

An Automata-Based Constraint Programming Framework for Optimal Classical Planning

Damien Van Meerbeeck¹, Arnaud Lequen¹, Gilles Pesant², Jendrik Seipp¹

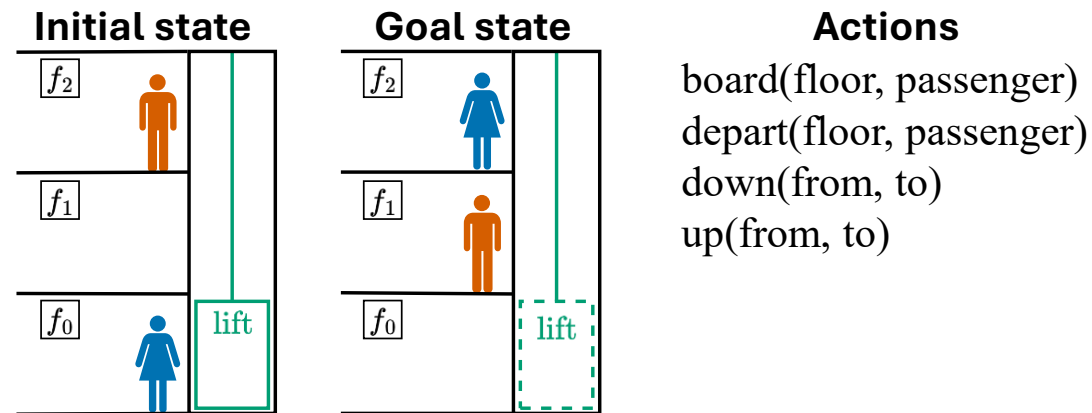
¹ Machine Reasoning Lab, Linköping University, Sweden

² Quosséça Lab, Polytechnique Montréal, Canada

CP 2026, 22 July 2026

Optimal Classical Planning

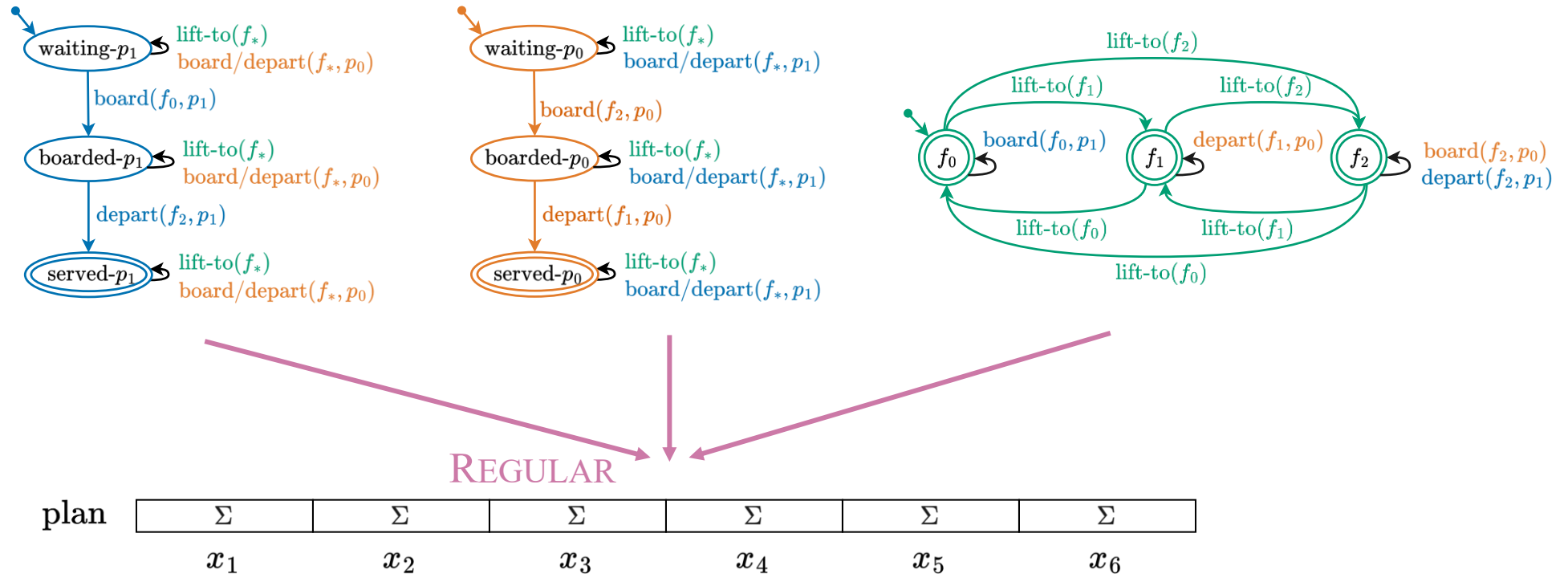
- **Optimal Classical planning** is the challenge of finding a shortest sequence of actions (**plan**) transforming a world from an **initial state** to a **goal state**.



