Engineering Conflict-Free Multiagent Systems[*]

Akın Günay and Pınar Yolum

Computer Engineering Department, Bogazici University, Istanbul, Turkey
{akin.gunay,pinar.yolum}@boun.edu.tr

Abstract. Organizations have served an important metaphor for designing and engineering multiagent systems. Usually, organizations are specified over abstract roles, which specify what the enacting agents should or should not do. In order to ensure that organizations work properly, we need to check that the roles have been designed correctly in the first place. Accordingly, we analyze various relations between roles to check if they include conflicting terms, which can prevent the enacting agents to conform with the obligations and prohibitions. We identify various types of conflict that may occur in a role specification and between different roles in an organization. In order to formalize the discussion we represent role specifications as sets of commitments. Then using the conflict relations between commitments, we define basic principles that should be followed in order to develop role-based conflict-free multiagent organizations.

1 Introduction

Organizations have been an important metaphor in understanding, designing, and engineering multiagent systems [1, 6]. Organizations allow division of tasks, assignment of these tasks to different individuals and provide a mechanism for disseminating task results between individuals. Many times, organizations are specified abstractly over roles, rather than individual agents. This allows a generic specification so that at different times different agents can be matched to various roles and realize the organization as a multiagent system.

Organizations exist to carry out specific tasks or reach specific goals. A correct specification of an organization should guarantee that the organization works in harmony and the overall goals of the organization are achieved. In order to achieve this objective the specification of the organization should be conflict-free. Since roles are the main building blocks of organizations, the first step of designing a conflict-free organization is to ensure that the roles in the organization are themselves conflict-free and the agents that attempt to carry out the roles are not prevented from satisfying the requirements of the roles out of their will. For example a program committee member role in a conference organization should not concurrently require and prohibit the enacting agent to review a paper. This would be conflicting and would prevent the agent to fulfill the requirements of the role event if it intends to do so, which in turn causes also the organization to fail to achieve its objective.

[*] This work is partially supported by Boğaziçi University Research Fund under grant BAP5694, and the Turkish State Planning Organization (DPT) under the TAM Project, 2007K120610.

Besides the conflicts within the role specification, another issue that may prevent an organization to work in harmony arises when agents are allowed to enact multiple roles simultaneously. In this case it is not enough only to ensure that the roles themselves are conflict-free, but also the specifications of the roles that may be enacted by the same agent should be conflict-free with each other. If an agent enacts to two roles at the same time, it should not be required to bring about a condition as part of the first role and required not to bring about the same condition as part of the second role. For example in a conference organization agents usually enact both to program committee member and author roles simultaneously. Program committee member role obligates the enacting agents to review assigned papers. On the other hand author role prohibits the enacting agents to see papers of other authors. When the mentioned obligation and prohibition of these two roles are considered, it makes the impression that these two roles should not be enacted by the same agent. However, these two roles actually can be enabled to the same agent by allowing the agent to access only to certain papers that are assigned to it. But, if this issue is not precisely specified, then the roles may conflict with each other by obligating the agent to review papers while prohibiting it from accessing to those papers.

Even if the roles in an organization are specified in a conflict-free manner while designing the organization, problems may still arise at run-time due to certain affairs that cannot be foreseen at design time. A major problem, which prevents the organization to function properly, occurs at run-time when the enacting agents suffer from lack of resources that are required to fulfill the obligations of their roles. For example, if five papers would be assigned to an agent enacting the program committee member role and the agent would have time only to review three papers, then the agent would fail to fulfill its obligation. In some organizations role specifications may include expected resource requirements of the role, which can be used by agents to check their compliance before enacting a role. However, making such predictions is usually challenging due to open nature of multiagent systems. For example, the specification of program committee member role may state that three papers are expected to be assigned to enacting agents for review. However, if the total number of submission would exceed the prediction that had been made while designing the system, more papers might be assigned to program committee members. Besides, in many situations it is not even possible to make such predictions about resources (e.g., scheduling of events).

Various works exist where multiagent systems are engineered through their specification as organizations [4, 24]. Once an organization has been specified, the general focus has been on assigning agents to particular roles. Most of these approaches rightly focused on whether the agent has the necessary capabilities to bring about the requirements of the roles. Aiming to create more robust and conflict-free multiagent organizations, here, we consider various other aspects that might cause to conflicts in designing and infeasibility while executing these organizations. To do so, we represent obligations and prohibitions of roles of an organization through commitments. We identify situations where commitments would conflict thereby leading to a role conflict. Similarly, we use the feasibility concept of commitments to capture infeasible situations at run-time and we discuss several methods to deal with such situations.

The rest of this paper is organized as follows. Section 2 gives the necessary background on commitments and describes how roles are denoted through commitments. Sections 3 and 4 develop what it means for roles to conflict and how they can be resolved. Finally, Section 5 discusses our work in relation to recent literature.

