Learning Hierarchical Policies by Iteratively Reducing the Width of Sketch Rules



Dominik Drexler[1] Jendrik Seipp[1] Hector Geffner[2,1]

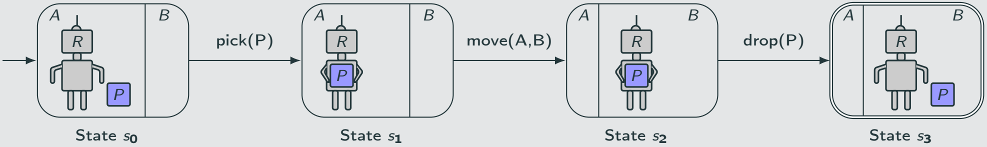
September 8, 2023 at KR conference

[1]Linköping University, Linköping, Sweden,
[2]RWTH Aachen University, Aachen, Germany

Classical Planning

- **Input:**
  1. **Domain** D:
     - Set of predicates
     - Set of action schemas
  2. **Instance** I:
     - Set of objects
     - Set of ground atoms for the initial state s_0 and goal states G
- **Output:**
  - A plan, i.e., sequence of ground actions from s_0 to $s \in G$

Example (Delivery)



Generalized Planning

- **Input:** Class of classical planning problems $\mathcal{Q}$ over common domain D
- **Output:** An algorithm $\mathcal{A}$ that solves any $P \in \mathcal{Q}$ in polynomial time w.r.t. input size

Example (Delivery)

Input: $\mathcal{Q}_{\text{Delivery}}$ consists of all problems of delivering packages, 1-by-1, in a grid.

Output: $\mathcal{A}$ is a hierarchical policy

- Note: has no solution for **intractable** classes (plan existence NP-hard) unless P = NP

Motivation for Hierarchical Policies

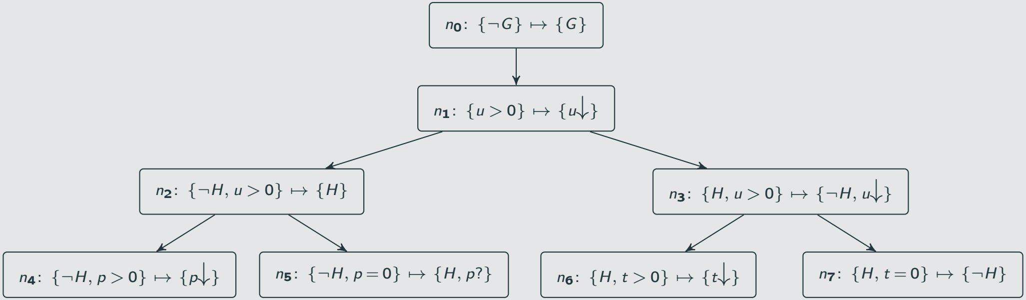
- **Hierarchical policies** involve the execution of sub-policies for achieving **subgoals**
- Subgoals are important in planning where they are exploited as **landmarks**
- Subgoals are important in RL where they appear as **intrinsic rewards**
- The **main challenge** in learning hierarchical policies is how to define a hierarchy of sub-policies for achieving subgoals
- We present a **width-based** characterization of hierarchical policies and how to learn them

Preview: Hierarchical Policy Π_2 for $\mathcal{Q}_{\text{Delivery}}$

Features Φ

- G: all packages delivered?
- H: holding a package?
- u: number of undelivered packages
- p: distance to nearest package
- t: distance to target cell

Hierarchical Policy Π_2



Planning Width [Lipovetzky and Geffner, 2012]

- Background theory of width
  - **Width** $w(P)$ measures the difficulty of a planning problem P
  - **Thm:** if $w(P) = k$ then IW(k) solves P optimally with resources $O(\exp(k))$
- Width in practice
  - Achieving a single goal atom: width is often small (1 or 2)
  - Achieving conjunctive goals: SIW(k) calls IW(k) to **achieve one goal atom at a time**
- Extensions
  - **Policy sketches** is a language that allows to define richer decompositions

Policy Sketches [Bonet and Geffner, 2021]

- A **sketch** R is a set of rules of form $C \mapsto E$ over Boolean and numerical features Φ with sets of feature conditions C and effects E
- **Sketch width** is max width of subproblems from **class of problems** $\mathcal{Q}$:

$$w_R(\mathcal{Q}) = \max_{P \in \mathcal{Q}, s \in S_R(P)} w(P[s, \bigcup_{r \in R} G_r(s)])$$

