New Admissible Heuristics for Domain-Independent Planning

Patrik Haslum, Blai Bonet, Héctor Geffner
AAAI 2005

Presentation by Kevin Urban

Content

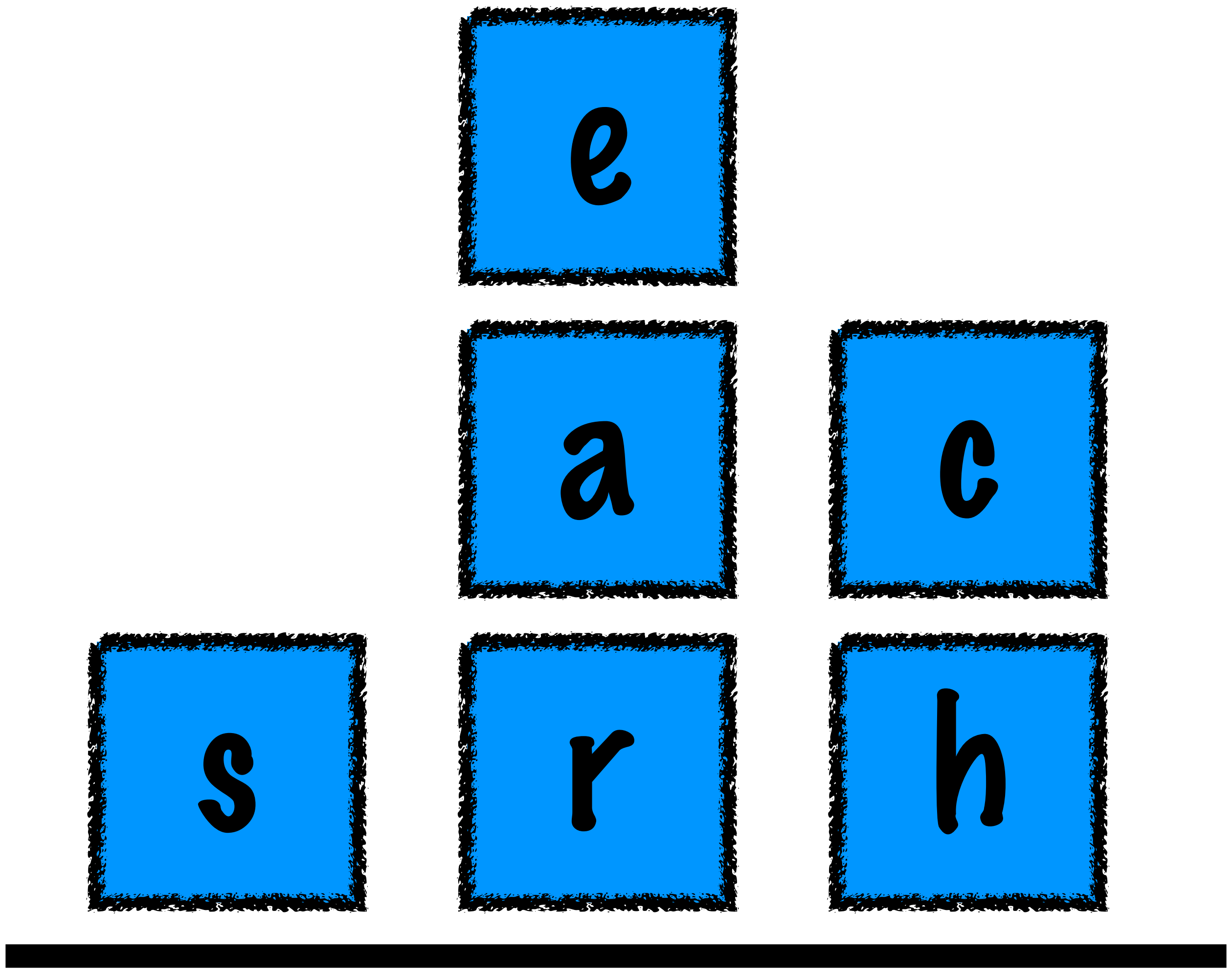
- STRIPS planning model
- “normal” PDBs, multi-valued variables & their limits
- constrained pattern databases
- h^m heuristics
- regression planning
- how to add additivity
- action partitioning
- evaluation

STRIPS planning model

STRIPS planning model

- **Atoms:** binary values
- **States *s*:** sets of atoms that are true
- **Actions *a*:**
  - ***pre(a):*** set of atoms that must be true before application
  - ***add(a):*** set of atoms that become true
  - ***del(a):*** set of atoms that become false
- **Initial state *I***
- **Set of goal states *G***

STRIPS — Blocksworld



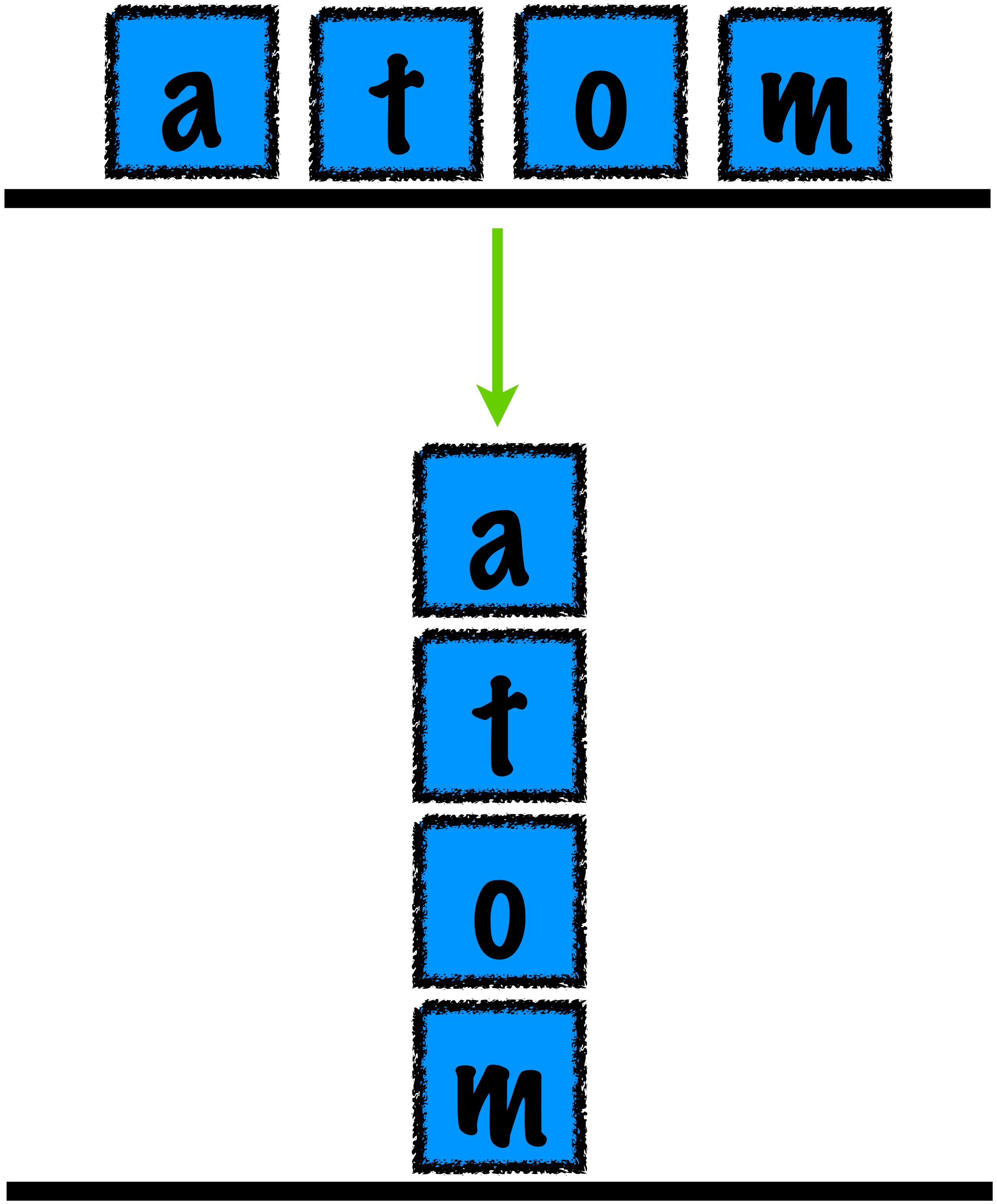
- **Atoms:**
  - on(e,a), on(a,r), on(c,h)
  - on-table(s), on-table(r), on-table(h)
  - clear(s), clear(e), clear(c)
- **Actions:**
  - move(e,a,c), ...
    - pre: { on(e,a), clear(e), clear(c) }
    - add: { on(e,c), clear(a) }
    - del: { on(e,a), clear(c) }
  - to-table(e,a), ...
  - from-table(s,e), ...