Plan: board(p_1 , f_0) $\rightarrow$ up(f_0 , f_2) $\rightarrow$ depart(f_2 , p_1) $\rightarrow$ board(f_2 , p_0) $\rightarrow$ down(f_2 , f_1) $\rightarrow$ depart(f_1 , p_0)

Classical Planning as CP with automata

- Previous work [1] showed that planning problems can be represented as factored set of automata that can model planning problems in CP through REGULAR constraints.



Motivation

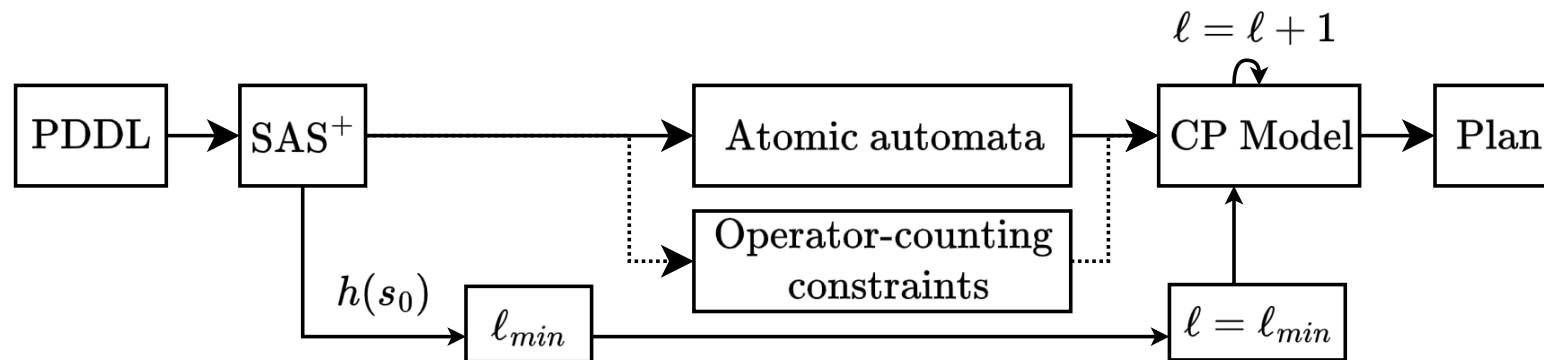
State-of-the-art planners use heuristics state-space search, but CP has some benefits:

- **Compact representation** to better scale for large planning tasks.
- **Non-sequential generation** of the plan.
- Use of declarative knowledge for **redundant constraints**.

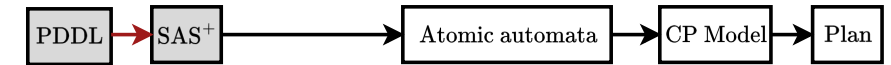
So far, the automata have been carefully handcrafted

Our contribution

- **End-to-end framework** So far, the automata have been carefully handcrafted from tasks in PDDL, we automate their creation from a standard PDDL description and use them for the modelling of a CSP.
- **Redundant constraints** Thanks to this modular framework we can also enhance the model with additional information in the form of redundant constraints.



PDDL to SAS⁺



PDDL
(lifted)



Predicates

(lift-at $f_?$)
(boarded $p_?$)
(served $p_?$)

Actions schemas

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

State Variables

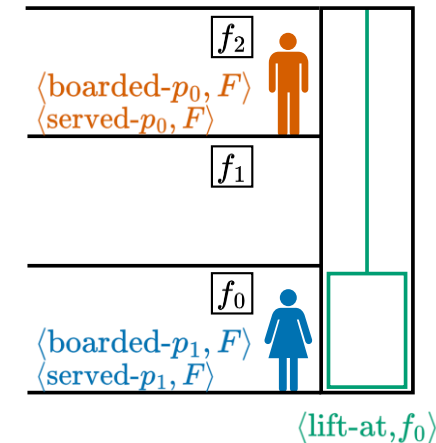
lift-at: $\{f_0, f_1, f_2\}$
boarded- p_0 : $\{T, F\}$
served- p_0 : $\{T, F\}$
boarded- p_1 : $\{T, F\}$
served- p_1 : $\{T, F\}$

SAS⁺
(grounded)

