ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-477

GOSU: computing GOal SUpport with commitments in multiagent systems

Özgür Kafalı[1] and **Akın Günay**[2] and **Pınar Yolum**[3]

Abstract. Goal-based agent architectures have been one of the most effective architectures for designing agents. In such architectures, the state of the agent as well as its goal set are represented explicitly. The agent then uses its set of actions to reach the goals in its goal set. However, in multiagent systems, most of the time, an agent cannot reach a goal only using its own actions but needs other agents to act as well. Commitments have been successfully used to regulate those interactions between agents. This paper proposes a framework and an environment for agents to manage the relations between their commitments and goals. More specifically, we provide an algorithm called GOSU to compute if a given set of commitments can be used to achieve a particular goal. We describe how GOSU can be implemented using the Reactive Event Calculus and demonstrate its capabilities over a case study.

1 Introduction

The Web is witnessing a shift of usership. The traditional Web has been meant to be used for humans to access Web pages. However, as the Web became a place to do business as well as daily activities, humans are in need of software to manage their tasks. The abstraction of an agent that can perceive the environment, reason on behalf of its user, and act as well as communicate in accordance with its users' goals is needed more than ever.

Various agent architectures exist. Among these, goal-based architectures have been especially useful in e-commerce, where the tasks that a user is taking can be mapped to goal representations. Different goal types, such as achievement or maintenance, have been identified and formalized in the literature [13]. The idea with goal-based architectures is that if the agent actions and goals are known, then the agent can act in order to achieve the goals. However, it is well-known now that no agent is an island. Agents must communicate and cooperate with others to satisfy certain goals. We capture these interactions as commitments [9]. Through the abstraction of goals and commitments two important aspects of cooperation can be addressed. First, each agent can represent and work toward its user's tasks and second each agent can interact and get help from other users if needed. By representing the user's goal, an agent can reason on its current state to check if the goals have been achieved and if not what actions need to be taken to achieve them. By representing, the user's commitments to others, it can manage the tasks that need to be fulfilled and the tasks that will be done by others in return.

[1] Department of Computer Science, Royal Holloway, University of London, United Kingdom, email: ozgur.kafali@rhul.ac.uk
[2] School of Computer Engineering, Nanyang Technological University, Singapore, email: akingunay@ntu.edu.sg
[3] Department of Computer Engineering, Bogazici University, Turkey, email: pinar.yolum@boun.edu.tr

Consider the dealings of a service provider with a user as a running example. The user has a goal of buying items and having them delivered in the following day. The user is not necessarily aware of the goals' of the service provider but there is a commitment between them that declares if the user pays a yearly premium, then the service provider will deliver the following day. Such dealings are common in e-commerce. When we, as humans are faced with such cases, we make inferences as to whether our existing commitments will enable us to reach our goal. When an agent faces this situation, it should do the same and decide whether its commitments are good enough to enable it to reach its goal.

This paper develops an algorithm for computing whether a given set of commitments are enough to satisfy an agent's goal. Each commitment in the set can be temporal, therefore representing the contractual agreements more realistically. Existing work that analyzes some of these relations have looked at the relations statically without taking into account the current state or constraints of the agent [8]. Contrary to that, here we represent the both current state of the agent and the constraints it has explicitly. Further, the commitments are associated with temporal properties to reflect many natural situations in real-life [1]. With the above setup, an agent can decide, whether it can achieve a particular goal, given its set of existing commitments, constraints, and current state.

The rest of this paper is organized as follows. Section 2 describes our technical framework with background on goals and commitments. Section 3 develops our algorithm for computing relations between goals and commitments. Section 4 examines the algorithms over a case study. Finally, Section 5 discusses the work in relation to related work.

2 Framework

A *goal* of an agent represents what the agent aims to achieve when acting in a multiagent system. For instance, the user might have the goal to receive an item by the next day that follows the purchase of the item. Each agent in a multiagent system has its own goals, which are not necessarily shared or agreed upon with others. The lifecycle of a goal has been studied before [10, 13]. In this paper we consider only *active* achievement goals of an agent. However, our work can be extended to work with a more complex goal lifecycle. An active goal is *satisfied* when the goal is achieved and *failed*, otherwise.

A *commitment* is a contractual binding between a debtor and a creditor. A commitment is denoted by $C(x, y, p_{ant}, p_{con})$ and states that the debtor agent x is committed to the creditor agent y to satisfy the consequent p_{con}, if the antecedent p_{ant} holds [9]. For instance, $C(provider, user, paid, delivered)$ denotes that the service provider is committed to the user to deliver an item (i.e., $delivered$

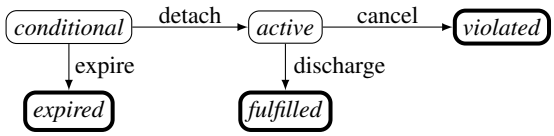

Figure 1. Lifecycle of a commitment.

holds), if the user purchases the item (i.e., $paid$ holds). While the goals of an agent can be private, a commitment between two agents is public such that both parties are aware of its existence.

