Symbolic Model Checking of Institutions

Francesco Viganò
Università della Svizzera italiana
via G. Buffi 13, 6900
Lugano, Switzerland
francesco.vigano@lu.unisi.ch

Marco Colombetti
Politecnico di Milano
piazza Leonardo Da Vinci 32
Milano, Italy
marco.colombetti@polimi.it

ABSTRACT

Norms defined by institutions and enforced by organizations have been put forward as a mechanism to increase the efficiency and reliability of electronic transactions carried out by agents in open systems. Despite several approaches have been proposed to *model* protocols in terms of institutional concepts (e.g., obligations and powers) and to *monitor* the actual compliance of agents' behavior at runtime, little work has been done to formally guarantee that such systems of norms ensure certain desirable properties. In this paper we describe a framework to verify institutions, which is characterized by a metamodel of institutional reality, languages to describe institutions and to specify their properties, and a tool to model check them. Finally, to evaluate our approach, we model and verify the Dutch Auction institution, a widely used interaction protocol, showing that the verification of institutional rules constitutes a necessary step to define sound institutions.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems; I.2.11 [**Distributed Artificial Intelligence**]: Languages and structures; D.2.4 [**Software/Program Verification**]: Model Checking

General Terms

Verification

Keywords

Institutions, Norms, Model Checking

1. INTRODUCTION

Electronic institutions have been proposed to design, analyze, and regulate open multiagent systems where agents are developed by different organizations and their internal mental states are not accessible [25, 13, 12, 16]. Unfortunately, the term "electronic institution" is often used to refer to either the *rules* that regulate open multiagent systems, the *organization* that enforces them [13], the *software implementation* of institutional rules [12], or a specific *formalism* to describe them [14]. For this reason, in this paper we will adopt the term "institution" to refer to a set of rules and concepts regulating agent interactions and which may be enforced by an organization (an electronic institution according to [13]). This distinction closely reflects the use of the term "institution" as it has been exploited in [30, 31], where an institution is any collectively accepted system of rules which creates institutional facts [30], and in [26], where institutions are the rules of the game in a society which may be enforced by a coercive third party.

According to [25, 13, 7, 15], in open systems institutions essentially play two fundamental roles: (i) they define a set of *norms* which make more predictable the behavior of other agents and (ii) they describe the *ontology* of the interaction context. For instance, the institution of the Dutch Auction not only introduces a norm which obliges an auctioneer to decrease the price of a good under certain conditions, but also defines the very concept of decreasing the price [15]. In [30, 31] Searle claims that there exists a strict relation among normative and ontological aspects of institutions: "institutional facts are matters of deontic" relations [31] (institutionalized powers [19], obligations, prohibitions, etc.). Indeed, when a seller quotes a good it fixes its price (an institutional fact) and at the same time it creates new powers and obligations not only for itself, but also for other agents. For example, the seller cannot ask clients to pay a higher price, while it is empowered (but not obliged) to apply a discount.

To investigate the interdependencies existing among deontic relations and the ontology defined by an institution, in [36] we proposed FIEVeL (*F*unctions for *I*nstitutionalized *E*nvironments *V*erification *L*anguage), a language to model institutions in terms of the notion of *status function*, that is "a *status* to which a *function* is assigned" [30, pag. 40]. Our main tenet is that all status functions, even those imposed on events and objects, can be reduced to statuses imposed on agents, which are named *agent status functions*. In particular, we describe agent status functions in terms of deontic relations, which represent what actions are empowered, obliged, forbidden, or permitted for an agent. In doing so, institutional events can be characterized in terms of what status functions are imposed or revoked, which helps to clarify how each institutional event changes agent deontic relations.

In contrast with [24], where norms (also named *social*

ICEC'07, August 19–22, 2007, Minneapolis, Minnesota, USA.

laws) are assumed to be respected by agents because they are designed and encoded by a single organization, in open systems it is unrealistic to expect that autonomous agents will always comply with norms. For this reason, the research on institutions has been mainly focused on developing languages and tools to *model* [13, 7, 28, 15, 16] and to *monitor* [12, 35, 8, 16] agent interactions in terms of institutional concepts (roles, obligations, etc.) with the purpose of avoiding or detecting violations of norms. In doing so, institutions play a crucial role to increase the efficiency of electronic transactions carried out by agents [25], but raise the problem of ensuring that such rules are not characterized by contradictory norms and provide agents with all the needed powers to fulfill their objectives. This is especially important when institutions are complex and it is prohibitive to foresee all possible evolutions admitted by them.

Automated formal verification [5, 27] should be considered as an important step for the development of institutions, because it can increase the reliability of institutions by ensuring that they satisfy certain properties. The development of formal frameworks to verify institutions is therefore essential, but only few attempts have been proposed in the literature to introduce automated formal methods to verify them [6, 18, 36]. In [6, 18] the authors propose two frameworks to model check institutions described according to the language discussed in [13], but only certain aspects of institutions are verified. For instance, both approaches do not consider *normative rules* [13]. In [36] we proposed to model check FIEVeL institutions by translating its constructs into Promela, the input language of the SPIN model checker [17]. The main disadvantages of the approach described in [36] reside in the time and memory required to verify institutions, and in the lack of a high-level specification language to define desirable properties in terms of the very same concepts used to model institutions.

To provide designers with a succinct notation to specify properties of institutions and to decrease the time required to verify them, in this paper we propose a new approach to model check FIEVeL institutions. In particular, we define the semantics of FIEVeL constructs in terms of what sorts, symbols, and axioms of an ordered many-sorted first-order temporal logic (OMSFOTL) are induced by them. Moreover, we introduce a specification language whose expressions are equivalent to formulae written in a OMSFOTL and which allows designers to refer to any symbol defined by a FIEVeL institution. This is particularly important because it increases the reuse of both models and properties of institutions, since typically institutions describe rules that do not depend on the number of agents, objects, etc. involved in the interaction. Finally, assuming that domains of sorts referred by institutions are finite, it is possible to translate them into propositional models and their properties into CTL formulae [5], which allows us to apply OBDD-based model-checking techniques and efficient algorithms to verify whether an institution satisfies a given property.

The remainder of this paper is structured as follows. Section 2 presents a set of concepts that we perceive as essential to describe institutions and which constitute the metamodel of the modeling language discussed in Section 3. To exemplify its syntax and the semantics of its main constructs, we report our model of the Dutch Auction inspired by the formalization proposed in [16]. Section 4 introduces a concise notation suitable for specifying properties of institutions and a few properties of the Dutch Auction, while Section 5 presents a tool to simulate and verify institutions. Finally in Section 6 we provide a comparison of our approach with related works and in Section 7 we draw some directions for future works.

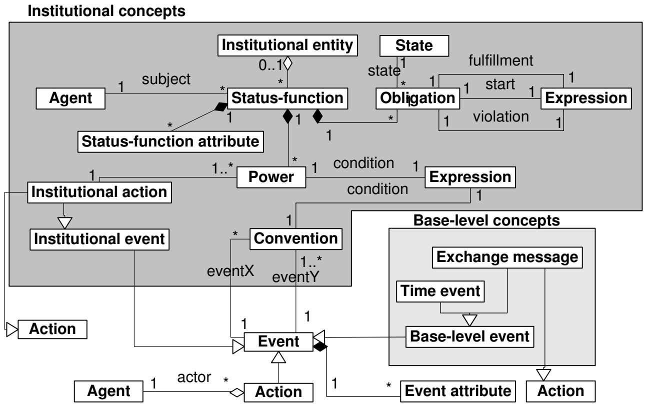


Figure 1: The institutional metamodel.

