An Automata-Based Constraint Programming Framework for Optimal Classical Planning

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

[1] Machine Reasoning Lab, Linköping University, Sweden
[2] Quosséça Lab, Polytechnique Montréal, Canada

CASP:ER Workshop, 29 June 2026

LINKÖPING UNIVERSITY

Machine Reasoning Lab

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

POLYTECHNIQUE MONTRÉAL

Motivation

- **Classical planning** is the challenge of finding a sequence of action (**plan**) transforming a world from **initial state** to a **goal state**.
- State of the art planners use state-space search with heuristics.

- **Constraint Programming (CP):** Paradigm for solving combinatorial problems.
- Benefits of CP for classical planning:
  - Use declarative knowledge for constraints (Landmarks, Operator Counting, …).
  - Generate the plan in a non-sequential way.

Classical Planning

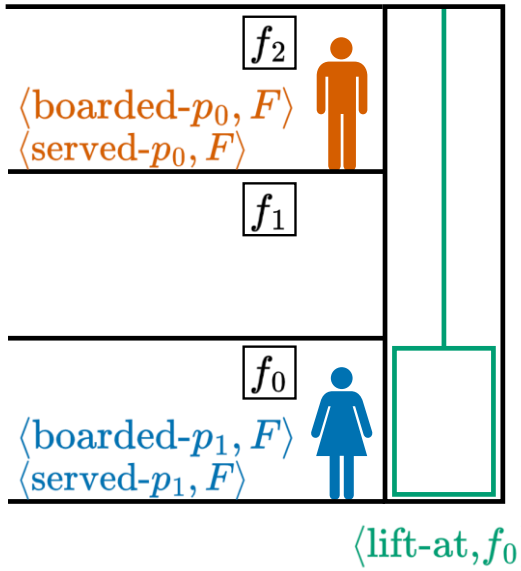
Predicates

boarded-p_0 lift-at f_0
served-p_0 lift-at f_1
boarded-p_1 lift-at f_2
served-p_1

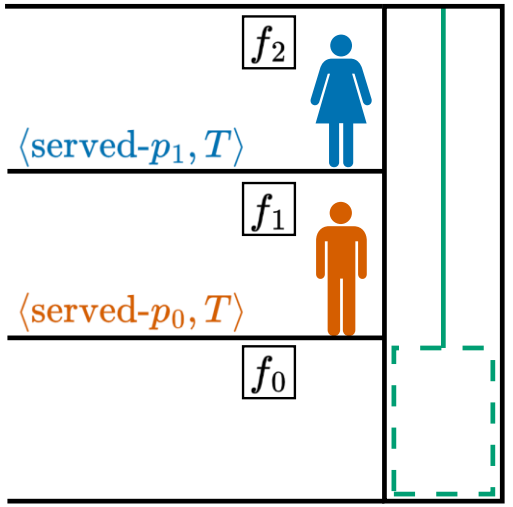
Actions

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

Initial state



Goal state

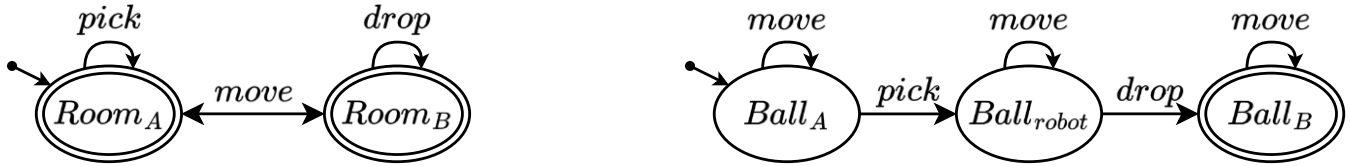


Plan

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

Classical Planning as CP with automata

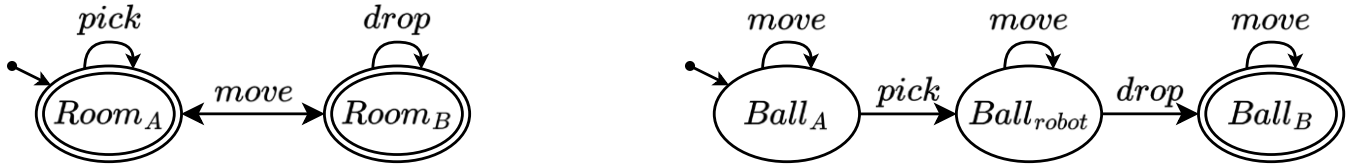
- Planning tasks can be represented as a set of automata and, trough REGULAR constraint, we can model the dynamics of the task.



