Theory of Activities

May 22, 2018

1 The Physical Domain

The domain has four rooms located side by side ($library$, $kitchen$, $office1$ and $office2$) and connected. The robot rob_1, can move from one room to the next. A *secure* room 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 is in the same location as the object, it can *put_down* an object that it is holding and it can only hold one object at a time. There are two exogenous actions, one that can change the location of any object, and one that can lock a secure room. The agent may not be aware of these exogenous action when they happen. We are also going to define three different defined fluents, two of which we will be using as possible goals.

1.1 AL

Sorts:

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

Static relations:

$$next\_to(office_1, office_2).$$

$$next\_to(kitchen, office_1).$$
$$next\_to(library, kitchen).$$

Inertial fluents:

$$inertial\_fluent = loc(thing, room) + in\_hand(robot, object) + locked(secure\_room).$$

Possible goals:

$$possible\_goal = tidy\_all(room) + tidy\_book(book, room).$$

Defined fluents:

$$defined\_fluent = possible\_goal + missing\_book(room).$$

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:

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

State Constraints:

$$next\_to(L1, L2) \quad \textbf{if} \quad next\_to(L2, L1).$$

$$\neg loc(T, L2) \quad \textbf{if} \quad loc(T, L1),\ L1 \neq L2.$$
$$loc(O, L) \quad \textbf{if} \quad loc(R, L),\ in\_hand(R, O).$$
$$\neg in\_hand(R, O1) \quad \textbf{if} \quad in\_hand(R, O2),\ O1 \neq O2.$$

Executability Conditions:

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

1.2 The Theory of Activities

In our ToA domain of our agent will also have a list of *possible goals* and one of them will be selected. The agent will need to specify one or more *activities* that would achieve the selected goal. He may have a list or different existing activites. If there are existing activities that achieve the goal, the agent will choose and return one of those activites. If the agent cannot find a successfull existing *activity*, he will be inconsistent.

An *activity* will be represented by a duple consisting on *name* and *plan*. A *name* is a unique identifier used to refer to the *activity*, and a plan is a sequence of 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 that may serve as a *goal* are those of the sort *possible_goal*.

In order to create the Action Language for the new domain, we will 1-adapt and 2-extend the original Action Language for the physical domain. We will adapt it by

re-defining the sort $inertial\_fulent$ as $physical\_inertial\_fluens$, $defined\_fluents$ as $physical\_defined\_fluents$, and the sort $agent\_action$ as $physical\_agent\_action$. We will define the following new sorts:

- A sort $activity\_name = 1, \ldots, max\_name$ to represent the name of an $activity$.
- A sort $index = \{-1, 0, max\_len\}$ to represent the index of an action as part of an $activity$.
- A sort $mental\_agent\_action = \{select\_activity(activity\_name)\}$ to represent the action of choosing and activity.

We also the following relations that give shape to the concept of $activity$.

$$activity\_component(activity\_name, index, physical\_agent\_action).$$
$$activity\_goal(activity\_name, possible\_goal).$$

We create a hierarchical structure of $actions$ as follows:

$$agent\_action = mental\_agent\_action + physical\_agent\_action,$$
$$action = agent\_action + exo\_action.$$

As well as the previous statements included in the physical domain, we will include:

Possible goalst:

$$\begin{array}{rll} tidy\_book(B, R) & \textbf{if} & loc(B, R), \neg in\_hand(B). \\ missing\_book(R) & \textbf{if} & \neg tidy\_book(B, R). \\ tidy\_all(R) & \textbf{if} & not\ missing\_book(R). \end{array} \quad (1)$$

2 The Architecture: Reasoning Tasks and Behaviour.

2.1 Introducing new relations

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

- $impossible(A, I)$ means that action A is impossible at step I.

- *candidate*(AN) means that the goal of AN is the selected goal, and therefore a candidate to be the selected Activity.
- *has_component*(AN, K) means that activity AN has a component at index K.
- *success*

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** $p_0, \ldots, p_m$
Tthe ASP contains: $holds(l, I+1)$:- $holds(p_0, I), \ldots, holds(p_m, I), occurs(a, I)$.

For every state constraint: l **if** $p_0, \ldots, p_m$
the ASP contains $holds(l, I)$:- $holds(p_0, I), \ldots, holds(p_m, I)$.

The ASP contains the CWA for defined fluents:
-$holds(F, I)$:- $\#physical\_defined\_fluent(F),\ not\ holds(F, I)$.

For every executability condition: **impossible** a **if** $p_0, \ldots, p_m$
the ASP contains: $impossible(a, I)$:- $holds(p_0, I), \ldots, holds(p_m, I)$.
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 Reasoning Axioms

Defining candidates:

$$\begin{array}{lcl} candidate(AN) & \leftarrow & activity\_goal(AN, G), \\ & & selected\_goal(G). \end{array} \quad (2)$$

Selecting candidates: If it is not impossible to select a candidate, it will be selected.

$$occurs(select\_activity(AN), 0) \leftarrow not\ impossible(select\_activity(AN), 0). \quad (3)$$

It is impossible to select an activity at any step other than 0, it is impossible to select an activity if it is not a candidate, it is not possible to select an activity if the selected goal holds at 0, and it is impossible to select two different activities .

$$\begin{array}{lcl} impossible(select\_activity(AN1), 0) & \leftarrow & occurs(select\_activity(AN2), 0), \\ & & AN1 \neq AN2. \\ impossible(select\_activity(AN), 0) & \leftarrow & not\ candidate(AN). \\ impossible(select\_activity(AN), I) & \leftarrow & 0 < I. \\ impossible(select\_activity(AN), I) & \leftarrow & selected\_goal(G), holds(G, 0). \end{array} \quad (4)$$

This rule ensures that the selected exisiting activity has the minimal sequence to reach the goal among all existing acitvities:

$$\begin{array}{lcl} occurs(PAA, I) & \stackrel{+}{\leftarrow} & occurs(select\_activity(AN), 0), \\ & & activity\_component(AN, I, PAA), \\ & & occurs(A, I-1), \\ & & 0 < I. \end{array} \quad (5)$$

Planning module:

$$\begin{array}{lcl} success & \leftarrow & holds(G, I),\ selected\_goal(G). \\ & \leftarrow & not\ success. \end{array} \quad (6)$$