Situation Calculus

- State-based representation where the states are denoted by terms.
- A situation is a term that dentotes a state.
- There are two ways to refer to states:
  - *init* denotes the initial state
  - $do(A, S)$ denotes the state resulting from doing action A in state S, if it is possible to do A in S.
- A situation also encodes how to get to the state it denotes.

Example States

- $init$
- $do(move(rob, o109, o103), init)$
- $do(move(rob, o103, mail),$
  $do(move(rob, o109, o103),$
  $init)).$
- $do(pickup(rob, k1),$
  $do(move(rob, o103, mail),$
  $do(move(rob, o109, o103),$
  $init))).$

Using the Situation Terms

- Add an extra term to each dynamic predicate indicating the situation.
- Example Atoms:

$$at(rob, o109, init)$$

$$at(rob, o103, do(move(rob, o109, o103), init))$$

$$at(k1, mail, do(move(rob, o109, o103), init))$$

Axiomatizing using the Situation Calculus

- You specify what is true in the initial state using axioms with *init* as the situation parameter.
- Primitive relations are axiomatized by specifying what is true in situation $do(A, S)$ in terms of what holds in situation S.
- Derived relations are defined using clauses with a free variable in the situation argument.
- Static relations are defined without reference to the situation.

Initial Situation

$sitting\_at(rob, o109, init).$

$sitting\_at(parcel, storage, init).$

$sitting\_at(k1, mail, init).$

Derived Relations

$adjacent(P_1, P_2, S) \leftarrow$

$\quad between(Door, P_1, P_2) \wedge$

$\quad unlocked(Door, S).$

$adjacent(lab2, o109, S).$

$\ldots$

When are actions possible?

$poss(A, S)$ is true if action A is possible in state S.

$$poss(putdown(Ag, Obj), S) \leftarrow$$
$$carrying(Ag, Obj, S).$$

$$poss(move(Ag, Pos_1, Pos_2), S) \leftarrow$$
$$autonomous(Ag) \land$$
$$adjacent(Pos_1, Pos_2, S) \land$$
$$sitting\_at(Ag, Pos_1, S).$$

Axiomatizing Primitive Relations

Example: Unlocking the door makes the door unlocked:

$$unlocked(Door, do(unlock(Ag, Door), S)) \leftarrow$$
$$poss(unlock(Ag, Door), S).$$

Frame Axiom: No actions lock the door:

$$unlocked(Door, do(A, S)) \leftarrow$$
$$unlocked(Door, S) \wedge$$
$$poss(A, S).$$

Example: axiomatizing *carried*

Picking up an object causes it to be carried:

$$carrying(Ag, Obj, do(pickup(Ag, Obj), S)) \leftarrow$$
$$poss(pickup(Ag, Obj), S).$$

Frame Axiom: The object is being carried if it was being carried before unless the action was to put down the object:

$$carrying(Ag, Obj, do(A, S)) \leftarrow$$
$$carrying(Ag, Obj, S) \wedge$$
$$poss(A, S) \wedge$$
$$A \neq putdown(Ag, Obj).$$

More General Frame Axioms

The only actions that undo *sitting_at* for object *Obj* is when *Obj* moves somewhere or when someone is picking up *Obj*.

$$sitting\_at(Obj, Pos, do(A, S)) \leftarrow$$
$$poss(A, S) \wedge$$
$$sitting\_at(Obj, Pos, S) \wedge$$
$$\forall Pos_1 \;\; A \neq move(Obj, Pos, Pos_1) \wedge$$
$$\forall Ag \;\; A \neq pickup(Ag, Obj).$$

The last line is equivalent to:

$$\sim \exists Ag \; A = pickup(Ag, Obj)$$

which can be implemented as

$$sitting\_at(Obj, Pos, do(A, S)) \leftarrow$$
$$\cdots \wedge \cdots \wedge \cdots \wedge$$
$$\sim is\_pickup\_action(A, Obj).$$

with the clause:

$$is\_pickup\_action(A, Obj) \leftarrow$$
$$A = pickup(Ag, Obj).$$

which is equivalent to:

$$is\_pickup\_action(pickup(Ag, Obj), Obj).$$

STRIPS and the Situation Calculus

- Anything that can be stated in STRIPS can be stated in the situation calculus.
- The situation calculus is more powerful. For example, the "drop everything" action.
- To axiomatize STRIPS in the situation calculus, we can use $holds(C, S)$ to mean that C is true in situation S.

$holds(C, do(A, W)) \leftarrow$	
$\quad preconditions(A, P) \wedge$	The preconditions of
$\quad holdsall(P, W) \wedge$	of A all hold in W.
$\quad add\_list(A, AL) \wedge$	C is on the
$\quad member(C, AL).$	addlist of A.
$holds(C, do(A, W)) \leftarrow$	
$\quad preconditions(A, P) \wedge$	The preconditions of
$\quad holdsall(P, W) \wedge$	of A all hold in W.
$\quad delete\_list(A, DL) \wedge$	C isn't on the
$\quad notin(C, DL) \wedge$	deletelist of A.
$\quad holds(C, W).$	C held before A.