Norm Consistency in Electronic Institutions

M. Esteva[1], W. Vasconcelos[2], C. Sierra[1], J. A. Rodríguez-Aguilar[1]

[1] Institut d'Investigació en Intel·ligència Artificial
CSIC, Campus UAB
08193 Bellaterra, Spain
{marc, sierra, jar}@iiia.csic.es

[2] Department of Computing Science
University of Aberdeen
AB24 3UE, Aberdeen, United Kingdom
wvasconc@csd.abdn.ac.uk

Abstract. Electronic institutions are a formalism to define and analyse protocols among agents with a view to achieving global and individual goals. In this paper we elaborate on the verification of properties of electronic institutions based on the dialogues that agents may hold. Specifically, we provide a computational approach to assess whether an electronic institution is *normatively consistent*. In this manner, given an electronic institution we can determine whether its norms prevent norm-compliant executions from happening. For this we strongly rely on the analysis of the dialogues that may occur as agents interact by exchanging illocutions in an electronic institution.

1 Introduction

An important aspect in the design of heterogeneous multiagent systems concerns the *norms* that should constrain and influence the behaviour of its individual components [1–3]. Electronic institutions have been proposed as a formalism to define and analyse protocols among agents with a view to achieving global and individual goals [4, 5]. In this paper we propose a definition for norms and a means of assessing whether an electronic institution is *normatively consistent*. In other words, given an electronic institution specification, we want to determine whether its norms prevent norm-compliant executions from taking place.

Norms are a central component of electronic institutions. As such, it is fundamental to guarantee that they are not wrongly specified, leading to unexpected executions of electronic institutions, and that an electronic institution indeed complies with its norms. For this purpose we strongly rely on the analysis of the dialogues that may occur as agents interact by exchanging illocutions in an electronic institution. Since the execution of an electronic institution can be regarded as a multi-agent dialogue, we can analyse such dialogue to assess whether it abides by the institutional norms. Hence, execution models of electronic institutions can be obtained as dialogue models. Thus, our approach can be regarded as a model checking process based on the construction of models for the enactment of electronic institutions. Hereafter, the purpose of the verification process is to evaluate whether such models satisfy the institutional norms.

In the next section we define the components of an electronic institution. In the ensuing section we introduce a precise definition of norms and subsequently show how they can be verified. Finally we present our conclusions, compare our research with related work and give directions for future work.

2 Electronic Institutions

In general terms, electronic institutions (EIs) structure agent interactions, establishing what agents are permitted and forbidden to do as well as the consequences of their actions. Next, we put forth definitions of the components of

an EI – these are more thoroughly described in [6]. We assume in this paper the existence of a finite and non-empty set $Ag = \{ag_1, \ldots, ag_n\}$ of unique agent identifiers $ag_i \neq ag_j, i \neq j, 1 \leq i, j \leq n$.

2.1 Dialogic Frameworks and Dialogues

In the most general case, each agent immersed in a multi-agent environment is endowed with its own inner language and ontology. In order to allow agents to interact with other agents we must address the fundamental issue of putting their languages and ontologies in relation. For this purpose, we propose that agents share, when communicating, what we call the *dialogic framework* that contains the elements for the construction of the agent communication language expressions. Furthermore the dialogic framework also defines the roles that participating agents can play.

Definition 1. *A dialogic framework is a tuple $DF = \langle O, L_O, P, R\rangle$ where O stands for an ontology (vocabulary); L_O stands for a content language to express the information exchanged between agents using ontology O; P is the set of illocutionary particles; and R is the set of internal roles.*

Within a dialogic framework the content language allows for the encoding of the knowledge to be exchanged among agents using the vocabulary offered by the ontology. The expressions of the agent communication language are defined as below:

Definition 2. *The language $\mathcal{L}_{DF}$ of a dialogic framework $DF = \langle O, L_O, P, R\rangle$ is the set of expressions $\iota(ag, r, ag', r', p, t)$ such that $\iota \in P$; $ag, ag' \in Ag$, the set of agent identifiers; $r, r' \in R$; $p \in L_O$ a variable-free expression of L_O; $t \in \mathbb{N}$ is a time tag.*

That is, the language of a dialogic framework is the collection of all the grounded, variable-free expressions that agents employing the dialogic framework may exchange. Intuitively, $\iota(ag, r, ag', r', p, t)$ denotes that agent ag incorporating role r sent to agent ag' incorporating role r' contents p at instant t.

We also need to refer to expressions which may contain variables. We provide the following definition with this aim:

Definition 3. *The pattern language $\mathcal{L}^*_{DF}$ of a dialogic framework $DF = \langle O, L_O, P, R\rangle$ is the set of expressions $\iota(ag^*, r^*, ag'^*, r'^*, p^*, t^*)$ such that $\iota \in P$; ag^*, ag'^* are agent variables or agent identifiers from the set Ag; r^*, r'^* are role variables or role identifiers in R; $p^* \in L_O$ is an expression which may contain variables; and t^* is either a time variable or a value in $\mathbb{N}$.*

Henceforth we shall refer to $\mathcal{L}_{DF}$ expressions as *illocutions*, represented generically as $\mathbf{i}$, and to $\mathcal{L}^*_{DF}$ expressions as *illocution schemes*, represented generically as $\mathbf{i}^*$. It follows from the definitions above that $\mathcal{L}_{DF} \subseteq \mathcal{L}^*_{DF}$.

