End-to-End Classical Planning using CP and Belief Propagation

Damien Van Meerbeeck[1], Gilles Pesant[2], Jendrik Seipp[1]

[1] Machine Reasoning Lab, Linköping University, Sweden

[2] Quosséça Lab, Polytechnique Montréal, Canada

Classical Planning

Classical Planning is the challenge of finding a sequence of actions transforming an initial situations into one that satisfies a goal condition. In **Optimal Classical Planning**, the plan must minimize the sum of action costs.

Motivation

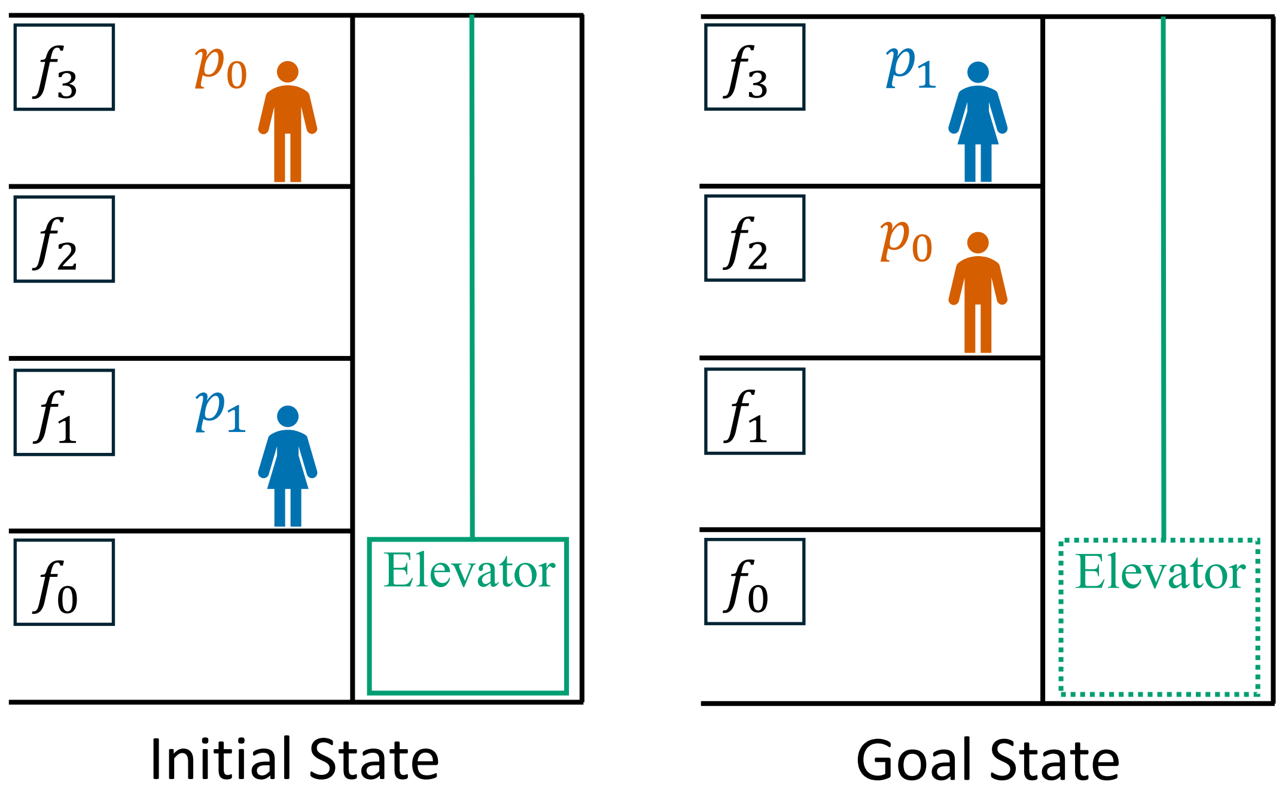
- Solve planning tasks using **Constraint Programming** (CP).
- Let users specify **additional constraints** on the plan in a declarative way.
- Use the power of CPBP solvers to **scale better for large planning tasks**.

