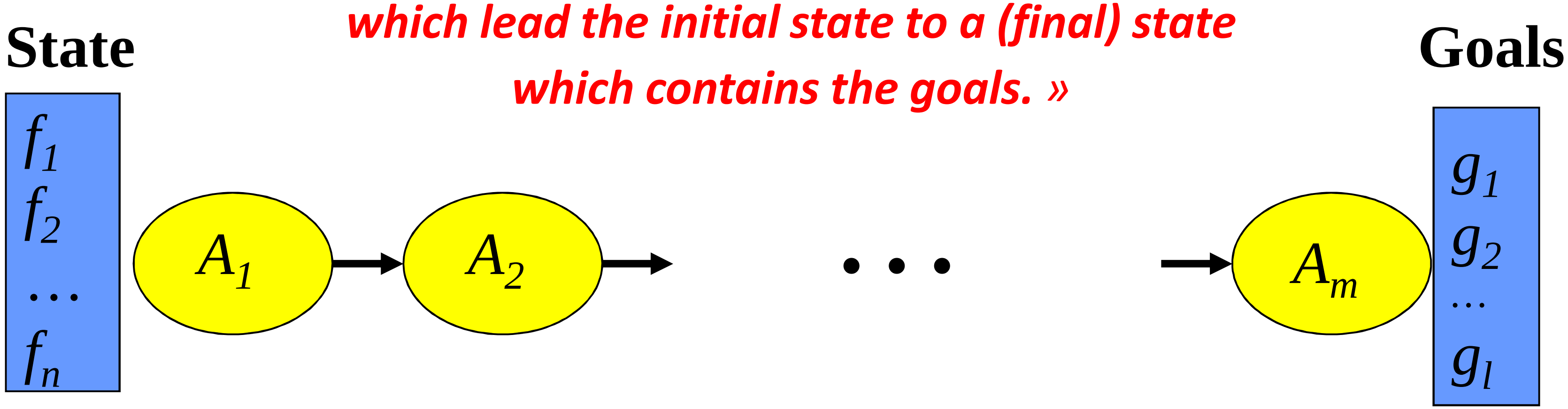
SAT solvers for planning

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. »

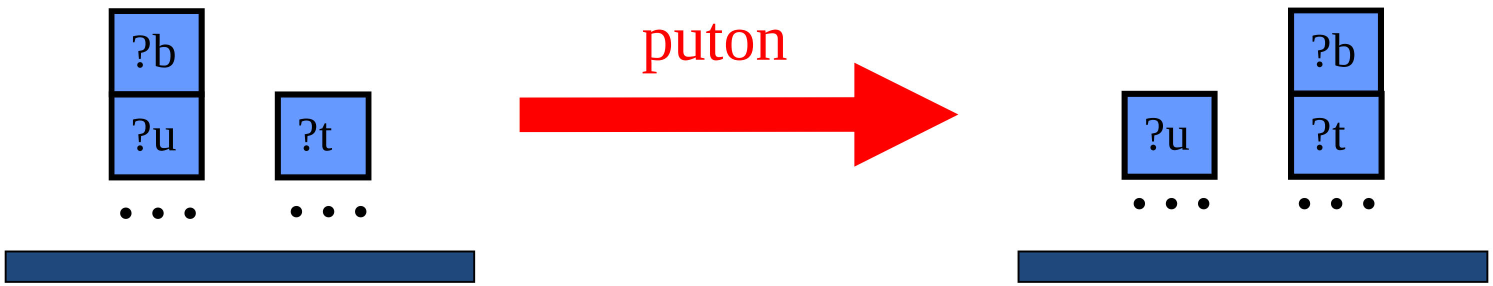


- Action planning = plan synthesis = generation of action plans: Activity of constructing a plan.
- Planner = task planner = action planner = A.I. planner: software which constructs a plan.

