SAT-based conformant planning

E. Giunchiglia

University of Genova

C. Castellini, P. Ferraris, A. Tacchella

Goal

See whether the Planning as Satisfiability idea can be extended to Expressive Action Languages allowing for

1. Concurrency, Constraints, Nondeterminism, environmental changes, . . . : [Giunchiglia, KR'00]
2. in an effective way: [Ferraris&Giunchiglia, AAAI'00], [Castellini, Giunchiglia, Tacchella, ECP'01].

Talk overview

1. Conformant Planning via SAT
2. $\mathcal{C}$-plan overview
3. Optimization 1: Pruning possible plans,
4. Optimization 2: Introducing backjumping and learning
5. Experimental Analysis
6. Conclusions

Conformant planning

A planning problem is a triple $\langle I, D, G \rangle$ in which

- I is a formula representing the possible initial states,
- D is a formal representation describing how actions affect the world, and
- G is a formula representing the goal states.

In conformant planning, the problem is to find a *sequence* of actions which, if executed sequentially starting from any initial state, is ensured to lead to a goal state.

$\Rightarrow$ if D is deterministic and I is a singleton, then

conformant planning = classical planning ...

... however, I am not making any assumption about I, D, G, and focus is on correct, complete, optimal approaches.

A conformant planning problem

There are $\#B$ bombs. Each bomb may or may not be armed. There are also $\#C$ containers, and if a bomb is placed in a container, then it is no longer dangerous, even if it is armed.

Starting from an initial state in which the set of armed bombs is unknown, the goal is to reach a “safe” state, i.e., a state in which which each bomb is no longer dangerous.

The idea

Consider a planning problem $\pi = \langle I, D, G \rangle$. Let tr_i^D be a propositional formula corresponding to the transition relation of D.

We may divide the problem of finding a "valid" plan for π into two parts:

1. *generate* "possible" plans of length n by satisfying

$$I_0 \wedge \wedge_{i=0}^{n-1} tr_i^D \wedge G_n.$$

2. *test* whether a generated plan $\alpha^1; \ldots; \alpha^n$ is "valid" by checking that

$$I_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} tr_i^D \models G_n.$$

The idea

Consider a planning problem $\pi = \langle I, D, G \rangle$. Let tr_i^D be a propositional formula corresponding to the transition relation of D.

We may divide the problem of finding a “valid” plan for π into two parts:

1. *generate* “possible” plans of length n by satisfying

$$I_0 \wedge \wedge_{i=0}^{n-1} tr_i^D \wedge G_n.$$

2. *test* whether a generated plan $\alpha^1; \ldots; \alpha^n$ is “valid” by checking that

$$I_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} tr_i^D \models G_n.$$

If the transition relation is not total then the “test” does not work!

Conformant planning via SAT

Consider a planning problem $\pi = \langle I, D, G \rangle$. Let tr_i^D be a propositional formula corresponding to the transition relation of D.

We may divide the problem of finding a "valid" plan for π into two parts:

1. *generate* "possible" plans of length n by satisfying

$$I_0 \wedge \wedge_{i=0}^{n-1} \mathit{tr}_i^D \wedge G_n.$$

2. *test* whether a generated plan $\alpha^1; \ldots; \alpha^n$ is "valid" by checking that

$$I_0 \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge \wedge_{i=0}^{n-1} \mathit{trt}_i^D \models G_n \wedge \neg Z_n.$$

   where trt_i^D is defined as

$$(\mathit{tr}_i^D \wedge \neg Z_i \wedge \neg Z_{i+1}) \vee ((Z_i \vee \neg \mathit{Poss}_i^D) \wedge Z_{i+1}),$$

   (Z_i is a new fluent, Poss_i^D is true for an action α in a state σ if α is executable in σ).