Pipeline

Input planning task

Planning task in first order planning definition language (**PDDL**):

- **Objects**
- **Predicates**
- **Initial** and **goal states**
- **Actions** with preconditions and effects on predicates



Lifted actions (of unit cost):

- board(floor, passenger)
- depart(floor, passenger)
- down(from, to)
- up(from, to)

Ground finite-domain task

Use the **Fast Downward** planning system to **translate** the input task into a ground finite-domain task (**SAS⁺**).

Task Variables :

- Boarded $p_0 : \{F, T\}$
- Served $p_0 : \{F, T\}$
- Boarded $p_1 : \{F, T\}$
- Served $p_1 : \{F, T\}$
- Elevator : $\{f_0, f_1, f_2, f_3\}$

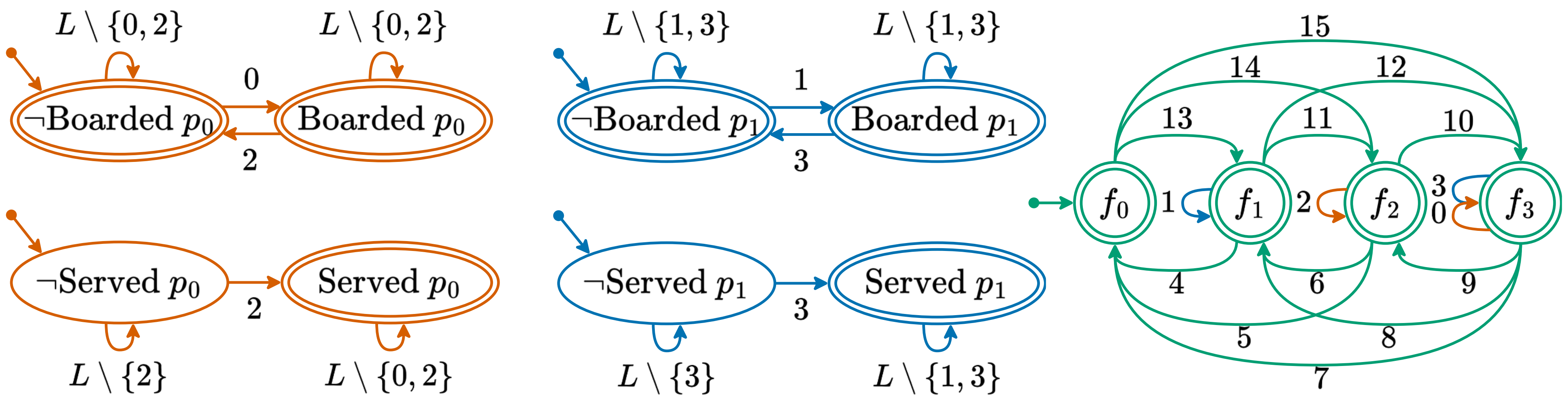
Grounded actions:

0	board(f_3, p_0)
1	board(f_1, p_1)
2	depart(f_2, p_0)
3	depart(f_3, p_1)

4	down(f_1, f_0)	10	up(f_2, f_3)
5	down(f_2,f_0)	11	up(f_1, f_2)
6	down(f_2, f_1)	12	up(f_1, f_3)
7	down(f_3, f_0)	13	up(f_0, f_1)
8	down(f_3, f_1)	14	up(f_0, f_2)
9	down(f_3, f_2)	15	up(f_0, f_3)

Automata

Project the task to each variable of the SAS⁺ task, to obtain a **Factored Transition System**. Additionally, group parallel actions (*Group*) and prune irrelevant actions (*Group+Prune*).

