# Open Agent Systems ???

Frank Dignum[1], Virginia Dignum[1], John Thangarajah[2], Lin Padgham[2], Michael Winikoff[2]

[1] Dept. Information and Computing Sciences, Utrecht University
The Netherlands
{dignum,virginia}@cs.uu.nl

[2] RMIT University
Melbourne, Australia
{johthan,linpa,winikoff}@cs.rmit.edu.au

**Abstract.** E-institutions are envisioned as facilities on the Internet for heterogeneous software agents to perform their interactions and thus forming truly open agent systems. We investigate how these heterogeneous agents can determine whether an institution is one in which they can participate. We propose a layered approach which is supported through a (traditional) middle agent that is part of the environment. Starting with a basic compatibility of message types, each extra layer ensures a higher degree of compatibility, but requires also extra sophistication in both the information required and the matching algorithms. In this paper, we describe reasoning about how an agent should take on a specific role, message matching, and protocol compatibility. We explore the issues in the context of an actual accommodation agent built in JACK, and a travel agency institution built in ISLANDER.

## 1 Introduction

The notion of *"open agent systems"* is a popular and appealing one. However, there are many question marks associated with making such a vision a reality! Increasingly, the Internet is being viewed as an open interaction space where many heterogeneous agents co-exist. As in human societies, electronic interaction spaces must deal with issues inherent to open environments, namely heterogeneity of agents; trust and accountability; exception handling (detection, prevention and recovery from failures that may jeopardise the global operation of the system); and societal change (capability of accommodating structural changes). *Electronic Institutions* have been proposed as a way to implement interaction conventions for agents to regulate their interactions and establish commitments in an open environment [1]. In theory, such open environments have open admission policies, and therefore potentially fluid membership, and in such an environment, any agent could join a given institution. However, it is not clear how an agent should determine whether a particular institution is one that makes sense for it to join. Is the institution one in which it will be able to successfully achieve its goals? Can it successfully exchange the expected messages for a given role in the institution? Currently, in practice, agents are designed so as to be able to operate exclusively with a single given institution, thus basically defying the open nature of the institution.

We consider the situation in which many different institutions and agents coexist, and where the institutions and the agents pursue their own goals, but are dependent on each other to realize them. We take as an example the domain of travel, where institutions represent travel agencies and agents can be seen as representing individuals (or enterprises) seeking or offering certain services (trips, hotels, air tickets, ...). Institutions are developed by (possibly competing) entities with the view to offering an interaction space around a certain topic (flights, tourism in Victoria, eco-tourism, etc.) enforcing certain regulations on the interaction. The institutions may be more or less regulated allowing for more or less autonomy and openness to agents. An agent may want to join different institutions in order to fulfill its goals (e.g. either to compare prices, to get the most clients, or to create a total package from the different possibilities offered by different institutions).

We further assume that institutions and agents are developed independently from each other, hence structures are needed to support and guide the search and joining process. The greater the level of standardization, the simpler it will be to provide such support. The aim of this work is to give guidance on what information needs to be provided by agents and by institutions, in order to reason about compatibility. We aim to move away from the current situation where agents are hand-crafted to participate in a particular institutional space. Our aim is to move towards agents being able to determine at run time whether they can sensibly participate in a given institution. It may be clear that a "correct" decision about participation is very complex and involves reasoning about the semantics of what the agent and the institution try to achieve and their compatibility. In this paper we will not attempt to give a complete solution for this problem in one step, but rather sketch a way in which the solution to this decision problem can be approximated. This is done by involving more aspects of the problem in different stages and also adding more semantic matchmaking aspects to a basis of more syntactic matchmaking solutions.

While standards facilitate this kind of reasoning and support (and are indeed a kind of baseline for our approach), our aim is to keep the requirements on agents and institutions as minimal (and realistic) as possible. Where possible, we attempt to use existing (W3C) standards, or information that could be expected to fairly readily be made available. Although few real standards have been developed specifically for agents (see FIPA[3] for the state of the art), we can make use of standards such as WSDL[4] for message formats and OWL-S[5] descriptions for message contents, as well as other W3C standardized information such as WS-CDL[6] for protocol descriptions. In the case of institutions there are far fewer examples, and de-facto or decreed standards have not yet developed. ISLANDER is one of the few implemented tools for the specification and management of institutions [2]. It provides a combination of graphical and textual representations of the institution whilst the complete specification can also be obtained as an XML document. Currently it requires that agents are developed specifically to fit within a particular ISLANDER specified institution. However, we use it to explore

[3] http://www.fipa.org/

[4] http://www.w3.org/TR/wsdl

[5] http://www.w3.org/Submission/OWL-S

[6] http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/

the general issues involved in attempting to incorporate arbitrary agents within such an institution.