2 Technical Background

In the rest of the paper we use propositional symbols to represent domain dependent concepts. We assume that these concepts are defined in a domain ontology that is available to all agents. We use the logical connectives $\wedge$ and $\rightarrow$ in their common semantics and $\neg$ as logical negation. We use the variables x, y for agents, p, q, u, w for propositions, c_i for commitments and r_i for roles. Constant symbols start with capital letters. In general we use a conference organization as a running example, as it is widely used in the previous work about organizations. Besides, to make our discussion more refined in some examples we also use some other related domains when these domains are more intuitive.

2.1 Commitments

C(x, y, q, p) denotes a commitment from the debtor agent x to the creditor agent y to bring about the consequent p, if the antecedent q holds [22]. The antecedent of a commitment can be a conjunction of propositions, however we assume that the consequent of the commitment is a single proposition. Let us consider the following example from the conference organization domain. In this domain a member of conference's program committee (PCMember) is committed to review a paper (PaperReviewed), if the program chair (Chair) assigns the paper to the program committee (PaperAssigned). We denote this using the following commitment:

$$\mathsf{C(PCMember, Chair, PaperAssigned, PaperReviewed)}$$

A commitment is a dynamic entity with a lifecyle [7, 17, 22]. A commitment's state evolves as the result of certain actions that are performed by the participating agents. In order to focus on our contributions, in this paper we use a simplified lifecycle of commitments as follows: A new commitment is created by its debtor using the Create(x, y, q, p) action, which creates the conditional commitment c =C(x, y, q, p). When the antecedent of the commitment starts to hold, the creditor performs Detach(c, q) action that makes the commitment active. To represent an active commitment we replace the antecedent of the commitment to $\top$ symbol (i.e., C(x, y, $\top$, p)). When the consequent starts to hold, the debtor performs Discharge(c, p) action that makes the commitment fulfilled. On the other hand, if the debtor cancels its commitment using Cancel(c) operation, then the commitment is canceled. In addition to committing to bring about a condition, agents may also commit to maintain a condition [2, 13, 23]. For instance, the web admin of a conference should keep the Web site of the conference running. We represent such commitments by drawing a line on top of the corresponding condition.

$$\mathsf{C(WebAdmin, Chair, \top, \overline{KeepWebRunning})}$$

Such commitments cease to exist only when the creditor of the commitment c releases the debtor by performing the Release(c) action.

2.2 Role Specifications

A role describes what an agent should and should not be doing. For instance, an agent in the program committee member role should review assigned papers by the program chair. This can be seen as the agent's obligation. On the other hand, the same agent should not be reviewing a colleague's paper. This can be seen as a prohibition. We represent each such clause (i.e., obligation or prohibition) as a separate commitment. Considering the program committee member role the obligation to review assigned papers can be represented using the following commitment:

$$\mathsf{C(PCMember, Chair, PaperAssigned, PaperReviewed)}$$

Similarly, the prohibition to not to review colleagues papers can be represented using the following commitment:

$$\mathsf{C(PCMember, Chair, AuthorIsColleague, \overline{\neg PaperReviewed})}$$

A role exists as part of an organization committee role exists because a conference is being organized. Depending on the progress of this organization, each role can be active or passive. The state of the roles are triggered by some conditions. For instance, the program committee member role becomes active when the paper submission is closed. That is, the requirements of the roles will apply on after this time. Similarly, the role becomes passive after the authors are notified about the acceptance of papers. From this point on, the role is not binding.

Definition 1 (Role). *A role is a tuple $\langle \lambda, \mathbb{L}, \mathbb{C} \rangle$, where λ is the unique label of the role, $\mathbb{L}$ is formulae to define the conditions that activate and passivate the role, and $\mathbb{C}$ is a set of commitments that the enacting agent should create.*

λ	PCMember
$\mathbb{L}$	*Activate*:{SubmissionClosed}, *Passivate*:{AuthorsNotified}

$\mathbb{C}$	C(PCMember, Chair, PaperAssigned, PaperReviewed)
	C(PCMember, Chair, AuthorIsColleague, $\overline{\neg\text{PaperReviewed}}$)

Table 1. Specification of the program committee member role.

We present a simplified role specification for a program committee member in Table 1. The unique label of the role is PCMember. The role is activated when the paper submission is closed and passivated when the authors are notified about the results. The program committee member is obligated to review assigned papers by the program chair (represented by the first commitment). The program committee member is prohibited to review papers if an author is also a colleague (represented by the second commitment). Note that our role specification does not require any particular organization structure. It can be used as part of any organizational schema that has roles as part of it.