Algorithm: $\mathcal{C}$-SAT

$P := I_0 \wedge \wedge_{i=0}^{n-1} tr_i^D \wedge G_n;$ $\quad V := I_0 \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} trt_i^D \wedge \neg(G_n \wedge \neg Z_n);$

function $\mathcal{C}$-SAT() **return** $\mathcal{C}$-SAT_GEN$_{\text{DLL}}$(*cnf*(P), $\{\}$).

function $\mathcal{C}$-SAT_GEN$_{\text{DLL}}$(φ, μ)

if $\varphi = \{\}$ **then return** $\mathcal{C}$-SAT_TEST(μ);

if $\{\} \in \varphi$ **then return** *False*;

if $\{$ a unit clause $\{L\}$ occurs in φ $\}$ **then return** $\mathcal{C}$-SAT_GEN$_{\text{DLL}}$(*assign*(L, φ),$\mu \cup \{L\}$);

L := $\{$ a literal occurring in φ $\}$;

return $\mathcal{C}$-SAT_GEN$_{\text{DLL}}$(*assign*(L, φ),$\mu \cup \{L\}$) **or** $\mathcal{C}$-SAT_GEN$_{\text{DLL}}$(*assign*$(\overline{L}, \varphi)$,$\mu \cup \{\overline{L}\}$).

function $\mathcal{C}$-SAT_TEST(μ)

α := $\{$the "partial" plan in $\mu\}$;

foreach $\{$"total" plan $\alpha^1; \ldots; \alpha^n$ which extends $\alpha\}$

if not SAT($\wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge V$) **then exit with** $\alpha^1; \ldots; \alpha^n$;

return *False*.

Algorithm: $\mathcal{C}$-SAT

P := $I_0 \wedge \wedge_{i=0}^{n-1} tr_i^D \wedge G_n$; V := $I_0 \wedge \neg Z_0 \wedge \wedge_{i=0}^{n-1} trt_i^D \wedge \neg(G_n \wedge \neg Z_n)$;

function $\mathcal{C}$-SAT() **return** $\mathcal{C}$-SAT_GEN$_{\mathrm{DLL}}$(*cnf*(P), $\{\}$).

function $\mathcal{C}$-SAT_GEN$_{\mathrm{DLL}}$(φ, μ)

- **if** $\varphi = \{\}$ **then return** $\mathcal{C}$-SAT_TEST(μ);
- **if** $\{\} \in \varphi$ **then return** *False*;
- **if** $\{$ a unit clause $\{L\}$ occurs in φ $\}$ **then return** $\mathcal{C}$-SAT_GEN$_{\mathrm{DLL}}$(*assign*(L, φ),$\mu \cup \{L\}$);
- L := $\{$ a literal occurring in φ $\}$;
- **return** $\mathcal{C}$-SAT_GEN$_{\mathrm{DLL}}$(*assign*(L, φ),$\mu \cup \{L\}$) **or** $\mathcal{C}$-SAT_GEN$_{\mathrm{DLL}}$(*assign*$(\overline{L}, \varphi)$,$\mu \cup \{\overline{L}\}$).

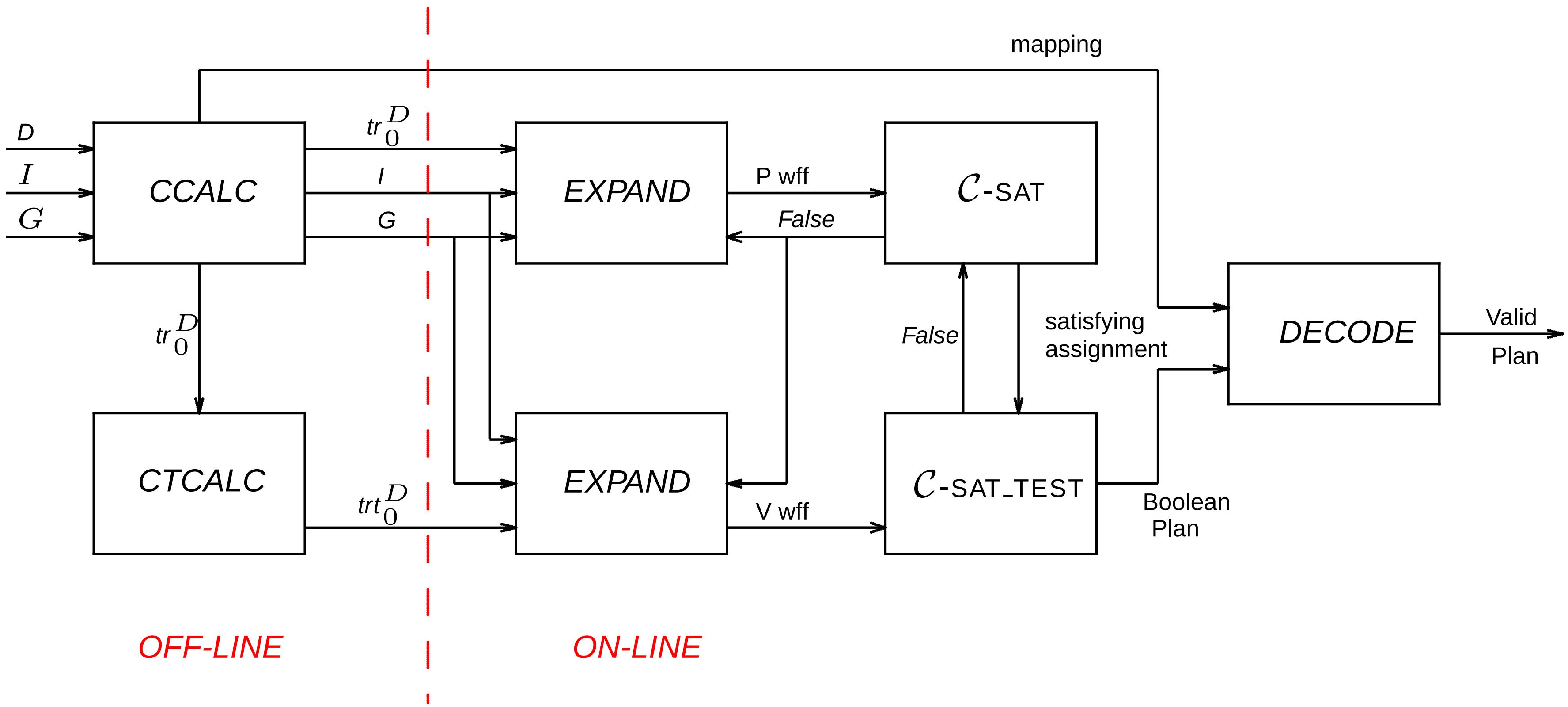
function $\mathcal{C}$-SAT_TEST(μ)

