Linköping University

Automated Planning

Relaxed Planning Graphs

Jonas Kvarnström
Department of Computer and Information Science
Linköping University

jonas.kvarnstrom@liu.se – 2019

Basic Idea

jonkv@ida

Apply delete relaxation

Create a graph
efficiently representing many ways
of achieving the goal
in the relaxed problem

Extract one possible solution π from the graph
(not necessarily optimal!)

$$h_{FF}(\mathrm{n}) = |\pi|$$
or
$$h(\mathrm{n}) = \mathrm{cost}(\pi) \geq h^{+}(s)$$

Running Example

jonkv@ida

- **Running example** due to Dan Weld (modified):
  - Prepare and serve a surprise dinner,
    take out the garbage,
    and make sure the present is wrapped before waking your sweetheart!

s_0 = {clean, garbage, asleep}

g = {clean, ¬garbage, served, wrapped}

Action	Preconds	Effects
cook()	clean	dinner
serve()	dinner	served
wrap()	asleep	wrapped
carry()	garbage	¬garbage, ¬clean
roll()	garbage	¬garbage, ¬asleep
clean()	¬clean	clean

Running Example (2)

jonkv@ida

- Let's apply **delete relaxation**
  - Prepare and serve a surprise dinner,
    take out the garbage,
    and make sure the present is wrapped before waking your sweetheart!

s_0 = {clean, garbage, asleep}
g = {clean, served, wrapped}

Action	Preconds	Effects	
cook()	clean	dinner	
serve()	dinner	served	
wrap()	asleep	wrapped	
carry()	garbage	–	**Pointless actions:**
roll()	garbage	–	**No effects!**
clean()	–	clean	

RPG 1: Propositions

jonkv@ida

- Now we want to find a **relaxed plan**
  - What is **true** initially?
    ➔ first **proposition level** in a **relaxed planning graph**

Proposition level 0

garbage

clean

asleep

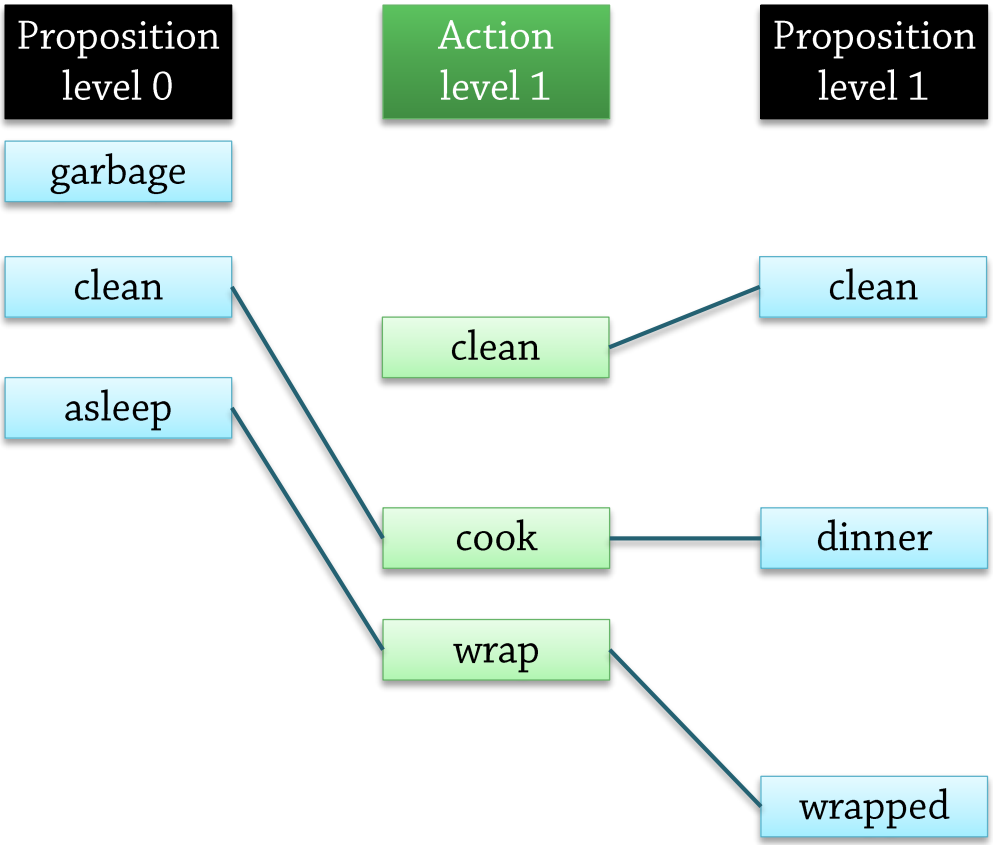
Planning Graph introduced by GraphPlan

Heuristics based on Relaxed Planning Graph pioneered by FF (FastForward)

RPG 2: Actions and Effects

jonkv@ida

- Next step:
  - Which **actions** could be executed first?
  - Which **effects** would we get?



Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean

Build a graph with actions linking to preconds and effects

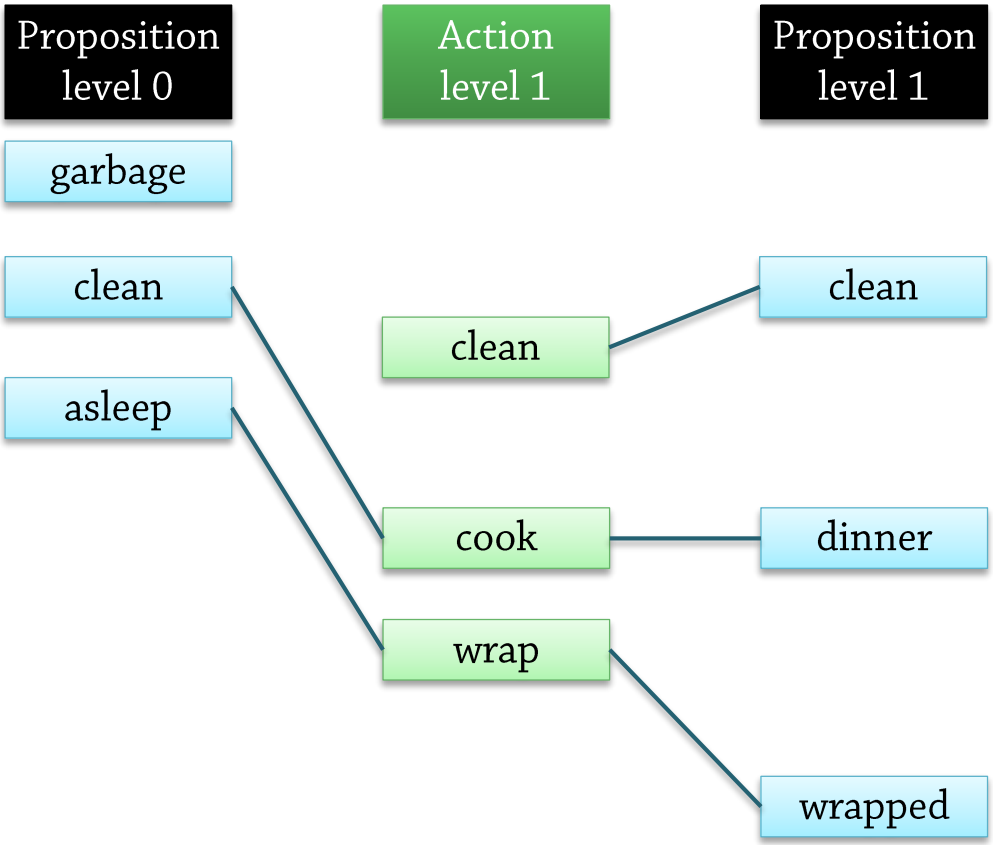
Assumes conjunctive preconds, effects!