Temporal constraints may be associated with the antecedent and consequent of a commitment, in order to capture real world situations, such as business contracts, more precisely. In this paper we consider interval constraints over the antecedents and consequents of commitments. These temporal constraints correspond to the fact that the antecedent or the consequent need to be realized during the associated interval. For instance, in order to get a discount, the user might have to accept an offer within 24 hours. In another example, the service provider might be committed to deliver a purchased item within three days after the payment.

The lifecycle of a commitment has been studied extensively in the literature, e.g., [1, 15]. Here, we use a simplified commitment lifecycle that is sufficient to study whether an agent can support a goal of interest with respect to its commitments. Figure 1 shows this lifecycle, where rounded rectangles represent the states of the commitment (bold ones are terminal states) and edge labels are the operations on the commitment. Initially, the commitment is created in *conditional* state. If the antecedent starts to hold (e.g., $provider$ gets paid), the commitment is detached and becomes *active*. If the antecedent fails to hold (e.g., $user$ does not pay), the commitment becomes *expired*. If the consequent starts to hold (e.g., the item is delivered on time), the commitment is discharged and becomes *fulfilled*. Finally, if the consequent fails to hold while the commitment is active (e.g., the item is not delivered on time), it becomes *violated*.

2.1 Formalization

In this section we formalize our framework elements (e.g., goals, commitments, etc.). Below, Φ is a set of propositional symbols and $\mathcal{L}_\Phi$ is a language of propositional logic over Φ, with operators $\wedge, \vee, \rightarrow, \neg$ in traditional semantics and symbols $\top$ and $\bot$ to denote true and false sentences, respectively. Agn is a set of agent identifiers and Act is a set of action symbols.

$P(\phi, t_s, t_e)$ denotes a *property* where ϕ is a disjunctive normal form formula in $\mathcal{L}_\Phi$ and $t_s, t_e \in \mathbb{Z}^+$. A property defines an interval constraint for the satisfaction of a propositional formula. Technically, the property is *satisfied*, if ϕ holds at some time t between t_s and t_e (i.e., $t_s \leq t \leq t_e$). If ϕ does not hold at a particular time between t_s and t_e, then the property is *pending*. On the other hand, if ϕ does not hold at any time between t_s and t_e, then the property is *failed*.

$A(x, p, a, \phi_e)$ denotes an *action* where $x \in Agn$ is the agent that can take the action $a \in Act$, if property p, which is the precondition, holds. ϕ_e is a conjunction in $\mathcal{L}_\Phi$ that represents the effect of this action. $G(p)$ denotes a *goal* where p is a property. The goal is *satisfied* if p is satisfied and *failed* if p is failed. $C(x, y, p_{ant}, p_{con})$ denotes a *commitment*. $x, y \in Agn$ are the debtor and creditor agents, respectively. p_{ant} and p_{con} are properties that represent the antecedent and consequent of the commitment, respectively.

An *agent* is a tuple $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C} \rangle$, where $x \in Agn$ is an agent identifier, $\mathcal{G}$ is a set of goals, $\mathcal{A}$ is a set of actions and $\mathcal{C}$ is a set of commitments. x is the unique identifier of the agent. $\mathcal{G}$ is the agent's goal set. $\mathcal{A}$ is the union of two disjoint action sets $\mathcal{A}_x \subseteq Act$ and $\mathcal{A}_{\bar{x}} \subseteq Act$. $\mathcal{A}_x$ consists of the actions that can be performed by x (i.e., $\forall A(y, p, a, \phi_e) \in \mathcal{A}_x : y = x$). $\mathcal{A}_{\bar{x}}$ consists of the actions that can be taken by the other agents (i.e., $\forall A(y, p, a, \phi_e) \in \mathcal{A}_{\bar{x}} : y \neq x$). Intuitively, the latter set captures the beliefs of x about the other agents' actions. Finally, $\mathcal{C}$ is the set of commitments that x enacts. Below, we use $\mathcal{P}^x$ as the set of all properties in a given agent x (i.e., properties considered in x's goals, actions and commitments).

