I-ABM: Combining Institutional Frameworks and Agent-based Modelling for the Design of Enforcement Policies

Abstract. Computer science advocates institutional frameworks as an effective tool for modelling policies and reasoning about their interplay. In practice, the rules or policies, of which the institutional framework consists, are often specified using a formal language, which allows for the full verification and validation of the framework (e.g. the consistency of policies) and the interplay between the policies and actors (e.g. violations). However, when modelling large-scale realistic systems, with numerous decision-making entities, scalability and complexity issues arise making it possible only to verify certain portions of the problem without reducing the scale.

In the social sciences, agent-based modelling is a popular tool for analysing how entities interact within a system and react to the system properties. Agent-based modelling allows the specification of complex decision-making entities and experimentation with large numbers of different parameter sets for these entities in order to explore their effects on overall system performance.

In this paper we describe how to achieve the best of both worlds, namely verification of a formal specification combined with the testing of large-scale systems with numerous different actor configurations. Hence, we offer an approach that allows for reasoning about policies, policy making and their consequences on a more comprehensive level than has been possible to date. We present the Institutional Agent-Based Model (I-ABM) methodology to combine institutional frameworks with agent-based simulations). We furthermore present J-Inst*AL*, a prototypical implementation of this methodology using the Inst*AL* institutional framework whose specifications can be translated into a computational model under the answer set semantics, and an agent-based simulation based on the JASON tool. Using a simplified contract enforcement example, we demonstrate the functionalities of this prototype and show how it can help to assess an appropriate fine level in case of contract violations.

Keywords: Policies, Institutional framework, Agent-based model, Logic programming, Enforcement

1. Introduction

The focal point of this paper is the combination of the best of both institutional frameworks (as a mechanism for formalizing a collection of policies) and agent-based models (as a mechanism for empirically studying the interaction of a large number of participants, referred to as agent) to both formally and empirically explore policy outcomes.

This brief introduction demands qualification and clarification with respect to the several concepts which it brings together:

– In line with the substantial literature (see for example Noriega (1997), North (1994), Ostrom (1990) and Vázquez-Salceda (2003)) and for the

purposes of this paper, an institutional model (collection of norms and/or policies) serves to express normative constraints over individual actions.

– Following Vickers (1973), we view policies as instruments to implement norms, which are used by policy makers to encourage members of society to modulate their behaviour with respect of norms. In other words, norms are a precise specification of what society expects and what its participants are permitted and/or obliged to do. Consequently, policies are the combination of norms and enforcement mechanisms that aim to bring about adherence to the said norms.

The I-ABM methodology and the J-Inst*AL* prototype presented in this paper address the evaluation of policy outcomes in two stages:

– The formal modelling of policies underpinned by the construction of a mathematical or logical foundation for the formal model, making it amenable to formal analysis, including correctness, completeness and computational complexity. This subsequently enables formal verification, through the translation of the mathematical model into an equivalent computational model, that permits the testing of static properties. The application of this process to a different scenario is presented in Balke et al. (2011), but is not pursued further here.

– An ABM, on the other hand, allows for an empirical validation in terms of policy requirements by enabling the study of the effects (the reaction of the target audience of the policies) in a variety of realistic settings, possibly with minor variations of the policies. The experiments should for example take into account different reactions to and interaction with the policies of the target audience. Hence, the testing of dynamic properties is also achievable.

The modelling of institutional frameworks has been subject to research for several decades (a comprehensive discussion appears in Jones and Sergot (1993) and more recently in Grossi (2007)). Looking at the literature on the computational implementation of policies and institutional frameworks, one finds that they are often encoded as formulas in some logic language, where some components of the formula represent notions of obligation, permission and prohibition, which are linked by rules (Alberti et al., 2012). The advantages of this form of implementation are three-fold:

Intuitive formulation: Rules correspond intuitively to conditional statements, such as that some consequence (e.g. the obligation to perform an action) follows from a state of affairs.

Verification: If an institutional framework is so represented, the properties of the framework can often be verified (e.g. the consistency of policies) or possible institutional states reasoned about (e.g. violations).

Comprehensive background literature: The underlying properties of logic languages and the (computational) implementation and verification of systems, have been widely studied, providing comprehensive resources for modelling purposes as well as mature implementations (Ligeza, 2006).

There are however drawbacks too, specifically with respect to scalability and model complexity: although it is theoretically possible to model large scale complex systems with many different decision entities, verifying every possible state is an exponential task (Ågotnes et al., 2007). Institutional frameworks typically focus on the institutional properties and abstract away the decision making entities. Modelling this decision making – especially when taking into account complex human behaviour – in order to analyse the impact of policies in more detail, though feasible, seems at least difficult to achieve with logic language models. Furthermore, running experiments with several different settings for the different decision entities in the system to test and analyse the policies under different circumstances requires new models for each experimental setting, again increasing the complexity of the approach. Finally, an additional problem is that policy formulations are often expressed in quantitative and narrative form which is difficult to represent in logic programming.

One research area that addresses directly the interpretation of quantitative and narrative information, as well as the modelling of complex large scale systems with different decision entities, is that of agent-based modelling (ABM) (also often referred to as agent-based simulation) (Bonabeau, 2002). In broad terms, an agent-based model describes how a computational representation of a scenario is constructed in terms of the agents that populate it, their behaviours and the policies that govern them, for the purpose of exploring the empirical properties of the parameters that characterise the setting. In ABM the actors of a system (which could for example be individuals, firms or even larger entities), are each represented by a software agent that acts, and interacts, within a simulation or computational setting, and whose actions are determined by some internal decision procedure, typically depending on internal state, including a memory of past interactions. The traditional focus of ABM is to model the individual decision making of the system entities on a micro level in the agents, and then to let the agents engage or interact with one another to observe the macro results of this interaction at a system level. There are many approaches to the encoding of agent decision making, such as (i) rule-based designs, (ii) BDI or BOID architectures (Rao and Georgeff, 1995; Broersen et al., 2001), which are conceptually more similar to human decision making, (iii) extensions of those incorporating cognition and emotion (Andrighetto et al., 2007), or (iv) Jager's Consumat model (Jager, 2000).

The two main advantages of ABM with respect to modelling and analyzing the impact of policies on human entities are that it (i) supports representing human decision making, and (ii) is typically designed for testing with different parameter settings. This allows hypotheses about whether, why and how the represented system behaves in certain circumstances.

Thus, the purpose of ABM is not to focus on the formal verification of the system, but on representing the behaviour and decision making of entities in order to observe the system behaviour as these interact. This makes it difficult using an ABM on its own to reason about the consistency of the policies of an institutional framework or the states of the framework represented. For an efficient and effective modelling and analysis of policies and institutional frameworks, the logic language representation and the ABM are important. That is why in this paper we advocate the I-ABM methodology for combining of these two approaches to bring together the advantages of both and reason about policies and policy making on a more comprehensive level than so far feasible. We present the J-Inst*AL* prototype combining an institutional framework whose specifications can be translated into a computational model under the answer set semantics (Gelfond and Lifschitz, 1991), and an ABM. Using a simple contract enforcement example, we demonstrate the functionalities of this prototype.

The remainder of the paper is structured as follows. In the next two sections we outline the foundations of our work by presenting the concepts of institutional frameworks and agent-based simulations and their complementary advantages and disadvantages. In Section 4, we present the main contribution of this paper: the I-ABM methodology and the Inst*AL* prototype to combine agent-based simulations and institutional frameworks In Section 5, we demonstrate the functionalities of our methodology and prototype by applying it to a simple contract enforcement case study. Afterwards we compare our approach to related work in Section 6. The paper ends with a short summary and conclusions.

2. Institutional Frameworks

The usual product of a modelling process, in which the focus is on capturing and defining the behaviour of its various components, is a specification of how each kind of component shall act and interact. In contrast, an institutional modelling process focuses on capturing the forces (physical, social, legal, as desired) that affect the elements and hence regulate their behaviour.

Institutional models are specified using a number of formalisms, depending on which aspects modellers consider important, but in essence there are two perspectives: states (that identify institutional positions) and events (that bring about changes in institutional positions). For example, Opera (Okouya

and Dignum, 2008) uses boolean combinations of facts to express institutional states while Inst*AL* (Cliffe, 2007) and its underlying formal model use events to bring about updates to an institutional state. These aspects are inextricably linked in the sense that we can write down a (institutional) system trace as:

$$s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} s_2 \xrightarrow{e_2} s_3 \ldots s_u$$

where s_i denotes an (institutional) system state and e_i an (institutional) event at time-step i.

The objective of much research on institutional system concerns the establishment of properties in the limit—using such traces—such as whether a particular state is ever (or never) reachable and which sequences of events can precede or succeed a particular state (Clarke et al., 1986). Clearly this is very useful but only establishes static properties, whereas it is also critical to verify dynamic characteristics arising from specific traces, leading to better understanding of policies and their consequences in practice.

