Commitments Among Agents

Ashok U. Mallya • *North Carolina State University* • *aumallya@ncsu.edu*
Michael N. Huhns • *University of South Carolina* • *huhns@sc.edu*

We sent one of our agents off to buy a book for us last month, but the book never arrived. The publisher claimed they had sent it, but the shipping company they chose could not find a record of it. Unfortunately, the credit card debit succeeded!

There appear to be several points of failure in this transaction, which we typically describe in terms of "blame." Software agents, however, require terminology that is more formal, which we can supply in the form of temporal commitments.

Commitments are a powerful representation for modeling multiagent interactions. Previous approaches have considered the semantics of commitments and how to check compliance with them (see sidebar). However, these approaches do not capture some of the subtleties that arise in real-life applications such as e-commerce, in which contracts and institutions have implicit temporal references. In this column, we describe a rich representation for the temporal content of commitments that lets us capture realistic contracts and avoid ambiguities. Consequently, this approach lets us reason about whether, and at what point, a commitment is satisfied or breached, and whether it is or ever becomes unenforceable.

Properties for Business Applications

Business contracts commonly 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 commitment-based approaches.

- *Time intervals*. Contracts often involve time bounds, which simplify decisions about the satisfaction or breach of commitments. Their existence in contracts is a significant reason traditional representations (such as paper documents)

rely on them. Practical commitments often must be satisfied either in a fixed, bounded interval or at a specified instant in the future.
- *Maintenance*. Current work on commitments has concentrated on conditions for their *achievement*, whereas real-life commitments are as likely to be about the *maintenance* of certain conditions. For example, a typical service-level agreement could involve committing to maintaining network connectivity during business hours.
- *Temporal anaphora*. A particular variety of time bounds arises in the notion of *temporal anaphora*, an implicit range of salient times within which a specified action occurred or will occur.[1] A promise such as "I will send you the goods," or a claim such as "I tried to call you five times," involves such a time range. 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 model time as a series of discrete moments rather than intervals, are commonly used in distributed system specifications. Such logics are inadequate to express the above requirements. Accordingly, we developed an extension of Emerson's Computation Tree Logic (CTL)[2] that can capture these cases of interest. In CTL, the world at every moment branches into all scenarios that are possible at that moment. Only one scenario actually happens and is called the *real* scenario, but we can reason about all possible scenarios.

Real-World Challenges

To motivate our study of temporal aspects of commitments, we use situational examples that arise in practical applications of Web services. (We present

solutions to these examples later.) 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 a hotel room at which to stay.

- *Example one.* The travel agent wants the passenger to fly on a particular day while still reserving the right to choose any flight on that day. If the airline offers such a deal, it becomes committed to maintaining a condition – a booked ticket – over an extended time period. We must be able to specify such maintenance conditions in commitments.
- *Example two.* The car rental company might offer a one-week free rental in January. This is a maintenance condition in another time period. We must be able to capture such temporal intricacies without bloating the domain language.
- *Example three.* Some commitments might 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 text on the coupon states that it can only be used during a future spring break.
- *Example four.* The car rental company might offer a warranty that cannot be used during the period in which the warranty is valid. The company might guarantee that its cars will not break down for at least two days, promising an immediate replacement if one does. However, if the company is closed on weekends, then a customer who rents a car on a Friday would not benefit from the warranty if the car broke down on Saturday.

Our method's main contribution to reasoning about commitments is that it applies a richer temporal representation and shows how to detect when a commitment is satisfied or breached. Further, it keeps the temporal aspects
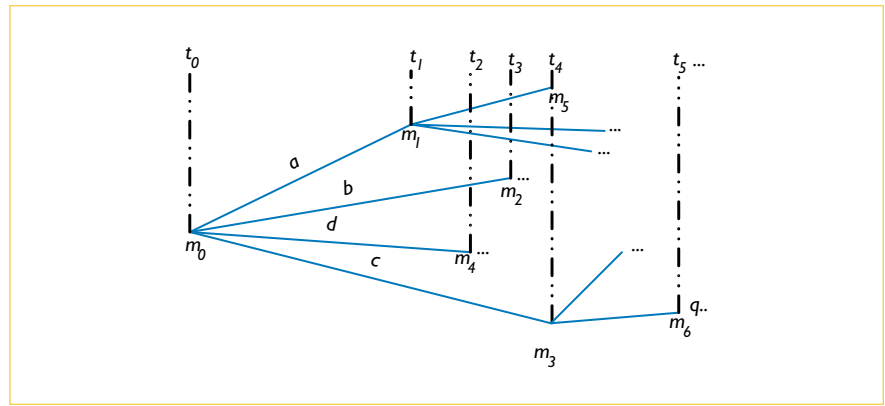

