Seminar: Search and Optimization

16. The FF Planning System: Fast Plan Generation Through Heuristic Search

Jörg Hoffmann and Bernhard Nebel

Silvan Sievers and Florian Pommerening

Universität Basel

December 6, 2012

Relaxation Heuristics

Delete Relaxations

Delete relaxation of STRIPS task

- As described in Lukas's talk
- Ignore all delete effects
- Easier task (NP-hard instead of PSPACE-hard)
- Every variable set to true stays true
- No operator is needed more than once

Solutions to delete relaxation

- Can be used as heuristic for original task
- Optimal solution is admissible estimate
- Any solution is a (possibly inadmissible) estimate

Relaxed Planning Graphs

Relaxed planning graph (RPG)

- Alternating action and fact layers
- First fact layer contains initial state
- Action layers contain applicable actions in last fact layer
- Other fact layers contain effects of actions
- No-op actions to maintain facts from previous layers

Heuristics based on the RPG

- $h^{\max}$: assumes positive interaction
- h^{add}: assumes no interaction, counts actions twice
- h^{FF}: next!

FF Heuristic

The FF Heuristic

Use cost of relaxed plan as estimate

- Not necessarily the optimal plan (h^+)
- Inadmissible
  - Proof: task without delete effects

Solution discovered by Graphplan

- Count operators at most once

Try to get short plans

- Prefer no-op operators
- Prefer easy operators
  - Difficulty of operator o: h^{add}-value of o

The FF Heuristic

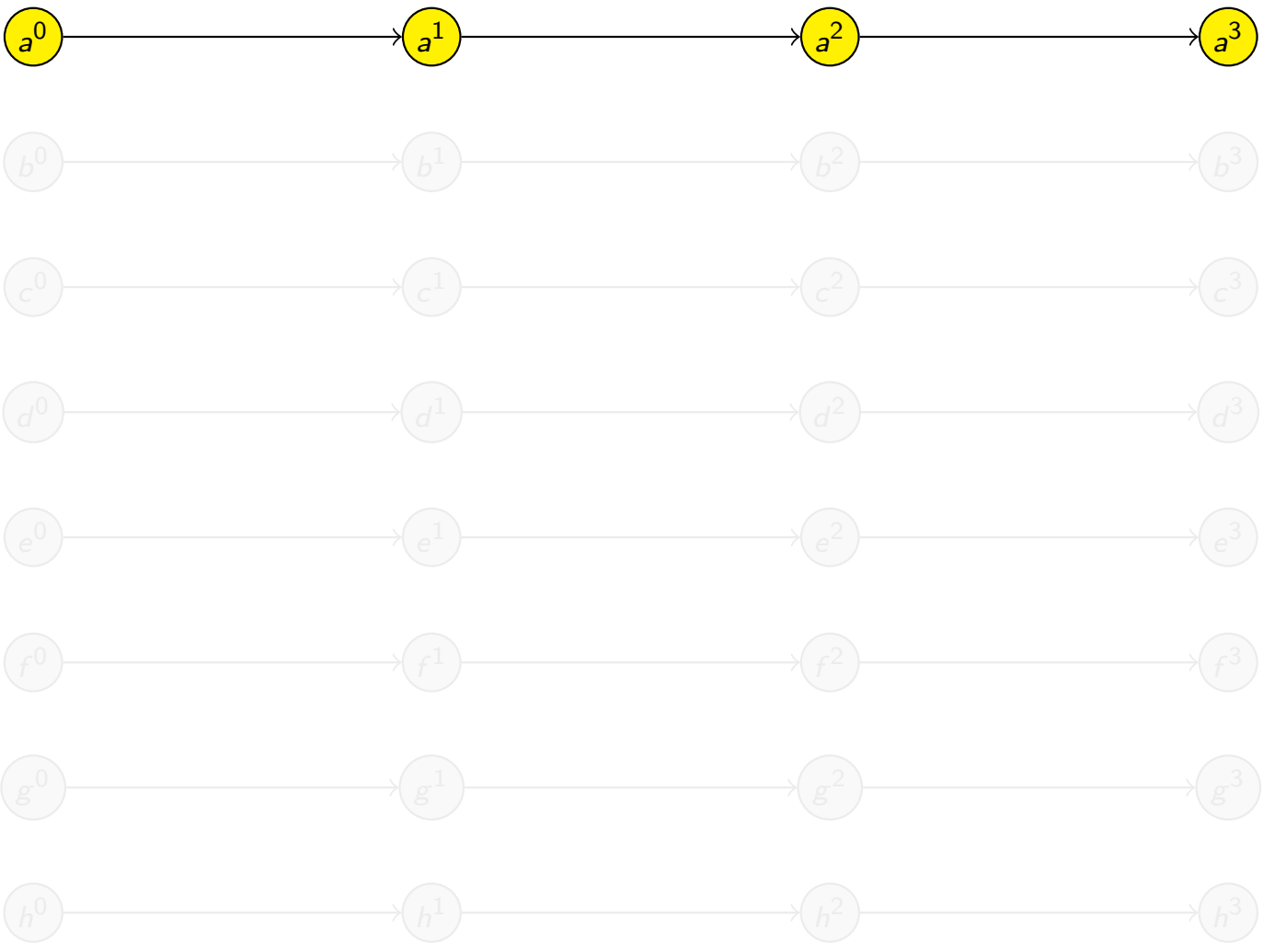
Construction

- Build relaxed planning graph
- Mark goal node
- For each layer continuously apply rules in order of priority
  1. Marked action or goal node
     - $\rightarrow$ Mark all predecessors
  2. Marked variable node without marked predecessors
     1. Has no-op predecessor $\rightarrow$ Mark it
     2. Otherwise $\rightarrow$ Mark the easiest predecessor

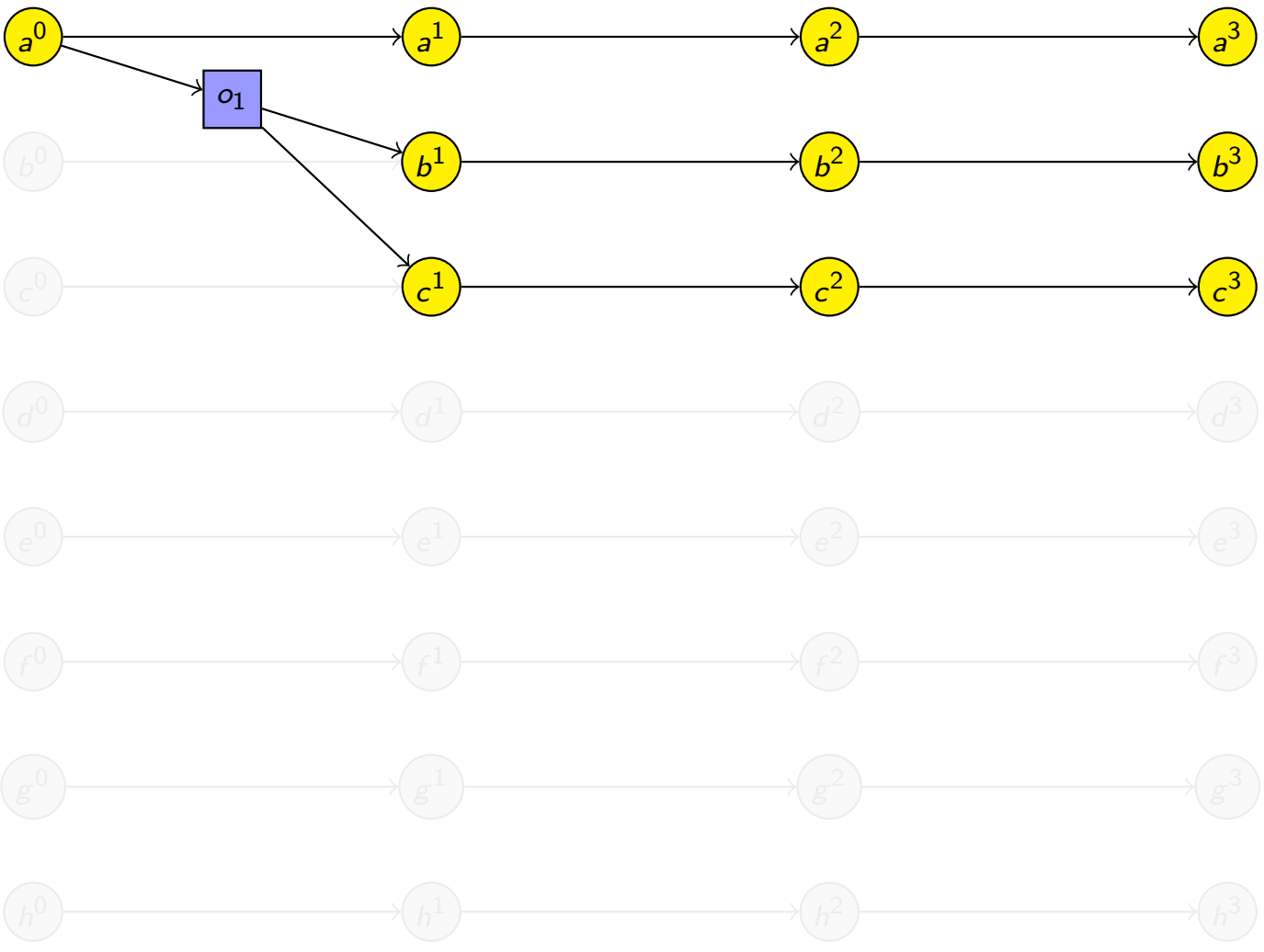
Heuristic value: number of marked actions

