On the Formal Specification of Electronic Institutions

Marc Esteva, Juan A. Rodríguez-Aguilar,
Carles Sierra, Pere Garcia, Josep L. Arcos

Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain.
{marc,jar,sierra,pere,arcos}@iiia.csic.es
http://www.iiia.csic.es

Abstract. In this article we argue that open agent organisations can be effectively designed and implemented as institutionalized electronic organisations (*electronic institutions*) composed of a vast amount of heterogeneous (human and software) agents playing different roles and interacting by means of speech acts. Here we take the view that the design and development of electronic institutions must be guided by a principled methodology. Along this direction, we advocate for the presence of an underlying formal method that underpins the use of structured design techniques and formal analysis, facilitating development, composition and reuse. For this purpose we propose a specification formalism for electronic institutions that founds their design, analysis and development.

1 Introduction

According to [20], human interactions are guided by institutions, which provide a structure to everyday life. Institutions represent the rules of the game in a society, including any (formal or informal) form of constraint that human beings devise to shape human interaction. Therefore, they are the framework within which human interaction takes place, defining what individuals are forbidden and permitted and under what conditions. Institutions must be created (f.i. the Constitution), and may evolve over time (f.i. the common law). A distinguishing feature of institutions is the clear distinction drawn between the rules and the players. Establishing a stable structure to human interaction appears as the *raison d'être* of institutions.

At this point it is convenient to refer to the crucial distinction between institutions and organisations. Following the rational view of organisations provided by [10] "organisations are social units (or human groupings) deliberately constructed and reconstructed to seek specific goals". Organisations include political, economic, social and educational bodies (f.i. political parties, trade unions, clubs, universities). In [24] *goal specificity* and *formalization* are identified as the two main characteristics of organisations, meaning by formalization the attempt to standardize and regulate the behavior of the roles interacting within an organization. Therefore roles are understood as standardized patterns of behavior required of all agents playing a part in a given functional relationship in the context of an organization.

The way organisations are created and how they evolve are fundamentally influenced by the institutional framework, and, in turn they influence how the institutional framework evolves. Organisations must conform to the rules of institutions in order to receive legitimacy and support.

In this paper we adopt as a main goal the design and development of efficient and robust open agent organisations. At the outset we wonder whether we can follow a mimetic strategy that permits us to borrow and adapt well-know social mechanisms and concepts that have proven valuable when employed for articulating human societies. For this purpose, we firstly make explicit the major issues arising in our principal task taking inspiration on the analysis already presented in [8].

- *Heterogeneity.* The society must be capable of coping with and supporting heterogeneous agents, i.e. agents developed in different languages, with different purposes (objectives) and preferences, and endowed with varying degrees of sophistication.
- *Trust and Accountability.* Agents are not always expected to reliably follow the rules of the society due to multiple reasons (bugs, bounded rationality, etc.), and therefore the society must be equipped with the appropriate mechanisms for monitoring and conveniently managing malicious, fraudulent agent behavior.
- *Exception Handling.* We must contemplate the capability of detecting, preventing, and recovering from failures resulting from unintended disfunctional behaviors that may jeopardize the overall functioning of the society.
- *Societal change.* Agent societies are not static; they may evolve over time by altering, eliminating or incorporating rules. Hence the need of demanding flexible agent societies capable of accommodating future changes.

We observe that indeed human societies successfully deal with the issues above by deploying institutions that:

- determine what individuals are prohibited from doing and what are permitted to undertake;
- costlinessly ascertain violations and the severity of the punishment to be enacted; and
- evolve and are altered by human beings.

We uphold that agent organisations can be effectively designed and implemented as *institutionalized agent organisations* —that henceforth we shall term *electronic institutions*, or *e-institutions* for shorter. From this follows that one of our main purposes will be to model the creation, evolution, and consequences of the rules defining institutions and their incorporation into agent organisations.

Notice that the kind of electronic institutions that we will be considering are populated by a vast amount of heterogeneous (human and software) agents playing different roles and interacting by means of speech acts [25]. In other words, we make the assumption that any interacting activity taking place within our electronic institutions is purely dialogic.

Throughout this paper, we focus on the macro-level (societal) aspects referring to the infrastructure of electronic institutions, instead of the micro-level (internal) aspects of agents. Such a task is widely admitted by the multi-agent community as highly delicate [18]. This fact makes us advocate for adopting a principled, engineering approach founded on a formal specification of electronic institutions, continuing the work presented in [22,26], that founds their design, analysis and development of architecturally-neutral electronic institutions. As the most salient result, we obtain a graphical specification language that is intended to be embedded into a development environment that assists e-institutions' designers at all the stages of the development cycle.

The rest of the paper is organized as follows. In Section 2 we argue on the need of a formal method that underpins the specification, analysis and development of electronic institutions. Next, in Section 3 we provide a mathematically sound, unambiguous, abstract definition of electronic institutions. In order to illustrate how to specify in practice electronic institutions, we offer in Section 4 an example corresponding to the fish market an electronic auction house. In Section 5 we summarize and highlight the major benefits arising from our contribution, and discuss the open issues that can be faced taking our proposal as the starting point.

2 A Formal Specification Approach

Formal methods in software building comprise two main activities, namely *formal specification* and *verified design*, that is the precise specification of the behaviour of a piece of software, so that such software can be subsequently produced, and the proof of whether the implementation complies with the specification. Thus, the role of any formal method is to provide a clear and precise description of *what* a system is supposed to do, rather than a formulation of *how* it operates [9]. The presence of an underlying formal model will support the use of structured design techniques and formal analysis, facilitating development, composition and reuse.

In this section we take such formal approach with the purpose of specifying software infrastructures for electronic institutions. Our formal specification will be based on a purposely devised specification language enabled to produce both visual and textual specifications of infrastructures for electronic institutions. In other words, such specification language shall serve to produce sound an unambiguous specifications of the *rules* of the game of an electronic institution. It is apparent that at a further stage the specification language can be extended in order to allow for the specification of the *players* of the game. Such extension is expected to support the specification of micro (internal) aspects of agents, i.e. their architectures and logics. As a result, it would be possible the complete specification (infrastructure and agents) of an electronic institution.