Plan				
	pick *drop* *move*	*pick* *drop* *move*	*pick* *drop* *move*	*pick* *drop* *move*

Classical Planning as CP with automata

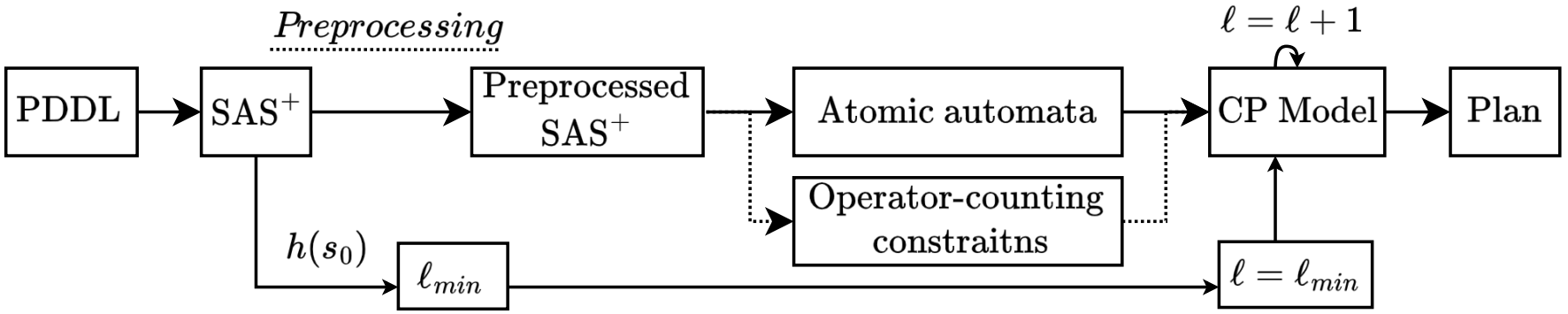
- Planning tasks can be represented as a set of automata and, trough REGULAR constraint, we can model the dynamics of the task.



Plan				
	pick ~~*drop*~~ *move*	*pick* *drop* *move*	*pick* *drop* *move*	~~*pick*~~ *drop* *move*

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

End-to-end Framework



SAS$^+$

State variables V

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 A

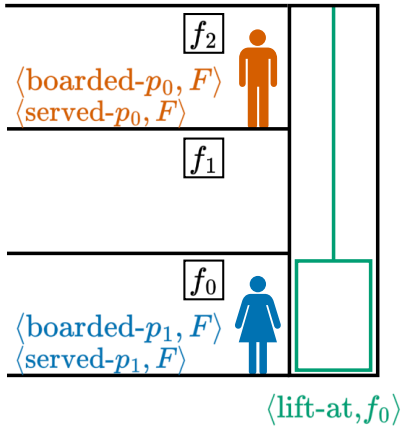
up(f_0, f_1)
up(f_0, f_2)
up(f_1, f_2)

down(f_1, f_0)
down(f_2, f_0)
down(f_2, f_1)

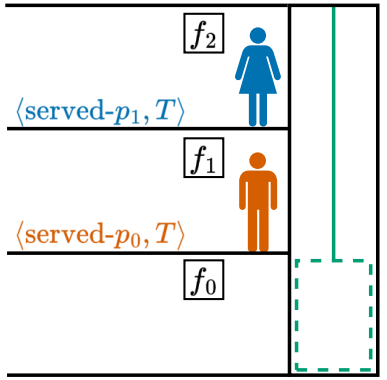
board(f_2, p_0)
depart(f_1, p_0)

board(f_0, p_1)
depart(f_2, p_1)

Initial state s_I



Goal state G



SAS^+ to Automata

State variables V

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 A

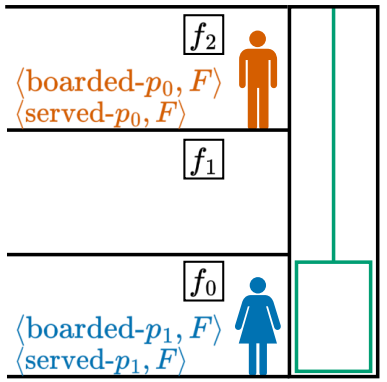
$\text{up}(f_0, f_1)$
$\text{up}(f_0, f_2)$
$\text{up}(f_1, f_2)$

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

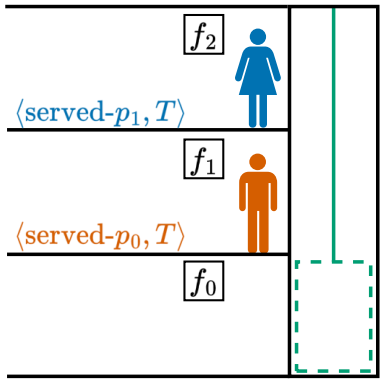
$\text{board}(f_2, p_0)$
$\text{depart}(f_1, p_0)$

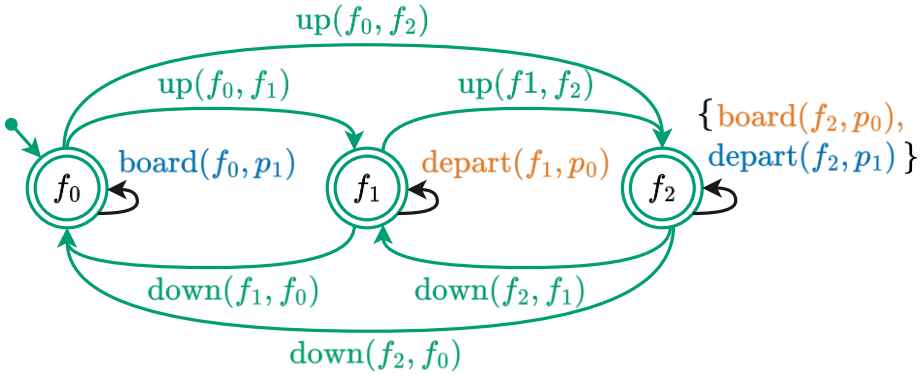
$\text{board}(f_0, p_1)$
$\text{depart}(f_2, p_1)$

