Institut d'Investigació en Intel·ligència Artificial
Universitat Politècnica de Catalunya
PhD Program in Artificial Intelligence

Value Engineering for Autonomous Agents

by Nieves Montes

***Directors**: Prof. Carles Sierra and Dr. Nardine Osman (IIIA-CSIC)*
***Tutor**: Prof. Cecilio Angulo (UPC)*

Paper-Based Thesis
Tesis per compendi d'articles

Barcelona, November 2023

A mis padres,
por todo lo que soy.

Abstract

The topic of this thesis is the engineering of values for autonomous agents. This is realised through the formulation, design and implementation of new functionalities for autonomous agents that enable reasoning in terms of values. In particular, we argue for the role of prescriptive norms as value-promoting mechanisms. Hence, value-driven agents should be able to autonomously determine which regulations (such as obligations, permissions or prohibitions) make the Multiagent System they inhabit better promote some values of interest. We lay the foundations of our work on Schwartz's Theory of Basic Human Values to establish a consequential connection between values and norms, considering that norms are aligned with respect to values if the outcomes they incentivise satisfy the goals that capture the meaning of values in a particular context. Another feature of Schwartz's theory that has been previously overlooked in the literature is the strong social dimension of values. That is, agents should be able to reason not just in terms of their own, but also of the values of others in their community. This points to Theory of Mind (i.e. the cognitive ability to perceive, interpret and reason about others in terms of their mental states) as an outstanding component of value-based reasoning.

This thesis is structured around three main contributions (published in journal papers) plus their integration. The first contribution establishes the norm-value relationship as a consequential one in nature, and proposes a methodology for the automated synthesis and analysis of optimally value-aligned normative systems. The second contribution tackles the limitations of the first, and defines the Action Situation Language to systematically express a wide range of rules that may be implemented in a Multiagent System. This language is complemented by a game engine that automatically interprets interaction descriptions and builds their semantics as Extensive Form Games, which are later analysed with standard game-theoretical tools. This leads to a distribution over game outcomes, which are evaluated in terms of their desirability with respect to val-

ues. The third contribution introduces Theory of Mind-related functionalities into an existing Belief-Desire-Intetion agent architecture, and combines them with abductive reasoning capabilities.

The three contributions are integrated in a novel functionality that enables agents to reason about prescriptive norms in terms of dynamic values. This means that an autonomous agent can, at runtime, switch its value perspective to the one it estimates that another agent has. Such perspective-dependent value-based normative reasoning functionality, with its inherent social orientation, constitutes a novel contribution to the community of values for autonomous agents and paves the way for possible applications such as value-based negotiation over normative systems. In summary, value engineering is a principled and systematic approach to computational ethics, which provides an innovative tool set for integrating ethical values into the design of autonomous agents.

Keywords – value engineering, values in autonomous agents, norms, normative multiagent systems, theory of mind

Resumen

El tema de esta tesis es la ingeniería de valores para agentes autónomos, lograda mediante la formulación, diseño e implementación de nuevas funcionalidades que permiten a agentes autónomos razonar en términos de valores. En particular, defendemos el papel de las normas prescriptivas como mecanismos de promoción de valores. Entonces, los agentes impulsados por valores deben poder determinar de forma autónoma qué regulaciones (tales como obligaciones, permisos o prohibiciones) promueven mejor algunos valores de interés en el Sistema Multiagente que habitan. Fundamentamos nuestro trabajo en la Teoría de Schwartz de Valores Humanos Básicos para establecer una conexión entre valores y normas basada en consecuencias, considerando que las normas están alineadas con respecto a los valores si los resultados que incentivan satisfacen las metas que captan el significado de dichos valores en un determinado contexto. Otra característica de la teoría de Schwartz que ha sido pasada por alto previamente en la literatura es la fuerte dimensión social de los valores. Es decir, los agentes deberían poder razonar, no sólo en términos de sus propios valores, sino también de los de otros en su comunidad. Esto apunta a la Teoría de la Mente (es decir, la capacidad cognitiva de percibir, interpretar y razonar sobre los demás en términos de sus estados mentales) como un componente destacado del razonamiento basado en valores.

Esta tesis se estructura en torno a tres contribuciones principales (publicadas en revistas académicas), además de su integración. La primera contribución establece la relación entre normas y valores mediante consecuencias, y propone una metodología para la síntesis y análisis automatizado de sistemas normativos óptimamente alineados con valores. La segunda contribución aborda las limitaciones de la primero, y define el *Action Situation Language* para expresar sistemáticamente una amplia gama de reglas que pueden implementarse en un Sistema Multiagente. Este lenguage se complementado con un intérprete que procesa automáticamente la descripción de la interacción y construye su semán-

tica como juego en forma extendida, que luego es analizado con herramientas estándares de teoría de juegos. Esto conduce a una distribución sobre estados finales del juego, que se evalúan en términos de su conveniencia con respecto a ciertos valores. La tercera contribución presenta funcionalidades relacionadas con la Teoría de la Mente, integrándolas en la arquitectura *Belief-Desire-Intention* existente y combinándolas con razonamiento abductivo.

Las tres contribuciones se integran en una novedosa funcionalidad que permite a agentes autónomos razonar sobre normas prescriptivas en términos de valores dinámicos. Esto significa que un agente autónomo puede, durante su ejecución, cambiar su perspectiva de valores a la que estima que tiene otro agente. Este enfoque, basado en valores y con una orientación social inherente, constituye una contribución novedosa a la investigación de valores para agentes autónomos y allana el camino a posibles aplicaciones como la negociación automática sobre sistemas normativos basada en valores. En resumen, la ingeniería de valores es un enfoque sistemático y de principios a la ética computacional, que proporciona un conjunto de herramientas innovadoras para integrar valores éticos en el diseño de agentes autónomos.

Palabras clave – ingeniería de valores, valores en agente autónomos, normas, sistemas multiagente normativos, teoría de la mente

Resum

El tema d'aquesta tesi és l'enginyeria de valors per a agents autònoms, aconseguida mitjançant la formulació, disseny i implementació de noves funcionalitats que permeten a agents autònoms raonar en termes de valors. En particular, defensem el paper de les normes prescriptives com a mecanismes de promoció de valors. Aleshores, els agents impulsats per valors deuen poder determinar de forma autònoma quines regulacions (com ara obligacions, permisos o prohibicions) promouen millor alguns valors d'interès en el Sistema Multiagent que habiten. Fonamentem el nostre treball a la Teoria de Schwartz de Valors Humans Bàsics per establir una connexió entre valors i normes basada en conseqüències, considerant que les normes estan alineades respecte als valors si els resultats que incentiven satisfan les metes que capten el significat d'aquests valors en un context determinat. Una altra característica de la teoria de Schwartz que ha estat passada per alt prèviament a la literatura és la forta dimensió social dels valors. És a dir, els agents haurien de poder raonar, no només en termes dels seus propis valors, sinó també dels d'altres a la seva comunitat. Això apunta a la Teoria de la Ment (és a dir, la capacitat cognitiva de percebre, interpretar i raonar sobre els altres en termes dels seus estats mentals) com un component destacat del raonament basat en valors.

Aquesta tesi s'estructura al voltant de tres contribucions principals (publicades en revistes acadèmiques), a més de la seva integració. La primera contribució estableix la relació entre normes i valors mitjançant conseqüències, i proposa una metodologia per a la síntesi i anàlisi automatitzada de sistemes normatius òptimament alineats amb valors. La segona contribució aborda les limitacions de la primera, i defineix l'*Action Situation Language* per expressar sistemàticament una àmplia gamma de regles que es poden implementar en un Sistema Multiagent. Aquest llenguatge es complementa amb un intèrpret que processa automàticament la descripció de la interacció i construeix la seva semàntica com un joc en forma estesa, que després és analitzat amb eines es-

tàndards de teoria de jocs. Això condueix a una distribució sobre estats finals del joc, que s'avaluen en termes de la seva conveniència respecte a certs valors. La tercera contribució presenta funcionalitats relacionades amb la Teoria de la Ment, integrant-les a l'arquitectura *Belief-Desire-Intention* existent i combinant-les amb raonament abductiu.

Les tres contribucions s'integren en una nova funcionalitat que permet a agents autònoms raonar sobre normes prescriptives en termes de valors dinàmics. Això vol dir que un agent autònom pot, durant la seva execució, canviar la seva perspectiva de valors a la que estima que té un altre agent. Aquest enfocament, basat en valors i amb una orientació social inherent, constitueix una contribució nova a la investigació de valors per a agents autònoms i aplana el camí a possibles aplicacions com la negociació automàtica sobre sistemes normatius basada en valors. En resum, l'enginyeria de valors és un enfocament sistemàtic i de principis a l'ètica computacional, que proporciona un conjunt d'eines innovadores per integrar valors ètics al disseny d'agents autònoms.

Paraules clau – enginyeria de valors, valors en agents autònoms, normes, sistemes multiagent normatius, teoria de la ment

Acknowledgments

I would like to thank my supervisors Carles Sierra and Nardine Osman for their guidance and encouragement throughout this thesis. The circumstances under which this journey started were far from ideal, but we made the most of them. I would also like to thank Michael Luck and Odinaldo Rodrigues from King's College London for their warn welcome and stimulating discussions during my research visit, and to the TAILOR project's connectivity fund for providing the funding. A shout-out also goes out to Pedro Messeguer, for organizing the wonderful *Bojos per la IA* seminars at IIIA-CSIC.

Research would be close to impossible without all the supporting staff who make sure that the lights are on and the gears keep turning. Hence, I would like to thank all the administrative and IT support staff at IIIA-CSIC, for always helping me and never making any comment about my ignorance on administrative matters.

Finally, I would like to thank my family, and especially my parents for their unconditional love and support throughout not just this thesis, but my whole life. To my friends, for the much-needed distractions, laughs, walks and dinners. And to Ivan, for his unwavering steadiness and patience through the good and through the not-so-good.

To all of you, a much deserved *Thank you*.

Contents

II Contributions

III Closing Remarks

Part I

Background

Chapter 1

Introduction

The topic of this thesis is the engineering of values into artificial software agents and Multiagent System (MAS). To set the stage, it is first necessary to start by clarifying the topic this thesis is concerned with. To that end, in this chapter we first introduce the conceptual framework for *values* that we adhere to. Second, we present and discuss the topic of *value engineering*: we clarify what we understand by the engineering of values, what is the motivation to pursue it, and relate it to some computational constructs used in the Autonomous Agents and Multiagent Systems (AAMAS) community. Third, we present the overall research goal that this thesis aims to attain, and the sub-goals that derive from it. Fourth, we outline at a high level the strategies pursued to achieve our research goal, and point to the papers in Part II where they are developed. We conclude this chapter with an outline of the rest of the document and a summary of all the publications that have resulted from this thesis.

1.1 Schwartz's Theory of Basic Human Values

1.1.1 The Framework

This thesis revolves around the concept of *values* and, in particular, their role in social life. In other words, we are interested in the functionalities of autonomous agents that can help them reason and act upon their environment, as well as in relation to the other agents with whom that environment is shared, in terms of the values that are deemed most important.

Values are an abstract and fuzzy concept originating in moral philosophy. However, any potential enactment of values in a computational context needs to make specific commitments on the implementation strategy that it is going

to pursue. Inevitably, the strategies followed to turn the abstract concept of "values" into computational entities must be grounded in a solid theory of values, which provides a definition for them, outlines their function in human social life and hints about how value structures are organised. In this section, we review one such conceptual framework. Specifically, we rely on Schwartz's Theory of Basic Human Values (STBHV) (Schwartz, 1992, 2012), which has been selected for three main reasons. The first relates to the merits of the theory: it is a well-established theoretical framework for values in the social sciences, and many of its features are compatible with those of other frameworks (Rohan, 2000). Second, STBHV is more grounded in the fields of social psychology and sociology than philosophy, since it conceived values as profoundly social constructs (as will be reviewed shortly). This feature is particularly relevant to this thesis since we are interested in the role of values as regulators of social life, i.e. how do they affect the interactions taking place at the heart of a MAS, both between an agent and its shared environment and between several agents. Third and last, not only is Schwartz's theory popular within the social sciences, but it has had a very positive reception in the AAMAS community too. Many previous approaches to values in autonomous agents have adhered to it, as will be illustrated by the literature review in Section 2.2.

The concept of *value* has a long history in moral philosophy, resulting in a variety of definitions for this term (Boudon, 2017, chapter 1). Despite the confusion that such an abundance of definitions generates, STBHV provides a succinct characterization of values through the following five features:

> Values (1) are concepts or beliefs, (2) pertain to desirable end states or behaviours, (3) transcend specific situations, (4) guide selection or evaluation of behaviour and events, and (5) are ordered by relative importance. (Schwartz, 1992, p. 4)

Hence, Schwartz's theory conceives values as very general, abstract guiding principles that individuals and groups utilise to generate judgements on a variety of constructs: actions, strategies, conventions, policies, outcomes... The features presented above, however, do not fully specify how values become operational, i.e. how are they instantiated in a situation to which they are relevant. To address this, the STBHV also proposes that the content of every value is realised into a *motivational goal*:

> (...) the primary content aspect of a value is the type of goal or motivational concern that it expresses. (...) values represent, in the form

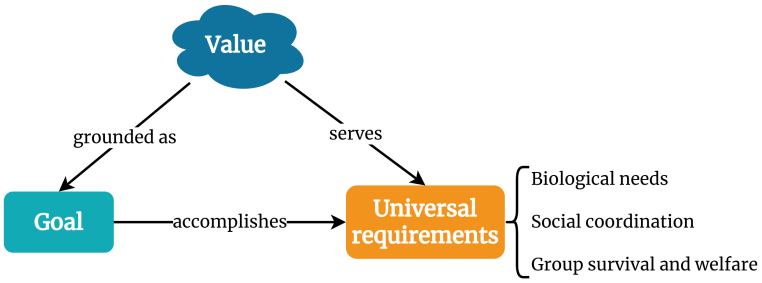


Figure 1.1: The three main components of Schwartz's Theory of Basic Human Values and the relationships among them.

> of conscious goals, three universal requirements of human existence to which all individuals and societies must be responsive: needs of individuals as biological organisms, requisites of coordinated social interaction, and survival and welfare needs of groups. (Schwartz, 1992, p. 4)

Thus, values, when activated, take the form of a motivational goal whose function is to allow an individual and their community at large to thrive. For example, according to STBHV, the value *power* is instantiated as the search for "social status and prestige, control or dominance over people and resources", while the value *universalism* is defined as an "understanding, appreciation tolerance, and protection for the welfare of all people and for nature" (Schwartz, 2012).

In summary, the STBHV is composed of the following three concepts: (1) the abstract values; (2) the defining goals that values motivate in specific situations; and (3) the ultimate functions that those goals seek to achieve. Figure 1.1 presents a diagram with the relationships among the three concepts. Note the cloud-shaped box around values underlining their abstract nature. Of the three concepts outlined in Figure 1.1, the one that is typically the least acknowledged is the function that values serve (i.e. the universal requirements values fulfil). When values are instantiated as motivational goals, they serve as heuristics that help by-pass lengthy (and potentially unfeasible) reasoning about everyday events and ultimately fulfil universal requirements of human existence. For example, when asked to justify why you paid for your groceries at the supermarket, most people would answer something along the line of "because it is the right thing to do". Very few would state something similar to "refusing to pay for my items could motivate everyone else to do the same, throwing society into a down-spiral and eventually landing in a state of anarchy that could threaten

my very existence". It is much more handy (and accounts for much shorter explanations) to make a moral argument and appeal to a shared sense of duty.

So far, we have covered the parts of STBHV where the features of values and their operational consequences as motivational goals are defined. Such motivational goals must be fulfilled or maintained by individuals and are hence instantiated at the individual agent level. Nonetheless, STBHV conceives values as profoundly social constructs

> Individuals cannot cope successfully with these requirements of human existence on their own. Rather, people must articulate appropriate goals to cope with them, communicate with others about them, and gain cooperation in their pursuit. Values are the socially desirable concepts used to represent these goals mentally and the vocabulary used to express them in social interaction. (Schwartz, 2012, p.4)

Therefore, values provide a sort of shared vocabulary to help humans coordinate and, ultimately, ensure the survival of the individual and, by extension, of the group. In essence, values provide a syntax that, if coupled with a shared semantics, helps humans relate to others by providing an understanding of what motivates them and what do they consider to be important in life. Having an idea of what values are important to another agent also generates expectations about their behaviour and/or their attitudes towards certain rules, laws or norms that are currently implemented or under discussion for implementation. Naturally, the accuracy of an agent's understanding of another in terms of their values depends on the extent to which their semantics overlap. For example, consider agents α and β, and value v. For β, v is grounded as goal g_v^{β}, while α believes that β grounds the semantics of v as goal $g_v^{\alpha,\beta}$. Fundamentally, the accuracy of α's explanations and/or expectations about β's behaviour depends on the degree to which $g_v^{\alpha,\beta}$ correctly reflects g_v^{β}.

1.1.2 Classification of Values

The merit of Schwartz's Theory of Basic Human Values is not only in the conceptualization of values presented earlier. In fact, this theory is often described as a landmark contribution to the field of intercultural studies. That is the case because STBHV delineates a set of ten broad value types (achievement, power, security, conformity, tradition, hedonism, stimulation, self-direction, universalism and benevolence) that have been found to be consistent across all the

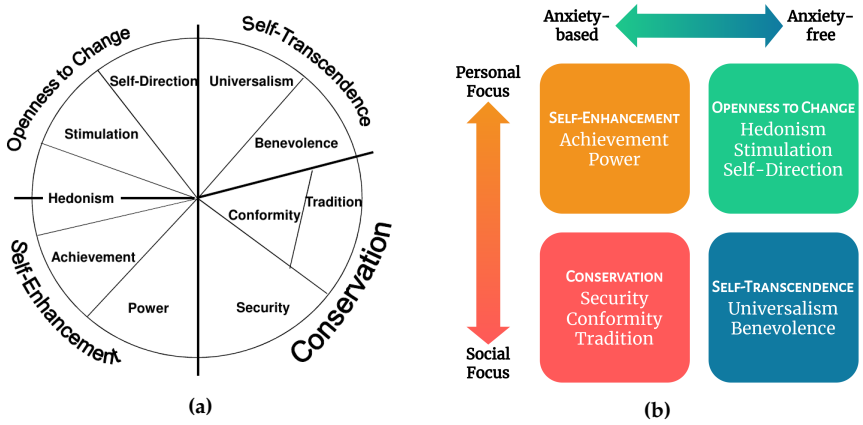


Figure 1.2: Organization of Schwartz's basic values based on (a) their convergence or conflict relationships and (b) their relationship to anxiety (horizontal axis) and focus (vertical axis). [Extracted and adapted from Schwartz, 2012, respectively].

world's major cultures (Schwartz, 2012). In other words, this set of values, along with their motivational goals and the relationships of conflict and compatibility among them, is found to be constant across societies with widely different cultural practices. Additionally, the set of ten value types is comprehensive, meaning that any justification provided by individuals for their behaviour can be related to one of the goals that values motivate (Schwartz, 1994). Although we do not rely on Schwartz's classification of values to develop the novel agent functionalities of this thesis, we briefly outline them here for completeness.

Given the set of universal value types mentioned above, STBHV offers several criteria to classify and arrange them within a value structure. The most popular ordering criterion is the relationship of congruence or conflict that arises between goals motivated by different values. This principle gives rise to a cyclic value structure, where values whose motivational goals are often compatible sit next to one another, while values that usually enter into conflict are opposite to one another. The resulting value structure appears in Figure 1.2a.

Another possibility considered by Schwartz is to organise values along a one-dimensional axis based on their relationship to a particular variable. The first such ranking variables included in the STBHV is anxiety. According to this principle, values go from being fully anxiety-based (seeking to avoid conflict, maintain the current order or actively control threat) to fully anxiety-free values (pursuing personal growth and self-expansion). This results in one half of the basic value types (achievement, power, security, conformity and tradition) on the anxiety-based end of the axis, while the other half (hedonism, stimulation, self-direction, universalism and benevolence) fall on the anxiety-free end. This

classification is displayed in the horizontal axis in Figure 1.2b.

The second ranking variable in the STBHV for organizing value types looks to their *focus*, i.e. whose interest a value is serving. On one end of the resulting one-dimensional axis are the value types with a *personal focus*: achievement, power, hedonism, stimulation and self-direction. The motivational goals for these types regulate how individuals express their own interests and characteristics. In contrast, on the other end we find values with a *social focus*: security, conformity, tradition, universalism and benevolence. These relate to how individuals interact in social settings, affect and influence one another. This last organizing principle corresponds to the vertical axis in Figure 1.2b.

1.2 Motivation: Value Engineering

So far, we have introduced the conceptual framework for values that this thesis follows. However, the aim of this thesis is not to provide arguments for or against the merits of STBHV, nor to compare it against any other theoretical framework from moral philosophy. Rather, this thesis is concerned with the *engineering* of these values, that is, their conversion from abstract notions into computational entities and the development of novel agent functionalities that operate on such value-inspired computational entities.

When we talk about *value engineering*, we are referring to the collection of strategies, techniques and methods that can be leveraged to *embed* values into autonomous agents. In turn, to *embed a into b* means to "make *a* an integral component of *b*".[1] Therefore, to engineer values into autonomous agents and/or the MAS they compose, these values have to be incorporated as fundamental constituents of the computational machinery that the system runs on. However, since values are abstract concepts, there is no straightforward way to map them to the computational entities that autonomous agent models are built upon. Although they are deeply intertwined, we loosely differentiate approaches that engineer values by targeting the *individual* through agent-level constructs such as beliefs, goals and plans; and those that target the *collective* through MAS-level constructs such as organizations, institutions and some interpretation of norms.

Despite the challenge it presents, we believe that the work on the engineering of values in autonomous software agents is important, timely, and relevant. The vision of the AAMAS community is the evolution towards a massive MAS, where a large set of heterogeneous agents with varying degrees of autonomy

[1] https://www.merriam-webster.com/dictionary/embed

and sophistication interact with one another and with their human users (Sierra, 2022). If the interactions that these software agents engage in are to respect the values that humans hold dear, it becomes mandatory to incorporate them during their development. In other words, the values that are deemed to be important, given the area where an autonomous agent is going to be utilised, have to be embedded, or *engineered*, into the system prior to deployment.

The engineering of values into autonomous agents can take place through a variety of avenues (see Sections 2.1 and 2.2). Remarkably, in this thesis we want to constantly and explicitly consider the role that values have in social life, since they ought to be engineered into autonomous agents that will share their environment with other agents as well as possibly humans. Thus, it is not enough to consider the engineering of values for a single agent. We must also establish through which mechanism an agent perceives, reasons about and relates to the value structures of those around it. This social dimension of values is barely considered in the current literature on values for autonomous agents, as manifested by the review of the state-of-the-art in Chapter 2 and the gaps identified there.

In this thesis we identify two high-level computational constructs to tackle the challenge of embedding values into autonomous agents. In tandem, these two entities can work together to achieve the vision of value-aware socially-oriented agents outlined above. First, we consider the *prescriptive norms* that regulate agent behaviour, and subsequently also MAS behaviour. These norms restrict agent autonomy to a greater or lesser extent, and constitute the institutional environment where agents are situated. Since the regulation of a MAS through prescriptive norms is applied to the system as a whole, we refer to this component as being *top-down* or targeting the *collective*.

Nonetheless, this thesis is concerned with the development of novel agent functionalities informed by values and oriented towards the social environment. Hence, the second component we consider is a cognitive ability which is particularly important for human social life: Theory of Mind (ToM). This ability, when implemented for agents, allows them to reason about the mental states (i.e. the beliefs, goals desires, intentions), and in particular about their values. As opposed to the previous component, this approach does not target the MAS level. Rather, it is a *bottom-up* approach since it targets one by one the constituent components of a MAS (i.e. the individual agents). Hence, we also refer to the use of ToM to engineer values into autonomous agents as an *individual* approach.

1.2.1 The Role of Norms

The first component for the engineering of values into autonomous agents and MASs that we have identified is the set of *prescriptive norms* implemented in the MAS. As mentioned, this is a top-down component that focuses on regulations that apply to the collective.

In a MAS, the interactions that agents have with their environment and with one another are regulated by the set of norms (or *normative system*) in place. In fact, norms are one of the most studied elements to engineer MASs, and much previous work exists on their representation, synthesis, emergence and adoption (Aldewereld et al., 2018; Andrighetto et al., 2012). Within the AAMAS community, the term *norms* is used to denote two related but distinct concepts: *conventions* and *prescriptions* (Grossi et al., 2012).

Conventions are patterns of behaviour that spread through a population and emerge as the dominant agent strategy in a bottom-up fashion (Morris-Martin et al., 2019). In essence, conventions are guidelines on behaviour that agents acquire through, for example, an imitation process in a social network (Li et al., 2020). Hence, their representation is usually contained within the agents' internal data structures.

In contrast, prescriptions are rules and laws that establish obligations, prohibitions or permissions on actions or on the state of affairs of the system. Often, prescriptions include a pre-condition for the rule to be enforced (for example, which subset of agents it applies to and under what provisions), a deontic modality (obligation, permission or prohibition) and the action or formula targeted by it (van der Torre, 2003). Usually, prescriptive norms are represented in data structures external to the agents, to which they generally have access to and can reason about. In contrast to conventions, prescriptions are usually implemented in a top-down fashion, with a central authority or mechanism in charge of their design and enforcement. Hence, they are frequently not the result of an emergence process but of a centralised design task, and apply to the set of agents as a whole. Throughout this thesis, we adopt the prescriptive view of norms. Therefore, from now on, every time we use the term *norms*, we are referring to prescriptions.

In the context of software agents, prescriptive norms may be implemented using one of two possible enforcement strategies: regimentation or sanction (Savarimuthu & Cranefield, 2011). Regimented norms render some of the actions that can be executed by agents and/or the states they are able to achieve

impossible. Norms enforced through sanctions establish obligations or prohibitions that cannot be perfectly regimented. To incentivise compliance, they are always accompanied by sanctions for detected violations or additional rewards to praise compliance. Usually, the triggering of a sanction or a reward creates an additional obligation for an enforcer agent to enact the sanction or reward in question (Hübner et al., 2011). However, sometimes the enforcement of sanctions and rewards is abstracted away as part of the platform where the MAS is executing (Fagundes et al., 2016). Despite there being these two distinct enforcement mechanisms, within a MAS there may coexist regimented norms with norms enforced through sanctioning.

Now that we have established what norms are, their classification, and the precise meaning we adopt, we focus on the role of norms as a potential avenue to engineer values into autonomous agents and MAS.

While values remain abstract context-independent concepts, norms deal with concrete actions and states of the system. The operational semantics of a MAS is defined as that of a transition system (Plotkin, 1981). When a set of norms is implemented in a MAS, the state transitions that the system undergoes change. This happens because either some transitions are eliminated altogether (e.g. a regimented prohibition makes an action impossible or a post-transition state unreachable), or the post-transition state changes (e.g. because a sanctioning mechanism is triggered or a new obligation to sanction is created). Consequently, the ultimate *outcomes* (i.e. the states at the end of a long sequence of state transitions) that the system is likely to achieve depend on the implemented norms. In other words, the probability distribution over the set of attainable outcomes is a function of the norms that the system is subject to and the institutional structure that oversees their application (i.e. regimentation and/or sanctioning).

It is precisely through the norm-dependent outcomes that the connection between norms and values can be established. Norms can, if carefully designed, promote the achievement of outcomes where a given value (or a set of values)

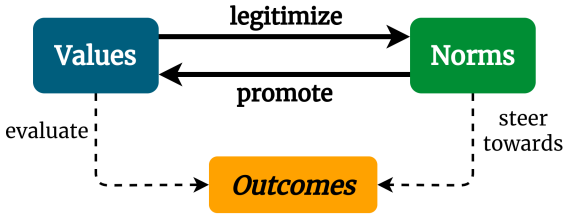


Figure 1.3: Relationship between values and norms through outcomes.

is being respected, promoted, or abided by. When the implementation of a new norm n increases the likelihood of an outcome that is deemed positive with respect to value v, we state that *n is aligned with respect to v*. Hence, we view the relationship between norms and values as *consequential* in nature. A norm is not moral or social in itself (a position defended by deontological ethics, see Alexander and Moore, 2021), it is so to the extent that the effects it brings about in the system are in agreement with the values we are interested in, in this case these are pro-social values. Hence, the alignment of a set of norms with respect to a set of values can be empirically evaluated by implementing the norms on the system (or on a simulated model of it), waiting for it to achieve some outcomes and evaluating those.

Our position is summarised by Figure 1.3. At the surface, values and norms form a feedback loop: values legitimise the enforced norms, while norms promote certain values. However, at a more fundamental level, the two are linked by the outcomes that norms steer the system towards and that are favourably evaluated in regard to values.

1.2.2 The Role of Theory of Mind

So far, we have covered the use of prescriptive norms as a possible avenue to engineer values into autonomous agents. Norms are a top-down or collective component, that apply at the MAS level. As discussed in the previous Section, the justification for a norm to be implemented is its alignment with respect to some values. Therefore, values, which are *individual* agent constructs, motivate the adoption of norms for the *collective* of agents.

In order for this connection to take place, some bridge from the individual agent values to the collective norms must be established. We envision such a bridge as a negotiation process, in which agents bargain about which values to implement in their shared environment based on their degree of alignment with respect to their values. We do not expect that the norms that the participating agents eventually agree upon to be optimally aligned with respect to any of the agent's values. Rather, they represent an *aggregation* of the values of the agents involved.

In a setting where an agent has an incentive to align the norms in place to its values more closely, even at the expense of compromising and making some concessions, it is especially useful to be able to reason not just in terms of one own's values, but also in terms of the values of others. Even though in this

thesis we do not develop any specific negotiation strategies for bargaining over norms, the ability to evaluate a norm proposal in terms of the value-motivated goals of the interlocutor is an essential requirement to bring forward proposals for norms that stand any chance of being accepted. Therefore, agents need cognitive abilities to be able to cope not just with their values and the goals they ground, but also with the values of others and the goals they motivate for them.

The human cognitive ability that allows us to perceive, interpret and reason about others in terms of their mental states (e.g. beliefs, goals, desires, emotions, intentions) is called Theory of Mind (ToM) (Malle, 2022). Although it is not an exclusively human capacity, it is considered essential for successful participation in social life (M. C. Corballis, 2011). In fact, the lack of reciprocity in social interactions displayed by some people with autism spectrum disorder is, in part, attributed to a deficient development of ToM abilities (Baron-Cohen et al., 1985). Despite its prevalence in social life, ToM is not innate. It is empirically established that children develop a ToM at around four years old, when they start to correctly assign false beliefs to others, that they themselves know to be true, in the famous Sally-Anne test (Korkiakangas et al., 2016).

There exist two distinct accounts of ToM within philosophy and psychology: Theory-Theory of Mind (TT) and Simulation-Theory of Mind (ST) (Röska-Hardy, 2008). The TT account views the cognitive abilities assigned to ToM as the consequence of a theory-like body of implicit knowledge. This knowledge is conceived as a set of general rules and laws about the deployment of mental concepts, analogous to a theory of folk psychology. In contrast, the ST account views ToM not as an inference ability, but as the capacity to use one's own cognitive processes and mechanisms to build a model of the minds of others and the processes happening therein. Hence, to attribute mental states, one imagines oneself as being in the other agent's position. Once there, humans apply their own cognitive processes, engaging in a sort of simulation of the minds of others. This internal simulation is intimately related to empathy, since it essentially consists of experiencing the world from the perspective of someone else.

To recap, prescriptive norms can steer a MAS towards outcomes that are better aligned with respect to a given set of values. Agents, when reasoning about which norms to implement, will have to consider not only the alignment with respect to their values, but also with respect to the values of others, since norms are implemented on the MAS as a whole. From this remark, we conclude that software agents, just as humans, will need to be endowed with ToM capabilities

if they are to successfully reason about norms and values from the perspective of their peers. However, traditional agent architectures such as Belief-Desire-Intetion (BDI) (Rao, 1996) and Reinforcement Learning (RL) (Sutton & Barto, 2018) do not include ToM as an essential functionality. Therefore, in order to develop value-aware socially-oriented agents, it is necessary to either expand existing agent architectures or design new ones with dedicated components to achieve functionalities analogous to those that ToM provides in humans.

1.3 Research Goals

The topic of this thesis is the engineering of values in autonomous agents and MASs. We advocate to achieve this through the use of *prescriptive norms* and to maintain a strong *social orientation* for this capability.

Main Research Goal (MRG): Develop a novel agent functionality to empower software agents to reason about the alignment (*i.e.* the suitability) of a set of prescriptive norms (or normative system) with respect to a value or set of values, either from the perspective of the agent's own values (*i.e.* the values that have been handed down to it by its user) or from the perspective of other agents it shares its environment with.

In order to achieve the **MRG**, it is best to split into smaller sub-goals or research questions that need to be answered, to tackle the several aspects that the **MRG** touches upon:

Research Question #1: How should *values* be *represented* in a way that is suitable to evaluate outcomes (that may refer to a variety of contexts or domains) in terms of their adherence to the value in question?

Research Question #2: How should *norms* be *represented* in a way that allows to connect them (either exactly or approximately) to the possible outcomes that the MAS may achieve, which are evaluated in terms of their value promotion by the value representation provided in **RQ#1**?

Research Question #3: Derived from the previous two research questions, how can prescriptive *norms* be *designed*, given a set of value representations, to maximise their alignment with respect to the given values? How can this process be automated?

Research Question #4: How can existing agent architectures be expanded with Theory of Mind capabilities so that software agents can perceive the state of the system from the perspective of other agents situated in their same environment?

Research Question #5: Following from **RQ#4**, how can an agent's Theory of Mind capabilities be used to evaluate the alignment of a set of norms with respect to the values held by other agents in the MAS?

The research developed in this thesis does not focus on any Artificial Intelligence (AI) technique in particular. Rather, it explores how already existing techniques can be adapted and/or expanded for the purpose of engineering values into autonomous agents.

This thesis aims to contribute to the field of Artificial Social Intelligence (ASI). ASI is an emergent sub-field within AI that deviates from the traditional paradigm of a single agent situated in an environment pursuing its individual goals, and takes as its starting point an agent situated within a larger community integrated by other software agents and also, possibly, humans. Despite its youth, ASI (which is sometimes referred to as *cooperative intelligence*) is a nascent field that nonetheless has been gaining traction recently, as researchers advocate for its importance in the current day and age (Dafoe et al., 2021) and research programs are being crafted (Dafoe et al., 2020; Paiva, n.d.). Within this research landscape, this thesis starts from the assumption that socially competent software agents have to incorporate values explicitly while being able to reason empathetically.

1.4 Three Contributions to Value Engineering

The main content of this thesis in Part II is constituted by three journal papers. These are standalone and self-contained publications. The contributions transition from a mostly *collective perspective* (focused on the representation of norms and values, and the optimisation of the alignment of norms with respect to values from a social planner perspective) to a mostly *individual perspective* (focused on the reasoning abilities of agents about their peers). These are all finally integrated in Part III. In this section, we briefly outline the contributions made in each paper, itemised by the techniques that they use. Naturally, all the details can be found in the complete papers included in Part II. We also indicate which research goals are addressed by each contribution, however the reader can find a detailed discussion on this issue in Chapter 4.

1.4.1 An Optimisation Approach

The first contribution to value engineering in this thesis focuses on leveraging prescriptive norms to ensure that a MAS as a whole abides by certain desirable values. It starts from the assumption, exposed in Section 1.2.1, that norms have the potential to promote values by steering the system towards certain outcomes where the values that we deem important are respected to a large degree. When this is the case, we state that the normative system in place is *aligned with respect to some value* v.

However, the challenge of finding *which* norms are the most aligned with respect to some given value v remains. To tackle this problem, we present a systematic norm synthesis methodology based on the maximisation of their *degree of value alignment*. We describe the problem building upon previous research for computing alignment (Sierra et al., 2019) and frame it as an optimisation task:

$$N^* = \underset{N \in \mathcal{N}}{\operatorname{argmax}} \ \mathsf{Algn}_{N,V} \tag{1.1}$$

where $\mathcal{N}$ is the space of possible normative systems, V is the set of values of interest and $\mathsf{Algn}_{N,V}$ is the degree of alignment of normative system N (an element in $\mathcal{N}$) with respect to V.

The general methodology we propose follows clear steps concerned with formulating the problem and its resolution. As part of the problem formulation, we define the *set of norms* or *normative system* $N = \{n_i\}$ that regulate the transitions between consecutive states. Every norm n_i is tied to a set of *normative parameters* P_i. For example, a norm that regulate electoral processes is parameterised on the minimum age of participants that are allowed to vote. All the normative parameters $\bigcup_i P_i$, together with their domains and constraints, define the search space of Equation (1.1). Also, we define the set of values of interest V. Formally, the meaning of value v in some domain is captured by its *semantics function*, $f_v : \mathcal{S} \to [-1, 1]$. This function evaluates states of the system, with $f_v(s) \approx -1, 0, 1$ indicating that state s strongly opposes, is neutral or strongly promotes value v.

Besides the norm synthesis methodology, we also provide an analytical toolbox to examine the results. This toolbox is composed of two instruments: the Shapley values of individual norms, and the degree of compatibility between values. The Shapley value of individual norms is a metric imported from cooperative game theory. Given a normative system $N = \{n_i\}$ that has been

optimised for value v, the *Shapley value of norm $n_i \in N$ with respect to value v* quantifies how much credit should be given to norm n_i for the degree of alignment with respect to v achieved by the overall set of norms. In addition to its definition, we also examine which properties of the Shapley value in the context of cooperative game theory can be imported to our framework.

The second tool in our toolbox is the *compatibility between values*. Given a normative system N^* that has been optimised for a specific value v_1, how capable is it of promoting a different value v_2? This metric quantifies how good is a normative system at compromising between different values. Following the lead of value compatibility, we define the Compatibility Maximizing Normative System (CMNS) as the set of norms that do not align optimally with respect to any value in particular, but that rather maintain the maximum degree of compatibility among all values within set V.

The journal paper with a detailed exposition of this work, accompanied by the code implementing our methodology, the analysis toolbox and a running example is included as Contribution 1 in Part II of this thesis and can be referenced as:

> Montes, N., & Sierra, C. (2022a). Synthesis and properties of optimally value-aligned normative systems. *Journal of Artificial Intelligence Research*, *74*, 1739–1774. https://doi.org/10.1613/jair.1.13487

Contribution 1 lays out the representation of values that we adhere to through this thesis (**RQ#1**), as well as a suitable representation of norms to enable reasoning about them with respect to values (**RQ#2**). Despite its limitations, the methodology we present enables the automated synthesis of normative systems using their degree of value alignment as the search target (**RQ#3**).

1.4.2 A Game Theoretical Approach

The previous piece of research has a major limitation: since norms are represented as sets of optimizable parameters, every time a user wants to introduce an additional regulation into the normative system, the methodology has to be re-executed from scratch. This hinders the ability of agents to dynamically examine the effects of implementing or retracting norms in the MAS where they participate.

To address this limitation, in the next piece of research we define a much more flexible and comprehensive norm representation and interpretation sys-

tem. We name it the Action Situation Language (ASL), which is inspired by the Institutional Analysis and Development (IAD) framework developed by Ostrom, 2005. The IAD framework is a policy analysis theory that identifies and delineates the generic building blocks that make up all social interactions. According to this framework, social interactions are determined by three sets of exogenous variables: the *biophysical conditions*, the *attributes of the community* and the *rules* of the interaction.

The IAD framework is remarkable in its clarity when it comes to differentiating between the components of social interactions, despite there being an immense variety of situations where humans and other entities engage with one another. In our work, we leverage this clarity to build a computational model of the IAD framework. We define the novel Action Situation Language (ASL). This is a machine-readable logic programming language (implemented in Prolog) whose syntax is highly tailored to the exogenous variables outlined in the IAD framework. Its central construct are the *rule statements*, expressed through an "`if <Condition> then <Consequence> where <Constraints>`" syntax.

Rule priorities are an important innovation of ASL. When interpreting an action situation description, rules with higher priority prevail over rules of lower priority if the two enter in conflict. This allows action situation descriptions to be *extended* in order to examine the impact that the introduction of new higher-priority rules has on the outcomes of the social interaction, without rewriting a new ASL description from scratch.

Beyond describing its syntax, ASL is complemented by a game engine that takes as input an action situation description and automatically generates its semantics as an Extensive Form Game (EFG). During this process, the game engine queries the rule base and handles conflicts between rules as they arise, by always following the rule with higher priority. Once the game model has been built, standard game theoretical tools (e.g. the Nash equilibrium) can be applied to the game to predict the most likely outcomes. These, then, are evaluated in terms of their adherence to the values that are deemed important by the community of agents that participate in the interaction. If this evaluation is not satisfactory, new rules with higher priorities can be introduced, the subsequent EFG model built and the new most likely outcomes predicted, in hopes that they are better aligned with respect to the values of interest.

The journal paper detailing this work is included in this thesis as Contribution 2 in Part II, including references to the open-source code that implements

ASL and its accompanying game engine. It can be found at:

> Montes, N., Osman, N., & Sierra, C. (2022b). A computational model of Ostrom's Institutional Analysis and Development framework. *Artificial Intelligence*, *311*, 103756. https://doi.org/10.1016/j.artint.2022.103756

Contribution 2 is mainly directed at providing an expressive norm representation language (**RQ#2**). Through its combination with game theoretical tool and the value representation scheme from Contribution 1, it enables reasoning about norms in terms of the values they promote or demote.

1.4.3 A Theory of Mind Approach

The two approaches presented so far respond to a *collective perspective* to value engineering: reasoning about which norms to implement on a MAS as a whole, based on their degree of value alignment. Nonetheless, in this thesis we are interested in implementing these norm reasoning capabilities at the *individual agent level*. Furthermore, we believe that for agents to successfully participate in a MAS in an ethical manner, they need to be able to relate to and reason about one another in terms of not just their own values, but of the values of their peers too. In other words, they must be endowed with a Theory of Mind (ToM).

Therefore, in our last contribution we propose a new agent model, which we call TomAbd. In this model, agent i is able to put itself in the shoes of another agent l by building an *estimation* of the belief base that l is operating with:

$$T_{i,l} = \{\phi \mid T_i \models \texttt{believes}(l, \phi)\} \tag{1.2}$$

where T_i is the agent's current belief base logic program (i.e. a logic program composed of facts and clauses) and $T_{i,l}$ is the estimation i makes of l's belief base.

By adopting the estimated program $T_{i,l}$ as its own (effectively attempting to perceive the world in the same way that l does), i can search for the missing beliefs that could complement $T_{i,l}$ to satisfactorily explain the behaviour that i observes l is displaying. This inference step, from the observation of an action by a peer to an explanation for it, is called *abductive reasoning*, and it yields a set of abductive explanations that the observing agent i can then use to build a more complete representation of the current state of the system and/or of the other agents within it.

The reasoning previously outlined uses *first-order* ToM. In other words, the agent adopts the perspective of agent l "directly". However, a distinguishing feature of the TomAbd agent model is that agent i can adopt the perspective of other agents down to an arbitrary level of recursion. For example, a TomAbd agent i may want to understand how an intermediary agent j perceives and reasons about agent l. This corresponds to *second-order* ToM reasoning. Analogously, i may want to estimate what is j estimation of ... k's estimation of l's belief base, thus adopting the perspective of the actor l through an arbitrary number of intermediaries. This corresponds to general n^{th}-order ToM reasoning.

To engage in higher-order ToM, the agent has to iteratively apply Equation (1.2):

$$T_{i,j,\ldots,k,l} = \{\phi \mid T_{i,j,\ldots,k} \models \texttt{believes}(l, \phi)\} \tag{1.3}$$

We have implemented and fully documented the TomAbd agent model in Jason (Bordini et al., 2007), an agent-oriented programming language based on the Belief-Desire-Intetion (BDI) architecture. As a benchmark, we have applied it to the Hanabi domain, a cooperative card game of imperfect information, and tested its performance in this domain along a number of metrics. The paper introducing the TomAbd agent model in detail is included as Contribution 3 in Part II of this thesis. It can be found at:

> Montes, N., Luck, M., Osman, N., Rodrigues, O., & Sierra, C. (2023a). Combining theory of mind and abductive reasoning in agent-oriented programming. *Autonomous Agents and Multi-Agent Systems*, *37*(2). https://doi.org/10.1007/s10458-023-09613-w

Contribution 3 is specialized towards providing the ToM capabilities (**RQ#4**) that the novel agent functionality envisioned in our **MRG** needs.

1.5 Publications

Besides the main publications mentioned in Section 1.4, there are others that are a direct consequence of the work developed in the course of this thesis, and that contain initial accounts of the detailed contributions in Part II. First, a reduced version of Contribution 2 first appeared at:

> Montes, N., Osman, N., & Sierra, C. (2021). Enabling game-theoretical analysis of social rules. In *Frontiers in artificial intelligence and applications*.

IOS Press. https://doi.org/10.3233/faia210120

Additionally, an extended abstract of Contribution 2 for presentation at an international conference was presented at:

Montes, N., Osman, N., & Sierra, C. (2023). A computational model of Ostrom's institutional analysis and development framework (extended abstract). *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2023/786

A preliminary account of Contribution 3 was presented as:

Montes, N., Osman, N., & Sierra, C. (2022a). Combining theory of mind and abduction for cooperation under imperfect information. In *Multi-agent systems* (pp. 294–311). Springer International Publishing. https://doi.org/10.1007/978-3-031-20614-6_17

Additionally, the participation in the Doctoral Consortium at a couple of internation conferences yielded the following two publications:

Montes, N. (2022a). Engineering pro-social values in autonomous agents – collective and individual perspectives. In *Multi-agent systems* (pp. 431–434). Springer International Publishing. https://doi.org/10.1007/978-3-031-20614-6_26

Montes, N. (2022b). Engineering socially-oriented autonomous agents and multiagent systems. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2022/833

Finally, a non peer-reviewed pre-print of a position paper presenting our position on values, norms and the relationship between the two is available at:

Montes, N., Osman, N., Sierra, C., & Slavkovik, M. (2023). Value engineering for autonomous agents. https://doi.org/10.48550/ARXIV.2302.08759

1.6 Document Outline

This section has introduced the necessary background, motivation, research goals and research outputs of this thesis. In the rest of this document, the discussion is structured as follows. In Chapter 2, we review relevant and recent work related to the topic of value engineering, with special attention to those approaches that are comparable to ours. Next, Part II includes the three main contributions of this thesis, which have been presented in Section 1.4. Part III starts with Chapter 3, which integrates the three contributions into a unified agent functionality for normative reasoning with respect to value from an arbitrary perspective. Finally, Chapter 4 concludes the thesis by reflecting on the initial research goals, listing the code repositories and functionalities that have resulted from this work, making an overall evaluation and pointing to potential future research directions.

Chapter 2

State-of-the-Art

The goal of this chapter is to provide the context for the research developed in this thesis. We review the current state-of-the-art concerning values and autonomous agents, starting from *collective* approaches (i.e. norm-related) and transitioning into *individual* ones (i.e. proposals to agents' cognitive abilities). We start with a review of the literature on norm synthesis and then narrow our focus to *value-guided* norm synthesis. Then, we move on to value-based practical reasoning and follow it up with recent approaches to reasoning about others, which we have outlined in Chapter 1 as a neglected requirement in the literature on values for autonomous agents. We conclude the chapter by identifying the gaps in the literature and the research opportunities that these afford. Given that values are a relatively recent concern of the AAMAS community, there does not exist an abundance of publications on the topic. Therefore, we choose to provide a relatively extensive review of each of the few cited works.

2.1 Value-Aligned Norm Synthesis

In the AAMAS community, norms are represented as obligations, prohibitions or permissions to perform certain actions, or to achieve some state of the system. The field of automated norm synthesis has traditionally considered the problem of finding the set of regimented norms (i.e. constraints) that are designed and enforced exogenously by an outside authority or mechanism. Pioneering work by Shoham and Tennenholtz (1995) and Onn and Tennenholtz (1997) defined the problem of finding the set of restrictions on actions that, for a given domain, ensure conflict-free operation while still allowing enough leeway for agents to achieve their goals. Hence, the problem is tightly linked to planning, and

prescriptive norms are referred to as "social laws". The set of social laws is synthesised *offline*, i.e. prior to system deployment. The domain originally considered is the multi-robot navigation problem, where a set of autonomous goal-oriented robots move around in a grid environment, and collisions should be avoided.

Following this tradition, recent work by Nir et al. (2020) revisits the problem. There, authors have the same goal as Shoham and Tennenholtz (1995), i.e. to find the set of constraints on agent actions that prevent conflicts while still allowing them to fulfil their goals. The domain of application is formalised using a multiagent version of the Stanford Research Institute Problem Solver (STRIPS), a popular planning formalism. Then, the set of possible social laws is conceived as a mapping within the space of multiagent STRIPS domains, where the original domain is modified as a consequence of the enforced norms. The authors frame the problem as a search task through the space of social laws, and solve it for various benchmark scenarios using various combinations of search algorithms (greedy best first, breadth first and depth first), heuristics, pruning techniques, and tie-breaking criteria.

So far, all the reviewed publications perform *offline* norm synthesis, where the set of prescriptions to be enforced is computed prior to run-time. However, offline synthesis is not suitable to regulate *open* MASs, where agents may enter and leave the environment during operation, and hence not all the system parameters (e.g. the number of agents and/or their type) are known *a priori*.

To address this shortcoming, Morales et al. (2013) present the Intelligent Robust Norm synthesis (IRON) system for *online* synthesis of normative systems. It consists of a norm engine that continuously monitors the system operation. When a conflict is encountered, it generates the restriction that would have prevented it. Then, it integrates this new norm into the normative system that is currently enforced, and applies a generalization step to avoid redundancies. This results in concise and succinct normative systems where duplicities are avoided.

Primarily, the objective of both online and offline norm synthesis is to find the set of *effective* prescriptions, i.e. those that ensure conflict-free operation. Other desiderata from the resulting normative system are, for example, compactness and liberalism (i.e. preserving the freedom of choice of the agents as much as possible, as in Morales et al., 2015). However, the incorporation of moral values into the norm synthesis process is a much more recent concern and, consequently, research on the topic is scarce. Furthermore, all existing

publications on value-guided norm synthesis consider only the *offline* scenario. Implicitly, this entails that the set of values and the preferences over them are static and do not change whatsoever as the system evolves.

Notably, Serramià et al. have developed complementary approaches (one quantitative and one qualitative) to incorporate values into the norm synthesis process. First, Serramià et al. (2018) consider a total preference ordering over the set V that reflects the overall values of the society. From this preference ordering, every value is assigned an integer utility based on its position in the ranking. Concerning the norm-value relationship, they define a *value support function*, which is provided as an input. This function relates every possible norm to the set of values supported by it. Then, the utilities of values are propagated to utilities on norms through the moral support function. This allows to formulate the problem of finding the most general, compact and value-aligned normative system as a multi-objective optimization problem. Once this conversion is complete, the problem can be solved using off-the-shelf solvers.

Second, Serramià et al. (2020) build upon Serramià et al. (2018) to develop a related but complementary *qualitative* approach. There, instead of having values inducing utilities over norms, the preference ordering over values is directly propagated as a preference ordering over entire normative systems through the value support function. Hence, the problem becomes to find the sound normative system that is the most preferred based on the values it promotes, i.e. the maximally *value-aligned* set of norms. The resulting problem has a very large search space (exponential in the number of possible norms), which the authors encode as a Linear Programming (LP) problem, and solve with off-the-shelf solvers.

2.2 Value-Based Practical Reasoning

So far, we have reviewed work on value engineering for MASs through prescriptive norms, a type of approach we refer to as *collective*, since they leverage a construct that is implemented on the society of agents as a whole. Now, we transition towards *individual* approaches, i.e. those that embed values by integrating them in the internal cognitive architecture of autonomous agents. In the current section, we focus on previous work that incorporates values into the *practical reasoning* of agents, i.e. deliberation oriented towards action selection or the justification of a recently performed action.

Atkinson and Bench-Capon (2016) propose an argumentation scheme for

practical reasoning based on an action-labelled transition system augmented with values. Its basic components are the set of states, the actions that label transitions between states and an interpretation function that assigns a set of atomic propositions to each state. Within this system, agents seek to fulfil their goals (which may be of type *achievement*, *remedy*, *maintenance* or *avoidance*) while promoting their values. Values function as a partition on the state transitions. In other words, given a value v, all transitions between two arbitrary states s and s' are classified as *promoting*, *demoting*, or *being neutral* with respect to v. This information is encoded into a logic program describing the conditions of the current state, which the agent uses to reason about which are the best actions to reach its target goals while abiding by its values.

There exists several publications with proposals to integrate values into the Belief-Desire-Intetion (BDI) model of practical reasoning. The BDI model of autonomy is inspired by the philosophical theory of the intentional stance by Searle (1983) and Dennett (1989), which defends that humans interpret the behaviour of entities (either alive entities such as other humans and animals, or technical artefacts) in terms of their mental states, such as their beliefs, desires and intentions. Most BDI agent architectures maintain a set of beliefs, a set of goals or desires to be achieved and a library of plans. Essentially, plans are templates for action that the agent should follow in order to achieve a goal in some context (Bordini et al., 2007, chapter 3).

In Cranefield et al. (2017), the authors introduce values into the BDI agent model by adopting the value structure of Schwartz's Theory of Basic Human Values (STBHV), presented in Figure 1.2a, and assuming that agents cannot pursue values that are in direct conflict (i.e. that are opposite to one another). Each value is characterised by its *target level* of satisfaction, its *current level* of satisfaction and its *current importance*. Then, they expand classical BDI goals with the *value-based reasoning* annotation $\mathcal{VBR}(G)$ (where G is a goal), a construct which indicates that the agent should consider values and not the default deliberation function when deciding about which means to select to fulfil G. $\mathcal{VBR}(G)$ is implemented as a set of annotations on the plans responsible for handling G. The agent program is then compiled into a set of constraints that capture the ethical implications of selecting some means to fulfil G. Hence, when the agent deliberates about which course of action to take in order to achieve G, it invokes an external constraint solver (instead of the default BDI deliberation function) that selects the plan that is most aligned with its values.

Staying with the BDI model of practical reasoning, Szabo et al. (2020) expand

the normative BDI model of Criado et al. (2010) with an explicit representation of values, based again on the STBHV. Originally, Criado et al. (2010) model norms as a new component of BDI agents, composed of a deontic operator and a literal representing the condition that an agent is obliged or prohibited to bring about. These norms are introduced into the agent reasoning cycle by generating desires to comply with them, i.e. to reach (avoid) states where the condition that the norm prescribes (prohibits) is fulfilled. Such desires are generated provided that the agent's willingness to comply with the norm in question surpasses a certain threshold.

In the extension by Szabo et al. (2020), values label ethically relevant propositional formulas with information about whether they promote or demote some value, and to which degree. All this information is aggregated into a quantity that expresses the degree to which states of the world satisfying formula γ are aligned with respect to the agent's overall value system. This aggregated metric is then incorporated into the BDI reasoning cycle through the computation of the willingness to comply with norms and the generation of intentions, through an assessment of the degree of value alignment in the intention's post-condition.

The practical reasoning approaches reviewed here operate on a lot of symbolic hand-coded information about values and their relationship to states, transitions, norms, plans and actions. On the upside, they benefit from easy integration with explainability techniques. Notably, Winikoff et al. (2021) extend the BDI model with a folk-psychological module that provides explanations for the behaviour of an agent based on its beliefs, desires and "valuings". These valuings are somehow a reflection of the agent's value system, and they are implemented (following the lead of Cranefield et al., 2017) as annotations on goals. These annotations capture the assessment that the agent makes of the post-conditions associated to the achievement of that goal. In summary, these valuings establish context-dependent preferences over the options available to achieve an initial goal, depending on which sub-goals the agent decides to pursue.

2.3 Reasoning About Others

In the publications reviewed so far, the modelling of values is (i) agnostic with respect to which specific values are being considered, and (ii) static with respect to the set of values that the agent is reasoning about. In other words, the agent has a fixed set of values it uses to reason, and that set is immutable during

the agent's lifetime. However, this thesis is specifically interested in the social dimension of values. As we argued in Chapter 1, autonomous software agents need to be able to interact with others in a manner that respects their values and, down the line, reach agreements. In other words, they need a Theory of Mind (ToM).

Within AI, implementations of ToM are often included under the umbrella of *modelling others*. Work on modelling others borrows the techniques and methods from other AI disciplines and applies them to building a representation of other agents (Albrecht & Stone, 2018). As a consequence, work ToM for autonomous software agents has so far been developed in a somewhat fragmented fashion, with every camp within the field implementing it according to their own techniques and methods. Furthermore, more often than not these techniques are solely applied to purely competitive domains, where they are referred to as *opponent modelling* (Baarslag et al., 2015; Nashed & Zilberstein, 2022).

In machine learning, prominent work by Rabinowitz et al. (2018) models ToM as a meta-learning process, where an architecture of several Deep Neural Network (DNN) components is trained on past trajectories of a variety of agents, including random, Reinforcement Learning (RL) and goal-directed ones. The target of the model is to predict an agent's action at the next time-step given their prior movements. The component of this architecture responsible for ToM capabilities is the *mental net*, which parses trajectory observations into a generic mental state embedding. Yet, it is not understood what kind of mental states (i.e. beliefs or goals) these embeddings represent.

Wang et al. (2022) also use an architecture based on DNN models to reach consensus in multiagent cooperative settings. Their ToM net explicitly estimates the goal that others are currently pursuing based on local observations. Another machine learning approach by Jara-Ettinger (2019) proposes to formalise the acquisition of a ToM as an Inverse Reinforcement Learning (IRL) problem. Yet, these learning approaches have drawn some criticism for their inability to mimic the actual operation of the human mind, as the direct mapping from past to future behaviour bypasses the modelling of relevant mental attitudes, such as desires and emotions (Cuzzolin et al., 2020).

ToM approaches have also been investigated from a game theoretical perspective, in particular the effect of the ToM level used. To understand what the "level of ToM" is, we need to grasp the *order* of ToM statements. Formally, ToM statements can be expressed using the language of epistemic logic, which studies the logical properties of knowledge, belief, and related concepts (van

der Hoek, 1993). For example, the belief of agent i is expressed using modal operator B_i. Then, formula $B_i\phi$ is read as "agent i believes ϕ".

ToM statements can be expressed by nesting beliefs about the state of the world. Therefore, statement $B_iB_j\phi$ is read as "i believes that j believes ϕ". This corresponds to a *first-order* ToM statement from the perspective of i. Subsequent nesting results in statements of higher order. For example, $B_iB_jB_k\phi$ is read as "i believes that j believes that k believes ϕ", a *second-order* ToM statement. This recursion can be extended down to an arbitrary level. In general, an n-th order ToM statement is expressed as $B_iB_{j_1} \ldots B_{j_{n-1}}B_{j_n}\phi$ and is read as "i believes that j_1 believes $\ldots$ that j_{n-1} believes that j_n believes ϕ".

In de Weerd et al. (2012) and de Weerd and Verheij (2011), the authors prove that while first-order and second-order ToM present a clear advantage with respect to opponents with ToM abilities of lower order (or no ToM capacity at all), the benefits of using higher-order ToM are outweighed by the complexity it entails. de Weerd et al. (2022) also prove that high-order ToM is beneficial in highly dynamic negotiation domains, and that the magnitude of the benefit increases with the uncertainty of the scenario.

Additionally, symbolic approaches to ToM have for the most part built upon the BDI model of agency. This is a natural choice since ToM aims to estimate mental states, like beliefs, desires and intentions, that are explicitly modelled in BDI languages. For example, A. Panisson et al. (2018) implement ToM for deceptive purposes by studying the effects that announcements have on the beliefs of others and their ripple-down effects on their desires and actions in response. They focus on the requesting and sharing of (possibly untruthful) information, and provide the operational semantics on the agent's transition system (which includes the models of others that the agent maintains) for these communicative actions (A. R. Panisson et al., 2019). Their approach is very much in line with Theory-Theory of Mind (TT). They use dedicated predicates to infer mental states, such as desires, given prior beliefs. These inferences are made from within the agent program, a feature which we consider qualifies as adherence to the theoretical version of ToM.

Sarkadi et al. (2019) extend the previous model by incorporating elements of trust and the modelling of several agent profiles based on their attitudes. In this extension, they distinguish between Theory- and Simulation-Theory of Mind (ST) components within their model. They argue that the TT component handles the assignment of prior beliefs to other agents, while the ST component handles belief additions based on agent announcements and the inferences

derived from them.

Finally, Harbers et al. (2009) establish a different criterion for classifying ToM approaches into TT and ST. They develop two separate ToM implementations, one identified with TT and the other with ST, for applications in virtual training systems. Both architectures maintain knowledge bases for the beliefs, logical rules and goals of other agents. The difference between the ST and TT approaches is found in the reasoner that is applied to the knowledge bases assigned to other agents. The TT architecture applies rules about how other agents combine their beliefs, goals and plans, which are explicitly included as part of the agent's own knowledge. In contrast, the ST architecture uses the agent's native reasoner, making it more lightweight. Besides this, other advantages are found for the ST architecture with respect to the TT one, namely code reusability and flexibility to deal with non-BDI agents.

2.4 Takeaways

As is natural with any emergent research field, there are several gaps in the current literature on values in AAMAS. First, concerning norm synthesis approaches reviewed in Section 2.1, the view adopted on the norm-value relationship is predominantly deontological (Serramià et al., 2020; Serramià et al., 2018). The set of values a norm promotes is provided as an input to the problem through the value support function, with no consideration of the states brought about by the norms. This is at odds with the *consequentialist* position derived form the STBHV. In other words, value-guided norm synthesis approaches do not really consider the *meaning* that a value acquires in a particular context. As a result, values end up being used as inert labels on which it is not possible to reason.

Meanwhile, some approaches to value-based practical reasoning do adopt a more consequentialist position of the norm-value relationship. e.g. Szabo et al. (2020) use values to label propositional formulas that either apply or not to the state of the system. However, the reviewed individual approaches conceive an autonomous agent in isolation, reasoning exclusively about its own actions, plans or conventions to adhere to. They do not conceive the possibility that autonomous agents might need to reason with respect to a set of values that does not reflect to their own position, but that of their peers. In all the reviewed literature, software agents are provided with the set of values they should consider at initialization time, and this set remains unchanged throughout the

agent's lifetime.

In contrast, it is the position of this thesis that, in order to incorporate the social dimension of values into autonomous agents, it is necessary to endow them with the necessary machinery to switch their value position at run-time, to put themselves in the shoes of other agents and try to understand the world in terms of their values. Hence, it is necessary to introduce an artificial Theory of Mind that allows agents to relate to one another by reasoning from each other's perspectives.

However, while we consider ToM to be a necessity for socially and value-oriented autonomous agents, a functioning ToM does not automatically guarantee that agents will abandon their individual pursues in favour of a compromise with their larger group or that they will act in a more pro-social manner. This is made abundantly clear by the many applications of techniques for modelling others in adversarial domains, while applications for cooperative teamwork are still a minority. Also, it is important to note that none of the existing approaches for estimating other agents' beliefs in the BDI architecture (which we favour in this thesis since beliefs are explicitly represented using first-order logic and are thus interpretable) specifically consider the estimation of value of others, even though values are, in essence, a particular type of belief.

Considering the discussion above, we believe there is an opportunity to develop research that addresses some of these deficiencies. In particular, by linking norms and values through the consequences brought about by norms and evaluated with respect to values, and by enabling agents to dynamically switch their value position using ToM, the capabilities of current approaches to values in autonomous agents can be significantly enhanced. In Part II we include three publications, first presented in Section 1.4, that make gradual contributions in this direction, and which culminate in Chapter 3 of Part III of this thesis.

Part II

Contributions

Contribution 1

Synthesis and Properties of Optimally Value-Aligned Normative Systems

Journal of Artificial Intelligence Research

Full citation:

Montes, N., & Sierra, C. (2022a). Synthesis and properties of optimally value-aligned normative systems. *Journal of Artificial Intelligence Research*, *74*, 1739–1774. https://doi.org/10.1613/jair.1.13487

Journal of Artificial Intelligence Research 74 (2022) 1739-1774 Submitted 11/2021; published 07/2022

Synthesis and Properties of Optimally Value-Aligned Normative Systems

Nieves Montes NMONTES@IIIA.CSIC.ES
Carles Sierra SIERRA@IIIA.CSIC.ES
Artificial Intelligence Research Institute (IIIA-CSIC)
Campus UAB Carrer de Can Planas, Zona 2
08193 Bellaterra, Barcelona

Abstract

The value alignment problem is concerned with the design of systems that provably abide by our human values. One approach to this challenge is through the leverage of prescriptive norms that, if carefully designed, are able to steer a multiagent system away from harmful outcomes and towards more beneficial ones. In this work, we first present a general methodology for the automated synthesis of value aligned normative systems, based on a consequentialist view of values. In the second part, we provide analytical tools to examine such value aligned normative systems, namely the Shapley value of individual norms and the compatibility of several values under a fixed set of norms. We illustrate all of our contributions with a running example of a society of agents where taxes are collected and redistributed according to a set of parametrised norms.

