Safe Abstraction in Fast Downward

University of Basel, 27.01.2025
Bachelor’s Thesis Presentation
Joan Moser

Outline

Background

Planning and abstractions

Safe Abstraction

Safe abstractions in Fast Downward

Evaluation

Effect of safe abstraction on search

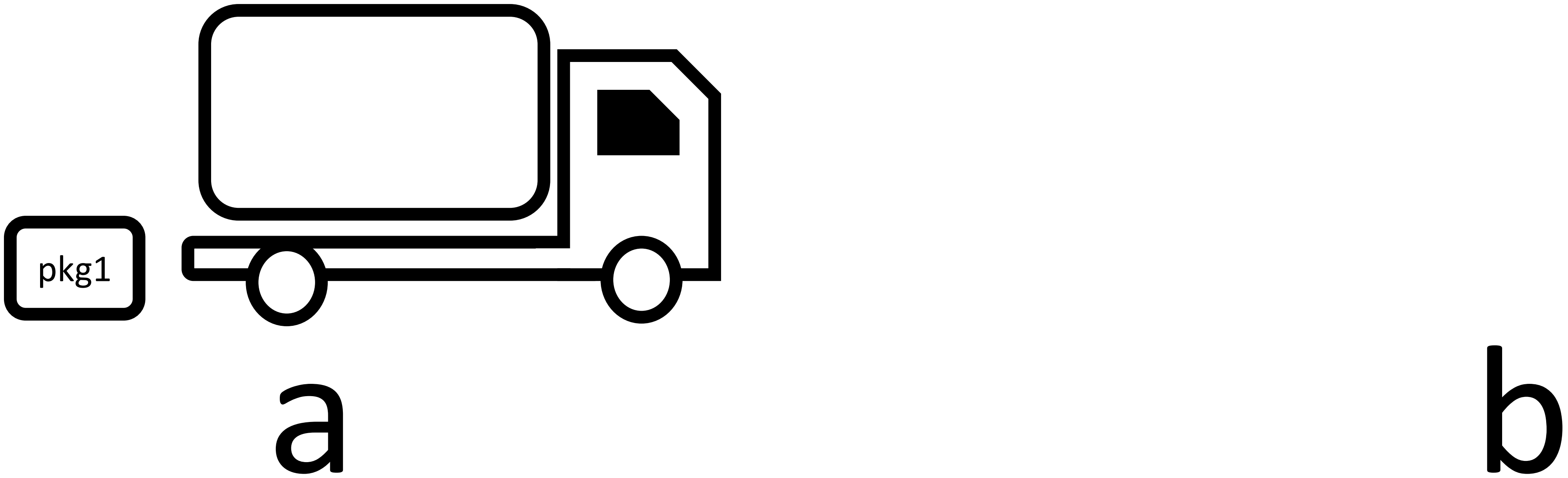
Conclusion

What have we learned? What is left to do?

Background

Planning and abstractions

Planning – Trucks Problem



Planning – Variables

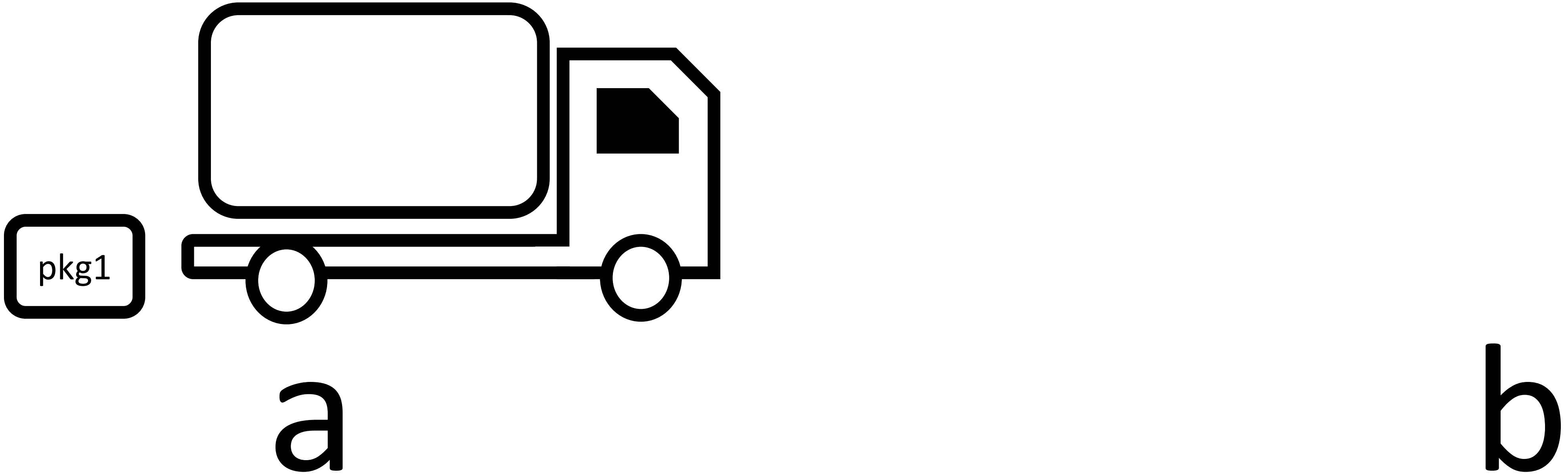
Variables in the trucks problem:

• Location of truck	*truck*	*(at-A, at-B)*
• Location of package	*p*	*(at-A, at-B, in truck)*
• Occupancy of truck	*cargo*	*(empty, contains-p)*

A state is a value assignment over all variables.

In this case:

$$truck \mapsto at{-}A$$
$$p \mapsto at{-}A$$
$$cargo \mapsto empty$$



Planning – Operators

Operators in the trucks problem:

- Driving the truck (a to b, b to a)
- Drop the package (at a, at b)
- Pick-up the package (at a, at b)

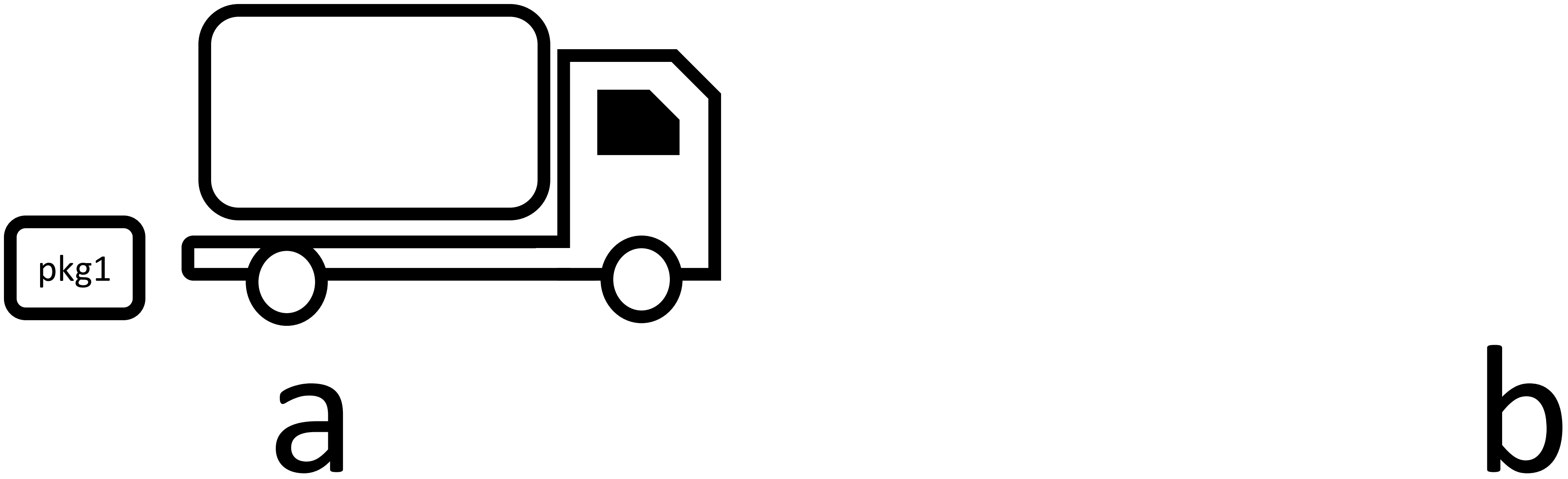
Example: *pick-up-truck-a-p*

Preconditions:

