Scheduling Non-Enforceable Contracts Among Autonomous Agents

Thomas Tesch Karl Aberer
GMD-IPSI Integrated Publication and Information Systems Institute
German National Research Center for Information Technology
Dolivostraße 15, D-64293 Darmstadt, Germany
{tesch,aberer}@darmstadt.gmd.de

Abstract

With the emergence of fast and standardized communication infrastructures over which separately designed agents of different organizations can interact in real-time, there is an increasing demand for cooperation mechanisms that allow to carry out inter-organizational cooperations in a safe way. The lack of external control over an agent's decisions, resources and actions hamper the usage of traditional transaction and workflow technology to make self-interested agents cooperate, i.e., agents cannot not be forced from a mediating cooperation instance to continue a cooperation. The challenge is, therefore, to design a cooperation mechanism that motivates cooperating agents to carry out a specified contract and, in case of unilateral defection, ensures that none of the cooperators can benefit from the situation. In this paper, we present a domain independent framework how non-enforceable cooperations can be made safe against unilateral defection. We have developed a utility-based scheduling algorithm that keeps a cooperation in equilibrium and that motivates agents to continue a cooperation as long as it is for all participants beneficial.

1 Introduction

Today, formerly centralized and isolated databases are connected to each other and are starting to interoperate and cooperate using electronic networks and EDI, while still maintaining their autonomy. Complete integration of the various resources is not adequate for technical and organizational reasons. Hence, new functionality is required to support interactions among autonomous systems. With the growth of a fast and inexpensive standardized communication infrastructure over which separately designed information systems belonging to different organizations can interact in real-time, there is an increasing demand for cooperation mechanisms that allow to carry out inter-organizational transactions in a safe way. This demand is strengthened by the trend to fast reorganizing enterprise structures that require to engage in short term inter-enterprise alliances as a response to dynamic market developments. We refer to an autonomous information system that carries out tasks and contracts as an agent.

The lack of external control over an agent's decisions, resources and actions hamper the usage of traditional transaction and workflow technology to make agents cooperate. Agents acting in non-cooperative environments are self-motivated and try to maximize their own benefits. A self-interested agent tends to defect a cooperation unilaterally if it expects a benefit from defection. The challenge is, therefore, to design a cooperation mechanism that motivates cooperating agents to carry out a cooperation or, in case of unilateral defection, minimize losses for the opponents.

We present a domain independent framework how inter-agent cooperations can be made safe. We assume that a cooperation is safe when it is in an equilibrium state. An equilibrium state is a state in which all cooperating agents have almost the same utility (positive or negative) from the cooperation. We introduce a contract manager that mediates among cooperating agents and tries to keep the cooperation in equilibrium. The contract manager comprises a scheduler that composes inter-agent transactions in order to reach a subsequent equilibrium state atomically from a given equilibrium state. The final goal is to reach a state in which no further improvement for all participants is possible, i.e., all agents have reached their maximum utility.

The approach addresses complex inter-agent cooperations that cannot be pre-coordinated and controlled from a central instance. Agents are acting rather dynamically in order to achieve their goals. Therefore, they are in an ongoing process of adapting their next actions towards their current goals. With the contract manager presented here, solutions for interactions are found that cannot be achieved by direct communication of the participants. With this work we propose an innovative paradigm for future transaction and workflow models, in order to cope with the specific conditions of inter-organizational cooperations. It utilizes existing research results from the domain of distributed artificial intelligence in order to progress towards this goal. We believe that contract managers mediating among autonomous information systems will play a similar role for inter-organizational cooperations than workflow management systems for intra-organizational cooperations.

In Section 2 we present a description of the problem domain and outline the solution proposed in this paper. Section 3 describes how inter-agent cooperations are specified. Afterwards, we present in Section 4 the scheduling algorithm for keeping the cooperation in equilibrium and illustrate it by example. We compare our approach in Section 6 to related approaches in areas of database transactions, transactional workflows and distributed artificial intelligence before we summarize and conclude in Section 7.

2 Cooperation among Agents

An agent is embedded into an organizational structure where it controls a set of local resources and has specific capabilities, actions that it can perform, and certain tasks and responsibilities possibly delegated by a human agent. The actions can be database actions performed as ordinary database transactions, or communications, such as providing some piece of information or sending a message to another agent. An agent tries to act towards a specific goal. Agents are autonomous in view of their cooperating counterparts. They decide autonomously which actions are performed best in order to achieve a goal. The interests of an agent can change depending on what it perceives from the environment [10].