Although a dialogic framework defines a set of illocutions that agents may exchange, we consider that agents, as human beings, engage in conversations. Conversations structure agents' interactions, by imposing an order on the illocutions exchange and represent the context where exchanged illocutions must be interpreted. As a conversation evolves, a *dialogue*, an ordered sequence of all illocutions exchanged among agents, is generated.

Dialogues represent the history of conversations and the analysis of the properties of a conversation can be conducted on the basis of its dialogues. We hence formally introduce the notion of dialogue as a core element upon which we carry out the analysis of conversations and, ultimately, of dialogic institutions:

Definition 4. *Given a dialogic framework DF and its language $\mathcal{L}_{DF}$, we define a dialogue over $\mathcal{L}_{DF}$ as any non-empty, finite sequence $\langle \mathbf{i}_1, \ldots, \mathbf{i}_n\rangle$ such that $\mathbf{i}_i = \iota_i(ag_i, r_i, ag'_i, r'_i, p_i, t_i) \in \mathcal{L}_{DF}, 1 \leq i \leq n$, and $t_i \leq t_j, 1 \leq i \leq j \leq n$.*

From the definition above we obtain all the possible dialogues that a group of agents using a dialogic framework may have. We next define the set of all possible dialogues of a dialogic framework:

Definition 5. *Given a dialogic framework DF, we define the dialogue set over $\mathcal{L}_{DF}$, noted as $\mathcal{D}_{DF}$, as the set containing all possible dialogues over $\mathcal{L}_{DF}$.*

Clearly, the set $\mathcal{D}_{DF}$ of all possible dialogues is infinite, even though the components of the corresponding dialogic framework DF are finite – the very same illocution can be uttered an infinite number of times with different time stamps.

A single dialogue solely contains only *grounded* illocutions. If we consider instead a sequence of illocution schemes $\mathbf{i}^*$, the very same sequence may produce multiple dialogues as values are bound to the free variables in the illocution schemes. Therefore, we can employ a sequence of illocution schemes for representing a whole set of dialogues that may occur. And thus, we can think of undertaking the analysis of a set of dialogues from the sequence of illocution schemes that generates them.

Definition 6. *Given a dialogic framework DF and its pattern language $\mathcal{L}^*_{DF}$, we define a dialogue scheme over $\mathcal{L}^*_{DF}$ as any non-empty, finite sequence $\langle \mathbf{i}^*_1, \ldots, \mathbf{i}^*_n \rangle$ such that $\mathbf{i}^*_i \in \mathcal{L}^*_{DF}, 1 \leq i \leq n$.*

In order to relate dialogue schemes and dialogues we rely on the concept of *substitution*, that is, the set of values for variables in a computation [7, 8]:

Definition 7. *A substitution $\sigma = \{x_0/T_0, \ldots, x_n/T_n\}$ is a finite and possibly empty set of pairs x_i/T_i, x_i being a first-order variable and T_i an arbitrary first-order term.*

A dialogue scheme and a dialogue are thus related:

Definition 8. *Given a dialogic framework DF, we say that a dialogue scheme $\langle \mathbf{i}^*_1, \ldots, \mathbf{i}^*_n \rangle \in \mathcal{L}^*_{DF}$ is a scheme of a dialogue $\langle \mathbf{i}_1, \ldots, \mathbf{i}_n \rangle \in \mathcal{L}_{DF}$ iff there is a substitution σ that when applied to $\langle \mathbf{i}^*_1, \ldots, \mathbf{i}^*_n \rangle$ yields (or unifies with) $\langle \mathbf{i}_1, \ldots, \mathbf{i}_n \rangle$, that is, $\langle \mathbf{i}^*_1, \ldots, \mathbf{i}^*_n \rangle \cdot \sigma = \langle \mathbf{i}_1, \ldots, \mathbf{i}_n \rangle$.*

The application of a substitution to a dialogue scheme $\langle \mathbf{i}^*_1, \ldots, \mathbf{i}^*_n \rangle \cdot \sigma$ is defined as the application of the substitution σ to each $\mathbf{i}^*_i$, that is, $\langle \mathbf{i}^*_1, \ldots, \mathbf{i}^*_n \rangle \cdot \sigma = \langle \mathbf{i}^*_1 \cdot \sigma, \ldots, \mathbf{i}^*_n \cdot \sigma \rangle$. The application of a substitution to $\mathbf{i}^*_i$ follows the usual definition [8]:

1. $c \cdot \sigma = c$ for a constant c.
2. $[\iota(ag^*, r^*, ag'^*, r'^*, p^*, t^*)] \cdot \sigma = \iota(ag^* \cdot \sigma, r^* \cdot \sigma, ag'^* \cdot \sigma, r'^* \cdot \sigma, p^* \cdot \sigma, t^* \cdot \sigma)$.
3. $x \cdot \sigma = T \cdot \sigma$ for a variable x such that $x/T \in \sigma$; if $x/T \notin \sigma$ then $x \cdot \sigma = x$.