Pattern Databases

Pattern Databases — Multi-valued variables

- Used for pattern databases: Multi-valued variables
- Set of atoms with at-most-one semantics (*mutex*)
- Implicit in many STRIPS problems
- Can be extracted automatically

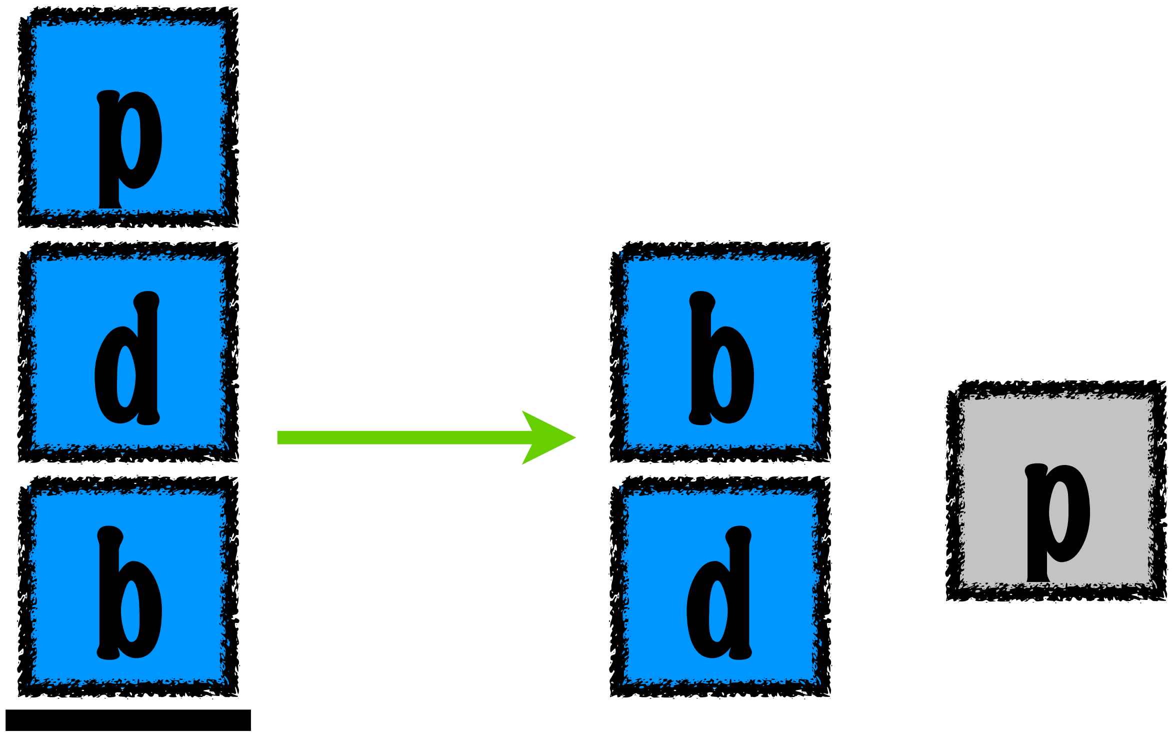
$h^A(s)$: Multi-valued variables — Example



- e.g. variable ***pos(t)***:
  - mutex for {on(t,a), on(t,o), on(t,m), on-table(t)}
  - domain: {a, o, m, table}
- pattern databases with pos(x)
- yields $h^{pos(x)}$(init) = 1 for each PDB except pos(m)
- additive since actions only change one variable
- sum yields 3. (***Optimal!***)

$h^A(s)$: Multi-valued variables — Example 2

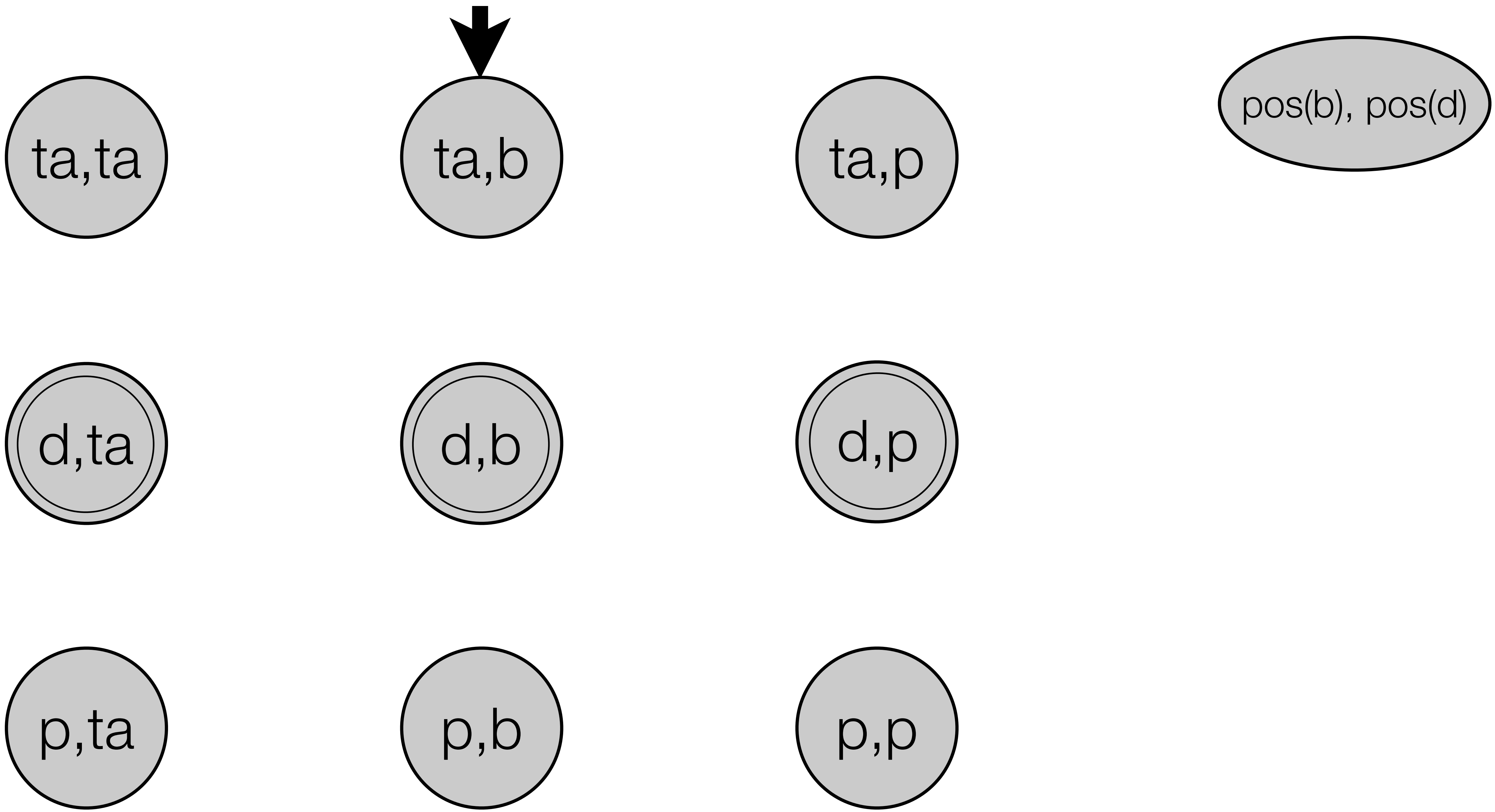
I = {
clear(p),
on(p,d),
on(d,b),
on-table(b)
}



