Towards a Theory of Intentional Agents

Justin Blount[*] **and Michael Gelfond**
Department of Computer Science
Texas Tech University
Lubbock, TX 79409 USA

Marcello Balduccini
College of Computing and Informatics
Drexel University
Philidelphia, PA 19104 USA

Abstract

This paper describes the $\mathcal{AIA}$ architecture for agents whose behavior is driven by their intentions and who reason about, and act in, changing environments. The description of the domain, written in action language $\mathcal{AL}$, includes both a description of agent's environment and the *Theory of Intentions* which describes properties of intentions. The architecture is designed to make agents capable of explaining unexpected observations (by means of diagnostic reasoning) and determining which of his actions are *intended* at the present moment. Reasoning tasks are reduced to computing answer sets of CR-Prolog programs constructed automatically from the agent's knowledge.

Introduction

This paper presents a new declarative methodology for design and implementation of intelligent agents. We limit our attention to a single agent satisfying the following assumptions:

- the agent's environment, its mental state, and the effects of occurrences of actions can be represented by a transition diagram. States of the diagram contain physical properties of the world as well as mental attitudes of the agent. Transitions are caused by actions and reflect possible changes in these states;
- the agent is capable of making correct observations, remembering the domain history, and correctly recording the *results* of his *attempts* to perform actions[1];
- normally, the agent is capable of observing the occurrence of all relevant exogenous[2] actions.

The assumptions are satisfied by a number of interesting domains but in many cases they are, of course, too restrictive. In our future work we plan to expand our architecture to systems in which agents are capable of communicating with each other, and to agents who can only determine a likelihood of success and failure of some of their actions and observations. These problems are, however, orthogonal to the main issues considered here. The topic of this paper is to a certain degree related to that of execution monitoring (desJardins et al. 1999). A precise evaluation of the relationship will be the subject of future research.

Our approach to agent design (referred to as $\mathcal{AIA}$) borrows a substantial number of ideas from an earlier work on the AAA architecture (Balduccini and Gelfond 2008). In this work declarative action language $\mathcal{AL}$ (Baral and Gelfond 2000) was used to concisely define a transition diagram of the agent's domain together with the domain's recorded history. Slightly generalized causal laws and history records of $\mathcal{AL}$ are used for the same purpose in our approach. (The use of the former can be traced to (Baral and Gelfond 2000). The use of the latter is new.) The main difference between AAA and $\mathcal{AIA}$ is in the organization of the basic control loop of the agent. In both cases the agent uses its knowledge about the domain to perform diagnostic and planning tasks. However, in our approach the loop is centered around the notion of intention which is absent in AAA. The use of intentions simplifies and generalizes the loop and allows the agent to more naturally persist in its commitment to achieve its goals. Moreover, it allows an outside observer (including the agent designer) to reason about agent's behavior, e.g. to prove that the agent will never perform any unintended actions. The $\mathcal{AIA}$'s focus on intentions and their primary role in intelligent behavior makes the architecture a special case of the well known BDI methodology for the design of intelligent agents (Rao and Georgeff 1991) and (Wooldridge 2000). These BDI systems are grounded in declarative logics for reasoning about an agent's beliefs, desires, and intentions. These logics are not executable and not clearly connected to executable systems. In our approach we remedy this problem. In formalizing the notion of intention we borrow some basic intuition about such properties of intentions as *persistence* and *non-procrastination* from (Baral and Gelfond 2005), where the authors formalized the behavior of an agent intending to execute a sequence of actions in ASP. While the theory from (Baral and Gelfond 2005) has been used for question answering from natural language (Inclezan 2010), for activity recognition (Gabaldon 2009), and

[*]Present address: Southwest Research Institute, San Antonio TX 78238 USA

[1]We say *attempt* because though the agent hopes his action is executable, it may in fact not be, and in this case we assume that the domain is not changed. The *result* of an attempt to perform an action is either the occurrence or non-occurrence of the action.

[2]Actions occurring in the environment that are not performed by the agent are called *exogenous*.

for other intelligent tasks, it is not sufficient for the goal-oriented reasoning of *intentional* agents. The technical features of our theory of intentions are quite different and substantially more advanced.

The technical realization of $\mathcal{AIA}$ is based on two main technical contributions:

- Introduction of formal theory of intentions ($\mathcal{TI}$).
- Development of an algorithm which takes the agent's knowledge (including the theory of intentions), explains the unexpected observations and computes an action the agent will intend to perform. (Note, that when necessary, the second task can require planning).

The $\mathcal{TI}$ represents properties of intentions as a collection of statements of an action theory of $\mathcal{AL}$. This ensures declarativity and allows for seamless integration of $\mathcal{TI}$ with agent's knowledge about the world and its own actions and history. Existence of a reasoning algorithm ensures that the declarative specification of an agent can be actually executed. The algorithm is based on the reduction of the task of explaining observations and finding an intended action to the problem of computing answer sets of a program of CR-Prolog (Balduccini and Gelfond 2003) automatically generated from the agent's knowledge. As usual answer sets are computed by a general purpose algorithm implemented by a comparatively efficient answer set solver (Balduccini 2007). A prototype implementation of a software called $\mathcal{AIA}$ *Agent Manager* allows to test this methodology. The following example informally describes the agent and illustrates its intended behavior by a number of simple (but non-trivial) scenarios. $\mathcal{AIA}$ *Agent Manager* is used to build the agent that is proven to correctly automate this behavior.

