Reasoning about the Intentions of Agents

Justin Blount and Michael Gelfond

Computer Science Department,
Texas Tech University,
Lubbock TX USA
{justin.blount,michael.gelfond}@ttu.edu

Abstract. In this paper we further develop the formal theory of intentions suggested by C. Baral and M. Gelfond in 2005. In this work the authors formalized the behavior of an agent intending to execute a sequence of actions. The resulting axioms for intentions written in Knowledge Representation language Answer Set Prolog allowed to easily express such properties of intentions as persistence and non-procrastination. This paper expands this work to allow reasoning with intentions in the presence of unexpected observations, and intentions to achieve goals. The theory is formulated in the extension of Answer Set Prolog, called CR-Prolog.

1 Introduction and Motivation

The paper is a contribution to the line of research aimed at developing theories of commonsense reasoning and acting in a changing environment. In particular we are interested in better understanding the mental attitudes of an agent, their relations with the physical world and with the agent's observations and actions. The subject has been of great interest to researchers in philosophy, logic, artificial intelligence and other areas for some time. There is a large body of literature addressing it from various perspectives. Some researchers concentrate on *conceptualization* — structuring the world in terms of some basic concepts or abstractions, clarifying the meaning of these concepts and studying their properties. The goal of this research is to better understand fundamental notions we use to think about the world such as beliefs, knowledge, defaults, causality, intentions, probability, etc., and to learn how one ought to use them for rational reasoning and acting. Others put their main efforts into *building agents* – entities which observe and act upon an environment and direct their activities towards achieving goals. This work led to a number of approaches to agent architecture which differ by underlying assumptions about properties of the domain and agent's tasks, languages used to represent the agent's knowledge, algorithms for solving different agent tasks such as planning, diagnostics, etc., and organization of the overall behavior of the agent. Even though both directions of research are closely interrelated there still is a substantial gap between them. This paper belongs to the growing number of works which attempt to combine achievements of both approaches. From the former we borrow the insight on the primary role of intentions for understanding the behavior of rational agents. The later gave us

a well developed AAA agent architecture (see, for instance, [1]), which assumes that the world can be viewed as a discrete dynamic system, models the agent's knowledge and reasoning methods in knowledge representation languages based on the answer set semantics [13], and organizes the agent's behavior by a simple observe-think-act loop.

As our own contribution we further develop the formal theory of intentions suggested in [7] and use the theory to improve the observe-think-act loop of AAA architecture. In [7] the authors formalized the behavior of an agent intending to execute a sequence of actions. The resulting axioms for intentions written in Answer Set Prolog allowed to easily express such properties of intentions as persistence and non-procrastination[1]. The theory has been used for question answering from natural language [16], for activity recognition [10] and other intelligent tasks. In many situations however viewing intention as a relation between an agent and a sequence of actions can be an oversimplification. For instance, the sequence of actions α intended for execution by an agent can be a plan for achieving a certain goal. If the goal were to be unexpectedly achieved thanks to some outside events it would be natural for an intelligent agent to stop persisting in his intention to complete the execution of α. Similarly, if the agent discovers that, due to the changed state of the world, completion of α did not lead to the achievement of his goal it will be rational to either try to find another plan or stop trying to achieve the goal all together. The following example, used throughout the paper, contain scenarios illustrating such behavior.

Example 1. [Bob meeting John]
Consider a row of four rooms, r_1,r_2,r_3,r_4 connected by doorways, such that an agent may move along the row from one room to the next. We say that two people *meet* if they are located in the same room. Assume that initially our agent Bob is in r_1 and he intends to meet with John who, as Bob knows, is in r_3. This type of intention is frequently referred to as an *intention to achieve* the goal. Bob's knowledge of the domain allows him to design a simple plan to achieve this goal: move from r_1 to r_2 and then to r_3. To fulfill his intention of meeting John Bob *intends to execute* the planned activity. If John remains in r_3 then, after the plan intended by Bob is executed, he and John will meet. Now suppose that, as Bob is moving from r_1 to r_2, John moves from r_3 to r_2. In this case it will be rational for Bob to recognize the unexpected achievement of his goal and not continue moving to r_3. Suppose now that Bob moved from $r1$ to $r2$ and then to $r3$, but John was not there. Bob must recognize that his plan failed. Further analysis however should allow Bob to conclude that, while he was executing his plan, John must have moved to $r4$. Since Bob is committed to achieving his goal, his intention to meet John persists, and Bob will come up with a new plan to meet John. With this new plan Bob will form and execute a new intended activity designed for this purpose (in our case simply move to

[1] We say that the agent does not procrastinate if he executes his intended actions the moment it becomes possible to do so. The agent is persistent in his intentions if he normally intends to perform an action even after the failure to do so. (for discussion of these properties see for instance [28]).

room $r4$). During the execution of his plan, Bob may abandon his intention to meet John in favor of achieving some other higher priority goal.

The theory from [7] is not sufficient for modeling this type of goal-oriented reasoning. *In this paper we present theory $\mathcal{I}$ of intentions, which simplifies [7] and expands it by including both, intentions to achieve goals and intentions to execute activities. The former establishes a relationship between beliefs and intentions of the agent with his planning process. The latter includes reasoning with unexpected observations and situations requiring replanning or abandoning the intention.* To achieve this we substantially modify the old theory. First we expand the notion of what is intended from action sequences to *goals* and *activities* defined as *a plan aimed at achieving a goal*. Second we expand the notion of a state of the domain by combining the *physical state* of the environment with the *mental state of the agent*. In the new theory the persistence axiom for intentions becomes a simple special case of the inertia axiom from [21]. The theory is formulated in the extension of Answer Set Prolog, called CR-Prolog [4]. The CR-Prolog inference engine *crmodels* [2] allows automatization of fairly subtle reasoning with intentions (including that discussed in Example 1). We believe that our approach allows to better understand properties of intentions — especially a classical problem of the relationship between agent's beliefs, intentions, and actions. The new theory can be used for specifying and reasoning about agents. It can also become a part of agent's architecture and be used by an agent to achieve its goals. Of course more needs to be done to fully incorporate these ideas into the AAA agent architecture. In particular we need to adopt our approach to agents with multiple, possibly prioritized, goals; design more efficient incremental inference engines for CR-Prolog which will be tailored toward consecutive calls with slightly expanding input knowledge bases, etc. But all this and more will, hopefully, be subject of future work. In the next section we will give a simple formalization of the dynamic domain from Example 1 in answer set programming (ASP) [20, 23, 5], and use it to provide the reader with the background necessary for understanding the AAA agent architecture. Section 3 contains precise description of the basic concepts and axioms of our theory $\mathcal{I}$ of intentions. The axioms are illustrated by reasoning about scenarios from Example 1. To simplify the presentation the material is divided into several parts describing different features of the theory. We conclude by the short discussion of related work.

2 The Background

We start with a brief description of a methodology for building agents [6] based on answer set programming (ASP) [20, 23]. In this approach, the reasoning and acting agents (referred to as ASP agents) view the world as a *discrete dynamic system* with a *transition diagram* whose nodes correspond to physically possible states of the system and whose arcs are labeled by actions. A *transition* $\langle \sigma, a, \sigma' \rangle$ indicates that the execution of action a in a state σ may move the system to state σ'. Thus paths of the transition diagram correspond to possible *trajectories* of the system.