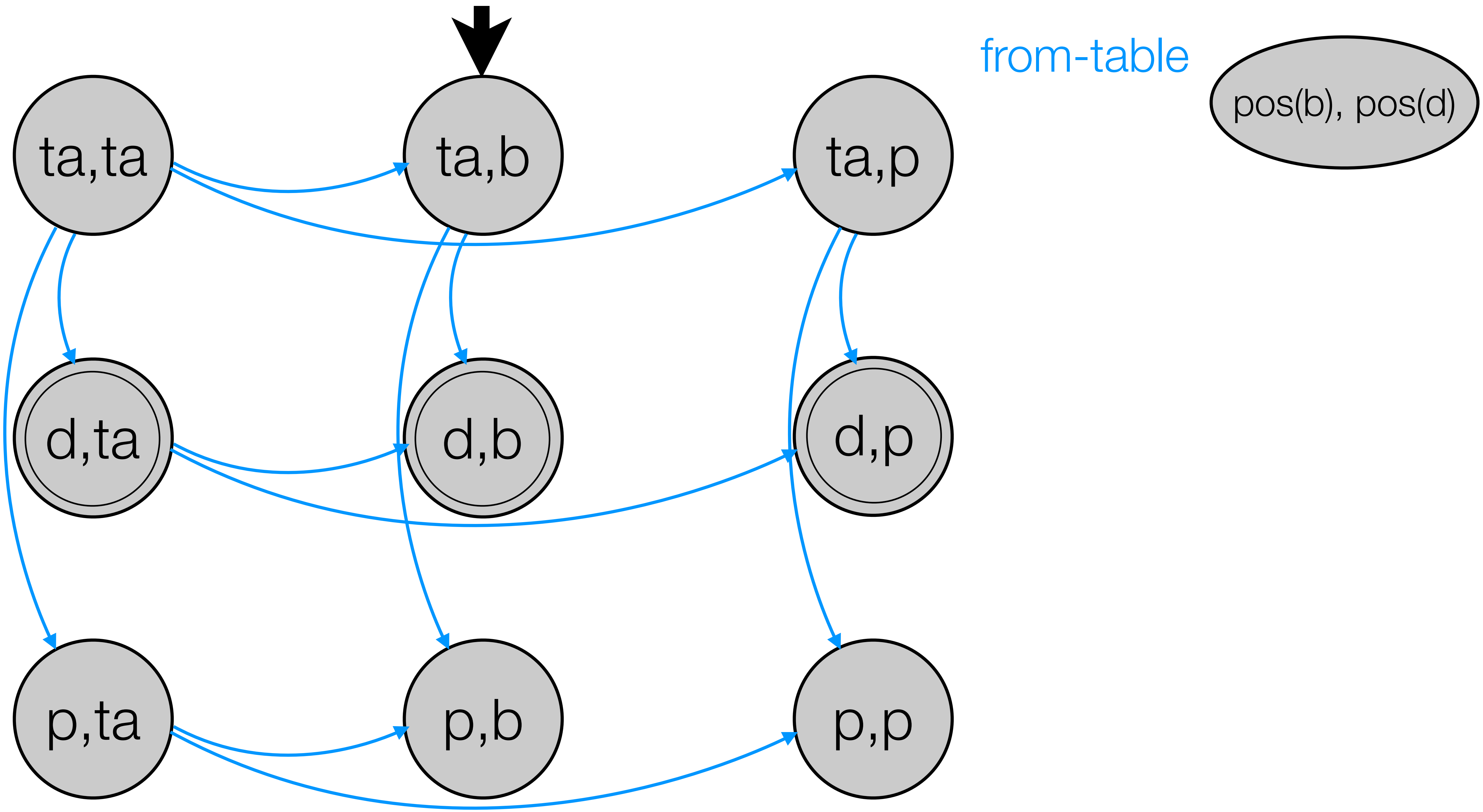
G = {
on(b,d)
}

-> PDB for {pos(b), pos(d)}

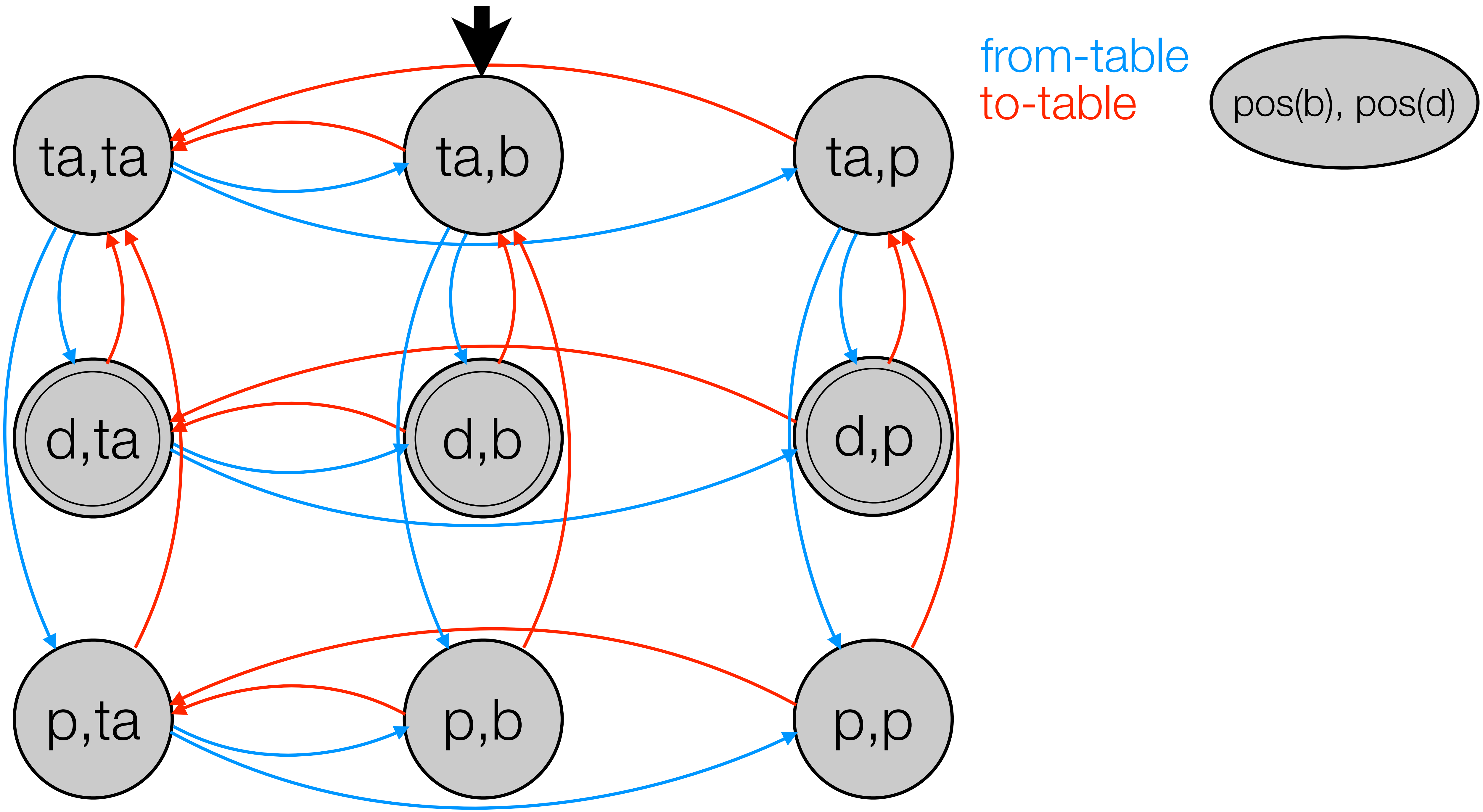
Pattern Database



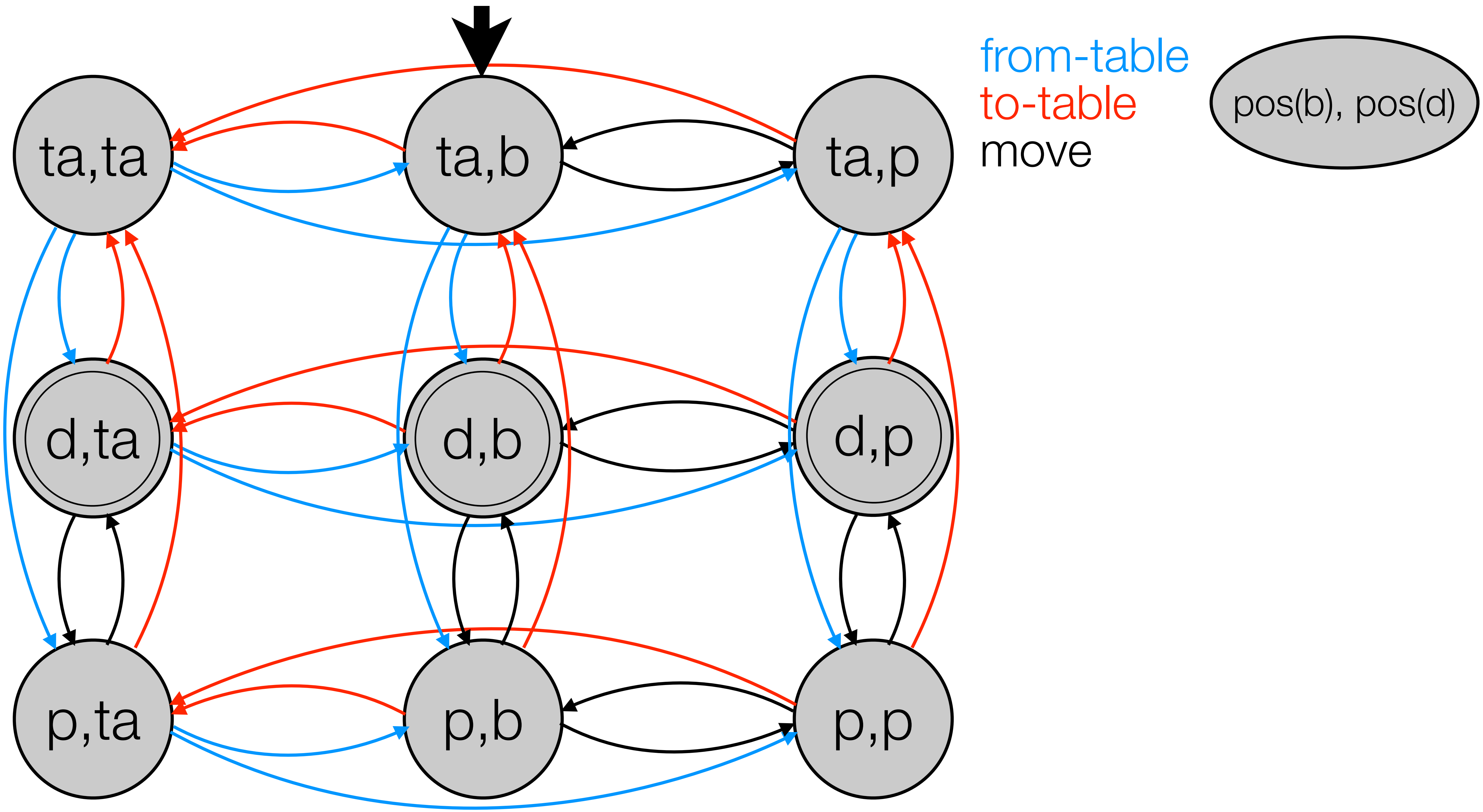
Pattern Database



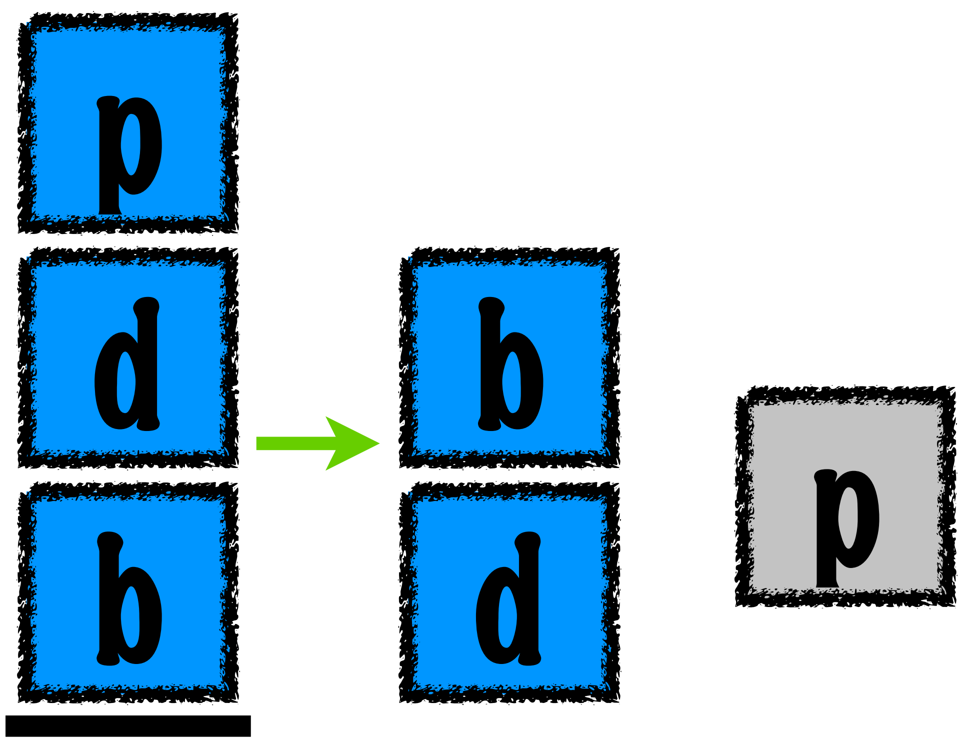
Pattern Database



