Prioritized Goals and Subgoals in a Logical Account of Goal Change – A Preliminary Report

Shakil M. Khan and Yves Lespérance

Department of Computer Science and Engineering
York University, Toronto, ON, Canada M3J 1P3
{skhan, lesperan}@cse.yorku.ca

Abstract. Most previous logical accounts of goal change do not deal with prioritized goals and do not handle subgoals and their dynamics properly. Many are restricted to achievement goals. In this paper, we develop a logical account of goal change that addresses these deficiencies. In our account, we do not drop lower priority goals permanently when they become inconsistent with other goals and the agent's knowledge; rather, we make such goals inactive. We ensure that the agent's chosen goals/intentions are consistent with each other and the agent's knowledge. When the world changes, the agent recomputes her chosen goals and some inactive goals may become active again. This ensures that our agent maximizes her utility. We also propose an approach for handling subgoals and their dynamics. We prove that the proposed account has some intuitively desirable properties.

1 Introduction

There has been much work on modeling agent's mental states, beliefs, goals, and intentions, and how they interact and lead to rational decisions about action. As well, there has been a lot of work on modeling belief change. But motivational attitudes and their dynamics have received much less attention. Most formal models of goals and goal change [1–7] assume that all goals are equally important and many only deal with achievement goals (one exception to this is the model of prioritized goals in [7]). Moreover, most of these frameworks do not guarantee that an agent's goals will properly evolve when an action is performed or an event occurs, e.g. when the agent's beliefs/knowledge changes or a goal is adopted or dropped. Also, they do not model the dependencies between goals and the subgoals and plans adopted to achieve these goals. For instance, subgoals and plans adopted to bring about a goal should be dropped when the parent goal becomes impossible, is achieved, or is dropped. Dealing with these issues is important for developing effective models of rational agency. It is also important for work on BDI agent programming languages, where handling declarative goals is an active research topic [8, 9].

In this paper, we present a formal model of prioritized goals and their dynamics that addresses some of these issues. Specifically, we propose a framework, where an agent can have multiple goals at different priority levels, possibly inconsistent with each other. We define intentions as the maximal set of highest priority goals that is consistent given the agent's knowledge. Our formalization of goals and goal dynamics ensures that the

agent strives to maximize her utility. Our model supports the specification of general temporally extended goals, not just achievement goals, and handles subgoals and their dynamics.

We start with a (possibly inconsistent) initial set of *prioritized goals* or desires that are totally ordered according to priority, and specify how these goals evolve when actions/events occur and the agent's knowledge changes. We define the agent's *chosen goals* or intentions in terms of this goal hierarchy. Our agents maximize their utility; they will abandon a chosen goal ϕ if an opportunity to commit to a higher priority but inconsistent with ϕ goal arises. To this end, we keep all prioritized goals in the goal base unless they are explicitly dropped. At every step, we compute an optimal set of chosen goals given the hierarchy of prioritized goals, preferring higher priority goals, such that chosen goals are consistent with each other and with the agent's knowledge. Thus at any given time, some goals in the hierarchy are active, i.e. chosen, while others are inactive. Some of these inactive goals may later become active, e.g. if a higher priority active goal that is inconsistent with the inactive goal becomes impossible. We also show how the dependencies between goals and subgoals can be modeled. Finally, we prove some interesting properties about the dynamics of chosen goals.

As mentioned above, our formalization of prioritized goals ensures that the agent always tries to maximize her utility, and as such a limitation of our framework is that it displays an idealized form of rationality. In Section 5, we discuss how this relates to Bratman's theory of practical reasoning [10]. We use an action theory based on the situation calculus [11] along with our formalization of paths in the situation calculus as our base formalism.

The paper is organized as follows: in the next section, we outline our base framework. In Section 3, we formalize *paths* in the situation calculus to support modeling goals. In Section 4, we present our model of prioritized goals. In section 5, we present our formalization of goal dynamics and discuss some of its properties. In Section 6, we discuss what it means for an agent to have a subgoal and how subgoals change as a result of changes to their parent goals. Then in the last section, we summarize our results, discuss previous work in this area, and point to possible future work.

2 Action and Knowledge

Our base framework for modeling goal change is the situation calculus [11] as formalized in [12]. In the situation calculus, a possible state of the domain is represented by a situation. There is a set of initial situations corresponding to the ways the agents believe the domain might be initially, i.e. situations in which no actions have yet occurred. Init(s) means that s is an initial situation. The actual initial state is represented by a special constant S_0. There is a distinguished binary function symbol do where $do(a, s)$ denotes the successor situation to s resulting from performing the action a. Thus the situations can be viewed as a set of trees, where the root of each tree is an initial situation and the arcs represent actions. Relations (and functions) whose truth values vary from situation to situation, are called relational (functional, resp.) fluents, and are denoted by predicate (function, resp.) symbols taking a situation term as their last argument. There is a special predicate Poss(a, s) used to state that action a is executable in situation s.

Our framework uses a theory D_{basic} that includes the following set of axioms:[1] (1) action precondition axioms, one per action a characterizing Poss(a, s), (2) successor state axioms (SSA), one per fluent, that succinctly encode both effect and frame axioms and specify exactly when the fluent changes [12], (3) initial state axioms describing what is true initially including the mental states of the agents, (4) unique name axioms for actions, and (5) domain-independent foundational axioms describing the structure of situations [14].

Following [15, 16], we model knowledge using a possible worlds account adapted to the situation calculus. $K(s', s)$ is used to denote that in situation s, the agent thinks that she could be in situation s'. Using K, the knowledge of an agent is defined as:[2] $\text{Know}(\Phi, s) \stackrel{\text{def}}{=} \forall s'.\ K(s', s) \supset \Phi(s')$, i.e. the agent knows Φ in s if Φ holds in all of her K-accessible situations in s. K is constrained to be reflexive, transitive, and Euclidean in the initial situation to capture the fact that agents' knowledge is true, and that agents have positive and negative introspection. As shown in [16], these constraints then continue to hold after any sequence of actions since they are preserved by the successor state axiom for K. We also assume that all actions are public, i.e. whenever an action (including exogenous events) occurs, the agent learns that it has happened. Note that, we work with knowledge rather than belief. Although much of our formalization should extend to the latter, we leave this for future work.

3 Paths in the Situation Calculus

To support modeling temporally extended goals, we introduce a new sort of *paths*, with (possibly sub/super-scripted) variables p ranging over paths. A path is essentially an infinite sequence of situations, where each successor situation along the path can be reached by performing some *executable* action in the preceding situation. We introduce a predicate OnPath(p, s), meaning that the situation s is on the path p. Also, Starts(p, s) means that s is the starting situation of path p. A path p starts with the situation s iff s is the earliest situation on p:[3]

Axiom 1 $\text{Starts}(p, s) \equiv \text{OnPath}(p, s) \wedge \forall s'.\ \text{OnPath}(p, s') \supset s \leq s'$.