Now, we define the semantics of an agent specification with respect to a transition system. Given $\mathcal{L}_\Phi$ and an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C} \rangle$, a *transition system* is a tuple $\langle S, s_0, T, \delta, L \rangle$, where:

- S is a set of states such that each state $s \in S$ is a composition of the following variables:
  - A variable for each proposition $\phi \in \Phi$ that captures the value of ϕ, which is equal to either $\top$ or $\bot$.
  - A variable for each $p \in \mathcal{P}^x$ that captures the state of p, which is equal to one of the values $Pending$, $Satisfied$ or $Failed$.
  - A variable for each goal $g \in \mathcal{G}$ that captures the state of g, which is equal to one of the values $Active$, $Satisfied$ or $Faied$.
  - A variable for each commitment $c \in \mathcal{C}$ that captures the state of c, which is equal to one of the following values $Conditional$, $Active$, $Expired$, $Fulfilled$ or $Violated$.
  - A variable $clock$ that represents the time associated to the state as an integer value.
- $s_0 \subset S$ is the initial state of the transition system.
- $T = Act$ is the set of transition labels that is identical to Act.
- $\delta : S \times Act \mapsto S$ is the transition function.
- L is a labeling function that assigns the values to the variables of the states in S.

Below, the entailment relation $s \models \phi$ denotes that the formula $\phi \in \mathcal{L}_\Phi$ holds in state s with respect to the labeling of the variables that correspond to the propositions of Φ in s. For convenience we use pSt, gSt, cSt, and clk functions to access the variables that capture the states of the properties, goals, commitments and clock in a given state s, respectively (e.g., $\mathsf{cSt}(s, c)$ is the value of the variable that represents the state of the commitment c in state s).

A transition $a \in T$ is enabled in a state s only if the precondition of the corresponding action in $\mathcal{A}$ holds in s. Technically, transition a is enabled in s, if $A(x, p, a, \phi_e) \in \mathcal{A}$ and $pST(p, s) = Satisfied$.

When a transition a from state s to s' happens, the labeling function L assigns the values of the variables in s' with respect to s and a as follows. The propositions in Φ are assigned to $\top$ and $\bot$ values with respect to the effects of a. The variable $clock$ in s' is set to $\mathsf{clk}(s)+1$. The value of a variable that capture the states of a pending property $p = P(\phi, t_s, t_e)$ is set by the following rules:

$$\frac{\mathsf{pSt}(p, s) = Pending \text{ and } s' \models \phi \text{ and } t_s \leq \mathsf{clk}(s') \leq t_e}{\mathsf{pSt}(p, s') \leftarrow Satisfied}$$

A pending property is satisfied, if the proposition of the property holds within its time interval.

$$\frac{\mathsf{pSt}(p, s) = Pending \text{ and } t_e < \mathsf{clk}(s')}{\mathsf{pSt}(p, s') \leftarrow Failed}$$

A pending property is failed to be satisfied, if the proposition of the property does not hold at any moment within its time interval (i.e., the property is still pending after t_e). $Satisfied$ and $Failed$ are terminal states for a property.

The value of a variable that capture the state of a goal $g = G(p)$ is set by the following rules:

$$\frac{\mathsf{gSt}(g,s) = Active \text{ and } \mathsf{pSt}(p,s') = Satisfied}{\mathsf{gSt}(g,s') \leftarrow Satisfied}$$

An active goal is satisfied, if the property of the goal is satisfied.

$$\frac{\mathsf{gSt}(g,s) = Active \text{ and } \mathsf{pSt}(p,s') = Failed}{\mathsf{gSt}(g,s') \leftarrow Failed}$$

An active goal fails, if the property of the goal is failed. $Satisfied$ and $Failed$ states of a goal are terminal.

Finally, the values of the variables that capture the states of a commitment $c = C(x, y, p_{ant}, p_{con})$ is set by the following rules which correspond to the commitment lifecycle in Figure 1.

$$\frac{\mathsf{cSt}(c,s) = Conditional \text{ and } \mathsf{pSt}(p_{ant},s') = Satisfied}{\mathsf{cSt}(c,s') \leftarrow Active}$$

$$\frac{\mathsf{cSt}(c,s) = Conditional \text{ and } \mathsf{pSt}(p_{ant},s') = Failed}{\mathsf{cSt}(c,s') \leftarrow Expired}$$

A conditional commitment becomes active, if the antecedent is satisfied, and becomes expired, if the antecedent is failed to be satisfied.

$$\frac{\mathsf{cSt}(c,s) = Active \text{ and } \mathsf{pSt}(p_{con},s') = Satisfied}{\mathsf{cSt}(c,s') \leftarrow Fulfilled}$$

$$\frac{\mathsf{cSt}(c,s) = Active \text{ and } \mathsf{pSt}(p_{con},s') = Failed}{\mathsf{cSt}(c,s') \leftarrow Violated}$$

An active commitment becomes fulfilled, if the consequent is satisfied, and becomes violated, if the consequent is failed to be satisfied. $Expired$, $Violated$ and $Fulfilled$ are terminal commitment states.

