Distributed Coordination of an Agent Society Based on Obligations and Commitments to Negotiated Agreements

Jiangbo Dang, Devendra Shrotri, and Michael N. Huhns

University of South Carolina
Department of Computer Science and Engineering
Columbia, SC 29208 USA
{dangj,shrotri,huhns}@engr.sc.edu

Summary. This chapter discusses coordination from a commitment basis. Typically, commitments are established via a process of negotiation between the parties—the debtor and creditor—involved in the commitment. We define obligations to be those commitments, sometimes termed norms or social commitments, without a clearly identifiable creditor. The establishment of a commitment occurs in response to the adoption of a goal or the acceptance and performance of a task. Using a service-oriented computing (SOC) context, we describe an efficient negotiation process for establishing commitments. We then show how commitments and obligations can be used to monitor and control the aggregate behavior of a group of agents to yield coordinated progress towards the agents' overall objective.

1 Introduction

In service-oriented multiagent environments, the participating agents are distinguished by the services they provide, the services they seek and the negotiated service agreements to which they commit. As an example, participants in typical real-world business environments interact by exchanging goods and providing services to each other. In seeking and providing services, they form associations by negotiating on service agreements, make promises, commit to products, quality, and service levels, fulfill what they promised, and attempt to achieve their intended goals.

The coherent behavior of systems in such an environment is governed by interactions among the agents, and we believe that commitments and obligations are the proper abstraction to characterize the interactions for monitoring and control of the systems. We hypothesize that a commitment is an appropriate abstraction for managing, monitoring, and assuring large-scale distributed coordination.

1.1 The Coordination Problem

Coordination is a ubiquitous problem for distributed systems, where the objective is to achieve coherent and efficient operation while making rapid progress toward system-wide goals. The problem can appear in many forms, ranging from managing access to shared resources to engaging the expertise of multiple participants in reaching an overall goal.

In this chapter, we make several assumptions to limit the scope of the coordination problem that we are considering. First, we assume that the problem can be cast in terms of a known set of agents performing a dynamic set of tasks to reach a globally known goal. Second, we assume that there might be thousands of individual tasks that need to be coordinated, but not millions and not just a few. Third, we assume that the time and resources needed to perform an individual task are generally available (not scarce). Fourth, we assume that the time needed to perform an individual task is much less than the time needed to reach the goal, allowing time for tasks to be created, modified, redone, cancelled, or reassigned. The individual tasks might be discrete (e.g., the task to remove an obstacle) or continuous (e.g., the task to prevent the introduction of an obstacle). Fifth, we assume that the tasks are organized into a workflow, which may evolve as commitments are made, resources are expended, and tasks are decomposed and performed. Sixth, we assume that the agents are each aware of and have accepted the global goal, but are otherwise self-interested and autonomous. (Sen [21] has shown that societies of purely selfless agents are inefficient.) Finally, we assume that the environment where the coordinated behavior takes place has the following characteristics.

1.2 A Service-Oriented Computing Environment

A typical real-world multiagent service-oriented environment is partially observable, stochastic, sequential, dynamic, and continuous. This environment consists of two classes of agents: participating agents and non-participating agents.

The participating agents either play the role of a service provider or that of a service seeker. These service providers and service seekers negotiate and reach a service agreement. Negotiation is a process by which agents communicate and compromise to reach an agreement on matters of mutual interest while maximizing their utilities. We believe that these negotiated agreements associate or bind these participating agents with each other and that this association can be best represented as the binary relationship of commitments.

In addition to this class of participating agents, there is another class of non-participating agents in this environment; these are agents that act more like impartial arbiters. The nonparticipating agents provide the context to a commitment relationship, termed a Sphere of Commitment (SoCom) [25]. Every agent in the environment is autonomous, hence at any point in time any

agent may choose to either abide by its commitment or stray from it. The non-participating arbiters can be used to capture a participating agent's behavior with regard to its commitments. Historical information about a participating agent's behavior can be utilized to measure its commitment adherence for future interactions.

We assume that the service providers and service seekers have already identified each other. How service seekers and service providers locate each other, how they identify compatible providers or seekers and what structure of communication and protocol they use are questions beyond the scope of this chapter.

It is further assumed that in this commitment-driven service-oriented environment the partial view that an agent has is governed solely by the commitment relationships in which it participates. In other words, agents have knowledge of other agents with whom they are associated via commitment relationships. Furthermore, it is assumed that the knowledge about a commitment relationship is governed by commitment operations, i.e., an agent has knowledge about a commitment association only through operations that affect that commitment. For example, when a service-seeking agent and a service-providing agent participate in a commitment relationship, each will have knowledge of the other agent's commitment actions and each will have knowledge of when the commitment gets created, fulfilled, revoked, etc. However, knowledge such as how that commitment is fulfilled, why it was not fulfilled, or why it was canceled is not available to the participating agents.

The typical environment for commitments is dynamic and nondeterministic; hence its temporal dimension is best represented as branching time. The underlying temporal parameter moves forward and branches out like a tree. Also, an agent's beliefs, desires and intentions define its internal state of mind. We use Rao and Georgeff's BDI framework [19], Emerson's CTL framework [5], Singh and Huhns's definitions for commitments and operations on them [24], and Shrotri and Huhns's definitions of commitments in terms of BDI [22].

2 Modeling and Representation

Goals are achieved via interleaved phases of planning and execution. Planning, which may be done by humans or by the agents responsible for goal achievement, yields sets of executable tasks and the dependencies among them. The dependencies will be primarily temporal, e.g., one task must be performed before another, but they also might be conditional, e.g., one task must be performed only if another fails. The resultant ordering of the tasks is a workflow, which can exist at several levels of generality as tasks are either aggregated into composite tasks or decomposed into subtasks.

Each task has associated with it a number of attributes that are used by an agent to perform the coordination. Each task will have a latest finish time

(deadline) by which the task must be completed, earliest start time, expected duration, priority, and worth. Temporal values allow the agent to reason about when a task can be performed. A task's priority and worth represent the value of the task to the goal. Task assignment to a particular agent leads to determination of values of several additional attributes: expected quality of a result, expected cost, and expected risk.

Tasks are associated with agents via a process of negotiation as described in Section 4. The resultant assignments, especially when dependent tasks are assigned to different agents, are monitored via commitments. A commitment is a well-defined data structure with an algebra of operations that have a formal semantics. A commitment has the form $C(a; b; p; G)$, where a is its creditor, b is its debtor, p the condition the debtor will bring about, and G the organizational context for the given commitment. The operations on commitments are *create*, *discharge*, *delegate*, *assign*, *cancel*, and *release*. Commitments capture the dependencies among the agents with regard to the tasks.

Note that tasks, interactions, and commitments are not completely known a priori, but can enter the system dynamically. We do not assume that each agent knows a priori all the possible tasks that it might be asked to perform. When it has been assigned and authorized to perform a task, then its commitment is formed. The dynamic nature of task assignment necessitates the ability of the system to reason about commitments in a principled way, thus enabling the agents to have optimized ways of dynamically forming and breaking commitments as new tasks enter the system.

Explicit representation of commitments helps coordination in the following two ways:

1. Commitment is an abstraction that explicitly refers to inter-agent dependencies, either through task temporal dependencies, task preconditions, or through contingencies (i.e., alternative ways of performing a task), thus allowing agents to recognize focus points in the revision process where coordination with other agents is needed; focusing the distributed search this way benefits the efficiency of coordination.
2. During the process of revising its local plan, an agent first tries to revise task timings that do not involve commitments; this heuristic modularizes the revision as much as possible, making it more scalable.

The following structures for tasks, goals, and task performers (agents) are consistent with the above assumptions, and also consistent with the TAEMS formulation [9]:

Task: a unit of work to be performed in furtherance of an overall goal

- duration (time needed to perform)
- effort required
- deadline (when task must be finished)
- resources required (consumable and non-consumable)
- utility, including cost and quality

- revocable?
- compensation (if result of task must be revoked and it is not revocable)

Agent: a performer of one or more tasks

- capabilities, including access to resources
- limitations

Goal: an overall mission or objective to be achieved

- workflow or goal decomposition

3 Negotiated Commitments

In supply chains, e-commerce, and Web services, the participants negotiate contracts and enter into binding agreements with each other by agreeing on functional and quality metrics of the services they request and provide. The functionality of a service is the most important factor, especially for discovering services. Once discovered, however, services are engaged, composed, and executed by the participants' negotiating over QoS metrics to maximize their profits.

