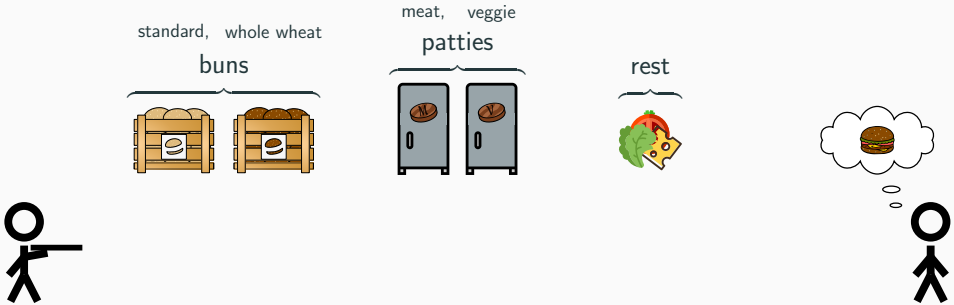
A Lifted Backward Computation of h^{add}

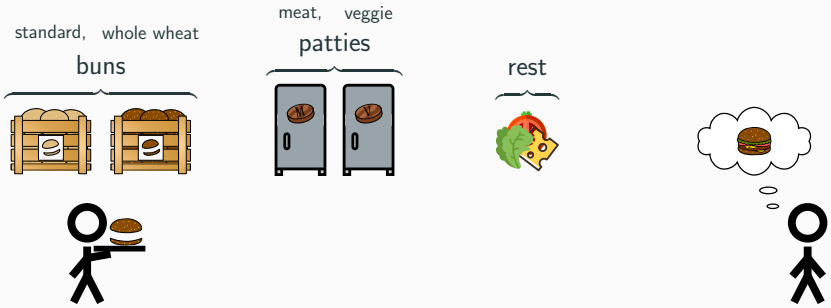
Pascal Lauer[1], Álvaro Torralba[2], Daniel Höller[1], Jörg Hoffmann[1]

[1]Saarland University, Saarland Informatics Campus, Saarbrücken, Germany, lastname@cs.uni-saarland.de
[2]Aalborg University, Denmark, alto@cs.aau.dk