RPG 3: Interpretation

jonkv@ida

- Here we see:
  - Which propositions can we **make** true in one step?
  - Which actions would we need?

Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean



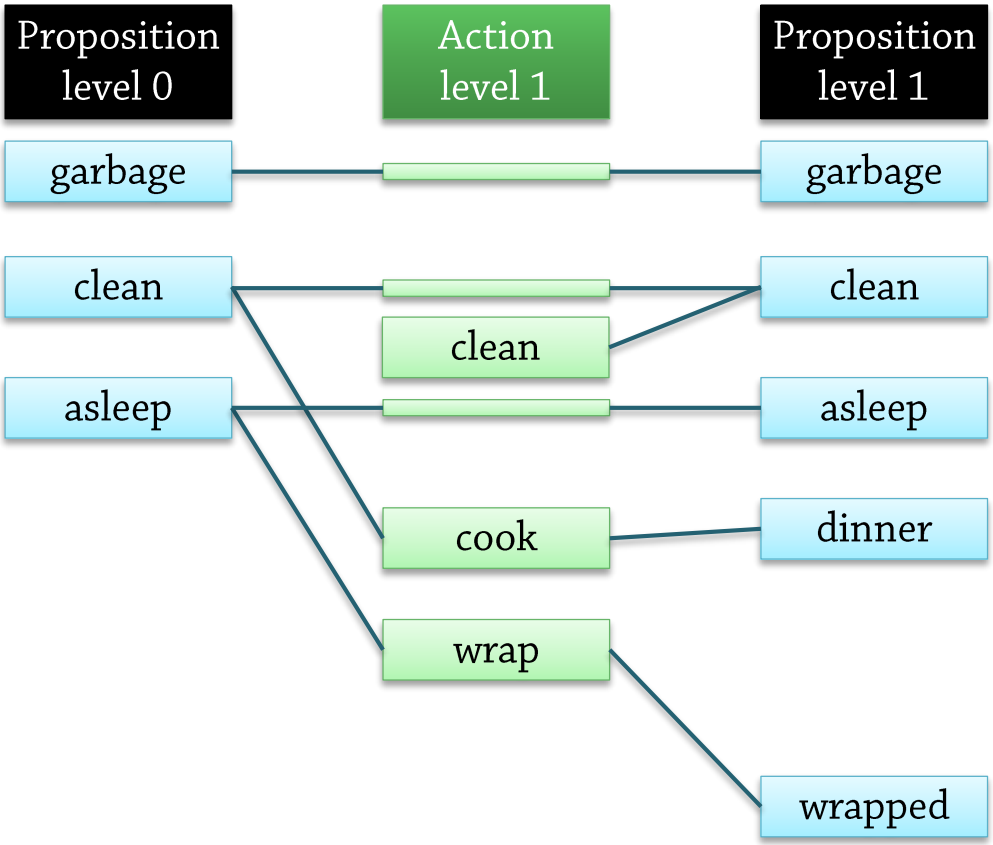
But wait!

Prop level 1 is missing "garbage", which could *remain* true from prop level 0…

RPG 4: Maintenance Actions

jonkv@ida

- Solution: ”No-op” or ”maintenance” actions
  - One for each proposition (fact) that exists
  - No need to treat *persistence* separately



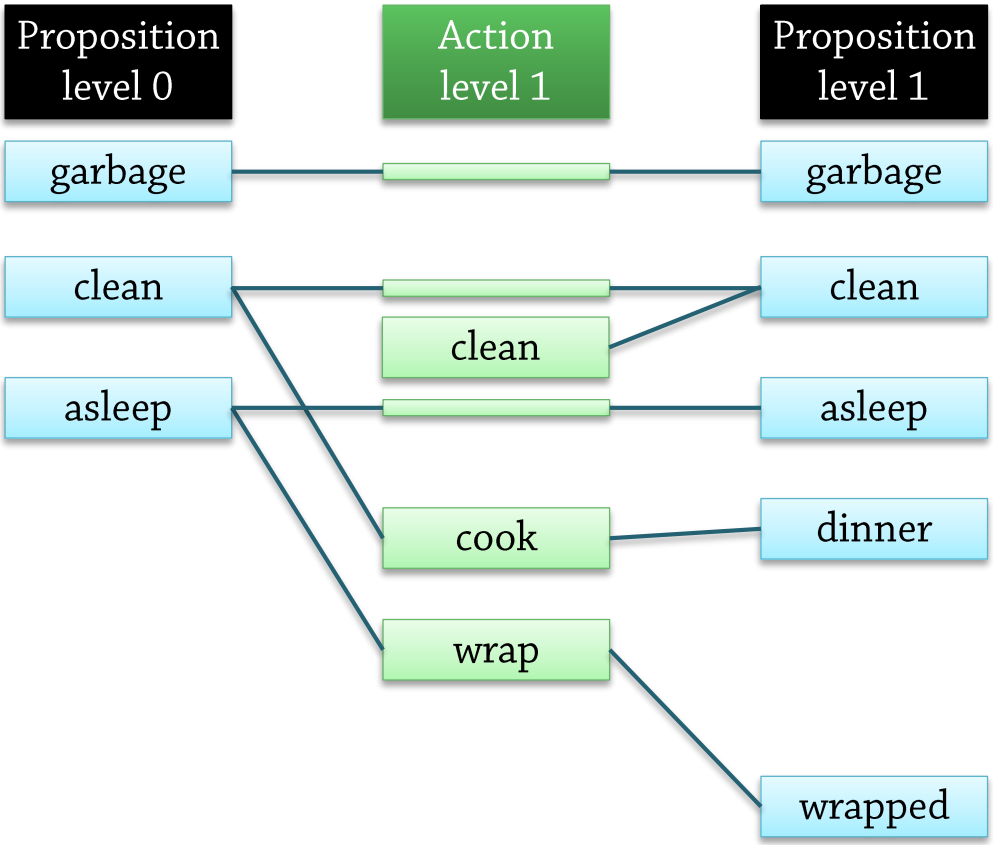
Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean

noop-clean
precond: clean
effect: clean
noop-garbage
…

RPG 5

jonkv@ida

- What does this **mean** for the **actions**?



Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean
noop…		

First action could be clean, cook or wrap

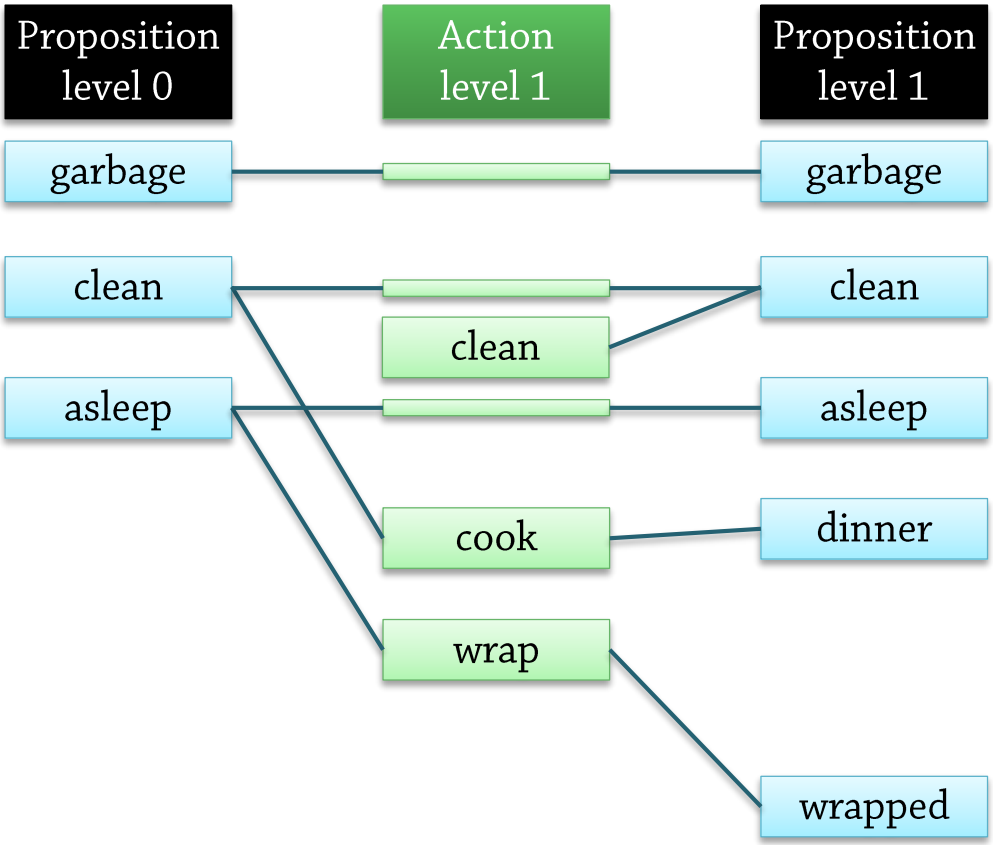
First actions could be any combination of clean, cook or wrap

None can invalidate the others' preconditions: No negative effects!

RPG 6

jonkv@ida

- What does this **mean** for the **facts**?



Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean
noop…		

We can **choose** actions that achieve **any** subset of **{ garbage, clean, asleep, dinner, wrapped }** and we don't have to care about their order!

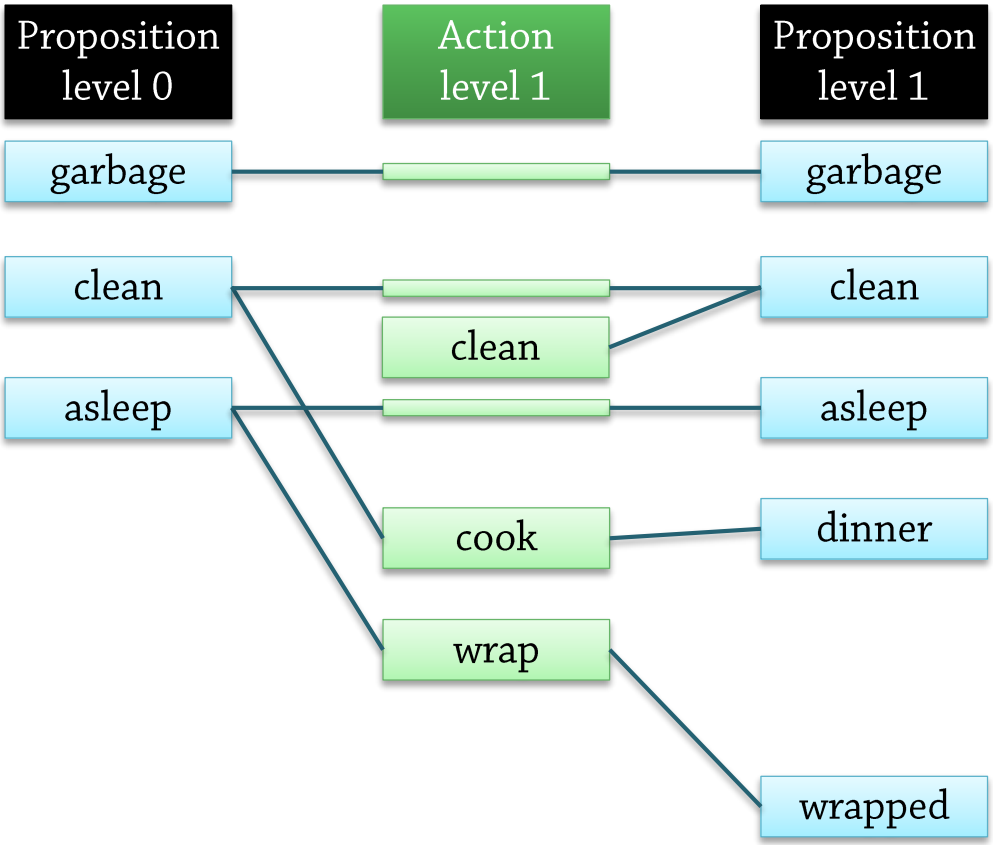
Given delete relaxation!

In **reality**, negative effects interfere… but we aim for a heuristic!

RPG 7

jonkv@ida

- Can we reach the goal now?
  - No, can't achieve **served** yet...



Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean
noop…		

We need *dinner* **before** *serve*

Level 1 is only for actions whose preconds are true at the start

Chains of dependencies

➔

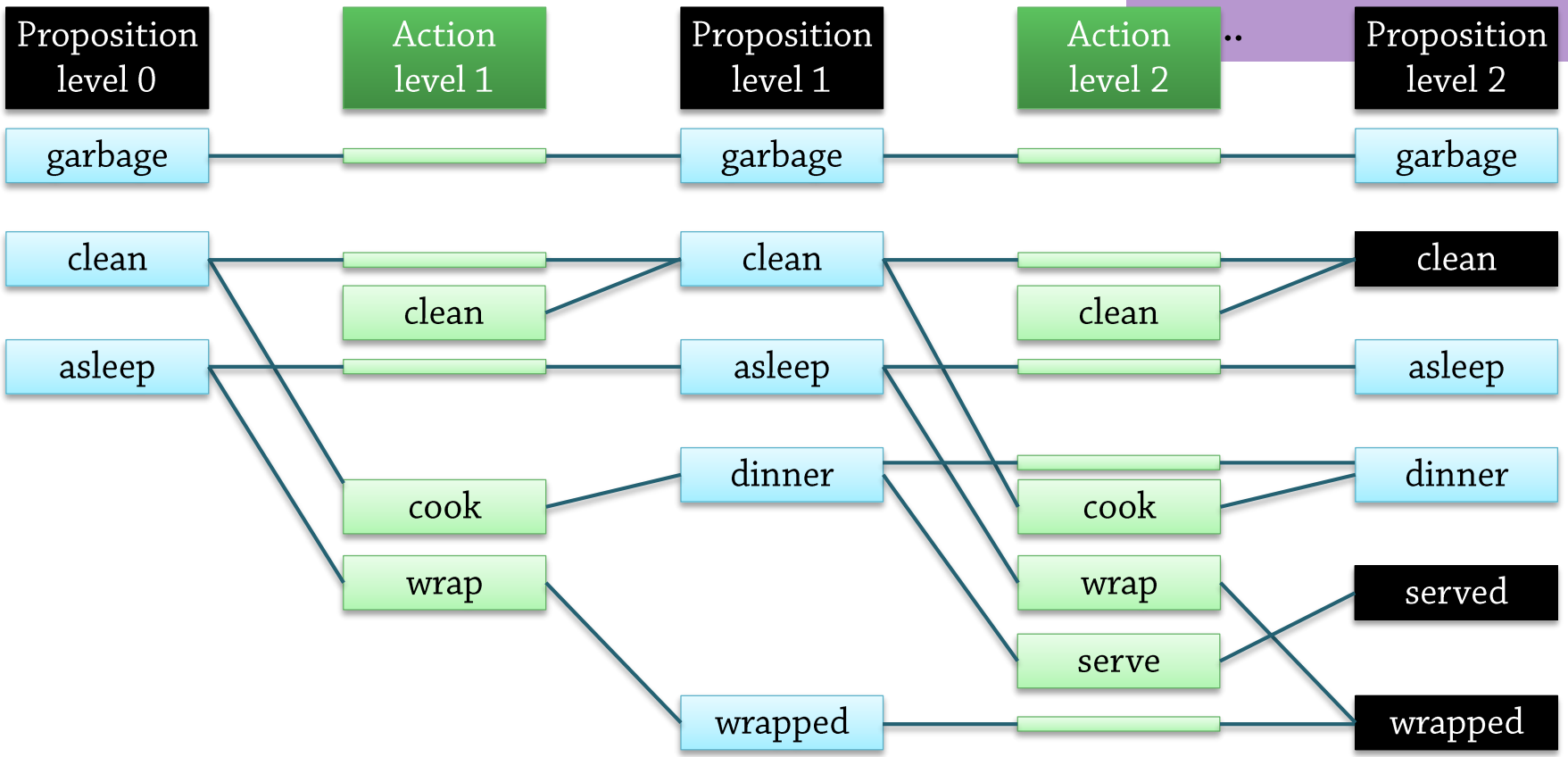
Many levels in the graph

RPG 8

jonkv@ida

- Let's add another level
  - Achieves all goals
  - Can select actions from the graph

Action	Prec	Effects
cook	clean	dinner
serve	dinner	served
wrap	asleep	wrapped
clean	–	clean



g = {clean, served, wrapped}

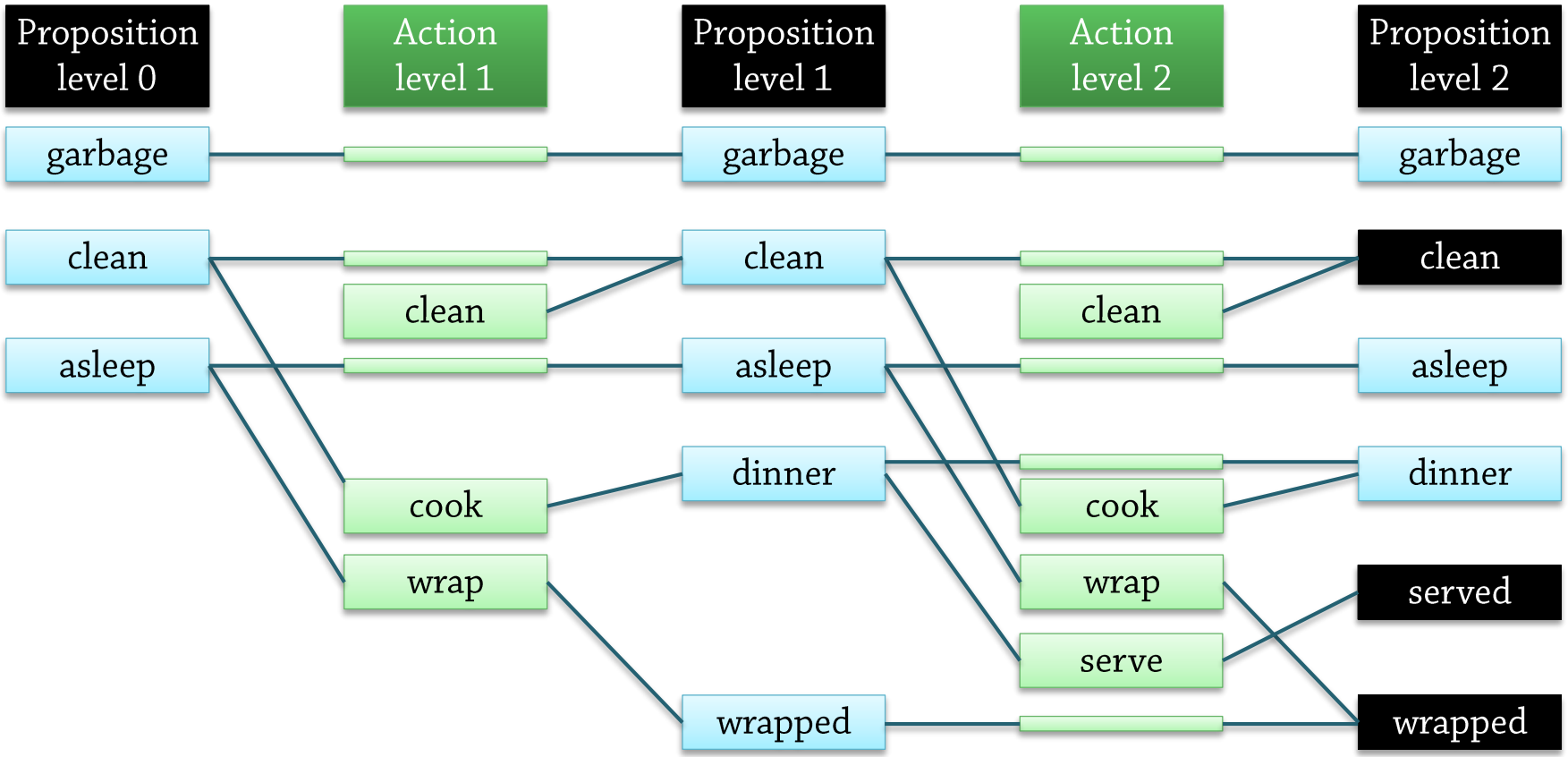
Heuristic Function:
Based on Solution Extraction

Solution Extraction

jonkv@ida

- For each goal fact, choose one action achieving it
  - clean ➔ noop-clean **or** clean
  - served ➔ serve
  - wrapped ➔ wrap **or** noop-wrapped