2.2 Goal Support

Now, we are ready to define when an active goal of an agent is supported. Basically, a goal g of an agent x is supported, if g can be satisfied at some future moment as a result of the agents' actions. However, since other agents are autonomous, it is not rational to expect them to perform certain actions unless they are committed to do so. For instance, the service provider would not deliver items unless she is committed to do so. Accordingly, we first define when one agent's beliefs about other agents' actions are rational.

Rational belief constraint: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ over language $\mathcal{L}_\Phi$ and the corresponding transition system $\langle S, s_0, T, \delta, L\rangle$, the agent's beliefs are rational only if there exists a commitment $C(y, x, P(\psi, t_{a_s}, t_{a_e}), P(\phi, t_{c_s}, t_{c_e}))$ for each action $A(y, P(\psi', t'_s, t'_e), a, \phi') \in \mathcal{A}_{\bar{x}}$ such that $\psi \rightarrow \psi'$, $\phi' \rightarrow \phi$ and $t'_s \leq t_{a_s} < t_{a_e} \leq t'_e$.

That is, it is rational for x to believe that y will perform an action a, if y is the debtor of a commitment c, such that the effect of a implies the consequent of c and the antecedent of c implies the precondition of a. Hence, when c becomes active, a is enabled and moreover y is committed to do a.

Now, we define an accessibility relation between states, which essentially shows that it is possible to move from one state to another state via a given set of actions.

Algorithm 1: *bool* GOSU$(s, g, \mathcal{A}, \mathcal{C})$

Input: g, goal to check for support
Input: s, current state
Input: $\mathcal{A}$, set of actions
Input: $\mathcal{C}$, set of commitments
Output: $true$ if g is supported in s, $false$ otherwise

1 **if** $\mathsf{gSt}(g,s) = Satisfied$ **then**
2 **return** $true$;
3 **else if** $\mathsf{gSt}(g,s) = Failed$ **then**
4 **return** $false$;
5 **else**
6 **foreach** $A(y,p,a,\phi_e) \in \mathcal{A}$ *such that* $\mathsf{pSt}(p,s) = Satisfied$ **do**
7 $s' \leftarrow \mathsf{progress}(s,a,g,\mathcal{A},\mathcal{C})$;
8 **if** GOSU$(g,s',\mathcal{A},\mathcal{C})$ **then**
9 **return** $true$;
10 **return** $false$;

Accessible state: Given two states s and s', s' is accessible from s (denoted as $s \rightsquigarrow s'$), if there is a set of transitions such that $s \times a_i \times s_i \times \ldots \times a' \times s'$.

Finally, we define support for a goal in the context of commitments. The idea we capture is that, an agent with a goal can possibly reach its goal if it has commitments such that when the other agents involved in these commitments fulfill their actions, then the goal can be satisfied. For the other agents to fulfill their commitments, the agents should as well have the necessary actions, with the right temporal constraints (defined above as rational belief constraint).

Support: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ that satisfies the rational belief constraint over language $\mathcal{L}_\Phi$ and the corresponding transition system $\langle S, s_0, T, \delta, L\rangle$, an active goal $g \in G$ is *supported* in $s \in S$, if there exists a state s' that is accessible from s (i.e., $s \rightsquigarrow s'$) and $\mathsf{gSt}(g,s') = Satisfied$.

3 Computing Goal Support

Algorithm 1 proposes our procedure, which we call GOSU, to compute whether a goal g is supported in a given state s. GOSU is based on the definition of accessible state. Basically, GOSU checks whether there exists a state s' in which g is satisfied and s' is accessible from s. To realize this, GOSU uses depth-first search strategy.

GOSU has the following four input parameters: (i) g is the goal to check for support, (ii) s is the current state, (iii) $\mathcal{A}$ is the set of actions, and (iv) $\mathcal{C}$ is the set of commitments. GOSU returns $true$ if g is supported in s. Otherwise, it returns $false$.

GOSU first checks the situation of the goal in the current state of the agent. If g is already in the satisfied state in s, then there is no need to check for future states and GOSU returns $true$ (lines 1-2). Similarly, if g is already in the failed state in s, GOSU immediately returns $false$ (lines 3-4) since it is not possible to satisfy g any more in any future state that is accessible from s.

If g is neither satisfied nor failed in the current state s, GOSU starts to explore the states s' that are directly accessible from s. For this purpose, GOSU iterates over the actions in $\mathcal{A}$, which have a satisfied precondition in s (line 6). For each such action a, GOSU creates the state s' that is accessible from s as a result of performing a using the auxiliary progress function (line 7). This function uses the transition rules (see Section 2.1) to create s'. We do not repeat the details of this

function here for brevity. After s' is created, GOSU checks whether g is supported in s' (line 8). This recursive process goes on until a state s' is found in which g is satisfied. In this case, GOSU returns $true$ (line 9). This concludes that g is supported given the current context of the agent. On the other hand, if all the actions that can be performed in s are considered, but none of them reaches a state s' in which g is satisfied, GOSU returns $false$ (line 10). That is, there does not exist an accessible state s' from the current state s in which g is satisfied. This concludes that g is not supported in the agent's current context.