Figure 1. Schematic representation of our model of time. Using computation tree logic, our model defines the possible futures from a given moment in time, m_0.

independent of the domain-specific semantics of the condition that the commitment is about. Thus, we can reason about the temporal aspects in a domain-independent manner.

A Temporal Commitment Framework

Commitments are obligations that one agent has to another[3]. Formally, a commitment $C(d, x, y, p)$ 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 d. The commitment is satisfied when the condition p holds.

A framework's essential elements for representing and reasoning about commitments are, first, a way to describe commitments and operations on commitments. Second, because of their temporal nature, there must be a way to describe moments and time intervals. Third, because commitments extend into the uncertain future, there must be a way to describe alternative outcomes.

Describing Commitments

For the first requirement, the following operations create, satisfy, and transform commitments:

- `create(x, c)` establishes the commitment c in the system. This can only be performed by c's debtor x.
- `cancel(x, c)` cancels the com-

mitment c. This can only be performed by c's debtor x. Generally, making another commitment compensates cancellation.
- `release(y, c)` releases c's debtor x from commitment c. This only can be performed by the creditor y.
- `assign(y, z, c)` replaces y with z as c's creditor.
- `delegate(x, z, c)` replaces x with z as the c's debtor.
- `discharge(x,c)` c's debtor x fulfills the commitment.

The operators are applied to commitments during a time interval.

Describing Time Intervals

For the second requirement, we define two temporal quantifiers:

- *Existential.* $[t_1, t_2]p$ means that p is true at one or more moments in the interval beginning at t_1 and ending at t_2.
- *Universal.* $\overline{[t_1, t_2]}p$ means that p is true at every moment in the interval beginning at t_1 and ending at t_2.

To reason about future possibilities, we need to consider the different things that might occur during future time intervals.

Describing Alternative Outcomes

A branching-time logic, such as CTL (shown in Figure 1), satisfies the third requirement. It defines the following operators that apply over a particular *scenario* (defined as a path into the

future from a moment in time):

- *Until.* A statement *pUq*, read *p until q*, is true at a moment m_1 on a scenario if and only if *q* is true at some moment m_2 in the future and *p* is true at all moments from m_1 until m_2.
- *Eventually.* A statement *Fp*, read *eventually p*, means that *p* will be true at some point in the future of the given scenario.
- *Always.* A statement *Gp*, read *always p*, means that *p* is true at every moment on the given scenario.

The operator *Ap* denotes that *p* holds in all scenarios that are valid at the present moment.

Finally, we need three predicates to indicate whether a commitment *c* has been satisfied, breached, or still holds, written `satisfied(c)`, `breached(c)`, `holds(c)`, respectively. Also, `active(c)` is an abbreviation for a commitment that has not been cancelled, delegated, assigned, released, or discharged.

Using the Framework

With these definitions, we can define the semantics of the language for commitments, which for a model *M* will have rule statements such as this one that defines the `satisfied(c)` predicate:

$M \models_m satisfied(c)$ iff ($\exists m_3 : m_3 \preceq m$ and $M \models_{m3} discharge(x, c)$ and ($\exists m_1 : m_1 \prec m_3$ and $M \models_{m1} create(x, c)$ and ($\forall m_2 : m_1 \preceq m_2 \prec m_3 \Rightarrow M \models_{m2} active(c)$)))

This says that a commitment is satisfied if and only if it was created and later discharged by an agent *x*, and if it was continuously active between the time it was created and the time it was discharged. We can also impose constraints on the model, such as

$M \models_m (cancel(x, c) \Rightarrow AG\neg holds(c))$,

which states that a cancelled commitment no longer holds. Mallya and colleagues have described a complete semantics for this model elsewhere.[4]

Resolving Temporal Commitments

A temporal commitment is resolvable if its satisfaction or breach can be determined at some moment. Under certain conditions, we can ascertain the irresolvability of a temporal commitment even before the specified time interval occurs.

