# Modelling and Testing the Domain Knowledge of an Intentional Robot

Rocio Gomez and Heather Riley

March 2, 2018

## 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 four rooms located side by side ($office_1$, $office_2$, *kitchen* and *library*) and connected. The robot, which we call $rob_1$, 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.

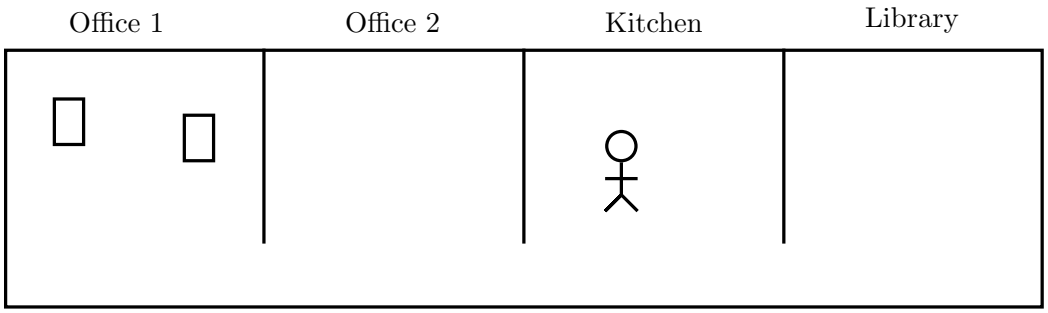


Figure 1: The physical environment

### 1.1 The Physical Environment Domain

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

Sorts:

$$
\begin{array}{l}
secure\_room = \{library\}. \\
room = secure\_room \ + \{kitchen, office_1, office_2\}. \\
robot = \{rob_1\}. \\
book = \{book_1, book_2\}. \\
object = book. \\
thing = object \ + \ robot.
\end{array}
$$

Static relations:

$$
\begin{array}{ll}
next\_to(office_1, office_2). & next\_to(office_2, kitchen). \\
next\_to(kitchen, library). &
\end{array}
$$

Inertial fluents:

$$
\begin{array}{l}
loc(thing, room). \\
in\_hand(robot, object). \\
locked(secure\_room).
\end{array}
$$

Robot actions:

$$
\begin{array}{l}
move(robot, room). \\
pickup(robot, object). \\
put\_down(robot, object). \\
unlock(robot, secure\_room).
\end{array}
$$

Exogenous actions:

$$
\begin{array}{l}
exo\_move(object, room). \\
exo\_lock(secure\_room).
\end{array}
$$

Causal Laws:

$$move(R, L) \quad \textbf{causes} \quad loc(R, L)$$

$$
\begin{array}{rll}
pickup(R,O) & \textbf{causes} & in\_hand(R,O). \\
put\_down(R,O) & \textbf{causes} & \neg in\_hand(R,O). \\
unlock(R,L) & \textbf{causes} & \neg locked(L). \\
exo\_lock(L) & \textbf{causes} & locked(L). \\
exo\_move(O,L) & \textbf{causes} & loc(O,L).
\end{array}
$$

State Constraints:

$$
\begin{array}{rll}
next\_to(L1,L2) & \textbf{if} & next\_to(L2,L1). \\
\neg loc(T,L2) & \textbf{if} & loc(T,L1),\ L1 \neq L2. \\
loc(O,L) & \textbf{if} & loc(R,L),\ in\_hand(R,O). \\
\neg in\_hand(R,O1) & \textbf{if} & in\_hand(R,O2),\ O1 \neq O2.
\end{array}
$$

Executability Conditions:

$$
\begin{array}{rrll}
\textbf{impossible} & move(R,L) & \textbf{if} & loc(R,L). \\
\textbf{impossible} & move(R,L2) & \textbf{if} & loc(R,L1),\ \neg next\_to(L1,L2). \\
\textbf{impossible} & move(R,L2) & \textbf{if} & loc(R,L1),\ locked(L1). \\
\textbf{impossible} & move(R,L) & \textbf{if} & locked(L). \\
\textbf{impossible} & unlock(R,L) & \textbf{if} & \neg locked(L). \\
\textbf{impossible} & unlock(R,L1) & \textbf{if} & loc(R,L2),\ \neg next\_to(L2,L1),\ L2 \neq L1. \\
\textbf{impossible} & put\_down(R,O) & \textbf{if} & \neg in\_hand(R,O). \\
\textbf{impossible} & pickup(R,O1) & \textbf{if} & in\_hand(R,O2). \\
\textbf{impossible} & pickup(R,O) & \textbf{if} & loc(R,L1),\ loc(O,L2),\ L1 \neq L2. \\
\textbf{impossible} & exo\_move(O,L) & \textbf{if} & loc(O,L) \\
\textbf{impossible} & exo\_move(O,L) & \textbf{if} & locked(L). \\
\textbf{impossible} & exo\_move(O,L2) & \textbf{if} & loc(O,L1), locked(L1). \\
\textbf{impossible} & exo\_move(O,L) & \textbf{if} & in\_hand(R,O). \\
\textbf{impossible} & exo\_lock(L) & \textbf{if} & locked(L).
\end{array}
$$