1. Introduction

In recent years, the term *value alignment* has been used to refer to the challenge of building artificial intelligence (AI) systems that comply, uphold, and respect the moral values that our societies care most about. This concern is rooted in the increasing power, autonomy and ubiquity of these technologies. The complexity of some of the algorithms that power AI systems, coupled with the sensitivity of the areas where they are deployed, entail the risk that us, the humans, might lose control over the systems whose primary purpose is precisely to serve us.

Within the multiagent systems (MAS) community, the value alignment problem translates into ensuring that the interactions at the heart of a society of agents are ethically appropriate. At least a subset of these agents are assumed to be software-enabled. Some approaches to value alignment in MAS introduce values as central elements in the reasoning schemes of agents' architectures (Atkinson & Bench-Capon, 2016). However, these methods assume complete access to the inner workings of the agents, a perk that does not apply in situations where the host of the interaction platform is not in charge of developing the agents.

To overcome this limitation, we turn to a widely studied element within the MAS literature: *prescriptive norms* (Savarimuthu & Cranefield, 2011). Prescriptive norms consist of regulations, constraints and directives on the behaviour of agents, possibly accompanied by monitoring and sanctioning provisions for detected violations. Usually, prescriptive norms are imposed by a system designer or central authority. When prescriptive norms are ap-

plied to software-enabled agents, they are often referred to as *technical norms* (van de Poel, 2020).

In this work, we claim that prescriptive norms have the potential to act as the main value-promoting mechanism within a MAS and that they should be leveraged to ensure that the moral values we deem most relevant are upheld by the system. If suitably designed, prescriptive norms are able to steer a MAS towards more ethically compliant outcomes, and are therefore a splendid avenue to engineer moral values into a society of agents. As we elaborate further in the text, prescriptive norms promote or demote end-states, which must then be assessed in terms of their degree of compliance with respect to some value. For this reason, we conceive *value alignment* to be a property of the implemented norms with respect to the values we intend to embed.

Even under the assumption that norms have the potential and should be aligned with respect to values, the challenge remains in finding *which* norms are actually the most aligned. The work presented in this paper tackles this problem by defining and deploying a novel and general methodology to automatically synthesise normative systems based on their degree of value promotion. Furthermore, we also provide an analytical toolbox to help the MAS designer to extract insights about the optimal normative systems that are returned by the value-guided search. Hence, the contributions of this paper are two-fold:

1. First, we present a systematic methodology to synthesise normative systems based on maximal value alignment. Formally, we seek to solve the following optimisation problem:

$$N^* = \arg\max_{N \in \mathcal{N}} \mathsf{Algn}_{N,V} \tag{1}$$

where $\mathcal{N}$ is the space of possible normative systems, V is the set of values of interest and $\mathsf{Algn}_{N,V}$ is the degree of alignment of normative system N (an element in $\mathcal{N}$) with respect to the values in V.

Our approach is differentiated from previous ones in that we take an explicitly consequentialist view of the relationship between norms and values. We present and discuss the underlying assumptions of our methodology, to help the reader understand the engineering choices that are made later on. This conceptual commitment allows us to quantify the value alignment of a set of norms with respect to a value (or set of values) through state features that are designated as proxies for the value in question.

Despite unequivocally adopting a consequentialist position, the methodology we put forward is general enough to be applied to a wide range of MAS, from very simple ones with blindly obedient agents, to others where participants are endowed with complex decision-making models. Our automated synthesis can be applied to either situation, as long as the norms governing the transitions between consecutive states are linked to a set of optimisable parameters. Moreover, no restrictions are placed on the values that such parameters may take: they can be either continuous or categorical, bounded within a domain or not.

The main steps of our synthesis approach are:

(a) Define the set of variables that completely specify the state of the MAS. These variables define the state space $\mathcal{S}$ that the system will visit as it transitions due to the actions taken by its populating agents and the norms in place.

(b) Define the norms n_i in the normative system N that regulate the transitions between consecutive states of the MAS. Every norm targets a particular aspect of the transition. All norms are parametric on a set of values, whose domains and constraints have to be specified. This parametrisation provides the search space $\mathcal{N}$ over which the maximising search in eq. (1) is performed.

(c) Define the set of values of interest V and the functions that capture their meaning in the context of the MAS. This step provides the *alignment* target function for the search in eq. (1).

(d) Choose a suitable optimisation algorithm and perform one maximising search for every value of interest. Use the search space defined in step (b) and the alignment established in step (c) as the objective function for the search.

2. In the second part of the paper, we provide analytical tools to examine the resulting optimal normative systems. These tools are intended to shine light on *how* are the optimal normative systems previously obtained operating in order to achieve their (hopefully) large degree of alignment, and whether they represent a good compromise between several competing values. Such insights should help the system designer reflect on the choices made prior to running the normative system search, and adapt accordingly if any of the information provided by these tools is deemed as unacceptable.

   The analytical toolbox we contribute to enrich the synthesis methodology is made up of two complementary metrics:

   (a) The concept of *Shapley value of an individual norm* within a given normative system is made possible by adopting the notion that any optimal normative system is a coalition of individual norms working together to promote some value. Our notion of Shapley value is imported straight from the cooperative game theory literature. It examines the interactions among the individual norms, given an alignment function with respect to some value. It helps establish whether an individual norm is critical or unimportant when it comes to promoting a specific value.

   (b) The notion of *value compatibility* quantifies how successful or neglectful of other values are norms that have been optimised for different goals. In other words, how much of a compromise is a normative system able to achieve between competing interests. This metric is complementary to the Shapley value. Just as the Shapley values examine the interaction among individual norms for a fixed value, compatibility looks at interactions between values under a fixed set of norms. Additionally, we also present and discuss the *compatibility maximising normative system*, which is intended to achieve the largest degree of harmony among values.

This paper is organised as follows. In Section 2 we review other works from the literature related to ours. Then, in Section 3 we discuss the philosophical assumptions that our value

alignment formal model is built upon. The formal model itself is presented in Section 4. Next, the optimisation that searches the most value-aligned normative systems is addressed in Section 5. The second part of this paper is comprised of Sections 6 and 7, which examine the Shapley values of individual norms and the compatibility among values, respectively. Finally, the main take-away points, conclusions and future work are presented in Section 8. All of our contributions are illustrated with a running example of a toy social model.

2. Related Work

In the AI literature, the most straightforward approach to incorporate values into autonomous agents comes from the practical reasoning community (Atkinson & Bench-Capon, 2016). There, values are explicitly incorporated into the reasoning schemes of agents for action and plan selection (van der Weide et al., 2010; Visser et al., 2015; Teze et al., 2019) and decisions on rule compliance (Szabo et al., 2020; Bench-Capon & Modgil, 2017). The upside of these strategies for value embedding is their easy explainability and transparency. On the downside, they require a lot of explicitly encoded information, as well as complete access to the inner architecture of the agents. This perk might not be available in some cases, particularly when the host of the platform where agents interact is not in charge for the development of the participating agents.

To address this shortcoming, an alternative approach focuses on the design and implementation of adequate norms for value promotion. Norms in multiagent systems are viewed from one of two perspectives: *conventions* and *prescriptions* (Conte & Castelfranchi, 1999; Grossi et al., 2012). Conventions are patterns of behaviour that spread through a population and emerge as the dominant agent strategy (Morris-Martin et al., 2019), often following an evolutionary process (Sandholm, 2009). Hence, conventions are part of the agents' internal constructs that emerge as the result of an adaptation process, without a central authority involved in the adoption of the convention.

In contrast, prescriptive norms are obligations, prohibitions and permissions that provide guidance on the behaviour of agents. This guidance may be either regimented, where forbidden behaviours are rendered unavailable when the prescription is implemented; or non-regimented, where agents have the ability to disobey a rule, though they might be encouraged to abide by it through some monitoring and sanctioning mechanism (Morris-Martin et al., 2019). Although work on prescriptive norms usually sticks to one of two models, regimented norms can be viewed as an extreme case of non-regimented norms. For example, the representation used by Fagundes et al. (2016) includes a detection probability for every norm. A norm can be made regimented by imposing perfect enforcement, i.e. probability of detection equal to 1. Overwhelmingly, prescriptive norms are synthesised and imposed on the agents forming a MAS by a central authority or mechanism (see e.g. all works cited in the following paragraph), according to some notion of optimality.

Originally, the purpose of prescriptive norms (also referred to as *social laws*) was to ensure conflict-free, coordinated operation of a team of robots (Shoham & Tennenholtz, 1995; Onn & Tennenholtz, 1997). Subsequent more contemporary solutions to achieving coordination through rules include online design that modifies and refines the norms in place at run-time in an open MAS (Morales et al., 2013), and guarantees on the evolutionary stability of the resulting normative system (Morales et al., 2018).

Despite their popularity as coordination mechanisms, the leverage of prescriptive norms as an avenue to embed moral values into a MAS is only recently being explored. Most notably, work by Serramià et al. has tackled this problem both from a qualitative (Serramià et al., 2020) and a quantitative utility-based (Serramià et al., 2018) perspective. Theirs is the work most similar to the one we present here. A key difference should be noted between their approach and ours. Serramià et al. implicitly assume a *deontological* view of the norm-value relationship. The values supported by every candidate norm are encoded in a value support function, without further justification. This function is then fed as an input to the problem-solving algorithm, which is responsible for finding the most value-aligned set of norms under some consistency requirements. This deontological view is also implicitly adopted by Ajmeri et al. (2020). In their work, personal assistant agents reason about the norms in place, and the values and goals of the user (including preferences over values) to select ethically appropriate actions.

In contrast to their approach, in this work we take an explicitly *consequentialist* view of the norm-value relationship. We claim that the support that a norm has for a value has to be empirically assessed by the outcomes that are brought about by it. Hence, computation of value alignment is based on features of the MAS state that serve as proxies for the value under examination.

In this paper, we extend our previous work on the synthesis of value-aligned normative systems (Montes & Sierra, 2021) on several fronts. First, we fully develop the philosophical foundations on values and norms (Section 3) that underlie many of the technical decisions made later on. Second, we exemplify value alignment with respect to an aggregation of two values, in addition to alignment with respect to both values separately (Section 4.2). Third, we expand the discussion on the Shapley value of individual norms (Section 6) by examining its properties from the cooperative game theory literature, and assessing which of these are relevant in our value alignment context. Finally, we also expand the discussion on value compatibility (Section 7) by introducing the Compatibility Maximising Normative System (CMNS) and running an optimisation search to find it in the context of our running example.

3. Underlying Assumptions

Before jumping to the technical part of the paper, we present the philosophical foundations that underlie our formal model of value alignment. With this exercise, we intend to build robust foundations for our computational model and provide sound justifications for the choices we will make during its formulation (Section 4). Our formalisation of values is built on two main points. The first relates to the concept of values as formal objects and their function within a society of agents. The second concerns the concept of norms, their function and their relationship to values.

3.1 The Nature of Values

First, we present our assumptions on *values*. Values are very abstract concepts that have been the object of intense study in philosophy for centuries (Macintyre, 1998). Currently, one of the most widely accepted theories of moral values in psychology and sociology is Schwartz's theory of basic human values (Schwartz, 1992, 2012). The main success of this

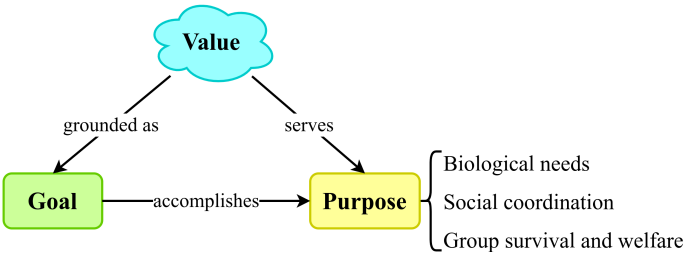


Figure 1: The three components of Schwartz's theory of basic human values and their relationships.

framework has been the identification of a spectrum of moral values that is universally reproduced across cultures. Nevertheless, the conceptualisation of values that it works with is also quite standard across the social sciences and humanities, and explicitly states the characteristics of values that are often implicit in other theories of morality (Rokeach, 1972; Feather, 1995; Spates, 1983). Schwartz's theory establishes the main features of *values* in relation to the *goals* that instantiate their meaning in a particular context and the ultimate *purpose* that they serve. The three concepts are deeply interrelated and their connections are displayed in schematic form in Figure 1.

The features of values that Schwartz's theory outlines and that are relevant to our formalisation are: values (1) are concepts or beliefs; (2) transcend specific situations; (3) refer to desirable goals, end states or behaviours; and (4) serve as evaluation criteria. Features (1) and (2) establish the nature of values and their relationship to physical (or virtual) reality. Values are general, abstract guiding principles (note the cloud-shaped box in Figure 1) that are not linked to any particular social context. Values are omnipresent constructs, that may or may not be relevant to the current context. When a value is relevant to a specific situation, it instantiates an explicit *goal* (bottom left in Figure 1) whose attainment helps further that value. For example, value "equality" in a tax policy context can refer to the effective redistribution of wealth, while in a domestic context it might relate to the even split of house chores between partners. Although the value itself remains unchanged and it is relevant to both this scenarios, its "content" is tailored to the scenario it applies to. To express the relationship between an abstract value and the meaning it takes in a particular context, we say that *an explicit goal g grounds the semantics of value v in context C*.

Features (3) and (4) state the usage of values. Essentially, values serve as moral measuring standards to make judgments about the outcome of a plan, the current state of the world and/or the actions that lead to it. All of these judgments evaluate adherence to a moral stance: whether they respect, uphold and promote the value of interest. Of course, the specific criteria that evaluates whether an action or a situation complies with a value depends on the context where the judgment is made. Since values are instantiated as situation-dependent goals, in order to assess whether the current state of the world or a particular strategy adhere to a value, we should examine how close they are or lead to the goal that is grounding the meaning of the value in that context.

In summary, values are abstract concepts that become operational in the form of explicit goals in a particular context to enable judgments on the world around us. However, beyond values and their grounding goals, there is another fundamental aspect of values left: their *purpose*. According to Schwartz's theory, values are socially desirable constructs to help us cope with three requirements of human existence: the biological needs of individuals, requisites of coordinated social interactions, and survival and welfare needs of groups. Luckily, many of us do not have to think about sheer survival when making everyday decisions. Rather, we justify our actions in terms of our moral values, even if by acting ethically we are, down the line, engaging in beneficial behaviour from an evolutionary perspective.

We argue that, of the three concepts discussed thus far, the *goals* that capture the meaning of values are the best avenue to derive a mathematical formalisation to be embedded into an artificial MAS, for three reasons. First, although values are very abstract, their grounding goals are not, and their fulfilment can be empirically evaluated. Second, we choose goals over purpose since, for technically enabled agents, concerns about evolutionary survival do not really apply. Third, by modelling the goals that ground values, we grant complete control on the *meaning* of values over to the system designer. Agents do not learn how to gradually de-abstract values depending on the context they are in. Rather, the designer is responsible for deciding how does a particular value manifest in the context where the MAS will be operating.

3.2 The Role of Norms

Now that we have established the characteristics of values and how we intend to represent them, we turn to the other main protagonist of this work: norms. As introduced early on, we approach norms from the prescriptive perspective: rules and regulations handed out by a central authority or system designer, that dictate or provide guidance the behaviour of agents, and that affect the outcomes they are able to achieve. This view of norms as prescriptions is differentiated to the view of norms as conventions, where they refer to socially acceptable and expected behaviour emanating from the agents themselves. From here on, whenever we use the word "norms", we are talking about prescriptions.

In this work, we accept the possibility of norms to be *regimented*, i.e. perfectly enforceable. Although regimented norms are not very representative of real-life interactions, in virtual environments they are often technically possible to achieve. In fact, in our running example we will assume norm regimentation, so that we do not need to add extra degrees of freedom related to the agents' decision-making models.

Norms play a central role in our value alignment model because, in our view, they have the potential to ensure ethically compliant outcomes in a society of agents. By modifying incentives and providing opportunities, norms have the ability to steer a community of agents towards particular outcomes. When norms are implemented, the incentive structure and constraints they impose on the agents causes some states to be promoted over others, by making them obligatory or more desirable (e.g. by assigning a penalty to alternative states). Hence, it is expected that some states will be more likely and easily achievable than others. Consequently, the resulting outcome (i.e. the last state visited by the MAS) is strongly dependent on the norms currently implemented. If norms are carefully designed, they facilitate the achievement of outcomes where the goals that ground the meaning of

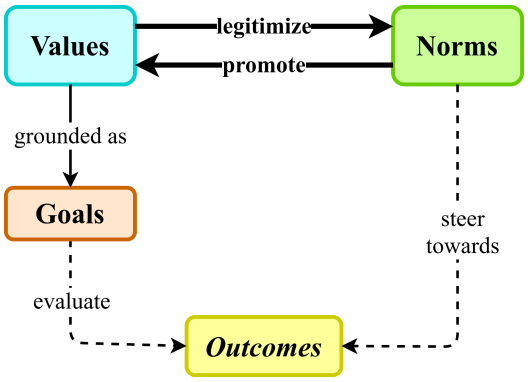


Figure 2: The relationship between *norms* and *values* goes through the grounding *goals* and the achieved *outcomes*.

values are fulfilled to a greater extent than if no regulations were imposed. When a set of norms are successful in this endeavour, we say that they are *aligned with respect to the values* they sought to promote. Hence, we conceive alignment as being a property of norms with respect to values.

Figure 2 illustrates the relationship between values and norms as a diagram. At a shallow level, the two form a feedback loop: norms promote values, and values legitimise the enforcement of norms. At a deeper level, however, the relationship between norms and values is held together by the outcomes that norms steer the system towards and that are favourably evaluated by values. Note that, while norms are context-dependent (they are crafted with a particular domain, e.g. online trade, in mind), values are not. Therefore, values should first be de-abstracted into their grounding goals. Then, these goals can be employed as the standards for which the eventual outcomes (and consequently the norms that lead to them) are held against.

4. Formal Model of Value Alignment

Our computational model to quantify the alignment of a set of norms (or normative system) with respect to some value is inspired by Sierra et al. (2019). In the present framework, a MAS consists of a set of agents G that interact with one another and their environment. The world is modelled as a labelled transition system (Gorrieri, 2017):

Definition 1. The *world* is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, where $\mathcal{S}$ is a set of *states*, $\mathcal{A}$ is a set of *actions* and $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a set of *transitions* between states labelled by an action.

At any point in time, the MAS is in state $\mathbf{s} \in \mathcal{S}$. This is understood as a *global state*, encoding all the information there is to know about the system. Among $\mathcal{S}$, the initial state is denoted by $\mathbf{s}_0$. Changes in the global state of the system are brought about by the actions $\mathcal{A}$ that agents take. Just as $\mathbf{s}$ is understood to be the global state of the system, action $a \in \mathcal{A}$ refers to the *joint* action profile that is executed by the whole of the agents. Executing joint action a in state $\mathbf{s}$ moves the system to a new state $\mathbf{s}'$. We denote transition $(\mathbf{s}, a, \mathbf{s}')$ with the notation $\mathbf{s} \xrightarrow{a} \mathbf{s}'$.

To simulate long-term evolution of the system, single transitions are not enough. Rather, several transitions are concatenated forming a *path*:

Definition 2. A *path* $p = \mathbf{s}_0 \xrightarrow{a_1} \mathbf{s}_1 \xrightarrow{a_2} ... \xrightarrow{a_n} \mathbf{s}_n$ over the world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$ is a sequence of transitions in $\mathcal{T}$ between states in $\mathcal{S}$, starting at initial state $\mathbf{s}_0$.

We denote the set of paths over the world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$ of length n (visiting $n+1$ states) as $\mathcal{P}_n$. The set of all paths (of any length) over the world is denoted as $\mathcal{P}$. Given a path $p \in \mathcal{P}$, the function $\mathsf{out} : \mathcal{P} \to \mathcal{S}$ returns the last state (or the *outcome*) of path p, $\mathsf{out}(p) = \mathbf{s}_n$.

Running Example

We illustrate all of our contributions with a running example of a toy social model, reminiscent of the public goods game. In this society, a set of technologically-enabled agents are endowed with some initial wealth. To facilitate the exchange of resources, a common fund is set up. Agents contribute to the common fund with a tax amount that is dependent on their economic status. However, a subset of evader agents try to skip the payment. They might get caught and obliged to pay the original taxes plus an additional fine. Finally, the common fund is invested, grown by a fix amount and redistributed back to the agents, regardless of their evading tendencies, in a way that is also dependent on the economic status of the individuals.

In our running example, the MAS is composed of a set of 200 agents $G = \{1, 2, ..., 200\}$. Of these 200, we set 10 to be "evader" agents, which will try to skip payment when the tax collection and redistribution scheme is imposed. This composition of the society (10 out of 200 evader agents) is an arbitrary choice made for exemplification purposes.

All agents have some wealth, which is initialised randomly according to a uniform distribution between 0 and 100. At any given time-step, every agent is characterised by its current wealth x_i as well as an integer that denotes the wealth segment it belongs to. To find the wealth segments, all agents are ranked from highest to lowest wealth and then split into 5 equally populated groups. The agents that belong to the poorest group are assigned to segment #1, while those in the wealthiest group are assigned segment #5. Therefore, an agent at any state is characterised by the tuple $(x_i, seg_i) \in (\mathbb{R}^+ \times \{1, 2, 3, 4, 5\})$, where x_i is agent i's wealth (which is always non-negative) and seg_i is the wealth segment or group it is assigned to. Consequently, the global state space is $\mathcal{S} = (\mathbb{R}^+ \times \{1, 2, 3, 4, 5\})^{|G|}$, where $|G|$ is the number of agents (200). In principle, in the unregulated situation agents can exchange money as they see fit at any transition.

4.1 Norms

In an unregulated world, actions may lead to undesired outcomes. As we have already argued, to avoid this we introduce normative systems as a way to steer the system away from harmful results and in the direction of valued outcomes.

Normative systems are composed of individual norms. An individual norm n_i is a regulation that targets a specific aspect of a state transition. The distinction between different aspects of a state transition will be clarified when we go through the norms in the running example. For the time being, we simply consider a norm to be a regulation that constraints how transitions $\mathcal{T}$ between states in $\mathcal{S}$ take place. The complete transition is

determined by the norms plus the decision-making models of the agents, who choose among the actions available to them.

In this paper, we work with parametric norms. Every n_i is linked to a set of *normative parameters* P_i, alongside with their domains and any possible constraints. When we talk about norms, we refer to the uninstantiated parameters, with no quantities assigned. As an instance closely related to our running example, consider income taxes, which partially determine your wealth increment from one month to the next. In many countries, income tax rates are regulated by a norm that taxes in an incremental way with respect to salary. To instantiate it in a concrete case, numerical parameters have to be set to, for example, decide how the income range is divided into several groups and which rate is applied to every group.

A normative system N is a set of individual norms $\{n_i\}$ with all of their parameters instantiated to some quantity, respecting any domain-dependent constraints. Every norm in N is responsible for regulating an aspect of a transition between two states. Together, the application of the norms in N to the world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$ restricts the original set of transitions to a subset of those, $\mathcal{T}^N \subseteq \mathcal{T}$. Consequently, the set of possible paths $\mathcal{P}$ is also restricted to a subset of the original, $\mathcal{P}^N \subseteq \mathcal{P}$. We use notation $\mathbf{s} \xrightarrow{N} \mathbf{s}'$ to denote the transition between two states as regulated by normative system N.

Note that individual norms by themselves do not determine transitions. Rather, a coalition of (in general) several norms is necessary. Also note that the individual norms that make up a normative system are fully instantiated, as all the parameters they depend on are assigned a quantity. Any normative system N, then, belongs to the family $\mathcal{N}$ of all normative systems composed of the same set of norms but whose parameters take different quantities. The dimensionality of $\mathcal{N}$ depends on the number of parameters needed to specify all norms of a normative system. Essentially, the family of normative systems $\mathcal{N}$ defines a search space consisting of the domains of the normative parameters of its individual norms.

Our methodology does not indicate how should the various aspects of state transitions be itemised into several norms. This is an engineering choice that must be made by the system designer. There is, nonetheless, a guideline that we believe should be respected when formulating the normative system. If there are constraints involving several parameters, these parameters should all be related to the same individual norm. An example of this restriction being respected can be found next, in norm n_2 of our running example.

Running Example

In our social model, we introduce a set of norms (1) to regulate the collection and redistribution of taxes and (2) to randomly detect evader agents and impose a fine on top of their taxes. Thus, within this model, transitions happen under the regulation of normative system $N = \{n_1, n_2, n_3, n_4\}$, where:

- $\mathbf{n_1}$ is the norm specifying the tax rate for every wealth segment. It is parametric on the set $P_1 = \{collect_j\}_{j=1,\ldots,5}$, where $collect_j$ corresponds to the fraction of their current wealth that every member of wealth segment j must contribute to the common fund at every transition. All $collect_j$ components have their values bounded in the range $[0, 1]$.

n_2 is the norm specifying how should the invested funds (which grow by a fixed 5% rate) be redistributed back to the agents. It is parametric on the set $P_2 = \{ redistribute_j\}_{j=1,...,5}$, where $redistribute_j$ corresponds to the fraction of the invested common fund that is allocated to the j-th wealth segment. This reimbursement is then shared equally among all agents in the group. Again, all $redistribute_j$ components are bounded in the range $[0, 1]$. Also, the following linear constraint holds:

$$\sum_{j=1}^{5} redistribute_j = 1 \tag{2}$$

implying that the totality of the common fund is reimbursed back to the agents.

n_3 is the norm that determines the probability of detecting evaders, who at every transition attempt to skip payment. It is parametric on a single value $P_3 = \{catch\}$, which corresponds to the probability that any evader agent will be detected and be made to pay its corresponding taxes plus an additional fine on any state transition. To emulate the struggle of fiscal authorities, the range of *catch* is bounded in $[0, 1/2]$, despite corresponding to a probability that could in principle take values in the range $[0, 1]$.

n_4 is the norm specifying how harsh should the punishment be on the detected evaders. Similarly to n_3, it is parametric on a single value, $P_4 = \{fine\}$. Whenever an evader is caught, the amount that it is obliged to contribute to the common fund is equal to the taxes it was trying to evade in the first place (that are dependent on its wealth segment), plus the additional fraction given by *fine*. If the total payment would result in the agent having to contribute with an amount greater than its total current wealth, then the payment equals to the totality of the current wealth of the agent. The value of *fine* is bounded in the range $[0, 1]$.

This example shows how to define a parametric normative system as a set of norms, each of them targeting a concrete aspect of the system's transitions. The family of normative systems $\mathcal{N}$ we work with is the one with components $\{n_1, n_2, n_3, n_4\}$. Our search space, then, is determined by the domains of the normative parameters $P_N = \{collect_j, redistribute_j, catch, fine\}_{j=1,...,5}$ plus the constraint in eq. (2). It should be noted that, despite bearing a remote resemblance with real-life tax codes, the example model is not intended to be a reliable reflection of actual tax policy. It is just a simple example to illustrate our methodology in action.

Second, in our example we do not consider any reasoning schemes by the agents. Evader agents always attempt to evade their payment, while non-evader agents are always compliant and pay their part. This choice is not a feature of our general methodology, but of this example only. We have made this choice in order to reduce the degrees of freedom of our example and focus on the main topic of this work, the synthesis of value-aligned norms.

4.2 Value Alignment

So far, we have modelled a society as a set of agents that transition between states by executing actions. Also, we have introduced norms as restrictions on the feasible transitions.

Now, we want to quantify how effective are those norms at promoting the values we want to embed in the system, i.e. how well *aligned* they are with respect to some values of interest.

In the underlying assumptions of our model (Section 3), we established that values are grounded as goals that evaluate, among other objects, the outcomes that the system achieves. Mathematically, we model these grounding goals as functions over the states of the system that ought to be maximised. As the state space $\mathcal{S}$ is different depending on the MAS in question, the goal that encapsulates the meaning of any value has to be defined in terms of features of that state space, and will not, in general, be applicable to other contexts.

Definition 3. Given a world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$ and a set of values V, the *semantics* of value $v \in V$ in the world is a function $f_v : \mathcal{S} \to [-1, 1]$ that evaluates the states of the world, where $f_v(\mathbf{s}) \sim -1, 0, +1$ indicates that state $\mathbf{s}$ strongly opposes, is neutral or strongly promotes value v, respectively.

Note that Definition 3 entails, as anticipated in Section 3.1, that values need first to be de-abstracted into a function that captures its meaning for the particular domain at hand. Our approach demands this step, and is unable to work with abstract values that have not been grounded first.

As the semantics of any value are given by a function that grades the states of the world, the goal capturing the meaning of the value, then, aims at maximising the corresponding function, by achieving a state that is as compliant towards the value in question as possible. Often, however, one is not just interested in promoting a single value, but rather would like to achieve compliance with respect to several values. Given a set of values $V = \{v_1, ..., v_m\}$ and their semantics functions $f_1, ..., f_m$, we propose that the semantics of the set of values in V should be grounded by an *aggregation* function $F_V : [-1, 1]^m \to [-1, 1]$. F_V takes in the compliance with respect to every individual value and merges them into a single metric. To shorten notation, we denote $F_V(f_1(\mathbf{s}), ..., f_m(\mathbf{s}))$ as $F_V(\mathbf{s})$.

We set the range of all value semantics functions to be bounded in order to facilitate comparison between values. This will become particularly relevant in Section 7, when we look into incompatibilities between values. If the range was unbounded, it would be difficult to establish which value is being actually more aggressively pursued. Therefore, it is highly convenient to grade states in a continuous bounded domain. The choice of the bounds at ± 1 is made out of convenience for the ease of working with unit quantities.

In summary, values evaluate states. How to extend such an assessment to the norms in place? Figure 2 gives a clear hint. Norms steer the system towards (hopefully) beneficial outcomes, that are assessed with the semantics function that capture the meaning of a value in that particular world. Mathematically, we understand an *outcome* to be the final state of a path p in the world, i.e. $\mathsf{out}(p)$. Since norms restrict the available paths, they also limit the final states where they can end. And, just like any other state, the outcome of a path can be graded according to the semantics function of a value.

We put all of these ideas together to define the alignment of a normative system with respect to a value:

Definition 4. Given a world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, a normative system N that applies to it, and a set of values $V = \{v_1, ..., v_n\}$ with semantics functions $f_1, ..., f_n$, the *alignment* $\mathsf{Algn}_{N,v}$ *of*

normative system N with respect to value $v \in V$ is computed as:

$$\mathsf{Algn}_{N,v} = \mathbb{E}\left[f_v\left(\mathsf{out}(\mathcal{P}^N)\right)\right] \tag{3}$$

where $\mathcal{P}^N$ is the random variable of the subset of paths restricted under the normative system N.

Equation (3) states that, in order to compute the alignment of a set of norms with respect to some value, the evolution of the system under the norms in N has to be simulated and let to achieve some outcome. Then, this final state is assessed in terms of the meaning that value v takes in the world, the semantics function f_v. In order to have a statistically significant quantity, the expected value should be computed over a sufficiently large random sample of norm-regulated paths using, e.g. Monte Carlo sampling (Lemieux, 2009). For computational convenience, we also propose to restrict the length of the sampled paths to a fixed number, and hence compute the expected value over $\mathcal{P}^N_n$, for some fixed n.

In line with the consequentialist view we present in Section 3, the alignment in eq. (3) is computed by considering the ultimate consequences (i.e. the end-state) that the implementation of a set of norms brings about. However, other alternatives are possible. For example, one may prefer to compute the alignment by considering the value semantics function applied to all the states that are visited during a path. In that case, one would also need to specify how to aggregate the evaluation of f_v over all the visited states (e.g. average, minimum...).

Once the alignment for a normative system has been established in absolute terms, we can compare several norm sets with one another. To do so, we define the *relative alignment* between two normative systems:

Definition 5. Given a world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$, two normative systems N_1 and N_2 that apply to it, and a value v with semantics function f_v, the *relative alignment between N_1 and N_2* $\mathsf{RAlgn}_{N_1/N_2,v}$ *with respect to value v* is computed as:

$$\mathsf{RAlgn}_{N_1/N_2,v} = \mathsf{Algn}_{N_1,v} - \mathsf{Algn}_{N_2,v} \tag{4}$$

Equations (3) and (4) can be readily extended to compute the (relative) alignment with respect to a set of values V. Instead of f_v, the evaluation of path outcomes is made with an aggregation function F_V.

Running Example

In our example tax model, we are interested in the two values $V = \{equality, fairness\}$. We define semantics functions with respect to the two values individually and for their aggregation as well. These two values will exemplify goals that are not correlated and that are achieved through different taxing strategies.

First, we ground the meaning of value *equality* in the context of our model as, of course, economic equality. To do so, we use the well-known Gini index indicator (Gini, 1912), a widespread metric to quantify wealth and income inequality (The World Bank, Development Research Group, 2019). The Gini index is bounded between 0 (for perfect equality) and 1 (for perfect inequality). In our model, we consider that a normative system N is highly

aligned with respect to equality if, by the end of a fixed-length path, the Gini index is as low as possible:

$$f_{eq}(\mathbf{s}) = 1 - 2 \cdot GI(\mathbf{s}) \tag{5}$$

where $GI(\mathbf{s})$ is the Gini index for the wealth distribution at global state $\mathbf{s}$, which is computed as:

$$GI(\mathbf{s}) = \frac{\sum\limits_{i,j \in G^2} |x_i - x_j|}{2 \cdot |G|^2 \cdot \bar{x}} \tag{6}$$

where x_i denotes the wealth of agent $i \in G$ and $\bar{x}$ the average of the distribution. Note that in eq. (5) we introduce an affine transformation in order to map perfect equality ($GI \sim 0$) to maximum alignment ($f_{eq} \sim 1$) and perfect inequality ($GI \sim 1$) to minimum alignment ($f_{eq} \sim -1$).

Second, we ground the semantics of value *fairness* to mean that evader individuals should be punished for their evasion. Hence, we consider that fairness is being highly promoted if, by the end of a fixed-length path, as many evaders as possible belong to the poorest wealth segment:

$$f_{fair}(\mathbf{s}) = 2 \cdot \hat{\mathbb{P}}\left[seg_i = 1 | evader_i\right] - 1 \tag{7}$$

where $\hat{\mathbb{P}}\left[seg_i = 1 | evader_i\right]$ denotes the estimated probability that the wealth segment of an agent i is the lowest one, provided that they are an evader agent. This estimation is computed as the proportion of evader agents in segment 1 at the final global state. Again, an affine transformation is introduced to map the probability range $[0, 1]$ to the alignment range $[-1, 1]$.

Given that, in our virtual society, there are more agents per wealth group at any time-step (40) than evader agents (10), in the best-case scenario all evaders would end up in segment #1. Consequently, the upper bound for function (7) is $+1$. If there were more evaders than agents per wealth segment, the semantics function would need to be modified so that the potential maximum alignment does not fall below 1.

Third, we turn to the alignment for the aggregation of both values *equality* and *fairness*. Here we take a demanding position. We consider that the set of values $V = \{equality, fairness\}$ are being upheld overall when both values are being simultaneously promoted:

$$F_V(\mathbf{s}) = \begin{cases} -f_{eq}(\mathbf{s}) \cdot f_{fair}(\mathbf{s}) & \text{if } f_{eq}(\mathbf{s}) < 0 \text{ and } f_{fair}(\mathbf{s}) < 0 \\ f_{eq}(\mathbf{s}) \cdot f_{fair}(\mathbf{s}) & \text{otherwise} \end{cases} \tag{8}$$

where $f_{eq}(\mathbf{s})$ and $f_{fair}(\mathbf{s})$ correspond to the semantics functions in eqs. (5) and (7) respectively. The piece-wise definition of $F_V(\mathbf{s})$ is necessary in order to avoid that negative alignment with respect to both *equality* and *fairness* would result in positive alignment with respect to their aggregation.

5. Search of Optimal Normative Systems

The purpose of this section is to demonstrate how to find optimally aligned normative systems with respect to some values, as stated in eq. (1). The search space is determined by the normative parameters, with their domains and constraints, and the objective function

to optimise corresponds to the alignment with respect to the value of interest, computed by eq. (3) as the expectation of the value semantics function on a sample of outcome states.

To perform the search, in this paper we use a Genetic Algorithm (GA) as our optimisation method. This choice, however, is not a defining feature of our methodology. Any optimisation strategy that is suitable, given the domains and constraints of the normative parameters, is apt to perform the task.

Genetic Algorithms (Luke, 2013) are a family of versatile search methods where a population of candidate solutions is maintained. Given the nature of our problem, a candidate in this population consists of a fully instantiated normative system, with a numerical quantity assigned to all of its parameters. Candidates are selected for breeding based on their fitness, i.e. how well aligned is the N instance with respect to the value for which we are optimising. A crossover operation is performed over highly aligned normative systems, in hopes of generating two even better instances. When enough offspring are generated, they substitute the original population. The process is repeated iteratively until some stopping criteria is met.

We initialise our population of candidate normative systems randomly within the bounds allowed for every parameter. For the selection with replacement step, we employ the 1 vs. 1 tournament technique (Miller & Goldberg, 1995). Two normative systems are drawn at random from the population, and the fittest of the two is selected for crossover. This step is repeated once more, to select the other parent to breed with the winner of the previous tournament.

Typically, the crossover step is performed with bit-wise operations, since GAs are mostly employed in optimisation tasks over discrete search spaces. In the case concerning this work, however, we are dealing with a continuous search space defined by the bounds and constraints of the normative parameters as described in Section 4.1. To handle this, we turn to a crossover technique suitable for continuous spaces, intermediate recombination (Mühlenbein & Schlierkamp-Voosen, 1993). This method is controlled by hyperparameter $p \geq 0$, which determines the explorability of the search. The larger p is, the more exploratory the search is. Additional tweaks are introduced in order to ensure that the linear constraint (2) is satisfied.

To enhance the exploitability of the search, we introduce elitism (Baluja & Caruana, 1995) into the algorithm. This technique consists of replacing a small number of the worst newly generated candidates with the same number of the best aligned candidates from their parent generation. This popular variant of genetic optimisation guarantees that the alignment of the most promising normative system will not decrease from one generation to the next. We denote by k the number of "elite" candidates that are carried over from one generation to the next.

For the stopping criteria, we take advantage of the fact that the target alignment functions are bounded up to 1, and hence we set the search to stop when a very promising normative system instance with alignment over some high threshold is found. In order to avoid excessively long searches, the algorithm is halted after a large number of total iterations, and a moderate number of partial iterations, i.e. rounds of the search for which the most promising candidate is not updated. An overview of the hyperparameters of our GA implementation is presented in Table 5 in Appendix A.

Table 1: Optimisation results for the running example with respect to the two values of interest plus their aggregation: optimal normative parameters defining the normative system, and their associated optimal alignment.

Value and semantics function	Optimal normative parameters	Optimal alignment $\text{Algn}^*_{\text{N,v}}$
Equality, eq. (5)	$collect = [20\%, 29\%, 26\%, 35\%, 27\%]$	0.95
	$redistribute = [20\%, 22\%, 19\%, 26\%, 13\%]$	
	$catch = 44\%$	
	$fine = 61\%$	
Fairness, eq. (7)	$collect = [1\%, 30\%, 37\%, 72\%, 66\%]$	0.93
	$redistribute = [2\%, 23\%, 42\%, 24\%, 9\%]$	
	$catch = 45\%$	
	$fine = 56\%$	
Aggregation, eq. (8)	$collect = [2\%, 79\%, 56\%, 65\%, 59\%]$	0.66
	$redistribute = [2\%, 28\%, 25\%, 35\%, 10\%]$	
	$catch = 31\%$	
	$fine = 77\%$	

Other than their versatility (for example in adapting the search to a continuous search space like ours), GAs are particularly attractive for the task at hand because they do not require the analytical formulae of the gradient of the target function with respect to their arguments, i.e. the normative parameters in our case. In other search methods, particularly those based on gradient ascent/descent (Ruder, 2016), disposing of the gradient function is extremely desirable, if not outright necessary. Yet, given the nature of the problem we are tackling, obtaining such an expression is an unnecessarily demanding task, as it would require deriving an expression of the alignment explicitly as a function of the normative parameters.

Although our running example is relatively small, we resort to inexact methods for the optimisation task. In fact, we would encourage readers interested in implementing this methodology for larger, more complex scenarios to stick with meta-heuristics methods (like the GA implemented here or simulated annealing), precisely because they do not impose any requirements of the optimisation target concerning continuity or differentiability. Regardless of the domain on is working with, the optimisation target that this methodology needs to maximise is the alignment, which is empirically estimated by generating a sample of simulation runs and evaluating their outcomes. Hence, this optimisation target is not differentiable. However, note that the computational requirements for the optimisation search are expected to grow with the size of the social model one is working with. Deriving this growth in resources for the optimisation search as a function of the size of the MAS is object for future work.

Running Example: Optimisation Results

In the context of our running example, searching the optimally aligned normative systems means finding which taxing policies maximise the promotion of values *equality* and *fairness* (and their aggregation), according to the meaning we have imbued in these values in eqs. (5) and (7).

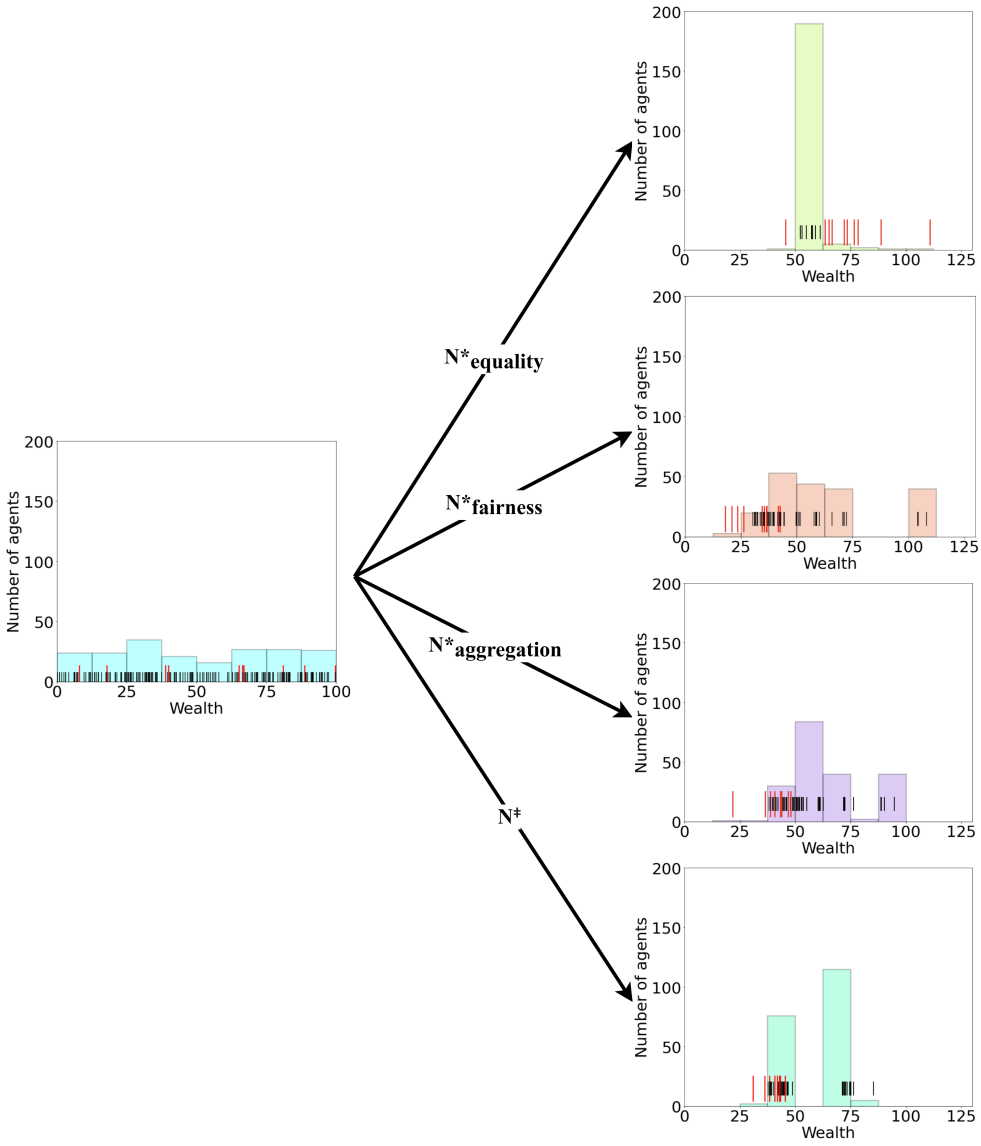


Figure 3: Wealth distribution and rug plot indicating the location of law-abiding agents (regular black marks) and evaders (longer red marks), at the initial state (left) and after a sample path of 10 transitions under the optimal normative system for equality (right first), for fairness (second), for their aggregation (third) and for the normative system that preserves the maximum compatibility between the two values (fourth, see Section 7).

Table 1 presents the optimisation results for the two values we have modelled (*equality* and *fairness*) plus their aggregation. The optimal alignment for equality and fairness separately are very satisfactory, with large positive values > 0.9. For the aggregation of values, the optimal alignment is still fairly good, over 0.6. This decrease in the optimal alignment

for the aggregation with respect to the values individually speaks to how demanding the aggregation of values in eq. (8) is, as it is necessary that both equality and fairness be promoted to a large degree simultaneously in order for the alignment with respect to their aggregation be also high.

We provide an intuitive interpretation of the optimal normative parameters obtained. For equality, the differences between the components of *collect* are small across wealth groups. In practice, this means that wealthier agents contribute to the common fund with more resources in absolute terms, as their wealth is larger. The even redistribution rates across groups then ensure that all agents receive a similar portion of the invested funds. The moderate values for the *collect* and *redistribute* parameters in the optimal model with respect to equality correspond to a compromise between funnelling enough resources from rich to poor agents in order to shrink the wealth distribution, but not channelling too many as to swap them, which would be detrimental towards lowering the Gini index.

For fairness, the normative parameters indicate that another mechanism is in place in order to push evaders towards the poorest group. It is worth noting that neither the probability of catching evaders nor the fine they are imposed are particularly large, they are similar or even smaller than those found for the optimal normative system with respect to equality. Rather, it appears that evaders are pushed towards group #1 by retrieving a lot of resources from the upper wealth groups, where undetected evaders manage to sneak, and then redirecting them towards the middle class. This middle segment is vastly composed of law-abiding citizens, since detected evaders belong to the lower groups and undetected ones belong to the upper ones. Hence, the norms enforce fairness by identifying the wealth groups most likely to include cheaters and directing their wealth elsewhere. It does not collect many taxes from group #1, but the norms keep the cash flow in and out of that group very limited, so that already poor evaders do not have any avenue to enrich themselves.

The optimal parameters with respect to the aggregation of the two values are the most difficult of the three to interpret, since they have to achieve a compromise between effectively punishing evaders and keeping wealth inequality under control. On one hand, the cash flow in and out of wealth group #1 is extremely limited (both $collect_1$ and $redistribute_1$ are ~ 0.02). Since this feature is common to the optimal parameters with respect to fairness, we suspect that its function is similar to the one it played in the optimal norms for fairness only: keep evaders in the lowest wealth segment once they have been detected and pushed there. Additionally, the fine rate to achieve the aggregation of both values is the highest across all optimal normative systems, close to 80%. This indicates that in order to achieve fairness without hurting equality too much, the norms promoting the aggregation of the two values rely more on fines that exclusively target evaders in order to punish them, while the normative system optimised for fairness alone did not consider the collateral harm it could inflict on non-evader agents.

On the other hand, the trends of the *collect* and *redistribute* parameters for the wealth segments other than #1 are somewhat parallel between the optimal normative system for equality alone and the aggregated values. This observation seems to indicate that equality is mostly promoted through a similar taxing strategy to the case when it was only that value being considered.

Figure 3 provides a visual representation of the evolution of the society under each of the three optimal normative systems.[1] These plots visually transmit the fact that the different norm sets lead the system towards different outcomes. Clearly, the final distribution under the optimal normative system for equality is very narrow, but does allow most evaders to become the richest individuals in the population. On the contrary, the final wealth distribution under the optimal norms for fairness is much broader but does push evaders towards the lower positions. The final distribution under the optimal norms for the aggregation of the two values is somewhat in the middle, with a compromise between a moderately narrow wealth distribution and pushing evaders towards the lowest positions in the wealth ranking.

6. Shapley Values of Individual Norms

So far, we have been able to synthesise, in an automated fashion, normative systems that are optimally aligned with respect to values. We have illustrated the methodology in the context of our running example. This synthesis is always contingent on the understanding we have of those values in the context where the MAS operates. Now, the second part of this work begins, where we provide an analytical toolbox to examine the optimal normative systems we have attained more closely. These tools are aimed at providing insights to the system designer on the output of the synthesis process. In particular, we quantify the contribution of every individual norm in an optimal normative system to the overall alignment through their *Shapley values* in this section. In Section 7 we examine *value compatibility*, i.e. how successful an overall set of norms is at compromising between competing values. These metrics should help the system designer reflect on the engineering choices made prior to the automated search and inform any changes to the choice of normative parameters and/or alignment function, before iterating the synthesis-analysis process until satisfactory results are obtained, both regarding a high degree of alignment of the norms with respect to the values of interest and acceptable metrics regarding their Shapley values and compatibility measurements.

In this section, we look at the interaction between individual norms through their Shapley values. We are interested in quantifying how much is a particular norm contributing towards the overall alignment of a normative system with respect to some value. For example, are the rate of evader detection and the fine imposed relevant when it comes to achieving equality, or are they not?

In order to quantify the importance of individual norms, we take the view of any normative system N as a grand coalition of individual norms $\{n_i\}$ working together to achieve high alignment with respect to some value. In order to allocate the credit to the different norms for achieving such promotion, we import the notion of Shapley value (Shapley, 1951) from cooperative game theory, and adapt it to our context:

Definition 6. Given a normative system $N = \{n_i\}$, a value v for which the semantics function f_v in world $(\mathcal{S}, \mathcal{A}, \mathcal{T})$ has been defined, the *Shapley value of norm n_i with respect*

1. Short videos displaying all the intermediate states between the initial and the final global states of a sample path are available, for the four normative systems in Figure 3, at `https://github.com/nmontesg/aamas21/tree/main/videos` and at the online appendix.

to value v is given by:

$$\begin{aligned}\phi_i(v) &= \sum_{N' \subseteq N \setminus \{n_i\}} \frac{|N'|!\,(|N| - |N'| - 1)!}{|N|!} \cdot \Big(\mathsf{Align}_{N' \cup \{n_i\}, v} - \mathsf{Align}_{N', v} \Big) = \\ &= \sum_{N' \subseteq N \setminus \{n_i\}} \frac{|N'|!\,(|N| - |N'| - 1)!}{|N|!} \cdot \mathsf{RAlign}_{N' \cup \{n_i\} / N', v}\end{aligned} \tag{9}$$

Definition 6 can be readily extended to a set of values V for which an aggregation function F_V has been defined.

The sum in eq. (9) is taken over all normative systems N' from which at least individual norm n_i is not included. Then, the Shapley value for n_i is computed through the relative alignment between the introduction of norm n_i ($\mathsf{Align}_{N' \cup \{n_i\}, v}$) and its absence ($\mathsf{Align}_{N', v}$). Not that if other norms besides n_i are absent from N', they are not to be reintroduced in $N' \cup \{n_i\}$.

Two issues need to be addressed to clarify how the computation ought to be performed: (i) what does it mean for an individual norm n_i to be absent from a normative system N'; and (ii) which normative systems should the sum in eq. (9) include. We address (i) first to be able to answer (ii) later.

Consider an arbitrary normative system instance N, from which we wish to remove some subset of individual norms $\{n_i, n_j, n_k ...\}$ to obtain a new normative system $N' = N \backslash \{n_i, n_j, n_k ...\}$. We denote the numerical quantities upon which norm n_i is parametric in normative system N as $P_i^{(N)}$. To proceed with the removal, we first need to introduce another normative system instance, N_{bsl}, which we refer to as the *baseline* normative system. N_{bsl} belongs to the same family of normative systems as N, meaning that it has the same set of individual norms $\{n_i\}$ related to the same parameters and subject to the same bounds and constraints. However, the numerical quantities for the parameters linked to N_{bsl} are (manually) set in a way as to reflect the lack of evolution. That is, when N_{bsl} is implemented on the MAS, the initial state remains unchanged after an arbitrary number of transitions.

Once N_{bsl} is defined, to remove norm subset $\{n_i, n_j, n_k ...\}$ from N we substitute the values of the parameters of all the norms in the removal set, $P_i^{(N)}, P_j^{(N)}, P_k^{(N)} ...$, by their baseline counterparts, $P_i^{(bsl)}, P_j^{(bsl)}, P_k^{(bsl)} ...$. Consequently, the normative parameters of system $N' = N \setminus \{n_i, n_j, n_k ...\}$ is composed of the original parameter quantities for the non-removed norms, $P_l^{(N)}$ for $n_l \notin \{n_i, n_j, n_k ...\}$, plus the baseline parameters for the removed norms, $P_m^{(bsl)}$ for $n_m \in \{n_i, n_j, n_k ...\}$.

Now that we know how to remove subsets of norms from a normative system, we have answered question (i). Essentially, a norm is absent when the quantities of the parameters it depends upon have been substituted by their baseline counterparts. Now we are in a position to provide a straightforward answer to issue (ii). The sum in eq. (9) is taken over $N' \subseteq N \setminus \{n_i\}$, i.e. all normative systems similar to the input normative system N from which *at least* the individual norm n_i has been removed. Hence, to obtain all terms in the summation, one must first remove n_i from N by substituting its parameters by their baseline. Then, substitute the parameters linked to other norms according to all possible combinations of the remaining ones $\{n_j\}_{j \neq i}$. Finally, $N' \cup \{n_i\}$ is obtained by setting $P_i^{(bsl)}$ back to the original parameter values $P_i^{(N)}$. This final step is only performed for n_i, not for

any other norms $n_j \neq n_i$ that may also be absent from N'. Therefore, the sum in eq. (9) contains $2^{|N|-1}$ terms.

Finally, we clarify that the Shapley value includes factorial terms on the size of the original normative system $|N|$ (which is a fixed quantity across all normative systems in the same family) and to the trimmed one $|N'|$. It should be noted that $|N'|$ only counts the individual norms that have *not* been substituted by baselines, as those that have been are considered as absent.

Concerning the baseline normative system, for the time being we do not provide a systematic method to find an adequate baseline given an arbitrary normative system family. The only restriction we impose is that the baseline normative system, just as all other normative systems in the same family, has to respect the domain-dependent constraints. For our running example, luckily, the simplicity of the scenario allows to define N_{bsl} from mere intuition. However, we assert that a good choice for a baseline normative system is one such that, for any sampled path, the initial global state is kept unchanged after a sequence of transitions of arbitrary length:

$$\mathbf{s_0} \xrightarrow{N_{bsl}} \mathbf{s_0} \xrightarrow{N_{bsl}} ... \xrightarrow{N_{bsl}} \mathbf{s_0} \tag{10}$$

This is a fairly demanding requirement, since we are not referring to the expectation over a sample of random paths, but to deterministic equality over every single possible path. Provided we are able to design a baseline where the above property holds, then the alignment of the baseline normative system simply corresponds to the assessment of the initial state according to the semantics function of value v:

$$\mathsf{Align}_{N_{bsl},v} = f_v(\mathbf{s}_0) \tag{11}$$

Running Example

In our example model, setting the baseline parameters can be manually made thanks to its simplicity:

$$N_{baseline} = \left\{ \begin{array}{l} n_1 \sim collect = [0, ..., 0] \\ n_2 \sim redistribute = [\frac{1}{5}, ..., \frac{1}{5}] \\ n_3 \sim catch = 0 \\ n_4 \sim fine = 0 \end{array} \right\} \tag{12}$$

Note that the choice of the *redistribute* list respects constraint (2). We have experimentally checked that this choice of parameters does indeed leave the initial global state of the system unchanged.

6.1 Properties of the Shapley Value

In the context of traditional cooperative game theory, it can be proven that the Shapley value is the only payoff distribution scheme that fulfils all of the following properties (Peters, 2008):

1. *Efficiency*: All of the payoff obtained by the coalition is allocated to some player.
2. *Null player*: If a player is *null*, then his Shapley value equals to zero.

3. *Symmetry*: If two players are symmetrical, then their Shapley values are equal.

4. *Additivity*: For any player, the Shapley value in the additive cooperative game equals to the sum of the Shapley values of the separate games.

Next, we review these properties in the context of normative systems, and provide definitions for these concepts within our value alignment context. Some of these properties can be readily integrated with our norm synthesis methodology and provide valuable insights. Others are not so interesting. All results are expressed in terms of alignment with respect to a single value, however they are all extensible to the alignment with respect to the aggregation of values.

First, we start with the *efficiency* property (1). A payoff distribution is efficient if all the reward achieved by the coalition is distributed back to its members. In our value alignment context, it is not material reward that we are allocating but recognition of a norm as crucial to effectively promote some value.

As formulated in Definition 6, the Shapley value for an individual norm is actually *not* efficient with respect to the absolute alignment of the normative system it belongs to:

$$\sum_{n_i \in N} \phi_i(v) \neq \mathsf{Algn}_{N,v} \tag{13}$$

However, the Shapley value of individual norms *does* maintain the efficiency property with respect to the *relative* alignment between the normative system being examined and the baseline:

Proposition 1. Given a normative system instance N, a baseline N_{bsl} for the family of normative systems to which N belongs and a value v (with semantics function f_v), the Shapley values of individual norms $n_i \in N$ are *efficient* with respect to the relative alignment between N and N_{bsl}:

$$\sum_{n_i \in N} \phi_i(v) = \mathsf{Algn}_{N,v} - \mathsf{Algn}_{N_{bsl},v} = \mathsf{RAlgn}_{N/N_{bsl},v} \tag{14}$$

The proof is provided in Appendix A.

Given the result in Proposition 1, the interpretation of the Shapley value in the normative systems context is slightly different from the one made in classical cooperative game theory. In that field, it is routinely assumed that the empty coalition does not achieve any utility. In our context, the "norm-less" situation (where all the normative parameters are set to their baseline values) may, in general, have non-zero alignment.

Therefore, the Shapley values of individual norms are not allocating credit for the absolute alignment that the normative system achieves. Instead, the Shapley value is allocating credit for alignment *relative to the baseline*. As explained previously, the baseline normative system should be set in a way as to halt the progress of the MAS under study (i.e. the initial state is kept unchanged after an arbitrary number of consecutive transitions). Consequently, the Shapley values are allocating the credit for the progress, from an ethical standpoint, from the initial state to the outcome achieved by the norms in place.

In case we wanted to have the Shapley values of individual norms be efficient with respect to the *absolute* alignment (i.e. have eq. (13) be fulfilled with an equality symbol), it

would be necessary to design a baseline such that $\mathsf{Algn}_{N_{bsl},v} = 0$. Computationally, it is an open question whether such a baseline exists given an arbitrary value semantics function, and whether its uniqueness is guaranteed. Additionally, from an interpretation perspective, we expect that the resulting baseline values for the normative parameters might be difficult to interpret and justify, since it would not reflect the lack of progress in the system, but rather the introduction of just enough regulation as to shift the system towards an ethically neutral outcome. Therefore, despite one might think that it is more desirable to have the Shapley values be efficient with respect to the absolute alignment, we believe that it is actually more informative to have them be efficient with respect to the relative alignment between N and N_{bsl}.

The second desirable property of the Shapley value that we examine is the role of *null norms* (2). The result from classical cooperative game theory is directly importable into our normative systems context. First, we define what a null norm is, in analogy to the concept of a null player:

Definition 7. A norm n_i is a *null norm* within normative system N with respect to value v if, for any $N' \subseteq N \setminus \{n_i\}$, it holds that $\mathsf{Algn}_{N' \cup \{n_i\},v} = \mathsf{Algn}_{N',v}$.

It is straightforward to prove from the above definition and eq. (9) that any null norm has zero Shapley value, $\phi_i(v) = 0$, meaning that it should not be given any credit for steering the system towards an ethically compliant state. However, the opposite is not necessarily true. A Shapley value of zero is only indicative of a null norm if the normative system it belongs to is *monotone*:

Definition 8. A normative system N is *monotone with respect to value* v (with semantics function f_v) if $\mathsf{Algn}_{N_2,v} \geq \mathsf{Algn}_{N_1,v}$, $\forall N_1, N_2 \subseteq N$ such that $N_1 \subset N_2$. (It is *strictly monotone* if $\mathsf{Algn}_{N_2,v} > \mathsf{Algn}_{N_1,v}$).

In a monotone normative system, switching any normative parameter from the baseline back to the original quantities necessarily increases the alignment. It can be interpreted as a normative system where all of its individual norms, to a greater or lesser extent, are contributing towards the alignment whenever they are reintroduced into the coalition.

In a monotone normative systems, the following holds:

Proposition 2. If normative system N is monotone with respect to value v and norm n_i has zero Shapley value with respect to v, $\phi_i(v) = 0$, then n_i is a null norm with respect to v.

The proof is provided in Appendix A.

Property (3) of the traditional Shapley value states that symmetrical players have identical Shapley values. In the context of normative systems, we have:

Definition 9. Two different norms $n_i, n_j \in N$ are *symmetric* with respect to value v if, for any $N' \subseteq N \setminus \{n_i, n_j\}$, it holds that $\mathsf{Algn}_{N' \cup \{n_i\},v} = \mathsf{Algn}_{N' \cup \{n_j\},v}$.

Indeed, norms that achieve the same level of promotion after they are separately introduced should be allocated the same amount of credit for the alignment realised:

Table 2: Shapley values for all the individual norms conforming the optimal normative systems with respect to the value for which they are optimised.

Value	Norm	Shapley value
Equality	n_1	0.50
	n_2	0.03
	n_3	0.07
	n_4	0.01
Fairness	n_1	0.19
	n_2	0.45
	n_3	0.46
	n_4	0.42
Aggregation	n_1	0.00
	n_2	0.27
	n_3	0.25
	n_4	0.31

Proposition 3. If $n_i, n_j \in N$ are two *symmetric norms* with respect to value v, then $\phi_i(v) = \phi_j(v)$.

Again, Proposition 3 is proven in Appendix A.

Finally, property (4) of the Shapley value that sets it apart from other payoff allocation schemes is the additivity property. However, this property is not as interesting as the other ones in our context, as its applicability is very limited. It could be applied in cases where aggregation functions over sets of values are defined as linear combinations of the individual values. To illustrate the limited scope of this property, we point to the running example in this paper, where we have defined an aggregation function for two values, however not as a linear combination. For this reason, we will not analyse this property in detail.

Running Example

Table 2 presents the Shapley values of every individual norm in the optimal normative systems in Table 1, with respect to the value for which they have been optimised. Additionally, for the three cases it has been experimentally checked that the efficiency property in eq. (14) holds. Those results are presented in Table 6 in Appendix A.

For value *equality*, the norm with the highest Shapley value is by far n_1, which is related to the collection of taxes. All other norms have Shapley values ~ 0, including the other norm of economic nature, n_2, linked to the redistribution of the common fund. These results reinforce our explanation for the optimal normative parameters with respect to equality in Table 1, where we conjectured that the wealth distribution is shrunk right after taxes are collected.

In contrast, for value *fairness*, the situation is the opposite, with norms n_2, n_3 and n_4 all having similar and large Shapley values, significantly above that of norm n_1. This would indicate that to punish evaders, it is most important to detect them (n_3), as unde-

tected evaders would automatically rise as the wealthiest members in the society. Then, the common fund needs to be very unevenly redistributed (see the optimal parameters in Table 1) towards the middle class, which is mostly composed by law-abiding citizens, as we have argued in Section 5. Note that imposing a fine on evaders is important, yet it is on approximately the same level as the redistribution of taxes. This indicates that, when it comes to punishing evaders, the norms that exclusively target them (n_3 and n_4) are just as relevant as the rule n_2 that directs their resources elsewhere, even at the expense of harming non-evader agents.

Last of all, the Shapley values for the optimal norms with respect to the aggregated values appear to be closer to those for fairness, with similar quantities for n_2, n_3 and n_4. Surprisingly, the norms related to tax recollection n_1, which stood out when it came to value *equality*, has Shapley value of zero for the aggregation.

6.2 Monotonicity in Low Dimensional Normative Systems

In order to assert whether the norms with zero Shapley values are actually null or not, we check the monotonic behaviour of every optimal normative system with respect to the value for which it has been optimised. To do so, we use a rather inelegant, brute-force approach. Due to the reduced number of parameters of our running example, however, this approach is still feasible, although not recommended in general.

The pseudo-code to check whether the optimal normative systems N^* are monotonic appears in Algorithm 1. Note that we only check pairs of subsets N_1, N_2 (being N_1 included within N_2, $N_1 \subset N_2$) for which N_2 includes only one more individual norm than N_1. It can be very easily proven that the check in Algorithm 1 is equivalent to a monotonic normative system:

Proposition 4. Given a normative system N and a value v (with semantics function f_v), if $\forall N' \subset N$, $\forall n \in N \setminus N'$ it holds that $\mathsf{Algn}_{N' \cup \{n\},v} \geq \mathsf{Algn}_{N',v}$ (i.e. Algorithm 1 returns `True`) iff N is monotone with respect to v.