In this work we solely focus on the specification of infrastructures. As a first step, we identify the core notions on which our current conception of electronic institution, and so our specification language, is to be founded:

- *Agents and Roles.* Agents are the players in an electronic institution, interacting by the exchange of speech acts, whereas roles are defined as standardized patterns of behaviour. The identification and regulation of roles is considered as part of the formalization process of any organisation [24]. Any agent within an electronic institution is required to adopt some role(s). A major advantage of using roles is that they can be updated without having to update the actions for every agent on an individual basis. Recently, the concept of role is becoming increasingly studied by software engineering researchers [21,16], and even more recently by researchers in the agents' community [4,17,28,6,5].
- *Dialogic framework.* Some aspects of an institution such as the objects of the world and the language employed for communicating are fixed, constituting the context or framework of interaction amongst agents. In a dialogic institution, agents interact trough speech acts. Institutions establish the acceptable speech acts by defining the ontology (vocabulary) —the common language to represent the "world"— and the common language for communication and knowledge representation which are bundled in what we call dialogic framework. By sharing a dialogic framework, we enable heterogeneous agents to exchange knowledge with other agents.
- *Scene.* Interactions between agents are articulated through agent group meetings, which we call *scenes*, with a well-defined communication protocol. We consider the protocol of a scene to be the possible dialogues agents may have.
- *Performative structure.* Scenes can be connected, composing a network of scenes, the so-called performative structure, which captures the existing relationships among scenes. The specification of a peformative structure contains a description of how the different roles can legally move from scene to scene. Agents within a performative structure may be possibly participating in different scenes, with different roles, and at the same time.
- *Normative Rules.* Agent actions in the context of an institution may have consequences that either limit or enlarge its subsequent acting possibilities. Such consequences will impose obligations to the agents and affect its possible paths within the performative structure.

Next, in Section 3 we offer a detailed analysis of the notions introduced above that helps us gradually construct a formal specification of an electronic institution infrastructure.

3 Electronic Institutions' Structure

First of all, and for notational purposes, we introduce some definitions that will apply henceforth. Let $Agents = \{a_1, \ldots, a_n\}$ be a finite set of agent identifiers, and $Roles = \{r_1, \ldots, r_m\}$ a finite set of role identifiers respectively. At the specification level we regard agents and roles simply as symbols and type names. Then by $a_i : r_j$ we mean that agent a_i is of type r_j. Moreover, we define the sets $V_A = \{x_1, \ldots, x_A\}$ and $V_R = \{\rho_1, \ldots, \rho_R\}$ as a finite set of agent variables and a finite set of role variables respectively.

3.1 Roles

We have already identified the notion of role as central in the specification of electronic institutions. Roles allow us to abstract from the individuals, the agents, that get involved in an institution's activities. In order to take part in an electronic institution, an agent is obliged to adopt some role(s). Thereafter an agent playing a given role must conform to the pattern of behaviour attached to that particular role. Therefore, all agents adopting a very same role are guaranteed to have the same rights, duties and opportunities. Notice that we can think of roles as agent types. More precisely, we define a role as a finite set of dialogic actions. Such actions are intended to represent the capabilities of the role. For instance, an agent playing the buyer role is capable of submitting bids and an agent playing the auctioneer role can offer goods at auction.

There are several issues concerning role/role relationships. As an agent may possibly play several roles at the same time, role/role associations standing for conflict of interests must be defined with the purpose of protecting the institution against an agent's malicious behaviour.

Thus, in general, the management of agent/role and role/role relationships becomes a fundamental issue for electronic institutions. *Role-based Access Control Models* (RBAC) [23] developed in the computer security arena offer well-founded mechanisms for handling both types of relationships. Fundamentally, RBAC has been successfully employed for managing the security administration of organisations [3,2]. But more interestingly, RBAC has been accurately formalized [12,14,13]. In order to solve the agent/role and role/role relationships, we borrow and adapt the formalisation of the rules for RBAC offered by the NIST RBAC model [14,13,12], which extends the basic RBAC model by adding role hierarchies, cardinality, and conflict of interests relationships.

Although RBAC manages three types of associations, namely associations between users (agents) and roles, associations between roles, and associations between roles and permissions[12], in this section we solely concentrate on the role/role associations.

When analysing human institutions we observe that the roles played by humans are hierarchically organised. There are many ways of constructing of a role hierarchy to express different types of relationships among roles. Here we consider that roles are organised as a partially ordered set (poset), represented as a pair $\mathcal{R} = \langle Roles, \succeq \rangle$, reflecting a role hierarchy. Then if $r \succeq r'$ holds we say that r subsumes r' or an agent playing role r is also enabled to play role r'.

We also realise that there are conflicting roles within an institution. For instance, in the fish market the *boss* and *buyer* roles are mutually exclusive in the sense that no one can *ever* act as the boss of the market and as a buyer. In a bank, the *teller* and *auditor* roles are also considered as mutually exclusive. We define a policy of static separation of duty (ssd) to mean that roles specified as mutually exclusive cannot be both authorised to an agent.

We represent the static separation of duties as the relation $ssd \subseteq Roles \times Roles$. A pair $(r, r') \in ssd$ denotes that r, r' cannot be authorised to the very same agent. Observe that the static separation of duties will be considered when assigning roles to an agent entering the institution and it restricts the possible assignations of roles to an agent.

Summarising, the adoption of a policy of separation of duties will allow us to express constraints on the role/role associations indicating conflicts of interests. It seems obvious that ssd must satisfy some constraints respect to the relation $\succeq$.

Finally, we explicitly state the requirements identified so far as necessary for managing the role/role relationships:

(i) The static separation of duties relation is symmetric. Formally, for all $r_i, r_j \in Roles \quad (r_i, r_j) \in ssd \Rightarrow (r_j, r_i) \in ssd$.
(ii) If a role subsumes another role, and that role is in static separation of duties with a third role, then the first role is in static separation of duties with the third one. Formally, for all $r, r_i, r_j \in Roles, r \succeq r_i, (r_i, r_j) \in ssd \Rightarrow (r, r_j) \in ssd$.
(iii) If a role subsumes another role, then the two roles cannot be in static separation of duties. Formally, $r_i \succeq r_j \Rightarrow (r_i, r_j) \notin ssd$.

Observe that from the last requirements the following properties can be inferred:

(i) For all $r \in Roles \Rightarrow (r, r) \notin ssd$ since the $\succeq$ relation is reflexive.
(ii) If $(r_i, r_j) \in ssd$ then for all $r \succeq r_i$ and for all $s \succeq r_j \Rightarrow (r, s) \in ssd$.
(iii) If exists r such that $r \succeq r_i$ and $r \succeq r_j \Rightarrow (r_i, r_j) \notin ssd$.

3.2 Dialogic Framework

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 successfully 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* [19], composed of a communication language, a representation language for domain content and an ontology. By sharing a dialogic framework, we enable heterogeneous agents to exchange knowledge with other agents.

