Resolving Commitments Among Autonomous Agents *

Ashok U. Mallya[1], Pınar Yolum[2], Munindar P. Singh[1]

[1] Department of Computer Science, North Carolina State University, Raleigh NC 27695-7535, USA
[2] Department of Artificial Intelligence, Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

Abstract. Commitments are a powerful representation for modeling multiagent interactions. Previous approaches have considered the semantics of commitments and how to check compliance with them. However, these approaches do not capture some of the subtleties that arise in real-life applications, e.g., e-commerce, where contracts and institutions have implicit temporal references. The present paper develops a rich representation for the temporal content of commitments. This enables us to capture realistic contracts and institutions rigorously, and avoid subtle ambiguities. Consequently, this approach enables us to reason about whether and when exactly a commitment is satisfied or breached and whether it is or ever becomes unenforceable.

1 Introduction and Objectives

Protocols help streamline the complex interactions that can take place between autonomous, heterogeneous agents in a multiagent system. A special application setting is e-commerce, where the agents represent different parties that do business on-line.

The role of commitments in modeling such rich interactions is widely recognized, because they enable the key content of an interaction to be represented and reasoned about, especially in the face of opportunities or exceptions. Commitments are thus more expressive than traditional formalisms such as finite state machines. Yolum and Singh [1] show how to build and execute commitment-based protocols and to reason about such protocols in the event calculus. Fornara and Colombetti [2] capture key aspects of the commitment lifecycle and further advance the idea of commitments as a data structure. Some compliance aspects of commitment protocols in a branching-time temporal logic with potential causality have also been studied by Venkataraman and Singh [3].

Motivation. The above approaches show that commitments provide a viable representational framework for designing, executing, and validating flexible protocols in multiagent systems. However, current approaches take a limited view of the temporal aspects of commitments. This can prove to be a drawback for their use in real systems, since business deals usually involve many clauses and have subtle time periods of reference. The following is an informal list of some properties that are relevant in practice, but not naturally handled by current approaches.

* We thank the anonymous referees for their comments and suggestions that helped improve the text. We also thank Mario Verdiccio for his suggestions. This research was supported by the National Science Foundation under grant DST-0139037.

– *Time Intervals*. Contracts often involve time bounds, which simplify decisions about the satisfaction or breach of commitments, which is one of the reasons traditional representations (e.g., paper documents) rely on them. Practical commitments often must be satisfied either within a fixed, bounded interval or at a specified instant in the future.
– *Maintenance*. Current work on commitments has concentrated on achievement conditions, whereas real-life commitments are as likely to be about the maintenance of certain conditions. For example, a typical service-level agreement (SLA) may involve committing to maintaining network connectivity during business hours.
– *Temporal anaphora*. A particular variety of time bounds arise in the notion of temporal anaphora, as introduced by Partee [4]. A promise such as "I will send you the goods" or a claim such as "I tried to call you five times" involves an implicit range of salient times within which the specified action occurred or will occur. Although we are not concerned here with commonsense reasoning, our representational framework for commitments should be able to accommodate the results of such reasoning.

Point-based temporal logics, which are commonly used in distributed system specifications, are inadequate to express the above requirements. Accordingly, we develop an extension of the well-known branching-time logic, Computation Tree Logic (CTL) developed by Emerson [5], which can capture the cases of interest here.

Challenges. We use examples from situations that arise in practical applications of web services to motivate our study of temporal aspects of commitments. We consider the example of a travel agent, who wishes to book an airline ticket to a certain destination, a rental car to use while there, and also a hotel room to stay at.

Example 1. The travel agent wants the passenger to be able to fly on one particular day, reserving the right to choose any flight on that day. If the airline is willing to offer such a deal, it becomes committed to maintaining a condition – a booked ticket – over an extended period of time. We need to be able to specify such maintenance conditions in commitments. ∎

Example 2. The car rental company might offer, for some reason, one weeks free rental in the month of January. This is a maintenance condition within another time period. We need to be able to capture such temporal intricacies without bloating the domain language. ∎

Example 3. Some commitments may violate constraints about time that commonsense reasoning would have detected. Such a situation can arise, for example, when a hotel offers an electronic discount coupon that expires today, but the coupon can be used only in some future time period, say, a special spring break offer that expires much before spring break. ∎

Example 4. Another interesting example is when a warranty cannot be verified within the period over which the warranty is valid. Consider a customer who rents a car from a company which guarantees that the car will not break down for at least a two days, and promises a replacement car if it does. However, if the car were rented on a Friday and the company is closed on the weekends, then the customer is at a disadvantage. ∎

Example 1 is solved in Section 4.1, Examples 2 and 3 in Section 4.2, and Example 4 in Section 4.3.

Contribution. Our main contribution is in applying a richer temporal representation to commitments and showing how the satisfaction or breach of a commitment can be detected. Further, the temporal aspects of commitments are independent of the domain-specific semantics of the condition that the commitment is about, so that we can reason about the temporal aspects of commitments in a domain-independent manner.

Organization. Section 2 introduces background concepts, Section 3 develops our technical approach, Sections 4 explains our results on the resolution of commitments that use temporally qualified propositions, and Section 5 summarizes our proposal and identifies directions of further research.

2 Background: Time and Commitments

We next briefly explain our model of time and our temporal logic, and introduce the notion of commitments.

2.1 The Temporal Framework

We use a discrete, branching-time model, as shown in Figure 1. The temporal model has the following features:

- F_1 The world is a set of discrete *moments* in time. $\mathbb{M}$ is the set of all possible moments, partially ordered by $\prec$. The past is linear, and the future branches.
- F_2 Each moment m is given a timestamp $\tau(m) \in \mathbb{T}$, totally ordered by $<$. If $m_0 \prec m_1$ then $\tau(m_0) < \tau(m_1)$.
- F_3 A *scenario* S at a moment is a maximal set of moments containing the given moment and all moments along some branch in the future of the given moment. $\mathbb{S}_m$ denotes the set of all scenarios at a moment m. A scenario $S \in \mathbb{S}_m$ has the following properties:
  - S is *rooted*; i.e., $m \in S$.
  - S is *linear*; i.e., $(\forall m_1, m_2 \in S : (m_1 = m_2$ or $m_1 \prec m_2$ or $m_2 \prec m_1)$ and $m \preceq m_1)$.

