STRIPS

Sven Koenig, USC

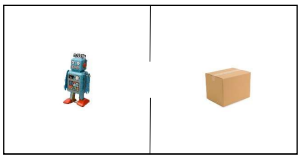
Russell and Norvig, 3rd Edition, Section 10.1

These slides are new and can contain mistakes and typos.
Please report them to Sven (skoenig@usc.edu).

Encoding Planning Problems

- Qualification problem
  - Problem of specifying when an action can be applied
- Frame problem
  - Problem of specifying what an action does NOT change
- Ramification problem
  - Problem of specifying what changes an action makes implicitly (e.g. when you drive a car, everything in it changes its location)

Logic



- We could encode planning problems in some form of logic.
- In situation calculus, for example, …
  - One adds an argument to each predicate that specifies the state s in which it is true. For example,
    - At(Robot, Room1, s)
  - One defines Result(s,a) as a function that maps a state and operator (= action) to the successor state resulting from applying the operator in the state. For example,
    - FORALL s FORALL a (At(Robot, Room2, Result(s,a)) EQUIV (a=Go(Room1, Room2) OR At(Robot, Room2, s) AND NOT a=Go(Room2,Room1)))
  - Then, one tries to prove whether there exists an operator sequence that transforms the start state to the goal state. If so, one finds such an operator sequence as a by-product of the proof.

Logic

- Reasoning in logic is very powerful but also slow.
- Furthermore, we do not want to find any operator sequence that transforms the start state to the goal state but one of minimal (or small) cost.
- Thus, we study efficient special-purpose knowledge representation formalisms and reasoning procedures that represent and solve planning problems.

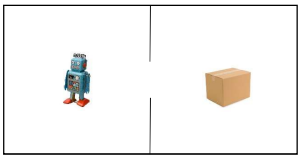
STRIPS

- STRIPS (Stanford Research Institute Problem Solver, around 1971): AI researchers no longer use the planner but we still use versions of the language that encodes planning problems.
  - The start and goal state are sets of propositions in form of non-negated ground(ed) predicates. A non-negated ground(ed) predicate is a predicate name followed by a list of constants that specify the arguments. The goal state can be partially specified.
  - Operator schemata consist of a name, a set (= list) of variable names, a set of preconditions (PRECOND) and a set of effects (EFFECTS).
  - Operators are instantiated operator schemata. An instantiated operator schema is one where each variable is bound to a constant. Variables can be bound to ANY constant.

STRIPS

- STRIPS (Stanford Research Institute Problem Solver, around 1971): AI researchers no longer use the planner but we still use versions of the language that encodes planning problems.
  - PRECOND is a set of non-negated predicates. A non-negated predicate is a predicate name followed by a list of arguments that are either constants or variables. If PRECOND of an operator unifies with a state, then the operator is applicable in the state.
  - EFFECTS is a set of (non-negated and/or negated) predicates. If an operator is applicable in a state, then the successor state (that results from the application of the operator in the state) is obtained from the state by adding all non-negated predicates (= add effects) from EFFECTS and deleting all negated predicates (= delete effects) from EFFECTS. The order of additions and deletions should not matter.
  - Our textbook uses slightly different conventions. Make sure to follow our ones.

Simple STRIPS Example



Start state:

At(Robot,Room1)
At(Box,Room2)
Pushable(Box)
Unequal(Room1,Room2)
Unequal(Room2,Room1)

Goal state:

At(Box,Room1)

Operator schemata:

Go(x,y) "Move Robot from x to y"
PRECOND: At(Robot, x), Unequal(x,y)
EFFECT: At(Robot, y), NOT At(Robot,x)

Push(b,x,y) "Let Robot push b from x to y"
PRECOND: At(b,x), At(Robot,x), Pushable(b), Unequal(x,y)
EFFECT: At(b,y), At(Robot,y), NOT At(b,x), NOT At(Robot,x)

Simple STRIPS Example

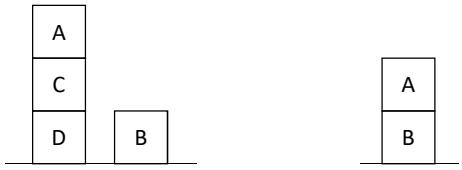
- Comments
  - Pushable(Box), Unequal(Room1,Room2) and Unequal(Room2,Room1) are in all states that can be reached from the start state by operator applications. Such predicates are often used to specify properties of objects, including their types.
  - Without the precondition Unequal(x,y), one could apply the operator Go(Room1,Room1) in the start state. The order of the addition of At(Robot,Room1) and the deletion of At(Robot,Room1) matters, which we do not want. Thus, we prevent the application of the operator Go(Room1,Room1) in any state via the precondition Unequal(x,y).
  - Both Unequal(Room1,Room2) and Unequal(Room2,Room1) need to be part of the start state to make it possible to apply Go(Room1,Room2) and Go(Room2,Room1), respectively, in the appropriate states.

Simple STRIPS Example

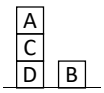
- Comments
  - In operator schema Go(x,y), we want x and y to be rooms. We could "type" them by adding Room(x) and Room(y) to PRECOND and Room(Room1) and Room(Room2) to the start state. (This is how Pushable(b) is used in the operator schema Push(b,x,y).) However, this is unnecessary here since the precondition Unequal(x,y) is only satisfied if x and y are rooms (that is, Room1 or Room2).
  - The goal state is only partially specified. We only want Box to be in Room1. We do not care about other predicates, such as where Robot is.

More Complex STRIPS Example

- Blocks world (for which we give a minimalistic STRIPS specification) This domain consists of a set of cube-shaped blocks sitting on a table. The blocks can be stacked, but only one block can fit directly on top of another. A robot arm can pick up a block and move it to another position, either on the table or on top of another block. The arm can pick up only one block at a time, so it cannot pick up a block that has another one on it. The goal will always be to build one or more stacks of blocks, specified in terms of which blocks are on top of which other blocks.



More Complex STRIPS Example



Start state

Goal state

Start state:

Clear(A) Clear(B)
On(A,C) On(C,D) On(D,Table) On(B,Table)
Unequal(A,B) Unequal(A,C) Unequal(A,D)
Unequal(B,A) Unequal(B,C) Unequal(B,D)
Unequal(C,A) Unequal(C,B) Unequal(C,D)
Unequal(D,A) Unequal(D,B) Unequal(D,C)
Block(A) Block(B) Block(C) Block(D)

Goal state:

Clear(A)
On(A,B)
On(B,Table)

Operator schemata:

MoveToBlock(b,x,y) "Move block b that is on top of x (block or table) onto top of block y"
PRECOND: On(b,x), Clear(b), Clear(y), Block(y), Unequal(b,y)
EFFECT: On(b,y), Clear(x), NOT On(b,x), NOT Clear(y)

← MoveToBlock(b,x,b) cannot be allowed since a block cannot be put onto itself.

MoveToTable(b,x) "Move block b that is on top of block x onto the table"
PRECOND: On(b,x), Clear(b), Block(x)
EFFECT: On(b,Table), Clear(x), NOT On(b,x)

← MoveToTable(b,Table) cannot be allowed since the order of the effects does matter for it.