Proof. The forward implication (Algorithm 1 returns `True` $\implies$ N monotone) can be proven by considering that consecutively adding new individual norms must at least maintain or improve the alignment, for any $N_2, N_1 \subseteq N$ such that $|N_2| > |N_1|$ it must hold that $\mathsf{Algn}_{N_2,v} \geq \mathsf{Algn}_{N_1,v}$.

Algorithm 1: Brute-force approach to check the monotonic behaviour of an optimal normative system.

```
foreach N_1 ⊂ N do
    compute Algn_{N_1,v}
    foreach n_i ∈ N \ N_1 do
        N_2 ← N_1 ∪ {n_i}
        compute Algn_{N_2,v}
        if Algn_{N_1,v} > Algn_{N_2,v} then
            return False
return True
```

The reverse implication (N monotone $\implies$ Algorithm 1 returns `True`) follows from the fact that, because N is monotone, $\forall N_1, N_2 \subset N$ such that $N_1 \subseteq N_2$ then $\mathsf{Algn}_{N_2,v} \geq \mathsf{Algn}_{N_1,v}$, this is in particular true for cases such that $|N_2| = |N_1| + 1$. □

One may think that monotonic normative systems are a promising subclass of normative systems for which optimisation with the target of maximum value alignment can be simplified. That is to say, instead of optimising for all normative parameters at once (like we do with the GA in our running example), one could start at the baseline normative system and optimise for one normative parameter at a time.

However, in order for this procedure to be correct, it would need to hold that the optimal normative system in a family with respect to the value of interest is indeed monotonic, even before such an optimal normative system has been computed. At this point, we are unable to provide such a guarantee, and hence recommend users of this methodology to stick with meta-heuristic methods. In fact, the following results concerning our running example prove that of the three optimal set of norms computed, only one is monotonic.

Running Example

Running Algorithm 1 over all the optimal normative systems (using the alignment function for the values for which they have been optimised) returns the following results: $N^*_{equality}$ is indeed monotone, while $N^*_{fairness}$ and $N^*_{aggregation}$ are not.

By the results reported in Table 2, we can assert that for value *equality*, all individual norms except n_1 (related to tax collection) are *null norms* (or close to null). This observation reinforces the dominant role that the *collect* rates have when it comes to promoting equality. By the optimal parameters in Table 1, we already hypothesised that the wealth distribution is shrunk right after taxes are collected. This conjecture seems to be supported by the finding that n_1 is the only non-null norm when it comes to equality (under its optimal normative system).

7. Value Compatibility

The Shapley values have allowed us to examine, given a value and its semantics, the relationships between the constituent norms in a normative system. In this section, we study an analogous issue: given a normative system, what is the relationship among several values that may (or not) be supported by it.

Schwartz's theory of basic human values establishes that actions executed in pursue of some value may have consequences that are either congruent or in conflict with other values (Schwartz, 2012). Such thinking can be extended to our policy design context. Implementing normative systems that aggressively promote some value might have collateral consequences, either positive or negative, for the promotion of other values. To quantify such relationships, we introduce the concept of *value compatibility* as an extension to the framework presented in Section 4.

Definition 10. Given a normative system N, a set of values $V = \{v_1, v_2, ..., v_k\}$ with semantics functions $f_1, f_2, ..., f_k$ are *compatible to degree d* (or d-compatible) under N if, for all values in V, it holds that $\mathsf{Algn}_{N,v} \geq d$.

It immediately follows that, if some set of values are compatible to degree d, they are also compatible to any degree $d' \leq d$. If a normative system is highly specialised towards some value at the expense of others, the compatibility will be low, possibly negative. Normative systems that reach a compromise will presumably maintain much higher compatibility between the values, even if not aligned with any of them to the maximum possible amount ~ 1.

The concept of compatibility works with alignment functions with respect to single values, $\mathsf{Algn}_{N,v}$. Nonetheless, we can easily expand it to aggregations over sets of values:

Definition 11. Given a set V of d-compatible values under normative system N, an aggregation function $F_V : [-1, 1]^n \to [-1, 1]$ is said to *preserve the compatibility* if $\mathsf{Algn}_{N,V} \geq d$.

Trivially, aggregation functions based on linear combinations of the alignment for the individual values do preserve d-compatibility by setting $d = \min_{v \in V} \mathsf{Algn}_{N,v}$.

In our framework, the concept of value compatibility makes sense and can potentially provide a lot of insight into the optimal normative systems obtained. However, an analogous concept for norms (i.e. norm compatibility, see e.g. the relationship among exclusive norms in Serramià et al., 2020) is not applicable in our approach. By construction, norms in a normative system control different aspects of a state transition. Intuitively, all the individual norms that make up a normative system are compatible with one another by construction.

Running Example

The compatibility computations for the optimal normative systems obtained in our tax policy model are presented in Table 3. It clearly manifests that the optimal normative system for equality is very neglectful of the fairness of the system, as it has negative alignment with respect to it. Unsurprisingly, its alignment with respect to the aggregation of values is also very poor. Hence, despite being very effective at reducing wealth inequality, under the optimal norms for equality the values *equality* and *fairness* are very incompatible.

Meanwhile, the norms optimised for fairness do uphold equality to a much larger extent than the other way around. Under this optimal normative system, the two values are compatible to degree ~ 0.6. This is a purely collateral effect, since the semantics function for fairness in eq. (7) does not encode any information related to the width of the wealth distribution. This result is visually confirmed by the second row in Figure 3. In its quest

Table 3: The optimal normative system with respect to value $\mathbf{v_i}$ (see Table 1) has its alignment computed with respect to values $\mathbf{v_j}$. For example, in the first row the optimal normative system with respect to equality is examined from the perspective of fairness and the aggregation of both values.

		$\mathbf{v_j}$		
		Equality	**Fairness**	**Aggregation**
$\mathbf{v_i}$	**Equality**	-	-0.28	-0.26
	Fairness	0.60	-	0.56
	Aggregation	0.71	0.88	-

to treat evader agents harshly, the optimal normative system for fairness also impoverishes many law-abiding citizens, most of which end up in the lower half of the wealth range. This happens because the retrieval of resources is, at least in part, done through a very uneven redistribution of taxes, a policy that affect all agents equally regardless of their evader status. Hence, the initial wealth distribution is narrowed as a consequence and the Gini index is decreased, although this is not the primary objective of the optimal norms for fairness.

Last row in Table 3 displays the alignment under the optimal norms for our aggregation function of the two values. It stands out as the one with the highest quantities: actually, the optimisation for the aggregation of the two values leads to a much higher compatibility degree than the optimisation for any of the two values separately, over 0.7. This is a consequence of our demanding aggregation function in eq. (8). The downside of this ambitious approach is that it does not preserve the compatibility as established in Definition 11, except under the optimal normative system for equality (for which the values are actually *incompatible*).

Also, the last row in Table 3 shows that the optimal normative system for the aggregation of values is more aligned with respect to fairness than equality. Similarly, the Shapley values in Table 2 for the aggregation of values are most similar to those of the optimal norms for fairness. Hence, we can conclude that the aggregation of values is definitely relying more on the promotion of fairness than equality.

7.1 Compatibility Maximising Normative System

Of course, we are most interested in finding the *maximum* compatibility degree for an arbitrary set of values. In fact, given a normative system N and a set of values V, their maximum compatibility degree is given by:

$$d_{max} = \min_{v \in V} \mathsf{Algn}_{N,v} \tag{15}$$

Equation (15) can, just as any other function, serve as an optimisation target. By finding the normative system that maximises the right-hand side of eq. (15), we obtain the normative system for which the set of values in question are the most compatible. Then, we can define the *compatibility maximising normative system* (CMNS) $N^{\ddagger}$ as:

$$N^{\ddagger} = \arg\max_{N \in \mathcal{N}} \min_{v \in V} \mathsf{Algn}_{N,v} \tag{16}$$

Note that eq. (16) is actually not an optimisation for an alignment function, as defined by the formal model in Section 4. One could think of taking a set of values $V = \{v_1, ..., v_n\}$ with semantics functions $f_1, ..., f_n$, setting their aggregation function as the minimum $F_V(\mathsf{s}) = \min_{v \in V} f_v(\mathsf{s})$, and then performing an optimisation of the type we have presented in Section 5. However, this would not be equivalent to searching the CMNS, since the expected value and the minimum operators are not in general commutable.

The notion of the CMNS echoes that of Pareto optimality (Lockwood, 2008). In the game theory literature, an outcome is *Pareto optimal* if there is no other outcome in which all participants are at least as well off, and at least one participant is better off. In this work, the CMNS is the normative system such that no other normative system can simultaneously

have larger alignment with respect to all values of interest. Therefore, Pareto optimality and compatibility maximisation are analogous, although not identical, concepts. While Pareto optimality fails if at least one participant in a game is better off (while all others retain the same utility), the CMNS fails if an alternative normative systems is encountered such that alignment with respect to *all* values is improved.

Running Example

To find the CMNS of our running example, we run the optimisation search using the same version of a GA as in Section 5, but taking eq. (16) as the target function. The set of values for which the alignment is computed (and then the minimum taken) is $V = \{equality, fairness\}$. To compute the expected values for the alignment, we again perform Monte-Carlo sampling with a sample of 500 paths of 10 transitions each.

Table 4: Optimisation results for the CMNS: optimal normative parameters and maximum compatibility degree, and alignments with respect to values *equality*, *fairness* and their aggregation.

$\mathbf{N}^{\ddagger}$	$collect = [4\%, 60\%, 74\%, 33\%, 58\%]$	$\mathbf{d}^{\ddagger}_{\mathbf{max}}$	0.74
	$redistribute = [3\%, 37\%, 35\%, 16\%, 9\%]$	$\mathsf{Algn}_{N^{\ddagger},\text{eq}}$	0.74
	$catch = 45\%$	$\mathsf{Algn}_{N^{\ddagger},\text{fair}}$	0.74
	$fine = 85\%$	$\mathsf{Algn}_{N^{\ddagger},\text{aggr}}$	0.56

The results of the optimisation for maximum alignment appear in Table 4. First, the optimal compatibility degree is at 0.74, slightly higher than the compatibility achieved under the optimal normative system for the aggregated values. Also, this compatibility degree is attained by promoting both values equally, although this requirement was not originally encoded in the objective function of the search. In summary, the CMNS does keep the compatibility degree between the two values (equality and fairness) to the largest degree among all optimal normative systems found in this work. However, the compatibility degree achieved by the optimal norms for the aggregation is only slightly below, where promotion of fairness is significantly boosted at the expense of a minor misalignment with respect to equality.

Second, we review the optimal parameters for the CMNS. The trends of the *collect* and *redistribute* parameters with the wealth segments are most similar to those of the optimal norms for the aggregation of values (see Table 1). Surprisingly, $N^{\ddagger}$ is the normative system that treats evaders the harshest. It has one of the highest *catch* rates (tied with that from the optimal normative system for fairness), and the largest *fine* across all normative systems analysed. However, despite being the most punitive norm set, its alignment for value *fairness* is actually lower than that of $N^{*}_{\text{aggregation}}$. This finding reinforces our observation that evaders are not actually punished by exclusively targeting them through n_3 and n_4, as we hypothesised in Section 5.

In Figure 3, bottom right row, the outcome that is reached under the CMNS is shown. The comparison with the optimal normative system for the aggregation of values shows that, despite some qualitative similarities in the normative parameters, the outcomes they lead to

are definitely different. Instead of reaching a wealth distribution halfway between N^*_{equality} and N^*_{fairness}, $N^\ddagger$ splits agents into two wealth groups by their wealth (above and below 50 units). Evaders all fall into the first group, hence ensuring fairness is at least moderately promoted. Meanwhile, because the two peaks, although separate, are still narrow, equality is also maintained.

8. Conclusions

In this work, we have proposed a solution to the problem of automated synthesis of normative systems based on value promotion. To do so, we have committed to a consequentialist position regarding the relationship between norms and values, and have delegated the responsibility to provide the meaning of values to the system designer. The methodology we have presented to tackle the task is fairly general and allows the designer to tailor it to the MAS at hand: choices need to be made regarding the norms in place, the semantics function of values and the optimisation strategy to perform the search.

The running example to illustrate our methodology shows how to apply each step of the methodology to a very simple model. In it, we have not introduced any reasoning schemes by the agents that would allow them to decide whether or not to abide by the norms, and which action to take among those that are designated as legal. We have omitted the introduction of individual reasoning schemes to keep the work focused on the automated synthesis of norms. However, we anticipate that the optimal normative system will depend on the composition of the society, namely the concrete reasoning schemes that are implemented and the proportion among them, analogously to the results obtained by Fagundes et al. (2016).

The results obtained in our running example for our alignment-maximising search have been very satisfactory, although the model has a reduced number of normative parameters and hence the search space is relatively small. Although it is possible to hypothesise about the role of the obtained normative parameters (i.e. with the aid of visual representations), the interpretability of the resulting normative parameters quantities is very much aided by the Shapley values of individual norms. The role that this indicator plays in our normative systems context is similar to the explainable AI literature, particularly in the estimation of feature importance in supervised machine learning models (Štrumbelj & Kononenko, 2013; Lundberg & Lee, 2017). Additionally, some of the desirable properties that give the Shapley value a privileged position in cooperative game theory have been proven to apply to the area of this work.

The last contribution of this paper is a numerical quantification in the compatibility between values. We have defined this concept formally and then performed a search for the normative system that maximises it in our running example. However, one could argue that the results for the CMNS are not much of an improvement over those obtained with a demanding aggregation function, and a "regular" optimisation using the derived alignment function as the search target.

Overall, this paper contains a substantial contribution to the field of automated synthesis of value-aligned normative systems, a field where there is not a load of work to build upon. We foresee that extensions of this work will study the scalability of our methodology to more complex models dependent upon more normative parameters, and the application

to real-life problems of policy design and analysis. Interesting future work should also attempt to integrate the synthesis methodology presented here with a framework that, prior to running the optimising search, automatically de-abstracts the value of interest into a semantics function, taking into account the domain at hand. Such an integration would allow practitioners to work with values directly as abstract entities.

Code Availability

All the code to go along with this work has been integrally developed in Python 3. It is available under an MIT license at `https://github.com/nmontesg/aamas21` and as an online appendix.

Acknowledgments

This work has been supported by the AppPhil project (RecerCaixa 2017), the CIMBVAL project (funded by the Spanish government, project #TIN2017-89758-R), the EU WeNet project (H2020 FET Proactive project #823783) and the EU TAILOR project (H2020 #952215).

Appendix A. Supplementary Information

Table 5: Hyperparameters of the Genetic Algorithm to find the optimally value-aligned normative systems. The first six refer to hyperparameters of the GA itself, and are covered in Section 5. The latter two refer to the Monte Carlo sampling: number of state transitions per path and total amount of paths sampled to compute the alignment.

Hyperparameter	**Value**
Population size	100
p (intermediate recombination)	0.25
k (elitism)	5
Maximum total iterations	500
Maximum partial iterations	50
Fitness threshold	0.9
Path length	10
Path sample size	500

Proofs of Propositions 1 to 3

In order to prove Propositions 1 to 3, we will use an equivalent definition of the Shapley value from the one in eq. (9). For a complete formulation of the Shapley value solution

Table 6: Optimal alignment found for every value, alignment of the baseline normative system and sum over the Shapley values with respect to that value. It can be checked that the sum over the Shapley values is efficient with respect to the relative alignment between the optimal normative system and the baseline, see eq. (14).

Value	$\mathsf{Algn}^{*}_{\mathbf{N},\mathbf{v}}$	$\mathsf{Algn}_{\mathbf{N_{bsl}},\mathbf{v}}$	$\sum_{\mathbf{n_i}\in\mathbf{N}^{*}} \phi_{\mathbf{i}}(\mathbf{v})$
Equality	0.95	0.34	0.61
Fairness	0.93	-0.59	1.52
Aggregation	0.66	-0.22	0.88

concept from the perspective of classical cooperative game theory, the reader is directed to Chalkiadakis et al. (2011, Ch. 2).

Given a normative system N, we denote by Π_N the set of all permutations over N. There are $|N|!$ permutations in total. For a permutation $\pi \in \Pi_N$, we denote by $S_\pi(n_i)$ the set of predecessors of norm n_i in permutation π. For example, in the social model considered throughout the paper, we have $N = \{n_1, n_2, n_3, n_4\}$. A permutation is $\pi = (n_1, n_3, n_4, n_2)$. Then, $S_\pi(n_1) = \{\}$, $S_\pi(n_2) = \{n_1, n_3, n_4\}$, $S_\pi(n_3) = \{n_1\}$ and $S_\pi(n_4) = \{n_1, n_3\}$.

The marginal contribution of norm n_i to the alignment for value v in permutation π is defined as:

$$\Delta^{v}_{\pi}(n_i) = \mathsf{Algn}_{S_\pi(n_i)\cup\{n_i\},v} - \mathsf{Algn}_{S_\pi(n_i),v} = \mathsf{RAlgn}_{S_\pi(n_i)\cup\{n_i\}/S_\pi(n_i),v} \tag{17}$$

Δ^{v}_{π} measures the improvement in the alignment (with respect to value v) when norm n_i joins its predecessors in permutation π.

Then, the Shapley value for norm n_i can be defined as the average over all permutation of its marginal contribution to the alignment:

$$\phi_i(v) = \frac{1}{|N|!}\sum_{\pi\in\Pi_N} \Delta^{v}_{\pi}(n_i) \tag{18}$$

Proof of Proposition 1. π_j denotes the norm in the j-th position in permutation π.

$$\begin{aligned}
\sum_{n_i \in N} \phi_i(v) &= \sum_{n_i \in N} \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \Delta^v_\pi(n_i) = \\
&= \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \sum_{n_i \in N} \Delta^v_\pi(n_i) = \\
&= \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \mathsf{Algn}_{\{\pi_1\},v} - \mathsf{Algn}_{\{\},v} + \mathsf{Algn}_{\{\pi_1,\pi_2\},v} - \mathsf{Algn}_{\{\pi_1\},v} + \\
&\qquad + ... + \mathsf{Algn}_{\{\pi_1,...,\pi_n\},v} - \mathsf{Algn}_{\{\pi_1,...,\pi_{n-1}\},v} = \\
&= \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \mathsf{Algn}_{\{\pi_1,...,\pi_n\},v} - \mathsf{Algn}_{\{\},v} = \\
&= \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \mathsf{Algn}_{N,v} - \mathsf{Algn}_{N_{bsl},v} = \\
&= \frac{1}{|N|!} |N|! \left(\mathsf{Algn}_{N,v} - \mathsf{Algn}_{N_{bsl},v}\right) = \mathsf{RAlgn}_{N/N_{bsl},v}
\end{aligned}$$

□

Proof of Proposition 2. Because N is monotone (see Definition 8), it must be that $\Delta^v_\pi(n_i) \geq 0,\ \forall \pi \in \Pi_n, \forall n_i \in N$.

Then, if a norm has $\phi_i(v) = \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \Delta^v_\pi(n_i) = 0$, it must be that every term equals zero, $\Delta^v_\pi(n_i) = 0,\ \forall \pi \in \Pi_N$. This is equivalent to having $\mathsf{Algn}_{N' \cup \{n_i\},v} = \mathsf{Algn}_{N',v},\ \forall N' \subseteq N \backslash \{n_i\}$, as every $N' \subseteq N \backslash \{n_i\}$ can be identified with $S_\pi(n_i)$ for some $\pi \in \Pi_N$. This derivation corresponds to n_i being a null norm (see Definition 7). □

Proof of Proposition 3. Suppose $n_i, n_j \in N$ are symmetric norms (see Definition 9). Given a permutation π, we denote by π' the permutation that is obtained from π by swapping n_i and n_j. First, we prove that $\Delta^v_\pi(n_i) = \Delta^v_{\pi'}(n_j)$.

Suppose n_i precedes n_j in π. Then, $S_\pi(n_i) = S_{\pi'}(n_j) = N'$:

$$\begin{aligned}
\Delta^v_\pi(n_i) &= \mathsf{Algn}_{N' \cup \{n_i\},v} - \mathsf{Algn}_{N',v} \\
\Delta^v_{\pi'}(n_j) &= \mathsf{Algn}_{N' \cup \{n_j\},v} - \mathsf{Algn}_{N',v}
\end{aligned}$$

Because n_i and n_j are symmetric, it holds that $\mathsf{Algn}_{N' \cup \{n_i\},v} = \mathsf{Algn}_{N' \cup \{n_j\},v}$. Consequently, $\Delta^v_\pi(n_i) = \Delta^v_{\pi'}(n_j)$.

Now suppose that n_i goes after n_j in π. Now, $N' = S_\pi(n_i) \backslash \{n_j\}$:

$$\begin{aligned}
\Delta^v_\pi(n_i) &= \mathsf{Algn}_{N' \cup \{n_j\} \cup \{n_i\},v} - \mathsf{Algn}_{N' \cup \{n_j\},v} \\
\Delta^v_{\pi'}(n_j) &= \mathsf{Algn}_{N' \cup \{n_i\} \cup \{n_j\},v} - \mathsf{Algn}_{N' \cup \{n_i\},v}
\end{aligned}$$

Again, because n_i and n_j are symmetric, the second terms of the right-hand sides are equal, and $\Delta^v_\pi(n_i) = \Delta^v_{\pi'}(n_j)$.

Then, because the map between π and its swapped permutation π' is one-to-one, we have:

$$\phi_i(v) = \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \Delta^v_\pi(n_i) = \frac{1}{|N|!} \sum_{\pi \in \Pi_N} \Delta^v_{\pi'}(n_v) = \phi_j(v)$$

□

References

Ajmeri, N., Guo, H., Murukannaiah, P. K., & Singh, M. P. (2020). Elessar: Ethics in norm-aware agents. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, p. 16–24, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Atkinson, K., & Bench-Capon, T. (2016). States, goals and values: Revisiting practical reasoning. *Argument & Computation*, *7*(2-3), 135–154.

Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, p. 38–46, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bench-Capon, T., & Modgil, S. (2017). Norms and value based reasoning: justifying compliance and violation. *Artificial Intelligence and Law*, *25*(1), 29–64.

Chalkiadakis, G., Elkind, E., & Wooldridge, M. (2011). Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *5*(6), 1–168.

Conte, R., & Castelfranchi, C. (1999). From conventions to prescription. towards an integrated view of norms. *Artificial Intelligence and Law*, *7*(4), 323–340.

Fagundes, M. S., Ossowski, S., Cerquides, J., & Noriega, P. (2016). Design and evaluation of norm-aware agents based on normative markov decision processes. *International Journal of Approximate Reasoning*, *78*, 33–61.

Feather, N. T. (1995). Values, valences, and choice: The influences of values on the perceived attractiveness and choice of alternatives.. *Journal of Personality and Social Psychology*, *68*(6), 1135–1151.

Gini, C. (1912). *Variabilità e Mutuabilità. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche*. Facoltà di Giurisprudenza della R. Università di Cagliari.

Gorrieri, R. (2017). Labeled transition systems. In *Monographs in Theoretical Computer Science. An EATCS Series*, pp. 15–34. Springer International Publishing.

Grossi, D., Tummolini, L., & Turrini, P. (2012). *Norms in Game Theory*, chap. Chapter 12, pp. 191–197. No. 8 in Law, Governance and Technology. Springer, Dordrecht.

Lemieux, C. (2009). *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer New York.

Lockwood, B. (2008). Pareto efficiency. In *The New Palgrave Dictionary of Economics*, pp. 1–5. Palgrave Macmillan UK.

Luke, S. (2013). *Essentials of Metaheuristics* (second edition). Lulu.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Macintyre, A. (1998). *A Short History of Ethics: A History of Moral Philosophy from the Homeric Age to the Twentieth Century*. University of Notre Dame Press.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, *9*.

Montes, N., & Sierra, C. (2021). Value-guided synthesis of parametric normative systems. In *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '21, p. 907–915, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems. (Best paper award finalist).

Morales, J., López-Sanchez, M., Rodríguez-Aguilar, J. A., Wooldridge, M., & Vasconcelos, W. (2013). Automated synthesis of normative systems. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, p. 483–490, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Morales, J., Wooldridge, M., Rodríguez-Aguilar, J. A., & López-Sánchez, M. (2018). Off-line synthesis of evolutionarily stable normative systems. *Autonomous Agents and Multi-Agent Systems*, *32*(5), 635–671.

Morris-Martin, A., Vos, M. D., & Padget, J. (2019). Norm emergence in multiagent systems: a viewpoint paper. *Autonomous Agents and Multi-Agent Systems*, *33*(6), 706–749.

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49.

Onn, S., & Tennenholtz, M. (1997). Determination of social laws for multi-agent mobilization. *Artificial Intelligence*, *95*(1), 155–167.

Peters, H. (2008). The shapley value. In *Game Theory*, pp. 241–258. Springer Berlin Heidelberg.

Rokeach, M. (1972). *The nature of human values*. Free Press.

Ruder, S. (2016). An overview of gradient descent optimization algorithms..

Sandholm, W. H. (2009). Evolutionary game theory. In *Encyclopedia of Complexity and Systems Science*, pp. 3176–3205. Springer New York.

Savarimuthu, B. T. R., & Cranefield, S. (2011). Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems*, *7*(1), 21–54.

Schwartz, S. H. (1992). Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries. In *Advances in Experimental Social Psychology*, pp. 1–65. Elsevier.

Schwartz, S. H. (2012). An overview of the Schwartz theory of basic values. *Online Readings in Psychology and Culture*, *2*(1).

Serramià, M., López-Sánchez, M., & Rodríguez-Aguilar, J. A. (2020). A qualitative approach to composing value-aligned norm systems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, p. 1233–1241, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Serramià, M., López-Sánchez, M., Rodríguez-Aguilar, J. A., Morales, J., Wooldridge, M., & Ansotegui, C. (2018). Exploiting moral values to choose the right norms. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. ACM.

Shapley, L. S. (1951). *Notes on the N-Person Game - II: The Value of an N-Person Game*. RAND Corporation, Santa Monica, CA.

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, *73*(1-2), 231–252.

Sierra, C., Osman, N., Noriega, P., Sabater-Mir, J., & Perelló-Moragues, A. (2019). Value alignment: A formal approach. In *Responsible Artificial Intelligence Agents Workshop (RAIA) in AAMAS 2019*.

Spates, J. L. (1983). The sociology of values. *Annual Review of Sociology*, *9*(1), 27–49.

Štrumbelj, E., & Kononenko, I. (2013). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, *41*(3), 647–665.

Szabo, J., Such, J. M., & Criado, N. (2020). Understanding the role of values and norms in practical reasoning. In Bassiliades, N., Chalkiadakis, G., & de Jonge, D. (Eds.), *Multi-Agent Systems and Agreement Technologies*, pp. 431–439, Cham. Springer International Publishing.

Teze, J. C. L., Perelló-Moragues, A., Godo, L., & Noriega, P. (2019). Practical reasoning using values: an argumentative approach based on a hierarchy of values. *Annals of Mathematics and Artificial Intelligence*, *87*(3), 293–319.

The World Bank, Development Research Group (2019). Gini index (world bank estimate, 1967-2019).. Accessed 7th June 2021, `http://data.worldbank.org/indicator/SI.POV.GINI`.

van de Poel, I. (2020). Embedding values in artificial intelligence (AI) systems. *Minds and Machines*, *30*(3), 385–409.

van der Weide, T. L., Dignum, F., Meyer, J. J. C., Prakken, H., & Vreeswijk, G. A. W. (2010). Practical reasoning using values. In *Lecture Notes in Computer Science*, pp. 79–93. Springer Berlin Heidelberg.

Visser, S., Thangarajah, J., Harland, J., & Dignum, F. (2015). Preference-based reasoning in BDI agent systems. *Autonomous Agents and Multi-Agent Systems*, *30*(2), 291–330.

Contribution 2

A Computational Model of Ostrom's Institutional Analysis and Development Framework

Artificial Intelligence

Full citation:

Montes, N., Osman, N., & Sierra, C. (2022b). A computational model of Ostrom's Institutional Analysis and Development framework. *Artificial Intelligence*, *311*, 103756. https://doi.org/10.1016/j.artint.2022.103756

Artificial Intelligence 311 (2022) 103756

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A computational model of Ostrom's Institutional Analysis and Development framework

Nieves Montes *, Nardine Osman, Carles Sierra

Artificial Intelligence Research Institute (IIIA-CSIC), UAB Campus, Carrer de Can Planas, Zona 2, 08193 Bellaterra (Barcelona), Spain

ARTICLE INFO

Article history:
Received 29 November 2021
Received in revised form 28 June 2022
Accepted 1 July 2022
Available online 8 July 2022

Keywords:
Institutional Analysis and Development framework
Rules
Normative multiagent systems
Game theory
Logic programming

ABSTRACT

The Institutional Analysis and Development (IAD) framework developed by Elinor Ostrom and colleagues provides great conceptual clarity on the immensely varied topic of social interactions. In this work, we propose a computational model to examine the impact that any of the variables outlined in the IAD framework has on the resulting social interactions. Of particular interest are the rules adopted by a community of agents, as they are the variables most susceptible to change in the short term. To provide systematic descriptions of social interactions, we define the Action Situation Language (ASL) and provide a game engine capable of automatically generating formal game-theoretical models out of ASL descriptions. Then, by incorporating any agent decision-making models, the connection from a rule configuration description to the outcomes encouraged by it is complete. Overall, our model enables any community of agents to perform *what-if* analysis, where they can foresee and examine the impact that a set of regulations will have on the social interaction they are engaging in. Hence, they can decide whether their implementation is desirable.

1. Introduction

The Institutional Analysis and Development (IAD) framework is a conceptual toolbox put forward by Elinor Ostrom and colleagues in an effort to identify and delineate the universal common variables that underlie the immense variety of human interactions [1]. The framework identifies *rules* as one of the core constructs that determine the structure of interactions, and acknowledges their potential to steer a community towards more beneficial and socially desirable outcomes.

This work presents the first attempt to turn the IAD framework into a computational model that allows communities of agents to perform *what-if* analysis on a given rule configuration. To do so, we define the Action Situation Language (ASL) whose syntax is highly tailored to the components of the IAD framework and that is used to write formal descriptions of social interactions. The ASL is complemented by a game engine that generates the semantics of social interactions as extensive-form games (EFGs). These EFGs can then be analyzed with the standard game-theoretical tools to predict which outcomes are being most incentivized, and evaluated according to the overall social benefit they bring about. All the code to go along with this work is open-sourced under an MIT license on the AI4EU platform and GitHub. Beyond the implementation of the fundamental algorithms, we include support for customized visualization of the generated game trees.

* Corresponding author.
E-mail address: nmontes@iiia.csic.es (N. Montes).

https://doi.org/10.1016/j.artint.2022.103756

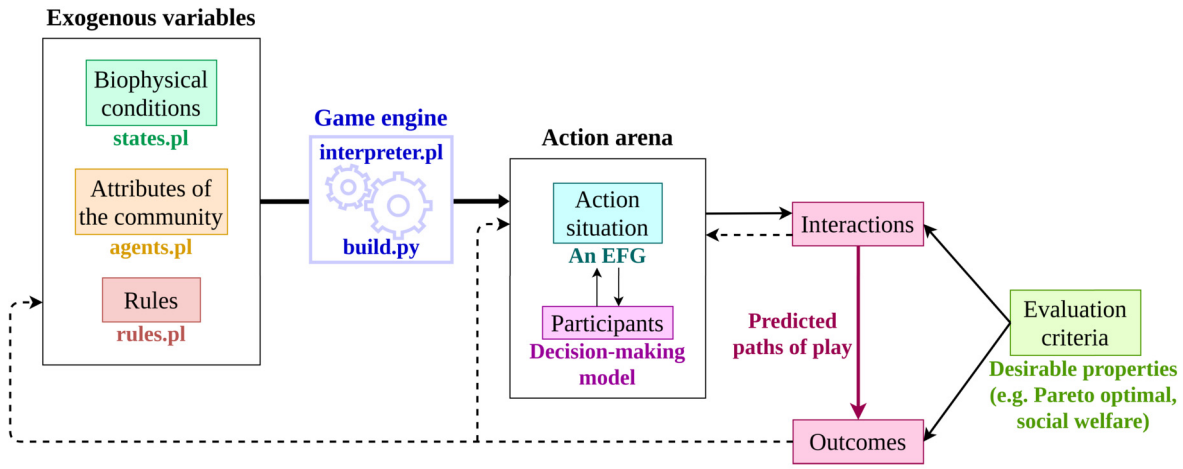


Fig. 1. Outline of the Institutional Analysis and Development framework, adapted from [2, p. 15]. Colored text outside boxes indicates either the scripts that contain information on the boxed component, or the game-theoretical concepts that represent it.

This paper is organized as follows. We start by presenting the necessary background on the IAD framework, outline our contributions and review some related work in the rest of this Introduction. Then, we present the syntax of the Action Situation Language in Section 2. Next, in Section 3, we provide a detailed explanation of the process of rule interpretation – a crucial step to turn action situation descriptions into games – and go through the game semantics generation process. The last technical part, Section 4, reviews some issues related to implementation and the integration of the resulting game representations with game-theoretical tools. Finally, we close with some illustrative examples in Section 6 and make our concluding remarks in Section 7.

1.1. The Institutional Analysis and Development framework

Within the field of policy analysis, the Institutional Analysis and Development (IAD) framework, put forward by Ostrom and colleagues [2], represents a comprehensive theoretical effort to identify and delineate the universal building blocks that make up any social interaction. Its outline is presented in Fig. 1. In the center part, any social interaction is referred to as an *action arena*. In it, a set of *participants* (the agents) find themselves in an *action situation*, which is the social space they may enter, take actions in and jointly bring about outcomes.

According to the IAD framework, action arenas are affected by three sets of exogenous variables that jointly combine to structure it (Fig. 1 left). These are the *biophysical conditions*, the *attributes of the community* and the *rules* of the interaction. The first two are fairly straightforward to define. The biophysical conditions refer to the relevant characteristics of the environment where the interaction takes place, such as land topology and location of resources. The attributes of the community encompass variables intrinsically linked to the participants, such as age, gender, ethnicity and/or belonging to one or several subgroups. Last of all, the meaning of the term *rules* is wide enough to require a detailed clarification that we provide below.

The IAD framework acknowledges the four common uses of the term *rules* in everyday language [3, Ch. 6], according to their scope: instructions, precepts, regulations and principles. First, instructions are understood as a set of steps to effectively achieve some desirable outcome in a given context. Good contemporary examples are Ikea assembly guides. In second place, precepts are somewhat similar to instructions, in the sense that they also directly concern the actions to be taken by an agent. However, their scope is more general. Instead of specifying the particular actions that an agent should perform in a specific situation or context, precepts provide widely applicable principles to help guide decision-making in a range of situations. Good examples are the five precepts from the Buddhist faith [4], which should be regarded by any person adhering to Buddhism regardless of the situation they are confronted with.

In the third place, regulations are, possibly, the most intuitive meaning of rules. They refer to statutes and ordinances that constrain or provide alternative avenues for a course of action. Typically, regulative rules are understood as being passed down from a central authority responsible for their crafting and enforcement. However, small communities can also self-impose regulations on themselves in order to ensure sustainability, fairness, and other desirable goals. For example, there are many cases of small communities of fishers, loggers and crop farmers who craft their own regulations regarding how much fish, wood or water is each member entitled to [5].

Finally, the last of the meanings that rules take are as physical principles. These refer to the laws of nature that inevitably play a part in determining what actions and/or outcomes are physically possible and the effects they have on the environment. If you drop an object, it will fall downwards. Additionally, if it is made of a fragile material such as glass or ceramic, it will most certainly break.

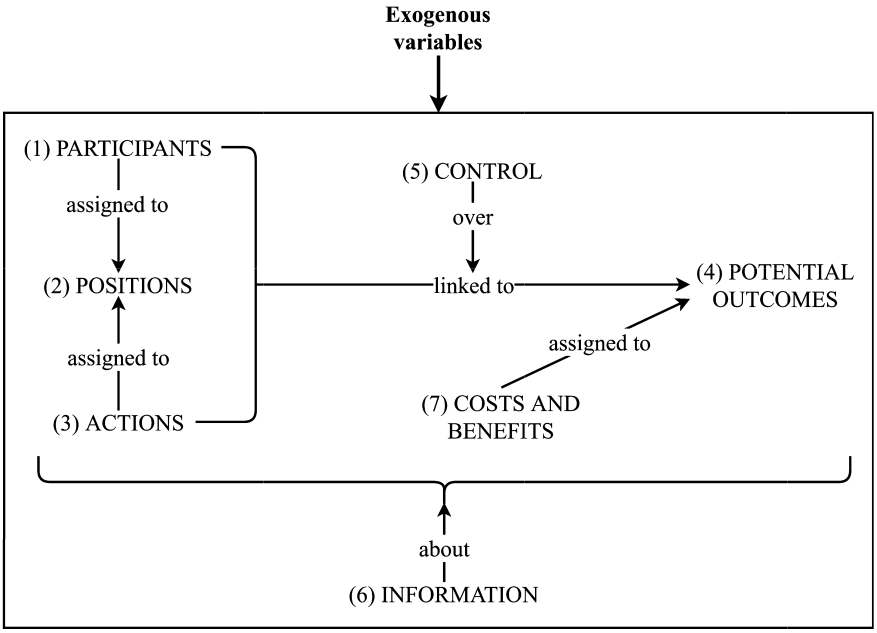


Fig. 2. Internal structure of an action situation, adapted from [2, p. 33].

There is a major difference between the first two and the last two meanings. While instructions and precepts indicate to an agent (either directly as instructions, or indirectly as precepts) what actions to perform provided the situation at hand, regulations and principles condition the structure of the situation itself. Together, regulations and physical principles jointly determine what actions are possible and/or allowed, what their effects are, potential sanctions if a prohibited action is performed (or failure to perform an obliged action) and, consequently, which outcomes may be attained. Once all of that information is gathered, instructions and precepts are invoked on that specific situation to output the particular course of action to take. Hence, instructions and precepts directly target the *actions* to take given a social situation, while regulations and principles determine the social situation itself.

In this work, we take the view that the function of rules is to mold the structure of the situation agents find themselves in, i.e. to modify the incentives and opportunities they face. Hence, the term *rule* will be used to encapsulate both regulations and natural principles. There is a fundamental difference, however, between the two. While regulations are human-made, and hence subject to revision and change, natural principles are not and therefore they are essentially untouchable. We address this distinction in our logical language by distinguishing between *default* rules and additional regulations through a priority relationship between rule statements. Also, we make the distinction between rules that reflect natural principles and the biophysical conditions introduced early on. Physical laws control the *dynamics* of the environment (drop an object and it will land on the ground) while biophysical conditions refer to static elements (like land topology).

In the computational model of the IAD framework we present, we leave out instructions and precepts, since our Action Situation Language does not include an avenue to model them in a systematic manner. We make this choice because we are interested in the constructs that shape the social interactions (i.e. regulations and physical laws). Therefore, this work is not concerned with the individual decision-making (which takes into account instructions and precepts) that agents perform once they are faced some situation. For decision-making, we rely on game-theoretic schemes, which are directly applicable on the extensive-form game representations of social interactions that are automatically built by our tool.

In addition to identifying the variables that condition an action arena, the IAD framework also determines the components that together make up any action situation. There are in total seven variables at play (see Fig. 2): (1) the participants who are allowed to enter; (2) the positions or roles that they take on; (3) the actions assigned to those roles; (4) the potential outcomes that may be reached; (5) the linkage between actions (or sequences of actions) to outcomes and the control that agents have over it; (6) the information available to participants about all other variables (including what information is available to *others*); and (7) the material reward and costs assigned to outcomes and/or actions.

Once participants populate an action situation, it becomes fully instantiated. By introducing some decision-making model for every agent (such as traditional rationality notions like the Nash equilibrium), a prediction of how the interaction is expected to play out and the eventual outcomes (the “Interactions” and “Outcomes” boxes in Fig. 1) that are likely to be reached can be constructed. Finally, these outcomes can be evaluated in terms of some desirable properties (“Evaluation criteria” box in Fig. 1), such as optimality, efficiency, or various metrics of social welfare [6].

If the evaluation is not satisfactory, changes to the exogenous variables should be made in pursue of more desirable outcomes (according to the evaluation criteria of choice). Of the three sets of exogenous variables, biophysical conditions and the attributes of the community are fixed in the short term. In contrast, rules are relatively malleable. In particular, human-crafted regulations are very much susceptible to review and modification. However, the "default" rules (those that reflect natural principles) cannot be changed.

Despite being introduced several decades ago [7], the IAD framework is currently being used in policy analysis studies. Most recent examples include scenarios of pollution and waste management in widely diverse areas of the world [8,9] as well as conservation policy [10]. Research efforts on the theoretical front are also on-going in order to better integrate the IAD framework with existing formal legal systems [11].

1.2. Contributions

The main contribution of this work is a computational model of Ostrom's IAD framework. This model enables communities of agents to formally perform *what-if* analysis of potential new regulatory rules they may be considering to adopt. We provide new tools and integrate them with existing concepts, to compose the complete connection from rule specification to evaluation in terms of the joint outcomes that are encouraged by the regulations in place.

In order to write rule configurations in a systematic manner, we present our novel Action Situation Language (ASL). This is a machine-readable logical language (implemented in Prolog) whose syntax is highly tailored to the exogenous variables outlined in the IAD framework (see Fig. 1). ASL is complemented by a game engine that takes as input a valid action situation description and automatically generates its semantics as an extensive-form game (EFG). EFGs are abstract and very general models, prevalent in the microeconomics field [12, Ch.9], that can be instantiated to represent a wide variety of social interactions among an arbitrary number of agents. Although environmental and community attributes also play a role in generating the EFG semantics, we are particularly interested in the impact that rules have on the resulting formal model. In fact, an essential component of the game engine is a rule interpreter, whose function is to query the rule base, process their implications and solve conflicts between contradicting rules.

In fact, the two main innovations we present (ASL plus its game engine) bridge the gap between the normative multi-agent systems (norMAS) and game theory fields. In norMAS, a great deal of work has been devoted to the study of *norms*, *rules* and other constraining mechanisms to achieve coordination and socially beneficial behavior among autonomous agents [13–15]. In parallel, game theory has provided a powerful toolbox to model multiagent interactions of competitive, cooperative and hybrid nature. Very well established game theoretical solution concepts are prevalent across the MAS literature (e.g. [16,17]). However, in game theory, the rules that configure the structure of the interaction become irrelevant once the formal model has been built, and they are often expressed in non-systematic, plain natural language. With ASL, such rules can be expressed in a systematic manner and their semantically equivalent formal game is automatically generated by the game engine.

The choice of EFGs as the semantics for an ASL description is motivated by the availability of many game-theoretical solution concepts, such as traditional rationality notions (e.g. Nash or correlated equilibrium, subgame perfect equilibria, etc.) and social properties of outcomes (e.g. Pareto efficiency, social welfare), that can be readily applied to any model built by the game engine. Due to the prevalence of these well-established concepts in the game theory literature, we do not see the need to provide new solution concepts of our own. Introducing such models of agent decision-making (either "rational" in the traditional sense or not) amounts to modeling the *participant* component of an action arena (see Fig. 1). This step then paves the way to compute the most likely outcomes and evaluating them according to their optimality, efficiency, or social welfare. At this point, the process that takes in a rule configuration and evaluates its impact is complete, and the community of agents involved is informed about the repercussion that such regulations would have on them, were they to be adopted.

1.3. Related work

Originally, the IAD framework was complemented by the Institutional Grammar (IG) [18]. The IG parses institutional statements (which include strategies, norms in the sense of conventions, and regulative rules) into five fields: the attributes (A) of the participants to whom the statement applies; the deontic (D) modality (permitted, forbidden or obliged); the aim (I) of the statement, meaning the action or outcome to whom the deontic applies; the condition (C) under which the statement applies; and the or-else (O) field which states the consequences of non-compliance. Put together, these fields constitute the ADICO syntax. The three types of institutional statements are distinguished by the fields that are necessary to describe them: AIC for strategies, ADIC for conventions and ADICO for rules. Lately, the IG has spurred renewed interest, with extensions to the original proposal including the nesting of statements [19] and the distinction between different levels of granularity in the parsing [20].

Although the early version of the IG did contain examples of formal games built from institutional statements [see 2, Ch. 5-6], no attention has been paid at automating this process, as the ADICO syntax is not designed as a machine-readable language. However, one of its most interesting features, which we will import into ASL to some extent, is the classification of rules based on the component of the action situation that they target, according to the aim (I) field.

Although not being machine-readable, some works have attempted to make the ADICO syntax operational in agent-based models [21,22]. There, the ADICO syntax is used to represent agents' strategies and shared conventions. However, these works are limited in scope, since they use very restricted forms of institutional statements (AIC and ADIC statements obtained from combinations of a pre-defined set of possibilities for every field) and only target the modeling of common-pool resource situations, i.e. natural resources that are jointly exploited by a community of farmers, fishers, loggers, etc., and whose easy access makes it very difficult to forcefully exclude anyone from accessing it [5]. Although the IAD framework indeed accounts for the analysis of this type of scenarios, it is intended to identify and analyze the components of a wide variety of social interactions. Due to the limitations in these previous works, we choose not to build on top of them and move away from the ADICO syntax by defining our own machine-readable language which is able to model a large range of action situations by leveraging the generality of game theoretical models.

Another work with the same objective as ours (turning the IAD framework into a general-purpose operational computational tool) has been developed by [23] as the Modeling Agent systems based on Institutional Analysis (MAIA) framework. Itsworkflow is somewhat similar to ours: input an action situation description into a web application (we feed an ASL description into the game engine) that automatically generates an executable script for an agent-based simulation (our engine generates a game theoretical model). Beyond technical differences ([23] employs Java plus HTML, while we use a combination of Prolog and Python), our contributions diverge in the encoding of institutional statements, as [23] stick to the ADICO syntax, while we propose a new *if-then-where* syntax.

However, the most significant difference between [23] and the present work lies in the approach to the "Participants" component in Fig. 1. Our computational model of the IAD framework is agnostic with respect to the decision-making model participants follow once they find themselves within an action arena. Hence, modeling participants and generating an action situation representation are independent tasks and, in principle, the same decision-making model can be applied across a wide diversity of situations, e.g. Nash equilibria computation can be applied to any extensive-form game. In contrast, the MAIA framework requires a criterion for decision-making to be explicitly provided as an input to their simulation generator, and therefore it needs for the participants to be modeled beforehand and crafted for every particular simulation. Such criterion is tailored to the context at hand, and is not, in principle, exportable to other situations.

On another front, the field of General Game Playing (GGP) within the AI community has come up through the years with machine-processable languages for the specification of general games. Most prominently, the Game Description Language (GDL) [24] is a high-level language for the specification of games with a finite number of players and legal moves. GDL provides compact descriptions of deterministic classical games (such as chess and checkers) and also admits a form of restricted imperfect information in the form of simultaneous moves, a feature that we incorporate into our language.

Since its creation, some extensions have been added to GDL in order to improve its expressive power. Most notably, GDL-II [25] incorporates the possibility of imperfect information and random moves by nature, although limiting those to a uniform probability distribution. Later, yet another addition resulted in the introduction of GDL-III, where epistemic games in which the rules depend on the knowledge of the agents can be represented by introducing player introspection [26]. Beyond game playing, GDL (in its original version) has been used for more socially relevant applications, such as mediated dispute resolution [27] and automated negotiation [28].

For comparison purposes, ASL and GSL descriptions of the benchmark Iterated Prisoner's Dilemma game are displayed in Listings 1 to 4. Although both GDL and our ASL are logical languages for game specification, some of the features of ASL make it much better suited than GDL for modeling socioeconomic interactions. First, when using the term "rules of the game" in relation to GDL, it is referring to the complete game description (i.e. the logical program). In contrast, by "rules" in this work we refer to one of the components describing an action situation, i.e. to the exogenous variable "rules" in Fig. 1, separate from biophysical conditions and attributes of the community.

Second, although the authors in [25] include a qualitative description of a procedure to turn GDL descriptions into extensive-form games (and vice-versa), this is not a central contribution of their work. Instead, they provide a logic for reasoning about GDL game descriptions based on a variant of the Situation Calculus [29]. Differently, we put a lot of focus on the interpretation and the translation of ASL descriptions into EFGs. These are the most prevalent models to represent social interactions in microeconomics and policy analysis (see the examples in Section 6). They are abstract and general enough to capture a wide variety of interactions, while at the same time being amenable for analysis by implementing notions of rationality that are prevalent across the social sciences.

Finally, the feature that sets ASL apart from GDL is the fact that ASL descriptions are meant to be extensible. That is, an ASL description is intended to be expanded with additional higher-priority rules. In other words, the same ASL description can give rise to two different multiagent interactions, depending on whether new rules in addition to the default ones are included or not. Differently, GDL game descriptions are static and not meant for modification. This is reflected in the fact that GDL does not incorporate any mechanism to solve conflicts between rules, while ASL does.

A game description system previous to GDL was the Game Language (Gala) [30]. The focus of the Gala system was on the efficient computation of solutions of large imperfect information game trees, and it suffers from some of the same drawbacks that make it unsuitable for the representation of socioeconomic interactions. In particular, Gala does not consider the rules of the game separate from other relevant exogenous variables either, nor are its descriptions meant to be extended.

2. ASL syntax

Our intention is to define the syntax of ASL as fully machine-readable, yet also relatively syntactically friendly to make it accessible to social science scholars. In order to completely describe an action situation, our language must specify the three sets of exogenous variables that affect it (see Fig. 1):

- **Attributes of the community**: the agents susceptible of taking part in the interaction, plus any relevant characteristics: age, gender, ethnicity, etc.
- **Biophysical and environmental conditions**: land topology, location of resources, etc.
- The **rules** structuring the situation, in particular the following four *types*, according to which aspect of the action situation they address:
  - **Boundary rules**: which agents are allowed to enter the action situation. For example, in many countries it is required to be over 18 years old to participate in an electoral process.
  - **Position rules**: what roles do the participants take on. For example, candidate, voter, etc.
  - **Choice rules**: what actions are available to the various roles under the current conditions. For example, an agent with the role *voter* can take the action to vote for one (or none) of the candidates.
  - **Control rules**: what are the effects of those actions. In a majority rule electoral process, the candidate with the most votes gets appointed to the position in contention.

Additionally, the following information is also necessary:

- The initial conditions when the interaction starts.
- The termination conditions under which the interaction halts.
- Which facts describing the state of the system can be simultaneously true (for example, an agent cannot be at two different locations at the same time).

As explained in Section 1.1, we consider rules in the sense of regulations and physical principles. Concerning the former, rule statements in ASL completely encapsulate human-made regulations. In fact, boundary, position and choice rules (which deal with providing agents with access to the social interaction, a role in it and actions to affect it, respectively) are not in any way related to the natural principles governing the environment. Concerning physical principles, these are captured both by control rules that dictate the dynamics of the system (see Section 3) and incompatibilities between facts, which are expressed through a dedicated predicate symbol.

ASL descriptions follow the standard syntax of logic programming and are thus composed of constant symbols, function symbols, predicate symbols and variables. Expressions are classified as one of the following:

- A *term* is a variable, or a function symbol with terms as arguments.
- A *literal* is a predicate symbol (or its negation) with terms as arguments. Terms and literals that do not contain any free variables are called *ground terms* and *ground literals* respectively.
- A *clause* is an expression of the form $h : -b_1, ..., b_n$, where the head h is a non-negated literal and the body $b_1, ..., b_n$ are literals, with the meaning that $b_1, ..., b_n$ together imply h.

Also, it is worth mentioning *lists*, which are ordered sets of elements are enclosed by "[" and "]" (the empty list is written as "[]"). Also, the anonymous variable is represented by a single underscore `_`. Its different occurrences may represent different literals. Regular variables, in contrast, start with a capital letter (e.g. `Agent, Action`) and their occurrences are all instantiated to the same ground literal within the scope of a clause.

ASL descriptions define, primarily, how a multiagent system evolves and transitions between states. A state s_t is defined as a finite set of ground literals, $s_t = \{f_1, ..., f_n\}$ (if p is an m-ary predicate symbol and $\alpha_1, ..., \alpha_m$ are ground terms, then $p(\alpha_1, ..., \alpha_m)$ is a ground term). The predicate symbols used to describe a particular action situation depend on the domain at hand and are a design choice by the user. Hence, the truth of a fact (i.e. a ground literal) f_i in state s_t holds iff $f_i \in s_t$. How the facts are initialized and evolve is a matter for the building of the EFG semantics from the ASL description (Section 4.2) and the interpretation of control rules (Section 3.2).

The keywords of ASL are gathered in Table 1. Most of these appear in `rule/4` arguments, and only `agent/1`, `initially/1`, `terminal/0` and `incompatible/2` are used as standalone predicates. In fact, the `agent/1` predicate symbol appears both within `rule` statement and as a standalone predicate. We start by reviewing the predicates that do not appear within rules. First, `agent(Ag)` denotes `Ag` as an individual susceptible of entering the action situation.

Table 1
Action Situation Language keywords, sorted into reserved predicate symbols (with their arity) and operators (with their type in parentheses).

Predicates		Operators	
`agent/1`	`rule/4`		
`participates/1`	`role/2`	`if` (prefix)	`then` (infix)
`can/2`	`does/2`	`where` (infix)	$\sim$ (prefix)
`initially/1`	`terminal/0`	`withProb` (infix)	`and` (infix)
`incompatible/2`			

Thus, this predicate provides information on the attributes of the community. If needed, domain-dependent predicates of the type `feature_name(Ag,Val)` can be added to encode agent attributes. For example, `age(alice,34)`.

Second, `initially(F)` indicates that literal `F` holds true at the start of the interaction, prior to any action being executed. For example, to indicate that at the start of the interaction, all agents, regardless of their role, are at the origin of coordinates, we need to include the clause `initially(at(Ag, position(0,0))) :- role(Ag,_).` `terminal/0` plays the opposite role, as it returns true whenever the conditions for halting the interaction are met. For example, to indicate that the interaction stops the moment an agent makes it to a finish line placed horizontally at some height y_{fl}, we need to include the clause `terminal :- at(Ag, position(_,Y)), Y>= y_fl.`

Finally, `incompatible(F,L)` states that literal `F` cannot be simultaneously true with the literals in list `L`. Formally, `incompatible(`f`,`L`)` means that $f \notin s_t$, where $s_t = \{l_i \mid l_i \in L\}$ is the state built from the literals in list L. For example, to indicate that agents cannot be at two different positions simultaneously, we need to include the clause `incompatible(at(Ag,Pos1),L) :- member(at(Ag,Pos2),L), Pos1\==Pos2.`

This example may raise the doubt of why we have chosen to have the second argument to `incompatible/2` literals be a list, instead of just a literal. We believe that having a list allows for greater flexibility in ASL descriptions. For example, suppose fact f_1 is only incompatible with facts f_2 and f_3 *simultaneously*, meaning that f_1 cannot be part of a state only if f_2 and f_3 are both part of it. This statement could not be expressed if the second argument to `incompatible/2` were a single literal. With our current syntax, it can be captured by the clause `incompatible(`f_1`,L) :- member(`f_2`,L), member(`f_3`,L).`

We move on now to `rule/4` predicates. All of its clauses, regardless of the component they target, follow the general template in Fig. 3, with the following four arguments:

1. An identifier `Id` that denotes the action situation where the rule is to be applied.
2. The `Type` of the rule, one of either *boundary*, *position*, *choice* or *control*.
3. The `Priority` of the rule. This is a non-negative integer that determines which statement is to prevail in case several rules lead to contradicting consequences. The rule statements that are supposed to reflect the physical principles of the domain are assigned priority 0 and are referred to as the *default rules*, while additional human-made regulations have strictly positive priorities. The reserved overwriting operator $\sim$ is introduced in order to have high priority rule nullify the effects of lower priority rules. We use the term *overwriting* instead of *negation* operator since ASL, as a logic programming language, follows negation as failure.
4. The content of the rule is expressed with an `if-then-where` statement (the three are all ASL reserved operators, see Table 1). The content of the `Condition` and `Consequence` fields is subject to syntactic constraints according to the type of the rule in question. We review these syntactic constraints in detail in the next section. The `Constraints` field always consists of a list of literals and constraints, whose free variables unify with those in `Condition` and `Consequence`. The separation of rule pre-conditions into a short `Condition` and a `Constraints` field is not technically indispensable, but rather a stylistic choice to help keep the syntax concise.

Besides predicate symbols, Table 1 also displays reserved operators, all of which appear within the scope of `rule/4` literals. Since ASL is implemented in Prolog, action situation descriptions can also make use of built-in Prolog predicates (notably `member(Elem,List)`, which has already been invoked) and operators, such as those for comparing terms. The least familiar of these are `Term1@<Term2` (also `@<=`, `@>`, `@>=`), which is interpreted as `Term1` preceding `Term2` in the standard order of terms (i.e. also considering characters). Additionally, The Prolog library for constraint logic programming

```
    Rule ::=    rule(
                    Id,
                    Type,
                    Priority,
                    if Condition then Consequence where Constraints
                ).
    Type ::=    boundary | position | choice | control
Priority ::=    0 | 1 | ... | ∞
```

Fig. 3. General syntax for *if-then-where* rules.

Table 2
Syntactic restrictions for the `Condition` and `Consequence` fields for every of the proposed rule types. α stands for a literal, i.e. a predicate symbol with terms as arguments.

Rule type	`Condition`	`Consequence`
Boundary	`agent(Ag)`	[∼]`participates(Ag)`
Position	`participates(Ag)`	[∼]`role(Ag,R)`
Choice	`role(Ag,R)`	[∼]`can(Ag,Ac)`
Control	`joint_action`	[`consequence`$_1$ `withProb` p_1, `consequence`$_2$ `withProb` p_2, ...]
	`joint_action` ::= `does(Ag,Ac)` [and `joint_action`]	
	`consequence` ::= α [and `consequence`]	

over real numbers is autoloaded with the ASL interpreter, part of the game engine. This library provides support for numerical constraints with syntax {*Constraints*} (for example, {`Payoff` < 10}).

2.1. Syntax by rule type

As introduced, ASL considers four rule types (boundary, position, choice and control) that target different action situation components in Fig. 2. First, the boundary rules are aimed at regulating the *participants* (1) components of action situations, as they designate which agents are able to enter the interaction. Second, position rules are responsible for assigning participants to their *roles* or *positions* (2). A participant may take on multiple roles. Third, choice rules assign *actions* (3) to roles, not to participants nor agents directly. Hence, an agent that is designated as a participant but is not assigned any role is irrelevant to the evolution of the interaction. Finally, control rules state what is the effect of actions on the system. Hence, this last rule type is directly responsible for the *control* (5) component of action situations.

Note that there is some disconnection between our four rule types and the seven variables within an action situation in Fig. 2. There are no dedicated rule types for the outcomes (4), costs and benefits (7), and information (6) variables. For the first two (outcomes, and costs and benefits), we argue that control rules are in charge. As they effectively regulate how does the state of the world evolve, they are also indirectly determining what *outcomes* are possible. Additionally, if one considers monetary and material rewards to be relevant in the current action situation, it is just enough to introduce a *payoff* predicate, initialized, for example, with `initially(payoff(Ag,0)) :- role(Ag,some_role)`. Then, its evolution can be regulated with control rules, that map (possibly joint) actions to monetary gains. A simple example of the use of control rules to regulate payoffs comes with the Iterated Prisoner's Dilemma example in Section 2.3.

As for the information component, it is left unaddressed in this early version of ASL. As we explain in Section 4, we only consider a restricted version of imperfect information in the semantics of any ASL description (much like in GDL game specifications). Although introducing information constraints to limit players' observability would certainly make for an interesting extension, we do not include it here as we anticipate that it would greatly increase the complexity of the resulting formal games, potentially opening the door for incomplete information and imperfect recall games.

As previously announced, the content of rule statements follows the syntax `if Condition then Consequence where Constraints`, with additional restrictions on the `Condition` and `Consequence` fields depending on the rule type. These restrictions are displayed in Table 2. Note that the boundary, position and choice rules all have an analogous syntax: one `agent/1`, `participates/1` or `role/2` literal as the `Condition`, and `participates/1`, `role/2` or `can/2` as the `Consequence`, respectively. Also, their `Consequence` literal might be preceded by the overwriting operator ∼, although it only makes sense to use it with non-default rules. The overwriting operator ∼ should not be confused with strong negation. The operator ∼ is used to have higher priority rules overwrite lower priority rules in case conflict arises between the consequences of different rule statements. Our computational model still uses the closed-world assumption and consequently negation as failure.

In contrast to the other rule types, control rules may have in their `Condition` multiple `does/2` literals concatenated by the `and` operator to account for the possibility that some effects are only brought about by joint actions, i.e. by having several agents perform some action simultaneously. If one wished to express the fact that several different actions `Act`$_1$ ... `Act`$_n$, performed by several different agents `Ag`$_1$ ... `Ag`$_n$, lead to the same effect (analogous to an `or` operator), one needs to include several control rules, one per each agent-action pair: `if does(Ag1,Act1) then Conseq`, ..., `if does(Agn,Actn) then Conseq`.

The `Consequence` of control rules, instead of a single literal, is a list where each of its members consists of (possibly several) literals concatenated with the `and` operator. In contrast with the other rule types, the literals that can be included in the `Consequence` field of control rule are not syntactically restricted. In general, they contain domain-specific predicate symbols, chosen to specifically reflect the state properties that are relevant to the action situation at hand.

The whole conjunction of predicates that makes up one potential consequence is assigned some probability with the operator `withProb`. The introduction of this operator is completely motivated by the desire to make control rules expressive enough as to encapsulate non-deterministic environments, where actions performed by the agents may have random

effects. In order for a control rule to be valid, the probability distribution over the potential consequences must be well-defined, i.e. p_i must fall in the range $[0, 1]$ for all i and must add up to unity.[1] Of course, deterministic environments can be expressed by having all control rules have one single consequence in their `Consequence` list, assigned probability equal to one.

In addition to stochastic effects, actions performed by agents may have different effects depending on the *context* where they are performed, *i.e.* the facts that hold true in the current state of the system. Such dependencies are handled through the `Constraints` field. Suppose the joint action profile $\mu = \texttt{does}(ag_1, ac_1)\ \texttt{and}\ \texttt{does}(ag_2, ac_2), \ldots$ has consequences c_1 when the state of the system is in s_1, consequences c_2 when in s_2, etc. To reflect this, the action situation description would need to include one control rule for every c_i, s_i pair, with the following content: `if` μ `then` c_i `where` $[s_i]$.

Regimentation and deontic modalities In the norMAS field, the term *norms* often refers to constraints and prescriptions on the behavior of agents intended to coordinate their actions [13]. Typically, norms are represented using a *deontic modality* (prohibited (P) or obliged (O), following [31]). Although the terminology "deontic modality" is inspired by deontic logic [32], P and O are rather used as operators to express constraints on actions (as in [33]) or states (as in [34,35]). This is in line with the operational semantics that we define for ASL in Section 4. Additionally, such normative constraints can be regimented or not [36]. In regimented domains, the nature of the application allows for the perfect enforcement of the desired constraints. Meanwhile, in non-regimented domains, some monitoring and sanctioning mechanism is implemented in order to deter agents from violating the norms.

In this short section, we provide guidelines on how prohibitions and obligations on *actions* can be specified through ASL rule statements, both in the regimented and sanctioning versions. We should note that, in order for any action to be possible in the first place, it needs to be explicitly included in a choice rule and assigned to the role capable of executing it.

Regimented constraints are addressed through choice rules, as they effectively allow or deny the possibility to execute some action. For example, suppose that the default rules (that model the "unconstrained" situation) are to be overwritten in order to prohibit some action, by banning the possibility of executing it. This situation is equivalent to introducing a higher priority choice rule of the form:

```
rule(Id,choice,N,if role(Ag,Role) then ~can(Ag,forbidden_action) where [ConditionsForProhibition
                                                                        ]).
```

where `N` is a strictly positive integer and `Role` is the position (following Fig. 2) that was originally assigned the action in question.

Similarly, regimented obligation can also be modeled through choice rules. To do so, we use the following equivalence between prohibition and obligation:

$$O\left(a_i \mid \psi\right) \Leftrightarrow P\left(a_j \mid \psi\right),\ \forall a_j \neq a_i \tag{1}$$

which states that, provided that some condition ψ holds true at the current state, the obligation to perform action a_i is equivalent to the prohibition of performing any other action a_j. Then, a regimented obligation can be expressed in ASL as:

```
rule(Id,choice,N,if role(Ag,Role) then ~can(Ag,Ac)
where [Ac\=obliged_action,ConditionsForObligation]).
```

Non-regimented norms assume that it is not possible to completely ban the execution of forbidden actions, and instead they attempt to discourage agents away from them. In ASL, this approach can be expressed through control rules. A deterrence for a forbidden action follows the template:

```
rule(Id,control,N,if does(Ag,forbidden_action) then [Punishment withProb P,
       NoPunishment withProb 1-P] where [ConditionsForProhibition]).
```

where `P` is the probability of detection violation.