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))))
```

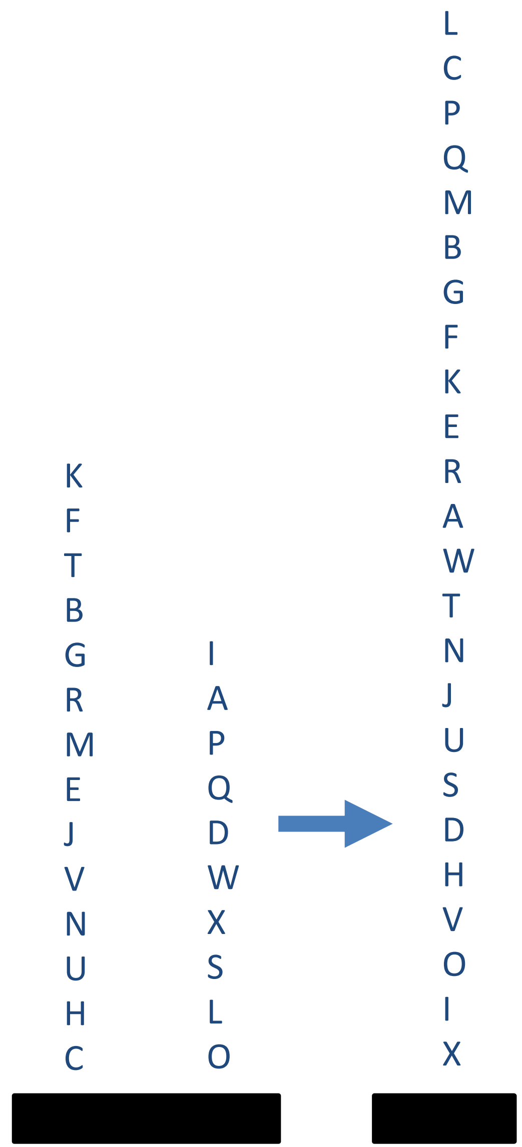
puton ?b ?u ?t	
(clear ?b) (on ?b ?u) (clear ?t)	(not (on ?b ?u)) (clear ?u) (on ?b ?t) (not (clear ?t))



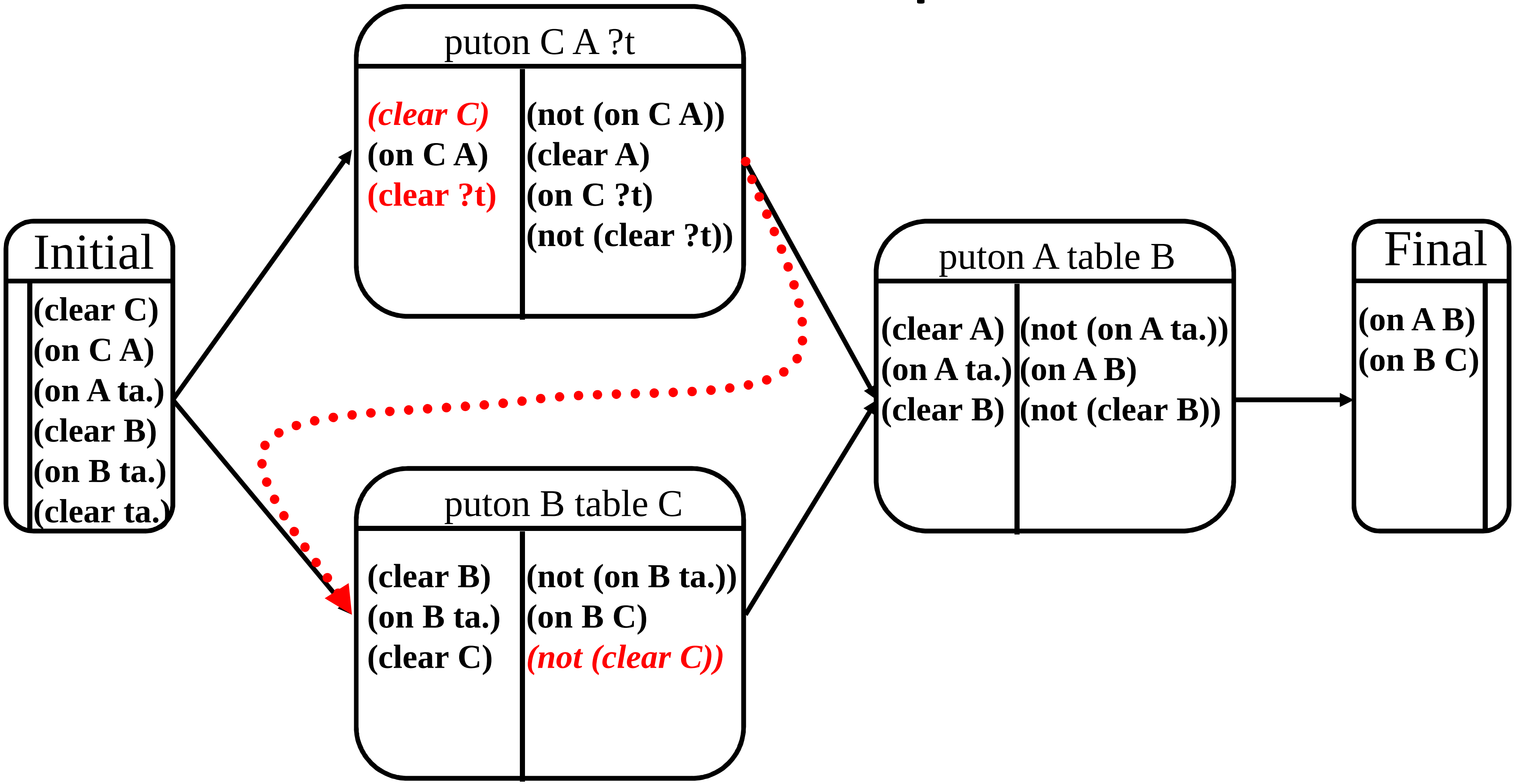
- Qualification / ramification problem

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



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:

   $$initial\_state \wedge all\_plans\_n \wedge goals$$

3. Run a SAT solver
4. IF solution found THEN decode   // SUCCESS
5. Increment *n*

- Improvement: Try plan lengths in parallel.