Negotiation is a process by which agents communicate and compromise to reach agreement on matters of mutual interest while maximizing their individual utilities. Negotiation for QoS-aware services is currently limited to primitive QoS verification methods or sorting and matching algorithms. We extend current techniques by presenting an optimal negotiation procedure that considers the cost to reach an agreement for QoS-aware service engagement and contracting.

3.1 Research Issues

Semantic Web services, as envisioned by Berners-Lee, are intended to be applied not statically by developers, but dynamically by the services themselves through automatic and autonomous selection, composition, and execution. Dynamic selection and composition first require service requestors to discover service providers that satisfy the requestors' functional requirements. Second, the requestors and providers negotiate non-functional requirements (QoS), including cost and qualities such as response time, accuracy, and availability.

In general, negotiation is the technique for reaching mutually beneficial agreement through communication among agents. Negotiation in QoS-aware services involves a sequence of information exchanges between parties to establish a formal agreement among them, whereby one or more parties will provide services to one or more other parties. The agreement typically involves QoS issues [26]. By QoS, we refer to the non-functional properties of services, such as performance, cost, reliability, and security. To meet the requirements of service requestors, multiple issues, including both functional and non-functional,

need to be taken into account during service advertisement, discovery, composition, and delivery. Preist [17] discussed how negotiation plays an important role in reaching a service agreement for a service.

Current standards for Web services do not support QoS negotiations. As a result, several researchers have attempted to merge negotiation from the MAS domain into QoS-aware Web services. Ran [18] proposes to enrich current UDDI registries by extending the SOAP message format and the UDDI data structures to describe QoS information. Petrone [16] proposed a conversation model to enrich the communication and coordination capabilities of Web services by adapting agent-based concepts to the communications among services and users. In [18, 8] researchers extend the Web service model by introducing a third party broker, certifier, or QoS manager for QoS enactment and enforcement. Their work includes simple QoS verification or match algorithms and permission for the broker to negotiate and make decisions on behalf of the requestors. This is problematic, especially in situations where price and payment issues are involved.

Maximilien and Singh [13] propose a Web service agent framework (WSAF) with a QoS ontology. When a service consumer needs to use a service, WSAF will create a service agent that can capture a consumer's QoS preference and select the most suitable service.

Negotiating for services involves both functional and non-functional issues. We can not apply existing multiple-issue negotiation models to service negotiation and contracting directly, because existing models often make the limiting assumption that agents know the private information of their opponents, and their theoretic models do not take computational cost into consideration. Therefore, these models do not fit the environment of on-line QoS negotiation for services.

Many researchers have investigated multiple-issue negotiation [10, 14, 6]. Fatima et al. [6] presented an optimal agenda and procedure for two-issue negotiation by introducing two negotiation procedures: *issue-by-issue negotiation* and *package deal*. For n-issue negotiation where $n > 2$, which is common in negotiation over QoS issues, the computational cost to reach a package deal might exceed the benefits obtained by optimizing the participants' utilities. By considering both utility optimization and computational efficiency, Dang and Huhns [2] propose the *coalition deal* that is suitable for multiple-issue negotiation, especially in the case of QoS negotiation for services.

In [10] agents know the incomplete preference information about their opponents and exploit this information to reach negotiation efficiency. This work is thus limited to cooperative negotiation, where agents care about not only their own utilities, but also equity and social welfare, which is not common in most application environments.

The outcome of multiple-issue negotiation depends on not only strategies, but also the procedure by which issues will be negotiated. Different procedures yield different outcomes. Based on an incomplete information assumption, Fatima et al. [6] discussed two procedures for multiple issue negotiation: *issue-*

by-issue and *package deal*. For two-issue negotiation, they determined the equilibrium strategy for these procedures and analyzed the optimal agenda and procedure. Since their analysis is limited to two-issue negotiation, they concluded that the package deal is the procedure that provides agents with optimal utilities; they did not address the computational cost. However, the computational cost becomes crucial when more issues are involved. We focus on the optimal strategy of efficiently negotiating multiple QoS issues to reach an agreement that gives both the requestor and the provider their maximum utilities.

We hypothesize that a coalition deal negotiation can overcome these limitations. As shown in [2], this is the optimal strategy for service negotiation over multiple issues when computation cost is considered. The coalition deal mitigates the computational cost problem by making a trade-off between optimal utility and computational efficiency. This chapter makes four contributions to the advancement of QoS-aware service negotiation and contracting. First, it describes the coalition deal negotiation for reaching utility optimization and computational efficiency. Second, it generalizes the analysis of an optimal negotiation procedure to multiple-issue negotiation over more than two issues. Third, it tailors negotiation components to fit QoS-aware negotiation. Fourth, it focuses on agents' own information; no agent has any information, such as reserve price, about its opponent.

3.2 QoS Scenario for Negotiation

In order to illustrate the coalition deal for n-issue negotiation over the QoS metrics of a service, we present a motivating scenario. Consider how one site, a requestor, might arrange to get a stock quote from a service provider. In this scenario, a service requestor a (a.k.a. the creditor if a commitment is established) locates a *GetStockQuote* Web service provided by b (a.k.a. the debtor if a commitment is established) that meets its functionality requirements. The *GetStockQuote* service takes the requestor's inquiring stock number as an input and a currency symbol as an argument, and provides a stock quote.

During the procedure of service selection, QoS becomes an important factor to both a and b. Before reaching a service contract, they need to negotiate over (1) **payment method** indicates the way a user pays for inquiries (e.g., pay per inquiry and pay for bundle); (2) **inquiry cost** indicates the cost per inquiry; (3) **update interval** represents how often the stock quote information is updated; (4) **response time** is the round-trip time between sending an inquiry and receiving the response; (5) **availability** represents the probability that this service is available and ready for immediate use; (6) **service plan cost** is the plan cost for service with agreed-upon quality.

Agents a and b could negotiate each issue individually using issue-by-issue negotiation, but some issues are related to each other and isolating them will degrade the utility and increase the risk of a conflict deal. A package deal

allows both a and b to make trade-offs among all six issues, but the computation is intractable with exponential cost. By using a coalition deal, we can partition six issues into two partitions where strongly related issues are in the same partition. For example, payment method, inquiry cost and update interval belong to partition one, while response time, availability, and service plan cost belong to partition two. a and b can negotiate two partitions in parallel, where each partition is settled as a package deal and independently of other partitions. By pursuing a coalition deal, agents can reach a service agreement while optimizing their utilities with efficient computation. The coalition deal is explored in the next section.

4 Coalition Deal Negotiation

A service is what an agent performs when it works on and completes a task. Negotiating for tasks has four components: (1) a negotiation set, which represents the possible proposal space for both functionality and QoS metrics of a service; (2) a protocol, which defines the legal proposals that an agent can make, as defined in a service description and constrained by negotiation history; (3) a strategy, which determines what proposals the agents will make, decided by an agent's private preference and affected by the service discovery result; and (4) a rule enforced by a mediator to determine when a deal has been struck and what the agreement is. We focus on the negotiation procedure of multiple-issue negotiation for services, which adopts Rubinstein's alternating offers protocol.

As described in our motivating scenario, let a denote the service requestor and b the service provider. From a service viewpoint, a has a task and tries to find a service to perform it. From a task viewpoint, b has a service and is capable of fulfiling certain tasks, so b tries to find a task to work on. We assume that each agent only has complete information about its own negotiation parameters. For some private information, such as the opponent's deadline, we can use the negotiation protocol in [20] to make truth-telling about a negotiation deadline the dominant strategy. We use S_a (S_b) to denote the set of negotiation parameters for agent a (b) and describe the negotiation model similarly to that in [6].

4.1 Single-Issue Negotiation

Consider a and b negotiating over an issue set I, where $I = A$ and A is one issue, say, the inquiry price. The agents' parameter sets are defined as

$$\begin{aligned} S_a &= \left\langle P_a^A, U_a^A, T_a^A, \delta_a^A \right\rangle \\ S_b &= \left\langle P_b^A, U_a^A, T_b^A, \delta_b^A \right\rangle \end{aligned} \tag{1}$$

where P_a^A, U_a^A, T_a^A, and δ_a^A denote agent a's reserve price over issue A, utility function over issue A, bargaining deadline, and time discounting factor,

respectively. Agent b's negotiation parameters are defined analogously. The agents' utilities at price p and at time t are defined as in [6]:

$$U_a^A(p,t) = \begin{cases} (P_a^A - p)(\delta_a^A)^t & \text{if } t \leq T_a \\ 0 & \text{if } t > T_a \end{cases}$$
$$U_b^A(p,t) = \begin{cases} (P_b^A - p)(\delta_b^A)^t & \text{if } t \leq T_b \\ 0 & \text{if } t > T_b \end{cases} \quad (2)$$

The value for δ_a^A is > 1 when agent a is patient and gains utility with time, < 1 when a is impatient and loses utility with time, and $= 1$ when a's utility is independent of time. The same holds for agent b. We only consider $\delta_a^A \leq 1$, which is common in a service-oriented environment.