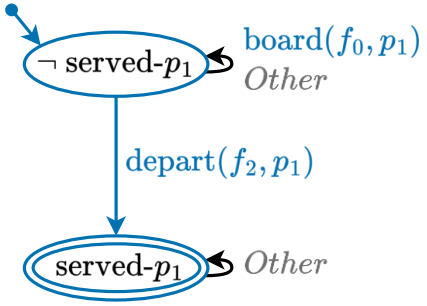
Initial state s_I

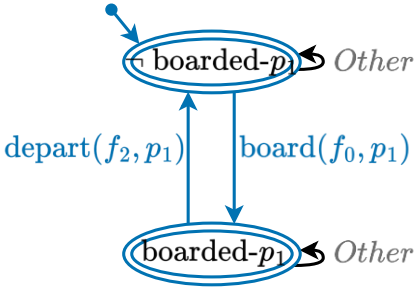


Goal state G









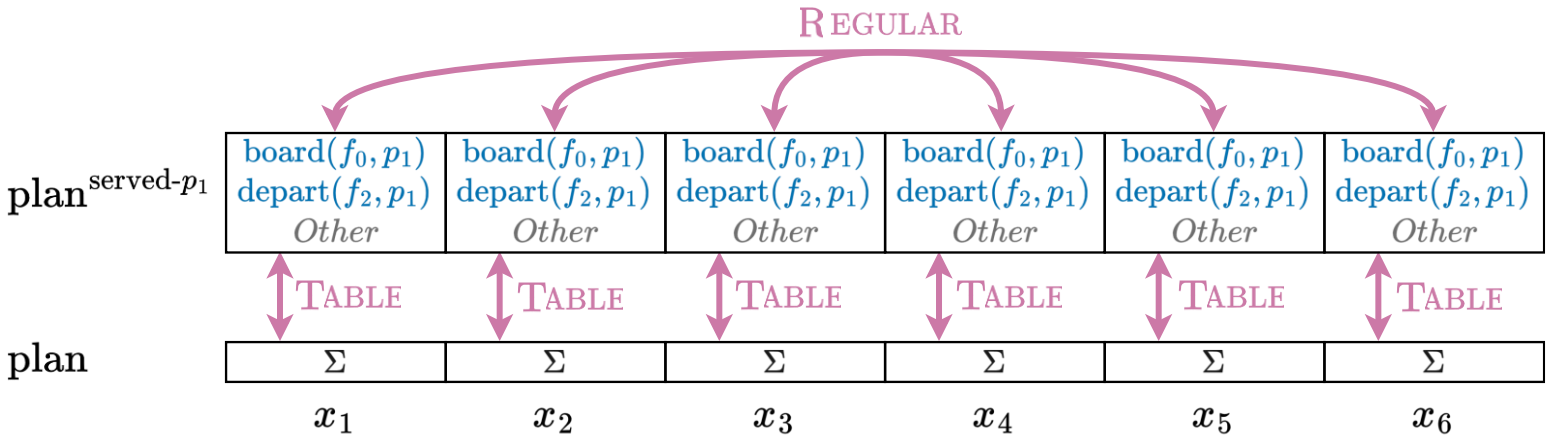
CP model

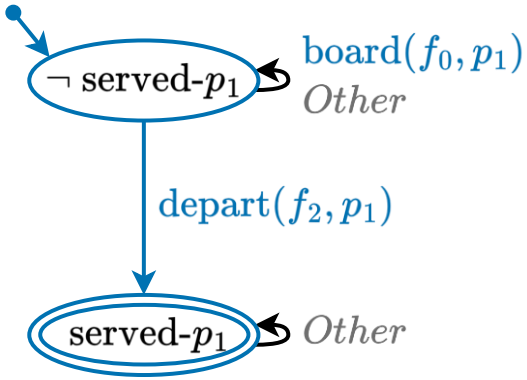
Variables

- plan : array of integer variables with alphabet .
- plan^{A_i} : plan for each automaton with reduced alphabet.

Constraints

- REGULAR over plan^{A_i} for each automaton A_i.
- TABLE constraints for each action of each individual plan^{A_i} to the plan.

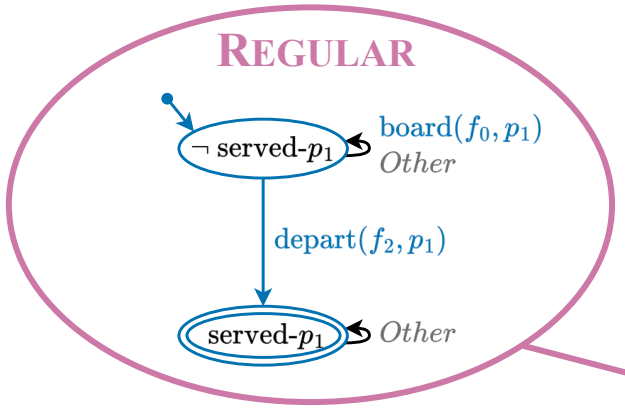




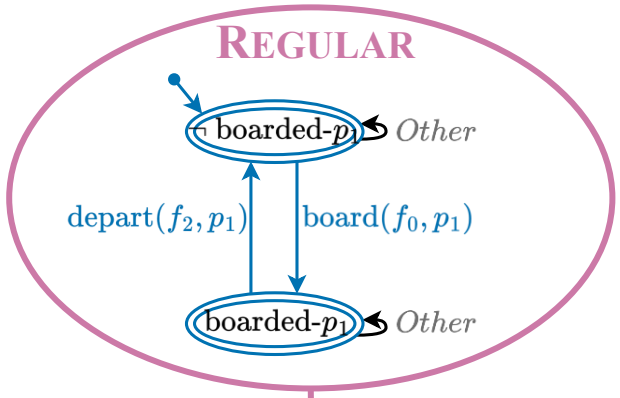
Factored automata – Localized information