- $truck \mapsto at\text{-}A$ ✓
- $p \mapsto at\text{-}A$ ✓
- $cargo \mapsto empty$ ✓

Effects:

- $p \mapsto in\text{-}truck$
- $cargo \mapsto contains\text{-}p$



Planning – Goal

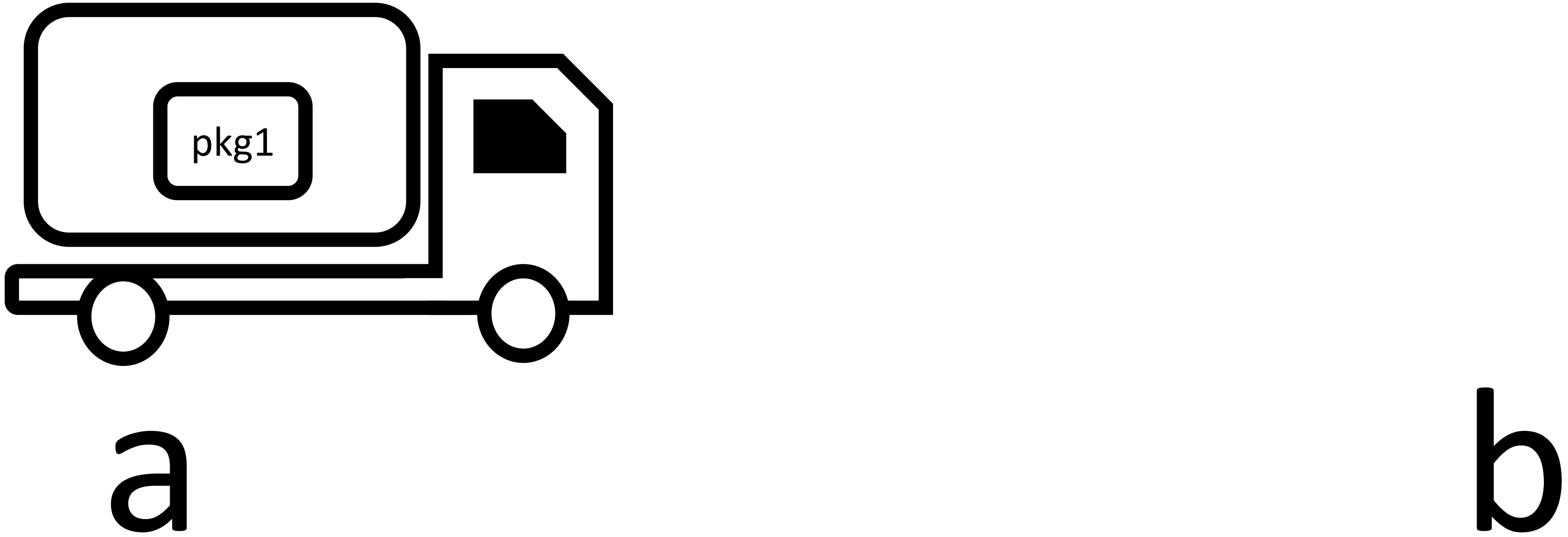
Goal is a partial state

Goal in the trucks problem:

$p \mapsto$ *at-B*

A **plan** is the sequence of operators used to get to a goal

- *pick-up-truck-a-p*
- *drive-truck-a-b*
- *drop-truck-b-p*
- *drive-truck-b-a*



Abstraction

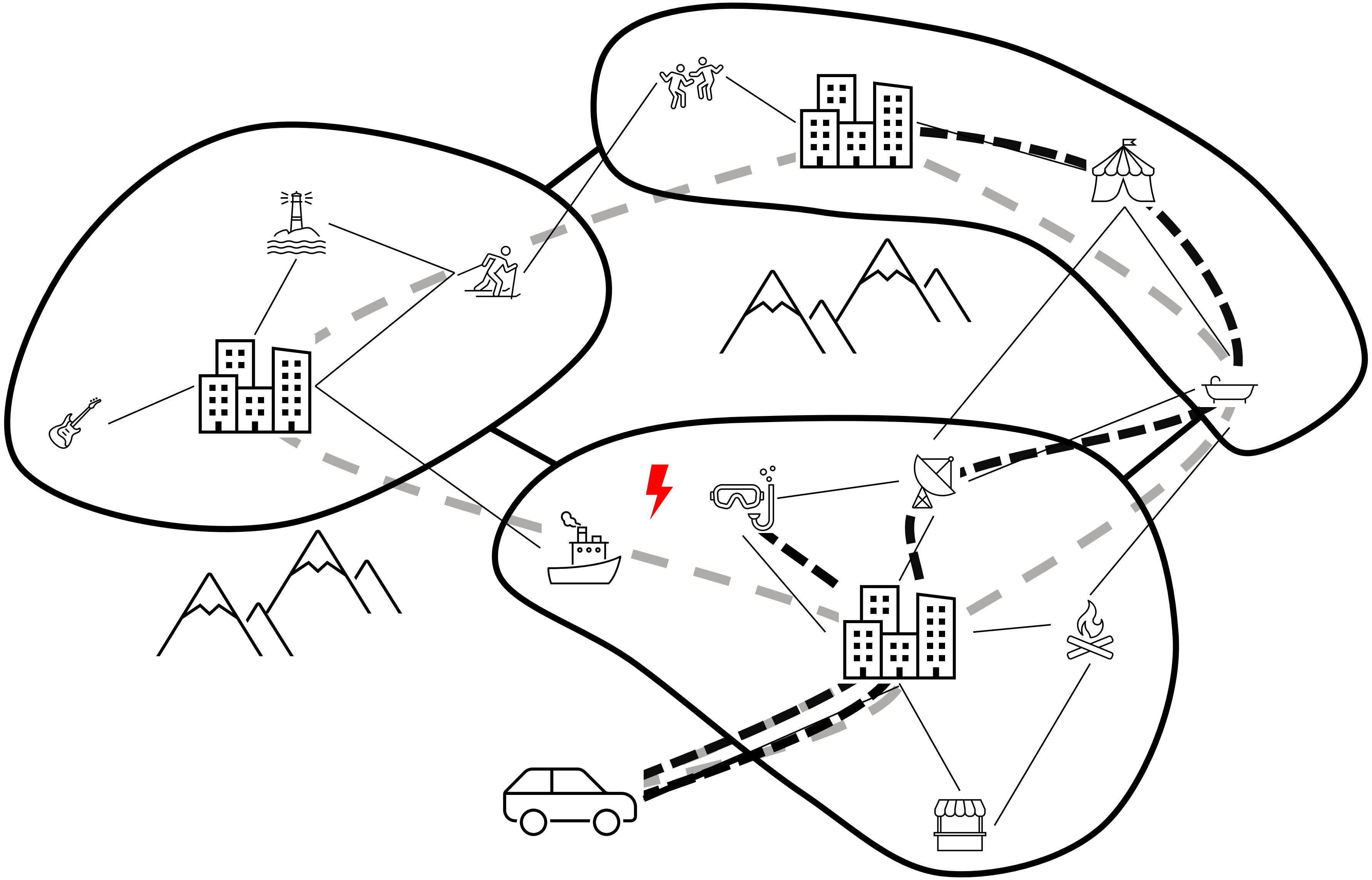
Problem:

- State space to be searched can grow exponential in problem description

Idea:

- Ignore some details
- Focus on bigger picture
- Guide search for a plan

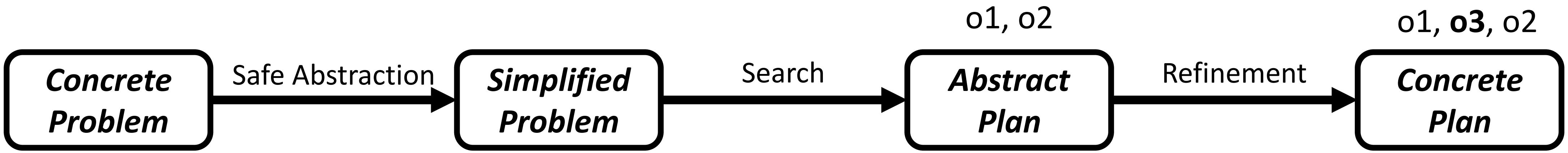
Abstraction – Road-trip example



Safe Abstraction

Safe abstractions in Fast Downward

Safe (Variable) Abstraction



How to find safe variables?
How to refine abstract plan?

