

# Modelling and Testing the Domain Knowledge of an Intentional Robot

Rocio Gomez and Heather Riley

November 8, 2017

## 1 The Planning Robot and the Intentional Robot

In this document we describe a domain that we will use to illustrate the capabilities of Theory of Intentions. The domain has three rooms located side by side (*office*, *kitchen* and *library*) and connected. The robot, which we call *rob*, can move from one room to the next. A room that is *secure* can be *locked* or *unlocked*. The robot cannot move to or from a locked room; it can *unlock* a locked room. The domain objects can be located in any of the rooms. The robot can *pickup* an object if the robot and the object are in the same room, and it can *putdown* an object that it is holding; the robot can only hold one object at a time. The domain includes two exogenous actions, one that can change the location of any object, and the other that can lock a secure room. The agent may or may not be aware of these exogenous action when they happen.

### 1.1 The Physical Environment Domain

The description of the domain in action language  $\mathcal{AL}$  consists of the following.

Sorts:

$$\begin{aligned} \textit{secure\_room} &= \{\textit{library}\}. \\ \textit{room} &= \textit{secure\_room} + \{\textit{kitchen}, \textit{office}\}. \\ \textit{robot} &= \{\textit{rob1}\}. \\ \textit{book} &= \{\textit{book1}, \textit{book2}\}. \\ \textit{object} &= \textit{book}. \\ \textit{thing} &= \textit{object} + \textit{robot}. \end{aligned}$$

Static relations:

*next\_to(library, kitchen).*  
*next\_to(kitchen, office).*

Inertial fluents:

*loc(thing, room).*  
*in\_hand(robot, object).*  
*locked(secure\_room).*

Robot actions:

*move(robot, room).*  
*pickup(robot, object).*  
*put\_down(robot, object).*  
*unlock(robot, secure\_room).*

Exogenous actions:

*exo\_move(object, room).*  
*exo\_lock(secure\_room).*

Causal Laws:

*move(R, L)* **causes** *loc(R, L).*  
*pickup(R, O)* **causes** *in\_hand(R, O).*  
*put\_down(R, O)* **causes**  $\neg in\_hand(R, O).$   
*unlock(R, L)* **causes**  $\neg locked(L).$   
*exo\_lock(L)* **causes** *locked(L).*  
*exo\_move(O, L)* **causes** *loc(O, L).*

State Constraints:

*next\_to(L1, L2)* **if** *next\_to(L2, L1).*  
 $\neg loc(T, L2)$  **if** *loc(T, L1), L1  $\neq$  L2.*  
*loc(O, L)* **if** *loc(R, L), in\_hand(R, O).*  
 $\neg in\_hand(R, O1)$  **if** *in\_hand(R, O2), O1  $\neq$  O2.*

Executability Conditions:

**impossible**  $move(R, L)$  **if**  $loc(R, L)$ .  
**impossible**  $move(R, L2)$  **if**  $loc(R, L1), \neg next\_to(L1, L2)$ .  
**impossible**  $move(R, L2)$  **if**  $loc(R, L1), locked(L1)$ .  
**impossible**  $move(R, L)$  **if**  $locked(L)$ .  
**impossible**  $unlock(R, L)$  **if**  $\neg locked(L)$ .  
**impossible**  $unlock(R, L1)$  **if**  $loc(R, L2), \neg next\_to(L2, L1), L2 \neq L1$ .  
**impossible**  $put\_down(R, O)$  **if**  $\neg in\_hand(R, O)$ .  
**impossible**  $pickup(R, O1)$  **if**  $in\_hand(R, O2)$ .  
**impossible**  $pickup(R, O)$  **if**  $loc(R, L1), loc(O, L2), L1 \neq L2$ .  
**impossible**  $exo\_move(O, L)$  **if**  $loc(O, L)$ .  
**impossible**  $exo\_move(O, L)$  **if**  $locked(L)$ .  
**impossible**  $exo\_move(O, L2)$  **if**  $loc(O, L1), locked(L1)$ .  
**impossible**  $exo\_move(O, L)$  **if**  $in\_hand(R, O)$ .  
**impossible**  $exo\_lock(L)$  **if**  $locked(L)$ .

Defaults:

$loc(O, library)$  **if**  $\#book(O), not \neg loc(O, library)$ .  
 $loc(O, office)$  **if**  $\#book(O), \neg loc(O, library), not \neg loc(O, office)$ .

## 1.2 Scenarios

- **Scenario 1: Planning**

Initially rob1 is in the office and book1 and book2 are in the library. Rob1 is requested to have book1 in the kitchen. Rob1 should be able to come up with a plan to achieve the requested goal.

