# Course "Automated Planning: Theory and Practice"
# Chapter 12: Delete Relaxation

| | |
|---|---|
| Teacher: | Marco Roveri - `marco.roveri@unitn.it` |
| M.S. Course: | Artificial Intelligence Systems (LM) |
| A.A.: | 2023-2024 |
| Where: | DISI, University of Trento |
| URL: | `https://bit.ly/3zOkGk8` |

Last updated: Monday 6th November, 2023

# TERMS OF USE AND COPYRIGHT

The material (text, figures) in these slides is authored by Jonas Kvarnström and Marco Roveri.

# RE-ACHIEVING CONDITIONS

- To make actions applicable and achieve goals:
  - We often have to re-achieve what was already achieved

  - Example: Driving
    - Initial state: { `(at A) (have-fuel)` }
    - Goal: { `(at D) (have-fuel)` }
    - Actions: `(drive ?x ?y)` - must be in ?x, must follow road from ?x to ?y, must `(have-fuel)`, consume fuel, is no longer in ?x, it is in ?y!

      `(refuel)` - must have no fuel, it make `(have-fuel)` true!
    - Solution:

```
(drive A B)
(refuel)
(drive B C)
(refuel)
(drive C D)
(refuel)
```

# RE-ACHIEVING CONDITIONS (CONT.)

- *Suppose* conditions always remained achieved
  - If `(have-fuel)` is true, it always remains true
  - New Solution:

```
(drive A B)
(drive B C)
(drive C D)
```

Can we exploit this observation to construct a relaxation?

# POSITIVE AND NEGATIVE EFFECTS

- Let's consider the classical representation used in Ghallab et al. [2]:
  - Precondition = set of literals that must be true
  - Goal = set of literals that must be true
  - Effects = set of literals (making atoms true or false)
  - Suppose we have a solution <`A1`,`A2`>:
    - Initially `(have-fuel)`
    - Action `drive` $\Longrightarrow$ requires `(have-fuel)`, makes `(have-fuel)` false
    - Action `refuel` $\Longrightarrow$ requires `(not (have-fuel))`, makes `(have-fuel)` true
  - Symmetry
    - Positive effects can *achieve* positive conditions, *un-achieve* negative conditions
    - Negative effects can *achieve* negative conditions, *un-achieve* positive conditions

# POSITIVE AND NEGATIVE EFFECTS (CONT.)

- Let's consider the PDDL's plain `:strips` level
  - Forbids negative preconditions/goals
    - Precondition = set of atoms (no negations!)
    - Goal = set of atoms (no negations!)
    - Effects = set of literals (making atoms true or false)
  - In this setting:
    - Positive effects are never "problematic":
      Adding more facts to the state can only make *more* preconds/goals satisfied
    - Only negative effects can "un-achieve" goals or preconditions
    - And negative effects can only "un-achieve" goals or preconditions:
      We never *need* them

# DELETE RELAXATION

- Assuming positive conditions, let's remove all negative effects
  - Example: `(unstack ?x ?y)`
    - Before transformation:

```
:precondition (and (handempy) (clear ?x) (on ?x ?y))
:effect       (and (not (handempy)) (holding ?x) (not (clear ?x))
                   (clear ?y) (not (on ?x ?y)))
```

    - After transformation:

```
:precondition (and (handempy) (clear ?x) (on ?x ?y))
:effect       (and (holding ?x)
                   (clear ?y))
```

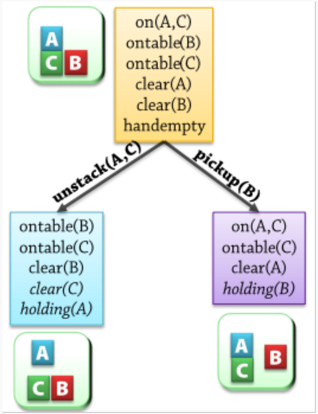
  - A fact that is true stays true

Is this a relaxation?