  For example, in Figure 1, the path $m_0 m_1 m_5 \ldots \in \mathbb{S}_{m_0}$

We use an extension of Emerson's Computational Tree Logic (CTL) [5]. We now introduce the components of CTL.

1. *Booleans*. $\neg$ and $\vee$ carry their usual meaning. true, false, $\rightarrow$, and $\wedge$ are obvious abbreviations.
2. *Linear time*. These operators apply over a particular scenario.
   U: A proposition $p \mathsf{U} q$, read *p until q*, is true at a moment m_i on a scenario, iff q holds at some moment m_x in the future on the given scenario and p holds at all moments between m_i and m_x.

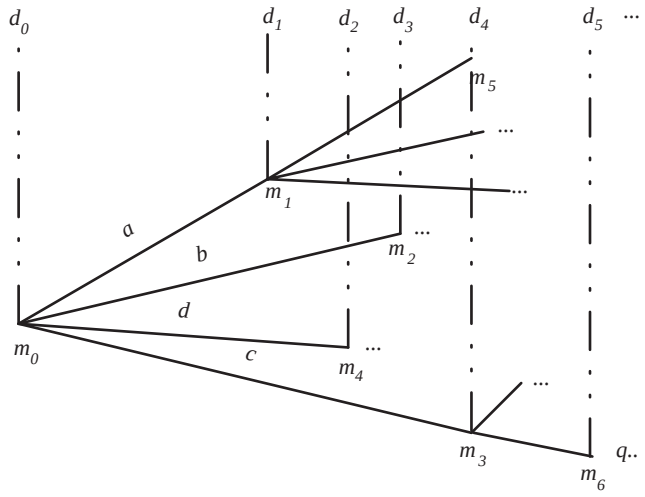

Fig. 1. A schematic representation of our model of time

F: A proposition Fp, read *eventually p*, means that p holds at some point in the future in the given scenario. Fp abbreviates true Up.
G : A proposition Gp, read *always p*, means that p always holds in the future on the given scenario. Gp abbreviates $\neg$F$\neg p$

3. *Branching quantifiers*. The operator A denotes *in* all *scenarios* at the present moment.

2.2 Temporal Qualification

The temporal commitment structure specified by Fornara and Colombetti [2] forms the basis for our temporal commitment scheme. Every condition specified in the commitment language has a time interval, a temporal quantifier, and a proposition in the domain language.

We use timestamps to denote endpoints of time intervals. Two timestamps d_l and d_u are used to represent an interval that begins at d_l and ends at d_u, both instants inclusive. For any such time interval, $d_l \leq d_u$. We introduce the following *temporal quantifiers* to quantify over instants in the interval:

1. $[\,]$ is an existential quantifier over a time interval. $[d_1, d_2]p$ means the proposition p has to hold at one or more instants in the interval beginning at d_1 and ending at d_2.
2. $\overline{[\,]}$ is a universal quantifier over a time interval. That is, $\overline{[d_1, d_2]}p$ means the proposition p has to hold at every instant in the interval beginning at d_1 and ending at d_2.

2.3 Commitments

Commitments are obligations that one agent has towards another, as Castelfranchi describes [6]. Formally, a commitment C*(id, x, y, p)*, relates a debtor x, a creditor y, and a condition p in such a way that x becomes responsible to y for satisfying the condition p; the commitment has a unique identifier id. The commitment is said to be satisfied when the condition p holds. There can be at most one commitment with a particular identifier in our entire model.

Commitment Operations. Commitments are created, satisfied, and transformed in certain ways. According to Singh [7], the following operations can be performed on commitments.

1. CREATE*(x, c)* establishes the commitment *c* in the system. This can only be performed by *c*'s debtor *x*.
2. CANCEL*(x, c)* cancels the commitment *c*. This can only be performed by *c*'s debtor *x*. Generally, cancellation is compensated by making another commitment.
3. RELEASE*(y, c)* releases *c*'s debtor *x* from commitment c. This only can be performed by the creditor *y*.
4. ASSIGN*(y, z, c)* replaces *y* with *z* as *c*'s creditor.
5. DELEGATE*(x, z, c)* replaces *x* with *z* as the *c*'s debtor.
6. DISCHARGE*(x,c)* c's debtor x fulfills the commitment.

We note that a commitment has to be created using the CREATE operation for it to exist. Further, we assume equivalence of the performance of a DISCHARGE operation and the satisfaction of the condition p. That is, we assume that the DISCHARGE operation brings about p, and conversely, if p occurs, the DISCHARGE operation is assumed to have happened. This assumption does not impoverish the theory, and we defer a discussion on it to Section 5.

We also introduce two predicates in Section 3.1 that help us capture the notion of fulfillment of a commitment rigorously.

Commitment Predicates. For every operation on commitments listed above, we introduce a corresponding predicate which has the same name as the operation, but with lowercase letters. The predicates, instantiated with proper parameters, are true at the moment at which the corresponding operation is performed. For example, if an agent x performs a CREATE*(x,c)* operation at a moment m_i, then the predicate *create(x,c)* is said to have the truth value true at the moment m_i.

Commitment Identifiers. Every commitment is assumed to have a unique identifier that helps to distinguish it from other commitments that may have the same debtor, the same creditor, and the same condition. For example, if I promise to pay you $5 twice, then a single payment of $5 should not suffice. The predicates in question also apply to specific commitments, i.e., respecting their unique identifiers. For example, I may cancel one of my two commitments to pay $5 without automatically canceling the other commitment. The identifiers come from a domain $\mathbb{D}$, which can be thought of as formed of the natural numbers.

3 Technical Framework

This section introduces the concept of time intervals, describes the formal language for our scheme, and introduces two key predicates dealing with the resolution of commitments.

3.1 The Formal Language

The following is a grammar for our formal language, $\mathcal{T}$, expressed in Backus-Naur Form (BNF). Here, tokens begenning with an uppercase letter denote nonterminals, tokens begenning with a lowercase letter denote lexical items that are not analyzed by this grammar, $agent$ stands for any agent symbol, $\longrightarrow$ and $|$ are meta-symbols of the BNF, and all other symbols are terminals. T is the unique starting symbol for the language of $\mathcal{T}$.

