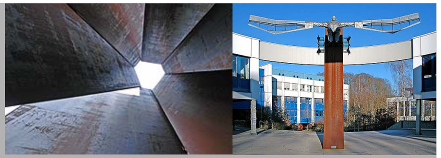
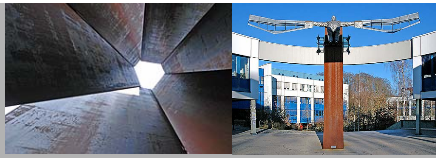
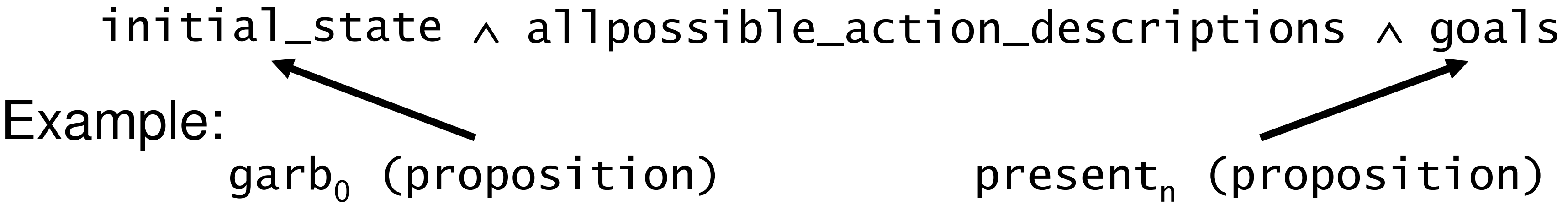


Planning as Satisfiability (SAT-Plan)



SAT-Plan

- Translate the planning problem into a satisfiability problem for length *n* of Plan



`initial_state` $\wedge$ `allpossible_action_descriptions` $\wedge$ `goals`

Example:

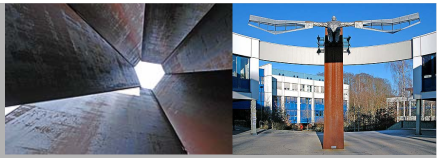
$\texttt{garb}_0$ `(proposition)`

$\texttt{present}_n$ `(proposition)`

- Use a satisfiability solver to solve the problem: find a model (true for action $\in$ Plan false for other actions)

- Solution to the satisfiability problem corresponds to a plan

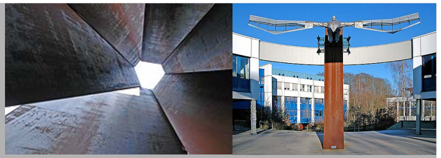
Quick Review of Satisfiability



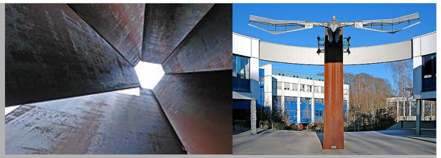
- Propositional calculus
  - no predicates
  - each variable represents an entire proposition
- Boolean formulas
  - Usually represented in Conjunctive Normal Form (CNF)
  - Conjunct of clauses; each clause is a disjunct of literals
    $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee w) \wedge (\neg z \vee \neg v \vee w)$
- Solution: an assignment of values to the variables that makes the entire formula true
- The corresponding decision problem is NP-complete (It was the first problem ever shown to be NP-complete [Cook, U. of Toronto]
- This problem has been very heavily studied:

→ Fast “satisfiability solvers”

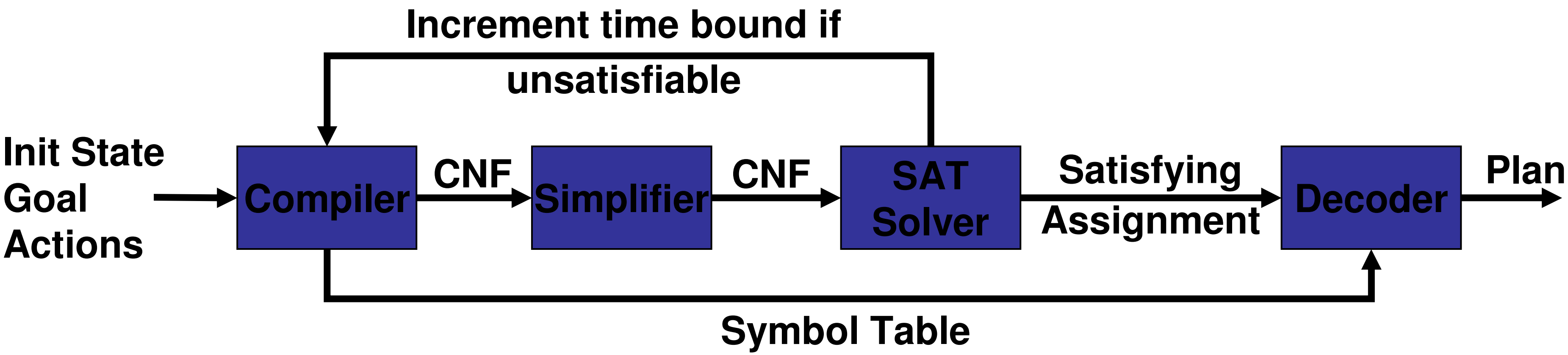
SAT-Plan General Approach



- Input: Planning problem
  - Compile it
  - Guess a plan length *n*
  - Generate apropoitional forumula *L* such that *L* is satisfiable iff a plan of lenght *n* exists
  - Use a symbol table to keep track of the correspondence between the propositional variables an the plan elements
  - Use fast (linear time) techiques to simplify the formula
  - Use a satisfiability solver to solve the simplified formula
  - Using the symbol table, decode the solution to get a plan



Final Process:



Increment time bound if unsatisfiable

Init State
Goal
Actions

Compiler

CNF

Simplifier

CNF

SAT Solver

Satisfying Assignment

Decoder

Plan

Symbol Table