In the sections ahead we provide a concrete example of a simple ISLANDER institution representing a travel agency, and an accommodation agent previously developed for a different project. We explore how it might be determined whether the accommodation agent can participate in the Travel Institution. We provide an architecture that minimizes the requirements on both agents and institutions by introducing a third party service to reason about compatibility. We also identify the kind of information that agents and institutions will need to make available in order for this to work.

## 2 Application Scenario

In order to test the issues raised in the previous section, we use the domain of travel organizations as an application scenario. The motivations for this choice are twofold. On the one hand, it is a familiar domain, which does not require a lot of background description. Travel applications are widely available on the internet, and the idea of developing agents that can interact seamlessly with several existing applications is a realistic and useful one. On the other hand, we make use of an accommodation agent that was developed as part of a project to provide a testbed of travel and tourism agents, which is implemented using JACK[7]. This agent, which we will refer to as *JACK-acc-agent* in the remainder of this paper, when queried, returns a list of accommodation options. The service description is available in both WSDL and OWL-S and the ontology is defined in OWL-S. The fact that this agent has been developed independently of this project further serves our purpose of analyzing the complexity of interactions in heterogeneous environments.

Since most travel applications existing online are not based on MAS frameworks, we have developed a test institution using the ISLANDER platform [8]. ISLANDER is a graphical tool for the specification and verification of electronic institutions. The core notions of an ISLANDER electronic institution specification are [1]:

**Agents and Roles** Agents are the players in the institution, whereas roles are defined as standardized patterns of behavior. Agents within an electronic institution are required to adopt some role(s).

**Dialogic framework** Provides the context for interaction, in terms of domain concepts (ontology) and communication language (illocutions).

**Scene** Interactions between agents are articulated through agent group meetings, called scenes, with a well-defined communication protocol.

**Performative structure** Scenes can be connected, composing a network of scenes that captures the existing (ordering) relations among scenes. Further, it describes how roles can legally move from scene to scene.

**Normative Rules** Agent actions 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.

[7] JACK is an agent development platform available from Agent Oriented Software, http://www.agent-software.com.au

[8] http://e-institutor.iiia.csic.es/software.html

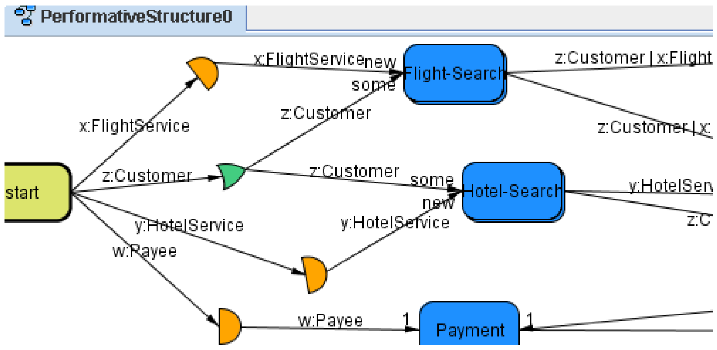

**Fig. 1.** Partial view of the travel agency in ISLANDER.

The travel agency institution facilitates the search, reservation, and purchase of flights and accommodation for potential customers. Some of the possible *roles* within this institution are customer, flight-service, hotel-service and payee (e.g. bank). Agents that enter this institution will take up one or more of these roles. Roles in ISLANDER can be either internal, played by agents within the organization, or external, which can be in principle played by any agent. Our focus in this paper is on these external roles.

The interactions between roles are captured within *Scenes*. For example the *hotel-search-scene* captures the interaction protocol between a customer and a hotel-service. In this scene the customer requests accommodation options from the hotel-service, who in return supplies a list of hotel options. The protocol is defined as a finite state machine. The other scenes of our travel institute include: a *flight-search-scene*, a *reservation-scene* and a *payment-scene*.

The relationships between scenes and the roles that enter and leave particular scenes are captured by the *Performative Structure* in ISLANDER (Figure 1). For example, a customer and a hotel-service may enter the hotel-search-scene and leave this scene with two options: exit the institution or move to the reservation-scene. The performative structure therefore constrains roles to particular scenes and also provides a view of the path an agent or service that enters an institution may take during its stay. As an agent moves between scenes it may well change its role. This is also specified as part of the Performative Structure. For example an agent playing the flight-service role in the flight search scene, may play a payee role in the payment scene. The performative structure specifies the permitted role changes as the agents move between scenes.

The *Ontology* defined in ISLANDER specifies the terms that can be used in the message patterns that can be used by the agents playing roles in a particular scene. The ontology is also used when defining constraints and norms.