- Compact representation but localizes information that could be used for pruning.

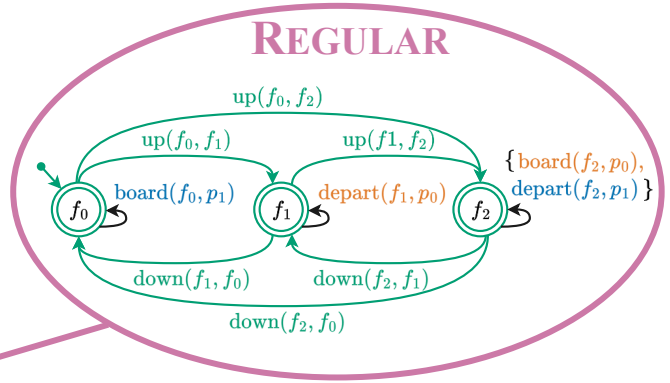
Action $\text{depart}(f_2, p_1)$ is required

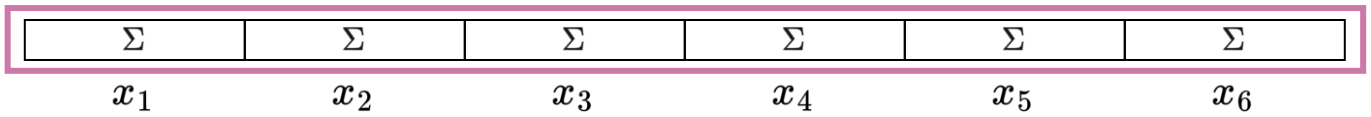


Action $\text{depart}(f_2, p_1)$ requires $\text{board}(f_0, p_1)$



Action $\text{depart}(f_2, p_1)$ requires $\text{up}(f_0, f_2)$ or $\text{up}(f_1, f_2)$





Operator-counting constraints

- In the planning literature, **operator-counting constraints** are linear systems of inequalities that express knowledge about the number of occurrences of action in all plans.
- **Landmark Constraints** (LMC)
  These constraints are formed of disjunctive action landmark, which give an intuition about actions that needs to 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)
  also called state-equation constraints, they give the intuition that if an atom is in the goal but not the initial state, then any plan must establish it once more than it deletes it.
  - $x_{\text{up}(f_0,f_2)} + x_{\text{up}(f_1,f_2)} - x_{\text{down}(f_2,f_0)} - x_{\text{down}(f_2,f_1)} \geq 0$
  - $x_{\text{board}(f_1,p_0)} - x_{\text{depart}(f_1,p_0)} \geq 0$

Operator-counting to redundant constraints

- To reduce the number of occurrence to keep track we can count the number of actions that occur together with the same sign.