In addition to actions such a system normally includes a number of objects of different sorts (e.g. people, areas, etc.) and relations between them (usually called fluents). The transition diagram of a discrete dynamic system is normally given by a collection of axioms of some action language [14], referred to as a *system description*. Axioms are later translated into their logic programming counterparts. The translations for different languages are well understood, so to limit the amount of background material needed to understand this paper we assume that a dynamic system that the agent uses to model its domain is represented directly in logic programming. We use the following example to clarify the details of such a representation. (We mostly follow the approach from [3] and [12] which contain logic programming rules corresponding to system descriptions of our favorite action language $\mathcal{AL}$.)

Example 2. [Bob's World]
Let us consider a formalization of Bob's world from Example 1. The world has two agents and four rooms which will be represented as follows:

$$agent(b).\ \ agent(j).\ \ room(1..4).$$

The next four rules define connectivity between rooms. We use possibly indexed variables A and R to range over agents and rooms respectively.

$$connected(1,2).\quad connected(2,3).\quad connected(3,4).$$
$$connected(R1,R2) \leftarrow connected(R2,R1).$$

To reason about trajectories of the system we need names for steps of such trajectories, defined as:

$$step(0..n).$$

Constant n indicates maximum length of a trajectory. In what follows we use possibly indexed variable I to range over steps. We also include the following rules describing Bob's physical actions, exogenous actions (i.e. actions not performed by Bob), and and fluents (relevant properties of the domain).

$$physical\_action(move(b,R1,R2)) \leftarrow connected(R1,R2).$$
$$exogenous\_action(move(j,R1,R2)) \leftarrow connected(R1,R2).$$

$$fluent(in(A,R), inertial).$$
$$fluent(meet(b,j), defined).$$

The first fluent, which holds if agent A is located in room R, is subject to the rule of inertia which says that *things normally stay as they are*. The use of inertia for representing effects of actions was first suggested in [21]. The ease of representing such defaults in Answer Set Prolog made possible a simple formalization of this

idea by the rules:

$$\begin{array}{ll} holds(F, I+1) \leftarrow & holds(F, I), \\ & not\ \neg holdsF, I+1), \\ & I < n, \\ & fluent(F, inertial). \\ \neg holds(F, I+1) \leftarrow & \neg holds(F, I), \\ & not\ holds(F, I+1), \\ & I < n, \\ & fluent(F, inertial). \end{array} \tag{1}$$

(Note that we are only numbering general, domain independent rules of our program.) Let us recall that in Answer Set Prolog $\neg$ represents so called *strong* or *classical negation*, i.e. $\neg p$ is read as "p is false"; the second negation, *not* is a non-monotonic default negation. Statement *not* p means that the agent has no reason to believe that p is true. Under this reading (captured by the Answer Set Semantics of logic programs) the first inertia axiom simply says that the agent must assume that F is true at $I+1$ if F is true at I and there is no reason to believe that F is false at $I+1$. The second fluent, $meet(b, j)$, holds at step I iff at that step both, b and j, are located in the same room. In our language this will be expressed by the rules:

$$\begin{array}{ll} holds(meet(b, j), I) & \leftarrow holds(in(b, R), I), \\ & \quad holds(in(j, R), I). \\ \neg holds(meet(b, j), I) & \leftarrow not\ holds(meet(b, j), I). \end{array}$$

In general, the second rule is subsumed by the Closed World Assumption (CWA) [26] for defined fluents given by:

$$\begin{array}{ll} \neg holds(F, I) \leftarrow & fluent(F, defined), \\ & not\ holds(F, I). \end{array} \tag{2}$$

We will also have the following descriptions of the *direct effects* of actions,

$$holds(in(A, R2), I+1) \leftarrow occurs(move(A, R1, R2), I).$$

state constraints, describing relationship between fluents,

$$\begin{array}{ll} \neg holds(in(A, R2), I) \leftarrow & holds(in(A, R1), I), \\ & R1 \neq R2. \end{array}$$

and *executability conditions*.

$$\begin{array}{ll} \neg occurs(move(A, R1, R2), I) & \leftarrow \neg holds(in(A, R1), I). \\ \neg occurs(move(A1, R1, R2), I) & \leftarrow occurs(move(A2, R1, R2), I), \\ & \quad A1 \neq A2. \\ \neg occurs(move(A1, R1, R2), I) & \leftarrow occurs(move(A2, R2, R1), I). \end{array}$$

The last two rules indicate that only one person can go through a door at a time. Finally, for a later discussion, we need the following rules describing exogenous action *delay* that prevents an agent from moving:

$$exogenous\_action(delay(A)).$$

$$\neg occurs(move(A, R1, R2), I) \leftarrow occurs(delay(A), I).$$

Let us denote the resulting program by Π^n. If $n = 1$ then we simply write Π. If σ is a collection of fluent literals (i.e. fluents and their negations) then

$$holds(\sigma, I) = \{holds(f, I) : f \in \sigma\} \cup \{\neg holds(f, I) : \neg f \in \sigma\}.$$

If a is a collection of elementary actions then

$$occurs(a, I) = \{occurs(e, I) : e \in a\},$$

and finally

$$\Pi(\sigma, a) = \Pi \cup holds(\sigma, 0) \cup occurs(a, 0).$$

A collection σ of fluent literals is a *state* of a transition diagram T defined by Π if for every inertial fluent f either f or $\neg f$ is in σ, a defined fluent is in σ iff it satisfies its definition (in our case $meet(b, j) \in \sigma$ iff there is some room r such that $in(b, r), in(j, r) \in \sigma$) and σ satisfies the state constraints of the program (in our case an agent cannot be located in more than one room). A triple $\langle \sigma, a, \sigma' \rangle$ is a *transition of* T if σ is a state and there is an answer set S of $\Pi(\sigma, a)$ such that $f \in \sigma'$ iff $holds(f, 1) \in S$ and $\neg f \in \sigma'$ iff $\neg holds(f, 1) \in S$. It is not difficult to show that σ' is a state of T. We hope that this brief description is sufficient to understand how transition diagrams can be defined by logic programs. For more accurate and general definitions of $\mathcal{AL}$ with defined fluents consult [12].

In addition to the system description given by a logic program the agent's memory contains the system's *recorded history* – the record of execution of all the actions performed by the agent together with a collection of truth values of fluents and occurrences of exogenous actions observed by the agent up to the current step j of the system's trajectory. The history is recorded using two new relations *obs* and *hpd*; $obs(F, V, I)$ *is true if at step* I *fluent* F *was observed to have truth value* V; $hpd(E, I)$ *is true if an event* E *was observed to have happened at* I. (Names different from *holds* and *occurs* are used to indicate that unlike the latter these statements can not be falsified by new information.) For instance, Bob's initial situation from Example 1 wrt the physical domain would contain the following statements describing Bob and John's initial positions:

$$\{obs(in(b, 1), true, 0), obs(in(j, 3), true, 0)\}.$$

To be able to reason about history of the system we need to expand our program Π^n by the following axioms. The next four axioms are a new version of the Reality Check from [3]. They ensure that observations do not contradict the

agent's expectations. The first two of these have the form of Answer Set Prolog constraints (rules with empty heads), and are encoded by:

$$\begin{array}{l} \leftarrow obs(F, false, I), \\ \quad holds(F, I), \\ \quad current\_step(I1), \\ \quad I < I1. \\ \leftarrow obs(F, true, I), \\ \quad \neg holds(F, I). \\ \quad current\_step(I1), \\ \quad I < I1. \end{array} \tag{3}$$