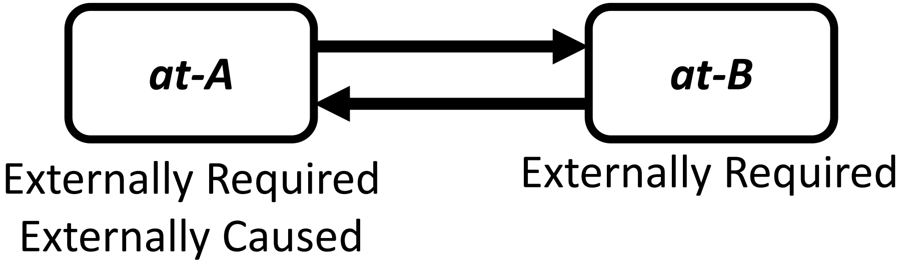
Safe (Variable) Abstraction

How to find safe variables?

Free Domain Transition Graph (Free DTG)

truck (location of truck)

- *at-A*
- *at-B*



drive-truck-a-b

pre: ***truck*** ↦ ***at-A***

eff: ***truck*** ↦ ***at-B***

drive-truck-b-a

pre: ***truck*** ↦ ***at-A***

eff: ***truck*** ↦ ***at-B***

A variable can be abstracted safely if, in the free DTG:

- All ex. required values are strongly connected. ✓
- Every ex. required value can be reached from any ex. caused value. ✓
- The goal value (if present) can be reached from each ex. required value. ✓

drop-truck-b-p

pre: ***truck*** ↦ ***at-B***, *p* ↦ *in-truck*, *cargo* ↦ *contains-p*

eff: *p* ↦ *at-B*, *cargo* ↦ *empty*

drop-truck-a-p

pick-up-truck-a-p

pick-up-truck-b-p

Safe (Variable) Abstraction

How to remove safe variables?

drive-truck-a-b
pre: ***truck*** ↦ ***at-A***
eff: ***truck*** ↦ ***at-B***

→

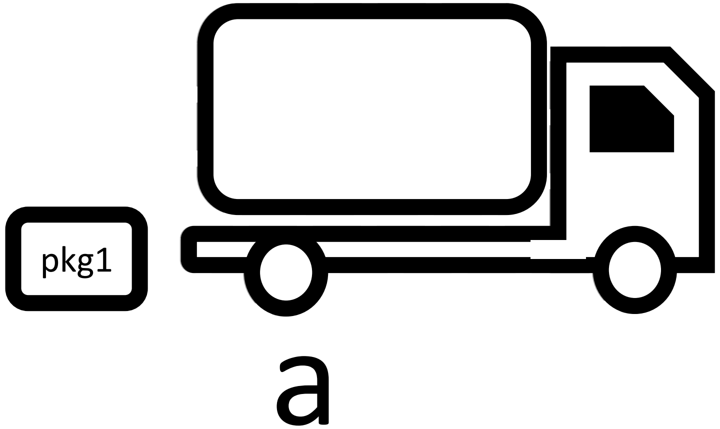
drive-truck-a-b
pre:
eff:

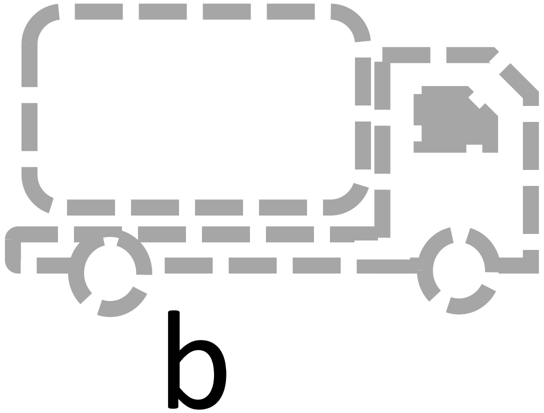
pick-up-truck-a-p
pre: ***truck*** ↦ ***at-A***, *p* ↦ *at-A, cargo* ↦ *empty*
eff: *p* ↦ *in-truck, cargo* ↦ *contains-p*

→

pick-up-truck-a-p
pre: *p* ↦ *at-A, cargo* ↦ *empty*
eff: *p* ↦ *in-truck, cargo* ↦ *contains-p*

Remove variable from goal condition (if present)





Safe (Variable) Abstraction

How to refine abstract plan?

pick-up-truck-a-p

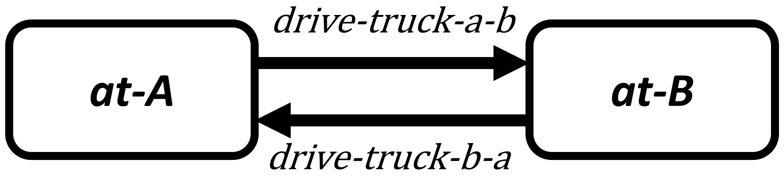
pre: ***truck*** ↦ ***at-A***, *p* ↦ *at-A*, *cargo* ↦ *empty*

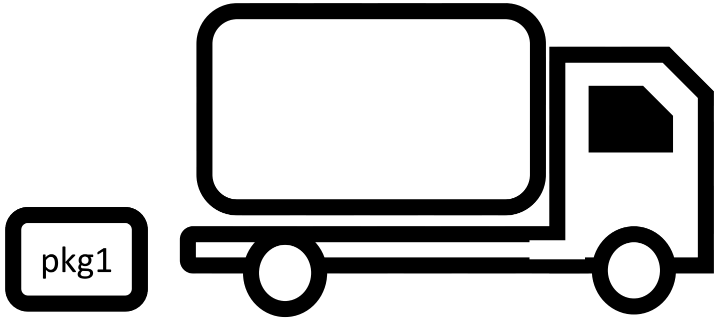
eff: *p* ↦ *in-truck*, *cargo* ↦ *contains-p*

drop-truck-b-p

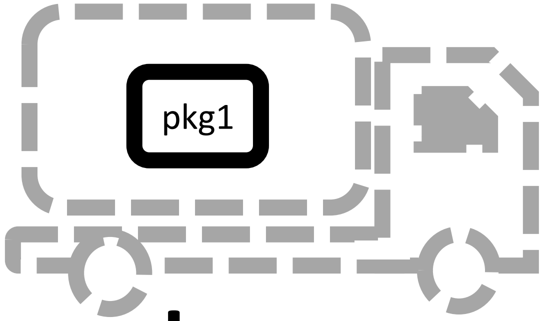
pre: ***truck*** ↦ ***at-B***, *p* ↦ *in-truck*, *cargo* ↦ *contains-p*

eff: *p* ↦ *at-B*, *cargo* ↦ *empty*





a



b

Operator Composition

Can we abstract more variables?

Remaining operators:

- *drop-truck-a-p*
- *drop-truck-b-p*
- *pick-up-truck-a-p*
- *pick-up-truck-b-p*

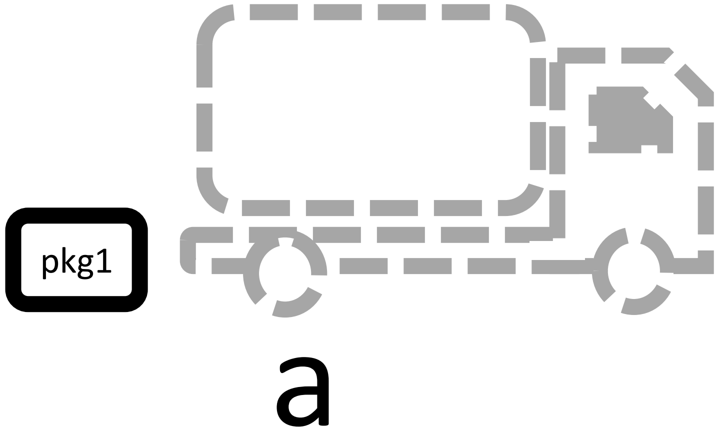
All change variables ***cargo*** and ***p***