Example 1 *[Bob and John]*
Consider an environment that contains our agent Bob, his colleague John, and a row of four rooms, $r1$, $r2$, $r3$, $r4$ where consecutive rooms are connected by doorways, such that either agent may $move$ along the row from one room to the next. The door between rooms $r3$ and $r4$ is special and can be $locked$ and $unlocked$ by both Bob and John. If the door is $locked$ then neither can $move$ between those two rooms until it is $unlocked$. Bob and John *meet* if they are located in the same room.

Scenario 1: Planning to achieve the goal
Initially Bob knows that he is in $r1$, his colleague John is in $r3$, and the door between $r3$ and $r4$ is unlocked. Suppose that Bob's boss requests that he meet with John. This causes Bob to intend to meet with John. This type of intention is referred to as an *intention to achieve* a goal. Since Bob acts on his intentions, he uses his knowledge of the domain to choose a plan to achieve his goal. Of course Bob does not waste time and chooses the shortest plan that he expects to achieve his goal, that is to move from $r1$ to $r2$ and then to $r3$. A pair consisting of a goal and the plan aimed at achieving it is called an *activity*. To fulfill his intention of meeting John, Bob *intends to execute* the activity consisting of the goal to meet John and the two step plan to move from $r1$ to $r3$. The process of executing an activity begins with a *mental*[3] action to *start* the activity. Assuming there are no interruptions, the process continues with the execution of each action in the plan (in this case, moving to $r2$, then to $r3$). After meeting John in $r3$ the process concludes with an action to $stop$ the activity.

Scenario 2: Unexpected achievement of goal
Now suppose that as Bob is moving from $r1$ to $r2$, he observes John moving from $r3$ to $r2$. In this case it will be rational for Bob to recognize the unexpected achievement of his goal, $stop$ executing his activity, and not continue moving to $r3$.

Scenario 3: Not expected to achieve goal and replanning
Now suppose that as Bob is moving from $r1$ to $r2$ he observes John moving from $r3$ to $r4$. Bob should recognize that in light of this new observation the continued execution of his activity is not expected to achieve the goal, i.e. his activity is *futile*. As a result, he should $stop$ executing his activity and $start$ executing a new one (containing a plan to move to $r3$ and then to $r4$) that is expected to achieve the goal of meeting John.

Scenario 4: Abandon goal
During the execution of his activity, Bob's boss may withdraw the request for Bob to meet with John. In this case Bob no longer intends to achieve the goal of meeting John. He should $stop$ executing the activity with the goal to do so and not continue moving toward John.

Scenario 5: Failure to achieve goal, diagnosis, and replanning
Suppose now that Bob moved from $r1$ to $r2$ and then to $r3$, but observes that John is not there. Bob must recognize that his activity failed to achieve the goal. Further analysis should allow Bob to conclude that, while he was executing his activity, John must have moved to $r4$. Bob doesn't know exactly when John moved and there are three possibilities. John could have moved while Bob was 1) $starting$ his activity, 2) moving to $r2$, or 3) moving to $r3$. In any case since Bob is committed to achieving his goal of meeting John his intention to do so persists. Bob will come up with a new activity (containing a plan to move to $r4$) to achieve the goal of meeting John.

Scenario 6: Unexpected failure to execute, diagnosis, and replanning
We continue from scenario 5. Bob $starts$ executing his activity containing a plan to move to $r4$. Then, believing that the door is unlocked, Bob attempts to move from $r3$ to $r4$, but is unable to perform the action. This is unexpected, but Bob realizes that John must have locked the door after moving to $r4$. Bob's new activity contains the same goal to meet John and a plan to unlock the door before moving to $r4$.

Scenario 7: Revision of explanations and replanning
Suppose that Bob is in $r1$ and has just $started$ his activity to move to $r3$. Bob is then told that John is no longer in $r3$. Bob reasons that there are two possible explanations. John must have moved to either $r2$ or $r4$ while Bob was $starting$ his activity. In the first case, Bob's activity is still expected to achieve his goal after he moves to $r2$, but in the second

[3]Actions that directly affect an agent's mental state are referred to as *mental* actions, while those actions that directly affect the state of the environment are referred to as *physical* actions.

case his activity is not. Bob is optimistic and since there is a possibility that his activity will achieve the goal, he continues executing it by moving to $r2$. Then he observes that John is not in $r2$ and realizes that the explanation that John moved there is invalid. Bob's only remaining explanation is that John moved to $r4$. With this he reasons that his activity is not expected to achieve the goal, so he *stops* it. Bob's intention to meet John persists so he *starts* a new activity containing a plan to move to $r3$ and then to $r4$.

Despite the comparative simplicity of the tasks illustrated by these scenarios we are unaware of any systematic declarative methodology which will allow us to easily build an agent capable of the type of reasoning needed to perform them.