Encoding

- Goals: $on(A,B)@T \wedge on(B,C)@T$
- Initial state: $clear(C)@0 \wedge on(C,A)@0 \wedge clear(B)@0$
  $(\wedge \neg on(A,C)@0 \wedge \neg on(A,B)@0 \wedge \neg on(B,C)@0 \wedge \neg on(B,A)@0$
  $\wedge \neg on(C, B)@0 \wedge \neg clear(A)@0)$ // *hypothèse du monde fermé*
- Axiom schemas on preconditions:

$\forall$ ***x***, $\forall$ ***y***, $\forall$ ***z***, $\forall$ ***t*** :

$$puton(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})@\boldsymbol{t} \Rightarrow on(\boldsymbol{x},\boldsymbol{y})@\boldsymbol{t} \wedge clear(\boldsymbol{x})@\boldsymbol{t} \wedge clear(\boldsymbol{z})@\boldsymbol{t}$$

- Axiom schemas on effects:

$\forall$ ***x***, $\forall$ ***y***, $\forall$ ***z***, $\forall$ ***t*** :

$$on(\boldsymbol{x},\boldsymbol{y})@\boldsymbol{t} \wedge clear(\boldsymbol{x})@\boldsymbol{t} \wedge clear(\boldsymbol{z})@\boldsymbol{t} \wedge puton(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})@\boldsymbol{t} \Rightarrow clear(\boldsymbol{y})@\boldsymbol{t}+1 \wedge on(\boldsymbol{x},\boldsymbol{z})@\boldsymbol{t}+1$$

- One operator at a time:

$\forall$ ***x***, $\forall$ ***y***, $\forall$ ***y'***, $\forall$ ***z***, $\forall$ ***z'***, $\forall$ **t** / ***y*** <> ***y'*** $\wedge$ ***z*** <> ***z'*** :

$$\neg (puton(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})@\boldsymbol{t} \wedge puton(\boldsymbol{x} ,\boldsymbol{y'} ,\boldsymbol{z'})@\boldsymbol{t})$$

- Frame axiom schemas:

$\forall$ p, $\forall$ t:

$$\begin{cases} p@(t+1) \Rightarrow (p@t \vee a_1{}^{p}@t \vee \ldots \vee a_n{}^{p}@t) \\ \neg p@(t+1) \Rightarrow (\neg p@t \vee a_1{}^{\neg p}@t \vee \ldots \vee a_n{}^{\neg p}@t) \end{cases}$$

Algorithms

- <u>Conflict-directed Clause Learning:</u>

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.