Definition 1. We define a *dialogic framework* as a tuple $DF = \langle O, L, I, CL, Time \rangle$ where

- O stands for an ontology (vocabulary);
- L stands for a representation language for domain content;

– I is the set of illocutionary particles;
– CL is the (agent) communication language;
– $Time$ is a discrete and partially ordered set of instants.

Within a dialogic framework the representation language (KIF [15], first-order logic, etc.) allows for the encoding of the knowledge to be exchanged among agents using the vocabulary offered by the O ontology. The propositions built with the aid of L, the “inner” language, are embedded into an “outer language”, CL, which expresses the intentions of the utterance by means of the illocutionary particles in I. We take this approach in accord to speech act theory [25], which postulates that utterances are not simply propositions that are true or false, but attempts on the part of the speaker that succeed or fail.

We consider that CL *expressions* are constructed as formula of the type $\iota(\alpha_i : \rho_i, \alpha_j : \rho_j, \varphi, \tau)$ where $\iota \in I, \alpha_i$ and α_j are terms which can be either agent variables or agent identifiers, ρ_i and ρ_j are terms which can be either role variables or role identifiers, $\varphi \in L$ and τ is a term which can be either a time variable or a value in $Time$. We say that a CL expression is an *illocution* when α_i, α_j are agent identifiers, ρ_i and ρ_j are role identifiers and τ is a value in $Time$ time-stamping the illocution. We say that a CL expression is an *illocution scheme* when some of the terms corresponds to a variable. This distinction will be valuable when specifying scenes in the following section.

During the execution of a scene, variables in CL expressions will be bound to concrete values. For the purpose of allowing to change the value of a bound variable, we will denote it differently. By $\tilde{t}$ we denote that once bound t it can be assigned new values. In this way, we provide a way of representing that the value of a variable can be overwritten.

Henceforth we shall employ question marks to differentiate variables in CL expressions, including the expression corresponding to the contents, from other terms.

Next we present examples of CL expressions:

– $request(?x_i : pra, ?x_j : pra, appointment(?t), ?t')$ is an illocution scheme.
– $request(KQLAT : pra, marc : pra, appointment(tomorrow), 5)$ is an illocution that instantiates the schema above. We interpret this CL expression as a request sent by agent $KQLAT$ playing the pra (personal representative assistant) role to agent $marc$ playing the same pra role for arranging an appointment to meet *tomorrow*.

Finally, we would like to stress the importance of the dialogic framework as the component containing the ontologic elements on the basis of which any agent interaction can be specified as illustrated next when introducing the notion of scene. Thus, a dialogic framework must be regarded as a necessary ingredient to specify scenes.

3.3 Scene

Before formally defining a scene, we must precisely understand what a scene is. Recall that the whole activity within an electronic institution was described above as a composition of multiple, well-separated, and possibly concurrent, dialogic activities, each one involving different groups of agents playing different roles. For each activity, interactions between agents are articulated through agent group meetings, which we call *scenes*, that follow well-defined communication protocols. In fact, no agent interaction within an institution takes place out of the context of a scene. We consider the protocol of each scene to model the possible dialogic interactions between roles instead of agents. In other words, scene protocols are patterns of multi-role conversation.

A scene protocol is specified by a graph where the nodes of the graph represent the different states of the conversation and the arcs connecting the nodes are labelled with illocution schemes that make scene state change. The graph has a single initial state (non-reachable once left) and a set of final states representing the different endings of the conversation. There is no arc connecting a final state to some other state.

Normally the correct evolution of a conversation protocol requires a certain number of agents for each role involved in it. Then a minimum and maximum number of agents for role is defined and the number of agents playing each role has to be always between them.

Because we aim at modelling multi-agent conversations whose set of participants may dynamically vary, scenes will allow that agents either join in or leave at some particular moments during an ongoing conversation. For this purpose, we differentiate for each role the sets of access and exit states. The incorporation or exit of agents has to satisfy the restriction mentioned above about the number of agents for role. Obviously, the final states have to be an exit state for each role, in order to allow all the agents to leave when the scene is finished. On the other hand, the initial state has to be an access state for the roles whose minimum is greater than zero, in order to start the scene.

Definition 2. Formally, a scene is a tuple [1]:

$$S = \langle R, DF, W, w_0, W_f, (WA_r)_{r\in R}, (WE_r)_{r\in R}, \Theta, \lambda, min, Max\rangle$$

where

- R is the set of roles of the scene;
- DF is a dialogic framework;
- W is a finite, non-empty set of scene states;
- $w_0 \in W$ is the initial state;
- $W_f \subseteq W$ is the non-empty set of final states;
- $(WA_r)_{r\in R} \subseteq W$ is a family of non-empty sets such that WA_r stands for the set of access states for the role $r \in R$;
- $(WE_r)_{r\in R} \subseteq W$ is a family of non-empty sets such that WE_r stands for the set of exit states for the role $r \in R$;
- $\Theta \subseteq W \times W$ is a set of directed edges;

[1] When we need to differentiate the elements of two scenes s and s' we will use a superindex s or s'.

- $\lambda : \Theta \longrightarrow CL$ is a labelling function;
- $min, Max : R \longrightarrow I\!N$ $min(r)$ and $Max(r)$ return respectively the minimum and maximum number of agents that must and can play the role $r \in R$;

Notice that not every illocution scheme is valid to label an arc. In general, a CL expression $\iota(\alpha_i : \rho_i, \alpha_j : \rho_j, \varphi, \tau)$ can label an arc if it satisfies:

- α_i and α_j are agent variables;
- ρ_i and ρ_j are either role variables or role identifiers in R_s; and
- τ is a time variable;

These variables will be bound to concrete values during the execution of the scene. For example, agent variables in an illocution scheme will be bound respectively to the identifier of the agent that has uttered the illocution and to the identifier of the agent who has received the illocution. Then at each moment the bindings of the variables will be the context of the scene execution.

Next, we describe an example of a scene extracted from the case of study presented in Section 4, that we subsequently employ to illustrate how designers can construct graphical specifications of scenes.

Case Study We have selected as example, the scene representing the main activity within the marketplace: the auctioning of goods in the auction room. This scene is governed by the auctioneer making use of the downward bidding protocol (DBP) that next we state explicitly:

[**Step 1**] The auctioneer chooses a good out of a lot of goods that is sorted according to the order in which sellers deliver their goods to the sellers' admitter.

[**Step 2**] With a chosen good, the auctioneer opens a *bidding round* by quoting offers downward from the good's starting price, previously fixed by the sellers' admitter, as long as these price quotations are above a *reserve price* set by the seller.