As we envisage our travel institution to be an open system, then the accommodation agent *JACK-acc-agent* should be able to determine that it could join this travel institution taking up the role of a *hotel-service*, and participating in the *hotel-search* scene (as long as all conditions were met). Such a process requires two types of abilities. On the one hand, the technical ability of finding and using the API to connect to the institution application, and on the other hand the cognitive ability of determining the applicability of some institutional role to the realization of its goals. Some of the questions that such an agent will need to answer, in order to recognize this opportunity, are:

- What roles will the agent be able to take on in the organization? How does it move between roles and scenes?
- Can the accommodation agent send and receive the right message types for the organization? Will it be able to understand the content of the messages sent by the customer role and vice versa?
- Will it be able to follow the appropriate interaction protocols to conduct its business within the organization?

In the following section we describe an overall architecture to support the kind of open system described in this example. The details in further sections will address the above issues.

## 3 System architecture

In order to achieve our aim of having minimum requirements imposed on the agents and on the institutions, we propose an architecture where most of the reasoning about whether a particular agent is equipped to successfully participate in a particular institution, is located in an infrastructure support service, which we conceptualize as a middle agent. The term *middle-agent* has been used to describe various services such as *yellow page agents* [3], directory facilitators[9], brokers, and match-makers [4]. The LARKS system [5] provides a match making between agents based on service descriptions. Although this comes closer to what we propose than something like UDDI (Universal Description, Discovery and Integration) directories[10] it can better be seen as a potential part of our Middle Agent than as an alternative. In our situation we do not only match services, but also protocols and combination of services. To our knowledge, no solution for this type of matchmaking has been proposed yet. We will call this entity an *Institutions Middle Agent (IMA)*.

Similarly to the matchmaker middle agent, the IMA will maintain listings, and provide information regarding compatibility of institutions/agents with requests. Agents and institutions will register with the IMA, providing some information about their structure (such as WSDL descriptions, and other identified information). The IMA will then do the required reasoning to answer the questions identified in our example. An overview of the architecture is given in Figure 2.

It is important to note here that this figure represents the simplest IMA architecture, consisting of only one middle agent. In practice, more than one IMA may be necessary (e.g. to avoid bottlenecks) which will be interconnected. From the perspective of the agents and institutions using the IMA layer, this difference should be irrelevant. That is, whether one or more IMAs are present in the mediation layer, its basic overall functionality is as depicted in figure 2. The interaction of multiple IMAs is the subject of future research.

[9] http://www.fipa.org/specs/fipa00023/SC00023K.html
[10] http://www.uddi.org

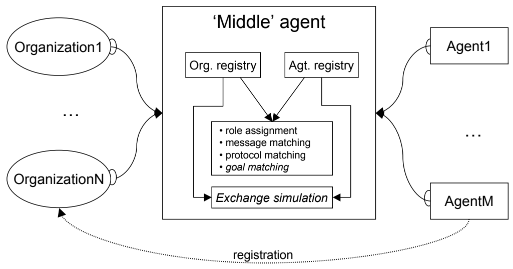

**Fig. 2.** Institutions Middle Agent (IMA) Architecture.

The functionalities of the IMA are divided over a number of levels of complexity. The lower levels require fewer assumptions of the required input and are meant to be used in situations where only simple standards exist and agents and institutions have little reflexive capabilities (i.e. cannot provide much information about the way they function). At the highest level more input is needed and the reasoning involves some very hard and complex issues, but also more guarantees can be given about the actual match between an agent and an institution.

*Assign agents to roles within an institution*. In order to do this the institution must register with the IMA some list of roles, and the messages and protocols associated with each role. On this most basic level the IMA checks whether the agent can play a role in the institution based on role names. This is explored further in section 4.

*Check for message compatibility.* On the next level of compatibility, the IMA will require some knowledge of the messages understood and expected by both the agent and the institution. Besides checking for message format compatibility, this also includes a kind of ontology matching on the content of the messages (which might in turn be outsourced to specialized ontology matchers). This is explored further in section 5.

*Check for protocol compatibility*. Even if the messages are compatible, if the expected pattern of message exchange is not compatible then successful interaction cannot be guaranteed. This is further discussed in section 6.

*Perform goal related reasoning*. On top of these basic compatibility issues the IMA can further check whether the goals of the agent and the institution are compatible. Is one institution better for the agent to achieve its goals than another? This level of reasoning could be supported if agents and institutions carried a protocol representation that included interaction goals. A possible way to achieve this is suggested by Cheong and Winikoff [6], but (due to space restrictions) will not be discussed further in this paper.

*Simulate a run of interactions*. The highest level of compatibility can be ensured through a simulated run of interactions. This allows to check not only whether the protocols are compatible but also whether they bring about the desired outcome. Space limitations do not allow us to elaborate on this issue.