- Marked actions define relaxed plan

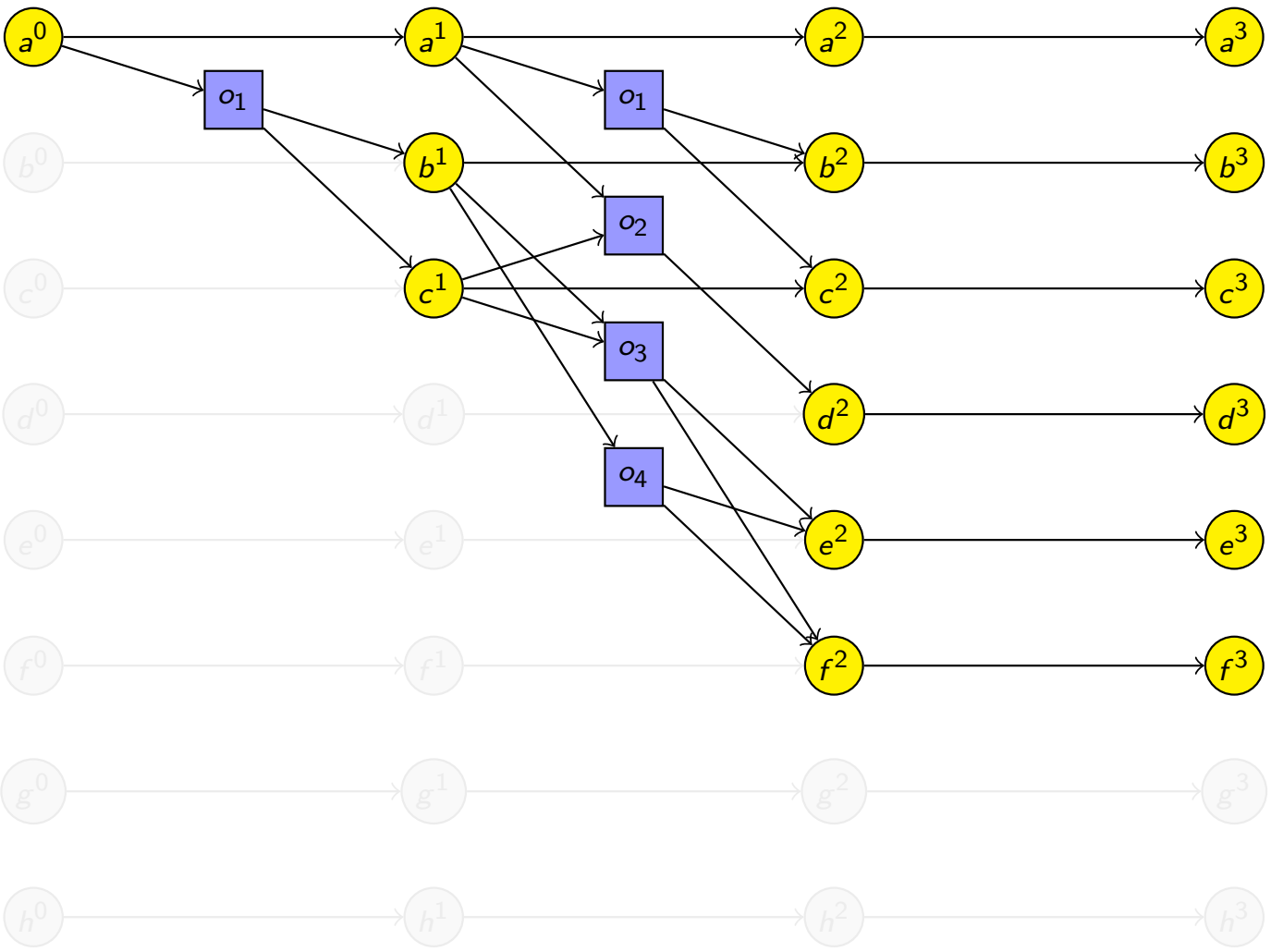
Example


a^0
a^1
a^2
a^3
b^0
b^1
b^2
b^3
c^0
c^1
c^2
c^3
d^0
d^1
d^2
d^3
e^0
e^1
e^2
e^3
f^0
f^1
f^2
f^3
g^0
g^1
g^2
g^3
h^0
h^1
h^2
h^3

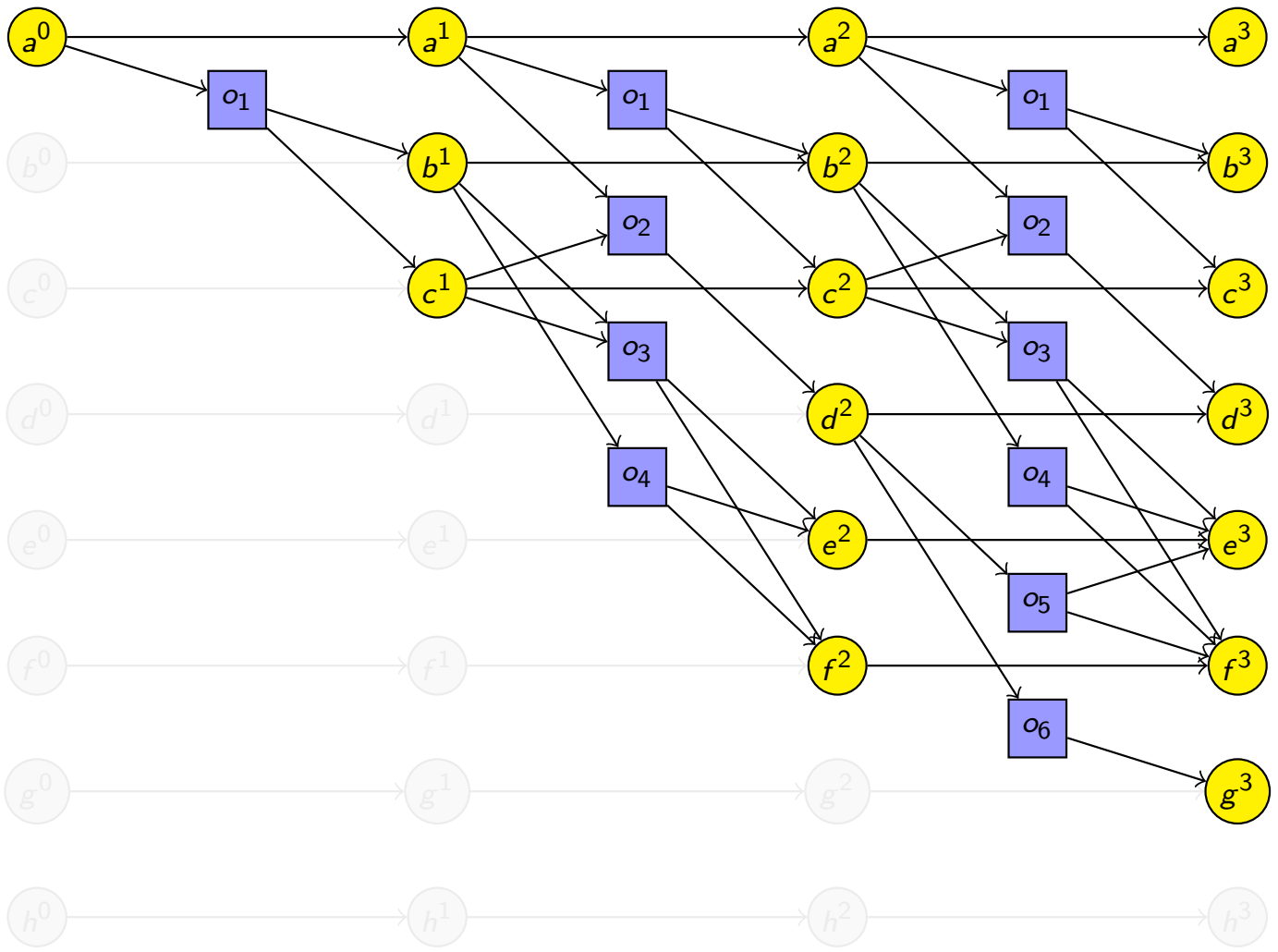
Example


a^0
a^1
a^2
a^3
o_1
b^0
b^1
b^2
b^3
c^0
c^1
c^2
c^3
d^0
d^1
d^2
d^3
e^0
e^1
e^2
e^3
f^0
f^1
f^2
f^3
g^0
g^1
g^2
g^3
h^0
h^1
h^2
h^3