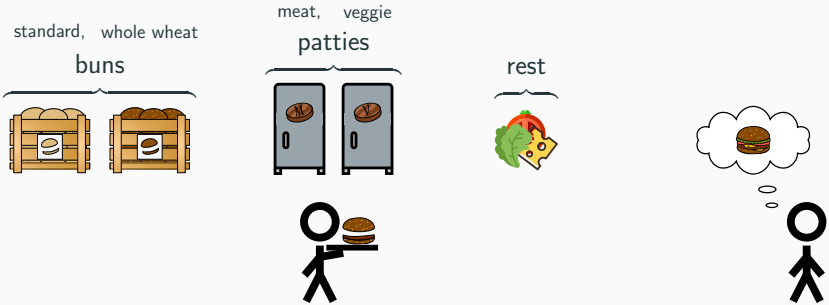
Running Example: Childsnack



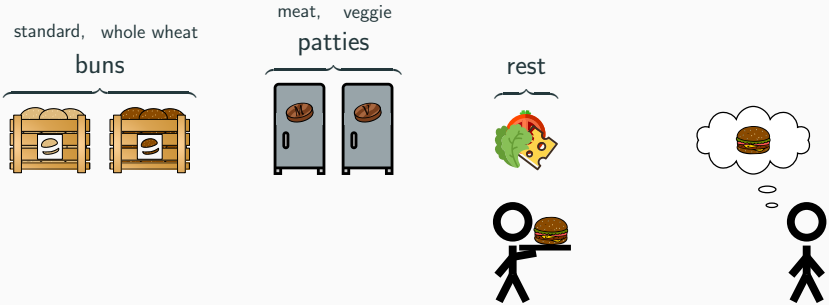
Running Example: Childsnack



Running Example: Childsnack



Running Example: Childsnack



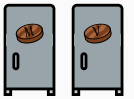
Running Example: Childsnack

standard, whole wheat

buns

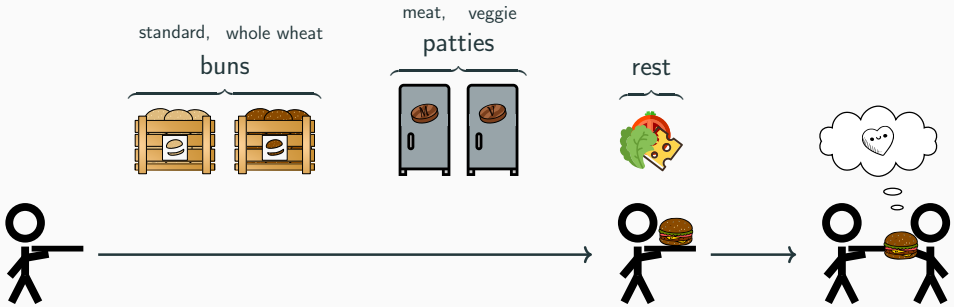
meat, veggie

patties



rest

Running Example: Childsnack



Encoding the Actions into PDDL

make(?bun, ?patty, ?rest):

pre : {inKitchen(?bun), inKitchen(?patty), inKitchen(?rest)}

add : {burger(?bun, ?patty, ?rest)}

del : {inKitchen(?bun), inKitchen(?patty), inKitchen(?rest)}

serve(?child, ?bun, ?patty, ?rest):

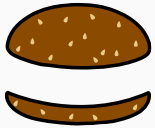
pre : {burger(?bun, ?patty, ?rest),

likes(?child, ?bun), likes(?child, ?patty), likes(?child, ?rest)}

add : {served(?child)}

del : {burger(?bun, ?patty, ?rest)}

Encoding the Planning Task



Planning task encoding formal and concrete

(Lifted) Plannings task $(\mathcal{P}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G})$

- $\mathcal{P}$ is a set of predicates.
  { inKitchen, likes, burger, served }
- $\mathcal{O}$ is a set of objects.
  { bun_{W1}, bun_{W2}, ..., patty_{M1}, ..., c_1 }
- $\mathcal{A}$ is a set of actions.
  { make, serve }
- $\mathcal{I}$ is a state, called initial state.
  { likes(c_1, patty_{M1}), ..., inKitchen(bun_{W1}, ...) }
- $\mathcal{G}$ is a set of grounded atoms.
  { served(c_1) }

Grounding not Possible

n contents

burger($?c_1$,...,$?c_n$)

The Search Setting

Search by grounding only the applicable actions per state. [Corrêa et al. 2020]

Examples of existing heuristics to guide the search:

- h^{UR} [Lauer et al. 2021]
- h^{max}, h^{add} [Corrêa et al. 2021]
- h^{FF} [Corrêa et al. 2022]
- Landmark Heuristics [Wichlacz et al. 2022; 2023]

Here: h^{add}

h^{add} [Bonet and Geffner, 2001]

$$\sum_{f \in \mathcal{G}} h^{add}(f)$$

inKitchen(...)	...	likes(...)	...	burger(...)	...	burger(...)	served(c_1)
0	...	0	..	∞	...	∞	∞
0	...	0	..	1	...	1	∞
0	...	0	..	1	...	1	2

h^{add} [Bonet and Geffner, 2001]

$$\sum_{f \in \mathcal{G}} h^{add}(f)$$

inKitchen(...)	...	likes(...)	...	burger(...)	...	burger(...)	served(c_1)
0	...	0	..	∞	...	∞	∞
0	...	0	..	1	...	1	∞
0	...	0	..	1	...	1	2

[Helmert, 2009; Corrêa et al., 2021]

Determine achievable facts without enumerating all actions.

h^{add} [Bonet and Geffner, 2001]

$$\sum_{f \in \mathcal{G}} h^{add}(f)$$

burger(?c$_1$,...,?c$_n$)

inKitchen(...)	...	likes(...)	...	burger(...)	...	burger(...)	served(c$_1$)
0	...	0	..	∞	...	∞	∞
0	...	0	..	1	...	1	∞
0	...	0	..	1	...	1	2

[Helmert, 2009; Corrêa et al., 2021]

Determine achievable facts without enumerating all actions.