Defaults:

$$
\begin{array}{rll}
loc(O,library) & \textbf{if} & \#book(O),\ not\ \neg loc(O,library). \\
loc(O,office_1) & \textbf{if} & \#book(O),\ \neg loc(O,library),\ not\ \neg loc(O,office_1).
\end{array}
$$

## 1.2 Scenarios

- **Scenario 1 (planning):** Robot $rob_1$ is in the kitchen holding $book_1$, and believes $book_2$ is in the kitchen and library is unlocked. The computed plan is: `move(`$rob_1$`, library)`, `put_down(`$rob_1$`, `$book_1$`)`, `move(`$rob_1$`, kitchen)`, `pickup(`$rob_1$`, `$book_2$`)`, `move(`$rob_1$`, library)`, `put_down(`$rob_1$`, `$book_2$`)`.
- **Scenario 2 (unexpected success):** Assume that $rob_1$ in scenario-1 has moved to the library and put down $book_1$, and observes $book_2$ in the library. The robot should be able to explain this observation (e.g., $book_2$ was moved there) and realize the goal has ben achieved.
- **Scenario 3 (not expected to achieve goal, diagnose and replan, case 1):** Assume $rob1$ in scenario-1 starts moving $book_1$ to library, but observes $book_2$ is not in the kitchen. Now, $rob_1$ should realize the plan will fail, explain the observation and compute a new plan.
- **Scenario 4 (not expected to achieve goal, diagnose and replan, case 2):** Assume $rob1$ is in the kitchen holding $book_1$, and believes $book_2$ is in $office_1$ and library is unlocked. The robot plans to first put $book_1$ in the library before fetching $book_2$ from $office_1$. Before $rob_1$ moves to the library, it suddenly observes $book_2$ in the kitchen. Now, $rob_1$ should realize the plan will fail, explain the observation and compute a new plan.
- **Scenario 5 (failure to achieve the goal, diagnose and replan):** Assume $rob_1$ in scenario-1 is putting $book_2$ in the library, after having put $book_1$ in library earlier, and observes that $book_1$ is not there. Now, $rob_1$'s intention should persist and it should explain observation(s), plan and execute actions until the goal is achieved.

## 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 $(1 \ldots max\_name)$, the length of

plans to a maximum length $(1 \ldots max\_len)$. The fluents of the physical environment or the *physical fluents* that may serve as a *goal* are called *possible goals*.

Sorts:

$$
\begin{aligned}
&secure\_room = \{library\}.\\
&room = \{kitchen, office_1, office_2\} + secure\_room.\\
&robot = \{rob_1\}.\\
&book = \{book_1, book_2\}.\\
&object = book.\\
&thing = object \ + \ robot.\\
&index = \{-1, \ldots, max\_len\}.\\
&activity\_name = \{1, \ldots, max\_name\}.\\
&boolean = \{true, false\}.
\end{aligned}
$$

$$
\begin{aligned}
physical\_inertial\_fluent = \ & loc(thing, room)+\\
& in\_hand(robot, object)+\\
& locked(secure\_room).\\
possible\_goal = \ & my\_goal.\\
physical\_defined\_fluent = \ & possible\_goal.
\end{aligned}
$$

$$
\begin{aligned}
mental\_inertial\_fluent = \ & active\_goal(possible\_goal)+\\
& next\_available\_name(activity\_name)+\\
& current\_action\_index(activity\_name, index).\\
mental\_defined\_fluent = \ & active\_activity(activity\_name)+\\
& next\_action(activity\_name, action)+\\
& in\_progress\_activity(activity\_name)+\\
& in\_progress\_goal(possible\_goal).
\end{aligned}
$$

$$
\begin{aligned}
&defined\_fluent = physical\_defined\_fluent + mental\_defined\_fluent.\\
&inertial\_fluent = physical\_inertial\_fluent + mental\_inertial\_fluent.
\end{aligned}
$$

$$
\begin{aligned}
physical\_agent\_action = \ & move(robot, room)+\\
& pickup(robot, object)+\\
& put\_down(robot, object)+\\
& unlock(robot, secure\_room).\\
mental\_agent\_action = \ & start(activity\_name)+
\end{aligned}
$$

$$stop(activity\_name).$$

$$agent\_action = mental\_agent\_action + physical\_agent\_action + \{finish\}.$$

$$\begin{aligned} physical\_exogenous\_action &= exo\_move(object, room) + \\ &\quad exo\_lock(secure\_room). \\ mental\_exogenous\_actions &= select(possible\_goal) + \\ &\quad abandon(possible\_goal). \end{aligned}$$

$$exogenous\_action = physical\_exogenous\_action + mental\_exogenous\_action.$$

Static relations:

$$\begin{aligned} &next\_to(office_1, office_2). \\ &next\_to(office_2, kitchen). \\ &next\_to(kitchen, library). \\ &activity\_component(activity\_name, index, physical\_agent\_action). \\ &activity\_length(activity\_name, index). \\ &activity\_goal(activity\_name, possible\_goal). \end{aligned}$$

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:

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

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

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

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

State Constraints:

$$\neg current\_action\_index(AN, K1) \quad \textbf{if} \quad current\_action\_index(AN, K2), \; K1 \neq K2. \tag{5}$$

$$active\_activity(AN) \quad \textbf{if} \quad \neg current\_action\_index(AN, -1). \tag{6}$$

$$\neg active\_goal(G) \quad \textbf{if} \quad G. \tag{7}$$

$$\begin{aligned} in\_progress\_activity(AN) \quad \textbf{if} \quad & active\_activity(AN). \\ & activity\_goal(AN, G), \\ & active\_goal(G). \\ in\_progress\_goal(G) \quad \textbf{if} \quad & active\_activity(AN). \\ & activity\_goal(AN, G), \\ & active\_goal(G). \end{aligned} \tag{8}$$

$$\begin{aligned} next\_action(AN, PAA) \quad \textbf{if} \quad & current\_action\_index(AN, K), \\ & activity\_component(AN, K+1, PAA), \\ & in\_progress\_activity(AN). \end{aligned} \tag{9}$$

$$\neg next\_available\_name(AN) \quad \textbf{if} \quad next\_available\_name(AN1), \; AN \neq AN1. \tag{10}$$

$$my\_goal \quad \textbf{if} \quad \%\ definition\ added\ at\ run\ time. \tag{11}$$

Executability Conditions:

$$\begin{aligned} &\textbf{impossible} \quad start(AN) \quad \textbf{if} \quad active\_activity(AN). \\ &\textbf{impossible} \quad stop(AN) \quad \textbf{if} \quad \neg active\_activity(AN). \end{aligned} \tag{12}$$

$$\begin{aligned} &\textbf{impossible} \quad PAA, MAA. \\ &\textbf{impossible} \quad MAA1, MAA2 \quad \textbf{if} \quad MAA1 \neq MAA2. \end{aligned} \tag{13}$$

$$\begin{array}{l} \textbf{impossible} \quad PAA,\ finish. \\ \textbf{impossible} \quad MAA,\ finish. \end{array} \tag{14}$$

$$\begin{array}{lll} \textbf{impossible} & select(G) & \textbf{if} \quad active\_goal(G). \\ \textbf{impossible} & abandon(G) & \textbf{if} \quad \neg active\_goal(G). \end{array} \tag{15}$$

$$\begin{array}{l} \textbf{impossible} \quad PAA, MEA. \\ \textbf{impossible} \quad PEA, MEA. \\ \textbf{impossible} \quad MAA, MEA. \end{array} \tag{16}$$

# 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 $\mathcal{AIA}$. The architecture of the agent's behaviour is specified by the following $\mathcal{AIA}$ 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 fo 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.
- *selected_goal_holds(G,I)* means that a selected goal G holds 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

$$\begin{array}{rcl} holds(F,0) & \leftarrow & obs(F,true,0). \\ \neg holds(F,0) & \leftarrow & obs(F,false,0). \end{array} \tag{17}$$

Reality check axioms which guarantee the agent's observations of the past do not contradict his expectations.

$$\begin{array}{rcl} \leftarrow & & current\_step(I1), \\ & & I \leq I1, \\ & & obs(F,false,I), \\ & & holds(F,I). \\ \leftarrow & & current\_step(I1), \\ & & I \leq I1, \\ & & obs(F,true,I), \\ & & \neg holds(F,I). \end{array} \tag{18}$$

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

$$\begin{array}{rcl} occurs(A,I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & hpd(A,true,I). \\ \neg occurs(A,I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & hpd(A,false,I). \end{array} \tag{19}$$

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

that action.

$$\begin{array}{rcl} occurs(AA, I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & attempt(AA, I), \\ & & not\ impossible(AA, I). \\ & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & occurs(AA, I), \\ & & not\ attempt(AA, I). \end{array} \tag{20}$$

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.

$$\begin{array}{rcl} impossible(select(G), I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & occurs(select(G1), I), \\ & & G \neq G1. \\ impossible(select(G), I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & holds(active\_activity(AN), I). \\ impossible(select(G), I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & holds(active\_goal(G1), I). \end{array} \tag{21}$$

Initial observations must be legal:

$$\begin{array}{r} holds(current\_action\_index(AN, -1), 0). \\ \neg holds(active\_goal(G), 0). \\ holds(next\_available\_name(1), 0). \end{array} \tag{22}$$

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

$$\begin{array}{rcl} observed\_result(AA, I) & \leftarrow & current\_step(I1), \\ & & I \leq I1, \\ & & hpd(AA, B, I). \\ & \leftarrow & current\_step(I1), \\ & & I \leq I1, \\ & & attempt(AA, I), \\ & & not\ observed\_result(AA, I). \end{array} \tag{23}$$

The agent always observes the actions performed by his controller.

$$\begin{array}{lll} \leftarrow & current\_step(I1), & \\ & I < I1, & \\ & occurs(select(G), I), & \\ & not\ hpd(select(G), true, I). & \\ \leftarrow & current\_step(I1), & \\ & I < I1, & \\ & occurs(abandon(G), I), & \\ & not\ hpd(abandon(G), true, I). & \end{array} \tag{24}$$

## 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.

$$\begin{array}{lll} occurs(PEA, I) & \stackrel{+}{\leftarrow} & current\_step(I1), \\ & & I < I1, \\ & & explaining(I). \end{array} \tag{25}$$

$$\begin{array}{lll} unobserved(PEA, I) & \leftarrow & current\_step(I1), \\ & & I < I1, \\ & & explaining(I), \\ & & occurs(PEA, I), \\ & & not\ hpd(PEA, true, I). \end{array} \tag{26}$$

$$\begin{array}{lll} number\_unobserved(N, I) & \leftarrow & current\_step(I), \\ & & N = \#count\{EX : unobserved(EX, IX)\}, \\ & & explaining(I). \end{array} \tag{27}$$

## 2.5 Rules for Finding Intended Actions

Activities must be unique.

$$
\begin{array}{rcl}
different\_component(AN, AN1) & \leftarrow & activity\_component(AN, K, AA), \\
 & & activity\_component(AN1, K, AA1), \\
 & & AA \neq AA1. \\
 & & \\
identical\_components(AN, AN1) & \leftarrow & activity\_length(AN, L), \\
 & & activity\_length(AN1, L), \\
 & & not\ different\_component(AN, AN1). \\
 & & \\
identical\_activities(AN, AN1) & \leftarrow & activity\_goal(AN, G), \\
 & & activity\_goal(AN1, G), \\
 & & identical\_components(AN, AN1). \\
 & & \\
 & \leftarrow & identical\_activities(AN, AN1), \\
 & & AN \neq AN1.
\end{array}
\tag{28}
$$

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.

$$
\begin{array}{rcl}
no\_activity\_for\_goal(G, I) & \leftarrow & current\_step(I), \\
 & & explanation(N, I), \\
 & & holds(active\_goal(G), I), \\
 & & \neg holds(in\_progress\_goal(G), I).
\end{array}
\tag{29}
$$

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.

$$
\begin{array}{rcl}
no\_goal\_for\_activity(AN, I) & \leftarrow & current\_step(I), \\
 & & explanation(N, I), \\
 & & holds(active\_activity(AN), I), \\
 & & activity\_goal(AN, G), \\
 & & \neg holds(active\_goal(G), I).
\end{array}
\tag{30}
$$

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

$$\begin{aligned} active\_goal\_activity(AN, I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & holds(in\_progress\_activity(AN), I). \end{aligned} \tag{31}$$

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, the intended next action is to finish.

$$\begin{aligned} intended\_action(finish, I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_goal\_for\_activity(AN, I). \end{aligned} \tag{32}$$

When the selected goal holds, the intended next action is to finish.

$$\begin{aligned} selected\_goal\_holds(G, I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & holds(G, I), \\ & occurs(select(G), I1), \\ & I1 \leq I. \end{aligned} \tag{33}$$

$$\begin{aligned} intended\_action(finish, I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & selected\_goal\_holds(G, I). \end{aligned} \tag{34}$$