- **Scenario 2: Unexpected achievement of goal**

Initially rob1 is in the office and book1 and book2 are in the library. Rob1 is requested to have book1 in the kitchen. When rob1 reaches the kitchen on its way to the library he observes that book1 is there. He realises he has reached the goal unexpectedly, so he stops his activity.

- **Scenario 3: Not expected to achieve goal and re-planning**

Initially rob1 is in the office and book1 and book2 are in the library. Rob1 is requested to have book1 in the kitchen. Rob1 plans to go to the library, take book1 and go back to the kitchen with the book. When rob1 is in the kitchen on

his way to the library, he observes that book1 is now in the office. His current plan would not achieve the goal, so he stops his plans and comes up with another plan to move back to the office pick up book1 and take it to the kitchen.

- **Scenario 4: Abandon goal**

Initially rob1 is in the office and book1 and book2 are in the library. Rob1 is requested to have book1 in the kitchen. Rob1 plans to go to the library, take book1 and go back to the kitchen with the book. When rob1 is in the kitchen on its way to the library, he is asked to abandon the goal, as it is not required any longer.

- **Scenario 5: Unexpected failure to execute, diagnosis, and re-planning**

Initially rob1 is in the office and book1 and book2 are in the library. Rob1 is requested to have book1 in the kitchen. Rob1 plans to go to the library, take book1 and go back to the kitchen with the book. When rob1 attempts to move from the kitchen to the library, he observes his attempt has failed and realises that the library is locked. He stops his plan, and comes up with a new plan to unlock the door, move to the library, take book1 and go back to the kitchen with the book.

- **Scenario 6: Failure to achieve goal, diagnosis, and re-planning**

Let us imagine now that initially rob1 and book1 are in the kitchen and book2 is in the library. Let us imagine that rob1 has been requested to tidy up, making sure that both books are in the library. Rob1 plans to take book1 and go to the library with it. Rob1 is now in the library with book1 and, as he puts down book1 in the library he sees that book2 is not there. He realises that, although he has completed his plan, he has not achieved his goal. He diagnosis the situation and realises that book2 must have been moved to either the kitchen or the office. Since office is the next default location after library, rob1 assumes that book2 is now in the office and comes up with a plan to go to the office, take book2 and move to the library with it.

### 1.3 The Intentional Agent (based on $\mathcal{TI}$ )

In the model of our intentional agent we include the *actions* of my agent as *physical agent actions*, the *exogenous actions* of the other agents as *physical exogenous actions*, and the *inertial* and *defined fluents* from the physical environment, as and *physical inertial fluents* and *physical defined fluents* respectively. The rest of the fluents and actions introduced here will represent mental states and mental actions of the intentional agent.

We also need to introduce the concept of possible activities of an agent. An *activity* will be represented by a triple consisting on *name*, *plan* and *goal*. A *name* is a unique identifier used to refer to the *activity*, a *goal* is a *physical inertial fluent* and a *plan* is

a sequence of *physical agent actions*, which will lead to the realisation of the *goal*. We limit the names of activities to a collection of integers ( $1 \dots max\_name$ ), the length of plans to a maximum length ( $1 \dots max\_len$ ). The fluents of the physical environment or the *physical fluents* that may serve as a *goal* are called *possible goals*.

Sorts:

```

secure_room = {library}.
room = {kitchen, office} + secure_room.
robot = {rob1}.
book = {book1, book2}.
object = book.
thing = object + robot.
index = {-1, ..., max_len}.
activity_name = {1, ..., max_name}.
boolean = {true, false}.

physical_inertial_fluent = loc(thing, room) +
                           in_hand(robot, object) +
                           locked(secure_room).

possible_goal = my_goal.
physical_defined_fluent = possible_goal.

mental_inertial_fluent = active_goal(possible_goal) +
                           next_available_name(activity_name) +
                           current_action_index(activity_name, index).

mental_defined_fluent = active_activity(activity_name) +
                           next_action(activity_name, action) +
                           in_progress_activity(activity_name) +
                           in_progress_goal(possible_goal).

defined_fluent = physical_defined_fluent + mental_defined_fluent.
inertial_fluent = physical_inertial_fluent + mental_inertial_fluent.

physical_agent_action = move(robot, room) +
                           pickup(robot, object) +
                           put_down(robot, object) +

```

$$\begin{aligned}
& \text{unlock}(\text{robot}, \text{secure\_room}). \\
\text{mental\_agent\_action} &= \text{start}(\text{activity\_name}) + \\
& \quad \text{stop}(\text{activity\_name}). \\
\text{agent\_action} &= \text{mental\_agent\_action} + \text{physical\_agent\_action} + \{\text{finish}\}. \\
\\
\text{physical\_exogenous\_action} &= \text{exo\_move}(\text{object}, \text{room}) + \\
& \quad \text{exo\_lock}(\text{secure\_room}). \\
\text{mental\_exogenous\_actions} &= \text{select}(\text{possible\_goal}) + \\
& \quad \text{abandon}(\text{possible\_goal}). \\
\\
\text{exogenous\_action} &= \text{physical\_exogenous\_action} + \text{mental\_exogenous\_action}.
\end{aligned}$$