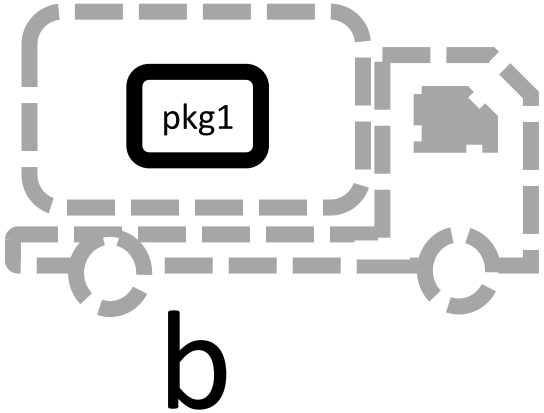
Idea: Combine ***pick-up*** and ***drop*** operators

Pick-up operator always followed by ***drop*** operator

pick-up-truck-a-p
    pre: …, ***cargo*** ↦ ***empty***
    eff: …, *cargo* ↦ *contains-p*
drop-truck-b-p
    pre: …, *cargo* ↦ *contains-p*
    eff: …, ***cargo*** ↦ ***empty***

Before and after sequence: ***cargo*** ↦ ***empty***





Operator Composition

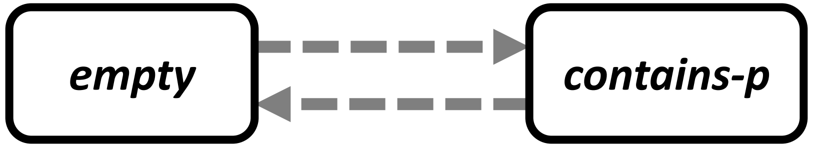
Can we abstract more variables?

Remaining operators:

pick-up-a-drop-a
pick-up-a-drop-b
pick-up-b-drop-a
pick-up-b-drop-b

pick-up-a-drop-b
pre: $p \mapsto a, cargo \mapsto empty$
eff: $p \mapsto b$

All change variables only change ***p***

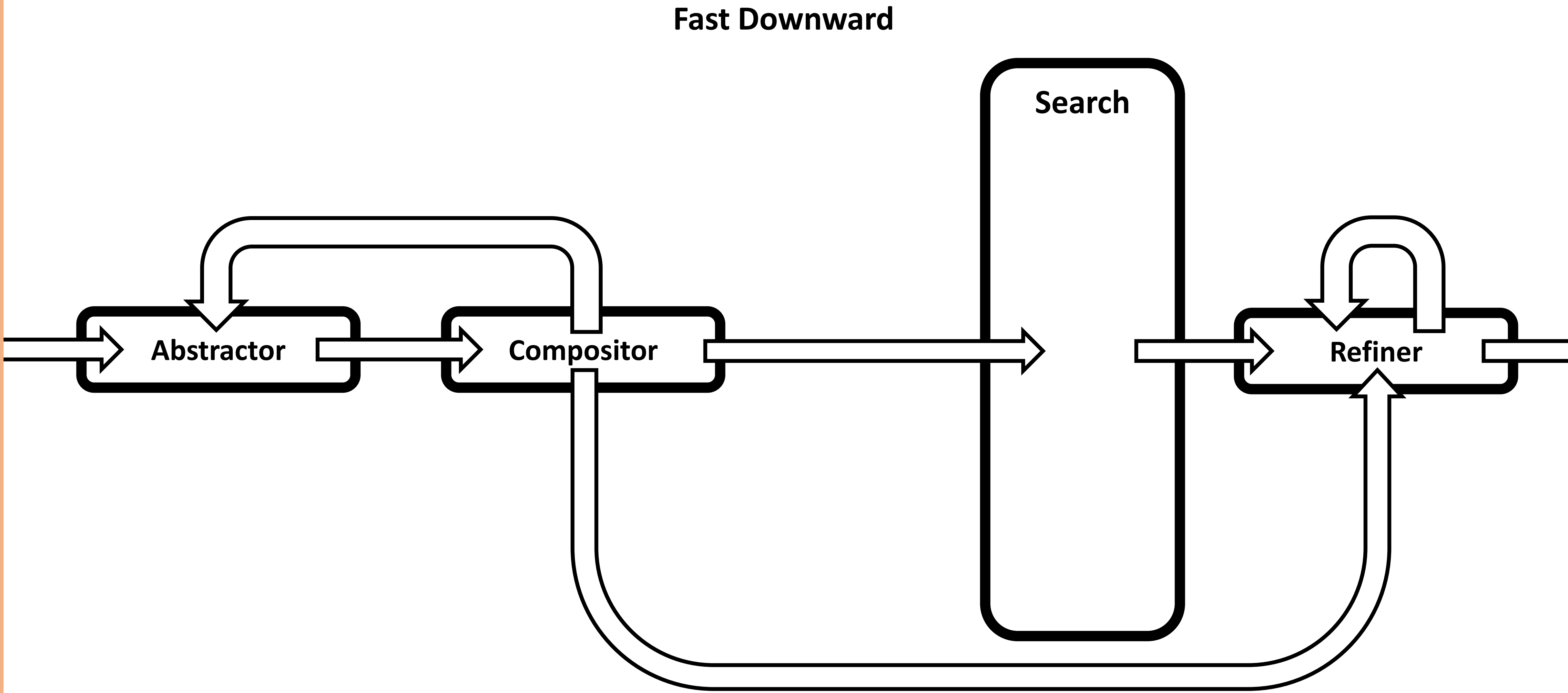


Externally Required
Externally Caused

A variable can be abstracted safely if, in the free DTG:

- All ex. required values are strongly connected. ✓
- Every ex. required value can be reached from any ex. caused value. ✓
- The goal value (if present) can be reached from each ex. required value. ✓