Background

We start with briefly describing syntax and informal semantics of Action Language $\mathcal{AL}$. For the formal semantics see (Gelfond and Kahl 2013). $\mathcal{AL}$ is parametrized by a sorted signature containing three special sorts *actions*, *fluents* and *statics*, (properties which can (can not) be changed by actions). The fluents are partitioned into two sorts: *inertial* and *defined*. Together, statics and fluents are called *domain properties*. A *domain literal* is a domain property p or its negation $\neg p$. If domain literal l is formed by a fluent, we refer to it as a *fluent literal*; otherwise it is a *static literal*.

Definition 1 (Statements of AL) *Language $\mathcal{AL}$ allows the following types of statements:*

1. Causal Laws*:*

$$a \textbf{ causes } l_{in} \textbf{ if } p_0, \dots, p_m$$

2. State Constraints*:*

$$l \textbf{ if } p_0, \dots, p_m$$

3. Executability Conditions*:*

$$\textbf{impossible } a_0, \dots, a_k \textbf{ if } p_0, \dots, p_m$$

where $k \geq 0$, $a, a_0, \dots, a_k$ are actions, l is an arbitrary domain literal, called the head *of constraint, l_{in} is a literal formed by an inertial fluent, and $p_0, \dots, p_m$ is a possibly empty set of domain literals*[4]*. Moreover, no negation of a defined fluent can occur in the heads of state constraints.*

The collection of state constraints whose head is a defined fluent f is referred to as the *definition of f*. As in logic programming definitions, f is true if it follows from the truth of the body of at least one of its defining rules. Otherwise, f is false.

Definition 2 *[System Description]*
A system description of $\mathcal{AL}$ is a collection of statements of $\mathcal{AL}$ over some (implicitly defined) signature.

In this paper we expand the syntax of $\mathcal{AL}$ by requiring its signature to contain activities – triples consisting of a goal, a plan aimed at achieving this goal, and a name. We name activities by natural numbers. For example, Bob's activity for meeting with John, denoted by 1, is given by the following:

$$\langle 1, [move(b, r1, r2), move(b, r2, r3)], meet(b, j)\rangle \quad (1)$$

[4]If the set is empty, then keyword **if** is omitted.

from Scenario 1 consists of a plan to move to $r2$ and then to $r3$ in order to meet John; 1 is the name of the activity. For simplicity of representation in $\mathcal{AL}$ this activity will be represented by the following statics:

$$\begin{array}{l} activity(1). \\ comp(1, 1, move(b, r1, r2)). \\ comp(1, 2, move(b, r2, r3)). \\ length(1, 2). \\ goal(1, meet(b, j)). \end{array} \quad (2)$$

where $comp(X, Y, A)$ states that A is the Y^{th} element of the plan of activity X, and $length(X, N)$ says that the plan of activity X has length N. In this example both components of the activity's plan are actions. In general, they can be other, previously defined, activities.

Normally, a system description of $\mathcal{AL}$ is used together with a recorded history of the domain — a collection of agent's observations. In traditional action theories such histories determine past trajectories of the system the agent believes to be possible. Such trajectories are called *models* of a history. If no such model exists the history is deemed inconsistent. For instance, given an action theory with inertial fluents f, g and p, agent actions a_1 and a_2, an exogenous action b and causal laws

a_1 **causes** f
a_2 **causes** g **if** f, p
b **causes** $\neg p$

a history

$$\Gamma = \begin{cases} obs(p, true, 0),\ obs(f, false, 0)\ hpd(a_1, 0), \\ hpd(a_2, 1),\ obs(g, false, 2) \end{cases}$$

is inconsistent. (Here $obs(F, V, I)$ says that the observed value of fluent F at the I^{th} step of the trajectory is V; $hpd(A, I)$ indicates that action A happened at step I.)

In this work we *expand the notion of domain history and to modify the notion of history's model*. The new domain history includes two more types of statements: $attempt(A, I)$ and $\neg hpd(A, I)$. The former indicates that the agent attempted to execute action A at step I. If at that point the preconditions for executability of A are satisfied then action A is successful and, therefore, the domain history will contain $hpd(A, I)$; otherwise it will contain $\neg hpd(A, I)$ stating that A did not happen. The notion of models is modified to allow the agent to explain unexpected observations by assuming the occurrence of a minimal collection of occurrences of exogenous actions missed by the agent. According to the new definition, a history Γ' that extends Γ from above:

$$\Gamma' = \Gamma \cup \{attempt(a_1, 0),\ attempt(a_2, 1)\}$$

is not inconsistent. For example, it has a model $\langle\{\neg f, \neg g, p\}, \{a_1, b\}, \{f, \neg g, \neg p\}, \{a_2\}, \{f, \neg g, \neg p\}\rangle$.

Theory of Intentions

Now we are ready to start the description of the vocabulary of our theory of intentions. Intuitively, Bob's mental state is primarily described by two inertial fluents: $active\_goal(g)$ that holds when the agent intends to achieve goal g, and

$status(m, k)$ that describes the agent's intent to execute activity m and the current state of the process of execution. More precisely $status(m, k)$ where L is the length of m and $0 \leq k \leq L$ holds when k is the index of the component of m that has most recently been executed, and $status(m, -1)$ holds when the agent does not intend to execute m[5]. The inertial property of these two fluents elegantly captures the natural persistence of the agent's intentions.