More realistically, the designer wants to discover properties such as which (type of) participants are not adhering to the policies prescribed by the system (e.g. not satisfying an obligation or carrying out a prohibited action or reaching an undesired institutional state). Policy compliance can be achieved by using appropriate enforcement mechanisms, like penalties that are imposed on those participants that deviate from the policy. These enforcement mechanisms often come at a cost (e.g. police, courts of law), indicating that with respect to the dynamic properties of an institutional system a balance has to be found between full normative compliance and enforcement costs.

2.1. INST*AL*

We use Inst*AL* (Cliffe et al., 2007) for the specification of the institutions that govern the agents in our simulation. Inst*AL* specifications are written in terms of events (rather than states), which permits the establishment of a connection between the violation of a norm and the participant causing this violation. Furthermore, this approach has a computational model derived from a mathematical foundation. We provide a brief introduction to the formalism but refer the interested reader to Cliffe (2007) and De Vos et al. (2011). An example of formalism can can be found in Section 5.

2.1.1. *Syntax*

Inst*AL* is based on a set-theoretic model, and thus is expressed through sets and relations between their elements. The formalisation is given in Figure 1, the elements of which we now describe in more detail, followed by the semantics, and which together provide a mathematical alternative to the logic-

based approaches such as Opera (Okouya and Dignum, 2008) or deontic logic (Governatori and Rotolo, 2004).

In the terminology used for an institutional framework ($\mathcal{I}$), we distinguish two types of events ($\mathcal{E}$): *exogenous* ($\mathcal{E}_{ex}$) and *institutional* ($\mathcal{E}_{inst}$) events. From the perspective of a policy domain, the first represent events external to the institutional framework, such as the actions of the participants: e.g. signing a contract; while institutional events are a mechanism for the legal interpretation of these actions. Events, both exogenous and institutional, may trigger institutional events in a direct reflection of "counts-as" (Jones and Sergot, 1996), to allow for events to be interpreted differently according to context. For example, raising your hand in class counts as indicating you want to ask a question while raising a hand during an auction indicates you wish to bid.

Events change the state of the institutional system by initiating and terminating fluents ($\mathcal{F}$). Fluents are properties of the state that can be initiated (become true) and remain true until they are terminated (made false). A state is represented by the fluents that are true in this state. Fluents not in the state are considered false. The institution starts with an initial state Δ. Two types of fluents exist: *domain* ($\mathcal{D}$) and *institutional fluents*. For the latter we distinguish three types of institutional fluents: power ($\mathcal{W}$) ($\texttt{pow}(\texttt{e})$ indicates that the event e is currently recognised by the institution and can bring about institutional state), permission ($\mathcal{P}$) ($\texttt{perm}(\texttt{e})$ denotes that the event e is permitted to occur) and obligation fluents ($\mathcal{O}$) ($\texttt{obl}(\texttt{e},\texttt{d},\texttt{v})$ indicates that the event must occur before the deadline event d otherwise a the violation event e occurs). Empowerment indicates that the institutional framework is able to interpret the occurrence of an exogenous event in the institutional context and possibly generate corresponding institutional events (based on count-as). A typical example is the marriage institutional framework: only an individual with power vested in him/her can marry two people, otherwise the marriage is not legally recognised. An event which is not permitted will still occur within the framework but will also raise a violation event. Events are empowered/permitted if their associated `pow`/`perm` fluent is true in the current state. Various interpretations of obligations exist in the literature (Governatori and Rotolo, 2004; Prakken and Sergot, 1996). In our model, obligations are used to indicate that an event should occur before a second event otherwise a violation is raised. An obligation is terminated as soon as the obligation is fulfilled or violated. In other works, obligation remain valid (Governatori and Rotolo, 2004; Prakken and Sergot, 1996). In the Inst*AL* model, permission needs to be granted, i.e. events are only permitted if its permission is part of the state. Deontic prohibition is through the absence of permission.

Institutional frameworks are interested in tracking violations to their norms. In Inst*AL* this is done through violation events ($\mathcal{E}_{viol}$). They can occur in two ways: either they are explicitly specified by the designer or are generated by

$\mathcal{I} = \langle \mathcal{E}, \mathcal{F}, \mathcal{C}, \mathcal{G}, \Delta \rangle$, where

1. $\mathcal{E} = \mathcal{E}_{ex} \cup \mathcal{E}_{inst}$ with $\mathcal{E}_{inst} = \mathcal{E}_{act} \cup \mathcal{E}_{viol}$
2. $\mathcal{F} = \mathcal{W} \cup \mathcal{P} \cup \mathcal{O} \cup \mathcal{D}$
3. $\mathcal{C} : \phi \times \mathcal{E} \rightarrow 2^{\mathcal{F}} \times 2^{\mathcal{F}}$ where
   $C(X, e) = (\mathcal{C}^{\uparrow}(\phi, e), \mathcal{C}^{\downarrow}(\phi, e))$ and where
   (i) $\mathcal{C}^{\uparrow}(\phi, e)$ initiates a fluent
   (ii) $\mathcal{C}^{\downarrow}(\phi, e)$ terminates a fluent
4. $\mathcal{G} : \phi \times \mathcal{E} \rightarrow 2^{\mathcal{E}_{inst}}$
5. State Formulae: $\phi = 2^{\mathcal{F} \cup \neg \mathcal{F}}$
6. Initial State: Δ

Figure 1. Formal specification of the institutional framework

the institution (see the later text on the Inst*AL*'s semantics) by the system with the occurrence of a non-permitted event or an unsatisfied obligation.

Changes in an institutional state are achieved by two relations: (i) the generation relation ($\mathcal{G}$), which implements counts-as by specifying how the occurrence of one (exogenous or institutional) event generates another (institutional) event, subject to the empowerment of the institutional event and the conditions on the state, and (ii) the consequence relation ($\mathcal{C}$), which specifies the initiation and termination of fluents, subject to the performance of some action in a state matching some condition. Conditions on a state (ϕ) are expressed by a set of fluents that should be true or false.

2.1.2. *Semantics*

The semantics of an institution is defined over a sequence, called a *trace*, of exogenous events. Starting from the initial state (Δ), each exogenous event is responsible for a state change, through initiation and termination of fluents. This is achieved by a three-step process: (i) the transitive closure of $\mathcal{G}$ with respect to a given exogenous event determines all the generated (institutional) events, (ii) to this, all violation events corresponding to the occurrence of event which was not permitted are added, and (iii) the specified violation event v for an obligation $\texttt{obl}(\texttt{e}, \texttt{d}, \texttt{v})$ for which the deadline event d has occurred. (iv) resulting in the set of all events whose consequences determine the new state, (v) the application of $\mathcal{C}$ to this set of events identifies all fluents that are initiated and terminated with respect to the current state, so determining the next state. For each trace, the model can therefore compute a sequence of states that constitutes the model of the institutional framework for that trace.

While a formal specification is useful for verification purposes, it does not offer the designer any tools to support the verification. Manual verification is

difficult, tedious and error-prone. To provide better support for the designer, an institutional specification language using semi-natural language was created and a computational model of the formal specification was developed (Cliffe et al., 2007). The computational model is expressed using answer set programming, which generates *all* the traces of the institutional model as solutions of the program. Constraints can be added to the computational model to select those traces that are relevant. Using the approach the specification can be analysed and verified. In the next paragraphs we give a brief overview of the specification language and the computational model. For specific details, we invite the reader to look at (Cliffe et al., 2007).

2.1.3. *InstAL*

An Inst*AL* specification starts with declarations of types and of the events and fluents (parameterized by types) that comprise the model. The next part defines the rules that make up the generation relation (using the keyword `generates`) and the consequence relation (using the keywords `initiates` and `terminates`). Finally comes the specification of the initial state of the institution. An example of an Inst*AL*specification is given in Section 5.

2.1.4. *Answer Set Programming*

Both the formal model and the Inst*AL* specification are realised by a translation to AnsProlog (Baral, 2003), a declarative non-monotonic programming language that uses logic programs under the answer set semantics (Gelfond and Lifschitz, 1991) to represent the problem as a logic program and derive the solutions as its answer sets. Declarative programming languages have the advantage that the problem being modelled and the requirements for the solution can be described without the need to provide an algorithm to solve the problem. The interpreter/compiler finds the solutions to the problem that meet the specified requirements. The basic components of the language are atoms, elements that can be assigned a truth value. An atom can be negated using *negation as failure*, something is false unless it is proven to be true. *Literals* are atoms a or negated atoms $\mathrm{not}\ a$. Atoms and literals are used to create rules of the general form: $a \leftarrow b_1, ..., b_m, \mathrm{not}\ c_1, ..., \mathrm{not}\ c_n$, where a, b_i and c_j are atoms. Intuitively, this means *if all atoms b_i are known/true and no atom c_j is known/true, then a must be known/true*. We refer to a as the head and $b_1, ..., b_m, \mathrm{not}\ c_1, ..., \mathrm{not}\ c_n$ as the body of the rule. Rules with empty body are called *facts*. Rules with empty head are referred to as *constraints*, indicating that no solution should be able to satisfy the body. A *program* is a set of rules. The semantics of AnsPrologis defined in terms of *answer sets*, i.e. assignments of true and false to all atoms in the program such that the rules are satisfied in a minimal and consistent fashion. A program may have zero or more answer sets, each corresponding to a solution of the problem being modelled. An AnsProlog program is given to a so-called answer set

solver which returns the answer sets of the program. For our implementation we use CLINGO (Gebser et al., 2007), a state-of-the-art answer set solver.