Pattern Database



$h^A(s)$: Multi-valued variables — Example 2

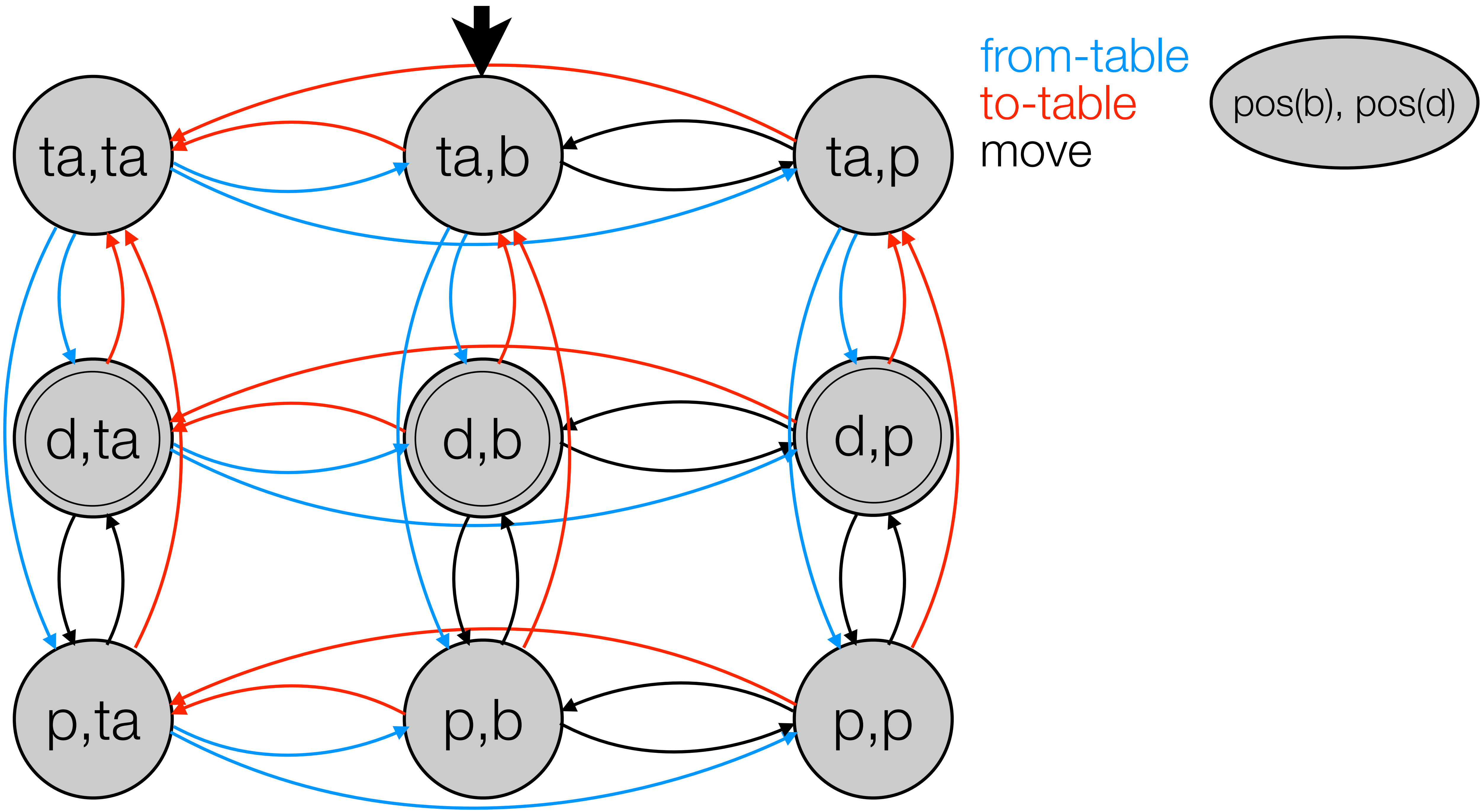


- PDB with {pos(b), pos(d)} yields h(s) = 1
- PDBs with all pos(x) still yield h(s) = 1
- ***problem***: pos(d) drops preconditions clear(d), clear(b)