[**Step 3**] For each price called by the auctioneer, several situations might arise during the open round:

Multiple bids. Several buyers submit their bids at the current price. In this case, a collision comes about, the good is not sold to any buyer, and the auctioneer restarts the round at a higher price. Nevertheless, the auctioneer tracks whether a given number of successive collisions is reached, in order to avoid an infinite collision loop. This loop is broken by randomly selecting one buyer out of the set of colliding bidders.

One bid. Only one buyer submits a bid at the current price. The good is sold to this buyer if his credit can support his bid. Whenever there is an unsupported bid the round is restarted by the auctioneer at a higher price, the unsuccessful bidder is punished with a fine, and he is expelled out from the auction room unless such fine is paid off.

No bids. No buyer submits a bid at the current price. If the reserve price has not been reached yet, the auctioneer quotes a new price which is obtained by decreasing the current price according to a price step. If the reserve price is reached, the auctioneer declares the good *withdrawn* and closes the round.

[**Step 4**] The first three steps repeat until there are no more goods left.

Graphical Representation The purpose of the formal definition above is to give a a mathematically sound definition of scene. However in practice scenes will be graphically specified. For instance, Figure 1 depicts the graphical specifications of the scene presented above.

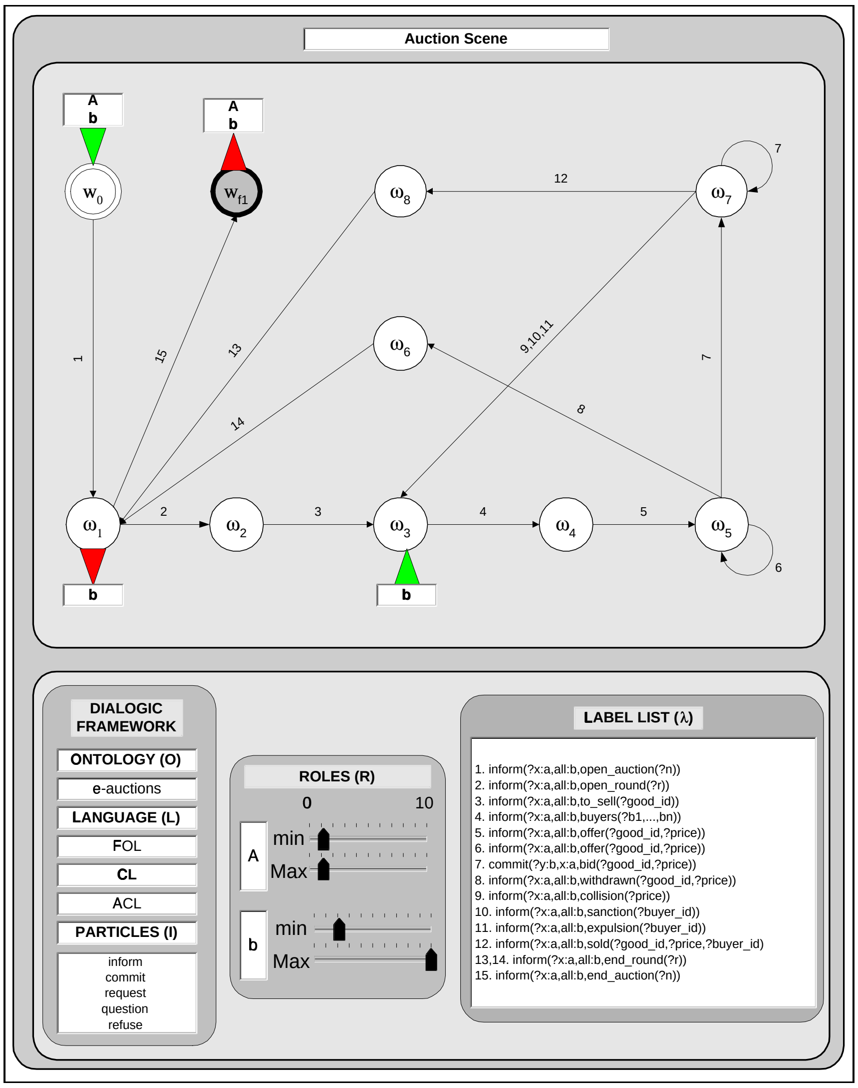
As to the *auction* scene (see Figure 1), we observe that the specification of the role set requires the participation of exactly one auctioneer(A) and, at least, two buyers(b)(min), although up to ten buyers might be allowed to participate(Max). The graph depicts the states of the scene, along with the edges representing the legal transitions between scene states, and labelled with illocution schemes of the communication language of the dialogic framework. The information contents of such schemes is expressed in first-order logic (*FOL*) making use of the concepts in the *e-auctions* ontology. The type of performative is specified in the set I — which contains the types of performatives identified in [7]— listed in the *Particles* area.

Notice that some states are identified as access and exit states. Apart from the initial and final states, the w_1 state is labelled as an access and exit state for buyers —meaning that after either a collision, sanction or expulsion, new buyers might be admitted into the scene— while ω_3 is uniquely an access state.

3.4 Performative Structure

The notion of performative structure is the most complex and interesting of this formalism, since it models the relationships among scenes. Notice that although conversations (scenes) are currently admitted as the unit of communication between agents, limited work has been done concerning the modelling of the relationships among different scenes. This issue arises when these conversations are embedded in a broader context, such as, for instance, organisations and institutions. If this is the case, it does make sense to capture the relationships among scenes. Thus, while a scene models a particular multi-agent dialogic activity, more complex activities can be specified by establishing relationships among scenes that allow:

- to capture *causal dependencies* among scenes (f.i. a patient cannot undergo an operation without being previously diagnosed by a doctor);
- to define *synchronisation* mechanisms involving scenes (f.i. within an exchange house, we might synchronise traders before allowing them to start off a negotiation scene);



1. inform(?x:a,all:b,open_auction(?n))
2. inform(?x:a,all:b,open_round(?r))
3. inform(?x:a,all:b,to_sell(?good_id))
4. inform(?x:a,all:b,buyers(?b1,...,bn))
5. inform(?x:a,all:b,offer(?good_id,?price))
6. inform(?x:a,all:b,offer(?good_id,?price))
7. commit(?y:b,x:a,bid(?good_id,?price))
8. inform(?x:a,all:b,withdrawn(?good_id,?price))
9. inform(?x:a,all:b,collision(?price))
10. inform(?x:a,all:b,sanction(?buyer_id))
11. inform(?x:a,all:b,expulsion(?buyer_id))
12. inform(?x:a,all:b,sold(?good_id,?price,?buyer_id)
13,14. inform(?x:a,all:b,end_round(?r))
15. inform(?x:a,all:b,end_auction(?n))

Fig. 1. Graphical Specification of an Auction Scene

– to establish *parallelism* mechanisms involving scenes (f.i. in an auction house, several auctions might be run at the same time);
– to define *choice points* that allow roles leaving a scene to choose their destination (f.i. an agent attending a conference is expected to opt for one of various, simultaneous talks);
– to establish the *role flow policy* among scenes, i.e. which paths can be followed by the roles leaving a scene and which target scenes they can reach. In a conference centre, a speaker, after finishing off his talk, is permitted

to leave the conference room and make it for another conference room to attend an ongoing talk. Notice that, for this particular case, the migration of this speaker also involves the adoption of another role.