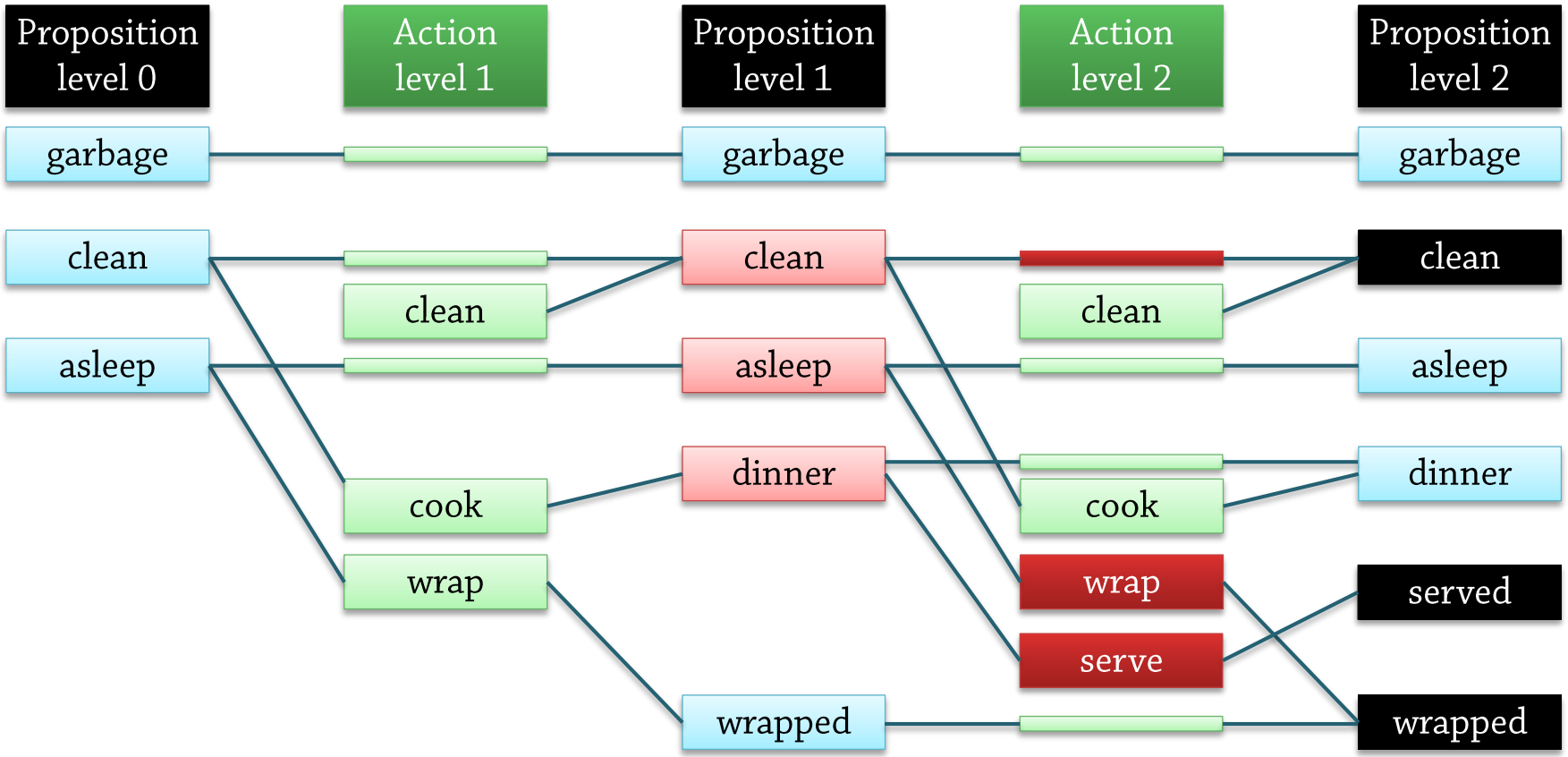
2*1*2 = 4 alternatives!
All **work**,
but some may result in
shorter plans… Why?



Solution Extraction (2)

jonkv@ida

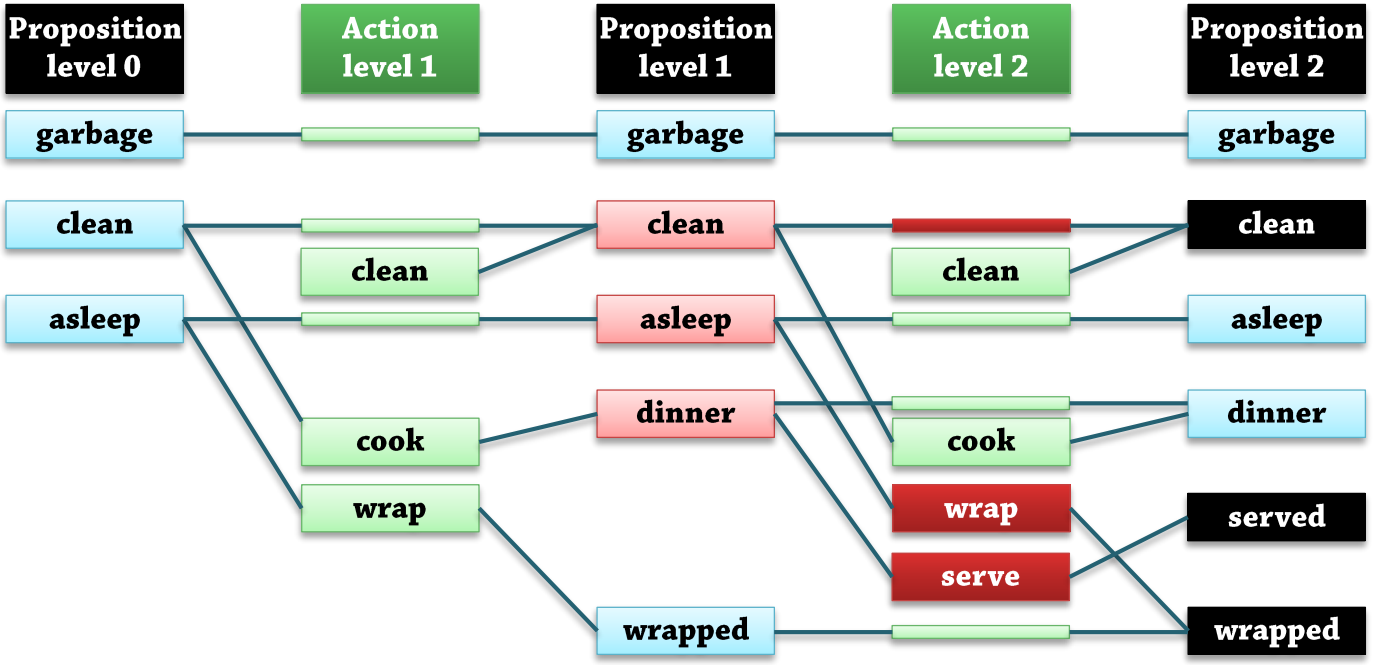
- For all **selected** actions in level 2 (marked red):
  - Must first achieve their preconditions!
  - This is a new goal to achieve by selecting actions at level 1



Solution Extraction (3)

jonkv@ida

- Unlike backward search in ***goal space***:
  - Simpler concept of relevance: No negative effects that interfere
  - At each level, select **sets** of actions, together achieving **all** goal facts
    - No need to consider "what the single selected action didn't achieve"
    - Simpler backward chaining: Instead of γ^{-1}, just conjoin preconds of selected actions
  - Already built a graph from the initial state
    - And no possibility of negative effect interference ➔ we *can* reach the initial state