- α := $\{$the "partial" plan in $\mu\}$;
- **foreach** $\{$"total" plan $\alpha^1; \ldots; \alpha^n$ which extends $\alpha\}$
  - **if not** SAT($\wedge_{i=0}^{n-1} \alpha_i^{i+1} \wedge V$) **then exit with** $\alpha^1; \ldots; \alpha^n$;
- **return** *False*.

For any fixed n,

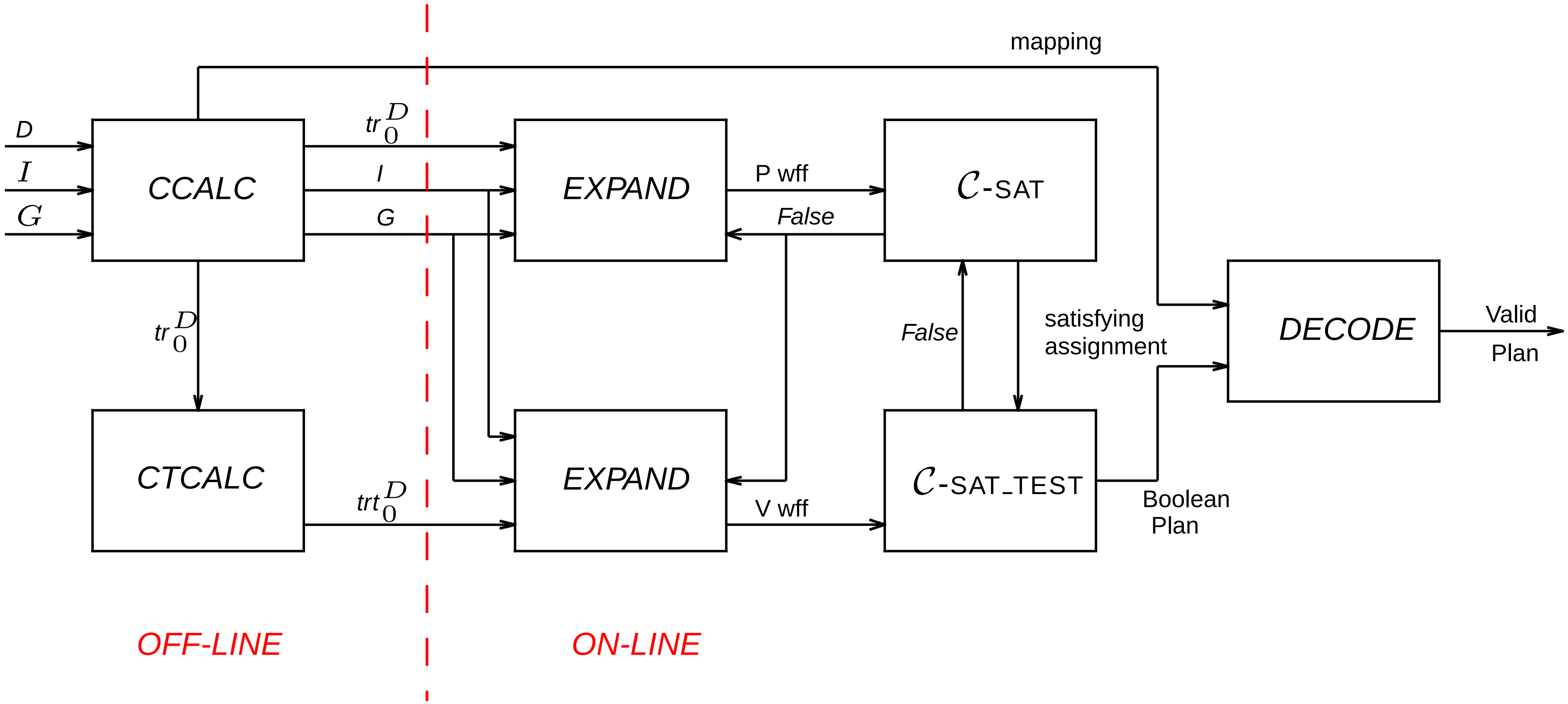
$\mathcal{C}$-SAT is correct and complete