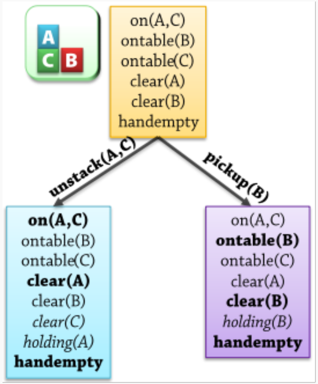
  - Positive conditions $\Longrightarrow$
    - No solution can *depend on a fact being false* in a visited state
    - No solution can *disappear* because we avoid making facts false

# DELETE RELAXATION: EXAMPLE

## STS FOR THE ORIGINAL PROBLEM



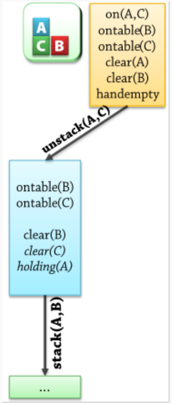
## DELETE-RELAXED STRIPS PROBLEM



No physical "meaning"!

# DELETE RELAXATION: EXAMPLE (CONT.)

## STS FOR THE ORIGINAL PROBLEM



Initial state does not change

$=$

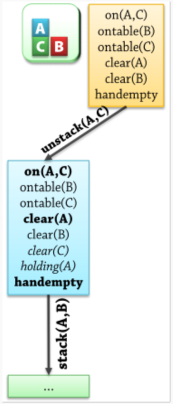
Same "origin", fewer facts removed

$\subseteq$

Different "origin" but same *action sequence*, fewer facts removed

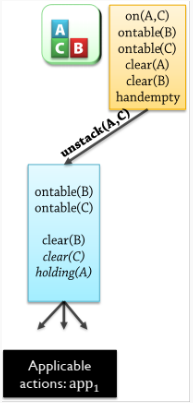
$\subseteq$

## DELETE-RELAXED STRIPS PROBLEM



# DELETE RELAXATION: EXAMPLE (CONT.)

## STS FOR THE ORIGINAL PROBLEM

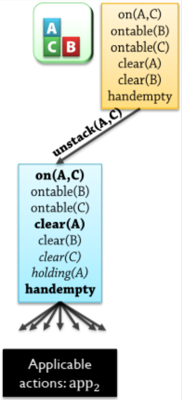


$\subseteq$

No action requires the *absence* of a fact

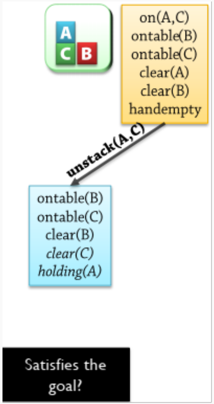
$\subseteq$

## DELETE-RELAXED STRIPS PROBLEM



# Delete Relaxation: Example (cont.)

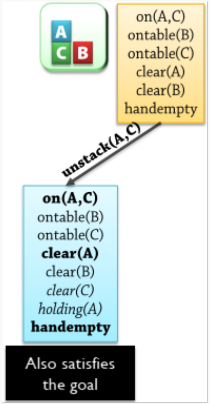
### STS for the original problem



No goal requires the *absence* of a fact

⟹

### Delete-relaxed STRIPS problem



# DELETE RELAXATION

- Negative effects are also called "delete effects"
  - They delete facts from the state
- So this is called delete relaxation
  - "*Relaxing* the problem by getting rid of the *delete effects*"
- "Relaxed plan for P" = plan for the delete-relaxed version of P

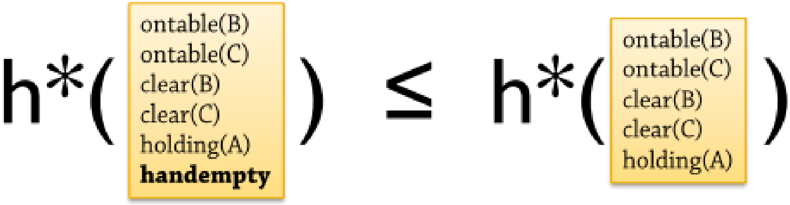
Delete relaxation does not mean that we "delete the relaxation" (anti-relax)!

Delete relaxation is only a relaxation if preconditions and goals are positive!