Complexity Analysis

Complexity of computing h^{add}

- In general: **EXPTIME**-complete
- Bounded predicate arity*: in **P**

Complexity Analysis

Complexity of computing h^{add}

- In general: **EXPTIME**-complete
- Bounded predicate arity*: in **P**

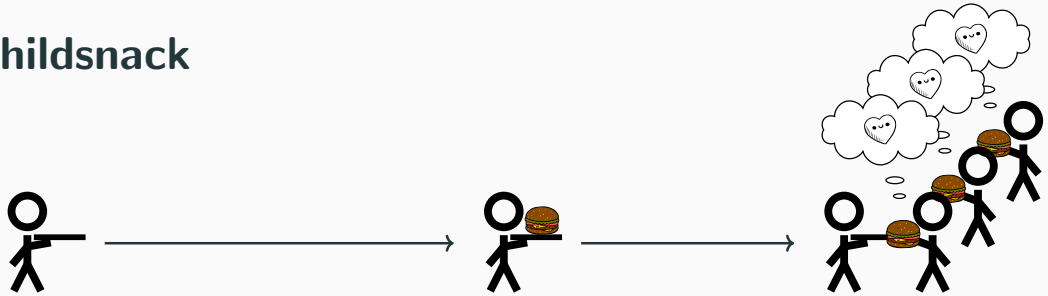
Question: Are there more tractable subclasses?

- If h^{add} is bounded (per goal fact)*: in **P**

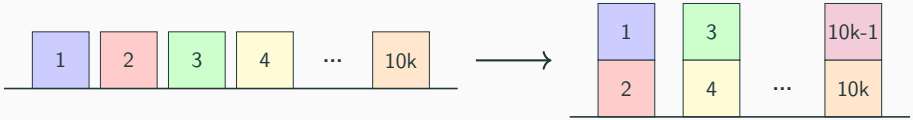
---

*Assuming all evaluated queries to be acyclic

Childsnack

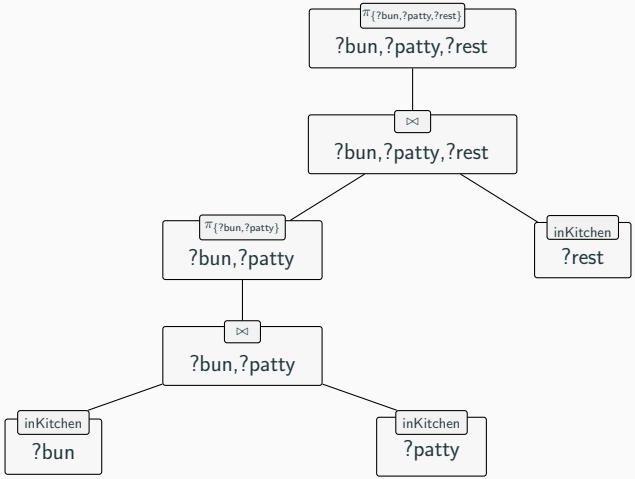


Blocksworld



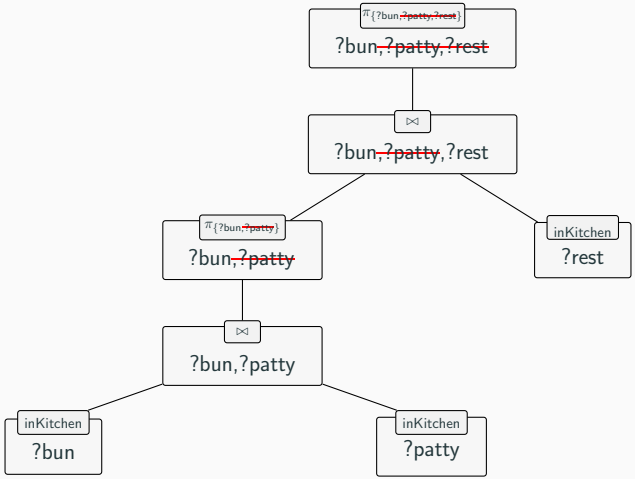
Satisfiability vs. Generating Output

$\mathrm{pre}(\mathsf{make}) = \{\mathsf{inKitchen}(?\mathsf{bun}),\ \mathsf{inKitchen}(?\mathsf{patty}),\ \mathsf{inKitchen}(?\mathsf{rest})\}$



Satisfiability vs. Generating Output

$\mathrm{pre}(\mathsf{make}) = \{\mathsf{inKitchen}(?\mathsf{bun}),\ \mathsf{inKitchen}(?\mathsf{patty}),\ \mathsf{inKitchen}(?\mathsf{rest})\}$



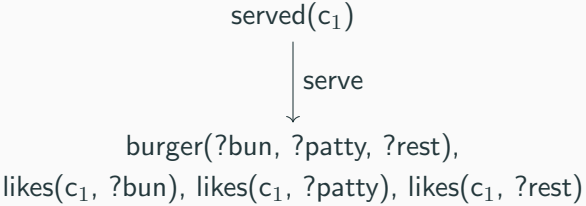
A Lifted Backward Computation of h^{add}

(Backward is a necessity.)