Implementation



Evaluation

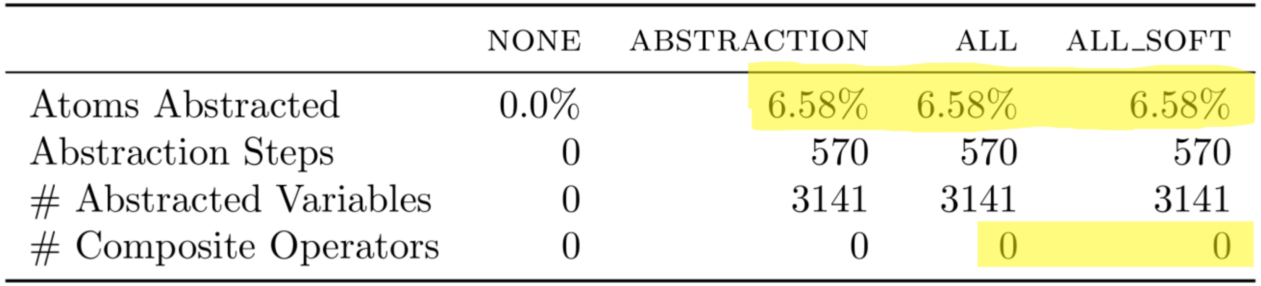
Effect of safe abstraction on search

Abstraction Results

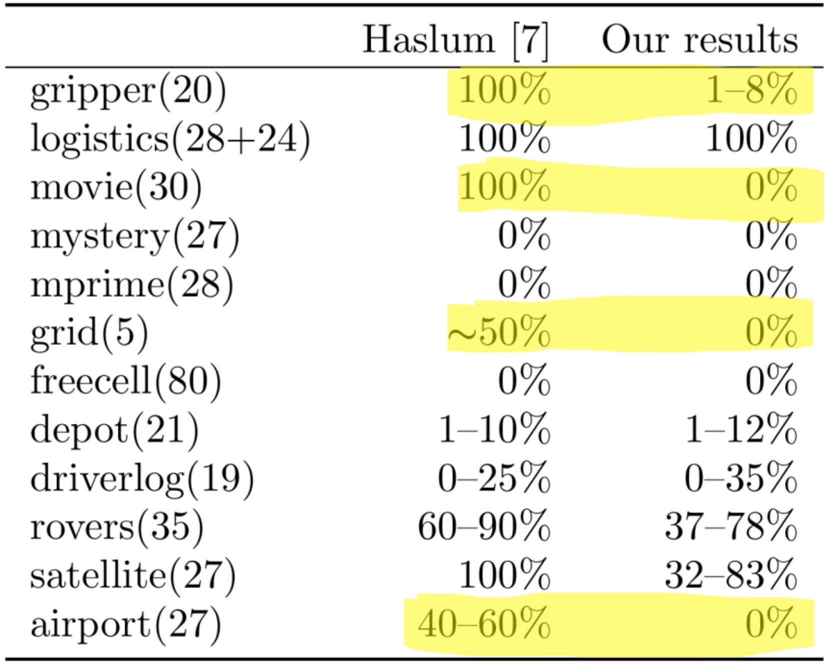
Configurations

- NONE
- ABSTRACTION
- ALL
- ALL_SOFT

Changes behaviour of safe abstraction

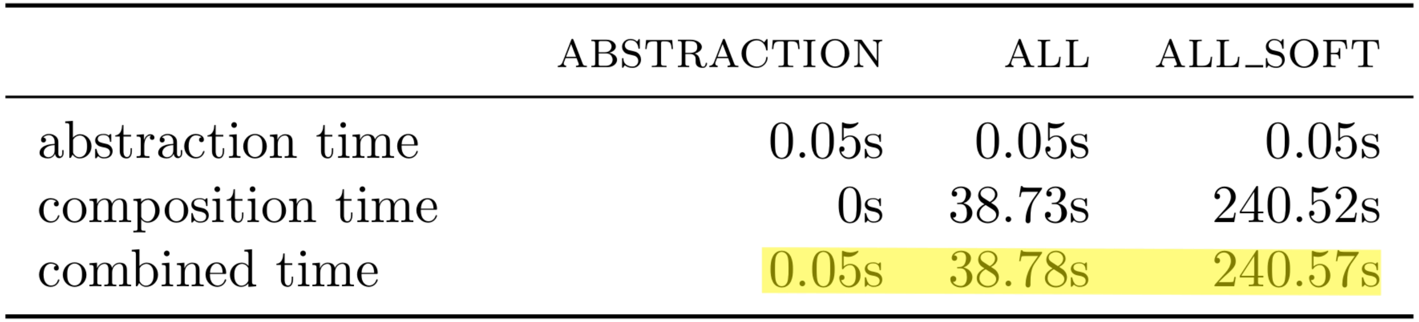


	NONE	ABSTRACTION	ALL	ALL_SOFT
Atoms Abstracted	0.0%	6.58%	6.58%	6.58%
Abstraction Steps	0	570	570	570
# Abstracted Variables	0	3141	3141	3141
# Composite Operators	0	0	0	0



	Haslum [7]	Our results
gripper(20)	100%	1–8%
logistics(28+24)	100%	100%
movie(30)	100%	0%
mystery(27)	0%	0%
mprime(28)	0%	0%
grid(5)	∼50%	0%
freecell(80)	0%	0%
depot(21)	1–10%	1–12%
driverlog(19)	0–25%	0–35%
rovers(35)	60–90%	37–78%
satellite(27)	100%	32–83%
airport(27)	40–60%	0%

Abstraction Results



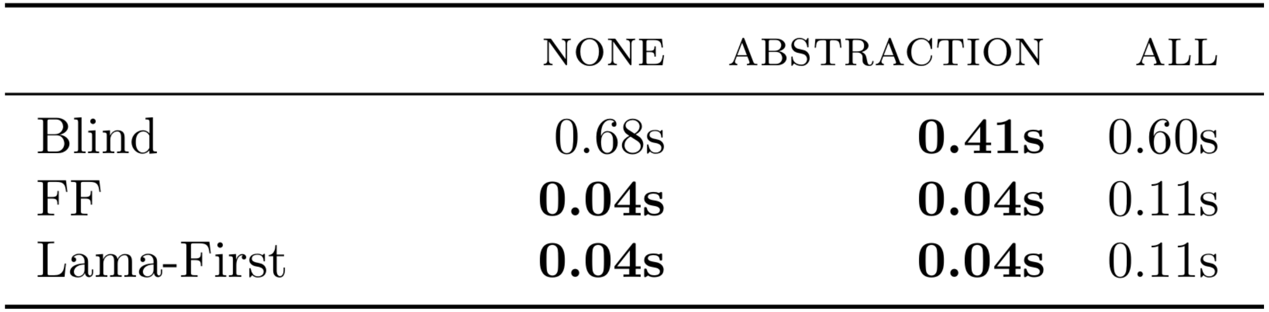
	ABSTRACTION	ALL	ALL_SOFT
abstraction time	0.05s	0.05s	0.05s
composition time	0s	38.73s	240.52s
combined time	0.05s	38.78s	240.57s