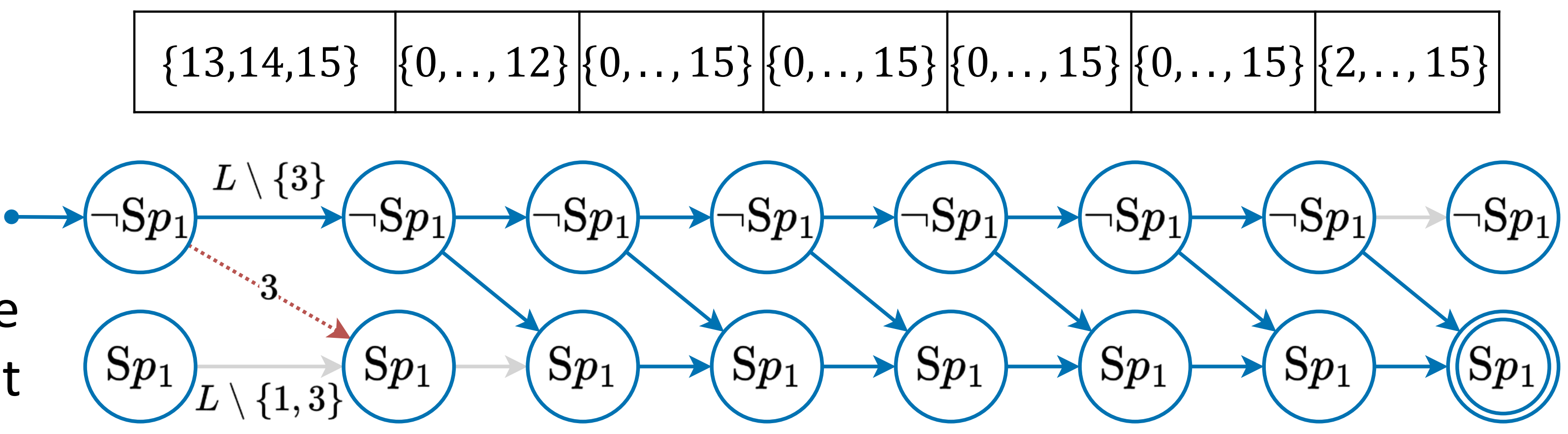


CP Model

Model of the planning task in CP:

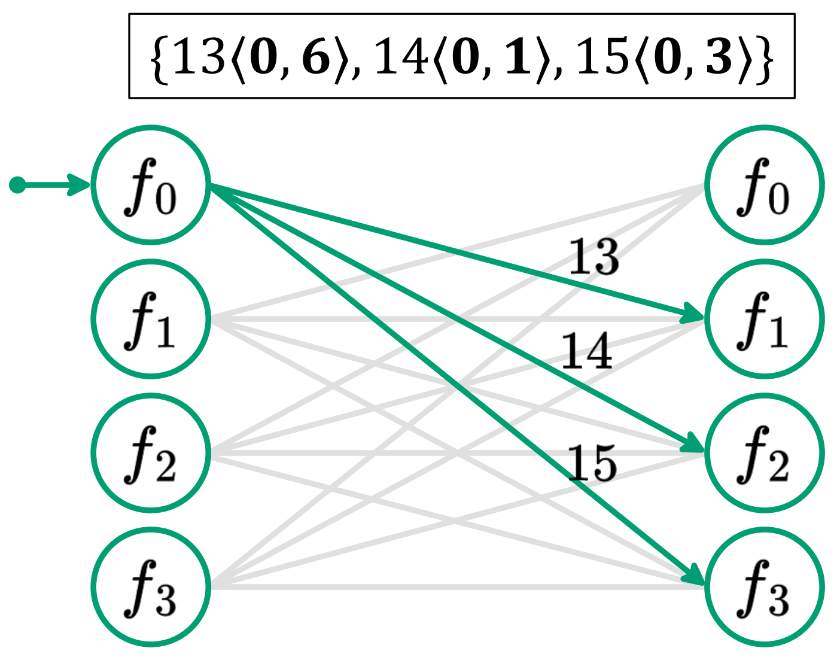
- **Plan**: An **array of integer variables** with a fixed plan length.
- **Task Variables**: **Regular Constraint** for each task variable to enforce their automaton on the plan.