The first case defines the application of a substitution to a constant c. Case 2 describes the application of a substitution to a generic illocution scheme $\mathbf{i}^* = \iota(ag^*, r^*, ag'^*, r'^*, p^*, t^*)$: the result is the application of σ to each component of the scheme. Case 3 describes the application of σ to a generic variable x: the result is the application of σ to the term T to which x is associated (if $x/T \in \sigma$) or x itself if x is not associated to terms in σ.

2.2 Scenes

Within the framework of an electronic institution, agent conversations are articulated through agent group meetings, called *scenes*, that follow well-defined interaction protocols. In other words, not all sequences in $\mathcal{D}_{DF}$ make sense, so some structure upon dialogues seems unavoidable.

A scene protocol is specified by a directed graph whose nodes represent the different states of a dialogic interaction among roles. From the set of states we

differentiate an initial state (non reachable once left) and a set of final states representing the different dialogue ends. In order to capture that final states represent the end of a dialogue they do not have outgoing arcs. The arcs of the graph are labelled with illocution schemes (whose sender, receiver, content, and time tag may contain variables). We formally define scenes as follows:

Definition 9. *A scene is a tuple* $S = \langle R, DF, W, w_0, W_f, \Theta, \lambda, min, Max\rangle$ *where* R *is the set of scene roles;* DF *is a dialogic framework;* W *is the set of scene states;* $w_0 \in W$ *is the initial state;* $W_f \subseteq W$ *is the set of final states;* $\Theta \subseteq W \times W$ *is a set of directed edges;* $\lambda : \Theta \longrightarrow \mathcal{L}^*_{DF}$ *is a labelling function, which maps each edge to an illocution scheme in the pattern language of the* DF *dialogic framework;* $min, Max : R \longrightarrow \mathbb{N}$ $min(r)$ *and* $Max(r)$ *return the minimum and maximum number of agents that must and can play role* $r \in R$*.*

We formally define the dialogue schemes of a scene:

Definition 10. *The dialogue schemes* $\mathcal{D}^*_S$ *of a scene* $S = \langle R, DF, W, w_0, W_f, \Theta, \lambda, min, Max\rangle$*, is the set of sequences* $\langle (s^*, w_2, \lambda(w_1, w_2)), \ldots, (s^*, w_n, \lambda(w_{n-1}, w_n))\rangle$*, where* s^* *is the identifier for scene* S *or a variable, and* $w_1 = w_0$*, and* $w_n \in W_f$*.*

The dialogue schemes of a scene are the sequence of labels $\lambda(w, w')$ of all paths connecting its initial state w_0 to a final state $w_n \in W_f$. The s and w's are required to precisely identify the context in which an illocution was uttered – this is essential to our notion of norms, as we shall see below.

The dialogue schemes of a scene allow us to obtain all the concrete dialogues accepted by the scene via appropriate substitutions assigning values to all variables of the illocutions. We define the set of all (concrete) dialogues of a scene as follows:

Definition 11. *The dialogues* $\mathcal{D}_S$ *of a scene* $S = \langle R, DF, W, w_0, W_f, \Theta, \lambda, min, Max\rangle$*, is the set of sequences* $\langle (s^*, w_2, \lambda(w_1, w_2)), \ldots, (s^*, w_n, \lambda(w_{n-1}, w_n))\rangle \cdot \sigma$*, where* s *is the identifier for scene* S*,* $w_1 = w_0$*, and* $w_n \in W_f$*, and* σ *is a substitution providing values to all variables of the illocution schemes* $\lambda(w, w')$*.*

The dialogues accepted by a scene are the ground versions of its dialogue schemes, that is, the sequence of labels of a path through the scene with all variables replaced with constants. Given a dialogue scheme we can derive infinite ground versions of it – however, as we shall see below, we provide means to express constraints on the values that the illocutions' variables may have. Extra constraints limit the number of possible applicable substitutions and, hence, concrete dialogues.

2.3 Performative Structures

While a scene models a particular multi-agent dialogic activity, more complex activities can be specified by establishing networks of scenes (activities), the so-called *performative structures*. These define how agents can legally move among different scenes (from activity to activity). Agents within a performative structure can participate concurrently in different scenes.

Definition 12. *Performative structures are recursively defined as:*

- *A scene* S *is a performative structure.*
- *If* PS_1 *and* PS_2 *are performative structures,* $PS_1.PS_2$ *is a performative structure, where* $PS_1.PS_2$ *means that the execution of* PS_1 *is followed by the execution of* PS_2*.*

– *If PS_1 and PS_2 are performative structures, $PS_1|PS_2$ is a performative structure, where $PS_1|PS_2$ stands for the interleaved execution of PS_1 and PS_2.*
– *If PS is a performative structure, PS^n is a performative structure, where PS^n stands for n executions of PS, where $n \in \mathbb{N}, n \geq 0$.*

A performative structure defines all the dialogues that agents may have within an electronic institution, by fixing the scenes in which agents can be engaged and how agents can move among them. Notice that the execution of a performative structure must be regarded as the execution of its different scenes. Moreover, executions of different scenes can occur concurrently.

