Artificial Intelligence

Turing machines from the perspective of STRIPS and the situation calculus

A snapshot (ID) \([\text{Left, Right, Q}]\) can be construed as a state, on which a Turing machine can perform one of several actions (move right, move left, write)

\[
[\text{Left, Right, Q}] \xrightarrow{\text{action}} [\text{Left', Right', Q'}]
\] (1)

To bring out the locality of these actions (i.e. the bounded effects of any one such action), let us describe a snapshot in a manner that decouples the head from the tape, specifying tape contents independently from the head position.

1 Describing a snapshot

That is, rather than describing a snapshot \([\text{Left, Right, Q}]\) by a single ternary predicate \(\text{id(Left, Right, Q)}\) (with lists as objects/arguments), let us introduce predicates \(\text{intState(Q)}, \text{headPos(Loc)}, \text{at(Symbol,Loc)}\) so that an initial snapshot \(([[],[a0,a1,...,an], q0])\) is described by the facts

\[
\text{intState(q0)}.
\text{headPos(0)}.
\text{at(a0,0)}.
\text{at(a1,right(0)).}
\cdot
\cdot
\text{at(an,right(right(...(0))))}.
\]

where a location \(\text{Loc}\) is one of
\(0, \text{right(0)}, \text{right(right(0))}, ..., \text{left(0)}, \text{left(left(0))}, ...\)

2 Move Right — a first stab

Recall that STRIPS characterizes an action by Precondition, Delete and Add lists. Let us take the case of the action move right.

Precondition list:

\[ [ \text{mr(Q,X,Qnew)}, \text{intState(Q)}, \text{at(X,Loc)}, \text{headPos(Loc)} ] \]

Delete list:

\[ [ \text{intState(Q)}, \text{headPos(Loc)} ] \]

Add list:

\[ [ \text{intState(Qnew)}, \text{headPos(right(Loc))} ] \]

Two complications:

(a) blanks
(b) unnormalized terms such as \(\text{right(left(0))}\) (instead of 0).
3  Move Right — a second stab

Fix (a) and (b) via the predicates

$$\text{derAt}(X, \text{Loc}) :- \text{at}(X, \text{Loc}),!.$$  
$$\text{derAt}(b-k, \text{Loc}).$$

$$\text{normalizeRight}(\text{left}(L),L) :- !.$$  
$$\text{normalizeRight}(L,\text{right}(L)).$$

and similarly for \text{normalizeLeft}.

Revised Precondition list:

\[
\begin{align*}
\text{mr}(Q,X,Q_{\text{new}}), & \quad \text{intState}(Q), \quad \text{derAt}(X, \text{Loc}), \quad \text{headPos}(\text{Loc}), \\
& \quad \text{normalizeRight}(\text{Loc},L_{\text{new}}) 
\end{align*}
\]

Revised Add list:

\[
\begin{align*}
\text{intState}(Q_{\text{new}}), & \quad \text{headPos}(L_{\text{new}}) 
\end{align*}
\]

4  The predicates listed

4.1  Static

Missing from (1) are

\[
\text{mr}(Q,X,Q_{\text{new}}), \quad \text{ml}(Q,X,Q_{\text{new}}), \quad \text{wl}(Q,X,Y,Q_{\text{new}})
\]

as well as the help functions [sic]

\[
\text{normalizeRight}(L,L_{\text{right}}), \quad \text{normalizeLeft}(L,L_{\text{left}})
\]

for avoiding mixed terms \text{right(left}(0)), \text{left(right}(0)), etc.

4.2  Dynamic

4.2.1  Primitive

\[
\text{intState}(Q), \quad \text{headPos}(\text{Loc}), \quad \text{at}(\text{Symbol},\text{Loc})
\]

4.2.2  Derived

\[
\text{derAt}(X,\text{Loc})
\]

5  Problems: STRIPS

1. Write out the clauses for \text{normalizeLeft}.
2. Write out the Precondition, Delete and Add lists for move left.
3. Write out the Precondition, Delete and Add lists for write.
6 Applying the Situation Calculus

Relativize all dynamic predicates to a state via predicates

\[ \text{holdPrim(PrimAtom, Situation)}, \text{ holds(Atom, Situation)}. \]

6.1 The initial ID with blank input

\[ \text{holdPrim(intState(q0), init)}. \]
\[ \text{holdPrim(headPos(0), init)}. \]

The absence of facts of the form \( \text{holdPrim(at(X,Loc), init)} \) amounts to the assumption that the input is blank.

6.2 Encoding the Precondition, Delete and Add lists

Encode Precondition, Delete and Add lists by predicates \( \text{poss(Action, Situation)} \), \( \text{delL(Action, Situation, PrimAtom)} \) and \( \text{addL(Action, Situation, PrimAtom)} \).