Static relations:

$$\begin{aligned}
& \text{next\_to}(\text{library}, \text{kitchen}). \\
& \text{next\_to}(\text{kitchen}, \text{of fice}). \\
& \text{activity\_component}(\text{activity\_name}, \text{index}, \text{physical\_agent\_action}). \\
& \text{activity\_length}(\text{activity\_name}, \text{index}). \\
& \text{activity\_goal}(\text{activity\_name}, \text{possible\_goal}). \\
& \text{immediate\_child\_activity}(\text{activity\_name}, \text{activity\_name}). \\
& \text{immediate\_child\_goal}(\text{possible\_goal}, \text{possible\_goal}). \\
& \text{descendant}(\text{activity\_name}, \text{activity\_name}). \\
& \text{minor\_activity}(\text{activity\_name}). \\
& \text{minor\_goal}(\text{possible\_goal}).
\end{aligned}$$

In the next section we use possible indexed variables  $AN$  to represent activity names, and similarly for indices  $K$ , possible goals  $G$ , mental agent actions  $MAA$ , physical agent actions  $PAA$ , agent actions  $AA$ , physical exogenous actions  $PEA$ , mental exogenous actions also called mental exogenous actions  $MEA$  and exogenous actions  $EA$ .

The  $\mathcal{AL}$  statements of the  $\mathcal{TI}$  are:

Causal Laws:

$$\begin{aligned}
\text{start}(AN) \quad \mathbf{causes} \quad \text{current\_action\_index}(AN, 0). \\
\text{stop}(AN) \quad \mathbf{causes} \quad \text{current\_action\_index}(AN, -1).
\end{aligned} \tag{1}$$

$$\begin{aligned}
\text{select}(G) \quad \mathbf{causes} \quad \text{active\_goal}(G). \\
\text{abandon}(G) \quad \mathbf{causes} \quad \neg \text{active\_goal}(G).
\end{aligned} \tag{2}$$

$$\begin{aligned}
PAA \text{ causes } current\_action\_index(AN, K + 1) \text{ if } & next\_action(AN, PAA), \\
& current\_action\_index(AN, K), \\
& activity\_component(AN, K + 1, PAA).
\end{aligned} \tag{3}$$

$$start(AN) \text{ causes } next\_available\_name(AN + 1) \text{ if } next\_available\_name(AN). \tag{4}$$

$$\begin{aligned}
stop(AN1) \text{ causes } current\_action\_index(AN, K + 1) \text{ if } & current\_action\_index(AN, K), \\
& activity\_component(AN, K + 1, AN1), \\
& next\_action(AN, stop(AN1)).
\end{aligned} \tag{5}$$

$$stop(AN) \text{ causes } current\_action\_index(AN1, -1) \text{ if } descendant(AN1, AN). \tag{6}$$

State Constraints:

$$\neg current\_action\_index(AN, K1) \text{ if } current\_action\_index(AN, K2), \tag{7} \\
K1 \neq K2.$$

$$active\_activity(AN) \text{ if } \neg current\_action\_index(AN, -1). \tag{8}$$

$$\neg active\_goal(G) \text{ if } G. \tag{9}$$

$$\begin{aligned}
in\_progress\_activity(AN) \text{ if } & active\_activity(AN), \\
& activity\_goal(AN, G), \\
& active\_goal(G). \\
in\_progress\_goal(G) \text{ if } & active\_activity(AN), \\
& activity\_goal(AN, G), \\
& active\_goal(G).
\end{aligned} \tag{10}$$

$$\begin{aligned}
next\_action(AN, PAA) \text{ if } & current\_action\_index(AN, K), \\
& activity\_component(AN, K + 1, PAA), \\
& in\_progress\_activity(AN).
\end{aligned} \tag{11}$$

$$\neg \text{next\_available\_name}(AN) \quad \text{if} \quad \text{next\_available\_name}(AN1), \quad (12)$$

$$AN \neq AN1.$$

$$\text{my\_goal} \quad \text{if} \quad \% \text{ definition added at run time.} \quad (13)$$

$$\text{immediate\_child\_activity}(AN1, AN) \quad \text{if} \quad \text{activity\_component}(AN, K + 1, AN1), \quad (14)$$