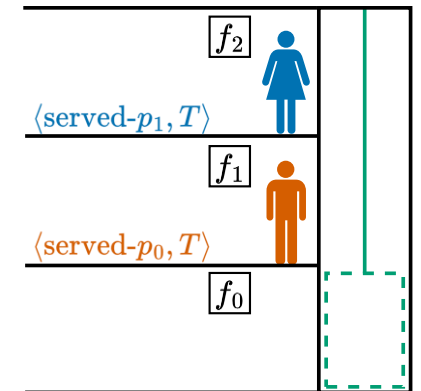
Actions

up(f_0, f_1) board(f_2, p_0)
up(f_0, f_2) depart(f_1, p_0)
up(f_1, f_2) board(f_0, p_1)
down(f_1, f_0) depart(f_2, p_1)
down(f_2, f_0)
down(f_2, f_1)

Initial state



Goal state





SAS⁺ to automata

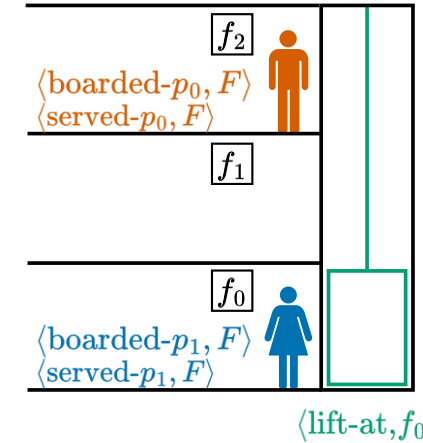
State Variables

lift-at: $\{f_0, f_1, f_2\}$
 boarded- p_0 : $\{T, F\}$
 served- p_0 : $\{T, F\}$
 boarded- p_1 : $\{T, F\}$
 served- p_1 : $\{T, F\}$

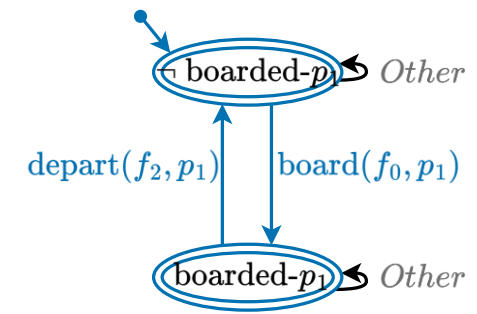
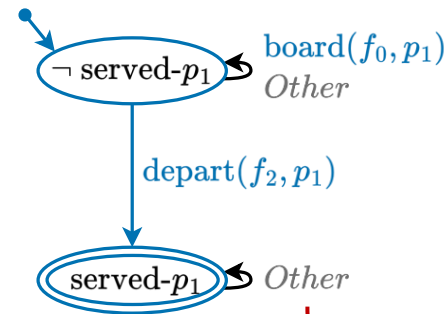
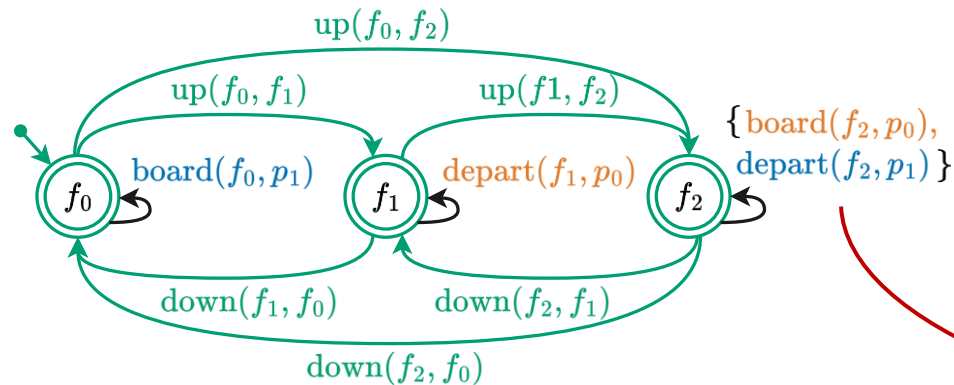
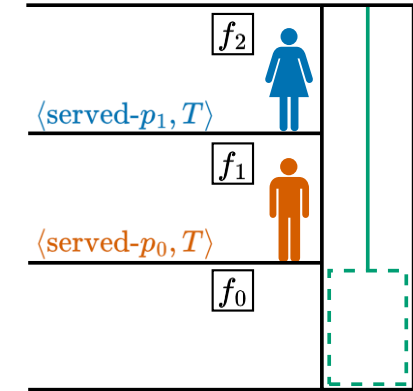
Actions

up(f_0, f_1) board(f_2, p_0)
 up(f_0, f_2) depart(f_1, p_0)
 up(f_1, f_2) board(f_0, p_1)
 down(f_1, f_0) depart(f_2, p_1)
 down(f_2, f_0)
 down(f_2, f_1)