E.g. for Action = moveRight,

\[ \text{poss(moveRight, Situation) :-} \]
\[ \text{mr(Q,X,Qnew)}, \]
\[ \text{holdPrim(intState(Q), Situation)}, \]
\[ \text{holds(derAt(X,Loc), Situation)}, \]
\[ \text{holdPrim(headPos(Loc), Situation)}. \]
\[ \% \text{ can drop: normalizeRight(Loc,Lnew)}. \]

\[ \text{delL(moveRight, Situation, intState(Q)) :-} \]
\[ \text{holdPrim(intState(Q), Situation)}. \% \text{ can drop this} \]

\[ \text{delL(moveRight, Situation, headPos(Loc)) :-} \]
\[ \text{holdPrim(headPos(Loc), Situation)}. \% \text{ can drop this} \]

\[ \text{addL(moveRight, Situation, intState(Q)) :-} \]
\[ \text{mr(Q0,X,Q)}, \]
\[ \text{holdPrim(intState(Q0), Situation)}, \]
\[ \text{holds(derAt(X,Loc), Situation)}, \]
\[ \text{holdPrim(headPos(Loc), Situation)}. \]

\[ \text{addL(moveRight, Situation, headPos(Loc)) :-} \]
\[ \text{holdPrim(headPos(L0), Situation)}, \]
\[ \text{normalizeRight(L0,Loc)}. \]

6.3 The fate of primitive predicates (beyond init)

Encode the effects of Delete via
holdPrim(PrimAtom, do(Action, Situation)) :-
    poss(Action, Situation),
    holdPrim(PrimAtom, Situation),
    not(delL(Action, Situation, PrimAtom)).

Encode the effects of Add via

holdPrim(PrimAtom, do(Action, Situation)) :-
    poss(Action, Situation),
    addL(Action, Situation, PrimAtom).

**Question.** What happens if a PrimAtom is in both the Delete and Add lists of an action?

### 6.4 The derived predicates

\[ \text{holds(derAt(X, Loc), Situation)} :\! \]
\[ \text{holdPrim(at(X, Loc), Situation)}, !. \]

\[ \text{holds(derAt(b-k, Loc), Situation).} \]

### 7 Problems: Situation Calculus

4. Write out the corresponding clauses for move left.
5. Write out the corresponding clauses for write.
6. Try out your answers, with the additional knowledge base

\[ \text{holdPrim(at(0,0),init).} \]
\[ \text{holdPrim(at(1,right(0)),init).} \]
\[ \text{mr(q0,0,q1).} \]
\[ \text{ml(q1,1,q2).} \]
\[ \text{wl(q2,0,1,q0).} \]

and the queries

\[ ?- \text{holdPrim(at(X,0),init).} \]
\[ X = 0 \]

\[ ?- \text{holdPrim(headPos(Loc),do(moveRight,init)).} \]
\[ \text{Loc} = \text{right}(0) \]
?- holdPrim(at(X,0), do(moveRight,init)).
X = 0

?- holdPrim(headPos(Loc), do(moveLeft, do(moveRight,init))).
Loc = 0

?- holdPrim(intState(Q), do(moveLeft, do(moveRight,init))).
Q = q2

?- holdPrim(at(X,Loc), do(write, do(moveLeft, do(moveRight,init)))).
X = 1, Loc = right(0) ;
X = 1, Loc = 0 ? ;
no
Prolog Code

normalizeRight(left(L),L) :- !.
normalizeRight(L,right(L)).

holdPrim(intState(q0), init).
holdPrim(headPos(0), init).

poss(moveRight, Situation) :-
  mr(Q,X,Qnew),
  holdPrim(intState(Q), Situation),
  holds(derAt(X,Loc), Situation),
  holdPrim(headPos(Loc), Situation).

delL(moveRight, Situation, intState(Q)) :-
  holdPrim(intState(Q), Situation).

delL(moveRight, Situation, headPos(Loc)) :-
  holdPrim(headPos(Loc), Situation).

addL(moveRight, Situation, intState(Q)) :-
  mr(Q0,X,Q),
  holdPrim(intState(Q0), Situation),
  holds(derAt(X,Loc), Situation),
  holdPrim(headPos(Loc), Situation).

addL(moveRight, Situation, headPos(Loc)) :-
  holdPrim(headPos(L0), Situation),
  normalizeRight(L0,Loc).

holdPrim(PrimAtom, do(Action,Situation)) :-
  poss(Action,Situation),
  holdPrim(PrimAtom,Situation),
  not(delL(Action,Situation,PrimAtom)).

holdPrim(PrimAtom, do(Action,Situation)) :-
  poss(Action,Situation),
  addL(Action,Situation,PrimAtom).

holds(derAt(X,Loc), Situation) :-
  holdPrim(at(X,Loc),Situation),!.

holds(derAt(b-k,Loc), Situation).

not(P) :- P,!; fail ; true.