In the standard situation calculus, paths are implicitly there, and a path can be viewed as a pair (s, F) consisting of a situation s representing the starting situation of the path, and a function F from situations to actions (called *Action Selection Functions* (ASF) or strategies in [5]), such that from starting situation s, F defines an infinite

[1] We will be quantifying over formulae, and thus we assume D_{basic} includes axioms for encoding of formulae as first order terms, as in [13]. We will also be using lists of integers, and assume that D_{basic} includes axiomatizations of integers and lists.

[2] A state formula $\Phi(s)$ takes a single situation as argument and is evaluated with respect to that situation. Φ may contain a placeholder constant *now* that stands for the situation in which Φ must hold. $\Phi(s)$ is the formula that results from replacing *now* by s. Where the intended meaning is clear, we sometimes suppress the placeholder.

[3] In the following, $s < s'$ means that s' can be reached from s by performing a sequence of executable actions. $s \leq s'$ is an abbreviation for $s < s' \vee s = s'$.

sequence of situations by specifying an action for every situation starting from s. Thus, one way of axiomatizing paths is by making them correspond to such pairs (s, F):

Axiom 2

$\forall p.\ \mathrm{Starts}(p, s) \supset (\exists F.\ \mathrm{Executable}(F, s) \wedge \forall s'.\ \mathrm{OnPath}(p, s') \equiv \mathrm{OnPathASF}(F, s, s'))$,

$\forall F, s.\ \mathrm{Executable}(F, s) \supset \exists p.\ \mathrm{Starts}(p, s) \wedge \forall s'.\ \mathrm{OnPathASF}(F, s, s') \equiv \mathrm{OnPath}(p, s')$.

This says that for every path there is an executable ASF that produces exactly the sequence of situations on the path from its starting situation. Also, for every executable ASF and situation, there is a path that corresponds to the sequence of situations produced by the ASF starting from that situation.

$$\mathrm{OnPathASF}(F, s, s') \stackrel{\mathrm{def}}{=} s \leq s' \wedge \forall a, s^*.\ s < do(a, s^*) \leq s' \supset F(s^*) = a,$$
$$\mathrm{Executable}(F, s) \stackrel{\mathrm{def}}{=} \forall s'.\ \mathrm{OnPathASF}(F, s, s') \supset \mathrm{Poss}(F(s'), s').$$

Here, $\mathrm{OnPathASF}(F, s, s')$ [6] means that the situation sequence defined by (s, F) includes the situation s'. Also, the situation sequence encoded by a strategy F and a starting situation s is executable iff for all situations s' in this sequence, the action selected by F in s' is executable in s'.

In our framework, we will use both state and path formulae. A state formula is a formula that has a free situation variable in it, whereas a path formula is one that has a free path variable. State formulae are used in the context of knowledge while path formulae are used in that of goals. We use $\Phi(s), \Psi(s), \cdots$ and $\phi(p), \psi(p), \cdots$ possibly with decorations to represent state and path formulae, respectively. Note that, by incorporating infinite paths in our framework, we can evaluate goals over these and handle arbitrary temporally extended goals; thus, unlike some other situation calculus based accounts where goal formulae are evaluated w.r.t. finite paths (e.g. [7]), we can handle for example unbounded maintenance goals.

We next define some useful constructs. A state formula Φ *eventually holds* over the path p if Φ holds in some situation that is on p, i.e. $\Diamond\Phi(p) \stackrel{\mathrm{def}}{=} \exists s'.\ \mathrm{OnPath}(p, s') \wedge \Phi(s')$. Other Temporal Logic operators can be defined similarly, e.g. always Φ: $\Box\Phi(p)$.

An agent *knows* in s that ϕ has become *inevitable* if ϕ holds over all paths that start with a K-accessible situation in s, i.e. $\mathrm{KInevitable}(\phi, s) \stackrel{\mathrm{def}}{=} \forall p.\ \mathrm{Starts}(p, s') \wedge K(s', s) \supset \phi(p)$. An agent knows in s that ϕ is impossible if she knows that $\neg\phi$ is inevitable in s, i.e. $\mathrm{KImpossible}(\phi, s) \stackrel{\mathrm{def}}{=} \mathrm{KInevitable}(\neg\phi, s)$.

Thirdly, we define what it means for a path p' to be a suffix of another path p w.r.t. a situation s:

$$\mathrm{Suffix}(p', p, s) \stackrel{\mathrm{def}}{=} \mathrm{OnPath}(p, s) \wedge \mathrm{Starts}(p', s)$$
$$\wedge\ \forall s'.\ s' \geq s \supset \mathrm{OnPath}(p, s') \equiv \mathrm{OnPath}(p', s').$$

Finally, $\mathrm{SameHistory}(s_1, s_2)$ means that the situations s_1 and s_2 share the same history of actions, but perhaps starting from different initial situations:

Axiom 3 $\mathrm{SameHistory}(s_1, s_2) \equiv \mathrm{Init}(s_1) \wedge \mathrm{Init}(s_2)$
$$\vee\ (\exists a, s_1', s_2'.\ s_1 = do(a, s_1') \wedge s_2 = do(a, s_2') \wedge \mathrm{SameHistory}(s_1', s_2')).$$

4 Prioritized Goals

Most work on formalizing goals only deals with static goal semantics and not their dynamics. There are two main categories of motivational attitudes, namely goal [1, 17] (AKA choice [2], wish [10] or preference), and intention. While goals are sometimes allowed to be inconsistent [10], intentions are mostly required to be consistent. Another difference is that agents are committed to their intentions, but not necessarily to their goals [10]. Intention is sometimes primitive [17, 3] and sometimes a defined concept, specified in terms of goals [1, 2, 4]. In this section, we formalize goals or desires with different priorities, which we call *prioritized goals* (p-goals, henceforth). These p-goals are not required to be mutually consistent and need not be actively pursued by the agent. Using this, we define the consistent set of *chosen goals* or intentions (c-goals, henceforth) that the agent is committed to. In the next section, we formalize goal dynamics by providing a SSA for p-goals. The agent's c-goals are automatically updated when her p-goals change. We deal with subgoals and their dynamics in Section 6.