Initial state

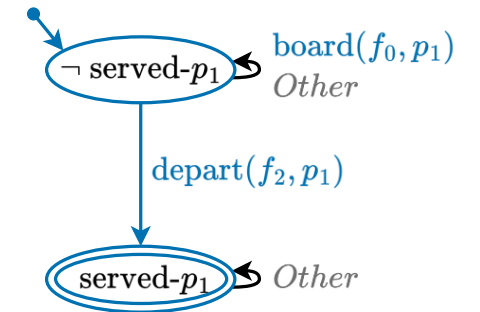
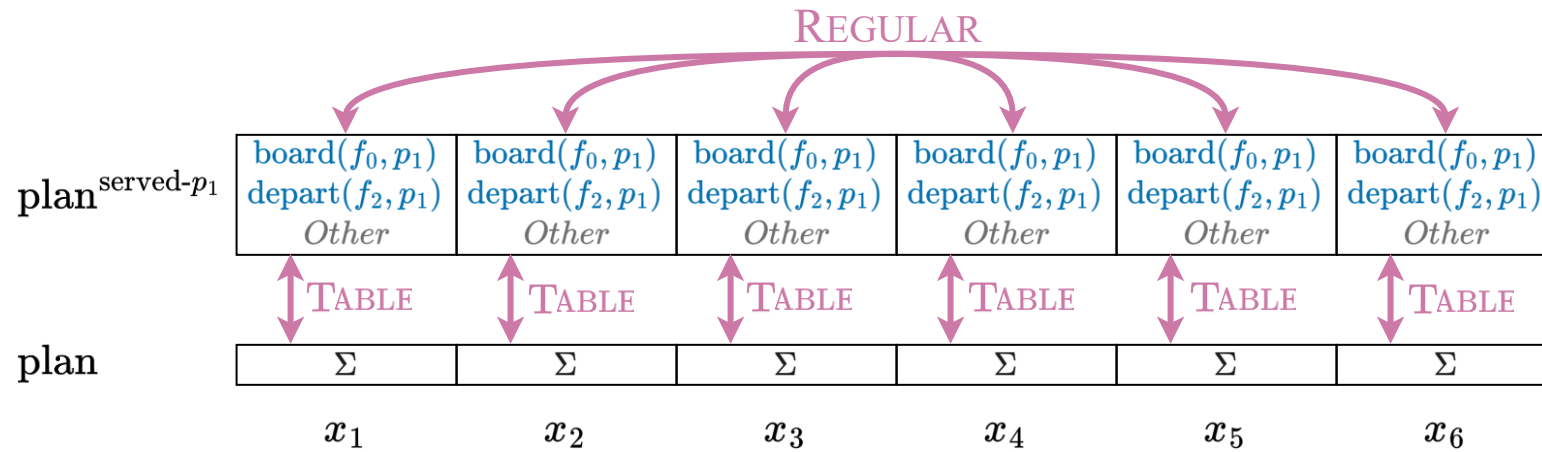