- **Thm:** if $w_R(\mathcal{Q}) = k$ then $\mathrm{SIW_R}(k)$ solves $P \in \mathcal{Q}$ with resources $O(\exp(k))$

Example (Delivery; 2-width sketch)

$\{u > 0\} \mapsto \{u\downarrow\}$; Decrease # undelivered packages

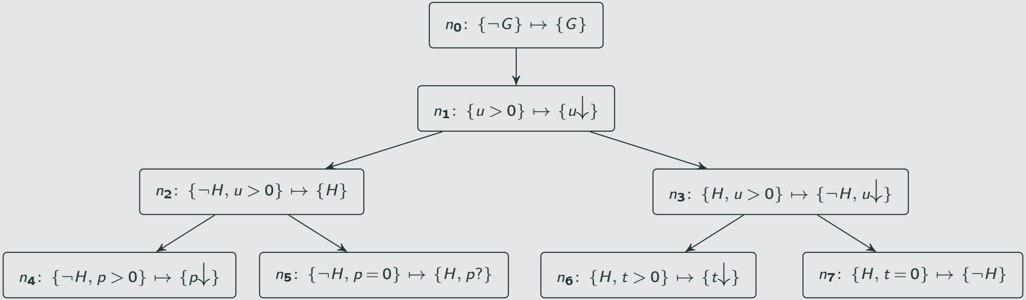
Example (Delivery; 1-width sketch)

$\{\neg H, u > 0\} \mapsto \{H\}$; Get hold of undelivered package
$\{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$; Deliver package

Hierarchical Policies: Formulation

- A hierarchical policy Π for a class of problems $\mathcal{Q}$ is a single rooted tree where every **node** n has a **sketch rule** $r(n)$ with features over $\mathcal{Q}$

Example (Hierarchical policy Π_2 for Delivery)



Valid Hierarchical Policies

- A **valid hierarchical policy** recursively decomposes the target class of problems $\mathcal{Q}$ into easier (smaller width) classes of subproblems $\mathcal{Q}'$
- The decomposition has constraints depending on three types of a node n
  1. **Root node** n:
     - The rule $r(n)$ is $\{\neg G\} \mapsto \{G\}$ where G is true only in the goal of any $P \in \mathcal{Q}$
     - The class of subproblems $\mathcal{Q}_n = \mathcal{Q}$
  2. **Inner node** n:
     - The rules $r(n')$ of the children n' of n define a sketch R whose **sketch width** $w_R(\mathcal{Q}_n)$ is **strictly less** than the **width** $w(\mathcal{Q}_n)$ of class $\mathcal{Q}_n$
     - The class of subproblems $\mathcal{Q}'_n$ is derived from R and $\mathcal{Q}_n$
  3. **Leaf node** n:
     - The **width** $w(\mathcal{Q}_n)$ of class $\mathcal{Q}_n$ is **zero** meaning that each $P \in \mathcal{Q}_n$ is solvable by executing a single action

Learning Hierarchical Policies

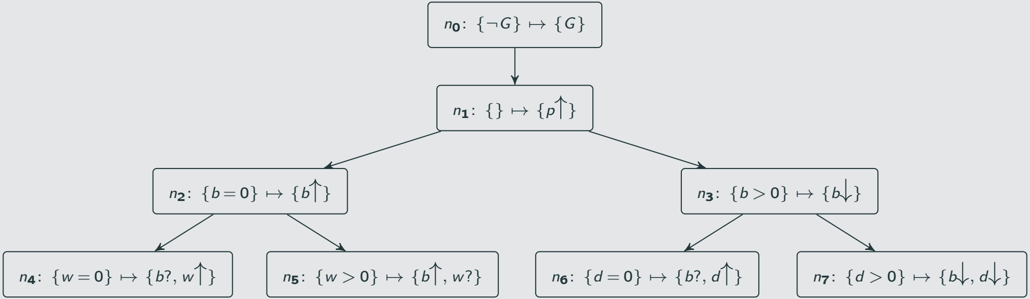
- Input:
  - Set of small training instances: $\mathcal{P} \subset \mathcal{Q}$
  - Width parameter: k
  - Maximum number of rules per learned sketch: m
- Initially, the hierarchical policy Π_k contains a single root node n_0 with $\mathcal{Q}_{n_0} = \mathcal{P}$
- Iteratively refine leaf nodes n with width $w(\mathcal{Q}_n) > 0$ as follows
  - Find sketch R decomposing $\mathcal{Q}_n$ with width $w_R(\mathcal{Q}_n) = w(\mathcal{Q}_n) - 1$
  - Compute set of subproblems $\mathcal{Q}_{n'}$ for each child n' with rule $r(n')$ from R
- We implemented the main operation of learning a sketch in ASP with Clingo [Gebser et al., 2019]