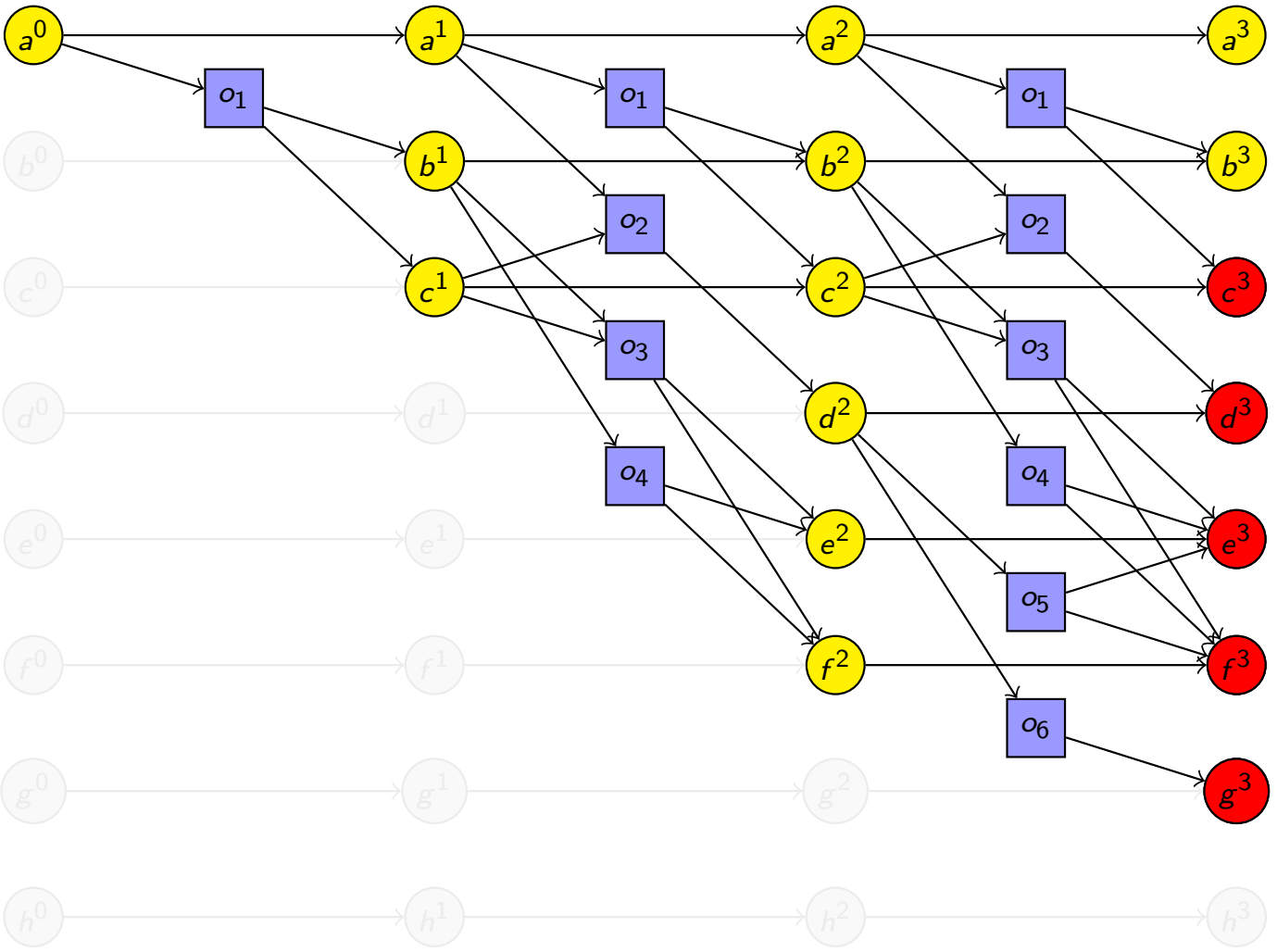
Example


a^0
o_1
b^0
c^0
d^0
e^0
f^0
g^0
h^0
a^1
o_1
b^1
o_2
c^1
o_3
d^1
o_4
e^1
f^1
g^1
h^1
a^2
b^2
c^2
d^2
e^2
f^2
g^2
h^2
a^3
b^3
c^3
d^3
e^3
f^3
g^3
h^3

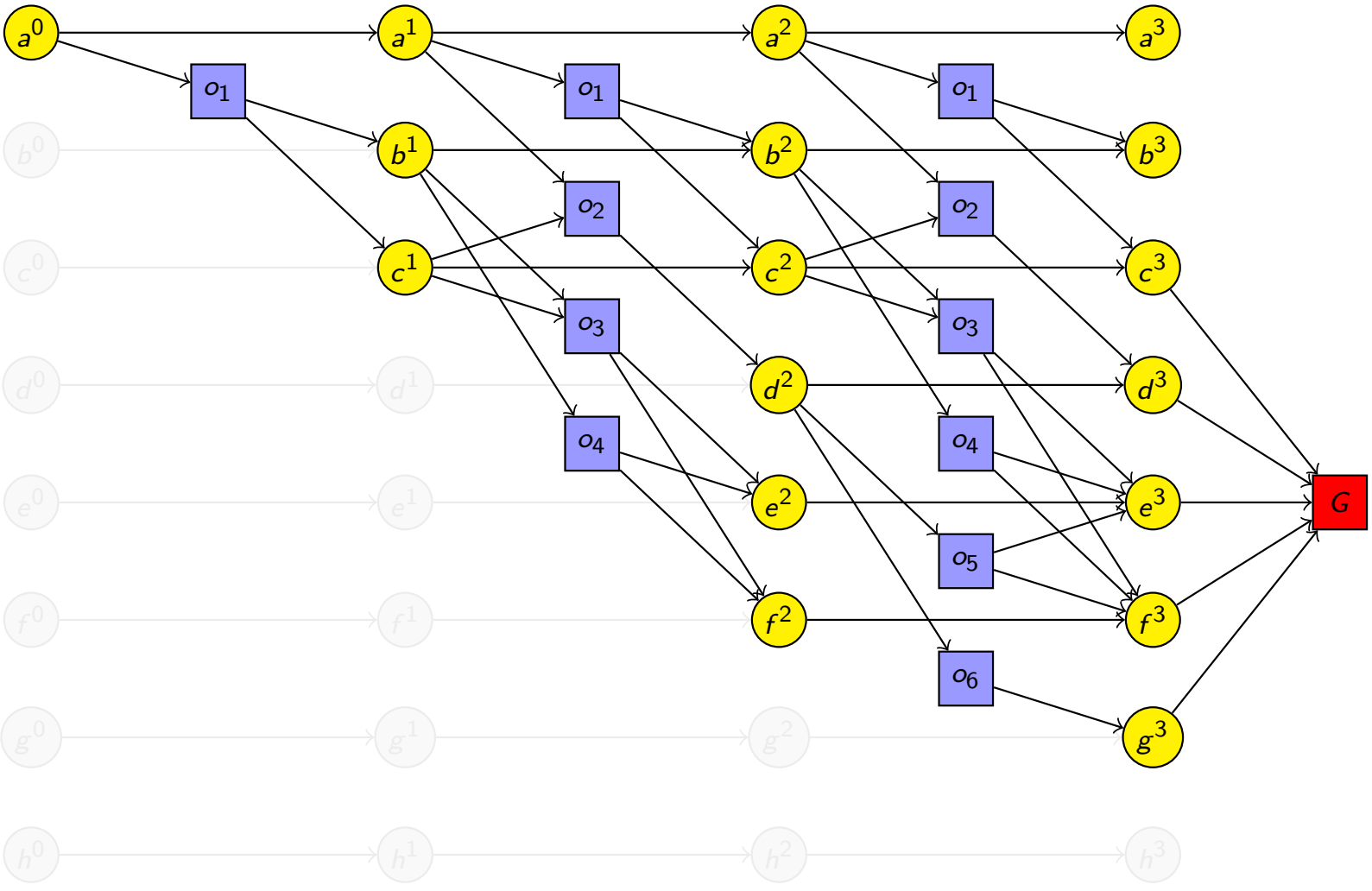
Example


a^0
o_1
a^1
o_1
a^2
o_1
a^3
b^0
b^1
b^2
b^3
o_2
o_2
c^0
c^1
c^2
c^3
o_3
o_3
d^0
d^1
d^2
d^3
o_4
o_4
e^0
e^1
e^2
e^3
o_5
f^0
f^1
f^2
f^3
o_6
g^0
g^1
g^2
g^3
h^0
h^1
h^2
h^3