We can define the dialogues accepted by performative structure PS, denoted by $\mathcal{D}_{PS}$ using the definition of performative structures above. For that we must define the operator $\oplus : \mathcal{D}_{PS_1} \times \mathcal{D}_{PS_2} \rightarrow \mathcal{D}_{PS_1|PS_2}$ that given two input dialogues d_{PS_1} and d_{PS_2} merges their illocutions taking into account their time stamps. More formally, given dialogues $d_1 = \langle \mathbf{i}_1^1, \ldots, \mathbf{i}_n^1 \rangle$ and $d_2 = \langle \mathbf{i}_1^2, \ldots, \mathbf{i}_m^2 \rangle$, $d_1 \oplus d_2 = \langle \mathbf{i}_1, \ldots, \mathbf{i}_{n+m} \rangle$, where $\mathbf{i}_i = \mathbf{i}_j^1$ or $\mathbf{i}_i = \mathbf{i}_k^2$, for some $j, k, 1 \leq j \leq n, 1 \leq k \leq$. Furthermore, for any two illocutions $\mathbf{i}_i = \iota_i(ag_i, r_i, ag_i', r_i', p_i, t_i)$ and $\mathbf{i}_l = \iota_l(ag_l, r_l, ag_l', r_l', p_l, t_l)$, such that $1 \leq i \leq l \leq n + m$, then $t_i \leq t_l$. The concatenation operator "$\circ$" over sequences is defined in the usual fashion.

We can now define the dialogues accepted by each performative structure.

Definition 13. *The dialogues $\mathcal{D}_{PS}$ of a performative structure PS are thus obtained:*

– *If $PS = S$, i.e., a scene, then $\mathcal{D}_{PS} = \mathcal{D}_S$ as in Def. 11.*
– *If $PS = PS_1.PS_2$ then $\mathcal{D}_{PS} = \{d_{PS_1} \circ d_{PS_2} | d_{PS_1} \in \mathcal{D}_{PS_1}, d_{PS_2} \in \mathcal{D}_{PS_2}\}$.*
– *If $PS = PS_1|PS_2$ then $\mathcal{D}_{PS} = \{d_{PS_1} \oplus d_{PS_2} | d_{PS_1} \in \mathcal{D}_{PS_1}, d_{PS_2} \in \mathcal{D}_{PS_2}\}$.*
– *If PS is of the form PS^n then $\mathcal{D}_{PS} = \{d_0 \circ \ldots \circ d_n | d_i \in \mathcal{D}_{PS}, 0 \leq i \leq n\}$.*

3 Norms in Electronic Institutions

Agents' actions *within* scenes may have consequences that either limit or enlarge their future acting possibilities. Some actions may create commitments for future actions, interpreted as obligations, and some actions can have consequences that modify the valid illocutions or the paths that a scene evolution can follow. These consequences are captured by a special type of rules called *norms* which contain the actions that will activate them and their consequences. Notice that we are considering dialogic institutions, and the only actions we consider are the utterance of illocutions. In order to express actions within norms and obligations we set out two predicates:

– $uttered(s, w, \mathbf{i}^*)$ denoting that a grounded illocution unifying with the illocution scheme $\mathbf{i}^*$ has been uttered at state w of scene S identified by s.
– $uttered(s, \mathbf{i}^*)$ denoting that a grounded illocution unifying with the illocution scheme $\mathbf{i}^*$ has been uttered at some state of scene S identified by s.

Therefore, we can refer to the utterance of an illocution within a scene or when a scene execution is at a concrete state.

3.1 Boolean Expressions

In some cases the activation of a norm will depend on the values bound to the variables in the illocution schemes and on the context of the scene (the previous bindings) where the illocution is uttered. With this aim, we incorporate boolean

functions over illocution schemes' variables as antecedents and consequents of norms.

The expressions over illocution schemes variables have the following syntax: e_i *op* e_j; where e_i and e_j are expressions of the appropriate type. The types of variables must be of any basic type: string, numeric and boolean, a type defined in the ontology, or a multi-set of them. We are currently using this reduced set of operators:

- $<, \leq, \geq, >$: $numeric \times numeric \rightarrow boolean$
- $\vee : boolean \times boolean \rightarrow boolean$
- $=, \neq$: $\alpha^{=} \times \alpha^{=} \rightarrow boolean$ where α represents a basic type or any type defined in the ontology.
- $\in, \notin$: $\alpha^{=} \times \alpha^{=} multiset \rightarrow boolean$, where α represents a basic type or any type defined in the ontology that may have equality.
- $\subset, \subseteq$: $\alpha^{=} multiset \times \alpha^{=} multiset \rightarrow boolean$, where α represents a basic type of any type defined in the ontology.