Learned Valid Hierarchical Policy Π_2 for $\mathcal{Q}_{\textbf{Miconic}}$

Features Φ

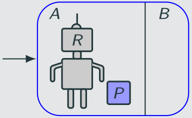
- G: all people served?
- w: # waiting people that are boardable
- d: # people unboardable at destination
- b: # boarded people
- p: # served people

Hierarchical Policy Π_2



Hierarchical Execution

Example (Delivery)



State s_0

$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$

$n_1\colon \{u > 0\} \mapsto \{u\downarrow\}$

$n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$

$n_3\colon \{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$

$n_4\colon \{\neg H, p > 0\} \mapsto \{p\downarrow\}$

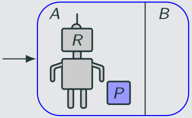
$n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$

$n_6\colon \{H, t > 0\} \mapsto \{t\downarrow\}$

$n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

Example (Delivery)



State s_0

$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$

s_0 $n_1\colon \{u > 0\} \mapsto \{u\downarrow\}$

$n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$

$n_3\colon \{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$

$n_4\colon \{\neg H, p > 0\} \mapsto \{p\downarrow\}$

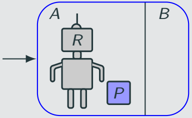
$n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$

$n_6\colon \{H, t > 0\} \mapsto \{t\downarrow\}$

$n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

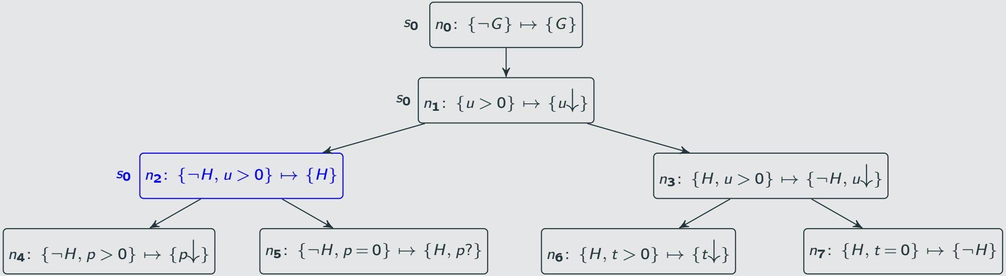
Hierarchical Execution

Example (Delivery)



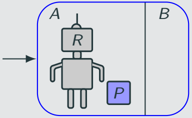
State s_0

$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$



Hierarchical Execution

Example (Delivery)



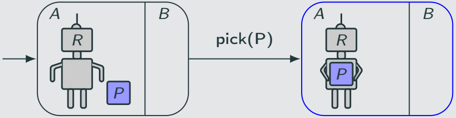
State s_0

$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

- s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$
  - s_0 $n_1\colon \{u > 0\} \mapsto \{u\downarrow\}$
    - s_0 $n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$
      - $n_4\colon \{\neg H, p > 0\} \mapsto \{p\downarrow\}$
      - s_0 $n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$
    - $n_3\colon \{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$
      - $n_6\colon \{H, t > 0\} \mapsto \{t\downarrow\}$
      - $n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

Example (Delivery)



State s_0

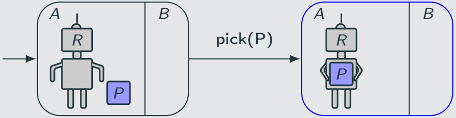
$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1