The two mental actions $start(m)$ and $stop(m)$ directly affect the mental state of the agent by initiating and terminating the agent's intent to execute activity m. A special exogenous mental action $select(g)$, which can be thought of as being performed by the agent's controller, causes the agent's intent to achieve goal g. Similarly special exogenous action $abandon(g)$ causes the agent to abandon his intent to achieve g. The agent has a special action $wait$, which no executability conditions or effects (physical or mental) and can be seen as doing nothing. Since action $wait$ has no affects, it is neither a mental or physical action. All other agent and exogenous actions are said to be *physical*. While the agent's and exogenous mental actions do not affect the state of the physical environment, some physical actions may affect the agent's mental state. The properties of the above actions and fluents are expressed by the following axioms of $\mathcal{AL}$:

$$\begin{array}{lll} \neg status(M, K1) & \textbf{if} & status(M, K2), \\ & & K1 \neq K2. \end{array} \tag{3}$$

Defined fluent $active(M)$ is true when M has a status that is not equal to -1.

$$active(M) \quad \textbf{if} \quad \neg status(M, -1). \tag{4}$$

Action $start$ sets the value of $status$ to 0, and action $stop$ returns the activity to a status of -1.

$$\begin{array}{lll} start(M) & \textbf{causes} & status(M, 0). \\ stop(M) & \textbf{causes} & status(M, -1). \end{array} \tag{5}$$

There are natural executability conditions for these actions. An agent can neither $start$ an active activity, nor $stop$ an inactive one.

$$\begin{array}{llll} \textbf{impossible} & start(M) & \textbf{if} & active(M). \\ \textbf{impossible} & stop(M) & \textbf{if} & \neg active(M). \end{array} \tag{6}$$

To simplify our theory we assume that an agent cannot execute a physical and mental action or multiple mental actions simultaneously.

$$\begin{array}{llll} \textbf{impossible} & A_1, A_2 & \textbf{if} & physical\_agent\_act(A_1), \\ & & & mental\_agent\_act(A_2). \\ \textbf{impossible} & A_1, A_2 & \textbf{if} & mental\_agent\_act(A_1), \\ & & & mental\_agent\_act(A_2), \\ & & & A_1 \neq A_2. \end{array} \tag{7}$$

An agent cannot execute a physical agent action and $wait$ simultaneously. Similarly for a mental agent action.

$$\begin{array}{llll} \textbf{impossible} & A, wait & \textbf{if} & physical\_agent\_act(A). \\ \textbf{impossible} & A, wait & \textbf{if} & mental\_agent\_act(A). \end{array} \tag{8}$$

[5]The agent's mental state also includes the static fluents which describe his activities.

Defined fluent $child(M1, M)$ is true when $M1$ is the current component of M and defined fluent $child\_goal(G1, G)$ is true when G and $G1$ are the goals of M and $M1$.

$$\begin{array}{rll} child(M1, M) & \textbf{if} & comp(M, K+1, M1), \\ & & status(M, K). \\ child\_goal(G1, G) & \textbf{if} & child(M1, M), \\ & & goal(M, G), \\ & & goal(M1, G1). \end{array} \tag{9}$$

Defined fluent $descendant(M1, M)$ is defined recursively in terms of defined fluent $child$.

$$\begin{array}{rll} descendant(M1, M) & \textbf{if} & child(M1, M). \\ descendant(M2, M) & \textbf{if} & descendant(M1, M), \\ & & descendant(M2, M1). \end{array} \tag{10}$$

Sub-activities and subgoals are represented by defined fluent $minor(M)$ and $minor(G)$ and are defined by the axioms:

$$\begin{array}{rll} minor(M1) & \textbf{if} & child(M1, M). \\ minor(G1) & \textbf{if} & child\_goal(G1, G). \end{array} \tag{11}$$

We refer to those activities and goals that are not $minor$ as *top-level*. Special exogenous actions $select$ and $abandon$ activate and deactivate a goal respectively.

$$\begin{array}{rll} select(G) & \textbf{causes} & active\_goal(G). \\ abandon(G) & \textbf{causes} & \neg active\_goal(G). \end{array} \tag{12}$$

There are natural executability conditions for $select$ and $abandon$. A goal that is active cannot be selected and an inactive or minor goal cannot be abandoned.

$$\begin{array}{rll} \textbf{impossible}\ select(G) & \textbf{if} & active\_goal(G). \\ \textbf{impossible}\ abandon(G) & \textbf{if} & \neg active\_goal(G). \\ \textbf{impossible}\ abandon(G) & \textbf{if} & minor(G). \end{array} \tag{13}$$

We assume that no physical exogenous action may occur concurrently with a special exogenous action. Similarly for physical and mental agent actions.

$$\begin{array}{rll} \textbf{impossible}\ A_1, A_2 & \textbf{if} & special\_exog\_act(A_1), \\ & & physical\_exog\_act(A_2). \\ \textbf{impossible}\ A_1, A_2 & \textbf{if} & special\_exog\_act(A_1), \\ & & physical\_agent\_act(A_2). \\ \textbf{impossible}\ A_1, A_2 & \textbf{if} & special\_exog\_act(A_1), \\ & & mental\_agent\_act(A_2). \end{array} \tag{14}$$

Top-level goals that are achieved are no longer active.

$$\begin{array}{rll} \neg active\_goal(G) & \textbf{if} & \neg minor(G), \\ & & goal(G). \end{array} \tag{15}$$

The following four axioms describe the propagation of the intent to achieve a goal to its child goal (i.e. goals that are minor). Of course, the parent goal may be a top-level goal or it may also be minor. Note too that the four rules are disjoint, that is for a particular minor goal $G1$ at most one of these axioms will be applicable.