3 Role Conflicts

While acting in a multiagent system, an agent usually participates in multiple commitments at the same time. In such situations, the commitments of an agent may conflict. For instance, if a program committee member agent would be (somehow) obligated and prohibited to review a paper simultaneously because of its two different commitments, these commitments would cause a conflict and as result of this conflict the agent would be unable to fulfill both commitments. In other words the agent would inevitably violate one of its commitments because of the conflict between the commitments and nothing that the agent can do can fix this. This is an undesired situation and should be avoided whenever possible. In this section we first give a formal definition of such commitment conflicts. Then in the rest of the section we discuss such conflicts in the context of role specifications.

As the above example demonstrates, a conflict occurs between two commitments of an agent when the consequences of the commitments are inconsistent with each other and hence cannot be brought about at the same time. In this context we provide a simplified version of a commitment conflict between two commitments as follows [9].

Definition 2 (Commitment Conflict). *Given two (active or conditional) commitments* c_i =*C(x, y, w, u) and* c_j =*C(x, y', w', u'),* c_i *and* c_j *conflict with each other, denoted by* $c_i \otimes c_j$*, if;*

- *it is not the case that* w $\rightarrow$ q *and* w' $\rightarrow$ $\neg$q*, and*
- *it is the case that* u $\rightarrow$ p *and* u' $\rightarrow$ $\neg$p.

In order for two commitments to conflict, their consequences should be inconsistent. That is, while one commitment is telling the agent to do one thing, the second commitment should say the opposite. We capture this by requiring that while one consequent implies a proposition, the other one implies the negation of the same proposition. The second condition in the previous definition captures this. If both commitments are active commitments (i.e., their antecedents already hold), then this is the only condition that needs to be checked. However, if either or both of the commitments are in the conditional state, then first a check is necessary to see if both commitments can actually end up in active state. For this to happen, the antecedents of both commitments should not be inconsistent; that is they should be able to hold at the same time. This will enable both commitments to become active at the same time. The first condition in the previous definition ensures this.

3.1 Conflicts in a Role Specifications

Development of a role-based organizational model requires accurate analysis of obligations and prohibitions that are associated with each role. However, such analysis is a challenging task and may create conflicts if not done properly. If a role is not designed carefully and the specification includes conflicting obligations and prohibitions (i.e., conflicting commitments), then the enacting agent ends up in a situation where it inevitably violates its commitments, which in turn affects proper working of the organization. For instance, consider again the program committee member role and the associated commitments to review assigned papers and not to review colleagues' papers, which are represented as follows:

$$c_1 = \mathsf{C(PCMember, Chair, PaperAssigned, PaperReviewed)}$$

$$c_2 = \mathsf{C(PCMember, Chair, AuthorIsColleague, \overline{\neg PaperReviewed})}$$

These two commitments are in conflict ($c_1 \otimes c_2$), since the consequences are inconsistent, while the antecedents are not. Once a paper is assigned to a program committee member (i.e., PaperAssigned holds), then c_1 is detached. Hence, the program committee member is obligated to review the paper (i.e., committed to bring about PaperReviewed). On the other hand, if the author of the assigned paper is a colleague, then c_2 is also detached (i.e., AuthorIsColleague holds). Hence, the program committee member is prohibited to review the paper (i.e., committed not to bring about PaperReviewed). When both commitments become active, then the conflict between the commitments prevents the program committee member to fulfill c_1 and c_2 simultaneously.

This conflict occurs due to the imprecise description of the commitments' antecedents. The above conflict can easily be avoided by making the antecedents more specific.

$$c'_1 = \mathsf{C(PCMember, Chair, PaperAssigned \wedge \neg AuthorIsColleague, PaperReviewed)}$$

$$c'_2 = \mathsf{C(PCMember, Chair, PaperAssigned \wedge AuthorIsColleague, \overline{\neg PaperReviewed})}$$

The modified antecedents are inconsistent (i.e., cannot hold at the same time). Hence, the commitments do not conflict anymore. More specifically, if the author of the assigned paper is not a colleague, then the antecedent of c'_2 fails to hold. Hence, the only active commitment can be c'_1. On the other hand, if the author of the paper is a colleague, then the antecedent of c'_1 fails to hold and the only active commitment can be c'_2.

The above conflict is a result of poor analysis of the situation. However it can be resolved with certain modifications of the commitments. On the other hand, some fallacies in the analysis may lead to fatal cases in which conflicts cannot be avoided. Such situations usually occur when a role is overloaded with obligations and prohibitions that actually should not be a part of the role. For instance, in a blind-review process the program committee member is prohibited to see author information of the papers. Assume that the program committee member role is analyzed inaccurately and the task to assign reviewers to submitted papers, which is normally a task of the program chair, is associated with the program committee member role. In this case the following commitments are part of the role specification of the program committee member:

$$c_1 = \mathsf{C(PCMember, Author, PaperSubmitted, ReviewerAssigned)}$$

$$c_2 = \mathsf{C(PCMember, Author, \top, \overline{\neg AuthorInfoAccessed})}$$

Note that in order to assign a paper, the program committee member should see the author information of the paper (i.e., $\mathsf{ReviewerAssigned} \rightarrow \mathsf{AuthorInfoAccessed}$). Hence, the propositions PaperAssigned and $\neg$AuthorInfoAccessed are inconsistent. Accordingly, c_1 and c_2 do conflict with each other. Different than the previous example, here it is not possible to avoid this conflict by modifying the commitments, which indicates that the existence of one of these commitments in the context of the program committee member role is conceptually wrong. That is c_2 should not be a part of this role specification in the first place. The above discussion leads us to the following result.