Interactions among agents considered in this paper are inter-organizational and, thus, purely competitive, involving self interest and utility maximization. As agents are autonomous, the factors which influence their behavior are private and not available to their opponents. Thus, agents do not know what utilities their opponents place on various outcomes, they do not know what reasoning models they employ, they do not know their opponents constraints, and they do not know whether an agreement is even possible.

Therefore, cooperations among self-interested agents in open environments include several risks.

- First, self-interest implies that agents are only motivated to engage in a cooperation if it promises to be individually beneficial. This means, an agent cooperates as long as the cooperation maximizes its expected utility compared to alternative plots. The decision whether the continuation or the abortion of a cooperation is more beneficial is taken locally by the agent and cannot be controlled from outside.

- Second, the local reasoning of an agent is based on uncertain information which leads to suboptimal individual decisions. In the light of new information, an agent can change its course of action. The motivation for the continuation of a contract is influenced by individual interests that may turn out to be conflicting with current contracts.

- Third, agents may not truthfully provide all information they have in the cooperation. It is not always possible to verify information provided by an agent. An individual rational agent may expect an advantage from providing malicious information or from holding back information.

- Fourth, in open networks it is easily possible to vanish in an ongoing cooperation by dropping the network connection, killing the own process etc. If there is no real world entity connected with the software agent, unilateral defection can be a rational option for the owner. Due to a lack of a legal framework in international computer networks, it is not realistic to assume that a contract can subsequently be enforced by common laws [14].

A service enabling cooperation among agents has to deal with the mentioned problems. The major challenge of a cooperation instance is to give guarantees to the cooperators that cannot be enforced by the cooperation instance itself, i.e., the cooperation instance can neither force an agent to fulfill its obligations nor can it hinder an agent to take a different course of action. The task of the cooperation instance is to coordinate the exchange among the agents in a way that the local exchange constraints are not violated and none of the agents becomes cheated.

Suppose a WWW shopping agent has 3 items in its shopping basket. It does need only 2 of them but the warehouse offers one item only in a package with another one. Both parties have to exchange goods and payment bilaterally, e.g., due to a missing common law enforceable among both (agents are located in different countries). Therefore, neither the electronic store trusts the customer nor vice versa. Furthermore, both parties have different interests in the deal. The customer only wants item 1 and item 2 whereas the provider is eager to sell item 2 and item 3 in a package. Since both parties are not aware of their different interests, it is impossible for them to perform a successful exchange. When the provider has delivered item 1 and item 2 the customer might defect the cooperation without purchasing item 3. On the other hand, the provider might not be interested to continue the deal after item 2 and item 3 are delivered.

The conflict is to be resolved by the introduction of a third instance that is aware of the different interests called the *contract manager*. Both partners tell the contract manager their personal interests in the deal (utility). The task of the contract manager is to form a contract acceptable for both parties and to schedule the exchanges in a way that considers the individual interests, i.e., none of the participants should be motivated during the exchange to defect the cooperation. The exchange can be performed in multiple steps in which the interests of the individual agents can change. We believe that these properties are of interest for complex asynchronous exchanges like interactive Web shopping scenarios of long duration or open-ended cooperations like subscriptions in digital libraries. Our mechanism is of special interest when there is no common law that can enforce misbehaviors afterwards.

The participants indicate to the contract manager the actions they are willing to perform by submitting them to the contract manager. We assume that actions that are submitted are under the control of the contract manager, i.e., the contract manager decides when they are executed. This can be the result of the authority of the contract manager resulting from means external to the actual cooperation. In a second phase, the contract manager notifies the participants about the existing submissions. The agents can calculate their utilities for the different possible solutions and send them back to the contract manager. The task of the contract manager is to compose the submitted actions into a sequence that satisfies the utilities of all participants and keeps the contract state in equilibrium. If no solution is found, the agents are encouraged to either re-evaluate their utilities or to submit further actions. Otherwise, the contract manager coordinates the execution of the selected sequence. The resulting state is an equilibrium state in which all participant reach almost the same amount of utility. Afterwards, the agents can submit further operations, adapt their utility function and continue the cooperation. With the proposed mechanism, they may change their interests during a contract is active. The set of actions that the agents are allowed to submit can be fixed in a preceding negotiation phase or, in case of open-ended cooperations, can be dynamically proposed and extended..

The fact that cooperations are non-enforceable is counteracted with a specific scheduling strategy. One of the scheduling criterions is to keep the defection motivation of each participant as low as possible. A low defection motivation is assumed if the state of the contract is in equilibrium, i.e., none of the partners has invested significantly more than its opponents. As long as the contract is in equilibrium, both participants loose/win the same in case of defection. Even if one of the participants could do better without disadvantaging its opponents in terms of utility, an equilibrium state is preferable because it increases the probability of a continuation. Note, the fact that a contract is kept in equilibrium is no guarantee for well-behaved cooperations but it increases the probability of well-behaved cooperations.