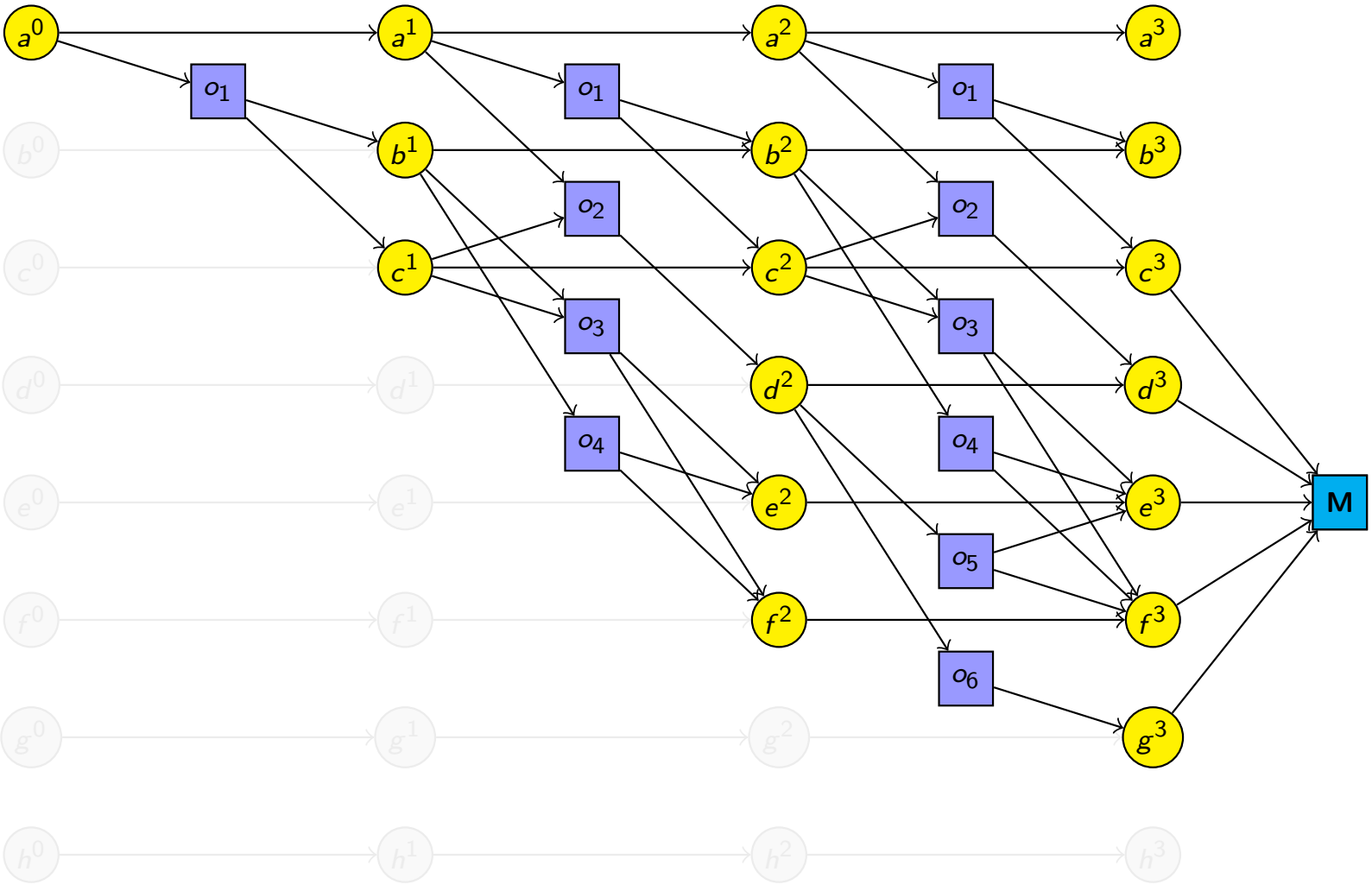
Example


a^0
o_1
b^0
c^0
d^0
e^0
f^0
g^0
h^0
a^1
o_1
b^1
o_2
c^1
o_3
d^1
o_4
e^1
f^1
g^1
h^1
a^2
o_1
b^2
o_2
c^2
o_3
d^2
o_4
e^2
o_5
f^2
o_6
g^2
h^2
a^3
o_1
b^3
o_2
c^3
o_3
d^3
o_4
e^3
o_5
f^3
g^3
h^3

Example

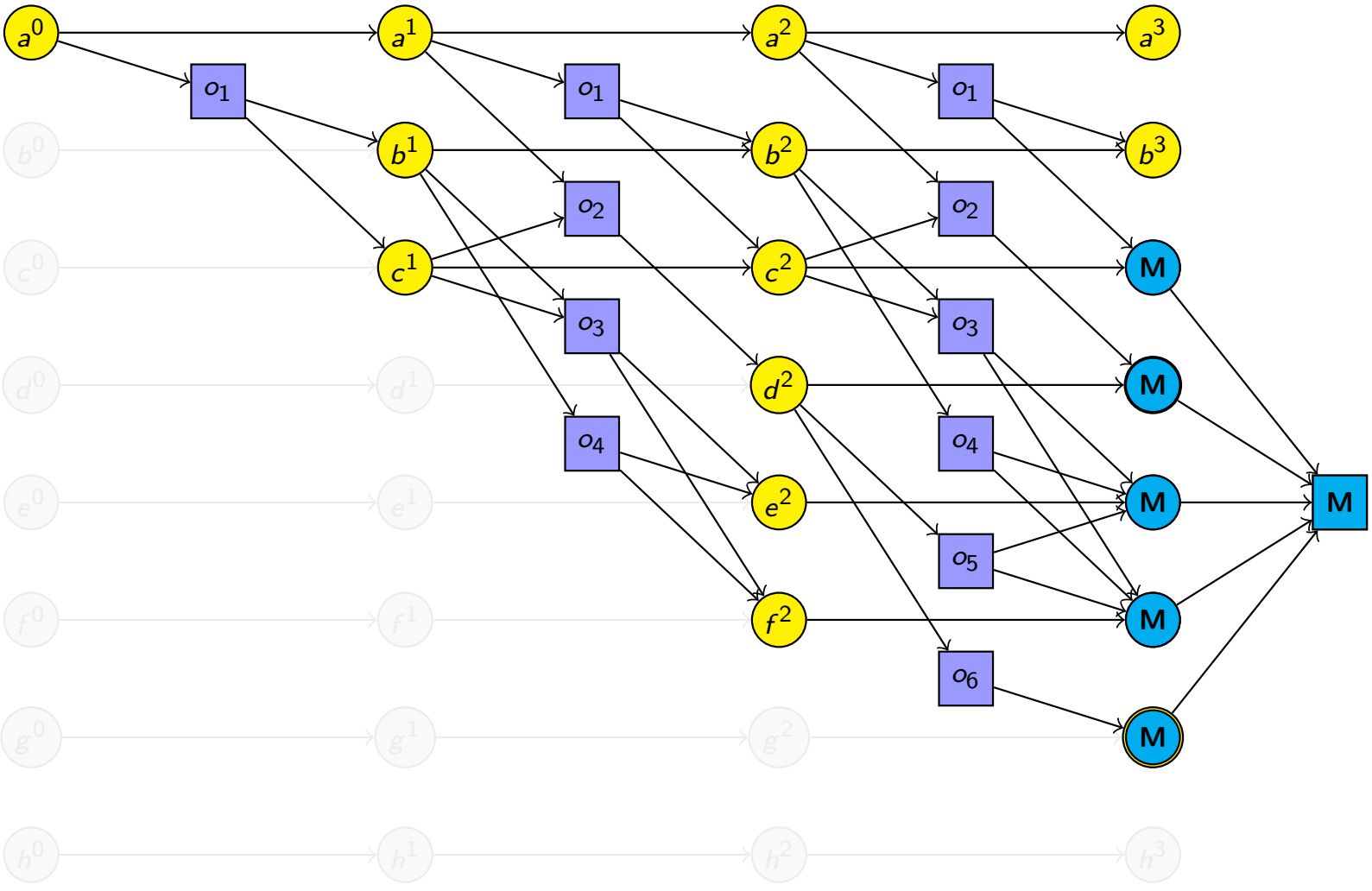

a^0
o_1
a^1
o_1
a^2
o_1
a^3
b^0
b^1
b^2
b^3
o_2
o_2
c^0
c^1
c^2
c^3
o_3
o_3
d^0
d^1
d^2
d^3
o_4
o_4
e^0
e^1
e^2
e^3
G
o_5
f^0
f^1
f^2
f^3
o_6
g^0
g^1
g^2
g^3
h^0
h^1
h^2
h^3