Solutions to examples one and two are straightforward; we can represent the temporal structure of the events in the examples using nested time intervals. We move on to the more interesting examples here.

Solution to Example Three

To model example three, the hotel *H* makes a commitment to a guest *g*. The commitment is $C(d, H, g, [t_1, t_1 + 24hrs]([t_3, t_3 + 7days]q)$, where $t_1 + 24$ hrs $< t_3$ because it is not Spring break yet; $[t_1, t_1 + 24hrs]$ denotes the interval "today" (say, a day in January); $[t_3, t_3 + 7days]$ denotes the interval when spring break happens; and *q* is an atomic proposition that denotes something that the coupon offers. In this case, $[t_3, t_3 + 7days]q$ cannot be resolved at least until $t_3 + 7days$, and $[t_1, t1 + 24hrs](\cdot)$ must be resolved at most by $t_1 + 24hrs$ for it not to be breached. But because $t_1 + 24hrs < t_3 + 7days$, this condition cannot be resolved. Hence, the commitment cannot be satisfied.

Solution to Example Four

We can model example four by the commitment $C(d, R, c, ([t_1, t_1 + 2days] great\_car \lor [t_1, t_2]replace\_car))$, where the literal *great_car* means the car has not broken down; the literal *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. In this model, t_1 represents the instant at which the car is rented on Friday and t_2 denotes the closing of the rental company on Friday. Hence $t_2 < t_1 + 2days$.

We see that there exists a moment in the set of all moments of the disjuncts at which every literal is either breached or is unresolvable because its time interval has passed. If the car breaks down a day after it was rented, that is, at $t_3 = t_1 + 1$ day, then the only proposition that is not yet breached at that point is the guarantee by the renter to replace it. However, the latest upper bound of the instant at which we could have ascertained the truth of this proposition has passed.

Formally, $\exists m_x, m_{early} \preceq m_x \preceq m_{late}$: $\forall i:(M \models_m breached(L_i)) \lor (\tau(m_x) > r_+(L_i))$. In this example, $\tau(m_{early}) = t_1$, $\tau(m_{late}) = t_1 + 2days$, i = 2, $L_1 = [t_1, t_1 + 2days] great\_car$, $L_2 = [t_1, t_2]replace\_car$, and the r_+ operator finds the latest instant at which the satisfaction of its proposition can be determined. The solution shows that the warranty is unfavorable to the customer.

Advantages of Deadlines

The concept of deadlines is necessary for practical use of commitments. Traditionally, deadlines have been hidden in the atomic propositions, but as we have shown, an explicit formulation of temporal commitments is highly desirable. It ensures uniform treatment of operational characteristics across domains. Our approach to developing such a system not only allows for the expression of statements that involve deadlines, but also decouples the temporal quantification from the proposition, allowing us to reason about the temporal aspect without regard to the propositions' meaning.

Future Directions

Our work on the temporal aspects of commitments is far from complete. In the larger scheme of things, agents would use commitments while interacting within the bounds of some protocol. Commitments between participants would shape the protocol and it, in turn, would constrain the commitments made within it. We therefore focus on agent interaction protocols and the flexibility in execution that commitments afford to these protocols.

A direction for further research is to

Related Literature

Our theory of deadlines is similar to that of Fornara and Colombetti,[1] which introduces the concept of a *commitment lifecycle*, explaining how operations on commitments create, modify, delete, and satisfy commitments. Their work is a good first step toward operationalizing commitments, but it does not focus on developing semantics for temporal commitments as we have done.

Our work lies in the middle ground between two orthogonal streams of research: the semantics of interaction protocols, and the implementation of business processes. We have embodied desirable properties of both streams in this theory.

Interaction Protocols
Dignum and colleagues describe a *temporal deontic logic* that helps specify obligations and constraints so that a planner can take deadlines into account while generating plans.[2] However, their approach is based on the notion of obligations, and it does not give operational methods for obligations. Once a deadline has passed and a certain rule has been violated, the logic has nothing to say about the effects on the system. Nevertheless, their approach is semantically rich and detailed in the kinds of deadlines and constraints it allows agents to model. For example, the deadline "as soon as possible," which cannot be modeled in our grammar, can be modeled using theirs. However, our system is closer to being implemented.