They establish consistency of observations and expectations for the *past* history of the agent[2]. The first says that for every step I in the past and any fluent F, it is impossible to predict that F is true and to observe that it is false. (Similarly for the second rule.) The next two axioms

$$\begin{array}{lll} inconsistent\_obs(F, false, I) & \leftarrow & obs(F, false, I), \\ & & \quad holds(F, I). \\ inconsistent\_obs(F, true, I) & \leftarrow & obs(F, true, I), \\ & & \quad \neg holds(F, I). \end{array} \tag{4}$$

are new. They define relation $inconsistent\_obs(F, V, I)$ which is true when the observations made at the current step are inconsistent with expectations. Axioms 3 and 4 are important for diagnostics. The three remaining axioms are auxiliary.

$$\begin{array}{lll} occurs(E, I) & \leftarrow & hpd(E, I). \\ holds(F, 0) & \leftarrow & obs(F, true, 0). \\ \neg holds(F, 0) & \leftarrow & obs(F, false, 0). \end{array} \tag{5}$$

(We use possibly indexed variables E to range over all actions.)
A history H of length j, where $j \leq n$, recorded by the agent defines a collection of models – trajectories of length j believed by the ASP agent to be possible pasts of the system. This framework allows to accurately define the notions of plan and diagnosis and reduce planning, diagnostics, and other reasoning tasks of the agent to computing answer sets of logic programs. The background knowledge introduced in this section will be used in conjunction with the theory of intentions we proceed to develop in the rest of the paper.

3 Theory of Intentions

In this section we introduce a new theory $\mathcal{I}$ of intentions. The basic notions of our theory are that of *goal* and *activity*. An activity can be viewed as a pair consisting of a goal and a plan for achieving it. An activity can be decomposed into sub-activities, until at the bottom where we find elementary (or atomic) actions. For

[2] In addition to these axioms [3] contained similar axioms for the current step.

simplicity of presentation we assume that the goal consists of a single fluent, that all activities are performed by a single designated agent, and that fluents that can serve as goals of this agent are stored in the system description using relation *possible_goal*. A description of an activity will include its goal, plan – described by a sequence of components, and length. Here is an example:

Example 3. [Bob's Activity]
Consider Bob's world from Example 2. Bob's activity, m, will be represented as:

$$activity(m).$$
$$goal(m, meet(b, j)).$$
$$component(m, 1, move(b, r1, r2)).$$
$$component(m, 2, move(b, r2, r3)).$$
$$length(m, 2).$$

Here $goal(M, F)$ holds if F is a fluent which is the goal of activity M and $component(M, K, C)$ holds if C (elementary action or another activity) is the K'th component of the M's plan; $length$ is the number of the activity components. Note that indexing of components starts with 1.

Our theory of intentions is perhaps best understood in the context of the observe-think-act loop of the AAA architecture. An action of selecting a goal (performed by an agent or an outside controller) is recorded in the agent's history. The agent will use the theory of intentions together with the background information to find a plan to achieve this goal, and to form the intention to execute it. The agent proceeds by executing components of the intended activity and maintaining the past history of these executions together with observations which could have been made during this process. New observations may trigger cancellation of the intention, additional planning to achieve the intended goal, and creation of new intended activities.

During this process the agent's memory contains a system description in the form of logic program, Π^n, domain history up to the current step, theory $\mathcal{I}$ of intentions, all of the possible goals of the agent, and activities that the agent has created to achieve some of these goals. The memory can be updated by new activities formed as the result of planning and by the record of newly occurring actions and observations. We will often refer to the agent's knowledge about goals it is actively trying to achieve and activities it is actively trying to execute as its *mental state*. If a particular activity is *in_progress*, i.e. not finished, the mental state will define the next action of the activity to be executed. Otherwise the activity may be a success, a failure, or cancelled; *success* indicates that the goal of the activity has been achieved, $failure$ indicates that the goal of the activity has not been achieved even after the execution of all the activity's components, and *cancelled* holds if the goal of the activity has been abandoned. Theory $\mathcal{I}$ of intentions can be viewed as a collection of axioms defining the transformation of the agent's mental state.

To simplify the presentation we will divide our story into two parts. In the first part we assume that the agent's goal and the corresponding intended activity are already present in the mental state of the agent, and concentrate on

the axioms describing the regular process of fulfilling intention to execute this activity. The second part will deal with committing to goals, forming intended activities, and reacting to unexpected observations.

3.1 Fulfilling Intentions to Execute

Axioms in this section, referred to as Basic Axioms, will be given as regular rules of Answer Set Prolog, and could be run on any of the standard ASP solvers like *smodels*, *dlv* and *clingo* [19, 22, 11]. To use these solvers we need to restrict the possible length of its activities by some constant, say max_len, and define a new sort

$$index(-1..max\_len). \tag{6}$$

Defined fluent

$$fluent(active(M), defined). \tag{7}$$

indicates that the agent is currently intending to execute activity M. The current state of this process is described by inertial fluent,

$$fluent(status(M, K), inertial). \tag{8}$$

If $0 \leq K \leq L$ where L is the length of M then K is the index of the component of M that has most recently been executed; $K = -1$ indicates that activity M is inactive. The first axiom below says that the fluent $status(M, K)$ is a function of M, and the second is a default that says initially the agents activities are normally inactive.

$$\begin{array}{ll} \neg holds(status(M, K1), I) \leftarrow & holds(status(M, K2), I), \\ & K1 \neq K2. \\ holds(staus(M, -1), 0) \leftarrow & not\ \neg holds(status(M, -1), 0). \end{array} \tag{9}$$

In addition to the agent's capability to interact with its environment by executing physical actions, the agent's *mental* actions directly affect its mental state. We assume that the execution of a mental action is not accompanied by the execution of any other actions (agent or exogenous). This is encoded by the following executability condition:

$$\begin{array}{ll} \neg occurs(E, I) \leftarrow & mental\_action(E1), \\ & occurs(E1, I), \\ & E \neq E1. \end{array} \tag{10}$$

We believe that the assumption is reasonably natural since mental activities are usually much faster than physical ones and an agent trying to perform multiple mental activities simultaneously is bound to run into problems. Moreover, the assumption substantially simplifies our theory.
Activities are moved into and out of the mental state of the agent by mental actions *start* and *stop*.

$$\begin{array}{l} mental\_action(start(M)). \\ mental\_action(stop(M)). \end{array}$$

Action *start* sets the value of *status* to 0, and action *stop* returns the activity to a status of -1. These direct affects are given by the following axioms:

$$\begin{array}{rl} holds(status(M,0),I+1) & \leftarrow occurs(start(M),I). \\ holds(status(M,-1),I+1) & \leftarrow occurs(stop(M),I). \end{array} \tag{11}$$

There are also natural executability conditions for these actions. An agent can neither start an active activity, nor stop an inactive one. These are encoded by the next two axioms:

$$\begin{array}{rl} \neg occurs(start(M),I) & \leftarrow holds(active(M),I). \\ \neg occurs(stop(M),I) & \leftarrow \neg holds(active(M),I). \end{array} \tag{12}$$

