BUILDING A COMMONSENSE THEORY OF TRAVEL

This is an attempt on the step-by-step explanation of the development of the commonsense theory of travel.

Michael Gelfond, Nov 1, 04

Modeling trips: names and positions.

The basic object of travel module T is that of a TRIP - a short journey over a set route.

We assume that names of trips and there possible stops are given by classes (unary relations)

$$trip(J) \qquad location(L)$$

At any given step the trip may be in transit (en_route) or at one of its possible stops. Its possible positions are given by a class

$$postion = location \cup \{en\_route\}$$

Modeling Trips: itineraries.

Description of a trip j must have its origin and destination given by relations:

$$origin(j, l) \qquad dest(j, l).$$

In addition it may also contain a more detailed itinerary given by a list of statements of the form:

$$leg\_of(j, l_1, l_2)$$

"j's next stop after l_1 is l_2".

Trip's Actions (depart and stop)

There are two main actions which can be "performed" by a trip j:

$$depart(j) \qquad\qquad stop(j, l)$$

"j departs its current location; j stops at l"

If j is a trip with the origin c_0, destination c_2, and an intermediate stop c_1 then departure, a_0, of j will normally trigger actions

$a_1 = stop(j, c_1)$, $a_2 = depart(j)$, $a_3 = stop(j, c_2)$.

The corresponding trajectory is

$\langle c_0, a_0, en\_route, a_1, c_1, a_2, en\_route, a_3, c_2 \rangle$

T_0 - the theory of trip's actions

Smodels notation is used to declare typed variables.

```
#domain location(L;L1;L2;L3).
#domain position(Pos;Pos1;Pos2;Pos3).
#domain trip(J;J1;J2;J3).

#domain action(A;A1;A2;A3).
#domain fluent(Fl;Fl1;Fl2;Fl3).
#domain step(S;S1;S2;S3).
```

Fluents are inertial; *step* **is the set of natural numbers from** 0 **to some** $n-1$ **used to denote steps of the domain histories.**

The language of T_0

T_0 will be used in conjunction with descriptions of trips and their possible positions. T_0's actions and fluents are

```
action(depart(J,L)).     actor(depart(J,L),J).
action(stop(J,L)).       actor(stop(J,L),J).
fluent(at(J,Pos)).
```

T_0 uses relations of LP based action theories:

$$obs(Fl, S),\ hpd(A, S),\ holds(Fl, S),\ occurs(A, S)$$

The first two are used to record (always accurate) observations. The last two, abbreviated by h and o, denote tentative, defeasible conclusions of the agent.

Dynamic causal laws:

After the departure J is en_route:

```
h(at(J,en_route),S+1) :-
            o(depart(J,L),S).
```

Stops are successful:

```
h(at(J,L),S+1) :-
            o(stop(J,L),S).
```

Observation of a stop not mentioned in the detailed itinerary is recorded by:

```
emergency_stop(J,S+1) :- neq(L1,L),
            leg_of(J,L1,L2),
            o(depart(J,L1),S),
            hpd(stop(J,L),S+1).
```

Defeasible triggers:

Default 1: unless otherwise specified we assume that an *en_raute* trip goes directly to its destination.

```
o(stop(J,L),S) :-
          h(at(J,en_route),S),
          dest(J,L),
          not ab(1,J,S),
          not -o(stop(J,L),S).
```

The default is used in the absence of a detailed itinerary and information about emergency stops.

Defeasible triggers:

Default 2: normally, a trip stops when required by its itinerary.

```
o(stop(J,L2),S+1) :-
              leg_of(J,L1,L2),
              o(depart(J,L1),S),
              not emergency_stop(J,S+1),
              not -o(stop(J,L),S+1).
```

Default 2 overrides default 1.

```
ab(1,J,S+1) :-
      leg_of(J,L1,L2),
      o(depart(J,L1),S).
```

Defeasible triggers:

Default 3: after a stop the trip normally continues to its destination:

```
o(depart(J,L),S+1) :-
            o(stop(J,L),S),
            not dest(J,L),
            not emergency_stop(J,S),
            not -o(depart(J,L),S+1).
```

Note that in the presence of emergency the default is not applicable. In our formalization the trip will stay at the place of emergency stop until its departure is explicitly specified.

Executability Conditions

```
-o(depart(J,L),S) :-
          h(at(J,en_route),S).
-o(stop(J,L),S) :-
          -h(at(J,en_route),S).
```

State Constraints

A trip can only stop at one place at a time

```
-o(stop(J,L),S) :-
           o(stop(J,L1),S),
           neq(L,L1).
```

We also need to say that position of an object is unique:

```
object(J).
#domain object(O;O1;O2;O3).
-h(at(O,Pos1),S) :-
           h(at(O,Pos2),S),
           neq(Pos1,Pos2).
```

Domain Independent Axioms

1. Inertia:

```
h(Fl,S+1) :-
      h(Fl,S),
      not -h(Fl,S+1).
-h(Fl,S+1) :-
      -h(Fl,S),
      not h(Fl,S+1).
```

2. Happened-occur connection:

```
o(A,S) :-
       hpd(A,S).
```

Computing the trip's trajectories

T_0 can be viewed as a function which takes as an input $X = D \cup H$ where

(a) D is a collection of atoms defining trips, their positions and itineraries referred to as a trip domain;

(b) H is a history of the domain,

and returns the domain trajectory $T_0(X)$ extracted from the answer set W of $T_0 \cup X$.

Displaying the trajectories.