Goal state



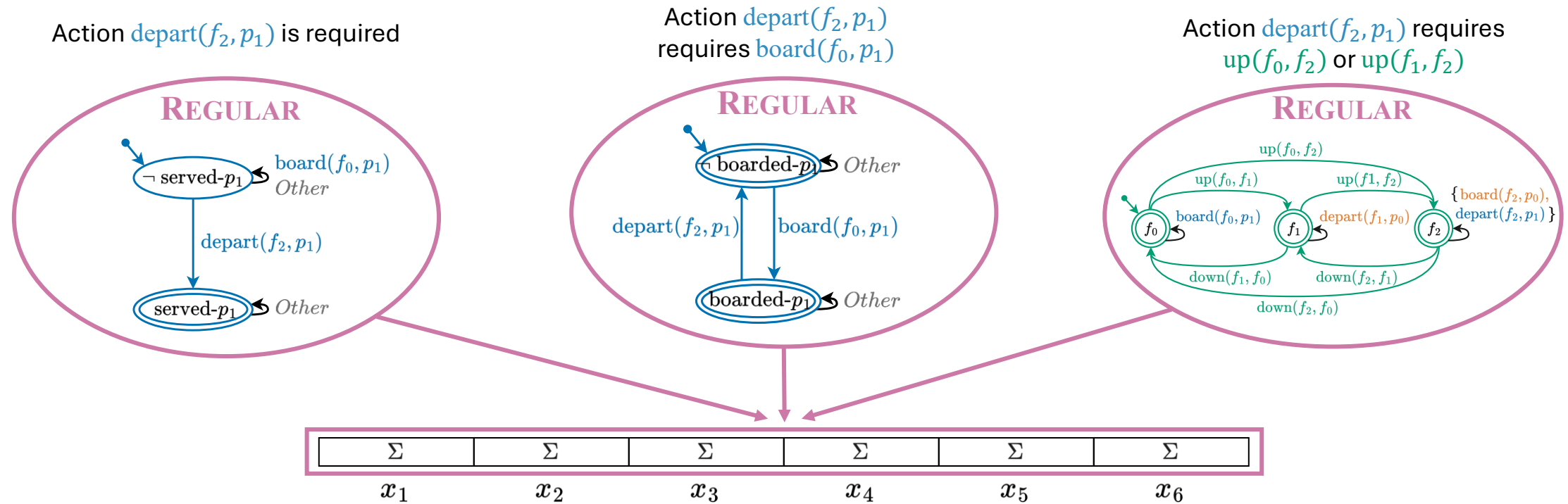
Reduced alphabet

CP model



CP model – with imperfections

- Compact representation but localizes information.



Operator-counting constraints

- **Operator-counting constraints** are linear systems of inequalities that express knowledge about the number of occurrences of actions in all plans.
- **Landmark Constraints (LMC)**
At least one of these actions must be part of the plan.
 - $x_{\text{depart}(f_2, p_1)} \geq 1$
 - $x_{\text{board}(f_0, p_1)} \geq 1$
 - $x_{\text{up}(f_0, f_2)} + x_{\text{up}(f_1, f_2)} \geq 1$
- **Net Change Constraints (NCC)**
If an atom is in the goal but not the initial state, any plan must establish it once more than it deletes it.
 - $x_{\text{up}(f_0, f_2)} + x_{\text{up}(f_1, f_2)} - x_{\text{down}(f_2, f_0)} - x_{\text{down}(f_2, f_1)} \geq 0$

Operator-counting to redundant constraints

- To reduce the number of variables, we count the occurrences of actions that appear together with the same sign in all opc and link the resulting variable and the plan with **AMONG** constraints.

Operator-counting constraints

$$x_{\text{depart}(f_2, p_1)} \geq 1$$

$$x_{\text{board}(f_0, p_1)} \geq 1$$

$$x_{\text{up}(f_0, f_2)} + x_{\text{up}(f_1, f_2)} \geq 1$$

$$x_{\text{up}(f_0, f_2)} + x_{\text{up}(f_1, f_2)} - x_{\text{down}(f_2, f_0)} - x_{\text{down}(f_2, f_1)} \geq 0$$

CP constraints

$$Occ_{\{\text{depart}(f_2, p_1)\}} \geq 1$$

$$Occ_{\{\text{board}(f_0, p_1)\}} \geq 1$$

$$Occ_{\{\text{up}(f_0, f_2), \text{up}(f_1, f_2)\}} \geq 1$$

$$Occ_{\{\text{up}(f_0, f_2), \text{up}(f_1, f_2)\}} \geq Occ_{\{\text{down}(f_2, f_0), \text{down}(f_2, f_1)\}}$$

$$Occ_{\{\text{depart}(f_2, p_1)\}} + Occ_{\{\text{board}(f_0, p_1)\}} + Occ_{\{\text{up}(f_0, f_2), \text{up}(f_1, f_2)\}} + Occ_{\{\text{down}(f_2, f_0), \text{down}(f_2, f_1)\}} + Occ_{\text{Remaining}} = l$$