The next axiom defines $active(M)$ in terms of *status*:

$$holds(active(M),I) \leftarrow \neg holds(status(M,-1),I). \tag{13}$$

An activity $M1$ that is the current component of an activity M is a current sub-activity of M. It is recursively defined in terms of *status* as follows:

$$fluent(curr\_subact(M1,M),defined).$$

$$\begin{array}{rl} holds(curr\_subact(M1,M),I) & \leftarrow component(M,K+1,M1), \\ & \quad holds(status(M,K),I). \\ holds(curr\_subact(M1,M),I) & \leftarrow holds(curr\_subact(M2,M),I), \\ & \quad holds(curr\_subact(M1,M2),I). \end{array} \tag{14}$$

An activity $M1$ that is a current sub-activity and its goal are said to be a *minor*. These definitions are encoded as follows:

$$\begin{array}{c} fluent(minor(M),defined). \\ fluent(minor(G),defined). \end{array}$$

$$\begin{array}{rl} holds(minor(M1),I) & \leftarrow holds(curr\_subact(M1,M),I). \\ holds(minor(G),I) & \leftarrow holds(minor(M),I), \\ & \quad goal(M,G). \end{array} \tag{15}$$

The stopping of an activity returns its current sub-activities and their goals to an inactive state. These additional affects are given by the following:

$$\begin{array}{rl} holds(status(M1,-1),I+1) & \leftarrow occurs(stop(M),I), \\ & \quad holds(curr\_subact(M1,M). \\ \neg holds(active(G1),I+1) & \leftarrow occurs(stop(M),I), \\ & \quad holds(curr\_subact(M1,M), \\ & \quad goal(M1,G1). \end{array} \tag{16}$$

The next inertial fluent from the mental state of the agent is $active(G)$. It is true at step I if at this step the agent intends to achieve goal G. (We use possibly indexed variable G to range over fluents that are the possible goals of the agent.)

$$fluent(active(G),inertial).$$

Initially the agent has no active goals. This is encoded by the following default:

$$\neg holds(active(G), 0) \leftarrow not\ holds(active(G), 0). \tag{17}$$

There are two actions, *select* and *abandon* a goal G which, in this paper, are viewed as usually performed by the agent's controllers. In a sense they are viewed as input to our theory of intentions. Action *select* activates the goal, and action *abandon* returns the goal to an inactive state.

$$\begin{array}{rl} holds(active(G), I+1) & \leftarrow occurs(select(G), I). \\ \neg holds(active(G), I+1) & \leftarrow occurs(abandon(G), I). \end{array} \tag{18}$$

There are also two natural executability conditions for these actions. An active goal cannot be selected, and an inactive goal cannot be abandoned. Moreover, in our theory we want to prevent the agent's outside controllers from micromanaging agent's activity. To achieve that we do not allow the abandoning of minor goals.

$$\begin{array}{rl} \neg occurs(select(G), I) & \leftarrow holds(active(G), I). \\ \neg occurs(abandon(G), I) & \leftarrow \neg holds(active(G), I). \\ \neg occurs(abandon(G), I) & \leftarrow holds(minor(G), I). \end{array} \tag{19}$$

The *active/inactive* state of a goal G of an activity M propagates to the goals of M's current sub-activities.

$$\begin{array}{rl} holds(active(G1), I) & \leftarrow holds(active(G), I), \\ & \quad goal(M, G), \\ & \quad holds(curr\_subact(M1, M), I), \\ & \quad goal(M1, G1). \\ \neg holds(active(G1), I) & \leftarrow \neg holds(active(G), I), \\ & \quad goal(M, G), \\ & \quad holds(curr\_subact(M1, M), I), \\ & \quad goal(M1, G1). \end{array} \tag{20}$$

To further describe the mental state of the agent executing activity M to achieve goal G it will be convenient to introduce several other fluents defined in terms of fluents $status(M)$ and $active(G)$. Execution of an active activity M is *cancelled* at step I if M's goal is no longer *active*.

$$fluent(cancelled(M), defined).$$

$$\begin{array}{rl} holds(cancelled(M), I) & \leftarrow holds(active(M), I), \\ & \quad \neg holds(active(G), I), \\ & \quad goal(M, G). \end{array} \tag{21}$$

Execution of an active and uncancelled activity M achieves *success* at step I if M's goal becomes true at I.

$$fluent(success(M), defined).$$

$$\begin{aligned} holds(success(M), I) \leftarrow\ & holds(active(M), I), \\ & \neg holds(cancelled(M), I), \\ & goal(M, F), \\ & holds(F, I). \end{aligned} \tag{22}$$

The next axiom returns goal G to an inactive state after being achieved.

$$\begin{aligned} \neg holds(active(G), I+1) \leftarrow\ & occurs(stop(M), I), \\ & goal(M, G), \\ & holds(success(M), I). \end{aligned} \tag{23}$$

If execution of an uncancelled activity M is completed at I but M's goal remains unfulfilled we say that M is a $failure$ at I.

$$fluent(failure(M), defined).$$

$$\begin{aligned} holds(failure(M), I) \leftarrow\ & length(M, K), \\ & holds(status(M, K), I), \\ & \neg holds(success(M), I), \\ & \neg holds(cancelled(M), I). \end{aligned} \tag{24}$$

Our definition of plans naturally leads to the assumption that *the failure of part of the plan indirectly causes the failure of the entire plan.* (This of course is not necessarily true in more complex situations, e.g. when conditional plans are allowed). This is encoded by the following rule. (Note that we only refer to failure of the activity which is neither cancelled nor unexpectedly successful).

$$\begin{aligned} holds(failure(M), I) \leftarrow\ & holds(status(M, K), I), \\ & component(M, K+1, M1), \\ & holds(failure(M1, I), \\ & \neg holds(success(M), I), \\ & \neg holds(cancelled(M), I). \end{aligned} \tag{25}$$

If an active activity is neither cancelled, a failure or a success then it is said to be *in_progress*.

$$fluent(in\_progress(M), defined).$$

$$\begin{aligned} holds(in\_progress(M, I)) \leftarrow\ & holds(active(M), I), \\ & \neg holds(cancelled(M), I), \\ & \neg holds(success(M), I), \\ & \neg holds(failure(M), I). \end{aligned} \tag{26}$$

A current sub-activity is *irrelevant* if one of its ancestors is a success, failure or is cancelled.

$$fluent(irrelevant(M), defined).$$

$$\begin{aligned} holds(irrelevant(M1, I)) \leftarrow\ & holds(curr\_subact(M1, M), I), \\ & holds(success(M), I). \\ holds(irrelevant(M1, I)) \leftarrow\ & holds(curr\_subact(M1, M), I), \\ & holds(failure(M), I). \\ holds(irrelevant(M1, I)) \leftarrow\ & holds(curr\_subact(M1, M), I), \\ & holds(cancelled(M), I). \end{aligned} \tag{27}$$

Another defined mental fluent, $intended\_action(M, E)$, is true at step I if mental or physical agent action E of M is intended for execution at I.

$$fluent(intended\_action(M, E), defined).$$

If the first not yet executed component of an active activity is an action then it is intended for execution. (We use possibly variable PA to range over physical actions of the agent.)

$$\begin{aligned} holds(intended\_action(M, PA), I) \leftarrow\ & holds(status(M, K), I), \\ & component(M, K+1, PA), \\ & holds(in\_progress(M), I), \\ & \neg holds(irrelevant(M), I). \end{aligned} \tag{28}$$