Search Results

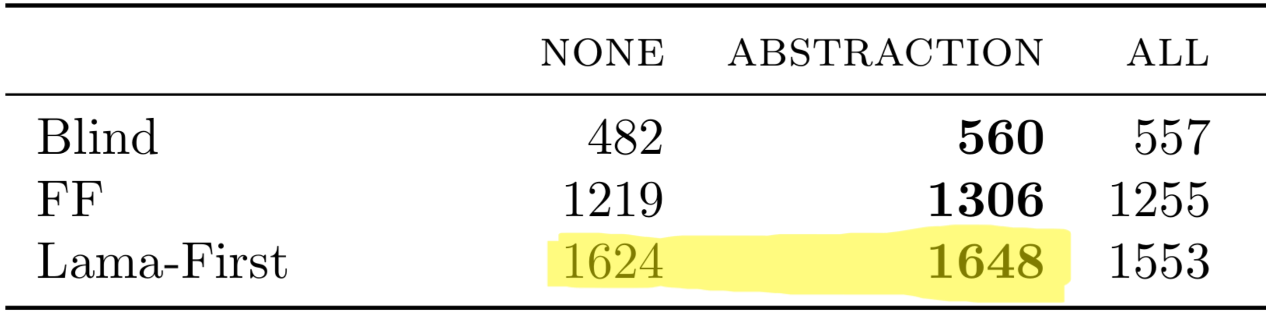
Search Algorithms

- Lama-First
- FF
- Blind

Changes behaviour of search

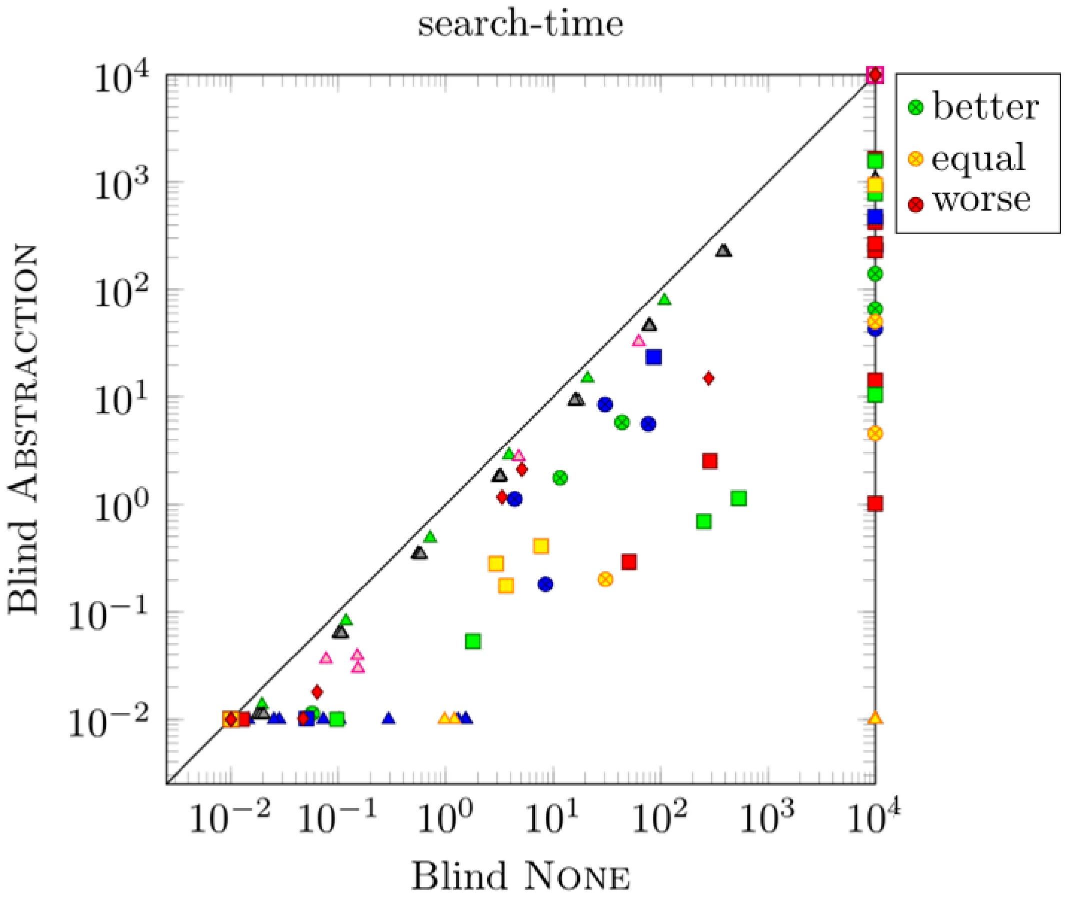


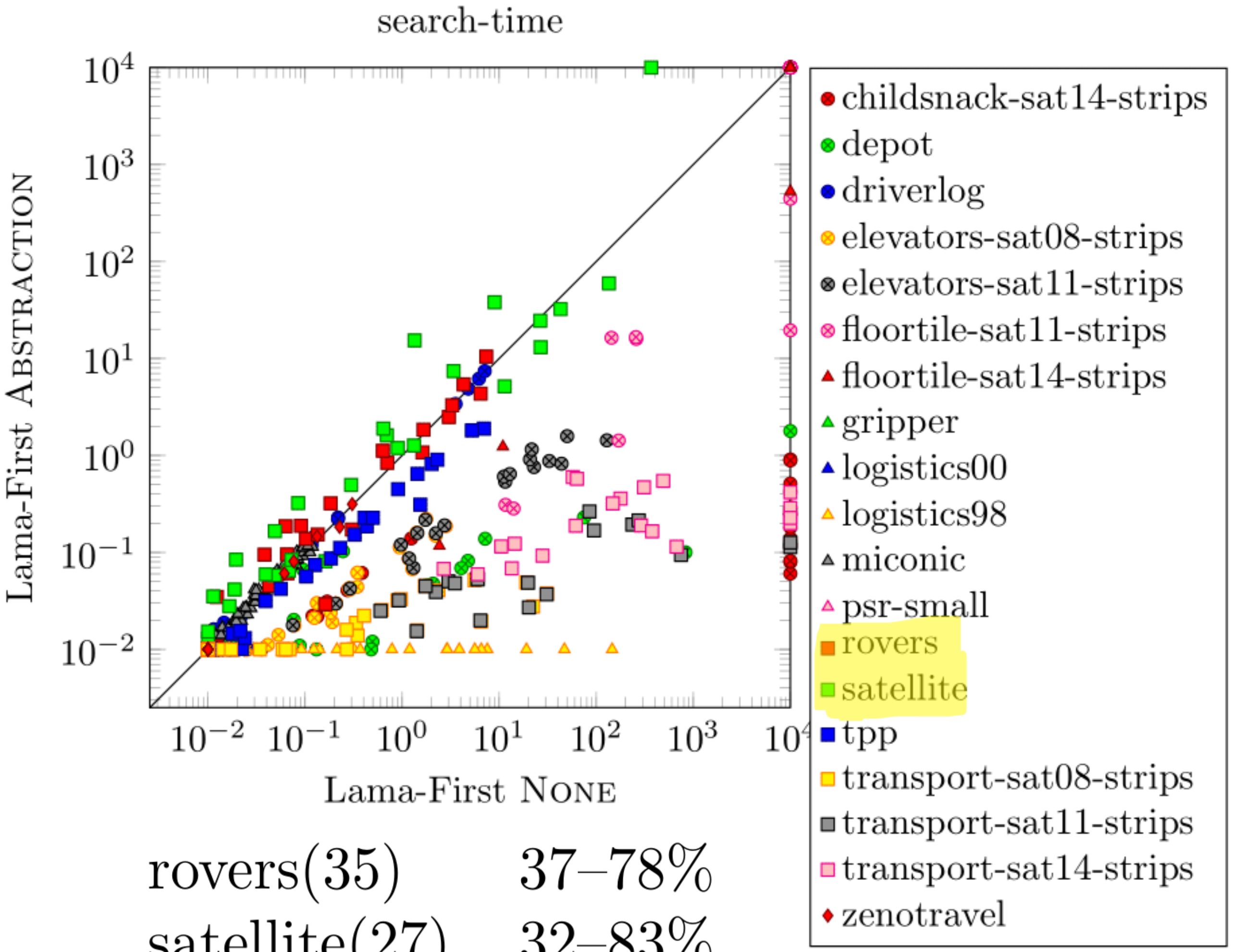
	NONE	ABSTRACTION	ALL
Blind	0.68s	**0.41s**	0.60s
FF	**0.04s**	**0.04s**	0.11s
Lama-First	**0.04s**	**0.04s**	0.11s



	NONE	ABSTRACTION	ALL
Blind	482	**560**	557
FF	1219	**1306**	1255
Lama-First	1624	**1648**	1553

Search Results





rovers(35)	37–78%
satellite(27)	32–83%

Conclusion

What have we learned?

What is left to do?