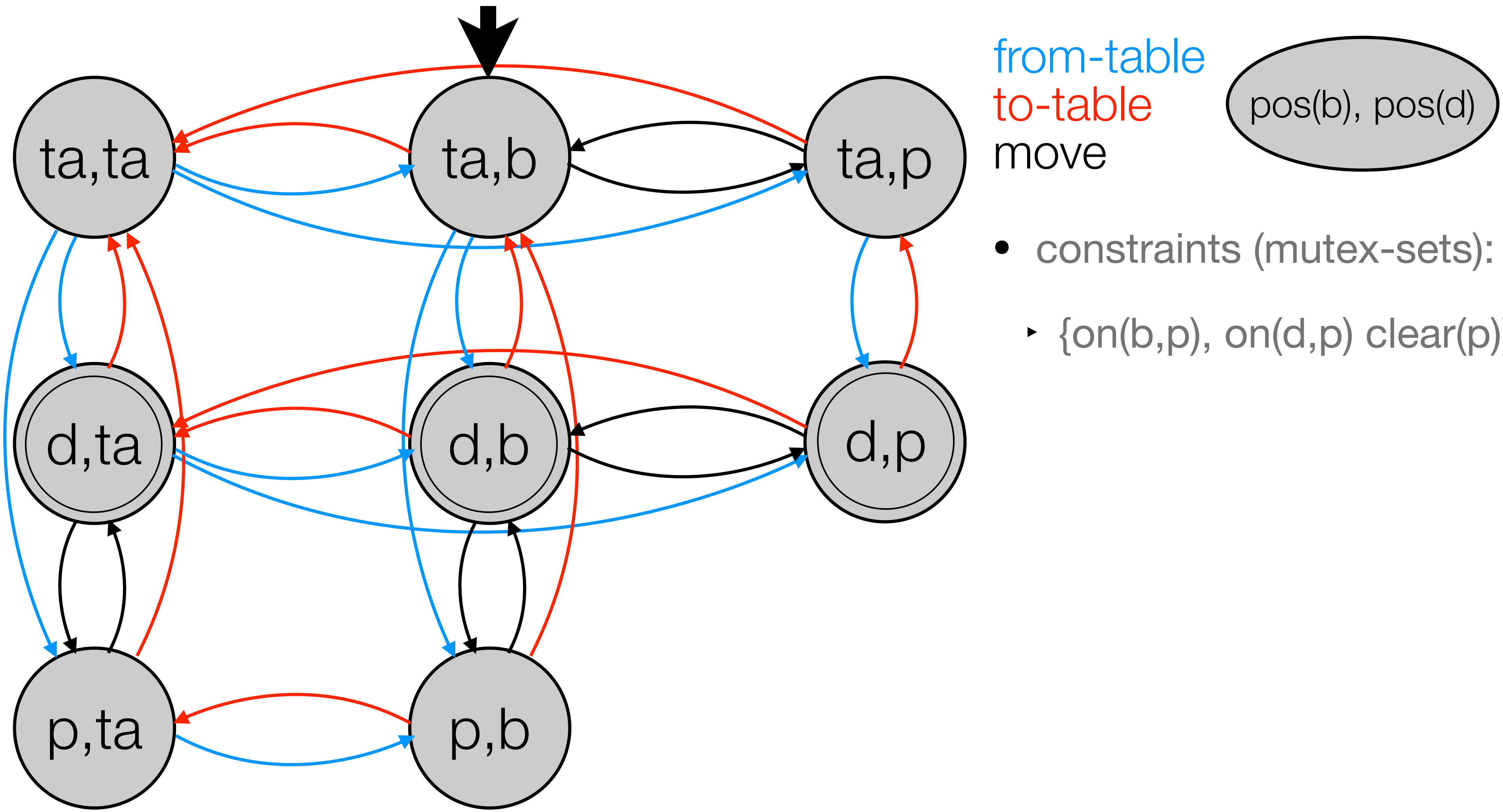
$h_C^A(s)$: Constrained PDBs

- Define mutex sets C_i (constraints)
- When constructing the PDBs,
  prune states and actions that violate constraints
- example: {on(p,d), on(b,d), clear(d)}