Example



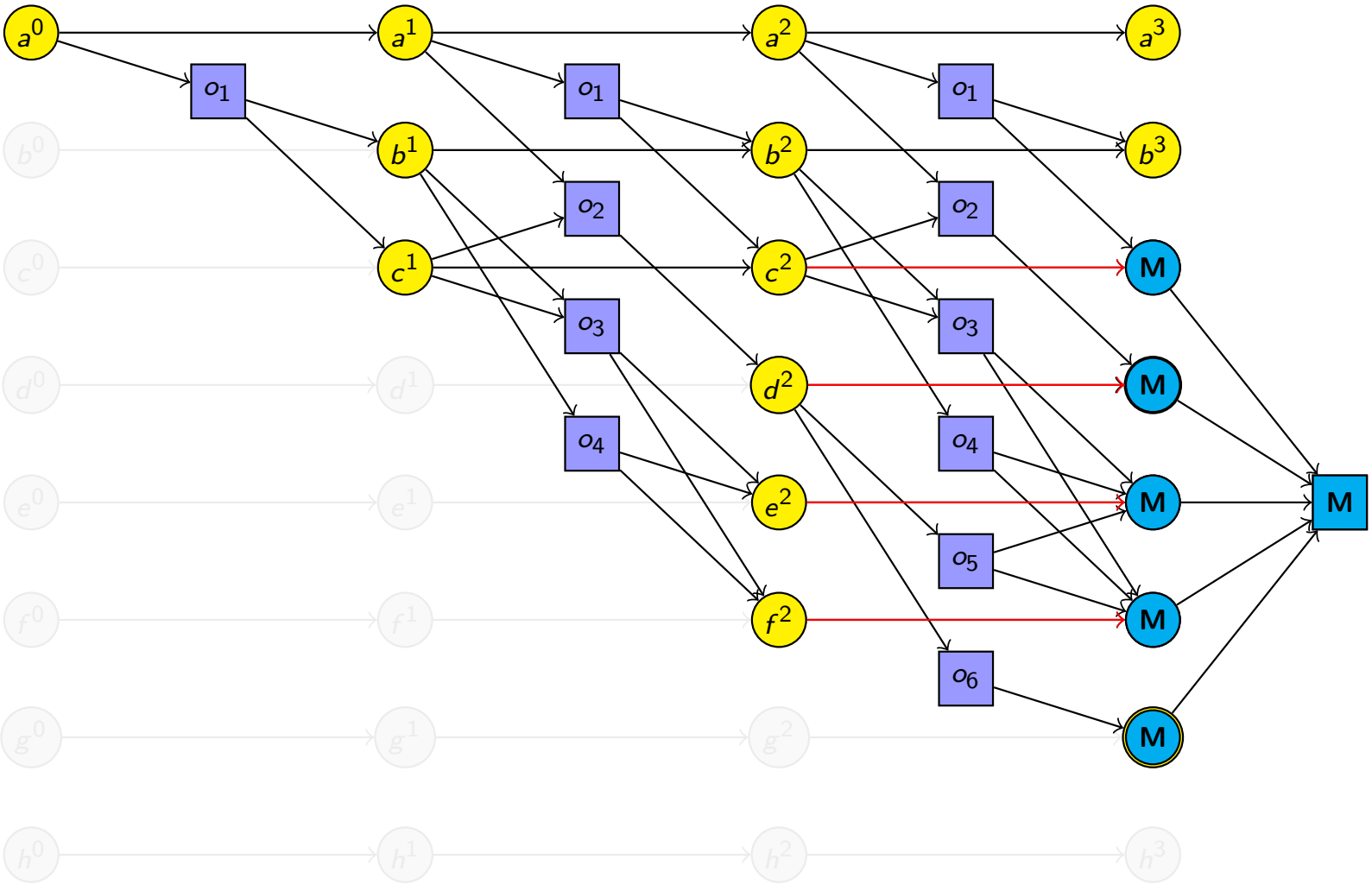
Mark goal node

Example



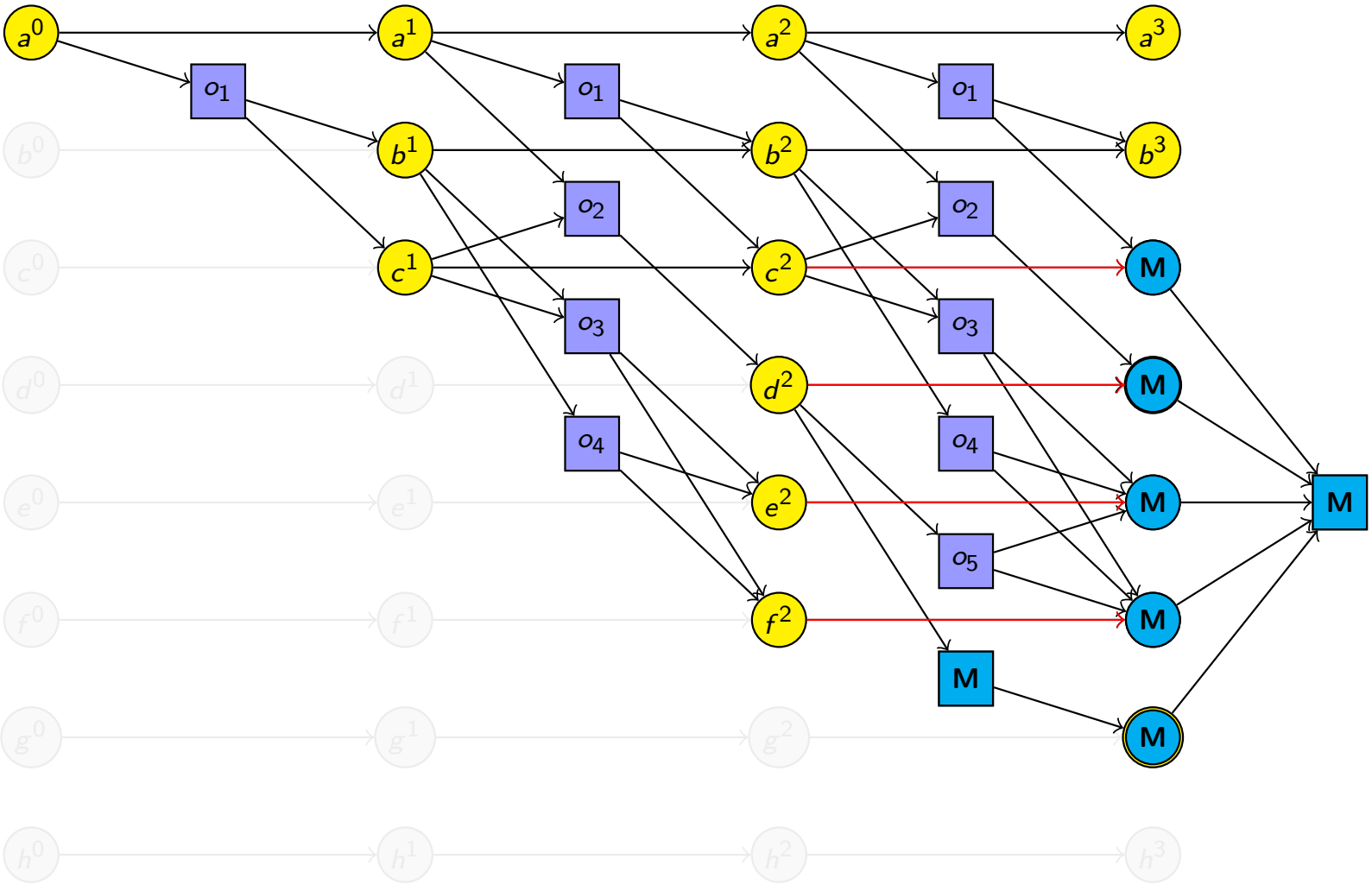
Rule 1: Mark all predecessors of goal node

Example



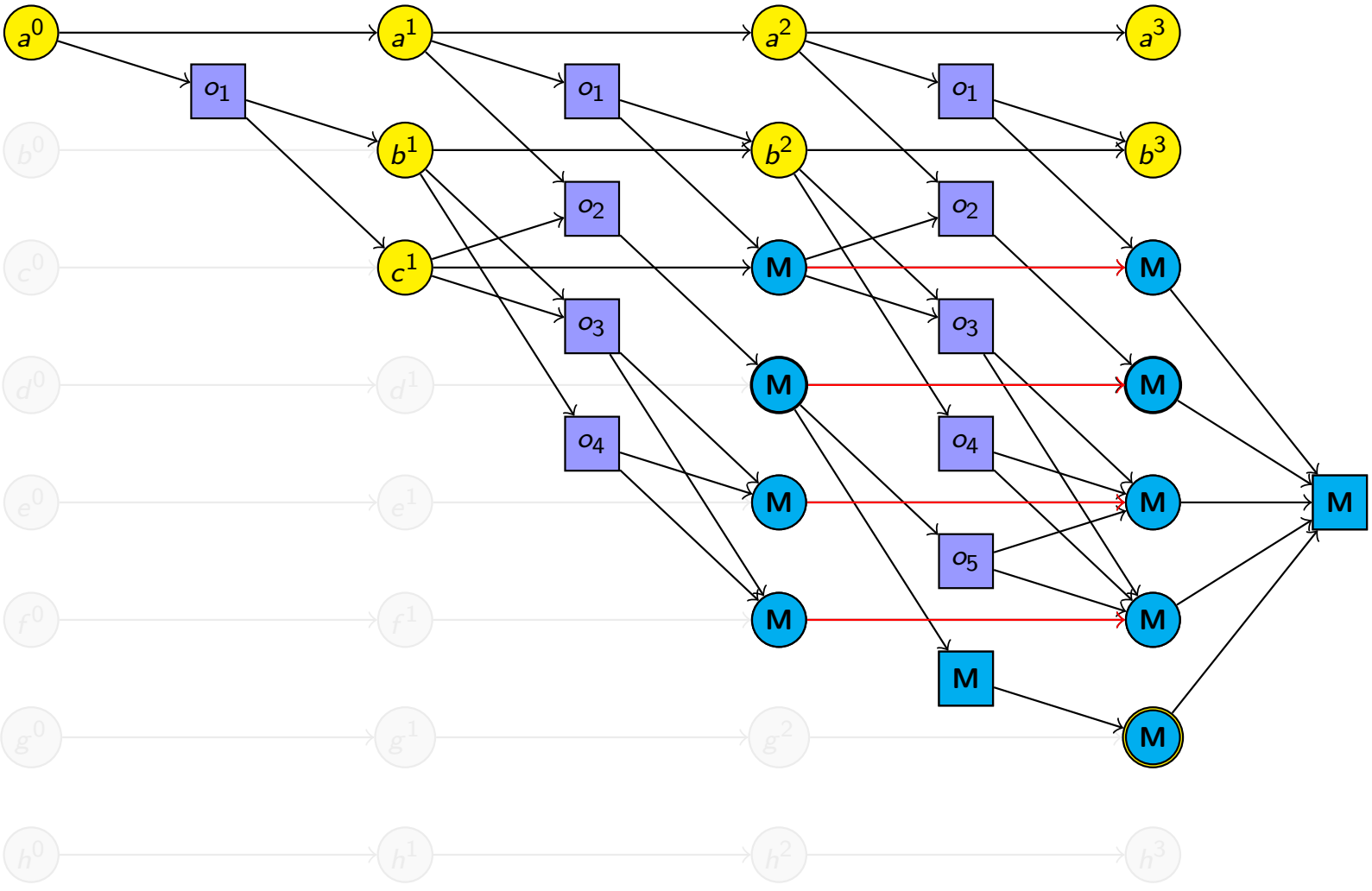
Rule 2.1: Mark no-op predecessors of variable nodes