In single-issue negotiation, the preferences of the agents are symmetric, in that a deal which is more preferred from one agent's point of view is guaranteed to be less preferred from the other's point of view. At the beginning of the negotiation, an agent makes an offer that gives it the highest utility and then incrementally concedes as the negotiation progresses by offering its opponent a proposal that gives it lower utility. Because of the symmetric preference of agents, agents have to concede to offer deals that are more likely to be accepted by their opponents if they prefer reaching an agreement to the conflict deal. An outcome is *individual rational* if it gives an agent a utility that is no less than its utility from the conflict outcome. The maximum possible utility that agent a (b) can get from an outcome over issue A is denoted $U_{max,a}^A$ ($U_{max,b}^A$) and it is individual rational to both agents.

Agent a's strategy (denoted σ_a) is a mapping from the previous negotiation proposals $p_{a,t'<t}$ and S_a to the action $Ac_{a,t}$ that it takes at time t: $\sigma_a : p_{a,t'<t} \times S_a \rightarrow Ac_{a,t}$ is defined as:

$$Ac_{a,t} = \begin{cases} \text{Quit} & \text{if } t \geq T_a \\ \text{Accept} & \text{if } U_a^A(p_{b,t}^A, t) \geq U_a^A(p_{a,t+1}^A, t+1) \\ \text{Offer } p_{a,t+1}^A \text{ at t+1, otherwise.} & \end{cases} \quad (3)$$

where $p_{b,t}^A$ is the offer made by agent b over issue A at time t. $p_{a,t+1}^A$ is defined analogously. Let $P_{a,t}^A$ denotes the offer that agent a makes at time t in equilibrium, drawn from agent a's equilibrium strategy. $P_{a,t}^A$ is determined by:

$$P_{a,t}^A = (U^{-1})_a^A((1 - y_{a,t}^A) \times U_{max,A}^A) \quad (4)$$

where $y_{a,t}^A$ is agent a's yield-factor [6] at time t.

4.2 Multiple-Issue Negotiation

We next consider multiple-issue negotiation over issue set I of k issues, where $I = \{I_1, I_2, \ldots, I_k\}$. The agents' parameter sets can then be defined as follows:

$$S_a = \langle P_a^I, U_a^I, T_a, \delta_a \rangle$$
$$S_b = \langle P_b^I, U_b^I, T_b, \delta_b \rangle \quad (5)$$

where $P_a^I = \{P_a^i \mid i \in I\}$ denotes agent a's reserve prices over I and P_a^i denotes a's reserve price over issue i, $U_a^I = \{U_a^i \mid i \in I\}$ denotes agent a's utility functions over I, T_a, and δ_a denote agent a's bargain deadline and discount factor. Agent b's negotiation parameters are defined analogously. We assume that an agent's utility from issue set I is the sum of its utilities from all issues, then we have:

$$U_{a,t}^I = \sum_{i \in I} U_{a,t}^i,\ U_{max,a}^I = \sum_{i \in I} U_{max,a}^i \quad (6)$$

Two procedures for multiple-issue negotiation have been discussed [6]: *package deal* and *issue-by-issue negotiation*. For a package deal, an offer includes a value for each issue under negotiation. Thus for k issues an offer is a package of k values, one for each issue. This allows trade-offs to be made between issues. Agents can either accept a complete offer or reject a complete offer. For issue-by-issue negotiation, each issue is settled separately and an agreement can take place either on a subset of issues or on all of them.

We first describe the procedure for a package deal. Assume that the agents use the same protocol as described in the previous section for single issue negotiation, but instead of making an offer on a single issue, an agent offers a set of offers (an offer consists of a set of values for issues from I, all of which give it equal utility). This is because when there is more than one issue, an agent can make trade-offs across issues, resulting in a set of offer sets, all of which give it equal utility. As an example, Figure 1(a) illustrates the utility for 4-issue negotiation with two package deals of two issues each. Here, we focus on the utility frontiers for the issue set $I = \{A, B\}$. In this figure the agents' utilities are measured along two axes, and the origin represents the conflict outcome. The segment AA' is the utility frontier for issue A and BB' that for issue B. The utility frontier for I is $A''B''C''D''$ (i.e., the sum of all possible utilities from issue A and issue B). The points along LL' are pairs of values for issue A and issue B that give equal utility to agent a, but different utilities to agent b. L is Pareto-optimal since it is the only one, from all possible pairs along LL', that lies on the segment $A''B''C''D''$. Because an agent does not know its opponent's utility function, it does not know which of the possible pairs along LL' is Pareto-optimal. Therefore, agent a makes trade-offs across A and B, and then offers a set of pairs that correspond to points along LL'. The slopes of segments AA' and BB' represent how the agents value the issues A and B. Agent a is said to value issue A more (less) than b if the increase in a's utility for a unit change for issue A is higher (lower) than the increase in b's utility for a unit change for issue A. Therefore, the slope of the segment represents the agents' utility preference for a issue, and is named comparative interest in [6].

We define $P^I_{a,t} = \left\langle P^{I_1}_{a,t}, P^{I_2}_{a,t}, \ldots, P^{I_k}_{a,t} \right\rangle$ as agent a's current optimal utility offer for agent b that satisfies $U^I_b(P^I_{a,t}) = \mathbf{argmax}\ U^I_b(p^I_{a,t})$ where $p^I_{a,t} \in P_t({U^I_{a,t}}')$ and $P_t({U^I_{a,t}}') = \left\{ \left\langle p^{I_1}_{a,t}, \ldots, p^{I_k}_{a,t} \right\rangle \mid U^I_a(p^{I_1}_{a,t}, \ldots, p^{I_k}_{a,t}, t) = {U^I_{a,t}}' \right\}$. Therefore, agent a's action $Ac_{a,t}$ for the package deal procedure is defined as

$$Ac_{a,t} = \begin{cases} \text{Quit} & \text{if } t \geq T_a \\ \text{Accept} & \text{if } U^I_a(P^I_{b,t}, t) \geq {U^I_{a,t+1}}' \\ \text{Offer } P_{t+1}({U^I_{a,t+1}}') \text{ at t+1,} & \text{otherwise.} \end{cases} \tag{7}$$

Agent a is playing its equilibrium strategy if ${U^I_{a,t+1}}' = (1 - y^I_{a,t+1})U^I_{max,a}$, where $U^I_{max,a}$ is the maximum possible utility agent a can get from issue set I [6]. The equilibrium strategy for agent b is defined analogously. We now turn to the issue-by-issue procedure. Agent a's action $Ac_{a,t}$ is defined as follows and proved in [6]:

$$Ac_{a,t} = \begin{cases} \text{Quit} & \text{if } t \geq T_a \\ \text{for issue } i \in I & \begin{cases} \text{Accept if} & U^i_{a,t}(p^i_{b,t}) \geq U^i_{a,t}(p^i_{a,t+1}) \\ \text{Offer } p^i_{a,t+1} & \text{otherwise.} \end{cases} \end{cases} \tag{8}$$

where $p^i_{a,t}$ satisfies the constraints for the equilibrium strategy described in Section 4.1.

4.3 Coalition Deal Negotiation

We discussed two negotiation procedures: issue-by-issue negotiation and package deal. The outcome of negotiation depends on different negotiation strategies and procedures. For our example *GetStockQuote*, issue-by-issue negotiation and package deal may produce different negotiation outcomes and give agents different utilities. We assume that both a and b prefer agreement to the conflict deal for every issue. In issue-by-issue negotiation, for example, agents agree on the issue of payment method with pay for bundle, and they also reach agreement that p is the inquiry cost. Since agents negotiate these issues independently, it is possible that p is too high to a if a chooses to pay for the bundle as its payment method. That means issue-by-issue negotiation may degrade agents' utilities. In package deal negotiation, agents can make a set of values over six issues and propose offers and counter offers by crossing over issues. Agents may combine different payment methods with different inquiry costs to reach mutually beneficial agreement over the two issues. However, the package deal also leads to an exponential growth in the computation cost to generate the offer sets. Most tasks (services), of course, are more complex than our example, and when they are composed this computation problem is significant. To make negotiating for tasks both optimum and efficient, we introduce the coalition deal.