Definition 3 (Incoherent Role Specification). *A role specification r is incoherent, denoted by $I(r)$, if there is a conflict between two commitments that are part of the role. Formally:*

$$\exists\, c_i, c_j : c_i \in r.\mathcal{C} \wedge c_j \in r.\mathcal{C} \wedge c_i \otimes c_j \Rightarrow I(r)$$

Conjecture 1 *If a role specification is incoherent, then the agent that enacts the role cannot fulfill all commitments.*

Conjecture 1 points out to a conceptual problem in the design of the incoherent role, which should be fixed either by modifying the commitments or reconsidering the involvement of some commitments as part of the role specification.

3.2 Conflicts between Roles

In many situations agents in an organization enact more than one role at the same time. For instance, in the conference organization an agent that enact to the program committee member role may also enact for the paper author role at the same time. In such situations, even if the role specifications enacted by the agent are individually coherent, conflicts may still arise between the agent's commitments that belong to different roles. Assume that an agent enacts both the program chair and program committee member roles at the same time. The program chair role involves the obligation to assign a reviewer to the submitted papers as represented by the following commitment:

$$c_1 = \mathsf{C(Chair, Author, PaperSubmitted, ReviewerAssigned)}$$

On the other hand the program committee member role prohibits the enacted agent to see the author information of papers as the following commitment states:

$$c_2 = \mathsf{C(PCMember, Author, }\top\mathsf{, }\overline{\neg\mathsf{AuthorInfoAccessed}}\mathsf{)}$$

As we have mentioned in the previous section, these two commitments conflict since assignment of a paper to a reviewer requires the agent to access the author information of the paper which is inconsistent with the prohibition of the program committee member role to access author information of papers. Assuming that the program chair and program committee member roles are coherent in themselves, such a conflict indicates that these two roles should not be enacted by the same agent at the same time. We call such roles as *mutually exclusive* roles.

However, in many real life situations conflicts between the commitments of different roles may not always require the roles to be mutually exclusive. For instance, a program committee member is allowed to see the contents of the papers she has been assigned. On the other hand, an author should not be seeing the contents of others' papers. However, if an agent enacts to both roles (assuming the agent is qualified for both), the agent can actually access the content of other authors' papers without violating its role as an author. In this case the commitment of to the program committee member overrides the commitment of to the author role. Hence, the prohibition of the author role to not view others' papers is not valid anymore. When such a case is a part of the modeled organization, then the override relation of the commitments should be explicitly represented.

Definition 4 (Override Relation of Commitments). *Given two commitments c_i and c_j, $c_i \gg c_j$ denotes that c_i overrides c_j.*

Definition 5 (Mutually Exclusive Roles). *Two roles r_i and r_j are mutually exclusive, denoted by $r_i \bowtie r_j$, if there is a conflict between at least one commitment of each role and there is no override relation between the conflicting commitments. Formally;*

$$\exists c_i, c_j : c_i \in r_i \wedge c_j \in r_j \wedge c_i \otimes c_j \wedge c_i \not\gg c_j \wedge c_j \not\gg c_i \Rightarrow r_i \bowtie r_j$$

Conjecture 2 *If two roles are mutually exclusive, then these roles cannot be enacted by the same agent simultaneously.*

Again, understanding whether two roles are mutually exclusive can be detected during design time. Since, an agent that is enacting mutually exclusive roles are doomed to violate some of its commitments, it is wise to design the organization to avoid mutually exclusive roles.

4 Run-time Feasibility

The reason of the conflicts we describe in the previous section is the inconsistency of the commitments (i.e., obligation and prohibition of an agent simultaneously to bring about the same condition). Such conflicts should be avoided at design time in order to come up with a conflict-free multiagent system.

Even though a multiagent system is designed free of conflicts by making the role specifications coherent and by restricting enactment of mutually exclusive roles by the same agent, agents may still face up with challenges at run-time when they enact the roles. One important reason for this is the lack of resources needed to satisfy the commitments that the agents are involved in. Such situations arise when the commitments of an agent are *infeasible* [10].