Example



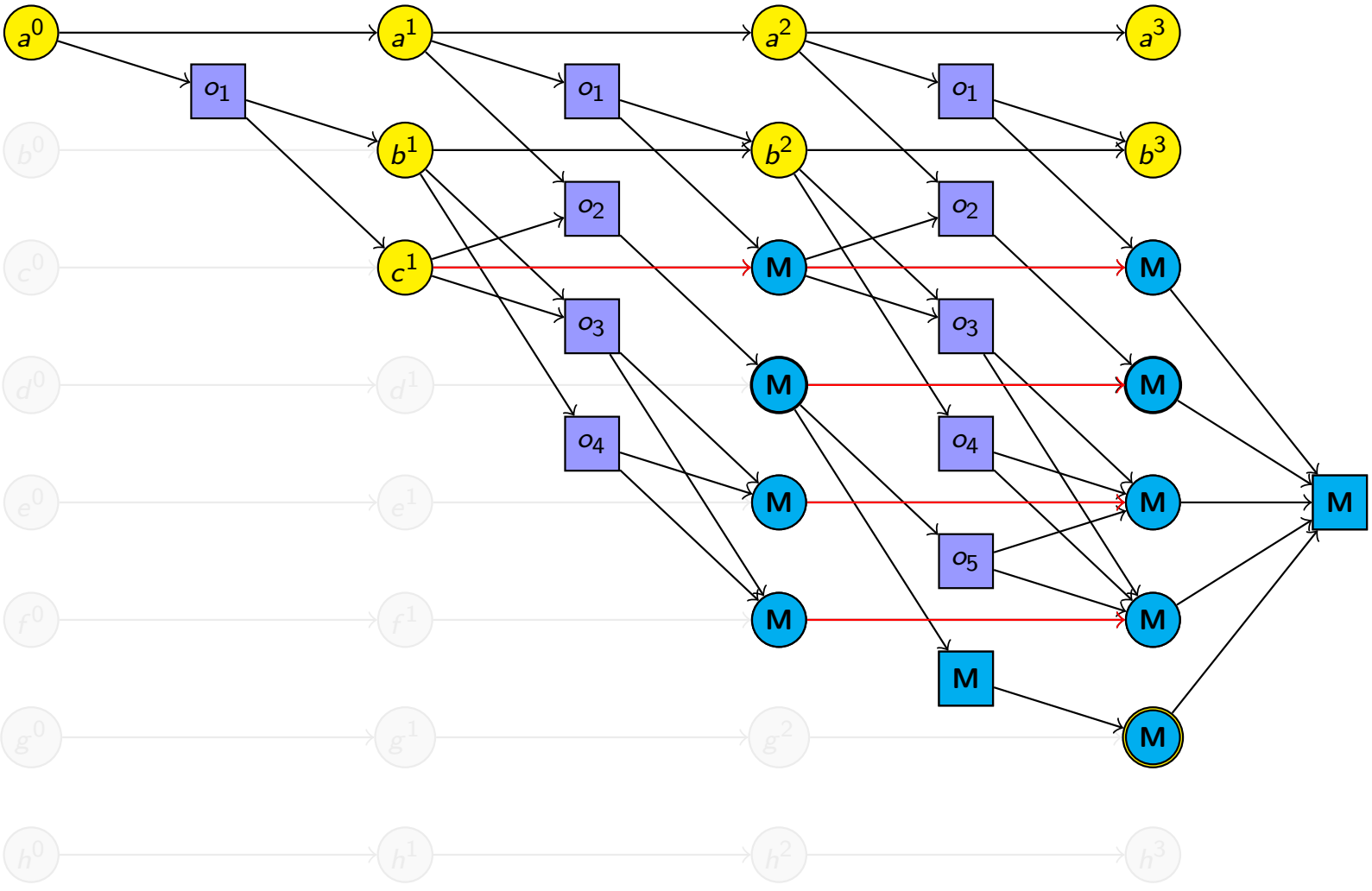
Rule 2.2: Mark easiest predecessor of variable node

Example



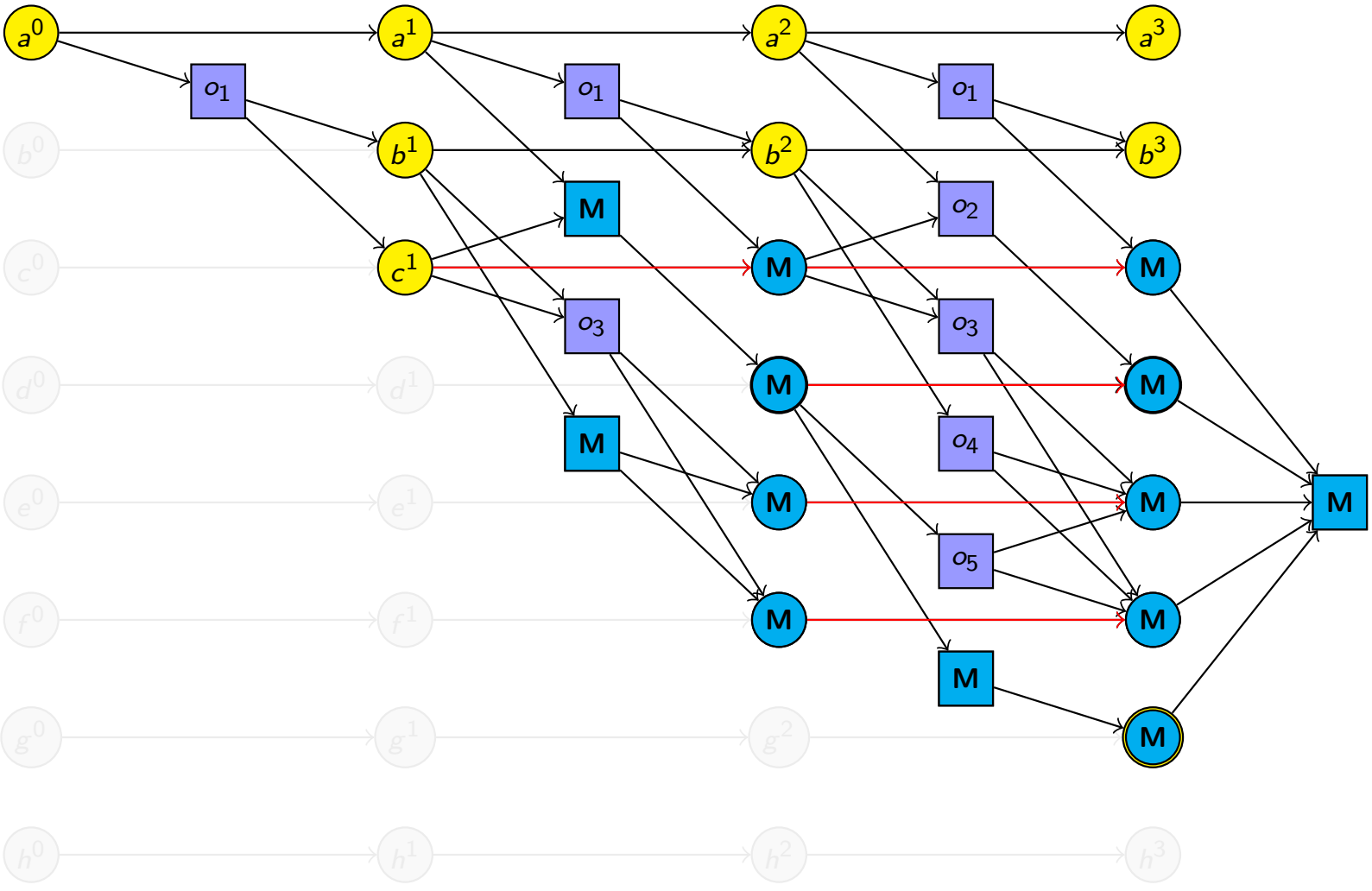
Rule 1: Mark all predecessors of action nodes