$$\text{current\_action\_index}(AN, K).$$

$$\text{immediate\_child\_goal}(G1, G) \quad \text{if} \quad \text{immediate\_child\_activity}(AN1, AN), \quad (15)$$

$$\text{goal}(AN, G),$$

$$\text{goal}(AN1, G1).$$

$$\text{descendant}(AN1, AN) \quad \text{if} \quad \text{immediate\_child\_activity}(AN1, AN). \quad (16)$$

$$\text{descendant}(AN1, AN) \quad \text{if} \quad \text{descendant}(AN1, AN), \quad (17)$$

$$\text{descendant}(AN2, AN1).$$

$$\text{minor\_activity}(AN1) \quad \text{if} \quad \text{immediate\_child\_activity}(AN1, AN). \quad (18)$$

$$\text{minor\_goal}(G1) \quad \text{if} \quad \text{immediate\_child\_goal}(G1, G). \quad (19)$$

$$\text{active\_goal}(G1) \quad \text{if} \quad \text{immediate\_child\_goal}(G1, G), \quad (20)$$

$$\text{active\_goal}(G),$$

$$\text{goal}(AN1, G1),$$

$$\neg G1,$$

$$\text{current\_action\_index}(AN1, -1).$$

$$\neg \text{active\_goal}(G1) \quad \text{if} \quad \text{immediate\_child\_goal}(G1, G), \quad (21)$$

$$\neg \text{active\_goal}(G).$$

$$\neg \text{active\_goal}(G1) \quad \text{if} \quad \text{immediate\_child\_goal}(G1, G), \quad (22)$$

$$\text{active\_goal}(G),$$

$$\text{goal}(AN1, G1),$$

$$\text{current\_action\_index}(AN1, K),$$

$$\text{length}(AN1, K).$$



$$\begin{aligned}
next\_action(AN, start(AN1)) \quad \textbf{if} \quad & current\_action\_index(AN, K), \\
& activity\_component(AN, K + 1, AN1), \\
& in\_progress\_activity(AN), \\
& \neg active\_activity(AN1).
\end{aligned} \tag{23}$$

$$\begin{aligned}
next\_action(AN, AA) \quad \textbf{if} \quad & current\_action\_index(AN, K), \\
& activity\_component(AN, K + 1, AN1), \\
& in\_progress\_activity(AN), \\
& in\_progress\_activity(AN1), \\
& next\_action(AN1, AA).
\end{aligned} \tag{24}$$

$$\begin{aligned}
next\_action(AN, stop(AN1)) \quad \textbf{if} \quad & current\_action\_index(AN, K), \\
& activity\_component(AN, K + 1, AN1), \\
& in\_progress\_activity(AN), \\
& active\_activity(AN1), \\
& goal(AN1, G1), \\
& \neg active\_goal(G1).
\end{aligned} \tag{25}$$

Executability Conditions:

$$\begin{aligned}
\textbf{impossible} \quad start(AN) \quad \textbf{if} \quad & active\_activity(AN). \\
\textbf{impossible} \quad stop(AN) \quad \textbf{if} \quad & \neg active\_activity(AN).
\end{aligned} \tag{26}$$

$$\begin{aligned}
\textbf{impossible} \quad PAA, MAA. \\
\textbf{impossible} \quad MAA1, MAA2 \quad \textbf{if} \quad & MAA1 \neq MAA2.
\end{aligned} \tag{27}$$

$$\begin{aligned}
\textbf{impossible} \quad PAA, finish. \\
\textbf{impossible} \quad MAA, finish.
\end{aligned} \tag{28}$$

$$\begin{aligned}
\textbf{impossible} \quad select(G) \quad \textbf{if} \quad & active\_goal(G). \\
\textbf{impossible} \quad abandon(G) \quad \textbf{if} \quad & \neg active\_goal(G). \\
\textbf{impossible} \quad abandon(G) \quad \textbf{if} \quad & minor\_goal(G).
\end{aligned} \tag{29}$$

$$\begin{aligned}
\textbf{impossible} \quad PAA, MEA. \\
\textbf{impossible} \quad PEA, MEA. \\
\textbf{impossible} \quad MAA, MEA.
\end{aligned} \tag{30}$$

## 2 The Architecture: Reasoning Tasks and Behaviour of Our Intentional Agent.

The rules presented in this section correspond to the implementation of the reasoning tasks of the intentional agent, based on *AZA*. The architecture of the agent's behaviour is specified by the following *AZA* loop:

1. interpreting observations;
2. find an 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.

As we can see there are two major reasoning tasks in the control loop: *interpreting observations* and *finding an intended action*. The rules that define these two tasks are presented below.