In general, a role specification covers the capabilities of an agent that are required to enact the role. However, the actual resources that are required while performing these capabilities are usually revealed only at run-time[1]. For instance, remember the commitment of the program committee member role to review the assigned papers. By the role specification it is clear that the agent that enacts this role should have the capability to

[1] We assume that the resource types and their amounts that are required to perform certain capabilities are represented in a domain ontology.

review an assigned paper. However, the actual number of papers that may be assigned is not pointed out in the role specification. If the agent would be capable to review only a certain number of papers and the number of assigned papers would exceed this limit at run-time, then the commitments of the agent would be infeasible and the agent would have no way to fulfill its commitments.

Note that in some situations the amount of resources required while enacting the role can be defined at design-time. For instance, the maximum number of papers that would be assigned to a program committee member agent might be defined in the role specification. However, due to the open nature of multiagent systems making such assumptions is usually not possible. For instance, the actual number of submissions at run-time might exceed the number that has been predicted at design-time. In such a situation more papers than it is defined in the role specification might be assigned to program committee member agents.

The issue is even more challenging if temporal constraints are introduced to the designed multiagent system. For instance, consider an agent that plays a professor role. The agent is obligated to conduct research and submit papers to conferences to publish results. The agent is also obligated to attend a conference to make a presentation, if a submitted paper is accepted. These obligations are represented by the following commitments in the professor role specification.

$$c_1 = \mathsf{C}(\mathsf{Professor}, \mathsf{Department}, \top, \mathsf{PaperSubmitted})$$

$$c_2 = \mathsf{C}(\mathsf{Professor}, \mathsf{Department}, \mathsf{PaperAccepted}, \mathsf{ConferenceAttended})$$

Another obligation of the professor role is giving lectures, if a lecture is assigned.

$$c_3 = \mathsf{C}(\mathsf{Professor}, \mathsf{Department}, \mathsf{LectureAssigned}, \mathsf{LectureGiven})$$

Assuming that the agent cannot attend to a conference and give a lecture simultaneously (i.e., ConferenceAttended and LectureGiven cannot hold at the same time) one of c_2 or c_3 would be violated by the agent, if these two events are scheduled to the same time. However, it is not possible to determine exact times of these events while designing the professor role. On the other hand, it is also not rational to remove one of c_2 or c_3 from the role specification, only because they may introduce a potential infeasibility at run-time. Nevertheless, in order to build a solid multiagent system, both the agents that aim to enact roles and the design of the multiagent system should be fitted out with mechanisms in order to deal with infeasibility.

In the rest of this section we first give a definition of feasibility adopting the definition we have developed in our previous work [10]. Then we define rules-of-thumb in the context of roles and commitments to deal with infeasibility, once it is detected.

Definition 6 (Running Multiagent System). *A running multiagent system m is a two-tuple $\langle \mathcal{A}, \mathcal{C} \rangle$, in which $\mathcal{A}$ is a set of agents and $\mathcal{C}$ is a set of commitments created by the agents in $\mathcal{A}$.*

Note that we do not mention the roles in the definition of a running multiagent system. However, we assume that the agents in the system create commitments in accordance with the role specifications they enact following to the role based design of the multiagent systems.

Definition 7 (Snapshot). *s_t is a snapshot of the commitments in $m.\mathcal{C}$ and their states in a running multiagent system m at moment t.*

Hence, from a snapshot one can induce what commitments have been created, which one have been fulfilled, and so on.

Definition 8 (Commitment Feasibility). *In a running multiagent system m, a given set of active or conditional commitments $C \subseteq m.\mathcal{C}$ in snapshot s_t is feasible, if there is a snapshot $s_{t'}$ in which every commitment in C is fulfilled and $t < t'$.*

The feasibility definition states that a commitment set is feasible at a given moment, if the multiagent system may progress in such a way that all commitments in the initial set are fulfilled. A multiagent system progresses based on the rules of commitment lifecycle. For example, an active commitment can either be fulfilled or violated, but a violated commitment can never become fulfilled. By generating possible future snapshots, one can infer it is possible for all the commitments to be successfully fulfilled. Hence, feasibility does not guarantee fulfillment of commitments, but only states that this is possible. On the other hand, once infeasibility of commitment is detected, then it is certain that the commitments cannot be fulfilled as they are.

Achieving Feasibility by Delegation: In many organizational models (e.g., hierarchies) agents who enact certain roles may have the power to delegate some of their responsibilities to other agents [12]. For instance, a professor may delegate lecturing duty to a teaching assistant (TA). In terms of commitments, delegation can be performed using the Delegate(c, a) operation, which states that the commitment c is delegated to the agent a. This operation is equal to releasing c and creating a new commitment with identical conditions, but new debtor a [17]. Although delegation provides an intuitive mechanism to resolve infeasibility, it should be used carefully. In this context, two major issues should be considered. The first issue is the problem to precisely define the delegation power over roles. This can be achieved by introducing new commitments to define delegation power. Although the same functionality can be achieved by expanding the notion of role specifications, commitment based approach is advantageous, since it eliminates introduction of new structures to the role specifications. For instance, in the specification of a teaching assistant role there might be a commitment, which states if a commitment is delegated (CommitmentDelegated) by the Professor (actually the agent who enacts this role), then the teaching assistant is committed to accept the delegation, i.e, create the delegated commitment (CreateCommitment).

