Lazy is Efficient
(in plan recognition)

Gal A. Kaminka

The MAVERICK Group

Computer Science Department

(Gonda Brain Research Center, Nanotech Center)

Bar Ilan University, Israel

Abstract Mind Architecture
(in artificial intelligence)

Two Processes

Perception/World Modeling:	Action Selection/Decision Making:
Vision, sensor processing, sensor fusion, ...	Planning, plan execution, goal deliberation, reacting...

My Scientific Problem:

The Nature of the Social Mind

The computational mechanisms that allow the mind to reason about, and interact with, others?

(differently than *about* or *with* inanimate objects)

Abstract Mind Architecture
(in artificial intelligence)

Two Processes

Perception/World Modeling:	Action Selection/Decision Making:
Theory of Mind: Intents, plans of other minds (in AI: Plan Recognition)	**Coordination:** Manipulating, acting w.r.t other minds

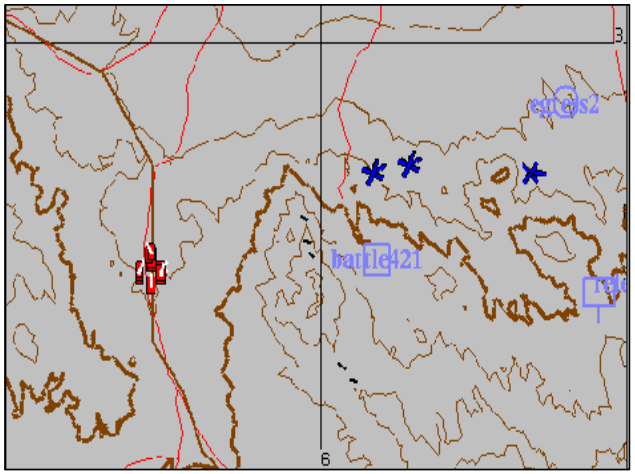
This talk

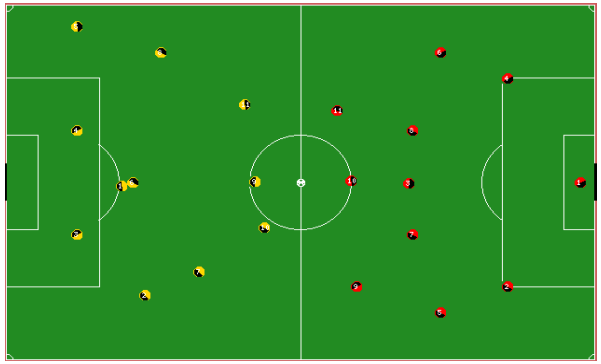
Since 1995….

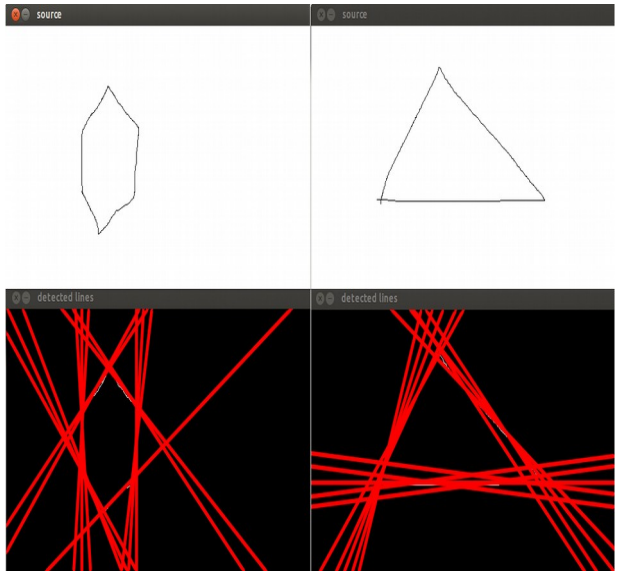
- Tracking teammates (RESL, w/ Tambe)
- Overhearing teams (YOYO, YOYO*, w/ Tambe, Pynadath)
- Overhearing using Colored Petri-nets (w/ Gutnik)
- **Symbolic Recognition (SBR, w/ Avrahami-Zilberbrandt)**
- Decision-theoretic rec. (UPR, w/ Avrahami-Zilberbrandt)
- Intent detection (w/ Bochek-Dokow)
- **Mirroring (w/ Vered)**

See: www.cs.biu.ac.il/~galk/publications/class_rescat.html

Dynamic, Continuous, Multi-Agent







Recognition Challenges

- Online: Observations incrementally received
- Complex state observations
  - Observe state, not agent actions
  - State is factored (composed of multiple features)
- Situated (reactive)
  - Observed agents react, deviate from own plans
  - What goal/plan is true <u>now</u>
- Observations stem from continuous world
  - Sensitivity to discretization (loss of information)

SBR: Symbolic Behavior Recognition [IJCAI 2005, MOO 2004-2005]

- Online, keyhole recognition, discrete observations
- Applied to vision tracker data (e.g., airport security)

SBR: Symbolic Behavior Recognition
[IJCAI 2005, MOO 2004-2005]