Next, we present formal properties of GOSU and provide proof sketches.

Proposition: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ that satisfies the rational belief constraint over language $\mathcal{L}_\Phi$ as input, GOSU *terminates*.

Proof sketch: GOSU is a depth-first search procedure that terminates if the state space is finite. The state space may be infinite in two situations: (i) there are infinitely many actions in $\mathcal{A}$, or (ii) there are cycles in the state space. We assume that $\mathcal{A}$ is finite. Hence, first situation is not possible. Moreover, the monotonically increasing clock variables in the states make them unique and prevent cycles which may occur due to non-monotonicity of propositional symbols. Hence, second situation is not possible either. Therefore, GOSU terminates.

Note that cycles may occur between commitments. However, those are eventually violated due to temporal constraints. If the property involved in such cyclic commitments affects agent's goal, then GOSU returns false. Moreover, agents can repeatedly take the same action causing loops. However, since our goal definition is temporal, those branches will terminate when goal becomes failed over time.

Soundness: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ that satisfies the rational belief constraint over language $\mathcal{L}_\Phi$ and the corresponding transition system $\mathcal{T}$, GOSU is sound if the following conditions hold:

- if GOSU$(g, s, \mathcal{A}, \mathcal{C})$ returns $true$, then g is supported in s of $\mathcal{T}$ with respect to the support definition (see Section 2.2),
- if GOSU$(g, s, \mathcal{A}, \mathcal{C})$ returns $false$, then g is not supported in s of $\mathcal{T}$ with respect to the support definition.

Proposition: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ that satisfies the rational belief constraint over language $\mathcal{L}_\Phi$ as input, GOSU is *sound*.

Proof sketch: Suppose that $S'_\mathcal{T}$ is the set of directly accessible states from s in transition system $\mathcal{T}$ that corresponds to $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ and S'_{GOSU} is the set of directly accessible states from s which is created by GOSU using progress function. The first condition does not hold only if GOSU creates some extra states (i.e., $S'_{\text{GOSU}} \backslash S'_\mathcal{T} \neq \emptyset$). The second condition does not hold only if GOSU does not create all states (i.e., $S'_\mathcal{T} \backslash S'_{\text{GOSU}} \neq \emptyset$). Neither first nor the second case is possible since progress function utilizes the same rules that are used by $\mathcal{T}$ to create accessible states.

Completeness: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ that satisfies the rational belief constraint over language $\mathcal{L}_\Phi$ and the corresponding transition system $\mathcal{T}$, GOSU is complete if the following conditions hold:

- if g is supported in s of $\mathcal{T}$ with respect to the support definition (see Section 2.2), then GOSU$(g, s, \mathcal{A}, \mathcal{C})$ returns $true$,
- if g is not supported in s of $\mathcal{T}$ with respect to the support definition, then GOSU$(g, s, \mathcal{A}, \mathcal{C})$ returns $false$.

Proposition: Given an agent $\langle x, \mathcal{G}, \mathcal{A}, \mathcal{C}\rangle$ that satisfies the rational belief constraint over language $\mathcal{L}_\Phi$ as input, GOSU is *complete*.

Proof sketch: Completeness can be proved in a similar manner to soundness.

4 Implementation and Case Study

We use the dealings of a service provider (*seller*) with a user (*buyer*) to demonstrate the workings of our approach. According to the contract among them, the seller commits to its prime customers (who pay a yearly premium) that their orders will be delivered within the following day. However, the seller requires payments to be confirmed before dispatching the items. In our scenario, the bank has the constraint that it confirms buyers' payments during weekdays only.

```
initiates(_, goalNotSupported, T):-
   \+ goalSupported(T).

terminates(_, goalNotSupported, T):-
   goalSupported(T).

initiates(exec(confirm(bank, buyer, Item)),
                confirmed(Item),
   T):-
      item(Item),
      (T mod 7) >= 1,
      (T mod 7) =< 5.

ccreate(
   exec(offer(Seller, Buyer, Item, Deadline)),
   c(T, Seller, Buyer, and(paid(Item), confirmed(Item)),
                       delivered(Item), Deadline),
   T):-
      prime(Buyer),
      item(Item).

ccreate(exec(offer(Bank, Buyer, Item, Deadline)),
   c(T, Bank, Buyer, paid(Item),
                     confirmed(Item), Deadline),
   T):-
      item(Item).
```

Listing 1. Domain model in $\mathcal{REC}$.