G_1 $T \longrightarrow \mathsf{A}Expr \mid \neg\mathsf{A}Expr$
G_2 $Expr \longrightarrow Prop \mid Oper$
G_3 $Prop \longrightarrow \neg Prop \mid Prop \vee Prop \mid Prop \mathsf{U} Prop \mid [I, I]Prop \mid \overline{[I, I]}Prop \mid a$
G_4 $I \longrightarrow date \mid variable \mid date + duration \mid variable + duration$
G_5 $Oper \longrightarrow \neg Oper \mid Oper \vee Oper \mid Oper \mathsf{U} Oper \mid breached(C) \mid satisfied(C) \mid$
$create(agent, C) \mid cancel(agent, C) \mid delegate(agent, agent, C)$
$\mid assign(agent, agent, C) \mid release(agent, C) \mid discharge(agent, C)$
G_6 $C \longrightarrow \mathsf{C}(identifier, agent, agent, Prop)$

In the grammar, a is an atomic proposition in the domain, $identifier$ is a unique commitment identifier, $date$ is a timestamp, $date \in \mathbb{T}$, $variable$ is a time variable that is bound to a timestamp , and $duration$ is a length of time used to construct simple additive expressions with time variables. Section 3.2 explains how time variables are bound to timestamps.

As a convention, we use p and q to denote simple propositions and p_t and q_t to denote temporally qualified propositions.

New Predicates for Resolving Commitments. We propose two predicates, ***breached(c)*** and ***satisfied(c)***, indicating violation and fulfillment of the given commitment, respectively.

3.2 Semantics

We now describe the semantics for the language $\mathcal{T}$. For a proposition p, $M \models_m p$ means that a model M satisfies proposition p at moment m. $M \models_{S,m} p$ means that the model M satisfies p at moment m in the scenario S. When resolving nested interval expressions, $M \models_{S,m,m_l,m_u,\mathtt{E}} p$ means that M satisfies p at a moment m in the scenario S within the interval that begins at m_l and ends at m_u, both m_l and m_u being in the scenario S. Two constants `E` and `U` are used as subscripts to denote whether the interval is to be interpreted as existentially or universally quantified, respectively.

An *interpretation* $\mathbb{I}$ labels each moment with the atomic propositions and the ground commitment predicates that are true at that moment. Ground commitment predicates

here refer to $create(\cdot,\cdot)$, $assign(\cdot,\cdot,\cdot)$, $delegate(\cdot,\cdot,\cdot)$, $cancel(\cdot,\cdot)$, $release(\cdot,\cdot)$, and $discharge(\cdot,\cdot)$. In a practical system, these elements would be specified in some manner external to the logic. For instance, a create operation might be taken to hold wherever a user submits a form over the Web.

Let Φ be a set containing the atomic propositions a and ground commitment predicates. Then $\mathbb{I} : \mathbb{M} \mapsto \varphi(\Phi)$.

Let $M = \langle \mathbb{A}, \mathbb{M}, \prec, \mathbb{I}, \mathbb{T} \rangle$ be a model for the formal language $\mathcal{T}$, where $\mathbb{M}$, $\mathbb{T}$, and $\prec$ have the meaning as explained in Section 2, $\mathbb{A}$ is a set of agent symbols, and $\mathbb{I}$ is an interpretation as defined above.

The semantics uses a *substitution* for time variables that occur in the bounds of intervals. If p is an expression, and $\boldsymbol{x}$ is a vector of all time-variables in the expression, then, $p\,|_{\boldsymbol{d}}^{\boldsymbol{x}}$ is the expression produced by a uniform, concurrent, and element-wise substitution of $\boldsymbol{x}$ by $\boldsymbol{d}$. An expression that has no time-variables is called *ground*. A ground expression is evaluated through date arithmetic. In the following, if t is ground, then $[\![t]\!]$ is the timestamp corresponding to it. The details of the arithmetic are not formalized here.

The semantics of $\mathcal{T}$ is given next. Here, c refers to a commitment of the form $\mathsf{C}(id, j, k, p)$ and d_i's denote timestamps. Also, $active(c)$ is an abbreviation for $\neg($ $cancel(j, c) \vee delegate(j, \cdot, c) \vee assign(k, \cdot, c) \vee release(k, c) \vee discharge(j, c))$.

The semantic rule R_1 creates ground expressions of time-variables. Rules R_2 through R_6 specify the meaning of the temporal and logical operators that are used. Rules R_7 and R_8 specify the meanings of intervals, while rules R_9 and R_{10} begin the evaluation of nested intervals. Rules R_{11} and R_{12} use the result of date arithmetic applied to ground time-expressions.

R_1 $M \models_{S,m} p$ iff $\exists \boldsymbol{d} : M \models_{S,m} p\,|_{\boldsymbol{d}}^{\boldsymbol{x}}$, where $p \notin \Phi$, $\boldsymbol{x}$ is a vector of variables the occur in p, $\boldsymbol{d}$ is a vector of timestamps, and $|\boldsymbol{d}| = |\boldsymbol{x}|$.
R_2 $M \models_{S,m} p$ iff $p \in \mathbb{I}(m)$ where $p \in \Phi$.
R_3 $M \models_{S,m} \neg p$ iff $M \not\models_{S,m} p$.
R_4 $M \models_{S,m} p \vee q$ iff $(M \models_{S,m} p$ or $M \models_{S,m} q)$.
R_5 $M \models_{S,m} p \mathsf{U} q$ iff $(\exists m_1 \in S : m \preceq m_1$ and $M \models_{S,m_1} q$ and $(\forall m_2 : m \preceq m_2 \preceq m_1 \Rightarrow M \models_{S,m_2} p))$.
R_6 $M \models_m \mathsf{A} p$ iff $(\forall S : S \in \mathbb{S}_m M \models_{S,m} p)$.
R_7 $M \models_{S,m} [d_l, d_u]p$ iff $M \models_{S,m,m_l,m_u,\mathtt{E}} p$, where $\tau(m_l) = d_l$ and $\tau(m_u) = d_u$.
R_8 $M \models_{S,m} \overline{[d_l, d_u]}p$ iff $M \models_{S,m,m_l,m_u,\mathtt{U}} p$, where $\tau(m_l) = d_l$ and $\tau(m_u) = d_u$.
R_9 $M \models_{S,m,m_l,m_u,\mathtt{E}} p$ iff $\exists m_x \in S : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} p$.
R_{10} $M \models_{S,m,m_l,m_u,\mathtt{U}} p$ iff $\forall m_x \in S : m_l \preceq m_x \preceq m_l$ and $M \models_{S,m_x} p$.
R_{11} $M \models_{S,m} [t_l, t_u]p$ iff $\exists d_l, d_u : d_l = [\![t_l]\!]$ and $d_u = [\![t_u]\!]$ and $M \models_{S,m} [d_l, d_u]p$.
R_{12} $M \models_{S,m} \overline{[t_l, t_u]}p$ iff $\exists d_l, d_u : d_l = [\![t_l]\!]$ and $d_u = [\![t_u]\!]$ and $M \models_{S,m} \overline{[d_l, d_u]}p$.