Notice that illocutions are tagged with the time at which the illocution is uttered. We consider such tags as numeric, and so, we can apply to them the same operations as to numeric expressions. Hence, order among the utterance of illocutions can be expressed via numeric operators over them.

Moreover, when a scene is executed we keep all the bindings produced by the uttered illocutions. Therefore, we can make reference to the last one or to a set of bindings for a giving variable(s) and use this in the expressions mentioned above. Concretely, we can apply the following prefix operators to obtain previous bindings:

- $!x$: stands for the last binding of variable x.
- $!^{k}_{w_i w_j} x$: stands for the multi-set of all the bindings of variable x in the k last subdialogues between w_i and w_j. $!^{1}_{w_i w_j} x$ is noted as $!_{w_i w_j} x$ for simplicity.
- $!^{*}_{w_i w_j} x$: stands for the multiset of the bindings of variable x in all subdialogues between w_i and w_j.
- $!^{k}_{w_i w_j} x$ (*cond*): stands for the multi-set of all the bindings of variable x in the k last sub-dialogues between w_i and w_j such that the substitution σ where the binding appears satisfies the *cond* condition.

3.2 Integrity Norms and Obligations

We now put forth formal definitions for two types of norms in electronic institutions, the integrity norms and obligations. In both definitions below we can also have $uttered(s, \mathbf{i}^*)$ subformulae.

Definition 14. *Integrity norms are first-order formulae of the form*

$$\left(\bigwedge_{i=1}^{n} uttered(s_i, w_{k_i}, \mathbf{i}_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j\right) \rightarrow \bot$$

where s_i are scene identifiers, w_{k_i} is a state k_i of scene s_i, $\mathbf{i}_{l_i}$ is an illocution scheme l_i of scene s_i and e_j are boolean expressions over variables from the illocution schemes $\mathbf{i}_{l_i}$.

Integrity norms define sets of actions that *must not* occur within an institution. The meaning of these norms is that if grounded illocutions matching the illocution schemes $\mathbf{i}_{l_1}, \ldots, \mathbf{i}_{l_n}$ are uttered in the corresponding scene states, and expressions $e_1, \ldots, e_m$ are satisfied, then a *violation* occurs ($\bot$).

Definition 15. *Obligations are first-order formulae of the form*

$$\left(\bigwedge_{i=1}^{n} \mathit{uttered}(s_i, w_{k_i}, \mathbf{i}^*_{l_i}) \wedge \bigwedge_{j=0}^{m} e_j\right) \rightarrow \left(\bigwedge_{i=1}^{n'} \mathit{uttered}(s'_i, w'_{k_i}, \mathbf{i}'^*_{l_i}) \wedge \bigwedge_{j=0}^{m'} e'_j\right)$$

where s_i, s'_i are scene identifiers, w_{k_i}, w'_{k_i} are states of s_i and s'_i respectively, $\mathbf{i}^*_{l_i}, \mathbf{i}'^*_{l_i}$ are illocution schemes l_i of scenes s_i and s'_i respectively, and e_j, e'_j are boolean expressions over variables from the illocution schemes $\mathbf{i}^*_{l_i}$ and $\mathbf{i}'^*_{l_i}$, respectively.

The intuitive meaning of obligations is that if grounded illocutions matching $\mathbf{i}^*_{l_1}, \ldots, \mathbf{i}^*_{l_n}$ are uttered in the corresponding scene states, and the expressions $e_1, \ldots, e_m$ are satisfied, then, grounded illocutions matching $\mathbf{i}'^*_{l'_1}, \ldots, \mathbf{i}'^*_{l'_n}$ satisfying the expressions $e'_1, \ldots, e'_{m'}$ must be uttered in the corresponding scene states.

Obligations assume a temporal ordering: the left-hand side illocutions must have time stamps which precede those of right-hand side illocutions. Rather than requiring that engineers manually encode such restrictions, we can automatically add them – given our definition above, we can add the extra boolean expressions $t^* \geq t^*_i, 1 \leq i \leq n, t'^* \leq t'^*_j, 1 \leq j \leq n', t^* < t'^*$ to our obligations, where t^*, t^*_i and t'^*, t'^*_j are the time stamps of, respectively, $\mathbf{i}^*_i$ and $\mathbf{i}'^*_j$, t^* being the greatest value of time stamp on the left-hand side illocutions (that is, the time stamp of the latest illocution) and t'^* the lowest value of time stamp on the right-hand side illocutions (that is, the time stamp of the earliest illocution).

3.3 Semantics of Norms

In order to define the semantics of our norms and obligations we need first to define the meaning of the predicates $\mathit{uttered}(s, w, \mathbf{i}^*)$ and $\mathit{uttered}(s, \mathbf{i}^*)$. We shall employ a function $\mathbf{K} : \mathbf{i}^* \times \mathcal{D} \times \sigma \mapsto \{\mathbf{true}, \mathbf{false}\}$, that maps illocution schemes, dialogues and substitutions to **true** and **false**[1].