C(TA, Professor, CommitmentDelegated, CreateCommitment)

Note that in our conference organization examples in the previous sections, assignment of a submitted paper from the program chair to a program committee member for review is also kind of a delegation. However, in that case the details of the delegation is explicit and well defined in the role specification as part of the conceptual design of the organization. The delegation mechanism that we discuss here is a loose one. In the above example, the professor is free to delegate any commitment to the teaching assistant, which takes us to the second important issue about the delegation. That is whether the deputy agent (e.g., the teaching assistant) is capable to fulfill the delegated commitment. Two things should be considered in this context. First, it is necessary to determine whether the capabilities of the deputy agent is compatible with the requirements of the commitment in order to fulfill it. For instance, if the teaching assistant is not capable of giving lectures, then the delegation does not resolve the infeasibility. Second, it is necessary to check that the result of the delegation does not cause to another infeasibility, due to lack of resources of the deputy agent. For instance, if the teaching assistant is scheduled for another event at the lecture time, then the infeasibility persists.

Conjecture 3 *If the commitments of an agent are infeasible and the agent has delegation power to resolve the infeasibility, then the agent should delegate one or more commitments, only if the deputy or deputies are capable of and have enough resources to fulfill the delegated commitment(s).*

As generally accepted in multiagent systems, the capabilities and state (e.g., existing resources to the agent) of an agent are private to that agent. Hence, in general the delegating agent may not know whether the deputy is capable of or has enough resources to fulfill the delegated commitment. Therefore, in order not to create a new infeasibility as a result of the delegation, an interaction should be carried out between the delegating agent and the deputy before the delegation.

Compensation of Violation: Delegation may not be used always to deal with infeasibility (e.g., the agent may not have power for delegation, etc.). In such situations, compensation may be used as an alternative method to expel the noxious effects of violation [19, 21]. In compensation, when a commitment is violated, another commitment (usually including a sanction) is created by the debtor to compensate the violation of the earlier commitment. Although compensation does not prevent violation, it still provides a mechanism to bring the system back into a healthy state. For instance, if the conference and the lecture of the professor would be scheduled for the same time and accordingly the professor would violate its commitment, the professor might commit to give another lecture at a later time to compensate the missing lecture. However, as in the case of delegation, it is necessary to take into account another infeasibility that may occur due to creation of the commitment for compensation (e.g., new schedule of the lecture for compensation may cause to another infeasibility).

Compensation can be performed using the commitment operation Compensate(c_i, c_j), which states that if the commitment c_i is violated then the commitment c_j is created as a compensation. As in the case of delegation, compensation can also be represented as a commitment in the role specification, without introducing a new notion. For instance, the compensation of a missing lecture can be represented using the following commitment:

C(Professor, Department, LectureMissed, ScheduleLectureToCompensate)

In this example, LectureMissed holds when the professor's commitment to give the lecture is violated and ScheduleLectureToCompensate holds only when a new commitment is created by the professor to give another lecture for compensation.

Note that in this example the commitment that will be created for compensation is well defined at design time, which is scheduling of another lecture. In some situations it may not be possible to determine a compensating commitment for each possible violation while designing the role specification. In such situations it may be a good practice to integrate a generic commitment, which states that when a commitment is violated then a compensating commitment should be created, into role specifications to enforce the agents to compensate their violated commitments. However, in such a situation context of the compensation should be agreed by both the debtor and the creditor that are subject to the violated commitment, which may require additional interaction between the agents.

C(x, y, CommitmentViolated, CreateCommitmentToCompensate)

Preference for Fulfillment Although delegation and compensation provide intuitive mechanism to deal with infeasibility, in many situations it may not be possible to apply these approaches (e.g., the role may not have delegation power, compensation cannot be scheduled, etc.). In such situations it is inevitable for the agent to violate one or more of its commitments. Even in this case, it is important to decide which commitments to fulfill and which ones to violate. This decision can be made by taking different factors into account. For example, consider an agent that has two commitments to two different creditors and has to violate one of these commitments because of infeasibility. If the agent would care about the opinion of one creditor more than the other, it would be reasonable for the agent to violate the commitment of the less cared creditor. On the other hand, if there would be many number of infeasible commitments and violation of a single commitment, which requires the most resources, would resolve the infeasibility, then it would be reasonable to violate that commitment instead of violating several commitments with less resource requirements.

In order to deal with such situations, if it is possible at design-time, a role specification may include a preference relation over the commitments which defines a partial or total order to indicate which commitments to fulfill and which ones to violate in the case of infeasibility.

