A Theory of Intentions for Intelligent Agents (Extended Abstract)

Justin Blount[1], Michael Gelfond[2], and Marcello Balduccini[3]

[1] Southwest Research Institute (justin.blount@gmail.com)
[2] Texas Tech University (michael.gelfond@ttu.edu)
[3] Drexel University (marcello.balduccini@gmail.com)

Abstract. We describe the $\mathcal{AIA}$ architecture for intelligent agents whose behavior is driven by their intentions and who reason about, and act in, changing environments. The description of the domain includes both the agent's environment and the general *Theory of Intentions*. The architecture is designed to enable agents to explain unexpected observations and determine which actions are *intended* at the present moment. Reasoning is 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 the 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;
- normally, the agent is capable of observing the occurrence of all relevant exogenous actions (actions not performed by the agent).

Our approach to agent design (referred to as $\mathcal{AIA}$) builds upon earlier work on the AAA architecture [1]. The main difference between AAA and $\mathcal{AIA}$ is in the organization of the 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.

Our work is obviously related to the BDI agent model [7, 9]. Space constraints prevent a thorough comparison, but we will mention some of the key differences. The BDI model is usually based on a rather complex logic, e.g. LORA [9], with multiple modalities that include beliefs, desires, intentions and time, as well as complex actions obtained from elementary actions by operators such as *if*, *while*, *choice*. By contrast,$\mathcal{AIA}$ is based on simpler, yet expressive logics that are directly executable. Revisions of beliefs, desires and intentions are also achieved quite differently in the two approaches, since BDI logics are monotonic, while $\mathcal{AIA}$ is based on non-monotonic logic. Finally, in BDI intentions are considered on par with beliefs and desires. In our work, on the other hand, intentions are precisely definable in terms of beliefs and desires. The hierarchical representation of activities in $\mathcal{AIA}$, which we introduce later in this paper, also paves the way towards establishing a connection between the flexibility of reasoning of the $\mathcal{AIA}$ architecture and the computational efficiency of HTN planning [8].

The main technical contributions of our work are the introduction of a formal theory of intentions ($\mathcal{TI}$) and the 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 its domain 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 [3] 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 [2]. 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.

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 $move$ between neighboring rooms. The door between $r3$ and $r4$ is special and can be $locked$ and $unlocked$ by both agents. 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 $r1$, 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, 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*[4] 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: Not expected to achieve goal and replanning. Suppose that as Bob is moving from $r1$ to $r2$ he observes John moving from $r3$ to $r4$. Bob should recognize

[4] 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.

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.
Scenario 3: Failure to achieve, diagnosis, and replanning. 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, but his intention will persist, and he will find a new activity (containing a plan to move to $r4$) to achieve his goal.
Scenario 4: Failure to execute, diagnosis, and replanning. 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 $r4$. $\diamond$

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.

The Representation Language

The representation language adopted in this work is an extension of action language $\mathcal{AL}$ [4]. The language is parametrized by a sorted signature containing three special sorts *actions*, *fluents*, and *statics* (properties which can, resp. cannot, 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*. Allowed statements are: *causal laws* (a **causes** l_{in} **if** $p_0, \dots, p_m$), *state constraints* (l **if** $p_0, \dots, p_m$), and *executability conditions* (**impossible** $a_0, \dots, a_k$ **if** $p_0, \dots, p_m$), where $k \geq 0$, a, a_i's are actions, l is a domain literal (the *head*), l_{in} is a literal formed by an inertial fluent, and p_i's are domain literals. 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, f is true if it follows from the truth of the body of at least one of its defining rules and is false otherwise. 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*, consisting of a goal, a plan aimed at achieving the goal, and a name. We name activities by natural numbers. For instance we can denote Bob's activity from Scenario 1 by $\langle 1, [move(b, r1, r2), move(b, r2, r3)], meet(b, j)\rangle$. In $\mathcal{AL}$ this will be represented by the statics $activity(1)$, $comp(1, 1, move(b, r1, r2))$, $comp(1, 2, move(b, r2, r3))$, $length(1, 2)$, $goal(1, meet(b, j))$, 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, i.e., a collection of agent's observations. In traditional action theories such histories determine past trajectories of the system, called *models*, the agent believes to be possible. If no such model exists the history is deemed inconsistent.

Compared to the AAA architecture, the present work also *expands the notion of domain history and modifies 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 $\neg hpd(A, I)$. The notion of model 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.