In general, the activity represented by a performative structure can be depicted as a collection of multiple, concurrent scenes. Agents navigate from scene to scene constrained by the rules defining the relationships among scenes. Moreover, the very same agent can be possibly participating in multiple scenes at the same time. Hence, it is our purpose to propose a formal specification of performative structures expressive enough to facilitate the specification of such rules.

From a structural point of view, performative structures' specifications must be regarded as networks of scenes. At execution time, a performative structure becomes populated by agents that make it evolve whenever these comply with the rules encoded by the specification. Concretely, an agent participating in the execution of a performative structure devotes his time to jointly start new scene executions, to enter active scenes where the agent interacts with other agents, to leave active scenes to possibly enter other scenes, and finally to abandon the performative structure.

At this point we must notice that the way agents move from scene to scene depends on the type of relationship holding among the source and target scenes. As mentioned above, sometimes we might be interested in forcing agents to synchronise before jumping into either new or existing scene executions, or offer choice points so that an agent can decide which target scene to incorporate into, and so on. Summarising, in order to capture the type of relationships listed above we consider that any performative structure contains a special elements that we call *transition*, devoted to mediate different types of connections among scenes. Each scene may be connected to multiple transitions, and in turn each transition may be connected to multiple scenes. In both cases, the connection between a scene and a transition is made by means of a directed arc. Then we can refer to the source and target of each arc. And given either a scene or a transition, we shall distinguish between its incoming and outgoing arcs. Notice that there is no direct connection between two scenes, or, in other words, all connections between scenes are mediated by transitions. We do not allow also the connection of transitions.

Agents will be moving from a scene instance (execution) to another by traversing the transition connecting the scenes and following the arcs that connect transitions and scenes. Transitions must be regarded as a kind of routers that contain local information about the scene instances that they connect. Therefore, instead of modelling some activity, they are intended to route agents towards their destinations in different ways, depending on the type of transition.

The arcs connecting transitions to scenes play also a fundamental role. Notice that as there might be multiple (or perhaps none) scene executions of a target scene, it should be specified whether the agents following the arcs are allowed to start a new scene execution, whether they can choose a single or a subset of

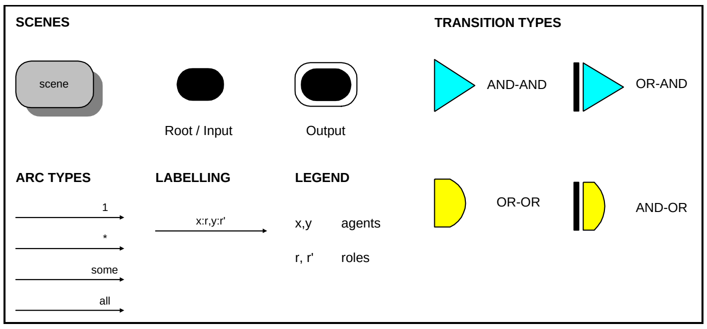


Fig. 2. Graphical Elements of a Performative Structure

scenes to incorporate into, or whether they must enter all the available scene executions.

We define a set of different types of transitions and arcs whose semantics will highly constrain the mobility of agents among the scene instances (the ongoing activities) of a performative structure. The differences between the diverse types of transitions that we consider are based on how they allow to progress the agents that they receive towards other scenes. Let us divide each transition into two parts: the *input*, through which it receives agents from the incoming arcs, and the *output*, through which agents leave following the outgoing arcs towards other scenes. Then, the following classification of transitions is based on the behaviour that they exhibit on their input and output sides:

– **And/And**: They establish synchronisation and parallelism points since agents are forced to synchronise at their input to subsequently follow the outgoing arcs in parallel.
– **Or/Or**: They behave in an asynchronous way at the input (agents are not required to wait for others in order to progress through), and as choice points at the output (agents are permitted to select which outgoing arc, which path, to follow when leaving).
– **And/Or**: They are a hybrid of the two types of transitions above: on the one hand, likewise *and/and* transitions, they force agents to synchronise on the input side, while on the other hand, likewise *or/or* transitions, they permit agents to individually make the choice of which path to follow when leaving.
– **Or/And**: They are also a hybrid of *and/and* and *or/or* transitions. Agents are not required to wait for others on the input side, but they are forced to follow all the possible outgoing arcs.

According to this classification, we define $\mathcal{T} = \{and/and, or/or, and/or, or/and\}$ as a set of transition types.

Depending on the path followed by the agents when traversing a transition, they may either start scenes or incorporate to one or more ongoing scenes. Thus there are also different types of paths, arcs, for reaching scenes after traversing transitions. We define $\mathcal{E} = \{1, some, all, *\}$ as the set of arc types. Following a *1-arc* constrains agents to enter a single scene instance of the target scene, whereas a *some-arc* is less restrictive and allows the agents to choose a subset of scene instances to enter, and an *all arc* forces the agents to enter all the scene instances to which the paths lead. Finally, a **- arc* fires the creation of a new scene instance of the target scene. Furthermore, each arc will be labelled with the collection of roles that are allowed to follow the arc.

Within the transitions agents are informed about current active scenes and request for their destination. Thus they constitute an intermediate state for agents that move from scene to scene. Transitions are employed by the institution for brokering purposes between agents and the active scenes connected to the transition. Thus within transitions agents will be informed about the active scenes that they can reach so that they choose their destination, i.e. either some active scene to join or some scene to be started. Each transition has an institutional agent that interact with arriving agents. The dialogue in transition can be specified as the dialogue in the scenes with the vision of the institutional agent and one arriving agent i.e. as a conversation with the institutional agent and another agent. When an agent arrives it is informed about its possibilities that depend on the type of the transition and the type of the arcs connecting the transition to the target scene(s). The difference between a transition and the rest of the scenes is that there is not a global state of the transition and any agent can be in a different state[2] . For example, new agents after arrive are in the initial state while other agents are in a middle state after been informed of their destination or waiting to synchronize with other agents or some agents can be in a final state waiting to move to their destinations. Agents requesting to move to a transition are always accepted and they are incorporated to the initial state. After they have selected their target scenes they wait in a final state until the target scene(s) arrives to an access state when.