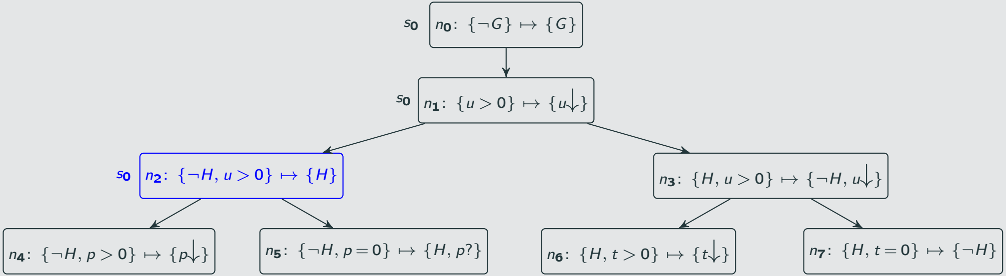
$\{\neg G, H, u = 1, p = \infty, t = 1\}$

- s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$
  - s_0 $n_1\colon \{u > 0\} \mapsto \{u{\downarrow}\}$
    - s_0 $n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$
      - $n_4\colon \{\neg H, p > 0\} \mapsto \{p{\downarrow}\}$
      - s_0 $n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$
    - $n_3\colon \{H, u > 0\} \mapsto \{\neg H, u{\downarrow}\}$
      - $n_6\colon \{H, t > 0\} \mapsto \{t{\downarrow}\}$
      - $n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

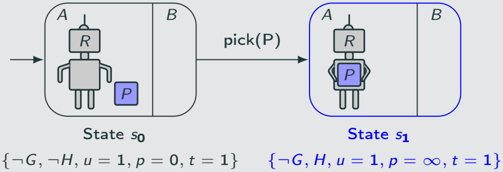
Example (Delivery)

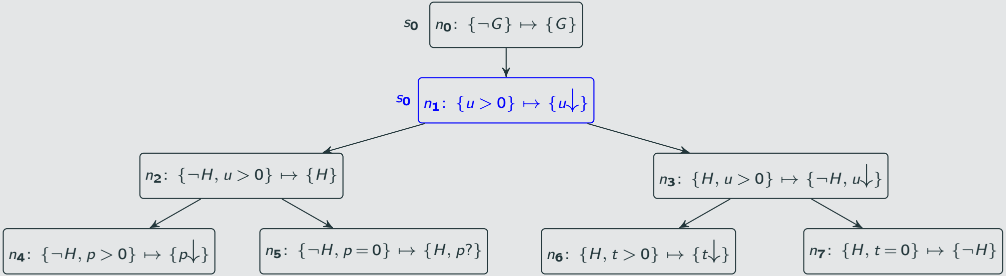


State s_0	State s_1
$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$	$\{\neg G, H, u = 1, p = \infty, t = 1\}$



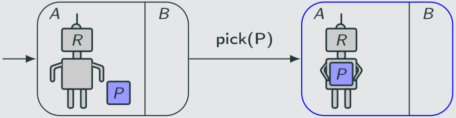
Example (Delivery)





Hierarchical Execution

Example (Delivery)



State s_0

$\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1

$\{\neg G, H, u = 1, p = \infty, t = 1\}$

s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$

s_0 $n_1\colon \{u > 0\} \mapsto \{u\downarrow\}$

$n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$

s_1 $n_3\colon \{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$

$n_4\colon \{\neg H, p > 0\} \mapsto \{p\downarrow\}$

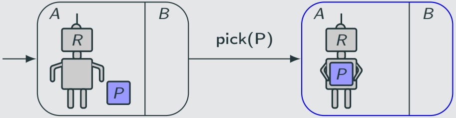
$n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$

$n_6\colon \{H, t > 0\} \mapsto \{t\downarrow\}$

$n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

Example (Delivery)



State s_0 $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1 $\{\neg G, H, u = 1, p = \infty, t = 1\}$

s_0 $n_0\colon\ \{\neg G\} \mapsto \{G\}$

s_0 $n_1\colon\ \{u > 0\} \mapsto \{u\!\downarrow\}$

$n_2\colon\ \{\neg H, u > 0\} \mapsto \{H\}$

s_1 $n_3\colon\ \{H, u > 0\} \mapsto \{\neg H, u\!\downarrow\}$

$n_4\colon\ \{\neg H, p > 0\} \mapsto \{p\!\downarrow\}$

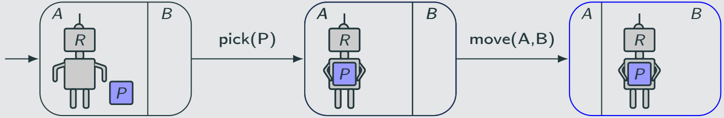
$n_5\colon\ \{\neg H, p = 0\} \mapsto \{H, p?\}$

s_1 $n_6\colon\ \{H, t > 0\} \mapsto \{t\!\downarrow\}$

$n_7\colon\ \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

Example (Delivery)