Partition refinement

CP variables

$$Occ_{\{\text{depart}(f_2, p_1)\}}$$

$$Occ_{\{\text{board}(f_0, p_1)\}}$$

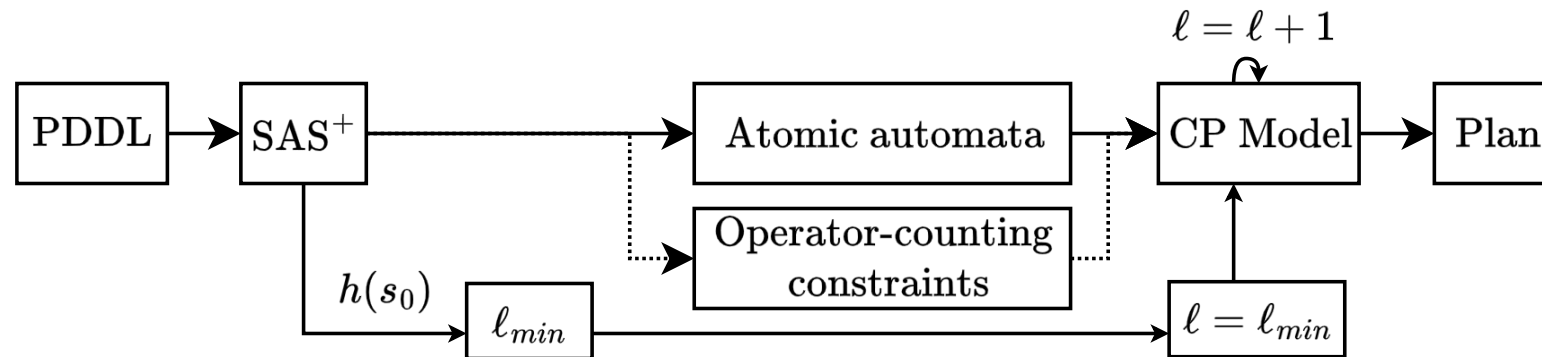
$$Occ_{\{\text{up}(f_0, f_2), \text{up}(f_1, f_2)\}}$$

$$Occ_{\{\text{down}(f_2, f_0), \text{down}(f_2, f_1)\}}$$

$$Occ_{\text{Remaining}}$$

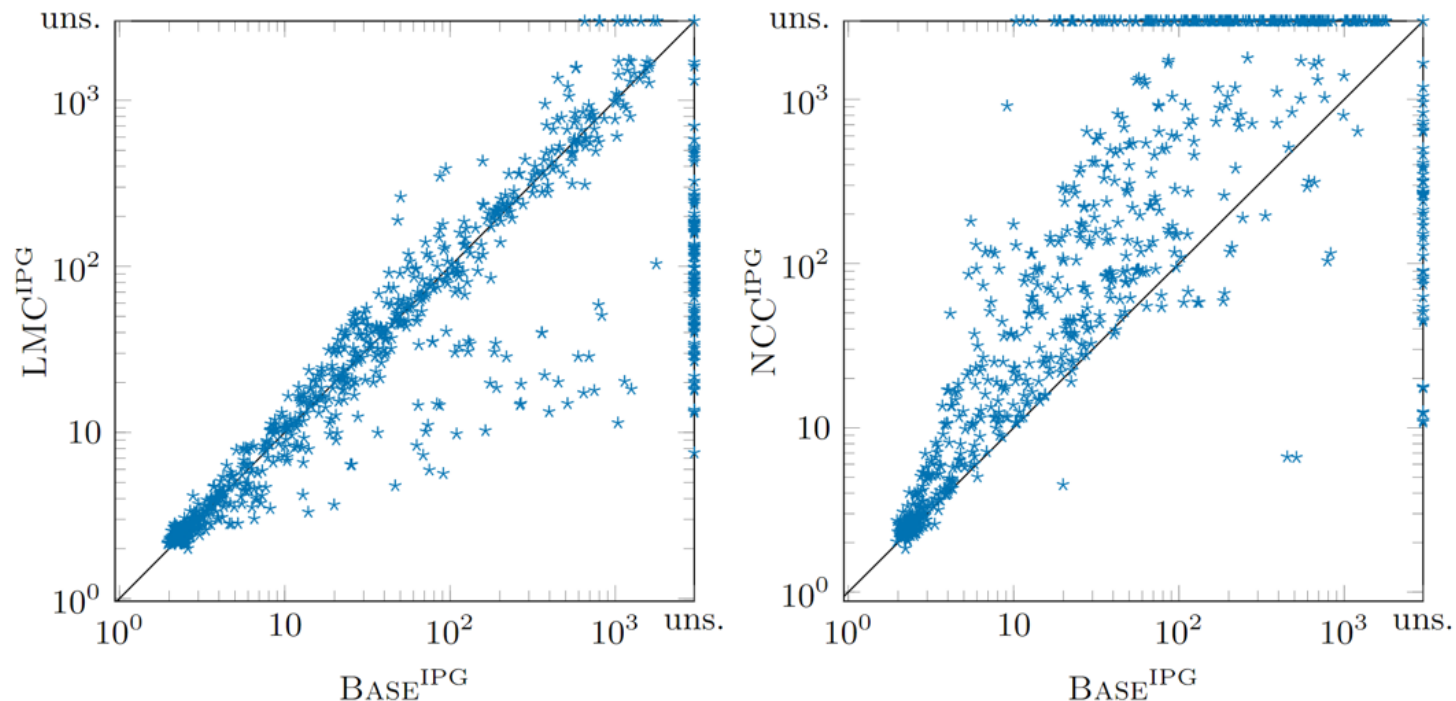
Experimental setup

- CP solver: CP-SAT
- Benchmark: 47 planning domains of the IPC 1998–2023 (1786 instances)
- Runtime: 30 min
- Memory limit: 8GB



Results – Operator-counting constraints

- LMC solves the most instances.
- NCC runs out of memory more often but still outperforms BASE over some domains.
- Operator-counting constraints brings valuable information but add computational overhead.