## 4 Agents and roles

The concept of a *role* is used in institutions as a place holder for the agents that will enact it at runtime. From the perspective of system design, roles have the advantage that they abstract from the specific design and internal specification of individual agents. Roles in an institution identify the activities and services necessary to achieve social objectives and enable to abstract from the specific individuals that will eventually perform them [7]. From the perspective of institution design, roles are the building blocks for the agent architecture, providing placeholders for the actual agents. The concept is concerned with the specification of patterns of conduct, that is, expectations, identities, and social positions; and with context and social structure. From the agent design perspective, roles specify the expectations of the society with respect to the agent's activity in the society. Roles also define normative behavioral repertoires for agents [8]. That is, a role can be seen as the abstract representation of a policy, service or function, within the institution. In the current practice of MAS design, roles take different functions in different approaches to open agent systems. Some define roles as a class that defines a normative behavioral repertoire of an agent, others see roles as the representation of the 'expectations' of the institution for that position, while others take a more (linguistic) interaction perspective and use roles to represent the different participants in a conversation or protocol.

One of the first problems one encounters when considering the take up of institutional roles by external agents is that tools currently used to implement agents do not contain the concept of a role to represent the position (i.e. place holder) of the agent in the institution. Although agents might be designed to fill a role in the system, they are not aware of these roles at run-time! As such, messages used by agents in these systems always require specific agent IDs as the sender or recipient of the message and cannot be addressed to any agent fulfilling a role. A primary functionality for the IMA is thus to enable agents to take up a role in an institution. There are two activities necessary to achieve this aim:

– Check compatibility between agent and role.
– Provide operational support for agent activity as role enactor within the institution.

At the most basic level, a compatibility check can be done by comparing the semantic meaning of the agent's identifier with that of the role identifier. Such a check is obviously not enough but it gives an initial idea of possible compatibility. E.g. a *buyer* agent may probably not be able to perform a *hotel-service* role. Of course, some ontological knowledge is required for this function. At this stage we assume this service to be provided by an ontology mapping service. Another possibility checks which roles an agent may be able to play by looking at the messages that the agent can send and receive, and comparing them against the message interface of the roles. We will elaborate further on this in section 5.

In the specific case of ISLANDER, roles describe a specific position in one dialogic scene. That is, in ISLANDER an agent probably has to take up several roles as it moves into different scenes in the performative structure of an institution (e.g. the agent playing the role of customer in the *hotel-search-scene*, may later have to play the role of payer in the *payment* scene). From the agent's perspective it would make more sense to think of all these roles together as the customer role. This extends the desired functionality of the IMA to create one institutional role to be taken up by one agent to move from entrance to exit of the institution. For ISLANDER this role is composed of the union of scene roles that form a coherent path through the dialogical framework of the institution.

After the IMA has determined that an agent is compatible with a role in an institution, the next step is to help the agent enter into that institution. In ISLANDER execution of electronic institutions is regulated by AMELI [2]. AMELI provides *governor* agents which mediate all interaction with external agents, and thus are able to enforce compliance with the institution's protocols and performative structure. However, other proposals for institution modeling relax this constraint by defining participation negotiation protocols, in which an agent can negotiate its institutional activity to best fit its capabilities [9]. The range of negotiation is defined by the institution so as to limit it to parameters and values that do not endanger institutional norms and objectives.

In this paper, we will limit ourselves to determining the necessary preconditions for an agent to take up a role (in terms of messages and protocol fulfillment). However this still does not say anything about the sufficient preconditions for role enactment. That is, why should the agent take up this role, how does an agent choose which role to perform, or is this the optimal role (and the optimal institution) for the aims of the agent? E.g. should an agent take up a role of *accommodation-service* or of *trip-service*? While one might be more generic (and lead to more customers) the other might lead to better customers. The definition of sufficient preconditions for role enactment must therefore include the matching of the goals of the agent to those of the role. Given that we cannot assume any level of introspection from the agents, and in fact most existing agent platforms do not enable introspection, it is clear that this is a field that requires much more study.

## 5 Message Compatibility

In the following, we assume that both agents and institutions are able to provide a WSDL description of their message types. It must be noted, that this is not something that can be generally expected of all current agent platforms. A possible way to solve this, is to develop dedicated services that are able to transform a specific message type representation into a standard such as WSDL. Such services can be used by the IMA to convert message specifications.