Not all of the agent's goals are equally important to her. Thus, it is useful to support a priority ordering over goals. This information can be used to decide which of the agent's c-goals should no longer be actively pursued in case they become mutually inconsistent. Following [6], we specify each p-goal by its own accessibility relation/fluent G. A path p is G-accessible at priority level n in situation s (denoted by $G(p, n, s)$) if all the goals of the agent at level n are satisfied over this path and if it starts with a situation that has the same history (in terms of the actions performed so far) as s. The latter requirement ensures that the agent's p-goal-accessible paths reflect the actions that have been performed so far. A smaller n represents higher priority, and the highest priority level is 0. Thus in this framework, we assume that the set of p-goals are totally ordered according to priority. We say that an agent has the p-goal that ϕ at level n in situation s iff ϕ holds over all paths that are G-accessible at n in s:

$$\text{PGoal}(\phi, n, s) \stackrel{\text{def}}{=} \forall p.\ G(p, n, s) \supset \phi(p).$$

To be able to refer to all the p-goals of the agent at some given priority level, we also define *only p-goals*.

$$\begin{aligned}\text{OPGoal}(\phi, n, s) \stackrel{\text{def}}{=} & \text{PGoal}(\phi, n, s) \\ & \wedge \forall p, s'.\ \text{Starts}(p, s') \wedge \text{SameHist}(s', s) \wedge \phi(p) \supset G(p, n, s).\end{aligned}$$

An agent has the *only p-goal* that ϕ at level n in situation s iff ϕ is a p-goal at n in s, and any path over which ϕ holds and that starts with a situation that has the same action history as in s is G-accessible at n in s.

A domain theory for our framework D includes the axioms of a theory D_{basic} as in the previous section, the axiomatization of paths i.e. axioms 1-3, domain dependent initial goal axioms (see below), the domain independent axioms 4-7 and the definitions that appear in this section and the next. The modeler must provide initial goal axioms of the following form:

Initial Goal Axioms (a) $\text{Init}(s) \supset ((G(p, 0, s) \equiv \text{Starts}(p, s') \wedge \text{Init}(s') \wedge \phi_0(p))$
$\wedge\ (G(p, 1, s) \equiv \text{Starts}(p, s') \wedge \text{Init}(s') \wedge \phi_1(p)) \wedge \cdots$

$$\wedge\ (G(p, k-1, s) \equiv \text{Starts}(p, s') \wedge \text{Init}(s') \wedge \phi_{k-1}(p))),$$

(b) $\forall n, p, s.\ \text{Init}(s) \wedge n \geq k \supset (G(p, n, s) \equiv \text{Starts}(p, s') \wedge \text{Init}(s'))$,

(c) $\text{Init}(s) \supset \mathit{NPGoals}(s) = k$.

The p-goals $\phi_0, \phi_1, \cdots, \phi_{k-1}$ (from highest to lowest priority) of the agent in the initial situations are specified by the Initial Goal Axiom (a); each of them defines a set of initial goal paths for a given priority level, and must be consistent. We assume that the agent has a finite number k of initial p-goals. For $n \geq k$, we make $G(p, n, s)$ true for every path p that starts with an initial situation in (b). Thus at levels $n \geq k$, the agent has the trivial p-goal that she be in an initial situation. We also have a distinguished functional fluent $\mathit{NPGoals}(s)$ that represents the number of prioritized goals that the agent has (i.e. the location of the first empty slot after the last p-goal). Initially $\mathit{NPGoals}$ is set to k in (c). Later, we will specify the dynamics of p-goals by giving SSAs for G and $\mathit{NPGoals}$.

We use the following as a running example. We have an agent who initially has the following three p-goals: ϕ_0 = □BeRich, ϕ_1 = ◇GetPhD, and ϕ_2 = □BeHappy at level 0, 1, and 2, respectively (see second column of Table 1). Assume that while all

G-Level	S_0, S_1'	S_1	S_2	S_3
4	TRUE	TRUE	□BeRich ∧ □WorkHard ∧ □BeEnergetic	TRUE
3	TRUE	□BeRich ∧ □WorkHard	□BeRich ∧ □WorkHard	TRUE
2	□BeHappy	□BeHappy	□BeHappy	□BeHappy
1	◇GetPhD	◇GetPhD	◇GetPhD	◇GetPhD
0	□BeRich	□BeRich	□BeRich	□BeRich

Table 1. Example of an Agent's PGoals and their Dynamics

of her p-goals are individually achievable initially, her p-goal ◇GetPhD is inconsistent with her highest priority p-goal □BeRich as well as with her p-goal □BeHappy, while the latter are consistent with each other. It is straightforward to specify a domain action theory such that it entails this. Also assume that, after the action *goBankrupt* happens in S_0, the p-goal □BeRich becomes impossible. Thus in our example, we have $\text{OPGoal}(\phi_i(p) \wedge \text{Starts}(p, s) \wedge \text{Init}(s), i, S_0)$, for $i = 0, 1, 2$. Also, for any $n \geq 3$, we have $\text{OPGoal}(\text{Starts}(p, s) \wedge \text{Init}(s), n, S_0)$.

Using p-goals, we next define c-goals. While p-goals or desires are allowed to be known to be impossible to achieve, an agent's c-goals or intentions must be realistic. Not all of the G-accessible paths are realistic in the sense that they start with a knowledge accessible situation. To filter these out, we define *realistic* p-goal accessible paths:

$$G_R(p, n, s) \stackrel{\text{def}}{=} G(p, n, s) \wedge \text{Starts}(p, s') \wedge K(s', s),$$

i.e., a path p is G_R-accessible at level n in situation s if it is G-accessible at n in s, and if p starts with a situation that is K-accessible in s. Thus G_R prunes out the paths from G that are known to be impossible. We will define c-goals in terms of realistic p-goals, so this ensures that agents' c-goals are realistic.

The idea of how we compute c-goal-accessible paths is as follows: the set of G_R-accessibility relations represents a set of prioritized temporal propositions that are can-

didates for the agent's c-goals. Given G_R, in each situation we want to compute the agent's c-goals such that it is the *maximal consistent* set of higher priority realistic p-goals. We do this iteratively starting with the set of all possible paths (i.e. paths that starts with a K-accessible situation). At each iteration we compute the intersection of this set with the next highest priority set of G_R-accessible paths. If the intersection is not empty, we thus obtain a new chosen set of paths at level i. We call a p-goal chosen by this process an *active* p-goal. If on the other hand, the intersection is empty, then it must be the case that the p-goal represented by this level is either in conflict with another active higher priority p-goal/a combination of two or more active higher priority p-goals, or known to be impossible. In that case, that p-goal is ignored (i.e. marked as inactive), and the chosen set of paths at level i is the same as at level $i-1$. We repeat this until we reach $i = NPGoals$. Axiom 4 computes this intersection:[4]

Axiom 4 $G_\cap(p,n,s) \equiv$ **if** $(n=0)$ **then** $\text{Starts}(p,s') \wedge K(s',s)$
else if $\exists p'.(G_R(p',n-1,s) \wedge G_\cap(p',n-1,s))$
then $(G_R(p,n-1,s) \wedge G_\cap(p,n-1,s))$
else $G_\cap(p,n-1,s)$.

C-goal accessible paths are the result of this intersection after all priority levels have been considered:

$$G_C(p,s) \stackrel{\text{def}}{=} G_\cap(p, NPGoals(s), s).$$

We define an agent's c-goals in terms of the G_C-accessible paths:

$$\text{CGoal}(\phi,s) \stackrel{\text{def}}{=} \forall p.\ G_C(p,s) \supset \phi(p),$$

i.e., the agent has the c-goal that ϕ if ϕ holds over all of her G_C-accessible paths.