Definition 16. *The semantics of $u = \mathit{uttered}(s, w, \mathbf{i}^*)$ or $u = \mathit{uttered}(s, \mathbf{i})$ wrt a set of dialogues $\mathcal{D}$ and a substitution σ, $\mathbf{K}(u, \mathcal{D}, \sigma) \mapsto \{\mathbf{true}, \mathbf{false}\}$, is:*

1. $\mathbf{K}(\mathit{uttered}(s, w, \mathbf{i}^*), \mathcal{D}, \sigma) = \mathbf{true}$ *iff there is a dialogue $\langle (s, w_1, \mathbf{i}^*_1), \ldots, (s, w_n, \mathbf{i}^*_n)) \rangle \cdot \sigma \in \mathcal{D}$ with $(s, w_i, \mathbf{i}^*_i) = (s, w, \mathbf{i}^*)$, for some $i, 1 \leq i \leq n$.*
2. $\mathbf{K}(\mathit{uttered}(s, w, \mathbf{i}^*), \mathcal{D}, \sigma) = \mathbf{true}$ *iff there is a dialogue $\langle (s, w_1, \mathbf{i}^*_1), \ldots, (s, w_n, \mathbf{i}^*_n)) \rangle \cdot \sigma \in \mathcal{D}$ with $(s, w_i, \mathbf{i}^*_i) = (s, w_i, \mathbf{i}^*)$, for some $i, 1 \leq i \leq n$.*

Our predicates are true if there is at least one dialogue $\langle (s, w_1, \mathbf{i}^*_1), \ldots, (s, w_n, \mathbf{i}^*_n)) \rangle \cdot \sigma$ in $\mathcal{D}$ with an element $(s, w_i, \mathbf{i}^*_i)$ (an illocution of the dialogue without its substitution σ applied to it) in it that is syntactically equal to $(s, w, \mathbf{i}^*)$. In the case of $\mathit{uttered}(s, \mathbf{i}^*)$ we do not care what the value of w is.

In the definition above, we can understand σ as parameter whose value is determined, confirmed or completed by the function. The substitution σ plays an essential role in finding the truth value of our boolean expressions.

We also need to define a semantic mapping for our boolean expressions e over illocution scheme variables. We shall use the same $\mathbf{K}$ function introduced above, extending it to cope with expressions e as introduced previously.

Definition 17. *The semantics of a boolean expression e wrt a set of dialogues $\mathcal{D}$ and a substitution σ, $\mathbf{K}(e, \mathcal{D}, \sigma) \mapsto \{\mathbf{true}, \mathbf{false}\}$, is*

$$\mathbf{K}(e_1 \mathit{\ op\ } e_2, \mathcal{D}, \sigma) = \mathbf{true} \mathit{\ iff\ } \mathbf{K}'(e_1, \sigma) \mathit{\ op\ } \mathbf{K}'(e_2, \sigma)$$

[1] We distinguish between the constants "$\top$" and "$\bot$" which are part of the syntax of formulae and the truth-values **true** and **false**. Clearly, $\mathbf{K}(\top, \mathcal{D}, \sigma) = \mathbf{true}$ and $\mathbf{K}(\bot, \mathcal{D}, \sigma) = \mathbf{false}$ for any $\mathcal{D}$ and σ.

The "*op*" operators are all given their usual definition. For instance, $\mathbf{K}(x \in Ag, PS, \sigma) = \textbf{true}$ iff $\mathbf{K}'(x, \sigma) \in \mathbf{K}'(Ag, \sigma)$, that is, the expression is true iff the value of variable x in σ belongs to the set of values comprising set Ag. The auxiliary mapping $\mathbf{K}' : e \times \sigma \mapsto \Im$, where $\Im$ is the union of all types in the ontology, is defined below.

Definition 18. *The value of a non-boolean expression e wrt a substitution* σ, $\mathbf{K}'(e, \sigma) \mapsto \Im$, *is:*

1. $\mathbf{K}'(c, \sigma) = c$ *for a constant* c.
2. $\mathbf{K}'(x, \sigma) = T', \mathbf{K}'(T, \sigma) = T', x/T \in \sigma$.
3. $\mathbf{K}'(f(T_1, \ldots, T_n), \sigma) = f(\mathbf{K}'(T_1, \sigma), \ldots, \mathbf{K}'(T_n, \sigma))$.
4. $\mathbf{K}'(\mathsf{Set}, \sigma) = \{c_0, \ldots\}$, $\mathsf{Set}/\{c_0, \ldots\} \in \sigma$.

Case 1 defines the value of a constant as the constant itself. Case 2 describes how to obtain the value of an arbitrary variable x appearing in illocution schemes. Case 3 describes how functions are evaluated: the meaning of a function is given by the application of the function to the value of its arguments. Finally, case 4 defines the value of a set as the set of values associated with the set name in the substitution σ – sets are treated like any other ordinary constant.