3 Formal Framework

A cooperation is a sequence of multi-agent agreements, called contracts, on taking a common course of action mediated through a contract manager. Within a cooperation there are a set of constraints reflecting the different interests of the participants. The constraints restrict the different states that are reachable during a cooperation. Conceptually, the cooperation can be considered to take place in a shared data space where each party has different benefits from future state changes. The goal of the contract manager is to mediate among the participants and find acceptable subsequent states by means of composing contracts.

For each contract, the cooperation space is defined as a tuple $C = (P, S_0, \mathcal{O}, \mathcal{U})$ where

- $P = \{p_1, \ldots, p_n\}$ is the set of participants,
- S_0 is the initial state of the cooperation,
- $\mathcal{O}$ is the set of actions offered as potential contributions for one step of the cooperation by the participants $p_i \in P$ with $O_i \subseteq \mathcal{O}$ is the subset of actions offered by agent p_i, and
- $\mathcal{U}$ is a set of utility functions with $U_i \in \mathcal{U}$ is the utility function of agent p_i.

3.1 Participants

We distinguish two kinds of roles which are involved in a cooperation: the different participants $P = \{p_1, \ldots, p_n\}$ and the contract manager as intermediary that mediates conflicting goals in the cooperation. Participants can act as producers, consumers or both. In the context of information business, producers are document sources, libraries, digital stores whereas consumers are customers with specific interests. We assume no general trust among the participants.

3.2 States

When a cooperation is initialized it is in a state S_0. Conceptually, the state of the cooperation is composed of the local states of the participating agents. Subsequent states S of a cooperation are defined by the initial state S_0 and the actions that have been scheduled resulting in S. With $S = S_0 \bullet [a_1, \ldots a_n]$ we denote the state resulting from the execution of a sequence of actions $[a_1 \ldots a_n]$ with a_i executed before a_j if $i < j$.

3.3 Actions

Within a cooperation each agent contributes with the execution of local actions. The execution of local actions has effects on the cooperating partners. Each action can thus affect multiple participants. This forms the basis of the cooperation. For instance, if agent i performs the action *pay*$_{i \to j}(x)$ this indicates that i performs a payment action having effect on j which receives the transferred amount x. Actions affecting two participants have an *initiator* which initiates and controls the execution and a *receiver* which is affected by the execution.

The set $\mathcal{O}$ contains all actions that are offered by the participants in one step of the cooperation. The entire set of actions that are to be exchanged are not considered formally here. This can either be a fixed set of actions the opponents have agreed upon in a prior negotiation process or actions can be dynamically proposed in each step of the cooperation. The subset $O_i \subseteq \mathcal{O}$ contains all actions offered by agent p_i in one step of the cooperation including [] referring to *no operation*. We define $\mathcal{A}$ as all possible execution sequences that can be constructed from the offered actions in $\mathcal{O}$ with $\mathcal{A} = \{[a_1, \ldots, a_n] | \forall i, j : 1 \leq i, j \leq n \wedge a_i, a_j \in \mathcal{O} \wedge a_i \neq a_j \wedge 1 \leq n \leq |\mathcal{O}|\}$.

The set of successor states $\mathcal{S}$ that can be constructed from the current state S_0 and the execution sequences $\mathcal{A}$ is $\mathcal{S} = S_0 \bullet \mathcal{A}$.

When an agent has once offered the execution of an action, it is assumed that the contract manager can perform the operation if required. When the contract manager schedules an action sequence forming a contract, the actions are performed by the participants by means of transactions. The involved state changes of the local participants are coordinated with a 2 phase commit protocol in which the contract manager acts as coordinator[11].

3.4 Utility

A key ingredient in the approach presented here, is the notion of utility which we will use to specify the value, to an agent, of total or partial achievement of a goal. The definition of a goal is subsumed by a worth oriented utility function [13] over the possible action sequences to be executed. Those states described by action sequences with the highest value might be thought of as satisfying the current goals best, while others, with lower utility values, only partially satisfy the goal.

The utility of each participant $p_i \in P$ is a mapping from all possible successor states of the current state to real numbers and defined as

$$\overline{U}_i : \mathcal{S} \to \Re$$

One way to determine the utility for a state $S \in \mathcal{S}$ given state S_0 with $S = S_0 \bullet [a_1, \ldots, a_n]$ is to calculate utilities based on values and costs for reaching S from the current state S_0. The utility is then defined as $U_i(S) = (V_i(S) - C_i(S))$ where $V_i(S)$ is the value expected from the execution of $[a_1, \ldots, a_n]$, $C_i(S)$ are the costs to be spent for performing $[a_1, \ldots, a_n]$.