2. THE INSTITUTIONAL METAMODEL

We express the semantics of status functions and related institutional concepts in terms of a many-sorted first-order logic [21] enriched with temporal operators and hierarchies of sorts. Despite formulae of an ordered many-sorted first-order temporal logic (OMSFOTL) can be translated into first-order temporal logic [1] or, assuming finite domains, into a temporal propositional logic like CTL* [10], we adopt OMSFOTL for two main reasons: (i), it represents an abbreviated form for long and complex formulae and (ii), institutions describe rules that typically are independent of the cardinality of domains used to describe them, which can be naturally expressed by allowing quantification over sorts. The signature of OMSFOTL is similar to the one of a first-order logic with the addition of a finite nonempty set of *sort symbols* $\mathbf{\Sigma}$, a *hierarchy of sorts* $\leq_\Sigma$ (where $\sigma_1 \leq_\Sigma \sigma_2$ means that sort σ_1 is a *subsort* of sort σ_2), and function ξ, which guarantees syntactic type checking of formulae by assigning a sort to every variable and every constant, and a signature (i.e. a sequence of sorts) to every function symbol and every predicate symbol. In the remainder of this section we explain what sorts, functions, and predicates are introduced to represent institutional concepts, providing also their signature ξ.

Figure 1 depicts some of the main sorts used in our approach (e.g., status function, obligation, event) and their relations, which are typically represented by introducing predicates or functions. For instance, relation *subject* is reflected in our logic by function *subject*, which refers to the agent a status function has been imposed to (see below). Finally, we report a set of axioms which characterize institutional reality by imposing restrictions on the admissible valuations of institutional models [2]. As we will see in Section 3, FIEVeL provides a concrete syntax to formalize institutions in terms of the very same concepts represented in Figure 1. For this reason we say that such set of concepts and their relations define a metamodel, since they constitute a model of our modelling language [20]. On the other hand, the institutional metamodel represents the *upper ontology* of institutional reality, since it introduces concepts that are extended to describe the ontology of institutions. For instance, any domain-dependent status function extends the notion of status function, which is abstractly defined as an aggregate of deontic position, by detailing what powers and obligations are associated to it.

Our metamodel of institutions is based on the notion of agent status function, that is, a status imposed on an agent and recognized as existing by a set of agents. A typical example of status function is the concept of "owner", since an agent owns an object not thanks to its own physical features, but only because a community of agents recognize so. Other examples of status functions are the notion of "president" or "employee", which have been usually regarded as *roles* [3]. Indeed the concept of status function shares several features with the concept of role (refer to [22, 3] for an overview), but we perceive it to be broader, since its definition does not presuppose a structured preexisting organization [3]. Moreover, the term status function better represents the fact that we are concerned with statuses whose existence depends on those agents that recognize them as existing and which are assigned to agents to create new institutionalized powers or to regulate their use.

The notion of agent status function induces sort σ_{sf} and the function *subject* ($\xi(subject) = \langle \sigma_{aid}, \sigma_{sf} \rangle$), which denotes the agent (σ_{aid}) the status function has been assigned to. A status function may be currently *assigned* or revoked ($\xi(assigned) = \langle \sigma_{sf} \rangle$), reflecting the fact that an agent acquires or looses certain deontic relations. Status functions of an institution are *modified* ($\xi(modified) = \langle \sigma_{sf} \rangle$) when certain institutional events happen (see below), otherwise they continue to be assigned (unassigned) to the same agent:

$$\mathbf{AG}\forall f(\neg \mathbf{X} modified(f) \rightarrow (assigned(f) \leftrightarrow \mathbf{X} assigned(f))) \tag{A.1}$$

$$\mathbf{AG}\forall f(\neg \mathbf{X} modified(f) \rightarrow \exists a(subject(f) = a \wedge \mathbf{X} subject(f) = a)) \tag{A.2}$$

Status functions are possibly empty aggregates of deontic relations that can be expressed in terms of two main concepts, *institutionalized power* [19] and *obligation*. As we will see, we represent powers as predicates related to the performance of institutional actions, while obligations induce a sort (σ_o) whose individuals reify norms of institutions. Obligations can be also used to express prohibitions by specifying suitable violation expressions, while we do not define a specific construct to explicitly represent the fact that an agent is permitted to perform an action as in [7, 16, 28, 35]. Instead, we consider that an agent is permitted to execute an action if the action can be performed without violating any norm.

Sort σ_o is characterized by the function *state* ($\xi(state) = \langle \sigma_{state}, \sigma_o \rangle$) and by a set of predicates (*start*, *fulfillment*, and *violation* of signature $\xi(violation) = \langle \sigma_o, \sigma_{sf} \rangle$) which are used to specify conditional norms and under what conditions an agent fulfills or violates them. When a status function is imposed, the state of a norm is set to *unfired* if predicate *start* is not satisfied:

$$\mathbf{AG}\forall o \forall f((ofstatus(o) = f \wedge \mathbf{X}(assigned(f) \wedge modified(f)) \wedge \neg start(o,f)) \rightarrow \mathbf{X} state(o) = unfired) \tag{A.3}$$

otherwise, it is set to *activated*:

$$\mathbf{AG}\forall o \forall f((ofstatus(o) = f \wedge \mathbf{X}(assigned(f) \wedge modified(f)) \wedge start(o,f)) \rightarrow \mathbf{X} state(o) = activated) \tag{A.4}$$

When a norm is activated, it may reach state *inactive* either because it is fulfilled, it is violated, or because it is associated to a revoked status function. For instance, the following axiom states that if an obligation has been *activated* and becomes *inactive*, then it means that either *violation* or *fulfillment* are evaluated to true, or the status associated to the obligation (referred by function *ofstatus*) has been revoked:

$$\mathbf{AG}\forall o \forall f((ofstatus(o) = f \wedge \mathbf{X} state(o) = inactive \wedge state(o) = active) \rightarrow (\mathbf{X} modified(f) \vee violation(o,f) \vee fulfillment(o,f))) \tag{A.5}$$

To automatically classify states and transitions with respect to norms of an institution, we introduce predicate *violated* of signature $\xi(violated) = \langle \sigma_o \rangle$. A norm is violated if and only if it was *activated*, the associated status function is not modified, *violation* holds while *fulfillment* is false:

$$\mathbf{AG}\forall o \forall f(ofstatus(o) = f \rightarrow (\mathbf{X} violated(o) \leftrightarrow (state(o) = active \wedge (violation(o,f) \wedge \neg fulfillment(o,f) \wedge \neg \mathbf{X} modified(f))))) \tag{A.6}$$

An institution and its physical environment evolve because events (σ_{ev}) occur or agents perform actions ($\sigma_{act} \leq_\Sigma \sigma_{ev}$). Each event type e induces a sort σ_e, a constant e of sort σ_e, and three predicates, $happens_e$, $prec_e$, and eff_e, which express if an event of type e happens, and what conditions must be satisfied before and after its occurrence. We classify events into two different sorts: *base-level events* (σ_{be}), like *time* events and *exchange-message* events, and *institutional events* (σ_{ie}), like the 18th birthday and the act of transferring the ownership. While base-level events affect only the environment of an institution, *institutional events* modify institutional reality by imposing or revoking status functions. Therefore, institutional events occur only because a community of agents recognizes their effects and cannot be directly produced by the environment or by an agent [30]. On the contrary, the occurrence of a base-level event is influenced only by its preconditions (e.g., a door can be open only if it was closed):

$$\mathbf{AG}\forall \overline{x}(\mathbf{X} happens_e(\overline{x}) \rightarrow prec_e(\overline{x})) \tag{A.7}$$

where $\overline{x}$ represents a set of variables x_i determined by the signature of predicate $happens_e$.