We finally define the meaning of our norms, depicting how the logical operators "$\wedge$" and "$\rightarrow$" are handled in our formalisation. Our atomic formulae are u or e, denoted generically as Atf; $Atfs$ denotes a conjunction of atomic formulae, $Atfs = Atf_1 \wedge \cdots \wedge Atf_n$. The logical operators are defined in the usual way:

Definition 19. *The semantics of a norm is given by*

1. $\mathbf{K}(Atfs_1 \rightarrow Atfs_2, \mathcal{D}, \sigma) = \textbf{true}$ *iff* $\mathbf{K}(Atfs_1, \mathcal{D}, \sigma) = \textbf{false}$ *or* $\mathbf{K}(Atfs_1, \mathcal{D}, \sigma) = \mathbf{K}(Atfs_2, \mathcal{D}, \sigma) = \textbf{true}$
2. $\mathbf{K}(Atfs_1 \wedge Atfs_2, \mathcal{D}, \sigma) = \textbf{true}$ *iff* $\mathbf{K}(Atfs_1, \mathcal{D}, \sigma) = \mathbf{K}(Atfs_2, \mathcal{D}, \sigma) = \textbf{true}$

Case 1 depicts the semantics of the "$\rightarrow$" operator: it yields **true** if the formulae on its left and right side evaluate to the same truth-value. Case 2 captures the semantics of the conjunction "$\wedge$": it yields **true** if its subformulae yield **true**. The base cases for the formulation above are u and e, whose semantics are represented in **Defs. 16** and **17** above.

3.4 Verification of Norms

We want to verify that the set of dialogues $\mathcal{D}_{PS}$ of a performative structure PS satisfies a set of norms $\mathcal{N}$, that is, $\mathcal{D}_{PS} \models \mathcal{N}$. One option is to verify that there exists at least one dialogue in $\mathcal{D}_{PS}$ such that

- All integrity norms are satisfied, that is, the performative structure does not contain situations in which a violation occurs.
- There are no pending obligations, that is, all acquired obligations (right-hand side of an obligation) are fulfilled.

We notice that the verification of norms in e-institutions as formalised in this work amounts to a restricted kind of first-order theorem proving. The restriction however does not affect the complexity of the task and attempts to automate it are limited by the semi-decidability of first-order theorem proving [9, 8]. Notwithstanding this theoretical result, we can adopt some practical simplifications to make the verification decidable: if we assume the sets from our ontology are all finite, then the verification process amounts to theorem proving with propositional logics, which is decidable. Our previous formalisation naturally accommodates the proposed simplification: an initial substitution $\sigma_0 =$

$\{\mathsf{Set}_1/\{c^1_1,\ldots,c^1_{n_1}\},\ldots,\mathsf{Set}_m/\{c^m_1,\ldots,c^m_{n_m}\}\}$ formally represents all sets from our ontology – it is essential to our approach that all sets be finite collections of constants. We can define our finite verification of a norm N wrt the set of dialogues $\mathcal{D}$ of a performative structure as $\mathbf{K}(N,\mathcal{D},\sigma_0\cup\sigma)=\mathbf{true}$, that is, we would like to obtain a substitution σ which (added to the initial σ_0) makes N hold in $\mathcal{D}$. Since the value of all variables should ultimately come from a set Set_i in our ontology and given that all these sets are finite and part of the initial substitution σ_0, then we can obtain σ that assign values to each illocution scheme variables – provided there are such values that satisfy the boolean expressions in N. We can extend this definition to cover sets of norms $\mathcal{N}=\{N_1,\ldots,N_p\}$: $\mathbf{K}(\mathcal{N},\mathcal{D},\sigma_0\cup\sigma)=\mathbf{true}$ iff $\mathbf{K}(N_1,\mathcal{D},\sigma_0\cup\sigma)=\cdots=\mathbf{K}(N_p,\mathcal{D},\sigma_0\cup\sigma)=\mathbf{true}$.

The substitution $\sigma_0\cup\sigma$ functions as a *model*: by using its values for the illocution scheme variables, we can construct a subset $\mathcal{D}'_{PS}\subseteq\mathcal{D}_{PS}$, $|\mathcal{D}'_{PS}|=1$ (*i.e.* exactly one dialogue scheme), such that $\mathcal{D}'_{PS}\models\mathcal{N}$. The only dialogue scheme in $\mathcal{D}'_{PS}$ consists of a single path through the scenes of the performative structure. This dialogue, together with its substitution *sigma* provides a norm-compliant execution for the institution.

The complexity of our verification is an exponential function on the number of values for each variable, in the worst case. The verification works by choosing a value for the illocution scheme variables from the appropriate sets and then checking the boolean expressions which might relate and constrain such values. A similar technique has been employed in [10] to obtain models for the enactment of e-institutions. This process can be made more efficient by using standard techniques from constraint logic programming [11].