The implementation of an institution consists of three parts: a *base component* which is independent of the institution being modelled, a *time component* and the *institution-specific component*. The base component deals with inertia of the fluents (i.e. making sure that initiated fluents remain true until they are terminated), the generation of violation events of non-permitted actions and unfulfilled obligations. Furthermore it terminates fulfilled and violated obligations. The time component defines the predicates for time and is responsible for generating a single exogenous event at every time instance. The mapping uses the following atoms: $\texttt{ifluent}(\texttt{p})$ to identify fluents, $\texttt{evtype}(\texttt{e},\texttt{t})$ to describe the type of an event, $\texttt{event}(\texttt{e})$ to denote the events, $\texttt{instant}(\texttt{i})$ for time instances, $\texttt{final}(\texttt{i})$ for the last time instance, $\texttt{occurred}(\texttt{e},\texttt{i})$ to indicate that the exogenous or empowered institutional event e took place at time i, $\texttt{observed}(\texttt{e},\texttt{i})$ that the exogenous event e was observed at time i, $\texttt{holdsat}(\texttt{p},\texttt{i})$ to state that the normative fluent p holds at i, and finally $\texttt{initiated}(\texttt{p},\texttt{i})$ and $\texttt{terminated}(\texttt{p},\texttt{i})$ for fluents that are initiated and terminated at time i.

The full details of the translation of the institutional specification to AnsProlog can be found in Cliffe et al. (2007), including a proof of the soundness and completeness of the translation. The result of the translation is a computational model that can generate all traces of a specified length and their corresponding sequence of models of all possible states of the institutional framework as the answer sets of the program. If desired, this computational model can be supplemented by a so-called trace program that specifies the specific exogenous events that need to take place at a certain time. The program will then only generate those traces that have these events occurring. If a complete – an exogenous event for each time instance – trace is provided, a single answer set is produces that specifies the sequences of states that are a consequence of this sequence of events.

3. Agent-based Simulations for Policy Analysis

Computational models in the natural sciences, engineering and legal research, typically rely on either equation- or logic-based modelling. This makes it hard to transfer them to the study of the interaction of human actors in a system, as human behaviour is complex and difficult to formalize mathematically (Helbing and Balietti, 2011) and poses severe issues in terms of computational complexity. However, when analysing the impact of policies on humans in a system, representing these humans appropriately in order to get a realistic picture of their interaction with the system is critical. One approach, that has become very popular in the social sciences, is to model

human behaviour using agent-based modelling (ABM) (Epstein and Axtell, 1996; Gilbert and Troitzsch, 2005; Macy and Willer, 2002) with "agents" representing the actors of the system to be modelled.[1].

In the next paragraphs we discuss the advantages and disadvantages of ABM in the context of modelling human behaviour, given the focus of this a paper is on policy modelling. Most of these are also valid comments in different contexts.

By using ABM for computational experiments, one may test in a systematic way different hypotheses related to attributes of the agents, their behavioural rules, and the types of interactions, and their effect on macro-level stylized facts about the system.

To understand better what is understood by the term agent and as a consequence by ABM, we will now look more closely at the definition of the term *agent* in the computer science context[2]. In this context agents are referred to as reactive systems (e.g. pieces of software) that exhibit some degree of *autonomy* in the sense that if being delegated a task or goal, the system determines how to achieve this goal. An important difference with other modelling approaches is that, rather than being given low-level detail on how to fulfil a task, the agents pursue goals actively and decide themselves how best to accomplish their goals with the (possibly limited) amount of resources they possess.

When looking at the properties of agents, agents are systems that are considered to be situated in some environment. Agents are capable of sensing their environment via sensors and have a number of possible actions that, based on their internal reasoning and decision making (with regard to their multiple and possible conflicting goal(s)), that they can decide to perform in order to affect the environment. This environment that the agents populate and interact with can be physical (e.g. robots that inhabit a physical world) or a software environment such as a computer simulation.

Besides being situated in and interacting with an environment, further properties are attributed to agents (Wooldridge and Jennings, 1995), which facilitate thinking of agents in terms of the human decision makers that they represent, interacting in a given policy setting.

Autonomy As mentioned earlier, agents operate independently in order to achieve goals being delegated to them by their principals. Thereby, they make independent decisions on how to achieve these delegated goals.

[1] The computational modelling technique corresponding to agent-based modelling is typically referred to as agent-based simulation or multi-agent simulation (Gulyás, 2005).

[2] The explanation of the terms agent and ABM in this paper follows that of Balke (2011) to a large extent.

Proactiveness Proactiveness is very closely linked to goal delegation. It implies that agents exhibit goal-directed behaviour. Hence, given a goal, an agent is expected to actively work to achieve this goal.

Reactivity Agents respond to changes in the environment. This also implies that if conditions on which they based their earlier decision change, they can adapt to these changes and change their plans for how to achieve their goals accordingly.

Social Ability This refers to the property of agents to be able to cooperate and coordinate activities with other agents (including a communication at knowledge level where agents are able to communicate their goals, beliefs and plans).

Humans act in the environment which is governed by different institutions and interact with other users, each being driven by their own objectives (e.g. financial gains) as well as being constrained by limitations (e.g. their physical limitations). The actions that the users perform are on the one hand based on their perception of their environment, (e.g. policies and institutions, or of the other users and their actions), and on the other hand on their resources, as well as considerations of how their actions could reflect on their own goals.

So far in this section we have mainly discussed single agents and their properties. However, what makes agents and ABM particularly relevant for policy modelling is their *social ability*, which allows us to analyse situations in which agents interact with each other. An ABM describes a system from the bottom-up, i.e. from the perspective of its constituent (possibly heterogeneous) units. The macro result on the global systems level is perceived as a result of the interaction of the constituent entities on the micro level. The overall aim is to observe emergent global system behaviour resulting from the sum of the individual actions of the units (Holland, 1992).

Although it might be theoretically feasible to model human behaviour with the help of formal methods, humans (and their behaviour) exhibit certain features that can more suitably modelled with ABMs (Bonabeau, 2002):

– Human behaviour tends to be complex and non-linear. For example, it exhibits memory effects, forms of learning and adaptation, path dependence, temporal correlations and non-Markovian components. As a result, for modelling purposes it is difficult to capture human behaviour in a purely analytical form. One particular problem is that the complexity of this analytical model increases exponentially as the complexity of behaviour increases.

– Human behaviour is characterised by stochasticity. Looking how the different approaches address this stochasticity, logic (programming) approaches do not tend to account for this, but rather focus on proving

properties assuming "ideal" conditions and agent behaviour. Mathematical equation-based approaches tend to use a "noise term" that is added to an aggregate equation. Finally ABM simulations allow one to add sources of randomness at specific and appropriate points in the agent's reasoning process, thereby allowing for more heterogeneity of the agents and a better representation of human decision making.

– Humans tend to act and base decisions on local information and their limited knowledge. Formal solutions (i.e. logic (programming) and mathematical solutions) very often assume global knowledge, whereas ABM can easily represent this local focus.

– Finally, humans and their interactions are heterogeneous. The heterogeneous interaction can generate network effects that may deviate a lot from predicted aggregate behaviour, based on the emergent network topology of the individuals interacting. Representing a system from the agent's perspective takes this into account. In contrast, purely mathematical systems typically assume global homogeneous mixing which mainly portrays aggregate behaviour and does not account for any network topology, etc.

In addition to these advantages, when trying to represent humans and their decision making process, ABM simulations offer further advantages with regard to the flexibility of the analysis. In an ABM simulation it is easy to add more (or take away) agents, i.e. scaling the system up (or down) as required. But not only is the number of agents easy to scale, the complexity of the agents (e.g. their degree of rationality or their ability to learn and evolve) as well as the level of description and aggregation of the system are also flexible. Thus, one can easily analyse a system with certain groups of agents or single agents, and work with different facets of the system description.

Despite all these advantages with respect to modelling human behaviour, due to the focus on bottom-up features, incorporating a top-down institutional perspective is difficult in a pure ABM. As a result, we integrate institutional frameworks, which offer the possibility to reason about top-down norms, with ABM in order to be able to form "institutional agent-based models".

