Relational Actions and Planning

- Agents reason in time.
- Agents reason about time.

Time passes as an agent acts and reasons.
Given a goal, it is useful for an agent to think about what it will do in the future to determine what it will do now.

Representing Time

Time can be modeled in a number of ways:

Discrete time Time is modeled as jumping from one time point to another.

Continuous time Time is modeled as being dense.

Event-based time Time steps don't have to be uniform; time steps can be between interesting events.

State space Instead of considering time explicitly, actions can map from one state to another.

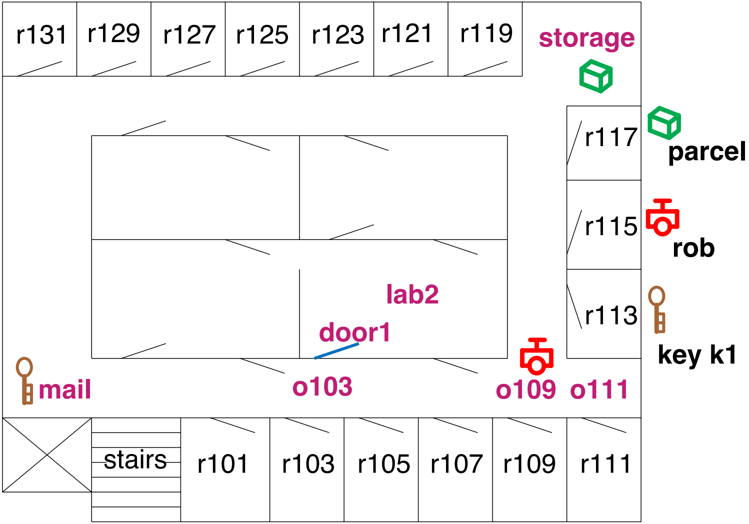
You can model time in terms of **points** or **intervals.**

Time and Relations

When modeling relations, you distinguish two basic types:

- **Static relations** are those relations whose value does not depend on time.
- **Dynamic relations** are relations whose truth values depends on time. Either
  - **derived relations** whose definition can be derived from other relations for each time,
  - **primitive relations** whose truth value can be determined by considering previous times.

The Delivery Robot World



Modeling the Delivery Robot World

- Individuals: rooms, doors, keys, parcels, and the robot.
- Actions:
  - move from room to room
  - pick up and put down keys and packages
  - unlock doors (with the appropriate keys)
- Relations: represent
  - the robot's position
  - the position of packages and keys and locked doors
  - what the robot is holding

Example Relations

- $at(Obj, Loc)$ is true in a situation if object *Obj* is at location *Loc* in the situation.
- $carrying(Ag, Obj)$ is true in a situation if agent *Ag* is carrying *Obj* in that situation.
- $sitting\_at(Obj, Loc)$ is true in a situation if object *Obj* is sitting on the ground (not being carried) at location *Loc* in the situation.
- $unlocked(Door)$ is true in a situation if door *Door* is unlocked in the situation.
- $autonomous(Ag)$ is true if agent *Ag* can move autonomously. This is static.

Example Relations (cont.)

- $opens(Key, Door)$ is true if key Key opens door $Door$. This is static.
- $adjacent(Pos_1, Pos_2)$ is true if position Pos_1 is adjacent to position Pos_2 so that the robot can move from Pos_1 to Pos_2 in one step.
- $between(Door, Pos_1, Pos_2)$ is true if $Door$ is between position Pos_1 and position Pos_2. If the door is unlocked, the two positions are adjacent.

Actions

- $move(Ag, From, To)$ agent Ag moves from location $From$ to adjacent location To. The agent must be sitting at location $From$.
- $pickup(Ag, Obj)$ agent Ag picks up Obj. The agent must be at the location that Obj is sitting.
- $putdown(Ag, Obj)$ the agent Ag puts down Obj. It must be holding Obj.
- $unlock(Ag, Door)$ agent Ag unlocks $Door$. It must be outside the door and carrying the key to the door.

Static Facts

$$sitting\_at(rob, o109).$$
$$sitting\_at(parcel, storage).$$
$$sitting\_at(k1, mail).$$

$$between(door1, o103, lab2).$$
$$opens(k1, door1).$$
$$autonomous(rob).$$

Derived Relations

$$at(Obj, Pos) \leftarrow sitting\_at(Obj, Pos).$$
$$at(Obj, Pos) \leftarrow carrying(Ag, Obj) \wedge at(Ag, Pos).$$
$$adjacent(o109, o103).$$
$$adjacent(o103, o109).$$
$$\ldots$$
$$adjacent(lab2, o109).$$
$$adjacent(P_1, P_2) \leftarrow$$
$$between(Door, P_1, P_2) \wedge$$
$$unlocked(Door).$$

Situation Calculus

- State-based representation where the states are denoted by terms.
- A **situation** is a term that denotes 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 encodes how to get to the state it denotes.
  - A state may be represented by multiple situations.
  - A state may be represented by no situations if it is unreachable.
  - A situation may represent no states, if an action was not possible.