Considering our example, messages used by the accommodation agent *JACK-acc-agent* are specified as follows:[11] (a # before a field name indicates that the field is optional)

```
<complex type       AccomodationInfo: ”accommodationContactInfo” type=string;
                    ”#Name” type=string; “#facilities” type=string; “#maxPrice” type=cost; “#minPrice” type=cost;
                    “#region” type=string; “#paymentMethods” type=string; ”address” type=string;/>
<message input:     “#Name” type=string; “#facilities” type=string; “#maxPrice” type=cost;
                    “#minPrice” type=cost; ”region” type=string />
<message output:    ”AccommodationResult” type=ListOfAccomodationInfo />
```

[11] The actual WSDL is too lengthy. This provides the relevant conceptual information, with modified syntax.

The following description of messages is extracted from the ISLANDER specification of the *hotel-search* scene:

```
<complex type   AccomodationInfo: ”accommodationContactInfo” type=string;
                “#Name” type=string; ”address” type=string; “#region” type=string; “#maxPrice” type=cost;
                “#minPrice” type=cost; “#paymentMethods” type=string;/>
<message        from role: customer to role: hotel-service “#Name” type=string; ”region” type=string;
                ”maxPrice” type=cost; ”minPrice” type=cost;/>
<message        from role: hotel-service to role: customer ”AccommodationResult” type=ListOfAccomodationInfo />
```

Given a WSDL style specification, such as that above, one of the functionalities of the IMA is the determination of message-level compatibility between the accommodation agent *JACK-acc-agent* and the institutional role of *hotel-service*. In open systems, exact matches are highly unlikely. In fact, the level of similarity in the example above, even though it is not an exact match, is much higher than what is likely to be found between compatible message sets in an open system. In our example, the lack of exact matching has to do with optional and mandatory fields, and with ordering of fields. Another important lack of matching is at the level of sender and receiver fields: while ISLANDER uses role names, the agent does not specify any receiver. As discussed in section 4, the IMA must keep track of the agent's role and the roles of the agents it interacts with, and possibly add this information to the messages being passed within the institution.

In order to realize that the accommodation agent *JACK-acc-agent* is compatible with the role of hotel-service, the IMA needs to determine (a) that all messages sent to the hotel-service role can be transformed into messages compatible with available input messages of the Accommodation Agent, and (b) that all output messages of the Accommodation Agent can be transformed into messages compatible with the messages that the hotel-service role can send.

Determining compatibility of one message specification with another requires ascertaining whether the sent message can be type-cast into what is expected by the receiver. In order to ensure that all instances of sent messages will be able to be transformed into something that is acceptable to the receiver we require that every mandatory field in the received message be compatible with a mandatory field in the sent message. Optional fields can be matched in the same way as mandatory ones, but if no match is found they can be ignored.

Where there is not an exact match between fields within a message we attempt to determine compatibility. Compatibility of compound fields requires compatibility of all their mandatory parts. Compatibility of (atomic) fields requires what we call *field name compatibility* and *value type compatibility*. Two atomic fields are *value type compatible* if the type of the field in the sent message is a subset of the type of that field in the received message. For example, if the sent message contains an integer, but a floating point number is expected to be received.

However, value type compatibility alone is not likely to yield semantic compatibility. Typically the name of a field carries semantic value. We refer to this level of compatibility as *field name compatibility*. For analysis of the semantic compatibility of two field names we require an ontology, and the ability to reason over that ontology. The ontology could be provided by the agent, by the institution, or could be an external ontology, or a combination of these. In our initial implementation, we make the simplification that the ontology matcher returns *true* only if there is an exact match of field names. However the more general case requires some form of ontological reasoning. We envisage this reasoning being done by a separate module (the "ontological reasoner") which could use more sophisticated reasoning to return true/false, or a probability representing the degree of a match. Malucelli *et al.* [10] propose such an ontological reasoning service to facilitate participation of agents in institutions.

In the example above we would hope that if the field "#region" type=string in the institutional message was replaced by "#suburb" type=string, it could still be matched by the ontological reasoner with the field "#region" type=string in the agent message, as region subsumes suburb. We do note however that this kind of ontological representation and reasoning is fraught with difficulty. The existence of agreed ontologies within an application domain can be expected to simplify the problem somewhat.

Finally, it is possible that an agent may need to transition through a number of scenes to obtain a successful outcome. Therefore, the institution will need to provide to the IMA the full set of messages involving the initial role, and all roles it can transform into within necessary scenes. For simplicity, at this stage role transformations should be mapped back to the name of their first occurrence.

## 6 Protocol Compatibility

A protocol defines the "rules of procedure" for a conversation, that is, describes possible sequences of messages that interacting agents must follow in order to achieve their goals. Given that agents can send and receive the right types of messages, the next step is to check whether they send these messages in the right order, that is, whether their protocols are compatible. For example, protocol checking should detect an incompatibility if agent $A$ can send $m_1$ followed by $m_2$, but the recipient expects $m_2$ followed by $m_1$.