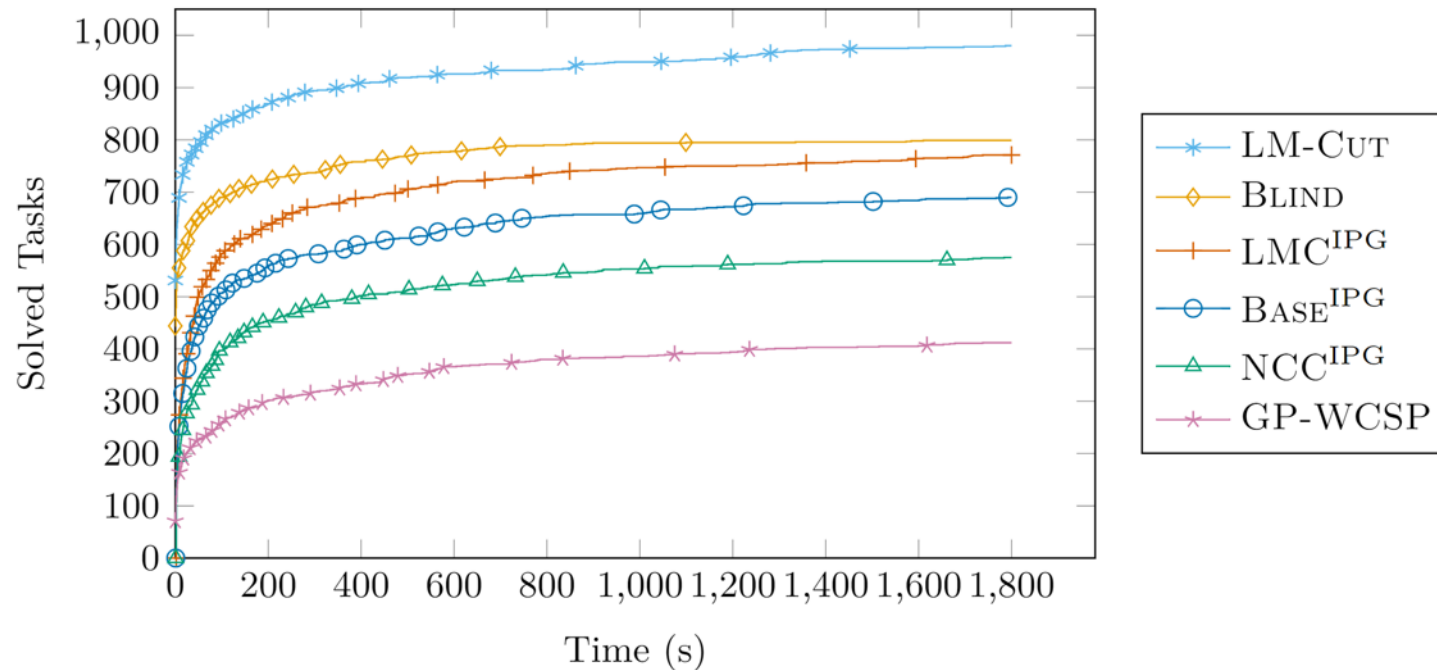
Results – Operator-counting constraints

- LMC solves the most instances.
- NCC runs out of memory more often but still outperforms BASE over some domains.
- Operator-counting constraints brings valuable information but add computational overhead.

	<i>DataNetwork (20)</i>	<i>Logistics (63)</i>	<i>Miconic (150)</i>	<i>ParcPrinter (50)</i>	<i>Pegsol (50)</i>	<i>Rovers (40)</i>	<i>Satellite (36)</i>	<i>Scanalyzer (50)</i>	<i>Tetris (17)</i>	<i>Trucks (30)</i>	<i>Woodworking (50)</i>	<i>Zenotravel (20)</i>	<i>Other (1210)</i>	<i>Total (1786)</i>
BASE ^{IPG}	17	13	58	30	38	4	7	23	5	5	35	9	446	690
LMC ^{IPG}	15	28	102	34	38	6	9	25	5	5	50	11	443	771
NCC ^{IPG}	12	13	38	50	42	5	5	11	9	8	45	8	323	569