### 2.1 Introducing new relations

Many of the axioms introduced in the following sections are related to the history. The history is updated and added to the ASP program at each step. It consists of three relations:

- $obs(F, B, I)$  means that fluent  $F$  is observed to have boolean value  $B$  at step  $I$ .
- $hpd(A, B, I)$  means that action  $A$  happened with boolean value  $B$  regarding its success at step  $I$ .
- $attempt(A, I)$  means that action  $A$  is attempted at step  $I$ .

The axioms that need to be added to the ASP program also involve the following relations:

- $occurs(A, I)$  means that action  $A$  occurs at step  $I$ .
- $holds(F, I)$  means that fluent  $F$  is believed at step  $I$ .
- $current\_step(I)$  means that the execution of the plan is currently at step  $I$ .

- *impossible*( $A, I$ ) means that action  $A$  is impossible at step  $I$ .
- *observed\_result*( $A, I$ ) means that the result of action  $A$  (did it happen successfully or unsuccessfully) has been observed at step  $I$ .
- *number\_unobserved*( $N, I$ ) means that the number of unobserved occurrences of exogenous actions up to step  $I$  is  $N$  (this is determined as the minimal number necessary to explain unexpected observations).
- *unobserved*( $PEA, I$ ) means that a physical exogenous action is assumed to have occurred at step  $I$ , though it wasn't observed.
- *explanation*( $N, I$ ) means that  $N$  unexpected observations need to be interpreted up to step  $I$ .
- *explaining*( $I$ ) is a flag for using the diagnosis feature.
- *identical\_activities*( $AN1, AN2$ ) means that activity  $AN1$  and activity  $AN2$  are the same (they have the same goal and components).
- *identical\_components*( $AN1, AN2$ ) means that activity  $AN1$  and  $AN2$  have the same component at every index.
- *different\_component*( $AN1, AN2$ ) means that activity  $AN1$  and  $AN2$  have different components at at least one index.
- *no\_activity\_for\_goal*( $G, I$ ) means that at step  $I$  there is an active goal but no activity with that goal is active.
- *no\_goal\_for\_activity*( $AN, I$ ) means that at step  $I$  there is an active activity but its goal is not active.
- *active\_goal\_activity*( $AN, I$ ) means that at step  $I$  there is an active activity and that activity's goal is also active.
- *intended\_action*( $A, I$ ) means that at step  $I$  the action that the agent intends to execute next is  $A$ .
- *projected\_success*( $AN, I$ ) means that at step  $I$ , continued execution of activity  $AN$  is expected to result in the goal being achieved.
- *futile\_activity*( $AN, I$ ) means that at step  $I$ , continued execution of activity  $AN$  is not expected to achieve the goal.
- *futile\_goal*( $G, I$ ) means that at step  $I$  no activity can be found that is expected to achieve the goal.
- *candidate*( $AN, I$ ) means that at step  $I$  activity  $AN$  is a candidate for the next activity to be started to achieve a goal.

- *some\_action\_occurred(I)* means that an action occurred at step I.
- *has\_component(AN, K)* means that activity AN has a component at index K.
- *has\_intention(I)* means that the agent has an intended action at step I.

## 2.2 Translating AL to ASP

The following steps should be followed in order to translate the AL description into an ASP program.

For every causal law *a* **causes** *l* **if** *p0, ..., pm*:

The ASP contains *holds(l, I+1) :- holds(p0, I), ..., holds(pm, I), occurs(a, I)*.

For every state constraint *l* **if** *p0, ..., pm*:

The ASP contains *holds(l, I) :- holds(p0, I), ..., holds(pm, I)*.

The ASP contains the CWA for defined fluents:

*-holds(F, I) :- #defined\_fluent(F), not holds(F, I)*.

For every executability condition **impossible** *a* **if** *p0, ..., pm*:

The ASP contains *impossible(a, I) :- holds(p0, I), ..., holds(pm, I)*.

It also contains *-occurs(A, I) :- impossible(A, I)*.

The ASP contains the inertia axioms:

*holds(F, I+1) :- holds(F, I), not -holds(F, I+1)*.

*-holds(F, I+1) :- -holds(F, I), not -holds(F, I+1)*.

The ASP contains the CWA for actions:

*-occurs(A, I) :- not occurs(A, I)*.

Once translation using the above steps has been completed, the axioms in the following section will also need to be added to the ASP program.