Example



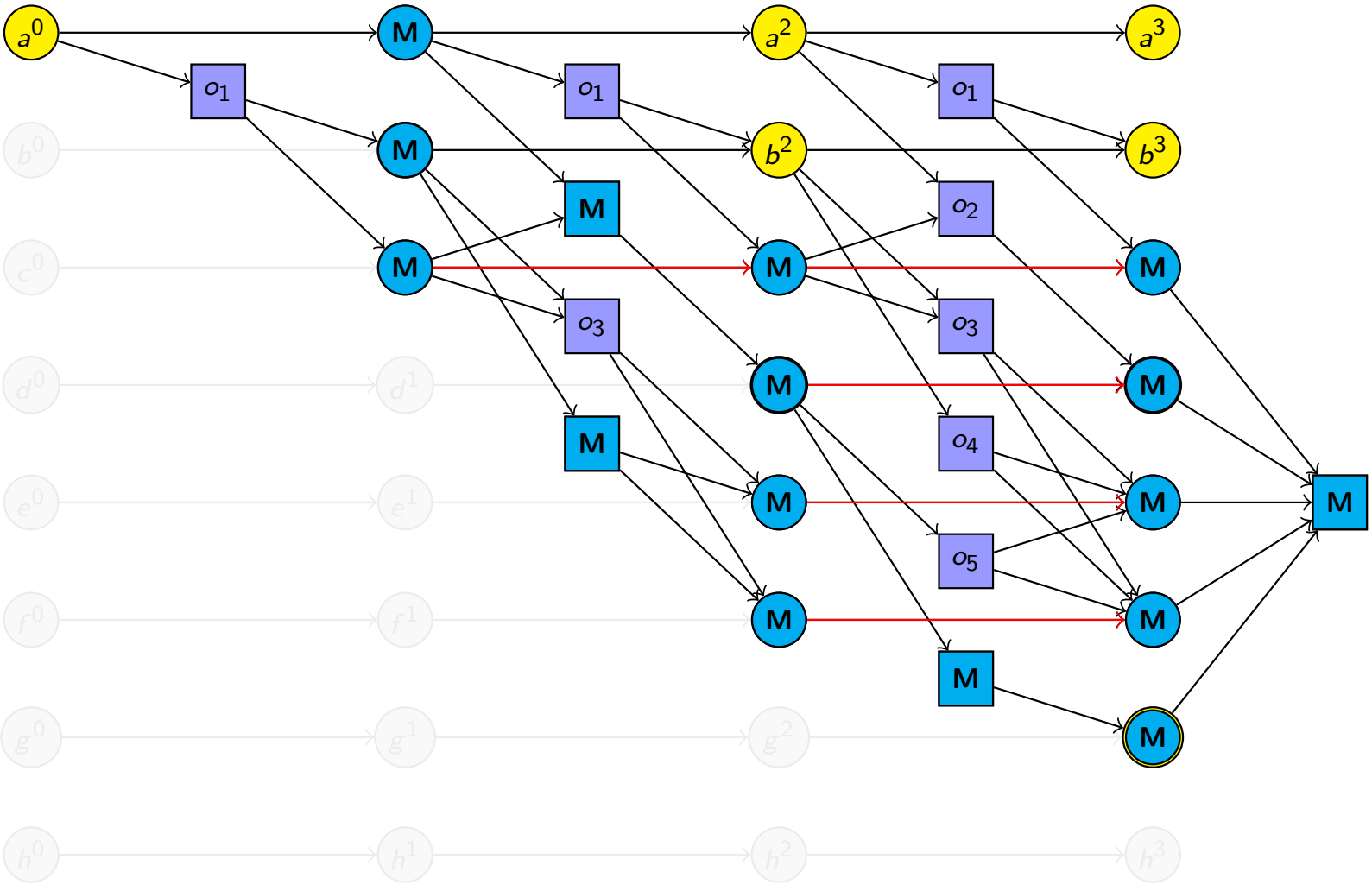
Rule 2.1: Mark no-op predecessor of variable node

Example



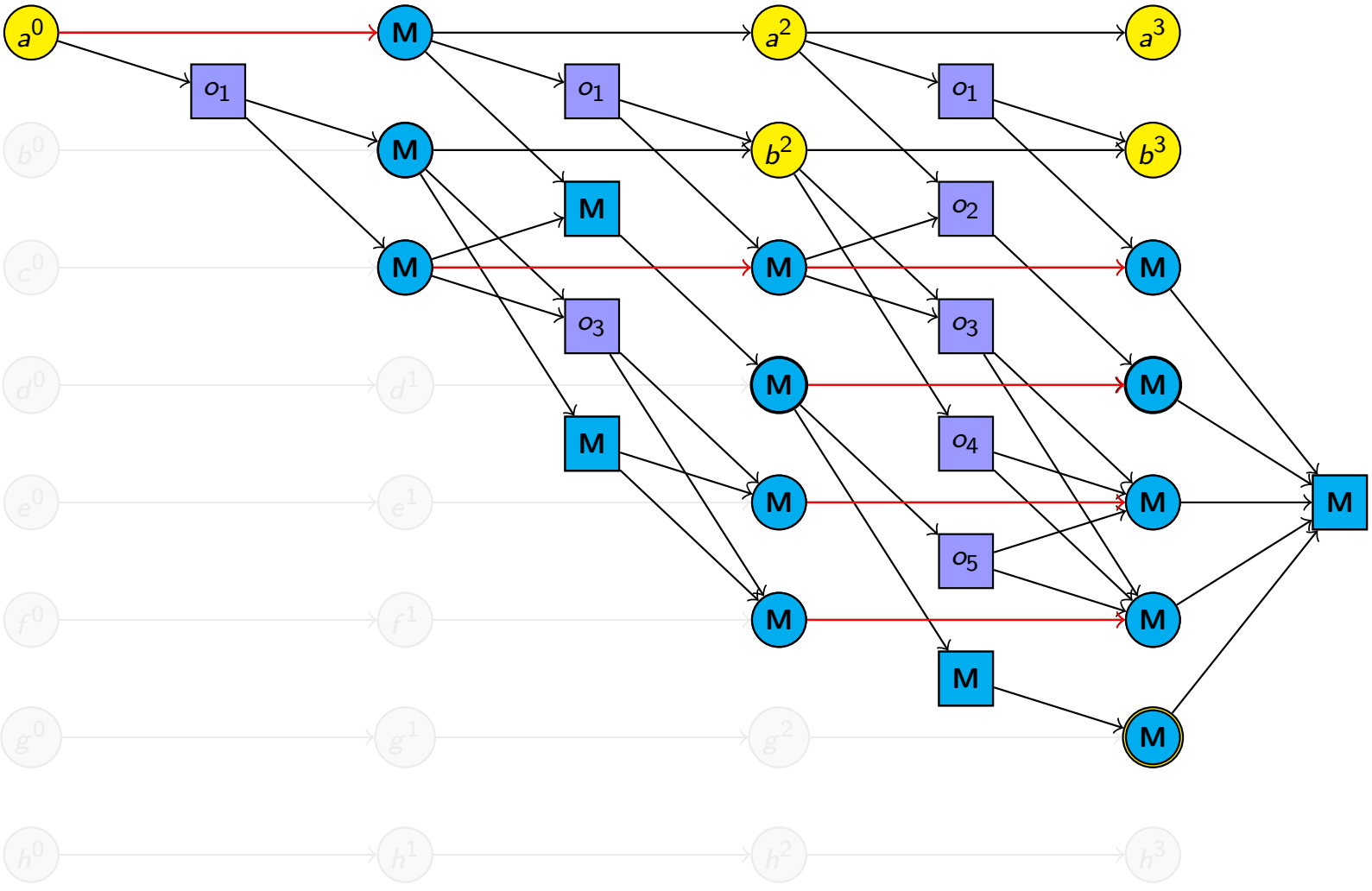
Rule 2.2: Mark easiest predecessor of variable nodes