Approach

$$\text{served}(c_1)$$

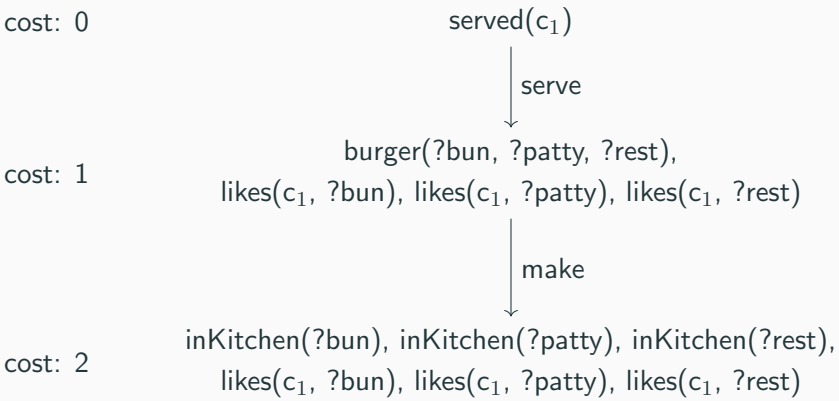
Approach


served(c_1)
serve
burger(?bun, ?patty, ?rest),
likes(c_1, ?bun), likes(c_1, ?patty), likes(c_1, ?rest)

Approach


served(c_1)
serve
burger(?bun, ?patty, ?rest),
likes(c_1, ?bun), likes(c_1, ?patty), likes(c_1, ?rest)
make
inKitchen(?bun), inKitchen(?patty), inKitchen(?rest),
likes(c_1, ?bun), likes(c_1, ?patty), likes(c_1, ?rest)

Approach


cost: 0
served(c_1)
serve
cost: 1
burger(?bun, ?patty, ?rest),
likes(c_1, ?bun), likes(c_1, ?patty), likes(c_1, ?rest)
make
cost: 2
inKitchen(?bun), inKitchen(?patty), inKitchen(?rest),
likes(c_1, ?bun), likes(c_1, ?patty), likes(c_1, ?rest)

Equality to h^{add}

$$h^{add}\ (...) = 0$$

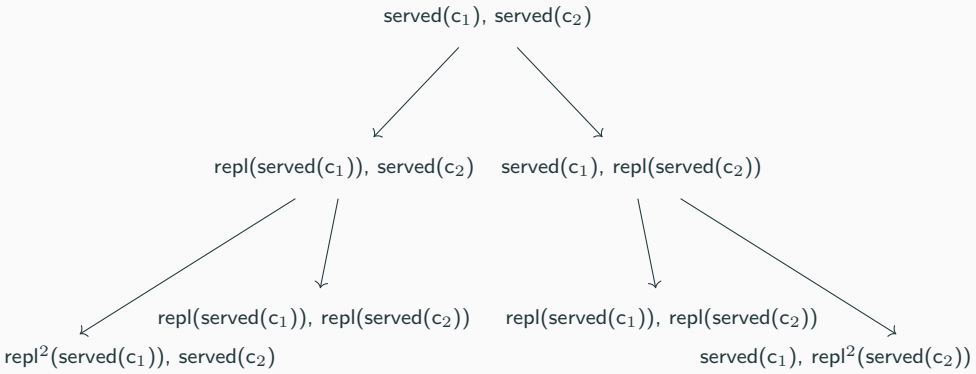
iff satisfied in current state

recursive step: For a multiset of atoms S (starting with $\mathcal{G}$)

original (grounded):	different repl. order:	lifted:
If $S = \{f\}$:		
$\min_{a \text{ achieves } f} c(a) + h^{add}(f)$	$\min_{a \text{ achieves } f \in S}$	$\min_{a \text{ achieves } p(?\vec{x}) \in S}$
	$c(a) +$	$c(a) +$
else:	$h^{add}(S \setminus \{f\}$	$h^{add}(S \setminus \{p(?\vec{x})\}$
$\sum_{f \in S} h^{add}(f)$	$\cup\ \text{pre}(a))$	$\cup\ \text{pre}(remap(p(?\vec{x})))$