$\mathcal{C}$-plan overview



- *CCALC* has been developed by Norman McCain,
- $\mathcal{C}$-plan first version has been implemented by Paolo Ferraris.

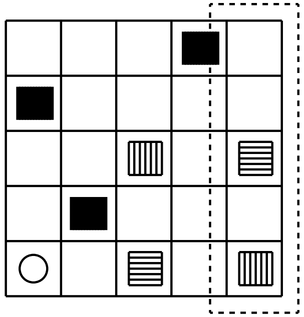
$\mathcal{C}$-plan overview



- *CCALC* has been developed by Norman McCain,
- $\mathcal{C}$-plan first version has been implemented by Paolo Ferraris.

$\mathcal{C}$-plan is correct and complete

Example: Simple robot navigation problem

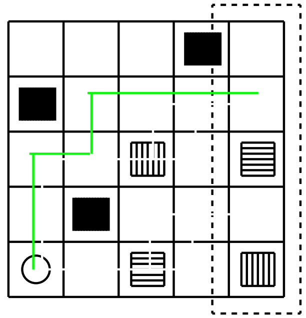


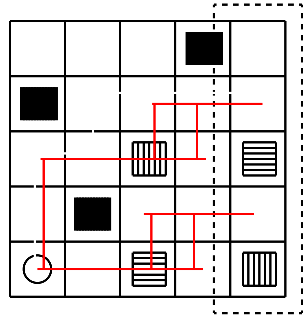
A robot (the circle) has to reach a position inside the dashed box.

Black locations are occupied by objects.

Either locations $\{(3,1),(5,3)\}$ or $\{(3,3),(5,1)\}$ are occupied.

$\mathcal{C}$-plan finds a valid plan but after generating many possible plans





Problems:

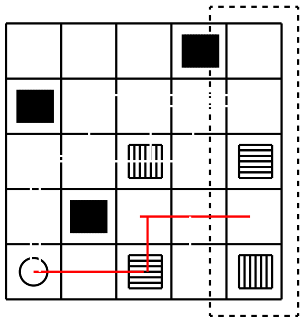
1. For each possible configuration of the obstacles, all possible plans are considered, and
2. For each fixed configuration of the obstacles, nothing is learned from previous attempts

Problem 1: all possible plans are generated

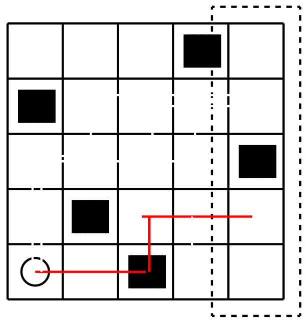
Solution: for plan generation, eliminate uncertainty in the planning problem

As soon as a possible plan is generated and rejected, the corresponding configuration of obstacles is discarded:

From



to



The new configuration is then considered for generating future possible plans.