State s_0: $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1: $\{\neg G, H, u = 1, p = \infty, t = 1\}$

State s_2: $\{\neg G, H, u = 1, p = \infty, t = 0\}$

s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$

s_0 $n_1\colon \{u > 0\} \mapsto \{u\!\downarrow\}$

$n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$

s_1 $n_3\colon \{H, u > 0\} \mapsto \{\neg H, u\!\downarrow\}$

$n_4\colon \{\neg H, p > 0\} \mapsto \{p\!\downarrow\}$

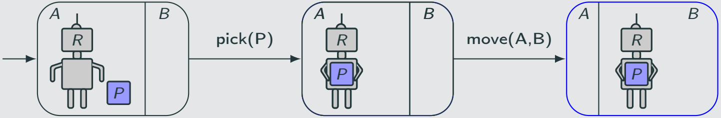
$n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$

s_1 $n_6\colon \{H, t > 0\} \mapsto \{t\!\downarrow\}$

$n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

Example (Delivery)



State s_0 $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1 $\{\neg G, H, u = 1, p = \infty, t = 1\}$

State s_2 $\{\neg G, H, u = 1, p = \infty, t = 0\}$

s_0 $n_0\colon \{\neg G\} \mapsto \{G\}$

s_0 $n_1\colon \{u > 0\} \mapsto \{u\downarrow\}$

$n_2\colon \{\neg H, u > 0\} \mapsto \{H\}$

s_1 $n_3\colon \{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$

$n_4\colon \{\neg H, p > 0\} \mapsto \{p\downarrow\}$

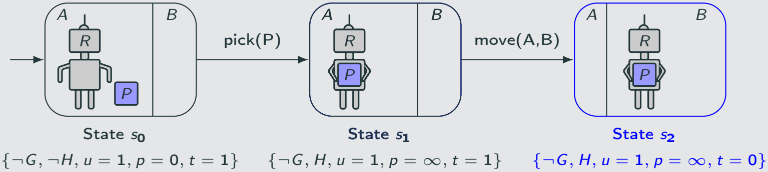
$n_5\colon \{\neg H, p = 0\} \mapsto \{H, p?\}$

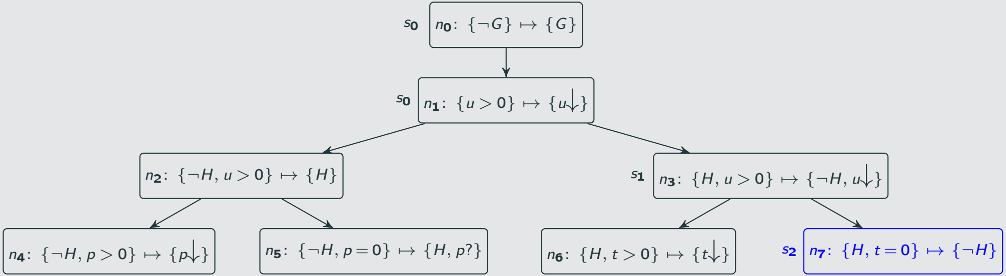
$n_6\colon \{H, t > 0\} \mapsto \{t\downarrow\}$

$n_7\colon \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

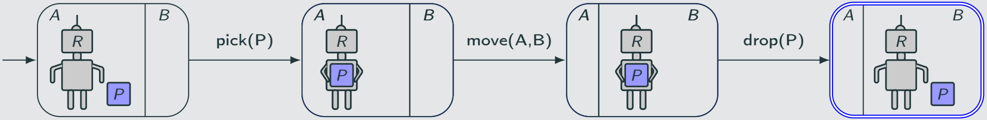
Example (Delivery)





Hierarchical Execution

Example (Delivery)

