Graph-Based Factorization of Classical Planning Problems

Martin Wehrle Silvan Sievers Malte Helmert
University of Basel, Switzerland

Setting

- Classical planning
- Problem reformulation

Classical Planning: Popular Solving Approaches

- Exploit independence of operators and variables
- Examples: factored planning, partial order reduction

Problem Formalization

- Usual focus: develop techniques for given problem formalization
- Little research on (automated) reformulation techniques
- Ideally: reformulation with fewer operator and variable dependencies

This Work

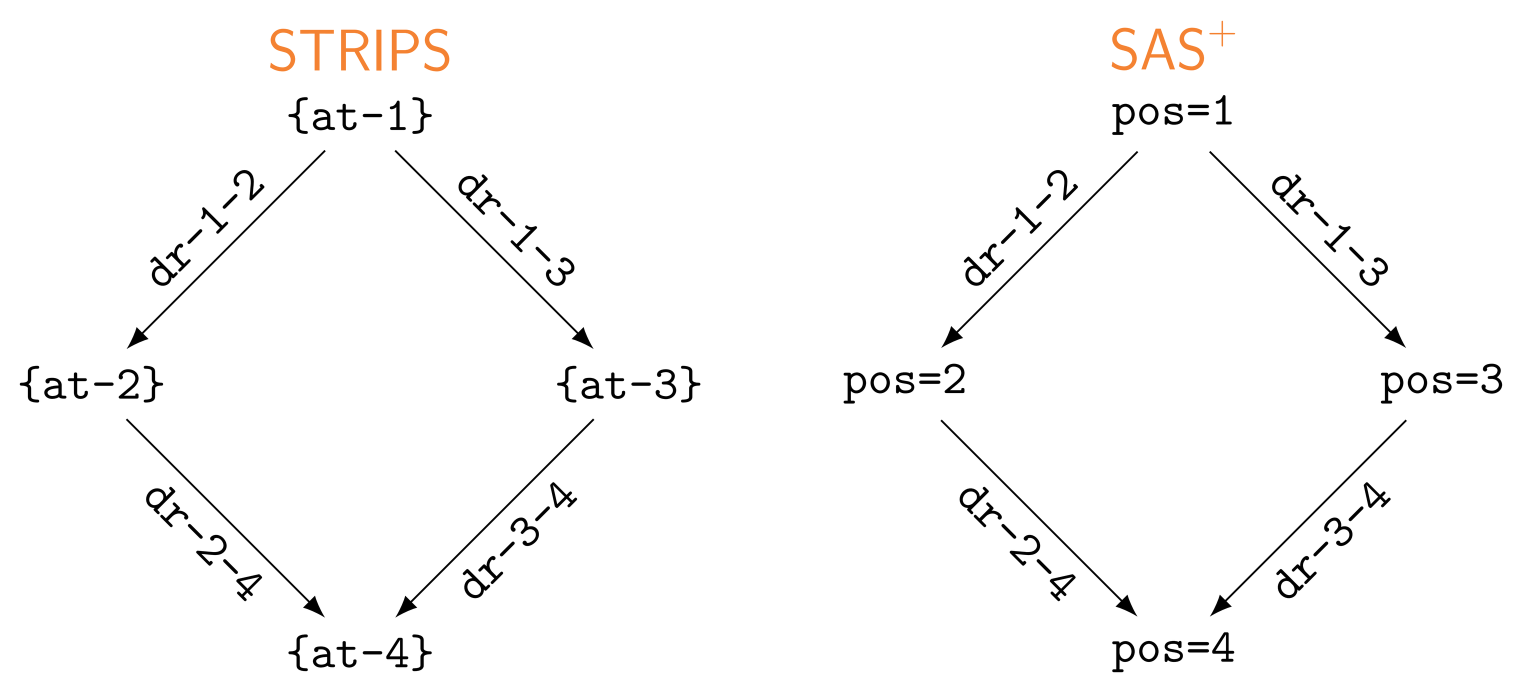
- Novel direction for reformulation ("problem factorization")
- Based on well-established theory on graph factorization

Cartesian Graph Factorization

- Well-studied problem in discrete mathematics since the 1960's
- Problem: given graph G without self-loops, find graphs $G_1, \dots, G_n$ such that the Cartesian product of $G_1, \dots, G_n$ yields G
- $G_1, \dots, G_n$ are the (unique) prime graphs of G
- Computation in polynomial time