Business Processes
Grosof and colleagues' courteous logic programs (CLPs)[3] can be used to specify business rules and instructions for deciding which to use among a set of conflicting rules. However, they do not provide semantics for the grammar that a CLP uses. They represent rules by if-then clauses, and all concepts beyond the structure of these clauses are domain specific, including temporal references. CLPs, however, are used in actual business systems.

References
1. N. Fornara and M. Colombetti, "Operational Specification of a Commitment-Based Agent Communication Language," *Proc. Int'l Joint Conf. Autonomous Agents and MultiAgent Systems*, ACM Press, 2002, pp. 535–542.
2. F. Dignum, H. Weigand, and E. Verharen, "Meeting the Deadline: On the Formal Specification of Temporal Deontic Constraints," *Foundations of Intelligent Systems, 9th Int'l Symp.*, (ISMIS '96), vol. 1079, *Lecture Notes in Computer Science*, pp. 243–252. Springer, 1996.
3. B.N. Grosof, Y. Labrou, and H.Y. Chan, "A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML," *Proc. 1st Annual ACM Conf. Electronic Commerce*, (EC 99), ACM Press.

---

apply the theory developed here to Venkataraman and Singh's compliance-checking scheme,[5] in which agents participating in a commitment-based protocol use a vector-clock to keep track of events they can perceive. A protocol violation by any agent can then be detected by combining what each agent has recorded. The temporal structure that Venkataraman and Singh use is weaker than ours, and an application of our work to this scheme would be interesting.

Another direction is to incorporate dialogue game protocols (DGPs)[6] into commitment machines (CMs).[7] DGPs are based on intuitive notions about the nature of human dialogue – the intent of conversations, and the aim of argumentation. Although the theory behind them has been around since Aristotle's time, they have only recently been proposed as a more flexible alternative to game- and auction-theoretic protocols for agent interactions. A CM is a commitment-based protocol execution framework that allows more flexibility than traditional formalisms, such as finite state machines. DGPs can model appropriate phases of a CM protocol. Identifying the DGP classes that can model a certain phase would afford participants the convenience of choosing any of the allowed DGP classes. A sound theory of such a composition and its compliance aspects would greatly benefit workflow development and Web-service composition. ⊡

References
1. B. Partee, "Nominal and Temporal Anaphora," *Linguistics and Philosophy*, vol. 7, no. 3, Aug. 1984, pp. 287–324.
2. E.A. Emerson, "Temporal and Modal Logic," *Handbook of Theoretical Computer Science*, volume B, 1990, pp. 995–1072.
3. C. Castelfranchi, "Commitments: From Individual Intentions to Groups and Organizations," *Proc. AAAI'93 Workshop AI and Theories of Groups and Organizations: Conceptual and Empirical Research*, AAAI, 1993.
4. A.U. Mallya, P. Yolum, and M.P. Singh, "Resolving Commitments Among Autonomous Agents," to appear in *Proc. AAMAS'03 Workshop Agent Communication Languages and Conversation Policies*, ACM, 2003.
5. M. Venkataraman and M.P. Singh, "Verifying Compliance with Commitment Protocols: Enabling Open Web-Based Multiagent Systems," *Autonomous Agents and Multiagent Systems*, vol. 2, no. 3, Sept. 1999, pp. 217–236.
6. P. McBurney and S. Parsons, "Dialogue Game Protocols," to appear in *Proc. Comm. Multi-Agent Systems: Background, Current Trends and Future*, vol. 2650, Lecture Notes in Artificial Intelligence, Springer-Verlag, 2003, pp. 317-331.
7. P. Yolum and M.P. Singh, "Commitment Machines," *Proc. 8th Int'l Workshop Agent Theories, Architectures, and Languages* (ATAL 01). Springer-Verlag, 2001.

Ashok U. Mallya is a PhD candidate in the Department of Computer Science at North Carolina State University, from which he received an MS degree. His current research interest is agent interaction protocols.

Michael N. Huhns is a professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology.