An Intentional Robot - Extending with a Refinement-Based Architecture

Rocio Gomez and Heather Riley

November 8, 2017

1 Adding Non-Determinism to our Model

In this section we describe a preliminary architecture to add non-determinism to an Intentional Agent. This addition is based on the Refinement-Based Architecture presented in [**?**]. The purpose of this work is to have a domain represented in two different granularities (one at coarse resolution, and another one at a fine resolution) and an architecture for an agent to be able to reason at two different levels. At this stage we are only going to focus on creating a fine-grain resolution model of our original domain.

2 Adding Refinement

The fine-resolution description is written in (AL_d). The (AL_d) language is an extension of the (AL) language that allows non-boolean fluents and non-deterministic causal laws. This concept has been introduced in [**?**].

When we look at our initial physical domain (at coarse resolution) we identify three rooms. Looking at the same environment at a fine-grained resolution we observe that each room has four cells in it.

2.1 Statics and Static Relations

These static relations refer to the layout seen in Fig. 1.

Following the steps of the proposed methodology, the static sorts and relations on the

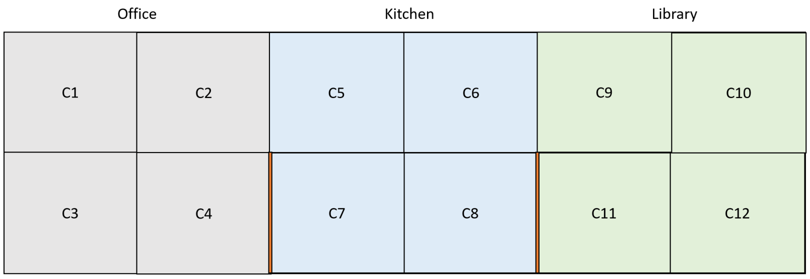

Figure 1: The layout cells in rooms

(AL) that will change or be added are:

$$secure\_room^* = \{library\}.$$
$$room^* = secure\_room^* + \{kitchen, office\}.$$

We add the newly discovered cells in the rooms.

$$cell = \{c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12\}.$$

Another static added to the description is

$$outcome = \{true, false, undet\}.$$

The static relation $next\_to(room, room)$ will adopt the name $next\_to^*(room^*, room^*)$, and we add the new static relations between cells as $next\_to(cell, cell)$. So we have:

$$next\_to^*(library, kitchen).\ next\_to^*(kitchen, office).$$
$$next\_to(c1, c2).\ next\_to(c1, c3).\ next\_to(c2, c4).\ next\_to(c3, c4).$$
$$next\_to(c5, c6).\ next\_to(c5, c7).\ next\_to(c6, c8).\ next\_to(c7, c8).$$
$$next\_to(c9, c10).\ next\_to(c9, c11).\ next\_to(c10, c12).\ next\_to(c11, c12).$$
$$next\_to(c4, c7).\ next\_to(c8, c11).$$

We add a static relation $comp(o_i, o)$ which holds iff object o_i is an newly discovered component of o.

$$comp(c1, office).\ comp(c2, office).\ comp(c3, office).\ comp(c4, office).$$
$$comp(c5, kitchen).\ comp(c6, kitchen).\ comp(c7, kitchen).\ comp(c8, kitchen).$$
$$comp(c9, library).\ comp(c10, library).\ comp(c11, library).\ comp(c12, library).$$

We add a static relation $next\_to\_door(C, R)$ which holds iff cell C is next to the door of room R.

$$next\_do\_door(c8, library).$$
$$next\_to\_door(c11, library).$$

2.2 Fluents

The physical fluents that will be modified are $loc^*(thing, room^*)$ and $locked^*(secure\_room^*)$. They both will become physical defined fluents. We also include new inertial fluents $loc(thing, cell)$. Physical inertial fluent $in\_hand(robot, object)$ as well as all other mental fluents remain the same.

We add two new types of fluents called knowledge inertial fluents:

$$\begin{aligned} knowledge\_inertial\_fluent \ = \\ & can\_be\_tested(my\_agent, physical\_inertial\_fluent) + \\ & directly\_observed(my\_agent, physical\_inertial\_fluent, outcome) + \\ & indirectly\_observed(my\_agent, physical\_defined\_fluent, outcome) \end{aligned}$$

and knowledge defined fluents:

$$\begin{aligned} knowledge\_defined\_fluent \ = \\ & may\_discover(my\_agent, physical\_defined\_fluent) + \\ & observed(my\_agent, physical\_fluent) \end{aligned}$$

The definition of the set of inertial fluents, defined fluents includes the new fluent subsets:

$$\begin{aligned} Inertial\ fluents \ &= physical\ inertial\ fluents + \\ & mental\ inertial\ fluents + \\ & knowledge\ inertial\ fluents. \\ Defined\ fluents \ &= physical\ defined\ fluents + \\ & mental\ defined\ fluents + \\ & knowledge\ defined\ fluents. \end{aligned}$$

2.3 Actions

We modify some physical actions: $move(robot, cell)$ and $exo\_move(object, cell)$.

There is a new action called the knowledge-producing action: $test(robot, physical\_inertial\_fluent, boolean)$

2.4 Axioms

Most axioms are syntactically identical to the axioms in the coarse-resolution model. Those that are changed or added are listed below.

We change four executability conditions related to the constraints of moving to a cell in a locked room, or moving an object to a cell in a locked room.

The original executability conditions were:

$$\begin{array}{rrcl}
\textbf{impossible} & move(rob1, R2) & \textbf{if} & loc(rob1, R1),\ locked(R1). \\
\textbf{impossible} & move(rob1, R) & \textbf{if} & locked(R). \\
\textbf{impossible} & exo_move(O, R) & \textbf{if} & locked(R). \\
\textbf{impossible} & exo_move(O, R2) & \textbf{if} & loc(O, R1), locked(R1).
\end{array}$$

and now we have:

$$\begin{array}{rrcl}
\textbf{impossible} & move(rob1, C2) & \textbf{if} & loc(rob1, C1), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2. \\
\textbf{impossible} & move(ro1, C1) & \textbf{if} & loc(rob1, C2), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2. \\
\textbf{impossible} & exo_move(O, C1) & \textbf{if} & loc(O, C2), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2. \\
\textbf{impossible} & exo_move(O, C2) & \textbf{if} & loc(O, C1), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2.
\end{array}$$

We also add an executability condition to ensure that a room can only be unlocked when a robot is in a cell next to the door.

$$\textbf{impossible} \quad unlock^*(rob1, R) \ \ \textbf{if} \quad loc(rob1, C),\ \neg next\_to\_door(C, R).$$

We also add an axiom that defines fluent $loc^*(thing, room)$ in terms of $loc(thing, cell)$.

$$loc^*(T, R) \quad \textbf{if} \quad loc(T, C),\ comp(C, R).$$

We aslo add a definition of the static relation $next\_to^*$

$$next\_to^*(R1, R2) \quad \textbf{if}\ next\_to^*(C1, C2),\ comp(C1, R1),\ comp(C2, R2).$$

3 AL_d of the fine-resolution version of our model.

Sorts:

$secure\_room^* = \{library\}.$

$room^* = secure\_room^* + \{kitchen, office\}.$

$cell = \{c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12\}.$

$robot = \{rob1\}.$

$book = \{book1, book2\}.$

$object = book.$

$thing = object\ +\ robot.$

$index = \{-1, \ldots, max\_len\}.$

$activity\_name = \{1, \ldots, max\_name\}.$

$boolean = \{true, false\}.$

$outcome = \{true, false, undet\}$

$physical\_inertial\_fluent = loc(thing, cell) +$
$\quad in\_hand(robot, object) +$
$\quad locked^*(secure\_room^*).$

$possible\_goal = my\_goal.$

$physical\_defined\_fluent = possible\_goal + loc^*(thing, room^*).$

$mental\_inertial\_fluent = active\_goal(possible\_goal) +$
$\quad next\_available\_name(activity\_name) +$
$\quad current\_action\_index(activity\_name, index).$

$mental\_defined\_fluent = active\_activity(activity\_name) +$
$\quad next\_action(activity\_name, action) +$
$\quad in\_progress\_activity(activity\_name) +$
$\quad in\_progress\_goal(possible\_goal).$

$knowledge\_inertial\_fluent\ =$
$\quad can\_be\_tested(my\_agent, physical\_inertial\_fluent)\ +$
$\quad directly\_observed(my\_agent, physical\_inertial\_fluent, outcome)\ +$
$\quad indirectly\_observed(my\_agent, physical\_defined\_fluent, outcome).$

$knowledge\_defined\_fluent\ =$
$\quad may\_discover(my\_agent, physical\_defined\_fluent)\ +$
$\quad observed(my\_agent, physical\_fluent).$

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

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

$$
\begin{aligned}
physical\_agent\_action = {} & move(robot, cell) + \\
& pickup(robot, object) + \\
& put\_down(robot, object) + \\
& unlock^*(robot, secute\_room^*). \\
mental\_agent\_action = {} & start(activity\_name) + \\
& stop(activity\_name). \\
agent\_action = {} & mental\_agent\_action + physical\_agent\_action + \{finish\}.
\end{aligned}
$$

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

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

Static relations:

$$next\_to(c1, c2).\ next\_to(c1, c3).\ next\_to(c2, c4).\ next\_to(c3, c4).$$
$$next\_to(c5, c6).\ next\_to(c5, c7).\ next\_to(c6, c8).\ next\_to(c7, c8).$$
$$next\_to(c9, c10).\ next\_to(c9, c11).\ next\_to(c10, c12).\ next\_to(c11, c12).$$
$$next\_to(c4, c7).\ next\_to(c8, c11).$$

$$comp(c1, office).\ comp(c2, office).\ comp(c3, office).\ comp(c4, office).$$
$$comp(c5, kitchen).\ comp(c6, kitchen).\ comp(c7, kitchen).\ comp(c8, kitchen).$$
$$comp(c9, library).\ comp(c10, library).\ comp(c11, library).\ comp(c12, library).$$
$$activity\_component(activity\_name, index, physical\_agent\_action).$$
$$activity\_length(activity\_name, index).$$
$$activity\_goal(activity\_name, possible\_goal).$$

$$next\_to\_door(c8, library).$$
$$next\_to\_door(c11, library).$$

Causal Laws:

$$\begin{array}{rcl}
move(rob1, C) & \textbf{causes} & loc(rob1, C) \\
pickup(rob1, O) & \textbf{causes} & in_hand(rob1, O). \\
put_down(rob1, O) & \textbf{causes} & \neg in_hand(rob1, O). \\
unlock^*(rob1, R) & \textbf{causes} & \neg locked^*(R). \\
exo_lock^*(R) & \textbf{causes} & locked^*(R). \\
exo_move(O, C) & \textbf{causes} & loc(O, C).
\end{array}$$

State Constraints:

$$\begin{array}{rcl}
next_to(R1, R2) & \textbf{if} & next_to(R2, R1). \\
\neg loc(T, R2) & \textbf{if} & loc(T, R1),\ R1 \neq R2. \\
loc(O, R) & \textbf{if} & loc(rob1, R),\ in_hand(rob1, O). \\
\neg in_hand(rob1, O1) & \textbf{if} & in_hand(rob1, O2),\ O1 \neq O2. \\
loc^*(T, R) & \textbf{if} & loc(T, C),\ comp(C, R). \\
next_to^*(R1, R2) & \textbf{if} & next_to^*(C1, C2),\ comp(C1, R1),\ comp(C2, R2).
\end{array}$$

Executability Conditions:

$$\begin{array}{rrcl}
\textbf{impossible} & move(rob1, C) & \textbf{if} & loc(rob1, C). \\
\textbf{impossible} & move(rob1, C2) & \textbf{if} & loc(rob1, C1),\ \neg next_to(C1, C2). \\
\textbf{impossible} & move(rob1, C2) & \textbf{if} & loc(rob1, C1), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2. \\
\textbf{impossible} & move(rob1, C1) & \textbf{if} & loc(rob1, C2), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2. \\
\textbf{impossible} & unlock^*(rob1, R) & \textbf{if} & \neg locked^*(R). \\
\textbf{impossible} & unlock^*(rob1, R1) & \textbf{if} & loc^*(rob1, R2),\ \neg next_to^*(R2, R1),\ R2 \neq R1. \\
\textbf{impossible} & unlock^*(rob1, R) & \textbf{if} & loc(rob1, C),\ \neg next_to_door(C, R). \\
\textbf{impossible} & put_down(rob1, O) & \textbf{if} & \neg in_hand(rob1, O). \\
\textbf{impossible} & pickup(rob1, O1) & \textbf{if} & in_hand(rob1, O2). \\
\textbf{impossible} & pickup(rob1, O) & \textbf{if} & loc(rob1, R1),\ loc(O, R2),\ R1 \neq R2. \\
\textbf{impossible} & exo_move(O, R) & \textbf{if} & loc(O, R) \\
\textbf{impossible} & exo_move(O, C1) & \textbf{if} & loc(O, C2), comp(C1, R1), comp(C2, R2), locked^*(R1), R1! = R2. \\
\textbf{impossible} & exo_move(O, C2) & \textbf{if} & loc(O, C1), comp(C1, R1), comp(C2, R1), locked^*(R1), R1! = R2. \\
\textbf{impossible} & exo_move(O, L) & \textbf{if} & in_hand(rob1, O). \\
\textbf{impossible} & exo_lock^*(R) & \textbf{if} & locked^*(R).
\end{array}$$

Defaults:

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

3.1 ToI Axioms

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:

$$\begin{array}{rcl}
start(AN) & \textbf{causes} & current_action_index(AN, 0). \\
stop(AN) & \textbf{causes} & current_action_index(AN, -1).
\end{array} \tag{1}$$

$$\begin{array}{rcl}
select(G) & \textbf{causes} & active_goal(G). \\
abandon(G) & \textbf{causes} & \neg active_goal(G).
\end{array} \tag{2}$$

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

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

State Constraints:

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

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

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

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

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

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

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

Executability Conditions:

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

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

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

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

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

3.2 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.

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

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.

$$\begin{aligned}
observed_result(AA, I) &\leftarrow current_step(I1), \\
&\quad\; I \leq I1, \\
&\quad\; hpd(AA, B, I). \\
&\leftarrow current_step(I1), \\
&\quad\; I \leq I1, \\
&\quad\; attempt(AA, I), \\
&\quad\; not\ observed_result(AA, I).
\end{aligned} \tag{23}$$

The agent always observes the actions performed by his controller.

$$\begin{aligned}
&\leftarrow current_step(I1), \\
&\quad\; I < I1, \\
&\quad\; occurs(select(G), I), \\
&\quad\; not\ hpd(select(G), true, I). \\
&\leftarrow current_step(I1), \\
&\quad\; I < I1, \\
&\quad\; occurs(abandon(G), I), \\
&\quad\; not\ hpd(abandon(G), true, I).
\end{aligned} \tag{24}$$

3.3 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.

$$\begin{aligned}
occurs(PEA, I) &\stackrel{+}{\leftarrow} current_step(I1), \\
&\quad\; I < I1, \\
&\quad\; explaining(I).
\end{aligned} \tag{25}$$

$$\begin{aligned}
unobserved(PEA, I) &\leftarrow current_step(I1), \\
&\quad\; I < I1, \\
&\quad\; explaining(I), \\
&\quad\; occurs(PEA, I), \\
&\quad\; not\ hpd(PEA, true, I).
\end{aligned} \tag{26}$$

$$\begin{aligned}
number_unobserved(N, I) &\leftarrow current_step(I), \\
&\quad\; N = \#count\{EX : unobserved(EX, IX)\}, \\
&\quad\; explaining(I).
\end{aligned} \tag{27}$$

3.4 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.

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

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 (because it has been reached or because it is a futile goal), the intended next action is to finish.

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

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.

$$
\begin{aligned}
occurs(AA, I1) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& I \leq I1, \\
& active\_goal\_activity(AN, I), \\
& holds(in\_progress\_activity(AN), I1), \\
& holds(next\_action(AN, AA), I1), \\
& not\ impossible(AA, I1).
\end{aligned} \tag{33}
$$

$$
\begin{aligned}
projected\_success(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& I < I1, \\
& holds(active\_activity(AN), I1), \\
& activity\_goal(AN, G), \\
& holds(G, I1).
\end{aligned} \tag{34}
$$

$$
\begin{aligned}
\neg projected\_success(AN, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& not\ projected\_success(AN, I).
\end{aligned} \tag{35}
$$

$$
\begin{aligned}
intended\_action(AA, I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& holds(next\_action(AN, AA), I), \\
& projected\_success(AN, I).
\end{aligned} \tag{36}
$$

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.

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

$$
\begin{aligned}
futile\_activity(AN, I) \quad \stackrel{+}{\leftarrow} \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& \neg projected\_success(AN, I).
\end{aligned} \tag{38}
$$

$$
\begin{aligned}
intended\_action(stop(AN), I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& active\_goal\_activity(AN, I), \\
& futile\_activity(AN, I).
\end{aligned} \tag{39}
$$

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.

$$\begin{aligned} candidate(AN, I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & holds(next\_available\_name(AN), I). \end{aligned} \tag{40}$$

$$\begin{aligned} activity\_goal(AN, G) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I). \end{aligned} \tag{41}$$

Only one activity can start at a time.

$$\begin{aligned} impossible(start(AN), I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & activity\_goal(AN1, G), \\ & occurs(start(AN1), I), \\ & AN \neq AN1. \end{aligned} \tag{42}$$

$$\begin{aligned} occurs(start(AN), I) \quad \leftarrow \quad & current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I), \\ & activity\_goal(AN, G), \\ & not\ impossible(start(AN), I). \end{aligned} \tag{43}$$

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.

$$\begin{aligned} \leftarrow \;& current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & occurs(start(AN), I), \\ & \neg projected\_success(AN, I), \\ & not\ futile\_goal(G, I). \end{aligned} \tag{44}$$

$$\begin{aligned} futile\_goal(G, I) \overset{+}{\leftarrow} \;& current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & occurs(start(AN), I), \\ & \neg projected\_success(AN, I). \end{aligned} \tag{45}$$

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

Auxiliary rule necessary to create candidate activities.

$$\begin{aligned} some\_action\_occurred(I1) \leftarrow \;& current\_step(I), \\ & explanation(N, I), \\ & occurs(A, I1), \\ & I \leq I1. \end{aligned} \tag{47}$$

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.

$$\begin{aligned} occurs(PAA, I1) \overset{+}{\leftarrow} \;& current\_step(I), \\ & explanation(N, I), \\ & no\_activity\_for\_goal(G, I), \\ & candidate(AN, I), \\ & occurs(start(AN), I), \\ & I < I1, \\ & some\_action\_occurred(I1 - 1). \end{aligned} \tag{48}$$

$$
\begin{aligned}
activity\_component(AN, I1 - I, PAA) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& I < I1, \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& occurs(PAA, I1).
\end{aligned} \tag{49}
$$

$$
\begin{aligned}
\leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& activity\_component(AN, K, PAA1), \\
& activity\_component(AN, K, PAA2), \\
& PAA1 \neq PAA2.
\end{aligned} \tag{50}
$$

$$
\begin{aligned}
has\_component(AN, K) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& activity\_component(AN, K, C).
\end{aligned} \tag{51}
$$

$$
\begin{aligned}
activity\_length(AN, K) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& has\_component(AN, K), \\
& not\ has\_component(AN, K + 1).
\end{aligned} \tag{52}
$$

And finally, the intended action:

$$
\begin{aligned}
intended\_action(start(AN), I) \quad \leftarrow \quad & current\_step(I), \\
& explanation(N, I), \\
& no\_activity\_for\_goal(G, I), \\
& candidate(AN, I), \\
& occurs(start(AN), I), \\
& projected\_success(AN, I).
\end{aligned} \tag{53}
$$

3.5 Automatic Behaviour

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

$$\begin{array}{rcl} has\_intention(I) & \leftarrow & intended\_action(AA, I). \\ & \leftarrow & current\_step(I), \\ & & explanation(N, I), \\ & & 0 < I, \\ & & not\ has\_intention(I). \end{array} \tag{54}$$

4 Currently not included

This Theory of Intentions description differs from that written by Justin Blount in that subgoals and sub-activities are not included.