Figure 2 depicts the graphical representations that we will employ to represent the performative structure's components introduced so far. From the point of view of the modeller, such graphical components are the pieces that serve to construct graphical specifications of performative structures.

Before stating a concrete definition of performative structure, there is a last element to be considered. Notice that although two scenes may be connected by a transition, the eventual migration of agents from a source scene instance to a target scene instance not only depends on the role of the agents but also on the results achieved by agents in the source scene. Thus, for instance, in the fish market, although a registration scene is connected to an auction scene, the access of a buying agent to the execution of the auction scene is forbidden if it has not successfully completed the registration process when going through a registration scene. In a conference centre environment, two agents are not allowed to meet to talk at a meeting room unless they have previously arranged and committed to some appointment. This fact motivates the introduction of

[2] Transitions can be seen as scenes where the state of the scene is the Cartesian product of the state of each agent

constraints over the arcs connecting scenes and transitions. We will require that agents satisfy the constraints, conditions, over the arc solicited to be followed when attempting to reach a destination scene. Therefore, conditions must also appear in our formal definition of performative structure.

Finally, we bundle all the elements introduced above to provide a formal definition of a performative structure specification:

Definition 3. Formally, given a meta-language ML, a performative structure is a tuple

$$PS = \langle S, T, s_0, s_\Omega, E, f_L, f_T, f_S, f_E, C, \mu \rangle$$

where

- S is a finite, non-empty set of scenes;
- T is a finite and non-empty set of transitions;
- $s_0 \in S$ is the *root* scene;
- $s_\Omega \in S$ is the *output* scene;
- $E = E^I \bigcup E^O$ is a set of arc identifiers where $E^I \subseteq (WE_S) \times T$ is a set of edges from exit states of scenes to transitions where $WE_S = \bigcup_{s \in S} \bigcup_{r \in R} WE_r^s$, and $E^O \subseteq T \times S$ is a set of edges from transitions to scenes;
- $f_L : E \longrightarrow 2^{V_A \times R}$ is the labelling function;
- $f_T : T \longrightarrow \mathcal{T}$ maps each transition to its type;
- $f_E : E \longrightarrow \mathcal{E}$ maps each arc to its type;
- $C : E \longrightarrow ML$ maps each arc to a meta-language expression of type boolean, i.e. a predicate, representing the arc's constraints;
- $\mu : S \longrightarrow I\!N$ sets the upper bound on the number of allowed simultaneous scenes for the scene scheme represented by each scene node.

In order to present the requirements to be satisfied by the elements above, some previous definitions are needed. Let $s \in S$ be a scene such that $s = \langle \mathcal{R}, DF, W, w_0, W_f, (WA_r)_{r \in R}, (WE_r)_{r \in R}, \Theta, L, min, Max \rangle$. Let var and $roles$ be two functions over arcs, defined respectively as $var : E \longrightarrow 2^{V_A}$ and $roles : E \longrightarrow 2^{Roles}$, which return for each arc the set of variables and the set of roles contained in the arc label. Then the following requirements must be satisfied by every performative structure:

(i) All the agent variables labelling the incoming arcs of a given transition must also appear on the outgoing arcs of the transition: for any $t \in T$ $\bigcup_{(s,t) \in E^I} var((s,t)) = \bigcup_{(t,s') \in E^O} var((t,s'))$

(ii) For every scene there is a path from the root transition to the output scene passing through the scene. For all $s \in S$ there is a path $s_0, t_1, \ldots, , s_m, t_m, s_\Omega$ such that $(s_i, t_i) \in E^I, (t_i, s_{i+1}) \in E^O \quad \forall i = 0, \ldots, m-1$ and $s_i = s$ for some i.

(iii) For every scene, the roles labelling its outgoing arcs must belong to its role set and for every role in its role set there is at least one outgoing arc labelled with it, i.e. for all $s \in S \quad \bigcup_{(s,t) \in E^I} roles((s,t)) = R^s$.

(iv) For every scene, the roles labelling its incoming arcs must belong to its role set and for every role of its role set there is at least one incoming arc labelled with it, i.e. for every scene $s \in S \quad \bigcup_{(t,s)\in E^O} roles((t,s)) = R^s$.

(v) For every scene, there must be at least one access state for every role labelling an incoming arc, i.e. for every arc $(t,s) \in E^O$, and for every role $r \in roles((t,s))$ there is an access state $w \in WA_r^s$ where WA_r^s is the set of access states for the role r in the scene s.

(vi) For every *and/and* transition, every outgoing arc has to be connected to a scene containing at least one access state for all the roles labelling the arc. More formally, for every (t,s) such that $f_T(t) = \mathit{and/and}$ then $\bigcap_{r\in roles((t,s))} WA_r \neq \emptyset$.

(vii) For every arc of type $*$ the initial state of the scene must belong to the set of access states for each role labelling the arc. Formally, for each arc $(t,s) \in E^O$ such that $f_E((t,s)) = *$ we require that $w_0 \in \bigcap_{r\in roles((t,s))} WA_r$.

(viii) Agents will not be allowed to progress through an arc of type $*$ if the number of instances of this scene has reached its maximum.

From the definition above, we can simply view a performative structure as a network of scenes interconnected by different types of transitions. The specification of a performative structure amounts to select a collection of scenes, and next create sound connections among them. Complementarity, the limits on the number of scenes, μ, need to be defined. Figure 3 shows how the graphical elements in Figure 2 are combined in order to produce the graphical specification of the performative structure corresponding to the fish market. Observe that the scene that Figure 1 represents, appears in the resulting specification as a particular node.

Notice that we demand any performative structure to contain a root and an output scene. The output scene does not model any activity, and so it must be regarded as the exit point of the performative structure. As to the root scene, it must be regarded as the starting point of any agent accessing the performative structure. Departing from the root scene, agents will make for other scenes within the performative structure.

3.5 Normative Rules

Such as depicted, a peformative structure constrains the behaviour of any participating agents at two levels:

(i) *intra-scene:* Scene protocols dictate for each agent role within a scene what can be said, by whom, to whom, and when.

(ii) *inter-scene:* The connections among the scenes of a performative structure define the possible paths that agents may follow depending on their roles. Furthermore, the constraints over output arcs impose additional limitations to the agents when attempting to reach a target scene.

And yet, we understand that an agent's actions within a scene may have consequences that either limit or enlarge its subsequent acting possibilities out of the scope of the scene.