If the first not yet executed component of an active activity is a sub-activity, then the agent will intend to start this sub-activity.

$$\begin{aligned} holds(intended\_action(M1, start(M1)), I) \leftarrow\ & holds(status(M, K), I), \\ & component(M, K+1, M1), \\ & holds(in\_progress(M), I), \\ & \neg holds(active(M1), I), \\ & \neg holds(irrelevant(M), I). \end{aligned} \tag{29}$$

Relevant activities that are a success, failure, or cancelled should be stopped.

$$\begin{aligned} holds(intended\_action(M, stop(M)), I) \leftarrow\ & holds(failure(M), I), \\ & \neg holds(irrelevant(M), I). \\ holds(intended\_action(M, stop(M)), I) \leftarrow\ & holds(success(M), I), \\ & \neg holds(irrelevant(M), I). \\ holds(intended\_action(M, stop(M)), I) \leftarrow\ & holds(cancelled(M), I), \\ & \neg holds(irrelevant(M), I). \end{aligned} \tag{30}$$

Intended actions propagate up from current sub-activities.

$$\begin{aligned} holds(intended\_action(M, Aa), I) \leftarrow\ & holds(intended\_action(M1, Aa), I), \\ & holds(curr\_subact(M1, M), I), \\ & holds(in\_progress(M), I). \end{aligned} \tag{31}$$

The next axioms define changes in the status of execution after execution of an intended action. The first axiom deals with the affect of executing an intended physical action.

$$\begin{aligned} holds(status(M, K+1), I+1) \leftarrow\ & occurs(PA, I), \\ & holds(intended\_action(M, PA), I), \\ & holds(status(M, K), I), \\ & component(M, K+1, PA). \end{aligned} \tag{32}$$

The second describes the effect of ending a sub-activity.

$$\begin{aligned} holds(status(M, K+1), I+1) \leftarrow\ & occurs(stop(M1), I), \\ & h(intended\_action(M1, stop(M1)), I), \\ & holds(status(M, K), I), \\ & component(M, K+1, M1). \end{aligned} \tag{33}$$

Next, we introduce the non-procrastination axiom which says that an agent tends to execute an intended action as soon as possible. This statement has a form of a typical default of Answer Set Prolog. The second axiom is an exception to this default that says, action $abandon(G)$ interrupts the execution of an intended activity by preventing the execution of the intended action.

$$\begin{array}{l} occurs(E,I) \leftarrow holds(intended\_action(M,E),I), \\ \qquad\qquad\quad not\ \neg occurs(E,I), \\ \qquad\qquad\quad I < n. \end{array} \tag{34}$$

$$\begin{array}{l} \neg occurs(E,I) \leftarrow occurs(abandon(G),I), \\ \qquad\qquad\qquad\quad holds(intended\_action(M,E),I). \end{array} \tag{35}$$

While executing an intended activity an agent should only execute physical actions that are intended.

$$\begin{array}{l} intended(PA,I) \leftarrow holds(intended\_action(M,PA),I). \\ \neg occurs(PA,I) \ \leftarrow holds(active(M),I), \\ \qquad\qquad\qquad\quad not\ intended(PA,I). \end{array} \tag{36}$$

Let us denote the resulting theory by $\mathcal{I}_0$. The theory contains Basic Axioms of $\mathcal{I}$ and is already capable of deriving some non-trivial conclusions and can serve as the basis for a deeper theory of intentions we introduce in the next section. The power of Basic Axioms can be illustrated by its use for formalizing several scenarios from Example 1.

Example 4. [Example 1 revisited]
To model the first scenario from Example 1 we need a program Π^n from Example 2, which describes Bob's world, and a history describing the initial state of Bob's world and his current mental state. History H_0 contains the initial positions of Bob and John (rooms 1 and 3 respectively). Since basic axioms assume that the agent already has an active goal and created an activity to achieve this goal we assume that the history also contains statement $holds(active(meet(b,j),0)$, the activity m from Example 3, and statement $holds(status(m,-1),true,0)$ which indicates that m is inactive. This description of the initial mental state of the agent will not be included in the histories used in conjunction with complete theory $\mathcal{I}$.

Let us start with the scenario of Example 1 in which Bob intends to execute activity m. This can be represented by the history

$$H_1 = H_0 \cup \{hpd(start(m),0)\}$$

To compute the trajectory defined by this history we will consider a program

$$B_1 = \Pi^n \cup \mathcal{I}_0 \cup H_1$$

obtained by combining general knowledge with the basic axioms of Theory of Intentions and the domain's history. The program has one answer set containing occurrences of actions

$$\begin{array}{l} \{occurs(start(m),0), occurs(move(b,1,2),1), \\ occurs(move(b,2,3),2), occurs(stop(m),3)\} \end{array}$$

and fluent literals: $\{holds(meet(b, j), 3), holds(success(m), 3)\}$.

This corresponds to the trajectory which will be performed by Bob in the absence of any intervening exogenous actions.

Persistence of intentions in the presence of some such actions can be illustrated by another scenario and corresponding program

$$H_2 = H_0 \cup \{hpd(start(m), 0), hpd(delay(b), 1)\}$$
$$B_2 = \Pi^n \cup \mathcal{I}_0 \cup H_2$$

(Recall that $delay(bob)$ prevents Bob from moving.) thas one answer set containing occurrences of actions

$$\{occurs(start(m), 0), occurs(delay(b), 1), occurs(move(b, 1, 2), 2),$$
$$occurs(move(b, 2, 3), 3), occurs(stop(m), 4)\}$$

and fluent literals: $\{holds(meet(b, j), 4), holds(success(m), 4)\}$.

As one can see Bob will continue the execution of his activity after the delay and will successfully end it at step 4. This happens because the execution of $delay(bob)$ prevents Bob from moving at step 1. However it does not change the status of the activity and therefore action $move(b, 1, 2)$ remains intended for execution, and hence actually happens, at step 2.

The scenario where Bob observes that John moved from room 3 to room 2 while Bob was moving from room 1 to 2 can be described by the following history and corresponding program

$$H_3 = H_0 \cup \{hpd(start(m), 0), hpd(move(b, 1, 2), 1), hpd(move(j, 3, 2), 1)\}$$
$$B_3 = \Pi^n \cup \mathcal{I}_0 \cup H_3$$

that has one answer set containing occurrences of actions

$$\{occurs(start(m), 0), occurs(move(b, 1, 2), 1),$$
$$occurs(move(j, 3, 2), 1), occurs(stop(m), 2)\}$$

and fluent literals: $\{holds(meet(b, j), 2), holds(success(m), 2))\}$

In this scenario Bob achieves his goal before completely executing his plan. His next intended action is to stop executing his activity, and not continue moving to room 3.