The following rules apply to the results of rules R_9 and R_{10} .

R_{13} $M \models_{S,m,m_l,m_u,\mathtt{E}} [d_l, d_u]p$ iff $\exists m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} [d_l, d_u]p$.
R_{14} $M \models_{S,m,m_l,m_u,\mathtt{E}} \overline{[d_l, d_u]}p$ iff $\exists m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} \overline{[d_l, d_u]}p$.
R_{15} $M \models_{S,m,m_l,m_u,\mathtt{U}} [d_l, d_u]p$ iff $\forall m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} [d_l, d_u]p$.
R_{16} $M \models_{S,m,m_l,m_u,\mathtt{U}} \overline{[d_l, d_u]}p$ iff $\forall m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} \overline{[d_l, d_u]}p$.

R_{17} $M \models_{S,m,m_l,m_u,\mathsf{E}} p\mathsf{U}q$ iff $\exists m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} p\mathsf{U}q$.
R_{18} $M \models_{S,m,m_l,m_u,\mathsf{U}} p\mathsf{U}q$ iff $\forall m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} p\mathsf{U}q$.
R_{19} $M \models_{S,m,m_l,m_u,\mathsf{E}} p \vee q$ iff $\exists m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} p \vee q$.
R_{20} $M \models_{S,m,m_l,m_u,\mathsf{U}} p \vee q$ iff $\forall m_x : m_l \preceq m_x \preceq m_u$ and $M \models_{S,m_x} p \vee q$.
R_{21} $M \models_{S,m,m_l,m_u,\mathsf{E}} \neg p$ iff $\exists m_x : m_l \preceq m_x \preceq m_u$ and $M \not\models_{S,m_x} p$.
R_{22} $M \models_{S,m,m_l,m_u,\mathsf{U}} \neg p$ iff $\forall m_x : m_l \preceq m_x \preceq m_u$ and $M \not\models_{S,m_x} p$.

Finally, we give the semantics for the $breached(\cdot)$ and $satisfied(\cdot)$ predicates.

R_{23} $M \models_m breached(c)$ iff ($\exists m_3 : m_3 \preceq m$ and $M \models_{m_3} \mathsf{AG}\neg discharge(j,c)$ and $\exists m_1 : m_1 \preceq m_3$ and $M \models_{m_1} create(j,c)$ and $(\forall m_2 : m_1 \preceq m_2 \preceq m_3 \Rightarrow M \models_{m_2} active(c))$).
R_{24} $M \models_m satisfied(c)$ iff ($\exists m_3 : m_3 \preceq m$ and $M \models_{m_3} discharge(j,c)$ and $(\exists m_1 : m_1 \prec m_3$ and $M \models_{m_1} create(j,c)$ and $(\forall m_2 : m_1 \preceq m_2 \prec m_3 \Rightarrow M \models_{m_2} active(c))))$

Further, we impose the following constraints on the model to capture operations on commitments.

C_1 A commitment cannot be created more than once with a given identifier.
$M \models_m create(j, \mathsf{C}(id,j,k,p)) \Rightarrow \forall m_1 : m \prec m_1 \Rightarrow (\forall j_1,k_1,p : M \not\models_{m_1} create(j_1, \mathsf{C}(id,j_1,k_1,p)))$.
C_2 When a commitment is assigned, it is no longer active, but a new commitment with the new creditor is created.
$M \models_m (assign(k,z,c) \Rightarrow \mathsf{AG}\neg active(c) \wedge create(j,c'))$ where $c \equiv \mathsf{C}(id,j,k,p)$ and $c' \equiv \mathsf{C}(id',j,z,p)$.
C_3 When a commitment is delegated, it is no longer active, but a new commitment with a different debtor is created.
$M \models_m (delegate(j,z,c) \Rightarrow \mathsf{AG}\neg active(c) \wedge create(z,c'))$ where $c \equiv \mathsf{C}(id,j,k,p)$ and $c' \equiv \mathsf{C}(id',z,k,p)$.
C_4 When a commitment is first created, it is active and not impossible to discharge.
$M \models_m (create(c) \Rightarrow active(c) \wedge \neg\mathsf{AG}\neg discharge(c))$
C_5 After a commitment has been breached, no operation can be performed on it.
$M \models_m (breached(c) \Rightarrow active(c))$

3.3 Commitment Life Cycle

Using the above semantics, we establish some simple but important lemmas indicating the stability of the $breached(\cdot)$ and $satisfied(\cdot)$ predicates.

When a commitment is first created, it is neither breached nor satisfied. Eventually, it might be breached and thus remain breached forever; or it may be be satisfied and remain satisfied forever. This has the effect of applying a three-valued logic for the satisfaction commitments.

Lemma 1. $M \models_m (create(c) \Rightarrow \neg breached(c) \wedge \neg satisfied(c))$.

Proof. By applying constraint C_4 to the semantic definitions R_{23} and R_{24}

Lemma 2. $M \models_m breached(c)$ *iff* $M \models_m \mathsf{AG} breached(c)$.

Proof. By semantic definition R_{23}

Lemma 3. $M \models_m satisfied(c)$ *iff* $M \models_m \mathsf{AG} satisfied(c)$.

Proof. By semantic definition R_{24}

Lemma 4. $M \models_m (\neg satisfied(c) \not\Rightarrow breached(c))$.

Proof. By Lemma 1.

Lemma 5. $M \models_m (\neg breached(c) \not\Rightarrow satisfied(c))$.

Proof. By Lemma 1.