Definition and Negotiation Model

We define coalition deal negotiation, which makes a better trade-off between issue-by-issue negotiation and the package deal procedure, to provide agents approximately optimized utilities with minimized computation costs.

Definition 1. *For a coalition deal, all negotiation issues are partitioned into disjoint partitions and each partition is negotiated independently of other partitions. Like the package deal, issues inside the same partition are negotiated as a whole and an offer includes a value for each issue in this partition. Furthermore, there is more than one partition in a coalition deal and at least one partition that has more than one issue.*

From this definition, we can see that issue-by-issue negotiation is a specific case of a coalition deal where one issue per partition. The package deal is also a coalition deal, where there is only one partition for all issues. Coalition deal negotiation provides (a) better utility, (b) less computational cost, (c) more flexible negotiation, and (d) better management of QoS metrics for services.

Consider multiple-issue negotiation with issue set I of k issues, where $I = \{I_1, I_2, \ldots, I_k\}$. From the definition, we know that there exists a partition IP of size s over I, where $IP = \{IP_j \mid 1 \leq j \leq s\}$. IP satisfies the constraint: $\forall 1 \leq m \leq s, 1 \leq n \leq s, m \neq n$, we have $IP_m \cap IP_n = \emptyset$ and $\cup_{j \in IP} \cup_{i \in j} i = I$. Similarly, agents' parameter sets can be defined as follows:

$$\begin{aligned} S_a &= \left\langle P_a^{IP}, U_a^{IP}, T_a, \delta_a \right\rangle \\ S_b &= \left\langle P_b^{IP}, U_b^{IP}, T_b, \delta_b \right\rangle \end{aligned} \tag{9}$$

where $P_a^{IP} = \{p_a^i \mid i \in j, j \in IP\}$ denotes agent a's reserve prices set over partitions of issue set I and p_a^i denotes a's reserve price over issue i, which belongs to partition j, $U_a^{IP} = \{U_a^i \mid i \in IP\}$ denotes agent a's utility functions over partition IP where U_a^i denotes agent a's utility function over one partition i from IP, T_a and δ_a denotes agent a's bargaining deadline and discount factor. Agent b's negotiation parameters are defined similarly. An agent's utility from partition IP of issue set I is the sum of its utilities from all partitions, so then we have

$$U_a^{IP} = \sum_{j \in IP} U_a^j = \sum_{j \in IP} \sum_{i \in j} U_a^i,\ U_{max,a}^I = \sum_{j \in IP} \sum_{i \in j} U_{max,a}^i \tag{10}$$

For a coalition deal, each partition is negotiated independently of other partitions. An agreement can take place either on some or all of the partitions. For each partition, an offer includes a value for each issue inside the partition that would be the same as the package deal for this partition. This allows trade-offs to be made between issues inside the partition. An agreement has to take place either on all or none of the issues inside the partition.

For each partition, we assume the agents use the same protocol as for the package deal, but instead of making a set of offers over issue set I, an agent

makes a set of offers over issues from this partition. An agent can make trade-offs only across issues in the same partition, resulting in a set of offer sets, all of which give it equal utility. As an example, Figure 1(a) illustrates the utility frontiers for issue set I where $I = \{A, B, C, D\}$. There exists a partition IP for I where $IP = \{\{A, B\}, \{C, D\}\}$. Let $IP_1 = \{A, B\}$, and $IP_2 = \{C, D\}$. The utility frontier for IP_1 is $A''B''C''D''$ and the utility frontier for IP_2 is $S''T''V''U''$. For IP_1, the points along LL' are pairs of values for IP_1 that give equal utilities to agent a but different utilities to agent b. The points along RR'' are pairs of values for IP_1 that give equal utilities to agent b but different utilities to agent a. The utility for IP is the sum of the utilities from IP_1 and IP_2 after these partitions are negotiated independently. If we only consider the optimal outcome from both negotiations over IP_1 and IP_2, All optimal outcomes for IP_1 lie on the segment $MB''K$, and all optimal outcomes for IP_2 lie on the segment $XT''Y$ as we described for the package deal. Therefore, the possible utility frontier for IP is represented by region $OM''P''QQ'P$ in Figure 1(b). For a partition IP_i of k_i issues, we define $P_{a,t}^{IP_i} = \left\langle P_{a,t}^{IP_i(1)}, \ldots, P_{a,t}^{IP_i(k_i)} \right\rangle$ as agent a's current optimal utility offer for agent b that satisfies $U_b^{IP_i}(P_{a,t}^{IP_i}) = \textbf{argmax}\ U_b^{IP_i}(p_{a,t}^{IP_i})$, where $p_{a,t}^{IP_i} \in P_t(U_{a,t}^{IP_i})$ and $P_t(U_{a,t}^{IP_i\prime}) = \left\{ \left\langle p_{a,t}^{IP_i(1)}, \ldots, p_{a,t}^{IP_i(k_i)} \right\rangle \mid U_a^{IP_i}(p_{a,t}^{IP_i(1)}, \ldots, p_{a,t}^{IP_i(k_i)}) = U_{a,t}^{IP_i\prime} \right\}$. $P_{t+1}(U_{a,t+1}^{IP_i\ \prime})$ is defined analogously. For a coalition deal, each partition is considered using the package deal negotiation protocol. Agent a's action $Ac_{a,t}$ for the coalition deal procedure is defined as follows:

$$Ac_{a,t} = \begin{cases} \text{Quit} & \text{if } t \geq T_a \\ \text{Accept package deal for } IP_i & \text{if } U_a^{IP_i}(P_{b,t}^{IP_i}) \geq U_{a,t+1}^{IP_i\ \prime} \\ \text{Offer } P_{t+1}(U_{a,t+1}^{IP_i\ \prime}) \text{ for } IP_i \text{ at t+1,} & \text{otherwise.} \end{cases} \quad (11)$$

Similarly, we define agent a as playing its equilibrium strategy for the package deal over a partition if $U_{a,t+1}^{IP_i\ \prime} = (1 - y_{a,t+1}^{IP_i}) U_{max,a}^{IP_i}$, where $U_{max,a}^{IP_i}$ is the maximum possible cumulative utility agent a can get from partition IP_i. The equilibrium strategy for agent a and agent b over other partitions is defined analogously.

Coalition Deal Utility

In previous sections, we discussed three different negotiation procedures: issue-by-issue, package deal, and coalition deal. These three procedures can generate different outcomes, and consequently give different utilities to the agents. To decide the optimal procedure that gives the agents highest utilities, we need to compare agents' utilities from these procedures for n-issue negotiation. Fatima et al. [6] introduced the zone of agreement for individual issues where both agents prefer agreement over no deal. An issue has a zone of agreement if its utility frontier lies in quadrant $Q1$. We discuss the common scenario of

service-oriented computing (SOC) in which both agents are *individual rational* (i.e., all issues have a zone of agreement ensured by the service description and the discovery procedure).

Lemma 1. *each agent's utility from the package deal is no worse than its utility from issue-by-issue negotiation for two-issue negotiation.*

Lemma 1 has been proven in [6]. In a service-oriented environment, there are many issues concerning functionality and quality that need to be negotiated during service engagement. Can we generalize Lemma 1 to cover more than two? Here, we compare agents' utilities from package deal and issue-by-issue negotiation for n-issue negotiation.

Theorem 1. *Each agent's utility from the package deal is no worse than its utility from issue-by-issue negotiation for n-issue negotiation, where* $n > 2$.

Theorem 1 has been proven in [2] by induction. From this theorem, we know that a package deal gives agents better utilities than issue-by-issue negotiation does. As stated in the previous section, a coalition deal provides approximately optimized utilities to agents. Then we prove that a coalition deal give agents utilities better than issue-by-issue negotiation does.

Theorem 2. *Each agent's utility from a coalition deal is no worse than its utility from issue-by-issue negotiation for n-issue negotiation, where* $n > 2$.

Theorem 2 has been proved by combining Theorem 1 and our assumption of additive utilities [2]. Both package deal and coalition deal give agents utilities better than issue-by-issue negotiation does. The remaining question is which procedure, package deal or coalition deal, gives agents better utilities. To answer this question, we first prove that the package deal gives agents utilities better than a coalition deal of two partitions.

Lemma 2. *Each agent's utility from the package deal is no worse than its utility from i-by-j negotiation for n-issue negotiation, where* $i \geq 1, j \geq 1, n > 2$, *and* $i + j = n$.

We have proven that the package deal gives agents utilities better than a coalition deal of two partitions for n-issue negotiation in [2]. For QoS negotiation for tasks, we need to extend Lemma 2 to the coalition deal with more than two partitions.