# Delete Relaxation (cont.)

- Since solutions are preserved when action are added:

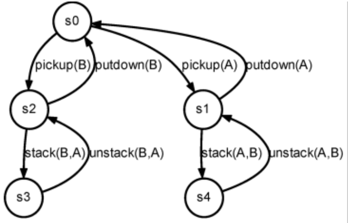
A state where additional facts are true can never be "worse"!
(Given positive preconds/goals)



h*(
ontable(B)
ontable(C)
clear(B)
clear(C)
holding(A)
**handempty**
) ≤ h*(
ontable(B)
ontable(C)
clear(B)
clear(C)
holding(A)
)

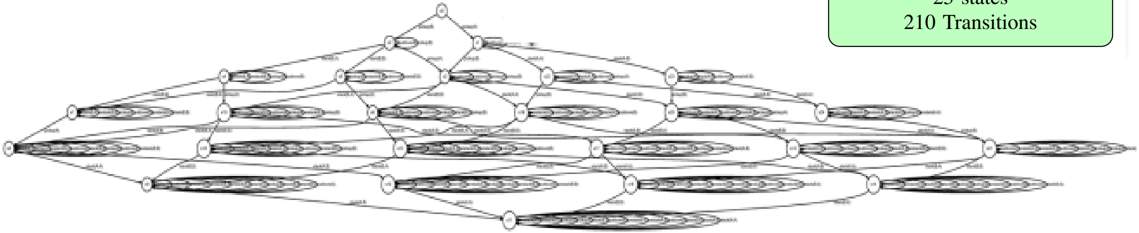
Given two states (sets of true atoms) $s_1$, $s_2$:
$s_2 \subset s_1 \rightarrow h^*(s_2) \geq h^*(s_1)$