Operator-counting constraints

$x_{depart(f_2,p_1)} \geq 0$

$x_{board(f_0,p_1)} \geq 0$

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

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

Partition refinement →

CP variables

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

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

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

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

$Occ_{Remaining}$

CP constraints

$Occ_{\{\text{depart}(f_2,p_1)\}} \geq 0$

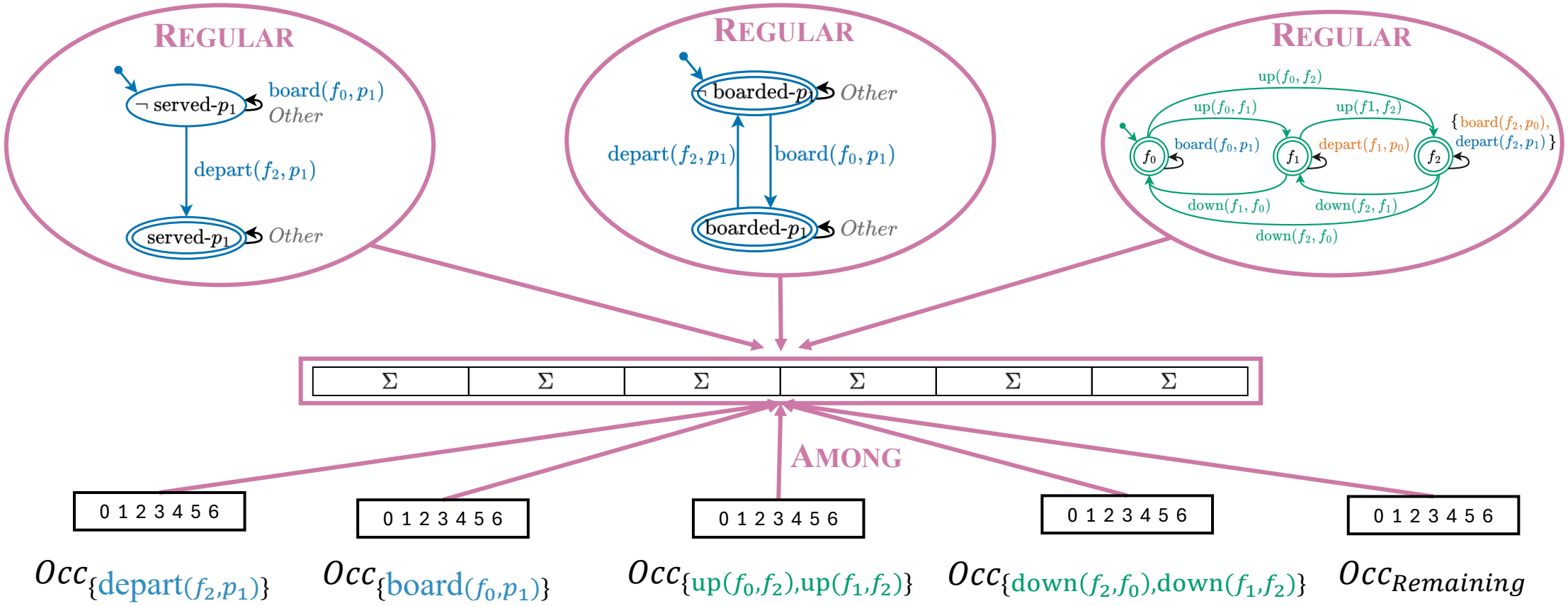
$Occ_{\{\text{board}(f_0,p_1)\}} \geq 0$

$Occ_{\{\text{up}(f_0,f_2),\text{up}(f_1,f_2)\}} \geq 0$

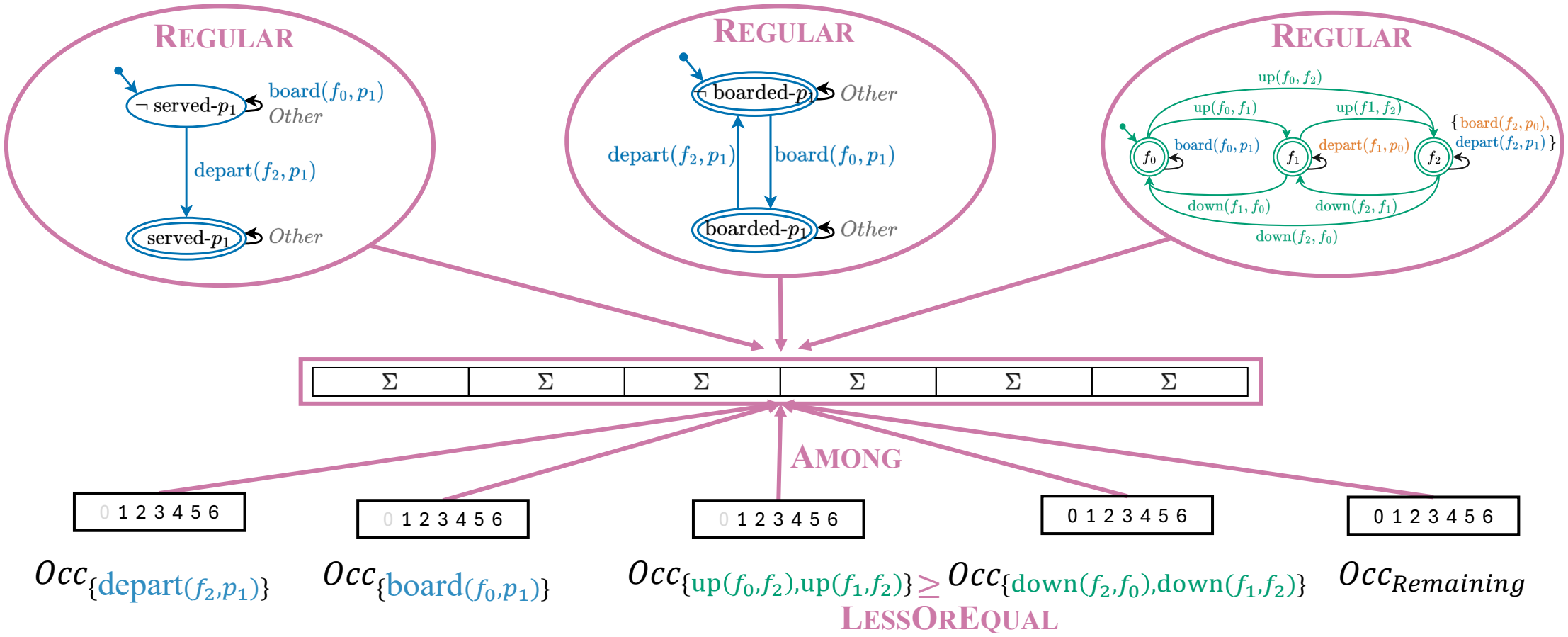
$Occ_{\{\text{up}(f_0,f_2),\text{up}(f_1,f_2)\}} - Occ_{\{\text{down}(f_2,f_0),\text{down}(f_1,f_2)\}} \geq 0$