An unachieved minor goal $G1$ of an activity $M1$ becomes $active$ when $M1$ is the next component of an ongoing activity M.

$$\begin{array}{rll} active\_goal(G1) & \textbf{if} & minor(G1), \\ & & child\_goal(G1, G), \\ & & active\_goal(G), \\ & & goal(M1, G1), \\ & & \neg G1, \\ & & status(M1, -1). \end{array} \tag{16}$$

A minor goal $G1$ is no longer active when it is achieved.

$$\neg active\_goal(G1) \text{ if } \begin{array}{l} minor(G1), \\ child\_goal(G1,G), \\ active\_goal(G), \\ G1. \end{array} \quad (17)$$

A minor goal $G1$ is no longer active when its parent is no longer active.

$$\neg active\_goal(G1) \text{ if } \begin{array}{l} minor(G1), \\ child\_goal(G1,G), \\ \neg active\_goal(G). \end{array} \quad (18)$$

A minor goal $G1$ of $M1$ is no longer active when $M1$ has been executed.

$$\neg active\_goal(G1) \text{ if } \begin{array}{l} minor(G1), \\ child\_goal(G1,G), \\ active\_goal(G), \\ \neg G1, \\ goal(M1,G1), \\ status(M1,K1), \\ length(M1,K1). \end{array} \quad (19)$$

Defined fluent $in\_progress(M)$ is true when M is an active and its goal G is active, and defined fluent $in\_progress(G)$ is true when G is the active goal of an active activity M.

$$\begin{array}{lll} in\_progress(M) & \text{if} & active(M), \\ & & goal(M,G), \\ & & active\_goal(G). \\ in\_progress(G) & \text{if} & active(M), \\ & & goal(M,G), \\ & & active\_goal(G). \end{array} \quad (20)$$

Defined fluent $next\_act(M,A)$ is true if agent action A is the next action of the ongoing execution of M. Since this fluent describes the ongoing execution, the initial starting and stopping of a top-level activity are never $next\_act$. However the starting or stopping of a sub-activity can be the $next\_act$ of the parent activity. Axiom (21) describes when this action is a physical agent action of M.

$$next\_act(M,A) \text{ if } \begin{array}{l} physical\_agent\_act(A), \\ status(M,K), \\ comp(M,K+1,A), \\ in\_progress(M). \end{array} \quad (21)$$

When the first not yet executed component of M is a sub-activity $M1$, then the next action of M is to start $M1$.

$$next\_act(M,start(M1)) \text{ if } \begin{array}{l} status(M,K), \\ comp(M,K+1,M1), \\ in\_progress(M), \\ \neg active(M1). \end{array} \quad (22)$$

The next action of an active sub-activity $M1$ propagates up to its parent M.

$$next\_act(M,A) \text{ if } \begin{array}{l} agent\_act(A), \\ status(M,K), \\ comp(M,K+1,M1), \\ in\_progress(M), \\ in\_progress(M1), \\ next\_act(M1,A). \end{array} \quad (23)$$

The next action of M after the completion of sub-activity $M1$ is to stop $M1$.

$$next\_act(M,stop(M1)) \text{ if } \begin{array}{l} status(M,K), \\ comp(M,K+1,M1), \\ in\_progress(M), \\ active(M1), \\ goal(M1,G1), \\ \neg active\_goal(G1). \end{array} \quad (24)$$

Executing the next physical action (rule 21) that is the current component of activity M increments the status of M.

$$\begin{array}{lll} A & \textbf{causes} & status(M,K+1) \\ & \textbf{if} & next\_act(M,A), \\ & & status(M,K), \\ & & comp(M,K+1,A), \\ & & physical\_agent\_act(A). \end{array} \quad (25)$$

Executing the next action of stopping a sub-activity $M1$ (rule 26) increments the status of parent M.

$$\begin{array}{lll} stop(M1) & \textbf{causes} & status(M,K+1) \\ & \textbf{if} & status(M,K), \\ & & comp(M,K+1,M1), \\ & & next\_act(M,stop(M1)). \end{array} \quad (26)$$

Stopping an activity returns its descendants to an inactive status.

$$\begin{array}{lll} stop(M) & \textbf{causes} & status(M1,-1) \\ & \textbf{if} & descendant(M1,M). \end{array} \quad (27)$$

Finally we introduce inertial fluent $next\_name(M)$. This fluent will allow the translation of a system description into ASP to contain only those activities that are relevant (i.e. those with names and not all of the possible flat activities contained in $\mathcal{D}$). Intuitively, $next\_name$ is the first name that is not yet relevant. As activities are deemed to be relevant and started, the value of $next\_name$ increments.

$$\neg next\_name(M) \text{ if } \begin{array}{l} next\_name(M1), \\ M \neq M1. \end{array} \quad (28)$$

$$\begin{array}{lll} start(M) & \textbf{causes} & next\_name(M+1) \\ & \textbf{if} & next\_name(M), \\ & & \neg minor(M). \end{array} \quad (29)$$

This completes the theory of intentions $\mathcal{TI}$.