Following [30] the occurrence of an institutional event is subordinated to the occurrence of another event conventionally associated to it. More precisely, an institutional event ie that is not an action occurs if and only if an event conventionally related to it happens:

$$\mathbf{AG}\forall \overline{x}((prec_{ie}(\overline{x}) \wedge \bigvee_{e \in \sigma_{ev}} \mathbf{X}(conv_{e-ie}(\overline{x}) \wedge happens_e(\overline{x}'))) \leftrightarrow \mathbf{X} happens_{ie}(\overline{x})) \tag{A.8}$$

where predicate $conv_{e-ie}$ represents the existence of a convention among event e and institutional event ie and $\overline{x}'$ reflects how arguments of event ie are mapped over arguments of event e. Instead, in the case of institutional actions a further condition must be satisfied, namely, the actor must be empowered to perform the institutional action ia:

$$\mathbf{AG}\forall\overline{x}((prec_{ia}(\overline{x}) \wedge \exists f(subject(f) = x_1 \wedge empowered_{ia}(f, \overline{x}) \wedge assigned(f) \wedge \bigvee_{a \in \sigma_{act}} \mathbf{X}(conv_{a-ia}(\overline{x}) \wedge happens_a(\overline{x}'))) \leftrightarrow \mathbf{X}happens_{ia}(\overline{x})) \tag{A.9}$$

where the first variable of $\overline{x}$ refers to the actor of action ia and predicate $empowered_{ia}$ reflects the fact that status functions may be conditionally empowered to perform institutional action ia. Finally, all types of events are characterized by the following axiom:

$$\mathbf{AG}\forall\overline{x}(\mathbf{X}happens_e(\overline{x}) \to \mathbf{X}eff_e(\overline{x})) \tag{A.10}$$

which states that if an event occurs, its effects take place. Axioms (A.6), (A.9), and (A.10) highlight the main difference between the absence of permission, due to the existence of a prohibition, and the absence of institutionalized power: if an agent is prohibited to perform institutional action ia but performs it anyway, the effects of the action take place and the obligation to not perform the action is violated; on the contrary, if an institutional action ia is not empowered, it cannot happen and its effects will not take place. As observed in [37], within a single institution it is always possible to regulate the performance of an institutional action either by revoking powers or by creating prohibitions. Instead, a base-level action can only be regulated by defining prohibitions to not execute it, unless we can modify the environment in such a way that it becomes impossible to perform it.

In the following section we will exemplify the syntax and semantics of FIEVeL, a modelling language for institutions which has been introduced in [36] and which allows designers to describe institutions in terms of the institutional concepts described by our metamodel. In particular, in this paper we present an improved version of the language, characterized by a simpler syntax and new constructs to describe norms.

3. FIEVEL, A MODELLING LANGUAGE FOR INSTITUTIONS

Figure 2 reports a few fragments of the Dutch Auction institution inspired by the formalization discussed in [16], where an *auctioneer* offers to sell a good at a certain price. In contrast with the English Auction, in the Dutch Auction the price is fixed by the auctioneer and not by participants, which may accept it or wait for a new quotation. For the sake of model checking we discretize the range of possible prices that can be declared by an auctioneer, assuming that the current price may be *toohigh* (no agent can afford it), *initial* (the first offer of the auctioneer), *medium*, and *reservation* (the lowest price offered by an auctioneer).

Once an auctioneer has quoted the good, the current price is considered valid for a certain time interval, after which the round is considered closed and, according to [16], three situations may arise:

```
basic-sorts
oid;
priceD={reservation,medium,initial,toohigh};
sOff={accept,noaccept};
...
base-events
message soldGood(agent:aid,good:oid);
message makeWithdrawGood(good:oid);
       ...
institution dutchAuction{
status-function auctioneer (goodA:oid,priceA:priceD,off:sOff){
key goodA;
powers
offer <-  ((good=goodA and off=noaccept) and ((priceA=medium
 and price=reservation) or ((price=medium and priceA=initial)
 or (price=initial and priceA=toohigh))));
sold <-  ((good=goodA and off=noaccept) and
 exists b1:bidder,forall b2:bidder (((assigned(b1) and
 assigned(b2)) -> subject(b1)=subject(b2) )));
withdrawGood <-  ((off=noaccept and good=goodA) and
 (priceA=reservation and not exists b1:bidder(assigned(b1)));
       ...
deontic
 n1
 start<->X((not priceA=reservation and off=noaccept) and
  (not exists b:bidder (assigned(b)));
 fulfillment<->Xhappens(offer,_,_,_)
 violation<->Xhappens(time);
 n2
 start<->(priceA=reservation and happens(time)) and
  X(not exists b:bidder (assigned(b)));
 fulfillment<->Xhappens(withdrawGood,_,_)
 violation<->Xhappens(time);
 n3
 start<->X(happens(time) and exists b1:bidder((
  assigned(b1) and forall b2:bidder (( assigned(b2) ->
  subject(b1)=subject(b2) )));
 fulfillment<->Xhappens(sold,_,_,_)
 violation<->Xhappens(time);
 n4
 start<->X(happens(time) and exists b1:bidder,
  exists b2:bidder (((assigned(b1) and assigned(b2)) and
  not subject(b1)=subject(b2) )));
 fulfillment<->Xhappens(collisionOffer,_,_,_)
 violation<->Xhappens(time);
 }
status-function customer(account :  priceD){...}
status-function bidder(goodB:oid){...}
       ...
institutional-events:
institutional-event endRound()
 pre exists a:auctioneer((off(a)=accept and assigned(a));
 eff a:auctioneer(off(a)=accept)-X->
    o:auctioneer assign(off(o)=noaccept,goodA(o)=goodA(a)
    subject(o)=subject(a),priceA(o)=priceA(a));
institutional-action sold(agent:aid,good:oid)
 pre exists b:bidder((goodB(b)=good and (assigned(b) and
   subject(b)=agent)));
 eff x:auctioneer revoke(goodA(x)=good);
institutional-action withdrawGood(good:oid)
 pre TRUE
 eff x:auctioneer revoke(goodA(x)=good);
institutional-action collisionOffer(good:oid,price:priceD)
 pre TRUE
 eff x:bidder revoke(goodB(x)=good),
     a:auctioneer assign(goodA(x)=good,priceA(a)=price,
      off(a)=accept);
       ...
conventions:
exch-Msg(makeWithdrawGood)[...]=c=>withdrawGood[good=c=>good];
exch-Msg(soldGood)[TRUE]=c=>sold[good=c=>good agent=c=>agent];
time[TRUE] =c=> endRound [ ]
       ...
}
```

Figure 2: Fragments of the Dutch Auction institution coded in FIEVeL.