4. I-ABM: Combining Institutional Frameworks and Agent-based Simulations

Summing up from the previous section, ABM combined with an institutional framework, offers three advantages: (i) the agents participate in the institutional framework, enabling the investigation of its dynamic properties, (ii) the

institutional framework regulates the behaviour agents, enabling the exploration of dynamic individual behaviours, and (iii) the combination of both allows an easier analysis of large scale complex settings (with different system and agent settings). Thus, we argue that institutional agent-based models open up new possibilities both for institutional framework research and for ABM research, which we seek to demonstrate by means of the contract law example in Section 5. We refer to the combination of agent based modelling techniques with institutional models as Institutional Agent-Based Modelling (I-ABM).

Following the presentation of the components of an I-ABM in the preceding sections, we now describe how they fit into the overall architecture of the I-ABM, explaining the interplay of the individual components. The next section addresses the case study. Our prototype I-ABM , using the Inst*AL* institutional framework and the JASON multi-agent environment as our simulation environment is referred to as J-Inst*AL*.

The UML component diagram in Figure 2 shows the components of an I-ABM, which consists of three components:

1. The institutional framework that contains the policy specifications and a mechanism to translate the specification into a computational model. In J-Inst*AL*, we use Inst*AL* as our specification language and use the program PYINSTAL[3] to translate the specification into AnsProlog.
2. The ABM that represents the actors and their behaviour, and we use JASON[4] as our simulation environment.
3. An institutional monitor, that monitors the actions of the participants, updates institutional state(s), keeps track of the different instantiations of the institutional framework and helps to translate the institutional specifications for the agents. For updating or querying institutional states, the J-Inst*AL*'s Java based monitor uses CLINGO[5].

Each of these three components has distinct tasks in our architecture. When analysing and verifying the effect of policies from a formal perspective the object both agent behaviour and institutional framework are considered together. Now, for the purpose of simulation, we need to separate agents and institution. The purpose of the institutional framework is to encapsulate the policy component and to observe and keep track of institutional state, not whole system behaviour. Thus, it only monitors the external events resulting from agents' actions and does not pre-determine agent behaviour. That is why in the context of simulation, we no longer are concerned with modelling agents and their decision making in the institutional framework, because that has now become the task of the ABM. For J-Inst*AL* we have chosen the JASON agent platform (Bordini et al., 2007) as the basis of our

[3] http://instsuite.cs.bath.ac.uk/
[4] http://jason.sourceforge.net/
[5] http://potassco.sourceforge.net/

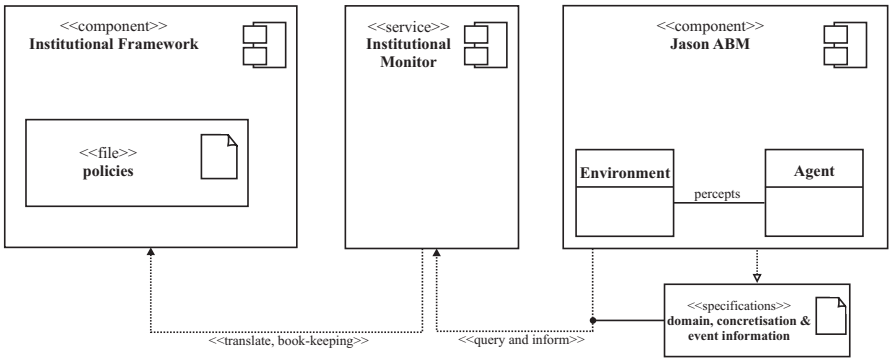

Figure 2. The institutional ABM architecture

ABM. Our architecture is not tied to JASON specifically: we chose it because it allows us to define BDI-agents using an extended version of AgentSpeak – a logic-based agent-oriented programming language – and because the implementation language is Java, which we could easily extend to communicate with the institutional framework.

In order to decouple the institutional framework and the ABM and to maintain the institutional state in the former, we introduce a special type of agent or entity, which is the institutional monitor. This component acts as a kind of gatekeeper between the ABM and the institutional framework. It (i) handles all the instantiations of the institutional framework, (ii) stores the institutional states and the respective grounding information, and (iii) helps to decouple the institutional framework and the agents. Because of the institutional monitor, the agents do not need to know any specifications of the internal structure, the semantics nor the syntax of the institutional framework, but can pass on information to the institutional monitor which then translates it into a form usable in the institutional framework component.

In Section 5 we present a simple contract case study to demonstrate our approach. In the case study agents can form contracts with one another and decide to live up to the contract or violate it. Using this example, the interaction between the three components just described works as follows: The ABM models the agents with their respective decision making processes and simulates their interaction with one another in the environment. When agents agree to interact (e.g. form a contract), they contact the institutional monitor with the specification of their agreed interaction to establish a new *contract* through a new instantiation of the institutional framework[6]. The specification passed to the institutional monitor includes a unique identifier that is used to distinguish the contract. The identifier is later used to iden-

[6] We assume here that the institutional framework includes policies that specify the procedure and the rules of contracting.

tify the correct instantiation of the institutional framework to be affected by the exogenous events generated from the agent's actions. The institutional monitor uses this specification to create an interaction-specific version of the institutional specification (i.e. it takes the general policies formulated in the institutional framework and replaces all the variables of the policies with the corresponding specific values (e.g. agent names,...) supplied by the agents). These specifications are then used to determine the initial state of the agents' contract (i.e. the initial state of the contract specific instantiation of the institutional framework). When the state needs updating (a new exogenous event takes place, such as for example a payment being made), using the contract identification, the institutional monitor retrieves the current state of the corresponding institutional framework. Using this as the initial state of the institution and the agent's action as a single event trace, the AnsPrologversion of the institutional framework provides the next state, which the institutional monitor stores for future use. In other words, the agents act, this action results in a new state of for the contract institution and the new state is stored. The role of ASP is to compute this new state.

Having the information for the initial contract as well as tracking the institutional state of each contract by analysing the respective exogenous events, the institutional monitor can act as a institutional query processor for the agents. Contracting agents can query the current state and discover the consequences of potential actions.

Possible queries include the following:

- queries about the current state, including the policies applying to that state (e.g. "What policies affect my current situation?" or "Given the current situation, following the policies, am I allowed to execute action ac_X?"),
- queries about the possible impact of the agent's own actions (e.g. "What is going to happen if I take action ac_X", and
- general queries on what might happen in the future (e.g. "What would happen if a series of actions (e.g. events $\mathcal{E}_{ex}(a)$,$\mathcal{E}_{ex}(b)$,$\mathcal{E}_{ex}(c)$ and $\mathcal{E}_{ex}(d)$) take place?").

Using this query information obtained via the institutional monitor, the agents can (but may not) incorporate the information into their decision making. This makes it possible to analyse the reactions of agents to the policies of the institutional framework (and the resulting effect of policies in a system). As agents themselves can decide to query the institutional monitor for institutional information, we are furthermore able to simulate policy awareness and its effects (e.g. agent might decide not to query the institutional monitor and make their decision without being aware of the exact institutional situation applying to them).

5. A Case Study: Simplified Contract Law

To demonstrate the application of I-ABM, we present a case study that analyses the effects of fines and detection probabilities on the fulfilment of contracts. In our view, contracts and policies have concepts in common with obligations, permissions, and prohibitions however they differ in terms of their focus. Whereas contracts are specific to their content, policies are more general and specify the procedure and rules of contracting. In the example, we are interested in the cumulative result of agents executing actions with respect to the deontic specifications (obligations, permissions,...). With respect to the case study, the aim of the simulation is to determine a suitable fine structure that maximizes the number of participants decide *not* to violate existing contract agreements. The simulation is used to determine a suitable fine, taking into account that only a certain number of violations will be actively enforced.

The case study itself serves to demonstrate the prototype, therefore we use a necessarily simplified example. We take our inspiration from contract law, although we have omitted what we consider to be the extraneous intricacies. We have also taken note of other work on optimal fine levels, such as that of Morales et al. (2011) on fines in traffic scenarios.

The example we use is as follows: When a buyer and a seller agree on a price for a good, both enter a contract binding one to pay and one to deliver the good. However, subsequent to entering the contract, either of them might receive a better offer. The seller could break the contract and sell to a buyer with a higher offer, or the buyer could accept an offer from a seller offering the good at a lower price intending not to pay for the first item bought. In either case this leads to a contract violation for which the wronged party could go to court in order to get compensation. Only a certain proportion of them will do so.

Incentives to encourage/discourage going to court over a contract violation are not part of this scenario. Furthermore, to keep the scenario simple, we assume that taking another contract is the only way to violate a contract and that when participants are committed to the contract they will pay and deliver on time. See (De Vos et al., 2011) for how other violations can be detected using an institutional framework. In the next sections, we first explain our simulation setting and then the institutional model that underpins our contract law example.

5.1. Simulation Set-up

In our simulation we have a number of agents that can act as both buyer and seller. The goods have no specific value and all agents have an "infinite" supply of money and goods. Agents are only interested in relative gains and

losses. Agents are permitted to hold only one contract as a buyer and one as a seller at any given time.

