SAT solvers for planning

Philippe Morignot
The planning problem

« Given action templates, a state and goals, find a sequence of instantiated actions, which lead the initial state to a (final) state which contains the goals. »

- Action planning = plan synthesis = generation of action plans: Activity of constructing a plan.
- Planner = task planner = action planner = A.I. planner: software which constructs a plan.
Planning Domain Definition Language: domain

(:action puton
 :parameters (?b ?u ?t - block)
 :precondition (and (clear ?b)
 (on ?b ?u))
 (clear ?t))
 :effect (and (not (on ?b ?u)) (clear ?u)
 (on ?b ?t) (not (clear ?t))))

- Qualification / ramification problem
Planning Domain Definition Language: problem

(define (problem blocks-24-1)
  (:domain blocks)
  (:objects X W V U T S R Q P O N M L K J I H G F E D C A B)
  (:init
    (CLEAR K) (CLEAR I) (ONTABLE C) (ONTABLE O)
    (ON K F) (ON F T) (ON T B) (ON B G) (ON G R)
    (ON R M) (ON M E) (ON E J) (ON J V) (ON V N)
    (ON N U) (ON U H) (ON H C) (ON I A) (ON A P)
    (ON P Q) (ON Q D) (ON D W) (ON W X) (ON X S)
    (ON S L) (ON L O) (HANDEMPTy))
  (:goal (and
    (ON L C) (ON C P) (ON P Q) (ON Q M) (ON M B)
    (ON B G) (ON G F) (ON F K) (ON K E) (ON E R)
    (ON R A) (ON A W) (ON W T) (ON T N) (ON N J)
    (ON J U) (ON U S) (ON S D) (ON D H) (ON H V)
    (ON V O) (ON O I) (ON I X))))
Partially-ordered partially-instantiated plan

Initial
(clear C) (on C A) (clear ?t)
(on C A) (clear A) (on C ?t) (not (clear ?t))

puton C A ?t
(clear C) (not (on C A)) (clear A) (on C ?t) (not (clear ?t))

puton B table C
(clear B) (not (on B ta.)) (on B ta.) (on B C) (not (clear C))

puton A table B
(clear A) (not (on A ta.)) (on A ta.) (on A B) (not (clear B))

Final
(on A B) (on B C)

donot (on A ta.) (on C A) (on C ta.) (clear C) (on B ta.) (clear B)

Planners

• Planners in a plan space (Dan Weld).
• Planners using forward search in a state space (Jorg Hoffman, Hector Geffner).
• Planners using backward search in a state space (M. Helmert).
• Planners using evolutionnary algorithms (M. Schoenauer)
• Planners using temporal logic (P. Doherty).
• Planners using constraint programming (V. Vidal).
• Planners using SAT solvers (H. Kautz & B. Selman, J. Rintanen).

Planning as Satisfiability: Heuristics

Jussi Rintanen

Institute for Integrated and Intelligent Systems, Griffith University, Queensland, Australia
Principle

1. Set the length of the plan to $n (= 1)$
2. Encode the planning problem of size $n$ as a propositional formula:
   \[
   \text{initial\_state} \land \text{all\_plans\_n} \land \text{goals}
   \]
3. Run a SAT solver
4. IF solution found THEN decode \hspace{1cm} // SUCCESS
5. Increment $n$

- Improvement: Try plan lengths in parallel.
Encoding

- **Goals:** \( \text{on}(A,B)@T \land \text{on}(B,C)@T \)
- **Initial state:** \( \text{clear}(C)@0 \land \text{on}(C,A)@0 \land \text{clear}(B)@0 \)
  \( \land \neg \text{on}(A,C)@0 \land \neg \text{on}(A,B)@0 \land \neg \text{on}(B,C)@0 \land \neg \text{on}(B,A)@0 \)
  \( \land \neg \text{clear}(A)@0 \)
  \( \text{hypothèse du monde fermé} \)
- **Axiom schemas on preconditions:**
  \[ \forall x, \forall y, \forall z, \forall t : \]
  \( \text{puton}(x, y, z)@t \Rightarrow \text{on}(x,y)@t \land \text{clear}(x)@t \land \text{clear}(z)@t \)
- **Axiom schemas on effects:**
  \[ \forall x, \forall y, \forall z, \forall t : \]
  \( \text{on}(x,y)@t \land \text{clear}(x)@t \land \text{clear}(z)@t \land \text{puton}(x,y,z)@t \Rightarrow \text{clear}(y)@t+1 \land \text{on}(x,z)@t+1 \)
- **One operator at a time:**
  \[ \forall x, \forall y, \forall y', \forall z, \forall z', \forall t : y <> y' \land z <> z' : \]
  \( \neg ( \text{puton}(x, y, z)@t \land \text{puton}(x, y, z')@t ) \)
- **Frame axiom schemas:**
  \[ \forall p, \forall t: \]
  \[ \begin{align*}
  p@\!(t+1) & \Rightarrow ( p@t \lor a_1 p@t \lor \ldots \lor a_n p@t ) \\
  \neg p@\!(t+1) & \Rightarrow ( \neg p@t \lor a_1 \neg p@t \lor \ldots \lor a_n \neg p@t )
  \end{align*} \]
Algorithms

• **Conflict-directed Clause Learning:**

The main loop of the CDCL algorithm (see Fig. 1) chooses an unassigned variable, assigns a truth-value to it, and then performs unit propagation to extend the current valuation $v$ with forced variable assignments that directly follow from the existing valuation by the unit resolution rule. If one of the clauses is falsified, a new clause which would have prevented considering the current valuation is derived and added to the clause set. This new clause is a logical consequence of the original clause set. Then, some of the last assignments are undone, and the assign-infer-learn cycle is repeated. The procedure ends when the empty clause has been learned (no valuation can satisfy the clauses) or a satisfying valuation has been found.

• **Heuristic for variable selection:** for a given goal, choose an action that achieves the goal and that can be taken at the earliest time at which the goal can become true.
Results

Figure 14: Number of instances solved by different planners
Applications (1 / 2)

• Disassemble a car engine (NOAH, Earl Sacerdoti 1974)
• Organize the military invasion of Iraq (SIPE, David Wilkins, 1980).
• Autonomy of a spatial probe around Jupiter (2000).
• Debug a xerox machine
Applications (2 / 2)