We also define what it means for an agent to have a c-goal at some level n:

$$\text{CGoal}(\phi,n,s) \stackrel{\text{def}}{=} \forall p.\ G_\cap(p,n+1,s) \supset \phi(p),$$

i.e. an agent has the c-goal at level n that ϕ if ϕ holds over all paths that are in the prioritized intersection of the set of G_R-accessible paths up to level n.

In our example, the agent's p-goals are $\Box$BeRich, $\Diamond$GetPhD, and $\Box$BeHappy in order of priority. The $G_\cap$-accessible paths at level 1 in S_0 are the ones that start with a K-accessible situation and where $\Box$BeRich holds. The $G_\cap$-accessible paths at level 2 in S_0 are the same as at level 1, since there are no K-accessible paths over which both $\Diamond$GetPhD and $\Box$BeRich hold. Finally, the $G_\cap$-accessible paths at level 3 in S_0 and hence the G_C-accessible paths are those that start with a K-accessible situation and over which $\Box\text{BeRich} \wedge \Box\text{BeHappy}$ holds. Also, it can be shown that initially our example agent has the c-goals that $\Box$BeRich and $\Box$BeHappy, but not $\Diamond$GetPhD.

Note that by our definition of c-goals, the agent can have a c-goal that ϕ in situation s for various reasons: 1) ϕ is known to be inevitable in s; 2) ϕ is an active p-goal at some level n in s; 3) ϕ is a consequence of two or more active p-goals at different levels in s. To be able to refer to c-goals for which the agent has a primitive motivation, i.e. c-goals that result from a single active p-goal at some priority level n, in contrast to those

[4] **if** ϕ **then** δ_1 **else** δ_2 is an abbreviation for $(\phi \supset \delta_1) \wedge (\neg\phi \supset \delta_2)$.

that hold as a consequence of two or more active p-goals at different priority levels, we define *primary* c-goals:

$$\text{PrimCGoal}(\phi, s) \stackrel{\text{def}}{=} \exists n.\ \text{PGoal}(\phi, n, s) \wedge \exists p.\ G_C(p, s) \wedge G(p, n, s).$$

That is, an agent has the primary c-goal that ϕ in situation s, if ϕ is a p-goal at some level n in s, and if there is a G_C-accessible path p in s that is also G-accessible at n in s. Thus if an agent has a primary c-goal that ϕ, then she also has the c-goal that ϕ, but not necessarily vice-versa. It can be shown that initially our example agent has the primary c-goals that □BeRich and □BeHappy, but not their conjunction.

5 Goal Dynamics

An agent's goals change when her knowledge changes as a result of the occurrence of an action (including exogenous events), or when she adopts or drops a goal. We formalize this by specifying how p-goals change. C-goals are then calculated using (realistic) p-goals in every new situation as explained above.

We introduce two actions for adopting and dropping a p-goal, $adopt(\phi)$ and $drop(\phi)$, and a third for adopting a subgoal ψ w.r.t. a supergoal ϕ, $adopt(\psi, \phi)$. The action precondition axioms for these are as follows:

Axiom 5 $\text{Poss}(adopt(\phi), s) \equiv \neg\exists n.\ \text{PGoal}(\phi, n, s)$,

$\text{Poss}(adopt(\psi, \phi), s) \equiv \neg\exists n.\ \text{PGoal}(\psi, n, s) \wedge \exists n'.\ \text{PGoal}(\phi, n', s)$,

$\text{Poss}(drop(\phi), s) \equiv \exists n.\ \text{PGoal}(\phi, n, s)$.

That is, an agent can adopt the p-goal that ϕ, if she does not already have ϕ as her p-goal at some level. An agent can adopt a subgoal ψ w.r.t. the parent goal that ϕ if she does not already have the p-goal that ψ at some level, and if at some level she currently has the parent goal that ϕ. The $drop(\phi)$ action is possible in s if ϕ is a p-goal at some level n in s.

In the following, we specify the dynamics of p-goals by giving the SSA for G and then discuss each case one at a time:

Axiom 6 (SSA for G) $G(p, n, do(a, s)) \equiv$

$\forall \phi, \psi.\ (a \neq adopt(\phi) \wedge a \neq adopt(\psi, \phi) \wedge a \neq drop(\phi) \wedge \text{Progressed}(p, n, a, s))$

$\vee\ \exists \phi.\ (a = adopt(\phi) \wedge \text{Adopted}(p, n, a, s, \phi))$

$\vee\ \exists \phi, \psi.\ (a = adopt(\psi, \phi) \wedge \text{SubGoalAdopted}(p, n, a, s, \psi, \phi)$

$\vee\ \exists \phi.\ (a = drop(\phi) \wedge \text{Dropped}(p, n, a, s, \phi))$.

The overall idea of the SSA for G is as follows. First of all, to handle the occurrence of a non-adopt/drop (i.e. regular) action a, we progress all G-accessible paths to reflect the fact that this action has just happened; this is done using the $\text{Progressed}(p, n, a, s)$ construct, which replaces each G-accessible path p' with starting situation s', by its suffix p provided that it starts with $do(a, s')$:

$$\text{Progressed}(p, n, a, s) \stackrel{\text{def}}{=} \exists p'.\ G(p', n, s) \wedge \text{Starts}(p', s') \wedge \text{Suffix}(p, p', do(a, s')).$$

Any path over which the next action performed is not a is eliminated from the respective G accessibility level.

Secondly, to handle adoption of a p-goal ϕ, we add a new proposition containing the p-goal to the agent's goal hierarchy. We assume that the newly adopted p-goal ϕ has the lowest priority. Thus in addition to progressing the G-accessible paths at all levels as above, we eliminate the paths over which ϕ does not hold from the $NPGoals(s)$-th G-accessibility level, and the agent acquires the p-goal that ϕ at level $NPGoals(s)$.

$$\text{Adopted}(p, n, a, s, \phi) \stackrel{\text{def}}{=} \textbf{if } (n = NPGoals(s)) \textbf{ then } (\text{Progressed}(p, n, a, s) \wedge \phi(p)) \textbf{ else } \text{Progressed}(p, n, a, s).$$

The third case of subgoal adoption is discussed in the next section.