Similarly to regimented norms, the leap from non-regimented prohibition to obligations is made thanks to Equation (1):

```
rule(Id,control,N,if does(Ag,Ac) then [Punishment withProb P, NoPunishment withProb 1-P]
                  where [Ac\=obliged_action,ConditionsForObligation]).
```

The addition of rules following the templates provided in this section is useful if one wishes to analyze the effects of introducing prohibition and obligation norms, either regimented or not, on a (possibly) unregulated social interaction. That is the reason why the priority `N` must be strictly positive, so it overwrites any conflicting default rules representing the unregulated interaction. This provides a big point of contact between this work and the concerns of the norMAS community [15], that deals with the representation and implementation of norms (obligations and prohibitions) in multiagent systems.

[1] If that is not the case, the game engine that interprets the rules will raise an error, and the game that corresponds to that ASL description will not be generated.

2.2. Valid ASL descriptions

Following the syntax of ASL, a valid action situation description $\mathbb{A}$ can be written as logic program composed of a finite set of clauses. We organize them into three exclusive subsets, one for every set of exogenous variables determining the structure of an action situation (see Fig. 1):

$$\mathbb{A} = \Delta \cup \Sigma \cup \Omega \tag{2}$$

where:

- Δ is the *agents base*, which includes the information on the agents and their attributes.
- Σ is the *states base*, which includes the information on biophysical features, plus clauses with `initially/1`, `terminal/0` and `incompatible/2` predicate symbols in their heads.
- Ω is the *rules base*, which contains the `rule/4` literals (i.e. bodyless clauses).

In order to ensure the unambiguous interpretation of an action situation description, some limitations are placed on the use of the reserved predicates within the body of clauses. Additionally, some minor directives are intended to ensure the clear separation between the community attributes, the biophysical features and the rules into three separate knowledge bases.

Definition 1 *(valid ASL description).* A *valid ASL description* $\mathbb{A}$ is a finite set of clauses split into three exclusive subsets $\Delta \cup \Sigma \cup \Omega$, where:

- `agent/1` predicate symbols appear only as the head of clauses in Δ.
- `initially/1`, `terminal/0` and `incompatible/2` predicate symbols appear only in the head of clauses in Σ. Also, `initially/1` clauses do not contain `can/1` nor `does/2` predicates in their bodies. This restriction reflects the fact that the initial state has to be completely determined *before* agents inspect what actions are at their disposal (`can/2`) or they choose one of those actions to perform (`does/2`). Furthermore, `incompatible/2` does not have a reserved predicate symbol as its first argument.
- `rule/4` predicate symbols appear only as literals (clauses with no body) in Ω, plus they follow the syntactic restrictions exposed in Table 2.

2.3. First example: iterated Prisoner's Dilemma

The best way to get a grasp on the ASL syntax is to go through a simple illustrative example. In this section, we show how the iterated version of the benchmark Prisoner's Dilemma game can be specified in ASL. Additionally, we take the opportunity to compare ASL to GDL in more detail, by differentiating the descriptions that both languages make of this benchmark. Its simplicity and familiarity to a wide range of audiences makes it a perfect candidate to be the first described with our language. Other more sophisticated examples are presented in Section 6.

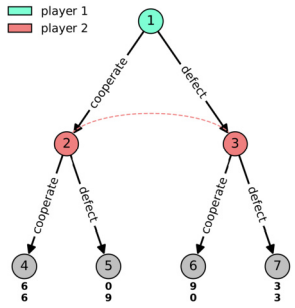
In the Prisoner's Dilemma game, two agents are put in identical positions, where they can choose one of two actions (either "cooperate" or "defect"), with no prior opportunity to communicate with one another. The normal and extensive-form representation of the game appear in Fig. 4. If an agent cheats on the other, the defector receives a large temptation payoff ($T = 9$) at the expense of the sucker ($S = 0$). If the two agents cooperate, they both receive identical reward payoffs ($R = 6$) that are larger than the punishment payoff ($P = 3$) they get when both choose to defect. This game has attracted a lot of interest from scholars in a wide range of disciplines because, although the most socially beneficial thing to do is for both agents to cooperate with one another, rational behavior stipulates that they should both defect, leading to worse individual and group reward. In game theoretical terms, the Nash equilibrium is not Pareto optimal. In its iterated version, agents typically start with wealth equal to zero. They play several consecutive rounds of the one-shot game, and their wealth is increased by the reward they get at every round. The interaction is halted after some pre-defined number of rounds have been completed.

The ASL description of the iterated Prisoner's Dilemma game appears in Listings 1 to 3. First, the agents and states knowledge bases appear in Listings 1 and 2 respectively. In the agents base, two agents are declared with no additional attributes. In the states base, the initial conditions are set to zero wealth for all prisoners. Also, a counter for the number of rounds is initialized. Finally, the `incompatible/2` clauses indicate that there can only be one `rounds/1` literal, as well as one `payoff/2` literal per agent in a state.

Second, the rule base is displayed in Listing 3. There are 8 rules in total: 1 boundary, 1 position, 2 choice and 4 control. They are all identified by the tag `ipd` and have priority equal to zero since they are the defaults. The boundary, position and choice rules are all very generic. All agents are allowed to participate by the boundary rules, they take on one role (that of a prisoner) according to the position rules and they are allowed to cooperate or defect with no further constraints by the choice rules.

	cooperate	defect
cooperate	6,6	0,9
defect	9,0	3,3

(a)



(b)

Fig. 4. The Prisoner's Dilemma game in the (a) normal-form and (b) extensive-form representations. Note the use of an information set (defined in Section 4.1) for player 2, denoted with the dashed line joining nodes 2 and 3, to capture the simultaneous nature of the moves.

```
1 agent(alice).
2 agent(bob).
```

Listing 1: Agents base Δ for the iterated Prisoner's Dilemma ASL description.

```
1 initially(payoff(P,0)) :- role(P,prisoner).
2 initially(rounds(0)).
3 terminal :- rounds(N),N>=3.
4 incompatible(rounds(_),L) :- member(rounds(_),L).
5 incompatible(payoff(P,_),L) :- member(payoff(P,_),L).
```

Listing 2: States base Σ for the iterated Prisoner's Dilemma ASL description, played for a total of three rounds.

```
1 rule(ipd,boundary,0,if agent(A) then participates(A) where []).
2
3 rule(ipd,position,0,if participates(A) then role(A,prisoner) where []).
4
5 rule(ipd,choice,0,if role(P,prisoner) then can(P,cooperate) where []).
6 rule(ipd,choice,0,if role(P,prisoner) then can(P,defect) where []).
7
8 % P1@<P2 avoids equivalent instantiations of some control rules
9 rule(ipd,control,0,if does(P1,_) and does(P2,_) then [rounds(M) withProb 1] where [P1@<P2,rounds(
      N),{M=N+1}]).
10 rule(ipd,control,0,if does(P1,cooperate) and does(P2,cooperate)then [payoff(P1,Y1) and payoff(P2,
      Y2) withProb 1] where [P1@<P2,payoff(P1,X1),payoff(P2,X2),{Y1=X1+6,Y2=X2+6}]).
11 rule(ipd,control,0,if does(P1,defect) and does(P2,defect) then [payoff(P1,Y1) and payoff(P2,Y2)
      withProb 1] where [P1@<P2,payoff(P1,X1),payoff(P2,X2),{Y1=X1+3,Y2=X2+3}]).
12 rule(ipd,control,0,if does(P1,cooperate) and does(P2,defect) then [payoff(P1,Y1) and payoff(P2,Y2
      ) withProb 1] where [payoff(P1,X1),payoff(P2,X2),{Y1=X1+0,Y2=X2+9}]).
```

Listing 3: Rule base for the iterated Prisoner's Dilemma ASL description.

The control rules exemplify the use of the `Constraints` field. The first one declares that, in order for the rounds counter to advance, two distinct participants must take some action. The other three control rules have identical structures as they all control the rewards received as a function of the joint actions at every stage of the game. Although there are four possibilities in the Prisoner's Dilemma game (see Fig. 4a), due to symmetry we need only three control rules. The dynamics of this environment are deterministic, hence all control rules have one single consequence with probability equal to unity.

Now that the syntax of ASL has been thoroughly explained, we are able to provide a comparison with that of GDL, whose description for the iterated Prisoner's Dilemma appears in Listing 4. The first small difference is that the GDL description is not split into components like the ASL one. Also, ASL separates the notions of *agent*, *participants* and *role*. In contrast, GDL merges the three concepts and declares any player participating in the game using the predicate `role/1`. Concerning

```
role(alice).
role(bob).

init(payoff(Ag,0)) :- role(Ag).
init(rounds(0)).
terminal :- rounds(N),N>=3.

legal(Ag,cooperate) :- role(Ag).
legal(Ag,defect) :- role(Ag).

next(rounds(M)) :- true(rounds(N)), M=N+1.
next(payoff(Ag1, Y)) :- does(Ag1,cooperate),does(Ag2,cooperate),role(Ag1),role(Ag2),Ag1@<Ag2,true
    (payoff(Ag1,X)),Y=X+6.
next(payoff(Ag1, Y)) :- does(Ag1,defect),does(Ag2,defect),role(Ag1),role(Ag2),Ag1@<Ag2,true(
    payoff(Ag1,X)),Y=X+3.
next(payoff(Ag1, Y)) :- does(Ag1,cooperate),does(Ag2,defect),role(Ag1),role(Ag2),true(payoff(Ag1,
    X)),Y=X+0.
next(payoff(Ag1, Y)) :- does(Ag1,defect),does(Ag2,cooperate),role(Ag1),role(Ag2),true(payoff(Ag1,
    X)),Y=X+9.
```

Listing 4: GDL description of the Iterated Prisoner's Dilemma game in infix syntax.

similarities, the `initially/1` and `terminal/0` clauses in ASL have the same function (and also practically the same spelling) as `init/1` and `terminal/0` clauses in GDL.

Regarding the `rule/4` literals in GDL, boundary and position rules do not have an equivalent in GDL since, as we have already mentioned, this latter language does not distinguish between agents, participants and roles, as it is not grounded in any social science theory but rather developed by and for the General Game Playing community. Choice rules in ASL are analogous to `legal/2` clauses in GDL, as both have the function of assigning actions to roles. Finally, control rules in ASL are analogous to `next/1` clauses in GDL, which contain as argument a literal that will hold true provided that actions in `does/2` predicates are performed and the facts included in `true/1` predicates are true in the current state.

The reader will have noticed that GDL clauses do not contain anything analogous to the rule priority introduced in ASL. This feature, we believe, sets ASL apart from GDL. It allows ASL description to be extended with, for instance, additional norms such as the ones suggested in the previous discussion on obligations and prohibitions. Therefore, it makes ASL much more suitable to the analysis of socioeconomic scenarios, where the interest is on the effect that changes in regulations will have on a given scenario, rather than on the construction of a new interaction model from scratch.

3. Rule interpretation

As introduced early on, an ASL description has its semantics automatically generated as a formal game model by a computational engine. To do so, the game engine has to repeatedly interpret the rules in place. This task is performed by the "interpreter.pl" script (see Fig. 1).

In this section, we go through the rule interpretation process in detail. We split it into two steps: (1) rule activation and (2) processing of consequences. Throughout, we denote an action situation description, split into its three components, again as $\mathbb{A} = \Delta \cup \Sigma \cup \Omega$. On several occasions, this set of clauses is expanded by ground literals on the participants $\phi = \{\texttt{participates}(ag_1), ...\}$, their roles $\rho = \{\texttt{role}(ag_1, r_1), ...\}$, the current state of the system $s_t = \{f_1, ..., f_n\}$ (where n is the number of necessary literals to completely describe the state of the system) and the actions executed by agents $\mu = \{\texttt{does}(ag_1, ac_1), ...\}$. All of these sets ($\phi$, ρ, s_t and μ) have a finite number of elements and do not belong to any of the components of the description (Δ, Σ or Ω), but are appended to the action situation description as a whole at concrete moments during the game construction procedure. As we will see in Section 4, the participants ϕ and roles ρ remain constant throughout the interaction, while the joint actions μ and the current state s_t change as the system evolves. Since the literals in s_t and μ change as the system evolve, we refer to them as *fluents*.

First, the interpreter has to find which rules in Ω are *active* given the current state of the system. In general, `rule/4` statements contain free variables that have to be instantiated to constants given the ASL description $\mathbb{A}$ and the current state of the system. When processing *boundary* rules, $\mathbb{A}$ does not need to be expanded. However, when processing *position* rules, the set of participant atoms ϕ has to be appended to $\mathbb{A}$. For processing *choice* rules, in addition to the set of participants ϕ, the set of roles ρ and the current state of the system s_t are necessary. Finally, for *control* rules, all of the previous extensions are necessary, plus the fluents denoting what actions μ agents are taking. When a rule is fully instantiated, we say that it has been *activated* or *triggered*. A rule may be activated multiple times, as many as possible instantiations of its free variables are possible.

The clause responsible for finding out active rules is `query_rule(Rule)`, shown in Listing 5. Its examination reveals that, in fact, a rule in ASL:

```
1 query_rule(rule(ID,Type,Priority,if Condition then Consequence where Constraints)) :-
2         rule(ID,Type,Priority,if Condition then Consequence where Constraints),
3         maplist(query,[Condition|Constraints]).
4
5 query(Q) :- call(Q).
6 query(A and B) :- query(A),query(B).
```

Listing 5: Prolog interpreter predicates to find which rules are activated given the current state of the system s_t.

```
rule(..., if Condition then Consequence where [Constraint1, Constraint2 ...]).
```

is equivalent to a traditional clause:

```
Consequence :- Condition, Constraint1, Constraint2, ... .
```

However, beyond its friendlier syntax as compared to traditional clauses, `rule/4` statements contain a `Type` argument, that allows to activate different kinds of rules depending on the step of the game construction process that the engine is in. Additionally, the `Priority` field helps sort out conflicts between norms with contradicting or incompatible consequences, and also filter out rules whose priority is over some threshold.

The rule activation step is common to all rule types, and it helps retrieve the consequences that rules have and that will have some effect on the resulting interaction. But first, these consequences need to be processed. Since boundary, position and choice rules all have analogous syntactic restrictions, the processing of their derived consequences is also shared. We deal with these three separately from the more complex control rules.

3.1. Processing of boundary, position and choice rules

As displayed in Table 2, boundary, position and choice rules have similar syntactic restrictions as their `Consequence` field contains a single literal. The consequences for all of these rule types are processed by the function Get-Simple-Conseqs (see Algorithm 1 in Appendix A). First, the (extended) action situation description is queried in order to find the active rules of the input type (Line 3). The $pr : f$ pairs are stored (where pr is the integer rule priority and f is the ground literal derived from the rule's `Consequence` field), and those whose priority is over some input threshold are ignored (Line 4).

The inclusion of such a *threshold* argument is, essentially, for convenience purposes. The reader will note that such an argument appears in all other algorithms presented in this work. The aim of this threshold is to allow an ASL user to write a single action situation description with rules of several priorities, and effortlessly obtain the formal representations of the social interaction when different rules are included, simply by tuning the *threshold* argument. This avoids the need to write (or uncomment) higher priority rules every time their impact in the interaction of interest wants to be assessed.

Next, the $pr : f$ pairs are ordered in descending order of `Priority` value (ties broken arbitrarily, Line 5). Then, the atom f derived from `Conseqs` is added to the output set if neither that same predicate nor its overwriting (with the prefix operator $\sim f$) have already been added to the output set by a higher priority rule (Lines 7-8). Finally, literals preceded by the overwriting operator are deleted before the set of facts is returned (Lines 9-10).

The rule interpreter relies on the fact that boundary, position and choice rules are *sound*. This means that Ω cannot contain two rules of the same type and priority with contradicting consequences (i.e. one rule overwrites the consequences of the other). For example, the following two rules should not both be included in the database:

```
rule(...,boundary,1,if agent(Ag) then participates(Ag) where [age(Ag,N),N>18]).
rule(...,boundary,1,if agent(Ag) then ~participates(Ag) where [age(Ag,N),N>18]).
```

If such rules were included, the output set of Get-Simple-Conseqs(..., boundary, 1) would not be deterministic, as it would depend on how the ties during the sorting procedure are broken.

Note that the ASL description $\mathbb{A}$ alone, without any additional fluents, is enough to process only the boundary rules. The set of participants $\phi = \{\texttt{participates}(ag_1), ...\}$ should be added to the action situation description before processing the consequences of position rules. Likewise, in order to process the consequences of choice rules both the roles $\rho = \{\texttt{role}(ag_1, r_1), ...\}$ and the current state $s_t = \{f_1, ..., f_n\}$ have to be added to $\mathbb{A}$. This data is necessary to ensure the proper activation of the various rule types.

To conclude the interpretation of simple rules, we illustrate how Algorithm 1 enforces the constraints of regimented prohibitions and obligations presented in Section 2.1, which are expressed through choice rules. We exemplify regimented prohibitions, the procedure for regimented obligations is analogous. Consider the following two choice rules:

```
rule(...,choice,0,if role(Ag,role) then can(Ag,ac) where []).
```

```
rule(...,choice,1,if role(Ag,role) then ~can(Ag,ac) where [condition_for_prohibition]).
```

The first is a default rule that assigns a generic action to a role regardless of the circumstances (note the empty list in the `Constraints` field). The second is a higher priority rule that bans these actions under some conditions.

Consider the execution of Algorithm 1 (with argument $type = control$) with an action situation description including the two rules above. Additionally, assume that the state description $s_t = \{f_1, ..., f_n\}$ fulfills the conditions for the second rule to be activated (i.e. $f_1 \wedge ... \wedge f_n \models \texttt{condition\_for\_prohibition}$). Then, kv on Line 6 of Algorithm 1 is assigned to:

$$kv \leftarrow [1 :\sim \texttt{can}(ag, ac), 0 : \texttt{can}(ag, ac)]$$

where we are assuming a generic instantiation $\texttt{Ag} \rightarrow ag$.

Then, the loop over kv in Line 7 of Algorithm 1 proceeds as follows:

Iteration	
1	$\mathcal{C} = \{\sim \texttt{can}(ag, ac)\}$
2	$f = \texttt{can}(ag, ac)$ and $\sim f \in \mathcal{C} \rightarrow \mathcal{C} = \{\sim \texttt{can}(ag, ac)\}$

Finally, $\sim \texttt{can}(ag, ac)$ is erased from $\mathcal{C}$ as it is a literal preceded by $\sim$. In the final output set, $\texttt{can}(ag, ac)$ is not included. Consequently, the constraint expressed by the rule with priority 1 has effectively removed action `ac` from the possible action available to an agent with some role `role`. This is indeed equivalent to a regimented prohibition.

3.2. Processing of control rules

Control rules have quite different syntax with respect to the other types, hence their consequences are processed by a different function, Get-Control-Conseqs (see Algorithm 2 in Appendix A). The added difficulty arises from the fact that possibly joint actions have, in general, stochastic consequences. In turn, every potential consequence does not just correspond to a single literal, but to several. Thus, it is not enough to simply return a set of ground literals, but rather a set of potential next states and a probability distribution over those, where every state is characterized by a set of fluents.

As control rules regulate how the state of the system transitions due to the actions performed by the agents, the pre-transition state $s_t = \{f_1, ..., f_n\}$ as well as the joint action whose execution we are examining $\mu = \{\texttt{does}(ag_i, ac_i), ...\}$ need to be added to the ASL description $\mathbb{A}$. Additionally, it must hold that for any participant ag performing some action ac_i (i.e. $\exists \texttt{does}(ag, ac_i) \in \mu$), then ag cannot be taking any other action simultaneously (i.e. $\nexists \texttt{does}(ag, ac_j)$ for any $ac_j \neq ac_i$).

The processing of control rule consequences starts off just as that of the other rule types. First, the instantiations of activated control rules are stored, together with their priority as a key (excluding those exceeding some input threshold) and sorted by descending priority (Lines 3-5). The set of potential next states $\mathcal{S}_{t+1}$ is initialized as a set containing only the empty set and unity probability assigned to the empty set (Lines 6-7).

Then, the function loops over the activated rules (Line 8). Every rule is composed of a probability distribution over joint consequences, which in turn contain several fluents concatenated by the `and` operator. The following check is performed (Lines 9-12): if any of the literals in any of the potential joint consequences is found to be incompatible with any literal in any of the provisional next states in $\mathcal{S}_{t+1}$, then that rule, despite having been activated, is ignored. This is done in order to avoid inconsistencies in the final probability distribution over the post-transition states. This check guarantees that the final probability distribution over $\mathcal{S}_{t+1}$ is correct, i.e. adds up to unity, provided that every control rule also has a proper probability distribution over its set of consequences.

If an activated control rule passes the check, its consequences are added to the set of potential next states (Lines 13-19). Every provisional post-transition state $s_{t+1} \in \mathcal{S}_{t+1}$ has its set of literals expanded with each of the joint consequences of the control rule being examined (Lines 16-17). The probability of the new expanded state is updated as the product between the probability of s_{t+1} prior to expansion times that of the joint consequences, provided by the active control rule (Line 18).

One final step is performed before returning the post-transition states and their probability distribution. A loop is run over the pairs of pre-transition state literals and the post-transition states (Line 20). If a pre-transition state literal is found to be compatible with the literals in the provisional post-transition state, it is dragged over and added to the potential next state (Line 21). When this loop is complete, the set of post-transition states $\mathcal{S}_{t+1}$ alongside with their probability distribution is returned. Note that, because of this final step, if we were to call Get-Control-Conseqs but no control rules were activated, the function would just return the pre-transition state ($\mathcal{S}_{t+1} = \{s_t\}$) with unit probability ($\mathcal{P}(s_t) = 1$).

The use of `incompatible/2` clauses together with the dragging of pre-transition state literals is the approach that ASL takes to tackle one of the problems that any action formalism has to inevitably address: the *frame problem* [37]. The frame problem states that when the state of an action is axiomatized, it should not be necessary to refer to the facts that are not affected by it. In ASL, control rules need only include the facts that do change in their `Consequence` field. By introducing the `incompatible/2` predicate symbol and having the rule interpreter drag the old ground literals that are consistent with those newly derived, the ASL allows for natural expression of control rules where only the facts affected by the actions in the `Condition` field need to be stated.

This approach to the frame problem that ASL takes is totally different from that taken by GDL. In GDL, given a state characterized by a set of fluents $s_t = \{f_1, ..., f_n\}$, all of the post-transition state literals have to be explicitly derived from `next/1` clauses (which, as explained in Section 2.3, are analogous to ASL's control rules). GDL does not make use of a predicate symbol analogous to ASL's `incompatible/2`, and therefore every fact that holds true after a transition from state s_t to state s_{t+1} has to be explicitly stated.

In the discussion of the previous Section 3.1 we introduced the notion of *soundness* for boundary, position and choice rules. An analogous concept applies to control rules. However, due to the more general nature of control rules in comparison with the other types (i.e. the literals in their `Consequence` can be anything), the soundness of control rules relies not on the direct comparison of their consequence literals, but on the use of the `incompatible/2` clauses.

To illustrate the notion of soundness in control rules, picture the following:

```
rule(...,control,N1,if does(Ag1,act1) and does(Ag2,act2) then Conseq1 ...).
rule(...,control,N2,if does(Ag1,act1) then Conseq2 where ...).
```

where `Conseq1` and `Conseq2` are lists of conjunctions of literals with a well-formed probability attached to each of them. The `Constraints` field has been omitted, and for the sake of discussion, assume it is equal for the two rules. Hence, since the conditions of the first rule imply those of the second, if the first rule is activated, then necessarily the second will be activated too.

When the two rules are triggered, three possible scenarios arise. First, the two rules may have compatible consequences. In this case, all the literals that appear in the consequence of one rule are compatible with all the literals in the consequences of the other. In this case, regardless of their priorities `N1` and `N2`, both rules will be processed by Get-Control-Conseqs.

Second, the two rules might not have compatible consequences, but the priority of one of them is strictly larger than the other ($\mathtt{N}_i > \mathtt{N}_j$). Then, the rule with the larger priority prevails, and the other is effectively discarded by Get-Control-Conseqs. The two scenarios discussed so far present no problem and are examples of *sound* control rules.

The third case corresponds to the different control rules having identical priorities and incompatibilities in their consequences. In this case, one of the two rules will prevail and the other discarded. However, it is not straightforward to predict which one will overcome the other, as that depends on the order in which they appear in the rule base Ω and/or the tie-breaking mechanism of the sorting subroutine in Line 5 of Algorithm 2. This is the problematic situation which we refer to as an *unsound* set of control rules.

We close this section by illustrating how the interpretation of control rules by Get-Control-Conseqs enforces the non-regimented norms presented in Section 2. Again, we illustrate punishment to deter participants from executing a forbidden action. The procedure for non-regimented obligations is analogous. Consider the two following control rules:

```
rule(...,control,0,if does(Ag,act) then [default1 withProb D1, ..., defaultN withProb DN] where
                                                                    []).
rule(...,control,1,if does(Ag,act) then [default1 withProb D1*(1-P), ..., defaultN withProb DN
                                *(1-P), Punish withProb P] where []).
```

The first rule expresses the effects on a generic action by some agent. In general, such an action is non-deterministic, with a set of default consequences each assigned a probability. To simplify the discussion, we leave the `Constraints` field as blank.

The second rule expresses the non-regimented prohibition of the generic action by a higher-priority control rule. Now, the probability of incurring punishment corresponds to P, while the probability of being undetected stands at $1 - P$. The relative proportion of the probabilities for the default consequences is preserved in the prohibiting rule with respect to the default one.

Additionally, consider the following set of `incompatible/2` clauses:

```
incompatible(default1, Punish).
                    ⋮
incompatible(defaultN, Punish).
```

When Get-Control-Conseqs is called on an ASL description containing such rules (with argument *threshold* ≥ 1, and assuming that the conditions are such to trigger the rules under discussion), the execution of the algorithm up to Line 7 (where the activated control rules are gathered), results in the kv local variable:

$$kv \leftarrow [\ 1\text{: } [\mathtt{default1\ withProb}\ D_1 * (1-P), ..., \mathtt{default1\ withProb}\ D_N * (1-P), \mathtt{Punish\ withProb}\ P],$$
$$0\text{: } [\mathtt{default1\ withProb}\ D_1, ..., \mathtt{defaultN\ withProb}\ D_N]\ ]$$

Then, the main loop in Get-Control-Conseqs (Lines 8 to 19 in Algorithm 2) first processes the first element in kv, i.e. the consequences derived from the rule with priority 1. This step results in the following post-transition states with probabilities:

$$\begin{gathered}\mathcal{S}_{t+1} = \{\{\texttt{default1}\}, ..., \{\texttt{defaultN}\}, \texttt{Punish}\} \\ \mathcal{P}(\{\texttt{default1}\}) = D_1 * (1 - P) \\ \vdots \\ \mathcal{P}(\{\texttt{defaultN}\}) = D_N * (1 - P) \\ \mathcal{P}(\texttt{Punish}) = P\end{gathered}$$

By the time the consequences derived from the control rule with priority 0 are processed, the incompatibility check in Lines 9 to 12 of Algorithm 2 returns true and discards the consequences derived from the default rule. Hence (and omitting the final steps of the algorithm, Lines 20-21), the returned post-transition states and their probabilities are the ones outlined above, where the punishment consequence is assigned some non-zero probability. The forbidden action is still a possibility by the acting agent (as the action is not banned by choice rules), but its execution might lead to the default consequences or to a deterring punishment.

4. ASL semantics

As reiterated throughout the text, an action situation description has its semantics grounded as an extensive-form game (EFG) with a restricted use of imperfect information. Furthermore, the formal game representation is augmented with a set of literals at those tree nodes that directly correspond to *states* of the system. In order not to get ahead of ourselves, we first need to review some common definitions from game theory and complement them with some definitions unique to this work. We then take a deep dive at how a formal game is built from an arbitrary action situation description, and which properties it is ensured to have as a consequence of the building mechanism. Finally, we present the complexity all the algorithms necessary to build the EFGs, included those introduced in Section 3.

4.1. Background on game theory

In the field of game theory, the simplest model of a multiagent interaction is provided by a normal-form game:

Definition 2 *(normal-form game).* A *normal-form game* is a tuple:

$$\text{NFG} = (P, (A_i)_{i \in P}, (U_i)_{i \in P})$$

where:

- P is the set of *players* (or agents).
- A_i is the set of *actions* available to player i.
- $U_i : A \to \mathbb{R}$ is the *utility function* for player i, which maps every possible joint action profile in $A = \bigtimes_{i \in P} A_i$ to a numeric quantity.

Normal-form games have been widely studied in a variety of fields, from microeconomics to evolutionary theory. However, they are not suitable to capture sequential interactions where agents might take several actions at different times. For this reason, normal-form games are sometimes referred to as stateless games. Moreover, normal-form games do not explicitly store information on possible random effects of joint actions.

To address these shortcomings, it is necessary to work with extensive-form games, where the strategic interaction is represented as a tree that agents transverse as they take actions. To account for stochastic effects, a new artificial player p_0 is added to the set of players, which is usually referred to as "nature" or "chance". We follow the notation by [38]:

Definition 3 *(extensive-form game).* An *extensive-form game* is a tuple:

$$\text{EFG} = (P, (X, E), T, W, \mathscr{A}, p, U)$$

where:

- P is the set of *players* (or *agents*).
- (X, E) is the *game tree*, where X is the set of nodes (or vertices) and $E \subseteq X \times X$ is the set of directed edges. One of the nodes $x_0 \in X$ is the *root* of the tree (with no incoming edges), such that for all other nodes $x \in X \setminus \{x_0\}$ there is a unique path from x_0 to x. The subset of nodes $Z \subseteq X$ with no outgoing edges are called the *terminal* (or *leaf*) nodes.

- $T : X \setminus Z \to P \cup \{p_0\}$ is the *turn function*, which assigns every non-terminal node to the player responsible for taking an action at that node, including chance moves. The turn function induces a partition $\Theta = \{\Theta_0, \Theta_1, ..., \Theta_{|P|}\}$ over the non-terminal nodes, by sorting them into subsets according to the player whose turn it is:

$$\Theta_i = \{x \in X \setminus Z \mid T(x) = i\}$$

- $W = \{W_i\}_{i \in P}$ is the *information partition* which, for every player $i \in P$, W_i corresponds to a partition of Θ_i. W_i splits all the nodes where i makes a move into mutually exclusive sets. Every $w \in W_i$ is called an *information set*. When player i makes a move, the information and actions available to i are exactly the same at any of the nodes that belong to the same information set.
  A game where all information sets are singletons is called *a perfect information game*. In this case, a player always knows precisely what state they are in before making the move. When that is not the case, games are said to have *imperfect information*. The information that is available to a player at a given information set is totally determined by the particular game being analyzed. Although information partitions are typically interpreted as partial observability, they can also be employed to simulate simultaneous moves. The prototype for this case can be found in the extensive form of the one-shot Prisoner's Dilemma game (see Fig. 4).
- $\mathscr{A} = (A(w))_{w \in W}$ denotes the *actions* available to the players, where $A(w)$ is the set of actions available at information set w for the player whose turn it is to move at that set. Actions label the outgoing edges from a decision (non-chance) node.
- p is a function that assigns to every chance node $x \in \Theta_0$ a *probability distribution* over its outgoing edges. Hence, it describes the nature of the environment and its stochastic effects.
- $U = (U_i)_{i \in P}$ is the *utility* function which assigns, for every agent, a numerical payoff for every terminal node, $U_i : Z \to \mathbb{R}$.

EFGs provide much richer representations of multiagent interactions than normal-form games. To generate the semantics of an ASL description, we will use a restricted form of EFGs to model all the possible ways by which a state s_t can evolve given the actions available to the participants at that state. We name this restricted form of EFGs as *game rounds*:

Definition 4 *(game round).* A *game round* is an extensive-form game with the following characteristics:

1. The root node is never a chance node.
2. There is, at most, one information set per player, $W_i = \{\Theta_i\}$, $\forall i \in P$ (players that do not take any action in a particular game round have an empty information partition).
3. For any two nodes x_1, x_2 that belong to the same information set, the length of the path from the root to x_1 and from the root to x_2 must be equal.
4. If $T(x) = p_0$ (x is a chance node), then all of its child nodes are terminal.

In practice, by condition 2 in Definition 4, in a game round every player has the opportunity to take only one action. Every depth level corresponds to the information set of one player (excluding the terminal nodes and possibly their immediate parents, which might be chance nodes). In a game round, every path of play corresponds to a joint action (i.e. every player only has the chance to select and perform one action), and the sequence of players whose turn it is to take is constant across paths. If the joint action has deterministic results, then the decision vertices are succeeded by a leaf node. Otherwise, if the joint action has stochastic effects, then a chance node succeeds the last decision node, before leading to a leaf node. Note that condition 3 ensures that game rounds always have *perfect recall*. This requirement will greatly facilitate the equilibria computations later on.

For example, the game in Fig. 4b holds the requirements to be a game round. Pointing this example raises the following question: why have we not chosen to model the interaction at a single time-step as a normal-form game, instead of going to the extent of using the much more loaded extensive form and then restricting it?

The answer is motivated by the ability of EFGs to explicitly store the information on probabilistic effects. This is not possible in normal-form games, where every action profile is mapped to a single payoff vector. Consider a joint action profile that leads to several reward allocations, each with some non-zero probability. In the normal-form representation, the payoff assigned to such an action profile would correspond to the weighted average over all the potential rewards, hence losing all information concerning the probability distribution over the outcomes. In our opinion, the loss of this information rules out normal-form games as suitable representations for action situations.

Given a state of the world, a game round models all the possible ways by which, through a single action each, agents might alter the state of the system. Since we want to model relatively complex action situations, with agents executing actions not just once but multiple times, several game rounds will be concatenated and merged into a larger extensive-form game. Next, we explain how game rounds are built from an action situation description and merged into an extensive-form game that grounds the semantics of an ASL description.

4.2. From action situation descriptions to games

Now, we move on to the process that takes an ASL description as input and constructs its game model semantics. This process is performed by the sequential generation of single game rounds that are concatenated into a larger game structure.

First, we address how a single game round is built by the function BUILD-GAME-ROUND (see Algorithm 3 in Appendix A). It starts off with the set of facts that characterize the pre-transition state s_t, which corresponds to the root of the game round tree. Then, the game tree is built in a *breadth-first* manner, by iterating over the agents that are able to take some action at s_t according to the choice rules (Lines 4-5 and 8-17). One information set is added for each of them (Line 9). At every iteration of this loop, the depth of the game round tree is increased by one level (Lines 11-16).

Once the tree skeleton has been built, it is time to find out which fluents hold true at the potential post-transition states $\mathcal{S}_{t+1}$ (Lines 19-32). Every terminal node corresponds to a different joint action executed from s_t, $\mu = \{\texttt{does}(ag_1, ac_1), ...\}$. Then, for every terminal node the joint action that leads to it from the root node is retrieved and added to the action situation description $\mathbb{A}$ (Lines 20-21). Next, GET-CONTROL-CONSEQS is called in order to find out the post-transition states that may be reached from that action profile (Line 23). In case there are no stochastic effects ($|\mathcal{S}_{t+1}| = 1$), the fluents of the single next state are assigned to the terminal node (Line 24). Otherwise, the terminal node is converted into a chance node, and additional descendants are added, one for every potential next state $s_{t+1} \in \mathcal{S}_{t+1}$ (Lines 25-31). The fluents at these new terminal nodes and the probabilities of the edge from the parent chance node are computed by GET-CONTROL-CONSEQS. Finally, the joint actions are deleted from the database before moving on to the next terminal node (Line 32).

By construction, the extensive-form game returned by BUILD-GAME-ROUND fulfills the properties of a game round according to Definition 4. Note that in every game round built by this function, only the root and terminal nodes correspond to actual states of the system, and they are assigned the fluents that hold true at that state (in fact the fluents that hold true at the root node are assigned before the tree emanating from it is constructed). The intermediate nodes between the root and the leafs do not correspond to actual states of the system. Hence, they are not assigned any fluents. They are auxiliary nodes, whose function is to capture the simultaneous nature of joint moves and the possibility for random effects in the environment.

BUILD-GAME-ROUND manages one last important point. For every joint action profile that is available at s_t, it checks whether executing joint action μ from s_t leads to termination (Line 22). If that is the case, the terminal node that corresponds to the execution of μ in state s_t will not be considered for further expansion (i.e. the construction of the game rounds that emanates from it) when the complete game for the action situation semantics is built.

Now that we know how to model the possible ways by which a state of the system s_t might evolve, the only step that is left is to concatenate multiple rounds into a larger game tree. This is precisely what the function BUILD-FULL-GAME does (see Algorithm 4). It only needs an action situation description as data, and maintains a queue of states susceptible to be expanded by BUILD-GAME-ROUND. The queue is initialized with the initial state of the interaction (Line 12), which is derived as the instantiations of the query `initially(F)` (Line 7). This is done *after* the participants and their roles have been added to the action situation description according to the boundary (Line 2) and position (Line 4) rules respectively, by calling GET-SIMPLE-CONSEQS with the appropriate *type* argument. As new game rounds are built, their respective terminal nodes are pushed to the queue as long as the joint actions leading to them from the root node in their game round do not trigger termination (Lines 28-30).

Additionally, every time a state is popped from the queue (Line 14) and its fluents added to the ASL description (Line 17), BUILD-FULL-GAME checks whether the termination conditions hold, *prior to the execution of any action* (Line 18). Adding this check to the one performed in BUILD-GAME-ROUND shows that, in an ASL description, a state may lead to termination either because the facts that characterize it command it, or because of the actions that have led to it from its pre-transition state.

Also, BUILD-FULL-GAME takes in a *max* argument that controls the maximum depth of the resulting game tree. The motivation for including this argument is as a safety stop in case the conditions for `terminal/2` clauses to return true are not met at all paths emanating from the root of the tree. If the action situation description is written in a such a way that the above situation arises, having a maximum allowed depth for the game tree being grown ensures termination of BUILD-FULL-GAME. Users confident in their ability to write action situation description where termination will always be dictated by `terminal/2` clauses returning true and do not wish for their EFGs representations to be cut short can use argument $max \rightarrow \infty$.

Finally, BUILD-FULL-GAME returns the extensive-form game that represents the action situation description, augmented by the facts that characterize the nodes that directly correspond to states of the system (those that are the root of game rounds plus the leafs). Hence, we can define the semantics of a valid ASL description by construction:

Definition 5 *(ASL semantics).* The *semantics of a valid ASL description* $\mathbb{A}$ correspond to an extensive-form game Γ with a restricted use of imperfect information, and a function $\mathcal{F}$ over a subset of the nodes in Γ. Γ and $\mathcal{F}$ are computed exactly by BUILD-FULL-GAME. Γ is built as the concatenation of game rounds, and is augmented with $\mathcal{F}$, which assigns a set of fluents for every node in Γ that correspond to a proper state of the system.

It should be noted that BUILD-FULL-GAME does not deal with the *utilities* assigned to the leaf nodes of the game tree. We choose not to assign utilities at construction time in order to allow flexibility in the valuations that agents make of

outcomes. If one is conducting a classical game-theoretical analysis based on material or monetary rewards, a predicate standing for such variable can be introduced and its evolution modeled through control rules, as stated in Section 2.1. Then, for every agent, the utility at a terminal node is extracted from the `payoff(Ag,x)` fluents assigned to that node. Alternatively, if one wishes to model other values that the agents might take into consideration in the particular action situation, the utilities can be assigned as a function of the fluents that hold true at every terminal node and/or the path of play from the root to the terminal node.

4.3. From games to action situation descriptions

Definition 5 specifies the semantics of any action situation description $\mathbb{A}$ as an EFG that respects the validity conditions outlined in Definition 1. To show the expressive power of the Action Situation Language, we now prove the equivalence between valid action situation description and extensive-form games that can be identified as a concatenation of game rounds. Despite that the focus of this work is on the transformation from an action situation description to an extensive-form game, we discuss here the inverse process for completeness purposes.

In order to continue, we must first clarify when an EFG is identifiable as a concatenation of game rounds. Consider an EFG Γ, and any of its subgames γ.[2] Now, take the largest subgame within γ, if any, and replace it by a terminal node. If after this transformation is applied to any of the subgames in Γ, it fulfills the conditions for a game round in Definition 4, then the overall game Γ is identified as a concatenation of game rounds.

We are now ready to establish the equivalence between EFGs and action situation descriptions:

Theorem 1. *An extensive-form game Γ is a concatenation of game rounds if and only if there exists an action situation description $\mathbb{A}$ such that $\mathbb{A}$ has its semantics grounded as Γ by* Build-Full-Game.

Proof. The reverse implication follows directly by construction from Definition 5. For the forward implication, we construct an action situation description *ad hoc*, and then prove that it certainly has its semantics grounded as the game in question.

Given Γ, start by having all of its players as agents: add `agent(`p`)` to the agent base Δ, $\forall p \in P$. Second, all agents have to be admitted as participants of the action situation and assigned a distinct role. Although in some cases (e.g. the Prisoner's Dilemma example in Section 2.3) several agents are designated the same role, for the sake of generality we assign a distinct role the every agent, named after themselves. Hence, the following two rules per participant are added to Ω:

```
rule(id,boundary,0,if agent(P) then participates(P) where []).
rule(id,position,0,if participates(P) then role(P,P) where []).
```

Third, denote the initial state as s_0 and denote all states as incompatible with one another. To do so, write the predicates `initially(`s_0`)` and `incompatible(_,L) :- length(L,1)` to the states base Σ (i.e. only one fluent s_i per state).

In the worst case, the input EFG Γ does not contain any regularities whatsoever, and so dedicated choice and control have to be added for every game round. To do so, it is necessary to first identify the chunks of the game tree that correspond to game rounds. Then, the dedicated rules need to be written for each of them.

The checking procedure discussed previously to identify whether an EFG corresponds to a concatenation of game rounds can be used to identify them. Take any subgame of the original extensive-form game, and replace its largest subgames into terminal nodes. The result corresponds to a game round.

Once every game round is delimited, denote its root as state s_i. For every player that takes some action within that game round, write a set of choice rules that assign to it the available actions, following the template:

```
rule(id,choice,0,if role(P,p) then can(P,a_ij^p) where [s_i]).
```

where a_{ij}^p denotes the j-th action that player p can take in state s_i.

The rule base Ω is completed by adding the control rules. In the worst case, one control rule has to be added for every *joint* action profile that players can take in every game round. Given a chunk of the overall game tree that has been identified as a game round, traverse it until either a chance node or a terminal node is encountered, whichever is first. Each distinct traversal will lead to a new control rule. Consider one of these paths from the root to either a chance or a terminal node in the game round. For every decision node x, add the literal `does(`$T(x), act$`)` to the `Condition` field of the control rule, where $T(x)$ is the turn function at node x and act is the action label of the outgoing edge. If the path ends in a terminal (non-chance) node, set the `Consequence` field to `[`s_{i+k} `withProb 1]` ($k = 1, 2...$). Otherwise, if a chance node is encountered, for every of its outgoing edges, append `[`s_{i+k+m} `withProb` p_m`]` $m = 1, 2...$ to the `Consequence` list. Finally, set `Constraint` to `[`s_i`]`, to indicate that the control rules only apply to the game round emanating from node s_i. Last of all, the individual terminal nodes of the overall game tree have to be designated as such, by adding `terminal :-` s_T clauses to Σ, one per terminal node.

[2] For a definition of subgame of an EFG, see [39, p.45].

Table 3
Time complexity of the algorithms presented in Appendix A.

Algorithm	Complexity
GET-SIMPLE-CONSEQS(..., *type*="boundary", ...)	$\mathcal{O}(R_b \cdot \log R_b)$
GET-SIMPLE-CONSEQS(..., *type*="position", ...)	$\mathcal{O}(R_p \cdot \log R_p)$
GET-SIMPLE-CONSEQS(..., *type*="choice", ...)	$\mathcal{O}(R_{ch} \cdot \log R_{ch})$
GET-CONTROL-CONSEQS	$\mathcal{O}(C^{R_{cnt}} \cdot (C \cdot L + F))$
BUILD-GAME-ROUND	$\mathcal{O}((R_{ch})^{Ag} \cdot C^{R_{cnt}} \cdot (C \cdot L + F))$
BUILD-FULL-GAME	$\mathcal{O}((R_{ch})^{(max+1)Ag} \cdot C^{(max+1)R_{cnt}} \cdot (C \cdot L + F))$

To prove that the *ad hoc* ASL description constructed above has its semantics grounded by BUILD-FULL-GAME as the original EFG Γ, we combine empirical and induction arguments. For an EFG that consists of $n = 1$ game rounds, we experimentally prove that such an action situation description has its semantics correctly grounded.[3] For a larger game with $n > 1$ game rounds, the subsequent game rounds are also individually correctly built, by the results obtained for $n = 1$. The subsequent game rounds are appended to the first one (whose root equals to the root node of the overall game tree) by changing the i sub-index in the action and state atoms (a^P_{ij} and s_i respectively) of the choice and control rules. □

The construction procedure outlined above is very tedious and results in a large ASL description (i.e. with a large amount of `rule/4`), even for small extensive-form games. However, it is generally expected that the extensive-form game will contain some symmetries or regularities that will avoid the need for a set of distinct choice and control rules per every game round.

In their original formulation, EFGs do not carry information on their nodes. However, according to the *ad hoc* ASL description above and the BUILD-FULL-GAME function, the resulting EFG will be expanded with a generic state fluent function $\mathcal{F}$ that assigns a generic state fluent s_i to those nodes that correspond to roots of game rounds. Also, in order to completely recover the original game, it is necessary to assign *a posteriori* the utilities to its terminal nodes. However, none of these two minor differences between the input EFG and the semantics grounded from the constructed ASL description are related to the structure of the game, i.e. the topology of the game tree.

4.4. Game construction complexity

Now that all the algorithms in this work have been reviewed, we discuss the complexity of grounding the semantics of an action situation description as an extensive-form game. Complexity results for the algorithms included in this work are presented in Table 3, which follows the following notation:

- $R_{b/p/ch/cnt}$: number of boundary/position/choice/control rules.
- C: maximum number of consequences per control rule (i.e. length of the `Consequences` list in control rules).
- L: maximum number of literals per consequence in control rules.
- F: maximum number of fluents that describe a state.
- Ag: number of agents in the agent base Δ.
- *max*: maximum allowed depth of the game tree, passed as a parameter to BUILD-FULL-GAME.

The complexity of GET-SIMPLE-CONSEQS is dominated by the sorting procedure in line 5 of Algorithm 1. Assuming that the Merge-Sort algorithm is employed, the complexity is $R \log R$, where R is the number of activated boundary, position and choice rules, which is assumed to be equal to the total number of rules of each type in the worst case. For the other algorithms, the complexity is dominated by the iterations over activated control rules, the number of consequences in their `Consequence` field C and the number of literals in every consequence L. To derive the complexity of BUILD-GAME-ROUND and BUILD-FULL-GAME, we assume that the number of actions available to every agent equals the number of choice rules in the ASL description.

The most important function for the purposes of this work is BUILD-FULL-GAME, as it is called once per every action situation configuration that we wish to assess. The complexity of this algorithm scales exponentially with the number of agents, the number of control rules, and the maximum allowed depth for the resulting EFG. Despite the exponential explosion, BUILD-FULL-GAME is well-suited for parallelization. The game rounds at every depth level of the overall game tree can be constructed independently from one another, i.e. the calls to BUILD-GAME-ROUND made within BUILD-FULL-GAME can be distributed over several computing nodes. Furthermore, information sets are, by construction, contained within one game round and do not include nodes belonging to different game rounds. Therefore, information sets do not need to be stored during computation and the construction of the overall EFG can be efficiently parallelized.

[3] See the `examples/adhoc` directory of the ASL distribution.

```
1  rule(ipd,choice,1,if role(P,prisoner) then ~can(P,defect) where [consecutiveDefections(P,N),N
       >=2]).
2  rule(ipd,control,1,if does(P,defect) then [consecutiveDefections(P,M) withProb 1] where [
       consecutiveDefections(P,N),{M=N+1}]).
3  rule(ipd,control,1,if does(P,cooperate) then [consecutiveDefections(P,0) withProb 1] where []).
4
5  initially(consecutiveDefections(P,0)) :- role(P,prisoner).
6  incompatible(consecutiveDefections(P,_),L) :- member(consecutiveDefections(P,_),L).
7
8  % rules to change the outcome of mutual defection
9  rule(ipd,choice,2,if role(P,prisoner) then can(P,defect) where []).
10 rule(ipd,control,2,if does(P1,defect) and does(P2,defect)
11     then [payoff(P1,Y11) and payoff(P2,Y12) withProb 0.5,
12           payoff(P1,Y21) and payoff(P2,Y22) withProb 0.5]
13     where [P1@<P2,payoff(P1,X1),payoff(P2,X2),{Y11=X1+0,Y12=X2+9,Y21=X1+9,Y22=X2+0}]).
```

Listing 6: Additional regulations with priority 1 to limit the number of consecutive defections (top), and with priority 2 to change the outcome of the mutual defection (bottom).

In the best-case scenario, there are more processing units than the maximum number of game rounds that have to be built at any depth level of the game tree. In such a case, the time complexity of Build-Full-Game would be reduced to that of Build-Game-Round, plus some communication overhead (i.e. the state fluents of a terminal node have to be transmitted to the host in charge of expanding the game round emanating from that node).

As an example, take the iterated Prisoner's Dilemma first introduced in Section 2.3. Its action situation description has the following parameter assignments: $R_b = R_p = 1$, $R_{ch} = 2$, $R_{cnt} = 4$, $C = 1$, $L = 2$, $F = 3$, $Ag = 2$ and $max = 3$. Hence, the complexity of Build-Full-Game for this ASL description is $\mathcal{O}(5 \cdot 2^8)$. If parallelization infrastructure is available and there are enough processing units ("enough" in this case meaning at least 16, which is the number of game rounds at the broadest level), then the complexity is reduced to that of Build-Game-Round, $\mathcal{O}(5 \cdot 2^2)$, a reduction by a factor of 2^6.

4.5. Follow-up on the iterated Prisoner's Dilemma

As an example of ASL semantics generation, we go back to the iterated Prisoner's Dilemma example presented earlier. The extensive form game built from the default rules presented there is shown in Fig. B.6 (B.1). The utilities that appear below the terminal nodes have been set by the `payoff(Agent,X)` fluents assigned to it.

To illustrate the introduction of higher priority rules and the impact that these have on the semantics, we propose two examples. First, we consider a ban on the number of consecutive defections that agents can take. Second, we introduce random changes to the outcome resulting from both agents playing "defect", where the outcome of the "defect"-"defect" joint action is as if one participant had defected while the other kept cooperating. The cheater and the sucker rewards are assigned randomly to the two agents. The intuition behind the first set of higher-priority rule is fairly straightforward, as it is a simple example of regimented norm (in this case a prohibition) to avoid the executing of too many actions considered to be detrimental.

The intuition behind the second set of higher-priority rules is not so obvious. To illustrate a possible real-world example that is captured by such rules, consider the following scenario. International talks to implement pollution-reduction policy to fight climate change (and the relatively little progress made during such talks) have often been identified as a Prisoner's Dilemma-type situation [40]. Consider two nations with competing economies that could introduce new regulations to reduce emissions, at an expense to their economic output. If they both, simultaneously, cooperate (i.e. transform their economies to reduce emissions), they will both be better off in the long run. If one defects (i.e. keeps polluting) while the other introduces environmental policy, the cooperator is expected to suffer a loss in competitiveness and therefore a hit to their economy, which the polluter takes advantage of. Finally, if both countries avoid emission reduction, the pollution levels will result in extreme weather events at an unmanageable frequency. Such events can partly destroy infrastructure, hence highly hindering the economic capacity of the location hit by them and providing an advantage to the country that has not suffered it, who can expect to sell their goods at a higher price. However, such events will, most likely, only hit infrastructure in one of the two countries at a time. Assuming that the two countries are equally likely to suffer from extreme weather events, this situation can be encapsulated by a random change in the "defect"-"defect" (or, in this case "keep polluting"-"keep polluting") outcome.

The additional rules needed to introduce those regulations appear in Listing 6. At the top, in order to limit the number of consecutive defections, it is enough to introduce one new choice rule with priority 1 that turns the action "defect" unavailable if the agent in question has defected twice in a row. This is an example of a prohibition rule in the regimented modality. Additionally, there are two new control rules that control the evolution of the `consecutiveDefections`/2 literals, as well as their initialization and the consideration that there may only be one `consecutiveDefections`/2

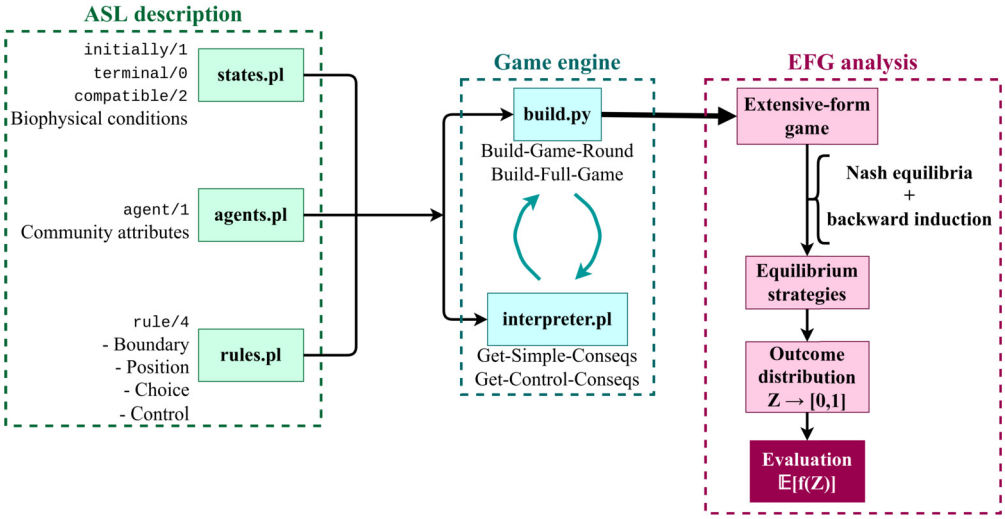


Fig. 5. Implementation of the computational model of the IAD framework.

literal per agent. The EFG that results from the interaction of this rule appears in Fig. B.7. Now, at some game rounds some agents only have action "cooperate" at their disposal, and the symmetry of the game tree is broken.

At the bottom of Listing 6, the rules that change the outcome of the mutual defection are introduced with priority 2. First, they need to undo the ban on several consecutive defections through a choice rule that recovers the action "defect" under any circumstances. Then, we present the control rule that is most representative of this regulation. Now, if both agents defect, the resulting outcome is *as if* one agent had cheated on the other. The selection for who becomes the *de facto* cheater and sucker is done by flipping an unbiased coin. The resulting game semantics for this configuration are displayed in Fig. B.8.

5. Implementation and computation of equilibria

Now that we have extensively presented the formal syntax and semantics of ASL, we go into the practical aspects of writing an action situation description, generating the EFG that derives from it and evaluating its structure.

The complete set-up to perform the *what-if* analysis of a rule configuration is displayed in Fig. 5. First, we write the ASL description with its clauses divided according to the separation $\mathbb{A} = \Delta \cup \Sigma \cup \Omega$ into agents, state-related information, and rules into three corresponding files states.pl, agents.pl and rules.pl.

These three files are fed to the game engine, that consists of the rule interpreter plus the game builder. The first is a Prolog script directly responsible for querying and processing the activated rules, and contains the implementations of Get-Simple-Conseqs and Get-Control-Conseqs. The second is a Python script that repeatedly communicates with the rule interpreter[4] and generates the action situation semantics through its implementation of the Build-Game-Round and Build-Full-Game functions.

Once the EFG is generated and utilities assigned to its terminal nodes, an assessment of its incentive structure can be performed with standard game-theoretical tools. In particular, the construction of EFGs as a concatenation of game rounds greatly facilitates their integration with *sequential rationality* solution concepts. Note that every game round is ensured to have perfect recall. Consequently, the resulting game tree also has perfect recall. A classic result in game theory by [41] establishes that in imperfect-information EFGs of perfect recall, for every *behavioral* strategy there is a corresponding equivalent *mixed* strategy, and vice-versa. In game theory, a behavioral strategy over an EFG for player i fixes, for every information set $w \in W_i$, a probability distribution over the available actions at w. Meanwhile, a mixed strategy for player i corresponds to a probability distribution over the sequences of actions that player i takes as they transverse the game tree.

Considering this result, we propose to use *backward induction* to compute the equilibrium strategies in an EFG generated from an ASL description. The general procedure operates as follows [39, p.46]:

1. Identify the *final* subgames of the overall EFG, i.e. those that do not have subgames of their own nested within them. Compute the equilibria strategy (or whichever solution concept) for the final subgames. The most common notion of rationality in game theory is expressed by the Nash equilibrium [42], however other options are possible, such as correlated equilibria [43].

[4] The communication between the Prolog script interpreter.pl and the Python script build.py is realized thanks to the open-source PySwip package.

2. Replace the subgames by terminal nodes whose utilities correspond to the expected average obtained by playing the equilibrium strategies.
3. Repeat the previous steps until the root node the expected utilities at the root node are computed.

There are several particularities of the EFGs we work with in this paper that make the above procedure particularly well-suited to compute the equilibrium strategies in action situations. First, the roots of all subgames of the overall EFG also coincide with the roots of the game rounds that have been concatenated to build it. Additionally, one should first take advantage of the fact that the game rounds that are closest to the leaf nodes of the overall game tree are also the smallest *subgames* within the tree, i.e. they are the *final* subgames of the overall EFG.

Second, these subgames in extensive form can be easily transformed into their induced normal form [44, Ch.5]. This is due to the fact that, by construction, players only make one move in a game round, and they are each assigned an information set relating "sibling" nodes at the same depth level of the game tree. Then, to compute the normal form from the extensive form of a game round, the chance nodes (if any) are substituted by terminal nodes with utility averaged over its children. The mapping from every joint action profile to a utility vector $\langle U_i \rangle_{i \in P}$ is computed by traversing all the paths in the game tree. The joint action profile corresponds to the actions by all players that are encountered along the path, while the utilities are extracted from the terminal nodes (which have originated from an average over the children of a chance node). Note that this procedure to convert an EFG into an NFG is only valid for game rounds fulfilling the properties of Definition 4, and does not generally apply to an arbitrary EFG.

Once the final game round has been converted from extensive into normal form, compute the equilibrium strategies at these normal-form game, using whichever solution concepts one deems appropriate. Finally, substitute the whole game round by a terminal node. Set its utilities to the average of the distribution induced by the previous equilibrium strategies together with the probabilities over chance nodes. Repeat these steps until the root of the original EFG is reached.

Note that the resulting equilibrium strategies computed by this procedure are given as *behavioral strategies*. For every game round, player i is assigned some probability distribution over the actions available to them at the only information set of player i at that game round. Consequently, the backward induction procedure assigns, for every information set in the game tree, one probability distribution for the actions available at that information set. This is precisely the definition of a behavioral strategy. Thus, following [41], we are guaranteed that there is some equivalent mixed strategy for the overall EFG.

Note also that the resulting equilibria strategies are not just rational but also *sequentially rational*. They correspond not just to equilibria over the whole game tree, but the restriction of the computed strategy to every subgame is also an equilibrium over it. This property is referred to as *subgame perfect equilibrium*.

So actually, limiting the use of imperfect information to simultaneous moves (hence ensuring perfect recall) does limit the expressive power of our language, but has major advantages when it comes to computing the equilibria that the game structure incentivizes. First, only "small" subgames have to be converted from the extensive to the normal form, as in every subgame any agent has at most one information set. Although this transformation can lead to exponential blow-up in the general case, we keep this complexity under control since it is single game rounds (where players take just one action) that have to be converted.

Second, instead of needing one big transformation from the whole EFG to a normal-form representation to find the mixed equilibria strategies directly, we can compute the equivalent behavioral strategies by performing several less intensive transformations over the game rounds only. An added advantage is that this approach avoids the contradictions often generated when turning a complex EFG into its normal-form representation. The computation of equilibria on the normal-form representation of a complete EFG may lead to equilibria that, although being rational are not *sequentially* rational, and hence constitute non-credible threats [45].

Once the resulting equilibria strategies are available, it is straightforward to find the probability distribution they induce, together with the probabilities assigned to chance moves, over the terminal nodes $Z$of the game tree (see the "Outcome distribution" box in Fig. 5). Finally, an evaluation criteria (which we generically denote as a function f over the leaf nodes Z) is defined, and its expected value is computed given the induced outcome distribution. This simple computation concludes the process from a rule configuration (an ASL description) to its evaluation. Now, the communities of agents involved can quantify whether the rules in place are in accordance with their idea of a "good" outcome.

The backward induction procedure outlined previously in this section is included with our ASL distribution. Step 2 of the procedure (i.e. the computation of Nash equilibria at every normal form subgame) is achieved by applying the incentive minimization approach of [44, p. 104]. Also, the computation of distribution over outcomes given the subgame perfect equilibria strategies is also included as part of the code distribution. The distribution over outcomes is computed as the product of the probabilities of all the action in the path from the root to that terminal node (or the probability of nature moves, if there are any). For all the example in this paper, we use the Nash equilibria and the computation of the distribution over outcomes implemented alongside the ASL distribution.

To exemplify the implementation issues presented in this section, we turn to the iterated Prisoner's Dilemma running example. We verify that the different rule configuration we have presented (the default ones in Section 2.3 and the additional

rules in Section 4.5) do indeed encourage different behaviors by the agents and hence lead to different outcomes. These results are presented in Tables B.4 to B.6 for the default rules, the rules limiting the number of consecutive defections and the rules banning the mutual defection, respectively. Tables B.4 to B.6 show, for every rule configuration, the equilibrium strategy induced by the structure of the EFG (obtained as the Nash equilibria) and the probability distribution that these equilibrium strategies induce over the terminal nodes. Note that equilibrium strategies are presented as behavioral strategies (i.e. a probability distribution over the action available at every information set).

As expected, the default Prisoner's Dilemma game has its equilibrium at pure defection by all players at all rounds of the game. However, the introduction of the limit on consecutive defections does encourage some cooperative actions by the participants at some decision points, and hence the outcomes with non-zero probability all lead to higher payoffs for both players. Finally, with the rules implementing the ban on mutual defection, agents go back to defecting at all rounds of the game. However, because of the randomness introduced, several outcomes are equally likely, with unequal distribution of payoffs.

6. Further examples

Before presenting the final conclusions, we illustrate the versatility of the ASL language with two more examples. These are more sophisticated than the very generic Prisoner's Dilemma, and also more interesting from the policy analysis and socio-economics perspective.

6.1. Axelrod's (meta)-norms game

First, we use ASL to model the norms and metanorms games, originally proposed by Robert Axelrod [46].[5] This action situation can be seen as a more elaborated version of the Prisoner's Dilemma. In the *norms* game, an individual i has the opportunity to defect and benefit at the expense of others. If they do not defect, no benefits or costs are incurred by anyone. If they do defect, then there is a fixed probability that they will be detected by a monitor j. If the cheater is detected, then the monitor can choose to punish them at a cost to themselves, but also imposing a large loss to the cheater.

In an extension called the *metanorms* game, the monitor j is themselves being watched by a meta-monitor k. Similarly to the norms game, the meta-monitor may detect with some probability if the monitor has neglected their duty (i.e. has chosen not to sanction i despite detecting their defection). In that case, the meta-monitor may decide whether to punish the monitor or not.

The norms and metanorms games are interesting examples for two reasons. First, they illustrate the use of non-regimented norms, as "bad" actions (defection by an individual or neglect by the monitor) always remain feasible actions. Furthermore, the agency of those responsible for monitoring and sanctioning is explicitly introduced into the game, instead of being idealized away as part of the environment dynamics (as in e.g. [34]).

The rules structuring the *norms* game are designated as the default ones and hence have priority zero. The meta-monitoring is introduced with additional regulations of priority 1. The resulting games and state fluent semantics appear in Fig. B.9 (see Appendix B.2). The norms and metanorms game trees are constructed with BUILD-FULL-GAME using arguments *threshold*=0 and *threshold*=1 respectively. The utilities that appear below the terminal nodes are set from `payoff(Ag,X)` literals, assigning to agent `Ag` the utility value `X`.

The state fluents for the nodes that correspond to actual states of the systems are also shown. For the terminal nodes, the probability distribution over the leaf nodes induced by the equilibrium strategies given the utilities appears next to their fluents. The equilibrium strategies themselves are displayed in Table B.7. These results show that, given the current payoff structure of this action situation, the introduction of a meta-monitor does not have any positive effect. Before k joins the interaction, agent i chooses to defect and the monitor j chooses not to sanction i if detected. However, the addition of k does nothing more that perpetuate "bad" agent behavior, as k chooses not to sanction j if their negligence is detected.

6.2. Ostrom's fishing game

The last example for which we write an ASL description first appeared in [47, Ch. 4].[6] It illustrates a theoretical analysis of how community-crafted rules are capable of transforming the opportunity structure that individuals face in common-pool resource environments. This is the richest example we present of this work, as it illustrates the use of community attributes, non-default position rules and biophysical features.