## 2.3 Rules for past history and observations

$$\begin{aligned}
 \text{holds}(F, 0) &\leftarrow \text{obs}(F, \text{true}, 0). \\
 \neg \text{holds}(F, 0) &\leftarrow \text{obs}(F, \text{false}, 0).
 \end{aligned} \tag{31}$$

Reality check axioms which guarantee the agent's observations of the past do not con-

tradict his expectations.

$$\begin{aligned}
&\leftarrow \text{current\_step}(I1), \\
&\quad I \leq I1, \\
&\quad \text{obs}(F, \text{false}, I), \\
&\quad \text{holds}(F, I). \\
&\leftarrow \text{current\_step}(I1), \\
&\quad I \leq I1, \\
&\quad \text{obs}(F, \text{true}, I), \\
&\quad \neg \text{holds}(F, I).
\end{aligned} \tag{32}$$

The occurrences of actions that are observed to have happened or not happened did actually occur or not occur.

$$\begin{aligned}
\text{occurs}(A, I) &\leftarrow \text{current\_step}(I1), \\
&\quad I < I1, \\
&\quad \text{hpd}(A, \text{true}, I). \\
\neg \text{occurs}(A, I) &\leftarrow \text{current\_step}(I1), \\
&\quad I < I1, \\
&\quad \text{hpd}(A, \text{false}, I).
\end{aligned} \tag{33}$$

If an observation did not occur is due to the violation of an executability condition for that action.

$$\begin{aligned}
\text{occurs}(AA, I) &\leftarrow \text{current\_step}(I1), \\
&\quad I < I1, \\
&\quad \text{attempt}(AA, I), \\
&\quad \text{not impossible}(AA, I). \\
&\leftarrow \text{current\_step}(I1), \\
&\quad I < I1, \\
&\quad \text{occurs}(AA, I), \\
&\quad \text{not attempt}(AA, I).
\end{aligned} \tag{34}$$

The agent's controller does not simultaneously select multiple goals and only selects a

goal when the agent has neither an active goal or an active activity.

$$\begin{aligned}
\text{impossible}(\text{select}(G), I) &\leftarrow \text{current\_step}(I1), \\
&I < I1, \\
&\text{occurs}(\text{select}(G1), I), \\
&G \neq G1. \\
\text{impossible}(\text{select}(G), I) &\leftarrow \text{current\_step}(I1), \\
&I < I1, \\
&\text{holds}(\text{active\_activity}(AN), I). \\
\text{impossible}(\text{select}(G), I) &\leftarrow \text{current\_step}(I1), \\
&I < I1, \\
&\text{holds}(\text{active\_goal}(G1), I).
\end{aligned} \tag{35}$$

Initial observations must be legal:

$$\begin{aligned}
&\text{holds}(\text{current\_action\_index}(AN, -1), 0). \\
&\neg \text{holds}(\text{active\_goal}(G), 0). \\
&\text{holds}(\text{next\_available\_name}(1), 0).
\end{aligned} \tag{36}$$

The agent always observes the results of his attempts to perform actions.

$$\begin{aligned}
\text{observed\_result}(AA, I) &\leftarrow \text{current\_step}(I1), \\
&I \leq I1, \\
&\text{hpd}(AA, B, I). \\
&\leftarrow \text{current\_step}(I1), \\
&I \leq I1, \\
&\text{attempt}(AA, I), \\
&\text{not observed\_result}(AA, I).
\end{aligned} \tag{37}$$

The agent always observes the actions performed by his controller.

$$\begin{aligned}
&\leftarrow \text{current\_step}(I1), \\
&I < I1, \\
&\text{occurs}(\text{select}(G), I), \\
&\text{not hpd}(\text{select}(G), \text{true}, I). \\
&\leftarrow \text{current\_step}(I1), \\
&I < I1, \\
&\text{occurs}(\text{abandon}(G), I), \\
&\text{not hpd}(\text{abandon}(G), \text{true}, I).
\end{aligned} \tag{38}$$

## 2.4 Diagnosis of Unexpected Observations

This limits the number of unobserved occurrences of exogenous actions to the minimal number of unobserved actions necessary to satisfy the unexpected observations.

$$\begin{aligned} occurs(PEA, I) \quad & \stackrel{+}{\leftarrow} \quad current\_step(I1), \\ & I < I1, \\ & explaining(I). \end{aligned} \tag{39}$$

$$\begin{aligned} unobserved(PEA, I) \quad & \leftarrow \quad current\_step(I1), \\ & I < I1, \\ & explaining(I), \\ & occurs(PEA, I), \\ & not \ hpd(PEA, true, I). \end{aligned} \tag{40}$$

$$\begin{aligned} number\_unobserved(N, I) \quad & \leftarrow \quad current\_step(I), \\ & N = \#count\{EX : unobserved(EX, IX)\}, \\ & explaining(I). \end{aligned} \tag{41}$$

## 2.5 Rules for Finding Intended Actions

Activities must be unique.