$h_C^A(s)$: Constrained PDBs

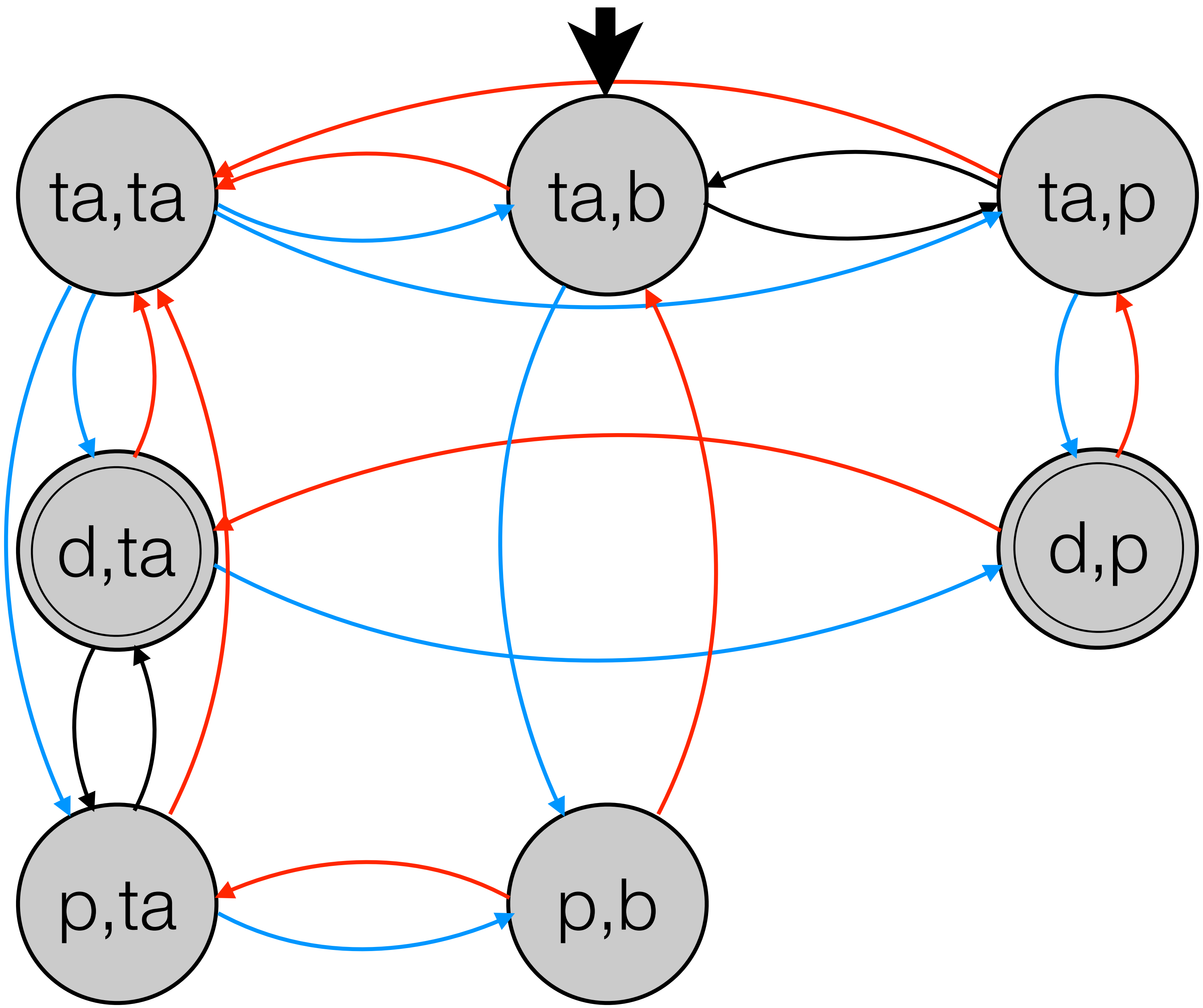


$h_C^A(s)$: Constrained PDBs

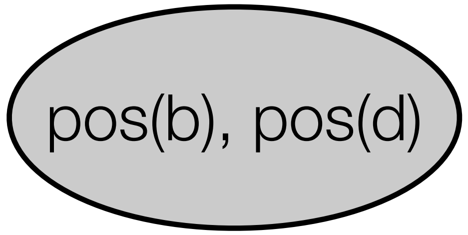


- constraints (mutex-sets):
  - {on(b,p), on(d,p) clear(p)}

$h_C^A(s)$: Constrained PDBs

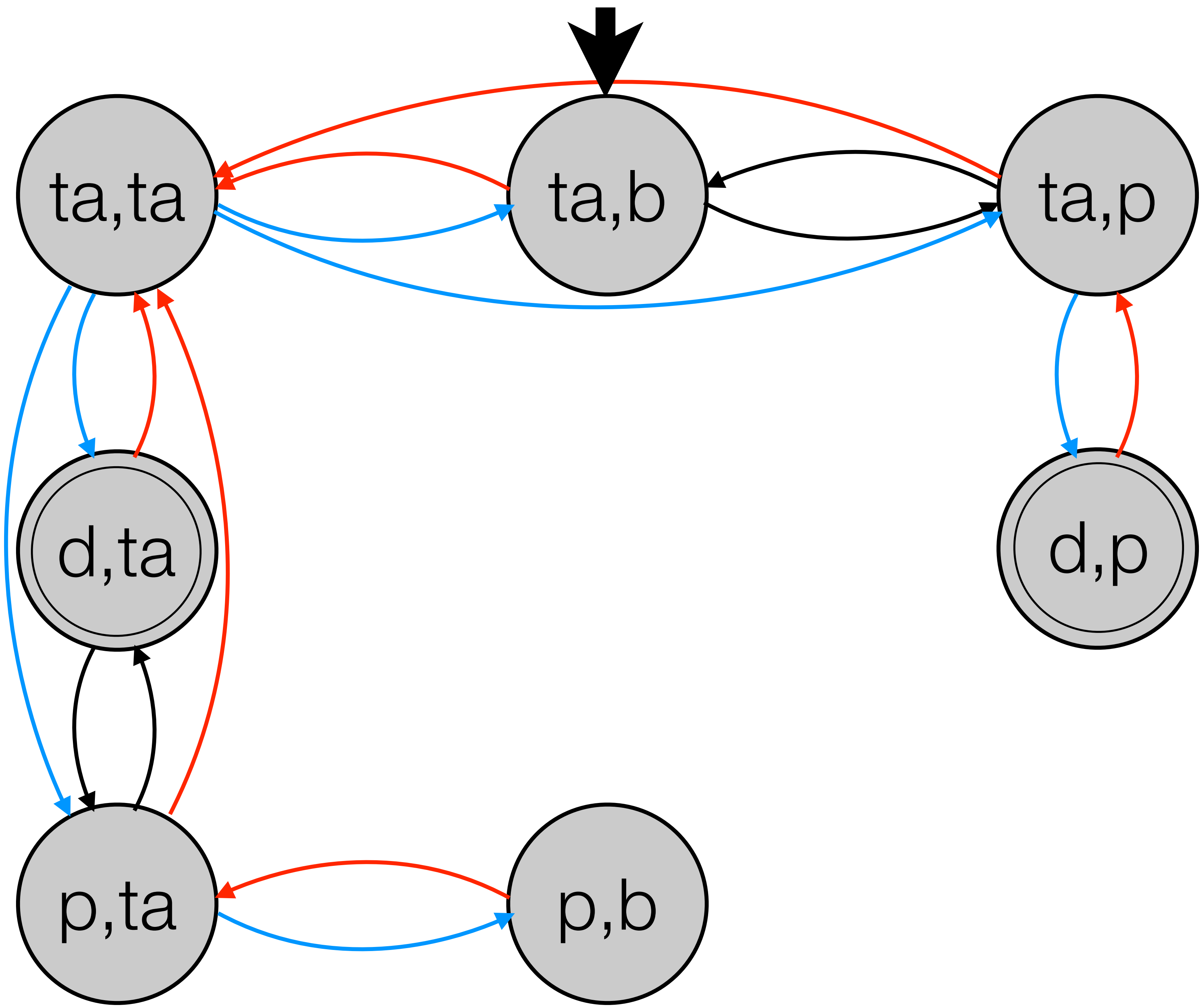


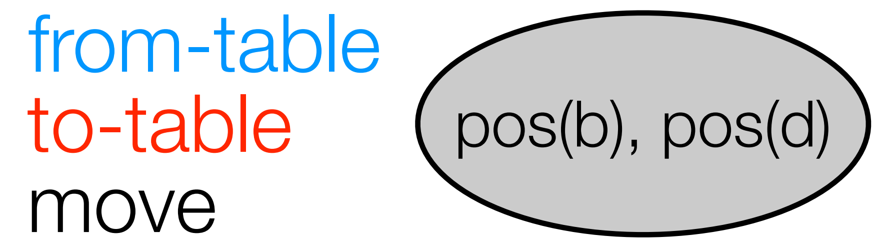
from-table
to-table
move



- constraints (mutex-sets):
  - {on(b,p), on(d,p) clear(p)}
  - {on(d,b), on(b,d)}

$h_C^A(s)$: Constrained PDBs





- constraints (mutex-sets):
  - {on(b,p), on(d,p) clear(p)}
  - {on(x,y), on(y,x)}
  - pre(from-table(x y): clear(x)
  - pre(to-table(x,y): clear(x))

$h^A_C(s)$: Properties

$$h^A(s) \leq h^A_C(s) \leq h^*(s)$$

- Is never worse than unconstrained PDBs
- Still admissible
- Conditions for additivity still applicable

h^m heuristics

h^m heuristics

- calculate the cost of achieving the most costly subgoal with size $\boldsymbol{m}$
- hint: $h^1 = h^{max}$

Regression Planning

- idea: search problem ***P*** backwards from goal
- search space ***R(P):***
  - states represent subset of atoms
  - transitions $(s, a, s') \in R(P)$:
    - $s \cap del(a) = \emptyset$
    - $s' = (s - add(a)) \cup pre(a)$

h^m heuristics

- calculate the cost of achieving the most costly subgoal with size $\boldsymbol{m}$ using regression

