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 can be modeled as jumping from one time point to another.
- **Continuous time** You can model time as being dense.
- **Event-based time** Time steps don’t have to be uniform; you can consider the time steps between interesting events.
- **State space** Instead of considering time explicitly, you can consider actions as mapping 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 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 previous times.

The Delivery Robot World

- r131, r129, r127, r125, r123, r121, r119
- storage
- parcel
- rob
- key k1
- mail
- door1
- lab2
- stairs
- r101, r103, r105, r107, r109, r111
- o103, o109, o111

© David Poole, Alan Mackworth, Randy Goebel, and Oxford University Press 1999
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.

- **opens(Key, Door)** is true if key `Key` opens door `Door`. This is static.
- **adjacent(Pos₁, Pos₂)** is true if position `Pos₁` is adjacent to position `Pos₂` so that the robot can move from `Pos₁` to `Pos₂` in one step.
- **between(Door, Pos₁, Pos₂)** is true if `Door` is between position `Pos₁` and position `Pos₂`. 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.
Initial Situation

sitting_at(rob, o109).
sitting_at(parcel, storage).
sitting_at(k1, mail).

Static Facts

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

Derived Relations

at(Obj, Pos) ← sitting_at(Obj, Pos).
at(Obj, Pos) ← carrying(Ag, Obj) ∧ at(Ag, Pos).
adjacent(o109, o103).
adjacent(o103, o109).
...

adjacent(lab2, o109).
adjacent(P1, P2) ←
  between(Door, P1, P2) ∧
  unlocked(Door).