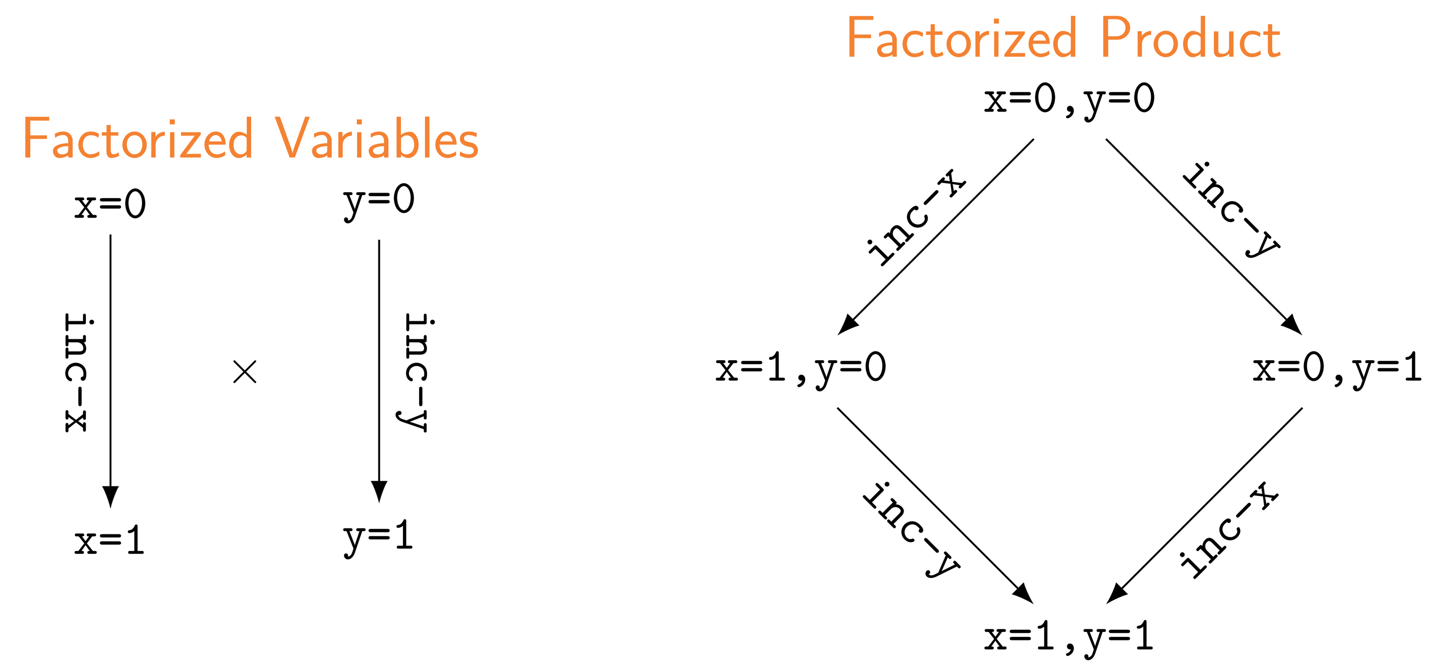
Motivating Example, Part I

- Objective: drive truck from location 1 to 4 (over 2 or 3)
- Typical formulations in STRIPS and SAS$^+$ tightly coupled (mutually interfering operators)
- Factored planning and partial order reduction do not fire



Motivating Example, Part II

- Factorization of variable pos into binary variables x and y



- Cartesian product (extended with edge labels) of factorized graphs structurally isomorphic to the graph corresponding to the original variable pos
- Factorized operators x and y are independent
- Factorization: decoupled but equivalent semantics

Factorization of Planning Problems

Self-Loops in Transition Systems

Self-loop in v's transition system if there is operator o with

- o reads v, but does not write to v, or
- o does not read v, but writes to v, or
- o does not mention v at all.

Variable Factorization

- Factorization of v into $v_1, \dots, v_n$ determined by factorization of v's transition system (self-loops removed) into $G_1, \dots, G_n$.
- Bijective mapping from v's values to value tuples of $v_1, \dots, v_n$

Operator Factorization

Factorization of operator o (given factorization of v into $v_1, \dots, v_n$):

- o reads v and writes to v: o changes exactly one factor v_i determined through factorization. Replace o's precondition and effect on v with v_i.

(Remaining cases cover self-loops)

- o reads v, but does not write to v: replace o's precondition on v with corresponding bijective precondition on $v_1, \dots, v_n$
- o does not read v, but writes to v: analogous
- o does not mention v: leave o unchanged

Theoretical Results

Π: a planning problem, and Π^f: its factorization

Theorem

The state space graphs of Π and Π^f isomorphic (modulo operator naming). Π^f preserves plan existence and optimal plan cost.

Theorem

The runtime bound for factored planning in Π^f can be exponentially lower than in Π.

Theorem

The size of the reachable state space with strong stubborn sets in Π^f can be exponentially smaller than in Π.
The number of generated nodes with iterative deepening search and sleep sets in Π^f can be exponentially smaller than in Π.

Evaluation on IPC Benchmarks

- Available graph factorization algorithms not directly applicable
- "Try-and-check" algorithm: factorize graph structure (ignoring labels), check if still valid when taking lables into account
- Graph structures in Floortile, Grid, Sokoban, Tetris and VisitAll factorizable, but fail the label check
  - Case study: modified VisitAll domain
  - no pruning vs. sleep sets pruning with IDA*; dashes: > 1800 seconds

	original formulation				factorized formulation			
	#nodes		runtime		#nodes		runtime	
size	nop	prune	nop	prune	nop	prune	nop	prune
	blind							
6	3853	3853	0.0	0.0	3853	987	0.0	0.0
9	2.5e+6	2.5e+6	12.2	14.4	2.5e+6	93613	12.2	0.6
12	—	—	—	—	—	1.1e+7	—	86.0
	iPDB							
12	81042	81042	0.7	0.8	72721	19102	4.5	4.2
16	1.9e+7	1.9e+7	112.6	125.5	1.3e+7	1.5e+6	107.7	22.9
20	—	—	—	—	—	1.6e+8	—	1710.8

Future Work

- Turn theory into practice
- More specialized factorization algorithms for atomic abstractions
- Weaker factorization: only reachable part of product relevant