In this example, two fishers have access to an open-water fishery. Starting at the shore, they may go to one of two fishing spots (one is assumed to be more productive than the other). If, after the first trip, both fishers meet at the same spot, they can choose to stay or leave. If they meet yet again at the same spot after that second action, they inevitably fight. The winner of the fight is determined by flipping a biased coin, whose probabilities depend on the relative strengths of every fisher.

[5] See the `examples/metanorms` directory of the ASL distribution.
[6] See the `examples/fishers` directory of the ASL distribution.

This is the first example that utilizes agent attributes (`speed/2` and `strength/2`). The relative difference between the strength and speed of the two agents is used to compute the probability that one will win a fight or a race over one of the fishing spots. Also, this is the first example that adds biophysical conditions (two fishing spots declared with predicate `fishing_spot/1`). The declaration of fishing spots determines some of the actions that the fishers may take, since they can only leave the shore for one of them. Overall, both the community attributes and the biophysical conditions in this example strongly influence the resulting EFG, however we keep them constant throughout the analysis as they are assumed to be non-changeable in the short to medium term. Also, note how the user is free to choose the predicate symbols (as long as they are not reserved keywords, see Table 1) for community attributes and biophysical conditions. This is the case because, unlike rules, community attributes and biophysical conditions do not require custom querying and interpretation functions.

The boundary and position rules in this action situation are analogous to the other examples, i.e. all agents participate and they all take on the same role, that of *fisher*. Initially, both agents start at the shore. The interaction halts when each fisher is in a different spot, or when a fight has occurred. At every state, one fisher may only be at one place, and there can only be one loser and one winner of fights or races (the introduction of a racing condition is introduced when we introduce higher-order rules).

The choice rules are also very easy to interpret. They state that fishers at the shore can go to any of the fishing spots. Once there, they may choose to stay or leave. As for the control rules, they assume that fishers always get to wherever they intended to go (i.e. boat engines never break down). The last control rule states that, if fishers are at the same fishing spot and they both take the same action, a fight will ensue (since they meet again either at the same spot or at another one). The semantics of this ASL description (using the rules with priority equal to zero) are displayed in Fig. B.10.

In order to avoid violent fights, additional higher priority rules can be introduced. The first option is to implement a *first-in-time, first-in-right* scheme. Now, if fishers depart from the shore with the same destination spot, they race to get there. The first to get there is guaranteed to keep the spot, while the slower fisher has to leave. The winner of the racing contest is not determined by their `strength/2` attribute, but by their `speed/2` instead. The corresponding semantics of this rule configuration appear in Fig. B.11.

Another possibility is the implementation of a *first-to-announce, first-in-right* scheme. In this case, one of the participating agents is randomly assigned to a new role, which we refer to as *announcer*. Before anyone has left the shore, the announcer has to broadcast the spot where they intend to fish. Then, if they hold their promise and go there, they are guaranteed it, i.e. they always win the race to get there. If the announcer goes to a different spot than the one they have broadcast, a race ensues between the two fishers, analogous to the *first-in-time, first-in-right* scheme.

The *first-to-announce, first-in-right* scheme shows the first instance of a non-default position rule. In this case, one randomly chosen fisher is designated as the announcer. Assuming this additional role provides the participant with new actions, as well as conditions the actions of others (e.g. fishers cannot leave the shore before the announcer has announced their fishing spot of choice). Note that the announcer role is in addition to the role of fisher, not in substitution of it. This extra position rule effectively breaks the symmetry that participants previously had, when they both assumed the same roles and hence could execute the same actions. This feature definitely adds to the richness and complexity of the interaction.

Note that rules for this latter configuration have priority 2, since they are added *on top of* the rules for the previous *first-in-time, first-in-right* scheme. This is the first example that illustrates a non-default position rule, such as the one that creates the *announcer* role. The game semantics for this rule configuration appear in Fig. B.12.

The utilities at the terminal nodes of Figs. B.10 to B.12 are set by assigning the following costs and benefits to actions and outcomes: a fisher keeps a spot to themselves or wins the fight over it ($v_1 = 10, v_2 = 5$), a fisher looses a fight ($d = -6$), a fisher travels between spots ($c = -2$). The resulting equilibrium strategies, obtained with an identical computation to the previous examples, appear under the corresponding game trees.

We evaluate the resulting outcome distributions qualitatively. For the default rule configuration, violent outcomes are predicted for over 50% of the paths of play (terminal nodes 14 through 27). Both the *first-in-time, first-in-right* and the *first-to-announce, first-in-right* schemes avoid violence, and hence promote more socially desirable outcomes. However, for the *first-in-time, first-in-right* configuration the fishers still go to the same spot and compete for it. This outcome is avoided in the *first-to-announce, first-in-right* scheme. Now, the announcer prefers to proclaim the most productive spot (spot 1) and remains faithful to their announcement. The other fisher, then, chooses to go for the other spot. Hence, honest announcements are encouraged, and any sort of competition is prevented with the *first-to-announce, first-in-right* scheme.

7. Conclusions

In this work, we have presented a complete computational model of Elinor Ostrom's Institutional Analysis and Development framework. It is based on the Action Situation Language, a novel logical language for the description of action situations, whose friendly syntax is highly tailored to the components identified in the IAD framework. In particular, the IAD framework identifies three sets of exogenous variables that are responsible for shaping social interaction: biophysical conditions, attributes of the community, and rules. The computational model we propose respects this distinction and stores the knowledge corresponding to each of the three exogenous variables separately. In this work, special attention is paid to the *rules* variable, as this is the only one susceptible to changes in the short term. Consequently, the Action Situation Lan-

guage includes a mechanism to solve conflicts between contradicting rules, that allows new rules with higher priority to override older rules with lower priority.

The Action Situation Language is complemented with a game engine that automatically generates the semantics of a description as an extensive-form game. The resulting EFG has some self-imposed limitations, however these greatly facilitate the computation of equilibrium strategies, the distribution over outcomes and their evaluation. We have illustrated the use of ASL and the complete process from an action situation description to the evaluation of its impact with some examples. Notably, the fishers action situation of Section 6.2 demonstrates the complete analysis, where the outcomes are evaluated in terms of the avoidance of violence, competition, and honesty by the participants. Additionally, it shows how the introduction of suitable regulations is able to steer the system towards more desirable end states.

The work presented here has several limitations and is susceptible to extensions in several directions. We point to five potential research directions that may take the work presented in this paper as a starting point:

1. The incorporation of an *information* rule type could greatly enhance the expressive capabilities of ASL, as it could potentially provide the syntax for games with imperfect information (beyond simultaneous moves), or games with imperfect recall. Work in this direction should explore what changes need to be incorporated to the Build-Full-Game algorithm to include more sophisticated schemes of information accessibility.
2. The *possibility for dynamic boundary and position rules* that are queried not just before the interaction kicks off, but also while it is ongoing. Such a refinement would allow agents to get in and out of an action situation and/or switch roles dynamically.
3. Studies on the *relationship between ASL and logical action formalisms* such as Situation Calculus, to expand the reasoning schemes applicable to an ASL description. For example, how should ASL rule statements be translated into Situation Calculus domain axiomatizations?
4. Work on the formal verification of an action situation description, in terms of its *validity*, *soundness* and *relevance* of the included rules. For example, a position rule assigning a participant to an agent with no actions available has no effect on the outcomes of the interaction. Interesting work could be developed to define this sort of mutual dependencies between rule statements and other components of the action situation description.
5. Studies on the *nesting of action situations*. In this work, we have focused on *operational* action situations, where agents interact directly with one another and their shared environment. However, ASL could also be used to describe *collective-choice* action situations (e.g. a negotiation domain or a voting procedure), whose outcomes result in the implementation of new rule statements on an operational action situation. The linkages between two such ASL descriptions are also worth exploring.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by the EU WeNet project (H2020 FET Proactive project #823783), the EU TAILOR project (H2020 #952215), the RHYMAS project (funded by the Spanish government, project #PID2020-113594RB-100) and the VALAWAI project (Horizon #101070930).

Appendix A. Algorithms

For every algorithm pseudo-code, the following is specified:

- **Input:** its arguments.
- **Output:** its return value(s) and/or data structure(s).
- **Data:** the information stored in the database being consulted. In general, all functions call upon the original action situation description $\mathbb{A}$. Additionally, the facts corresponding to the current state s_t and/or the action profile executed μ might also be necessary.
- **Description:** Short documentation on the procedure implemented by the function.

Algorithm 1: Function GET-SIMPLE-CONSEQS(*id*, *type*, *thres*).

Input : *id* ▷ string
type ▷ one of either "boundary", "position" or "choice"
thres ▷ non-negative integer
Output : $\mathcal{C}$ ▷ a set of ground atoms
Data : $\mathbb{A}$ ▷ an action situation description
$\phi = \{\texttt{participates}(ag_i)\}$ ▷ set of participant fluents (if *type*="position")
$\rho = \{\texttt{role}(ag_i, r_i)\}$ ▷ set of role fluents (if *type*="choice")
$s_t = \{f_1, ..., f_n\}$ ▷ set of state fluents (if *type*="choice")
Description : Get the participants, roles or available actions entailed by the boundary, position or choice rules respectively.

1 **Function** GET-SIMPLE-CONSEQS(*id, type, thres*)**:**
2 $kv \leftarrow []$
3 **foreach** instance of `?-query_rule(rule(`*id*`,`*type*`,`*pr*`,if Cond then Conseq where Constr))` **do**
4 **if** $pr \leq thres$ **then** kv.APPEND(pr : `Conseq`)
5 $kv \leftarrow$ SORT-BY-DESCENDING-KEY(kv)
6 $\mathcal{C} \leftarrow \{\}$
7 **for** (pr : f) pair in kv **do**
8 **if** $f \notin \mathcal{C}$ **and** $\sim f \notin \mathcal{C}$ **then** $\mathcal{C} \leftarrow \mathcal{C} \cup \{f\}$
9 $\mathcal{C} \leftarrow \{c \in \mathcal{C} \mid c \neq \sim f\}$
10 **return** $\mathcal{C}$

Algorithm 2: Function GET-CONTROL-CONSEQS(*id*, *thres*).

Input : *id* ▷ string
thres ▷ non-negative integer
Output : $\mathcal{S}_{t+1} = \{s^1_{t+1}, s^2_{t+1}, ...\}$ ▷ the set of potential next states, each corresponds to a set of fluents
$\mathcal{P} : \mathcal{S}_{t+1} \rightarrow [0, 1]$ ▷ a probability distribution over the next states
Data : $\mathbb{A}$ ▷ action situation description
$s_t = \{f_1, f_2, ...\}$ ▷ set of facts that hold true at the current state
$\mu = \{\texttt{does}(ag_1, ac_1), \texttt{does}(ag_2, ac_2), ...\}$ ▷ joint action profile
Description : Get the post-transition state fluents and their probabilities that derive from performing some joint action in a pre-transition state.

1 **Function** GET-CONTROL-CONSEQS(*id, thres*)**:**
2 $kv \leftarrow []$
3 **foreach** instance of `?-query_rule(rule(`*id*`,`*control*`,`*pr*`,if Cond then Conseqs where Constr))` **do**
4 **if** $n \leq thres$ **then** kv.APPEND(pr : `Conseqs`)
5 $kv \leftarrow$ SORT-BY-DESCENDING-KEY(kv)
6 $\mathcal{S}_{t+1} \leftarrow \{\{\}\}$
7 $\mathcal{P}(\{\}) = 1$
8 **for** (pr : *conseqs*) pair in kv **do** // loop over activated control rules
/* *conseqs* = [c_{11} and c_{12} and ... withProb p_1,
c_{21} and c_{22} and ... withProb p_2,
...] */
/* check that every fluent in the consequences is consistent with the facts already established in the potential next states */
9 **for** (c_{i1} and c_{i2} and ... withProb p_i) in *conseqs* **do**
/* c_{ij} refers to the j-th fluent of the i-th consequence in the list of consequences induced by the control rule */
10 $\mathcal{C}_i \leftarrow \{c_{ij}\}_{c_i = c_{i1} \text{ and } c_{i2} \text{ and } ...}$
11 **for** (c_{ij}, s_{t+1}) in $\mathcal{C}_i \times \mathcal{S}_{t+1}$ **do**
12 **if** `?-incompatible(`c_{ij}, s_{t+1}`)` returns `true` **then** move to the next (pr : *conseqs*) pair // aka go to line 8
/* the activated rule consequences are consistent with $\mathcal{S}_{t+1}$ */
13 $\mathcal{S}'_{t+1} \leftarrow \{\}$
14 **for** $s_{t+1} \in \mathcal{S}_{t+1}$ **do**
15 **for** (c_{i1} and c_{i2} and ... withProb p_i) in *conseqs* **do**
16 $\mathcal{C}_i \leftarrow \{c_{ij}\}_{c_i = c_{i1} \text{ and } c_{i2} \text{ and } ...}$
17 $\mathcal{S}'_{t+1} \leftarrow \mathcal{S}'_{t+1} \cup \{s_{t+1} \cup \mathcal{C}_i\}$
18 $\mathcal{P}(s_{t+1} \cup \mathcal{C}_i) \leftarrow \mathcal{P}(s_{t+1}) \cdot p_i$
19 $\mathcal{S}_{t+1} \leftarrow \mathcal{S}'_{t+1}$
20 **for** $(f_i, s_{t+1}) \in s_t \times \mathcal{S}_{t+1}$ **do** // drag compatible facts from s_t over to s_{t+1}
21 **if** `?-incompatible(`f_i, s_{t+1}`)` returns `false` **then** $s_{t+1} \leftarrow s_{t+1} \cup \{f_i\}$
22 **return** $\mathcal{S}_{t+1}$, $\mathcal{P}$

Algorithm 3: Function BUILD-GAME-ROUND(*id*, *thres*).

Input : id ▷ string
$thres$ ▷ non-negative integer
Output : γ ▷ game round
F ▷ set of fluents assigned to γ's terminal nodes
$\tau : Z \to \{0, 1\}$ ▷ function mapping whether termination conditions are met at a terminal node ($\tau = 1$) or not ($\tau = 0$).
Data : $\mathbb{A}$ ▷ action situation description
$s_t = \{f_1, f_2, ...\}$ ▷ set of facts
Description : Given a state characterized by a set of facts, model all the ways by which it may evolve as a game round.

Function BUILD-GAME-ROUND(*id, thres*)**:**
  $k \leftarrow 1$, $X \leftarrow \{k\}$, $x_0 \leftarrow k$, $k++$
  $E \leftarrow \{\}$
  /* set players to be those participants that can take some action */
  $M \leftarrow$ GET-SIMPLE-CONSEQS(id, choice, $thres$) = $\{\texttt{can}(ag_1, ac_1), \texttt{can}(ag_2, ac_2), ...\}$
  $P \leftarrow \{ag_i \mid \exists \texttt{can}(ag_i, ac) \in M\}$
  /* STEP 1: Build the game tree in a breadth-first manner */
  $w \leftarrow \{x_0\}$, $w' \leftarrow \{\}$ // current and next information sets
  $W \leftarrow \{\}$, $\mathscr{A} \leftarrow \{\}$ // information partition and actions
  for $player \in P$ **do**
    $W_{player} \leftarrow \{w\}$, $W \leftarrow W \cup \{W_{player}\}$
    $A(w) \leftarrow \{ac \mid \texttt{can}(player, ac) \in M\}$, $\mathscr{A} \leftarrow \mathscr{A} \cup \{A(w)\}$
    for $x \in w$ **do**
      $T(x) \leftarrow player$
      for $action \in A(w)$ **do**
        $X \leftarrow X \cup \{k\}$, $w' \leftarrow w' \cup \{k\}$
        $E \leftarrow E \cup \{(x, k)\}$, $label(x, k) \leftarrow action$
        $k++$
    $w \leftarrow w'$, $w' \leftarrow \{\}$
  /* STEP 2: Get the facts and chance moves at the terminal nodes */
  $p \leftarrow \{\}$ // probability over chance moves
  for $z \in Z \subseteq X$ **do** // Z is the subset of terminal nodes
    // action profile from root to terminal node
    $\mu = \{\texttt{does}(ag, ac)\}_{\forall ag = T(x_i),\ ac = label(x_i, x_{i+1}) \mid (x_i, x_{i+1}) \in \text{PATH}(x_0, z)}$
    $\mathbb{A} \leftarrow \mathbb{A} \cup \mu$ // assert action profile into database
    if `?-terminal` returns `true` **then** $t \leftarrow 1$ **else** $t \leftarrow 0$
    $\mathcal{S}_{t+1}, \mathcal{P} =$ GET-CONTROL-CONSEQS(id, $thres$)
    if $\mathcal{S}_{t+1} = \{s_{t+1}\}$ **then** $F(z) = s_{t+1}$, $\tau(z) = t$ // no stochastic effects
    else // stochastic effects
      $T(z) \leftarrow chance$
      for $s^i_{t+1} \in \mathcal{S}_{t+1}$ **do**
        $X \leftarrow X \cup \{k\}$, $E \leftarrow E \cup \{(z, k)\}$
        $p_z(z, k) \leftarrow \mathcal{P}\left(s^i_{t+1}\right)$
        $F(k) \leftarrow s^i_{t+1}$, $\tau(k) \leftarrow t$, $k++$
      $p \leftarrow p \cup \{p_z\}$
    $\mathbb{A} \leftarrow \mathbb{A} \setminus \mu$ // remove actions from the database
  $\gamma = (P, (X, E), T, W, \mathscr{A}, p)$
  return γ, F, τ

Algorithm 4: Function BUILD-FULL-GAME(*id*, *thres*, *max*).

Input : id ▷ string
$thres$ ▷ non-negative integer
max ▷ non-negative integer
Output : Γ ▷ extensive-form game
$\mathcal{F}$ ▷ set of fluents assigned to γ's state nodes
Data : $\mathbb{A}$ ▷ action situation description
Description : Given an action situation description, generate its extensive-form game semantics.

Function BUILD-GAME-ROUND(*id, thres, max*)**:**
  $\phi \leftarrow$ GET-SIMPLE-CONSEQS(id, boundary, $thres$) = {`participates`(ag_1), ...}
  $\mathbb{A} \leftarrow \mathbb{A} \cup \phi$
  $\rho \leftarrow$ GET-SIMPLE-CONSEQS(id, position, $thres$) = {`role`(ag_1, r_1), ...}
  $\mathbb{A} \leftarrow \mathbb{A} \cup \rho$
  $s_0 \leftarrow \{\}$
  foreach instantiation f_i of `?-initially(F)` **do** $s_0 \leftarrow s_0 \cup \{f_i\}$ `// initial facts`
  $P = \{ag_i\}_{\forall \texttt{participates}(ag_i) \in \phi}$
  $X \leftarrow \{1\}$, $x_0 \leftarrow 1$, $E \leftarrow \{\}$, $W \leftarrow \{\{\}, ..., \{\}\}_{\forall i \in P}$, $\mathscr{A} \leftarrow \{\}$, $p \leftarrow \{\}$
  $\mathcal{F}(1) \leftarrow s_0$
  $round(1) \leftarrow 0$
  $\mathcal{Q} \leftarrow$ QUEUE(1)
  while $\mathcal{Q}$ is not empty **do**
    $n \leftarrow \mathcal{Q}$.POP()
    if $round(n) \geq max$ **then continue**
    $s_t \leftarrow \mathcal{F}(n)$
    $\mathbb{A} \leftarrow \mathbb{A} \cup \{s_t\}$ `// assert node facts into database`
    if `?-terminal` returns `true` **then** $\mathbb{A} \leftarrow \mathbb{A} \setminus \{s_t\}$, **continue**
    $\gamma, F, \tau \leftarrow$ BUILD-GAME-ROUND(id, $thres$)
    $\mathbb{A} \leftarrow \mathbb{A} \setminus \{s_t\}$
    `/* append game round to overall game tree - superindex` γ `denotes the elements from the game round */`
    for $x \in X^\gamma$ **do** $x \leftarrow x + n - 1$ `// node re-labeling`
    $X \leftarrow X \cup X^\gamma$, $E \leftarrow E \cup E^\gamma$
    for $x \in X^\gamma \setminus Z^\gamma$ **do** $T(x) \leftarrow T^\gamma(x)$
    for $p \in P$ **do** $W_p \leftarrow W_p \cup W_p^\gamma$
    for $A(w) \in \mathscr{A}^\gamma$ **do** $\mathscr{A} \leftarrow \mathscr{A} \cup \{A(w)\}$
    forall $x \in X^\gamma \mid T(x) = chance$ **do** $p \leftarrow p \cup \{p_x^\gamma\}$
    for $z \in Z^\gamma$ **do** $\mathcal{F}(z) \leftarrow F(z)$
    forall $z \in Z^\gamma$ **do**
      $round(z) \leftarrow round(n) + 1$
      if $\tau(z) = 0$ **then** $\mathcal{Q}$.PUSH(z)
  $\Gamma = (P, (X, E), T, W, \mathscr{A}, p)$
  return Γ, $\mathcal{F}$

Appendix B. Example game semantics

B.1. Iterated Prisoner's Dilemma

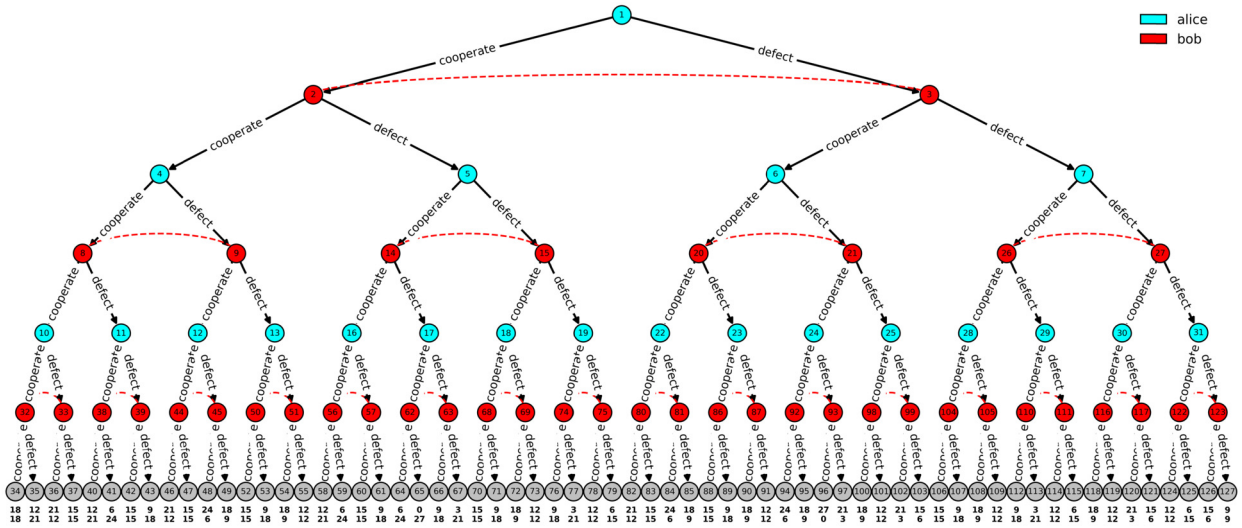


Fig. B.6. Game semantics for the iterated Prisoner's Dilemma with the default rule configuration.

Table B.4
Equilibrium strategies (top) and distribution over outcomes (bottom) for the default rule configuration of the iterated Prisoner's Dilemma game.

Agent	Information set	Action	Probability
alice	all	cooperate defect	0 1
bob	all	cooperate defect	0 1

Terminal node(s)	State fluents	Probability
127	`payoff(alice,9)`, `payoff(bob,9)`	1

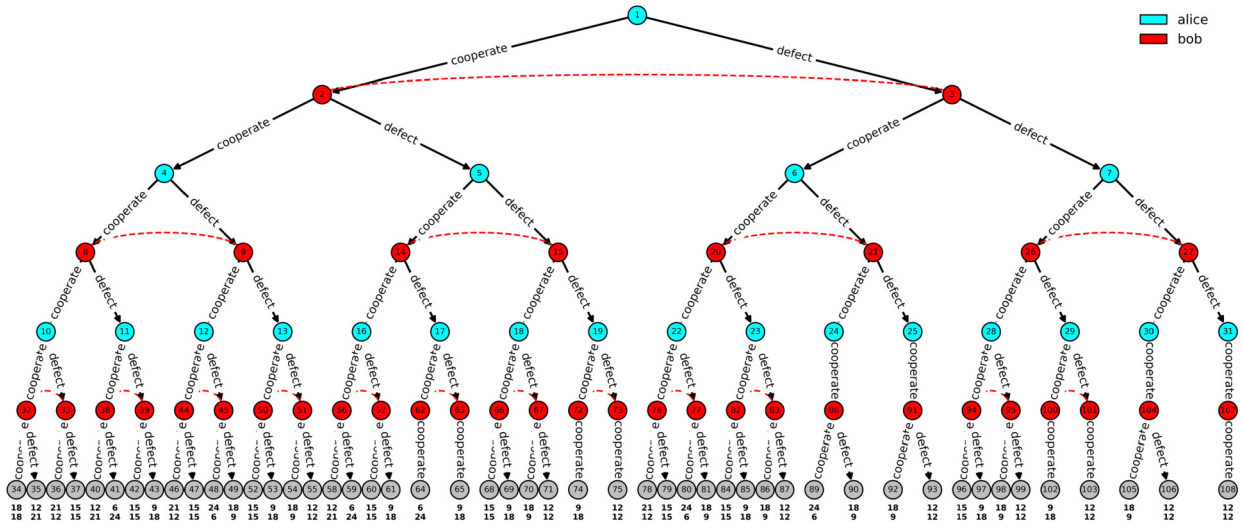


Fig. B.7. Game semantics for the iterated Prisoner's Dilemma with an additional rule for restricting to 2 the number of consecutive defections.

Table B.5
Equilibrium strategies (top) and distribution over outcomes (bottom) for the iterated Prisoner's Dilemma with an additional rule for restricting the number of consecutive defections. For the bottom table, only the state fluents that are common to all the outcomes with non-zero probability are shown.

Agent	Information set	Action	Probability
alice	{1}, {6}, {7}	cooperate	1/2
		defect	1/2
	{24}, {25}, {30}, {31}	cooperate	1
		defect	0
	all others	cooperate	0
		defect	1
bob	{2, 3}, {26, 27}, {14, 15}	cooperate	1/2
		defect	1/2
	{62, 63}, {72, 73}, {100, 101}, {107}	cooperate	1
		defect	0
	all others	cooperate	0
		defect	1

Terminal node(s)	State fluents	Probability
55		1
71, 75, 87, 93	`payoff(alice,12)`, `payoff(bob,12)`, ...	0.125
99, 103, 106, 108		0.0625

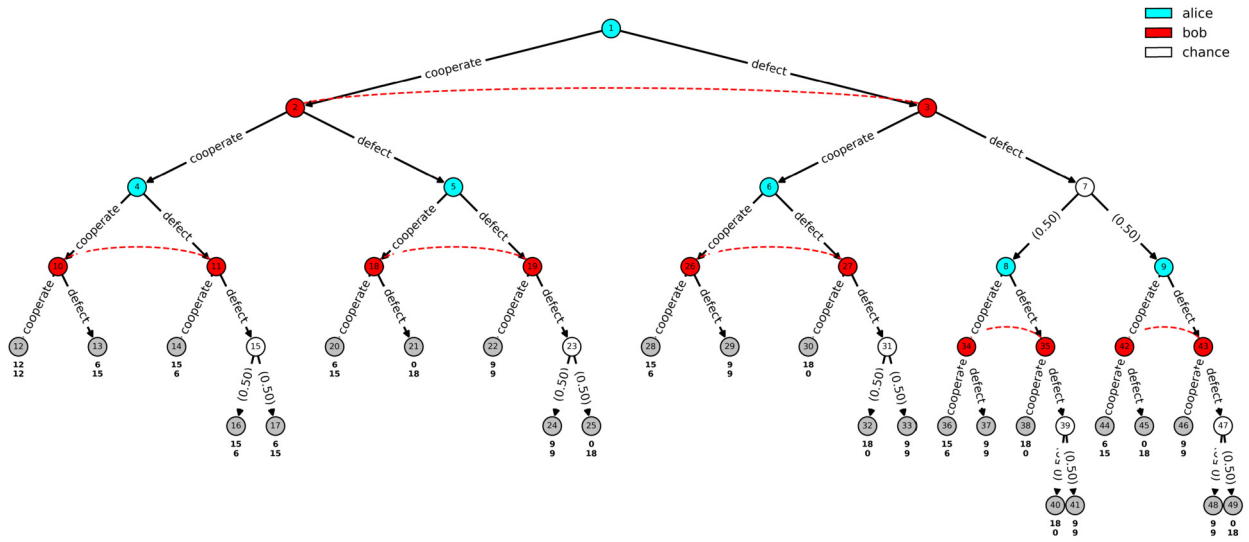


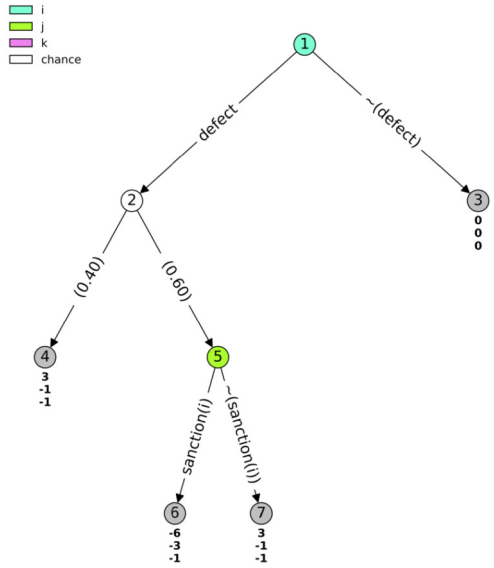
Fig. B.8. Game semantics for the iterated Prisoner's Dilemma with additional rules to change the outcome of a consecutive defection. For visualization purposes, termination conditions are met after agents play two game rounds instead of three.

Table B.6
Equilibrium strategies (top) and distribution over outcomes (bottom) for the iterated Prisoner's Dilemma with additional rules to change the outcome of a consecutive defection.

Agent	Information set	Action	Probability
alice	all	cooperate	0
		defect	1
bob	all	cooperate	0
		defect	1

Terminal node(s)	State fluents	Probability
40	`payoff(alice,18)`, `payoff(bob,0)`	0.25
41	`payoff(alice,9)`, `payoff(bob,9)`	0.25
48	`payoff(alice,9)`, `payoff(bob,9)`	0.25
49	`payoff(alice,0)`, `payoff(bob,18)`	0.25

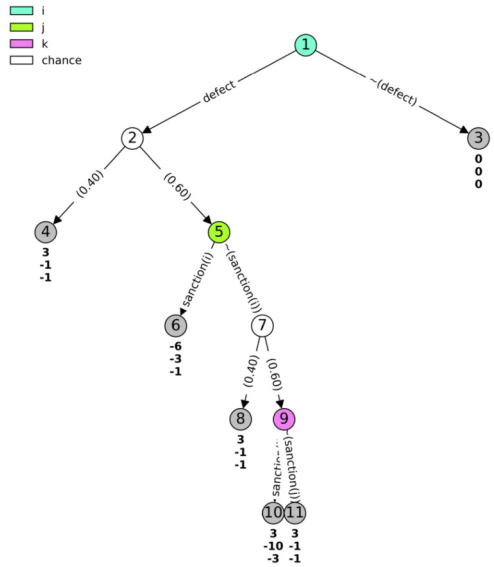
B.2. Axelrod's (meta)-norms game



(a)

	State fluents	p
1	payoff(i,0), payoff(j,0), payoff(k,0), time(0)	-
3	payoff(i,0), payoff(j,0), payoff(k,0), time(1)	0
4	payoff(i,3), payoff(j,-1), payoff(k,-1), time(1), (seen(j,i))	0.4
5	payoff(i,3), payoff(j,-1), payoff(k,-1), seen(j,i), time(1)	-
6	payoff(i,-6), payoff(j,-3), payoff(k,-1), time(2)	0
7	payoff(i,3), payoff(j,-1), payoff(k,-1), time(2)	0.6

(b)



(c)

	State fluents	p
1	payoff(i,0), payoff(j,0), payoff(k,0), time(0)	-
3	payoff(i,0), payoff(j,0), payoff(k,0), time(1)	0
4	payoff(i,3), payoff(j,-1), payoff(k,-1), time(1), (seen(j,i))	0.4
5	payoff(i,3), payoff(j,-1), payoff(k,-1), seen(j,i), time(1)	-
6	payoff(i,-6), payoff(j,-3), payoff(k,-1), time(2)	0
8	payoff(i,3), payoff(j,-1), payoff(k,-1), time(2), (seen(k,j))	0.24
9	payoff(i,3), payoff(j,-1), payoff(k,-1), seen(k,j), time(2)	-
10	payoff(i,3), payoff(j,-10), payoff(k,-3), time(3)	0
11	payoff(i,3), payoff(j,-1), payoff(k,-1), time(3)	0.36

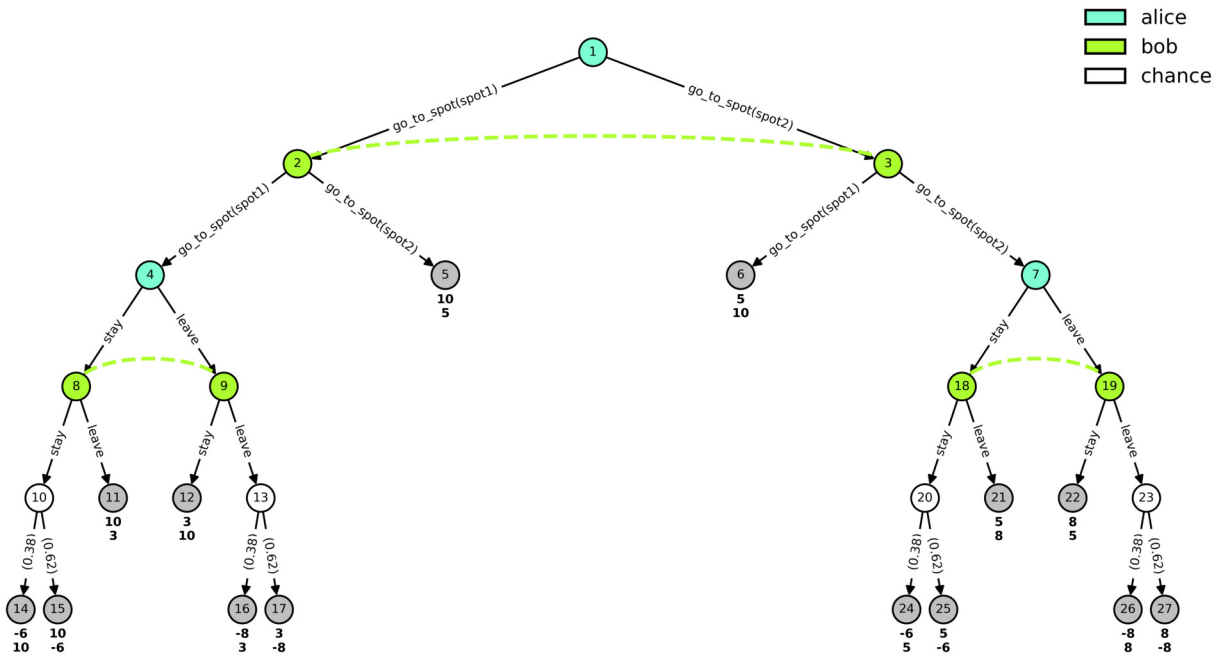
(d)

Fig. B.9. Semantics for Axelrod's norms (top) and metanorms (bottom) action situations, with the extensive game tree (left) and the corresponding state fluents (right). For the terminal nodes, their probability **p** induced by the equilibrium strategies and chance moves is also given.

Table B.7
Equilibrium strategies for Axelrod's norms and meta-norms games. For the norms game, only the first two rows apply (strategies for i and j). For the meta-norms game, all rows apply.

Agent	Information set	Action	Probability
i	{1}	defect ~defect	1 0
j	{5}	sanction(i) ~sanction(i)	0 1
k	{9}	sanction(j) ~sanction(j)	0 1

B.3. Ostrom's fishing game



(a)

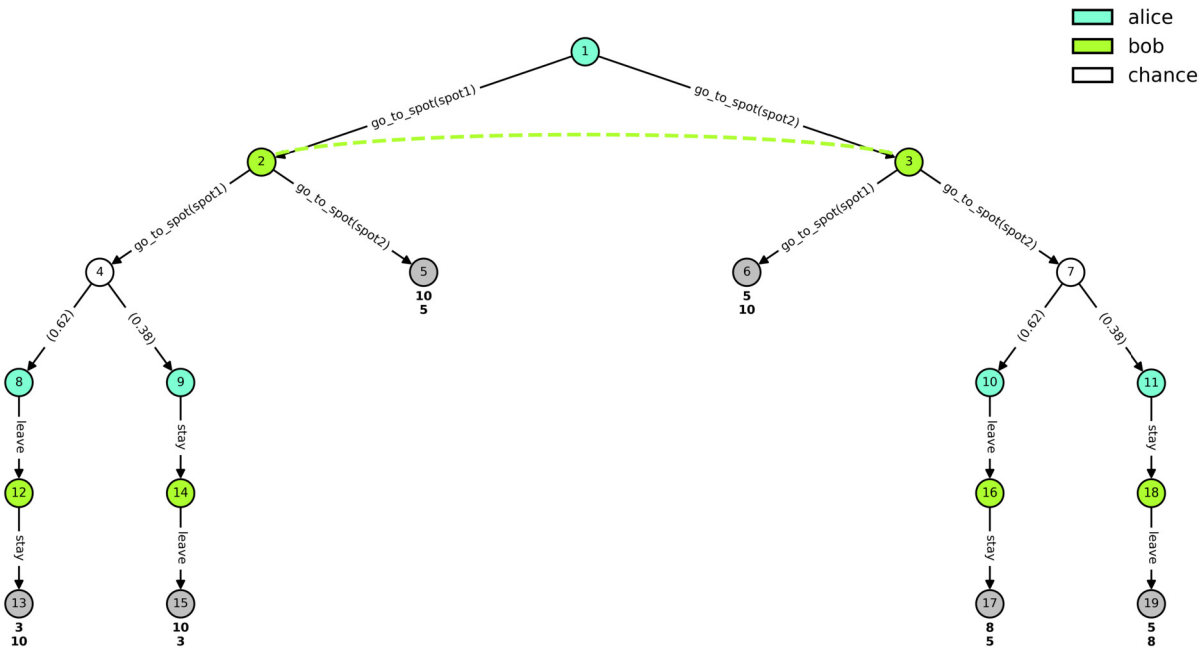
Agent	Information set	Action	Probability
alice	{1}	go_to_spot(spot1)	0.71
		go_to_spot(spot2)	0.29
	{4}	stay	0.82
		leave	0.18
	{7}	stay	0.41
		leave	0.59
bob	{2, 3}	go_to_spot(spot1)	0.84
		go_to_spot(spot2)	0.16
	{8, 9}	stay	1
		leave	0
	{18, 19}	stay	0.30
		leave	0.70

(b)

	State fluents	p		State fluents	p
1	at(alice, shore), at(bob, shore)	-	14	at(alice, spot1), at(bob, spot1), won_fight(bob)	0.18
4	at(alice, spot1), at(bob, spot1)	-	15	at(alice, spot1), at(bob, spot1), won_fight(alice)	0.31
5	at(alice, spot1), at(bob, spot2)	0.11	16	at(alice, spot2), at(bob, spot2), won_fight(bob)	0
6	at(alice, spot2), at(bob, spot1)	0.25	17	at(alice, spot2), at(bob, spot2), won_fight(alice)	0
7	at(alice, spot2), at(bob, spot2)	-	24	at(alice, spot2), at(bob, spot2), won_fight(bob)	0
11	at(alice, spot1), at(bob, spot2)	0	25	at(alice, spot2), at(bob, spot2), won_fight(alice)	0
12	at(alice, spot2), at(bob, spot1)	0.11	26	at(alice, spot1), at(bob, spot1), won_fight(bob)	0.01
21	at(alice, spot2), at(bob, spot1)	0.01	27	at(alice, spot1), at(bob, spot1), won_fight(alice)	0.01
22	at(alice, spot1), at(bob, spot2)	0.01			

(c)

Fig. B.10. Semantics for the fishers default action situation: game tree (a), equilibrium strategies (b) and state fluents with the probabilities over the terminal nodes induced by the equilibrium strategies and chance moves (c).



(a)

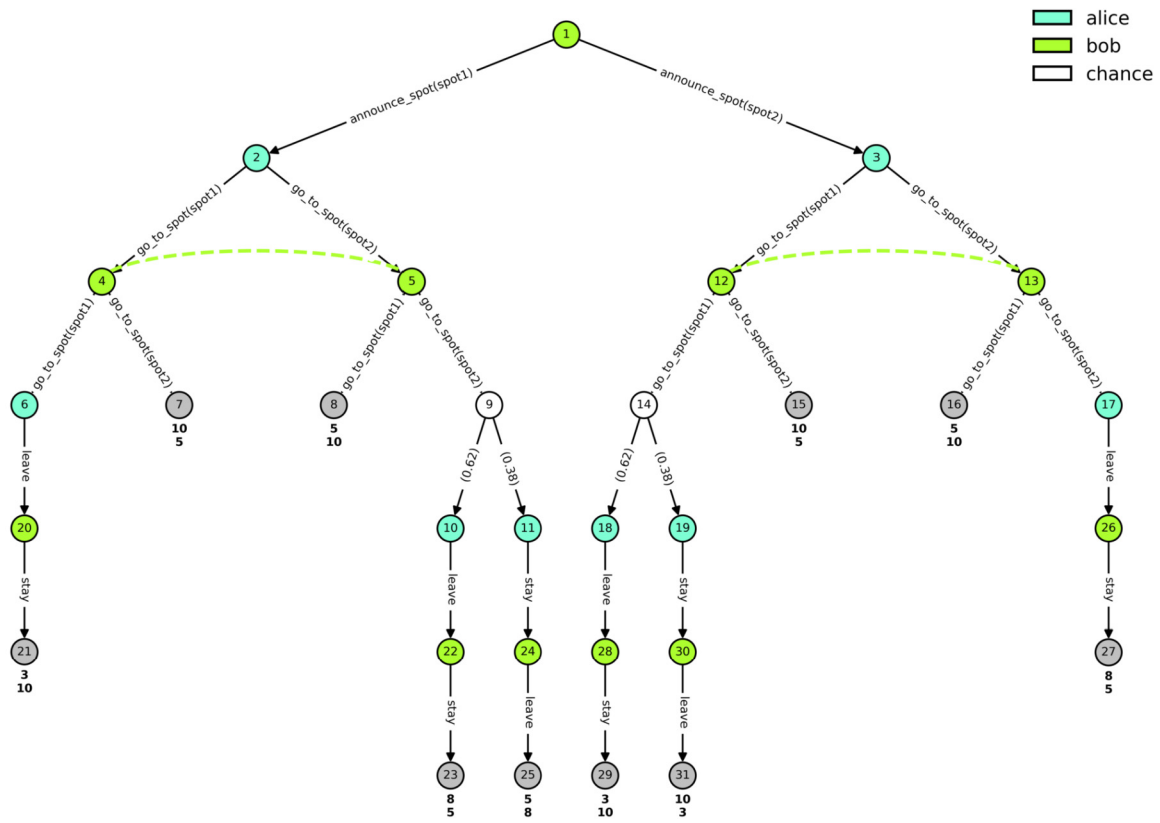
Agent	Information set	Action	Probability
alice	{1}	go_to_spot(spot1)	1
		go_to_spot(spot2)	0
	{8}	leave	1
	{9}	stay	1
bob	{2, 3}	go_to_spot(spot1)	1
		go_to_spot(spot2)	0
	{12}	stay	1
	{14}	leave	1

(b)

	State fluents	p
1	at(alice, shore), at(bob, shore)	-
5	at(alice, spot1), at(bob, spot2)	0
6	at(alice, spot2), at(bob, spot1)	0
8	at(alice, spot1), at(bob, spot1), won_race(bob)	-
9	at(alice, spot1), at(bob, spot1), won_race(alice)	-
10	at(alice, spot2), at(bob, spot2), won_race(bob)	-
11	at(alice, spot2), at(bob, spot2), won_race(alice)	-
13	at(alice, spot2), at(bob, spot1), won_race(bob)	0.62
15	at(alice, spot1), at(bob, spot2), won_race(alice)	0.38
17	at(alice, spot1), at(bob, spot2), won_race(bob)	0
19	at(alice, spot2), at(bob, spot1), won_race(alice)	0

(c)

Fig. B.11. Semantics for the fishers *first-in-time, first-in-right* action situation: game tree (a), equilibrium strategies (b) and state fluents with the probabilities over the terminal nodes induced by the equilibrium strategies and chance moves (c). The equilibrium strategies for information sets {10}, {11}, {16} and {18} have been omitted since they are not relevant for the game (i.e. agents never choose to go to spot 2).



(a)

Agent	Information set	Action	Probability
alice	{2}	go_to_spot(spot1) go_to_spot(spot2)	0 1
bob	{1}	announce_spot(spot1) announce_spot(spot2)	1 0
	{4, 5}	go_to_spot(spot1) go_to_spot(spot2)	1 0

(b)

	State fluents	p
1	at(alice, shore), at(bob, shore)	-
2	announced(bob, spot1), at(alice, shore), at(bob, shore)	-
3	announced(bob, spot2), at(alice, shore), at(bob, shore)	-
6	announced(bob, spot1), at(alice, spot1), at(bob, spot1), won_race(bob)	-
7	announced(bob, spot1), at(alice, spot1), at(bob, spot2)	0
8	announced(bob, spot1), at(alice, spot2), at(bob, spot1)	1.00
10	announced(bob, spot1), at(alice, spot2), at(bob, spot2), won_race(bob)	-
11	announced(bob, spot1), at(alice, spot2), at(bob, spot2), won_race(alice)	-
15	announced(bob, spot2), at(alice, spot1), at(bob, spot2)	0
16	announced(bob, spot2), at(alice, spot2), at(bob, spot1)	0
17	announced(bob, spot2), at(alice, spot2), at(bob, spot2), won_race(bob)	-
18	announced(bob, spot2), at(alice, spot1), at(bob, spot1), won_race(bob)	-
19	announced(bob, spot2), at(alice, spot1), at(bob, spot1), won_race(alice)	-
21	announced(bob, spot1), at(alice, spot2), at(bob, spot1), won_race(bob)	0
23	announced(bob, spot1), at(alice, spot1), at(bob, spot2), won_race(bob)	0
25	announced(bob, spot1), at(alice, spot2), at(bob, spot1), won_race(alice)	0
27	announced(bob, spot2), at(alice, spot1), at(bob, spot2), won_race(bob)	0
29	announced(bob, spot2), at(alice, spot2), at(bob, spot1), won_race(bob)	0
31	announced(bob, spot2), at(alice, spot1), at(bob, spot2), won_race(alice)	0

(c)

Fig. B.12. Semantics for the fishers *first-to-announce, first-in-right* action situation: game tree (a), equilibrium strategies (b) and state fluents with the probabilities induced over the terminal nodes by the equilibrium strategies and chance moves (c). Only the equilibrium strategies for the information sets that are actually visited during game play are included.

References

[1] E. Ostrom, Background on the Institutional Analysis and Development framework, Policy Stud. J. 39 (1) (2011) 7–27, https://doi.org/10.1111/j.1541-0072.2010.00394.x.
[2] E. Ostrom, Understanding Institutional Diversity, Princeton University Press, 2005.
[3] M. Black, Models and Metaphors: Studies in Language and Philosophy, Cornell University Press, Ithaca, NY, 1962.
[4] D. Cozort, J.M. Shields (Eds.), The Oxford Handbook of Buddhist Ethics, Oxford University Press, 2018.
[5] E. Ostrom, Governing the Commons, Cambridge University Press, 1990.
[6] J. Weymark, Social Welfare Functions, Oxford University Press, 2016, pp. 126–159, Ch. 5.
[7] L.L. Kiser, E. Ostrom, The Three Worlds of Action: A Metatheoretical Synthesis of Institutional Approaches, Michigan University Press, Ann Arbor, 1982, pp. 56–88, Ch. 2.
[8] S. Sarr, B. Hayes, D.A. DeCaro, Applying Ostrom's Institutional Analysis and Development framework, and design principles for co-production to pollution management in Louisville's Rubbertown, Kentucky, Land Use Policy 104 (2021) 105383, https://doi.org/10.1016/j.lusepol.2021.105383.
[9] T. Nguyen, T. Watanabe, Autonomous motivation for the successful implementation of waste management policy: an examination using an adapted Institutional Analysis and Development framework in Thua Thien Hue, Vietnam, Sustainability 12 (7) (2020) 2724, https://doi.org/10.3390/su12072724.
[10] D.N. Barton, K. Benavides, A. Chacon-Cascante, J.F. Le Coq, M.M. Quiros, I. Porras, E. Primmer, I. Ring, Payments for ecosystem services as a policy mix: demonstrating the Institutional Analysis and Development framework on conservation policy instruments, Environ. Policy Gov. 27 (5) (2017) 404–421, https://doi.org/10.1002/eet.1769.
[11] D.H. Cole, Laws, norms, and the Institutional Analysis and Development framework, J. Inst. Econ. 13 (4) (2017) 829–847, https://doi.org/10.1017/s1744137417000030.
[12] A. Mas-Colell, M.D. Whinston, J.R. Green, Microeconomic Theory, Oxford University Press, New York, 1995.
[13] Y. Shoham, M. Tennenholtz, On social laws for artificial agent societies: off-line design, Artif. Intell. 73 (1–2) (1995) 231–252, https://doi.org/10.1016/0004-3702(94)00007-n.
[14] S. Onn, M. Tennenholtz, Determination of social laws for multi-agent mobilization, Artif. Intell. 95 (1) (1997) 155–167, https://doi.org/10.1016/s0004-3702(97)00045-3.
[15] G. Andrighetto, G. Governatori, P. Noriega, L. van der Torre, Normative multi-agent systems (Dagstuhl seminar 12111), Dagstuhl Rep. 2 (3) (2012) 23–49, https://doi.org/10.4230/DagRep.2.3.23, http://drops.dagstuhl.de/opus/volltexte/2012/3535.
[16] C. Hahn, T. Phan, S. Feld, C. Roch, F. Ritz, A. Sedlmeier, T. Gabor, C. Linnhoff-Popien, Nash equilibria in multi-agent swarms, in: Proceedings of the 12th International Conference on Agents and Artificial Intelligence, SCITEPRESS - Science and Technology Publications, 2020, pp. 234–241.
[17] P. Caillou, S. Aknine, S. Pinson, Searching Pareto optimal solutions for the problem of forming and restructuring coalitions in multi-agent systems, Group Decis. Negot. 19 (1) (2009) 7–37, https://doi.org/10.1007/s10726-009-9183-9.
[18] S.E.S. Crawford, E. Ostrom, A grammar of institutions, Am. Polit. Sci. Rev. 89 (3) (1995) 582–600, https://doi.org/10.2307/2082975.
[19] C. Frantz, M.K. Purvis, M. Nowostawski, B.T.R. Savarimuthu, nADICO: a nested grammar of institutions, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 429–436.
[20] C.K. Frantz, S. Siddiki, Institutional Grammar 2.0: A Specification for Encoding and Analyzing Institutional Design, Public Administration, 2021.
[21] A. Ghorbani, G. Bravo, Managing the commons: a simple model of the emergence of institutions through collective action, Int. J. Commons 10 (1) (2016) 200–219, https://doi.org/10.18352/ijc.606.
[22] A. Smajgl, L.R. Izquierdo, M. Huigne, Modeling endogenous rule changes in an institutional context: the adico sequence, Adv. Complex Syst. 11 (02) (2008) 199–215, https://doi.org/10.1142/s021952590800157x.
[23] A. Ghorbani, P. Bots, V. Dignum, G. Dijkema, MAIA: a framework for developing agent-based social simulations, J. Artif. Soc. Soc. Simul. 16 (2) (2013), https://doi.org/10.18564/jasss.2166.
[24] M. Genesereth, N. Love, B. Pell, General game playing: overview of the AAAI competition, AI Mag. 26 (2005) 62–72, https://doi.org/10.1609/aimag.v26i2.1813.
[25] S. Schiffel, M. Thielscher, Representing and reasoning about the rules of general games with imperfect information, J. Artif. Intell. Res. 49 (2014) 171–206, https://doi.org/10.1613/jair.4115.
[26] M. Thielscher, GDL-III: a description language for epistemic general game playing, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, 2017, pp. 1276–1282.
[27] D. de Jonge, T. Trescak, C. Sierra, S. Simoff, R.L. de Mántaras, Using game description language for mediated dispute resolution, AI Soc. 34 (4) (2017) 767–784, https://doi.org/10.1007/s00146-017-0790-8.
[28] D. de Jonge, D. Zhang, GDL as a unifying domain description language for declarative automated negotiation, Auton. Agents Multi-Agent Syst. 35 (1) (2021), https://doi.org/10.1007/s10458-020-09491-6.
[29] R.B. Scherl, H.J. Levesque, Knowledge, action, and the frame problem, Artif. Intell. 144 (1–2) (2003) 1–39, https://doi.org/10.1016/s0004-3702(02)00365-x.
[30] D. Koller, A. Pfeffer, Representations and solutions for game-theoretic problems, Artif. Intell. 94 (1–2) (1997) 167–215, https://doi.org/10.1016/s0004-3702(97)00023-4.
[31] G.H. von Wright, Deontic logic, Mind 60 (237) (1951) 1–15, http://www.jstor.org/stable/2251395.
[32] M. Belzer, Deontic logic, in: Routledge Encyclopedia of Philosophy, Routledge, 1998.
[33] J. Morales, M. Lopez-Sanchez, J.A. Rodriguez-Aguilar, M. Wooldridge, W. Vasconcelos, Automated synthesis of normative systems, in: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2013, pp. 483–490.
[34] M.S. Fagundes, S. Ossowski, J. Cerquides, P. Noriega, Design and evaluation of norm-aware agents based on normative Markov decision processes, Int. J. Approx. Reason. 78 (2016) 33–61, https://doi.org/10.1016/j.ijar.2016.06.005.
[35] J. Szabo, J.M. Such, N. Criado, Understanding the role of values and norms in practical reasoning, in: N. Bassiliades, G. Chalkiadakis, D. de Jonge (Eds.), Multi-Agent Systems and Agreement Technologies, Springer International Publishing, Cham, 2020, pp. 431–439.
[36] D. Grossi, D. Gabbay, L. van der Torre, The norm implementation problem in normative multi-agent systems, in: Specification and Verification of Multi-agent Systems, Springer US, 2010, pp. 195–224.
[37] F. Lin, Situation Calculus, Foundations of Artificial Intelligence, vol. 3, Elsevier, 2008, pp. 649–669, Ch. 16.
[38] J. González-Díaz, I. García-Jurado, M.G. Fiestras-Janeiro, An Introductory Course on Mathematical Game Theory, American Mathematical Society and Real Sociedad Matemática Española, Providence, Rhode Island, USA and Madrid, 2010.
[39] S. Fatima, S. Kraus, M. Wooldridge, Principles of Automated Negotiation, Cambridge University Press, 2009.
[40] N. Gronewold, Game theory: climate talks destined to fail (Dec 2010), https://www.scientificamerican.com/article/game-theorist-predicts-failure-at-climate-talks/.
[41] H.W. Kuhn, 11. Extensive games and the problem of information, in: Contributions to the Theory of Games (AM-28), Vol. II, Princeton University Press, 1953, pp. 193–216.

[42] J.F. Nash, Equilibrium points in n-person games, Proc. Natl. Acad. Sci. USA 36 (1) (1950) 48–49, http://www.jstor.org/stable/88031.
[43] R.J. Aumann, Subjectivity and correlation in randomized strategies, J. Math. Econ. 1 (1) (1974) 67–96, https://doi.org/10.1016/0304-4068(74)90037-8.
[44] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, 2014.
[45] L. Hammond, J. Fox, T. Everitt, A. Abate, M. Wooldridge, Equilibrium refinements for multi-agent influence diagrams: theory and practice, in: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2021, pp. 574–582.
[46] R. Axelrod, An evolutionary approach to norms, Am. Polit. Sci. Rev. 80 (04) (1986) 1095–1111, https://doi.org/10.2307/1960858.
[47] E. Ostrom, R. Gardner, J. Walker, Rules, Games, and Common-Pool Resources, University of Michigan Press, 1994.

Contribution 3

Combining Theory of Mind and Abductive Reasoning in Agent-Oriented Programming

Autonomous Agents and Multi-Agent Systems

Full citation:

Montes, N., Luck, M., Osman, N., Rodrigues, O., & Sierra, C. (2023a). Combining theory of mind and abductive reasoning in agent-oriented programming. *Autonomous Agents and Multi-Agent Systems*, *37*(2). https://doi.org/10.1007/s10458-023-09613-w

Autonomous Agents and Multi-Agent Systems (2023) 37:36
https://doi.org/10.1007/s10458-023-09613-w

Combining theory of mind and abductive reasoning in agent-oriented programming

Nieves Montes[1] · Michael Luck[2] · Nardine Osman[1] · Odinaldo Rodrigues[2] · Carles Sierra[1]

Accepted: 6 July 2023

Abstract
This paper presents a novel model, called TomAbd, that endows autonomous agents with Theory of Mind capabilities. TomAbd agents are able to simulate the perspective of the world that their peers have and reason from their perspective. Furthermore, TomAbd agents can reason from the perspective of others down to an *arbitrary level of recursion*, using Theory of Mind of n^{th} order. By combining the previous capability with abductive reasoning, TomAbd agents can infer the beliefs that others were relying upon to select their actions, hence putting them in a more informed position when it comes to their own decision-making. We have tested the TomAbd model in the challenging domain of Hanabi, a game characterised by cooperation and imperfect information. Our results show that the abilities granted by the TomAbd model boost the performance of the team along a variety of metrics, including final score, efficiency of communication, and uncertainty reduction.

Keywords Theory of mind · Abductive reasoning · Agent-oriented programming · Social AI · Hanabi

N. Montes: Part of this work was done when the author was on a research visit to the Department of Informatics at King's College London.

✉ Nieves Montes
nmontes@iiia.csic.es

Michael Luck
michael.luck@kcl.ac.uk

Nardine Osman
nardine@iiia.csic.es

Odinaldo Rodrigues
odinaldo.rodrigues@kcl.ac.uk

Carles Sierra
sierra@iiia.csic.es

[1] Department of Multi-Agent Systems, Artificial Intelligence Research Institute (IIIA-CSIC), Campus de la UAB, 08193 Bellaterra, Barcelona, Spain

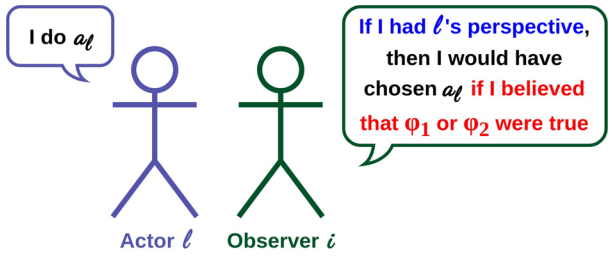
[2] Department of Informatics, King's College London, Bush House, London WC2B 4BG, UK

Published online: 11 August 2023

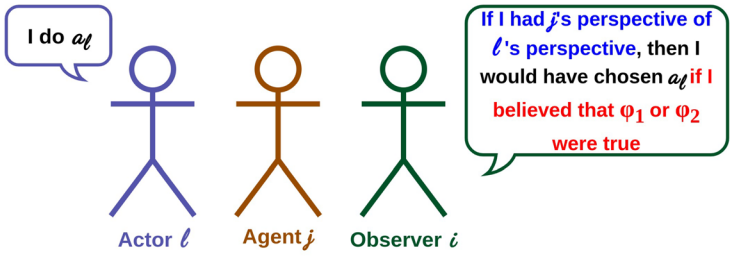
1 Introduction

The emergent field of social AI deals with the formulation and implementation of autonomous agents that can successfully act as part of a larger society, made up of other software agents as well as humans [1, 2]. In human social life, an essential requirement for effective participation is the ability to interpret and predict the behaviour of others in terms of their mental states, such as their beliefs, goals and desires. This ability to put oneself in the position of others and reason from their perspective is called Theory of Mind (ToM) and is closely related to feelings of empathy [3] and moral judgements [4].

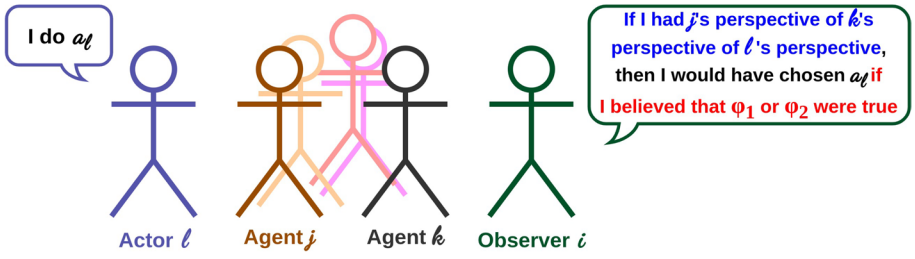
The work presented here starts from the assumption that, just as humans need a functioning ToM, if autonomous software agents are to operate satisfactorily in social contexts, they also need some implementation of the abilities that ToM endows humans with [5]. In particular, in domains where agents have to deal with partial observability, agents can benefit by engaging in the type of reasoning pictured in Fig. 1: agents can infer additional



(a) First-order ToM.



(b) Second-order ToM.



(c) Third-order ToM.

Fig. 1 Outline of the reasoning process captured by the TomAbd agent model

knowledge from observing the actions performed by others and deducing the beliefs that their peers were relying upon to select those actions. This process can be achieved directly as in Fig. 1a, where an observer adopts the perspective of an actor to provide an explanation for their action, or through one (Fig. 1b) or more (Fig. 1c) intermediaries, where the observer adopts the perspective of the actor through an arbitrary number of agents. Hence, agents can use other agents as "sensors" with the purpose of being in a more informed position when it comes to their own decision-making. The backward inference from observations (actions by others) to their underlying motivations is called *abductive reasoning* and, together with ToM, is a central component of the agent model presented here.

The main contribution of this work is the TomAbd agent model, which combines the two capacities mentioned above (Theory of Mind and abduction) to provide the reasoning displayed in Fig. 1. This paper builds on a previous, much-reduced, preliminary version [6]. Here, we propose a completely domain-independent model where agents observe the actions of others, adopt their perspective and generate explanations that justify their choice of action. We cover all the steps involved in this reasoning process: from the switch from the agent's perspective to that of a peer's, to the generation, post-processing and update of previous explanations as the state of the system evolves. In addition, we also provide a complementary decision-making function that takes into account the gathered abductive explanations.

We implement the TomAbd agent model in Jason [7], an agent-oriented programming language based on the Belief- Desire-Intention (BDI) architecture. Given the functionalities of our model, the ToM capabilities of TomAbd agents are strongly skewed towards the perception step of the BDI reasoning cycle (i.e. upon observation of an action by another agent, generate a plausible explanation for it). Nonetheless, we open up an avenue to introduce ToM reasoning into the deliberation step of the BDI cycle as well through a complementary decision-making function.

Furthermore, we have applied the TomAbd agent model to Hanabi, a cooperative card game that we use as our benchmark. We clearly indicate the specific domain-dependent choices necessary in this application, that need not be shared for other domains. We analyse the model's performance on a number of metrics, namely absolute team score and information gain and value. Our assessment quantifies the gains that can be unequivocally attributed to the ToM abilities of the agents.

This paper is organised as follows. In Sect. 2 we provide the necessary background on Theory of Mind, abductive reasoning and the Hanabi game. The central contribution of this paper, the TomAbd agent model, is exposed in detail in Sect. 3. Then, in Sect. 4 we cover some issues related to the implementation and potential customisations of the model components. Section 5 presents the performance results of the TomAbd agent model applied to the Hanabi domain. Finally, Sect. 6 compares our work with related approaches, and we conclude in Sect. 7.

2 Background

2.1 Theory of mind

The first building block of the TomAbd agent model is Theory of Mind (ToM). Broadly defined, ToM is the human cognitive ability to perceive, understand and interpret others in terms of their mental attitudes, such as their beliefs, emotions, desires and intentions [8].

Humans routinely interpret the behaviour of others in terms of their mental states, and this ability is considered essential for language and participation in social life [3].

ToM is not an innate ability. It is an empirically established fact that children develop a ToM at around the age of 4 [9]. It has been demonstrated that around this age, children are able to assign false beliefs to others, by having them undertake the Sally-Anne test [10]. The child is told the following story, accompanied by dolls or puppets: Sally puts her ball in a basket and goes out to play; while she is outside Anne takes the ball from the basket and puts it in a box; then Sally comes back in. The child is asked where will Sally look for her ball. Children with a developed ToM are able to identify that Sally will look for her ball inside the basket, thus correctly assigning a false belief to the character, that they themselves know to be untrue.