1. several *customer*s have submitted their bids: a collision is detected and norm $n4$ is fired, obliging the auctioneer to offer the good at a higher price before a certain time period elapses;

2. only a customer has bid: the auctioneer is obliged by norm $n2$ to sell the good, which also determines that the auction is closed;

3. no buyer has submitted a bid: if the current price is equal to the *reservation* price, norm $n3$ obliges the auctioneer to withdraw the good, otherwise the auctioneer offers a lower price to fulfill norm $n1$.

To activate norms we introduce event *endRound* (see Figure 2), which occurs when a time event happens and modifies the *auctioneer* status function: in doing so, according to Axiom (A.4), it activates one of the aforementioned norms. Moreover, if certain conditions are met, the auctioneer is empowered to declare sold the good, to withdraw it, or finally to declare that a collision has been detected, which also increases the current price.

In our formalization of the Dutch Auction we empower customers to perform bids only when their credit is higher than the current price. This means that, according to Axiom (A.9), customers have the physical possibility to send messages offering to buy the good, but such messages do not necessarily count as bids [15]. Therefore, if two agents send a message accepting a certain price but only one of them has the necessary credit, the system registers a single bid and the good is sold to the bidder. Instead, in [16] customers are prohibited from making bids they cannot afford, but bids not supported by a sufficient credit are considered by the auctioneer as any other bid: as a consequence, the auctioneer is obliged to restart the auction by offering a higher price even when there exists only an agent which can afford its bid. For this reason, although our formalism is able to model both scenarios, we prefer to limit the power of customers, in order to increase the efficiency of the interactions ruled by the Dutch Auction institution.

The semantics of the language exemplified in Figure 2 is given by providing a translation of its constructs into a set of symbols and formulae of OMSFOTL. For instance, the first lines of the model reported in Figure 2 induces sort σ_{oid}, σ_{priceD} (which also declares constants *toohigh*, *initial*, *medium*, and *reservation* of sort σ_{priceD}), and σ_{sOff}, which is used to represent whether a round has been terminated. Analogously, status function *auctioneer* is mapped onto sort $\sigma_{auctioneer} \leq_\Sigma \sigma_{sf}$ and its attributes are mapped onto function symbols. For instance, attribute *priceA* corresponds to a function symbol of signature $\xi(priceA) = \langle \sigma_{priceD}, \sigma_{auctioneer} \rangle$. According to Figure 2, only status function *auctioneer* is empowered to perform institutional action *sold* and an auctioneer can successfully declare a good sold when a round has terminated ($off = noaccept$) and only an agent has bid during the last round:

$$\begin{aligned}&\forall x \forall g \forall f\, \mathbf{AG}(empowered_{sold}(f, x, g) \leftrightarrow \exists s(f = s \wedge \\ &\quad (g = goodA(s) \wedge off(s) = noaccept \wedge \exists b1 \forall b2(\\ &\qquad (assigned(b1) \wedge assigned(b2)) \rightarrow \\ &\qquad\quad subject(b1) = subject(b2))))); \quad \text{(A.11)}\end{aligned}$$

where $\xi(g) = \sigma_{oid}$, $\xi(x) = \sigma_{priceD}$, $\xi(f) = \sigma_{sf}$, and $\xi(s) = \sigma_{auctioneer}$. Similarly, norm $n2$ induces sort $\sigma_{n2} \leq_\Sigma \sigma_o$ and states that a norm of sort σ_{n2} may be violated when a time event occurs:

$$\mathbf{AG} \forall o \forall f(violation(o, f) \leftrightarrow \mathbf{X} happens_{time}()) \quad \text{(A.12)}$$

where $\xi(o) = \sigma_{n2}$ and $\xi(f) = \sigma_{auctioneer}$. Analogously, a norm of sort σ_{n2} is considered fulfilled when the auctioneer withdraws the good:

$$\begin{aligned}&\mathbf{AG} \forall o \forall f(fulfillment(o, f) \leftrightarrow \mathbf{X} \exists a \exists g \\ &\quad (happens_{withdrawGood}(a, g))) \quad \text{(A.13)}\end{aligned}$$

where $\xi(g) = \sigma_{oid}$ and $\xi(a) = \sigma_{aid}$. Finally, norm $n2$ is activated when a time event has occurred, the current price is equal to the reservation price, and in the next state no agent has bid for the current price:

$$\begin{aligned}&\mathbf{AG} \forall o \forall f(start(o, f) \leftrightarrow (ofstatus(o) = f \wedge happens_{time}() \wedge \\ &\quad priceA(f) = reservation \wedge \mathbf{X} \neg \exists b(assigned(b)))) \\ &\quad \text{(A.14)}\end{aligned}$$

where $\xi(b) = \sigma_{bidder}$.

The declaration of institutional action *sold* is reflected by the definition of sort $\sigma_{sold} \leq \sigma_{ia}$ and predicates $happens_{sold}$, pre_{sold}, and eff_{sold}, whose signatures are determined by a set of attributes defined by the designer. In particular, we assume that all actions have an attribute *actor*. Notice that while attributes of status functions are mapped onto functions, attributes of event types are represented only by the signature of predicates *happens*, *pre*, and eff. For instance, institutional action *sold* induces predicate $happens_{sold}$ of signature $\xi(happens_{sold}) = \langle \sigma_{aid}, \sigma_{aid}, \sigma_{oid} \rangle$.

4. A SPECIFICATION LANGUAGE FOR INSTITUTIONS

In our framework, properties are specified with a language whose syntax strongly resembles an ordered many-sorted first-order logic with temporal operators. The only difference resides in the fact that we write "$x : \sigma$" to say that variable x is of sort σ. For instance, given that our metamodel ensures that sorts $\sigma_{auctioneer}$ and $\sigma_{customer}$ are disjoint, the following property requires that no agent has contemporary the status function of auctioneer and customer:

$$\begin{aligned}&\mathbf{AG} \neg \exists ag : \sigma_{aid}((\exists a : \sigma_{auctioneer}((subject(a) = ag) \wedge \\ &\quad assigned(a)) \wedge \exists c : \sigma_{customer}((assigned(c) \\ &\qquad \wedge subject(c) = ag)))) \quad \text{(P.1)}\end{aligned}$$