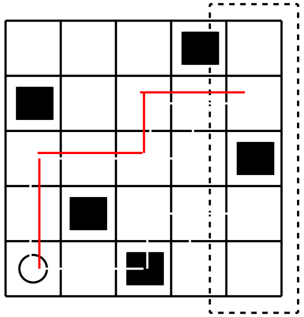
The system is still correct and complete.

Problem 2: nothing is learned from previus failures

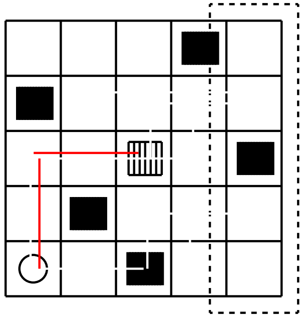
Solution: compute the reasons for failures, and introduce backjumping and learning

For a given configuration of objects, “learn” the reason for failure

From



to



Reasons are dynamically learnt and forgot, as in SAT.

The system is still correct and complete.

Comparative Analysis

Systems:

- Bonet's and Geffner's GPT [*AIPS'2000*]
- Cimatti's and Roveri's CMBP [*ECP'1999, JAIR'2000*]

Test cases:

- Purely parallel
- Purely sequential, low uncertainty
- Purely sequential, high uncertainty

Working environment:

- Pentium III, 850MHz, 512MBRAM running Linux SUSE 7.0
- Timeout at 1200s
- Systems stopped when memory requirements $>$ 512MB

Purely parallel

	GPT	CMBP		Cplan				
$\#B$-$\#C$	Total	#s	Total	#s	#pp	Last	Tot.search	Total
2-1	0.03	2	0.00	1	1	0.00	0.00	0.00
4-1	0.03	4	0.01	1	1	0.00	0.00	0.00
6-1	0.04	6	0.02	1	1	0.00	0.00	0.00
8-1	0.15	8	0.08	1	1	0.00	0.00	0.00
10-1	0.27	10	0.61	1	1	0.00	0.00	0.00
15-1	17.05	15	42.47	1	1	0.00	0.00	0.00
20-1	MEM	—	MEM	1	1	0.00	0.00	0.00

Purely Sequential, Low Uncertainty

	GPT	CMBP		Cplan				
$\#B$-$\#C$	Total	#s	Total	#s	#pp	Last	Tot.search	Total
2-1	0.10	3	0.00	3	6	0.00	0.00	0.01
2-5	0.04	2	0.01	1	1	0.00	0.00	0.00
2-10	0.05	2	0.03	1	1	0.00	0.00	0.00
4-1	0.04	7	0.00	7	540	0.12	0.15	0.65
4-5	0.23	4	0.79	1	1	0.00	0.00	0.00
4-10	2.23	4	11.30	1	1	0.00	0.00	0.01
6-1	0.09	11	0.04	11	52561	15.39	49.39	221.55
6-5	3.29	7	16.80	3	98346	56.92	57.34	419.53
6-10	74.15	—	MEM	1	1	0.00	0.00	0.01
8-1	0.41	15	0.20	—	—	—	—	TIME
8-5	32.07	11	112.48	—	—	—	—	TIME
8-10	MEM	—	MEM	1	1	0.00	0.00	0.01
10-1	2.67	19	1.55	—	—	—	—	TIME
10-5	MEM	15	974.45	—	—	—	—	TIME
10-10	MEM	—	MEM	1	1	0.00	0.00	0.04

Purely Sequential, Low Uncertainty

$\#B$-$\#C$	GPT	CMBP		Cplan				
	Total	#s	Total	#s	#pp	Last	Tot.search	Total
2-1	0.10	3	0.00	3	6	0.00	0.00	0.01
2-5	0.04	2	0.01	1	1	0.00	0.00	0.00
2-10	0.05	2	0.03	1	1	0.00	0.00	0.00
4-1	0.04	7	0.00	7	540	0.12	0.15	0.65
4-5	0.23	4	0.79	1	1	0.00	0.00	0.00
4-10	2.23	4	11.30	1	1	0.00	0.00	0.01
6-1	0.09	11	0.04	11	52561	15.39	49.39	221.55
6-5	3.29	7	16.80	3	98346	56.92	57.34	419.53
6-10	74.15	—	MEM	1	1	0.00	0.00	0.01
8-1	0.41	15	0.20	—	—	—	—	TIME
8-5	32.07	11	112.48	—	—	—	—	TIME
8-10	MEM	—	MEM	1	1	0.00	0.00	0.01
10-1	2.67	19	1.55	—	—	—	—	TIME
10-5	MEM	15	974.45	—	—	—	—	TIME
10-10	MEM	—	MEM	1	1	0.00	0.00	0.04