During the 1980s, the ToM hypothesis of autism gained traction, which states that deficits in the development of ToM satisfactorily explain the main symptoms of autism. This hypothesis argues that the inability to process mental states leads to a lack of reciprocity in social interactions [10]. Although a deficiency in the identification and interpretation of mental states remains uncontested as a cause of autism, it is no longer viewed as the only one, and the disorder is now studied as a complex condition involving a variety of cognitive mechanisms [11, 12].

Within philosophy and psychology, two distinct accounts of ToM exist: Theory ToM (TT) and Simulation ToM (ST) [13]. The TT account views the cognitive abilities assigned to ToM as the consequence of a theory-like body of implicit knowledge. This knowledge is conceived as a set of general rules and laws concerned with the deployment of mental concepts, analogous to a theory of folk psychology. This theory is applied inferentially to attribute beliefs, goals, and other mental states and predict subsequent actions.

In contrast, the ST account views the predictions of ToM not as a result of inference, but through the use of one's own cognitive processes and mechanisms to build a model of the minds of others and the processes happening therein. Hence, to attribute mental states and predict the actions of others, one imagines oneself as being in the other agent's position. Once there, humans apply their own cognitive processes, engaging in a sort of simulation of the minds of others. This internal simulation is very closely related to empathy, since it essentially consists of experiencing the world from the perspective of someone else. In this work, we adhere more closely to the ST account than to the TT one, as we view the former as having a clearer path to becoming operational. In our TomAbd model, agents simulate themselves to be in the position of another, and then apply abductive reasoning (covered in Sect. 2.2) to infer their beliefs.

Formally, ToM statements can be expressed using the language of epistemic logic, which studies the logical properties of knowledge, belief, and related concepts [14, 15]. The belief of agent i is expressed using modal operator B_i. Although modal operators also exist for other mental states such as desires and intentions [16], we focus on B, since the ToM abilities of our agent model are manifested by having the agent's own beliefs replaced by an estimation of the beliefs of others. Then, the statement $B_i\phi$ is read as "agent i believes that ϕ".

ToM statements can be expressed by nesting the previous beliefs about the state of the world. Therefore, statement $B_iB_j\phi$ is read as "i believes that j believes ϕ". This corresponds to a *first-order* ToM statement from the perspective of i. Subsequent nesting results in statements of higher order. For example, $B_iB_jB_k\phi$ is read as "i believes that j believes that k believes ϕ", a *second-order* ToM statement. This recursion can be extended down to an arbitrary nesting level. In general, an n-th order ToM statement is expressed as $B_iB_{j_1} \dots B_{j_{n-1}}B_{j_n}\phi$ and is read as "i believes that j_1 believes $\cdots$ that j_{n-1} believes that

j_n believes ϕ". The psychologist Corballis argued that, in fact, the ability to think recursively beyond the first nesting level, as in ToM statements of second order and beyond, is a uniquely human capacity that sets us apart from all other species [17, 18].

Within AI, implementations of ToM are often categorised under the umbrella of techniques for *modelling others* [19]. In the majority of cases, these techniques are applied to competitive domains, where they are referred to as *opponent modelling* [20, 21]. ToM for autonomous software agents has so far been developed in a somewhat fragmented fashion, with every camp within the field implementing it according to their own techniques and methods.

In machine learning, prominent work by Rabinowitz et al. [22] has modelled ToM as a meta-learning process, where an architecture composed of several deep neural networks (DNN) is trained on past trajectories of a variety of agents, including random, reinforcement learning (RL) and goal-directed agents, to predict action at the next time-step. The component of the architecture most related to ToM is the *mental net*, which parses trajectory observations into a generic mental state embedding. It is not specified what kind of mental states (i.e. beliefs or goals) these embeddings represent. In contrast, Wang et al. [23] also use an architecture based on DNNs for reaching consensus in multi-agent cooperative settings. Their *ToM net* explicitly estimates the goal that others are currently pursuing based on local observations. Finally, an alternative approach by Jara-Ettinger [24] proposes to formalise the acquisition of a ToM as an inverse reinforcement learning (IRL) problem. However, these approaches have drawn some criticism for their inability to mimic the actual operation of the human mind, as the direct mapping from past to future behaviour bypasses the modelling of relevant mental attitudes, such as desires and emotions [25]. By contrast, in our work ToM is used to derive explicit beliefs. We leave the expansion of the model to include other mental states, such as desires and intentions, for future work.

ToM approaches have also been investigated from an analytical game theoretical perspective. De Weerd et al. [26, 27] show that the marginal benefits of employing ToM diminish with the nesting level in competitive scenarios. In particular, while first-order and second-order ToM present a clear advantage with respect to opponents with ToM abilities of lower order (or no ToM capacity at all), the benefits of using higher-order ToM are outweighed by the complexity it entails. The same authors also prove that high-order ToM is beneficial in dynamic environments, with the magnitude of the benefits increasing with the uncertainty of the scenario [28]. It is therefore important to devise techniques that attempt to measure the information gained through the addition of ToM of any order, a concern also considered in this paper.

Finally, symbolic approaches to ToM have studied the effects of announcements on the beliefs of others and the ripple-down effects on their desires and the actions they motivate in response, for the purposes of deception and manipulation [29, 30].

2.2 Abductive logic programming

The second main component of the ToMAbd agent model is abductive reasoning. Abduction is a logical inference paradigm that differs from traditional deductive reasoning [31]. Classical deduction makes inference following the *modus ponens* rule: from knowledge of ϕ and of the implication $\phi \rightarrow \psi$, ψ is inferred as true. In contrast, abduction makes inferences in the opposite direction: from knowledge of the implication $\phi \rightarrow \psi$ and the *observation* of ψ, ϕ is inferred as a possible *explanation* for ψ.

Hence, instead of inferring conclusions deductively, abduction is concerned with the derivation of hypothesis that can satisfactorily explain an observed phenomenon. For this reason, abduction is broadly defined as "inference to the best explanation" [32], where the notion of *best* needs to be specified by some domain-dependent optimality criterion. Abduction is also distinct from the inference paradigm of *inductive reasoning* [33]. While induction works on a body of observations to derive a general principle, explanations inferred in abductive reasoning consist of *extensional* knowledge, i.e. knowledge that only applies to the domain under examination.

In the context of logic programming, the implementation of abductive reasoning is called Abductive Logic Programming (ALP) [34, 35], defined as follows.

Definition 1 An *Abductive Logic Programming theory* is a tuple $\langle T, A, IC \rangle$, where:

- T is a *logic program* representing expert knowledge in the domain;
- A is a set of ground *abducibles* (which are often defined by their predicate symbol), with the restriction that no element in A appears as the head of a clause in T; and
- IC is a set of *integrity constraints*, i.e. a set of formulas that cannot be violated.

Then, an abductive explanation is defined as follows.

Definition 2 Given an ALP theory $\langle T, A, IC \rangle$ and an observation Q, an *abductive explanation* Δ for Q is a subset of abducibles $\Delta \subseteq A$ such that:

- $T \cup \Delta \models Q$; and
- $T \cup \Delta$ verifies IC.

The verification mentioned in Definition 2 can take one of two views [34]. First, the stronger *entailment* view states that the extension of T with explanation Δ needs to derive the set of constraints, $T \cup \Delta \models IC$. Second, the weaker *consistency* view states that it is enough for the extended logic program not to violate IC, i.e. $T \cup \Delta \cup IC$ is satisfiable, or $T \cup \Delta \not\models \neg IC$. In this work, we adhere to the latter view. We do not model integrity constraints directly but rather their negation. We introduce into the agent program formulas that should *never* hold true through special rules called *impossibility clauses*. More details on this are provided in Sect. 3.1. Taking the consistency position allows us to work with incomplete abductive explanations that need not complement the current knowledge base to the extent that IC can be derived, but that nonetheless provide valuable information.

In practice, most existing ALP frameworks compute abductive explanations using some extension of classical Selective Linear Definite (SLD) clause resolution, or its negation-as-failure counterpart SLDNF [36–39]. The current state-of-the-art integrates abduction in Probabilistic Logic Programming (PLP), where the optimal explanation is considered to be the one that is compatible with the constraints and simultaneously maximises the joint probability of the query and the constraints [40].

The purpose of computing abductive explanations is to expand an existing knowledge base KB, which may or may not correspond to the logic program T used to compute explanation Δ in the first place. During knowledge expansion, which occurs one formula at a time, the following four scenarios may arise [34].

1. The new information can already be derived from the existing explanation, $KB \cup \Delta \equiv KB$, and hence Δ is *uninformative*.
2. KB can be split into two disjoint parts, $KB = KB_1 \cup KB_2$, such that one of them, together with the new information, implies the second, $KB_1 \cup \Delta \models KB_2$. In the worst case, the addition of Δ renders a part of the original knowledge base redundant.
3. The new information Δ violates the logical consistency of KB. To integrate the two, it is necessary to modify and/or reject a number of the assumptions in KB or in Δ that lead to the inconsistency.
4. Δ is independent and compatible with KB. This is the most desirable case, as Δ can be assimilated into KB in a straightforward manner.

In the TomAbd agent model, we deal with scenarios 1 and 3 through the post-processing of the generated abductive explanations by the explanation revision function (ERF). Essentially, uninformative explanations (scenario 1) as well as explanations that violate the integrity of the current belief base (scenario 3) are discarded. More details are provided in Sect. 3.3.

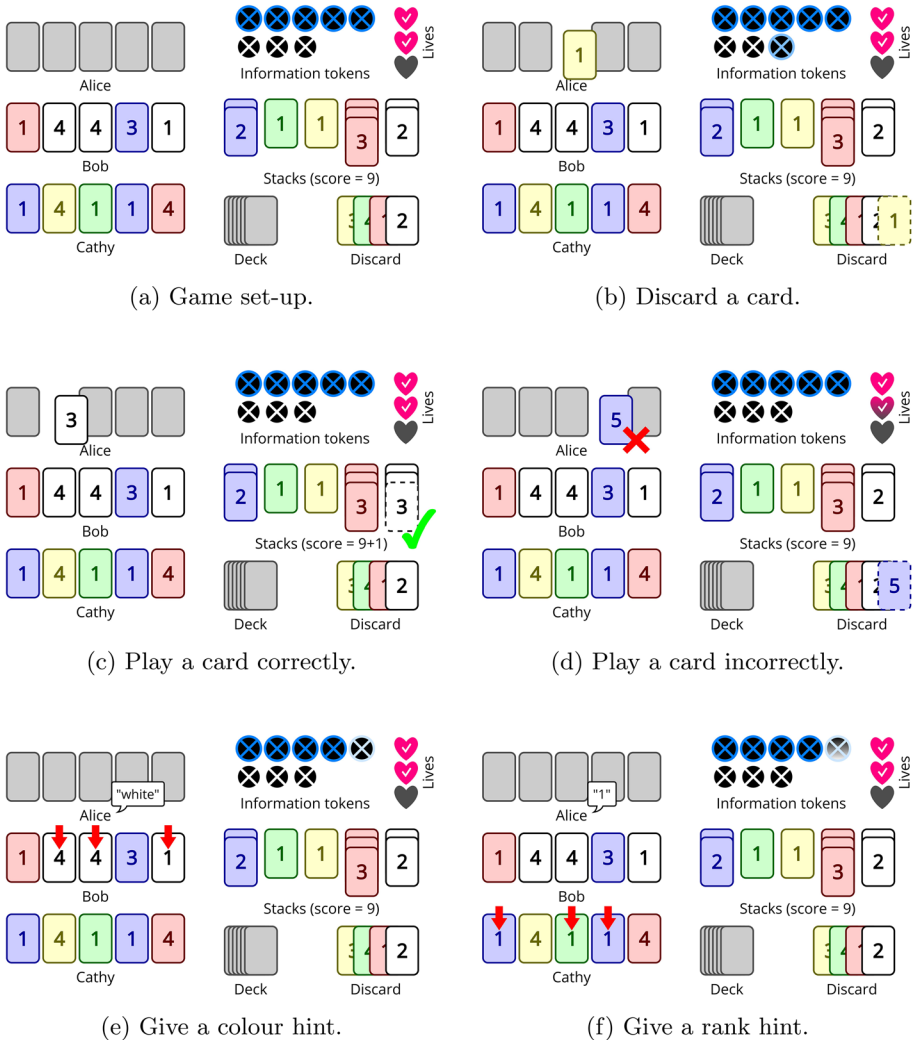
Hence, the addition of abductive explanations does not affect the correctness of KB, but it may affect its efficiency. The addition of δ into the knowledge base may subsume some information already there, as anticipated by scenario 2. However, in the TomAbd model, we do not check whether a new explanation renders part of the knowledge base redundant. We work with *dynamic* belief bases, which change as agents update their perceptions of the environment. When the system evolves and an agent's perception of it changes, some abductive explanations currently in the belief base need to be dropped because they are no longer correct, or they are now redundant. This operation is performed by the explanation update function (EUF), covered in Sect. 3.3. If, due to the addition of δ, a part of the belief base had been discarded, it would raise the issue of whether it needs to be recovered once the explanation that caused it to become irrelevant is dropped. We bypass this question by retaining all of the belief base upon adding an explanation, provided that this explanation has previously passed all the redundancy and consistency checks.

2.3 The Hanabi game

In this paper, we use the Hanabi game as a running example for the presentation of the TomAbd agent model and to evaluate its performance. Hanabi has been by other AI researchers as a testbed to test techniques for multi-agent cooperation [41, 42]. Hanabi is an award-winning[1] card game, where a team of two to five players work together towards a common objective. The goal of the team is to build stacks of cards of five different colours (blue, green, yellow, red and white), with the stacks composed of a card of rank 1, followed by a card of rank 2, and so on, until the stack is completed with a card of rank 5. A typical setup of an ongoing Hanabi game appears in Fig. 2a.

At the start of the game, players are handed four or five cards, depending on the size of the team. Players place their cards in a way such that everyone except themselves can see them. For example, the setup in Fig. 2a is drawn from the perspective of player Alice, who cannot see her own cards but has access to Bob's and Cathy's cards. Initially, no stack has any card on it (their size is 0). Additionally, eight information tokens (the round blue and

[1] https://www.spiel-des-jahres.de/en/games/hanabi/.



(a) Game set-up. (b) Discard a card.

(c) Play a card correctly. (d) Play a card incorrectly.

(e) Give a colour hint. (f) Give a rank hint.

Fig. 2 Basic set-up for the Hanabi game and actions that can be performed

black chips in Fig. 2) and three live tokens (the heart-shaped chips in Fig. 2) are placed on the table.

Players take turns in order, one at a time, in which they must perform one of three actions. First, they can *discard* a card (Fig. 2b). Here, the player picks a card from their hand and places it in the discard pile, which is observable by everyone. By doing so, they recover one spent information token (which is spent by giving hints) and replace the vacant slot in their hand with a card drawn from the deck. A player cannot discard a card if there are no information tokens to recover.

Second, players can *play* cards from their hand. They pick a card and place it on the stack of the corresponding colour. Players need not state in which stack they are going to play their card before they do so. In other words, they are allowed to play "blindly". There are two possible outcomes to this move. The card is correctly played if its rank is exactly 1

unit over the size of the stack of the card's colour. For example, in Fig. 2c Alice plays her white 3 card on the white stack, which has size 2 (i.e. there is a white 1 card at the bottom and a white 2 card in top of it). After a card is played correctly, the team score is increased by 1 unit (the score corresponds to the sum of the ranks at the top of each stack). Moreover, if a player correctly places a card of rank 5 and therefore completes one stack, one information token is recovered for the team, assuming there are some tokens left to recover. Finally, the player replaces the gap in their hand with a card from the deck.

The card is incorrectly played if the rank does not match the size of the stack plus 1. For example, in Fig. 2d Alice attempts to play a blue 5 while the blue stack has size of 2. If this happens, the player places the card they attempted to play in the discard pile, and replaces it with a new card from the deck. Furthermore, the whole team loses one of their life tokens.

Third, players can *give hints* to one another about the cards they hold. Hints are publicly announced, i.e. everyone hears them. Players can hint to one another about the colour (Fig. 2e) or rank of their cards (Fig. 2f). In order to give a hint, the moving player must spend one information token. The team must have at least one information token, which is spent when the hint is given. When players give hints to others, they must indicate all of the receiver's cards that match the colour or rank being hinted. For example, in Fig. 2e, Alice has to tell Bob where all of his white cards are. Alice is not allowed to tell Bob only the colour of a card in a single slot if he has other cards of the same colour. Analogously, in Fig. 2f, Alice tells Cathy which of her cards have rank 1, not mentioning their colour, regardless of any previous hints.

There are three possible ways in which a game of Hanabi might end. First, the players might manage to complete all of the stacks up to size 5, hence finishing the game with the maximum score of 25. Second, the team might lose all three life tokens. In this case, immediately after losing the third life token, the game finishes with the minimum score of 0. Third and last, after a player has drawn the final card from the deck, all participants take one more turn. After that, the game finishes with score equal to the sum of the size of the stacks.

The Hanabi game has three features that make it particularly interesting to test techniques for modelling others. This has led some researchers to point to Hanabi as the next great challenge to be undertaken by the AI community [41]. The first feature is the purely cooperative nature of the game, since all participants have a common goal, which is to build the stacks as high as possible. Consequently, players can benefit from understanding the mental state of others, such as their intentions with respect to their cards, or the short-term goals they want to achieve during the course of a game. Additionally, the effectiveness of the developed approaches can be experimentally assessed through the final score.

Second, players in Hanabi have to cope with *partial observability* (or *imperfect information*, the preferred term in the game theory community), as players can see everyone else's cards but not their own. To cope with this, players provide information to one another through hints. There are two facets to these hints. One is the explicit information carried by the hint, i.e. the colour or rank of the cards directly involved. The other facet is the additional implicit information that can be derived from understanding the intention of the player making a move when they provide a hint.

To understand this second facet, consider the situation displayed in Fig. 2a. It is Alice's turn to move, and she decides to give a colour hint to Cathy, pointing to her rightmost card as being the only red card she has. In principle, Cathy now only knows that her rightmost card is red, and all others are not. However, Cathy may be able to understand that Alice would only provide such a hint if she wanted her to play that card, and since it is red and

the red stack has size 3, Cathy's card must be a red 4. Cathy can draw such a conclusion from the observation of the current state of the game, and an assumption about the strategy that Alice is following. In the TomAbd agent model, this implicit information is identified with the abductive explanations that agents are able to generate by taking the perspective of the player making the move.

Finally, the third interesting feature of Hanabi is the fact that the sharing of information is quantified through discrete tokens that must be managed as a collective resource. Agents must manage the number of hint tokens available altogether, by balancing the need to provide a hint in the current state of the game versus discarding a card to recover a token that then becomes available for another hint.

Previous work on autonomous Hanabi-playing agents has followed one of two approaches: rule-based and reinforcement learning (RL) agents. Rule-based Hanabi bots [43–47] play following a set of pre-coded rules. In contrast, RL bots [41, 48–50] apply single-agent or multi-agent RL techniques to learn a policy for the game. Sarmasi et al. [51] have compiled a database of Hanabi-playing agents developed so far.

Our TomAbd agent models relies on a pre-coded strategy to decide what action to take next and hence aligns more closely with the rule-based approach. However, our agent model is agnostic with respect to the specifics of the strategy that the agent follows. In contrast, previous work on rule-based agents for Hanabi [43–47] has focused on the details of the developed strategies. Also, unlike both rule-based and RL agents, our agent model is domain-independent, and it is applied to Hanabi as a test case. The type of reasoning that TomAbd agents engage in is general but can be useful for this particular game.

Autonomous agents for Hanabi can be evaluated in three different settings: self-play, where all the participants of the team follow the same approach and strategy; cross-play, where teams are composed of heterogeneous software agents; and human-play, where teams include human players. The majority of the current research on Hanabi AI evaluates performance during self-play, as we do in this paper. In self-play, RL agents outperform rule-based agents, with the former routinely achieving average scores of around 23 points, while the latter struggle to break into 20 points for the average score. The current state-of-the-art for Hanabi AI combines both RL and rule-based techniques, and produced an average score of 24.6 in self-play [49]. To achieve that, first, one agent learns a game-playing policy while all other team members follow the same pre-coded strategy. Second, all agents use multi-agent learning, where they perform the same joint policy update after every iteration, if feasible. If not, they fall back on the same set of pre-coded rules.

Although RL agents display superior performance in self-play, they require a computationally intensive learning process. Additionally, in a recent survey [42] several types of rule-based or RL agents were paired with human players, forming teams of 2. Despite there being no statistically significant difference in game score between rule-based and RL teammates, humans perceived rule-based agents as more reliable and predictable, while expressing feelings of confusion and frustration more often when paired with RL teammates.

3 Agent model

In the current section, we detail the TomAbd agent model, which constitutes the core of this work. First, we outline the agent architecture, its components and introduce some necessary notation. Later, we explain how these components operate.

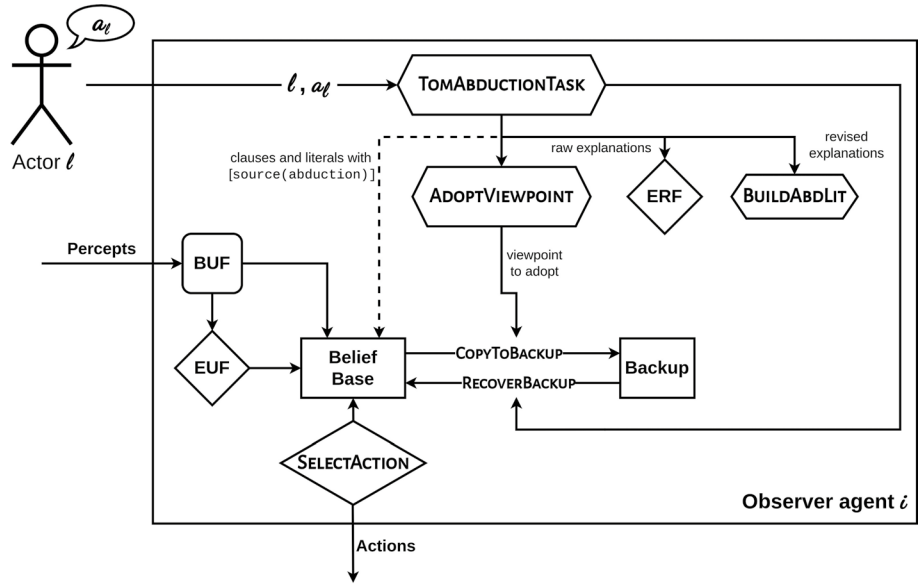


Fig. 3 Architecture of the TomAbd agent model

3.1 Preliminaries

TomAbd is a symbolic, domain-independent agent model with the ability to adopt the point of view of fellow agents, down to an arbitrary level of recursion. Consider the traditional multi-agent setting, where a set of agents $\mathcal{A} = \{i, j, k, \dots\}$ operate in a shared environment. For the remainder of Sect. 3, the explanations are presented from the perspective of an arbitrary observer agent i; i.e. we will be considering the cognitive processes that i autonomously undertakes when it observes its fellow agents taking actions.

The main components of the TomAbd agent model are presented in Fig. 3. Rectangles represent belief base (BB) data structures. Hexagons represent immutable functions, that are not customisable. Diamonds represent functions for whom only default implementations are provided, and that allow users to customise them according to their application's needs. The rounded square for BUF corresponds to the *belief update function*, a common functionality for situated agents. We do not define this function in our work, but tailor the default BUF method in our language of choice to include some operations on the gathered abductive explanations. Details on this are provided in Sect. 4.

The agent architecture is composed of the main BB data structure which contains the logic program that the agent is currently working with, plus a backup to store the agent's own beliefs when switching to another agent's perspective. At all times, the BB contains a logic program: a set of ground literals representing facts about the world and a set of rules representing relationships between literals. We denote by T_i the logic program of agent i; i.e. the content of their BB at initialization time. T_i is composed of the following components.

1. *Percepts* are ground literals that represent the information that the agent receives from the environment. Incoming percepts update the BB according to the *belief update func-*

tion (BUF in Fig. 3). For example, in a situation of Hanabi like the one displayed in Fig. 2, agent Alice would receive the following percepts:

```
has_card_colour(bob,4,blue)
has_card_rank(bob,4,3)
```

These indicate that Alice observes Bob having a card of colour blue and rank 3 in his fourth slot (counting from the left). We assume, implicitly, that the agents are limited by partial observability, meaning that they do not perceive all the information there is to know about the environment. In general, different agents will have access to different parts of the environment, and will receive different percepts. In the Hanabi game in particular, players do not *a priori* know about their own cards or about the order of cards in the deck.

2. *Domain-related clauses* are traditional logic programming rules that establish relationships between facts in the domain. For example, the following clause expresses that, in Hanabi, a card of colour `C` and rank `R` is playable if the size `Sz` of the corresponding stack is one unit below the rank of the card.
```
playable(C,R) :- colour(C), rank(R), stack(C,Sz), Sz=R-1.
```

3. *Impossibility clauses* have atom `imp` as their head and whose body contains literals that cannot hold simultaneously true. They capture the constraints of the domain, if there are any. For example, in the Hanabi game, the following clause states that a player `P` cannot have cards of two different colours, `C1` and `C2`, in the same slot `S`.[2]
```
imp :-
    player(P), slot(S), colour(C1), colour(C2),
    has_card_colour(P,S,C1), has_card_colour(P,S,C2), C1\== C2.
```

As stated in Sect. 2.2, we adopt the *consistency view* when it comes to verifying the expansion of the belief base with an abductive explanation. To incorporate integrity constraints into an agent's program, we need a mechanism that triggers an exceptional event when one or several constraints are violated. This is precisely the role of the impossibility clauses.

To clarify, consider an impossibility clause `imp :- Conj.`. The conjunction `Conj` in its body corresponds to a formula that should never hold true. In other words, its negation ¬`Conj` is equivalent to a traditional integrity constraint *IC* that can never be violated. Therefore, the derivation of `imp` indicates that *IC* has been violated. To avoid this, the generated abductive explanations undergo post-processing operations where they are filtered out if their expansion into the program causes the derivation of `imp`.

4. *Theory of mind clauses* are rules that are essential to the agent's cognitive ability to put itself in the shoes of others. They function as a meta-interpreter on the agent's current program to generate an estimation of another agent's program. ToM clauses have the literal `believes(Ag,F)` as their head, to express the fact that agent *i* believes that agent `Ag` knows about some fact `F`. In the Hanabi domain, the following ToM clause

[2] Note that Jason, our implementation language of choice, does not include any default mechanisms to check the consistency of an agent's BB, i.e. an agent may simultaneously believe b and $\sim b$. It hence becomes the responsibility of the agent developer to implement, if needed, additional mechanisms to avoid such inconsistencies, which the TomAbd agent model achieves through the introduction of impossibility clauses.

indicates that agent i believes that player `Agj` can see the card that a third player `Agk` has in their `S`-th slot, and, in particular, `Agj` can observe its colour `C`.

```
believes(Agj,has_card_colour(Agk,S,C)) :-
    player(Agj), player(Agk), slot(S), colour(C),
    has_card_colour(Agk,S,C), Agj\==Agk.
```

5. *Abducible clauses* have literal `abducible(F)` at their head, and express what missing beliefs can potentially be added to agent i's BB to obtain a more detailed representation of the state of the system. The definition of the literals that may be missing from an agent's BB is a domain-dependent component of the program. For example, in the Hanabi domain, the following belief indicates that, from the viewpoint of i, a player `P` may have, in their `S`-th slot, a card of colour `C` if i does not already hold a belief about the colour of the card in `S`, nor does i explicitly hold a belief explicitly indicating that `P` does not have a card of colour `C` in `S`.[3]

```
abducible(has_card_colour(P,S,C)) :-
    player(P),slot(S),colour(C),
    not has_card_colour(P,S,_),
    not ~has_card_colour(P,S,C).
```

6. *Action selection clauses* are a set of rules with head `action(Ag,Act) [priority(`n`)]` that indicate the pre-conditions for agent `Ag` to select and execute action `Act`. These clauses correspond to agent i's beliefs about the other agents' strategies (for instances where `Ag` $= j, j \neq i$) as well as, potentially, agent i's own strategy (for instance when `Ag` $= i$). The head is annotated with a `priority(`n`)` literal, where n is a number (any number, not necessarily an integer). These priorities state in which order the action selection clauses should be considered when they are queried. Details about the action selection are provided in Sect. 3.4.

   As an example for the Hanabi domain, the following clause indicates that a participant `P` should play their card in slot `S` if it is of a playable colour `C` and rank `R`.

```
action(Ag,play_card(S)) [priority(3.0)] :-
    player_turn(Ag), slot(S),
    has_card_colour(Ag,S,C), has_card_rank(Ag,S,R),
    playable(C,R).
```

   The action selection clauses are used by TomAbd agents to compute abductive explanations from the observation of actions by other agents. Hence, it is compulsory that they capture agent i's beliefs about the strategy agent $j \neq i$ is following. Nonetheless, the TomAbd model is flexible concerning whether such action selection clauses also implement agent i's own strategy. The action selection function presented in Sect. 3.4 certainly provides an avenue to use action selection clauses during the agent's own practical reasoning. However, this is a complement to the TomAbd model (whose focus is on the generation and maintenance of abductive explanations using ToM) rather than a fundamental component.

[3] We distinguish between strong negation (`~ Fact`) and negation as failure (`not Fact`). In epistemic logic notation, they are expressed as $B_i[\sim \phi]$ and $\sim B_i\phi$, respectively.

So far, we have presented the components of the logical program of a TomAbd agent. Now, we move on to explain how they are utilised. The distinguishing feature of our agent model is the ability to put themselves in the shoes of others. For example, when engaging in first-order ToM (recursive level 1), agent i changes their perception of the world to the way in which they believe that some other agent j is perceiving it. In epistemic logic notation, these are the beliefs denoted by $B_iB_j\phi$. In other words, in an attempt to perceive the world how i believes j is perceiving it, agent i's BB changes to $BB_i^j = \{\phi \mid \texttt{believes}(j, \phi)\}$, where BB_i^j denotes i's estimation of j's BB given i's current logic program.

However, the TomAbd agent model is not limited to first-order ToM. It can, in fact, switch its perception of the world to that of another agent *down to an arbitrary level of recursion*. For example, agent i may want to view the world in the way that they believe j believes that k is perceiving it. This corresponds to second-order ToM and is expressed as $B_iB_jB_k\phi$ in epistemic logic notation. In particular, i may want to estimate j's estimation of itself. This is equivalent to the previous case $B_iB_jB_k\phi$ with $i = k$, $B_iB_jB_i\phi$.

The nesting exposed in the previous paragraph can be extended to an arbitrary level of recursion: agent i attempts to view the world how it believes that j believes $\cdots$ that k believes that l views it. This is denoted by $B_iB_j \ldots B_kB_l\phi$. We define the sequence of agent perspectives [j, ..., k, l] recursively adopted by i as a *viewpoint*:

Definition 3 For agent i, a *viewpoint* is an ordered sequence of agent designators [j, ..., k, l] where there are no two consecutive equal elements and the first element is different from i.

Hence, when we talk about agent i *adopting viewpoint* $[j, \ldots, k, l]$ we mean the process by which agent i switches its own perspective of the world by the one it believes that j believes $\cdots$ that k believes that l has. To do this, agent i has to modify its own program T_i, contained in its main BB, by the estimation that it can build of j's estimation $\cdots$ of k's estimation of l's program. This new program will, in general, indeed be an *estimation* since agents have access to (possibly) overlapping but different features of the environment. We denote this estimated program by $T_{i,j,\ldots,k,}$ and define it as follows:

Definition 4 Given agent i with logic program T_i, *i's estimation of viewpoint* [j, ..., k, l] is a new logic program $T_{i,j,\ldots,k,l}$:

$$T_{i,j,\ldots,k,l} = \{\phi \mid T_{i,j,\ldots,k} \vDash \texttt{believes}(l, \phi)\} \tag{1}$$

Equation (1) indicates that, in order to estimate the BB of the next agent whose perspective is to be adopted, the agent must query its current BB to find all the ground literals that, according to the ToM rules, the next agent knows about. Therefore, agent i must substitute their current program by the set of unifications to the second variable in `believes(Ag,F)`. In other words, agent i runs the query `believes(`*ag* `,F)` in its BB and obtains a set of unifications for `F` as output, $\{\texttt{F} \mapsto \phi_1, \ldots, \texttt{F} \mapsto \phi_n\}$, where ϕ_i, $i = 1, \ldots, n$ denotes a groud literal (e.g. `has_card_colour(alice,3,red)`, `has_card_rank(bob,1,5)`). Then, agent i substitutes the contents in its BB by the set $\{\phi_1, \ldots, \phi_n\}$.

Algorithm 1 Function AdoptViewpoint(vp)

Input: vp ($[j, \ldots, k, l]$), a viewpoint according to Definition 3.
Result: agent i substitutes T_i by $T_{i,j,\ldots,k,l}$ in their BB.

1: CopyBBToBackup()
2: **for** p in vp **do**
3: $\quad BB_i^p \leftarrow \{\phi \mid BB \models \texttt{believes}(p, \phi)\}$
4: $\quad BB \leftarrow BB_i^p$
5: **end for**

The operationalisation of Definition 4 is presented in function AdoptViewpoint, Algorithm 1. It takes as its only argument a viewpoint as defined in Definition 3. Given this viewpoint, agent *i* adopts it, first, by saving a copy of its own BB in the backup. Then, *i* queries the ToM clauses with the next agent whose perspective is to be estimated as their first argument. The result of this operation becomes agent *i*'s new BB, and they move on to the next iteration.

3.2 The TomAbductionTask function

Function AdoptViewpoint captures the n^{th}-order Theory of Mind capabilities in the TomAbd agent model, for arbitrary integer value of *n*. However, the purpose of switching one's perspective is to be able to reason from the point of view of another agent. Therefore, it is not enough for *i* to invoke AdoptPerspective. It should, once the switch has occurred, infer the motivation for the actions taken by the other. This reasoning process is implemented in the core function of the TomAbd agent model, TomAbductionTask, in Algorithm 2.

Algorithm 2 Function TomAbductionTask($obsVp$, l, a_l)

Input: $obsVp$, observer viewpoint.
l, acting agent.
a_l, action.

Output: $\mathbf{\Phi'_{act}}$, actLit, $\mathbf{\Phi'_{obs}}$, obsLit: explanations $\mathbf{\Phi'}_{\{\cdot\}}$ that justify the election of action a_l by actor agent l, from the actor's and the observer's viewpoint, respectively; and the literals $\{\cdot\}$Lit containing those explanations in a format suitable to be added to a logic program.

1: $actVp \leftarrow obsVp$.Append(l)
2: AdoptViewpoint($actVp$)
3: $BB \leftarrow BB \cup \{\texttt{viewpoint(actor)}\}$
4: $\mathbf{\Phi} \leftarrow$ Abduce(action(l, a_l))
5: $\mathbf{\Phi'_{act}} \leftarrow$ ERF($\mathbf{\Phi}$)
6: actLit $\leftarrow$ BuildAbdLit($actVp$, $\mathbf{\Phi'_{act}}$)
7: RecoverBackup()
8: AdoptViewpoint($obsVp$)
9: $BB \leftarrow BB \cup \{\texttt{viewpoint(observer)}\}$
10: $\mathbf{\Phi'_{obs}} \leftarrow$ ERF($\mathbf{\Phi}$)
11: obsLit $\leftarrow$ BuildAbdLit($obsVp$, $\mathbf{\Phi'_{obs}}$)
12: RecoverBackup()
13: **return** $\langle \mathbf{\Phi'_{act}}, \text{actLit}, \mathbf{\Phi'_{obs}}, \text{obsLit} \rangle$

TomAbductionTask takes three arguments as input: an observer viewpoint, an acting agent l and the action l took a_l. The last two are straightforward to understand. The observer viewpoint is a list as defined in Definition 3. It indicates what ToM order i is engaging in, and through which other agents it is estimating the perception that the actor l has of the world. For example, suppose i would like to understand why l chose a_l directly. In this case, i observes its peer's action from its own perspective. The observer viewpoint would, in this case, correspond to the empty list ("[ ]"). However, i might want to understand why a third agent j thinks that l made their choice. Then, i is observing l's action through j, and hence the observer viewpoint is $[j]$. This viewpoint can be subsequently extended to any desired level of recursion. For example, agent i may want to estimate the impression that actor l thinks they are making on agent j when executing a_l. This corresponds to observer viewpoint $[l, j]$.

The first step of TomAbductionTask (Lines 1 and 2) is to build the *actor's viewpoint* by simply appending actor agent l to the observer viewpoint and to adopt it by calling AdoptPerspective. Now, agent i is in a position to reason from the perspective of the actor, possibly through a number of intermediate observers. In the simplest case, agent i is switching its logical program T_i to the program it estimates actor l to be working with, i.e. $T_{i,l}$. This case corresponds to agent i engaging in *first-order* ToM at the time of adopting the actor's viewpoint. Alternatively, agent i may switch its program T_i to the program they estimate that j_1 estimates that $\cdots$ j_{n-1} estimates that j_n is working with, $T_{i,j_1,\ldots,j_{n-1},j_n}$, this time engaging in n^{th}-*order* ToM.

Once the actor's viewpoint has been adopted, the agent uses ALP to generate abductive explanations that justify agent l's action a_l. The ALP theory that the agent uses is composed of its current BB (in the general case, $T_{i,j,\ldots,k,l}$), and the set of abducibles derived from it, which we denote as $A_{i,j,\ldots,k,l}$:

$$A_{i,j,\ldots,k,l} = \{\alpha \mid T_{i,j,\ldots,k,l} \models \texttt{abducible}(\alpha)\} \tag{2}$$

The set of plausible abductive explanations is computed by function Abduce in Line 4 of Algorithm 2, using the set of abducibles defined in Eq. (2). The pseudocode for this function is not provided, as it does not constitute any technical innovation. The input to this function is the query $Q = \texttt{action}(l, a_l)$. The Abduce function consists of an abductive meta-interpreter, based on classical SLD clause resolution with a small extension. To compute abductive explanations, this meta-interpreter attempts to prove the query Q as a traditional goal in SLD clause resolution. However, when it encounters a sub-goal that is not provable, before failing the query, it checks whether this sub-goal can be unified to any element in the set of abducibles $A_{i,j,\ldots,k,l}$. If so, the sub-goal is added to the explanation under construction in the branch being currently explored.

Function Abduce backtracks upon failure or completion of the query, just as traditional SLD solvers. Consequently, the output of this function is a set $\mathbf{\Phi}$ of m potential explanations. At the same time, every element in $\mathbf{\Phi}$ is itself a set of ground abducibles from $A_{i,j,\ldots,k,l}$:

$$\begin{aligned} &\mathbf{\Phi} = \{\Phi_1, \ldots, \Phi_m\}, \text{ where } \Phi_h = \{\phi_{h1}, \ldots, \phi_{hn_h}\} \\ &\quad \text{and } \phi_{hg} \in A_{i,j,\ldots,k,l}, \forall h, g \end{aligned} \tag{3}$$

Once the abductive explanations have been computed, they are first refined through the application of the *explanation revision function*, EERF, in Line 5. Then, they are transformed into a literal, that is, to a format suitable to be added to a logical program, through

the BuildAbdLit function in Line 6. Both of these steps are reviewed in detail in the next section.

At this point, the abductive explanations have been computed and post-processed, all from the perspective of the actor, i.e. whilst agent i's BB contains $T_{i,j,\ldots,k,l}$. However, agent i does not derive this information *only* so that it can build a better estimation of the actor's BB. It also reasons about how this information affects beliefs at the *observer's viewpoint level*. Therefore, agent i has to first return to its original program T_i by retrieving it from the backup (Line 7). Then, it adopts the *observer's* viewpoint (Line 9) and perform the same post-processing steps (explanation revision in Line 10 and format transformation in Line 11) from this new perspective. Eventually, agent i recovers its original program T_i from the backup in Line 12.

It should be noted that the TomAbductionTask function does not, by default, add the abductive explanations (or rather, the associated literals generated by BuildAbdLit) to agent i's program T_i (observe the dashed arrow from TomAbductionTask to the BB in Fig. 3). Rather, the function returns the revised explanations and their formatted literals. This choice has been made to allow flexibility to potential users. If necessary, users can perform further reasoning and modifications to the returned explanations. For example, agent i can decide whether to append the returned literals to their BB based on some trust metric it has towards the actor.

3.3 Explanation revision, assimilation and update

This section reviews the post-processing operations that are performed on the raw abductive explanations returned by the Abduce function. In the cases where the implementations provided are *defaults*, this is clearly indicated. Details of how these defaults can be overridden are provided in Sect. 4.

Algorithm 3 Function ERF($\mathbf{\Phi}$) (*default* explanation revision function)

Input: $\mathbf{\Phi} = \{\{\phi_{11}, \ldots, \phi_{1n_1}\}, \ldots, \{\phi_{m1}, \ldots, \phi_{mn_m}\}\}$, a set of m explanations, each being a set of ground abducibles.
Output: $\mathbf{\Phi}' = \{\{\phi'_{11}, \ldots, \phi'_{1n'_1}\}, \ldots, \{\phi'_{m'1}, \ldots, \phi'_{m'n'_{m'}}\}\}$, a set of m' refined explanations.

1: $\mathbf{\Phi}' \leftarrow \{\}$
2: **for** Φ_h in $\mathbf{\Phi}$ **do**
3: $\quad \Phi'_h \leftarrow \{\phi_{hg} \mid \phi_{hg} \in \Phi_h \text{ and } BB \not\models \phi_{hg}\}$
4: $\quad$ **if** $BB \cup \Phi'_h \not\models$ `imp` **then**
5: $\quad\quad \mathbf{\Phi}' \leftarrow \mathbf{\Phi}' \cup \{\Phi'_h\}$
6: $\quad$ **end if**
7: **end for**
8: **return** $\mathbf{\Phi}'$

During the execution of TomAbductionTask, two calls are made to the *explanation revision function* (EERF), one from the point of view of the actor and one from the point of view of the observer. The purpose of this function is to refine and/or filter the raw explanations based on the current content of agent i's BB, which is either the estimation of the actor's program $T_{i,j,\ldots,k,l}$ or the estimation of the observer's program $T_{i,j,\ldots,k}$.

The default implementation of the EERF function appears in Algorithm 3 and consists of two steps. First, in Line 3, the agent trims every explanation $\mathbf{\Phi}_h$ (a set of ground abducibles) to remove *uninformative* atoms. Admittedly, this step only makes a difference when EERF is called from the perspective of the *observer* (Line 10 in Algorithm 2), and not from the perspective of the *actor* (Line 5 in Algorithm 2). The abduction meta-interpreter does not add proven sub-goals to the explanation under construction. Therefore, from the perspective of the actor (where the raw abductive explanations are actually computed), there cannot be uninformative facts in the explanation sets.

The second step is a consistency check (Lines 4 to 6 in Algorithm 3). This check takes in every trimmed explanation and inspects whether it, together with agent i's current BB, entails any impossibility clause. Recall from the discussion in Sect. 3.1 that the derivation of `imp` is equivalent to an integrity constraint *IC* being violated. The impossibility clauses that this check considers include both domain-related and impossibility clauses derived from prior executions of TomAbductionTask. If no violation occurs, the explanation is returned as part of the set of revised explanations.

Here, we have only presented a basic EERF implementation that can be customised if needed. For example, the EERF could annotate every explanation $\mathbf{\Phi}_h$ with an uncertainty metric. Alternatively, it could operate differently depending on whether it is being called while the agent is working under the *actor* or the *observer* point of view. In fact, the belief addition operations in Lines 3 and 9 of Algorithm 2 are there precisely to allow for this possibility. Further details are provided in Sect. 4.

Algorithm 4 Function BuildAbdLit(vp, $\mathbf{\Phi}$)

Input: vp, a viewpoint
$\mathbf{\Phi} = \{\{\phi_{11}, \ldots, \phi_{1n_1}\}, \ldots, \{\phi_{m1}, \ldots, \phi_{mn_m}\}\}$, a set of (revised) explanations.
Output: Λ, a literal containing the input explanation, in a format suitable to be added to the agent's logical program.

1: $\Lambda \leftarrow \{$`imp [source(abduction)] :-`
$(\sim \phi_{11} \mid \ldots \mid \sim \phi_{1n_1})$, ... , $(\sim \phi_{m1} \mid \ldots \mid \sim \phi_{mn_m})$.$\}$
2: $vp' \leftarrow$ Reverse(vp)
3: **for** p in vp' **do**
4: $\quad \Lambda' \leftarrow$ `believes`(p, Λ)
5: $\quad \Lambda \leftarrow \Lambda'$
6: **end for**
7: **return** Λ

The set of revised explanations $\mathbf{\Phi'}$ is, like the set of raw explanations $\mathbf{\Phi}$, a *set of sets* of ground abducibles, see Eq. (3). Therefore, it is not in a suitable format to be added to agent i's BB, which is a logical program composed of facts and clauses. The conversion from a set of sets to a clause that can be added to a logical program is performed by function BuildAbdLit (short for "build abductive literal") in Algorithm 4.

To understand how this function operates, consider that a (revised) abductive explanation $\mathbf{\Phi} = \{\{\phi_{11}, \ldots, \phi_{1n_1}\}, \ldots, \{\phi_{m1}, \ldots, \phi_{mn_m}\}\}$ can be written as the following disjunctive normal form (DNF):

$$\mathbf{\Phi} = \left(\phi_{11} \wedge \ldots \wedge \phi_{1n_1}\right) \vee \ldots \vee \left(\phi_{m1} \wedge \ldots \wedge \phi_{mn_m}\right) \tag{4}$$

The formula in Eq. (4) must hold, meaning it has the status of a traditional *IC* discussed in Sect. 2.2. Therefore, its negation $\neg\mathbf{\Phi}$ must never hold true. If $\neg\mathbf{\Phi}$ is derived from the agent's program, it means that the formula in Eq. (4) has been violated, and an exceptional event (i.e. the derivation of `imp`) should be triggered. This observation leads to the use of $\neg\mathbf{\Phi}$ to build a new impossibility clause. This new clause has the same format as the domain-related impossibility clauses presented in Sect. 3.1 but its head `imp` is annotated with `source(abduction)` to denote that it is not domain-specific but derived from an abductive reasoning process. This step corresponds to Line 1 in Algorithm 4.

Nonetheless, this new impossibility clause does not consider the level of recursion, or, in other words, the *viewpoint*, where the explanation was generated. This information needs to be incorporated in Lines 2 to 6. In summary, if the agent is operating under viewpoint $[j, \ldots, k, l]$, BuildAbdLit nests the abductive impossibility clause constructed in Line 1 into the following literal:

$$\begin{aligned}\texttt{believes}(j, \ldots, \texttt{believes}(k, \texttt{believes}(l, \{\texttt{imp[source(abduction)]} : - \\ (\neg\phi_{11} \mid \ldots \mid \neg\phi_{1n_1}), \ldots, (\neg\phi_{m1} \mid \ldots \mid \neg\phi_{mn_m})\})) \ldots).\end{aligned} \tag{5}$$

Therefore, the next time agent i adopts viewpoint $[j, \ldots, k, l]$, the bare `imp [source(abduction)]` clause will become part of their BB (assuming the user has decided to add it to T_i in the first place).

Algorithm 5 Function EUF (*default* explanation update function)

Result: The agent removes from their BB the abductive literals whose associated explanation is no longer informative.

1: $\Sigma \leftarrow \{\}$
2: **for all** $\langle vp, \mathbf{\Phi}, \texttt{lit}\rangle$ in BB **do**
3: AdoptViewpoint(vp)
4: **if** $BB \models \mathbf{\Phi}$ **then**
5: $\Sigma \leftarrow \Sigma \cup \{\texttt{lit}\}$
6: **end if**
7: RecoverBackup()
8: **end for**
9: $KB \leftarrow KB \setminus \Sigma$

Finally, there is one last operation performed on the clauses and literals derived from TomAbductionTask, which is the *update* of those that have been incorporated into the original BB of the agent, T_i, as new percepts are received. We refer to this operation as the *explanation update function* (EEUF). In contrast to the other functions presented in this section, the EEUF is not executed within TomAbductionTask, but is called from the *belief update function* (BUF, see Fig. 3). The BUF is a standard function of the BDI agent reasoning cycle whose purpose is to update the BB depending on the percepts received from the environment and the messages passed on by other agents. Therefore, upon receiving percepts from the environment, the agent first modifies its ground percept beliefs, and then

updates clauses and literals derived from previous executions of TomAbductionTask, if there are any.

The default implementation of EEUF appears in Algorithm 5. In it, agent i discards previous abductive explanations if they are deemed to be no longer informative *at the viewpoint at which they were generated*. To do so, the agent loops over all the literals that originated from an abductive reasoning process, denoted by the tuple $\langle vp, \mathbf{\Phi}, \texttt{lit} \rangle$ composed of the viewpoint vp where the explanation $\mathbf{\Phi}$ originated and the associated literal (or clause) `lit` (Line 2). Agent i then adopts viewpoint vp with a routine call to AdoptViewpoint, and checks if explanation $\mathbf{\Phi}$ can be derived from the current BB, $T_{[i|vp]}$.[4] If so, the explanation is deemed to be no longer informative and its associated literal `lit` is added to a removal set.

3.4 Action selection

Algorithm 6 Function SelectAction (*default* action selection)

Output: a, an action.

1: **for all** clauses $\mathcal{H}$ with head `action(Ag,Act) [priority(`p`)]`, in descending order of p **do**
2: $\mathcal{H}$.Map(`Ag` $\mapsto i$)
3: $\mathcal{B} \leftarrow \mathcal{H}$.Body()
4: $\Gamma \leftarrow$ SkolemisedAbducibles($\mathcal{B}$)
5: **for all** γ in Γ **do**
6: $\Pi \leftarrow$ Instantiate(γ)
7: $\mathcal{A} \leftarrow \{\}$
8: **for all** π in Π **do**
9: **if** $BB \cup \pi \not\models$ `imp` **then**
10: $a_\pi \leftarrow \underset{a}{\text{argmax}}\{m \mid BB \cup \pi \models \texttt{action}(i, a)\texttt{[priority(}m\texttt{)]}\}$
11: $\mathcal{A} \leftarrow \mathcal{A} \cup \{a_\pi\}$
12: **end if**
13: **end for**
14: **if** $\mathcal{A} = \{a\}, \|\mathcal{A}\| = 1$ **then**
15: **return** a
16: **end if**
17: **end for**
18: **end for**
19: **return** *null*

The functions presented so far constitute the agent's core cognitive abilities combining Theory of Mind and abductive reasoning. However, the purpose of undergoing all this cognitive work is for agent i to be in a more informed position when it comes to i's own decision-making. To do so, agent i needs to consider the generated abductive explanations

[4] We use Prolog notation for lists $[H \mid T]$, where H is the first element (head) and T is the tail of the list, which is itself another list, possibly empty.

when reasoning about which action to perform next. We provide such a function, SELECTACTION in Algorithm 6, which takes into account *all* the impossibility clauses in the agent's BB, including those coming from the output of TOMABDUCTIONTASK.

Similarly to the EERF and EEUF, the provided implementation is a basic one, and it is customisable. The user can, for example, reason probabilistically about which action to take next, in case they have associated an uncertainty metric to the generated abductive explanations. The default implementation in Algorithm 6 takes a cautious approach, where an action is only selected if it is the action prescribed by the action selection clauses in all the possible worlds.

Additionally, the SELECTACTION function presented here is a complement to the other functionalities of the TOMABD agent model, and not a core component of the model. We provide a default querying mechanism to select an action given the action selection clauses and the set of current impossibility constraints. However, the agent developer might decide to use an alternative implementation that, for instance, does not use the action selection clauses to pick the action to execute next, or they might decide to not use the SELECTACTION function at all. This is enabled by the fact that this function has been wrapped in an internal action (IA) that can be called from within the agent code. More details on this point are provided in Sect. 4.

Algorithm 6 proceeds as follows. First, in Line 1, it retrieves action selection clauses in descending order of priority, so rules with higher priority take precendence over rules with lower priority. Then, the variable at the first argument in the head is unified with the identity of agent i in Line 2.

Second, the body of the clause is retrieved (Line 3) and the set of *skolemised abducibles* is built. This is done by function SKOLEMISEDABDUCIBLES (whose pseudo-code is not provided) in Line 4. This means that whenever an abducible in the rule body cannot be proven by the agent's BB (i.e. T_i), its free variables are substituted by Skolem constants. In general, one action selection clause will generate several skolemised forms of its abducibles.

Third, the agent searches for all of the potential instantiations of every skolemised form. This corresponds to the call to function INSTANTIATE in Line 6. Again, for every set of skolemised abducibles, there will be, in general, several possible ways of binding their variables. Each of these possible instantiations provides additional beliefs that can partly complement the agent's BB to obtain a more complete view of the current state of the world. However, it is not necessary to complement the agent view to the point of complete observability, just to add enough information to be able to query the action selection clause currently under consideration.

The agent only considers the complete instantiations of abducibles that, together with agent i's BB, do not lead to an impossibility clause. This check takes place in Line 9. For those instantiations that pass the check, agent i queries for the action of maximum priority that is entailed if the grounded abducibles were part of the BB. As we have seen, for this default implementation, action selection clauses with higher priority take precedence over clauses with lower priority. Hence, when querying for actions, the one with the highest priority is returned.

So, every action selection clause (i.e. an iteration of the loop in Line 1) leads to several sets of skolemised abducibles. In its turn, every set of skolemised abducibles (i.e. an iteration of the loop in Line 5) leads to several sets of ground abducibles. If each of these instantiations leads to the same action (Line 14), the SELECTACTION function returns the action in question and execution continues from the point where the function had been called.

If all the action selection clauses have been processed and no action has been selected, the SELECTACTION function returns *null*. The user is advised to deal with this

situation by including some contingency measure, e.g. use a *default action* when SELECTACTION return *null*. Further details are provided in Sect. 4.

To illustrate how the default SELECTACTION function works, consider the action selection clause provided as an example in Sect. 3.1:

```
action(Ag,play_card(S)) [priority(3.0)] :-
    player_turn(Ag), slot(S),
    has_card_colour(Ag,S,C), has_card_rank(Ag,S,R),
    playable(C,R).
```

Then, after the execution of Line 3, we have:

$\mathcal{B} \leftarrow$ `player_turn(`i`), slot(S),`
`has_card_colour(`i`,S,C), has_card_rank(`i`,S,R), playable(C,R).`

Now, suppose agent Alice, in the setting of Fig. 2a, has the following information in her BB, derived from a hint:

```
~has_card_rank(i,1,3) [source(hint)]
has_card_rank(i,2,3) [source(hint)]
~has_card_rank(i,3,3) [source(hint)]
has_card_rank(i,4,3) [source(hint)]
~has_card_rank(i,5,3) [source(hint)]
```

This means that Alice knows that she has cards of rank 3 in her 2nd and 4th slots, and that she has cards of rank different from 3 at all others.

Then, after execution of Line 4, we have:

$\Gamma \leftarrow$ {{`has_card_colour(`i`,1,`sk_1`), has_card_rank(`i`,1,`sk_2`)`}, {`has_card_colour(`i`,2,`sk_3`)`}, {`has_card_colour(`i`,3,`sk_5`), has_card_rank(`i`,3,`sk_6`)`}, {`has_card_colour(`i`,4,`sk_7`)`}, {`has_card_colour(`i`,5,`sk_9`), has_card_rank(`i`,5,`sk_{10}`)`}}

where sk_n are Skolem constants.

In addition, suppose that she has in her BB the following IC, derived from a previous execution of TOMABDUCTIONTASK:

```
imp [source(abduction)] :-
    ~has_card_colour(i,2,blue), ~has_card_colour(i,2,white).
```

This IC would have been derived by BUILDABDLIT (Algorithm 4) from the following set of (revised) abductive explanations:

$\mathbf{\Phi} =$ {{`has_card_colour(`i`, 2, blue)`}, {`has_card_colour(`i`, 2, white)`}}

meaning that, from a previous move by another player, agent i interpreted that they must have either a blue or a white car in the second slot.

Then, when looping through the second element of Γ, in Line 6, the following instantiations will be generated:

$$\begin{aligned}\Pi \leftarrow \{&\{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{blue})\},\\ &\{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{green})\},\\ &\{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{yellow})\},\\ &\{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{red})\},\\ &\{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{white})\}\}\end{aligned}$$

Of all the instantiations in Π, only two are compatible with the previous IC:

$$\{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{blue})\},\ \{\texttt{has\_card\_colour}(i,\texttt{2},\texttt{white})\}$$

When querying for which action to select, the two previous instantiations will lead to `play_card(2)` (due to the action clauses with priority 3.0), and this will be the return value of the function SelectAction.

4 Implementation

```
+trigger : context
    <- ... ;
    tomabd.agent.tom_abduction_task(
        +ObsVp,                 // a list
        +Actor,                 // an atom
        +Action,                // a ground literal
        -ActorVpExpls,          // a list of lists
        -ObsVpExpls,            // a list of lists
        -ActLit,                // a literal
        -ObsLit                 // a literal
    );
    ...
```

Listing 1: Usage of the tomabd.agent.tom_abduction_task IA. A "+" precedes variables that must be bound at invocation time, while a "-" precedes variables that are bound by the IA.

The agent model presented in Sect. 3 has been implemented in Jason [7], an agent-oriented programming language based on the BDI architecture. Jason implements and extends the abstract AgentSpeak language [52], offering a wide range of features and options for customisation. To utilize the TomAbd in their projects, the user is required to have prior knowledge on the Jason programming language [7, Chapter 3], the basics of the Jason reasoning cycle [7, Chapter 4] and the customisation of Jason components [7, Chapter 7]. Our

implementation is documented and publicly available under a Creative Commons license.[5] It has been packaged into a Java Archive (.jar) file to facilitate its use as an external library for developers who want to include it in their applications.

The core of the implementation consists of the tomabd.agent.TomAbdAgent class, a subclass of Jason's default agent class. It contains all the methods to implement the functions in Fig. 3 (plus some auxiliaries). Besides the main BB (inherited from Jason's default agent class), tomabd.agent.TomAbdAgent also has a backup BB. The BUF is part of Jason's default agent class, which we override in our implementation to include a call to EEUF after percepts have been updated. The abductive reasoner implementing the ABDUCE function is included as a set of Prolog-like rules in an AgentSpeak file, which the agent class automatically includes at initialization time.

A call to the main function of this agent model (TOMABDUCTIONTASK) is not included within Jason's native reasoning cycle, and hence does not constrain it in any way. Instead, an internal action (IA), tomabd.agent.tom_abduction_task, is provided as an interface to the agent's method. The usage of this IA is illustrated in Listing 1. Calling TOMABDUCTIONTASK through an IA allows the agent developer flexibility and control over when to trigger it from within the application-specific agent code. Furthermore, the invocation of TOMABDUCTIONTASK from an IA ensures that the whole function is executed within one act() step of the BDI reasoning cycle. Therefore, its execution does not interfere with changes in the BB that happen during other perceive() or act() steps (e.g. belief removal or addition operations) of the BDI reasoning cycle.

We have exposed the reasons why TOMABDUCTIONTASK is not called from within the agent's reasoning cycle, but using an IA interface. In summary, through an IA the TOMABD agent model provides additional functionalities to Jason agents, without restricting the use of other custom components nor placing constraints on the BDI reasoning cycle. Similar remarks apply to the SELECTACTION function and its counterpart IA tomabd.agent.select_action (also included in our implementation), which operates similarly to tomabd.agent.tom_abduction_task but provides an interface to SELECTACTION instead. It is called as tomabd.agent.select_action(A), where A is a free variable bounded by the IA to the return value of SELECTACTION.

```
+!trigger : context
    <- ... ;
    tomabd.agent.select_action(Action);
    Action.                                  // execute action on the environment

-!trigger[error(ia_failed),
    error_msg(''internal action tomabd.agent.select_action failed'')]
    <- ?default_action(DefAct);              // look for a default action in the BB
    DefAct.
```

Listing 2: Usage of the tomabd.agent.select_action IA and a possible contingency plan to handle its failure.

[5] https://github.com/nmontesg/tomabd.

As covered in Sect. 3.4, if using the default implementation of SelectAction (or any other implementation that may return *null*), contingency measures should be put into place to handle the possibility of failure. In Listing 2 we propose a strategy to do this. In the first plan, the agent uses the tomabd.agent.select_action IA to decide which action to perform next. If the IA is successful, the agent goes on to execute it as a standard action on the environment. If not, the second plan in Listing 2 handles the failure. The annotations in this plan, namely the error and error_msg literals, ensure that this plan handles only the failure of tomabd.agent.select_action, not of any other source of failure in the previous plan (e.g. the failure of execution of Action on the environment). In Listing 2, if tomabd.agent.select_action fails, the agent queries its BB to look for a default action, and executes it on the environment.

```
public class MyAgent extends TomAbdAgent {

    @Override
    public ListTermImpl erf(ListTermImpl expls) {
        Literal obsViewpoint = Literal.parseLiteral("viewpoint(observer)");
        Literal actViewpoint = Literal.parseLiteral("viewpoint(actor)");
        Unifier un = new Unifier();
        if (believes(actViewpoint, un)) {
            return erfAct(expls);
        } else if (believes(obsViewpoint, un)) {
            return erfObs(expls);
        } else {
            throw new RuntimeException("Error in erf: no valid viewpoint belief");
        }
    }

}
```

Listing 3: Customisation of the ERF function, based on whether the agent is currently adopting the actor or the observer's viewpoint.

In Listings 1 and 2, the IAs tomabd.agent.tom_abduction_task and tomabd.agent.select_action are invoked as part of the body of agent plans. Nonetheless, similarly to standard Jason IAs, they may also appear in the context of plans. If that is the case, the execution of the corresponding TomAbductionTask and SelectAction would be moved to the deliberate() step of the BDI reasoning cycle. Whether it is more desirable to have the mentioned functions execute in an act() step (by placing their corresponding IAs in the plan body) or in a deliberate() step (by placing them in the plan context) is a decision for the agent developer to take.

In summary, of the agent functions displayed in Fig. 3, only TomAbductionTask and SelectAction have a correspondinging IA interface, with SelectAction being the only one of the two that is customisable. Additionally, the EERF and the EEUF are also customisable, but these are called from within other functions and hence are not accompanied by an IA interface.

To override the default implementation of any of these functions, the developer needs to write new erf(), euf() and selectAction() methods in an agent subclass of tomabd.agent.TomAbdAgent. For example, Listing 3 provides an agent subclass with an

alternative implementation of EERF that applies a different revision function depending on whether the agent is currently working at the observer's or at the actor's perspective.

5 Results

5.1 Experimental setting

As a proof of concept, we have applied the TomAbd agent model to the Hanabi domain presented in Sect. 2.3, for teams of 2 to 5 players in self-play mode.[6] This means that the teams are homogeneous, composed exclusively of TomAbd agents. As for the `action/2` clauses that implement the team strategy, Hanabi has a thriving community of online players that have gathered a set of *conventions* for the game, called the H-group conventions.[7] These conventions comprise definitions (e.g. what constitutes a *save hint* or a *play hint*) and guidelines to follow during game play. We have taken inspiration from these conventions to devise our action selection clauses. However, while these conventions are itemised according to player experience, we have only made use of the introductory-level ones. Our goal is not to synthesise the playing strategy that achieves the maximum possible score, but to explore the usefulness of the capabilities of the TomAbd model in an example domain. We leave the exploration of more sophisticated conventions for future work.

To trigger the execution of the TomAbductionTask function, participants publicly broadcast their action of choice prior to execution. To handle these announcements, we define a Knowledge Query and Manipulation Language (KQML) custom performative, publicAction. Agents react to messages with this performative by executing the tomabd.agent.tom_abduction_task IA using first-order ToM. This means that, when adopting the other acting agent's viewpoint, agents do not take that perspective through any intermediate agents. Hence, agents work with program $T_{i,l}$, where i is the observer and l is the acting agent, when adopting the actor's viewpoint to generate explanations. Consequently, the variable ObsVp in Algorithm 2 is bound to the empty list "[ ]". Additionally, all the generated literals from the abductive explanations are immediately incorporated into the agent's program.

We evaluate the performance of the TomAbd agent model for the Hanabi domain, using the basic set of H-group conventions and first-order ToM. We ran 500 games with random seed 0 to 499, for every team size and switching on/off the call to TomAbductionTask. The simulations were distributed over 10 nodes at the high performance computing cluster at IIIA-CSIC.[8]

5.2 Score and efficiency

The results are first evaluated in terms of the absolute score at the end of every game. This is the most straightforward performance metric and one that allows comparison with other work on Hanabi AI. Beyond the absolute score, we also evaluate teams according to their

[6] The code that applies the TomAbd model to Hanabi is available at https://github.com/nmontesg/tomabd/examples/hanabi.

[7] https://hanabi.github.io/.

[8] https://www.iiia.csic.es/en-us/research/ars-magna/.

communication efficiency, which we define as the ratio between the final score and the total number of hints given during the course of a complete game. This metric quantifies how efficient the team is at turning communication (i.e. hints) into utility (i.e. score). Intuitively, a lower bound for the efficiency metric is $\frac{1}{2}$, as two hints are needed (one for colour and one for rank) to completely learn about a card's identity and be able to safely play it.