Finally, to handle dropping of a p-goal ϕ, we replace the propositions that imply the dropped goal in the agent's goal hierarchy by the "trivial" proposition that the history of actions in the current situation has occurred. Thus in addition to progressing all G-accessible paths as above, we add back all paths that share the same history with $do(a, s)$ to the existing G-accessibility levels where the agent has the p-goal that ϕ, and thus these G-accessibility levels now amounts to the "trivial" p-goal that CorrectHist(s, *path*).[5]

$$\text{Dropped}(p, n, a, s, \phi) \stackrel{\text{def}}{=} \textbf{if } \text{PGoal}(\phi, n, s) \textbf{ then } \text{Starts}(p, s') \wedge \text{SameHistory}(s', do(a, s)) \textbf{ else } \text{Progressed}(p, n, a, s).$$

The SSA for $NPGoals(s)$ is as follows:

Axiom 7 (SSA for NPGoals) $NPGoals(do(a, s)) = k \equiv$

$$a \neq adopt(\phi) \wedge a \neq adopt(\psi, \phi) \wedge NPGoals(s) = k \vee$$
$$a = adopt(\phi) \wedge NPGoals(s) + 1 = k \vee$$
$$a = adopt(\psi, \phi) \wedge \text{AdjustSubGoalAdopt}(\phi, s) = k.$$

That is, when the agent adopts a p-goal, her current $NPGoals$ is incremented by one. We discuss the adjustment of $NPGoals$ required for subgoal adoption in the next section. Finally, $NPGoals$ is not affected by any other action.

Returning to our example, recall that our agent has the c-goals/active p-goals in S_0 that □BeRich and □BeHappy, but not ◇GetPhD, since the latter is inconsistent with her higher priority p-goal □BeRich. On the other hand, in $S_1' = do(goBankrupt, S_0)$, the agent has the c-goal that ◇GetPhD, but not □BeRich nor □BeHappy; □BeRich is excluded from the set of c-goals since it has become impossible to achieve (i.e. unrealistic). Also, since her higher priority p-goal ◇GetPhD is inconsistent with her p-goal □BeHappy, the agent will make □BeHappy inactive.

Note that, while it might be reasonable to drop a p-goal (e.g. ◇GetPhD) that is in conflict with another higher priority active p-goal (e.g. □BeRich), in our framework we keep such p-goals around. The reason for this is that although □BeRich is currently inconsistent with ◇GetPhD, the agent might later learn that □BeRich has become impossible to bring about (e.g. after *goBankrupt* occurs), and then might want to pursue

[5] CorrectHist(s, *path*) is defined as Starts(*path*, s') ∧ SameHistory(s', s); here *path* is a placeholder that stands for a path and s represents the current situation.

$\Diamond$GetPhD. Thus, it is useful to keep these inactive p-goals since this allows the agent to maximize her utility (that of her chosen goals) by taking advantage of such opportunities. As mentioned earlier, c-goals are our analogue to intentions. Recall that Bratman's model of intentions limits the agent's practical reasoning – agents do not always optimize their utility and don't always reconsider all available options in order to allocate their reasoning effort wisely. In contrast to this, our c-goals are defined in terms of the p-goals, and at every step, we ensure that the agent's c-goals maximizes her utility so that it is the set of highest priority goals that is consistent given the agent's knowledge. Thus, our notion of c-goals is not as persistent as Bratman's notion of intention [10]. For instance as mentioned above, after the action *goBankrupt* happens in S_0, the agent will lose the c-goal that $\Box$BeHappy, although she did not drop it and it did not become impossible or achieved. In this sense, our model is that of an idealized agent. There is a tradeoff between optimizing the agent's chosen set of prioritized goals and being committed to chosen goals. In our framework, chosen goals behave like intentions with an automatic filter-override mechanism [10] that forces the agent to drop her chosen goals when opportunities to commit to other higher priority goals arise. In the future, it would be interesting to develop a logical model that captures the pragmatics of intention reconsideration by supporting control over it.

We now show that our formalization of prioritized goals has some desirable properties. Some of these (e.g. Proposition 3a) are analogues of the AGM postulates; others (e.g. adopting logically equivalent goals has the same result, etc.) were left out for space reasons. First we show that c-goals are consistent:

Proposition 1 (Consistency) $D \models \forall s.\ \neg\text{CGoal}(\text{False}, s)$.

Thus, the agent cannot have both ϕ and $\neg\phi$ c-goals in a situation s and there is a path that is G_C-accessible in s. Even if all of the agent's p-goals become known to be impossible, the set of G_C-accessible paths will be precisely the ones that starts with a K-accessible situation, and thus the agent will only choose the propositions that are known to be inevitable.

We also have the property of realism [1], i.e. if an agent knows that something has become inevitable, then she has this as a c-goal:

Proposition 2 (Realism) $D \models \text{KInevitable}(\phi, s) \supset \text{CGoal}(\phi, s)$.

Note that this is not necessarily true for p-goals and primary c-goals – an agent may know that something has become inevitable and not have it as her p-goal/primary c-goal, which is intuitive. In fact, this is the reason why we define p-goals in terms of G-accessible paths rather than G_R. A consequence of Proposition 1 and 2 is that an agent does not have a c-goal that is known to be impossible, i.e. $D \models \text{CGoal}(\phi, s) \supset \neg\text{KImpossible}(\phi, s)$.

We next discuss some properties of the framework w.r.t. goal change. Proposition 3 says that (a) an agent acquires the p-goal at some level n that ϕ after she adopts it, and (b) an agent acquires the primary c-goal (and c-goal) that ϕ after she adopts it in s, provided that she does not have the c-goal that $\neg\phi$ in s.

Proposition 3 (Adoption) (a) $D \models \exists n.\ \text{PGoal}(\phi, n, do(adopt(\phi), s))$,
(b) $D \models \neg\text{CGoal}(\neg\phi, s) \supset \text{PrimCGoal}(\phi, do(adopt(\phi), s))$.

We can also show that after dropping the p-goal that ϕ in s, an agent does not have the p-goal (and thus the primary c-goal) that ϕ.

Proposition 4 (Drop) $D \models \neg\exists n.\ \mathrm{PGoal}(\phi, n, do(drop(\phi), s))$.

Note that, this does not hold for CGoal, as ϕ could still be a consequence of two or more of her remaining primary c-goals.

The next few properties concern the persistence of these motivational attitudes. In the following, we prove a persistence property for achievement p-goals:

Proposition 5 (Persistence of Achievement PGoals)

$$D \models \mathrm{PGoal}(\Diamond\Phi, n, s) \wedge \mathrm{Know}(\neg\Phi(now), s) \wedge \forall\psi.\ a \neq drop(\psi) \supset \mathrm{PGoal}(\Diamond\Phi, n, do(a, s)).$$

This says that if an agent has a p-goal that $\Diamond\Phi$ in s, then she will retain this p-goal after some action a has been performed in s, provided that she knows that Φ has not yet been achieved, and a is not the action of dropping a p-goal. Note that, we do not need to ensure that $\Diamond\Phi$ is still known to be possible or consistent with higher priority active p-goals, since the SSA for G does not automatically drop such incompatible p-goals from the goal hierarchy.