The agents report their utilities for the reachable successor states to the contract manager. The normalized utility function

$$U_i : \mathcal{S} \to [0, 1]$$

is a mapping from all possible successor states of S_0 to the interval $[0, 1]$. The normalization is necessary in order to perform a fair comparison of interests. We require

$$\forall p_i \in P : \sum_{S \in \mathcal{S}} U_i(S) = 1$$

in order to ensure that all agents have the same amount of utility they can distribute among the cooperation space. The normalized utility function can be calculated from the ordinary utility function with a straight forward algebraic transformation [1].

The normalization ensures that understating the utility for a successor state $S \in \mathcal{S}$ will always result in an overstatement for some state $S' \in \mathcal{S}$ and vice versa. Assuming that the agents have no information on the utilities reported by their opponents, with overstating or understating the utility an agent risks to engage in a suboptimal contract. Based on this assumption, there are no beneficial lies. However, the cooperating opponents report their utilities only to the trusted third party which is unbiased towards the participants.

If the utility functions of the agents are not available to the contract manager, it can use bounds[15]. Participants can give lower bounds for $V_i(S)$ and upper bounds for $C_i(S)$ which can be used by the contract manager to calculate the normalized utility. However, a conservative choice of bounds will rule out some possible cooperations. Furthermore, estimation of agents utilities is a subtle task because it depends on the agents private goals.

4 Contract Management

4.1 Scheduling Algorithm

In our model we assume that cooperating agents exchange goods, information, or services among each other. A cooperation consists of multiple steps. In each step, the

[1]
$$U_i(S) = \frac{\overline{U}_i(S) + \overline{U}_{min,i}}{\sum_{S' \in \mathcal{S}} (\overline{U}_i(S') + \overline{U}_{min,i})}$$
with $\overline{U}_{min,i} = \min_{S \in \mathcal{S}} \overline{U}_i(S)$

participants try to compromise among the next actions to be performed by means of a contract. The concept of worth oriented utility allows agents to reach agreements, to compromise, in situations where agreement had previously been impossible. First, we generalize the utility evaluation principle before we present the protocol and scheduling mechanism of the contract manager.

The task of the contract manager is to find a suitable subsequent state reachable by a contract. Given the cooperation space $C = (P, S_0, \mathcal{O}, \mathcal{U})$ it searches for states with the properties indicated in the following. For each agent $p_i \in P$, we denote with $U_{i,nop} = U_i(S_0 \bullet [])$ the utility for *no operation* which is the execution of the empty sequence [].

Before the algorithm is given, we define some properties among the possible subsequent states:

Definition 1 *The set of potential successor states D_i for an agent $p_i \in P$ is defined as $D_i = \{S | S \in \mathcal{S} \wedge U_i(S) > U_{i,nop}\}$.*

Potential successor states denote those subsequent states that allow an agent to improve its situation. As long as there exists a potential successor state for an agent, the agent will continue the cooperation.

Definition 2 *The set of potential successor states D for the cooperation is defined as $D = D_1 \cap \ldots \cap D_n$.*

The scheduler ensures that each successor state that is reached, is a potential successor state for all participants.

Definition 3 *The set of equilibrium states E for a cooperation is defined as $E = \{S | \forall p, q \in P : |U_p(S) - U_q(S)| < \epsilon\}$.*

Equilibrium states are states which give the same amount of utility for both agents. It is expected that equilibrium states give the highest incentive for all participants to continue the cooperation. With ϵ we denote the maximum deviation in terms of utilities of the resulting state the contract manager is willing to tolerate. We assume that ϵ is chosen such that $S_0 \bullet [] \in E$ which implies that staying in the current state is in equilibrium.

Definition 4 *A cooperation is in a termination state if $D \cap E = \{\}$.*

The cooperation terminates if there are no potential successor states available that are in equilibrium. However, for multi step cooperations with a fixed set of actions to be exchanged, a termination state is also reached if all actions have been submitted and scheduled. In the last step, the equilibrium condition can be neglected because there are no remaining actions to be exchanged. Open-ended cooperations continue until no more potential successor states are found.

It is the task of the contract manager to find a solution among the state space that is composed from the possible execution sequences. Such a compromise should have following properties:

1. **Equilibrium:** in each subsequent state S, all participants should reach the same amount of their utility, i.e., $\forall p, q \in P : p \neq q \wedge |U_p(S) - U_q(S)| < \epsilon$. The constant ϵ specifies the maximum deviation of the utilities.