- <u>Heuristic for variable selection:</u> 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

				LAMA		
		Mp	M	2008	2011	FF
1998-GRID	5	5	3	5	5	5
1998-GRIPPER	20	20	20	20	20	20
1998-LOGISTICS	30	30	30	29	30	30
1998-MOVIE	30	30	30	30	30	30
1998-MPRIME	20	20	18	20	20	19
1998-MYSTERY	19	19	18	19	14	16
2000-BLOCKS	102	63	82	54	95	80
2000-FREECELL	60	45	32	59	59	60
2002-DEPOTS	22	22	22	18	22	22
2002-DRIVERLOG	20	20	19	20	20	16
2002-ZENO	20	20	18	19	20	20
2004-AIRPORT	50	50	48	38	38	39
2004-OPTICAL-TELEGRAPH	14	14	14	3	14	13
2004-PHILOSOPHERS	29	29	29	12	14	14
2004-PIPESWORLD-TANKAGE	50	38	11	38	41	22
2004-PIPESWORLD-NOTANKAGE	50	41	20	44	44	36
2004-PSR-SMALL	50	50	50	50	50	43
2004-SATELLITE	36	35	35	31	36	36
2006-PATHWAYS	30	30	30	28	28	20
2006-ROVERS	40	40	40	40	40	40
2006-STORAGE	30	30	25	21	20	18
2006-TPP	30	30	30	30	30	28
2006-TRUCKS	30	21	22	8	15	11
2008-CYBER-SECURITY	30	30	30	29	29	4
2011-BARMAN	20	10	0	17	20	0
2011-ELEVATORS	20	20	1	20	20	20
2011-FLOORTILE	20	20	20	2	6	5
2011-NOMYSTERY	20	17	17	13	18	4
2011-OPENSTACKS	20	0	0	18	20	20
2011-PARCPRINTER	20	20	20	12	20	20
2011-PARKING	20	0	0	20	20	8
2011-PEGSOL	20	20	19	19	20	20
2011-SCANALYZER	20	20	13	20	20	20
2011-SOKOBAN	20	2	0	13	19	17
2011-TIDYBOT	20	17	2	14	16	15
2011-TRANSPORT	20	4	0	16	19	9
2011-VISITALL	20	0	0	20	7	4
2011-WOODWORKING	20	20	20	16	20	4
1998-ASSEMBLY-ADL	24	24	23	24	23	24
2000-ELEVATOR-SIMPLE	150	150	150	149	150	150
2000-SCHEDULE-ADL	150	150	150	134	138	134
2002-SATELLITE-ADL	20	20	20	20	20	20
2004-AIRPORT-ADL	50	49	47	31	45	30
2004-OPTICAL-TELEGRAPH-ADL	48	39	41	19	1	17
2004-PHILOSOPHERS-ADL	48	48	48	23	14	14
2006-TRUCKS-ADL	29	16	22	17	14	11
2008-OPENSTACKS-ADL	30	18	15	30	30	30
total	1646	1416	1304	1332	1414	1238
weighted score	47	39.06	34.36	37.97	40.48	34.11
confidence interval low		34.82	-7.86	-6.09	-3.33	-9.69
confidence interval high		42.79	-1.98	4.14	6.36	-0.12

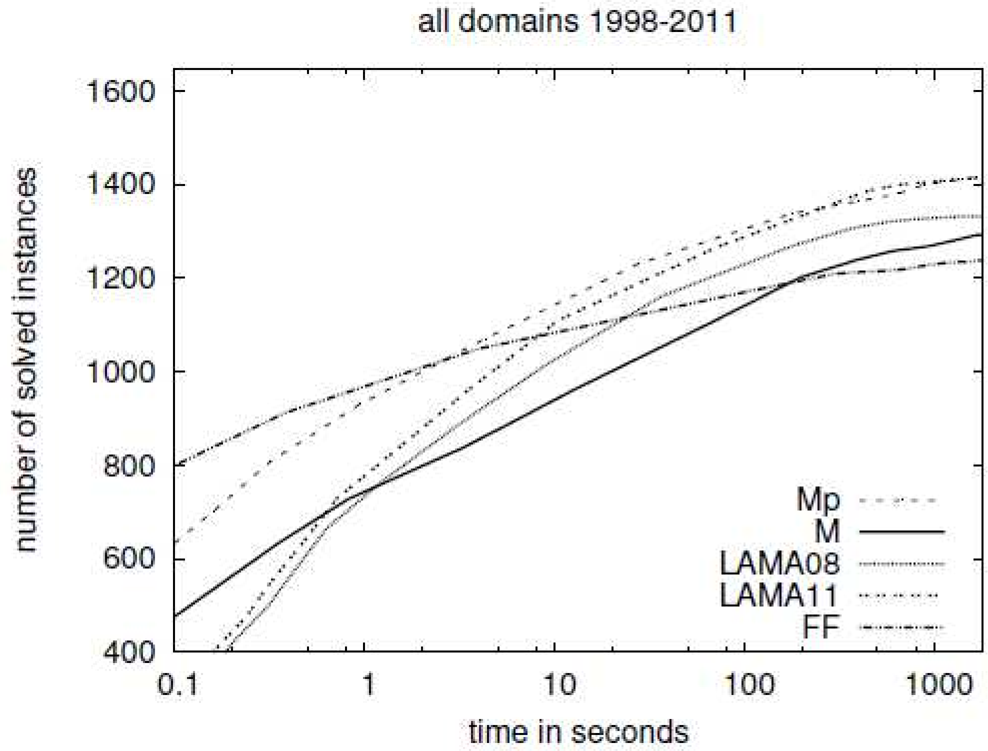


Figure 14: Number of instances solved by different planners

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)