For achievement chosen goals we have the following:

Proposition 6 (Persistence of Achievement Chosen Goals)

$$\begin{aligned} D \models\ & \mathrm{OPGoal}(\Diamond\Phi \wedge \mathrm{CorrectHist}(s), n, s) \wedge \mathrm{CGoal}(\Diamond\Phi, s) \\ & \wedge \mathrm{Know}(\neg\Phi(now), s) \wedge \forall\psi.\ a \neq drop(\psi) \wedge \neg\mathrm{CGoal}(\neg\Diamond\Phi, n-1, do(a, s)) \\ & \supset \mathrm{CGoal}(\Diamond\Phi, n, do(a, s)). \end{aligned}$$

Thus, in situation s, if an agent has the only p-goal at level n that $\Diamond\Phi$ and the correct history of actions in s has been performed, and if $\Diamond\Phi$ is also a c-goal in s (and thus she has the primary c-goal that $\Diamond\Phi$), then she will retain the c-goal that $\Diamond\Phi$ at level n after some action a has been performed in s, provided that:

- she knows that Φ has not yet been achieved,
- that a is not the action of dropping a p-goal,
- and that at level $n-1$ the agent does not have the c-goal that $\neg\Diamond\Phi$, i.e. $\Diamond\Phi$ is consistent with higher priority c-goals.

Note that this property also follows if we replace the consequent with $\mathrm{CGoal}(\Diamond\Phi, do(a, s))$ or $\mathrm{PrimCGoal}(\Diamond\Phi, do(a, s))$, and thus it deals with the persistence of (primary) c-goals. Note however that, it does not hold if we replace the OPGoal in the antecedent with PGoal; the reason for this is that the agent might have a p-goal at level n in s that ϕ and the c-goal in s that ϕ, but not have ϕ as a primary c-goal in s, e.g. n might be an inactive level because another p-goal at n has become impossible, and ϕ could be a c-goal in s because it is a consequence of two other primary c-goals. Thus even if $\neg\phi$ is not a c-goal after a has been performed in s, there is no guarantee that the level n will be active in $do(a, s)$ or that all the active p-goals that contributed to ϕ in s are still active.

We believe that the dropping of an unrelated p-goal will not affect persistence, and hence it should be possible to strengthen Proposition 5 and 6. Also, in the future we would like to generalize these two propositions to deal with arbitrary temporally extended goals.

6 Handling Subgoals

In this section, we deal with the dynamics of subgoals. As mentioned earlier, a subgoal must be dropped when the parent goal is dropped or becomes impossible. When adopting a subgoal ψ with respect to a supergoal ϕ, in addition to recording the newly adopted goal ψ, we need to model the fact that ψ is a subgoal of ϕ. This information can later be used to drop the subgoal when the parent goal is dropped. One way of modeling this is to ensure that the adoption of a subgoal ψ w.r.t. a parent goal ϕ adds new p-goals that contain *both this subgoal and this parent goal i.e.* $\psi \wedge \phi$ *at a lower priority than the parent goal* ϕ. This ensures that when the parent goal is dropped, the subgoal is also dropped. To see this, recall that to handle the dropping of a goal ϕ, we drop the p-goals at all G-accessibility levels that imply ϕ. Thus, if we drop the parent goal ϕ, it will also drop all of its subgoals including ψ, since the G-accessibility levels where the parent goal ϕ holds include the G-accessibility levels where the subgoal ψ holds. Note that, if there are more than one level where the supergoal ϕ is a p-goal, then we copy all these levels, i.e. for each level n where ϕ is a p-goal, we add a (lower priority) level to the goal hierarchy. As we will see, this ensures that the sub-subgoals and sub-sub-subgoals etc. are also properly dropped when the supergoal is dropped. Also, this means that dropping a subgoal does not necessarily drop the supergoal.

Before going over the formal details, let us mention some useful bookkeeping tools that we will use: Length(l) returns the number of elements in list l; Nth(l, i) returns the i-th element in list l, and -1 if $i >$ Length(l); Sort(l) returns a sorted version of list l. The part of the SSA for G that handles subgoal adoption is defined as follows:

$$\begin{aligned}\text{SubGoalAdopted}(p, n, a, s, \psi, \phi) \stackrel{\text{def}}{=} & (n < \mathit{NPGoals}(s) \wedge \text{Progressed}(p, n, a, s))) \vee \\ & (\mathit{NPGoals}(s) \leq n < \mathit{NPGoals}(s) + \text{Length}(\text{AddList}(\phi, s)) \\ & \quad \wedge \exists i, m.\ (n = \mathit{NPGoals}(s) + i \wedge m = \text{Nth}(\text{AddList}(\phi, s), i) \\ & \quad \wedge \text{Progressed}(p, m, a, s) \wedge \psi(p))) \vee \\ & (n \geq \mathit{NPGoals}(s) + \text{Length}(\text{AddList}(\phi, s)) \wedge \text{Progressed}(p, n, a, s)).\end{aligned}$$

That is, if the action involves the adoption of a subgoal ψ w.r.t. a supergoal ϕ, we adjust G to incorporate (possibly several) new p-goals. We will discuss each case in turn. First, note that the existing p-goals are just carried over by progressing them; this is handled by the first disjunct.

Secondly, we adjust G starting at level $\mathit{NPGoals}(s)$. We add a number of new levels that include the conjunction of the only p-goal and the subgoal at a lower priority for all the current only p-goals that imply the parent goal ϕ. For example, say at level i we have an OPGoal that ϕ_i and it implies the parent goal that ϕ; then we add at a lower priority the conjoined goal of the progressed version of ϕ_i and the subgoal ψ. Our formalization of this uses the abbreviation AddList(ϕ, s) which is a sorted list of levels such that the parent goal is implied by the only p-goal at this level. AddList is defined as: $\text{AddList}(\phi, s) \stackrel{\text{def}}{=} \text{Sort}([n \mid \text{PGoal}(\phi, n, s)])$. The length of this list indicates the number of lower priority goals that needs to be added. As discussed above, this ensures that the agent will drop the subgoal when the parent goal is dropped (but not necessarily vice-versa). Note that if this process adds two or more new p-goals to the

agent's goal hierarchy, we maintain the original ordering; e.g. suppose that the agent adopted ψ w.r.t. ϕ, that there are two G-accessibility levels m and n such that the agent has the only p-goal that ϕ_m at m and ϕ_n at n, that ϕ_m implies ϕ and ϕ_n implies ϕ, and that $n > m$. In that case, the SSA for G will add the p-goal $\phi_m \wedge \phi$ at level $NPGoals(s)$ and the p-goal $\phi_n \wedge \phi$ at level $NPGoals(s) + 1$.

Finally, all the remaining levels involving trivially true goals are just carried over by progressing them.

The part of the SSA for *NPGoals* that handles subgoal adoption is defined as follows:

$$\text{AdjustSubGoalAdopt}(\phi, s) \stackrel{\text{def}}{=} NPGoals(s) + \text{Length}(\text{AddList}(\phi, s)).$$

That is, when the agent adopts a subgoal w.r.t. a parent goal, her current *NPGoals* is incremented by the number of new p-goals adopted in this process.