In order to check protocol compatibility we need a notation to capture protocols. This notation must be expressible in a machine-readable form that can be used by the IMA. A number of notations could be used, including BPEL4WS, WS-CDL, and AUML[12]. In fact, for our discussion we do not make any assumptions about the notation. We use AUML because it is the de-facto standard for describing agent interactions, because its representation is compact, and because it has a machine-readable form [11].

Roughly speaking, the intuition behind protocol compatibility checking is the same as for message compatibility checking: it is the sender who has the choice of what to send, and thus the receiver needs to handle anything that the sender can send. For example, if at a given point in the interaction the sender can send two messages, *Confirm flight* and *Decommit flight*, then the recipient needs to be able to handle either option, and may handle additional messages. However, if the recipient cannot handle, say, *Confirm flight* then the interaction may fail if the sender chooses to send *Confirm flight*. The generalization of this intuition yields a definition of protocol compatibility: an agent's protocol is compatible with the institution's protocol if at any point in the interaction, the agent's protocol provides the agent with fewer options for messages that it can send,

[12] Agent UML: http://www.auml.org

and with more options for messages that it can receive with respect to the institution's protocol. Very similar ideas have also been developed independently in [12] and can also be found in [13, 14]. In [15] the notion of common protocols in open environments, and ways to achieve common protocols have also been discussed.

We can formalize the above intuitions by using $R_I$ (respectively $R_A$) to denote the institution's (respectively agent's) protocol messages that the agent can receive at a given point. Similarly, using $S_I$ (respectively $S_A$) to denote the institution's (respectively agent's) protocol messages that the agent can send at a given point in the interaction. The above definition of protocol compatibility can be formalized as requiring that at each point in the interaction $S_A \subseteq S_I$ and $R_I \subseteq R_A$.

We now briefly consider four cases where, at a given point in the interaction, there are differences between the protocols. If $S_A \subset S_I$ then the agent follows a sub-protocol of the one provided by the institution. For example, let the agent be a flight service that uses the protocol in Figure 3a while the institution uses protocol 3b. At the end of the protocol $S_A = \{\text{Confirm flight X}\}$ and $S_I = \{\text{Confirm flight X}, \text{Decommit flight X}\}$. This is OK, as long as the sub-protocol allows the agent to get from an entrance state to an end state in the institution, which is the case in this example. However, if the agent cannot get to an end state because it misses a necessary branch of the protocol then the agent's protocol is not suited.

If $R_I \subset R_A$ then the agent is able to handle more messages than can be sent, and this does not constitute a problem.

If $S_I \subset S_A$ then, at that point in the interaction, there are messages that the agent's protocol permits the agent to send which are not permitted by the institution's protocol, and which will not be expected by the message's recipient, thus making the protocols incompatible. Similarly, if $R_A \subset R_I$ then the protocols are incompatible because at that point in the interaction the institution's protocol indicates that the agent needs to be able to handle certain messages which are missing in the agent's protocol. For example, if the institution has the protocol in Figure 3b and the customer agent has protocol 3a then the customer agent does not expect a decommitment at the end, and will not be prepared to handle it.

Note that incompatibility does not mean that any interaction will *necessarily* fail, just that it *may* fail. It is quite possible, for example, that the flight service might choose to send a commit message, which the customer agent can handle, rather than a decommit, which it cannot.

More generally, instead of merely checking whether two protocols are compatible as-is, it is possible to determine what constraints would make the protocols compatible. For example, if the flight service is willing to accept a constraint that prevents it from sending a decommit message, then the two protocols (3b at the institution and 3a at the customer agent) become compatible. In order to be willing to adopt constraints, other constraints may need to hold. For example, if the institution guarantees that all transactions are performed within 1 minute then all options provided by the flight service are still available when chosen by the customer, and a decommit is unnecessary.

The discussion of compatibility so far has focused on a "pointwise" comparison between states. We now briefly consider compatibility if the two protocols are not of the same length. For example, if one protocol has a sub-sequence of messages not present

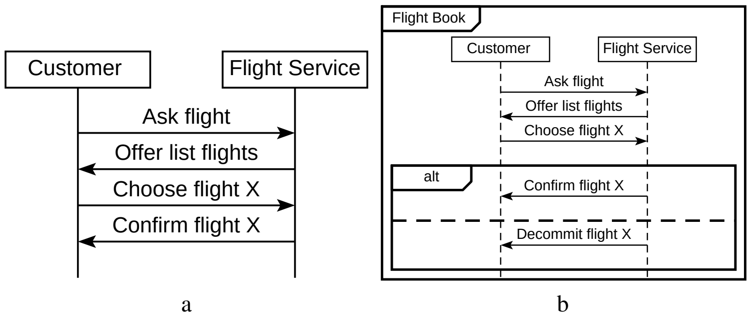

**Fig. 3.** Example protocol.

in the other. If this sub-sequence is optional then this case is captured above. Otherwise, if the sub-sequence occurs in the middle of a protocol then the protocols are not compatible.

