CompSci 366

Classical Planning: Regression Planning

Outline

- Review of Progression Planning(PP)
- Regression Planning Overview
- Regression Planning in Prolog

Review of Progression Planning(PP)

- PP:
  - Checks whether the new current state satisfies the goal (if it is then done!).
  - Finds all the ops ***applicable in current state***.
  - Selects one of those ops and ***“progresses” the state thru that op to get the new current state***.
  - Recurse with the ***new state and the old goal***.

Preview of Regression Planning (RP)

- RP:
  - Checks whether the current goal set is satisfied by the initial situation (if it is then done!).
  - Find ops that can ***satisfy some of the goals***.
  - Selects one of the ops and "***regresses" the goal set thru that op to get the new current goal set***.
  - Recurse with the ***new goal set and the old initial situation***.

What is regression?

- What does it mean to regress a logic expression, ***L***, thru an action description, ***A***?
- It means computing another logic expression, ***P***, such that

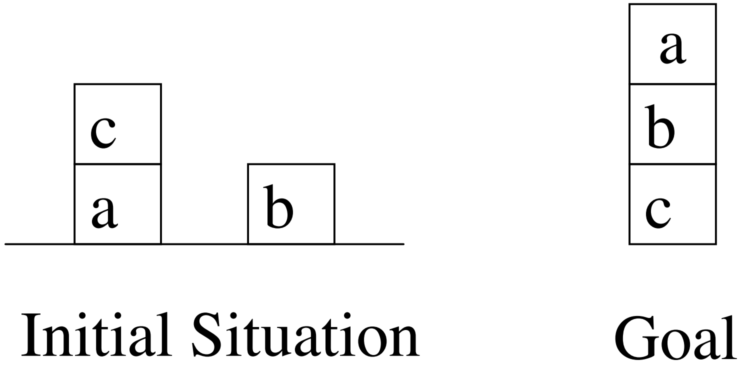
  if ***P*** is true in state ***S***

  then if ***A*** is applied to ***S***

  then ***L*** must be true in that new state.

Regression Example

Example Problem

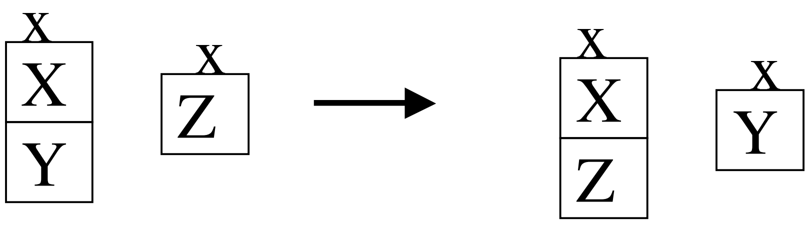


Regression Example cont’d

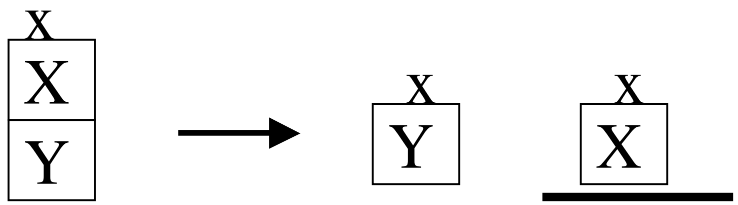
Domain Operators:

Note: x = ***clear***

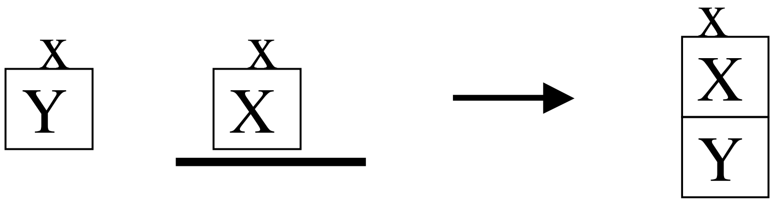
MBB(X,Y,Z):