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

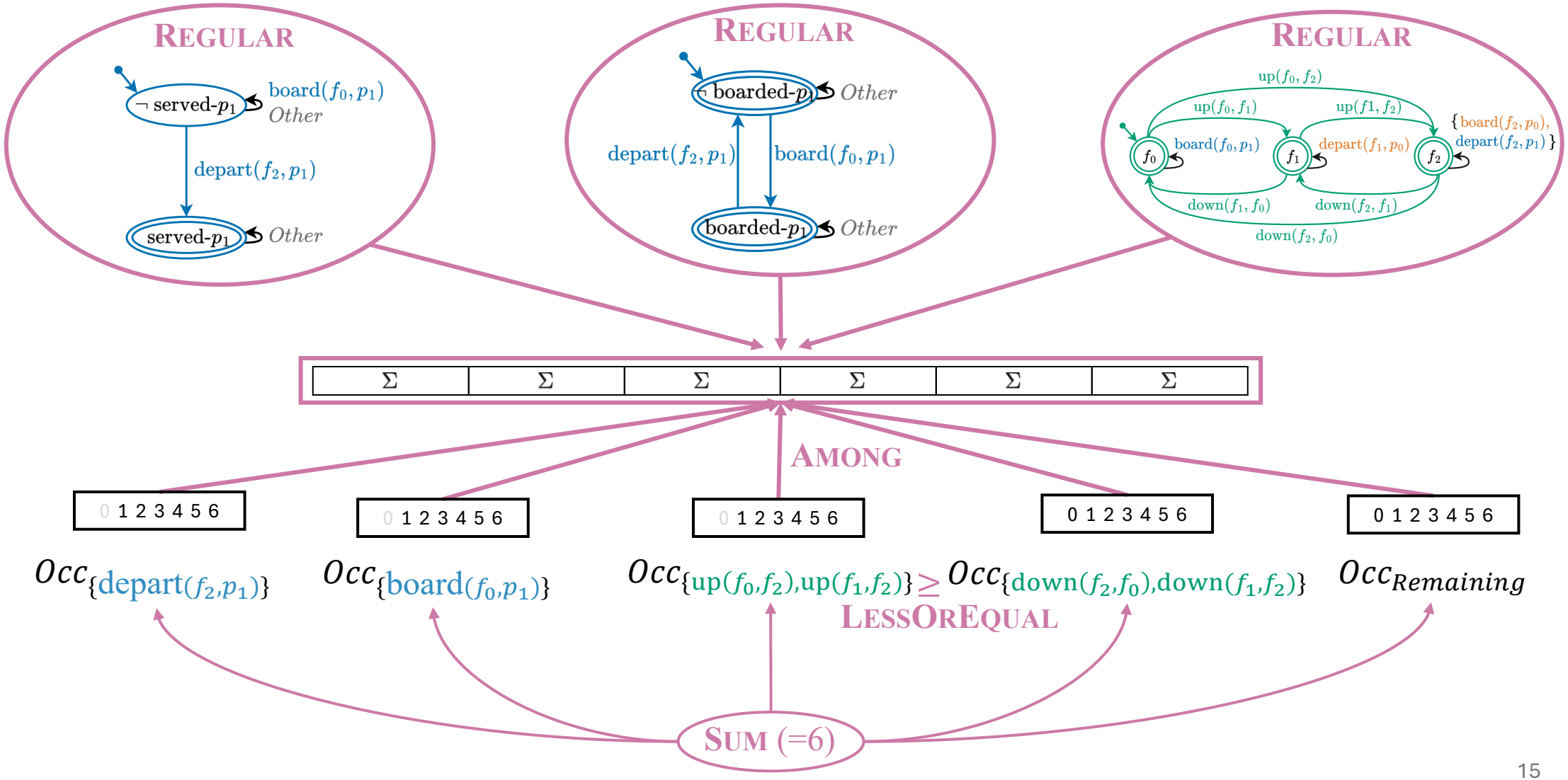
CP model with operator counting constrains