# Reachable State Space: BW size 2



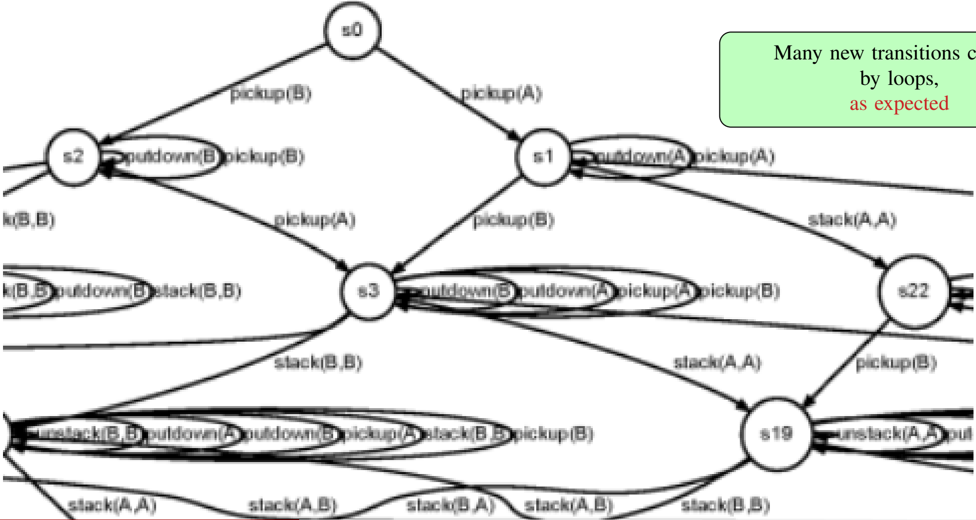
5 states
8 Transitions

⇓



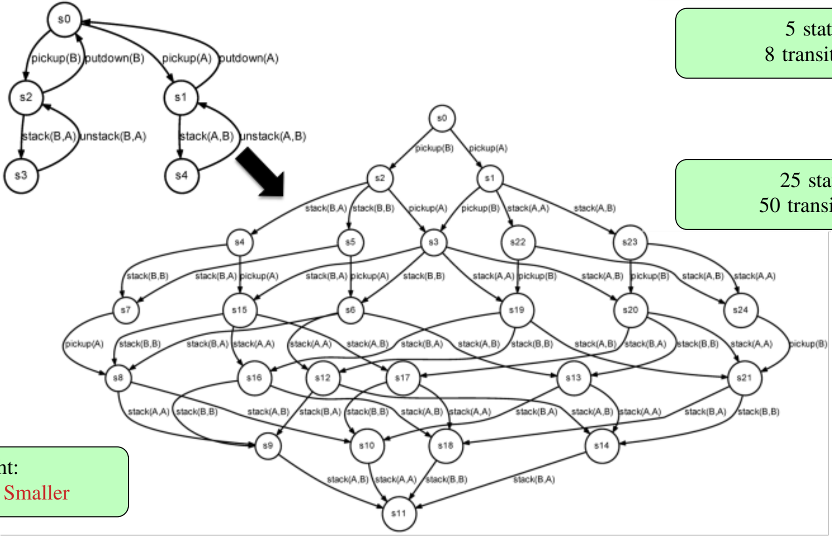
25 states
210 Transitions

# Reachable State Space: BW size 2 - Detailed view



Many new transitions caused
by loops,
as expected

# DELETE RELAXED: "LOOPS" REMOVED



5 states
8 transitions

25 states
50 transitions

Insight:
Relaxed ≠ Smaller

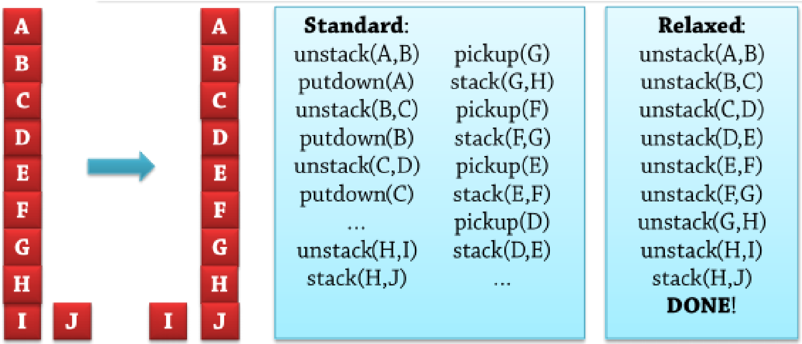
# OPTIMAL DELETE RELAXATION HEURISTIC

- If only delete relaxation is applied:
  - We can calculate the optimal delete relaxation heuristic, $h^{+}(n)$
  - $h^{+}(n) =$ the cost of an optimal solution
    to a delete-relaxed problem
    starting in node $n$

# ACCURACY OF $h^+$ IN SELECTED DOMAINS

- How close is $h^+(n)$ to the true goal distance $h^*(n)$?
  - Worst case asymptotic accuracy as problem size approaches infinity:
    - Blocks world: $\frac{1}{4} \implies h^+(n) \geq \frac{1}{4}h^*(n)$

Optimal plans in delete-relaxed Blocks World can be down to 25% of the length of optimal plans in "real" Blocks World and goals are positive!



**Standard:**

unstack(A,B)
putdown(A)
unstack(B,C)
putdown(B)
unstack(C,D)
putdown(C)
...
unstack(H,I)
stack(H,J)
pickup(G)
stack(G,H)
pickup(F)
stack(F,G)
pickup(E)
stack(E,F)
pickup(D)
stack(D,E)
...

**Relaxed:**

unstack(A,B)
unstack(B,C)
unstack(C,D)
unstack(D,E)
unstack(E,F)
unstack(F,G)
unstack(G,H)
unstack(H,I)
stack(H,J)
**DONE!**

# ACCURACY OF $h^+$ IN SELECTED DOMAINS

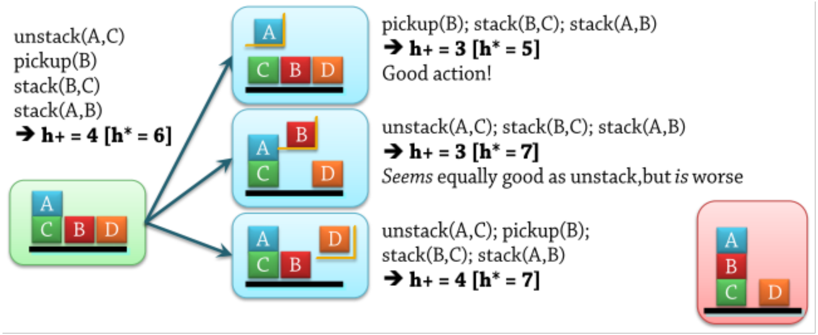
- How close is $h^+(n)$ to the true goal distance $h^*(n)$?
  - Worst case asymptotic accuracy as problem size approaches infinity:

| | | |
|---|---|---|
| Blocks world: | $\frac{1}{4}$ | $\Longrightarrow h^+(n) \geq \frac{1}{4} h^*(n)$ |
| Gripper domain: | $\frac{2}{3}$ | (single robot moving balls) |
| Logistics domain: | $\frac{3}{4}$ | (move packages using trucks, airplanes) |
| Miconic STRIPS: | $\frac{6}{7}$ | (elevators) |
| Miconic-Simple-ADL: | $\frac{3}{4}$ | (elevators) |
| Schedule: | $\frac{1}{4}$ | (job shop scheduling) |
| Satellite: | $\frac{1}{2}$ | (satellite observations) |

  - Details:
    - Malte Helmert and Robert Mattmüller
      *Accuracy of Admissible Heuristic Functions in Selected Planning Domains* [4]

# Example of Accuracy

- How close is $h^{+}(n)$ to the true goal distance $h^{*}(n)$?
  - In practice: Also depends on the problem instance!



unstack(A,C)
pickup(B)
stack(B,C)
stack(A,B)
➔ **h+ = 4 [h* = 6]**

pickup(B); stack(B,C); stack(A,B)
➔ **h+ = 3 [h* = 5]**
Good action!

unstack(A,C); stack(B,C); stack(A,B)
➔ **h+ = 3 [h* = 7]**
*Seems* equally good as unstack,but *is* worse

unstack(A,C); pickup(B);
stack(B,C); stack(A,B)
➔ **h+ = 4 [h* = 7]**

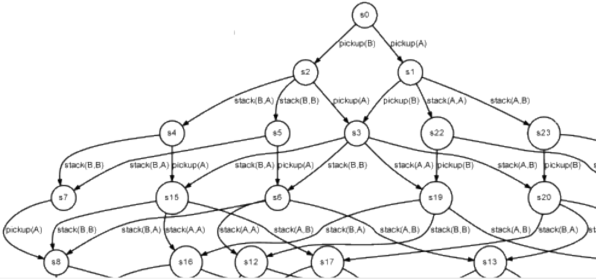
- Performance also depends on the search strategy
  - How sensitive it is to specific types of inaccuracy

# COMPUTING $h^+$

- Is $h^+$ easier to compute than $h^*$?
  - $h^*$ = length of optimal plan for arbitrary planning problem
    - Supports negative effects
    - If we can execute either `a1;a2` or `a2;a1`:
      `a1` removes $p$, `a2` adds $p \Longrightarrow$ net result: add $p$
      `a2` adds $p$, `a1` removes $p \Longrightarrow$ net result: remove $p$
      Both orders must be considered
  - $h^+$ = length of optimal plan after removing negative effects
    - If we can execute either `a1;a2` or `a2;a1`:
      Must lead to the same state (add `a1` before `a2`, or `a2` before `a1`)
      Sufficient to consider one order - simpler?
  - Incomplete analysis
    - But the worst case for $h^+$ is easier than the worst case for $h^*$

# Computing $h^*$ (cont.)

- Still difficult to calculate in general!
  - NP-equivalent (reduced from PSPACE-equivalent)
    - Since you must find optimal solutions to the relaxed problem
  - Even a constant-factor approximation is NP-equivalent to compute!
    - Finding $h(n)$ so that $\forall n.h(n) \geq c \cdot h^+(n)$
- Therefore, rarely used "as is"
  - But forms the basis of many other heuristics



# REFERENCES I

[1] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN 9781608459698. doi: 10.2200/S00513ED1V01Y201306AIM022. URL `https://doi.org/10.2200/S00513ED1V01Y201306AIM022`.

[2] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning - theory and practice*. Elsevier, 2004. ISBN 978-1-55860-856-6. 5

[3] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016. ISBN 978-1-107-03727-4. URL `http://www.cambridge.org/de/academic/subjects/computer-science/artificial-intelligence-and-natural-language-processing/automated-planning-and-acting?format=HB`.

[4] Malte Helmert and Robert Mattmüller. Accuracy of admissible heuristic functions in selected planning domains. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 938–943. AAAI Press, 2008. URL `http://www.aaai.org/Library/AAAI/2008/aaai08-149.php`. 19