An *intentional system description* $\mathcal{D}$ consists of a description of the agent's physical environment, a collection of activities, and the theory of intentions. Paths in the transition diagram $\mathcal{T}(\mathcal{D})$ correspond to physically possible *trajectories* of the domain. A state of the trajectory is divided into two parts: *physical* and *mental* consisting of all physical and mental fluent literals respectively.

The $\mathcal{AIA}$ Control Strategy

According to $\mathcal{AIA}$ *control strategy* the agent begins by *observing* his domain. Observations may not be consistent with the agent's expectations. In this case the agent preforms diagnostic reasoning to *explain* unexpected observations. An observation of an occurrence of special exogenous action

$select(g)$ (12), which initiates the agent's intent to achieve goal g, is particularly important to the agent. Our agent is motivated to change his domain[6] by the intention to achieve a goal. Because of his intention to achieve a goal, the agent finds an activity which may lead to achieving the goal, commits to this activity, and proceeds with its execution. At the center of this process is a task of deciding the intended action that the agent should attempt to perform at the current step in order to fulfill his intention to achieve the goal. *Intuitively, an agent's behavior is compatible* (i.e. is in accordance) *with his intentions when at each step he attempts to perform only those actions that are intended and does so without delay*. To precisely define such behavior and to formulate and prove correctness of the architecture we will introduce notions of *intended model* and *intentional history*.

In our architecture this behavior is specified by the following $\mathcal{AIA}$ *control loop*.

Observe the world and initialize history with observations;

1. interpret observations;
2. find an intended action e;
3. attempt to perform e and
   update history with a record of the attempt;
4. observe the world,
   update history with observations, and go to step **1**.

Figure 1: $\mathcal{AIA}$ control loop

The agent begins by *observing* his domain and initializing his history with the observations. In general the observations need not be complete. Let us assume however for simplicity that the agent's initial observations are complete. After initializing the history, the agent enters a loop consisting of four steps. In step **1** the agent uses diagnostic reasoning to explain unexpected observations. The agent explains these observations by hypothesizing that some exogenous actions occurred unobserved in the past. For simplicity we assume that the agent is looking for explanations with the smallest number of such actions. (It will not be difficult to consider actions with costs and to look for explanations with the smallest costs.) In step **2** the agent finds an *intended action*. An *intended action* is intuitively either to continue executing an ongoing activity that is expected to achieve its goal; to *stop* an ongoing activity whose goal is no longer active (either achieved or abandoned); to *stop* an activity that is no longer expected to achieve its goal; or to *start* a chosen activity that is expected to achieve his goal. Of course, there may be no way for the agent to achieve his goal or he may have no goal. In either case the agent's *intended action* is to $wait$. Step **3** corresponds to an output operation where the agent attempts to perform an intended action and updates his history with a record of his attempt. Step **4** corresponds to an input operation where the agent observes his domain. This includes not only values of fluents, but also occurrences or non-occurrences of exogenous actions and the result of his attempt to perform his intended action. This step concludes when the agent updates his history with the observations and returns to step **1**.

[6]Without the intention to achieve a goal he is not motivated to change his domain and does nothing (i.e. he $waits$)

The control loop relies on some natural assumptions about observation strategies of agents and the behavior of the agent's controller. In particular, we assume that at each step the agent observes the truth or falsity of its goal as well as the commands ($select$ and $abandon$) issued to the agent by its controller. We assume that multiple goals are not simultaneously $selected$ and that a goal is $selected$ only when the agent has neither an intended goal or activity. We also assume that every time the agent finds an explanation of unexpected observations it stores in the domain history the size of this explanation – the number of missed exogenous actions. All this information will be crucial for determining an action the agent will intend to perform.

In general, a history Γ of the domain defines trajectories in the transition diagram satisfying Γ. These trajectories define physically possible pasts of the domain compatible with observations in Γ and the assumptions about the agent observation strategy and power. This however does not mean that every action in such a model is intended by an agent. Suppose Bob procrastinates and he $waits$ instead of performing the intended action of starting activity 1, as in the following consistent history:

$$\Gamma_2 = \begin{cases} obs(in(b,r1),true,0), obs(in(j,r3),true,0), \\ attempt(wait,0), \\ hpd(select(meet(b,j),0), hpd(wait,0), \\ obs(meet(b,j),false,1), attempt(wait,1), \\ hpd(wait,1), obs(meet(b,j),false,2) \end{cases}$$

It can be shown, however, that for histories produced by the agent executing the $\mathcal{AIA}$ control loop this is impossible. Every agent's action in every model of such a history is intended. (Histories satisfying this property are called *intentional*.) This is exactly what we require from an intentional agent.

The Reasoning Algorithms

In this section we present a refinement of the $\mathcal{AIA}$ control loop (Figure 1) in which reasoning tasks performed by the agent at step n are reduced to computing answer sets of a CR-Prolog program constructed from intentional system description $\mathcal{D}$ of $\mathcal{AL}$ and the domain history Γ_n.