Let us go back to our example. Suppose that the agent knows that one way of always being rich is to always work hard, which in turns can be fulfilled by always being energetic. Assume that with this in mind, our agent adopts the subgoal that $\Box$WorkHard w.r.t. the p-goal that $\Box$BeRich, and then adopts the sub-subgoal that $\Box$BeEnergetic w.r.t. the subgoal that $\Box$WorkHard starting in S_0. Then the agent's goal hierarchy in $S_1 = do(adopt(\Box\text{WorkHard}, \Box\text{BeRich}), S_0)$ should include the p-goal that $\Box$WorkHard and in $S_2 = do(adopt(\Box\text{BeEnergetic}, \Box\text{WorkHard}), S_1)$ should also include the p-goal that $\Box$BeEnergetic. According to the SSA for G, our agent's goal hierarchy in S_1 and in S_2 will be as in Table 1.[6] In S_0, the supergoal $\Box$BeRich holds at level 0 and thus $\text{AddList}(\Box\text{BeRich}, S_0) = [0]$. Similarly in S_1, the supergoal $\Box$WorkHard holds at level 3 and thus $\text{AddList}(\Box\text{WorkHard}, S_1) = [3]$. Now, suppose that in S_2 the agent wants to drop the p-goal that $\Box$WorkHard. Then in $S_3 = do(drop(\Box\text{WorkHard}), S_2)$, she should no longer have $\Box$BeEnergetic as a p-goal, but should retain the supergoal that $\Box$BeRich. After the agent drops the p-goal that $\Box$WorkHard, by the SSA for G we can see that all the G-accessible levels where $\Box$WorkHard holds will be replaced by the only p-goal that $\text{CorrectHist}(S_2, path)$ (see S_3 in Table 1). This shows that dropping $\Box$WorkHard results in the dropping of all of its subgoals (in this case $\Box$BeEnergetic), but that its parent goal $\Box$BeRich is retained.

We define the SubGoal relation as follows:

$$\begin{aligned}\text{SubGoal}(\psi, \phi, s) \stackrel{\text{def}}{=} \exists n.\ &\text{PGoal}(\phi, n, s) \wedge \neg\text{PGoal}(\psi, n, s) \\ &\wedge \forall n.\ \text{PGoal}(\psi, n, s) \supset \text{PGoal}(\phi, n, s).\end{aligned}$$

This says that ψ is a subgoal of ϕ in situation s iff there exists an G-accessibility level n in s such that ϕ is a p-goal at n while ψ is not, and for all G-accessibility levels in s where ψ is a p-goal, ϕ is also a p-goal. Note that, while our formalization of subgoal dynamics allows a subgoal to have multiple parents, in this definition we assume that a subgoal can't have more than one parent. In the future, we will work on relaxing this constraint.

We now discuss some properties concerning the dynamics of subgoals and the dependencies between a subgoal and its parent goal. Proposition 7 states that (a) an agent

[6] For simplicity in Table 1, we only show the agent's relevant p-goals rather than its only p-goals (which in addition reflect the actions that have been performed so far, i.e. $\text{CorrectHist}(s)$).

acquires the p-goal that ψ after she adopts it as a subgoal of another goal ϕ in s, provided that she has the p-goal at some level in s that ϕ, and (b) she also acquires the primary c-goal that ψ after she adopts it as a subgoal of ϕ in s, provided that she has the primary c-goal in s that ϕ, and that she does not have the c-goal that $\neg\psi$ in s.

Proposition 7 (Subgoal Adoption)

(a) $D \models \exists m.\ \mathrm{PGoal}(\phi, m, s) \supset \exists n.\ \mathrm{PGoal}(\psi, n, do(adopt(\psi, \phi), s))$,

(b) $D \models \mathrm{PrimCGoal}(\phi, s) \wedge \neg\mathrm{CGoal}(\neg\psi, s) \supset \mathrm{PrimCGoal}(\psi, do(adopt(\psi, \phi), s))$.

The next property says that after dropping the p-goal that ϕ in s, an agent does not have the p-goal (and thus the primary c-goal) that ψ, provided that ψ is a subgoal of ϕ in s.

Proposition 8 (Supergoal Drop)

$$D \models \mathrm{SubGoal}(\psi, \phi, s) \supset \neg\exists n.\ \mathrm{PGoal}(\psi, n, do(drop(\phi), s)).$$

As with Proposition 4, this does not hold if we replace PGoal in the consequence with CGoal since ψ could be a consequence of a combination of other active p-goals.

The next two properties say that dropping a subgoal does not effect the parent goal.

Proposition 9 (Subgoal Drop)

(a) $D \models \mathrm{SubGoal}(\psi, \phi, s) \supset \exists n.\ \mathrm{PGoal}(\phi, n, do(drop(\psi), s))$,

(b) $D \models \mathrm{SubGoal}(\psi, \phi, s) \wedge \mathrm{PrimCGoal}(\phi, s) \supset \mathrm{PrimCGoal}(\phi, do(drop(\psi), s))$.

That is, (a) an agent retains the p-goal that ϕ after she drops a subgoal ψ of ϕ, and (b) she also retains the primary c-goal that ϕ after she drops a subgoal ψ of ϕ in s, provided that she has the primary c-goal that ϕ in s.

Finally, it can be shown that the SubGoal relation is transitive, i.e. if ψ_1 is a subgoal of ϕ in s, and if ψ_2 is a subgoal of ψ_1 in s, then ψ_2 must also be a subgoal of ϕ in s.

7 Discussion and Future Work

In this paper, we presented a formalization of prioritized goals, subgoals, and their dynamics. Our formalization ensures that an agent's chosen goals are always consistent and that goals and subgoals properly evolve as a result of actions and as a result of adopting and dropping goals. Although we made some simplifying assumptions, in this paper we have focused on developing an expressive framework that captures an idealized form of rationality without worrying about tractability. In would be desirable to study restricted fragments of the logic where reasoning is tractable. Also, before defining more limited forms of rationality, one should have a clear specification of what ideal rationality really is so that one understands what compromises are being done.

While in our account chosen goals are closed under logical consequence, primary c-goals are not. To this end, our formalization of primary c-goals is related to the non-normal modal formalizations of intentions found in the literature [3], and as such it does not suffer from the side-effect problem [1]. For instance, in our framework an agent can have the primary c-goal to get her teeth fixed and know that this always involves pain, but not have the primary c-goal to have pain.

Also, since we are using the situation calculus, we can easily represent procedural goals/plans, e.g. the goal to do a_1 and then a_2 can be written as: PGoal(Starts(p, s_1) $\wedge$ OnPath(p, s) $\wedge$ $s = do(a_2, do(a_1, s_1)), 0, S_0$). Golog [12] can be used to represent complex plans/programs. So we can model the adoption of plans as subgoals.