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.

$$\begin{aligned} occurs(AA, I1) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & I \leq I1, \\ & active\_goal\_activity(AN, I), \\ & holds(in\_progress\_activity(AN), I1), \\ & holds(next\_action(AN, AA), I1), \\ & not\ impossible(AA, I1). \end{aligned} \tag{35}$$

$$
\begin{aligned}
projected\_success(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& I < I1, \\
& holds(active\_activity(AN), I1), \\
& activity\_goal(AN, G), \\
& holds(G, I1).
\end{aligned}
\tag{36}
$$

$$
\begin{aligned}
\neg projected\_success(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& not\ projected\_success(AN, I).
\end{aligned}
\tag{37}
$$

$$
\begin{aligned}
intended\_action(AA, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& holds(next\_action(AN, AA), I), \\
& projected\_success(AN, I).
\end{aligned}
\tag{38}
$$

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.

$$
\begin{aligned}
\leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& \neg projected\_success(AN, I), \\
& not\ futile\_activity(AN, I).
\end{aligned}
\tag{39}
$$

$$
\begin{aligned}
futile\_activity(AN, I) \quad \stackrel{+}{\leftarrow} \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& \neg projected\_success(AN, I).
\end{aligned}
\tag{40}
$$

$$
\begin{aligned}
intended\_action(stop(AN), I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& futile\_activity(AN, I).
\end{aligned}
\tag{41}
$$

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.

$$\begin{aligned} candidate(AN, I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & holds(next\_available\_name(AN), I). \end{aligned} \tag{42}$$

$$\begin{aligned} activity\_goal(AN, G) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I). \end{aligned} \tag{43}$$

Only one activity can start at a time.

$$\begin{aligned} impossible(start(AN), I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & activity\_goal(AN1, G), \\ & occurs(start(AN1), I), \\ & AN \neq AN1. \end{aligned} \tag{44}$$

$$\begin{aligned} occurs(start(AN), I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I), \\ & activity\_goal(AN, G), \\ & not\ impossible(start(AN), I). \end{aligned} \tag{45}$$

Auxiliary rule necessary to create candidate activities.

$$\begin{aligned} some\_action\_occurred(I1) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & occurs(A, I1), \\ & I \leq I1. \end{aligned} \tag{46}$$

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.

$$\begin{aligned} occurs(PAA, I1) \quad \stackrel{+}{\leftarrow} \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I), \\ & occurs(start(AN), I), \\ & I < I1, \\ & some\_action\_occurred(I1 - 1). \end{aligned} \tag{47}$$

$$\begin{aligned} activity\_component(AN, I1 - I, PAA) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & I < I1, \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I), \\ & occurs(start(AN), I), \\ & occurs(PAA, I1). \end{aligned} \tag{48}$$

$$\begin{aligned} \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I), \\ & activity\_component(AN, K, PAA1), \\ & activity\_component(AN, K, PAA2), \\ & PAA1 \neq PAA2. \end{aligned} \tag{49}$$

$$
\begin{array}{lll}
has\_component(AN, K) & \leftarrow & current\_step(I), \\
& & explanation(N, I), \\
& & no\_activity\_for\_goal(G, I), \\
& & candidate(AN, I), \\
& & occurs(start(AN), I), \\
& & activity\_component(AN, K, C).
\end{array} \tag{50}
$$

$$
\begin{array}{lll}
activity\_length(AN, K) & \leftarrow & current\_step(I), \\
& & explanation(N, I), \\
& & no\_activity\_for\_goal(G, I), \\
& & candidate(AN, I), \\
& & occurs(start(AN), I), \\
& & has\_component(AN, K), \\
& & not\ has\_component(AN, K + 1).
\end{array} \tag{51}
$$

And finally, the intended action:

$$
\begin{array}{lll}
intended\_action(start(AN), I) & \leftarrow & current\_step(I), \\
& & explanation(N, I), \\
& & no\_activity\_for\_goal(G, I), \\
& & candidate(AN, I), \\
& & occurs(start(AN), I), \\
& & projected\_success(AN, I).
\end{array} \tag{52}
$$

## 2.6 Automatic Behaviour

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

$$
\begin{array}{lll}
has\_intention(I) & \leftarrow & intended\_action(AA, I). \\
& \leftarrow & current\_step(I), \\
& & explanation(N, I), \\
& & 0 < I, \\
& & not\ has\_intention(I).
\end{array} \tag{53}
$$