Theorem 3. *Each agent's utility from a coalition deal is no better than its utility from the package deal for n-issue negotiation, where* $n > 2$ *[2].*

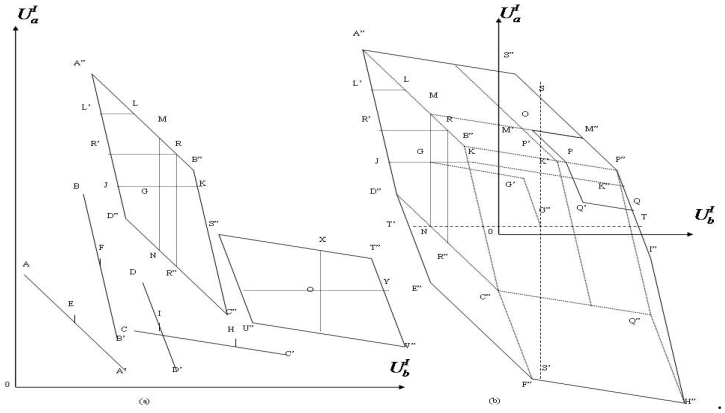

Fig. 1. Agents' utilities for 4-issue negotiation

Coalition Deal Efficiency

From Theorems 1, 2 and 3, we know that each agent's utility from the package deal is better than its utility from a coalition deal and issue-by-issue negotiation. Therefore, we should choose the package deal negotiation to maximize agents' utilities. However, we need to consider the computational costs, which can be the primary factor when negotiating for tasks.

Given an issue set $I = \{I_1, I_2, \ldots, I_n\}$ and a partition $IP = \{IP_1, IP_2, \ldots, IP_k\}$ over I, we define the unit computational cost for generating a price value for one issue as a constant. We assume that every issue in issue-by-issue negotiation can be negotiated in parallel and every partition in a coalition deal can also be negotiated in parallel. To compare the computational efficiency, we only need to compare the computational cost of generating an offer in each round of three different procedures. If we suppose agents need almost the same rounds of negotiation to reach an agreement in these three negotiation procedures, we can compare their computational costs by comparing the cost of generating an offer in each round.

An n-issue negotiation can be viewed as a distributed search through an n-dimensional space, where each issue has a separate dimension associated with it. In issue-by-issue negotiation, each issue is negotiated separately, Based on the above equilibrium strategy, agents will compute a value for each issue. Therefore, the computational cost in one round is $O(n)$, where n is the size of the issue set. In the package deal, an offer is a set including a value for each issue under negotiation. In each round, an agent can make trade-offs across all n issues to offer a set of offers that give it the same utilities. In the worst case, the computational cost in one round is $O(m^n)$, where we assume each issue may have m possible values.

The computation problem of generating an offer set is equivalent to searching in an n-dimensional space for all combinations of possible distributions of

given utility value among all n issues with a utility constraint. This problem is intractable and takes $O(m^n)$ time in the worst case. Even worse, we have to solve this problem every round during the package deal negotiation procedure. It means that it will be infeasible for an agent to consider every possible offer given a utility constraint. In coalition deal negotiation, issues are partitioned into k disjoint partitions and each partition is settled independently of the other partitions. Like the package deal, issues inside the same partition are negotiated as a whole and an offer includes a value for each issue in this partition. Therefore, the computation problem is reduced to the sum of k searches where the i-th search is in an n_i-dimension space, where $n_i << n$ and $\sum_{i=1}^{k} n_i = n$. This problem takes $O(km^{n_s})$ time in the worst case, where $n_s = \textbf{argmax}\ n_i$. Moreover, we can limit the maximum size of a partition to a constant C. Therefore, the computational cost of a coalition deal reduces to $O(nm^C)$. The time complexity will be $O(m^C)$ if we have several agents, one for each partition, work together to generate a coalition deal.

In our GetStockQuote service scenario, we divide six issues into two partitions. The computational cost is 6 in each round for issue-by-issue negotiation. In package deal, agents need to search through all possible offers in a 6-dimensional space to meet the given utility constraint. The computational cost is $O(a^6)$ in the worst case, where a is the size of possible value per issue. In a coalition deal, the computational cost is $O(a^3)$ in the worst case.

Coalition Deal Negotiation for Services

With much lower computational cost than that for the package deal, agents earn greater utilities from the coalition deal than from issue-by-issue negotiation. Besides computational cost and agent utility, another advantage of the coalition deal is that it is natural to partition issues into different categories and deal with each category separately. For example, in bilateral negotiation of a labor dispute, it would be easier if money issues such as salary and bonus are negotiated in a partition separately from issues such as working condition and healthcare. Of course, it is possible that both sides would benefit if they could deal with all issues as a package, but the negotiation might become infeasible.

In QoS-aware service contracting, self-interested service agents negotiate with each other over multiple issues besides QoS attributes to reach an agreement while maximizing their utilities. The optimal negotiation strategy for the coalition deal is: (1) Agents reveal their deadlines; honesty about their real deadline is enforced by the negotiation protocol. For example, the agent that has the latest deadline will receive better payoff at the time right before its deadline. (2) Each agent estimates individually the rounds this negotiation should have before the earliest deadline. (3) Agents are identified by their time discount factors (≤ 1) from their own utility functions. Agents choose either the *Boulware* or *conceder* discount functions by mapping their discount factor to different parameters. (4) Agents compute the expected cumulative

utility by their Boulware/conceder functions and generate a set of offers, all of which give them equal utility, by crossing over multiple issues inside one partition.

Since all partitions can be negotiated in parallel and independently, the fourth steps can be executed in parallel for each partition. A service agent can breed several negotiation agents, each for one partition. These negotiation agents cooperate to reach a service agreement with distributed computation. The coalition among these negotiation agents provides the framework for a possibly more flexible negotiation procedure in the future.

5 Commitments and Obligations

Now that we have described efficient multiple-issue negotiation, in this section we define commitments and obligations and describe various operations that the participating agents can perform on them. We briefly revisit earlier formalisms of commitments and their operations [24, 22], and then define an extension useful for coordination.

5.1 Commitments

Social commitments are legal abstractions associating one entity with another. These commitments are accessible publicly and represent an interaction between two participating entities. Commitments are binary relationships that bind two agents: a "debtor agent" that promises to provide a particular service for a "creditor agent." For example, service level agreements, QoS agreements, online purchases, and service contracts are all real-world instances of commitments.

Earlier works have treated all the information about a commitment as publicly available or accessible. It is more realistic to treat some of the information as partially accessible and some as private. To do this, we refine the commitment structure in [24, 22] with the key properties of accessibility.

First, the commitment properties that are publicly accessible are

Multiagency: Commitments associate one agent with another. The agent that promises or commits to satisfying a condition is called the debtor agent and the agent that wants the condition to be fulfilled by the debtor is called the creditor agent. Each commitment is directed from its debtor to its creditor.

Scope: Commitments have a well-defined scope, also known as a Sphere of Commitment (SoCom), which gives context to the commitment.

Manipulability: Commitments are modifiable. They can become fulfilled, breached, active, suspended, or revoked, which is public information about their current status.

The following two additional parameters are not properties of a commitment per se, but represent an agent's attitude towards its commitments. These are also public.

Commitment Adherence Rating: Agents may choose to respect or ignore their commitments. For effective coordination, fulfilling promises is critical and determines an agent's reputation. A participating agent's history of commitment adherence can be captured and translated into a this rating, which represents the agent's reputation in a domain. Nonparticipating arbiters can be used to measure and maintain this parameter.

Utility Weighting: This is a numerical coefficient in the range $(0, 1]$ that represents the relative importance of the committed promise on the overall utility that is desired by the creditor agent. This commitment property is used for multiple-issue commitments. For single-issue commitments, the value is always 1. It cannot be 0, as that would represent an issue on which the creditor agent is completely agnostic.

The next (partially-accessible) property is accessible to the debtor of the commitment and to the nonparticipating arbiters, defined as follows:

Utility Coefficient: Imagine a scenario where a debtor agent makes false promises to many service seekers and then does nothing to fulfill the promises. In the real world there are checks and measures in place to discourage such behavior. The Utility Coefficient, which represents the affect of debtor's behavior on its utility, provides similar discouragement. Its value in the range $[0, 1]$ captures whether a debtor receives all of the utility associated with a commitment (value 1) or none of the utility (value 0).

Lastly, we revisit two key commitment properties [22] and redefine them as properties that represent an agent's private or internal information.

Life: Commitments have a life cycle; they are created, remain active, and at some point cease to exist. Continuous commitments are beyond the scope of this formalism and are a subject of future research.