$$h^m(s) = \begin{cases} 0 & \text{if } s \subseteq s_0 \\ \min_{s':(s,a,s') \in R(P)} h^m(s') + cost(a) & \text{if } |s| \leqslant m \\ \max_{s' \subseteq s, |s'| \leqslant m} h^m(s') \end{cases}$$

- in our example: build tower **“atom”** from table

$$h^1(s_0) = 1$$

$$h^2(s_0) = 2$$

$$\dots$$

- **but:** costs for calculating $\boldsymbol{h^m}$ grow exponentially in $\boldsymbol{m}$

Additive h^m

- h^m generally not additive because actions are counted twice

➡ partition all actions into disjoint sets $\boldsymbol{A_i}$

- set cost in $h^m_{A_i}$ to 0 for all action not in $\boldsymbol{A_i}$
- don't count actions twice when summing up
- $\sum_i h^m_{A_i} = h^m_\Sigma$ is admissible

h^m_Σ — Example

- Recall: Tower assembly example: build tower **“atom”** from table
- Action Partitioning: $\boldsymbol{A_i}$ contains actions that move block $\boldsymbol{i} \in \{\boldsymbol{a,t,o,m}\}$
- we get: $h^m_{A_a} = 1$ $h^m_{A_t} = 1$ $h^m_{A_o} = 1$ $h^m_{A_m} = 0$
- $h^m_\Sigma = 3$ (**optimal!**)

$h^m_\Sigma(s)$: Properties

$$h^m_\Sigma(s) \leq h^*(s)$$

- The sum of cost partitioned $\boldsymbol{h^m}$ heuristics is admissible
- Action partitioning can be applied to any admissible heuristic!