The semantic rules R_{24} and R_{23} , and the Lemmas 2 and 3 are shown in Figures 2 and 3. Note that a commitment may be active even after it is breached. However, a commitment cannot be active after it is satisfied. Both these observations follows from the definition of the $active(\cdot)$ predicate.

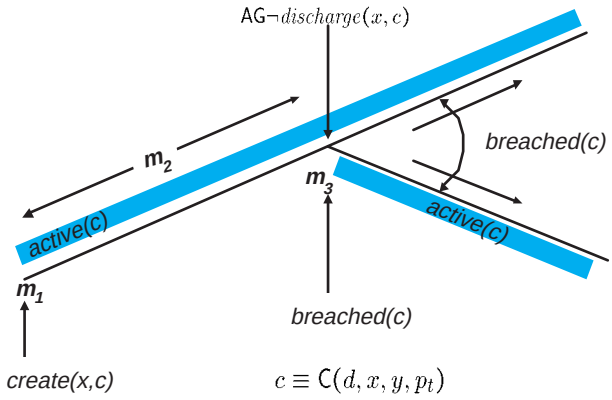

Fig. 2. Temporal behavior of the $breached(\cdot)$ predicate

4 Resolving Temporal Commitments

A temporal commitment is resolvable if its satisfaction or breach can be determined at some moment. Under certain conditions, the unresolvability of a temporal commitment can be ascertained even before the specified time interval occurs. We now discuss cases where a temporally quantified proposition is not resolvable, and develop methods to detect such cases. Based on the resolvability of such propositions, we can detect satisfaction or breach of temporal commitments.

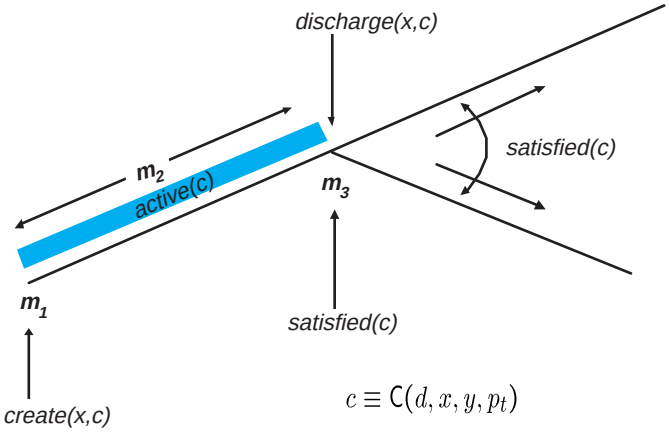

Fig. 3. Temporal behavior of the $satisfied(\cdot)$ predicate

4.1 Nested Interval Expressions

The language given in section 3.1 allows for propositions to be nested within multiple levels of time intervals. Although there are many nested intervals whose interpretation in common language does not make sense or induces redundancy, some nested time intervals do make sense in real-life situations. We give examples of both meaningful and meaningless nested interval propositions.

Allen [8] defines 13 possible temporal relationships between any two given time intervals. Figure 4 shows these 13 relationships for the two intervals contained in the proposition $[d_1, d_2](\overline{[d_3, d_4]}q)$; i.e., the intervals $[d_1, d_2]$ and $\overline{[d_3, d_4]}$. Here, time increases from left to right. The shaded portions are intervals of the type $\overline{[d_l, d_u]}$, and the unshaded portions are of the type $[d_l, d_u]$. 13 such cases can be constructed *for each combination* of temporal quantifiers applied to each of the intervals, but we show only one interval-quantifier combination pair as an example; i.e., the quantified intervals $[d_1, d_2]$ and $\overline{[d_3, d_4]}$.

If we consider nested temporally quantified propositions as being conditions of commitments, we can see which kinds of nesting will make the success of the commitment unresolvable. Cases 1.1, 3.1, 4.1, 5.1, and 6.1 are not resolvable, but cases 1.2, 2.1, 3.2, 4.2, 5.2, 6.2, 7.1, and 7.2 can be resolved for reasons listed below. The term *inner proposition* is used to refer to the temporal proposition $\overline{[d_3, d_4]}p$.

In cases 1.1, 3.1, 4.1, 5.1, and 6.1, the inner proposition's time interval does not complete until after the outer time interval completes. The inner interval has references to instants in the future. Since the future cannot be seen in advance, these cases cannot be resolved.

In cases 1.2, 2.1, 3.2, 4.2, 5.2, 6.2, 7.1, and 7.2 the inner proposition's success can be resolved at at least one instant within the interval of the outer proposition. Since the outer proposition has an existentially quantified time interval $[t_1, t_2]$ and we have at least one instant of resolvability, these cases can be resolved.

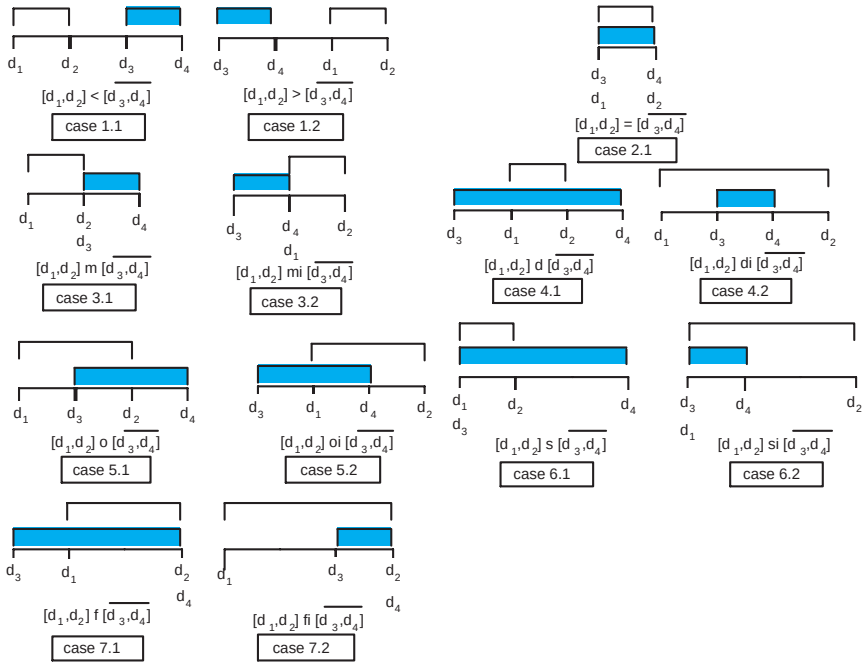

Fig. 4. Allen's intervals for $[d_1, d_2](\overline{[d_3, d_4]}p)$

A temporally quantified proposition can be used to represent events like that in Example 1; i.e., the passenger using one ticket on a particular day, on any flight of her choice.