Box plots for the results of performance in terms of score and efficiency are displayed in Fig. 4. Additionally, the experimental distributions are available in Fig. 8 in the Supplementary information. Visually, Fig. 4 conveys that the incorporation of ToM and abductive reasoning capabilities boosts performance, both in terms of score and efficiency. Furthermore, regardless of team size, the efficiency is over the lower bound for over 75% of the games when the ToM and abduction capabilities are used. In contrast, when these cognitive abilities are switched off, the efficiency falls below the lower bound for approximately 75% of the runs.

In Table 1 the means and standard deviations for the score and communication efficiency are provided. Moreover, the average percentage increase (comparing pairs of games with the same random seed with and without calls to TomAbductionTask) is displayed in the *Improvement* row for every team size. The average scores in Table 1, even with the ToM and abductive reasoning switched on, are still far from the current state-of-the-art in Hanabi AI, with average score of up to 24.6 [49]. Nonetheless, they are in line with the performance of current rule-based Hanabi-playing bots (see Table 1 by Siu et al. [42]). Moreover, recall that the goal of this work is not to synthesise the optimal team strategy for Hanabi, but to develop a domain-independent agent model capable of putting itself in the shoes of other agents and reasoning from their perspective. The Hanabi game was selected as a test bed for this model, alongside a very simplistic playing strategy. Yet, we anticipate that the results presented here could be improved through the introduction of more advanced playing conventions, such as "prompts" and "finesses".

To confirm the observation that performance is better when agents make use of the TomAbductionTask function, we used statistical testing. First, we applied the Shapiro-Wilk test of normality [53] to test that the score and efficiency distributions in Fig. 8 are normally distributed, under all the experimental conditions. We confirm that this is indeed the case for confidence level 99%. Second, we used the paired samples *t*-test [54] to confirm that the averages for the score and the efficiency, across all team sizes, are significantly better when the TomAbduction function is used. We used the paired samples version of the *t*-test, rather than the independent samples, because games with equal random seeds are related as far as the sequence of cards that are dealt from the deck is the same for all. The results confirm that the averages for the score and the efficiency are significantly better when the TomAbductionTask function is called with respect to when it is not, across all team sizes and for confidence level 99%. Therefore, we conclude that the use of the TomAbductionTask function quantitatively boosts the performance of teams, independently of their size.

Once we confirmed that, indeed, the execution of the TomAbductionTask function produces significantly better performance in terms of score and efficiency, we sought to quantify this improvement. As explained earlier, games of equal team size and random seed are related since the sequence of dealt cards is the same for both. For this reason, it makes sense to compare the score and the efficiency for games with the ToM capabilities on and off, while controlling for team size and seed. To do this, we computed the percentage increase in the score and efficiency when using the TomAbductionTask function, and then aggregated these values into the average across all random seed. These results are displayed in the *Improvement* row in Table 1. They show that there is indeed a notable percentage increase in both score and efficiency, and this improvement increases monotonically with

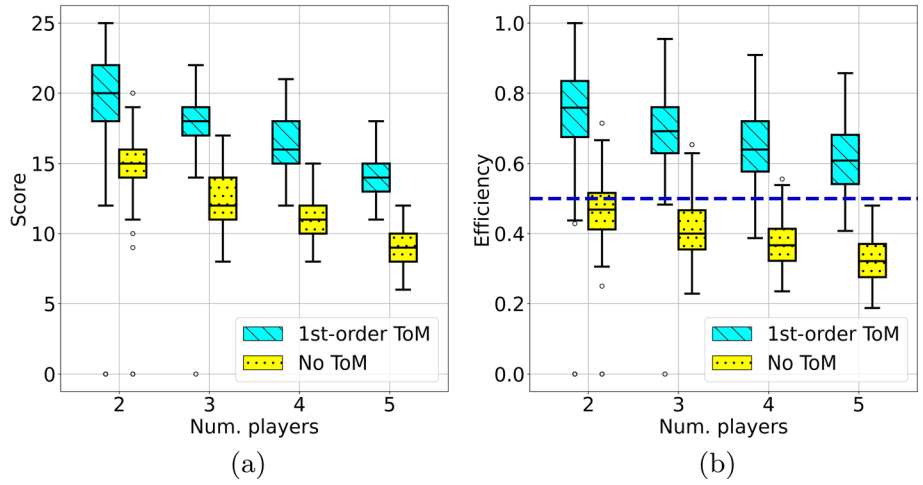


Fig. 4 Results for the score (**a**) and communication efficiency (**b**). Cyan ruled boxes correspond to games where agents make use of the capabilities of the TomAbd agent model, and yellow dotted boxes correspond to games where they do not. The dashed line on the efficiency plot indicates the bound of two hints per score point (Color figure online)

Table 1 Average, standard deviation and improvement when using the TomAbductionTask function for the score and communication efficiency

Num. players	TomAbductionTask	Score	Efficiency
2	Yes	18.61 ± 5.92	0.71 ± 0.22
	No	14.57 ± 2.93	0.46 ± 0.10
	Improvement	27%	54%
3	Yes	17.97 ± 1.94	0.70 ± 0.10
	No	12.52 ± 1.56	0.42 ± 0.07
	Improvement	45%	71%
4	Yes	16.50 ± 1.61	0.64 ± 0.09
	No	11.23 ± 1.36	0.38 ± 0.06
	Improvement	49%	75%
5	Yes	14.42 ± 1.37	0.62 ± 0.09
	No	9.23 ± 1.30	0.33 ± 0.06
	Improvement	59%	91%

Paired samples *t*-test confirmed that the average score and efficiency are significantly better when the TomAbductionTask function is used, regardless of team size

team size. For example, the increase in score is around 30% for teams of two players while it reaches almost 60% for the largest teams (five players).

5.3 Elapsed time

The results presented in the previous section clearly prove that the use of the TomAbductionTask function (using first-order ToM and with the selected action selection rules) has a positive effect on the team performance, both in their final score and the efficiency of communication. In this section, we analyse the computational load associated to this performance boost.

In Fig. 5 we present the results for the elapsed time of the TomAbductionTask function. Every box contains data on at least 4,000 runs of the function. The samples in Fig. 5 correspond to the execution of the TomAbductionTask function across different games with different random seeds, and at different stages of the game.

In all cases, the execution of the function has magnitude in the hundreds of milliseconds. As expected, the elapsed time tends to increase and fall within a larger range as the team size increases. This is due to the larger BB that agents have to manage when they are part of a larger team. This results in a larger space to search through in order to construct the abductive explanations. For example, for teams of size 2, agents have 10 percepts concerning the rank and colour of the cards of their fellow player. Meanwhile, for teams of size 5, agents have 32 percepts about the cards of others.

As explained in Sect. 4, the TomAbductionTask function (and also SelectAction) is executed through an IA at the discretion of the developer. Hence, it is not natively integrated into the BDI reasoning cycle. Nonetheless, there is one TomAbd-specific function that *is* called from the BDI reasoning cycle: EEUF, which is called from BUF, a central component of the sensing step in the Jason reasoning cycle. Hence, to quantify the burden put on the BDI reasoning cycle by the TomAbd agent model, we have to analyse the performance of EEUF.

In Fig. 6 we present the results for the elapsed time of EEUF. Every box contains at least 750 data points. The results are itemized by the number of explanations in the agent's BB at the time EEUF was executed, since our default implementation of EEUF loops over the literals in the BB that originated from an abductive reasoning process. There were no instances found with 4 or more explanations. The results in Fig. 6 show that, for the first order ToM we are using for the Hanabi domain, EEUF entails a negligible overhead on the execution time of the Jason reasoning cycle. Its execution time is around two orders of magnitude smaller than that of TomAbductionTask and, as expected, follows an approximately linear trend with respect to the number of abductive explanations in the BB. Nonetheless, we expect the execution time of EEUF to increase as higher-order ToM is introduced.

5.4 Information gain

The previous analyses quantify the overall outcome of a game, either in terms of score or efficiency, and their computational requirements. Now, in the current and the following section, we would like to quantify the amount and the value of the information that agents derive from the execution of the TomAbductionTask function.

The analysis that follows relies on some features that are specific to Hanabi and hence not generally exportable to other domains where the TomAbd agent model may be applied. The first enabling feature is the fact that Hanabi has a well-defined set of states that the game might be in at any given moment. These states are defined by the heights of the stacks, the available information tokens, the number of lives remaining and the cards in the

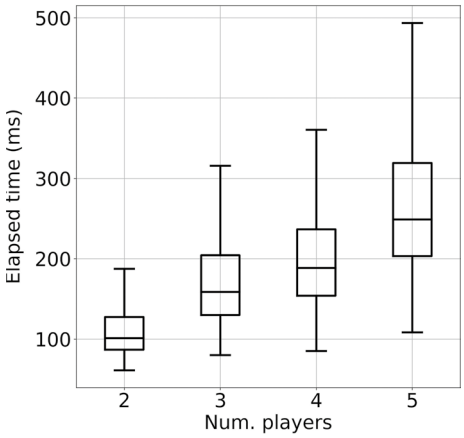


Fig. 5 Execution time of the TomAbductionTask function for the Hanabi domain with different team sizes

discard pile, which are all observable by all players. Additionally, states are also characterised by the cards at each player's hand, which are not common knowledge.[9]

In game theoretical terms, the above feature is referred to as Hanabi being a game of imperfect yet complete information. In other words, players in Hanabi do not in general have access to all the information characterising the current state of the game, but they can infer a finite set of states the game might be in. Additionally, using domain knowledge (namely, the number of duplicate identical cards, which depends on their rank) and, potentially, the abductive explanations currently in their BB, agents can compute, for every slot S in their hand, the marginal probability distribution for the colour and the rank of their card in S. By examining these probability distributions and comparing them to the true one (which assigns unit probability to the colour and rank of the actual card a player holds in S, and zero otherwise), we can quantify the information gain, which we present in the present section.

The second feature of Hanabi that enables the analysis on information value in Sect. 5.5 is the fact that, as any classical game, Hanabi has a set of well-defined end-states with an assigned numerical utility or score. This characteristic, together with the previous one, allows us to relate the reduction in uncertainty of the probability distributions over the cards in player's slots with the increase in score when the ToM and abductive reasoning capabilities of the TomAbd agent model are introduced.

We begin, then, by quantifying the gain in information derived from the combination of ToM and abductive reasoning. To help with this, consider Fig. 7. Agents maintain a marginal probability distribution over the identity of the card at each of their slots, i.e. the tuple $(\mathcal{C}, \mathcal{R})$ of random variables corresponding to the card's colour and rank. At every turn of the game, there are three distributions to consider: the pre-action distribution *before* the action is executed $\mathbb{P}_S^{\text{preAct}}$, the post-action distribution *after* the action is executed $\mathbb{P}_S^{\text{postAct}}$, and the post-explanation distribution after the action is executed *and* the abductive literals derived from the TomAbductionTask function have

[9] The sequence of cards in the deck, which is hidden to all players, might also be considered as part of the state description in Hanabi. However, we prefer to view it as a randomising device rather than as part of the state description. In any case, its treatment is not relevant to the analysis in Sects. 5.4 and 5.5.

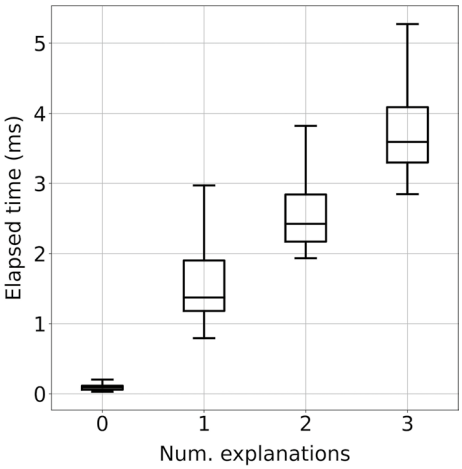


Fig. 6 Execution time of the EEUF for the Hanabi domain, as a function of the abductive explanations in the agent's BB at execution time

been introduced into the agent's BB $\mathbb{P}_S^{\text{postExpl}}$. In addition to these three distributions, the true identity of a card at slot S is denoted as $(\mathcal{C}^S, \mathcal{R}^S)$. We refer to $(\mathcal{C}^S, \mathcal{R}^S)$ as the *ground truth* for slot S. Trivially, the true probability distribution can be considered to be $\mathbb{P}_S^*(\mathcal{C}^S, \mathcal{R}^S) = 1$ and 0 otherwise.

To quantify the distance between two probability distributions, we use the Kullback–Leibler divergence [55], which defines the relative entropy from distribution Q to distribution P as:

$$D_{KL}(P\|Q) = \sum_{x_i \in \mathcal{X}} P(x_i) \log \left(\frac{P(x_i)}{Q(x_i)} \right) \tag{6}$$

If $P(x_i) = 0$ for some i, the contribution of the i-th term is assumed to be null.

The Kullback–Leibler distance quantifies how much information is lost when approximating P using Q or, alternatively, how much information is gained by refining Q into P. In the Hanabi game, we are working with probability distributions over the domain of card identities, which has size 25 (5 colours × 5 ranks). Therefore, for all our computations we take the logarithm in Eq. (6) with base 25.

We evaluate the gain provided by the abductive explanations by comparing the distance to the ground truth between the post-action and the post-explanation distributions at every game turn. Since the ground truth corresponds to a single card identity with probability 1, the Kullback–Leibler distance from the two aforementioned distributions to the ground truth is reduced to:

$$D_{KL}(\mathbb{P}_S^* \| \mathbb{P}_S^{\{\cdot\}}) = -\log \left(\mathbb{P}_S^{\{\cdot\}}(\mathcal{C}^S, \mathcal{R}^S) \right) \tag{7}$$

The results for the percentage reduction in distance between the post-action and the post-explanation distribution to the ground truth appear in Table 2. The reduction in distance is large across all teams sizes, starting at around 85% for teams of 2 players, and increasing monotonically with team size up to a 91% for teams of 5 players.

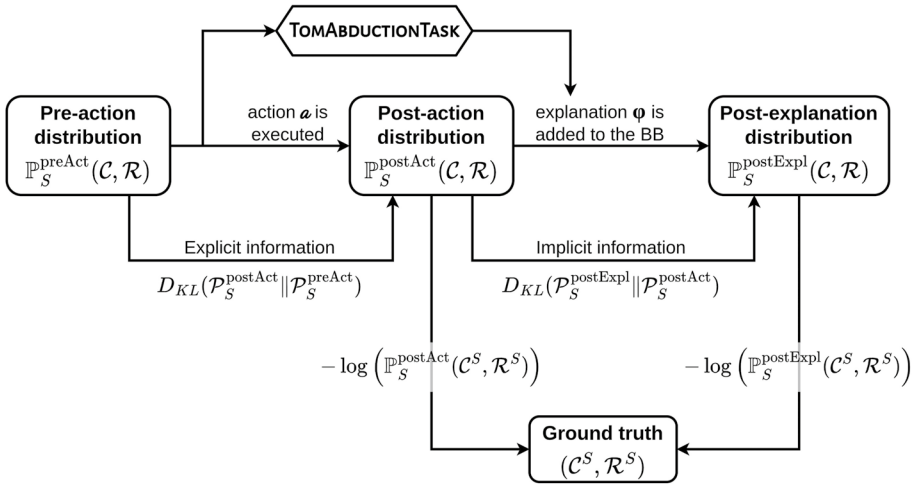


Fig. 7 Outline of the probabilistic analysis of the simulation results

Table 2 Reduction is distance to the ground truth from the post-action to the post-explanation distribution

Num. players	%
2	85.33
3	88.29
4	89.43
5	91.49

5.5 Information value

The previous results indicate that the incorporation of abductive explanations does shrink the distance to the ground truth to a very large extent. However, the analysis does not indicate how *valuable* the information derived from these abductive explanations is. In other words, how much score are agents able to draw from the information provided by abductive explanations.

To quantify the score value of abductive explanations, we define the two following quantities (see Fig. 7). First, we define the *explicit information gain* as the Kullback-Leibler distance from the pre-action distribution to the post-action distribution. Second, we define the *implicit information gain* as the Kullback–Leibler distance from the post-action distribution to the post-explanation distribution. The explicit information gain quantifies the knowledge acquired just from observing the progress of the game, as new cards are drawn and revealed. Meanwhile, the implicit information gain quantifies the knowledge derived only from the abductive explanations.

Next, we define the *total explicit information gain* (TEIG) as the sum across all slots S and moves m_i over the course of a game of the explicit information gain:

Table 3 Average percentage of implicit information and average score assigned to this implicit information

Num. players	% implicit info.	% implicit score
2	15.15	27.80
3	15.51	30.55
4	18.79	32.24
5	19.40	38.37

$$\text{TEIG} = \sum_{m_i} \sum_{S} D_{KL}(\mathbb{P}_S^{\text{postAct}} \| \mathbb{P}_S^{\text{preAct}}) \tag{8}$$

The *total implicit information gain* (TIIG) is defined analogously to Eq. (8), but using the distance from the post-action to the post-explanation distribution, $D_{KL}(\mathbb{P}_S^{\text{postExpl}} \| \mathbb{P}_S^{\text{postAct}})$, instead. The TEIG is defined for all games, regardless of whether agents are using the TomAbductionTask function. The TIIG is defined only for games where the mentioned function is active. For these games, we compute the percentage of information that is derived from the ToM and abduction capabilities as:

$$\%\,\text{implicit info.} = \frac{\text{TIIG}}{\text{TIIG} + \text{TEIG}} \cdot 100 \tag{9}$$

Then, to quantify the contribution of each type of information to score, we start by computing the *explicit score rate* (ESR) as the ratio of the score to the TEIG, for games where agents are *not* using the TomAbductionTask function. Once we have the ESR, we turn to games where agents *are* using this function, and we estimate the residual score that cannot be explained away by the explicit information that agents acquire by observing the evolution of the system as:

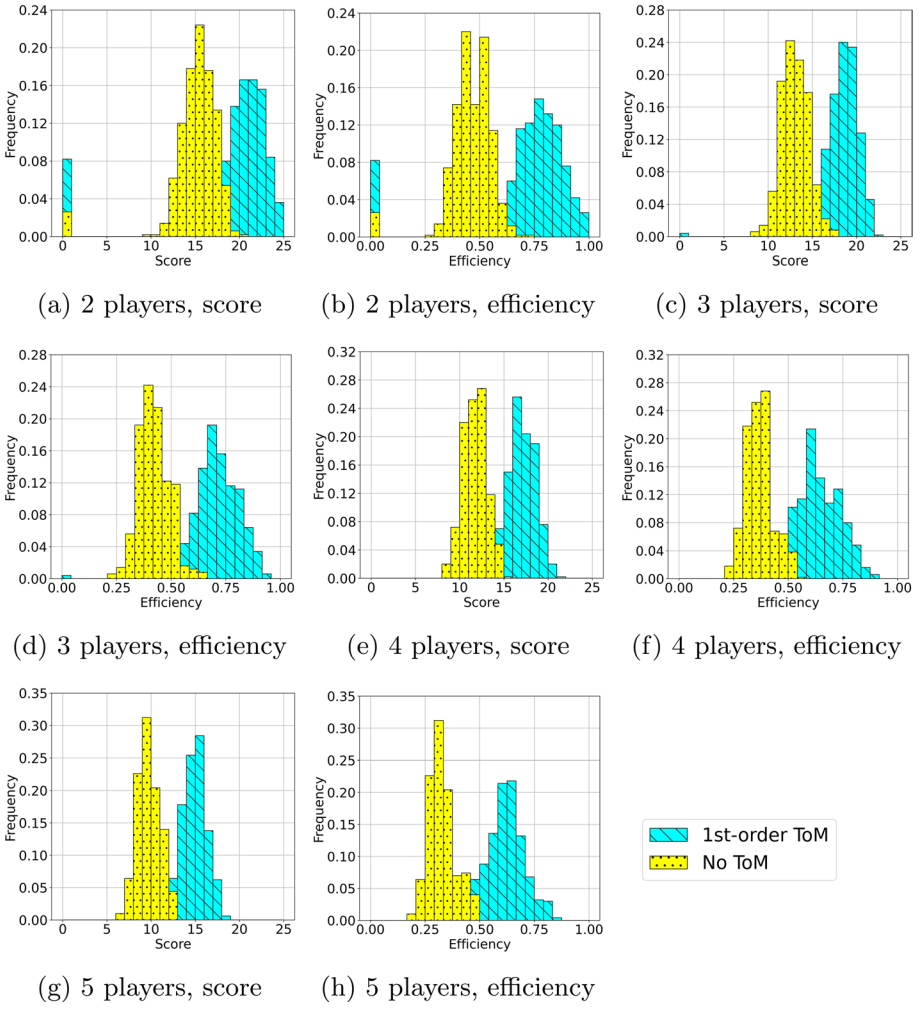
$$\text{residual score} = \text{score} - \text{ESR}_{\text{seed}} \cdot \text{TEIG} \tag{10}$$

where ESR_{seed} is the ESR for the game without calls to TomAbductionTask with the same random seed, and TEIG (TIIG) is the total explicit (implicit) information gain for the game that employs the TomAbductionTask function. We use the ratio between the residual score in Eq. (10) and the total score as the estimation of the contribution of the implicit information to the overall performance of the team.

The results for the average percentage of implicit information and the average percentage score that can be assigned to this explicit information appear in Table 3, for games where agents use the TomAbductionTask function. Across all team sizes, the information derived from the ToM and abduction capabilities accounts for between 15% and 20% of the total information. However, this implicit information accounts for disproportionate amount of the final score, between 27% and 40% of it. Therefore, when agents use the capabilities of the TomAbd model, the information derived from these capabilities ends up being overrepresented in the final score by a factor of between $\times 1.7$ and $\times 2.0$.

6 Related work

This section compares our contribution with related approaches. Previous work on Theory of Mind implementations in agent-oriented programming have, for the most part, used languages based on the BDI architecture [29, 30, 56, 57]. This is a natural choice that we share,



(a) 2 players, score (b) 2 players, efficiency (c) 3 players, score

(d) 3 players, efficiency (e) 4 players, score (f) 4 players, efficiency

(g) 5 players, score (h) 5 players, efficiency

Fig. 8 Experimental distributions of the absolute score and the communication efficiency. Cyan ruled boxes correspond to games where agents make use of the capabilities of the TomAbd agent model, and yellow dotted boxes correspond to games where they do not. The Shapiro-Wilk test confirmed that all experimental histograms follow a normal distribution (Color figure online)

since BDI-based languages provide constructs for the mental states that ToM estimates and operates on. Specifically, some ToM implementations are, like our TomAbd agent model, developed in Jason [29, 30, 56]. In contrast, other work uses Extended 2APL [57].

Panisson et al. [30] implement ToM for deceptive purposes. They focus on the communicative interventions, i.e. the requesting and sharing of (possibly untruthful) information, and provide operational semantics [56] for the effects that these actions have of the models that agents maintain of one another. Their approach is very much in line with the TT account of ToM. It uses dedicated predicates to infer additional mental states, such as goals and future actions, given prior beliefs. These inferences are made from within the agent

program, a feature which we consider qualifies as adherence to the theoretical version of ToM.

Sarkadi et al. [29] extend the previous model by incorporating elements of trust and modelling several agent profiles based on their attitudes. In this extension, they distinguish between TT and ST components within their model. They argue that the TT component handles the assignment of prior beliefs to other agents, while the ST component handles inferences based on those. In our work, we do not distinguish between TT and ST components, but consider that our approach overall aligns more closely with the ST account of ToM than with the TT account.

Harbers et al. [57] establish a different criterion for classifying ToM approaches into TT and ST. They develop two separate ToM implementations, one identified with TT and the other with ST, for applications in virtual training systems. Both architectures maintain knowledge bases for the beliefs, logical rules and goals of other agents. The difference between the ST and TT approaches is found in the reasoner that is applied to the knowledge bases assigned to other agents. The TT architecture applies rules about how other agents combine their beliefs, goals and plans, which are explicitly included as part of the agent's own knowledge. In contrast, the ST architecture uses the agent's native reasoner, making it more lightweight. Besides this, other advantages were found for the ST architecture with respect to the TT one, namely code reusability and flexibility to deal with non-BDI agents.

The choice to maintain belief bases for other agents, in addition to the agent's own belief base, is very different to the TomAbd agent model, where we generate estimations of the beliefs of others on demand at run-time, using the set of ToM rules as a meta-interpreter. This allows the TomAbd model to engage in higher-order ToM by recursively applying the set of ToM rules. In comparison, the maintenance of belief bases for other agents hinders the use of ToM beyond first-order. For every recursive path that the agent would like to take into account, i.e. what we refer to as the *viewpoint* in Definition 3, a different knowledge base would have to be initialised and updated throughout the agent's lifetime, resulting in a rapid combinatorial explosion in memory requirements. This limitation to first-order ToM is also shared by other work [29, 30].

There is an important difference in the focus of ToM between the works reviewed in this section and the TomAbd agent model of this paper. In related work [29, 30, 56, 57], the purpose of the ToM functionalities is to compute the action that best pursues the agent's goal, whether it is to deceive an opponent or to provide explanation to assist in staff training. Hence, ToM is directed towards the *deliberation* step of the BDI reasoning cycle. In contrast, in our approach, ToM is directed towards the *sensing* step, with the TomAbd model computationally implementing the cognitive processes to use other agents as sensors. Accordingly, the core function of the TomAbd agent model is TomAbductionTask, which uses abduction to compute explanations either about the state of the environment or the mental state of other agents. The execution of this function results in the agent being in a more informed position when it comes to its own decision-making.

It should be noted that, even if at this current stage the TomAbd agent model strongly links ToM with sensing, it provides an avenue to include these capabilities into the agent's deliberation stage too. The component directed towards practical reasoning, the SelectAction function, has not thus far received as much attention as TomAbductionTask. Nonetheless, as mentioned previously, this function is a *customisable* component of the model. This leaves a lot of room to develop further implementations that more explicitly use the ToM capabilities of the agent during the deliberation stage, for example by making calls to the AdoptPerspective procedure within SelectAction.

To summarise, the publications reviewed so far orient the ToM capabilities of agents towards deliberation. Nonetheless, work by Sindlar et al. [58], similarly to ours in its goals, focuses on mental state abduction, i.e. the inference of beliefs and goals of BDI agents given a sequence of observed actions. They use the APL agent programming language, where an agent is composed, among others, of goals achievement rules, analogous to Jason plans. An agent program and its observed actions are translated into an Answer Set Programming (ASP) program, which is then resolved with an off-the-shelf ASP solver. The authors argue that the ToM capabilities provided through this mode of reasoning have potential to enhance the social awareness and credibility of non-player characters in role-playing games [59].

In contrast to our current work, the approach by Sindlar et al. is restricted to first-order ToM and, in our opinion, leans heavily towards the theoretical account of ToM (TT), presented in Sect. 2.1. Finally, compared with the work we cite previously (where ToM is oriented towards action selection), Sindlar et al. do not offer details about how the obtained explanations are integrated into the abducing agent's own knowledge or decision-making.

7 Conclusions

In this paper, we have presented the novel TomAbd model, an agent architecture combining Theory of Mind and abductive reasoning. Its main functionality is the ability to perceive the state of the system through the eyes of their peers, and infer the beliefs that account for their most recent action using abductive reasoning. This core functionality is accompanied by other functions that handle how the abductive explanations are refined, updated and used during practical reasoning.

There are four features that make the TomAbd agent model stand out. First, the model is able to handle ToM of an arbitrary order without additional memory requirements. Second, our approach has a strong preference for a simulation account of ToM over a theory account. Third, we emphasise the role of ToM for sensing over deliberation. The goal of ToM in our model is to extract the information as perceived by other agents, hence using them as proxies for obtaining data about the world. Finally, we would like to highlight the user-friendliness and the flexibility of our implementation, which allows customisation of many of its components.

We have tested our model in the benchmark domain of Hanabi. Our results show that teams whose agents use ToM consistently perform better than those that do not, both in terms of absolute score and efficiency of communication. In terms of information gain, our analysis shows that the knowledge derived from the abductive reasoning component of the model greatly reduces uncertainty. Additionally, the information derived from the combination of ToM and abductive reasoning contributes to the final score in a disproportionate amount, with respect to the explicit information derived from the observation of the evolution of the game alone.

The TomAbd agent model presented here offers several directions for future work. First, within the Hanabi game domain, an option would be to investigate more sophisticated action selection rules. Additionally, it would be interesting to investigate the perception that human players have of TomAbd teammates, for strategies of different skill levels. This research could shine light on how well is human ToM captured by the agents, and how compatible is human ToM and the artificial ToM we have presented here.

Second, the TomAbd agent model can be applied to other domains where ToM capabilities may entail a potential benefit, with the goal of extracting the common general features that a domain must have in order for ToM to result in improved performance. This research could also expand the set of customised functions for explanation revision and update, as well as the incorporation of ToM in the deliberation stage. Furthermore, the application to other domains would require the development of additional metrics to quantify the benefits entailed by the agents' ToM capabilities, analogous to the information gain and information value metrics we present in this paper for Hanabi. Such metrics would naturally need to consider the domain properties such as whether there is a closed set of states and/or any heuristics available to quantify the value of MAS states.

Third, the flexibility of the TomAbd model could be enhanced by extending the type of constructs for which the TomAbd agent model is able to provide explanations. In other words, with small additional functionality, TomAbd agents could be adapted to compute abductive explanations not just for actions, but for mental states such as beliefs, goals and intentions. Of course, the mental state that is taken as input to the machinery of the TomAbd agent model must either be the result of some observation (e.g. agent *i* overhears agent *j* discuss its goals with a third party), or of other techniques, such as goal recognition, that aggregate granular observations into a mental state, i.e. a sequence of atomic actions into the goal or intention pursued by those actions.

Regardless of the modality of the observation, the process of generating an explanation for it would be analogous to that presented in the TomAbd agent model for actions. In summary, as long as some agent *i* has an input about another agent *j* (such as an action *j* has taken, a belief or a desire *j* holds, or an intention *j* is pursuing) and an estimation of the inference rules that *j* is using, *i* can provide an explanation for the input. Of course, its precision will depend on the accuracy of the input and of the inference rules that *i* believes *j* to have.

Last but not least, the computational requirements versus the performance benefits of using higher-order Theory of Mind, in the Hanabi game or in other domains, presents an interesting challenge. Note that, in the TomAbd agent model, the same mechanism that enables an agent to use first-order ToM also enables it to use ToM of any order (i.e. querying the `believes(Ag, Fact)` clauses and substituting the contents of its belief base), hence the ToM level that a TomAbd agent uses is, by construction, unbounded. Here too, many questions arise. For example, does performance plateau around a particular recursion level n_{pl}? Is n_{pl} a domain-independent quantity? How does it compare with respect to the maximum order of ToM that humans usually apply? Does this have any evolutionary implications? In other words, did humans develop ToM just far enough to obtain the maximum evolutionary advantage, but not any further to save resources?

To conclude, our work presents and tests a novel model for agents with Theory of Mind. It provides the cognitive machinery to adopt the perspective of a peer and reason from its perspective. It is inspired by the though processes that humans engage in when trying to understand the motivations for the behaviour of others. Our model endows autonomous agents with essential social abilities, that are becoming increasingly important in the current AI landscape (Fig. 8).

Acknowledgements N. Montes, N. Osman and C. Sierra would like to thank the TAILOR connectivity fund for funding their research visit to KCL.

Author Contributions N. Montes wrote the main manuscript text. All authors reviewed the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. N. Montes, N. Osmain and C. Sierra acknowledge funding from the Spanish funded VAE project

(#TED2021-131295B-C31), the EU VALAWAI project (HORIZON #101070930), and the EU TAILOR project (H2020 #952215).

Availability of data and materials All the code accompanying this work can be freely accessed https://github.com/nmontesg/tomabd. All the simulation data generated is freely available https://drive.google.com/file/d/1W66eZD-t-5YeQ4M4SHmdBuX60wRu1e0k/view?usp=sharing.

Declarations

Conflict of interest The authors declare that they have no competing interests that could have influence this work.

References

1. Dafoe, A., Bachrach, Y., Hadfield, G., Horvitz, E., Larson, K., & Graepel, T. (2021). Cooperative AI: machines must learn to find common ground. *Nature, 593*(7857), 33–36. https://doi.org/10.1038/d41586-021-01170-0
2. Paiva, A., et al. (2020). WP6 – Social AI: Learning and reasoning in social contexts. ICT-48 TAILOR: Foundations of Trustworthy AI - Integrating Reasoning, Learning and Optimization. https://www.tailor-social-ai.eu/home
3. Malle, B. (2022). In R. Biswas-Diener & E. Diener (Eds.), *Theory of mind.* Champagne, IL: DEF publishers.
4. Knobe, J. (2005). Theory of mind and moral cognition: Exploring the connections. *Trends in Cognitive Sciences, 9*(8), 357–359. https://doi.org/10.1016/j.tics.2005.06.011
5. Williams, J., Fiore, S. M., & Jentsch, F. (2022). Supporting artificial social intelligence with theory of mind. *Frontiers in Artificial Intelligence*. https://doi.org/10.3389/frai.2022.750763
6. Montes, N., Osman, N., & Sierra, C. (2022). Combining theory of mind and abduction for cooperation under imperfect information. arXiv:2209.15279 [cs.MA].
7. Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). *Programming multi-agent systems in agent speak using Jason*. New York, NY: Wiley.
8. Frith, C., & Frith, U. (2005). Theory of mind. *Current Biology, 15*(17), 644–645. https://doi.org/10.1016/j.cub.2005.08.041
9. Korkiakangas, T., Dindar, K., Laitila, A., & Kärnä, E. (2016). The Sally-Anne test: An interactional analysis of a dyadic assessment. *International Journal of Language & Communication Disorders, 51*(6), 685–702. https://doi.org/10.1111/1460-6984.12240
10. Baron-Cohen, S., Leslie, A. M., & Frith, U. (1985). Does the autistic child have a "theory of mind'' ? *Cognition, 21*(1), 37–46. https://doi.org/10.1016/0010-0277(85)90022-8
11. Tager-Flusberg, H. (2007). Evaluating the theory-of-mind hypothesis of autism. *Current Directions in Psychological Science, 16*(6), 311–315. https://doi.org/10.1111/j.1467-8721.2007.00527.x
12. Askham, A. V. (2022). Theory of mind in autism: A research field reborn. Spectrum. https://doi.org/10.53053/gxnc7576.
13. Röska-Hardy, L. (2008). Theory theory (simulation theory, theory of mind). In *Encyclopedia of neuroscience* (pp. 4064–4067). Berlin: Springer. https://doi.org/10.1007/978-3-540-29678-2_5984.
14. van der Hoek, W. (1993). Systems for knowledge and belief. *Journal of Logic and Computation, 3*(2), 173–195. https://doi.org/10.1093/logcom/3.2.173
15. Rendsvig, R., & Symons, J. (2021). Epistemic Logic. In Zalta, E.N. (ed.) *The stanford encyclopedia of philosophy*, Summer 2021 edn. Metaphysics Research Lab, Stanford University, Stanford, CA.

16. Meyer, J.-J.C., Broersen, J., & Herzig, A. (2015). BDI logics. In van Ditmarsch, H., Halpern, J.Y., van der Hoek, W. (eds.) *Handbook of epistemic logics*. College Publications, Rickmansworth, WD3 1DE. Chap. 10.
17. Corballis, M. (2007). The uniqueness of human recursive thinking. *American Scientist, 95*(3), 240. https://doi.org/10.1511/2007.65.240
18. Corballis, M. C. (2011). *The Recursive Mind: The Origins of Human Language, Thought, and Civilization* (p. 291). Princeton, NJ: Princeton University Press.
19. Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence, 258*, 66–95. https://doi.org/10.1016/j.artint.2018.01.002
20. Baarslag, T., Hendrikx, M. J. C., Hindriks, K. V., & Jonker, C. M. (2015). Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems, 30*(5), 849–898. https://doi.org/10.1007/s10458-015-9309-1
21. Nashed, S., & Zilberstein, S. (2022). A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research, 73*, 277–327. https://doi.org/10.1613/jair.1.12889
22. Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S.M.A., & Botvinick, M. (2018). Machine theory of mind. In Dy, J., Krause, A. (eds.) *Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research* (Vol. 80, pp. 4218–4227). PMLR, Stockholm, Sweden. https://proceedings.mlr.press/v80/rabinowitz18a.html.
23. Wang, Y., Zhong, F., Xu, J., & Wang, Y. (2022). Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind. In *International conference on learning representations*. https://openreview.net/forum?id=M3tw78MH1Bk.
24. Jara-Ettinger, J. (2019). Theory of mind as inverse reinforcement learning. *Current Opinion in Behavioral Sciences, 29*, 105–110. https://doi.org/10.1016/j.cobeha.2019.04.010. Artificial Intelligence.
25. Cuzzolin, F., Morelli, A., Cîrstea, B., & Sahakian, B. J. (2020). Knowing me, knowing you: Theory of mind in AI. *Psychological Medicine, 50*(7), 1057–1061. https://doi.org/10.1017/s0033291720000835
26. de Weerd, H., & Verheij, B. (2011). The advantage of higher-order theory of mind in the game of limited bidding. In *Workshop on reasoning about other minds: Logical and cognitive perspectives* (Vol. 751, pp. 149–164).
27. de Weerd, H., Verbrugge, R., & Verheij, B. (2012). Higher-order social cognition in the game of rock-paper-scissors: A simulation study. In Bonanno, G., Van Ditmarsch, H., Hoek, W. (eds.) *Proceedings of the 10th conference on logic and the foundations of game and decision theory (LOFT 2012)* (pp. 218–232).
28. de Weerd, H., Verbrugge, R., & Verheij, B. (2022). Higher-order theory of mind is especially useful in unpredictable negotiations. *Autonomous Agents and Multi-Agent Systems*. https://doi.org/10.1007/s10458-022-09558-6
29. Ştefan Sarkadi, Panisson, A.R., Bordini, R.H., McBurney, P., Parsons, S., & Chapman, M. (2019). Modelling deception using theory of mind in multi-agent systems. *AI Communications* 32, 287–302. https://doi.org/10.3233/AIC-190615.
30. Panisson, A., Mcburney, P., Parsons, S., Bordini, R., & Sarkadi, S. (2018). Lies, bullshit, and deception in agent-oriented programming languages. In *Proceedings of the 20th international trust workshop co-located with AAMAS/IJCAI/ECAI/ICML (AAMAS/IJCAI/ECAI/ICML* 2018).
31. Walton, D. (2014). *Abductive reasoning* (p. 320). Tuscaloosa, AL: University of Alabama Press.
32. Josephson, J. R., & Josephson, S. G. (1994). *Abductive inference: Computation, philosophy, technology* (p. 316). Cambridge, UK: Cambridge University Press.
33. Flach, P.A., & Kakas, A.C. (eds.): Abduction and induction: Essays on their relation and integration. Berlin: Springer (2000). https://doi.org/10.1007/978-94-017-0606-3.
34. Kakas, A., Kowalski, R., & Toni, F. (1993). Abductive logic programming. *Journal of Logic and Computation, 2*(6), 719–770. https://doi.org/10.1093/logcom/2.6.719
35. Denecker, M., & Kakas, A.C. (2002). Abduction in logic programming. In *Computational logic: Logic programming and beyond, essays in Honour of Robert A. Kowalski, Part I* (pp. 402–436). Berlin: Springer.
36. Denecker, M., & de Schreye, D. (1998). Sldnfa: An abductive procedure for abductive logic programs. *The Journal of Logic Programming, 34*(2), 111–167. https://doi.org/10.1016/S0743-1066(97)00074-5
37. Ray, O., & Kakas, A. (2006). Prologica: a practical system for abductive logic programming. In *Proceedings of the 11th International workshop on non-monotonic reasoning* (pp. 304–312).
38. Fung, T. H., & Kowalski, R. (1997). The iff proof procedure for abductive logic programming. *The Journal of Logic Programming, 33*(2), 151–165. https://doi.org/10.1016/S0743-1066(97)00026-5

39. Sadri, F., & Toni, F. (2000). Abduction with negation as failure for active and reactive rules. In E. Lamma & P. Mello (Eds.), *AI*IA 99: Advances in artificial intelligence* (pp. 49–60). Berlin: Springer.
40. Azzolini, D., Bellodi, E., Ferilli, S., Riguzzi, F., & Zese, R. (2022). Abduction with probabilistic logic programming under the distribution semantics. *International Journal of Approximate Reasoning, 142*, 41–63. https://doi.org/10.1016/j.ijar.2021.11.003
41. Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., Dunning, I., Mourad, S., Larochelle, H., Bellemare, M. G., & Bowling, M. (2020). The hanabi challenge: A new frontier for AI research. *Artificial Intelligence, 280*, 103216. https://doi.org/10.1016/j.artint.2019.103216
42. Siu, H.C., Peña, J.D., Chang, K.C., Chen, E., Zhou, Y., Lopez, V.J., Palko, K., & Allen, R.E. (2021). Evaluation of human-ai teams for learned and rule-based agents in hanabi. CoRR arXiv:2107.07630.
43. O'Dwyer, A. (2017). Quuxplusone/Hanabi: Framework for writing bots that play Hanabi. https://github.com/Quuxplusone/Hanabi/.
44. Osawa, H. (2015). Solving hanabi: Estimating hands by opponent's actions in cooperative game with incomplete information. In *AAAI workshop: Computer poker and imperfect information*http://aaai.org/ocs/index.php/WS/AAAIW15/paper/view/10167.
45. Cox, C., Silva, J. D., Deorsey, P., Kenter, F. H. J., Retter, T., & Tobin, J. (2015). How to make the perfect fireworks display: Two strategies for Hanabi. *Mathematics Magazine, 88*(5), 323–336. https://doi.org/10.4169/math.mag.88.5.323
46. van den Bergh, M. J. H., Hommelberg, A., Kosters, W. A., & Spieksma, F. M. (2017). Aspects of the cooperative card game hanabi. In T. Bosse & B. Bredeweg (Eds.), *BNAIC 2016: Artificial Intelligence* (pp. 93–105). Cham: Springer.
47. Walton-Rivers, J., Williams, P. R., Bartle, R., Perez-Liebana, D., & Lucas, S. M. (2017). Evaluating and modelling hanabi-playing agents. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 1382–1389). https://doi.org/10.1109/CEC.2017.7969465.
48. Hu, H., Lerer, A., Peysakhovich, A., & Foerster, J. (2020). "Other-play" for zero-shot coordination. In III, H.D., Singh, A. (eds.) *Proceedings of the 37th international conference on machine learning. proceedings of machine learning research* (Vol. 119, pp. 4399–4410). PMLR, Virtual event. https://proceedings.mlr.press/v119/hu20a.html.
49. Lerer, A., Hu, H., Foerster, J., & Brown, N. (2020). Improving policies via search in cooperative partially observable games. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, pp. 7187–7194). https://doi.org/10.1609/aaai.v34i05.6208.
50. Foerster, J., Song, F., Hughes, E., Burch, N., Dunning, I., Whiteson, S., Botvinick, M., & Bowling, M. (2019). Bayesian action decoder for deep multi-agent reinforcement learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) *Proceedings of the 36th international conference on machine learning. Proceedings of machine learning research* (Vol. 97, pp. 1942–1951). PMLR, Long Beach, CA. https://proceedings.mlr.press/v97/foerster19a.html.
51. Sarmasi, A., Zhang, T., Cheng, C.-H., Pham, H., Zhou, X., Nguyen, D., Shekdar, S., & McCoy, J. (2021). Hoad: The hanabi open agent dataset. In *Proceedings of the 20th international conference on autonomous agents and multiagent systems. AAMAS '21* (pp. 1646–1648). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.
52. Rao, A.S. (1996). AgentSpeak(l): BDI agents speak out in a logical computable language. In *Lecture notes in computer science* (pp. 42–55). Berlin: Springer. https://doi.org/10.1007/bfb0031845.
53. Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika, 52*(3–4), 591–611. https://doi.org/10.1093/biomet/52.3-4.591
54. Ross, A., & Willson, V. L. (2017). Paired samples T-Test, pp. 17–19. SensePublishers, Rotterdam. https://doi.org/10.1007/978-94-6351-086-8_4.
55. Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics, 22*(1), 79–86. https://doi.org/10.1214/aoms/1177729694
56. Panisson, A. R., Ştefan Sarkadi, McBurney, P., Parsons, S., & Bordini, R. H. (2019). On the formal semantics of theory of mind in agent communication. In *Agreement technologies* (pp. 18–32). Springer, Cham. https://doi.org/10.1007/978-3-030-17294-7_2.
57. Harbers, M., Bosch, K.V.d., & Meyer, J.-J. (2009). Modeling agents with a theory of mind. In *2009 IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology* (Vol. 2, pp. 217–224). https://doi.org/10.1109/WI-IAT.2009.153.
58. Sindlar, M., Dastani, M., & Meyer, J.-J. (2011). Programming mental state abduction. In *The 10th international conference on autonomous agents and multiagent systems - Volume 1. AAMAS '11* (pp. 301–308). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.

59. Sindlar, M. P., Dastani, M. M., & Meyer, J.-J.C. (2009). Bdi-based development of virtual characters with a theory of mind. In Z. Ruttkay, M. Kipp, A. Nijholt, & H. H. Vilhjálmsson (Eds.), *Intelligent Virtual Agents* (pp. 34–41). Berlin: Springer.

Part III

Closing Remarks

Chapter 3

Integrating the Three Approaches

In Part II, a variety of contributions are made: a methodology for the automated synthesis and analysis of norms based on their degree of value alignment (Contribution 1); a rich framework for representating and automatically interpreting norms to examine the outcomes incentivised by them (Contribution 2); and a ToM-endowed agent model with the ability to adopt and reason from the perspective of other agents (Contribution 3). In this chapter, we present the integration of these three approaches into an agent functionality for computing the degree of alignment of a set of norms with respect to a value (or set of values) not just from their own perspective, but also from the perspective of other agents, thus incorporating the strong social dimension of values discussed in Section 1.1.

3.1 An Integrated Approach to Value Engineering

The goal of this chapter is to integrate the contributions made in Part II into a novel agent functionality for computing the alignment of a set of norms (or normative system) with respect to a value of choice from any perspective, i.e. to reason about the alignment of norms not just from the agent's own value perspective, but also that of other agents. An agent endowed with such functionality will be able to perform value-alignment computations from an eminently *social orientation*. Formally, an agent α should not just be able to compute:

$$\mathsf{Algn}^{\alpha}_{N,v} \tag{3.1}$$

which denotes the alignment Algn of the norms N with respect to value v[1] from the perspective of agent α itself. Agent α should also be able to compute:

$$\mathsf{Algn}^{\alpha,\beta}_{N,v} \tag{3.2}$$

which denotes the alignment Algn that *agent α estimates that agent β has* for the set of norms in N with respect to value v. That is, Equation (3.2) denotes the alignment that α computes from the perspective of β. Therefore, in order to compute $\mathsf{Algn}^{\alpha,\beta}_{N,v}$, agent α will have to use first-order ToM capabilities (like those provided by Contribution 3) to change its perception on the proposed norms N to an estimation of β's perception of them. While the computation of the alignment from one's own perspective (i.e. $\mathsf{Algn}^{\alpha}_{N,v}$) does not require, in principle, such perspective switching capabilities, they become necessary once agents need to estimate the opinion of their peers.

Furthermore, Equation (3.2) can be extended to an arbitrary level of recursion (analogously to Equation (1) in Contribution 3):

$$\mathsf{Algn}^{P}_{N,v} \text{ where } P = \alpha, \beta, \ldots, \gamma, \delta \tag{3.3}$$

which denotes the alignment for norms in N with respect to value v from the perspective of P,[2] i.e. the alignment that α estimates that β estimates that ... γ estimates that δ has. Thus, just as the computation of Equation (3.2) required first-order ToM capabilities, to compute Equation (3.3) agent α must be endowed with *n-th order* ToM capabilities (again, like those provided by Contribution 3). The reader should keep in mind that the agent actually performing the computation of Equation (3.3) is α, i.e. the head element of P.

To compute Equation (3.1), we select the most important features of the contributions in Part II and integrate them into a new agent functionality, namely the computation of the *alignment of a set of norms with respect to a value from any perspective*. Mainly, the features selected from Part II are:

- **From Contribution 1**: The computation of value alignment as an expectation over outcome states of the semantics function that grounds the meaning of the value of interest in a particular context.

- **From Contribution 2**: The rich norm representation language (i.e. the

[1] The discussion on alignment with respect to *a value v* throughout this chapter can be trivially extended to the alignment with respect to the *aggregation of values in set V*.

[2] See Definition 3 in Contribution 3.

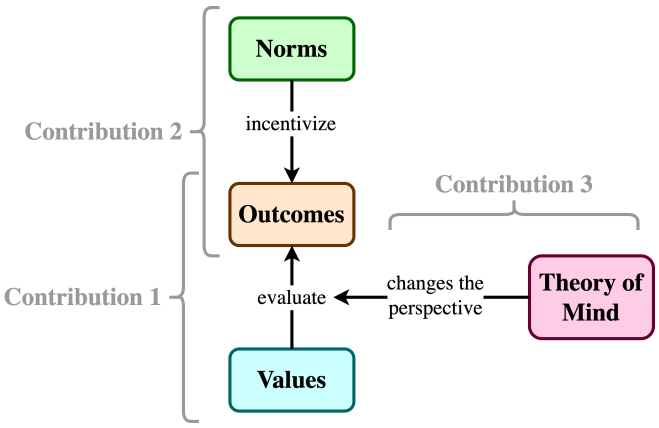


Figure 3.1: Integration overview.

Action Situation Language, ASL) and its accompanying game engine to interpret ASL descriptions. Together with a game theoretical solution concept, the probability distribution over outcome states (i.e. the set of terminal nodes in the resulting EFG tree) is automatically predicted.

- **From Contribution 3**: The n-th order ToM capabilities of the ToMAbd agent model, i.e. its ability to switch its view by building an estimation of the perspective that other agents have, and in general to perform such perspective switches down to an arbitrary level of recursion.

Figure 3.1 presents the integration of the three contributions of Part II as a diagram. Contribution 1 provides the grounding of values as a set of semantics functions that evaluate outcomes, while Contribution 3 changes the perspective that the agent has while doing this evaluation. Meanwhile, Contribution 2 provides a framework to predict the outcomes based on the set of norms in place.

Note that the integration proposed in this chapter materialises as a new agent functionality, and hence it is an *agent-centred* integration. While the agent model developed in Contribution 3 is naturally agent-centred, Contribution 1 and Contribution 2 correspond to general frameworks that are originally applied from the perspective of a social planner or a community of agents at large. Therefore, they need to be adapted such that their functionalities can be employed from within an agent's operation.

The main application that is envisioned for the agent functionality proposed here is value-guided automated negotiation over normative systems. There, two or more autonomous agents bargain over the set of norms to implement in the domain where they are interacting, based on the degree of alignment of

those norms with respect to their values. In order to make proposals that stand a chance of being accepted, an agent must be able to view proposals for normative systems in terms of their alignment with respect to several values from its own as well as the perspective of others, thus requiring ToM capabilities.

An interesting perspective on the process of negotiating over norms mentioned above is as a value-aggregating process. Agents come in to the bargain equipped with their individual values, which might be very diverse both on their meaning and priorities. From the negotiation process should come out a set of norms that are implemented on the system as a whole and that determine which outcomes are more likely to be reached. The selected norms, then, depend on the value preferences and interpretations that the participating agents have, and merge all of them into a single shared regulative body. However, whether the agreed-upon norms will be more responsive to a subset of agents over another will depend on how the negotiation is set up, and how much power does each individual hold within that process.

3.2 Formal Model

Mathematically, the alignment Algn of a set of norms N with respect to value v from perspective P is a function of the following:

$$\mathsf{Algn}^{P}_{N,v} = F\left(N, v, P \mid G, \mathcal{L}, \mathcal{U}, \mathcal{SC}, \mathfrak{TM}\right) \tag{3.4}$$

where:

- N is the set of *norms* (or normative system) whose alignment is being computed.
- v is the *value* of interest with respect to whom the alignment is being computed.
- P is the *perspective* from which the alignment is being computed. P is an ordered subset of G, where
- $G = \{\alpha, \beta, \gamma, \delta, ...\}$ is the set of *agents* in the system.
- $\mathcal{L}$ is the *logical language* that agent α (i.e. the one computing the alignment, the head element in P, see Equation (3.3)) uses to describe the system, i.e. the set of facts that jointly characterise a state according to α.

- $\mathcal{U}$: $\mathsf{Pow}(\mathcal{L}) \times G \rightarrow \mathbb{R}$ is the *utility assignment function*. Given a state $\mathbf{s}$ described by a set of facts expressed in language $\mathcal{L}$ and an agent $\omega \in G$, $\mathcal{U}(\mathbf{s}, \omega)$ returns the utility that agent ω has for state $\mathbf{s}$. As a reminder, the agent utilities are a set of functions $\{U_g\}_{g \in G}$ where $U_g : Z \leftarrow \mathbb{R}$ and G is the set of players in an Extensive Form Game (EFG) Γ and Z is the set of terminal nodes of the game tree in Γ. We understand these utilities in the classical game-theoretical sense as financial rewards or payoffs.

- $\mathcal{SC}$ is a game theoretical *solution concept*. Given a game model in normal or extensive form, $\mathcal{SC}$ predicts the equilibrium strategies that agents will converge to based on the utilities of the possible outcomes.

- $\mathfrak{TM}$: $\mathsf{Pow}(\mathcal{L}) \times G \rightarrow \mathsf{Pow}(\mathcal{L})$ is the *Theory of Mind function*. Given the current state of α's belief base BB (i.e. a set of facts and clauses expressed in language $\mathcal{L}$) and an agent $\omega \in G$, $\mathfrak{TM}(BB, \omega)$ returns the set of beliefs that α believes that ω has, also expressed in language $\mathcal{L}$.

Equation (3.4) makes a distinction between *arguments* (N, v and P before the "given" sign "|") and *parameters* (G, $\mathcal{L}$, $\mathcal{U}$, $\mathcal{SC}$ and $\mathfrak{TM}$ after the "given" sign "|"). This distinction is not strict, and it is made because any given agent is expected to compute the alignment numerous times for different instantiations of the *arguments*, while the *parameters* remain constant. Consider the value-based norm negotiation application mentioned in Section 3.1. There, a negotiating agent will have to evaluate, for a given domain, several proposals for N with respect to a variety of values and from a number of different perspectives. Meanwhile, the set of agents G, the language $\mathcal{L}$ used to describe the domain, the way that utilities are assigned to outcomes $\mathcal{U}$, the decision-making model $\mathcal{SC}$ and the mapping from one's beliefs to those of others (i.e. the information encoded in $\mathfrak{TM}$) are expected to remain constant.

All arguments in Equation (3.4) can be identified with constructs that have appeared throughout the contributions in Part II. The norms N correspond to the *rule base* Ω in an Action Situation Language (ASL) description, specified in Contribution 2. Value v has the same meaning as the values with respect to which norms are optimised in Contribution 1. The perspective P also has the same meaning as in Contribution 3, and it has been revisited in Section 3.1. The set of agents in G are those declared in the *agents base* Δ of an Action Situation Language (ASL) description. Meanwhile, the logical language $\mathcal{L}$ used to describe the state of the system is identified with the fluents and clauses included in an ASL description, and which are initialized by the `initially/1`

clauses in the *states base* Σ. Finally, the Theory of Mind function $\mathfrak{TM}$ is expressed by the *Theory of Mind clauses* from Contribution 3, which assign beliefs to other agents based on the state of the agent's own belief base.

Two parameters in Equation (3.4), namely the *utility assignment function* $\mathcal{U}$ and the *solution concept* $\mathcal{SC}$, do not have direct equivalents in Part II, although they are mentioned in Contribution 2. The utility assignment function $\mathcal{U}$ is necessary to complete the EFG model Γ built from an ASL description. The game engine developed in Contribution 2 is responsible for automatically querying the ASL description and building the game tree that it entails. However, an EFG is not completely specified until numerical utilities have been assigned to the terminal nodes in the game tree. This is the purpose of the utility assignment function $\mathcal{U}$. In turn, the solution concept $\mathcal{SC}$ leans on the utilities assigned by $\mathcal{U}$ to predict the equilibrium behaviour of agents and the induced probability distribution over outcomes. By default, the solution concept we apply throughout this chapter, similarly to Contribution 2, is the Nash equilibrium coupled with backward induction, i.e. subgame perfect Nash equilibrium.

Equation (3.4) specifies the functional dependencies of $\mathsf{Algn}^{P}_{N,v}$, but not the shape that such function takes. Here, we propose to compute the perspective-dependent alignment as:

$$\mathsf{Algn}^{P}_{N,v} = \sum_{z \in Z} \mathcal{P}(z) \cdot f^{P}_{v}(z) \tag{3.5}$$

where:

- Z is the set of *outcomes*, i.e. the set of terminal nodes in the tree of the EFG Γ, generated from the automated interpretation of an ASL description.
- $\mathcal{P} : Z \to [0, 1]$ is the probability distribution over outcomes induced by the utility assignment function $\mathcal{U}$ through the solution concept $\mathcal{SC}$ applied to Γ, together with any stochastic effects in Γ.
- $f_v : \mathsf{Pow}(\mathcal{L}) \to [-1, 1]$ is the *semantics function of value* v, as defined in Contribution 1. This function indicates whether a state, and a *terminal* state in particular, strongly adheres to (+1), is neutral ($\sim$ 0) or strongly opposes (-1) value v. Then, f^{P}_{v} denotes the semantics function of value v evaluated from perspective P. Further details on invoking the semantics function of a value from different perspectives are provided in Section 3.3.

It should be noted that Equation (3.5) is one proposal, and that other pos-

sibilities exist. For example, Equation (3.5) weights every outcome solely by its probability $\mathcal{P}(z)$. However, in domains where non-aligned outcomes are particularly detrimental and should be avoided, one may want to assign a large negative weight to outcomes where $f_v^P(z) \sim -1$. In our proposal, we are not biased towards or against outcomes based on their evaluation of the semantics function f_v, and simply weight every outcome by the probability of it being realised.

It should be noted that, in our proposal, the utilities obtained from the utility assignment function $\mathcal{U}$ and the solution concept $\mathcal{SC}$ are fixed for the agent and will not be part of its ToM capabilities. In other words, once the agent has interpreted an ASL description using the game engine from Contribution 2, it assigns the utilities and predicts the equilibrium strategies with its own $\mathcal{U}$ and $\mathcal{SC}$ functions. For the time being, an agent α does not work with its belief of the utility and solution concept of another agent β when adopting its viewpoint, which may yield a different probability distribution $\mathcal{P}$ than the one obtained with α's own $\mathcal{U}$ and $\mathcal{SC}$ functions. Doing so would be equivalent to possessing a ToM for $\mathcal{U}$ and $\mathcal{SC}$ too. In our proposal, the ToM capabilities are reserved for the semantics function f_v, thus the interpretation of value does generally change depending on whose perspective is being adopted. We expose this point in detail next.

3.3 The Role of the Perspective

Now, we discuss in further detail the ways in which the perspective P may affect the alignment of a set of norms with respect to a set of values. For starters, consider the base case where agent α is evaluating a set of norms N with respect to a value v from its own perspective. The resulting $\mathsf{Algn}_{N,v}^{\alpha}$ will depend on (i) the *semantics function* f_v that α has for value v, and (ii) the *data* that f_v takes as input. The semantics functions for the values of interest are provided through clauses in the agent's belief base with the following format:

$$\texttt{value(V, Id, Fv) :-}\ b \tag{3.6}$$

indicating that, under the conditions expressed by the clause body b, a value `V` is respected to degree f_v = `Fv` in the action situation identified by `Id`.

For example, consider value *equality*, whose semantics function were grounded through the Gini index in Contribution 1. This semantics function is expressed

by the following clause:[3]

```
value(equality, Id, Degree) :-
    .findall(X, income(Ag, X), L) &
    gini_index(L, GI) &
    Degree = 1 - 2*GI.
```
(3.7)

where `gini_index` is an auxiliary predicate that takes in a list `L` of income quantities and unifies `GI` with the corresponding Gini index. Hence, when computing the alignment with respect to value *equality* from its own perspective, α will take into consideration the set of `income/2` facts to which α itself has access. Note that we are not assuming that all action situations will include `income/2` literals to describe the state of the system. We are using them just for the sake of example.

Now, suppose that α wants to compute the alignment with respect to value equality from the perspective of β, $\mathsf{Align}^{\alpha,\beta}_{N,v}$. The first possibility is that α believes that β uses the same semantics function in Equation (3.7) to compute alignment with respect to *equality*. In other words, α believes that β has the same interpretation of value *equality* as itself. If this is the case, the following ToM clause is included in α's belief base:

```
believes(beta, EqSemFunc, Id) :-
    .relevant_rules(value(equality, Id, Degree), LR) &
    .member(EqSemFunc, LR).
```
(3.8)

stating that, in the action situation identified by `Id`, the clauses that β has to express the semantics function of value *equality* are those that are already present in α's belief base.

However, the result of computing $\mathsf{Align}^{\alpha,\beta}_{N,v}$ is not, in general, the same as $\mathsf{Align}^{\alpha}_{N,v}$. That is the case because when α adopts the perspective of β, the set of `income/2` facts to which it will have access will be, typically, different. For example, α may believe that β only has access to a subset of the `income/2` facts that α itself knows about. Alternatively, α may build an estimation of other `income/2` facts to which β has access, even if they are not part of α's original belief base and/or they do not accurately reflect β's information.

A second possibility is that α may believe that β grounds the meaning of

[3]The clauses displayed in this section are written in Jason agent code. It follows a syntax very similar to that of Prolog, but using ampersand "&" instead of comma for conjunction. Additionally, Jason built-in predicates are preceded by a dot ".".

value *equality* through a different semantics function. For example, suppose that α believes that β conceives value *equality* as the ratio between the minimum and the maximum incomes it knows about. This is captured by the following ToM clause:

```
believes(
    beta,
    {value(equality, Id, Degree) :-
        .findall(X, income(Ag, X), L) &
        .min(L, Min) &
        .max(L, Max) &
        Ratio = Min / Max &
        Degree = 2*Ratio - 1},
    Id
).
```
(3.9)

Hence, even if α uses the same `income/2` facts to compute the alignment with respect to value *equality* from its own and from β's perspective, the result would generally differ due to the different interpretations of this value that are used for both computations.

In general, computing the alignment from a different perspective will involve a combination of the two possibilities outlined in this section: a different interpretation of a value when the perspective of another agent is adopted, which furthermore takes as input different data that do not correspond to the agent's original belief base.

3.4 Computing the Alignment

To compute Equation (3.5), we present the ALIGNMENT function in Algorithm 1. This function provides an additional internal functionality to symbolic-based agents (such as BDI agents), which are assumed to maintain a belief base BB. As prerequisites, an agent executing the ALIGNMENT function must have access to (i) the game engine from Contribution 2 to automatically interpret ASL descriptions; and (ii) the ToM-related functionalities of the TOMABD agent model from Contribution 3. In particular, an agent must be able to copy its current belief base to a backup and recover it (functions COPYTOBACKUP and RECOVERBACKUP respectively), as well as replacing its current beliefs by an estimation of the beliefs that other agents have (function ADOPTVIEWPOINT). For further details on these functions, see Contribution 3.

To compute the perspective-dependent alignment $\mathsf{Algn}^{P}_{N,v}$, the agent must first have access to some data about the domain under examination. This data corresponds to the parameters in Equation (3.4). First, the agent must have access to the *default* ASL description of the domain upon which a new set of rules is being considered for adoption. This default ASL description is composed of the agents base Δ, the states base Σ, and the *default* rule base Ω_0. This default rule base contains the *default rules* regulating the action situation, i.e. those whose priority is equal to 0. In terms of the value-based negotiation over normative systems scenario introduced in Section 3.1, the set of default rules would correspond to the default outcome of the negotiation (i.e. the result of the negotiating process in case no agreement on another alternative is reached). Also, the action situation under examination is identified with identifier *id*. This is not a crucial input to Algorithm 1 at the moment, and it is rather made to keep consistency with Contribution 2 and with a potential extension to multi-context systems in mind (this point is explained in detail in Section 4.3).

Next, the agent must also take in a utility assignment function $\mathcal{U}$ to set the utilities of an EFG generated from an ASL description. Similarly, it must also have set the solution concept $\mathcal{SC}$ to compute equilibrium strategies in an EFG, and from which the probability distribution over outcomes is induced. Meanwhile, not central to this work is the optional parameter *max*, which can be set to limit the depth of the game tree during the game-building process.

Last, Algorithm 1 also has access to a set of *Theory of Mind clauses* $\mathfrak{TM}$. Overloading the notation, we refer to this set of clauses using the same symbol used as for the ToM function in Equation (3.4). In fact, the clauses mentioned in Algorithm 1 encode the ToM function from Equation (3.4). In other words, the ToM clauses $\mathfrak{TM}$ allow to build an estimation of another agent's belief base, taking as input the current state of the agent's own belief base. The clauses in $\mathfrak{TM}$ have a very similar format as the ToM clauses from Contribution 3:

$$\texttt{believes(Ag, F, Id) :-}\ b \tag{3.10}$$

meaning that agent `Ag` believes in fact `F` in the action situation identified by `Id` if the conditions in body b hold. Compared to Contribution 3, Equation (3.10) only has a small extension in the additional argument `Id` to identify the action situation in which the clause applies. At this stage, this is not a very relevant addition, but it opens the door to future work on autonomous agents that can operate on multi-context systems (again, more details are provided in Section 4.3).