Action Partitioning

"Our basic approach is to create one partition A_i for each goal atom g_i,

and assign actions to the partition where they appear to contribute the most to the sum"

Action Partitioning

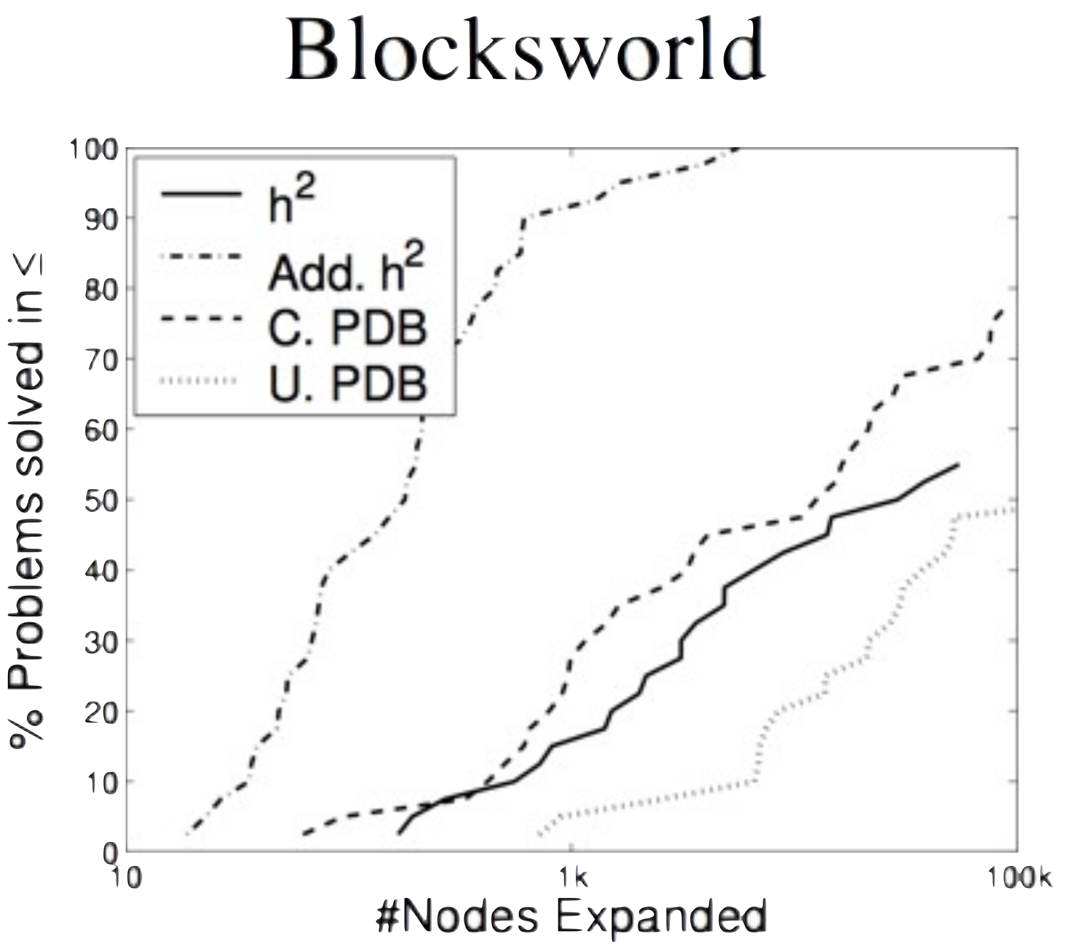
- start: assign each $\boldsymbol{g_i}$ an action set $\boldsymbol{A_i}$ containing all actions
- compute $h^1_A(g_i) - h^1_{A-\{a\}}(g_i)$ for each goal atom
  - measure for relevance of $\boldsymbol{a}$ towards $\boldsymbol{g_i}$
  - leave $\boldsymbol{a}$ inside the action set with greatest loss, remove from rest

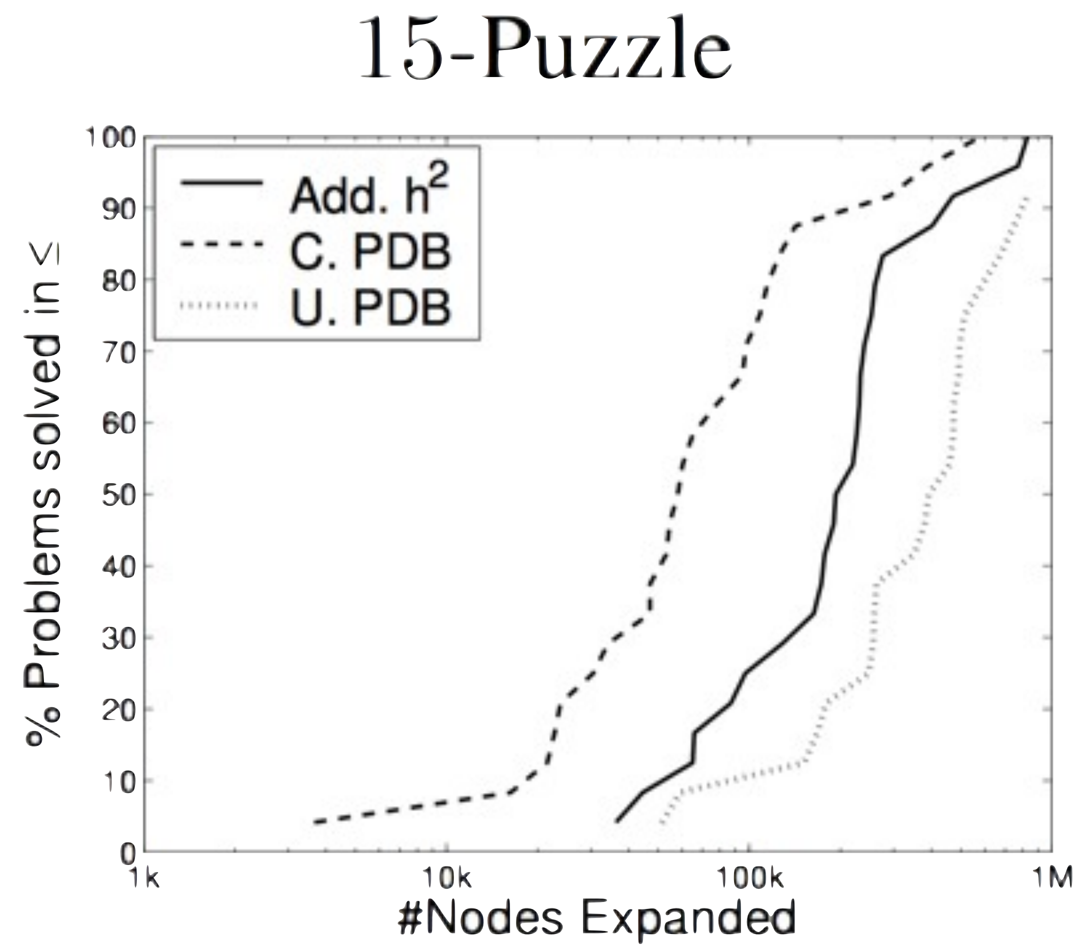
Action Partitioning

- **Problem:** relaxing a single action often not enough
  - recursive / backward calculation
  - branching on preconditions
- relax action sets instead of single actions:
  - do action partitioning similar to next presentation

Evaluation

Evaluation





- Unsolved: exceeding 1M nodes, 16h CPU, 1GB RAM
- constrained PDBs and additive $\boldsymbol{h^2}$ exceed “old” approaches
- which is better? domain dependent!
  (over all problems, $\boldsymbol{h^2}$ performs better, but uses more CPU time)

Thank you.

Questions?