The program, denoted by $\Pi(\mathcal{D}, \Gamma_n)$, consists of the following parts:

1. $\Pi(\mathcal{D})$ - the translation of $\mathcal{D}$ into ASP rules;
2. $\Pi(\Gamma_n)$ - rules for computing models of Γ_n.
3. $IA(n)$ - rules for determining intended actions at n.

The first part is not new (see for instance (Gelfond and Kahl 2013)). Construction of $\Pi(\Gamma_n)$ is based on the diagnostic module of AAA (Balduccini and Gelfond 2003). In addition to standard axioms (reality check and occurrence awareness axioms) it contains axioms encoding of our domain assumptions and the effects of a record $attempt(A, I)$ and

$\neg hpd(A, I)$.

$$\begin{array}{lll} occurs(A,I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & attempt(A,I), \\ & & not\ impossible(A,I). \\ & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & occurs(A,I), \\ & & not\ attempt(A,I). \end{array}$$

$$\begin{array}{lll} \neg occurs(A,I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & \neg hpd(A,I). \end{array}$$

The next rule is a consistency-restoring rule of CR-Prolog which allows us to compute minimal explanations of unexpected observations.

$$\begin{array}{lll} d(A,I2,I1) : occurs(A,I2) & \stackrel{+}{\leftarrow} & current\_step(I1), \\ & & physical\_exog\_act(A), \\ & & I2 < I1. \end{array}$$

The size of a minimal explanation is found by the rules

$$\begin{array}{lll} unobs(A,I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & physical\_exog\_act(A), \\ & & occurs(A,I), \\ & & not\ hpd(A,I). \end{array}$$

$$\begin{array}{lll} number\_unobs(N,I) & \leftarrow & current\_step(I), \\ & & N = \#count\{unobs(EX,IX)\}. \end{array}$$

where $\#count$ is an aggregate function.

Lemma 1 *[Computing models of Γ_n] If Γ_n is an intentional history of $\mathcal{D}$ then P_n is a model of Γ_n iff P_n is defined by some answer set A of $\Pi(\mathcal{D}) \cup \Pi(\Gamma_n)$. Moreover, for every answer set A of $\Pi(\mathcal{D}) \cup \Pi(\Gamma_n)$ $number\_unobs(x,n) \in A$ iff there are x unobserved occurrences of exogenous actions in A.*

The lemma ensures that the first step of $\mathcal{AIA}$ control loop, interpreting observations, is reduced to computing answer sets of a program.

To perform the second step – finding an intended action – we will need program $IA(n)$. It consists of an atom $interpretation(x,n)$ where x is the number of unobserved exogenous actions in the models of Γ_n and the collection of rules needed to compute an intended action. We begin with the following constraint, which requires the agent to adhere to the outcome of the reasoning completed in step **1**.

$$\begin{array}{ll} \leftarrow & current\_step(I), \\ & number\_unobs(N,I), \\ & interpretation(X,I), \\ & N \neq X. \end{array}$$

This constraint prevents the agent from assuming additional occurrences of exogenous actions. Next we notice that the collection of possible histories can be divided in four categories. The categories, which are uniquely determined by a mental state of the agent, are used in the rules for computing intended actions.

A history belongs to the first category if the agent has neither goal nor activity to commit to, i.e.

$$\begin{array}{lll} category\_1(I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & not\ active\_goal\_or\_activity(I). \end{array}$$

$$\begin{array}{lll} active\_goal\_or\_activity(I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & h(active\_goal(G),I). \\ active\_goal\_or\_activity(I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & h(active(M),I). \end{array}$$

In this case the intended action is $wait$.

$$\begin{array}{lll} intended\_action(wait,I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & category\_1(I). \end{array}$$

A history of category 2 corresponds to a mental state of the agent in which the top-level activity is still active but the goal of the activity is not.

$$\begin{array}{lll} category\_2(M,I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & \neg h(minor(M),I), \\ & & h(active(M),I), \\ & & goal(M,G), \\ & & \neg active\_goal(G). \end{array}$$

In this case the activity must be stopped.

$$\begin{array}{lll} intended\_action(stop(M),I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & category\_2(M,I). \end{array}$$

The history is of category 3 if it corresponds to a situation in which a top-level activity and its goal are both active

$$\begin{array}{lll} category\_3(M,I) & \leftarrow & current\_step(I), \\ & & interpretation(N,I), \\ & & \neg h(minor(M),I), \\ & & h(in\_progress(M),I). \end{array}$$

In this situation the intended action of the agent will simply be the next action of activity M. But there is an exception to this rule — the agent needs to check that this activity still has a chance to lead him to his goal. Otherwise, continuing the activity will be futile and it should be stopped. This behavior is achieved by the following rules:

$$\begin{array}{lll} occurs(A,I1) & \leftarrow & current\_step(I), \\ & & category\_3(M,I), \\ & & interpretation(N,I), \\ & & I \leq I1, \\ & & \neg h(minor(M),I1), \\ & & h(in\_progress(M),I1), \\ & & h(next\_action(M,A),I1), \\ & & not\ impossible(A,I1). \end{array}$$

$$
\begin{aligned}
projected\_success(M,I) \leftarrow\ & current\_step(I),\\
& interpretation(N,I),\\
& \neg h(minor(M),I),\\
& I < I1,\\
& h(active(M),I1),\\
& goal(M,G),\\
& h(G,I1).\\
\neg project\_success(M,I) \leftarrow\ & current\_step(I),\\
& interpretation(N,I),\\
& not\ project\_success(M,I).
\end{aligned}
$$