Purely Sequential, High Uncertainty

	GPT	CMBP		Cplan				
$\#B$-$\#C$	Total	#s	Total	#s	#pp	Last	Tot.search	Total
2-1	0.03	3	0.00	3	3	0.00	0.00	0.00
2-5	0.04	2	0.00	1	1	0.00	0.00	0.00
2-10	0.24	2	0.02	1	1	0.00	0.00	0.02
4-1	0.17	7	0.01	7	15	0.01	0.02	0.02
4-5	0.06	4	0.54	1	1	0.01	0.00	0.01
4-10	0.38	4	7.13	1	1	0.02	0.00	0.02
6-1	0.08	11	0.03	11	117	0.25	1.39	2.01
6-5	0.33	7	10.71	3	48	0.62	0.66	1.36
6-10	7.14	—	MEM	1	1	0.00	0.00	0.00
8-1	0.06	15	0.17	15	1195	12.23	147.25	184.29
8-5	2.02	11	90.57	3	2681	14.84	15.60	317.13
8-10	MEM	—	MEM	1	1	0.00	0.00	12.68
10-1	0.21	19	1.02	—	—	—	—	TIME
10-5	12.51	15	591.33	—	—	—	—	TIME
10-10	MEM	—	MEM	1	1	0.00	0.00	0.06

Purely Sequential, High Uncertainty

$\#B$-$\#C$	GPT	CMBP		Cplan				
	Total	#s	Total	#s	#pp	Last	Tot.search	Total
2-1	0.03	3	0.00	3	3	0.00	0.00	0.00
2-5	0.04	2	0.00	1	1	0.00	0.00	0.00
2-10	0.24	2	0.02	1	1	0.00	0.00	0.02
4-1	0.17	7	0.01	7	15	0.01	0.02	0.02
4-5	0.06	4	0.54	1	1	0.01	0.00	0.01
4-10	0.38	4	7.13	1	1	0.02	0.00	0.02
6-1	0.08	11	0.03	11	117	0.25	1.39	2.01
6-5	0.33	7	10.71	3	48	0.62	0.66	1.36
6-10	7.14	—	MEM	1	1	0.00	0.00	0.00
8-1	0.06	15	0.17	15	1195	12.23	147.25	184.29
8-5	2.02	11	90.57	3	2681	14.84	15.60	317.13
8-10	MEM	—	MEM	1	1	0.00	0.00	12.68
10-1	0.21	19	1.02	—	—	—	—	TIME
10-5	12.51	15	591.33	—	—	—	—	TIME
10-10	MEM	—	MEM	1	1	0.00	0.00	0.06

Summary

- SAT-based (conformant) planning is very flexible. It is easy to define procedures for planning in the presence of, e.g., for
  - concurrency,
  - constraints, and
  - nondeterminism
- In the presence of an uncertain initial state and/or nondeterminism, $\mathcal{C}$-PLAN
  - employs a "generate" and "test" approach
  - incorporates some optimizations, but many other are possible to improve performances, e.g., taking into account domain specific features, or by giving up optimality and/or completeness.
  - range of applicability is different from GPT's and CMBP's
  - performances are not directly correlated with the "degree of uncertainty" of the domain.

Other Symbolic approaches

- David Smith’s alternative “generate and test, anytime” approach,
- Jussi Rintanen’s QBF-based approach,
- Alessandro Cimatti’s BDD-based approach.

References

[Giunchiglia, 2000] Enrico Giunchiglia. Planning as satisfiability with expressive action languages: Concurrency, constraints and nondeterminism. In KR'2000.

[Giunchiglia, 2000] Paolo Ferraris, Enrico Giunchiglia. Planning as satisfiability in nondeterministic domains. In AAAI'2000

[Castellini *et al.*, 2001] Claudio Castellini, Enrico Giunchiglia, and Armando Tacchella. Improvements to SAT-based conformant planning. In ECP'2001.

[Castellini *et al.*, 2001] Claudio Castellini, Enrico Giunchiglia, and Armando Tacchella. Sat-based planning in complex domains: Concurrency, constraints and nondeterminism, 2001. Tech. Report.