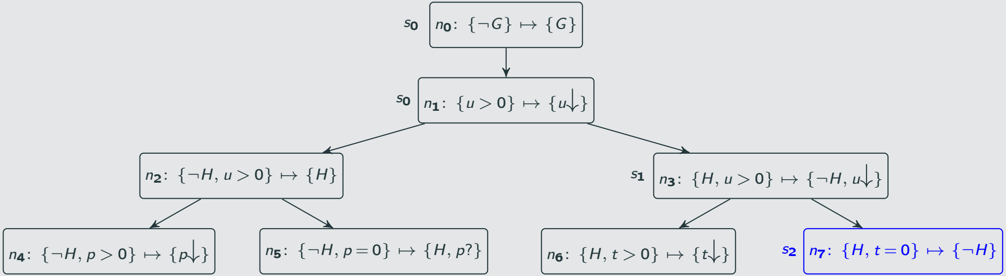


State s_0 $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1 $\{\neg G, H, u = 1, p = \infty, t = 1\}$

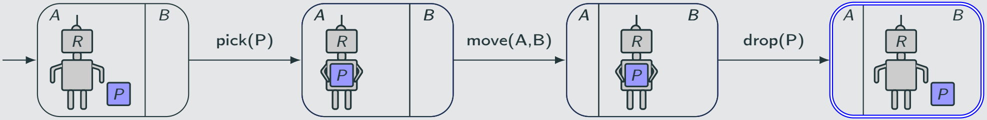
State s_2 $\{\neg G, H, u = 1, p = \infty, t = 0\}$

State s_3



Hierarchical Execution

Example (Delivery)



State s_0 — $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1 — $\{\neg G, H, u = 1, p = \infty, t = 1\}$

State s_2 — $\{\neg G, H, u = 1, p = \infty, t = 0\}$

State s_3 — $\{G, \neg H, u = 0, p = \infty, t = 0\}$

s_0 $n_0\colon\ \{\neg G\} \mapsto \{G\}$

s_0 $n_1\colon\ \{u > 0\} \mapsto \{u\!\downarrow\}$

$n_2\colon\ \{\neg H, u > 0\} \mapsto \{H\}$

s_1 $n_3\colon\ \{H, u > 0\} \mapsto \{\neg H, u\!\downarrow\}$

$n_4\colon\ \{\neg H, p > 0\} \mapsto \{p\!\downarrow\}$

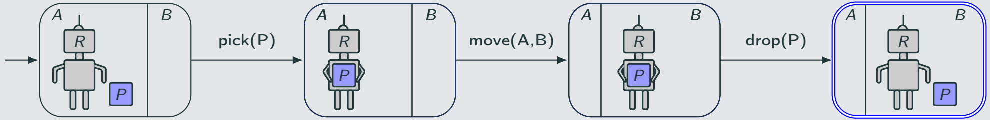
$n_5\colon\ \{\neg H, p = 0\} \mapsto \{H, p?\}$

$n_6\colon\ \{H, t > 0\} \mapsto \{t\!\downarrow\}$

$n_7\colon\ \{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

Example (Delivery)



State s_0 $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1 $\{\neg G, H, u = 1, p = \infty, t = 1\}$

State s_2 $\{\neg G, H, u = 1, p = \infty, t = 0\}$

State s_3 $\{G, \neg H, u = 0, p = \infty, t = 0\}$

s_0 n_0: $\{\neg G\} \mapsto \{G\}$

s_0 n_1: $\{u > 0\} \mapsto \{u\downarrow\}$

n_2: $\{\neg H, u > 0\} \mapsto \{H\}$

n_3: $\{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$

n_4: $\{\neg H, p > 0\} \mapsto \{p\downarrow\}$

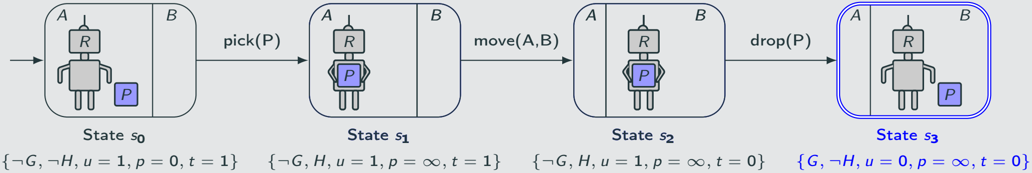
n_5: $\{\neg H, p = 0\} \mapsto \{H, p?\}$