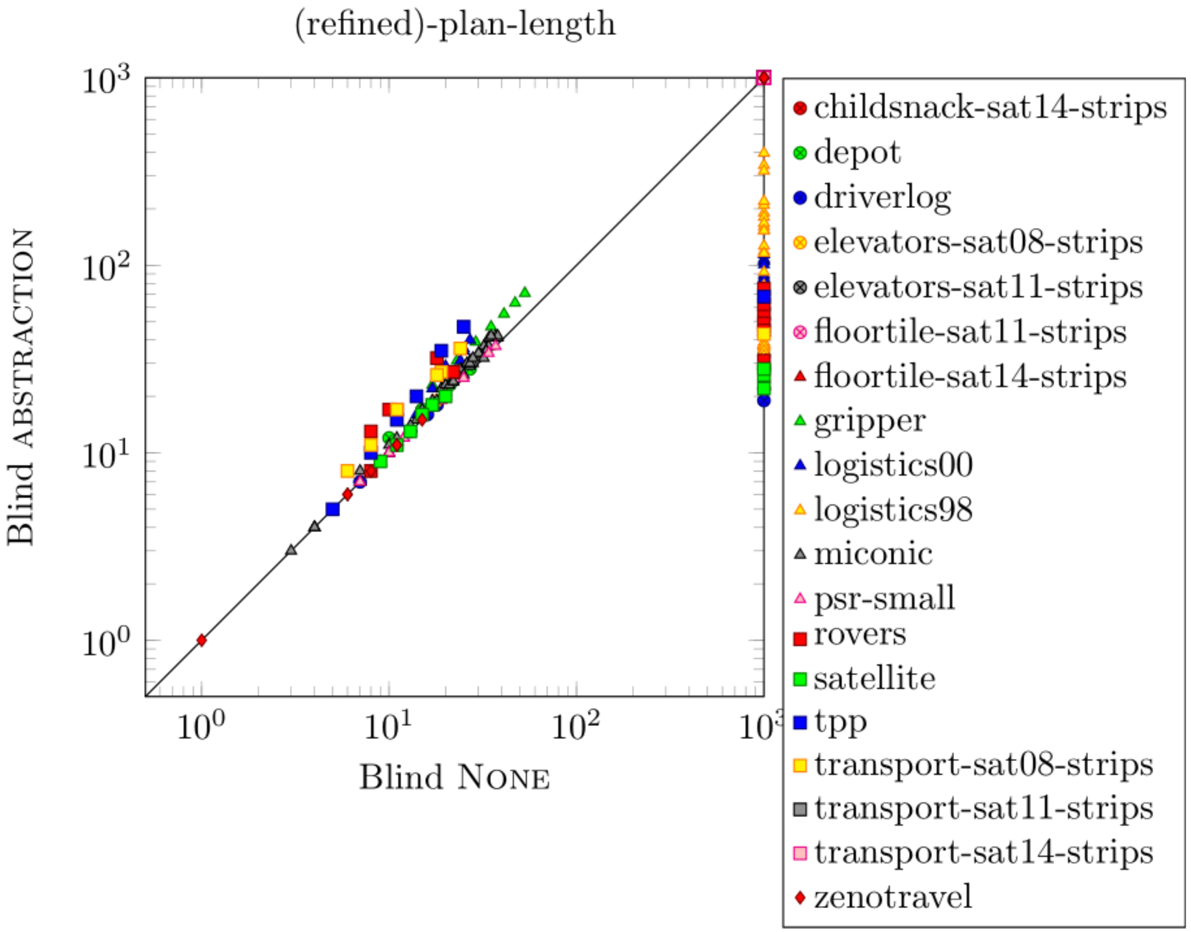
Conclusion

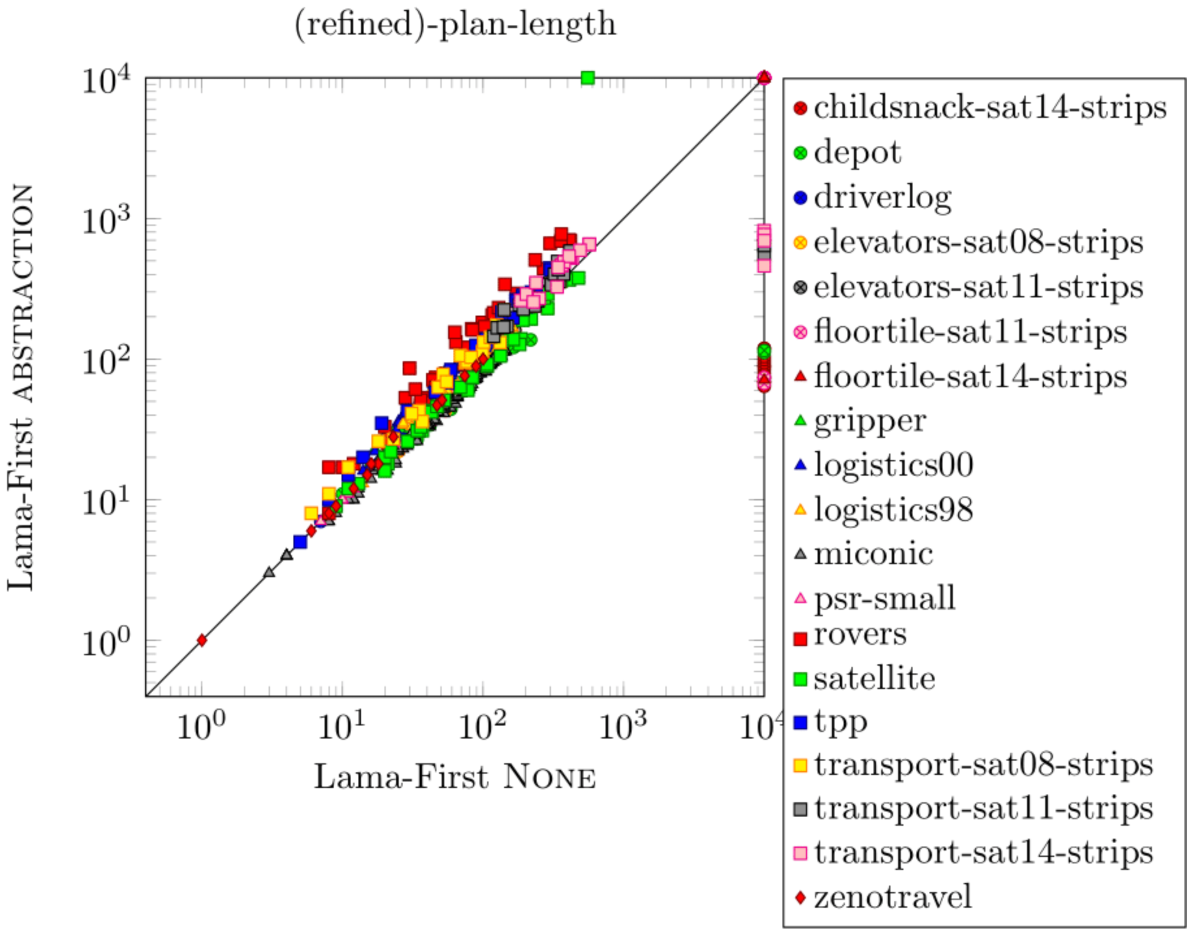
- Unable to reproduce operator composition
- Safe abstraction can improve time and memory usage
- Can have counter-intuitive influence on sophisticated search algorithm
- Effectiveness of safe abstraction greatly depends on problem encoding

Future?

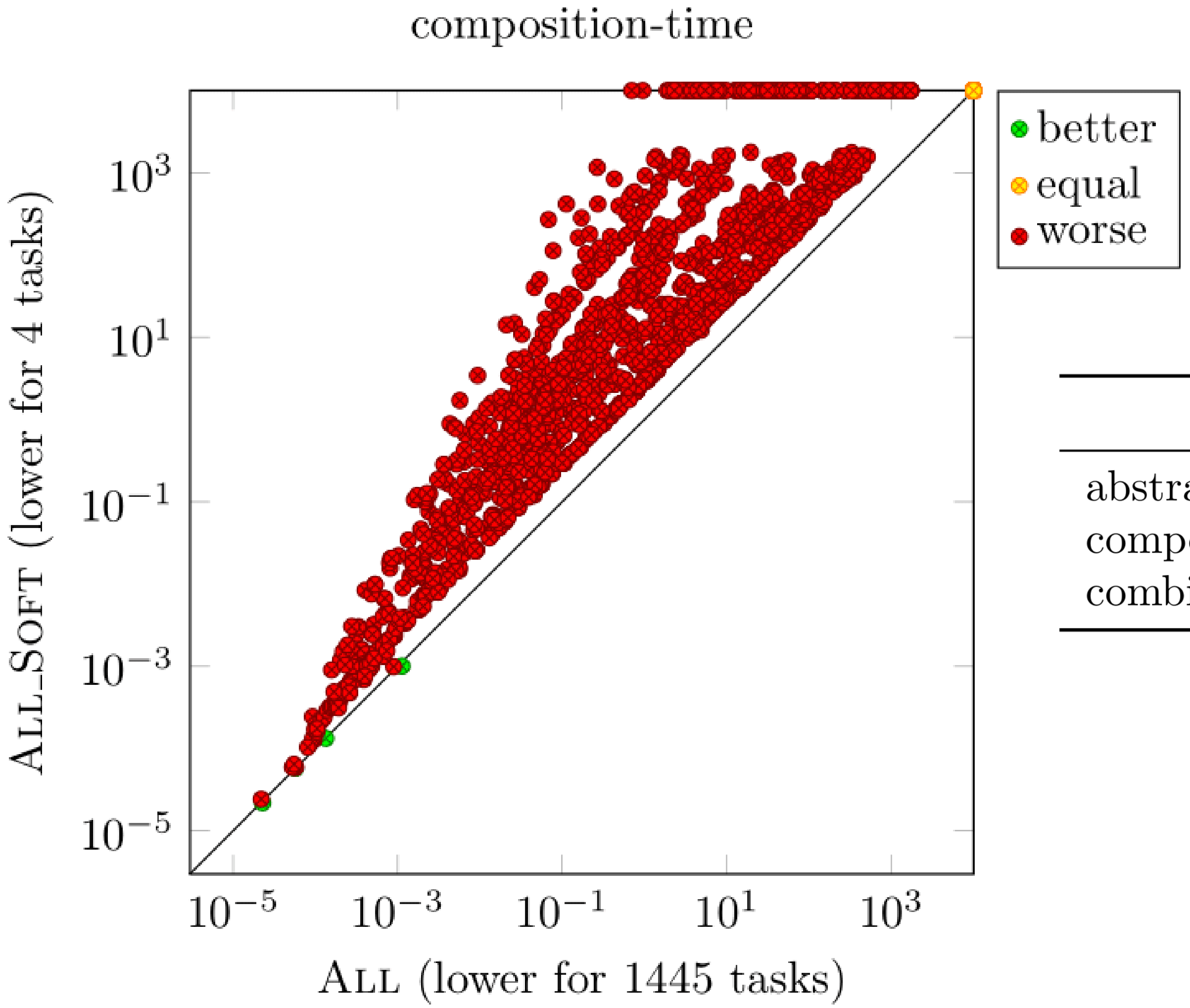
- Why are our results different from Haslum?
  - Is the problem the encodings?
  - Is the problem the implementation of composition?
  - What do the other techniques Haslum mentioned do?
- Why do some domains (rovers, satellite) behave counter to our intuition in Lama-First?