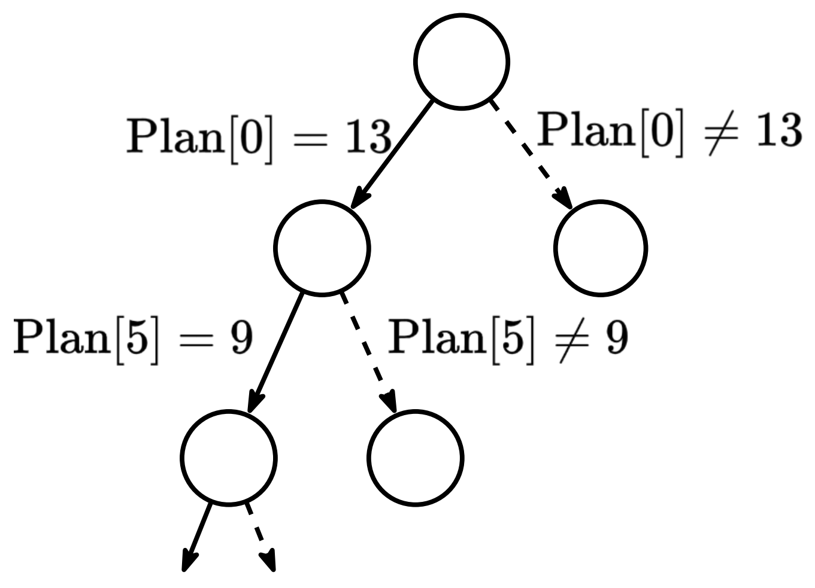
Time-unfolded automaton for the Regular Constraint



Search

Solve using **MiniCPBP** solver with **belief propagation** and the **maximum marginal** branching heuristic on the actions of the plan.


$\{13\langle \mathbf{0}, \mathbf{6}\rangle, 14\langle \mathbf{0}, \mathbf{1}\rangle, 15\langle \mathbf{0}, \mathbf{3}\rangle\}$
f_0
f_1
f_2
f_3
13
14
15
f_0
f_1
f_2
f_3

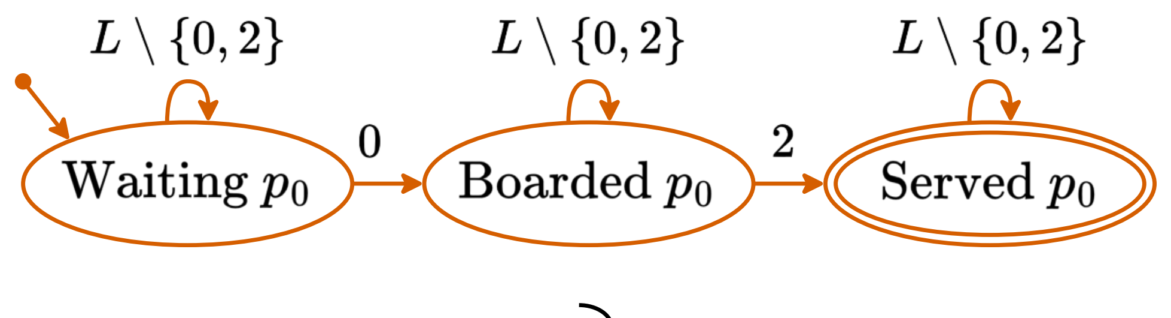


Preliminary Results

Planning Domain	Outcome	*NoGroup*	*Group*	*Group+Prune*	*Manual*
Miconic (150)	solved opt.	38	42	41	49
	out of memory	60	28	28	0
	out of time	52	80	81	101
Scanalyzer (41)	solved opt. (solved)	5 (11)	5 (15)	5 (21)	(33) 33
	out of memory	18	18	12	0
	out of time	18	18	24	8

Future Works

- Merging automatons



- Action Space reduction

up(f_2, f_3), up(f_1, f_3), up(f_0, f_3) } up(f_3)

- New constraints (landmarks, operator counting, ...)