Degree: We believe that when active, commitments do not necessarily remain in one constant state; they might age by becoming more or less important. This notion of commitment aging is captured by what we define as the degree of commitment. We believe that for a service-oriented coordination environment, the degree of commitment changes with changing beliefs, desires, and intentions. Also, specifically in the case of commitment cancellation or revocation, the commitment might not go from an active state to an inactive state instantaneously, but gradually decrease its degree until it becomes inactive.

Commitments are represented by a predicate C. The partially accessible commitment properties are represented inside angle brackets "$\langle \ldots \rangle$" and the private properties are represented inside square brackets "$[\ldots]$". Commitments have the form $C(i, a, b, p, S, W, \langle \mu \rangle, [d])$, where

i: is a unique identifier,
a: is the creditor agent,
b: is the debtor agent,
p: is the promise or the condition that the debtor will bring about,
S: is the context, also known as the *sphere of commitment*,
W: is the utility weighting,
μ: is the utility coefficient,
d: is the age or degree of commitment.

In this chapter, we do not use all of these properties, but mention them wherever pertinent. Throughout the rest of this chapter we refer to the creditor entity as a and the debtor entity as b.

5.2 Obligations

We believe that obligations are closely tied to the notions of duty and responsibility. An obligation is a promise that one makes to oneself; it is driven by the demand of ones own conscience or custom or socially accepted norms and it binds one to a specific course of action. We believe that obligations may also exist between a debtor agent and an abstract creditor agent, which cannot be represented as one concrete creditor, for instance society or say one's country. In this chapter however, we will consider only those obligations that represent promises one makes to oneself.

We believe that obligations can be represented as a special case of commitments. Obligations, unlike commitments, are best described as unitary and private in nature. In the described service-oriented environment, obligations are the abstractions of bindings that an agent imposes on itself. These obligations or internal bindings are visible only to the agent and are driven solely by agent's internal state of mind i.e. beliefs, desires and intentions. We believe that commitments' claim over a promise is stronger than that of obligations.

As we are dealing only with unitary obligations, the Multiagency and Utility Weighting properties are inapplicable. R and μ have special values set by the debtor agent itself. The other properties of Scope, Manipulability, Life, and Degree are treated the same as they are for commitments.

Obligations are represented by a predicate O, with their private properties inside square brackets "[...]". Obligations have the form: $O(i, b, p, S, \mu, [d])$, where

i: is a unique identifier,
b: is the debtor agent,
p: is the promise or the condition that the debtor will bring about,
S: is the context, also known as the *sphere of commitment*,
μ: is the utility coefficient,
d: is the age or degree of obligation.

5.3 Operations on Commitments and Obligations

As described above, our service-oriented environment is commitment-driven and participating agents' knowledge is governed solely by commitment operations. In this section, we describe commitment operations [23, 12] and their extension [22]. Commitments are treated as abstract data types that associate debtor, creditor, promise, and context. The seven fundamental commitment operations are

1. *Create (b, C(i, a, b, p, S))*: This operation establishes a commitment C in the situation S. This operation can only be performed by C's debtor.
2. *Discharge(b, C(i, a, b, p, S))*: This operation indicates that the inherent promise in the commitment C has been fulfilled; hence the commitment C has been satisfied.
3. *Revoke(b, C(i, a, b, p, S))*: This operation cancels the commitment C and can only be performed by C's debtor. This operation also reflects the autonomy of the participating entity.
4. *Release(a, C(i, a, b, p, S))*: This operation captures the situation where a creditor no longer wishes its debtor to fulfill its committed promise and releases it of its commitment. It can only be performed by C's creditor.
5. *Assign(a, z, C(i, a, b, p, S))*: This operation enables a commitment's creditor to designate another entity as the creditor. It can only be performed by C's creditor and replaces a with z as C's creditor.
6. *Delegate(b, z, C(i, a, b, p, S))*: This operation enables C's debtor to transfer its commitment promise to another agent. This operation can only be performed by C's debtor and replaces b with z as C's debtor.
7. *Suspend(b, C(i, a, b, p, S))*: This operation can only be performed by C's debtor, and describes a situation where the debtor has put its promised commitment on hold.

We use predicates to describe whether the commitment C has been satisfied, revoked, breached, or still holds, written as *satisfied(C)*, *revoked(C)*, *breached(C)*, and *active(C)*, respectively.

For obligations, only the following four of the above operations are applicable: *Create(b, C (i, b, p, S))*, *Discharge(b, C (i, b, p, S))*, *Revoke(b, C (i, b, p, S))*, and *Suspend(b, C (i, b, p, S))*. Obligations are unitary, internal, and private in nature; hence, assignment and delegation is not applicable. Because obligations can be treated as a special case of commitments, in the remainder of this chapter we use commitments as the basic abstraction for both binary and unitary agent bindings.

5.4 Negotiated Agreements as Commitment Promises

As described in Section 1, in service-oriented environments the participating agents, which play the roles of a service provider and a service seeker, negotiate and commit to a service agreement about the execution and completion

of a task. During the negotiation, the agents communicate and compromise to reach an agreement on matters of mutual interest while maximizing their utilities. In this section we will describe how the negotiated agreements, which associate or bind these participating agents with each other, can be best encapsulated as commitment promises.

Let b denote the service provider or the debtor agent and a denote the service seeker or the creditor agent as described in Section 4.3. Both a and b negotiate on issues related to the service and come to an agreement. How they communicate and their particular negotiation strategy is beyond the scope of this section.

We first consider agreements over a single issue. Specifically, a and b have negotiated and agreed upon an issue set $I = A$, where A represents one issue, such as the product price in an e-commerce transaction.

Expanding on the agents' negotiation parameters as defined in Section 4, we define the agents' agreement parameters as

$$\begin{aligned} S_a &= \langle P_a, U_a \rangle_A \\ S_b &= \langle P_a, U_b \rangle_A \end{aligned} \tag{12}$$

where, $P_a = P_b = P_{agreed}$ is the agreed price or the agreed parameter over issue A. This is public information.

U_a and U_b are utilities of the respective participating agents. The utilities are associated with the negotiated agreement on A. It is partially accessible information known to the owner agent and the non-participating arbiters. Note that the actual utility, $U_{actual} = \mu \times U$, is awarded to the agent once the commitment C reaches finality.

The negotiated agreement between agents a and b over issue A is a commitment in which the agreed parameter over A is the commitment promise. From section 5.1, and because this is a single-issue agreement so that $W = 1$, the commitment is represented as $C(i, a, b, P_{agreed}, S, 1, \langle \mu \rangle, [degree, age])$.

We now consider multiple-issue negotiated agreements. As an example, an online transaction between an online bookseller and a buyer would involve agreement from both sides on the multiple issues of book price, book condition (new or used), delivery method, etc. All these are sub-issues of the main issue of "buying a book."

Let there be an issue set I of k issues, where $I = \{I_1, I_2, \ldots, I_k\}$. Expanding on the agents' negotiation parameters as defined in Section 4, the agents' agreement parameter sets are defined as:

$$\begin{aligned} S_a &= \langle P_a, U_a \rangle_I \\ S_b &= \langle P_b, U_b \rangle_I \end{aligned} \tag{13}$$

where, $P_a = \sum_{i \in I} P_a^i$, and similarly, $P_b = \sum_{i \in I} P_b^i$

This means that the overall agreed price or agreed parameter over issue I, which comprises k sub-issues, is the summation of the agreed price or the

agreed parameter of all the sub-issues. Since a and b are in agreement, $P_a = P_b = P_{agreed}$, it is the overall agreed price or the overall agreed parameter over the issue A. This is public information, which is available to all the participants and the non-participating arbiters.

U_a and U_b are overall utilities of the respective participating agents. This utility is associated with the negotiated agreement on the issue I. This is partially accessible information, which means that it is known to the owner agent and the non-participating arbiters. Note that the actual utility, $U_{actual} = \mu \times U$ and will be awarded to the agent once the commitment C reaches some kind of finality. We know that $U_a = \sum_{i \in I} U_a^i$. Similarly, $U_b = \sum_{i \in I} U_b^i$. Which means that the overall utility for an agent to have an agreement on a parameter over issue set I, which comprises k sub-issues is the summation of utilities it gains on having an agreement on all the sub-issues.

Now we describe the concept of W in greater detail. As described above, the overall utility of the debtor agent b over the issue set I is the sum all the utilities ("sub-utilities") it gains over all the k sub-issues that make up the issue set I. We theorize that in the issue set I all of the sub-issues do not necessarily have an equally significant effect on its overall utility. In our book-selling example, let us assume that a service provider b and a service seeker a enter into a commitment relationship in which b promises to deliver a book to a. Of the many sub-issues that make up the complete transaction, the "color of the book cover" may not have as significant an impact on b's overall utility as does the "condition of the book" or the "delivery time". W represents the relative significance of sub-issues that make up an issue set.