Solution 1. If p represents the proposition that a ticket is on offer, then $[d_1, d_2]p$ can be used to denote that a ticket will be on offer for the period of time between d_1 and d_2. Hence a ticket valid for an entire day would be represented by $[d_1, d_1 + 24hours]p$ ∎

Nested temporal intervals can be used to denote maintenance conditions like the one in Example 2.

Solution 2. If d_1 denotes January 1 and t denotes a time variable, and p denotes that the company will rent a car for free, then the proposition $[d_1, d_1 + 31days]\overline{[t, t + 7days]}p$ denotes one week of free rental in the month of January. ∎

Note that it only requires a simple extension of our language to be able to specify timestamps in relation to one another.

We next formalize the notions of nested intervals and results about their resolution. These results can be used to detect some commitment violations before they occur.

Definition 1. *A temporally quantified proposition is positive-resolvable at an instant if its value is known to be true at that instant; it is negative-resolvable at an instant if its value is known to be false at that instant.*

Definition 2. *A temporal commitment is positive-resolvable at an instant if its satisfaction can be known at that instant; it is negative-resolvable at an instant if its breach can be known at that instant.*

We use the following notation to denote some important instants with respect to an interval. Below, p_t is a temporal proposition, and r denotes resolvability. Given an interval,
$r^+_\perp(p_t)$ represents the earliest instant at which p_t is positive-resolvable;
$r^+_\top(p_t)$ represent the latest instant in that interval at which p_t is positive-resolvable.
$r^-_\perp(p_t)$ represents the earliest instant at which p_t is negative-resolvable;
$r^-_\top(p_t)$ represents the latest instant in that interval at which p_t is negative-resolvable.

The following observations form the base cases for detecting resolvability of propositions that have intervals nested to any arbitrary depth and the resolvability of temporal commitments. Here, p is a proposition.

$$r^+_\perp([d_l, d_u]p) = d_l,\ r^+_\top([d_l, d_u]p) = d_u.$$
$$r^-_\perp([d_l, d_u]p) = d_u,\ r^-_\top([d_l, d_u]p) = d_u.$$
$$r^+_\perp(\overline{[d_l, d_u]}p) = d_u,\ r^+_\top(\overline{[d_l, d_u]}p) = d_u.$$
$$r^-_\perp(\overline{[d_l, d_u]}p) = d_l,\ r^-_\top(\overline{[d_l, d_u]}p) = d_u.$$

These observations imply that p_t is not positive-resolvable at any instant before $r^+_\perp(p_t)$, not negative-resolvable at any instant before $r^-_\perp(p_t)$, positive-resolvable at any instant after $r^-_\top(p_t)$, and negative-resolvable at any instant after $r^-_\top(p_t)$.

4.2 Resolving Nested Interval Expressions

Using the rules in Section 4.1, we can now see why some of the two-level interval nesting cases shown in Figure 4 were determined to be unresolvable.

In cases 1.1, 3.1, 4.1, 5.1, and 6.1, the earliest instant at which the satisfaction of $\overline{[d_3, d_4]}q$ can be determined is d_4, which is beyond d_2, the latest instant for the satisfaction of $[d_1, d_2]p$. As a consequence, the expression $[d_1, d_2](\overline{[d_3, d_4]}q)$ cannot be resolved, which is why commitments whose conditions are propositions of this type are disadvantageous for the creditor.

Solution 3. To model Example 3, the hotel H makes a commitment to a customer c. The commitment is $\mathsf{C}(d, H, c, \mathsf{A}[d_1, d_1 + 24hrs](\overline{[d_2, d_2 + 7days]}q)$, where $d_1 + 24hrs < d_2$ because it is not spring break yet, $[d_1, d_1 + 24hrs]$ denotes the interval "today" (say, a day in July), $\overline{[d_2, d_2 + 7days]}$ denotes the interval when spring break happens, and q is an atomic proposition that denotes some offer that the coupon offers. In this case, $\overline{[d_2, d_2 + 7days]}q$ cannot be resolved at least until $d_2 + 7days$, and $[d_1, d_1 + 24hrs](\cdot)$ *has* to be resolved at most by $d_1 + 24hrs$. But since $d_1 + 24hrs < d_2 + 7days$, this condition cannot be resolved. Hence the commitment cannot satisfied. Formally, $r^+_\top([d_1, d_1 + 24hrs](\cdot)) < r^+_\perp(\overline{[d_2, d_2 + 7days]}q)$. ∎

To summarize, the following conditions are necessary to ensure resolvability of a temporally quantified proposition:
A temporally quantified proposition of the form $[d_l, d_u]p_t$ must have at least one instant

in the interval d_l, d_u, at which p_t is resolvable. A temporally quantified proposition of the form $\overline{[d_l, d_u]}p_t$ must have p_t resolvable at all instants in the interval d_l, d_u.

For a commitment c, the following lemmas indicate the three valued logic of satisfaction due to the $satisfied(\cdot)$ and the $breached(\cdot)$ predicates.

Lemma 6.

$$M \models_m (create(x, c) \Rightarrow \mathsf{AF}(satisfied(c) \Rightarrow \neg active(c)) \ .$$

Proof. By the definition of the predicate $active(\cdot)$ and the semantic rule R_{24} .

Lemma 7.

$$M \models_m (create(x, c) \Rightarrow \mathsf{A}(breached(c) \Rightarrow \mathsf{G}active(c))) \ .$$

Proof. By the constraint C_5 and Lemma 2.

We have shown how unresolvable commitments can be detected. Such resolution results will enable earlier detection of protocol violations, and are of practical importance where an unresolvable commitment is as good (or as bad) as one that is breached.

4.3 Disjunctive Forms

Another important aspect of resolution concerns *disjunctive commitments* whose conditions are disjunctions of temporally-quantified propositions.