If there is an answer set of a program containing projected_success(M,I) then the next action is given by the rule:

$$
\begin{aligned}
intended\_action(A,I) \leftarrow\ & current\_step(I),\\
& interpretation(N,I),\\
& category\_3(M,I),\\
& h(next\_action(M,A),I),\\
& projected\_success(M,I).
\end{aligned}
$$

If no such answer set exists then the activity is futile. This is defined by the following cr-rule and constraint:

$$
\begin{aligned}
f(M,I) : futile(M,I) \overset{+}{\leftarrow}\ & current\_step(I),\\
& interpretation(N,I),\\
& category\_3(M,I),\\
& \neg projected\_success(M,I).
\end{aligned}
$$

$$
\begin{aligned}
\leftarrow\ & current\_step(I),\\
& interpretation(N,I),\\
& category\_3(M,I),\\
& \neg projected\_success(M,I),\\
& not\ futile(M,I).
\end{aligned}
$$

In this case the intended action is *stop*:

$$
\begin{aligned}
intended\_action(stop(M),I) \leftarrow\ & current\_step(I),\\
& interpretation(N,I),\\
& category\_3(M,I),\\
& futile(M,I).
\end{aligned}
$$

Due to the lack of space we omit the definition of category 4 in which there is an active goal but the agent does not yet have a plan to achieve it. In this case an intended action will begin executing either an activity containing a shortest plan for achieving this goal or *wait* if such activity does not exist. The planning uses cr-rules similar to that in (Blount and Gelfond 2012). The resulting program shows that, by mixing regular ASP rules with consistency restoring rules, CR-Prolog is capable of expressing rather non-trivial forms of reasoning. The following lemma ensures that step **2** of the $\mathcal{AIA}$ control loop – finding an intended action – is reduced to computing answer sets of $\Pi(\mathcal{D}, \Gamma_n)$.

Lemma 2 *[computing intended actions of Γ_n] Let Γ_n be an intentional history and x be the number of unobserved occurrences of exogenous actions in a model of Γ_n. Action e is an intended action of Γ_n iff some answer set A of $\Pi(\mathcal{D}, \Gamma_n) \cup \{interpretation(x,n).\}$ contains the atom $intended\_action(e,n)$.*

Conclusions

This paper describes the $\mathcal{AIA}$ architecture for agents whose behavior is driven by their intentions and who reason about, and act in, changing environments. We presented a formal model of an intentional agent and its environment that includes the theory of intentions $\mathcal{TI}$. Such a model was capable of representing activities, goals, and intentions. We presented an algorithm that takes the agent's knowledge, (including $\mathcal{TI}$), explains unexpected observations and computes the agent's intended action. Both reasoning tasks are reduced to computing answer sets of CR-prolog programs constructed automatically from the agent's knowledge.

References

Balduccini, M., and Gelfond, M. 2003. Logic Programs with Consistency-Restoring Rules. In Doherty, P.; McCarthy, J.; and Williams, M.-A., eds., *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, 9–18.

Balduccini, M., and Gelfond, M. 2008. The AAA Architecture: An Overview. In *AAAI Spring Symposium on Architecture of Intelligent Theory-Based Agents*.

Balduccini, M. 2007. CR-MODELS: An inference engine for CR-Prolog. In Baral, C.; Brewka, G.; and Schlipf, J., eds., *Proceedings of the 9th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'07)*, volume 3662 of *Lecture Notes in Artificial Intelligence*, 18–30. Springer.

Baral, C., and Gelfond, M. 2000. Reasoning Agents In Dynamic Domains. In *Workshop on Logic-Based Artificial Intelligence*. Kluwer Academic Publishers.

Baral, C., and Gelfond, M. 2005. Reasoning about Intended Actions. In *Proceedings of AAAI05*, 689–694.

Blount, J., and Gelfond, M. 2012. Reasoning about the intentions of agents. In Artikis, A.; Craven, R.; Kesim Çiçekli, N.; Sadighi, B.; and Stathis, K., eds., *Logic Programs, Norms and Action*, volume 7360 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 147–171.

desJardins, M.; Durfee, E.; Ortiz, C.; and M., W. 1999. A survey of research in distributed, continual planning. *AI Magazine* 13–22.

Gabaldon, A. 2009. Activity recognition with intended actions. In *IJCAI*, 1696–1701.

Gelfond, M., and Kahl, Y. 2013. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents*. Available at `http://redwood.cs.ttu.edu/~mgelfond/FALL-2012/book.pdf`.

Inclezan, D. 2010. Computing Trajectories of Dynamic Systems Using ASP and Flora-2. In *Proceedings of the Thirty Years of Nonmonotonic Reasoning (NonMon30)*.

Rao, A. S., and Georgeff, M. P. 1991. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Wooldridge, M. 2000. *Reasoning about Rational Agents*. The MIT Press, Cambridge, Massachusetts.