Definition 9. *Given two commitments c_i and c_j ,$c_i \rhd c_j$ states that c_i is preferred for fulfillment over c_j.*

For instance, considering the two following commitments of the professor role, the commitment to attend a conference may be preferred (based on importance from the point of the role) for fulfillment over the fulfillment of the commitment to give a lecture.

c_2 =C(Professor, Department, PaperAccepted, ConferenceAttended)

c_3 =C(Professor, Department, LectureScheduled, LectureGiven)

If this is the case, then the professor role specification should involve the preference order $c_2 \rhd c_3$ to state this situation. On the other hand, when it is not possible to define such a preference relation at design-time, then the agent itself should decide on which commitments to fulfill and which ones to violate in the case of infeasibility at run-time using other reasoning mechanisms.

5 Discussion

The abstraction of roles has been important in designing organization-based multiagent systems. However, understanding what goes into a role and how roles affect each other is an ongoing research question. This paper illustrates various problems that can arise in designing roles and suggests ways to overcome them. Once roles are conflict free, the next step is to study how the agents will enact these roles. To this end, various existing work studies, whether agents that enact these roles are fit to do so [3]. While this is important, there is not sufficient work on how an agent can stop enacting a role, how this would affect the agent's existing commitments, and so on. Intuitively, the agent can be released from its conditional commitments. With active commitments, the problem is more tricky. For instance, assume that a review period is over before a program committee member fulfills its commitment of reviewing a paper. Would this commitment be considered violated even if the role no longer exists and thus binds the agent?

Commitment conflicts and norm conflicts, which are closely related to commitment conflicts, are studied in multiagent research literature [9, 16, 20]. These studies focus on the formal definition of different conflict types, such as conflicts due to logical inconsistencies and temporal disputes. Here, we use a general notion of conflict based on the concurrent obligation and prohibition of the same condition via commitments. However, the points we emphasize here such as the conflict-free role specifications and mutually exclusive roles are mainly independent from the specific definition of conflict. Hence, these points should be taken into account while designing any role based multiagent system.

In this paper we adapt a generic definition of commitment feasibility from our previous work [10]. Computing an agent's commitments feasibility in practice is a challenging problem which requires to consider various resources of the agents, temporal constraints and also expectations of the agent in the interactions with others. In our previous work we use constraint satisfaction methods in order to achieve this objective. Here, we do not consider how feasibility of the agents commitments are computed. This can be done in different ways taking the structure of the considered multiagent system into account. On the other hand, we focus on what agents can do once an infeasibility is detected. However, the approaches that we present here are not the only responses that can be given by the agent to deal with infeasibility. For example, once infeasibility is detected an agent may take action to acquire more resources to resolve infeasibility. We left the investigation of such approaches as future work.

There are a lot of work on specifying organizations. AGR [6], Gaia [24], Tropos [1] and OperA [5] are important examples. AGR model specifies an organization through agents, groups, and roles. The interactions are generally specified with conditional actions. Both Gaia, Tropos and OperA focus on design of organizations with various rich constructs, such as capabilities, goals, and so on. Thus, their focus is on a global view of how these constructs exist together. Here, our aim is not on the interplay between various organizational constructs but on the conflicts that can arise within and among roles in an organization. On the other hand, integration of the concepts we present in this work into these frameworks would be beneficial for these approaches. Accordingly, we aim to extend at least one of these frameworks by introducing especially the conflict concept into the framework in our future research.

Formalization of organization level constraints is studied by van Riemsdijk *et al.* [15] using linear temporal logic in the context of MOISE$^+$ organizational modeling language [11]. An organizational constraint is a meta-constraint in the level of organization, which specifies how an organizational action such as enacting to a role restricts an agent's behavior (e.g., agent should adopt obligations of the enacted role) or cardinality requirements on the number of agents that enact a given role. While enforcing those constraints are obviously important, they are different than the constraints we are interested in here. Our concern here is at a lower-level to ensure that the roles are designed correctly to begin with.

Fornara, Viganò and Colombetti use institutional actions to define semantic of commitments mapping messages, which manipulate the state of a commitment, to institutional actions [8]. Beside institutional actions, they also consider norms in the form of event driven rules, which fire under certain conditions and cause creation of commitments that correspond to the norms. However, they are not concerned about the possible conflicts between the norms or roles as we have done here.

Dastani, Dignum and Dignum model a role as a set of goals and plans [3]. Obligations and prohibitions in the form of norms conceptually exist, but they are basically used to generate goals to be achieved and goals to be avoided, respectively. These type of conflicts are closely related to our definition of feasibility. However, our and their approaches consider this issue from different perspectives. In their perspective the conflict occurs with respect to a specific plan. On the other hand, in our perspective feasibility is independent from a specific plan or execution. Instead we use it as an indication of possible fulfillment and violation of agents' commitments.