In the language of Smodels this can be done by the following display rules and directives.

```
at(O,Pos,S) :-
        h(at(O,Pos),S).
do(A,S) :-
        o(A,S).
hide.
show do(A,B), at(A,B,C).
```

Example: domain D

Consider two trips defined as:

```
trip(j1).            trip(j2).
```

with possible positions defined by the program:

```
city(rome).
city(baghdad).          location(X) :- city(X).
city(boston).
...
position(L).
position(en_route).
```

Note that the last two axioms are part of every travel domain.

Example: domain D

Domain D is obtained by expending the above axioms by statements:

```
origin(J,boston).    dest(J,baghdad).
```

and the itinerary

```
leg_of(j2,boston,rome).
leg_of(j2,rome,berlin).
leg_of(j2,berlin,baghdad).
```

for $j2$.

Example: histories H_1 and H_2

Let H_1 **be**

```
hpd(depart(J,boston),0).    h(at(J,boston),0).
```

H_2 **be**

```
hpd(depart(J,boston),0).    h(at(J,boston),0).
hpd(stop(J,paris),1).
```

and $X_1 = D \cup H_1$ **and** $X_2 = D \cup H2$

Example: trajectory of j_1 in X_1

The trajectory of j_1 in X_1 is computed by setting n in the definition of *step* to 6 and finding the answer set of $T_0 \cup X_1$. It is

```
at(j1,boston,0)       do(depart(j1,boston),0),
at(j1,en_route,1)     do(stop(j1,baghdad),1)
at(j1,baghdad,2)
```

Since X_1 contains neither the itinerary for j_1 nor information about its other stops we conclude that the trip goes directly to Baghdad.

Example: trajectory of j_2 in X_1

Similarly for j_2

```
at(j2,boston,0)         do(depart(j2,boston),0)
at(j2,en_route,1)       do(stop(j2,rome),1)
at(j2,rome,2)           do(depart(j2,rome),2)
at(j2,en_route,3)       do(stop(j2,berlin),3)
at(j2,berlin,4)         do(depart(j2,berlin),4)
at(j2,en_route,5)       do(stop(j2,baghdad),5)
at(j2,baghdad,6)
```

j_2 goes to Baghdad making all the required intermediate stops.

Example: trajectory of j_1 in X_2

```
at(j1,boston,0)         do(depart(j1,boston),0)
at(j1,en_route,1)       do(stop(j1,paris),1)
at(j1,paris,2)          do(depart(j1,paris),2)
at(j1,en_route,3)       do(stop(j1,baghdad),3)
at(j1,baghdad,4)
```

As stated in its history the trip visits Paris, and than continues to its final destination, Baghdad.

Example: trajectory of j_2 in X_2

```
at(j2,boston,0)          do(depart(j2,boston),0)
at(j2,en_route,1)        do(stop(j2,paris),1)
at(j2,paris,2)
```

The trip makes an unplanned emergency stop in Paris. Until otherwise informed we assume that it stays there. If the history of j_2 is expended by

$hpd(depart(j2, paris), 2)$

the trip will proceed to Baghdad.

Features to Discuss

Even though construction of T_0 is influenced by the work on action theories it has several non-standard features:

1. Defeasible triggers which are implemented via prioritized defaults.

2. Encoding of executability conditions by statements with -$o(A, S)$ in the head instead of constraints. Ability to infer non-occurrence of actions allows to defeat the defaults.

3. Distinguishing between *hpd* and *occur* and the use of *hpd* in defining non-inertial fluent *emergency_stop*.

Another important influence is the methodology of building knowledge bases by lp-functions.

Modeling the trip's participants

To model travel by individuals we expand T_0 by adding a new variable declaration

```
#domain person(P;P1;P2;P3)
```

additional actions and fluents

```
action(embark(P,J))      actor(embark(P,J),P)
action(disembark(P,J))   actor(disembark(P,J),P)
action(go_on(P,J))       actor(go_on(P,J),P)
fluent(participant(P,J))
fluent(at(P,Pos))
```

and additional objects

```
object(P)
```

Direct Effects

P **joins the trip** J**:**

```
  h(participant(P,J),S+1) :-
                    o(embark(P,J),S).
```

P **ends the trip** J**:**

```
-h(participant(P,J),S+1) :-
            o(disembark(P,J),S).
```

Triggers

go_on **is a sequence of two simpler actions:**

```
o(embark(P,J),S) :-
            o(go_on(P,J),S).
o(depart(J,L),S+1) :-
            h(at(J,L),S+1),
            o(go_on(P,J),S).
```

Participants leave the trip at the destination.

```
o(disembark(P,J),S+1) :-
           h(participant(P,J),S),
           o(stop(J,L),S),
           dest(J,L).
```

State Constraints

Participants share the current location of their trip

```
h(at(P,Pos),S) :-
           h(participant(P,J),S),
           h(at(J,Pos),S).
```

Executability Conditions

Can't embark (disembark) if already done so:

```
-o(embark(P,J),S) :-
               h(participant(P,J),S).
-o(disembark(P,J),S),
               -h(participant(P,J),S).
```

Need to be at the right place:

```
-o(embark(P,J),S) :-
               h(at(P,L),S),
               -h(at(J,L),S).
-o(embark(P,J),S),
               h(at(J,en_route),S).
-o(disembark(P,J),S),
               h(at(J,en_route),S).
```

Features to Discuss

Testing the program (use travelers) on several histories in traveler1,..., traveler3, produces correct results. But how can we demonstrate the program correctness for reasonable histories?

1. Because of the prioritized defaults and the use of *hpd* our program probably does not fit into any action language. If it is true is it worth developing such a language?

2. Suppose we have proven correctness of "theory of trips". Can we have a reasonably general result to show that extension of "trip" by the travelers information is conservative?