2. **Monotonicity:** the resulting state S' after the execution of a contract should be better than the current state S in terms of utility, i.e., $\forall i \in P : U_i(S) <= U_i(S')$. This condition could be given up assuming the participating agents agree on this. This might be necessary to enable the agents to cross a local minimum in their utility functions.

All states that become visible for the participants during the cooperation should be equilibrium states. In those states, none of the participants is motivated to defect the cooperation because there is an opportunity to improve the own benefit in further cooperation steps. The equilibrium condition guarantees safeness of intermediate states by ensuring that the different utilities do only deviate by a fixed constant ϵ. This avoids states in which one agent has almost reached a termination state while others are still interested to compromise on remaining actions.

The Algorithm 1 searches the state space reachable from the submitted operations in one cooperation step. Input is the current state S_0 and the actions $\mathcal{O}$ offered by the participants. After all possible sequences have been constructed, all potential successor states that are in equilibrium are collected in H. If H contains more than one solution, the state where the product of the utilities is maximal is selected because it provides the best solution of all equilibrium deals [13].

5 Example

We revisit the example mentioned in Section 2. Assume there are three exchanges possible among the buyer and the electronic store $\{e_1, e_2, e_3\}$. The buyer is most interested in e_1, e_2 and the shopping agent only sells e_2, e_3 in a package. With e_i we denote in this example a full exchange, i.e., delivery and payment. The non-additive normalized utilities for some selected execution sequences are given in table 1. The entry *other* captures all remaining sequences unacceptable for both agents. States with utilities less than the empty

	U_{buyer}	U_{shop}
[]	0.0434	0.074
e_1	0.0869	0.111
e_1, e_2	0.434	0
e_2, e_3	0.0869	0.407
e_1, e_2, e_3	0.347	0.407
other	0	0

Table 1. Example with non-additive utilities

Algorithm 1 Search next equilibrium state

```
Input:
  $S_0$ {current state}
  $P = 1, \dots, n$ {Participants}
  $\mathcal{O} = \{O_1, \dots, O_n\}$ {Actions offered}
  $H \leftarrow \emptyset$ {Executions resulting in successor state}
  for all $i \in P$ do
    if $[] \notin O_i$ then
      $O_i \leftarrow O_i \cup []$ {insert empty sequence}
    end if
  end for
  $\mathcal{A} = \{[a_1, \dots, a_n] | \forall i, j : 1 \leq i, j \leq n \wedge a_i, a_j \in \mathcal{O} \wedge a_i \neq a_j \wedge 1 \leq n \leq |\mathcal{O}|\}$ {Construction of execution sequences}
  $\mathcal{S} \leftarrow S \bullet \mathcal{A}$ {Construction of state space}
  for all $h \in \mathcal{A}$ do
    for all $i, j \in P | i \neq j$ do
      if $|U_i(S_0 \bullet h) - U_j(S_0 \bullet h)| < \epsilon \wedge (U_i(S_0 \bullet h) < U_i(S_0 \bullet []) \wedge U_j(S_0 \bullet h) < U_j(S_0 \bullet []))$ then
        $H \leftarrow H \cup \{h\}$
      end if
    end for
  end for
  if $H \neq \emptyset$ then
    return $h$ such that $\prod_{i=1}^{n} U_i(S \bullet h)$ maximal
  else
    return [] {no operations are performed}
  end if
```

sequence [] are considered as undesirable states. The algorithm given in Section 4 identifies the sequence $[e_1, e_2, e_3]$ as possible successor state assuming $\epsilon >= 0.06$. Thus, the exchange sequence $[e_1, e_2, e_3]$ leads to a subsequent equilibrium state. Note, the interleaving of e_1, e_2 and e_3 during the execution is not of importance because the sequence is executed atomically by means of transactions.

We consider a second example with separated payment and delivery actions. Agent A has agreed to buy three documents (e.g., technical descriptions, digital soundtracks, etc.) from agent B. The details of the preceding negotiation process are not captured here. Both have agreed that the overall payment is \$150 for the documents d1, d2, and d3. Let's further assume that agent A pays in units of \$50 and each document has different cost and worth for both agents. The actions a_1, a_2, a_3 transfer the payment from A to B and b_1, b_2, b_3 corresponds to the delivery of the documents.

It is not possible to perform the whole exchange in one transaction because d1 is not in stock. However, both parties want to perform those parts of the exchange that are already possible. This can happen due to non-existence of certain items which are part of the agreement or simply different delivery sizes. Furthermore, direct payment of each item resulting in many payments might also be undesirable due to the increased transfer fees.