Disjunctive commitments regularly arise in common business interactions and can sometimes lead to what we call *the warranty paradox* — a situation where some subtle clauses render the warranty void before the customer can ascertain the quality of the good. This can happen, for instance, if ascertaining the quality of the good takes more time than the life of the warranty.

Intuitively, we reason as follows about the satisfiability of a disjunction of temporal propositions: *A disjunction of temporal propositions can potentially be satisfied if it has not already been satisfied, and at least one of the disjuncts is still resolvable.*

Let $p_1, p_2, \ldots p_n$ be temporally quantified propositions that occur in a disjunctive commitment of the form $\mathsf{C}(id, x, y, \mathsf{A}((p_1 \vee p_2 \ldots \vee p_i) \vee (p_{i+1} \vee p_{i+2} \vee \ldots \vee p_n))$. Here, $p_1 \vee p_2 \ldots \vee p_i$ represents some quality that the good satisfies, as claimed by the merchant x, and $p_{i+1} \vee p_{i+2} \vee \ldots \vee p_n$ represents the replacement for the good that the merchant promises to the customer y. If the quality assured by the merchant becomes false at some moment, then the replacement proposition should be positive-resolvable at that moment. Otherwise, the warranty is unfavorable for the customer.

Formally, this requirement is stated as

$$\begin{aligned} M \models_m \neg(p_1 \vee p_2 \ldots \vee p_i) \Rightarrow (\ & M \models_m (p_{i+1} \vee p_{i+2} \vee \ldots \vee p_n) \vee \\ & (\tau(m) < r^{+}_{\top}(p_{i+1} \vee p_{i+2} \vee \ldots \vee p_n) \wedge \\ & (M \not\models_m \neg(p_{i+1} \vee p_{i+2} \vee \ldots \vee p_n))) \end{aligned}$$

If none of the warranty disjuncts are resolvable at an instant at which the promise has not yet been satisfied, then the warranty is unfavorable to the customer.

Applying this requirement to Example 4 of Section 1 gives us the following.

Solution 4. Example 4 can be modeled by the commitment $\mathsf{C}(id, R, c, \mathsf{A}(\overline{[d_1, d_2]}\ good\_car \vee [d_1, d_3]replace\_car))$, where "$good\_car$" means the the car hasn't broken down, "$replace\_car$" represents the warranty that the rental company gives on the quality of the car, R represents the rental company, and c the customer. d_1 represents the time at which the car is rented on Friday, d_2 is the time at which the car should be returned on Monday, and d_3 denotes the closing of the rental company on Friday. Hence $d_3 < d_2$. We see that there exists a moment in between d_1 and d_3, at which the literal $([d_1, d_3]replace\_car)$ is beyond the upper bound of its positive-resolution. If the car breaks down on Saturday, then the only proposition that has not yet been resolved at that moment is the guarantee by the renter to replace it. However, the upper bound of positive resolvability of this proposition has passed. Formally, $\exists m : d_1 < d_3 < \tau(m) < d_3$ and $M \models_m \neg(\overline{[d_1, d_2]}great\_car)$ and $M \not\models_m ([d_1, d_3]replace\_car)$ and $r^+_\top([d_1, d_3]replace\_car) < \tau(m)$. ∎

Figure 5 shows Example 4.

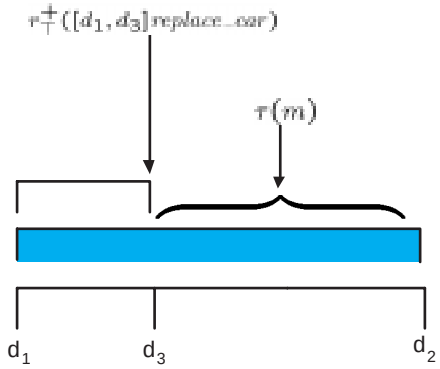

Fig. 5. Unfavorable warranty in Example 4

Thus we have shown how the warranty paradox can be captured in our scheme of temporal commitments.

5 Discussion

The concept of deadlines in commitments is doubtless necessary for practical uses of commitments. Traditionally, deadlines are hidden within the atomic propositions. However, an explicit formulation of temporal commitments, as developed above, is highly desirable. It offers a uniform treatment of operational characteristics across domains. We have shown how such a system of commitments with deadlines can be developed and used to reason about the possibility of satisfaction. Our approach not only allows for the expression of statements that involve deadlines, but also decouples the temporal

quantification from the proposition itself, thus allowing us to reason about the temporal aspect without regard to the meaning of the propositions. We now discuss related issues.

In Section 2.3, we assumed that all commitments are fulfilled by the DISCHARGE operation, and not just by the condition p holding. This assumption helps simplify the theory because it excludes a situation where a commitment $\mathsf{C}(d_1, x, y, p)$ is satisfied by p being brought about by some agent z rather than by agent x. However, the assumption does not weaken the scope of the commitment operations. We may require that only agent x bring about p. The domain language used to specify p should be rich enough to incorporate such constraints.

For instance, the condition p might be " turn the lights on", in which case any agent could turn the lights on to fulfill the commitment $\mathsf{C}(d_1, x, y, p)$. In fact, we might want to allow such a fulfillment. On the other hand, if the condition were subjective, like "teach a graduate course", we might want to make sure that only the intended party brings about the condition. We would therefore have to have a language for specifying p that can express that "*x* will teach a graduate course".

5.1 Literature

Literature related to our work can be classified according to two main orthogonal fields:

- The semantics of agent interaction protocols.
- The semantics and implementation of business processes.

A middle ground, where our work lies, is the operational aspects of commitments. Our work embodies desirable aspects of both the semantics and the business processes.

Semantics. Dignum *et al.* [9] describe a temporal deontic logic that helps specify obligations and constraints. The work focuses on specifying deadlines, so that a planner can take deadlines into account while generating plans. Their approach, however, is based on the notion of obligations, and no operational methods for obligations are given. Once a deadline has passed, and a certain rule has been violated, the logic has nothing to say about the effects on the system. This work, although semantically rich, has not been designed with an operational framework in mind. Nevertheless, Dignum *et al.*'s approach is detailed in the kinds of deadlines and constraints that it allows to be modeled. For example, deadlines like *. . . as soon as possible*, which cannot be modeled in our grammar, can be modeled using theirs. We have, however, a system that is closer to being operationalized than theirs.