Encounters between agents take place in rounds. In each round (in random order) each agent can make decisions and act upon these decisions, resulting in events which are then interpreted by the institutional framework. It is a random decision whether an agent acts as a buyer or a seller in a given round. In the next step, a buyer and seller agent are randomly matched. A contract is established when both the buyer and seller accept. The seller agent takes the initiative. If it does not have a contract as a seller, it makes a random bid from a list of valid bids. Bids are fixed between 50 and 140 with an increment of 10. If the buyer agent does not hold a buyer contract, it accepts the bid.

In the case of a pre-existing bid, the agents determines whether it is worth violating this contract in favour of a new one. This decision is based the perceived risk of enforcement, previous gains and penalties from violations and the profit margin on the current bid.

Specifically, the seller agent with an existing contract randomly selects a higher bid (if available). If it decides to violate the existing contract, it makes an offer to the buyer agent. Only when the buyer accepts the offer, does it become a violation.

As mentioned previously, we assume that each contract can only be violated by agreeing a new contract before honouring the existing one, we make the indirect assumption that an agent can only hold one contract at a time. We assume that every contract that is not violated is automatically honoured after five simulation rounds.

In our simulation, the institutional monitor is not only responsible for detecting contract violations but also acts as the enforcer. To simulate the cost of enforcement we assume that only a certain number of the violations will be penalised. Together with the level of the imposed fine, the number of violations penalised is one of the simulation parameters.

Table I provides an overview of all simulation parameters. The primary objective of this set-up is to analyse how the level of the fine for contract violations and the number of violations penalised (independent variables) affect the cooperation behaviour and overall number of cheats (dependent variable). We performed factorial experiments[7] alternating the two (independent) variables whose impact we want to analyse. The simulation experiments consist of running 50 experiments for each parameter combination given in Table I. This means 50 simulation runs for 56 parameter combinations, making 2,800 runs in all.

[7] That is, to run simulations with all possible combinations of the values of the subsets of chosen input variables across all factors. Factorial experiments enable study of the effect of each factor on the simulation output data, as well as the effects of interactions between factors, while cancelling out influences of other factors on a particular setting.

Table I. Simulation variables

Name	Range/Type	Simulation Parameter
$\mid \mathcal{A} \mid$	$[2, \infty]$	100
# Rounds	$[0, \infty]$	500
# Max. Sanctioned Violation per Round	$[0, \infty]$	1, 2, 3, 5, 8, 10, 20
Fine-Level	$[0, \infty]$	10, 25, 50, 75, 100, 125, 150, 200

5.2. THE INSTITUTIONAL MODEL

For the specification of the institutional framework we use Inst*AL*. Once written and verified, it is translated to AnsProlog to be used in the institutional monitor.

For the conceptual design of an institutional framework, we had two options: have one institutional framework that governs all the contracts within the system or have one for each contract. The latter requires the institutional monitor to be aware which participant is involved in which contract and institutional frameworks to share information. We have opted for the former as it is easier to detect when a participant engages in a second contract as a buyer or a seller, but our model could easily be adapted to the latter option using the multi-institution framework (see Cliffe (2007)) that extends the framework used here.

The institutional framework is structured in three phases:

1. Contract creation: buyers and sellers engage in contract formation during a round of the simulation.
2. Violation handling: once all contracts of a given round are agreed, the institutional framework knows which contracts have been violated.
3. Contract fulfilment: contracts that were successfully completed can be deleted.

A schematic representation of the working of the institutional framework is given in Figure 3. A contract is represented by an inertial fluent `contract (Seller,Buyer,Round)` initiated in the first phase. The `Round` is used to indicate to the institutional monitor when the contract was established, so it can be terminated when the contract is satisfied. The contract creation phase consists of a series of exogenous events of the type `setUpContract(Seller, Buyer,Round)`. At the start of the simulation, every agent has to have the permission to act as a buyer and a seller. Regardless of whether or not both `Buyer` and `Seller` have permission to engage in a new contract, a contract fluent will be generated. But, when permission is absent, the correct violation event will be triggered and the contract is scheduled for deletion, which will take place at the end of the simulation round. It is then up to the enforcement mechanism to deal with the violation. Agreeing to enter a contract takes away

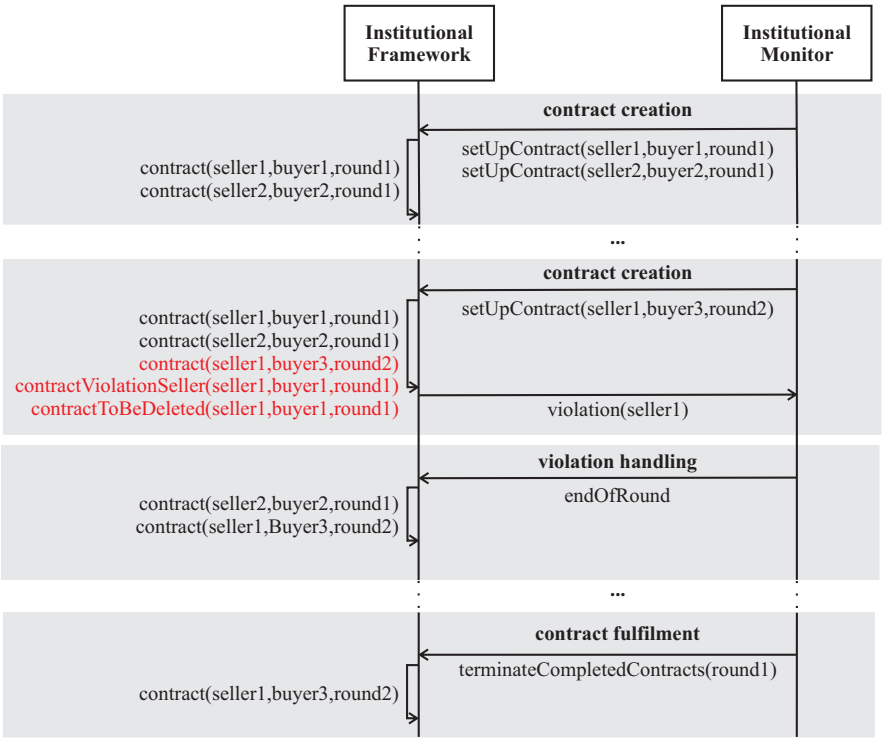

Figure 3. Schematic representation of the institutional framework over a period of time

the permission to enter further contracts until the current contract has been satisfied.

Figure 4 contains a fragment of the Inst*AL* specification of this phase, while the full specification is provided in appendix A. At the end of each round, when all potential buyers and sellers have decided whether they want to enter a contract or not, the institutional monitor signals the institutional framework that phase 2 can commence through the exogenous event `endOfRound`. The occurrence of this event terminates all the contracts that were scheduled for deletion due to a contract violation. Furthermore, it restores permission for participating non-violating agents to engage in new contracts. The main reason to make this a separate phase, rather than immediately deleting the contract and restoring permissions, is to allow us to track violations of both the buyer and seller of an existing contract rather than only penalising the first offender. Hence, we do not allow for mutually-agreed annulment of contracts.

The start of Phase 3 is signalled by the occurrence of the exogenous event `terminateCompletedContracts(Round)`. It indicates that all existing contracts created in the round have been successfully completed. Correspond-

```
setUpContract(Seller,Buyer,Round)
generates
   intSetUpContract(Seller,Buyer,Round);

viol(intSetUpContract(Seller,Buyer,Round))
generates
   contractViolationBuyer(Seller1,Buyer,Round1)
   if contract(Seller1,Buyer,Round1);

contractViolationBuyer(Seller,Buyer,Round)
initiates
   contractToBeDeleted(Seller,Buyer,Round);

intSetUpContract(Seller,Buyer,Round)
terminates
   perm(intSetUpContract(Seller1,Buyer,Round1)),
   perm(intSetUpContract(Seller,Buyer2,Round2));
```

Figure 4. Inst*AL* specification of the initial contract phase

ing contracts are terminated and their participants' permissions to engage in a new contract are restored.

5.3. The Institutional Monitor

Recalling the explanation of the institutional monitor in Section 4, its primary functions are (i) handling the instantiation of the institutional framework, (ii) storage of institutional states and associated grounding information, and (iii) decoupling the institutional framework from the agents. As a result of the design choices that we have explained earlier, we use only one instantiation of one institutional framework in our example. Thus, the institutional monitor does not need to track which contract is assigned to which agent, but instead tracks all agreements by all agents within the same instantiation of the institutional framework. The institutional monitor thus focuses on the latter two tasks. In the beginning of the simulation, the monitor determines the start state of the institutional framework by solving using the ASP solver CLINGO (Gebser et al., 2008). In the next step, whenever it perceives information from the agents that is relevant for the institutional framework (e.g. `buyer(agent1),seller(agent2)`) it determines the current round of the simulation and translates this information into the appropriate external event for the institutional framework `setUpContract(agent2,agent1,round1)` appends this event to the current state of the institutional framework and runs everything for one time-step with CLINGO in order to determine the new state of the institutional framework. Once a round is finished and all agents have

determined whether to take up new contracts or not, it is the institutional monitor which sends the `terminateCompletedContracts(round)` and `endOfRound` events to the institutional framework in order to complete the current round and receive a list of all agents that have violated the rules. From this list of violators, it randomly picks agents, up to the maximum number of punishable events, that are then fined[8]. In a classical setting, the fine would be executed by enforcement agents (i.e. a kind of police force in the system), who are told which agents to punish by the institutional monitor. The number of punishable events would then correspond to the capacity of this police force. For simplicity reasons we have not implemented these enforcement agents directly, but instead we use the institutional monitor to exact the punishment.