Recently, there have been a few proposals that deal with goal change. Shapiro *et al.* [18] present a situation calculus based framework where an agent adopts a goal when she is requested to do so, and remains committed to this goal unless the requester cancels this request; a goal is retained even if the agent learns that it has become impossible, and in this case the agent's goals become inconsistent. Shapiro and Brewka [7] modify this framework to ensure that goals are dropped when they are believed to be impossible or when they are achieved. Their account is similar to ours in the sense that they also assume a priority ordering over the set of (in their case, requested) goals, and in every situation they compute chosen goals by computing a maximal consistent goal set that is also compatible with the agent's beliefs. However, their model has some unintuitive properties: the agent's chosen set of goals in $do(a, s)$ may be quite different from her goals in s, although a did not make any of her goals in s impossible or inconsistent with higher priority goals, because inconsistencies between goals at the same priority level are resolved differently (this can happen because goals are only partially ordered). Note that, while one might argue that a partial order over goals might be more general, allowing this means that additional control information is required to obtain a single goal state after the agent's goals change. Also, we provide a more expressive formalization of prioritized goals – we model goals using infinite paths, and thus can model many types of goals that they cannot. Finally they model prioritized goals by treating the agent's p-goals as an arbitrary set of temporal formulas, and then defining the set of c-goals as a subset of the p-goals. But our possible world semantics has some advantages over this: it clearly defines when goals are consistent with each other and with what is known. One can easily specify how goals change when an action a occurs, e.g. the goal to do a next and then do b becomes the goal to do b next, the goal that $\Diamond\Phi \vee \Diamond\Psi$ becomes the goal that $\Diamond\Psi$ if a makes achieving Φ impossible, etc.

There has been much work on agent programming languages with declarative goals where the dynamics of goals and intentions and the dependencies between goals and sub-goals are modeled (e.g. [19], [9] and the references therein). However, most of these are not based on a formal theory of agency, and to the best of our knowledge, none maintains the consistency of (chosen) goals (e.g. when adopting a plan to achieve a goal, these frameworks do not ensure that this plan is consistent with the agent's other concurrent goals/plans). Also, most of these do not deal with temporally extended goals, and as a result they often need to accommodate inconsistent goal-bases to allow the agent to achieve conflicting states at different time points (e.g. the default logic based framework in [20]); chosen goals are required to be consistent. In [6], the authors present a situation calculus based agent programming language where the agent executes a program while maximizing the achievement of a set of prioritized goals. However, they do not formalize goal dynamics.

One limitation of our account is that we assume that the agent's p-goals are totally ordered in terms of priority. Also, newly adopted p-goals are assigned the lowest priority. A consequence of this is that an agent's c-goals depend on the adoption order of

her p-goals. For instance, given a fixed starting situation, an agent can end up with two different sets of c-goals by adopting ϕ followed by ψ, and by adopting ψ followed by ϕ. This has very different results when ϕ and ψ conflict with each other. We would like to address this by incorporating the priority of the p-goal as an argument to the *adopt* action, and handling this in the framework. Finally, one can argue that our agent spends too much resources trying to optimize her c-goals at every step. It would be interesting to develop an account where the agent is strongly committed to her chosen goals, and where the filter override mechanism is only triggered under specific conditions.

References

1. Cohen, P.R., Levesque, H.J.: Intention is Choice with Commitment. Artificial Intelligence **42**(2–3) (1990) 213–361
2. Sadek, M.D.: A Study in the Logic of Intention. In: Third Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR&R-92), Cambridge, MA (1992) 462–473
3. Konolige, K., Pollack, M.E.: A Representationalist Theory of Intention. In: Thirteenth Intl. J. Conf. on Artificial Intelligence (IJCAI-93), Chambéry, France (1993) 390–395
4. Singh, M.P.: Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications. Volume 799 of LNAI. Springer-Verlag, Germany (1994)
5. Shapiro, S., Lespérance, Y., Levesque, H.J.: Goals and Rational Action in the Situation Calculus - A Preliminary Report. In: Working Notes of the AAAI Fall Symposium on Rational Agency: Concepts, Theories, Models, and Applications, Cambridge, MA (November 1995) 117–122
6. Sardina, S., Shapiro, S.: Rational Action in Agent Programs with Prioritized Goals. In: Second Intl. J. Conf. on Autonomous Agents and Multi-Agent Sys. (AAMAS-03), Melbourne, Australia (2003) 417–424
7. Shapiro, S., Brewka, G.: Dynamic Interactions Between Goals and Beliefs. In: Twentieth Intl. J. Conf. on Artificial Intelligence (IJCAI-07), India (2007) 2625–2630
8. Winikoff, M., Padgham, L., Harland, J., Thangarajah, J.: Declarative and Procedural Goals in Intelligent Agent Systems. In: Eighth Intl. Conf. on Principles and Knowledge Representation and Reasoning (KR&R-02), Toulouse, France (2002) 470–481
9. Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E., eds.: Multi-Agent Programming: Languages, Platforms and Applications. Springer (2005)
10. Bratman, M.E.: Intentions, Plans, and Practical Reason. Harvard University Press, Cambridge, MA (1987)
11. McCarthy, J., Hayes, P.J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. Machine Intelligence **4** (1969) 463–502
12. Reiter, R.: Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge, MA (2001)
13. DeGiacomo, G., Lespérance, Y., Levesque, H.J.: ConGolog, a Concurrent Programming Language Based on the Situation Calculus. Artificial Intelligence **121** (2000) 109–169
14. Levesque, H.J., Pirri, F., Reiter, R.: Foundations for a Calculus of Situations. Electronic Transactions of AI (ETAI) **2**(3–4) (1998) 159–178
15. Moore, R.C.: A Formal Theory of Knowledge and Action. In Hobbs, J.R., Moore, R.C., eds.: Formal Theories of the Commonsense World. Ablex (1985) 319–358
16. Scherl, R., Levesque, H.: Knowledge, Action, and the Frame Problem. Artificial Intelligence **144**(1–2) (2003) 1–39

17. Rao, A.S., Georgeff, M.P.: Modeling Rational Agents with a BDI-Architecture. In Fikes, R., Sandewall, E., eds.: Second Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR&R-91), San Mateo, CA, Morgan Kaufmann Publishers (1991) 473–484
18. Shapiro, S., Lespérance, Y., Levesque, H.J.: Goal Change in the Situation Calculus. J. of Logic and Computation **17**(5) (2007) 983–1018
19. Sardina, S., deSilva, L., Padgham, L.: Hierarchical Planning in BDI Agent Programming Languages: A Formal Approach. In: Fifth Intl. J. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS-06), Hakodate, Japan (2006) 1001–1008
20. van Riemsdijk, M.B., Dastani, M., Meyer, J.J.C.: Semantics of Declarative Goals in Agent Programming. In: Fourth Int'l J. Conf. on Autonomous Agents and Multiagent Sys. (AAMAS-05). (2005) 133–140