We have implemented a prototype of our framework using the Reactive Event Calculus ($\mathcal{REC}$), which is a tool for tracking commitments at run time [1]. The Event Calculus [7] is a logic for modelling events and their effects through time. This is a suitable logic to realise our transition system as well as describing an agent's context. Listing 1 presents a sample code fragment from $\mathcal{REC}$, showing how the agent's domain can be modelled as part of above scenario. Events and their can effects can be described using the $initiates/3$ and $terminates/3$ predicates in Prolog fashion ($head \leftarrow body$), e,g,. an event initiates a fluent at a specific time if the certain preconditions hold at that time. Similarly, an event can terminate the existence of a fluent. Note that, events and fluents correspond to the actions and propositions of our transition system described in Section 2.1, respectively. The current state of a agent can be queried using the $holds\_at/2$ predicate. For brevity, we omit the details of the EC formalisation here[4].

In our scenario, we describe the bank's constraint on confirming payments as a precondition of the $confirm$ event (see Listing 1). Note that $exec$ is the prefix to describe events. According to the initiates clause, the fluent $confirmed$ is only initiated between Monday and Friday ($(Tmod7) >= 1$ and $(Tmod7) =< 5$). Similarly, commitments are represented as fluents and they change state based on events. For example, the seller's commitment to the buyer is initiated using an offer event. Note that for this commitment to be created, the buyer has to be a prime customer as a precondition.

Next, we consider several cases for the scenario. The following is an example narrative of events that we can feed $\mathcal{REC}$ with. For simplicity, we treat time points as days in the following discussion.

[4] The complete implementation can be downloaded from `http://mas.cmpe.boun.edu.tr/ozgur/code.html`.

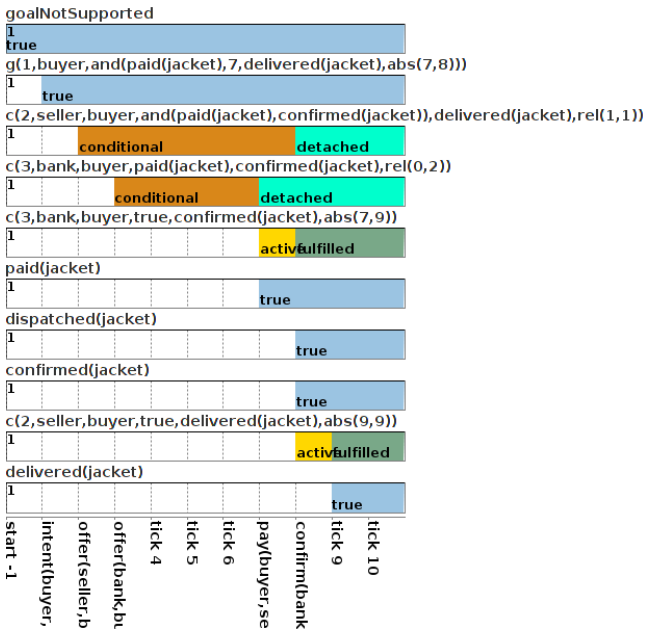

Figure 2. Goal not supported due to bank's constraint.

```
intent(buyer,and(paid(Item),7,
                 delivered(Item),abs(7,8))) 1
offer(seller,buyer,Item,rel(0,1)) 2
offer(bank,buyer,Item,rel(0,2)) 3
tick 4
tick 5
tick 6
pay(buyer,seller,jacket) 7
confirm(bank,buyer,jacket) 8
dispatch(courier,buyer,jacket) 8
tick 9
tick 10
```

First, the buyer's goal is created such that if she pays on Sunday, then she wants to receive the item by Monday. Then, the seller creates the commitment towards the buyer which states that paid and confirmed items are sent within one day (see Listing 1). Similarly, the bank creates the commitment towards the buyer. Following her goal, the buyer makes the purchase of a jacket on Sunday. However, due to the bank's constraint, the confirmation can only be done the following day. Upon confirmation of payment, the seller dispatches the buyer's order.

$\mathcal{REC}$ supports concurrent events ($confirm$ and $dispatch$ both happen at time 8). Moreover, we have added the functionality to support events that take time. For example, $dispatch$ initiates delivery in the next time point. When run with the above trace of events, $\mathcal{REC}$ produces the output shown in Figure 2. The horizontal axis shows the timeline of events that have occurred during execution. Notice a $tick$ event is associated with every non-occupied discrete time-point. This is required for $\mathcal{REC}$ to process properly since it is event-driven, i.e., a new event triggers $\mathcal{REC}$ to process further. The fluents are positioned vertically, and their truth values (and the corresponding states for commitments) are computed according to the events.