Business Protocols. Grosof *et al.* [10] develop semantics for systems that represent business rules using *Courteous Logic Programs*. Their approach uses explicit rules to use to decide between conflicting rules, and the emphasis is on the implementation and application of CLP to real businesses. The work however does not define the semantics for the grammar they use. Business rules are represented by general if-then clauses. Hence, all concepts beyond the structure of the if-then rules are domain specific, including the temporal references. Their work, however, has been applied to actual business systems, and proves the value of intelligent agents in business processes.

From the standpoint of real-world implementation, the *Business Transaction Protocol* proposed by the OASIS [11] addresses the need for long-lived interactions among web services as opposed to short-lived transactions in the classical sense of the word. This protocol also recognizes that many of the offers made in real-world businesses involve deadlines. For example, an airline participating as a web service provider in a BTP run could specify how long it is willing to hold an offer open when it agrees to take part in a transaction [12].

Commitment Operations. Our scheme for temporal quantification is related to a similar notion presented by Fornara and Colombetti [2]. Their approach seeks to operationalize commitments and define the life cycle of a commitment, but does not pay attention to the issue of deadlines and temporally sensitive commitments, whereas our approach develops interesting results about the resolution of fulfillment of commitments. Although Fornara and Colombetti have taken a good first step towards the operationalization of commitments in agent interaction, they stop short of developing a semantics for temporal commitments as we have done here.

Verdiccio and Colombetti [13] develop a theory of the evolution of commitments over time. Their work is closest to ours among all others discussed in this section. It specifies constraints on the creation and satisfaction of temporal commitments using a variant of CTL that has a branching past and a branching future as opposed to our linear-past, branching-future temporal structure. Our scheme is simpler in the temporal structure used, more general in the examples we can model, and easier to understand.

McBurney and Parsons [14] define the *Posit Spaces Protocol*, which is an agent interaction protocol. This protocol uses a central repository for storing all the commitments that have been made. The idea is simple and corresponds to the concept of the *Sphere of Commitment* that was proposed by Singh [15]. This work does not relate to our results directly, but is yet another demonstration of the use of commitments in modeling and building multiagent systems.

5.2 Directions

Our work on temporal aspects of commitments is far from complete. One direction for further research is to investigate ways to do away with rigid protocols by having agents commit to each other by taking small risks at a time to finally arrive at a state where both parties are committed so that there is no risk of a loss to one party. This is just an intuitive notion, and further work is required to assess the viability of this approach.

Another direction is to describe agent interaction protocols using the theory of universal causation developed by Giunchiglia *et al.* [16], so that commitment machines that exploit nonmonotonic reasoning can be automatically generated. A commitment machine, proposed by Yolum and Singh [1], is a novel way of representing interaction protocols using commitments. It specifies the states that are allowed in the interaction in terms of the commitments that hold in those states. A commitment machine allows greater flexibility in the enactment of a protocol since interacting agents have many ways of reaching final states. Nonmonotonic reasoning in commitment machines would allow greater flexibility as compared to the incremental inferencing approach used by

Yolum and Singh [1]. Chopra and Singh [17] present this idea in greater detail and clarity.

Venkataraman and Singh [3] develop a vector-clock based scheme to verify agents' compliance to a commitment protocol. However, they do not consider rich temporal structures as we have done here. It will be interesting to apply the theory developed here to their scheme of compliance checking.

References

1. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: Applying event calculus planning using commitments. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), ACM Press (2002)
2. Fornara, N., Colombetti, M.: Operational specification of a commitment-based agent communication language. In: Proceedings of the International Joint Conference on Autonomous Agents and MultiAgent Systems, ACM Press (2002) 535–542
3. Venkatraman, M., Singh, M.P.: Verifying compliance with commitment protocols: Enabling open Web-based multiagent systems. Autonomous Agents and Multi-Agent Systems **2** (1999) 217–236
4. Partee, B.: Nominal and temporal anaphora. Linguistics and Philosophy **7** (1984) 287–324
5. Emerson, E.A.: Temporal and modal logic. In van Leeuwen, J., ed.: Handbook of Theoretical Computer Science. Volume B. North-Holland, Amsterdam (1990) 995–1072
6. Castelfranchi, C.: Commitments: From individual intentions to groups and organizations. In: Proceedings of the AAAI-93 Workshop on AI and Theories of Groups and Organizations: Conceptual and Empirical Research. (1993)
7. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law **7** (1999) 97–113
8. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM **26** (1983) 832–843
9. Dignum, F., Weigand, H., Verharen, E.: Meeting the deadline: On the formal specification of temporal deontic constraints. In Ras, Z.W., Michalewicz, M., eds.: Foundations of Intelligent Systems, 9th International Symposium, (ISMIS '96). Volume 1079 of Lecture Notes in Computer Science., Springer (1996) 243–252
10. Grosof, B.N., Labrou, Y., Chan, H.Y.: A declarative approach to business rules in contracts: Courteous logic programs in XML. In Wellman, M.P., ed.: Proceedings 1st Annual ACM Conf. on Electronic Commerce, EC'99, Denver, CO, USA, 3–5 November 1999, ACM Press (1999)
11. OASIS: Business transaction protocol (2002) www.oasis-open.org/committees/business-transactions/documents/specification/2002-06-03.BTP_cttee_spec_1.0.pdf.
12. Dalal, S., Temel, S., Little, M., Potts, M., Webber, J.: Coordinating business transactions on the web. IEEE Internet Computing **7** (2003) 30–39
13. Verdicchio, M., Colombetti, M.: A logical model of social commitment for agent communication. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), ACM Press (2003)
14. McBurney, P., Parsons, S.: Posit spaces: A performative model of e-commerce. In: Proceedings of the International Joint Conference on Autonomous Agents and MultiAgent Systems. (2003) To appear.
15. Singh, M.P.: Multiagent systems as spheres of commitment. In: Proceedings of the International Conference on Multiagent Systems (ICMAS) Workshop on Norms, Obligations, and Conventions. (1996)

16. Giunchiglia, E., Lee, J., McCain, N., Lifschitz, V., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence (AIJ) (2003) To Appear.
17. Chopra, A., Singh, M.: Nonmonotonic commitment machines. In Dignum, F., ed.: Advances in Agent Communication. LNAI, Springer Verlag (2003) in this volume