Addendum_1 – Plan Length



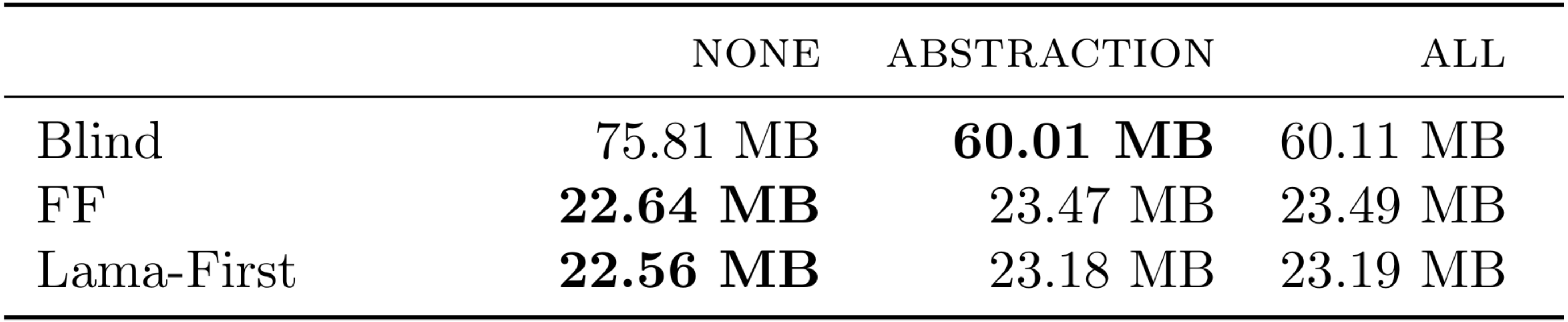


Addendum_2 – Composition Time



	ABSTRACTION	ALL	ALL_SOFT
abstraction time	0.05s	0.05s	0.05s
composition time	0s	38.73s	240.52s
combined time	0.05s	38.78s	240.57s

Addendum_3 – Memory



	NONE	ABSTRACTION	ALL
Blind	75.81 MB	**60.01 MB**	60.11 MB
FF	**22.64 MB**	23.47 MB	23.49 MB
Lama-First	**22.56 MB**	23.18 MB	23.19 MB

Addendum_4 – Free DTG example

Free DTG of *p*

Free Domain Transition Graph (Free DTG)

p (location of package)

- *at-A*
- *at-B*
- *in-truck*

drop-truck-a-p

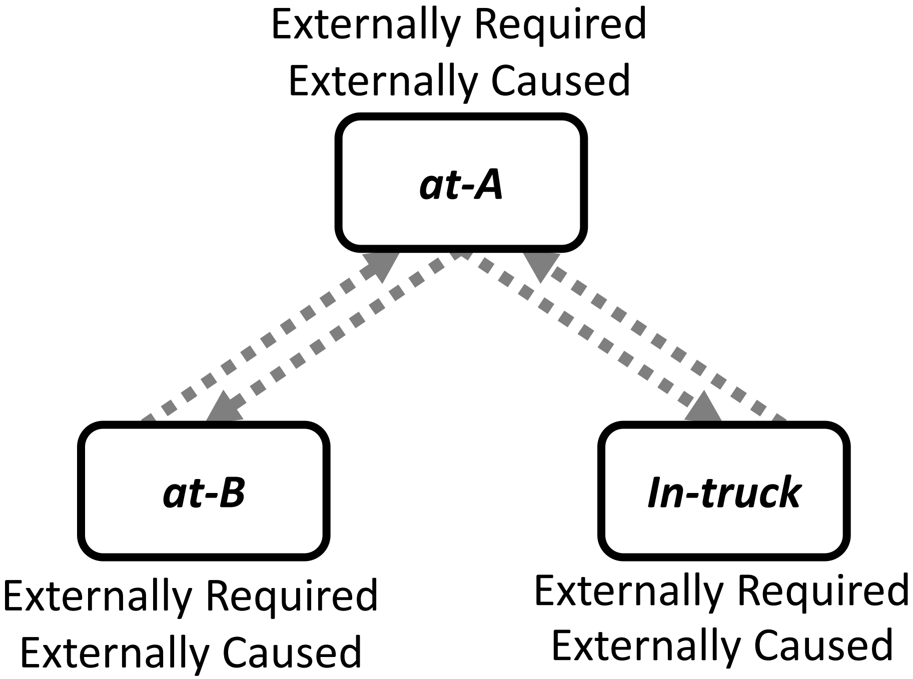
pre: *truck* ↦ *at-A*, ***p*** ↦ ***in-truck***, *cargo* ↦ *contains-p*

eff: ***p*** ↦ ***at-A***, *cargo* ↦ *empty*

drop-truck-b-p

pick-up-truck-a-p

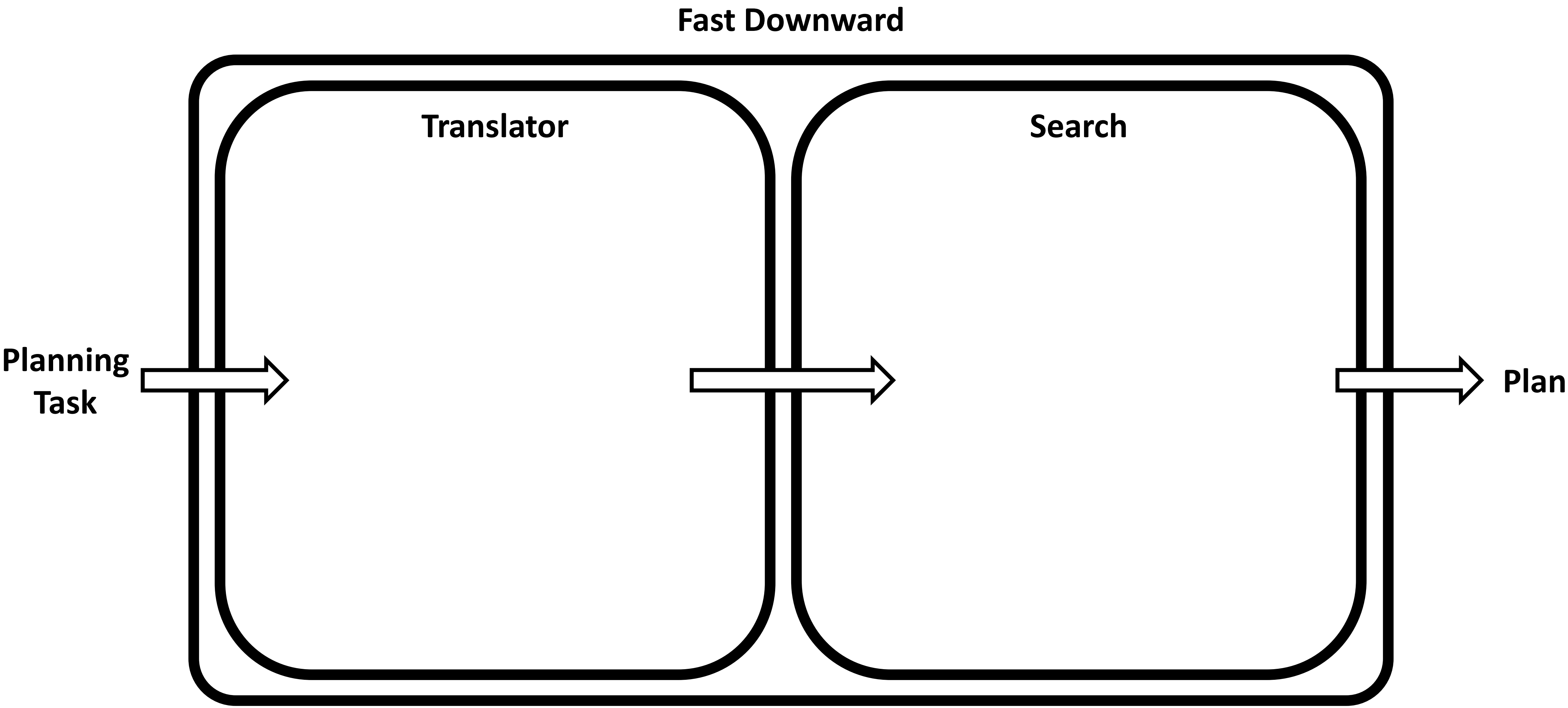
pick-up-truck-b-p



A variable can be abstracted safely if, in the free DTG:

- All ex. required values are strongly connected. ✗
- Every ex. required value can be reached from any ex. caused value. ✗
- The goal value (if present) can be reached from each ex. required value. ✗

Addendum_5 – Fast Downward


Fast Downward
Translator
Search
Planning Task
Plan