Such consequences have effect along two different directions. On the one hand some actions will introduce subsequent acting commitments that have to be interpreted as acting obligations. On the other hand, other consequences occurring locally within a scene may vary the paths that an agent can follow in the performative structure because they affect the satisfaction and dissatisfaction of the constraints labelling paths. Both types of consequences will be required to be kept by an institution for each agent on an individual basis. In general, for a given agent we shall refer to such consequences as the agent *context* within the institution.

For instance, a trading agent winning a bidding round within an auction house is obliged to subsequently pay for the acquired good. Considering the performative structure in Figure 3 that implies that the trading agent has to move at some time to the buyers' settlements scene to pay for the acquired good. Notice that although the auction scene is connected to the output scene, the path is disallowed to agents unless they fulfil their pending payments.

In order to represent the deontic notion of obligation, we set out the predicate *obliged* as follows:

- $obliged(x, \psi, s)$ = agent x is obliged to do ψ in scene s.

where ϕ is taken to be, in both cases, an illocution scheme.

Next we introduce a special type of rules, the so-called *normative rules* in order to capture which agent actions (illocutions) have consequences that need to be kept in its context. Given a performative structure and a metalanguage, the normative rules will have the following schema:

$$(s_1, \gamma_1) \wedge \ldots \wedge (s_m \gamma_m) \wedge \phi_1 \ldots \phi_n \Rightarrow \phi_{n+1} \wedge \ldots \wedge \phi_r$$

where $(s_1, \gamma_1), \ldots, (s_m, \gamma_m)$ are pairs of illocution schemes and scenes, and $\phi_1 \wedge \ldots \wedge \phi_r$ are meta-language predicates. Notice that some of these rules will be devoted to the triggering of obligations, while others will be used for inferring facts that will be subsequently employed to determine the access of an agent to other scenes.

Let us consider the following example that will help us illustrate how normative rules are specified[3]:

$$\begin{aligned}&(auction\_room, commit(?x : a, ?y : b, sell(?good, ?y, ?price, ?round))) \Rightarrow \\ &obliged(?y, commit(?y : b, ?z : c, pay(?good, ?price, ?card), buyers\_settlements)\end{aligned} \quad (1)$$

If an auctioneer commits to sell the good at auction at a given buyer, this is obliged to commit to a buyers' accountant in a *settlements* scene to pay for the good.

[3] a, b, c stand respectively for the auctioneer, buyer and accountant roles.

Summarising the deployment of normative rules is motivated by the need of an institution to infer agents' obligations as well as the consequences of agents' local actions (within scenes) that have effect out of the scope of a scene.

3.6 Electronic Institution

Finally, we can define an electronic institution choosing a performative structure and defining the rest of its components. These are the institutional roles, the hierarchy between roles, the policy of duties and the normative rules. The institutional roles define a set of roles that can not be played by the external agents. They are like the workers in a human institution. In the case of the hierarchy of roles and the ssd we can to take into account that both are applied to the set of roles of the institution which are all the roles appearing in the different scenes of the performative structure.

Definition 4. An electronic institution is defined as a tuple

$$\langle \mathcal{PS}, IR, \succeq, ssd, N_{PS} \rangle$$

where

- PS stands for a performative structure;
- IR is a subset of roles representing the institutional roles;
- $\succeq$ stands for the hierarchy partial order over the roles;
- ssd is the set of static separation of duties between roles; and
- N_{PS} stands for a set of normative rules.

Notice that the specification of an electronic institution must be regarded as a compositional activity to be undertaken by the institution designer. We can consider that specifications of dialogic frameworks, scenes and peformative structures are to be naturally organised into specification libraries.

Once completed a specification, it must go through a validation process that checks its well-formedness. Ultimately, if such a specification proves to be correct, its equivalent textual representation must be generated in order to be manipulated by the agents intending to participate in the institution. Moreover, the generated textual representations can be also employed for brokering purposes.

In the light of the complexity of the whole process, it is apparent the need of tools that assist the institution designer through the specification, validation, and translation of an electronic institution into a machine-readable format so that it can be easily parsed by agents.

4 Practical Specifications

The purpose of this section is to practically illustrate how to specify electronic institutions. For this purpose we concentrate on the modelling of the fish market an electronic auction house, whose complete graphical specification is shown in Figure 3.

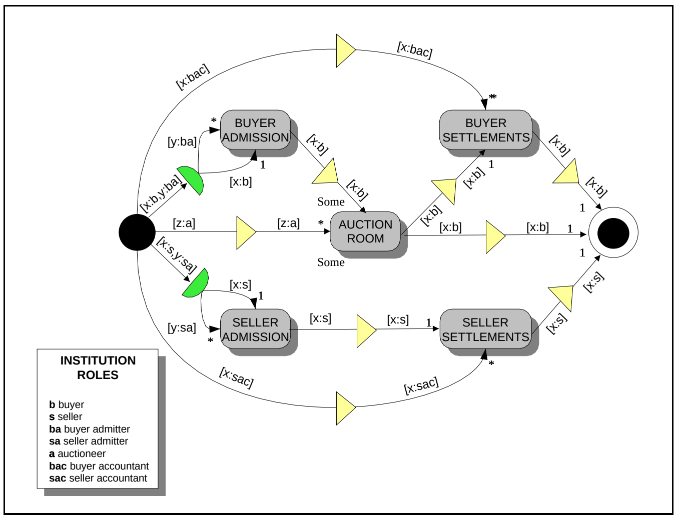


Fig. 3. Graphical Specification of the Fishmarket Performative Structure.

If we look at the example in figure 3 we can see that there are five scenes apart from the root and the output scene. We can observe the different paths that agents can follow depending on their role. Each scene is created by an institutional agent: buyer admitter, seller admitter, auctioneer, buyer accountant and seller accountant. This agents start one of the scenes and are in charge of it. They will dialogue with the external agents coming to the institutions.

We have two roles that external agents can play, the buyer and the seller roles. Both of them and also the institutional agents go for the root scene when entering the institutions. From the root scene we can see that buyers and sellers have different paths. On the one hand, buyers after entering in the root scene they can only go to the buyer admission scene. There they have to pass the admission process that is mediated by the buyer admitter who will ask the buyer for his login and password. If he is admitted, then he can go to the auction scene. As it can be some auction rooms at the same time, maybe each one auctioning a different kind of goods, he can choose a subset of them to go. For that reason the arc connecting the transition and the auction room is labelled of type *some*. When a buyer wants to leave the auction room it have two possibilities, to go to the output scene and leave the institution or to go to the buyer settlements. If he has won a round he must go to the buyer settlements to pay for the goods and the path going to the output scene is disabled for him. At the buyers settlements he has to pay for the goods corresponding to the round(s) that he has won and then he can leave the institution going to the output scene.