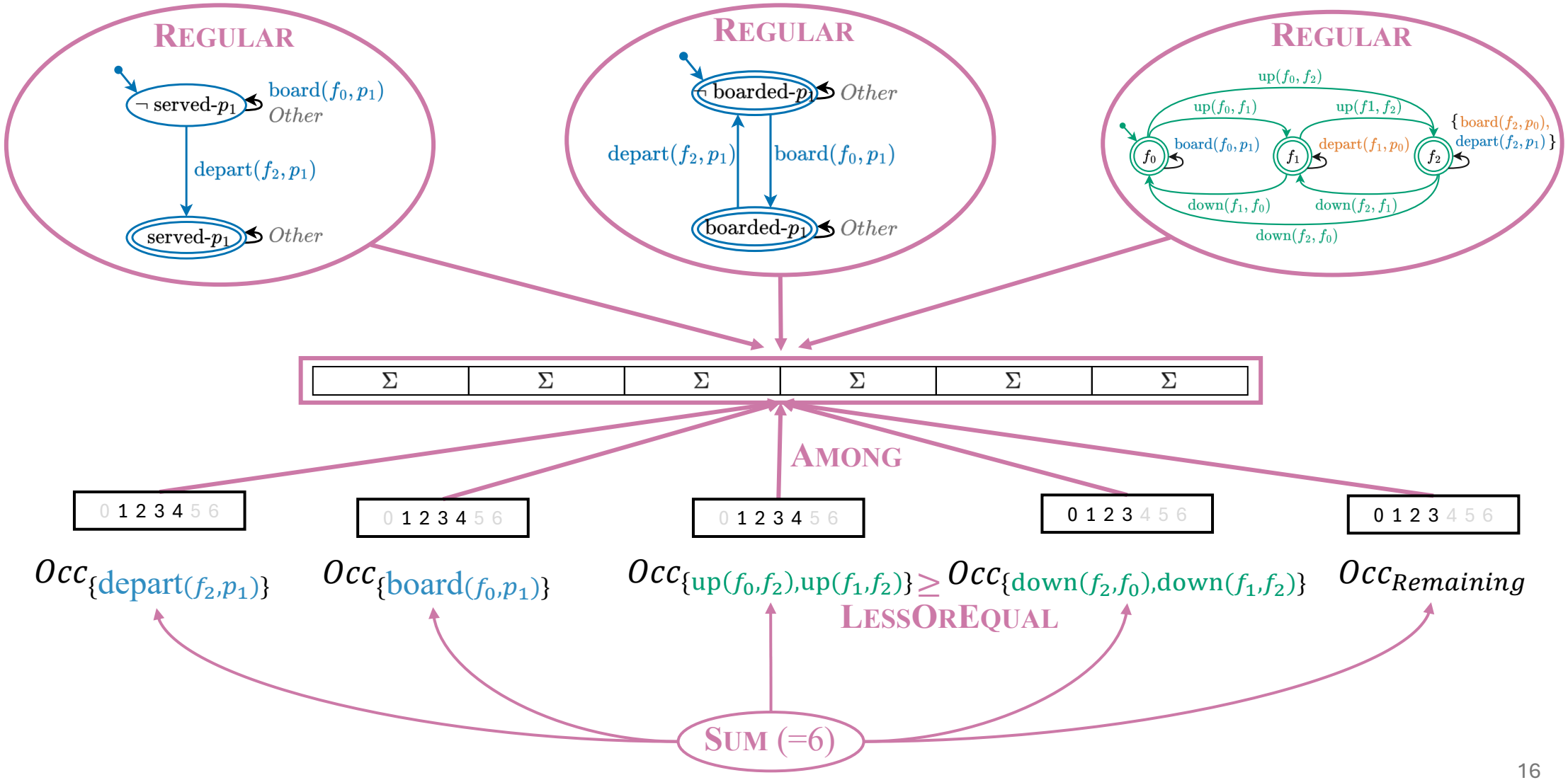
CP model with operator counting constrains



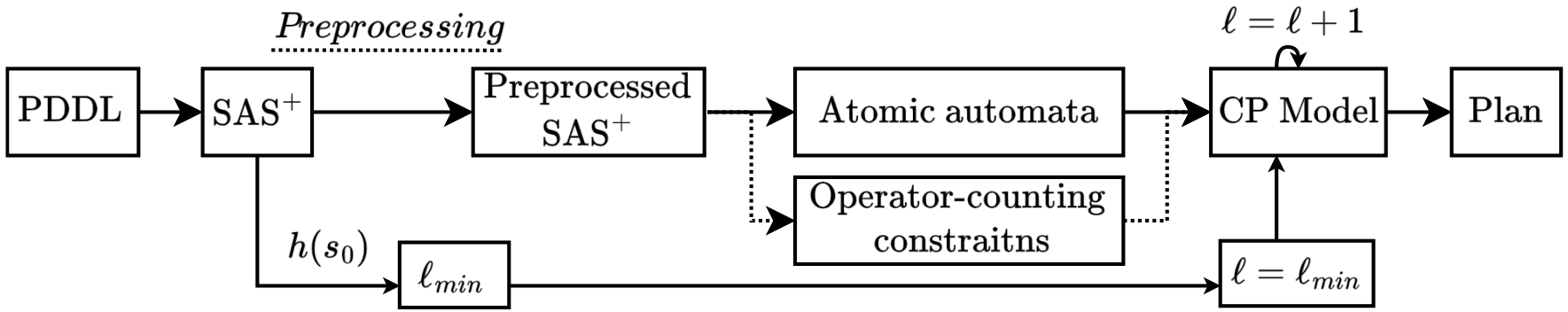
CP model with operator counting constrains