Now let us consider a small variation of the previous scenario, in which, instead of observing John's move to room 2 Bob observes John being in room 2 at step 2. The scenario can be described by the following history and its corresponding program

$$H_4 = H_0 \cup \{hpd(start(m), 0), hpd(move(b, 1, 2), 1), obs(in(j, 2), true, 2)\}$$
$$B_4 = \Pi^n \cup \mathcal{I}_0 \cup H_4$$

This time the behavior of $\mathcal{I}_0$ is less satisfactory. The answer set of program B_4 contains statement: $inconsistent\_obs(in(j, 2), true, 2)$ and contains neither

$h(success(m), 2)$ nor $h(failure(m), 2)$. This is not surprising since the observation is unexplained by the knowledge base of Bob. Bob does not know that John has left room 3 and therefore expects John to be there. This contradicts the observation. Consequently, the Reality Check Axiom (4) causes the statement above to appear in the answer set. To resolve unexpected observations the theory of intentions should have a diagnostic ability, which would allow Bob to restore consistency to his history by concluding that John moved from room 3 to room 2. This ability will be added to our theory in the next section.

3.2 Axioms for Unexpected Observations

So far our formalization was done in the original Answer Set Prolog. To represent knowledge needed for diagnostics and planning we will use an extension of Answer Set Prolog, called CR-Prolog. All the examples discussed in this paper were tested using CR-Prolog inference engine *crmodels*. A program of CR-Prolog consists of regular Answer Set Prolog rules and a (possibly empty) collection of so called *consistency restoring* rules of the form:

$$r : l_0 \stackrel{+}{\leftarrow} l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n$$

where r is the name of the cr-rule, l_i's are literals. The rule says "if $l_1, \ldots, l_m$ hold and there is not reason to believe $l_{m+1}, \ldots, l_n$, then l_0 *may* possibly hold. However, this possibility may be used only if there is no way to obtain a consistent set of beliefs by using only regular rules of the program. When an agent discovers an inconsistent observation, it must find an explanation (an unobserved past occurrence of an exogenous action) that resolves the inconsistency between observations and expectations. Mental action

$$mental\_action(find\_explanation).$$

is executed whenever the agent has an inconsistent observation.

$$occurs(find\_explanation, I) \leftarrow inconsistent\_obs(F, V, I). \tag{37}$$

Next we introduce an important consistency restoring rule which says that the agent looking for an explanation of an inconsistent observation may assume that any exogenous action could have occurred (unobserved) in the past.

$$diag(EA, I2, I1) : occurs(EA, I2) \stackrel{+}{\leftarrow} hpd(find\_explanation, I1), \quad I2 < I1. \tag{38}$$

Here variable EA ranges over exogenous actions. (As usual, a rule containing variables is viewed as a shorthand for the collection of all its ground instantiations which respect the meaning of variables).
The next two axioms encode the requirement that an explanation resolve all inconsistent observatations. An inconsistent observation of fluent F having truth

value V at I is resolved if the expected value of F at $I+1$ is V.

$$\begin{array}{l} \leftarrow inconsistent\_obs(F, true, I), \\ \quad current\_step(I+1), \\ \quad \neg h(F, I+1). \\ \leftarrow inconsistent\_obs(F, false, I), \\ \quad current\_step(I+1), \\ \quad h(F, I+1). \end{array} \tag{39}$$

For simplicity we limit possible explanations of the discrepancy between observations and prediction to a single unobserved occurrence of an exogenous action.

$$\begin{array}{rl} unobserved(EA, I) \leftarrow & occurs(EA, I), \\ & not\ hpd(EA, I). \\ \leftarrow & unobserved(EA1, I1), \\ & unobserved(EA2, I2), \\ & EA1 \neq EA2. \end{array} \tag{40}$$

Now Bob's general knowledge about his domain and about reasoning with intentions is given by a program

$$\Pi^n \cup \mathcal{I}_1$$

where $\mathcal{I}_1$ consists of $\mathcal{I}_0$ expanded by the six diagnostic rules above. The following example illustrates the use of these rules.

Example 5. [Finding Explanations]
To illustrate the behavior of an agent using our new theory let us go back to the last scenario of Example 4 with domain history H_4 recording unsuccessful execution of activity m. Recall that the answer set, say S_0, of the corresponding program contained statement: $inconsistent\_obs(in(j,2), true, 2)$ which was suppose to alert Bob to the need for an explanation. Let us now replace old theory $\mathcal{I}_0$ by $\mathcal{I}_1$ and consider a new

$$B_4 = \Pi^n \cup \mathcal{I}_1 \cup H_4$$

The answer set of the new program contains S_0 together with a statement $occurs(find\_explanation, 2)$. Bob will record the new action occurrence in his new history and use the corresponding program

$$\begin{array}{l} H_5 = H_4 \cup \{hpd(find\_explanation, 2)\} \\ B_5 = \Pi^n \cup \mathcal{I}_1 \cup H_5 \end{array}$$

to resolve the difficulty with unexplained observation. Note that now the current step of the program is 3 satisfying constraints 39 becomes a problem for regular rules of the program. This activates the 38 which finds the necessary explanations — inconsistency was caused by the failure of the agent to notice the occurrence of an exogenous action $move(j, 3, 2)$ at step 1. Consequently, the program will have one answer set containing actions

$$\{occurs(start(m), 0), occurs(move(b, 1, 2), 1), occurs(move(j, 3, 2), 1), \\ occurs(find\_explanation, 2), occurs(stop(m), 3)\}$$

and fluent literals: $\{holds(meet(b,j),2), holds(success(m),2))\}$

but not containing: $inconsistent\_obs(in(j,2),true,2)$.

3.3 Axioms for Planning

In previous examples illustrating our theory of intentions we discussed Bob's intent to execute a given activity m. But Example 1 begins with Bob intending to meet with John, i.e. with intention to *achieve a goal* followed by *planning*. Similarly in the scenario in which Bob didn't find John in room 3. To model this type of intentions we further expand our theory by removing assumptions from section 3. This time the only information available to the agent in the beginning of the process is the *physical state of the domain and the selected goal*. The latter is recorded by the statement: $hpd(select(G),0)$, where G is a goal. As an effect of the action $select(G)$ the agent will look for a plan from the current step to achieve the intended goal G. The plan and the goal will be represented as the agent's intended activity. We name activities generated by the agent by integers from the set:

$$created\_activity(0..max\_act). \tag{41}$$

where max_act is the maximum number of activities the agent can create. We also need rule

$$activity(X) \leftarrow created\_activity(X). \tag{42}$$

and some way to generate a new name for an activity to be created.

$$fluent(name(X), inertial) \leftarrow created\_activity(X). \tag{43}$$

$$holds(name(1),0). \tag{44}$$

$$holds(name(X+1), I+1) \leftarrow occurs(start(X,I), holds(name(X),I). \tag{45}$$

$$\neg holds(name(X2),I) \leftarrow created\_activity(X1), created\_activity(X2), holds(name(X1),I), X1 \neq X2. \tag{46}$$

The name, which is initially set to 1 is incremented as the result of execution of *start*. The last rule guarantees the name's uniqueness.
We also need defined fluent $in\_progress(G)$ that is true when the agent is executing an activity M with goal G.

$$fluent(in\_progress(G), defined)$$

$$holds(in\_progress(G), I) \leftarrow holds(active(M), I), goal(M,G). \tag{47}$$

An activity M is formed to achieve goal G when the goal is active but no activity with this goal is currently in progress. This is expressed by the following:

$$fluent(form\_activity(M,G), defined)$$

$$\begin{aligned} holds(form\_activity(M,G),I) \leftarrow\ & holds(active(G),I),\\ & \neg holds(in\_progress(G),I),\\ & \neg holds(minor(G),I),\\ & holds(name(M),I),\\ & current\_step(I1),\\ & I \leq I1. \end{aligned} \tag{48}$$

Note that the third literal in the body is needed since the agent only persist in his intentions to achieve non minor goals. The remaining two literals guarantee that activity M is formed in the agent's past (as opposed to being hypothesized at some future time).