Now, let us see whether the buyer's goals is supported with respect to Algorithm 1. Note that, GOSU is executed for each state of the agent to see whether the goal is supported at that specific time point (see the progression of the fluent $goalNotSupported$ in Figure 2). Initially, the goal is not supported since there is no commitment created until time 2. That is, GOSU searches through every possible future state where the seller's delivery action would be in the rational belief constraint of the agent. But, there is no commitment towards delivery yet. Therefore, the goal is not supported. After the commitment is created, now the seller's delivery action will be in the agent's rational belief constraint. That is, there is an accessible state in which the fluent $delivered$ can become satisfied. However, due to the bank's constraint on confirmation of payments, it can only be satisfied at time 9, which exceeds the deadline for the agent's goal. Therefore, the goal becomes failed and the fluent $goalNotSupported$ stays false.

Let us now consider another case where the deadline of the commitment is extended to $[2, 5]$ as follows:

```
offer(seller,buyer,Item,rel(2,5)) 2
```

Again, the goal is not supported since the buyer's goal is not covered by the seller's commitment. If we go back to the first case and the bank's constraint is removed so that payments can be confirmed any day of the week, then the goal will be supported as soon as the commitment is created, i.e., the fluent $goalNotSupported$ becomes false from time 3 onwards (see Figure 3).

5 Discussion

GOSU is intended for run-time monitoring of goals. The agent does not need global knowledge of the interactions in the protocol. It verifies goal support via its commitments as well as beliefs about other agents' actions. We use $\mathcal{REC}$ for our prototype, because it is suitable for state-based approaches and run-time (distributed) verification.

Goals and commitments have both been studied extensively in the literature. Most work consider these concepts in isolation. Different types of goals and their characteristics have been identified in the literature [13]. The authors propose a formal framework to describe various goal types such as achievement and maintenance. Here, we only focus on achievement goal types where the goal is satisfied if it is realized at one single time point. However, that work assume the agents realize the goals on their own and does not study the link between goals and commitments. Works such as [16, 3] study the lifecycle of commitments, their verification, as well as ways to implement commitments in agent systems. However, they are not concerned about how these commitments are related to the agent goals.

More recent work study goals and commitments in relation to each other. Marengo *et al.*[8] define control of a proposition (which potentially serves as a goal for the agent) and safety of a commitment. An agent has control over a proposition, if it can realize the proposition, either on its own or by means of a commitment from another agent that has direct control of the proposition. The underlying idea is that if the agent has such control over a (goal) proposition, then the proposition is attainable. Our notion of support is similar, however we consider a set of commitments (rather than a single commitment) as well as the temporal constraints in computation. Conceptually, our work is on run-time and includes temporal commitments and goals.

Another relevant work connecting goals and contracts is that of Weigand *et al.*[14]. They focus on designing organizations for a set of goals, whereas we focus on the execution of such organizations from the point of view of each individual agent. We have temporal goals and contracts to mimic dynamism. Moreover, we propose an algorithm for computing goal support.

Two important works are the generation of a set of commitments to realize a goal that an agent has. Telang, Meneguzzi and Singh [11]

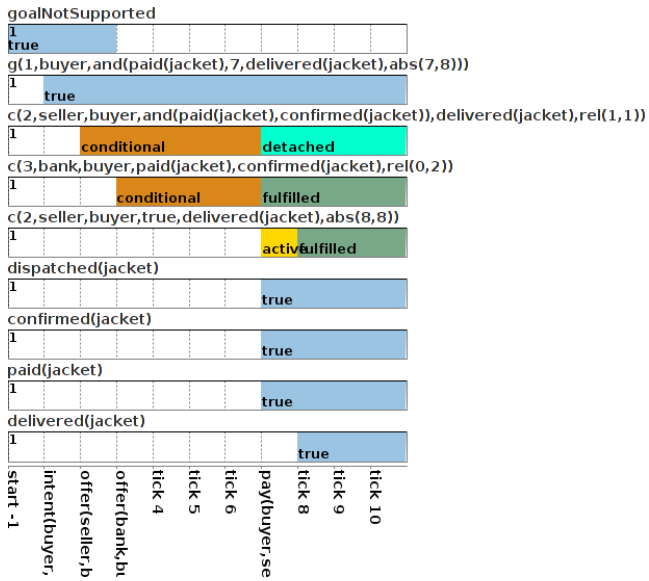

Figure 3. Goal supported (no constraint on confirmation of payments).

use Hierarchical Task Network (HTN) planning: given a set of agents and their goals, their objective is to come up with a global plan to satisfy these goals. The resulting plan is a set of commitments and the operations that are required to fulfill these commitments, which lead to the achievement of the agents' goals. By contrast, Günay, Winikoff, and Yolum [4] propose a distributed algorithm that can be run by any agent in the system to generate a commitment protocol, such that if the protocol is executed by the involved parties, the goal of the agent is realized. Neither of these two lines of work consider temporal aspects of commitments,

An important aspect of commitment support is the available resources. Günay and Yolum [5] incorporated resources into commitments and developed an algorithm to compute if the resources would be available to fulfill a set of commitments. The algorithm kept track of the available resources as well as the resources that will be made available through other commitments and used constraint satisfaction as a method to compute resource necessity. However, that work did not consider temporal commitments or constraints as we have done here. Hence, the two work can be thought as complementary.