5.4. Simulation Results

Figures 5 and 6 show the results of the simplified contract law example simulation with the parameters given in Table I. For legibility and due to space limitations, we only present a subset of results graphically and will comment on the remaining results in text. Thus, we only show the results for the first 150 rounds for example, as between the 150th and the 500th round no significant changes in the results appear. In the figure, for each of the fine levels, we plot the number of cheats per round for different maximum numbers of punishable events (i.e. 1, 3, 5 and 8 events).

The first thing to notice is that due to the set-up, the behaviour in all of the first few rounds is the same: there are no violations in the first round (as agents do not have an alternative offer at this point) and cheats typically briefly peak after that, as agents have not been fined before and therefore are more likely to risk cheating. After these first rounds, the fines and the number of maximum sanctioning events take their effect. Looking at the results more closely and starting with the lowest fine-level of 10 (i.e. a fine level which is equal to or lower then the gains an agent has from cheating), the clear result is that no matter the maximum number of punishable events, no successful stimulus for decreasing contract violation is in place. This changes once the fine increases. At a level of fine of 25, for example, for a maximum of 8 punishable events, a decrease in the number of cheats is clearly visible. This decrease increases proportionally with the fine. Thus, at a fine level of 150 or 200 respectively, for a maximum number of 3, 5 and 8 punishable events hardly any cheats can be detected in the long run[9]. These comparatively good cheat-reduction results of 3, 5 and 8 has significant practical value: if one considers that pun-

[8] If fewer agents have violated than punishable events, all violators are fined.

[9] The higher numbers of punishable events in our experiments (i.e. 10 and 20), that are not depicted in Figures 5 and 6, yield the same results that followed the same trends as described here, the decrease in cheating events produced by them was however faster.)

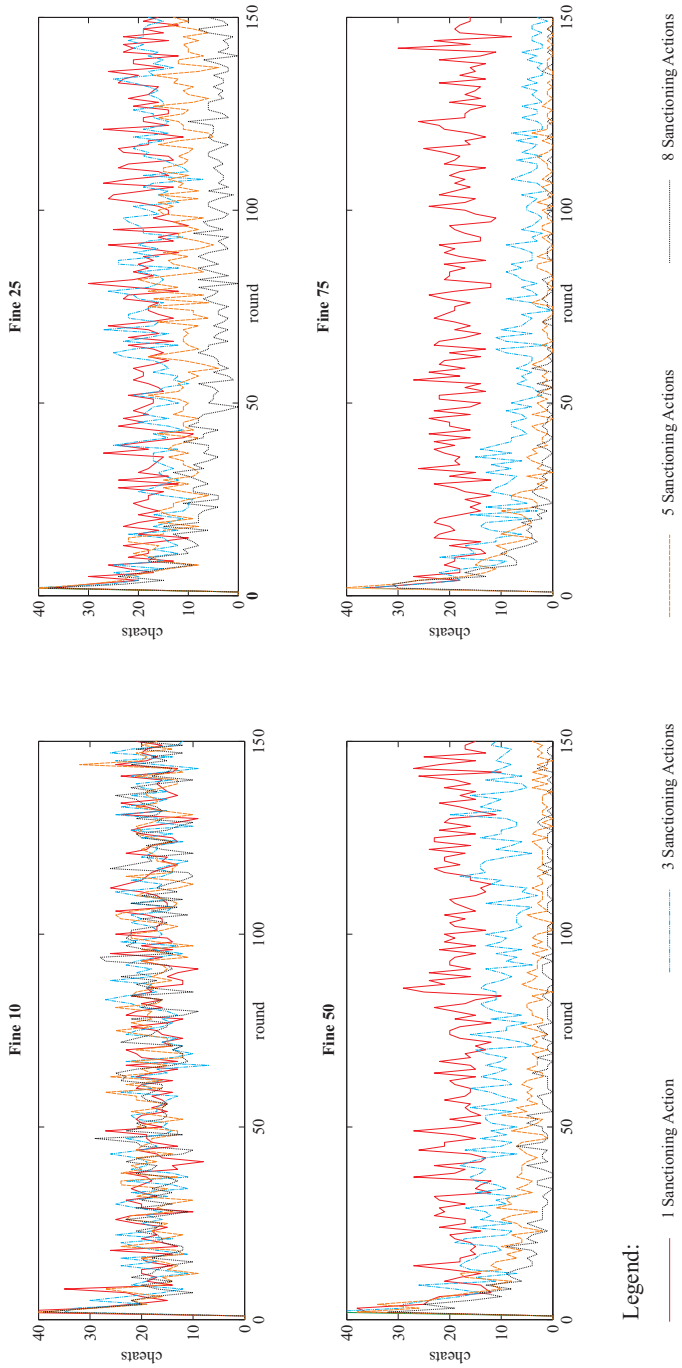

Figure 5. Simulation results (selection) – part 1

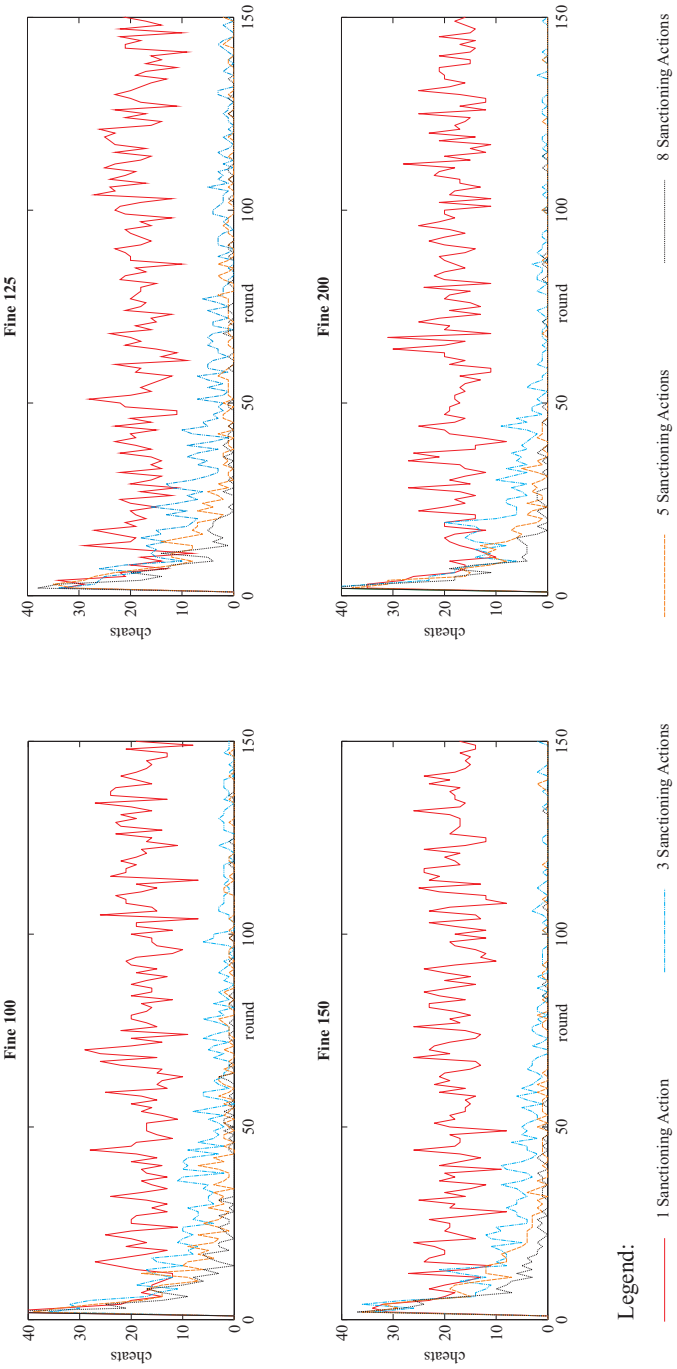

Figure 6. Simulation results (selection) – part 2

ishment does not come for free, but at a cost (e.g. the fixed costs for the staff employed to handle the legal procedures), achieving similarly good results of enforcement at lower costs is highly attractive. Our analysis also shows that it is risky to decrease the maximum number of punishable events too far. In the case that only one violation was punished, many violations went unpunished. As a result, the increase in fines does not help to reduce cheating significantly, as a large number of violators do not have to pay the fine and learn that cheating largely goes unpunished[10]. Other interesting observations (which were not in the initial simulation setup) are that if the total number of agents is decreased (as is the number of punishment agents, likewise), a minor reduction in cheating events can be detected. This might be due to the lower number of transaction partners (and thus alternative offers) agents are able to choose from in this setting. Furthermore, although not part of the initial research question of the simulation, an interesting side-effect is that the model allows us to think about policy awareness. The policies in our model, are stored within the institutional framework and not within the agents. As a result the agents have to query the institutional framework (via the institutional monitor) to learn about the policies affecting their contracts. If we set up the agents not to query the institutional monitor about the policies applying to their current state, etc.), more cheating takes place due to agents possibly not being aware that they could cause a violation[11]. By using an institutional ABM, we are consequently able to determine optimal combinations of fine-levels and a maximum number of punishable events and analyse their combined effects on the described contract-law case study.