More Goals

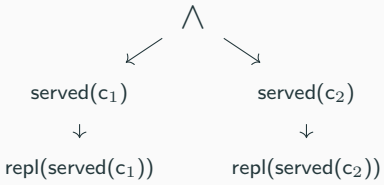
$\boxed{\text{served}(c_1)}_1 \quad , \quad \boxed{\text{served}(c_2)}_2$



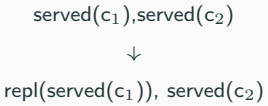
Optimizations

$$\boxed{\text{served}(c_1)}_1 \quad , \quad \boxed{\text{served}(c_2)}_2$$

Opt. 1: Split subsets with disconnected parameters



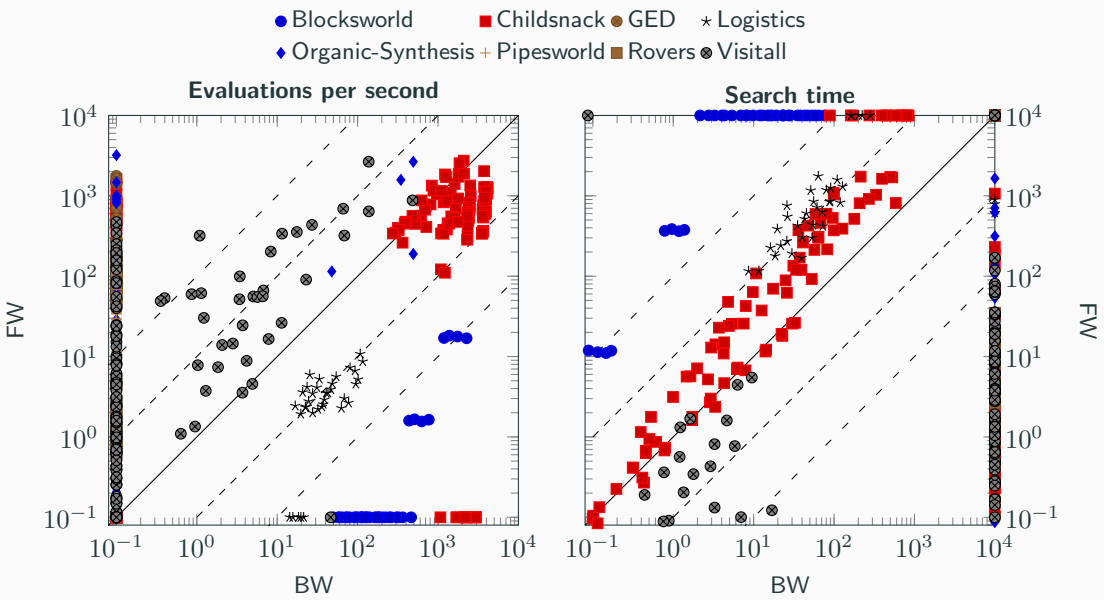
Opt. 2: Replace only elements of unsatisfied subset



Coverage

	Backward (BW)				FW	COMB
	—	Opt. 1	Opt. 2	Opt. 1+2		
Blocksworld (40)	0.0	2.5	5.0	**7.5**	2.5	**7.5**
Childsnack(144)	0.0	7.64	20.83	**24.31**	23.61	22.92
GED (312)	0.0	0.0	0.0	0.0	**43.27**	42.63
Logistics (40)	10.0	20.0	10.0	**90.0**	17.5	87.5
Org.-Synthesis (56)	0.0	0.0	5.36	7.14	**80.36**	**80.36**
Pipesworld (50)	0.0	0.0	0.0	0.0	**40.0**	**40.0**
Rovers (40)	0.0	0.0	0.0	0.0	**27.5**	**27.5**
Visitall (180)	7.78	10.0	17.78	20.56	**65.0**	64.44
Sum (862)	17.78	40.14	58.97	149.5	299.74	**372.85**
Sum orig. (862)	18	38	71	115	370	**396**

Runtime


Blocksworld
Childsnack
GED
Logistics
Organic-Synthesis
Pipesworld
Rovers
Visitall
Evaluations per second
Search time
FW
BW
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}

Thank you :)