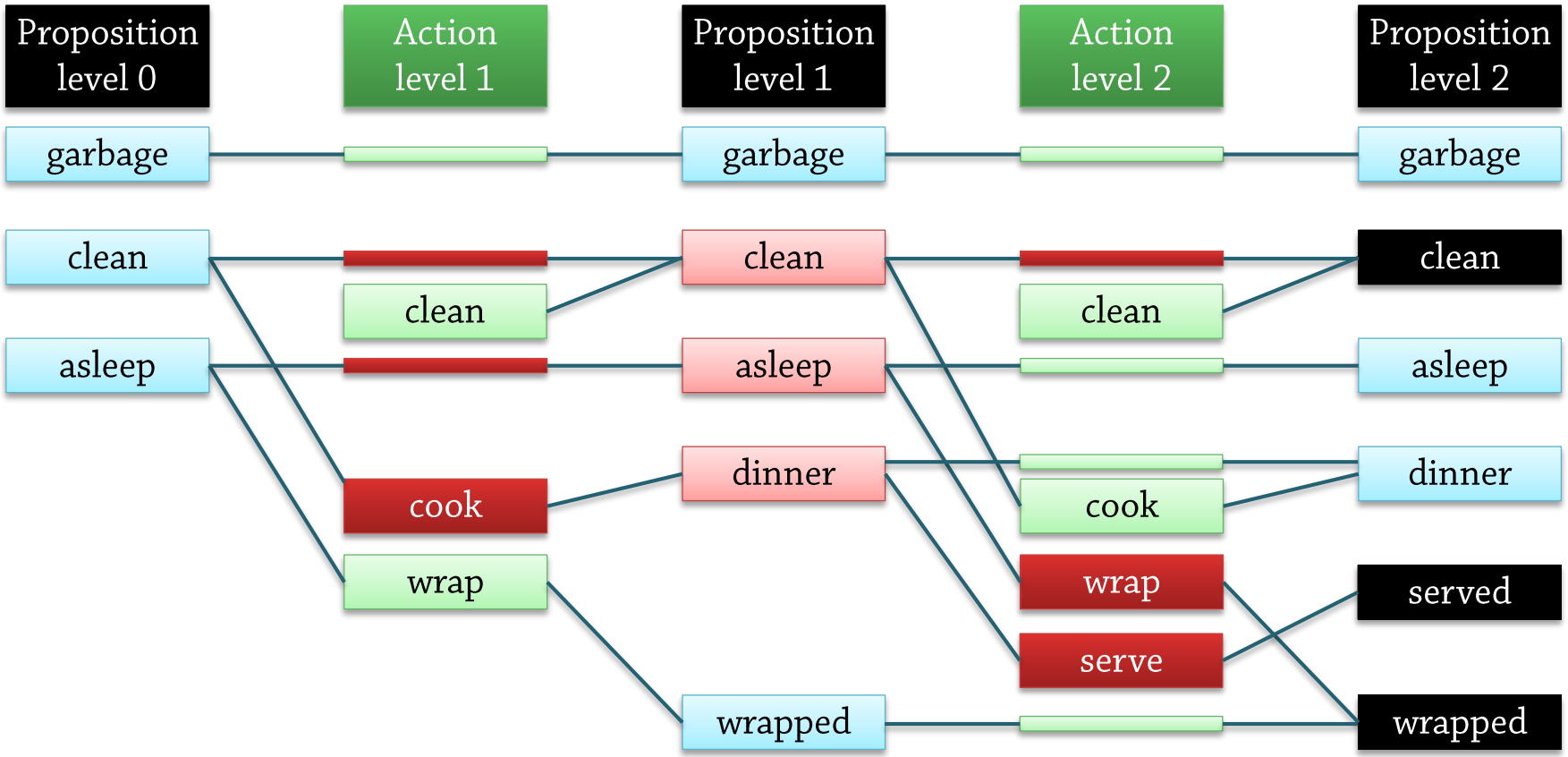


Solution Extraction (4)

jonkv@ida

- Final relaxed plan:
  - First cook
  - Then wrap and serve, in some order
  - $h_{FF}(n) = 3$, assuming the algorithm chose this order!

Relaxed plan:
Not a solution to the **original** problem!

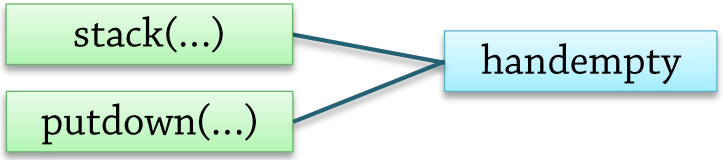


Solution Extraction (5)

jonkv@ida

- Does the choice of actions matter?
  - Choosing a noop action may mean fewer actual actions

  - Different actions chosen at one level:
    - May lead to different actions at previous levels
    - Which then leads to different preconds to satisfy…



  - And so on…

Actual solution extraction algorithm in FF uses backward search in the RPG + *heuristics* for this search!

- Not equivalent to $h^+(n)$: Would require an **optimal** relaxed plan
  - Would have to test different action selections
  - May require additional **levels** (with fewer selected actions per level)

Properties of Relaxed Planning Graphs

jonkv@ida

- The **relaxed planning graph** considers **positive** interactions
  - For example, when one action achieves multiple goals
  - Ignores **negative** interactions

  - Can extract a **Graphplan-optimal** relaxed plan
    (minimal number of levels / ”parallel” steps)
    in **polynomial** time

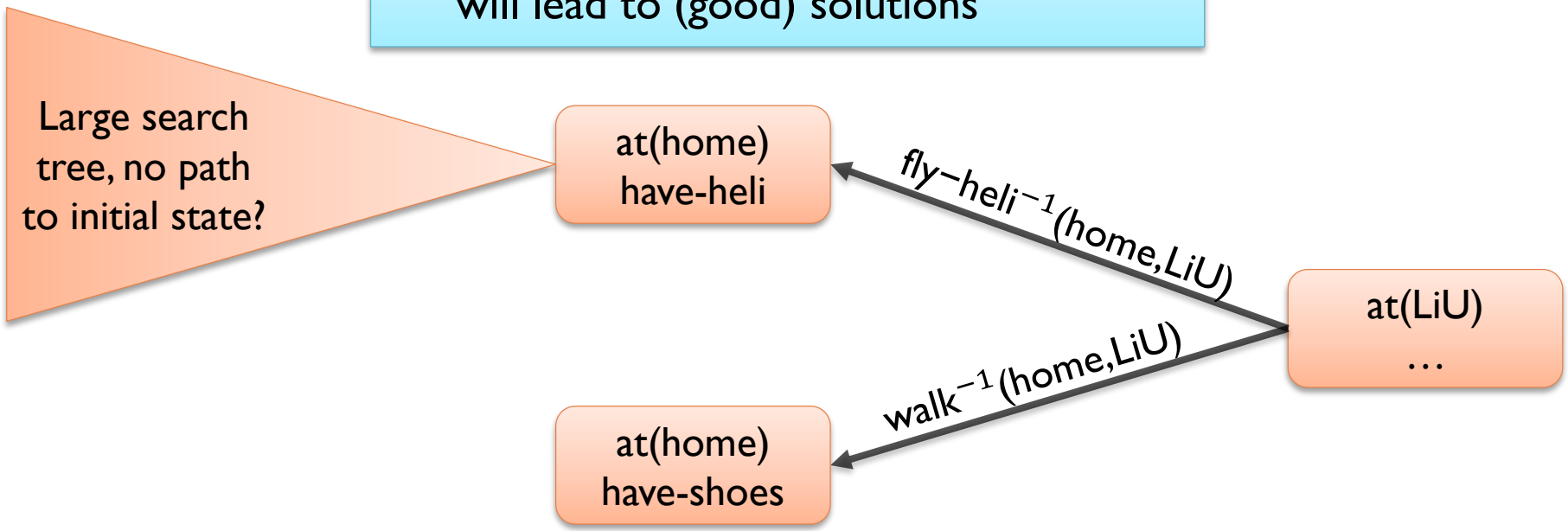
Original Inspiration: GraphPlan

Recap: Backward Search

jonkv@ida

BACKWARD SEARCH

- We know if the **effects** of an action can contribute to the goal
- Need **guidance** to determine which backward paths will lead to (good) solutions