$$\begin{aligned} different\_component(AN, AN1) \quad & \leftarrow \quad activity\_component(AN, K, AA), \\ & activity\_component(AN1, K, AA1), \\ & AA \neq AA1. \\ \\ identical\_components(AN, AN1) \quad & \leftarrow \quad activity\_length(AN, L), \\ & activity\_length(AN1, L), \\ & not \ different\_component(AN, AN1). \end{aligned} \tag{42}$$

$$\begin{aligned} identical\_activities(AN, AN1) \quad & \leftarrow \quad activity\_goal(AN, G), \\ & activity\_goal(AN1, G), \\ & identical\_components(AN, AN1). \\ \\ & \leftarrow \quad identical\_activities(AN, AN1), \\ & AN \neq AN1. \end{aligned}$$

The choice of the intended next action depends on the history and other added observations of the present state. A history can imply three different situations:

The situation in which we have an active goal, but no active activity for this goal, so the goal is not in progress. This situation happens at the beginning, just after the goal has been selected but the activity has not been created, or when an activity has stopped because it is futile and another activity is necessary.

$$\begin{aligned}
no\_activity\_for\_goal(G, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& \textcolor{red}{\neg minor\_goal(G)}, \\
& holds(active\_goal(G), I), \\
& \neg holds(in\_progress\_goal(G), I).
\end{aligned} \tag{43}$$

The situation in which we have an active activity  $AN$ , but the goal of the activity is not active any longer. This may be the case because the goal of the activity has been reached, or because the goal of the activity is futile.

$$\begin{aligned}
no\_goal\_for\_activity(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& \textcolor{red}{\neg minor\_activity(AN)}, \\
& holds(active\_activity(AN), I), \\
& activity\_goal(AN, G), \\
& \neg holds(active\_goal(G), I).
\end{aligned} \tag{44}$$

The situation in which we have an active activity  $AN$ , and an active goal, so the goal is in progress.

$$\begin{aligned}
active\_goal\_activity(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& \textcolor{red}{\neg minor\_activity(AN)}, \\
& holds(in\_progress\_activity(AN), I).
\end{aligned} \tag{45}$$

Now we give the rules that describe the agent's intended action for each situation. We start from the end. When the activity is active, but not its goal (because it has been reached or because it is a futile goal), the intended next action is to finish.

$$\begin{aligned}
intended\_action(finish, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_goal\_for\_activity(AN, I).
\end{aligned} \tag{46}$$



The following four rules determine the next intended action in the situation in which there is an active goal and an active activity. The first three rules will determine if the active activity  $AN$  still has a projected success (i.e. would achieve the goal according to the current situation). The fourth rule gives the intended action if the activity has projected success.

$$\begin{aligned}
occurs(AA, I1) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& I \leq I1, \\
& \neg minor\_activity(AN), \\
& active\_goal\_activity(AN, I), \\
& holds(in\_progress\_activity(AN), I1), \\
& holds(next\_action(AN, AA), I1), \\
& not\ impossible(AA, I1).
\end{aligned} \tag{47}$$

$$\begin{aligned}
projected\_success(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& \neg minor\_activity(AN), \\
& I < I1, \\
& holds(active\_activity(AN), I1), \\
& activity\_goal(AN, G), \\
& holds(G, I1).
\end{aligned} \tag{48}$$

$$\begin{aligned}
\neg projected\_success(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& not\ projected\_success(AN, I).
\end{aligned} \tag{49}$$

$$\begin{aligned}
intended\_action(AA, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& holds(next\_action(AN, AA), I), \\
& projected\_success(AN, I).
\end{aligned} \tag{50}$$

If the active activity has projected success, the intention is given by the above rule. If the activity does not have projected success, the activity is declared futile (next two

rules) and the intended action is to stop.

$$\begin{aligned}
&\leftarrow \text{current\_step}(I), \\
&\text{explanation}(N, I), \\
&\text{active\_goal\_activity}(AN, I), \\
&\neg \text{projected\_success}(AN, I), \\
&\text{not futile\_activity}(AN, I).
\end{aligned} \tag{51}$$

$$\begin{aligned}
\text{futile\_activity}(AN, I) \quad &\stackrel{\pm}{\leftarrow} \text{current\_step}(I), \\
&\text{explanation}(N, I), \\
&\text{active\_goal\_activity}(AN, I), \\
&\neg \text{projected\_success}(AN, I).
\end{aligned} \tag{52}$$

$$\begin{aligned}
\text{intended\_action}(\text{stop}(AN), I) \quad &\leftarrow \text{current\_step}(I), \\
&\text{explanation}(N, I), \\
&\text{active\_goal\_activity}(AN, I), \\
&\text{futile\_activity}(AN, I).
\end{aligned} \tag{53}$$