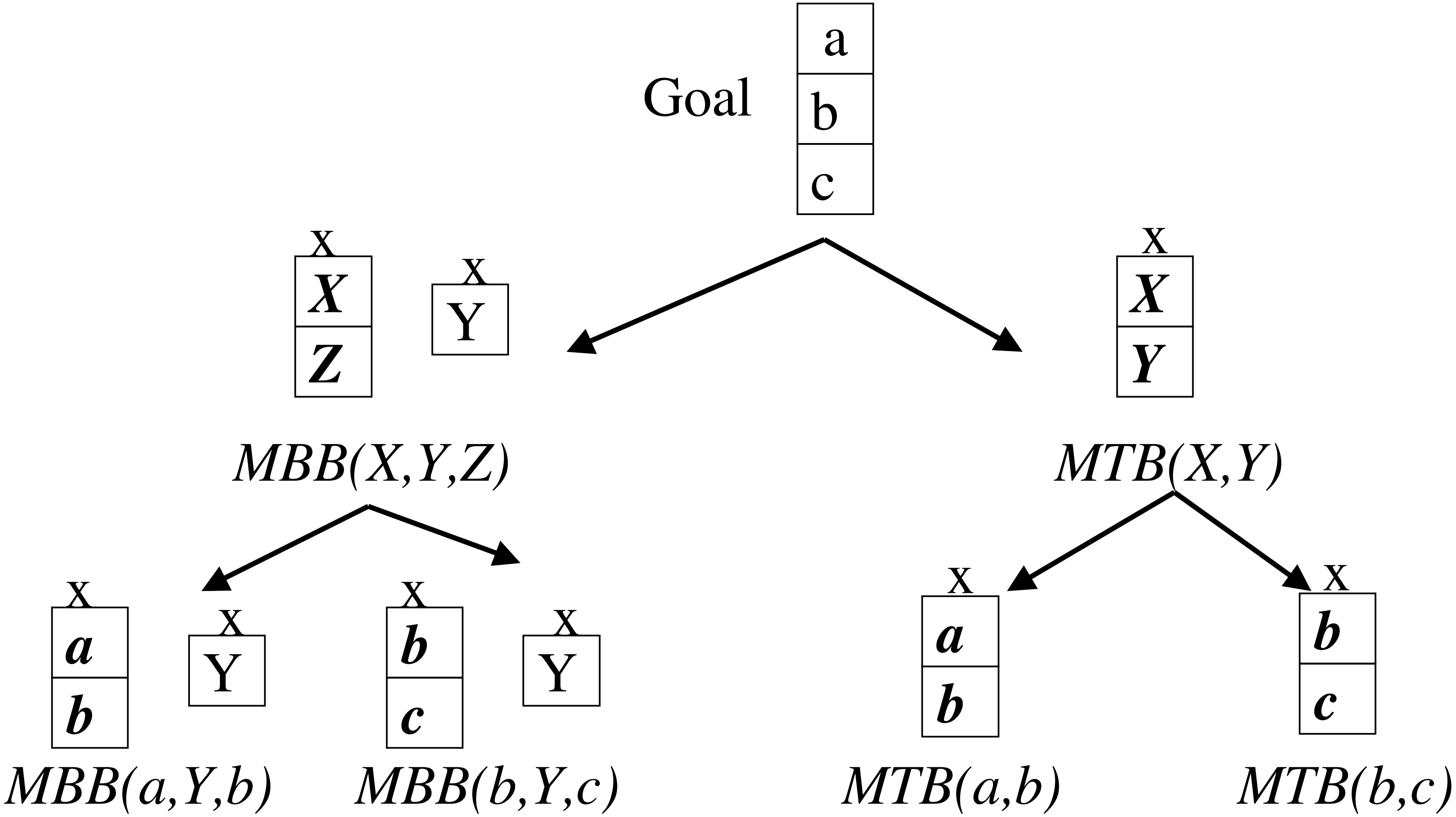
MBT(X,Y):



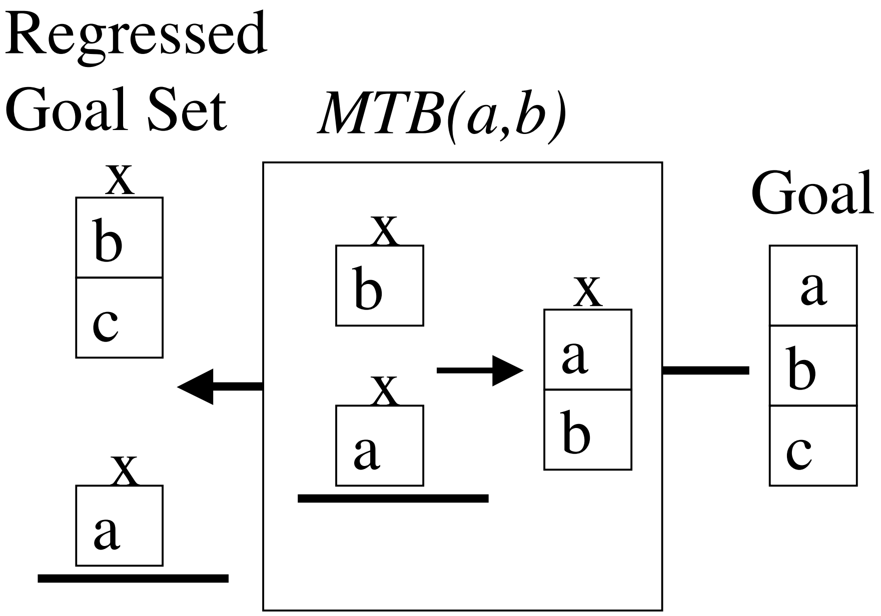
MTB(X,Y):