Considering the negotiated agreement between agents a and b over the issue set I as a commitment relationship, the relationship between a and b can be represented as: $C_{ab} = \sum_{i \in I} C_{ab}^i$, which means that the commitment relationship between a and b over negotiated agreements on the issue set I, which comprises k sub-issues, is the summation of all the commitments on all the k sub-issues. Note that $\sum_{i \in I} W_b^i = 1$. Thus, service-oriented environments where participating agents are involved in negotiated agreements over single or multiple issues can be modeled by our commitment-driven approach.

6 Commitment-Based Coordination Protocol

Organizational control is needed to ensure that the appropriate information is communicated among the coordinating agents, so that they can make effective decisions to advance the overall objective. The key information being communicated is of three types:

1. Static information, such as authority relationships
2. Dynamic information, such as policies, standard operating procedures, and communication protocols
3. Contextual information, such as the current state of the overall workflow or plan and the states of the relevant agents.

An important aspect of our approach is that it treats organizational control as an integral aspect of planning, particularly for coordinating in the face of exceptions. This is a reasonable approach, because the flexibility of an organization reflects the complexity of its plans, the dynamism of its environment, and the risks faced by its plans. Thus heuristic techniques for encoding and using coordination strategies are naturally extended into strategies that accommodate organizational structure and control.

Moreover, organizational structure can be used to control the complexity both of the design and configuration of agent systems and of the execution by individual agents. This improves scalability. Well-designed organizations naturally yield narrow interfaces so that changes are not unnecessarily propagated and the right information flows at the right time. We cast the problem of organization design as a natural next step to the representation and design of agent heuristics, where the heuristics are selected so as to capture and exploit organizational structure. For example, we could have heuristics to report exceptions or anticipated exceptions to a supervisory role; to delegate a commitment to a subordinate; to request a peer to accept a delegate; to assign a resource not needed to a peer; and so on. In this manner, the general approach for verifying correctness could be made more elaborate to take advantage of organizational structure. Moreover, a model of the agents' organization, policies, and authority can be integrated with coordinated decision making to ensure the compliance of decisions to organizational policies.

To make this discussion concrete, let's outline how inter-agent control and intra-agent control mesh:

1. One or more agents perceive or are notified of an event.
2. Each agent perceiving the event decides (a) whether the event changes its local plan, and (b) whether to communicate the change (by itself or along with additional results of its reasoning) to another agent.
3. If an agent decides the event does not affect it or any one of its dependent agents, then it filters out the event and continues on its prior execution path. If an agent decides that the event does not affect its own plan, but could possibly affect plans of its dependent agents, then it communicates the event to the affected agents.
4. If an agent decides that the change affects its own plan, it reconsiders its commitments and begins a renegotiation of those that cannot be met.
5. The actions proposed to meet commitments are subjected to a "filter" that detects any that are in opposition to policies. All agents have an obligation to act in accordance with appropriate and applicable policies.
6. If the coordinated commitment-revision process encounters difficulties, the agent who has the most severe difficulty is given its preference and the coordination continues.

The above process can be captured in a general and flexible manner through the use of commitments. As explained in Section 5, commitments provide a natural abstraction to encode relationships among autonomous,

heterogeneous parties. Commitments are important for organizational control, because they provide a layer that mediates across the declarative semantics of organizations from the operational behavior of the team members. Organizational control based on commitments by a reasoner in an agent has the advantages that:

1. Commitments can be assigned to roles, so that any unit that fills the role of "transport troops" will, e.g., inherit a commitment assigned to the role to move troops from location A to location B.
2. Commitments can be delegated, so that a captain who has the commitment to "transport troops" can delegate the commitment to Helicopter Unit 1.
3. Commitments can be reassigned. For example, if Helicopter Unit 1 fails to meet its commitment (the helicopters break down) then the captain can release Unit 1 from the commitment and delegate it to Unit 2.
4. Commitments can be negotiated. The captain might ask another captain (a peer) to take over a commitment that could not otherwise be met.
5. Commitments can fail to be met, in which case the failure can be communicated to an agent with the authority to release the original commitment and reassign it.

Commitments to follow required policies are a kind of obligation, and are managed by a deontic reasoner. An organizational model based on obligations and rights can enable agents to represent and reason about the relationship between the responsibilities of the agent or group being coordinated and applicable policies, decision-making constraints, authorities, and overall objectives. This feature decides which organizational policies apply for the current situation and marks as unacceptable any intended actions that are inappropriate.

7 Commitments in Plan Revision

It is clear from the above that coordination is not a one-shot effort that can be satisfied through one round of planning, but must be carried out repeatedly. Further, coordination includes challenges such as unexpected events and changing situations, and must respect not only physical constraints, but also organizational challenges.

One aspect of commitments involves scheduling algorithms so that an agent can manage multiple commitments in the face of external events. Each agent applies classification to identify the general class of an event, then the classification is used to choose heuristics most likely to lead to effective coordinated behavior. Each agent maintains the consistency of formally represented commitments leading to robust, yet flexible coordination reasoning.

This relies upon a temporal semantics for commitments, which naturally leads to heuristics for ensuring that tasks that can be scheduled are satisfactorily scheduled given the emerging constraints. Another aspect involves

reasoning about commitments more directly at the level of coordination as it relates to communication. To this end, it helps to develop additional representations based on commitments. Such representations can be thought of as patterns of coordination relationships.

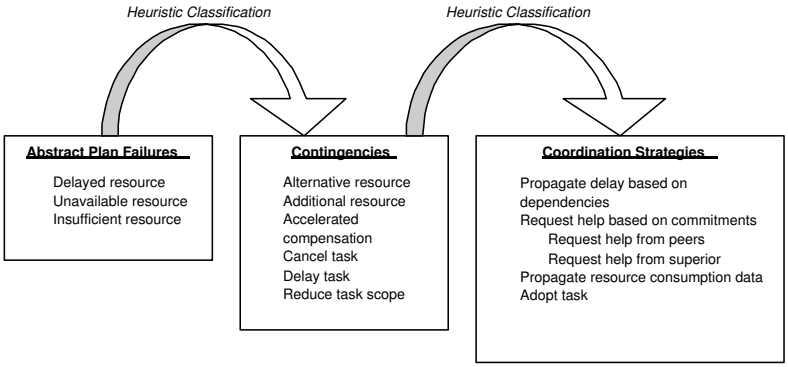

Fig. 2. Elements of a simple domain-independent mechanism for coordinating a response to conflicts and failures in plans

Commitments provide us with a basis for creating techniques that are generic and reusable. It is helpful to frame these as first-order patterns of interaction as well as second-order patterns of how other patterns are modified. These patterns would be indexed according to different situations and potential threats such as lost communications, ineffective participants, and so on. Figure 2 illustrates examples of how certain coordination strategies can be associated with potential plan failures. This is an example of heuristic classification in the sense of [1]. In our approach, this heuristic classification is supported by our semantics for commitments. Commitments are formally modeled via temporal logic; each agent's behavior is modeled via a simple finite-state machine (FSM).

To operationalize a commitment, we represent it as an FSM that processes commitments. The FSMs corresponding to different patterns can be combined with each other to yield the desired composite structure for the different agents. Figure 3 illustrates an example of a heuristic for handling a delayed resource. On the left is a part of an agent's FSM behavior where it deals with obtaining a resource from another party. An agent implemented according to this FSM would wait for the resource to arrive and then process it according to its current plan. However, such an agent would not be robust under certain kinds of enactment failures, specifically if the resource fails to materialize on time. The FSM on the right is an alternative for the same functional behavior. An agent implemented according to this FSM would be robust under the above failure, because it would time-out and generate reminders for the missing resource.