6. Related Work

The related work that we discuss covers institutional frameworks as well as the idea of linking them to agents and ABM. This section serves to point out this related work and explain where the work presented in this paper differs from previous work.

Starting with the formal work on the representation of institutional frameworks, it has to be noted that its representation has been a subject of research for several decades. Ever since the British Nationality Act was formalized using Logic Programming by Sergot et al. (1986), it has been recognised that non-monotonic formalisms are important to deal with many aspects of rule-based systems in general and institutional frameworks, as well policies and their analysis in particular (Alberti et al., 2011). Policies and norms,

[10] For maximum 2 punishments effects the results are similar the single punishment effect.

[11] An interesting line of research would be to analyse the learning or increase of awareness of agents with respect to policies if they are punished repeatedly, but this is not the subject of this paper.

specifically, have received much attention using different formalisms, among which defeasible logic (Governatori, 2005; Governatori and Rotolo, 2004) is considered particularly appropriate, along with various executable representations, such as RuleML in (Governatori, 2005). Herrestad (1991) argues that Sergot's formalisation of the library regulations is confronted with the Christholm paradox and that therefore logic programming formalisms are ill-suited for modelling this kind of knowledge. Herrestad's claim is based on the assumption that implication can be rewritten as a disjunction. However, we use answer set programming for the computational model of our institutional framework and here, Herrestad's claims are not valid, as answer set programming uses negation as failure whose semantics differs from classical negation (see Denecker (2004) for more information). Furthermore, with our approach there is no need to express that no disciplinary action is taken when books are returned on time. This is supported implicitly.

The idea of incorporating agents in institutional framework is not a new one. Several frameworks exist that put forward such ideas, the most prominent ones being Opera/Operetta, $\mathcal{M}$OISE$^+$ and Islander (an introduction to these different frameworks can be found in Sierra et al. (2007).

In contrast to these approaches, we do not only include agents in the framework, but link a complete ABM to it and allow for a full autonomy of the agents, only checking the impact their decisions have on the institutional framework as a whole. Other differences are listed below.

The focus in Opera (Okouya and Dignum, 2008) is the specification of (valid) institutional states: how the states were achieved is not addressed, nor is the matter of which agents are responsible for the actions that bring about these states. The model on which our work is based (Cliffe et al., 2007) is complementary to Opera, focusing instead on actions and whether agents are empowered, permitted or obliged to execute actions, as well as maintaining a complete trace of the institutional history. Furthermore, the computational model described here enables a direct, model-checking approach, both for the verification of the design-time model and in its utilisation by agents enquiring about the institutional state as it evolves.

$\mathcal{M}$OISE$^+$ (Hübner et al., 2007) is potentially the most similar to what we have set out: it is described as a middleware for organizational programming, which like us uses an ABM environment as the agent framework and extends this environment to allow agents to perceive their organization. However, it would appear from Hübner et al. (2007), that agent autonomy is quite highly constrained by the roles they take on at any one time, which on their part are restricted by the domain being modelled. Furthermore, it is notable that individual actions can be restricted by the group to which an agent belongs. In contrast, the agents in our model are simply subject to the policies of the institutional framework and can choose whether to observe them or not.

Islander (Esteva et al., 2002) – and associated tools – offers a comprehensive environment for the specification of electronic institutions through state diagrams labelled with speech acts, coupled with simulation via Ameli (Esteva et al., 2004), supported by JADE. The agents are constructed from the specification and are in essence regimented by speech acts, rather than being autonomous, perceiving an illocution and then taking some action that may or may not have meaning in the (single) pervading institutional context. Although it would appear to be possible to author agents that can operate in this environment, using an internal architecture of choice, they would be without access to any formal model of the institutional framework in which they act, and so are functionally blind.

The attraction of connecting BDI agents and norms and policies goes back to at least Dignum et al. (2000), which discusses a form of deontic logic to accommodate institutional concepts and a pseudo-code extension of the BDI agent loop. However, practical connections appear to be few, and even these do not appear to offer actual implementations. Meneguzzi and Luck (2009) discuss how BDI agents may assimilate (changes in) permissions, prohibitions and obligations, but unlike here, do not appear to address the utilization of policies in the agent reasoning process. Criado, Argente and Botti (2010) also describe a form of institution assimilation process via bridge rules, so agents may recognize and acquire norms (and thereby also policies) from their environment, but the recognition process is unclear, nor is it apparent whether the norms are acquired from observation or supplied by an institutional framework.

Summarizing, to address the shortcomings of the related work and to fulfil our goal to offer an approach that allows for reasoning about policies, policy making and their effects on a more comprehensive level than has been possible to date, we presented our approach which combines an institutional framework and an agent-based simulation, uniting the best of both worlds, namely verification of a formal specification combined with the testing of large-scale systems with numerous different actor configurations.

7. Summary and Conclusions

We have demonstrated how institutional frameworks can be integrated into ABM in order to reason about the impact of policies on a group of participants.The advantage of our prototype, J-Inst*AL*, is that it allows both for the formal verification of the institutional model, and the ability to run large scale simulation tests with different set-ups of the individuals in the system without facing complexity issues. In this paper, we focussed more on the integration of the two approaches rather than the benefits arising from formal verification. For a fuller explanation of this aspect, we refer to Balke et al. (2011). We de-

scribed in detail the architecture and the individual components of I-ABMand our prototype J-Inst*AL* and used a simplified contract-law example to show the functionality of our methodology. The results of our simulation experiments give an indication about an appropriate level of fine and a maximum number of punishable events required for the scenario described.

From the perspective of simulation set-up, a possible extension of our work is the analysis of a progressive fine (i.e. fining an agent more if it has broken contracts repeatedly), the inclusion of more sophisticated agents, as well as the obvious matter of the sophistication of our simplified contract example.

In our simulation, one institutional monitor object manages one institutional framework. We observe that often more than one institutional framework is active within an application. Furthermore, some of these may interact with one another. In (Cliffe et al., 2007), the authors present the concept of *multiple* institutional frameworks where events in one institutional framework cause events in another or change another institutional state. Extension of the institutional monitor to accommodate reasoning about multiple institutional frameworks is an important part of future work, along with the issue of using conventional distributed systems techniques, such as replication, as a means to avoid the institutional monitor becoming a bottleneck or single point of failure.

Acknowledgements

The research of Tina Balke is partially supported by funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 288147.

References

Ågotnes, T., W. van der Hoek, J. A. Rodríguez-Aguilar, C. Sierra, and M. Wooldridge: 2007, 'On the Logic of Normative Systems'. In: *JCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*.

Alberti, M., M. Gavanelli, and E. Lamma: 2012, '$Deon^{+}$: Abduction and Constraints for Normative Reasoning'. In: A. Artikis, R. Craven, N. Kesim Çiçekli, B. Sadighi, and K. Stathis (eds.): *Logic Programs, Norms and Action*, Vol. 7360 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 308–328.

Alberti, M., A. S. Gomes, R. Gonçalves, J. Leite, and M. Slota: 2011, 'Normative systems represented as hybrid knowledge bases'. In: J. Leite, P. Torroni, T. Ågotnes, G. Boella, and L. van der Torre (eds.): *Proceedings of the 12th international conference on Computational logic in multi-agent systems*, Vol. 6814 of *Lecture Notes in Computer Science*. pp. 330–346.

Andrighetto, G., R. Conte, P. Turrini, and M. Paolucci: 2007, 'Emergence In the Loop: Simulating the two way dynamics of norm innovation'. In: G. Boella, L. van der Torre, and H. Verhagen (eds.): *Normative Multi-agent Systems*.

Balke, T.: 2011, 'Towards the Governance of Open Distributed Systems – A Case Study in Wireless Mobile Grids'. Phd dissertation, University of Bayreuth.

Balke, T., M. D. Vos, and J. Padget: 2011, 'Analysing energy-incentivized cooperation in next generation mobile networks using normative frameworks and an agent-based simulation'. *Future Generation Computer Systems* **27**(8), 1092–1102.