Kafalı and Torroni consider exceptions in the context of temporal commitments [6]. They extend Chopra and Singh's work on misalignment [2] by integrating temporal aspects and effects of delegation. As a main theme, they identify what can go wrong in satisfying a commitment. The temporal aspects that are captured by Kafalı and Torroni are identical to those here, however their work does not have any notions of goals or their support.

This work opens up an interesting line of research. First, while this work has studied commitments in relation to achievement goals, understanding the dynamics of commitments in the existence of other goal types is crucial. Especially, maintenance goals play an important role in representing business policies. Hence, it would be useful to extend this work to handle such goals. Second, our framework here does not consider the suspension of goals or the conflicts that might exist between them [12]. Many times a goal can be suspended based on the context. When this is the case, commitments that serve to fulfill a goal need to be handled appropriately as well. This could require a commitment lifecycle that depends on a goal lifecycle. One other direction we are currently pursuing is the integration of trust and reputation within the relation of goals and commitments. This will enable us to make a more accurate judgment of which pending commitments are likely to be fulfilled. These are interesting directions that we will study in our future work.

ACKNOWLEDGEMENTS

This work has been supported by Bogazici University Research Fund under grant BAP 03A102P and by TUBITAK under grant 113E543.

REFERENCES

[1] Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni, 'Representing and monitoring social commitments using the event calculus', *Autonomous Agents and Multi-Agent Systems*, **27**(1), 85–130, (2013).

[2] Amit K. Chopra and Munindar P. Singh, 'Multiagent commitment alignment', in *Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 937–944, (2009).

[3] Mohamed El-Menshawy, Jamal Bentahar, Hongyang Qu, and Rachida Dssouli, 'On the verification of social commitments and time', in *Proceedings of the Tenth International Conference on Autonomous Agents and Multiagent Systems*, pp. 483–490, (2011).

[4] Akın Günay, Michael Winikoff, and Pınar Yolum, 'Dynamically generated commitment protocols in open systems', *Autonomous Agents and Multi-Agent Systems*, (2014). To appear.

[5] Akın Günay and Pınar Yolum, 'Constraint satisfaction as a tool for modeling and checking feasibility of multiagent commitments', *Applied Intelligence*, **39**(3), 489–509, (2013).

[6] Özgür Kafalı and Paolo Torroni, 'Exception diagnosis in multiagent contract executions', *Annals of Mathematics and Artificial Intelligence*, **64**(1), 73–107, (2012).

[7] R Kowalski and M Sergot, 'A logic-based calculus of events', *New Gen. Comput.*, **4**(1), 67–95, (1986).

[8] Elisa Marengo, Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Viviana Patti, and Munindar P. Singh, 'Commitments with regulations: reasoning about safety and control in REGULA', in *Proceedings of the 10th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 467–474, (2011).

[9] Munindar P. Singh, 'An ontology for commitments in multiagent systems', *Artificial Intelligence and Law*, **7**(1), 97–113, (1999).

[10] Pankaj R. Telang, Munindar P. Singh, and Neil Yorke-Smith, 'Relating goal and commitment semantics', in *Programming Multi-Agent Systems*, eds., Louise Dennis, Olivier Boissier, and RafaelH. Bordini, volume 7217 of *LNCS*, 22–37, Springer, (2012).

[11] R. Pankaj Telang, Felipe Meneguzzi, and Munindar P. Singh, 'Hierarchical planning about goals and commitments', in *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems*, AAMAS, pp. 877–884, (2013).

[12] M. Birna van Riemsdijk, Mehdi Dastani, and John-Jules Ch. Meyer, 'Goals in conflict: semantic foundations of goals in agent programming', *Autonomous Agents and Multi-Agent Systems*, **18**(3), 471–500, (2009).

[13] M. Birna van Riemsdijk, Mehdi Dastani, and Michael Winikoff, 'Goals in agent systems: A unifying framework', in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 713–720, (2008).

[14] Hans Weigand, Virginia Dignum, John-Jules Ch. Meyer, and Frank Dignum, 'Specification by refinement and agreement: Designing agent interaction using landmarks and contracts', in *Engineering Societies in the Agents World*, pp. 257–269, (2003).

[15] Pınar Yolum and Munindar P. Singh, 'Flexible protocol specification and execution: Applying event calculus planning using commitments', in *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*, pp. 527–534, (2002).

[16] Pınar Yolum and Munindar P. Singh, 'Enacting protocols by commitment concession', in *Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems*, pp. 116–123, (2007).