On the other hand, from the root scene sellers can only go to the seller admission scene. There they identify themselves with the sellers admitter and deliver their goods for being auctioned. From there they can go to the seller settlements scene where they receive the money for their goods after they have been auctioned at the auction scene. After receiving the money they can leave the institution moving from the seller settlements scene to the output scene.

5 Summary

In agree with [11], although organisational design is widely admitted as a fundamental issue in multi-agent systems, social concepts have been introduced in a rather informal way. Hence the need for formally incorporating organisational terms and concepts into multi-agent systems.

In this paper we have argued that agent societies can be effectively designed and constructed as electronic institutions. Then we have adopted a formal approach to specify electronic institutions. But why a formal technique? We believe that the following tip [1](page 215) answers the question:

> Use a formal technique when we cannot afford to have the requirements misunderstood.

An this is our case if we consider the high complexity of what we have identified as our main goal: the design and development of architecturally- neutral electronic institutions inhabited by heterogeneous (human and software) agents. In general, the presence of an underlying formal method underpins the use of structured design techniques and formal analysis, facilitating development, composition and reuse. Thus, we defend that successful methodologies must rely on unambiguous formal semantics, though developers are not aware of the existence of such a semantics as pointed out in [27].

Thus, we consider that the development of an electronic institution must be preceded by a precise specification that fully characterise the institution's rules. Some important advantages derive from the creation of specifications (models) of electronic institutions:

- An electronic institution model is a description of the modelled institution that can be used either as a specification or as a presentation. Such a model allows to investigate a new institution before being constructed. This possibility is particularly useful for institutions where design errors may jeopardise security or be expensive to correct.
- Graphical specifications are extremely easy to understand. They are similar to the informal diagrams employed by engineers and designers while designing, constructing and analysing a system.
- An electronic institution's specification offers an explicit description of both states and actions, in contrast to most description languages which describe either the states or the actions.
- The behaviour of an electronic institution model can be analysed, either by means of simulation or by means of more formal analysis methods.
- The process of creating the description and performing the analysis allows the modeller to gain a dramatically improved understanding of the modelled institution.

6 Acknowledgements

Marc Esteva enjoys the CIRIT doctoral scholarship 1999FI-00012. The research reported in this paper is supported by the ESPRIT LTR 25500-COMRIS *Co-Habited Mixed-Reality Information Spaces* project, the SLIE project IST-1999-10948 and the Spanish CICYT project SMASH, TIC96-1038-C04001.

References

1. *Software Requirements. Objects, Functions and States.* Prentice Hall International, Inc, 1993.
2. J. Barkley. Comparing simple role based access control models and access control lists. In *Proceedings of the Second ACM Workshop on Role-based Access Control*, 1997.
3. J. Barkley, A. V. Cincotta, D. F. Ferraiolo, S. Gavrilla, and D. R. Kuhn. Role based access control for the world wide web. In *20th National Information System Security Conference*, 1997.
4. M. Becht, T. Gurzki, J. Klarmann, and M. Muscholl. Rope: Role oriented programming environment for multiagent systems. In *Proceedings of the Fourth IFCIS Conference on Cooperative Information Systems (CoopIs'99)*, 1999.
5. O. Belakhdar and J. Ayel. Modelling approach and tool for designing protocols for automated negotiation in multi-agent systems. In W. van de Velde and J. W. Perram, editors, *Agents Breaking Away*, number 1038 in Lecture Notes in Artificial Intelligence, pages 100–115. Springer-Verlag, 1996.
6. R. J. A. Buhr, M. Elammari, T. Gray, and S. Mankowski. A high level visual notation for understanding and designing collaborative, adaptive behaviour in multi-agent system. In *Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS'98)*, 1998.
7. P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 65–72, Menlo Park, CA., jun 1995. AAAI Press.
8. C. Dellarocas and M. Klein. Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces. In *Proceedings ACM Conference on Electronic Commerce (EC-99)*, 1999.
9. A. Diller. *Z An Introduction to Formal Methods.* John Wiley & Sons, Inc, 1990.
10. A. Etzioni. *Modern Organizations.* Englewood Cliffs, NJ, Prentice-Hall, 1964.
11. J. Ferber and O. Gutknetch. A meta-model for the analysis of organizations in multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98)*, pages 128–135, 1998.
12. D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information Systems Security*, 1(2), 1999.
13. D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn. Role-based access control (rbac): Features and motivations. In *Annual Computer Security Applications Conference.* IEEE Computer Society Press, 1995.
14. S. I. Gavrila and J. F. Barkley. Formal specification for role based access control user/role and role/role relationship management. In *Proceedings of the Third ACM Workshop on Role-based Access Control*, 1998.

15. M. R. Genesereth and R. E. Fikes. Knowldege interchange format version 3.0 reference manual. Technical Report Report Logic–92–1, Logic Group, Computer Science Department, Standford University, June 1992.
16. I. Jacobson, M. Christerrson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering - A Use Case Driven Approach.* Addison-Wesley, 1996.
17. E. A. Kendall. Agent roles and role models: New abstractions for intelligent agent system analysis and design. In *Proceedings of Intelligent Agents for Information and Process Management (AIP'98)*, 1998.
18. V. R. Lesser. Reflections on the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, 1:89–111, 1998.
19. P. Noriega and C. Sierra. Towards layered dialogical agents. In *Third International Workshop on Agent Theories, Architectures, and Languages, ATAL-96*, 1996.
20. D. North. *Institutions, Institutional Change and Economics Perfomance.* Cambridge U. P., 1990.
21. D. Riehle and T. Gross. Role model based framework design and integration. In *Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'98)*, pages 117–133. ACM Press, 1998.
22. J. A. Rodríguez-Aguilar, F. J. Martín, P. Noriega, P. Garcia, and C. Sierra. Towards a formal specification of complex social structures in multi-agent systems. In J. A. Padget, editor, *Collaboration between Human and Artificial Societies*, volume 1624 of *Lecture Notes in Artificial Intelligence*, pages 284–300. Springer-Verlag, 2000.
23. R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role based access control models. *IEEE Computer*, 29(2), 1996.
24. W. R. Scott. *Organizations: Rational, Natural, and Open Systems.* Englewood Cliffs, NJ, Prentice Hall, 1992.
25. J. R. Searle. *Speech acts.* Cambridge U.P., 1969.
26. C. Sierra and P. Noriega. Institucions electrònics. In *Proceedings of the Primer Congrès Català d'Intel.ligència Artificial*, 1998.
27. M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
28. M. Wooldridge, N. R. Jennings, and D. Kinny. A methodology for agent-oriented analysis and design. In *Proceedings of the Third International Conference on Autonomous Agents (AGENTS'99)*, May 1999.