To increase the flexibility of our specification language, occurrences of events can be referenced with a generic predicate *happens*, whose signature is determined by the sort of its first argument. For instance, with the following formula we require that there exists a path where eventually the institutional action *sold* happens:

$$\mathbf{EF} happens(sold, \_, \_, \_) \quad \text{(P.2)}$$

where character "underscore" is used to express existential quantification. In this case, predicate *happens* is mapped onto predicate $happens_{sold}$ of signature $\xi(happens_{sold}) = \langle \sigma_{aid}, \sigma_{aid}, \sigma_{oid} \rangle$ (see Figure 2). Exploiting the flexibility provided by predicate *happens* and by quantifying over sort of institutional events (σ_{ie}), we can require that all institutional events defined by an institution may eventually happen:

$$\forall e : \sigma_{ie}(\mathbf{EF} happens(e)) \tag{P.3}$$

If Property (P.3) holds, it implies that, according to axioms (A.8) and (A.9), we have defined a proper set of conventions and powers. It is worth observing that while properties (P.1) and (P.2) regard specific aspects of the Dutch Auction institution, Property (P.3) states a general desirable feature that any institution ought to satisfy. Indeed, it would be irrational to define an event that cannot happen. This difference is reflected by properties themselves, since Property (P.3) is characterized only by symbols introduced by our metamodel, whereas symbols defined in Figure 2 appear in both properties (P.1) and (P.2). Moreover, while the former two properties are concerned with the *functionality* of our institution, stating that it is possible (impossible) to reach certain states of affairs, Property (P.3) captures an important aspect of the notion of event: it may occur. Thanks to quantification over sorts defined by our conceptual model (e.g., events, status functions, etc.) we envisage that it is possible to define a library of properties that should be satisfied by any institution, enhancing their reuse and ensuring that systems of rules governing open systems are sound with respect to the intended semantics of institutional concepts. For instance, conventions are introduced to link the occurrence of institutional events to the occurrence of other events. Therefore, given a convention which relates events of type x and y, it should be the case that there exists a path where eventually both of them contemporary happen:

$$\begin{aligned}\forall ev_x : \sigma_{ev} \forall ev_y : \sigma_{ie}(&convention(ev_x, ev_y) \rightarrow \\ &\mathbf{EF}(happens(ev_x) \wedge happens(ev_y)))\end{aligned} \tag{P.4}$$

Once we have checked that an auctioneer can sell or withdraw a good, we may require that the auctioneer sells or withdraws the good in all possible interactions regulated by the Dutch Auction:

$$\mathbf{AF}(happens(sold, \_, \_, \_) \vee happens(withdrawGood, \_, \_)) \tag{P.5}$$

If we assume that an auctioneer is autonomous and it is allowed to not comply with norms stated by the Dutch Auction, we should expect that Property (P.5) does not hold. Indeed, it may be the case that a good is never declared sold because an auctioneer obliged to do so ignores its obligations. In general, agents that autonomously act in open systems cannot be assumed to be compliant with norms [13, 15, 35]: as a consequence certain properties may not hold in an institution even if its rules are correctly stated.

To analyze whether an institution may lead a system into certain states when its norms are respected, we can exploit predicate *violated* and the fact that in our framework norms are reified as norm individuals. Therefore, it is possible to quantify over sort σ_o (and its subsorts induced by each norm), investigating how norms condition the evolution of an institution. For instance we can require that while an auctioneer is subject to all norms defined by the Dutch Auction institution, it is always the case that interactions terminate either with the good sold or withdrawn:

$$\begin{aligned}\mathbf{AF}(\exists o : \sigma_o(violated(o)) \vee happens(sold, \_, \_, \_) \vee \\ happens(withdrawGood, \_, \_))\end{aligned} \tag{P.6}$$

Actually, Property (P.6) is insufficient to guarantee that norms lead an auctioneer to declare close an auction. Indeed, if norms were inconsistent, Property (P.6) would be trivially satisfied. Therefore, we should also verify that there exists a path compliant with norms where an auctioneer sells or withdraws a good:

$$\begin{aligned}\mathbf{E}(\forall o : \sigma_o(\neg violated(o))\mathbf{U}(happens(sold, \_, \_, \_) \vee \\ happens(withdrawGood, \_, \_)))\end{aligned} \tag{P.7}$$

Together, properties (P.6) and (P.7) ensure that auctioneers have been provided with powers to successfully carry out their activities and with norms that detect paths where they violate their obligations.

To conclude this section it is worth noticing that whereas formulae described in sections 2 and 3 introduce a set of restrictions A on valuations which characterize respectively any institutional model and any model of the Dutch Auction, OMSFOTL formulae defined with our specification language constitute properties we want to test over models satisfying restrictions A. Moreover, we observe that while in [16] auctioneers are assumed to be always compliant with their norms, our model of the Dutch Auction introduces only norms that constrain the behavior of the auctioneer and it does not define prohibitions to regulate activities of customers, which are permitted to bid whenever they are empowered (see Section 3). As a consequence, since the main differences among our model and the one presented in [16] regard how an institution reacts to violations, if we assume that agents comply with their norms, Property (P.6) holds in our model if and only if it holds according to the rules defined in [16].

5. A TOOL FOR MODEL CHECKING INSTITUTIONS

In this section we will present a new approach to verify institutions modelled with FIEVeL and whose properties are specified with the language described in Section 4. While in our previous attempt to model check institutions [36] FIEVeL constructs were translated into the input language of an existing model checker, namely SPIN [17], in this paper we present a symbolic model checker that has been specifically developed to verify FIEVeL institutions and which is based on the CUDD library [33].

Given an institution described with our modelling language, which corresponds to a set of symbols and axioms of an ordered many-sorted first order temporal logic, to apply symbolic model-checking techniques [5] we define a mapping μ of symbols and a translation τ of axioms to obtain an equivalent propositional model. Models satisfying such formulae correspond to Kripke structures that can be conveniently represented as Ordered Binary Decision Diagrams (OBDDs) [4], a canonical representation of Boolean functions. For this reason, following [5] we will represent a Kripke structure as a tuple $M(\overline{v}) = (S(\overline{v}), S_0(\overline{v}), R(\overline{v}, \overline{v}'))$, where $S(\overline{v})$ is a set of states (assuming that each state corresponds to a valuation of atomic propositions $\overline{v}$), $S_0(\overline{v})$ is a set of initial states, and $R(\overline{v}, \overline{v}')$ is a total relation on $S(\overline{v})$ where $\overline{v}'$ is a second set of propositions used to represent states reachable from states encoded by variables $\overline{v}$.

Atomic propositions $\overline{v}$ are determined by defining a function μ which maps institutional symbols into a set of atomic propositions as follows:

- let N_σ be the cardinality of domain D_σ: each function f of sort σ induces a set of propositions $\overline{v}_{f_{x_1,...,x_n}}$ of cardinality $\log_2(N_\sigma)$ such that $\mu[f(x_1,...,x_n)] = \overline{v}_{f(x_1,...,x_n)}$ for each valuation of variables $\overline{x}$;

- each predicate P induces a proposition $v_{x_1,...,x_n}$ such that $\mu[P(x_1,...,x_n)] = v_{x_1,...,x_n}$ for each valuation of $\overline{x}$;

- constant symbols do not introduce any additional proposition: instead, each constant c is mapped as a sequence of truth values which correspond to the binary representation of the individual referenced by it ($\mu[c] = \mathit{binary}(I(c))$).

It can be shown that if domains and symbols defined by an institution are finite, the size of $\overline{v}$ is finite. Several optimizations can be introduced to reduce the number of atomic propositions, but we omit the details for the sake of brevity.