The goal for the activity which needs to be formed is given by the rule:

$$goal(M,G) \leftarrow holds(form\_activity(M,G),I). \tag{49}$$

To create components of this activity we will also need a sort for possible activity components which will be given by statements:

$$comp(PA). \quad comp(M). \tag{50}$$

(Possibly indexed variables C will range over these components, i.e physical actions or activities.) The activity's components are generated by the cr-rule:

$$\begin{aligned} plan(M,K,C,G,I) : component(M,K,C) \stackrel{+}{\leftarrow}\ & holds(form\_activity(M,G),I),\\ & current\_step(I),\\ & 0 < K. \end{aligned} \tag{51}$$

which says that if necessary any possible component can be used to form the activity. There are a number of natural constraints on an activity's components:

$$\begin{aligned} &\leftarrow component(M,K,M).\\ &\leftarrow component(M,K,C),\\ &\quad component(M,K,C1),\\ &\quad C1 \neq C.\\ &\leftarrow component(M,K,C),\\ &\quad length(M,K1),\\ &\quad K > K1. \end{aligned} \tag{52}$$

Of course we want to only generate activities which are expected to succeed in achieving the goal. This constraint is expressed as follows:

$$\begin{aligned} \leftarrow\ & current\_step(I),\\ & holds(form\_activity(M,G),I),\\ & not\ projected\_success(M,I). \end{aligned} \tag{53}$$

where $projected\_success(M,I)$ is a shorthand defined by the following axiom.

$$\begin{aligned} projected\_success(M,I) \leftarrow\ & current\_step(I),\\ & holds(success(M),I1),\\ & I < I1. \end{aligned} \tag{54}$$

Finally we define the length of activity:

$$\begin{aligned} has\_comp(M,K) &\leftarrow component(M,K,C). \\ length(M,K) &\leftarrow has\_comp(M,K), \\ &\quad\ not\ has\_comp(M,K+1). \end{aligned} \tag{55}$$

The following axiom says that a newly formed activity should be executed.

$$holds(intended\_action(M,start(M)),I) \leftarrow holds(form\_activity(M,G),I). \tag{56}$$

This completes the planning component, $\mathcal{I}_2$, which consists of axioms 41-56. Now we define our final theory $\mathcal{I}$:

$$\mathcal{I} =_{def} \mathcal{I}_1 \cup \mathcal{I}_2$$

3.4 More Examples

It is easy to show that addition of new axioms does not change the behavior of the theory on previous scenarios. To see how it works for planning and diagnostics let us look at the remaining scenarios from Example 1. Note that now Bob's general knowledge about his domain and about reasoning with intentions is given by a program

$$\Pi^n \cup \mathcal{I}$$

and his initial state is

$$\mathcal{H}_0 = \{holds(in(b,1),0), holds(in(j,3),0), hpd(select(meet(b,j)),0)\}$$

Example 6. [Intending to Achieve - Initial Planning]
Initially Bob's knowledge is represented by program

$$\mathcal{B}_0 = \Pi^n \cup \mathcal{I} \cup \mathcal{H}_0$$

where n is a maximal length of the domain trajectory. The program has one answer set containing description, $\mathcal{A}_1$ of activity 1 where $\mathcal{A}_1$ is

$$\begin{aligned} &goal(1,meet(b,j)). \\ &component(1,1,move(b,1,2)). \\ &component(1,2,move(b,2,3)). \\ &length(1,2)). \end{aligned}$$

actions

$$\begin{aligned} &\{occurs(select(meet(b,j)),0), occurs(start(1),1), \\ &occurs(move(b,1,2),2), occurs(move(b,2,3),3), occurs(stop(1)),4)\} \end{aligned}$$

and fluent literals: { holds(meet(b,j),4), holds(success(1),4)}.
The current step of the program is 0. At this point Bob created the activity to achieve his goal and expects to actually achieve it at step 4. According to our

observe-think-act loop after completion of his initial planning step Bob will store a record of the activity 1 as follows

$$\mathcal{H}_1 = \mathcal{H}_0 \cup \mathcal{A}_1.$$

and proceed executing this activity. In the absence of exogenous actions Bob and John will meet at step 4 of the trajectory.

Now let us look at the scenario from Example 1 in which Bob moved to room 3, didn't find John, found an explanation, and persisted with achieving his goal.

Example 7. [Unexpected Observations]
In this scenario Bob's knowledge is given by history

$$\mathcal{H}_2 = \mathcal{H}_1 \cup \{hpd(start(1),1), hpd(move(b,1,2),2), hpd(move(b,2,3),3),\\ obs(in(j,3),false,4), hpd(find\_explanation,4), hpd(stop(1),5)\}$$

Now Bob's knowledge is represented by program

$$\mathcal{B}_1 = \Pi^n \cup \mathcal{I} \cup \mathcal{H}_2.$$

This program has 3 answer sets differing only by when John moved to room 4. The answer sets contain actual occurrences of actions from $\mathcal{H}_2$, fluent literals,

$$\{holds(failure(1),4), holds(active(meet(b,j)),5)\}$$

statements,

$$goal(2, meet(b,j)).\\ component(2,1,move(b,3,4)).\\ length(2,1).$$

describing the newly created activity 2, and new actions

$$\{occurs(start(2),6), occurs(move(b,3,4),7), occurs(stop(2),8)\}$$

which are the result of replanning. Note the use of axioms 38 and 51 for diagnosing and replanning. Thanks to this axiom Bob will be able to persist in his intention, and plan to move to room 4 and meet John there. To proceed Bob will store this second activity in his knowledge base and continue the execution of actions and the corresponding recording as before. In the absence of further interfering Bob will meet John in room 4 at step 8.

4 Discussion

The theory $\mathcal{I}$ presented in the previous section formalizes properties of intentions intentions to achieve goals and intentions to execute activities. The theory, formulated in CR-Prolog, is meant to be used in conjunction with an agent's general knowledge about its world and its own abilities, as well as knowledge

about a particular history of the agent. Its axioms capture the agent's reasoning about its beliefs, explain the relationship between these beliefs and agent's intentions, and show how these beliefs and intentions direct the agent's diagnostic and planning activities. We also demonstrated that our axioms can be naturally incorporated into the well developed AAA agent architecture. Several examples illustrate that this, together with the reasoning abilities of solvers like *crmodels* make it possible to automate non-trivial commonsense behavior of agents.

As expected there still remain a large number of open questions. First, our axioms are not general enough to cover some interesting situations. For instance, the agent using these axioms may fail to recognize the futility of its intended activity even if he has enough information to do so. If in the beginning of execution of his plan to meet John Bob from Example 1 is informed that John has moved to the room inaccessible to Bob he would continue the execution of his intended activity. The failure of this activity to achieve its goal which should be apparent to Bob immediately will only be recognized after arriving in room 3. To deal with the problem one may attempt to suggest replanning every time a new observation is made about relevant objects of the domain but this would be a subject of future work. We also have to go beyond our simplifying assumptions and consider behavior of an agent trying to achieve several goals simultaneously, introduce priorities between goals, incorporate our theory into a more hierarchical control loop which will allow more intelligent selection of goals and better connection between agent's experience at achieving the goal and its abandonment, etc. The second direction of future work will be related to improving efficiency of our agent's reasoning with intentions. Here the most immediate need is for more efficient version of *crmodels* algorithm and implementation. Finally we plan to expand our framework by allowing reasonable use of probabilistic information. Our language of choice for representing such knowledge will be P-log [8][3].