Example Situations

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

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.

Derived Relations

$sitting\_at(rob, o109, init).$
$sitting\_at(parcel, storage, init).$
$sitting\_at(k1, mail, init).$

$$
\begin{aligned}
& adjacent(P_1, P_2, S) \leftarrow \\
& \qquad between(Door, P_1, P_2) \wedge \\
& \qquad unlocked(Door, S). \\
& adjacent(lab2, o109, S). \\
& \ldots
\end{aligned}
$$

When are actions possible?

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

$$
\begin{aligned}
& poss(putdown(Ag, Obj), S) \leftarrow \\
& \quad carrying(Ag, Obj, S).
\end{aligned}
$$

$$
\begin{aligned}
& poss(move(Ag, Pos_1, Pos_2), S) \leftarrow \\
& \quad autonomous(Ag) \land \\
& \quad adjacent(Pos_1, Pos_2, S) \land \\
& \quad sitting\_at(Ag, Pos_1, S).
\end{aligned}
$$

Axiomatizing Primitive Relations

Example: Unlocking the door makes the door unlocked:

$$
\begin{aligned}
&\mathit{unlocked}(\mathit{Door}, \mathit{do}(\mathit{unlock}(\mathit{Ag}, \mathit{Door}), S)) \leftarrow \\
&\qquad \mathit{poss}(\mathit{unlock}(\mathit{Ag}, \mathit{Door}), S).
\end{aligned}
$$

Frame Axiom: No actions lock the door:

$$
\begin{aligned}
&\mathit{unlocked}(\mathit{Door}, \mathit{do}(A, S)) \leftarrow \\
&\qquad \mathit{unlocked}(\mathit{Door}, S) \wedge \\
&\qquad \mathit{poss}(A, S).
\end{aligned}
$$

Example: axiomatizing *carried*

Picking up an object causes it to be carried:

$$\begin{aligned}&carrying(Ag, Obj, do(pickup(Ag, Obj), S)) \leftarrow\\ &\quad poss(pickup(Ag, Obj), S).\end{aligned}$$

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

$$\begin{aligned}&carrying(Ag, Obj, do(A, S)) \leftarrow\\ &\quad carrying(Ag, Obj, S) \wedge\\ &\quad poss(A, S) \wedge\\ &\quad A \neq putdown(Ag, Obj).\end{aligned}$$

Example: $sitting\_at$

An object is sitting at a location if:

- it moved to that location:

$$sitting\_at(Obj, Pos, do(move(Obj, Pos_0, Pos), S)) \leftarrow$$
$$poss(move(Obj, Pos_0, Pos).$$

- it was put down at that location:

$$sitting\_at(Obj, Pos, do(putdown(Ag, Obj), S)) \leftarrow$$
$$poss(putdown(Ag, Obj), S) \wedge$$
$$at(Ag, Pos, S).$$

- it was at that location before and didn't move and wasn't picked up.

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*.

$$
\begin{aligned}
& sitting\_at(Obj, Pos, do(A, S)) \leftarrow \\
& \quad poss(A, S) \land \\
& \quad sitting\_at(Obj, Pos, S) \land \\
& \quad \forall Pos_1 \;\; A \neq move(Obj, Pos, Pos_1) \land \\
& \quad \forall Ag \;\; A \neq pickup(Ag, Obj).
\end{aligned}
$$

The last line is equivalent to:

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

which can be implemented as

$$
\begin{aligned}
& sitting\_at(Obj, Pos, do(A, S)) \leftarrow \\
& \quad \cdots \land \cdots \land \cdots \land \\
& \quad \sim is\_pickup\_action(A, Obj).
\end{aligned}
$$

with the clause:

Planning

Given

- an initial world description
- a description of available actions
- a goal

a **plan** is a sequence of actions that will achieve the goal.

Example Planning

If you want a plan to achieve Rob holding the key $k1$ and being at $o103$, the query

$$ask\ carrying(rob, k1, S) \wedge at(rob, o103, S).$$

has an answer

$$\begin{aligned}S = do(&move(rob, mail, o103),\\ &do(pickup(rob, k1),\\ &\quad do(move(rob, o103, mail),\\ &\quad\quad do(move(rob, o109, o103), init)))).\end{aligned}$$

Planning as Resolution

- Idea: backward chain on the situation calculus rules.
- A complete search strategy (e.g., A^* or iterative deepening) is guaranteed to find a solution.
- When there is a solution to the query with situation $S = do(A, S_1)$, action A is the last action in the plan.
- You can virtually always use a frame axiom so that the search space is largely unconstrained by the goal. Search space is enormous.

Goal-directed searching

- Given a goal, you would like to consider only those actions that actually achieve it.
- Example:

$$ask\ \ carrying(rob, parcel, S) \wedge in(rob, lab2, S).$$

the last action needed is irrelevant to the left subgoal.

- So we need to combine the planning algorithms with the relational representations.