Given mapping μ and propositions $\overline{v}$, transition relation $R(\overline{v},\overline{v}')$ is obtained by translating the conjunction of axioms φ_i into propositional formulae ($R(\overline{v},\overline{v}') = \tau[\bigwedge \varphi_i]$). Assuming that there exists a constant symbol for each individual of every domain and, for simplicity, that only variables and constants can appear as arguments of functions and predicates, translation τ is defined as follows:

- $\tau[t_1 = t_2] = \bigwedge_{i=0}^{i<N_{\xi(t_1)}} \neg(\mu[t_1]_i \oplus \mu[t_2]_i)$ where $\mu[t]_i$ refers to the i-th proposition (or equivalently the i-th truth value) corresponding to the encoding of term t;

- $\tau[P(x_1,...,x_n)] = \mu[P(x_1,...,x_n)]$;

- $\tau[\forall x \varphi] = \bigwedge_{c \in \mathbf{C}_{\xi(x)}} \tau[\varphi[c/x]]$ where c ranges over constants of sort $\xi(x)$ and $\varphi[c/x]$ is the result of replacing every free occurrence of x in φ with an occurrence of c;

- $\tau[\neg\varphi] = \neg\tau[\varphi]$;

- $\tau[\varphi \wedge \varphi] = \tau[\varphi] \wedge \tau[\varphi]$;

- $\tau[\mathbf{X}\varphi] = \tau[\varphi]'$, that is, we apply translation τ and then variables $\overline{v}$ are substituted with variables $\overline{v}'$;

- $\tau[\mathbf{AG}\varphi] = \tau[\varphi] \wedge \tau[\mathbf{X}\varphi]$ if φ does not contain the *next* temporal operator ($\mathbf{X}$), otherwise $\tau[\mathbf{AG}\varphi] = \tau[\varphi]$.

Although in principle the OBDD corresponding to $R(\overline{v},\overline{v}')$ can be directly obtained by applying τ, in practice its generation tends to be extremely slow. To overcome such problem, we apply the algorithm described in [32] to convert propositional formula $\tau[\bigwedge \varphi_i]$ into conjunctive normal form (CNF), whose satisfying assignments are searched by invoking Minisat [9], an efficient SAT solver. In this case, $R(\overline{v},\overline{v}')$ is defined as the OBDD corresponding to the disjunction of all solutions found by the solver ($R(\overline{v},\overline{v}') = \bigvee_{0 \leq i \leq N} r_i(\overline{v},\overline{v}')$). Finally, the set of initial states is obtained by applying translation τ to a formula which describes the initial settings of an institution and its environment.

Kripke structure $M(\overline{v})$ can be exploited to interactively simulate the evolution of an institution and its environment. In particular, by taking advantage of mapping μ, given a state $s \in S(\overline{v})$ and transitions departing from it, we can provide a high-level description of state s in terms of institutional concepts, allowing the user to choose the following state by selecting one of the base-level events that may occur.

$\lvert\sigma_{aid}\rvert$	$\lvert\overline{v}\rvert$	*clauses*	*time (sec.)*	*solutions*
3	30	678	0.028	80
4	34	788	0.040	165
5	39	1009	0.078	346
6	42	1157	0.137	731

Table 1: Time spent by the SAT solver to find all assignments admitted by a propositional model of the Dutch Auction.

Table 1 reports how the cardinality of sort σ_{aid} affects the number of propositional variables $\overline{v}$ necessary to encode the Dutch Auction institution, the number of clauses generated by the translation of axioms induced by our metamodel into propositional formulae, the number of solutions, and the time spent by the SAT solver to find them. Observing Table 1, we can notice that the number of solutions N found by the SAT solver is considerably smaller than the number of states that can be encoded by variables $\overline{v}$. We can also observe that if a state $s(\overline{v})$ is reachable, then there exists a solution $r(\overline{v},\overline{v}')$ such that the valuation of variables $\overline{v}'$ is equal to the valuation of variables $\overline{v}$ corresponding to state $s(\overline{v})$. As a consequence, N represents an upper bound of the number of reachable states. This fact suggests that once we know the total number of assignments satisfying the CNF formula obtained as the conjunction of axioms induced by the metamodel or FIEVeL constructs, we can build a Kripke structure $M(\overline{w})$, equivalent to $M(\overline{v})$, but defined over a smaller set of variables. In doing so the time required to build the symbolic representation of the institution and the time required to verify its properties can be considerably reduced.

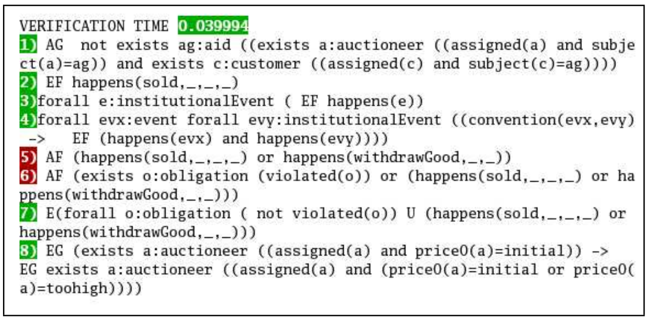
Given a set Φ of properties specified with the language described in Section 4 and a set of assignments of cardinality N, to construct the symbolic representation $M(\overline{w})$ we proceed as follows:

1. the number of variables $\overline{w}$ is determined by the logarithm in base 2 of N increased by 1, since it may be the case that some transitions depart from the initial state but none of them reach it;

2. we define a function ε which maps natural numbers comprised between 0 and N (which are also used to identify states of $M(\overline{w})$) into valuations of variables $\overline{v}'$ encountered by the SAT solver. Notice that function ε creates a correspondence among states of the Kripke structures $M(\overline{v})$ and $M(\overline{w})$;

3. for each property $\varphi \in \Phi$ we first remove all quantifiers as done for translation τ and subsequently replace each subformula not containing temporal operators by introducing an atomic proposition p_i. For instance, Property (P.2) induces the definition of a new proposition p and is transformed into propositional temporal formula $\mathbf{EF}p$;

4. for each proposition p_i we define a set of states $S_{p_i}(\overline{w})$ such that $s(\overline{w}) \in S_{p_i}(\overline{w})$ iff $\varepsilon(s(\overline{w})) \rightarrow \tau[\varphi_{p_i}]$, that is, a state $s(\overline{w})$ belongs to $S_{p_i}(\overline{w})$ if and only if it corresponds to a state $s(\overline{v})$ which satisfies propositional formula $\tau[\varphi_{p_i}]$;



```
VERIFICATION TIME 0.039994
1) AG  not exists ag:aid ((exists a:auctioneer ((assigned(a) and subje
ct(a)=ag)) and exists c:customer ((assigned(c) and subject(c)=ag))))
2) EF happens(sold,_,_,_)
3)forall e:institutionalEvent ( EF happens(e))
4)forall evx:event forall evy:institutionalEvent ((convention(evx,evy)
 ->   EF (happens(evx) and happens(evy))))
5) AF (happens(sold,_,_,_) or happens(withdrawGood,_,_))
6) AF (exists o:obligation (violated(o)) or (happens(sold,_,_,_) or ha
ppens(withdrawGood,_,_)))
7) E(forall o:obligation ( not violated(o)) U (happens(sold,_,_,_) or
happens(withdrawGood,_,_)))
8) EG (exists a:auctioneer ((assigned(a) and price0(a)=initial)) ->
EG exists a:auctioneer ((assigned(a) and (price0(a)=initial or price0(
a)=toohigh))))
```

Figure 3: The report generated by our model checker.