We can now define means to obtain models $\mathcal{D}'_{PS}\subseteq\mathcal{D}_{PS}$ for a given performative structure and set of norms. We employ the relationships **D** and **K**:

$$\mathbf{model}(PS,\mathcal{N},\mathcal{D}'_{PS})\leftarrow\mathcal{D}'_{PS}\subseteq\mathcal{D}_{PS}\ \wedge\ |\mathcal{D}'_{PS}|=1\ \wedge\ \mathbf{K}(\mathcal{N},\mathcal{D}'_{PS},\sigma)$$

That is, we obtain individual dialogue schemes from $\mathcal{D}_{PS}$ (one at a time) and then check via **K** if it satisfies the set of norms $\mathcal{N}$.

The *correctness* of our definitions can be formulated as: if $\mathbf{model}(PS,\mathcal{N},\mathcal{D}'_{PS})$ then $\mathcal{D}'_{PS}\models\mathcal{N}$. Whereas the *completeness* can be stated as: if $\mathcal{D}'_{PS}\models\mathcal{N}$ then $\mathbf{model}(PS,\mathcal{N},\mathcal{D}'_{PS})$. It is important to notice that if σ_0 contains only finite sets, then our **model** relationship is correct and complete, and its complexity is an exponential function on the number of illocution scheme variables and their possible values.

4 Conclusions, Related Work and Directions of Research

We have presented a formal definition of norms and shown how norms can be employed to verify electronic institutions. We provide a computational approach to assess whether an electronic institution is *normatively consistent*, that is, whether there is at least one possible enactment of it (by heterogeneous agents) in which norms will not be subverted. Given an electronic institution we can also determine whether its norms prevent any norm-compliant executions from happening.

Electronic institutions provide an ideal scenario within which alternative definitions and formalisations of norms can be proposed and studied. In [12] we find an early account of norms relating illocutions of an e-institution. In [13] we find a first-order logic formulation of norms for e-institutions: an institution conforms

to a set of norms if it is a logical model for them. Our work is an adaptation and extension of [12] but our approach differs in that we do not explicitly employ any deontic notions of obligations [1]. Our norms are of the form $Pre \rightarrow Obls$, that is, if *Pre* holds then *Obls* ought to hold. The components of *Pre* and *Obls* are utterances, that is, messages the agents participating in the e-institution send. This more pragmatic definition fits in naturally with the view of e-institutions as a specification of virtual environments which can be checked for properties and then used for synthesising agents [14, 15].

We are currently investigating means to automatically verify our norms, building dialogue schemes and sets of substitutions that can be used to restrict the behaviour of agents taking part in the enactment of an institution.

References

1. Dignum, F.: Autonomous Agents with Norms. Artificial Intelligence and Law **7** (1999) 69–79
2. López y López, F., Luck, M., d'Inverno, M.: Constraining Autonomy Through Norms. In: Proceedings of the 1st Int'l Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS), ACM Press (2002)
3. Verhagen, H.: Norm Autonomous Agents. PhD thesis, Stockholm University (2000)
4. Esteva, M., Rodríguez-Aguilar, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the Formal Specification of Electronic Institutions. Volume 1991 of LNAI. Springer-Verlag (2001)
5. Rodríguez-Aguilar, J.A.: On the Design and Construction of Agent-mediated Electronic Institutions. PhD thesis, IIIA-CSIC, Spain (2001)
6. Esteva, M.: Electronic Institutions: from specification to development. PhD thesis, Universitat Politècnica de Catalunya (UPC) (2003) IIIA monography Vol. 19.
7. Apt, K.R.: From Logic Programming to Prolog. Prentice-Hall, U.K. (1997)
8. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer-Verlag, New York, U.S.A. (1990)
9. Enderton, H.B.: A Mathematical Introduction to Logic. 2nd edn. Harcourt/Academic Press, Mass., USA (2001)
10. Vasconcelos, W.W.: Expressive Global Protocols via Logic-Based Electronic Institutions. In: Proc. 2nd Int'l Joint Conf. on Autonomous Agents & Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, ACM, U.S.A (2003)
11. Jaffar, J., Maher, M.J.: Constraint Logic Programming: A Survey. Journal of Logic Programming **19/20** (1994) 503–581
12. Esteva, M., Padget, J., Sierra, C.: Formalizing a Language for Institutions and Norms. Volume 2333 of LNAI. Springer-Verlag (2001)
13. Ibrahim, I.K., Kotsis, G., Schwinger, W.: Mapping Abstractions of Norms in Electronic Institutions. In: 12th. Int'l Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'03), Linz, Austria, IEEE Computer Society (2003)
14. Vasconcelos, W.W., Robertson, D., Sierra, C., Esteva, M., Sabater, J., Wooldridge, M.: Rapid Prototyping of Large Multi-Agent Systems through Logic Programming. Annals of Mathematics and A.I. (2004) Special Issue on Logic-Based Agent Implementation, to appear.
15. Vasconcelos, W.W., Sierra, C., Esteva, M.: An Approach to Rapid Prototyping of Large Multi-Agent Systems. In: Proc. 17th IEEE Int'l Conf. on Automated Software Engineering (ASE 2002), Edinburgh, UK, IEEE Computer Society, U.S.A (2002) 13–22