There is much prior work on intentions related to AI which had an impact on this paper. A substantial part of this work is based on the ideas presented in [9], where the authors modeled intentions in modal logic. This led to the development of BDI architecture and powerful logical systems allowing various forms of reasoning about intentions, beliefs, and commitments (for more details see for instance [28]). For a long time there has been a substantial gap between theoretical results in this area and actual use of these logical systems for agent design. In our opinion this was partly due to the power and complexity of the language allowing nested modal operators, to the lack of well-understood connections between these modal theories and theories about actions which often serve as a basis for the agent design, and to monotonicity of the corresponding logics which limits the ability of the agent to absorb new knowledge. Much of subsequent work, which probably started with [25], attempted to narrow this gap. There are several papers which are close to our approach. The closest of course is [7] which builds the theory of intentions on the foundations of action

[3] We actually plan to use version from [15] which replaces the logical bases for the language from ASP to CR-Prolog.

languages and ASP. This connects intentions with theories of actions and ASP based theory of rational belief and opens the way of using ASP inferences engines for reasoning about intentions. As discussed in the introduction we expand this earlier work and establish a closer connection of reasoning with intentions and AAA agent architecture. In [24] reasoning about intentions is placed in the framework of another action formalism – situation calculus [21, 27]. The relation between our work and that of [24] requires more investigation. Clearly there are substantial technical differences. We also believe that our work provides more powerful reasoning mechanisms which allow to deal with diagnosis, reason about indirect effects of actions, etc., as well as higher degree of elaboration tolerance. On another hand work based on situation calculus may allow more advanced interplay between beliefs of an agent and actual truth of fluents in various situations. Finally there is a number of papers which design and investigate agents architectures which include reasoning about plans, and revising them at run-time, following observations. Such plans can be seen as intentions. Such an approach is taken, for instance, by [17] which is somewhat similar to our work in their use of abductive logic programming. The action theories and logic programming reasoning mechanisms are, however, very different from those in our paper. Moreover, this work does not contain a fully developed theory of intentions, and is more concerned with agent's architecture.

5 Afterword

Finally we would like to congratulate Marek Sergot on his 60th birthday. We learned a lot from his work, for instance, one of the basic ideas of this paper — combining reasoning about actions and logic programming — has been learned by the authors from much earlier and very influential paper on Event Calculus [18]. We hope to learn much more in the future.

5.1 Acknowledgements

We would like to acknowledge NASA grant #NNX1OAI86G and thank Jarred Blount, Daniela Inclezan, and Vladimir Lifschitz for their valuable comments.

References

1. M. Balduccini and M. Gelfond. The aaa architecture: An overview. In *AAAI Spring Symposium on Architecture of Intelligent Theory-Based Agents*, 2008.
2. Marcello Balduccini. CR-MODELS: An inference engine for CR-Prolog. In C. Baral, G. Brewka, and J. Schlipf, editors, *Proceedings of the 9th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'07)*, volume 3662 of *Lecture Notes in Artificial Intelligence*, pages 18–30. Springer, 2007.
3. Marcello Balduccini and Michael Gelfond. Diagnostic reasoning with A-Prolog. *Journal of Theory and Practice of Logic Programming (TPLP)*, 3(4–5):425–461, Jul 2003.

4. Marcello Balduccini and Michael Gelfond. Logic Programs with Consistency-Restoring Rules. In Patrick Doherty, John McCarthy, and Mary-Anne Williams, editors, *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, pages 9–18, Mar 2003.
5. Chitta Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press, Jan 2003.
6. Chitta Baral and Michael Gelfond. Reasoning Agents In Dynamic Domains. In *Workshop on Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, Jun 2000.
7. Chitta Baral and Michael Gelfond. Reasoning about Intended Actions. In *Proceedings of AAAI05*, pages 689–694, 2005.
8. Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic reasoning with answer sets. *Journal of Theory and Practice of Logic Programming (TPLP)*, 9(1):57–144, 2009.
9. Cohen and Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
10. A. Gabaldon. Activity recognition with intended actions. In *IJCAI*, pages 1696–1701, 2009.
11. M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam answer set solving collection. *aicom*, 24(2):105–124, 2011.
12. M. Gelfond and D. Inclezan. Yet Another Modular Action Language. In *Proceedings of SEA-09*, pages 64–78. University of Bath Opus: Online Publications Store, 2009.
13. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
14. M. Gelfond and V. Lifschitz. Action Languages. *Electronic Transactions on AI*, 3, 1998.
15. Michael Gelfond and Nelson Rushton. Causal and probabilistic reasoning in p-log. In R. Dechter, H. Gener, and J. Halpern, editors, *Heuristics, Probabilities and Causality. A tribute to Judea Pearl*, pages 337–359. College Publications, 2010.
16. Daniela Inclezan. Computing Trajectories of Dynamic Systems Using ASP and Flora-2. Paper presented at NonMon@30: Thirty Years of Nonmonotonic Reasoning Conference, Lexington, Kentucky, 22-25 October, 2010.
17. Antonis Kakas, Paolo Mancarella, Fariba Sadri, Kostas Stathis, and Francesca Toni. Computational logic foundations of kgp agents. *Journal of Artificial Intelligence Research*, pages 285–348, 2008.
18. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, pages 4:67–95, 1986.
19. N. Leone, G. Pfeifer, W. Faber, F. Calimeri, T. Dell'Armi, T. Eiter, G. Gottlob, G. Ianni, G. Ielpa, C. Koch, S. Perri, and A. Polleres. The DLV system. In *Proceedings of the 8th European Conference on Artificial Intelligence (JELIA 02)*, pages 537–540, 2002.
20. Victor W. Marek and Miroslaw Truszczynski. *Stable models and an alternative logic programming paradigm*, pages 375–398. The Logic Programming Paradigm: a 25-Year Perspective. Springer Verlag, Berlin, 1999.
21. J. McCarthy and P. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

22. I. Niemela and P. Simons. Extending the Smodels System with Cardinality and Weight Constraints. In *Logics in Artificial Intelligence*. Kluwer Academic Publishers, 2000.
23. Ilkka Niemela. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. In *Proceedings of the Workshop on Computational Aspects of Nonmonotonic Reasoning*, pages 72–79, Jun 1998.
24. Pilar Pozos Parra, Abhaya Nayak, and Robert Demolombe. Theories of intentions in the framework of situation calculus. In *Declarative Agent Languages and Technologies (DALT 2004), LNCS 3476, Springer-Verlag (2005)*. Springer Verlag, 2005.
25. Anand S Rao. *AgentSpeak(L): BDI agents speak out in a logical computable language*, volume 1038, pages 42–55. Springer, 1996.
26. Raymond Reiter. *On Closed World Data Bases*, pages 119–140. Logic and Data Bases. Plenum Press, 1978.
27. Raymond Reiter. *Knowledge in Action – Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Sep 2001.
28. Michael Wooldridge. *Reasoning About Rational Agents*. The MIT press, 2000.