5. analogously, a state $s(\overline{w})$ is an initial state if and only if it corresponds to an initial state of the Kripke structure $M(\overline{v})$. Formally: $s(\overline{w}) \in S_0(\overline{w})$ iff $\varepsilon(s(\overline{w})) \in S_0(\overline{v})$;

6. the transition relation $R(\overline{w}, \overline{w}')$ is obtained as the disjunction of the encoding of solutions found by the SAT solver: $R(\overline{w}, \overline{w}') = \bigvee_{r(v,v')} \varepsilon^{-1}(v) \wedge \varepsilon^{-1}(v')$. It is worth observing that with the exception of the initial state, to build $R(\overline{w}, \overline{w}')$ we can omit solutions which start from unreachable states.

At the end of this process, we obtain a Kripke structure $M(\overline{w})$ which is equivalent to $M(\overline{v})$ and properties Φ can be verified by applying standard symbolic algorithms (see [5]). Notice that to built $M(\overline{w})$ it is not necessary to construct $M(\overline{v})$, but it is sufficient to analyze assignments found by the SAT solver.

As we can observe in Figure 3, when we asked our tool to verify properties described in the previous section, we obtained that properties (P.5) and (P.6) were violated by our formalization of the Dutch Auction. The violation of Property (P.5) was expected, since it stems from our assumption that even an auctioneer may be an autonomous agent and, as such, it may violate norms. Instead, the violation of Property (P.6) was unexpected, since it means that the system of norms of the Dutch Auction defined in [16] does not ensure that an auction ends even when norms are respected. For instance the following evolution is compatible with norms of the Dutch Auction described in Section 3: (i) two or more participants bid when the price is *initial* and norm $n4$ obliges the auctioneer to offer the good at price *toohigh*; (ii) none of the agents accepts price *toohigh* and the auctioneer is obliged by norm $n1$ to decrease the price to the *initial* value; at this point, as in (i) two or more agents bid for price *initial* and, if they will continue doing so and the auctioneer will respect norms $n4$ and $n1$, the system continues iterating among states (i) and (ii). The following property captures this fact, stating that there exists a state where once an auctioneer quotes a good, there exists a path where either the price if higher or is kept equals to the initial price:

$$\begin{aligned} &\mathbf{EG}(\exists a : \sigma_{auctioneer}(assigned(a) \wedge priceA(a) = initial) \\ &\qquad \rightarrow \mathbf{EG}\exists a : \sigma_{auctioneer}(assigned(a) \wedge \\ &\qquad (priceA(a) = initial \vee priceA(a) = toohigh))); \qquad \text{(P.8)} \end{aligned}$$

$\lvert\sigma_{aid}\rvert$	$\lvert\overline{w}\rvert$	$\lvert R(\overline{w})\rvert$	$\lvert S(\overline{w})\rvert$	*time (sec.)*
3	7	120	64	0.005
4	8	253	126	0.011
5	9	572	260	0.028
6	10	1347	550	0.040

Table 2: Cumulative time required to verify all properties discussed in Section 4.

As reported in Figure 3, Property (P.8) holds, which explains why norms do not ensure that a Dutch Auction will terminate. As discussed in Section 4, if we assume that agents comply with norms, our model represents an abstraction of the institution discussed in [16] and our results can be applied to that formalization. For instance, if two agents bid only when a good is offered at a price lower than a fixed threshold, they may lead an auctioneer to never declare closed an auction.

To conclude this section, we report results obtained by checking properties described in Section 4 on a laptop with installed Linux and equipped with a pentium 1.66 GHz and 1 GB of RAM. Table 5 shows how the number of propositional variables $\overline{w}$ and the size of reachable states vary by augmenting the number of agents. Column $|R(\overline{w})|$ shows the actual number of transitions of the Kripke structure $M(\overline{w})$, which is typically greater than the number of solutions found by the SAT solver (see column *solutions* of Table 1). This is achieved by universally abstracting certain variables (e.g., variables corresponding to predicate *violated*) and contributes to reducing the search space of the SAT solver. Finally, we can observe that the number of states that can be encoded with variables $\overline{w}$ is still considerably larger with respect to the actual size of reachable states $S(\overline{w})$. Unfortunately, the exact number of reachable states cannot be determined only by considering the set of assignments encountered by the SAT solver and requires the construction of the Kripke $M(\overline{v})$, which, due to the number of variables $\overline{v}$ used to encode it, tends to be computationally expensive when the number of solutions is big.

6. RELATED WORKS

In the literature there are several attempts to *model*, *simulate*, and *verify* institutions. Our approach provides a single language and a tool to carry out all these tasks, while other proposals typically support only one of them. For instance, in [15, 37] we proposed a notation based on an intuitive semantics of institutional concepts, which does not allow the development of a framework to verify agents' behavior. On the contrary, in [7] Cliffe et al. present a framework for the verification of institutions, but they do not provide a language to model institutional concepts. As a consequence, designers must manually encode and classify such concepts, which increases the size and complexity of institution descriptions and may lead to inconsistencies.

In [28] systems regulated by norms (named normative systems) are described by using a formalism inspired by the Event Calculus. The absence of an institutional metamodel, which for instance provides an axiom to state that every institutional action must be empowered in order to be successfully executed, obliges the authors of [28] to specify this fact for every single action and for every role. Therefore, the definition of a modelling language characterized by a metamodel of institutional reality provides a significant advantage, especially when many status functions (or roles, using the terminology of [28]) are empowered to perform the same institutional action. Furthermore, the definition of an automatic encoding of institutions described in FIEVeL into propositional models allows us to verify our systems, while in [28] the authors must rely on "systematic runs".

In [13] Esteva et al. present a graphical notation to model institutions as labeled transition systems. One of the main advantages of such language resides in the fact that designers have at their disposal a tool for editing institutions [11] and another tool to enact institutions and automatically monitor agents' interactions [12]. To verify such systems, in [18] and [6] the authors describe two syntactic translations of institutions into languages amenable to model checking. For instance, in [18] an institution is translated into a MABLE agent [38], but, since "the meaning of labels is not clear from the notation only" [34], both approaches allow designers to define and verify only properties regarding the syntactic structure of an institution. Moreover, the translation presented in [18] treats norms regarding the performance of communicative acts as if they were always fulfilled by agents, which seems counterintuitive given that institutions are introduced to regulate systems where such assumption is not satisfied [13]. As a consequence, the model checker may answer that a certain property holds when it is not the case and vice versa.

Indeed, reasoning about what properties are satisfied by an institution when agents follow their norms is important to evaluate the *functionality* of institutions (e.g., they allow agents to fulfill their objectives without violating their obligations and prohibitions), but properties holding when agents are not compliant should also be investigated. To do so, in [29] Raimondi and Lomuscio proposed a specification language characterized by a modal operator which discriminates computations compliant with an interaction protocol. The tool described in [29] is very efficient, but its input language requires designers to explicitly list the set of states that each agent may reach, and to classify them as *red* (an agent violates the protocol) or *green*. Although red states are such only because they violate a protocol [29], such classification is not inferred from the protocol but must be manually provided independently from it: therefore designers may introduce discrepancies among the protocol and the classification of states. In this respect, our modelling language offers a high-level description of norms, which are used to automatically determine and classify states and transitions. Moreover, in [29] systems and their properties are described in terms of a fixed set of agents, which reduce the reuse of properties and system descriptions to other possible situations admitted by an institution.