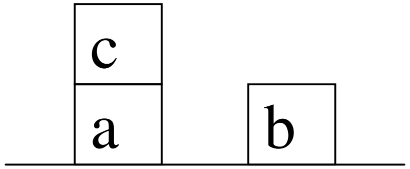
Find Some Ops that Satisfy Some Goals



Select Op & Regress Goal

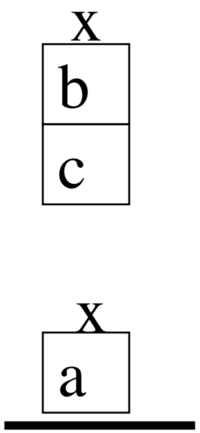


Does Initial State Satisfy Regressed Goal Set?

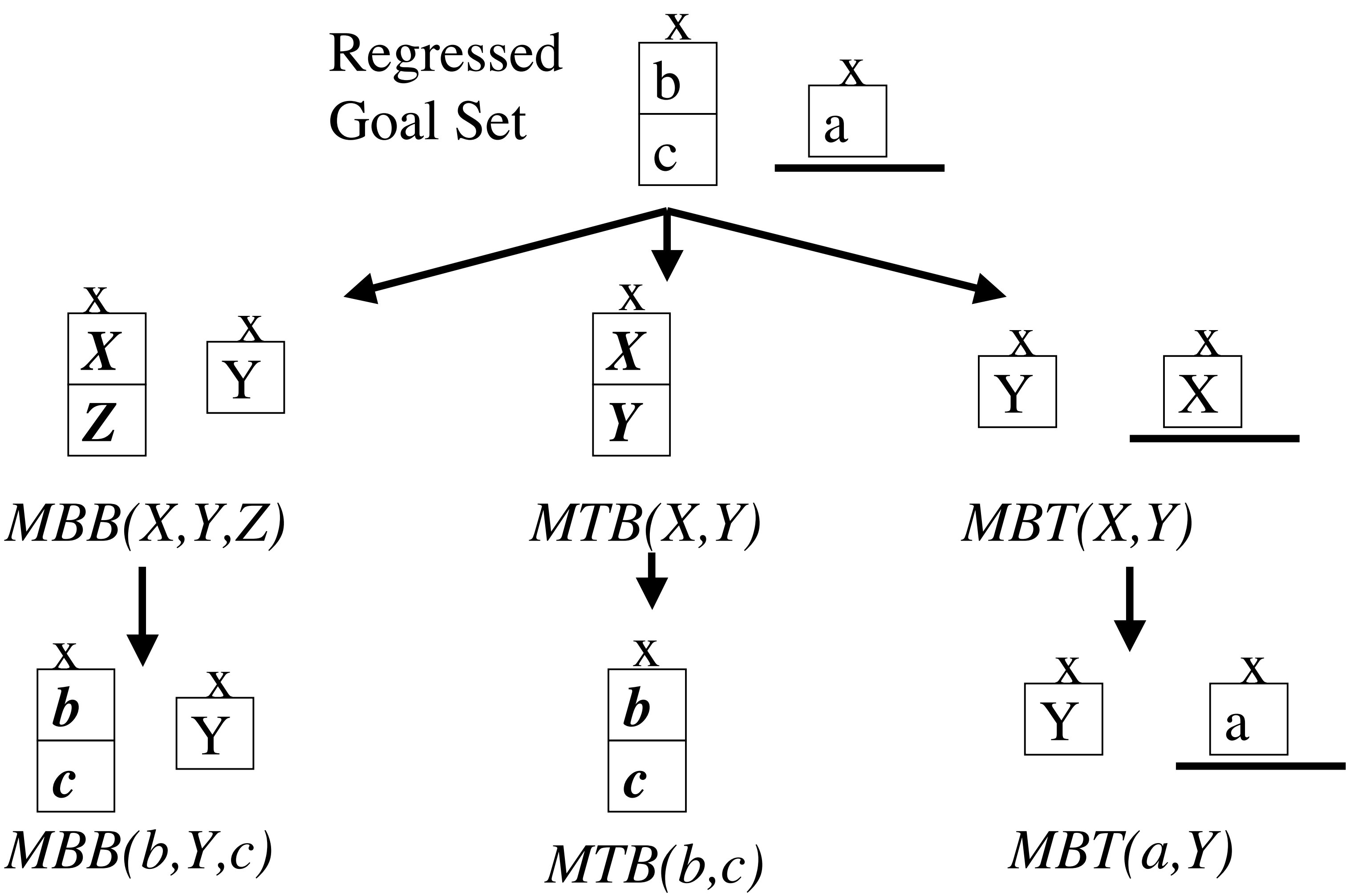


Initial Situation

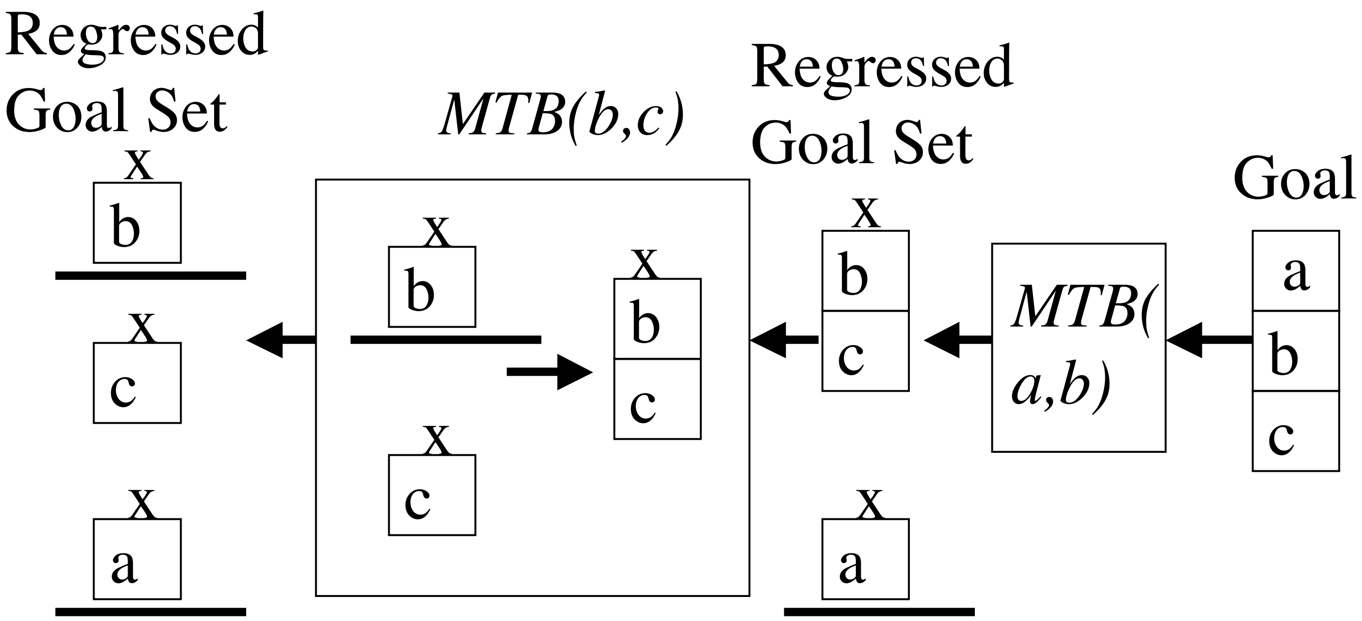
Regressed
Goal Set



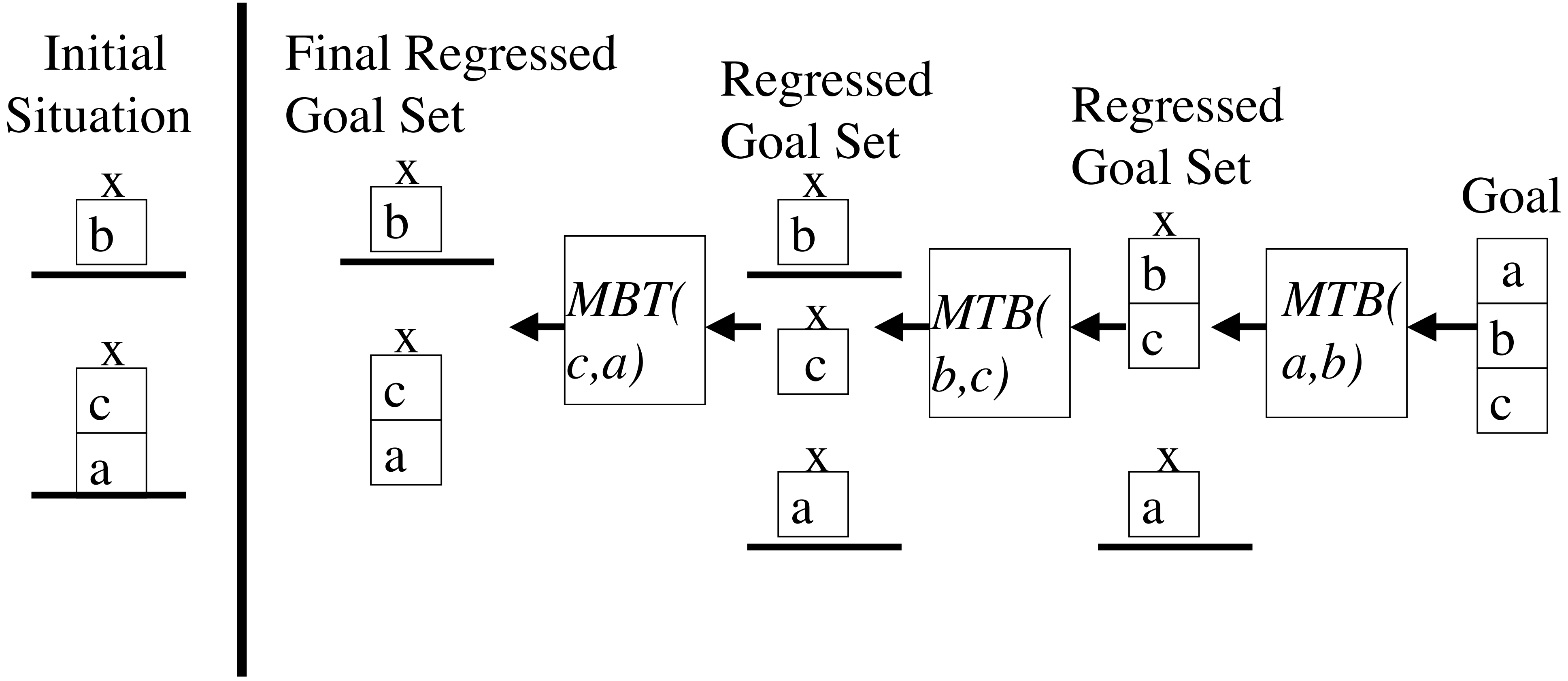
Find Some Ops that Satisfy Some Goals



Select Op & Regress Goal



Initial Situation Satisfies Regressed Goal Set



Programming Regression Planning in Prolog

- We now have a procedural understanding of regression planning.

- How do we go about implementing RP in Prolog?

Programming Regression Planning in Prolog cont’d

- Think declaratively: i.e., define what a problem being solved by a plan means.
- Want a definition that uses regression of goal sets rather than progression of states.
- Want simple definition - use divide & conquer approach.