We further assume in this example that both agents have an expected worth of 0.3 which is calculated from $e_A = C_A([a_1, a_2, a_3]) - V_A([b_1, b_2, b_3])$ and $e_B = C_B([b_1, b_2, b_3]) - V_B([a_1, a_2, a_3])$. The utility functions of A and B are presented in Tables 2 and 3. The rows and columns in the tables contain the actions forming the respective state. Each entry in the table describes the utility for all states containing the respective actions, i.e., in this example the utility function is additive such that an action is independent from previous and subsequent actions.

In the first step of the exchange A submits all possible payment actions and B offers all deliveries, i.e., $H_A = \{a_1, a_2, a_3\}$ and $H_B = \{b_1, b_2, b_3\}$. The contract manager calculates subsequent equilibrium states which are visualized in Table 7. With $\epsilon \leq 0.0004$ the successor states in equilibrium are $S_1 = [a_1, b_1]$, $S_2 = [a_1, a_2, b_2, b_3]$ and $S_3 = [a_1, a_2, a_3, b_1, b_2, b_3]$. If, for example, an item cannot be delivered immediately it may be an option to go first into S_2 which is an equilibrium state. Then, the remaining

u_A	[]	a_1	a_1, a_2	a_1, a_2, a_3
[]	0.00	-0.50	-1.00	-1.50
b_1	0.50	0.00	-0.50	-1.00
b_2	0.40	-0.10	-0.60	-1.10
b_3	0.90	0.40	-0.10	-0.60
b_1, b_2	0.90	0.40	-0.10	-0.60
b_1, b_3	1.40	0.90	0.40	-0.10
b_2, b_3	1.20	0.70	0.20	-0.30
b_1, b_2, b_3	1.80	1.30	0.80	0.30

Table 2. Utility U_A

u_B	[]	a_1	a_1, a_2	a_1, a_2, a_3
[]	0.00	0.50	1.00	1.50
b_1	-0.30	0.20	0.70	1.20
b_2	-0.20	0.30	0.80	1.30
b_3	-0.70	-0.20	0.30	0.80
b_1, b_2	-0.50	0.00	0.50	1.00
b_1, b_3	-1.00	-0.50	0.00	0.50
b_2, b_3	-0.90	-0.40	0.10	0.60
b_1, b_2, b_3	-1.20	-0.70	-0.20	0.30

Table 3. Utility U_B

actions can be executed in the next cooperation step.

Both parties have in this state a higher utility than in the current state and none of them has already reached its maximum utility which is in this example the full delivery in state S_3. Therefore, both parties will continue the exchange of the remaining items. Table 6 additionally shows the product of the agents utilities which are used as second scheduling criterion to find the best equilibrium state among the ones found.

Note, for the situation described in the example, there is no exchange possible with $\epsilon = 0$. The next state in which both agents have exactly the same utility is where the full delivery has occurred.

When the contract manager cannot find a subsequent equilibrium state there are different options:

- Notification of participants to submit further actions that allow to reach an equilibrium state.

- Increase ϵ which allows to mark more states as equilibrium states. This is critical since it affects the safeness of the contract state.

- Request the participants to relax their goals by means of changing the utility function.

A further option would be to compensate exchanges already performed to return to a previous equilibrium state. However, taking into account that agents can adapt their utility function for each exchange, this method appears not as promising. Furthermore, compensation of already received information is in many cases not possible.

U_A	[]	a_1	a_1, a_2	a_1, a_2, a_3
[]	0.0143	0.0095	0.0048	0.0000
b_1	0.0191	0.0143	0.0095	0.0048
b_2	0.0181	0.0134	0.0086	0.0038
b_3	0.0229	0.0181	0.0134	0.0086
b_1, b_2	0.0229	0.0181	0.0134	0.0086
b_1, b_3	0.0277	0.0229	0.0181	0.0134
b_2, b_3	0.0258	0.0210	0.0162	0.0115
b_1, b_2, b_3	0.0315	0.0267	0.0219	0.0172

Table 4. Normalized utility U_A

U_B	[]	a_1	a_1, a_2	a_1, a_2, a_3
[]	0.0139	0.0197	0.0255	0.0313
b_1	0.0104	0.0162	0.0220	0.0278
b_2	0.0116	0.0174	0.0231	0.0289
b_3	0.0058	0.0116	0.0174	0.0231
b_1, b_2	0.0081	0.0139	0.0197	0.0255
b_1, b_3	0.0023	0.0081	0.0139	0.0197
b_2, b_3	0.0035	0.0093	0.0150	0.0208
b_1, b_2, b_3	0.0000	0.0058	0.0116	0.0174