7. CONCLUSIONS

In this paper we have presented a framework which provides a high-level language amenable to model checking to describe institutions in terms of institutional concepts (e.g., status functions, institutional events, etc.) We have also introduced a concise notation to specify properties of institutions and a tool which, starting from a FIEVeL model and a description of basic domains, allows designers to simulate and verify institutions. To exemplify our approach we have modelled the Dutch Auction institution as it has been described in [16] and formalized a set of properties, showing that under certain conditions an interaction ruled by such institution may not terminate.

We are currently exploring how to improve performances of our tool by adopting a more efficient SAT procedure (like the one described in [23]) to find all solutions admitted by a CNF. In the future we will extend our framework to model interdependent institutions, analyzing what kinds of relations may be defined among two institutions and their effects over agent deontic relations.

8. ACKNOWLEDGMENTS

This research has been supported by the Swiss National Science Foundation project 200020-109525, "Artificial Institutions: specification and verification of open distributed interaction frameworks". The authors would like to acknowledge Federico Heras Viaga for making available a modified version of Minisat computing all assignments admitted by a CNF. Moreover, the authors would also like to thank Alessio Lomuscio and Franco Raimondi for their comments on an initial implementation of the symbolic model checker described in this paper.

9. REFERENCES

[1] M. Abadi. The Power of Temporal Proofs. *Theoretical Computer Science*, 65:35–83, 1989.

[2] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[3] G. Boella and L. van der Torre. The Ontological Properties of Social Roles: Definitional Dependence, Powers and Roles Playing Roles. In *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques*, 2005.

[4] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[5] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[6] O. Cliffe and J. Padget. A Framework For Checking Interactions Within Agent Institutions. In *Proceedings of the ECAI Workshop on Model Checking and Artificial Intelligence*, 2002.

[7] O. Cliffe, M. D. Vos, and J. Padget. Specifying and Analysing Agent-based Social Institutions using Answer Set Programming. In *Coordination, Organization, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *LNAI*, pages 99–113. Springer, 2006.

[8] S. Cranefield. Modelling and Monitoring Social Expectations in Multi-Agent Systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems II*, volume 4386 of *LNCS*, 2007.

[9] N. Eén and N. Sörensson. An Extensible SAT-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing.*, volume 2919 of *LNCS*, pages 502–518. Springer, 2004.

[10] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[11] M. Esteva, D. de la Cruz, and C. Sierra. ISLANDER: an electronic institutions editor. In *Proceedings of the*

1st Conference on Autonomous Agents and Multiagent Systems, pages 1045–1052, 2002.

[12] M. Esteva, J. A. Rodríguez-Aguilar, B. Rosell, and J. L. Arcos. AMELI: An Agent-based Middleware for Electronic Institutions. In *Proceedings of the 3rd Conference on Autonomous Agents and Multi-Agent Systems*, pages 236–243, 2004.

[13] M. Esteva, J. A. Rodríguez-Aguilar, C. Sierra, P. Garcia, and J. L. Arcos. On the Formal Specification of Electronic Institutions. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective*, volume 1991 of *LNAI*, pages 126–147. Springer, 2001.

[14] M. Esteva, J. A. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. Verifying Norm Consistency in Electronic Institutions. In *Proceedings of the Workshop on Agent Organizations: Theory and Practice*, pages 8–15, 2004.

[15] N. Fornara, F. Viganò, and M. Colombetti. Agent Communication and Artificial Institutions. *Autonomous Agents and Multi-Agent Systems*, 14(2):121–142, 2007.

[16] A. García-Camino, J.-A. Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos. A Rule-based Approach to Norm-Oriented Programming of Electronic Institutions. *ACM SIGecom Exchanges*, 5(5):33–40, 2006.

[17] G. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2003.

[18] M.-P. Huget, M. Esteva, S. Phelps, C. Sierra, and M. Wooldridge. Model Checking Electronic Institutions. In *Proceedings of the ECAI Workshop on Model Checking and Artificial Intelligence*, 2002.

[19] A. Jones and M. J. Sergot. A formal characterisation of institutionalised power. *Journal of the IGPL*, 4(3):429–445, 1996.

[20] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture–Practice and Promise*. Addison-Wesley Professional, 2003.

[21] M. Manzano. Introduction to many-sorted logic. In *Many-sorted logic and its applications*, pages 3–86. John Wiley & Sons, 1993.

[22] C. Masolo, L. Vieu, E. Bottazzi, C. Catenacci, R. Ferrario, A. Gangemi, and N. Guarino. Social Roles and their Descriptions. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 267–277, 2004.

[23] K. L. McMillan. Applying SAT Methods in Unbounded Symbolic Model Checking. In E. Brinksma and K. Larsen, editors, *Proceedings of the 14th Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 250–264. Springer, 2002.

[24] Y. Moses and M. Tennenholtz. Artificial Social Systems. *Computers and AI*, 14(6):533–562, 1995.

[25] P. Noriega. *Agent mediated auctions: The Fishmarket Metaphor*. PhD thesis, Universitat Autònoma de Barcelona, 1997.

[26] D. North. *Institutions, Institutional Change and Economics Performance*. Cambridge University Press, 1990.

[27] D. Peled. *Software reliability methods*. Texts in Computer Science. Springer, 2001.

[28] J. Pitt, L. Kamara, M. Sergot, and A. Artikis. Formalization of a voting protocol for virtual organizations. In *Proceedings of the 4th Conference on Autonomous agents and Multi-Agent Systems*, pages 373–380, 2005.

[29] F. Raimondi and A. Lomuscio. Automatic verification of deontic interpreted systems by model checking via obdd's. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence*, pages 53–57, 2004.

[30] J. R. Searle. *The construction of social reality*. Free Press, 1995.

[31] J. R. Searle. What is an institution? *Journal of Institutional Economics*, 1(01):1–22, 2005.

[32] D. Sheridan. The optimality of a fast cnf conversion and its use with sat. In *The 7th International Conference on Theory and Applications of Satisfiability Testing*, 2004.

[33] F. Somenzi. CUDD: CU Decision Diagram Package. http://vlsi.colorado.edu/ fabio/CUDD/.

[34] W. Vasconcelos. Logic-Based Electronic Institutions. In *Proceedings of the Workshop on Declarative Agent Languages and Technologies*, volume 2990 of *LNCS*, pages 221–242. Springer, 2004.

[35] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in Multiagent Systems: from Theory to Practice. *International Journal of Computer Systems Science & Engineering*, 20(4):225–236, 2005.

[36] F. Viganò and M. Colombetti. Specification and Verification of Institutions through Status Functions. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems II*, volume 4386 of *LNCS*, 2007.

[37] F. Viganò, N. Fornara, and M. Colombetti. An Event Driven Approach to Norms in Artificial Institutions. In *Coordination, Organization, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *LNAI*, pages 142–154. Springer, 2006.

[38] M. Wooldridge, M.-P. Huget, M. Fisher, and S. Parsons. Model Checking for Multiagent Systems: the Mable Language and its Applications. *International Journal on Artificial Intelligence Tools*, 15(2):195–226, 2006.