Example



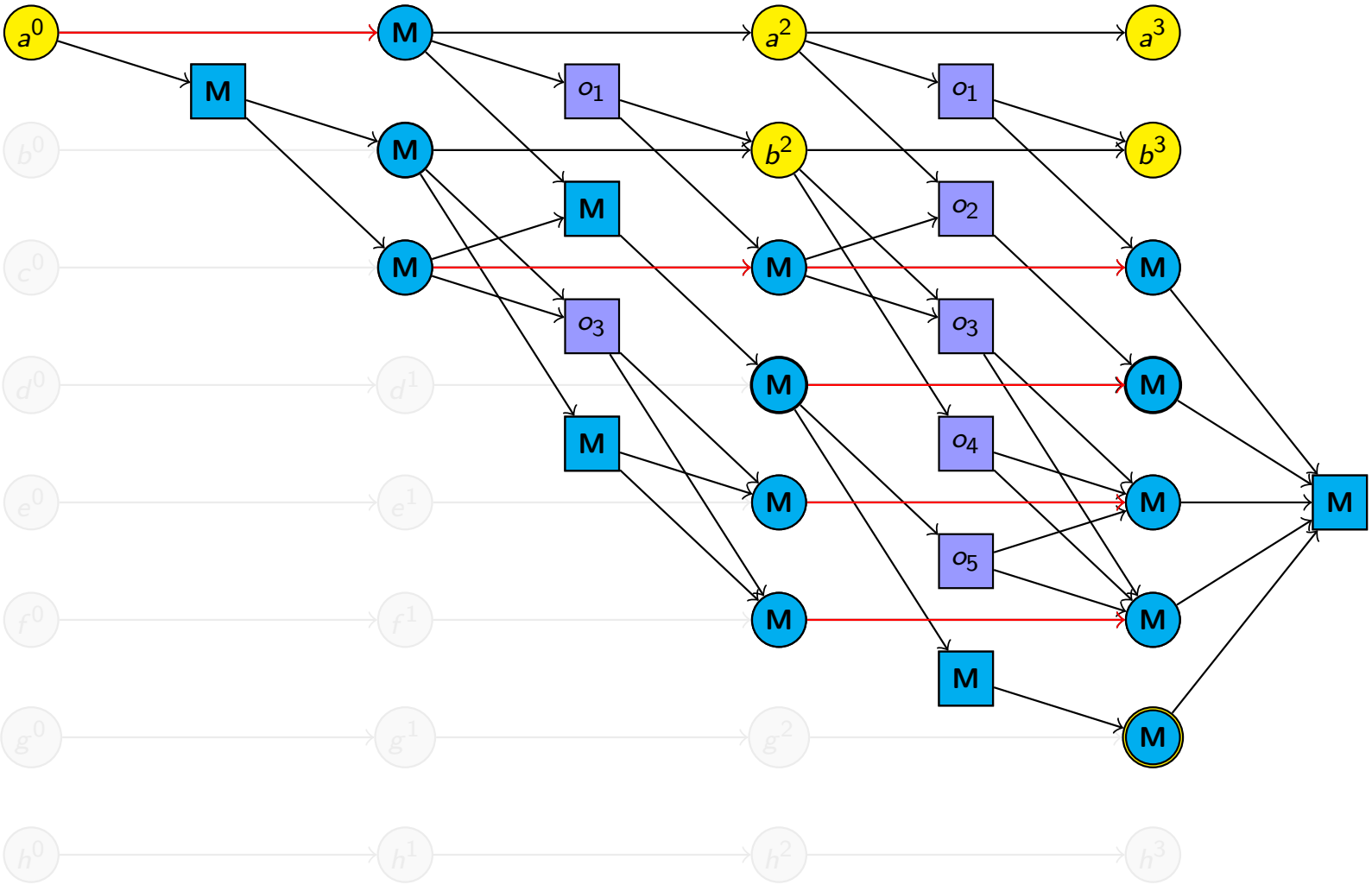
Rule 1: Mark all predecessors of action nodes

Example



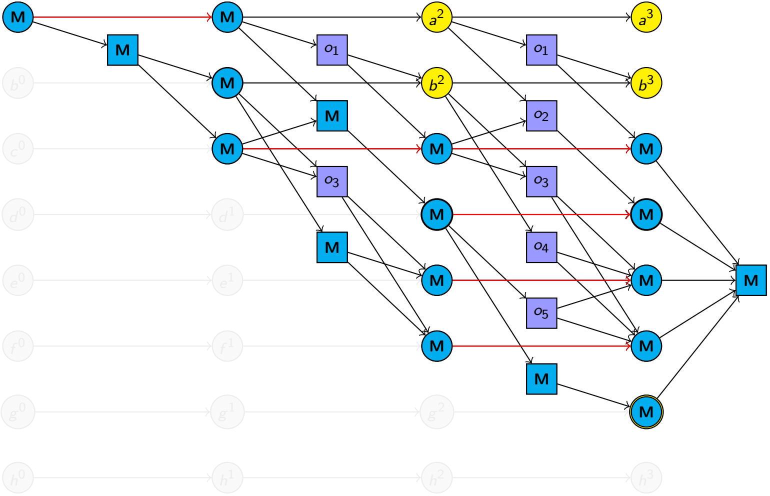
Rule 2.1: Mark no-op predecessor of variable node

Example



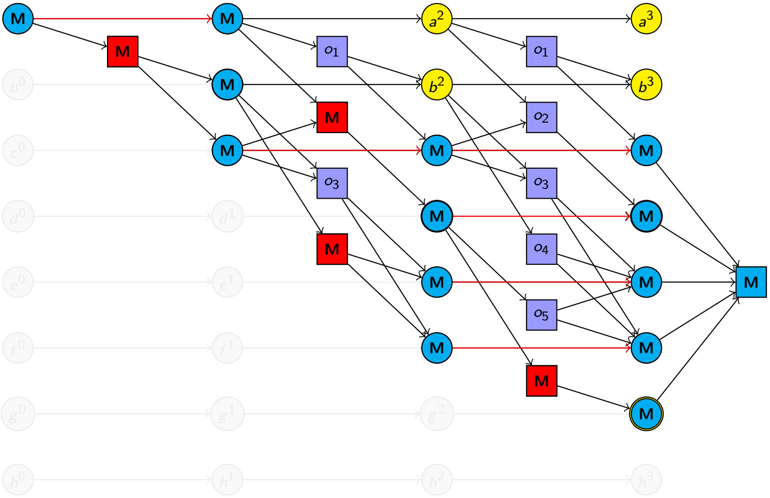
Rule 2.2: Mark easiest predecessor of variable node

Example



Rule 1: Mark all predecessors of action node

Example



$h^{\mathsf{FF}} = 4$

Implementation Tricks

Efficient implementation

- Connectivity graph: one action and one fact layer
- Pointers from actions to preconditions and back
- Pointers from effects to actions and back
- For each action and fact: store number of first layer of RPG containing it
- Store number of unsatisfied preconditions for actions
  - Queue action for application when all preconditions are satisfied

Evaluation

Comparison of h^{FF} with h^{add}

- Usually tighter bound on h^+
- No (overall) improvement for solution length of plans
- Significant improvement of run-time in 12/20 domains
- Degradation in 2/20 domains
- Best heuristic at the time
- Still one of the best heuristics for satisficing planning

Search Technique

Search Technique

Motivation

- Local search (as in HSP) because state evaluations are costly
- Use systematic search to avoid local optima

Enforced Hill-climbing (EHC)

- Hill-climbing: always choose a best successor
  - Even if it is not better
- Use breadth-first search to find a strictly better descendant
- Example: whiteboard

Enforced Hill-climbing ctd.

Properties

- Commits to chosen states: can get stuck in dead ends
- Incomplete search algorithm in the general case
- Complete if dead-end free task and goal-aware heuristic

How to deal with the completeness issue?

- Determining presence of dead ends: PSPACE-complete
- EHC usually fails quickly on tasks with dead ends
- Switch to systematic search once enforced hill-climbing failed
- Here: greedy best-first search

Evaluation

Example: Logistics

- States with better evaluation usually at very small depths

Comparison of EHC with HC

- Increase or decrease of run-time depending on domain
  - Overall slightly favoring EHC
- Solution length
  - Significantly improved in 8/20 domains
  - Degraded in 1/20 domains

Pruning Techniques

Pruning Techniques

General properties of the presented pruning techniques

- Generated as side effects of RPG computation
- Do not preserve completeness
- Integrated only into EHC (and not best first search) in order not to break completeness

In the following

- Helpful actions
- Added goal deletion
- Goal agenda

Helpful Actions

General idea

- Choose subset of applicable actions at each state
  - Called helpful actions
- Restrict successor generation to helpful actions

Naive approach

- Relaxed plan starts with action marked in first layer of RPG
  - Try starting the actual plan in same way
- Choose all these actions as helpful actions
- Example: whiteboard

Helpful Actions ctd.

Safer version

- Choose all actions adding marked variables in the first layer
- Example: whiteboard

Consequence

- Still incomplete (example in the paper)
- Prunes a potentially large part of the search space

Evaluation

Example: Logistics

- Helpful actions prune 60–95% of state successors
- Larger tasks have fewer helpful actions

Comparison of pruning vs. no pruning

- Run-time
  - Significant improvement in 13/20 domains
  - Degradation in 2/20 domains
- Solution length
  - Significantly improved in 7/20 domains


Helpful actions
are helpful

Added Goal Deletion

Motivation

- Domains with goal ordering constraints
- Avoid achieving goals and deleting them later

Method

- If relaxed plan for state s contains action which removes previously achieved goal, remove s from search space
- Example: Blocksworld
- Problem: Task where goal deletion is necessary

Goal Agenda

Motivation

- Again: domains with goal ordering constraints
- Achieve goals in the given order

Method

- Given: subsets of goals $G_1, \ldots, G_n$
- Start EHC in initial state for G_1
- If succeeded, stay in resulting state
- Continue EHC for $G_1 \cup G_2$
- Etc.

Combination of Techniques

Combination of different features of FF

Plan length

- EHC with h^{FF} often results in short plans

Run-time

- No observable synergies when including more features to h^{FF}
- Particularly good when combining EHC with helpful actions
  - Hill-climbing explores only one branch
  - EHC uses breadth-first search looking for better nodes
  - Pruning applicable actions decreases branching factor
  - More useful for EHC
- Best performance when using EHC with h^{FF} and helpful actions

Conclusion

Conclusion

Features of the Fast Forward planning system

- FF heuristic
- Enforced hill-climbing
- Three other techniques
  - Most important: Helpful actions

Performance

- Outperforms HSP, the former state of the art
- FF heuristic still very good today