However, if the sub-sequence occurs at the start or end of the protocol then the protocols may be compatible if the sub-sequence can be skipped because it is only important for the agent. For example, if a customer starts with choosing a flight without first asking for a list of possibilities. However, if the sub-sequence is important for the institution then it cannot be skipped. For example, if the institution needs an identification of the customer agent before interacting with it further.

Concluding this brief sketch of protocol compatibility issues we can state that there are some cases in which the protocols are not identical, but interactions might still be successful. Sometimes, this also depends on the content of the messages and sometimes on the internal architecture of the agent. As usual, the more knowledge one has, the more can be done. However, even with limited knowledge gained from standard descriptions a rudimentary check can be made that could suffice for the majority of the cases in practice.

## 7 Related work

Open systems assume that the heterogeneous agents are designed and run independently from each other and use their own motivations to determine whether to join an existing institution or another interaction space. However, currently, in most MAS, agents are simply designed from scratch so that their behavior complies with the behavior described by the role(s) it will take up in the society. Comprehensive solutions for open environments require complex agents that are able to reason about their own objectives and desires and thus decide and negotiate their participation in an organization. In [16], the following requirements were identified for the effective design of open agent environments:

1. Formal frameworks are needed for the specification of the society structure and goals with verifiable and meaningful semantics in a way that does not require the formal specification of participating agents.

2. Mechanisms are needed through which prospective participants can evaluate the characteristics and objectives of roles in the society, in order to decide about participation.
3. Tools must be available that support an individual agent to identify specific requirements and particularities of an organization and to adapt their architecture and functionality to the requirements of an assumed role.

That same paper proposed a first step on the road to this solution (cf. [16]) by introducing a formalism to compare the specification of agents and roles and determine whether an agent is suitable to enact a role. This solution is based on the OperA framework [9] that fulfills the first requirement above by providing a formal organizational model that is verifiable without the need to specify specific participating agents.

The work by Sierra et al, in the area of electronic institutions has been highly influential in the area of open agent systems [1]. They also faced the problem of how agents could interact with an institution while the agents and institutions are designed and implemented by different groups. They alleviated this problem by introducing the idea of "skeleton agents" [17]. These skeletons consist of the communication patterns that an agent needs to be able to function in the institution. The idea is that external parties use the skeletons as a basis for developing their agents and thus have an easy way of building agents that fit with the institution. The main restriction is that the agents are still assumed to be developed explicitly for one particular institution.

In [18] an architecture is described in which highly heterogeneous teams are able to cooperate. This is achieved by creating some semi-autonomous proxy agents that take care of the coordination layer between the agents. In fact what this seems to do is extract from the agents the part that they need in order to function within the coordinated team framework. This is similar to what we would like to extract from an agent, but we do not pre-suppose a certain type of interaction pattern as related to teamwork, but rather any type of interaction within an institution.

There is of course also some work done concentrating on the use of "middle agents" within the agent research community. Most notable is the work on RETSINA [19], being one of the first to realize the benefits of using middle agents to connect different parts of a system which each have their own goal and internal structure. The work that comes closest to our purpose for using a middle agent is that of task delegation. The middle agent matches task descriptions of demand and supply. However, in our case we do not match task descriptions but rather interaction patterns and possibly goals

In a similar way the work of [20] is relevant. In that work middle agents were used to manage subscriptions to different types of services. Services could be seen as very simple forms of institutions. The main simplification is that a service has a very simple interaction pattern and the interaction is with only one other party. Therefore, the match between the capabilities and tasks of an agent and a description of what a service can provide is relatively simple.

## 8 Discussion and future work

In this paper we have presented an approach to support open agent environments, where independently created agents can enter suitable institutions, with the assistance of an

IMA. This approach goes beyond the so called "skeleton agents" [17] that assist external parties to build agents that fit a specific institution. We assume standard descriptions such as WSDL and WS-CDL to describe different interaction patterns in order to perform the matching. Based on this, some basic compatibility checks can be made. However, a more semantic matching requires introspection of the agent about its goals and roles. Ontological reasoning also plays an important part in this matching.

The contributions of this paper are: the development of the IMA architecture for facilitating entry of agents into institutions, and the identification and description of reasoning processes (other than the ontological reasoning) which must be done by the IMA, including identification of potential roles the agent can take on, matching of messages between the agent and institution, and checking protocol compatibility.