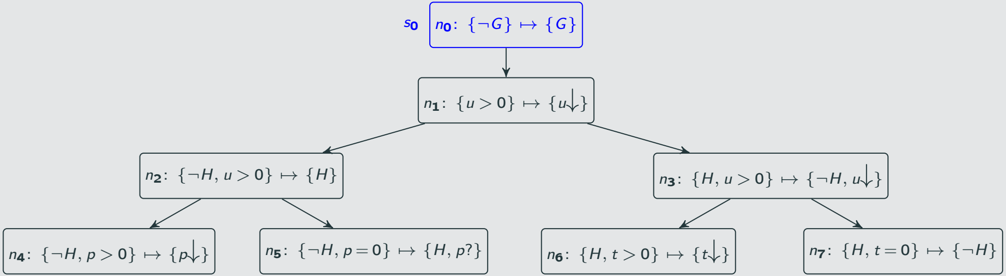
n_6: $\{H, t > 0\} \mapsto \{t\downarrow\}$

n_7: $\{H, t = 0\} \mapsto \{\neg H\}$

Hierarchical Execution

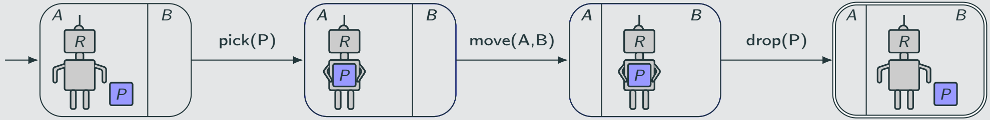
Example (Delivery)





Hierarchical Execution

Example (Delivery)



State s_0: $\{\neg G, \neg H, u = 1, p = 0, t = 1\}$

State s_1: $\{\neg G, H, u = 1, p = \infty, t = 1\}$

State s_2: $\{\neg G, H, u = 1, p = \infty, t = 0\}$

State s_3: $\{G, \neg H, u = 0, p = \infty, t = 0\}$

- n_0: $\{\neg G\} \mapsto \{G\}$
  - n_1: $\{u > 0\} \mapsto \{u\downarrow\}$
    - n_2: $\{\neg H, u > 0\} \mapsto \{H\}$
      - n_4: $\{\neg H, p > 0\} \mapsto \{p\downarrow\}$
      - n_5: $\{\neg H, p = 0\} \mapsto \{H, p?\}$
    - n_3: $\{H, u > 0\} \mapsto \{\neg H, u\downarrow\}$
      - n_6: $\{H, t > 0\} \mapsto \{t\downarrow\}$
      - n_7: $\{H, t = 0\} \mapsto \{\neg H\}$

Experiments: Planning

	LAMA		Π_2	
Domain	Coverage	Time (sec)	Coverage	Time (sec)
Blocks-clear (30)	**30**	32	**30**	29
Blocks-on (30)	**30**	23	**30**	23
Delivery (30)	4	999	**30**	22
Gripper (30)	**30**	2	**30**	2
Miconic (30)	**30**	7	**30**	7
Reward (30)	**30**	381	**30**	39
Spanner (30)	0	–	**30**	11
Visitall (30)	29	189	**30**	783
# Solved domains	5		**8**	

Table 1: Satisficing planning with resource limits 8 GB memory and 30 minutes time.

Summary

- Hierarchical policies are important in planning and RL
- There are no principled methods in generalized planning for learning them
- New width-based formulation: hierarchical policy is a tree with sketch rule $r(n)$ and classes of subproblems $\mathcal{Q}(n)$ for each node n where
  - $\mathcal{Q}(\text{root}) = \mathcal{Q}_{\text{target}}$
  - $\text{width}(\mathcal{Q}(n)) < \text{width}(\mathcal{Q}(\text{parent}(n)))$
  - $\text{width}(\mathcal{Q}(\text{leaf})) = 0$
- Method for learning hierarchical policies with no supervision from small instances
  - Based on ASP/Clingo
  - Uses pool of C_3 features
  - Interesting hierarchical policies obtained for number of benchmarks

Bonet, B. and Geffner, H. (2021).
General policies, representations, and planning width.
In Leyton-Brown, K. and Mausam, editors, *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, pages 11764–11773. AAAI Press.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2019).
Multi-shot ASP solving with clingo.
Theory and Practice of Logic Programming, 19:27–82.

Lipovetzky, N. and Geffner, H. (2012).
Width and serialization of classical planning problems.
In De Raedt, L., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P., editors, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 540–545. IOS Press.