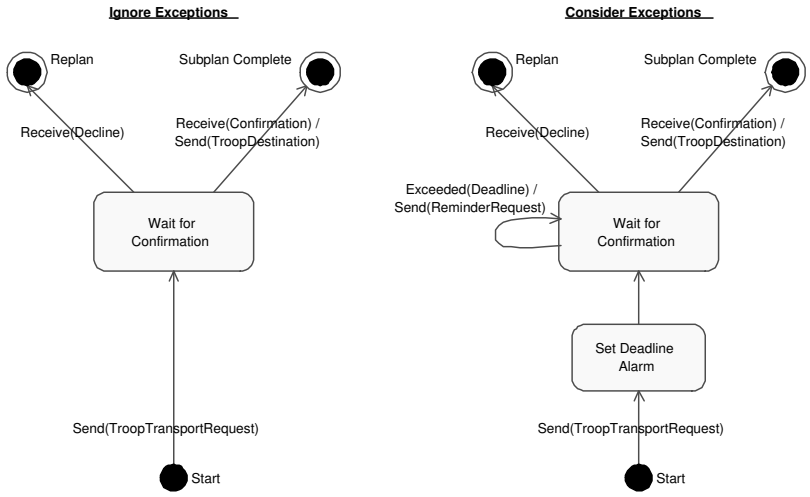

Fig. 3. Operationalizing commitments: an example of a finite-state machine for a coordinator that handles a delayed resource

The above heuristic is promising, but has an obvious shortcoming in that, if the resource is dead rather than merely delayed, the agent will keep generating reminders, whereas it should drop the current plan altogether. As agents are designed for a rich variety of scenarios, more such heuristics will be needed. They might be invented at run-time via machine learning or during configuration when a team of agents is deployed.

We can validate if a set of agents will function together in a manner to produce the right behaviors. It is known that checking the correctness of a distributed system of complex components is not tractable. The FSM representation of the heuristics hides irrelevant detail and enables the correctness verification known as model checking. Examples of the kinds of errors that can be detected early via model checking are: (1) if all the agents in a system are implemented according to the simple FSM on the left in the figure above, then such a system will hang when a resource dies, and (2) when resource sharing, if the receiver of a resource is implemented according to the FSM on the right, then we can confirm that reminders will be generated in case of a delay, but there might still be unnecessary delays because the resource provider cannot notify the resource consumer and the resource consumer will be unable to terminate its current plan if the resource is in fact dead. Similarly, we can create additional sets of FSMs and verify their correctness. Previous work on this problem used a Computational Tree Logic (CTL) model checker to create FSMs that would guarantee specified combinations of commitment patterns [28, 27].

In simple terms, the methodology combines the power of heuristics and the learning of agent behaviors, while providing a sound underpinning in terms of

commitments and their formal semantics. Heuristic classification is essential for practical knowledge acquisition and implementation; formalization gives us the essential guarantees of robustness and reliability that are necessary for mission-critical situations.

8 Conclusions

Commitments are a powerful representation for modeling intelligent interactions among agents distributed within an organizational structure. Previous approaches have considered the semantics of commitments and how to check compliance with them. However, for large-scale applications such as supply chains or military operations, these approaches do not capture implicit temporal task dependencies or the organizational authority and responsibilities among the participating entities. Our use of negotiated commitments for coordination lets us capture realistic task dependencies and avoid ambiguities. Consequently, it enables us to reason about whether, and at what point, a commitment is satisfied or breached, and whether it is or ever becomes unenforceable when replanning must be done.

Our use of deadlines for agent plans is similar to that for commitment life-cycles [7], which explains how operations can create, modify, delete, and satisfy commitments. This work operationalizes commitments, and we extend it to yield agent-internalized BDI semantics for temporal commitments.

The use of policy and organizational reasoning for coordination requires advances in the representation of policies in terms of commitments and obligations and an associated deontic reasoning mechanism. A temporal deontic logic for specifying obligations so that interaction protocols can take deadlines into account has been developed [4]. Other work on obligations [11] used them to represent and reason about policies, but did not incorporate commitments, as we do.

The choice of commitments as a basic data type for coordination enables the monitoring of performance by recording the satisfaction of prior commitments. This can be used to predict an agent's computational resource needs, and can be used to determine when an agent is not meeting expectations.

This chapter also investigates the coalition deal as a strategy for QoS-aware negotiation over commitments. Using equilibrium strategies, we prove that it makes better tradeoffs between utility optimization and computational efficiency than either the package deal or issue-by-issue negotiation.

Many real world systems are becoming service-oriented. In a service-oriented multiagent system, commitments represent agent associations and interactions. In such an environment, a participant agent's beliefs, desires, and intentions about the commitments in which it is involved are critical to modeling its behavior. By formalizing commitments in terms of BDI, we have provided the basic framework on which a more comprehensive commitment-driven coordination theory could be developed. The advantage

of this framework is that it blends two established formalisms—BDICTL [3] and commitments—that together can model a service-oriented multiagent system. Our future research involves exploration of how agents decide what to commit (integrating earlier works on "capability" [15] with commitments), when to revoke a commitment, how a commitment ages, and how historical information of an agent's commitment adherence can be utilized to predict agent behavior.

References

1. W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27(3):289–350, 1985.
2. J. Dang and M. N. Huhns. Optimal multiple-issue negotiation over qos metrics of web service. *USC CIT Technical Report TR-CIT05-01*, 2005.
3. M. Dastani and L. van der Torre. An extension of bdictl with functional dependencies and components. In *Proceedings of LPAR'02*, volume LNCS 2514, pages 115–129. Springer, 2002.
4. F. Dignum, H. Weigand, and E. Verharen. Meeting the deadline: On the formal specification of temporal deontic constraints. In *Foundations of Intelligent Systems, 9th International Symposium*, 1996.
5. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, pages 995–1072, 1990.
6. S. S. Fatima, M. Wooldridge, and N. Jennings. Optimal negotiation of multiple issues in incomplete information settings. In *Proc. Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*, pages 1080–1089, New York, USA, 2004. ACM.
7. N. Fornara and M. Colombetti. Operational specification of a commitment-based agent communication language. In *AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 536–542. ACM Press, 2002.
8. D. Gouscos, M. Kalikakis, and P. Georgiadis. An approach to modeling web service qos and provision price. In *Proc. Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03)*, pages 121–130, Roma, Italy, 2003.
9. B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The TAEMS white paper. 2004. http://mas.cs.umass.edu/research/taems/white.
10. C. M. Jonker and V. Robu. Automated multi-attribute negotiation with efficient use of incomplete preference information. In *Proc. Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*, pages 1056–1063, New York,USA, 2004. ACM.
11. L. Kagal, T. Finin, and A. Joshi. A policy based approach to security on the semantic web. In *Proc. Second International Semantic Web Conference*, 2003.
12. A. U. Mallya and M. N. Huhns. Commitments among agents. *IEEE Internet Computing*, 7(4):90–93, July-Aug 2003.
13. E. M. Maximilien and M. P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93, Sept-Oct 2004.

14. T. D. Nguyen and N. Jennings. Coordinating multiple concurrent negotiations. In *Proc. Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*, pages 1064–1071, New York,USA, 2004. ACM.
15. L. Padgham and P. Lambrix. Agent capabilities: Extending BDI theory. In *AAAI/IAAI*, pages 68–73, 2000.
16. G. Petrone. Managing flexible interaction with web services. In *Proc. Workshop on Web Services and Agent-based Engineering (WSABE 2003)*, pages 41–47, Melbourne, Australia, 2003.
17. C. Preist. A conceptual architecture for semantic web services. In *Proceedings of the Third International Semantic Web Conference 2004 (ISWC2004)*, Hiroshima,Japan, 2004.
18. S. Ran. A model for web services discovery with qos. *ACM SIGecom Exchanges*, 4(1):1–10, 2003.
19. A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.
20. T. Sandholm and N. Vulkan. Bargaining with deadlines. In *Proc. National Conference on Artificial Intelligence (AAAI)*, pages 44–51, Orlando, FL, 1999.
21. S. Sen. Reciprocity; a foundational principle for promoting cooperative behavior among self-interested agents. In *Proc. International Conference on Multi-Agent Systems*, pages 322–329, Menlo Park, CA, 1996.
22. D. Shrotri and M. N. Huhns. Formalization of multiagent commitments in a BDI_CTL framework. *USC CIT Technical Report TR-CIT05-02*, 2005.
23. M. P. Singh. Synthesizing coordination requirements for heterogeneous autonomous agents. *Autonomous Agents and Multi-Agent Systems*, 3(2):107–132, 2000.
24. M. P. Singh and M. N. Huhns. Social abstractions for information agents. In M. Klusch, editor, *Intelligent Information Agents*, Boston, MA, 1999. Kluwer Academic Publishers.
25. M. P. Singh and M. N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, London, UK, 2005.
26. SWSA. Semantic web services architecture requirements. Semantic Web Services Initiative Architecture committee (SWSA). http://www.daml.org/services/swsa/swsa-requirements.html.
27. M. Venkatraman and M. P. Singh. Verifying compliance with commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, 1999.
28. J. Xing and M. P. Singh. Engineering commitment-based multiagent systems: A temporal logic approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 891–898. ACM Press, 2003.