Odell *et al.*discuss temporal aspects of roles [14], where they consider active and suspended states of roles. To capture the transitions between these states (and also for creation and termination) they define a set of operations. The first operation is *classify*, which is used to occupy a to an agent. The complementing operation *declassify* is used to make the role unoccupied again. *Activate* and *suspend* operation are used for the transitions from suspended to active and active to suspended states, respectively. In this paper, we use a simplified model through making roles active and passive based on some preconditions. This simplification is enough in our case, since we consider conflicts only when roles are active. However, it is straightforward to integrate our definitions into a more complete model of roles with more states.

Telang and Singh extend the TROPOS methodology [1] with commitments to model agent interaction [18]. In order to identify the commitments during the development, they first determine the major roles and goals of these roles. These high-level goals are decomposed into more fine-grained sub-goals. Finally, the goals are mapped to tasks that are required be performed by the agents who enact the roles to achieve the goals. They use these tasks and their dependencies to identify the commitments required to model the interaction in the system. Our approach can be coupled with theirs as a post-processing capability to check if the resulting roles are conflict-free.

References

1. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems

8(3), 203–236 (2004)

2. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment tracking via the reactive event calculus. In: Proceedings of the 21st International Joint Conference on Artifical Intelligence. pp. 91–96 (2009)
3. Dastani, M., Dignum, V., Dignum, F.: Role-assignment in open agent societies. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 489–496 (2003)
4. DeLoach, S.A.: Multiagent systems engineering of organization-based multiagent systems. ACM SIGSOFT Software Engineering Notes 30(4), 1–7 (2005)
5. Dignum, V., Dignum, F., Meyer, J.J.: An agent-mediated approach to the support of knowledge sharing in organizations. Knowledge Engineering Review 19(2), 147–174 (2004)
6. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Proceedings of the Workshop on Agent Oriented Software Engineering. pp. 214–230 (2003)
7. Fornara, N., Colombetti, M.: Operational specification of a commitment-based agent communication language. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems. pp. 536–542 (2002)
8. Fornara, N., Viganò, F., Colombetti, M.: Agent communication and artificial institutions. Autonomous Agents and Multi-Agent Systems 14(2), 121–142 (2007)
9. Günay, A., Yolum, P.: Detecting conflicts in commitments. In: Sakama, C., Sardina, S., Vasconcelos, W., Winikoff, M. (eds.) Declarative Agent Languages and Technologies IX. LNAI, vol. 7169, pp. 51–66. Springer (2011)
10. Günay, A., Yolum, P.: Constraint satisfaction as a tool for modeling and checking feasibility of multiagent commitments. Applied Intelligence (2013), to appear.
11. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: Programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering 1(3/4), 370–395 (2007)
12. Kolp, M., Giorgini, P., Mylopoulos, J.: Multi-agent architectures as organizational structures. Autonomous Agents and Multi-Agent Systems 13(1), 3–25 (2006)
13. Mallya, A.U., Huhns, M.N.: Commitments Among Agents. IEEE Internet Computing 7, 90–93 (2003)
14. Odell, J.J., Parunak, H.V.D., Brueckner, S., Sauter, J.: Temporal aspects of dynamic role assignment. In: Giorgini, P., Müller, J.P., Odell, J. (eds.) Agent-Oriented Software Engineering IV. LNCS, vol. 2935, pp. 201–213 (2004)
15. van Riemsdijk, M.B., Hindriks, K.V., Jonker, C.M., Sierhuis, M.: Formalizing organizational constraints: a semantic approach. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems. pp. 823–830 (2010)
16. Sensoy, M., Norman, T.J., Vasconcelos, W.W., Sycara, K.: OWL-POLAR: A framework for semantic policy representation and reasoning. Web Semantics: Science, Services and Agents on the World Wide Web 12–13, 148–160 (2012)
17. Singh, M.P.: An ontology for commitments in multiagent systems. Artificial Intelligence and Law 7(1), 97–113 (1999)
18. Telang, P., Singh, M.: Enhancing tropos with commitments. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 417–435 (2009)
19. Torroni, P., Chesani, F., Mello, P., Montali, M.: Social commitments in time: Satisfied or compensated. In: Baldoni, M., Bentahar, J., van Riemsdijk, M.B., Lloyd, J. (eds.) Declarative Agent Languages and Technologies VII. LNCS, vol. 5948, pp. 228–243 (2010)
20. Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J.: Normative conflict resolution in multiagent systems. Autonomous Agents and Multi-Agent Systems 19(2), 124–152 (2009)

21. Wan, F., Singh, M.P.: Commitments and causality for multiagent design. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems. pp. 749–756 (2003)
22. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: Applying event calculus planning using commitments. In: Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems. pp. 527–534 (2002)
23. Yolum, P., Singh, M.P.: Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. Annals of Mathematics and Artificial Intelligence 42(1-3), 227–253 (2004)
24. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Transactions on Software Engineering Methodology 12(3), 317–370 (2003)