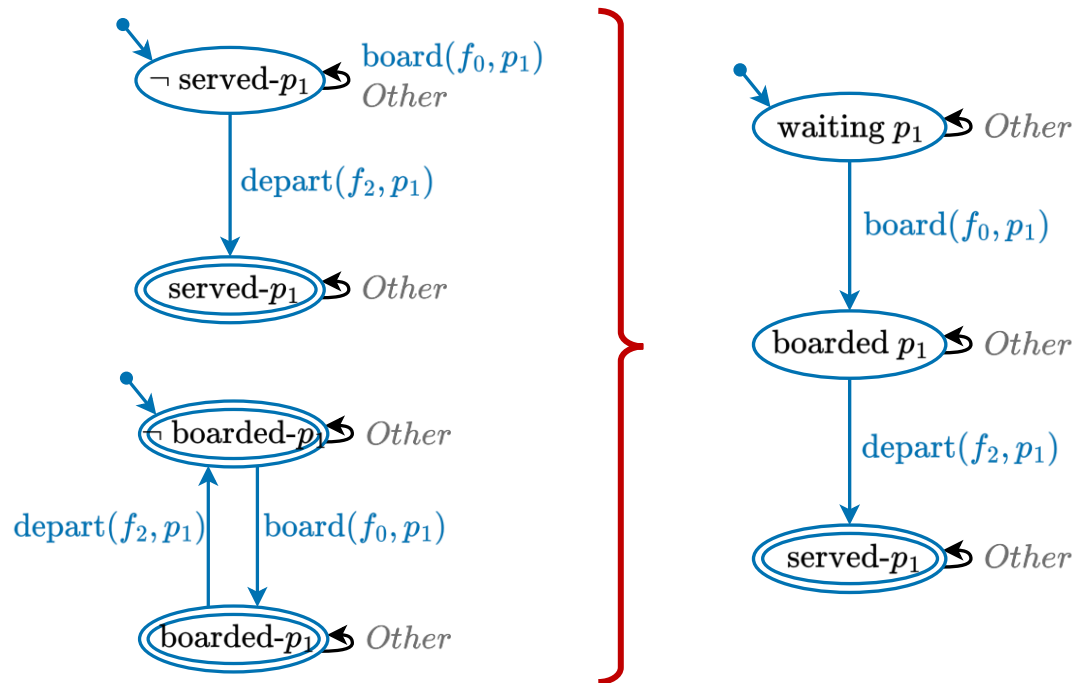
Results – Baselines

- Approach beats state-of-the-art CP-based optimal classical planner (GP–WCSP).
- Does not yet reach the level of state-space search planners (BLIND, LM–CUT).

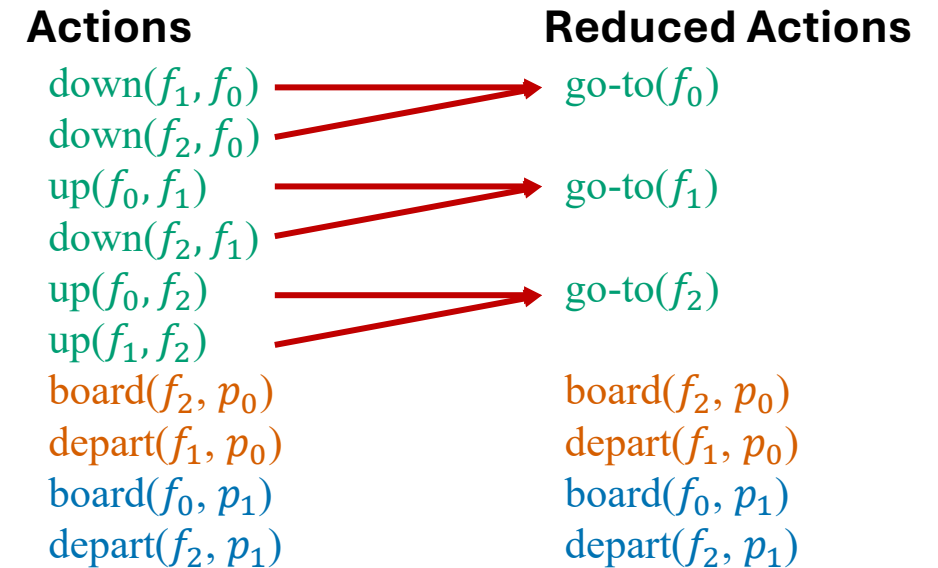


Future works – Automata Reformulation

Merging



Label reduction



Conclusion

- End-to-end Automata-based framework to solve planning tasks using CP.
- CP model for planning can be extended with additional constraints.
- Counting actions for CP with the help of operator-counting constraints.