CP model with operator counting constrains



Experimental setup



Results – Preprocessing

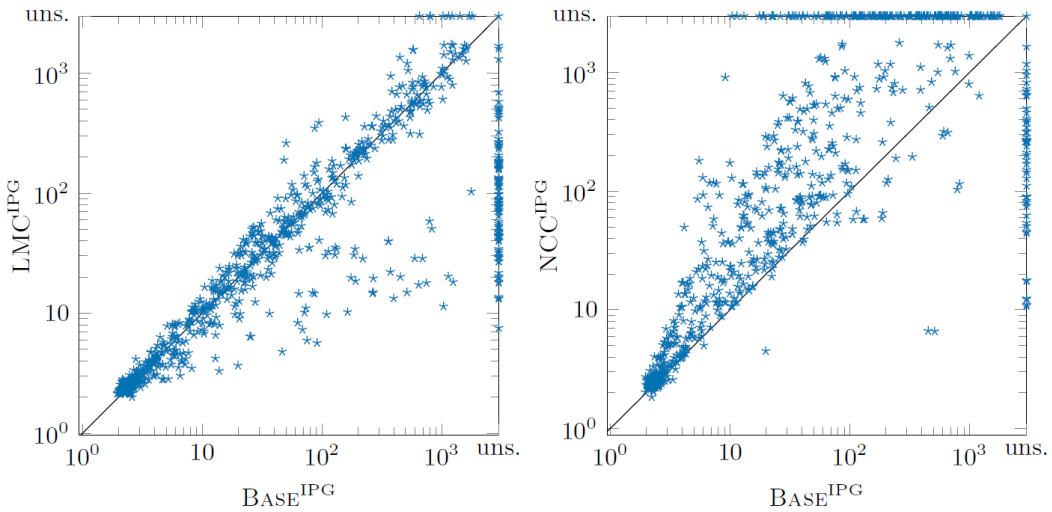
	BASE	BASE^{IP}	BASE^{IG}	BASE^{IPG}	Total (1786)
BASE	–	2	5	0	675
BASE^{IP}	**2**	–	6	0	676
BASE^{IG}	**6**	**7**	–	1	683
BASE^{IPG}	**7**	**7**	**7**	–	**690**

Comparison of different preprocessing techniques. We compare our Base model to models enriched with implied preconditions ($-^{IP}$), implied goals ($-^{IG}$), or both ($-^{IPG}$). The first four columns compare the different preprocessing techniques on a per-domain basis, where each cell holds the number of domains for which the approach in the row performed better than the approach in the column. The last column reports the number of tasks solved.

Results – Operator-Counting

	DataNetwork (20)	Logistics (63)	Miconic (150)	ParcPrinter (50)	Pegsol (50)	Rovers (40)	Satellite (36)	Scanalyzer (50)	Tetris (17)	Trucks (30)	Woodworking (50)	Zenotravel (20)	*Other* (1210)	Total (1786)
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

Number of solved tasks per domain for BASE^{IPG} and its two extensions with additional operator-counting constraints, LMC^{IPG} and NCC^{IPG}. Domain sizes (number of tasks) are shownin parentheses. A domain is detailed in the table if the absolute difference between LMC^{IPG} and BASE^{IPG} is at least 2 or if the difference between NCC^{IPG} and BASE^{IPG} is more than 2. The remaining domains are compiled in the Other column.

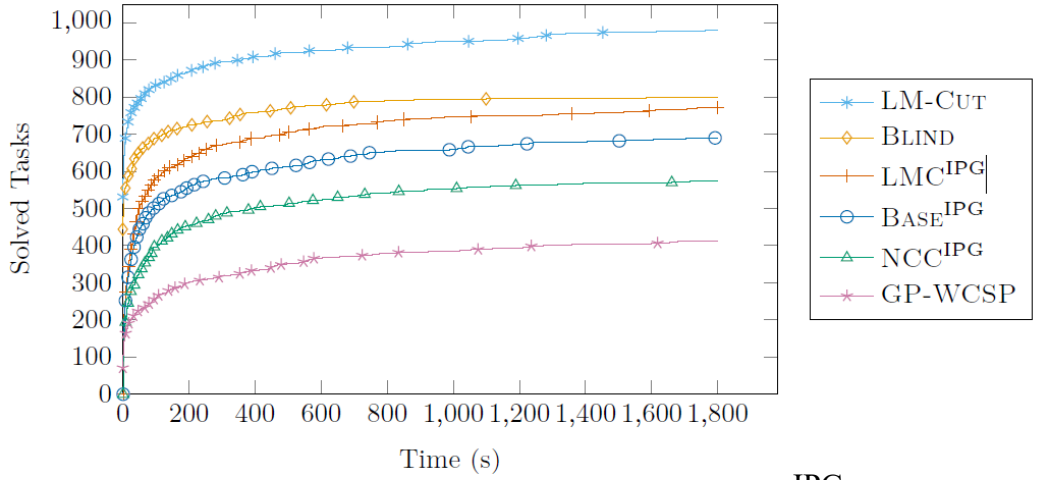


Comparison of overall runtime for BASE^{IPG} against LMC^{IPG} and NCCIPG. BASE^{IPG} is our base model, LMC^{IPG} is our model with landmark constraints, and NCCIPG is our model with net change constraints. All models are enriched with implied preconditions and goals. Points on the "uns." axes indicate runs that hit the time or memory limit.

Results – Baselines

	GP-WCSP	BASE^{IPG}	LMC^{IPG}	BLIND	LM-CUT	Total (1786)
GP-WCSP	–	3	2	3	1	401
BASE^{IPG}	**39**	–	7	13	7	690
LMC^{IPG}	**40**	**11**	–	15	10	771
BLIND	**41**	**26**	**23**	–	15	784
LM-CUT	**44**	**32**	**30**	**26**	–	**965**

Per-domain comparison of our models (BASE^{IPG} and LMC^{IPG}) and the baselines. Each cell holds the number of domains for which the approach in the row solved more tasks than the approach in the column. Bold numbers indicate that the approach in the row outperformed the one in the column on more domains than the reverse comparison.



Accumulated number of solved tasks over time for our models (–^{IPG}) and the baselines.

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.