The following rules determine the intended action in the situation in which we have an active goal  $G$ , but not an active activity yet. The intended action in this case is to either *start* and activity which execution is expected to achieve the goal  $G$  in as few occurrences of physical actions as possible, or to *finish* if there is no such activity. The activity with goal  $G$  under consideration is called *candidate*. The first intended action is to *start* a candidate that has a *total execution* that is *minimal*. There may be more than one candidate with a minimal total execution but the agent will only attempt to perform one of them.

$$\begin{aligned}
\text{candidate}(AN, I) \quad &\leftarrow \text{current\_step}(I), \\
&\text{explanation}(N, I), \\
&\text{no\_activity\_for\_goal}(G, I), \\
&\text{holds}(\text{next\_available\_name}(AN), I).
\end{aligned} \tag{54}$$

$$\begin{aligned}
\text{activity\_goal}(AN, G) \quad &\leftarrow \text{current\_step}(I), \\
&\text{explanation}(N, I), \\
&\text{no\_activity\_for\_goal}(G, I), \\
&\text{candidate}(AN, I).
\end{aligned} \tag{55}$$

Only one activity can start at a time.

$$\begin{aligned}
impossible(start(AN), I) \leftarrow & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{activity\_goal}(AN1, G), \\
& \text{occurs}(start(AN1), I), \\
& AN \neq AN1.
\end{aligned} \tag{56}$$

$$\begin{aligned}
occurs(start(AN), I) \leftarrow & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{candidate}(AN, I), \\
& \text{activity\_goal}(AN, G), \\
& \text{not impossible}(start(AN), I).
\end{aligned} \tag{57}$$

The following rule guarantees that candidates that have started (by rule given above) are expected to achieve the goal. If there is not a candidate that can achieve the goal (or not have a projected success), the goal is futile and the intended action is to finish.

$$\begin{aligned}
\leftarrow & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{occurs}(start(AN), I), \\
& \neg \text{projected\_success}(AN, I), \\
& \text{not futile\_goal}(G, I).
\end{aligned} \tag{58}$$

$$\begin{aligned}
futile\_goal(G, I) \stackrel{+}{\leftarrow} & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{occurs}(start(AN), I), \\
& \neg \text{projected\_success}(AN, I).
\end{aligned} \tag{59}$$

$$\begin{aligned}
intended\_action(finish, I) \leftarrow & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{futile\_goal}(G, I).
\end{aligned} \tag{60}$$

Auxiliary rule necessary to create candidate activities.

$$\begin{aligned}
\text{some\_action\_occurred}(I1) \quad \leftarrow \quad & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{occurs}(A, I1), \\
& I \leq I1.
\end{aligned} \tag{61}$$

Creating an candidate: The first rule generates a minimal uninterrupted sequence of occurrences of physical actions. The second rule creates components based on those occurrences. The third rule guarantees that multiple actions do not have the same index. The fourth and fifth rules describe the length of a the candidate activity.

$$\begin{aligned}
\text{occurs}(PAA, I1) \quad \stackrel{+}{\leftarrow} \quad & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{candidate}(AN, I), \\
& \text{occurs}(\text{start}(AN), I), \\
& I < I1, \\
& \text{some\_action\_occurred}(I1 - 1).
\end{aligned} \tag{62}$$

$$\begin{aligned}
\text{activity\_component}(AN, I1 - I, PAA) \quad \leftarrow \quad & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& I < I1, \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{candidate}(AN, I), \\
& \text{occurs}(\text{start}(AN), I), \\
& \text{occurs}(PAA, I1).
\end{aligned} \tag{63}$$

$$\begin{aligned}
\leftarrow \quad & \text{current\_step}(I), \\
& \text{explanation}(N, I), \\
& \text{no\_activity\_for\_goal}(G, I), \\
& \text{candidate}(AN, I), \\
& \text{activity\_component}(AN, K, PAA1), \\
& \text{activity\_component}(AN, K, PAA2), \\
& PAA1 \neq PAA2.
\end{aligned} \tag{64}$$

$$\begin{aligned}
has\_component(AN, K) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& activity\_component(AN, K, C).
\end{aligned} \tag{65}$$

$$\begin{aligned}
activity\_length(AN, K) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& has\_component(AN, K), \\
& not\ has\_component(AN, K + 1).
\end{aligned} \tag{66}$$

And finally, the intended action:

$$\begin{aligned}
intended\_action(start(AN), I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& projected\_success(AN, I).
\end{aligned} \tag{67}$$

## 2.6 Automatic Behaviour

The following rule forces the model to have an intention in the current state.

$$\begin{aligned}
has\_intention(I) \quad \leftarrow \quad & intended\_action(AA, I). \\
\leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& 0 < I, \\
& not\ has\_intention(I).
\end{aligned} \tag{68}$$