Theory of Intentions

The agent's mental state is primarily described by the two inertial fluents $active\_goal(g)$ and $status(m, k)$. The latter 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 agent's mental state by initiating and terminating its intent to execute activity m. Special exogenous mental actions $select(g)$ and $abandon(g)$, which can be thought of as being performed by the agent's controller, initiate and terminate the agent's intent to achieve goal g. Special agent action $wait$, which has no executability conditions or effects (physical or mental), can be seen as doing nothing. Since action $wait$ has no effects, it is neither a mental nor 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 a collection of axioms of $\mathcal{AL}$[6].

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

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

Action $start$ sets the value of $status$ to 0 and an agent cannot $start$ an active activity. Similarly action $stop$ deactivates, and an agent cannot $stop$ an inactive activity.

Defined fluent $child(M1, M)$ is true when $M1$ is the current component of M:

$$child(M1, M) \ \textbf{ if } \ comp(M, K+1, M1), status(M, K). \tag{2}$$

Similarly, $child\_goal(G1, G)$ is true when G and $G1$ are the goals of M and $M1$, and $descendant(M1, M)$ is defined recursively in terms of $child$. Sub-activities and subgoals are represented by defined fluent $minor(\cdot)$. We refer to activities and goals that are not $minor$ as *top-level*. Special exogenous action $select$ activates a goal, and $abandon$ deactivates a goal. A state constraint is also included, which ensures that top-level goals are no longer active once they have been achieved.

[5] The mental state includes statics, which describe activities.

[6] For space reasons we omit formal representations of some axioms.

The next axioms describe the propagation of the intent to achieve a goal to its child goal. Of course, the parent goal may be a top-level or minor goal.

The first axiom in (3) says that an unachieved minor goal $G1$ of an activity $M1$ becomes $active$ when $M1$ is the next component of an ongoing activity M. The second says that a minor goal $G1$ is no longer active when it is achieved:

$$\begin{array}{lll} active\_goal(G1) & \textbf{if} & \neg G1,\ minor(G1), child\_goal(G1,G), active\_goal(G), \\ & & goal(M1,G1), status(M1,-1). \\ \neg active\_goal(G1) & \textbf{if} & G1,\ minor(G1), child\_goal(G1,G), active\_goal(G). \end{array} \tag{3}$$

Not shown here are the third and forth axioms, which say that a minor goal $G1$ is no longer active when its parent is no longer active, and that a minor goal $G1$ of $M1$ is no longer active when $M1$ has been executed (i.e. its status is equal to its length). Defined fluents $in\_progress(M)$ and $in\_progress(G)$ are true when M and its goal G are both active. Defined fluent $next\_act(M,A)$ is true if agent action A is the next action of the ongoing execution of M. For a physical agent action of M, the axiom is:

$$\begin{array}{lll} next\_act(M,A) & \textbf{if} & phys\_agent\_act(A), status(M,K), \\ & & comp(M,K+1,A), in\_progress(M). \end{array} \tag{4}$$

Executing the next physical action of M increments the status of M:

$$\begin{array}{lll} A\ \ \textbf{causes}\ status(M,K+1) & \textbf{if} & next\_act(M,A), status(M,K), \\ & & comp(M,K+1,A), phys\_agent\_act(A). \end{array} \tag{5}$$

Along the same lines, stopping an activity causes its descendants to be inactive:

$$stop(M)\ \ \textbf{causes}\ \ status(M1,-1)\ \textbf{if}\ descendant(M1,M). \tag{6}$$

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

In our architecture the agent's behavior is specified by the following $\mathcal{AIA}$ *control loop*:

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**.

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. In step **2** the agent finds an *intended action,* i.e.: 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.

In general, a history Γ of the domain defines trajectories in the transition diagram satisfying Γ. These trajectories define possible pasts of the domain compatible with observations in Γ and the assumptions about the agent's observation strategy and ability.This however does not mean that every action in such a model is intended by an agent. This is the case, for example, if Bob procrastinates and $waits$ instead of performing the intended action $start(1)$. It can be shown, however, that this is impossible for histories produced by an agent executing the $\mathcal{AIA}$ control loop. Every agent's action in every model of such a history is intended. Such histories are called *intentional*, and 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 in which reasoning tasks are reduced to computing answer sets of a CR-Prolog program constructed from the intentional system description and the domain history.