Table 5. Normalized utility U_B

	[]	a_1	a_1, a_2	a_1, a_2, a_3
[]	0.00019879	0.00018775	0.00012148	0.00000000
b_1	0.00019879	0.00023192	0.00020984	0.00013253
b_2	0.00020984	0.00023192	0.00019879	0.00011044
b_3	0.00013253	0.00020984	0.00023192	0.00019879
b_1, b_2	0.00018554	0.00025180	0.00026285	0.00021867
b_1, b_3	0.00006405	0.00018554	0.00025180	0.00026285
b_2, b_3	0.00008946	0.00019437	0.00024407	0.00023855
b_1, b_2, b_3	0.00000000	0.00015462	0.00025401	0.00029819

Table 6. product of utilities $U_A * U_B$

	[]	a_1	a_1, a_2	a_1, a_2, a_3
[]	0.0004	-0.0101	-0.0207	-0.0313
b_1	-0.0087	0.0019	-0.0124	-0.0230
b_2	-0.0066	-0.0040	-0.0146	-0.0251
b_3	-0.0171	-0.0066	-0.0040	-0.0146
b_1, b_2	-0.0148	-0.0042	-0.0063	-0.0169
b_1, b_3	-0.0254	-0.0148	-0.0042	-0.0063
b_2, b_3	-0.0223	-0.0117	0.0012	-0.0094
b_1, b_2, b_3	-0.0315	0.0209	-0.0104	0.0002

Table 7. $|U_A - U_B| < \epsilon$ as positive numbers, others negative

6 Related Work

This section compares the problem that we are addressing to related work in the area of distributed database transactions, workflow management systems, and distributed artificial intelligence.

In the traditional view of a database transactions [4], the systems aims at maintaining a consistent state. Consistency is based on the assumption that a single transaction is correct and that no anomalies are introduced due to interleaved execution of transactions. Transaction are under the control of a single party. Two phase commit protocols are ways of ensuring that all participants of a transaction commit to the state change or none of them [11]. Our approach differs in that there are different interests among the participants and, thus, the transaction (cooperation) is under the control of several instances. We introduce a quality measure on states that give each participant its own view on the next preferred consistent state. In contrast to a traditional transaction manager which re-orders actions to a serializable schedule or a cooperative transaction manager which governs the execution sequences according to a specification, the role of the contract manager is to mediate among the different interests by generating schedules that reach equilibrium states.

Recent extensions of transaction models discuss complementary aspects of the work presented here. In [2] I-transactions are introduced. They provide atomicity for a set of actions triggered by a user upon different and not a-priori known DBMS. The work in [7] considers how exchanges can occur among chains of multiple parties connected through brokers and distrusting each other.

The ADEPT project [6, 12, 5] addresses the problem of agent interoperation in domains such as business process management. ADEPT takes a distributed approach where the disparate components of a business process are each represent by an agent. Agents retain control over their own resources , the tasks that they perform, and their communication and coordination with other agents. The architecture supports the encapsulation of services through a hierarchy of agencies, and so enables abstract services that are negotiated among the agents. Agents try to reach mutually acceptable agreements on the execution of services through a negotiation protocol. The approach emphasizes the negotiation aspect rather than how agreements can be enforced without central control. Other agent based workflow approaches are [18, 3, 9] which rely on distributed event condition action rules. To the best of our knowledge, decision autonomy of participating components is not considered.

Research in distributed artificial intelligence aims at models for communication and cooperation among intelligent automated systems. An overview is given in [8]. The mechanism presented in this paper exploits research results from game theory. Research on game theoretic approaches to agent negotiation [13, 21] has focused on the definition of cooperation protocols that give no benefit to agents that misrepresent or hide information. In [17] disclosure of information is acceptable, because it will help an agent to find an optimal solution for itself. This also holds for the solution presented here. It has been argued in [19] that some assumption of classical game-theoretic approaches limit the applicability of game theory to solve real problems.

[15, 14] has investigated how exchanges can be carried out among agents without the involvement of a third party. The basic idea of the proposed mechanism is making the present less important compared to the future (initially suggested by [1]). An exchange module schedules an agents deliveries in such a way that the opponent is not motivated to defect at any point during the exchange. The approach is based on linear increasing utility functions for each cooperating agent. Therefore, the tradeoff of an agent increases as long as the cooperation continuous. Although the exchange is scheduled in a way that an agents future gains are higher than its future costs it cannot be guaranteed that defection may be at no point individually rational. The approach requires that the agents know their utilities or at least upper and lower bounds of the opponents utility functions. Due to the fact that in the approach presented in this paper the utilities are only applicable by the contract manager, they are not shared between the agents.

Recent work on coalition formation analyzes coalitions among self-interested agents. By forming a coalition and acting jointly agents can save costs in contrast to acting individually. [20] presents a mechanism for coalition formation in task oriented domains. The work presented in [16] investigates coalitions among self-interested agents which need to solve combinatorial optimization problems.