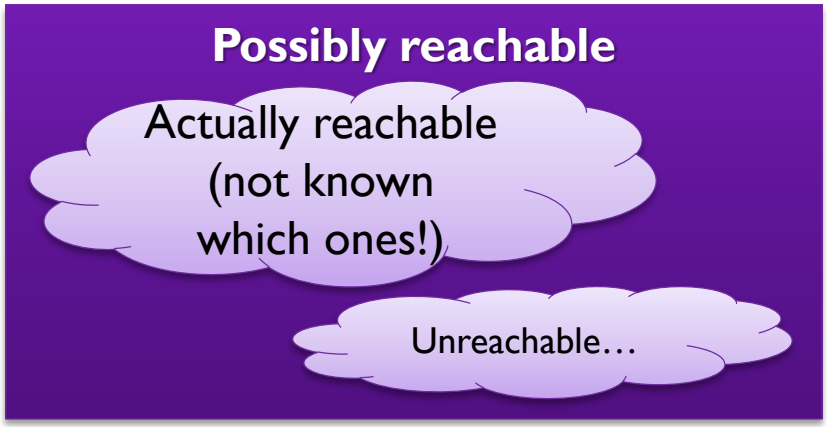


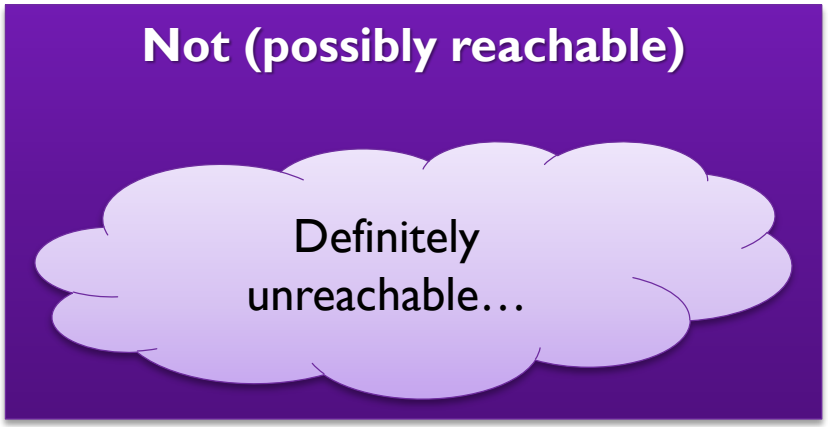
One approach: Use heuristics. But other methods exist…

Reachable States

jonkv@ida

- Suppose that we could quickly determine:
  - **possibly-reachable**(s_0, s) – may state s be reachable from s_0?

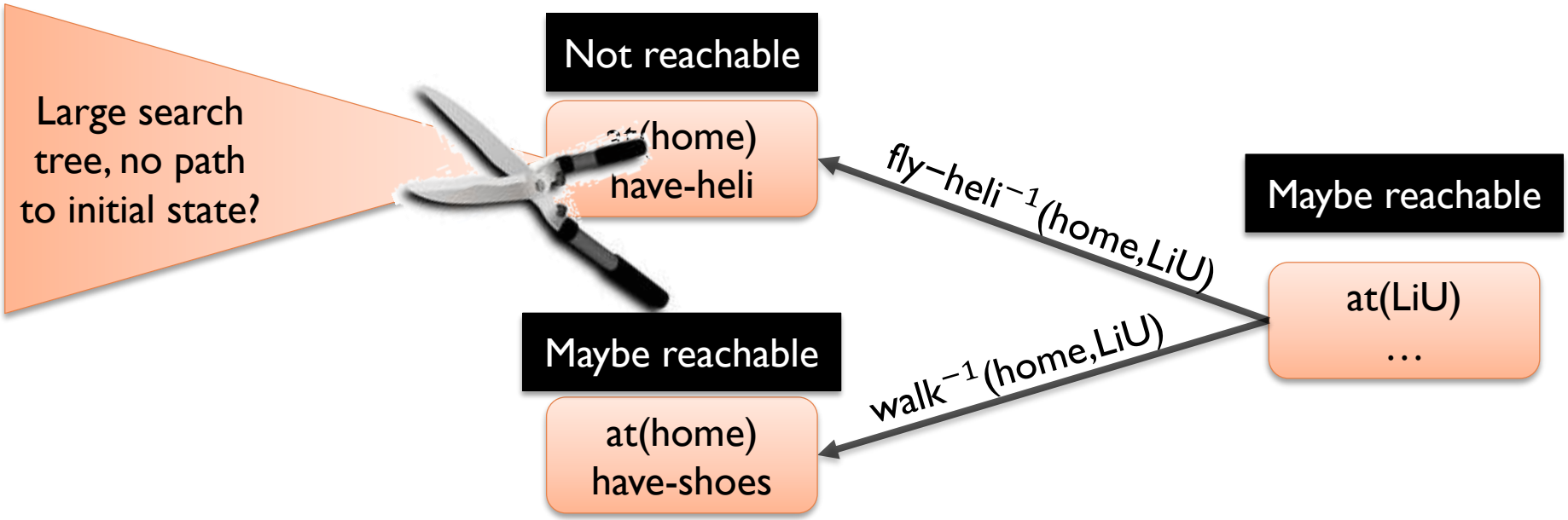




Reachable States

jonkv@ida

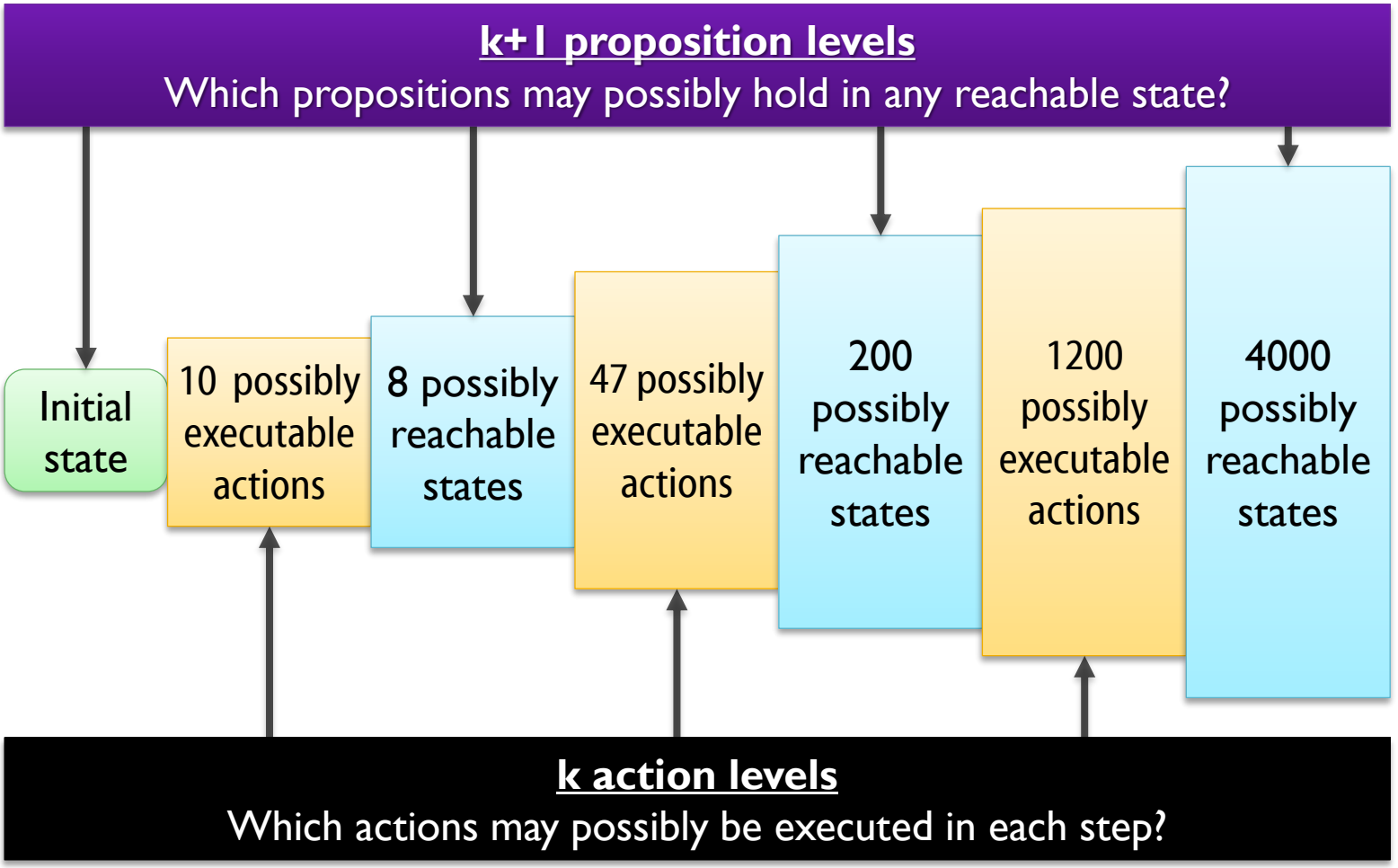
- Then we could **prune** many ”fruitless branches”:



Planning Graph

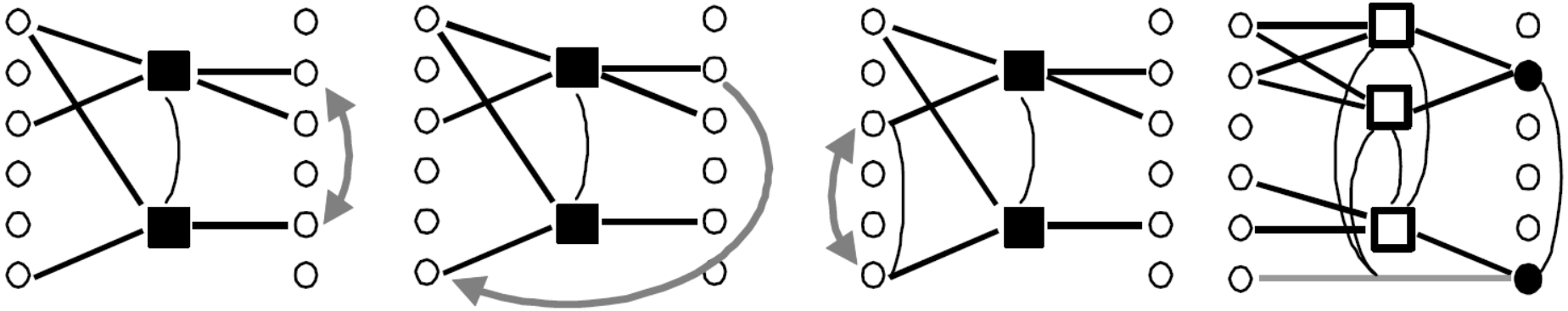
jonkv@ida

- A (non-relaxed) **Planning Graph**:
  - Useful to *generate states* – also useful in *backwards search!*



Negative Effects ➔ Mutual Exclusion

jonkv@ida



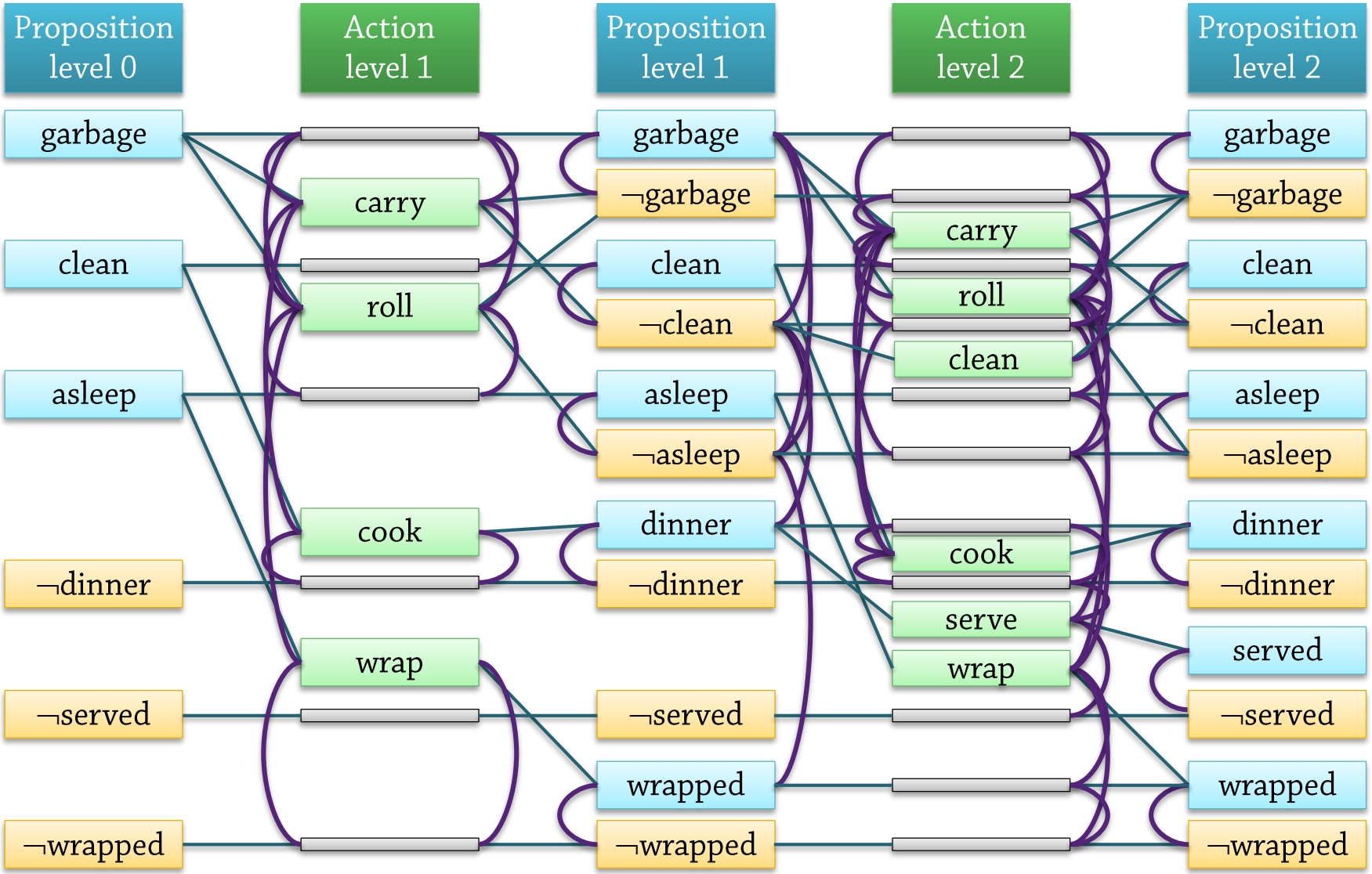
- Two **actions** at the same action level are mutex (can't be selected together) if
  - *Inconsistent effects:* an effect of one negates an effect of the other
  - *Interference:* one deletes a precondition of the other
  - ***Competing needs:*** they have mutually exclusive preconditions (*not shown*)
- Otherwise:
  - Both might appear at the same time step in a solution plan

Recursive propagation of mutexes

- Two **literals** at the same proposition level are mutex if
  - *Inconsistent support A:* one is the negation of the other,
  - *Inconsistent support B:* **all ways of achieving them are pairwise mutex**

Example

jonkv@ida



All goal literals are present in level 2, and none of them are (known to be) mutex!