- Specifically, induction/recursion!

Programming Regression Planning in Prolog cont'd

- Base case: what would be the simplest case where a plan solved a problem?
- The absolutely simplest case would be where the initial situation already satisfied the goal and we had the empty plan.
- We could say this in Prolog by:

```
solvedBy(problem(Init, Goals), [ ]) :-
     satisfiedBy(Goals, Init).
```

Programming Regression Planning in Prolog cont'd

- Inductive case: since the simplest case used the empty plan, perhaps we should do the induction on the length of the plan.
- We could do this in Prolog by:

```
solvedBy(problem(Init, Goals), Plan) :-
     append(Rest, [Step], Plan),
     achievesSome(Step, Goals),
     regressesThruTo(Goals, Step, NewGoals),
     solvedBy(problem(Init, NewGoals), Rest).
```

Programming Regression Planning in Prolog cont'd

- ***achievesSome(Step, Goals)'***s definition should be rather obvious.
- ***regressesThruTo(Goals, Step, NewGoals)***'s definition is probably less so.
- The goal of regression is to compute the "weakest" precondition, ***NewGoals***, for when it is guaranteed that ***Goals*** will be achieved after executing ***Step***.
- However, this is too expensive and so we compute a sufficient condition for guaranteeing that ***Goals*** will be achieved after executing ***Step***.

Programming Regression Planning in Prolog cont'd

- Regression should fail if resulting goal description is inconsistent (assume initial situation consistent then no valid plan should be able to achieve an inconsistent situation).
- Inconsistencies occur if there is a goal, ***G***, and ***not(G)*** is either an effect or a precondition of the step. ***Why???***

Programming Regression Planning in Prolog cont'd

- Regressed goals = Goals - Effects(Step) + Preconditions(Step), where "-" is set difference.