7 Conclusion

In this paper we have present a domain independent framework how inter-agent cooperations can be made safe against defection. A cooperation is safe when it is in an equilibrium state. An equilibrium state is a state in which all cooperating agents have almost the same utility from the cooperation. We have introduced the contract manager that mediates among cooperating agents and tries to keep the cooperation in equilibrium. The presented scheduling algorithm composes sequences resulting in a subsequent equilibrium state. The presented model motivates the participants in a cooperation to carry out the cooperation and, in case of unilateral defection, ensures that a defector cannot profit from the situation.

In future work we intend to extent the basic model presented here in several ways. First, inter-agent and intra-agent action dependencies should be taken in consideration, e.g., agents may want to define sequences they want to have scheduled atomically or in certain orders etc. Such preferences can currently only be expressed via utilities. For this problem class a more elaborated scheduling algorithm will be developed. Second, under specific circumstances it might be useful that the scheduler enters possibly on behalf of an agent non-equilibrium states. This functionality is required if one participant is willing to take risk in order to carry out a formerly impossible exchange. Third, a natural extension would be if the contract manager provides means for insurance. Depending on the assurance, the contract manager can take risk if it can calculate the probability for unilateral defection. Different kinds of insurance are possible, e.g., the insurance can compensate losses up to some value previously agreed.

Acknowledgments

The authors thank Peter Fankhauser for inspiring discussions on the topic and his valuable comments to the normalization of utility which helped to avoid safe beneficial lies by overstating or understating utility.

References

[1] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[2] D. Billard. Transactional services for the internet. In *Proc. of the Int. Workshop on the Web and Databases*, pages 11–17, Valencia, Spain, 1998.

[3] A. Geppert, M. Kradolfer, and D. Tombros. Realization of cooperative agents using an active object-oriented database management system. In *Proc. of the second Int. Workshop on Rules in Database Systems*, Athens, Greece, September 1995.

[4] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, Inc., 1993.

[5] N. R. Jennings, P. Faratin, P. Johnsen, T. J. Norman, P. O'Brien, and M. E. Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 1996.

[6] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, M. E. Wiegand, C. Voudouris, J. L. Alty, T. Miah, and E. Mamdani. Adept: Managing business processes using intelligent agents. In *Proc. of the BCS Expert Systems 1996 Conference*, pages 5–23, 1996.

[7] S. P. Ketchpel and H. Garcia-Molina. Making trust explicit in distributed commerce transactions. In *Proc. of the IEEE Int. Conference on Distributed Computing Systems*, 1996.

[8] S. Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94:79–97, 1997.

[9] P. C. Lockemann and H.-D. Walter. Object-oriented protocol hierachies for distributed workflow systems. *Theory and Practice of Object Systems*, 1(4):281–300, 1995.

[10] P. Maes. Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1), 1994.

[11] C. Mohan and B. Lindsay. Efficient commit protocols for the tree of processes model of distributed transactions. In *Proc. of the second ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 378–396, August 1983.

[12] T. J. Norman, N. R. Jennings, P. Faratin, and E. Mamdani. Designing and implementing a multi-agent architecture for business process management. In *Proc. of the third Int. Workshop on Agent Theories, Architectures, and Languages*, 1996.

[13] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computer*. MIT press, 1994.

[14] T. Sandholm and V. Lesser. An exchange protocol without enforcement. In *Proc. of the 11th Int. Workshop on Distributed Artificial Intelligence*, Washington, USA, 1994.

[15] T. W. Sandholm and V. R. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In *Proc. of the 14th Int. Joint Conference on Artificial Intelligence*, pages 694–701, Montreal, Canada, 1995.

[16] T. W. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94:99–137, 1997.

[17] C. Sierra, P. Faratin, and N. R. Jennings. A service-oriented negotiation model between autonomous agents. *Lecture Notes in Artificial Intelligence*, 1237:17–35, 1997.

[18] D. Tombros, A. Geppert, and K. R. Dittrich. Semantics of reactive components in event-driven workflow execution. In *Proc. of the 9th Int. Conference on Advanced Information Systems Engineering*, Barcelona, Spain, June 1997.

[19] D. Zeng and K. Sycara. How can an agent learn to negotiate? In *Proc. of the third Int. Workshop on Agent Theories, Architectures, and Languages*, 1996.

[20] G. Zlotkin and J. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings AAAI*, Seattle, WA, USA, 1994.

[21] G. Zlotkin and J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In *Proc. of the 11th Int. Joint Conference on Artificial Intelligence*, pages 912–917, Detroit, Michigan, USA, August 1989.