Algorithm 1: Function ALIGNMENT(N, v, P)

Input : N ▷ tuple $\langle \Omega[, thres] \rangle$ where Ω is a set of *higher priority rules* written in ASL, and *thres* is an optional threshold parameter used as a filter (a non-negative integer). By default, use $thres \sim \infty$.
f_v ▷ semantics function for value v.
P ▷ viewpoint from which the alignment is computed, as defined in Equation (3.4).

Data : $\mathbb{A} = \langle \Delta, \Sigma, \Omega_0 \rangle$ ▷ default ASL description composed of the agents base Δ, the states base Σ and the default rule base Ω_0 containing the set of rules whose priority equals 0.
id ▷ string, an identifier for the action situation under examination.
$\mathcal{U}$ ▷ utility assignment function to set the EFG utilities based on the fluents declared in Σ, as defined in Equation (3.4).
$\mathcal{SC}$ ▷ game-theoretical solution concept. By default, subgame perfect equilibria, as defined in Equation (3.4).
$\mathfrak{TM}$ ▷ set of Theory of Mind clauses, as defined in Equation (3.4).
max ▷ an optional non-negative integer to limit the depth of the generated EFG models. By default, $max \sim \infty$.

Output : $\mathsf{Algn}^P_{N,v}$ ▷ the alignment of norms in N with respect to value v from the viewpoint of P (a double).

Function ALIGNMENT *(N,v,P)*:
- $\Omega_0 \leftarrow \Omega_0 \cup \Omega$ // add rules to ASL description
- $\Gamma, \mathcal{F} \leftarrow$ BUILD-FULL-GAME(id, $thres$, max) // from Contribution 2
- **for** $(z, g) \in Z_\Gamma \times G$ **do** // G is the set of agents declared in Δ and Z_Γ is the set terminal nodes in Γ
  - $U_g(z) \leftarrow \mathcal{U}(\mathcal{F}(z), pl)$ // assign utilities to EFG
- $\mathcal{P}_E \leftarrow \mathcal{SC}(\Gamma)$ // equilibrium strategy assigns a probability to every edge in Γ
- **for** $z \in Z$ **do** // probability distribution over outcomes
  - $\mathcal{P}(z) \leftarrow \prod_{\text{edge} \in \text{path}(\text{root} \to z)} \mathcal{P}_E(\text{edge})$
- $BB \leftarrow \mathfrak{TM}$, $\mathsf{Algn} \leftarrow 0$
- **for** $z \in Z$ **do**
  - $BB \leftarrow BB \cup \mathcal{F}(z)$
  - CopyToBackup() // from Contribution 3
  - AdoptViewpoint(P) // from Contribution 3
  - $\mathsf{Algn} \leftarrow \mathsf{Algn} + \mathcal{P}(z) \cdot f_v(BB)$ // from Contribution 1
  - RecoverBackup() // from Contribution 3
  - $BB \leftarrow BB \setminus \mathcal{F}(z)$
- **return** Algn

Against the data that Algorithm 1 relies upon, the ALIGNMENT function takes as arguments a set of norms N, a value v and a perspective P, and returns the alignment of the norms in N with respect to value v from the perspective of P, $\mathsf{Algn}^{P}_{N,v}$. The norms in N are encoded as a set of *higher priority* ASL rules and, optionally, a threshold to select the maximum priority of the rules to be considered (by default, all higher-priority rules are included). Value v is provided as a *semantics function* f_v, which captures the meaning of v in the action situation under examination, as presented in Equation (3.6). Finally, the viewpoint P under which the alignment is computed is, as defined in Contribution 3, an ordered sequence of agents $[\beta, \gamma, \ldots, \delta]$ whose perspective is sequentially adopted when the outcomes of the generated EFG model are analyzed.

Algorithm 1 starts by adding the higher-priority norms in N to the default rule base (Line 2). Then, the game engine from Contribution 2 automatically builds the EFG model Γ that grounds the semantics of the interaction, provided that the norms in N were adopted (Line 3). To do so, the agent executes the BUILD-FULL-GAME function, thoroughly presented in Contribution 2. As a result of this execution, function $\mathcal{F} : Z_\Gamma \rightarrow \mathsf{Pow}(\mathcal{L})$ is also returned, which maps every outcome in Z_Γ (the set of terminal nodes in the game tree of Γ) to the set of fluents that characterises it.

As mentioned in Contribution 2, the framework there does not manage the terminal nodes utilities that fully characterise an EFG. The assignment of utilities to outcomes is performed by the custom utility assignment function $\mathcal{U}$, which is crafted for the action situation under examination (Lines 4-5). Once the game utilities are assigned, the solution concept $\mathcal{SC}$ computes the equilibrium strategies for Γ, which results in a prediction of the actions that agents will take in the game. This is represented as the function $\mathcal{P}_E : E_\Gamma \rightarrow [0, 1]$, which assigned to the edges in Γ's game tree, denoted by E_Γ, to the probability of that edge being traversed during game play (Line 6). From $\mathcal{P}_E$, the probability distribution over outcomes $\mathcal{P} : Z \rightarrow [0, 1]$ that is induced by the equilibrium strategies is computed (Lines 7-8). $\mathcal{P}(z)$ is calculated as the probability of the path from the root node of the game tree to the terminal node z, which in turn is the joint probability of every edge in the path. Since Γ is represented by a *tree* (rather than a general graph), there is only one path from the root of the tree to any of its terminal nodes.

So far, Algorithm 1 builds a game model reflecting the implementation of the norms in N using the framework presented in Contribution 2, and analyses the effects of such implementation as the distribution over outcomes. Next,

the ALIGNMENT function evaluates the EFG from perspective P with respect to value v. This process starts by initializing the agent's belief base BB with the set of ToM clauses $\mathfrak{TM}$ and the alignment to zero (Line 9). Then, Algorithm 1 loops over the set of outcomes (i.e. the terminal nodes in Z) to examine the contribution of each outcome to the alignment. At each terminal state, the agent adds the set of fluents $\mathcal{F}(z)$ that characterise it to its belief base (Line 11).

Following that, the agent proceeds to switch its perspective and adopt that of P. This is achieved by applying the ToM-related function presented in Contribution 3. First, the agent makes a copy of its current belief base to a backup using the COPYTOBACKUP function (Line 12), and then it substitutes its belief base by the estimation it is able to build of the beliefs from P's perspective using function ADOPTVIEWPOINT. Both COPYTOBACKUP and ADOPTVIEWPOINT are developed in Contribution 2.

As a reminder of the main ToM-related functionalities that the agent has, suppose agent α is the one computing the alignment from perspective $P = [\beta, \ldots, \gamma, \delta]$. Then, in Line 13 of Algorithm 1 α replaces its original belief base BB_α by:

$$BB_{\alpha,\beta,\ldots,\gamma,\delta} = \{\phi \mid BB_{\alpha,\beta,\ldots,\gamma} \models \texttt{believes}(\delta, \phi, id)\} \tag{3.11}$$

where the first iteration in Equation (3.11) is given by:

$$BB_{\alpha,\beta} = \{\phi \mid BB_\alpha \models \texttt{believes}(\beta, \phi, id)\} \tag{3.12}$$

There is only one small difference between Equations (3.11) and (3.12) and their counterparts in Contribution 3 (see Equation (1) there), and that is the identifier *id* in the ToM clauses. As mentioned previously, this is not a crucial difference for the time being.

After the execution of Line 13 in Algorithm 1, the agent has in its belief base the estimation they can make of the view of the world from P's perspective. Now, the agent adds to the alignment Algn the contribution of the terminal node z under examination. To do so, it applies the value semantics function to the current state of its belief base, which generally contains beliefs attributed to other agents, and weights the result by the probability of the outcome $\mathcal{P}(z)$. Before moving on to the next outcome in Z, the agent recovers its original belief base BB_α using function RECOVERBACKUP (also presented in Contribution 3) and removes the fluents assigned to the outcome that has just been evaluated to move on to the next one.

As mentioned at the end of Section 3.2, in our proposal the agent comput-

ing the alignment (which we have been denoting as α) does not consider the possibility that other agents assign utilities differently or apply a different solution concept to predict the distribution over outcomes, leaving all the effects of switching the perspective to fall exclusively in the value semantics function (as illustrated in Section 3.3). The current version of Algorithm 1 reflects this choice. Nonetheless, it could be easily modified to allow for the fact that agent α believes that other agents assign utilities and/or predict the outcomes of the game differently. This would require changing $\mathcal{U}$ in Line 5 (α's own utility assignment function) by $\mathcal{U}^P$ (α's belief about the utility assignment function for perspective P); and also changing $\mathcal{SC}$ in Line 6 (α's own solution concept) by $\mathcal{SC}^P$ (α's belief about the solution concept employed for perspective P). Obviously, this would require a substantial amount of additional information at α's disposal. To focus solely on the *value-related* implications of ToM, we set the agent computing the ALIGNMENT function to work with a single utility assignment function and solution concept.

Implementation

An agent class called `AlgnAgent` with the ALIGNMENT functionality from Algorithm 1 (as well as the necessary functionalities from the contributions in Part II) has been implemented in Jason, a Java-based agent-oriented BDI language (Bordini et al., 2007). The ALIGNMENT function is implemented as a method of the agent class. However, its execution can be triggered from the application-specific agent code through the Jason internal action `integration.alignment`, presented

```
+trigger : context
    <-  ... ;
        integration.alignment(
            +Value,
            +Path,
            +Id,
            +Threshold,
            +UtilityModule,
            +UtilityFunction,
            +Viewpoint,
            -Degree
        );
        ...
```

Listing 3.1: Usage of the `integration.alignment` internal action in an arbitrary Jason plan to compute the perspective-dependent alignment.

in Listing 3.1. There, arguments that need to be bound at invokation time are preceded by "+", while variables that are instantiated by the internal action are preceded by "-".

The internal action `integration.alignment` has the following arguments:

- `Value` is a ground term denoting the value (or set of values) with respect to whom the alignment is computed. `Value` may be bound to an atom referring to a single value (e.g. `equality`), or a literal to refer to the aggregation of several values (e.g. `aggregation(equality, fairness)`).
- `Path` is a string pointing to the location of the ASL description.
- `Id` is an atom denoting the identifier for the action situation under examination (e.g. `ipd`, `metanorms`, `fishers` from Contribution 2).
- `Threshold` is an integer to filter the rules considered during computation of the alignment (i.e. rules whose priority is higher than this threshold are not used during the game building process).
- `UtilityModule` is the name of the Python module where the utility assignment function is defined, and `UtilityFunction` is the name of such function. Both arguments are strings.
- `Viewpoint` is a list of atoms denoting the agents whose perspectives are sequentially adopted, e.g. `[alice, bob, charles]`.
- `Degree` is a double that is bounded by the internal action `integration.alignment`. It corresponds to the return value of Algorithm 1, i.e. the perspective-dependent alignment $\mathsf{Algn}^{P}_{N,v}$.

At present, the implementation of the ALIGNMENT function does not allow to tune the solution concept $\mathcal{SC}$ that is applied to the ASL-generated game, and it uses the Nash equilibrium coupled with backward induction, i.e. subgame perfect equilibrium. Beyond the ASL description (whose syntax is extensively covered in Contribution 2), the agents need the ToM clauses $\mathfrak{TM}$ to build estimations of other agent's belief bases. These clauses follow the structure presented in Equation (3.10).

3.5 Example

To illustrate the new agent functionality presented in this chapter, we turn to one of the examples from Contribution 2: the fisher's game. There, two

fisher agents in an open-water fishery compete over fishing spots of varying productivity, and possibly fight over them in the default rule configuration. In order to avoid aggressive and inefficient outcomes, agents may introduce higher-priority rules that establish new allocation schemes to incentivise fairness and honest behaviour by agents: in the *first-in-time, first-in-right* configuration, agents are entitled to fishing at a spot if they win the race to get there; while in the *first-to-announce, first-in-right* configuration, one agent is allowed to declare a spot beforehand and is entitled to it as long as the agent does go to declared spot. For a detailed description of this action situation, refer to Section 6.1 in Contribution 2. We only make one small change with respect to the description presented there, and that is that in the *first-to-announce, first-in-right* rules configuration, the agent who takes over the role of *announcer* is no longer chosen at random, but deterministically assigned to one of the agents, in this case `alice`.

3.5.1 Modelling

In this section, we evaluate the three rule configurations for the fisher's action situation from the perspective of agent `alice` with no ToM involved ($\mathsf{Algn}^{a}_{N,v}$) and from the perspective that agent `alice` estimates that agent `bob` has, hence using first-order ToM ($\mathsf{Algn}^{a,b}_{N,v}$). We compute the alignment from these two perspectives with respect to the following values: *equality*, *achievement*, *power*, *benevolence* and *conformity*. From the perspective of `alice`, these values take on the meaning conveyed by following semantics functions:

- *Equality*:
$$f_{eq}(z) = 1 - 2 \cdot GI(U_{\texttt{alice}}(z), U_{\texttt{bob}}(z)) \tag{3.13}$$
where $U_i(z)$ is the utility of agent i at outcome z and $GI(\cdot)$ is the Gini index given a set of income or wealth data points.

- *Achievement*:
$$f_{ach}(z) = \frac{U_{\texttt{alice}}(z)}{\mathsf{maxProd}} \tag{3.14}$$
where $\mathsf{maxProd}$ is the maximum productivity of the fishing spots in the environment.

- *Power*:
$$f_{pow}(z) = \begin{cases} +1 & \text{if } \texttt{alice} \text{ wins a fight or a race} \\ -1 & \text{if } \texttt{alice} \text{ looses a fight or race} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

- *Benevolence*:

$$f_{ben}(z) = \begin{cases} +1 & \text{if no fights or races have occurred} \\ -1 & \text{if a fight or a race has occurred} \end{cases} \tag{3.16}$$

- *Conformity*:

$$f_{con}(z) = \begin{cases} +1 & \text{if } \texttt{alice}\text{, in the } \textit{announcer} \text{ role,} \\ & \text{goes to the announced spot} \\ -1 & \text{if } \texttt{alice}\text{, in the } \textit{announcer} \text{ role,} \\ & \text{goes to a spot different from} \\ & \text{the one announced} \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

In the previous modelling, value *equality* is captured by a similar semantics function as that used in Contribution 1, i.e. it seeks to minimise the Gini index. Remember that a Gini index ~ 0 means perfect equality, while a Gini index ~ 1 means perfect inequality. Next, value *achievement* is modelled by `alice` as financial success, i.e. what is the ratio between its achieved utility and the maximum that can in principle be obtained. Finally, the last three values (*power*, *benevolence* and *conformity*) have two- or three-valued semantics functions. For `alice`, value *power* is manifested by winning any form of competition, while *benevolence* is translated as the absence of such competition. Finally, value *conformity* for `alice` means that, if given the opportunity to announce a spot, such an announcement should be honest (i.e. `alice` goes to the declared spot).

Of the five values being modelled, only *equality* is not recognised as a value category in Schwartz's Theory of Basic Human Values (STBHV) (see Section 1.1.2). In fact, as modelled here, value *equality* could be considered as a form of "financial benevolence". Value *benevolence*, according to STBHV, refers to concerns for the well-being of those in agent's in-group. In contrast, value *universalism* represents the concern for all people and nature, regardless of their kin or affiliation. For this example, value *universalism* does not apply. However, value *benevolence* can take several forms, i.e. concern for the financial well-being of others in the group, as modelled by Equation (3.13), or concern for the physical security of others in the group manifested by the absence of competitions, as modelled by Equation (3.16).

Overall, *achievement* and *power* are self-enhancement values with a *personal* focus, while *benevolence* and *conformity* are *socially-focused* values. Value *benev-*

olence is opposite to *achievement* and *power* in Schwartz's circumference value structure. Meanwhile, *conformity* is adjacent to *benevolence*, the former being anxiety-based and the latter being anxiety-free.

So far, we have all the information to compute the alignment of the various rule configurations with respect to the presented values from the perspective of agent `alice`, i.e. with no ToM involved ($\mathsf{Algn}^{a}_{N,v}$). Nonetheless, we are also interested in `alice`'s estimation of `bob`'s alignment ($\mathsf{Algn}^{a,b}_{N,v}$). To do so, `alice` must first be able to partially estimate the content of `bob`'s belief base. For this example, `alice` believes that `bob` shares all of her beliefs, except for when the two are in different locations. In that case, `alice` believes that `bob` assumes that her utility equals the productivity of the spot where `bob` is *not* located:

```
believes(bob, utility(alice, X), fishers) :-
    at(bob, S1) & at(alice, S2) & S1\==S2 & productivity(S2, X).
```
(3.18)

Bear in mind that this may not be the case, as `alice` may have incurred in extra costs associated with travel between spots.

In addition to the beliefs, `alice` must also have an estimation of `bob`'s value semantics functions in order to compute the alignment. For values *power*, *benevolence* and *conformity*, `alice` assumes that `bob`'s interpretation is the same as hers. In contrast, they differ for values *equality* and *achievement*. For *equality*, `alice` assigns the following semantics function to `bob`:

$$f_{\text{eq}}(z) = \begin{cases} 2 \cdot \frac{\min\limits_{pl \in Pls} U_{pl}(z)}{\max\limits_{pl \in Pls} U_{pl}(z)} - 1 & \text{if } \min\limits_{pl \in Pls} U_{pl}(z) > 0 \\ -1 & \text{otherwise} \end{cases} \tag{3.19}$$

In contrast to Equation (3.14), where `alice`'s sense of achievement depends only on her own gains, she believes that `bob`'s sense of achievement depends on his gains as compared to hers. Therefore, she assigns the following semantics function to him for value *achievement*:

$$f_{ach}(z) = \frac{2}{\pi} \cdot \arctan\left(U_{\texttt{bob}}(z) - U_{\texttt{alice}}(z)\right) \tag{3.20}$$

which is a sigmoid function that takes positive values when $U_{\texttt{bob}}(z) > U_{\texttt{alice}}$ and negative values when $U_{\texttt{bob}}(z) < U_{\texttt{alice}}$. The normalisation constant $\frac{2}{\pi}$ is added to bound its limits between 1 and -1.

Table 3.1: Perspective-dependent alignment results for the fishers example.

		Default	*First-in-time, first-in-right*	*First-to-announce, first-in-right*
Equality	Algn^{a}	0.50	0.76	0.84
	$\mathsf{Algn}^{a,b}$	-0.51	-0.15	0.00
Achievement	Algn^{a}	0.48	0.57	1.0
	$\mathsf{Algn}^{a,b}$	0.09	0.19	-0.87
Power	Algn^{a}	0.13	-0.24	0.0
	$\mathsf{Algn}^{a,b}$	-0.13	0.23	0.0
Benevolence	Algn^{a}	-0.02	-1.0	1.0
	$\mathsf{Algn}^{a,b}$	-0.02	-1.0	1.0
Conformity	Algn^{a}	0.0	0.0	1.0
	$\mathsf{Algn}^{a,b}$	0.0	0.0	0.0
Sum	Algn^{a}	1.09	0.09	3.84
	$\mathsf{Algn}^{a,b}$	-0.57	-0.73	0.13

3.5.2 Results

The results of computing the alignment, with no ToM and with first-order ToM, for all rule configurations in the fishers example and with respect to all values modelled in the previous section are presented in Table 3.1.

Looking at the alignment from `alice`'s perspective ($\mathsf{Algn}^{a}_{N,v}$), the most value-aligned rule configuration is the *first-to-announce, first-in-right* one. We can provide two pieces of evidence to support this claim. First, this is the rule configuration with the largest degree of alignment for all values except *power*, with respect to whom the alignment is neutral. Second, the *first-to-announce, first-in-right* rule configuration is the *Pareto optimal* for `alice`. In other words, if the rule configuration changed in order to pursue a higher alignment with respect to *power* (for instance, by adopting the default rule configuration), then the alignment with respect to at least one other value, and in fact several other values, would decrease.

When using first-order ToM and adopting the perspective of `bob`, `alice` finds that the three rule configurations are Pareto optimal with respect to values. This means that, for any set of norms N_1 in Table 3.1, switching to a different set of norms N_2 does always improve the alignment with respect to some value, but at the expense of diminishing it with respect to others. Hence, Pareto optimality cannot be the criteria that `alice` uses to determine which rule configuration is preferred by `bob`.

However, `alice` might conclude that the *first-to-announce, first-in-right* rule

configuration is the one preferred by `bob`, since the sum of its alignment across all values is the largest of the three rule sets examined (see the last row of Table 3.1). Hence, when engaged in negotiation, `alice` would propose *first-to-announce, first-in-right* rules since they best fit the alignment with respect to her values, while at the same time believing that `bob` perceives them as the best possible alternative. Nonetheless, the accuracy of this assessment by `alice` depends on her ability to estimate with exactitude `bob`'s alignment, which in turn depends on her ability to make sensible estimations of `bob`'s beliefs and his interpretation of values.

For this example, we are using a summation operation to aggregate the alignment with respect to all the value of interest. Thus, we are implicitly assuming all values to have the same degree of importance for `alice`. Nonetheless, this choice is just intended as an example. Naturally, `alice` could aggregate the alignment with respect to all the values under consideration differently, e.g. by weighing more heavily those that she considers more pressing. Unfortunately, an in-depth examination of the alignment aggregation process across values is outside the scope of the current work.

3.6 Conclusions

In this chapter, we have combined the contributions made in Part II of this thesis into a novel agent functionality. This new functionality enables autonomous agents to reason about the value alignment of a set of norms from any perspective. In other words, the agent can reason about the alignment of a set of norms with respect to its own values, or it can reason about the alignment with respect to the values it believes another agent holds. This has been achieved through the formulation and implementation of the Alignment function to compute the perspective-dependent value alignment of norms. This function combines the norm interpretation system of Contribution 2, the ToM abilities of Contribution 3 and the grounding of values as devices to assess outcomes of Contribution 1.

The integration work undertaken in this chapter addresses one of the main gaps in the literature on value and autonomous agents identified in Section 2.4: the fact that value-guided reasoning for agents is limited to a value structure and representation that is provided prior to run-time and remains static throughout the agent's lifetime. In our integration, through the use of ToM rules, the agent can compute the alignment with respect to a dynamic set of values that is generally constructed at run-time and that can be triggered on-demand, through the

provided internal action interface to the ALIGNMENT function. Even though in this thesis we focus on ToM, the dynamic nature of the values under consideration may also come from the agent's own evolving values or a prior agreement in a community on which value ought to be upheld.

Chapter 4

Conclusions

In this last chapter, we wrap-up this thesis. We first reflect on the contributions made with respect to the initial research goals and the larger AAMAS landscape. Second, we enumerate the software tools accompanying the contributions, and point to documentation with guidelines on their usage. Finally, we conclude with a high-level view on the main takeaways from this work, and point to future directions that could follow from it.

4.1 Revisiting the Research Goals

We begin by examining one by one the research questions presented in Section 1.3, and we then move on to evaluating the overall research goal. We reproduce here the research questions and answer them in relation to the contributions in Part II.

Research Question #1: *How should* values *be* represented *in a way that is suitable to evaluate outcomes (that may refer to a variety of contexts or domains) in terms of their adherence so the value in question?*

RQ#1 has been entirely addressed in Contribution 1. There, we establish that values are grounded as a *semantics function*, which can take on different forms depending on the context where the value is applied and/or the designer criteria. Such semantics functions evaluate states of a MAS, and in particular end-states or *outcomes*, considering the features that describe it. We adhere to this approach, which is originally presented in Contribution 1, throughout this whole thesis, and in particular during the integration of functionalities presented in Chapter 3.

Research Question #2: *How should* norms *be* represented *in a way that allows*

*to connect them (either exactly or approximately) to the possible outcomes that the MAS may achieve, which are evaluated in terms of their value promotion by the value representation provided in **RQ#1**?*

The first attempt at answering **RQ#2** has happened in Contribution 1, where norms are represented as a set of normative parameters that are bounded to a value in a given domain and, possibly, subject to some constraints. However, that is not an expressive norm representation language, and it was not clear how the introduction and/or retrieval of norms ought to be formalised. For this reason, we have opted to represent norms using the Action Situation Language (ASL) presented in Contribution 2. The ASL provides a syntactically friendly, flexible and *extensible* norms representation. In other words, it integrates mechanisms to handle the introduction or removal of norms to an existing normative system, thus bringing it closer to real-world policy-making where new rules and laws are introduced into an existing corpus.

Research Question #3: *Derived from the previous two research questions, how can prescriptive norms be designed, given a set of value representations, to maximise their alignment with respect to the given values? How can this process be automated?*

The synthesis of prescriptive norms based on their degree of value alignment has been completely automated in Contribution 1, albeit at the expense of using a norm representation scheme with limited expressiveness (see the response to **RQ#2**). The ASL has not, in the course of this thesis, been integrated with any automation tool to find the normative system most aligned with respect to an input value. Rather, exhaustive searches through the set of potential new regulations have been performed. Therefore, this research question has only been partially satisfied and could be pursued in future work.

Research Question #4: *How can existing agent architectures be expanded with Theory of Mind capabilities so that software agents can perceive the state of the system from the perspective of other agents situated in their same environment?*

This point has been extensively developed in Contribution 3. In particular, using the AdoptViewpoint function of the TomAbd agent model, an agent can adopt the perspective of another. Furthermore, this switch in perspective can happen at multiple levels of recursion (using high-order ToM), imposes minimal additional memory requirements, and happens on-demand at run-time, thus allowing agents to dynamically adopt and reason from the perspectives of many other agents.

Research Question #5: *Following from **RQ#4**, how can an agent's Theory of Mind capabilities be used to evaluate the alignment of a set of norms with respect to the values*

held by other agents in the MAS?

RQ#5 has been addressed in the previous Chapter 3. There, we have combined the value representation scheme of Contribution 1, the norms representation and interpretation framework of Contribution 2 and the ToM capabilities of Contribution 3 to enable agents to reason about the outcomes most incentivised by a set of norms not just from their own values, but from the value perspective of other agents.

By progressively addressing the Research Questions articulated at the beginning of this thesis, we believe we have satisfactorily fulfilled the Main Research Goal:

Main Research Goal (MRG): *Develop a novel agent functionality to empower software agents to reason about the alignment* (i.e. *the suitability) of a set of prescriptive norms (or normative system) with respect to a value or set of values, either from the perspective of the agent's own values* (i.e. *the values that have been handed down to it by its user) or from the perspective of another agent's values it shares its environment with.*

The functionality that the **MRG** refers to is the one formulated, implemented and illustrated in Chapter 3. We have provided an overview of the integration approach there in the response to **RQ#5**, which essentially culminates all the work developed in answering the previous research questions.

4.2 A Toolbox for Value Engineering

All the software developed in the course of this thesis is open-source, documented and publicly available for researchers to inspect and build upon. In this section, we gather and link to the available repositories and provide an overview of their functionality.

1. The repository accompanying Contribution 1 includes the scripts to reproduce the results there and additional materials such as animations displaying the evolution of the wealth distribution for the various optimal normative systems.[1] A user who wishes to optimise and analyse a normative system for their custom MAS has to define the domains of the normative parameters and implement how they affect the state transitions. Nonetheless, the optimisation and analysis on Shapley values and value

[1] https://github.com/nmontesg/normsynth

compatibility can be performed using the same scripts as we did for the example in Contribution 1.

2. The implementation of the ASL language, the automated game engine and some additional functionalities for the analysis of Extensive Form Game (EFG) is included in the code accompanying Contribution 2.[2] Users can write their own ASL descriptions following the syntax presented in Contribution 2 and use the automated game engine to build the corresponding EFG model. Besides that, users can also use complementary functions to compute the equilibrium strategies and the distribution over outcomes induced by those. Furthermore, users can define their custom evaluation functions over end-states and leverage the outcome probabilities to compute its expected value and other metrics.

3. The code implementing the TomAbd agent model from Contribution 3 is suited to be used in Jason or JaCaMo projects.[3] It includes the agent class with all the functionalities exposed in Contribution 3, together with the internal actions to trigger such functionalities. In order to facilitate its integration with MAS projects, it has been packaged into a Java Archive (.jar) file that only needs to be appended to the classpath of the project to be used.

4. Finally, the code implementing the integration of the three contributions from Part II presented in Chapter 3 is also suited to be used in Jason or JaCaMo projects.[4] It is built as an extension of the agent class implementing the TomAbd agent model from Contribution 3. Therefore, its usage (which is exposed in Section 3.4) needs only to add the compiled .jar file to the project classpath.

4.3 Takeaways and Future Work

In this thesis, we have provided a new outlook on values for autonomous agents by incorporating the emphatically social dimensions of values identified by Schwartz's Theory of Basic Human Values. By fulfilling our Main Research Goal, we have contributed to filling the gap in the literature on values and

[2] https://github.com/nmontesg/norms-games
[3] https://github.com/nmontesg/tomabd
[4] https://github.com/nmontesg/integration

AAMAS identified in Section 2.4. That is, the embedding of values into autonomous agents under the implicit assumption that the agents' value systems are isolated from one another, static and unchangeable. Instead, we strongly link prescriptive norms to values (since norms are the concrete, system-wide construct that is actually implemented in a MAS) and allow normative reasoning to be performed from the agent's own perspective, or that of another agent it shares its environment with. Hence, we have developed a functionality to perform *socially-oriented value-based normative reasoning*.

The work developed in the course of this thesis leaves the door open for many future research directions. The first and most obvious one is the use of the functionality culminating this thesis (the one presented in Chapter 3) in value-driven negotiation over normative systems. There, agents would evaluate proposals for the introduction of norms based on their degree of alignment, interpreting norms using the framework in Contribution 2 and evaluating their subsequent outcomes as in Contribution 1. The use of ToM would be useful for agents in order to make proposals that stand a chance of being accepted by their interlocutors, by analysing their degree of alignment with respect to the values of other before proposing them.

Second, we should note that in this thesis we have solely introduced values into autonomous agents through prescriptive norms. For example, in the analysis of the models generated from ASL descriptions, we have used game-theoretical models to predict the distribution over outcomes, without any consideration on how values may play a role in the agents' decision-making process. Hence, future research could extend the current work and incorporate values into several constructs, such as prescriptions (covered here), conventions and actions.

The third and last direction for future research relates to multi-context systems. In this thesis, we have assumed from beginning to end that agents are situated in one and only one domain, environment or context, such as the tax collection and redistribution example in Contribution 1 or the fishers example in Contribution 2. Nonetheless, autonomous agents can be enhanced by enabling them to operate in various domains simultaneously. Even though we argue for the grounding of values for every context separately, we foresee the possibility that values in different contexts dynamically relate to one another through some kind of bridge rules, as those utilised in multi-context systems (Sabater, 2002). For example, low value satisfaction in one context might drive an agent to seek high alignment with respect to that value in another context

by increasing its priority, in a process analogous to the phenomenon of psychological compensation (Bäckman & Dixon, 1992). Future work could formalise the syntax, semantics and implementation of rules bridging values that apply to different domains.

Acronyms

AAMAS Autonomous Agents and Multiagent Systems

AI Artificial Intelligence

ASI Artificial Social Intelligence

ASL Action Situation Language

BDI Belief-Desire-Intetion

CMNS Compatibility Maximizing Normative System

DNN Deep Neural Network

EFG Extensive Form Game

IAD Institutional Analysis and Development

IRL Inverse Reinforcement Learning

IRON Intelligent Robust Norm synthesis

LP Linear Programming

MAS Multiagent System

RL Reinforcement Learning

ST Simulation-Theory of Mind

STBHV Schwartz's Theory of Basic Human Values

STRIPS Stanford Research Institute Problem Solver

ToM Theory of Mind

TT Theory-Theory of Mind

Bibliography

Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, *258*, 66–95. https://doi.org/10.1016/j.artint.2018.01.002

Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., & Padget, J. (2018). *Social coordination frameworks for social technical systems*. Springer.

Alexander, L., & Moore, M. (2021). Deontological Ethics. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Winter 2021). Metaphysics Research Lab, Stanford University.

Andrighetto, G., Governatori, G., Noriega, P., & van der Torre, L. (2012). Normative Multi-Agent Systems (Dagstuhl Seminar 12111) (G. Andrighetto, G. Governatori, P. Noriega, & L. van der Torre, Eds.). *Dagstuhl Reports*, *2*(3), 23–49. https://doi.org/10.4230/DagRep.2.3.23

Askham, A. V. (2022). 'Theory of mind' in autism: A research field reborn. *Spectrum*. https://doi.org/10.53053/gxnc7576

Atkinson, K., & Bench-Capon, T. (2016). States, goals and values: Revisiting practical reasoning (K. Atkinson, F. Cerutti, P. McBurney, S. Parsons, & I. Rahwan, Eds.). *Argument & Computation*, *7*(2-3), 135–154. https://doi.org/10.3233/aac-160011

Aumann, R. J. (1974). Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, *1*(1), 67–96. https://doi.org/10.1016/0304-4068(74)90037-8

Axelrod, R. (1986). An evolutionary approach to norms. *American Political Science Review*, *80*(04), 1095–1111. https://doi.org/10.2307/1960858

Azzolini, D., Bellodi, E., Ferilli, S., Riguzzi, F., & Zese, R. (2022). Abduction with probabilistic logic programming under the distribution semantics. *International Journal of Approximate Reasoning*, *142*, 41–63. https://doi.org/https://doi.org/10.1016/j.ijar.2021.11.003

Baarslag, T., Hendrikx, M. J. C., Hindriks, K. V., & Jonker, C. M. (2015). Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, *30*(5), 849–898. https://doi.org/10.1007/s10458-015-9309-1

Bäckman, L., & Dixon, R. A. (1992). Psychological compensation: A theoretical framework. *Psychological Bulletin*, *112*(2), 259–283. https://doi.org/10.1037/0033-2909.112.2.259

Baluja, S., & Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, 38–46.

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., Dunning, I., Mourad, S., Larochelle, H., Bellemare, M. G., & Bowling, M. (2020). The hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, *280*, 103216. https://doi.org/10.1016/j.artint.2019.103216

Baron-Cohen, S., Leslie, A. M., & Frith, U. (1985). Does the autistic child have a "theory of mind" ? *Cognition*, *21*(1), 37–46. https://doi.org/https://doi.org/10.1016/0010-0277(85)90022-8

Barton, D. N., Benavides, K., Chacon-Cascante, A., Le Coq, J. F., Quiros, M. M., Porras, I., Primmer, E., & Ring, I. (2017). Payments for ecosystem services as a policy mix: Demonstrating the institutional analysis and development framework on conservation policy instruments. *Environmental Policy and Governance*, *27*(5), 404–421. https://doi.org/10.1002/eet.1769

Belzer, M. (1998). Deontic logic. In *Routledge encyclopedia of philosophy*. Routledge. https://doi.org/10.4324/9780415249126-y043-1

Bench-Capon, T., & Modgil, S. (2017). Norms and value based reasoning: Justifying compliance and violation. *Artificial Intelligence and Law*, *25*(1), 29–64. https://doi.org/10.1007/s10506-017-9194-9

Black, M. (1962). *Models and metaphors: Studies in language and philosophy*. Cornell University Press.

Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). *Programming multi-agent systems in agentspeak using jason*. John Wiley & Sons.

Boudon, R. (2017). *The origin of values: Sociology and philosophy of beliefs*. Routledge. https://doi.org/10.4324/9781315133645

Caillou, P., Aknine, S., & Pinson, S. (2009). Searching pareto optimal solutions for the problem of forming and restructuring coalitions in multi-agent systems. *Group Decision and Negotiation*, *19*(1), 7–37. https://doi.org/10.1007/s10726-009-9183-9

Chalkiadakis, G., Elkind, E., & Wooldridge, M. (2011). Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *5*(6), 1–168. https://doi.org/10.2200/s00355ed1v01y201107aim016

Cole, D. H. (2017). Laws, norms, and the institutional analysis and development framework. *Journal of Institutional Economics*, *13*(4), 829–847. https://doi.org/10.1017/s1744137417000030

Conte, R., & Castelfranchi, C. (1999). From conventions to prescription. towards an integrated view of norms. *Artificial Intelligence and Law*, *7*(4), 323–340. https://doi.org/10.1023/a:1008310107755

Corballis, M. (2007). The uniqueness of human recursive thinking. *American Scientist*, *95*(3), 240. https://doi.org/10.1511/2007.65.240

Corballis, M. C. (2011). *The recursive mind: The origins of human language, thought, and civilization: The origins of human language, thought, and civilization*. Princeton University Press.

Cox, C., Silva, J. D., Deorsey, P., Kenter, F. H. J., Retter, T., & Tobin, J. (2015). How to make the perfect fireworks display: Two strategies for Hanabi. *Mathematics Magazine*, *88*(5), 323–336. https://doi.org/10.4169/math.mag.88.5.323

Cozort, D., & Shields, J. M. (Eds.). (2018). *The oxford handbook of buddhist ethics*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780198746140.001.0001

Cranefield, S., Winikoff, M., Dignum, V., & Dignum, F. (2017). No pizza for you: Value-based plan selection in BDI agents. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2017/26

Crawford, S. E. S., & Ostrom, E. (1995). A grammar of institutions. *American Political Science Review*, *89*(3), 582–600. https://doi.org/10.2307/2082975

Criado, N., Argente, E., & Botti, V. (2010). A bdi architecture for normative decision making. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, 1383–1384.

Cuzzolin, F., Morelli, A., Cirstea, B., & Sahakian, B. J. (2020). Knowing me, knowing you: Theory of mind in AI. *Psychological Medicine*, *50*(7), 1057–1061. https://doi.org/10.1017/s0033291720000835

Dafoe, A., Bachrach, Y., Hadfield, G., Horvitz, E., Larson, K., & Graepel, T. (2021). Cooperative AI: Machines must learn to find common ground. *Nature*, *593*(7857), 33–36. https://doi.org/10.1038/d41586-021-01170-0

Dafoe, A., Hughes, E., Bachrach, Y., Collins, T., McKee, K. R., Leibo, J. Z., Larson, K., & Graepel, T. (2020). Open problems in cooperative AI.

de Jonge, D., Trescak, T., Sierra, C., Simoff, S., & de Mántaras, R. L. (2017). Using game description language for mediated dispute resolution. *AI & SOCIETY*, *34*(4), 767–784. https://doi.org/10.1007/s00146-017-0790-8

de Jonge, D., & Zhang, D. (2021). GDL as a unifying domain description language for declarative automated negotiation. *Autonomous Agents and Multi-Agent Systems*, *35*(1). https://doi.org/10.1007/s10458-020-09491-6

Denecker, M., & de Schreye, D. (1998). Sldnfa: An abductive procedure for abductive logic programs. *The Journal of Logic Programming*, *34*(2), 111–167. https://doi.org/https://doi.org/10.1016/S0743-1066(97)00074-5

Denecker, M., & Kakas, A. C. (2002). Abduction in logic programming. *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, 402–436.

Dennett, D. C. (1989). *The intentional stance* (7. printing). MIT Press.

de Weerd, H., Verbrugge, R., & Verheij, B. (2012). Higher-order social cognition in the game of rock-paper-scissors: A simulation study. In G. Bonanno, H. Van Ditmarsch, & W. Hoek (Eds.), *Proceedings of the 10th conference on logic and the foundations of game and decision theory (loft 2012)* (pp. 218–232).

de Weerd, H., Verbrugge, R., & Verheij, B. (2015). Negotiating with other minds: The role of recursive theory of mind in negotiation with incomplete information. *Autonomous Agents and Multi-Agent Systems*, *31*(2), 250–287. https://doi.org/10.1007/s10458-015-9317-1

de Weerd, H., Verbrugge, R., & Verheij, B. (2022). Higher-order theory of mind is especially useful in unpredictable negotiations. *Autonomous Agents and Multi-Agent Systems*, *36*(2). https://doi.org/10.1007/s10458-022-09558-6

de Weerd, H., & Verheij, B. (2011). The advantage of higher-order theory of mind in the game of limited bidding. *Workshop on Reasoning About Other Minds: Logical and Cognitive Perspectives*, *751*, 149–164.

Fagundes, M. S., Ossowski, S., Cerquides, J., & Noriega, P. (2016). Design and evaluation of norm-aware agents based on normative markov decision processes. *International Journal of Approximate Reasoning*, *78*, 33–61. https://doi.org/10.1016/j.ijar.2016.06.005

Fatima, S., Kraus, S., & Wooldridge, M. (2009). *Principles of automated negotiation*. Cambridge University Press. https://doi.org/10.1017/cbo9780511751691

Feather, N. T. (1995). Values, valences, and choice: The influences of values on the perceived attractiveness and choice of alternatives. *Journal of Personality and Social Psychology*, *68*(6), 1135–1151. https://doi.org/10.1037/0022-3514.68.6.1135

Flach, P. A., & Kakas, A. C. (Eds.). (2000). *Abduction and induction: Essays on their relation and integration*. Springer. https://doi.org/10.1007/978-94-017-0606-3

Foerster, J., Song, F., Hughes, E., Burch, N., Dunning, I., Whiteson, S., Botvinick, M., & Bowling, M. (2019). Bayesian action decoder for deep multi-agent reinforcement learning. In K. Chaudhuri

& R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning* (pp. 1942–1951). PMLR. https://proceedings.mlr.press/v97/foerster19a.html

Frantz, C., Purvis, M. K., Nowostawski, M., & Savarimuthu, B. T. R. (2013). nADICO: A nested grammar of institutions. In *Lecture notes in computer science* (pp. 429–436). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-44927-7_31

Frantz, C. K., & Siddiki, S. (2021). Institutional grammar 2.0: A specification for encoding and analyzing institutional design. *Public Administration*. https://doi.org/10.1111/padm.12719

Frith, C., & Frith, U. (2005). Theory of mind. *Current Biology*, *15*(17), R644–R645. https://doi.org/10.1016/j.cub.2005.08.041

Fung, T. H., & Kowalski, R. (1997). The iff proof procedure for abductive logic programming. *The Journal of Logic Programming*, *33*(2), 151–165. https://doi.org/https://doi.org/10.1016/S0743-1066(97)00026-5

Genesereth, M., Love, N., & Pell, B. (2005). General game playing: Overview of the aaai competition. *AI Magazine*, *26*, 62–72. https://doi.org/10.1609/aimag.v26i2.1813

Ghorbani, A., Bots, P., Dignum, V., & Dijkema, G. (2013). MAIA: A framework for developing agent-based social simulations. *Journal of Artificial Societies and Social Simulation*, *16*(2). https://doi.org/10.18564/jasss.2166

Ghorbani, A., & Bravo, G. (2016). Managing the commons: A simple model of the emergence of institutions through collective action. *International Journal of the Commons*, *10*(1), 200–219. https://doi.org/10.18352/ijc.606

Gini, C. (1912). *Variabilità e mutuabilità. contributo allo studio delle distribuzioni e delle relazioni statistiche*. Facoltà di Giurisprudenza della R. Università di Cagliari.

González-Díaz, J., García-Jurado, I., & Fiestras-Janeiro, M. G. (2010). *An introductory course on mathematical game theory*. American Mathematical Society; Real Sociedad Matemática Española.

Gorrieri, R. (2017). Labeled transition systems. In *Monographs in theoretical computer science. an EATCS series* (pp. 15–34). Springer International Publishing. https://doi.org/10.1007/978-3-319-55559-1_2

Gronewold, N. (2010). Game theory: Climate talks destined to fail. https://www.scientificamerican.com/article/game-theorist-predicts-failure-at-climate-talks/

Grossi, D., Gabbay, D., & van der Torre, L. (2010). The norm implementation problem in normative multi-agent systems. In *Specification and verification of multi-agent systems* (pp. 195–224). Springer US. https://doi.org/10.1007/978-1-4419-6984-2_7

Grossi, D., Tummolini, L., & Turrini, P. (2012). Norms in game theory. In *Agreement technologies* (pp. 191–197). Springer, Dordrecht. https://doi.org/10.1007/978-94-007-5583-3_12

Hahn, C., Phan, T., Feld, S., Roch, C., Ritz, F., Sedlmeier, A., Gabor, T., & Linnhoff-Popien, C. (2020). Nash equilibria in multi-agent swarms. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*, 234–241. https://doi.org/10.5220/0008990802340241

Harbers, M., Bosch, K. v. d., & Meyer, J.-J. (2009). Modeling agents with a theory of mind. *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, *2*, 217–224. https://doi.org/10.1109/WI-IAT.2009.153

Hu, H., Lerer, A., Peysakhovich, A., & Foerster, J. (2020). "Other-play" for zero-shot coordination. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (pp. 4399–4410). PMLR. https://proceedings.mlr.press/v119/hu20a.html

Hübner, J. F., Boissier, O., & Bordini, R. H. (2011). A normative programming language for multi-agent organisations. *Annals of Mathematics and Artificial Intelligence*, *62*(1-2), 27–53. https://doi.org/10.1007/s10472-011-9251-0

Jara-Ettinger, J. (2019). Theory of mind as inverse reinforcement learning [Artificial Intelligence]. *Current Opinion in Behavioral Sciences*, *29*, 105–110. https://doi.org/https://doi.org/10.1016/j.cobeha.2019.04.010

Josephson, J. R., & Josephson, S. G. (1994). *Abductive inference: Computation, philosophy, technology*. Cambridge University Press.

Kakas, A., Kowalski, R., & Toni, F. (1993). Abductive logic programming. *Journal of Logic and Computation*, *2*(6), 719–770. https://doi.org/10.1093/logcom/2.6.719

Kiser, L. L., & Ostrom, E. (1982). The three worlds of action: A metatheoretical synthesis of institutional approaches. Michigan University Press.

Knobe, J. (2005). Theory of mind and moral cognition: Exploring the connections. *Trends in Cognitive Sciences*, *9*(8), 357–359. https://doi.org/10.1016/j.tics.2005.06.011

Koller, D., & Pfeffer, A. (1997). Representations and solutions for game-theoretic problems. *Artificial Intelligence*, *94*(1-2), 167–215. https://doi.org/10.1016/s0004-3702(97)00023-4

Korkiakangas, T., Dindar, K., Laitila, A., & Kärnä, E. (2016). The Sally–Anne test: An interactional analysis of a dyadic assessment. *International Journal of Language & Communication Disorders*, *51*(6), 685–702. https://doi.org/https://doi.org/10.1111/1460-6984.12240

Kuhn, H. W. (1953). 11. extensive games and the problem of information. In *Contributions to the theory of games (AM-28), volume II* (pp. 193–216). Princeton University Press. https://doi.org/10.1515/9781400881970-012

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86. https://doi.org/10.1214/aoms/1177729694

Lemieux, C. (2009). *Monte carlo and quasi-monte carlo sampling*. Springer New York.

Lerer, A., Hu, H., Foerster, J., & Brown, N. (2020). Improving policies via search in cooperative partially observable games. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*, 7187–7194. https://doi.org/10.1609/aaai.v34i05.6208

Li, A., Zhou, L., Su, Q., Cornelius, S. P., Liu, Y.-Y., Wang, L., & Levin, S. A. (2020). Evolution of cooperation on temporal networks. *Nature Communications*, *11*(1). https://doi.org/10.1038/s41467-020-16088-w

Lin, F. (2008). Situation calculus. In *Handbook of knowledge representation* (pp. 649–669). Elsevier. https://doi.org/10.1016/s1574-6526(07)03016-7

Lockwood, B. (2008). Pareto efficiency. In *The new palgrave dictionary of economics* (pp. 1–5). Palgrave Macmillan UK. https://doi.org/10.1057/978-1-349-95121-5_1823-2

Luke, S. (2013). *Essentials of metaheuristics* (second). Lulu.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4768–4777.

Macintyre, A. (1998). *A short history of ethics: A history of moral philosophy from the homeric age to the twentieth century*. University of Notre Dame Press.

Malle, B. (2022). Theory of mind. In R. Biswas-Diener & E. Diener (Eds.), *Noba textbook series: Psychology*. DEF publishers.

Mas-Colell, A., Whinston, M. D., & Green, J. R. (1995). *Microeconomic theory*. Oxford University Press.

Meyer, J.-J. C., Broersen, J., & Herzig, A. (2015). BDI logics. In H. van Ditmarsch, J. Y. Halpern, & W. van der Hoek (Eds.), *Handbook of epistemic logics*. College Publications.

Miller, B. L., & Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, *9*.

Montes, N. (2022a). Engineering pro-social values in autonomous agents – collective and individual perspectives. In *Multi-agent systems* (pp. 431–434). Springer International Publishing. https://doi.org/10.1007/978-3-031-20614-6_26

Montes, N. (2022b). Engineering socially-oriented autonomous agents and multiagent systems. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2022/833

Montes, N., Luck, M., Osman, N., Rodrigues, O., & Sierra, C. (2023a). Combining theory of mind and abductive reasoning in agent-oriented programming. *Autonomous Agents and Multi-Agent Systems*, *37*(2). https://doi.org/10.1007/s10458-023-09613-w

Montes, N., Osman, N., & Sierra, C. (2021). Enabling game-theoretical analysis of social rules. In *Frontiers in artificial intelligence and applications*. IOS Press. https://doi.org/10.3233/faia210120

Montes, N., Osman, N., & Sierra, C. (2022a). Combining theory of mind and abduction for cooperation under imperfect information. In *Multi-agent systems* (pp. 294–311). Springer International Publishing. https://doi.org/10.1007/978-3-031-20614-6_17

Montes, N., Osman, N., & Sierra, C. (2022b). A computational model of Ostrom's Institutional Analysis and Development framework. *Artificial Intelligence*, *311*, 103756. https://doi.org/10.1016/j.artint.2022.103756

Montes, N., Osman, N., & Sierra, C. (2023). A computational model of Ostrom's institutional analysis and development framework (extended abstract). *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2023/786

Montes, N., Osman, N., Sierra, C., & Slavkovik, M. (2023). Value engineering for autonomous agents. https://doi.org/10.48550/ARXIV.2302.08759

Montes, N., & Sierra, C. (2022a). Synthesis and properties of optimally value-aligned normative systems. *Journal of Artificial Intelligence Research*, *74*, 1739–1774. https://doi.org/10.1613/jair.1.13487

Morales, J., López-Sanchez, M., Rodríguez-Aguilar, J. A., Wooldridge, M., & Vasconcelos, W. (2013). Automated synthesis of normative systems. *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, 483–490.

Morales, J., López-Sánchez, M., Rodríguez-Aguilar, J. A., Wooldridge, M., & Vasconcelos, W. (2015). Synthesising liberal normative systems. *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 433–441.

Morales, J., Wooldridge, M., Rodríguez-Aguilar, J. A., & López-Sánchez, M. (2018). Off-line synthesis of evolutionarily stable normative systems. *Autonomous Agents and Multi-Agent Systems*, *32*(5), 635–671. https://doi.org/10.1007/s10458-018-9390-3

Morris-Martin, A., Vos, M. D., & Padget, J. (2019). Norm emergence in multiagent systems: A viewpoint paper. *Autonomous Agents and Multi-Agent Systems*, *33*(6), 706–749. https://doi.org/10.1007/s10458-019-09422-0

Mühlenbein, H., & Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary Computation*, *1*(1), 25–49. https://doi.org/10.1162/evco.1993.1.1.25

Nash, J. F. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, *36*(1), 48–49. http://www.jstor.org/stable/88031

Nashed, S., & Zilberstein, S. (2022). A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research*, *73*, 277–327. https://doi.org/10.1613/jair.1.12889

Nguyen, T., & Watanabe, T. (2020). Autonomous motivation for the successful implementation of waste management policy: An examination using an adapted institutional analysis and development framework in thua thien hue, vietnam. *Sustainability*, *12*(7), 2724. https://doi.org/10.3390/su12072724

Nir, R., Shleyfman, A., & Karpas, E. (2020). Automated synthesis of social laws in strips. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(06), 9941–9948. https://doi.org/10.1609/aaai.v34i06.6549

O'Dwyer, A. (2017). Quuxplusone/Hanabi: Framework for writing bots that play Hanabi.

Onn, S., & Tennenholtz, M. (1997). Determination of social laws for multi-agent mobilization. *Artificial Intelligence*, *95*(1), 155–167. https://doi.org/10.1016/s0004-3702(97)00045-3

Osawa, H. (2015). Solving hanabi: Estimating hands by opponent's actions in cooperative game with incomplete information. *AAAI Workshop: Computer Poker and Imperfect Information*. http://aaai.org/ocs/index.php/WS/AAAIW15/paper/view/10167

Ostrom, E. (1990). *Governing the commons*. Cambridge University Press. https://doi.org/10.1017/cbo9780511807763

Ostrom, E. (2005). *Understanding institutional diversity*. Princeton University Press.

Ostrom, E. (2011). Background on the institutional analysis and development framework. *Policy Studies Journal*, *39*(1), 7–27. https://doi.org/10.1111/j.1541-0072.2010.00394.x

Ostrom, E., Gardner, R., & Walker, J. (1994). *Rules, games, and common-pool resources*. University of Michigan Press. https://doi.org/10.3998/mpub.9739

Paiva, A. (n.d.). Social ai: Learning and reasoning in social contexts. https://www.tailor-social-ai.eu/home

Paiva, A., et al. (2020). Wp6 – social ai: Learning and reasoning in social contexts [ICT-48 TAILOR: Foundations of Trustworthy AI – Integrating Reasoning, Learning and Optimization].

Panisson, A., Mcburney, P., Parsons, S., Bordini, R., & Sarkadi, S. (2018). Lies, bullshit, and deception in agent-oriented programming languages. *Proceedings of the 20th International Trust Workshop co-located with AAMAS/IJCAI/ECAI/ICML (AAMAS/IJCAI/ECAI/ICML 2018)*.

Panisson, A. R., Sarkadi, Ș., McBurney, P., Parsons, S., & Bordini, R. H. (2019). On the formal semantics of theory of mind in agent communication. In *Agreement technologies* (pp. 18–32). Springer International Publishing. https://doi.org/10.1007/978-3-030-17294-7_2

Peters, H. (2008). The shapley value. In *Game theory* (pp. 241–258). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-69291-1_17

Plotkin, G. D. (1981). *A structural approach to operational semantics*. Aarhus University.

Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. M. A., & Botvinick, M. (2018). Machine theory of mind. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (pp. 4218–4227). PMLR. https://proceedings.mlr.press/v80/rabinowitz18a.html

Rao, A. S. (1996). AgentSpeak(l): BDI agents speak out in a logical computable language. In *Lecture notes in computer science* (pp. 42–55). Springer Berlin Heidelberg. https://doi.org/10.1007/bfb0031845

Ray, O., & Kakas, A. (2006). Prologica: A practical system for abductive logic programming. *Proceedings of the 11th International Workshop on Non-monotonic Reasoning*, 304–312.

Rendsvig, R., & Symons, J. (2021). Epistemic Logic. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Summer 2021). Metaphysics Research Lab, Stanford University.

Rohan, M. J. (2000). A rose by any name? the values construct. *Personality and Social Psychology Review*, *4*(3), 255–277. https://doi.org/10.1207/s15327957pspr0403_4

Rokeach, M. (1972). *The nature of human values*. Free Press.

Röska-Hardy, L. (2008). Theory theory (simulation theory, theory of mind). In *Encyclopedia of neuroscience* (pp. 4064–4067). Springer. https://doi.org/10.1007/978-3-540-29678-2_5984

Ross, A., & Willson, V. L. (2017). Paired samples t-test. In *Basic and advanced statistical tests: Writing results sections and creating tables and figures* (pp. 17–19). SensePublishers. https://doi.org/10.1007/978-94-6351-086-8_4

Ruder, S. (2016). An overview of gradient descent optimization algorithms.

Sabater, J. (2002). Engineering executable agents using multi-context systems. *Journal of Logic and Computation*, *12*(3), 413–442. https://doi.org/10.1093/logcom/12.3.413

Sadri, F., & Toni, F. (2000). Abduction with negation as failure for active and reactive rules. In E. Lamma & P. Mello (Eds.), *Ai*ia 99: Advances in artificial intelligence* (pp. 49–60). Springer Berlin Heidelberg.

Sandholm, W. H. (2009). Evolutionary game theory. In *Encyclopedia of complexity and systems science* (pp. 3176–3205). Springer New York. https://doi.org/10.1007/978-0-387-30440-3_188

Sarkadi, S., Panisson, A. R., Bordini, R. H., McBurney, P., Parsons, S., & Chapman, M. (2019). Modelling deception using theory of mind in multi-agent systems. *AI Communications*, *32*, 287–302. https://doi.org/10.3233/AIC-190615

Sarmasi, A., Zhang, T., Cheng, C.-H., Pham, H., Zhou, X., Nguyen, D., Shekdar, S., & McCoy, J. (2021). Hoad: The hanabi open agent dataset. *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 1646–1648.

Sarr, S., Hayes, B., & DeCaro, D. A. (2021). Applying ostrom's institutional analysis and development framework, and design principles for co-production to pollution management in louisville's rubbertown, kentucky. *Land Use Policy*, *104*, 105383. https://doi.org/10.1016/j.lusepol.2021.105383

Savarimuthu, B. T. R., & Cranefield, S. (2011). Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems*, *7*(1), 21–54. https://doi.org/10.3233/mgs-2011-0167

Scherl, R. B., & Levesque, H. J. (2003). Knowledge, action, and the frame problem. *Artificial Intelligence*, *144*(1-2), 1–39. https://doi.org/10.1016/s0004-3702(02)00365-x

Schiffel, S., & Thielscher, M. (2014). Representing and reasoning about the rules of general games with imperfect information. *Journal of Artificial Intelligence Research*, *49*, 171–206. https://doi.org/10.1613/jair.4115

Schwartz, S. H. (1992). Universals in the content and structure of values: Theoretical advances and empirical tests in 20 countries. In *Advances in experimental social psychology* (pp. 1–65). Elsevier. https://doi.org/10.1016/s0065-2601(08)60281-6

Schwartz, S. H. (1994). Are there universal aspects in the structure and contents of human values? *Journal of Social Issues*, *50*(4), 19–45. https://doi.org/10.1111/j.1540-4560.1994.tb01196.x

Schwartz, S. H. (2012). An overview of the Schwartz theory of basic values. *Online Readings in Psychology and Culture*, *2*(1). https://doi.org/10.9707/2307-0919.1116

Searle, J. R. (1983). *Intentionality: An essay in the philosophy of mind*. Cambridge University Press. https://doi.org/10.1017/cbo9781139173452

Serramià, M., López-Sánchez, M., & Rodríguez-Aguilar, J. A. (2020). A qualitative approach to composing value-aligned norm systems. *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1233–1241.

Serramià, M., López-Sánchez, M., Rodríguez-Aguilar, J. A., Morales, J., Wooldridge, M., & Ansotegui, C. (2018). Exploiting moral values to choose the right norms. *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. https://doi.org/10.1145/3278721.3278735

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3-4), 591–611. https://doi.org/10.1093/biomet/52.3-4.591

Shapley, L. S. (1951). *Notes on the n-person game - ii: The value of an n-person game*. RAND Corporation. https://doi.org/10.7249/RM0656

Shoham, Y., & Leyton-Brown, K. (2014). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, *73*(1-2), 231–252. https://doi.org/10.1016/0004-3702(94)00007-n

Sierra, C. (2022). *Value engineering in autonomous agents* [Invited Talk at the 19th European Conference on Multi-Agent Systems (EuMAS 2022)]. url%20link%20to%20talk%20abstract%20if%20any

Sierra, C., Osman, N., Noriega, P., Sabater-Mir, J., & Perelló-Moragues, A. (2019). Value alignment: A formal approach. *Responsible Artificial Intelligence Agents Workshop (RAIA) in AAMAS 2019*.

Sindlar, M., Dastani, M., & Meyer, J.-J. (2011). Programming mental state abduction. *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, 301–308.

Sindlar, M. P., Dastani, M. M., & Meyer, J.-J. C. (2009). Bdi-based development of virtual characters with a theory of mind. In Z. Ruttkay, M. Kipp, A. Nijholt, & H. H. Vilhjálmsson (Eds.), *Intelligent virtual agents* (pp. 34–41). Springer Berlin Heidelberg.

Siu, H. C., Peña, J. D., Chang, K. C., Chen, E., Zhou, Y., Lopez, V. J., Palko, K., & Allen, R. E. (2021). Evaluation of human-ai teams for learned and rule-based agents in hanabi. *CoRR*, *abs/2107.07630*. https://arxiv.org/abs/2107.07630

Smajgl, A., Izquierdo, L. R., & Huigne, M. (2008). Modeling endogenous rule changes in an institutional context: The adico sequence. *Advances in Complex Systems*, *11*(02), 199–215. https://doi.org/10.1142/s021952590800157x

Spates, J. L. (1983). The sociology of values. *Annual Review of Sociology*, *9*(1), 27–49. https://doi.org/10.1146/annurev.so.09.080183.000331

Štrumbelj, E., & Kononenko, I. (2013). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, *41*(3), 647–665. https://doi.org/10.1007/s10115-013-0679-x

Sukthankar, G., Geib, C., Bui, H., Pynadath, D., & Goldman, R. P. (2014). *Plan, activity, and intent recognition: Theory and practice*. Elsevier Science & Technology Books.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction: An introduction*. Bradford Books.

Szabo, J., Such, J. M., & Criado, N. (2020). Understanding the role of values and norms in practical reasoning. In N. Bassiliades, G. Chalkiadakis, & D. de Jonge (Eds.), *Multi-agent systems and agreement technologies* (pp. 431–439). Springer International Publishing.

Tager-Flusberg, H. (2007). Evaluating the theory-of-mind hypothesis of autism. *Current Directions in Psychological Science*, *16*(6), 311–315. https://doi.org/10.1111/j.1467-8721.2007.00527.x

Teze, J. C. L., Perelló-Moragues, A., Godo, L., & Noriega, P. (2019). Practical reasoning using values: An argumentative approach based on a hierarchy of values. *Annals of Mathematics and Artificial Intelligence*, *87*(3), 293–319. https://doi.org/10.1007/s10472-019-09660-8

The World Bank, Development Research Group. (2019). Gini index (world bank estimate, 1967-2019) [Accessed 7th June 2021, http://data.worldbank.org/indicator/SI.POV.GINI].

Thielscher, M. (2017). GDL-III: A description language for epistemic general game playing. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 1276–1282. https://doi.org/10.24963/ijcai.2017/177

van den Bergh, M. J. H., Hommelberg, A., Kosters, W. A., & Spieksma, F. M. (2017). Aspects of the cooperative card game hanabi. In T. Bosse & B. Bredeweg (Eds.), *Bnaic 2016: Artificial intelligence* (pp. 93–105). Springer International Publishing.

van de Poel, I. (2020). Embedding values in artificial intelligence (AI) systems. *Minds and Machines*, *30*(3), 385–409. https://doi.org/10.1007/s11023-020-09537-4

van der Hoek, W. (1993). Systems for knowledge and belief. *Journal of Logic and Computation*, *3*(2), 173–195. https://doi.org/10.1093/logcom/3.2.173

van der Torre, L. (2003). Contextual deontic logic: Normative agents, violations and independence. *Annals of Mathematics and Artificial Intelligence*, *37*(1/2), 33–63. https://doi.org/10.1023/a:1020207321544

van der Weide, T. L., Dignum, F., Meyer, J. .-. C., Prakken, H., & Vreeswijk, G. A. W. (2010). Practical reasoning using values. In *Lecture notes in computer science* (pp. 79–93). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-12805-9_5

Visser, S., Thangarajah, J., Harland, J., & Dignum, F. (2015). Preference-based reasoning in BDI agent systems. *Autonomous Agents and Multi-Agent Systems*, *30*(2), 291–330. https://doi.org/10.1007/s10458-015-9288-2

von Wright, G. H. (1951). Deontic logic. *Mind*, *60*(237), 1–15. http://www.jstor.org/stable/2251395

Walton, D. (2014). *Abductive reasoning*. University of Alabama Press.

Walton-Rivers, J., Williams, P. R., Bartle, R., Perez-Liebana, D., & Lucas, S. M. (2017). Evaluating and modelling hanabi-playing agents. *2017 IEEE Congress on Evolutionary Computation (CEC)*, 1382–1389. https://doi.org/10.1109/CEC.2017.7969465

Wang, Y., Zhong, F., Xu, J., & Wang, Y. (2022). Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind. *International Conference on Learning Representations*. https://openreview.net/forum?id=M3tw78MH1Bk

Weymark, J. (2016). Social welfare functions. In *The oxford handbook of well-being and public policy* (pp. 126–159). Oxford University Press. https://doi.org/10.1093/oxfordhb/9780199325818.013.5

Williams, J., Fiore, S. M., & Jentsch, F. (2022). Supporting artificial social intelligence with theory of mind. *Frontiers in Artificial Intelligence*, *5*. https://doi.org/10.3389/frai.2022.750763

Winikoff, M., Sidorenko, G., Dignum, V., & Dignum, F. (2021). Why bad coffee? explaining BDI agent behaviour with valuings. *Artificial Intelligence*, *300*, 103554. https://doi.org/10.1016/j.artint.2021.103554