The program, denoted by $\Pi(\mathcal{D}, \Gamma_n)$, consists of: the translation of $\mathcal{D}$ into ASP rules ($\Pi(\mathcal{D})$); rules for computing models of Γ_n ($\Pi(\Gamma_n)$); and rules for determining intended actions at n ($IA(n)$). Construction of $\Pi(\mathcal{D})$ is based on the diagnostic module of AAA [6]. In addition to standard axioms, it contains axioms encoding our domain assumptions and the effects of a record $attempt(A, I)$ and $\neg hpd(A, I)$. A consistency-restoring rule of CR-Prolog allows us to compute minimal explanations of unexpected observations:

$$\begin{aligned} occurs(A, I2) &\overset{+}{\leftarrow} phys\_exog\_act(A), curr\_step(I1),\ I2 < I1. \\ unobs(A, I) &\leftarrow I < I1, phys\_exog\_act(A), occurs(A, I), not\ hpd(A, I). \\ number\_unobs(N, I) &\leftarrow curr\_step(I), N = \#count\{unobs(EX, IX)\}. \end{aligned}$$

The following lemma links the first step and the answer sets of $\Pi(\mathcal{D}) \cup \Pi(\Gamma_n)$.

Lemma 1. *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 $\Pi = \Pi(\mathcal{D}) \cup \Pi(\Gamma_n)$. Moreover, for every answer set A of Π, $number\_unobs(x, n) \in A$ iff there are x unobserved occurrences of exogenous actions in A.*

To perform the second step – finding an intended action – we will need program $IA(n)$. It consists of an atom $interpret(x, n)$ 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. A constraint requires the agent to adhere to the outcome of the reasoning completed in step **1** by preventing 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.

For example, a history belongs to category 1 if the agent has neither goal nor activity to commit to. In this case the intended action is to $wait$. This is defined by a rule in which literal $active\_goal\_or\_activity(I)$ is true when there is an active goal or activity at the current step I. Similarly a history is of category 2 if the agent's top-level activity is active but its goal is not, and in this case the intended action is to $stop$ the activity. A history is of category 3 if the agent's top-level activity and goal are both active. In this situation the intended action will 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 achieve his goal. Other rules cause an atom $proj\_success(M, I)$ to be part of an answer set if the $next$ action is intended. If no such answer set exists, then the activity is futile, and the intended action is to $stop$. This is achieved by a cr-rule:

$$futile(M,I) \overset{+}{\leftarrow} interpret(N,I), category\_3(M,I), \neg proj\_success(M,I).$$

Finally, category 4 corresponds to the case 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 task uses cr-rules similar to those in [5]. 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. *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 $\Pi(\mathcal{D}, \Gamma_n) \cup \{interpret(x, n).\}$ contains the atom $intended\_act(e, n)$.*

Conclusions

This paper describes the $\mathcal{AIA}$ architecture for intelligent 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. A prototype can be found at http://www.depts.ttu.edu/cs/research/krlab/software-aia.php.

References

1. Balduccini, M., Gelfond, M.: The AAA Architecture: An Overview. In: AAAI Spring Symposium on Architecture of Intelligent Theory-Based Agents (2008)
2. Balduccini, M.: CR-MODELS: An inference engine for CR-Prolog. In: Baral, C., Brewka, G., Schlipf, J. (eds.) Proceedings of the 9th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR'07). Lecture Notes in Artificial Intelligence, vol. 3662, pp. 18–30. Springer (2007)
3. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: Doherty, P., McCarthy, J., Williams, M.A. (eds.) International Symposium on Logical Formalization of Commonsense Reasoning. pp. 9–18. AAAI 2003 Spring Symposium Series (Mar 2003)
4. Baral, C., Gelfond, M.: Reasoning Agents In Dynamic Domains. In: Workshop on Logic-Based Artificial Intelligence. Kluwer Academic Publishers (Jun 2000)
5. Blount, J., Gelfond, M.: Reasoning about the intentions of agents. In: Artikis, A., Craven, R., Kesim Çiçekli, N., Sadighi, B., Stathis, K. (eds.) Logic Programs, Norms and Action, Lecture Notes in Computer Science, vol. 7360, pp. 147–171. Springer Berlin / Heidelberg (2012)
6. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents. Cambridge University Press (2014)
7. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning. pp. 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA (1991)
8. Sacerdoti, E.: The nonlinear nature of plans. In: Procs of IJCAI-75 (1975)
9. Wooldridge, M.: Reasoning about Rational Agents. The MIT Press, Cambridge, Massachusetts (2000)