Further work includes an actual implementation of the IMA other than the basic work done in this paper using one specific agent tool and one specific institution tool, generating plug-ins to extract message and protocol descriptions from agents and institutions' specifications, and exploration of more examples, using either standardized ontologies, or existing ontological reasoners of various types, to better understand what will work well in practice. This paper provides the framework and basis on which such work can be continued. It also identifies some aspects of ISLANDER or similar systems which are necessary for the proposed approach to work, such as institutional management of role (or role name) changes as the agent moves between different scenes within the institution.

## Acknowledgments

We would like to acknowledge the support of the Australian Research Council and Agent Oriented Software, under grant LP0453486, the Australian Department of Education, Science and Training, under grant CG040014. We also acknowledge the support of the Netherlands Organization for Scientific Research (NWO), under Veni grant 639.021.509. We thank the Australian Tourism Data Warehouse (ATDW) for the provision of live content for our travel and tourism agents. We also thank Carles Sierra and Juan Antonio Rodriguez for their assistance in understanding and working with ISLANDER.

## References

1. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: ATAL-2001. LNAI 2333, Springer (2001) 348–366
2. Arcos, J.L., Esteva, M., Noriega, P., Rodrguez, J.A., Sierra, C.: Engineering open environments with electronic institutions. Engineering Applications of Artificial Intelligence **18** (2005) 191–204
3. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer-Verlag, Berlin, Germany (2004)
4. Sycara, K.: Multi-agent infrastructure, agent discovery, middle agents for web services and interoperation. In: Multi-Agent Systems and Applications, LNAI 2086, Springer-Verlag (2001) 17–49

5. Sycara, K., Wido, S., Klusch, M., Lu, J.: Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. Autonomous Agents and Multi-Agent Systems **5** (2002) 173–203
6. Cheong, C., Winikoff, M.: Hermes: Implementing goal-oriented agent interactions. In: Programming Multi-Agent Systems, 3rd International Workshop, The Netherlands (2005)
7. Dignum, V., Dignum, F.: Coordinating tasks in agent organizations. or: Can we ask you to read this paper? In: Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (COIN@ECAI'06). LNAI, Springer (2006)
8. Odell, J., Parunak, H.V.D., Fleischer, M.: The role of roles in designing effective agent organizations. In Garcia, A., Lucena, C., Zambonelli, F., Omicini, A., Castro, J., eds.: Software Engineering for Large-Scale Multi-Agent Systems. Volume 2603 of LNCS., Springer-Verlag (2003)
9. Dignum, V.: A Model for Organizational Interaction: based on Agents, founded in Logic. SIKS Dissertation Series 2004-1. Utrecht University (2004)
10. Malucelli, A., Cardoso, H., Oliveira, E.: Enriching a MAS Environment with Institutional Services. In: Environments for Multi-Agent Systems II, 2nd International Workshop, The Netherlands (2005)
11. Winikoff, M.: Towards making agent UML practical: A textual notation and a tool. In: First international workshop on Integration of Software Engineering and Agent Technology (ISEAT 2005). (2005)
12. Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: Verification of protocol conformance and agent interoperability. In: CLIMA VI, Springer, LNCS 3900 (2005) 265–283
13. Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are two Web Services Compatible? In: TES'04, Springer, LNCS 3324 (2005) 15–28
14. Yellin, D., Strom, R.: Protocol specifications and component adaptors. ACM Transactions on Programming Languages and Systems (TOPLAS) **19** (1997) 292–333
15. Paurobally, S., Cunningham, J.: Achieving common interaction protocols in open agent environments. In: Challenges in Open Agent Systems '03 Workshop, Melbourne, Australia, AAMAS (2003)
16. Dastani, M., Dignum, V., Dignum, F.: Role assignment in open agent societies. In: Proceedings of Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS), ACM Press (2003)
17. Vasconcelos, W., Sabater, J., Sierra, C., Querol, J.: Skeleton-based agent development for electronic institutions. In: Proceedings of First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS), ACM Press (2002) 696–703
18. Scerri, P., Pynadath, D., Schurr, N., Farinelli, A., Gandhe, S., Tambe, M.: Team oriented programming and proxy agents: The next generation. In: Proceedings of 1st international workshop on Programming Multiagent Systems. LNAI 3067, Springer (2004)
19. Sycara, K., Paolucci, M., Velsen, M.V., Giampapa, J.: The RETSINA MAS Infrastructure. Autonomous Agents and Multi-Agent Systems **7** (2003) 29–48
20. Mbala, A., Padgham, L., Winikoff, M.: Design options for subscription managers. In: Proceedings of the 7th International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS). (2005)