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**.
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 depend 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

Diagram showing a delivery robot world with labeled sections:
- Stairs
- Mail
- Storage
- Lab 2
- Door 1
- Parcel
- Robot
- Key k1

Legend:
- r101, r103, r105, r107, r109, r111
- o103, o109, o111

D. Poole and A. Mackworth 2010
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

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

- \texttt{opens(Key, Door)} is true if key \textit{Key} opens door \textit{Door}. This is static.
- \texttt{adjacent(Pos}_1, Pos_2) is true if position \textit{Pos}_1 is adjacent to position \textit{Pos}_2 so that the robot can move from \textit{Pos}_1 to \textit{Pos}_2 in one step.
- \texttt{between(Door, Pos}_1, Pos_2) is true if \textit{Door} is between position \textit{Pos}_1 and position \textit{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.
sitting_at(rob, o109).
sitting_at(parcel, storage).
sitting_at(k1, mail).

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

\[ at(Obj, Pos) \leftarrow \text{sitting}_\text{at}(Obj, Pos). \]
\[ at(Obj, Pos) \leftarrow \text{carrying}(Ag, Obj) \land at(Ag, Pos). \]
\[ \text{adjacent}(o109, o103). \]
\[ \text{adjacent}(o103, o109). \]
\[ \ldots \]
\[ \text{adjacent}(lab2, o109). \]
\[ \text{adjacent}(P_1, P_2) \leftarrow \]
\[ \text{between}(Door, P_1, P_2) \land \]
\[ \text{unlocked}(Door). \]
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

- \textit{init}
- \texttt{do(move(rob, o109, o103), init)}
- \texttt{do(move(rob, o103, mail), do(move(rob, o109, o103), init)).}
- \texttt{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:**

\[
\text{at}(\text{rob}, \text{o109}, \text{init})
\]

\[
\text{at}(\text{rob}, \text{o103}, \text{do}(\text{move}(\text{rob}, \text{o109}, \text{o103}), \text{init}))
\]

\[
\text{at}(\text{k1}, \text{mail}, \text{do}(\text{move}(\text{rob}, \text{o109}, \text{o103}), \text{init}))
\]
You specify what is true in the initial state using axioms with \textit{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

\[
\begin{align*}
sitting \_at(rob, o109, init). \\
sitting \_at(parcel, storage, init). \\
sitting \_at(k1, mail, init).
\end{align*}
\]

adjacent(\(P_1, P_2, S\)) \leftarrow
\begin{align*}
&\text{between(Door, } P_1, P_2) \land \\
&\text{unlocked(Door, } S). \\
\end{align*}
adjacent(lab2, o109, S).
\ldots
When are actions possible?

$\text{poss}(A, S)$ is true if action $A$ is possible in situation $S$.

\[ \text{poss}(\text{putdown}(Ag, Obj), S) \leftarrow \text{carrying}(Ag, Obj, S). \]

\[ \text{poss}(\text{move}(Ag, Pos_1, Pos_2), S) \leftarrow \text{autonomous}(Ag) \land \text{adjacent}(Pos_1, Pos_2, S) \land \text{sitting\_at}(Ag, Pos_1, S). \]
Axiomatizing Primitive Relations

**Example:** Unlocking the door makes the door unlocked:

\[
\text{unlocked}(\text{Door}, \text{do}(\text{unlock}(\text{Ag}, \text{Door}), S)) \leftarrow \\
\text{poss}(\text{unlock}(\text{Ag}, \text{Door}), S).
\]

**Frame Axiom:** No actions lock the door:

\[
\text{unlocked}(\text{Door}, \text{do}(A, S)) \leftarrow \\
\text{unlocked}(\text{Door}, S) \land \\
\text{poss}(A, S).
\]
Example: axiomatizing \textit{carried}

Picking up an object causes it to be carried:

\[
\text{carrying}(Ag, Obj, do(pickup(Ag, Obj), S)) \leftarrow \\
\text{poss}(\text{pickup}(Ag, Obj), S).
\]

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

\[
\text{carrying}(Ag, Obj, do(A, S)) \leftarrow \\
\text{carrying}(Ag, Obj, S) \land \\
\text{poss}(A, S) \land \\
A \neq \text{putdown}(Ag, Obj).
\]
Example: \textit{sitting\_at}

An object is sitting at a location if:

- it moved to that location:
  \[
  \textit{sitting\_at}(\text{Obj}, \text{Pos}, \text{do}(\text{move}(\text{Obj}, \text{Pos}_0, \text{Pos}), S)) \leftarrow \text{poss}(\text{move}(\text{Obj}, \text{Pos}_0, \text{Pos})).
  \]

- it was put down at that location:
  \[
  \textit{sitting\_at}(\text{Obj}, \text{Pos}, \text{do}(\text{putdown}(\text{Ag}, \text{Obj}), S)) \leftarrow \text{poss}(\text{putdown}(\text{Ag}, \text{Obj}), S) \land \text{at}(\text{Ag}, \text{Pos}, S).
  \]

- it was at that location before and didn’t move and wasn’t picked up.
More General Frame Axioms

The only actions that undo \textit{sitting\_at} for object \textit{Obj} is when \textit{Obj} moves somewhere or when someone is picking up \textit{Obj}.

\[\text{sitting\_at}(Obj, Pos, do(A, S)) \leftarrow\]
\[\text{poss}(A, S) \land\]
\[\text{sitting\_at}(Obj, Pos, S) \land\]
\[\forall Pos_1 \quad A \neq \text{move}(Obj, Pos, Pos_1) \land\]
\[\forall Ag \quad A \neq \text{pickup}(Ag, Obj).\]

The last line is equivalent to:

\[\sim \exists Ag \quad A = \text{pickup}(Ag, Obj)\]

which can be implemented as

\[\text{sitting\_at}(Obj, Pos, do(A, S)) \leftarrow\]
\[\cdots \land \cdots \land \cdots \land\]
\[\sim \text{is\_pickup\_action}(A, Obj).\]

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

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

has an answer

$$S = do(move(rob, mail, o103),$$
$$do(pickup(rob, k1),$$
$$do(move(rob, o103, mail),$$
$$do(move(rob, o109, o103), init))))).$$
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:**

  \[
  \text{ask } \text{carrying}(\text{rob}, \text{parcel}, S) \land \text{in}(\text{rob}, \text{lab2}, S).
  \]

  the last action needed is irrelevant to the left subgoal.

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