Baral, C.: 2003, *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge Press.

Bonabeau, E.: 2002, 'Agent-based modeling: Methods and techniques for simulating human systems'. *Proceedings of the National Academy of Sciences of the United States of America* **99**(10), 7280–7287.

Bordini, R. H., J. F. Hübner, and M. Wooldridge: 2007, *Programming Multi-Agent Systems in AgentSpeak using Jason*, Wiley Series in Agent Technology. John Wiley & Sons.

Broersen, J., M. Dastani, J. Hulstijn, Z. Huang, and L. van der Torre: 2001, 'The BOID architecture: conflicts between beliefs, obligations, intentions and desires'. In: *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA, pp. 9–16.

Clarke, E. M., E. A. Emerson, and A. P. Sistla: 1986, 'Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications'. *ACM Transactions on Programming Languages and Systems* **8**(2), 244–263.

Cliffe, O.: 2007, 'Specifying and Analysing Institutions in Multi-Agent Systems using Answer Set Programming'. Ph.D. thesis, University of Bath.

Cliffe, O., M. De Vos, and J. Padget: 2007, 'Specifying and Reasoning about Multiple Institutions'. In: *Coin*, Vol. 4386 of *LNAI*. pp. 67–85.

Criado, N., E. Argente, and V. J. Botti: 2010, 'A BDI architecture for normative decision making'. In: W. van der Hoek, G. A. Kaminka, Y. Lespérance, M. Luck, and S. Sen (eds.): *International Conference on Autonomous Agents and Multiagent Systems*. pp. 1383–1384.

De Vos, M., J. Padget, and K. Satoh: 2011, 'Legal Modelling and Reasoning using Institutions'. In: S. Tojo (ed.): *Proceedings of JURISIN 2010*, Vol. 6797 of *LNCS*.

Denecker, M.: 2004, 'What's in a model? Epistemological Analysis of Logic Programming'. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*. pp. 106–113.

Dignum, F., D. Morley, E. Sonenberg, and L. Cavedon: 2000, 'Towards Socially Sophisticated BDI Agents'. *Multi-Agent Systems, International Conference on* **0**, 0111.

Epstein, J. M. and R. Axtell: 1996, *Growing Artificial Societies: Social Science from the Bottom Up*. Washington, DC, USA: The Brookings Institution.

Esteva, M., D. de la Cruz, and C. Sierra: 2002, 'ISLANDER: an electronic institutions editor'. In: *International Conference on Autonomous Agents and Multiagent Systems*. pp. 1045–1052.

Esteva, M., B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos: 2004, 'AMELI: An agent-based middleware for electronic institutions'. In: N. e. a. Jennings (ed.): *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, Vol. 1. Washington, DC, USA, pp. 236–243.

Gebser, M., R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele: 2008, 'Engineering an Incremental ASP Solver'. In: M. Garcia de la Banda and E. Pontelli (eds.): *Logic Programming*, Vol. 5366 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 190–205.

Gebser, M., B. Kaufmann, A. Neumann, and T. Schaub: 2007, 'Conflict-Driven Answer Set Solving'. In: *Proceeding of IJCAI07*. pp. 386–392.

Gelfond, M. and V. Lifschitz: 1991, 'Classical Negation in Logic Programs and Disjunctive Databases'. *New Generation Computing* **9**(3-4), 365–386.

Gilbert, N. and K. G. Troitzsch: 2005, *Simulation for the Social Scientist*. Open University Press, 2nd edition.

Governatori, G.: 2005, 'Representing Business Contracts in RuleML'. *International Journal of Cooperative Information Systems* **14**(2-3), 181–216.

Governatori, G. and A. Rotolo: 2004, 'Defeasible Logic: Agency, Intention and Obligation'. In: A. Lomuscio and D. Nute (eds.): *Deontic Logic in Computer Science*, Vol. 3065 of *LNAI*. Berlin, pp. 114–128.

Grossi, D.: 2007, 'Designing invisible handcuffs. Formal investigations in institutions and organizations for multi-agent systems'. Ph.D. thesis, Utrecht University.

Gulyás, L.: 2005, 'Understanding Emergent Social Phenomena: Methods, Tools and Applications for Agent-Based Modeling'. Ph.D. thesis, Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary.

Helbing, D. and S. Balietti: 2011, 'How to Do Agent-Based Simulations in the Future: From Modeling Social Mechanisms to Emergent Phenomena and Interactive Systems Design'. Working Paper 11-06-024, Santa Fe Institute.

Herrestad, H.: 1991, 'Norms and formalization'. In: *ICAIL'91: Proceedings of the 3rd international conference on Artificial intelligence and law*. pp. 175–184.

Holland, J. H.: 1992, *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press.

Hübner, J. F., J. S. Sichman, and O. Boissier: 2007, 'Developing organised multiagent systems using the MOISE'. *IJAOSE* **1**(3/4), 370–395.

Jager, W.: 2000, 'Modelling Consumer Behaviour'. Ph.D. thesis, University of Groningen.

Jones, A. J. and M. Sergot: 1996, 'A Formal Characterisation of Institutionalised Power'. *ACM Computing Surveys* **28**(4), 121. Read 28/11/2004.

Jones, A. J. I. and M. Sergot: 1993, 'On the Characterization of Law and Computer Systems: The Normative Systems Perspective'. In: *Deontic logic in Computer Science: Normative System Specification*. John Wiley and Sons Ltd., pp. 275–307.

Ligeza, A.: 2006, *Logical Foundations for Rule-Based Systems*, Vol. 11 of *Studies in Computational Intelligence*. Springer.

Macy, M. W. and R. Willer: 2002, 'From Factors to Actors: Computational Sociology and Agent-Based Modeling'. *Annual Review of Sociology* **28**, 143–166.

Meneguzzi, F. R. and M. Luck: 2009, 'Norm-based behaviour modification in BDI agents'. In: C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman (eds.): *International Conference on Autonomous Agents and Multiagent Systems (1)*. pp. 177–184.

Morales, J., M. López-Sánchez, and M. Esteva: 2011, 'Evaluation of an Automated Mechanism for Generating New Regulations'. In: J. Lozano, J. Gámez, and J. Moreno (eds.): *Advances in Artificial Intelligence*, Vol. 7023 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 12–21.

Noriega, P.: 1997, 'Agent mediated auctions: The Fishmarket Metaphor'. Ph.D. thesis, Universitat Autònoma de Barcelona.

North, D. C.: 1994, 'Institutions Matter'. Economic History 9411004, EconWPA.

Okouya, D. and V. Dignum: 2008, 'OperettA: a prototype tool for the design, analysis and development of multi-agent organizations'. In: *AAMAS (Demos)*. pp. 1677–1678.

Ostrom, E.: 1990, *Governing the Commons: the Evolution of Institutions for Collective Action*. Cambridge University Press. 18th printing (2006).

Prakken, H. and M. J. Sergot: 1996, 'Contrary-to-Duty Obligations'. *Studia Logica (SLOGICA)* **57**(1), 91–115.

Rao, A. S. and M. P. Georgeff: 1995, 'BDI-agents: from theory to practice'. In: *Proceedings of the First International Conference on Multiagent Systems*.

Sergot, M. J., F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory: 1986, 'The British Nationality Act as a logic program'. *Communications of the ACM* **29**(5), 370–386.

Sierra, C., J. Thangarajah, L. Padgham, and M. Winikoff: 2007, 'Designing institutional multi-agent systems'. In: *Proceedings of the 7th international conference on Agent-oriented software engineering VII*. Berlin, Heidelberg.

Vázquez-Salceda, J.: 2003, 'The role of norms and electronic institutions in multi-agent systems applied to complex domains. The HARMONIA framework'. Ph.D. thesis, Technical University of Catalonia.

Vickers, G.: 1973, 'Values, Norms and Policies'. *Policy Science* **4**, 103–111.

Wooldridge, M. J. and N. R. Jennings: 1995, 'Intelligent Agents: Theory and Practice'. *The Knowledge Engineering Review* **10**(2), 115–152.

Appendix

A. Full Case-Study Specification

The complete Inst*AL*specification language description of our case study is given in the following figures. Comments describing the comments are preceeded by %

```
% name of institution

institution buyerseller;

% types
type Agent;
type Round;

% exogeneous events
exogenous event setUpContract(Agent,Agent,Round);
      % seller, buyer, round identifier
exogenous event terminateCompletedContracts(Round);
exogenous event endOfRound;

% institutional events
inst event intSetUpContract(Agent,Agent,Round);

% violation events
violation event contractViolationBuyer(Agent,Agent,Round);
     % Seller, Buyer
violation event contractViolationSeller(Agent,Agent,Round);

% fluents
fluent contract(Agent,Agent,Round);  % Seller, Buyer
fluent contractToBeDeleted(Agent,Agent,Round);

%%%%%%%%%%%%%%%%%%%
```

Figure 7. Declaration of the institutional components