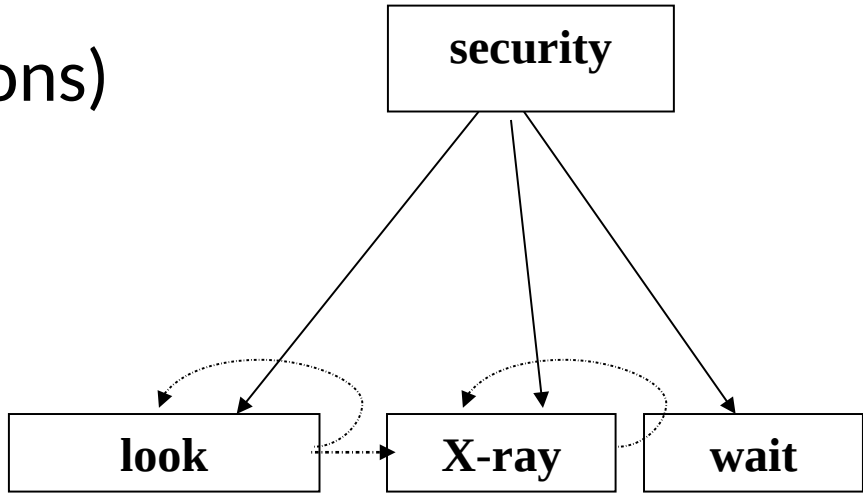
- Online, keyhole recognition, discrete observations
- Applied to vision tracker data (e.g., airport security)

What is agent doing?

Is its behavior anomalous?

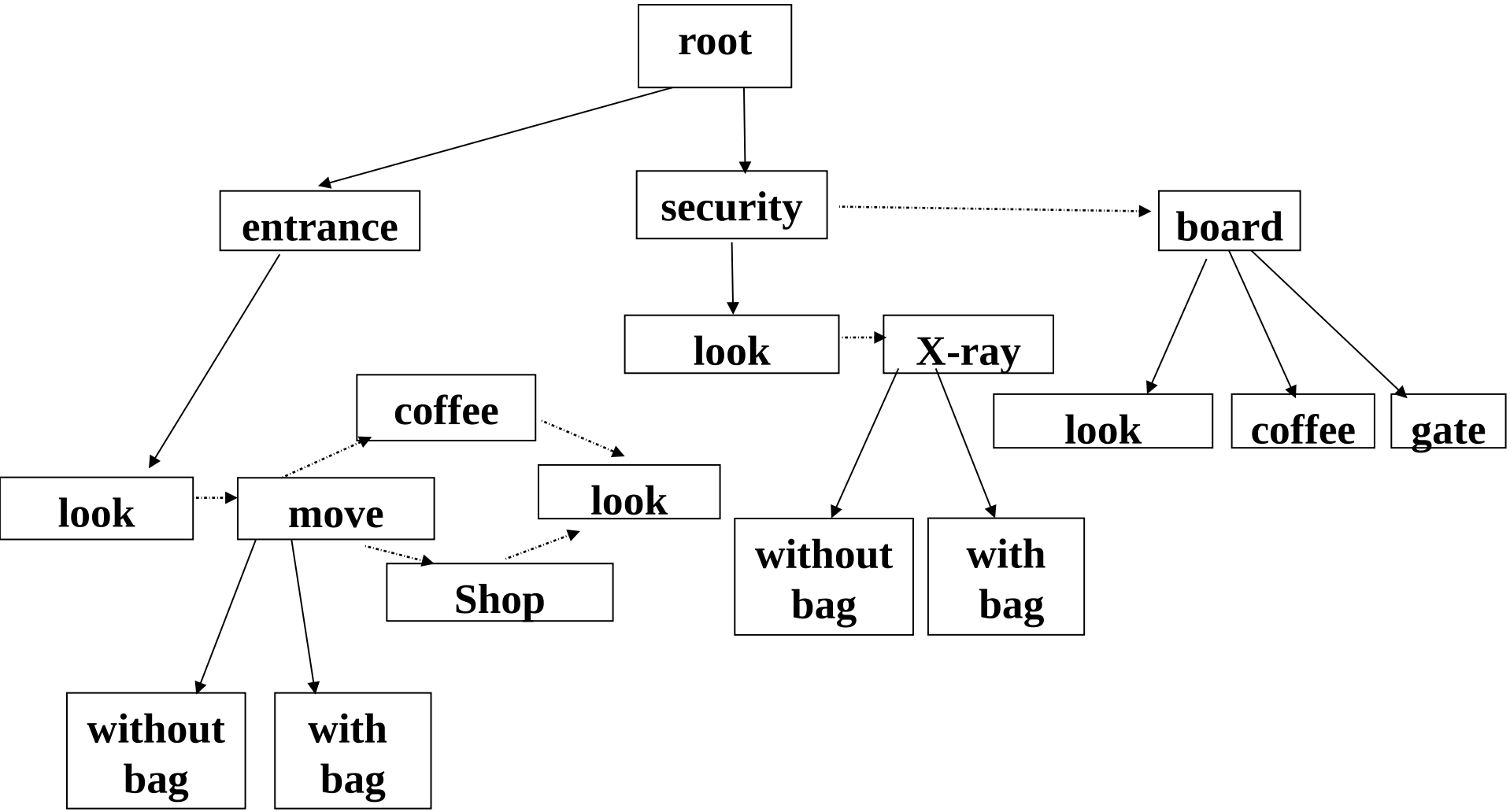
SBR Plan Library: Layered, Durative Actions

- Directed acyclic connected graph
- Vertices denote plan steps (actions)
- Edges
  - Vertical (decomposition) edges
  - Horizontal (sequential) edges

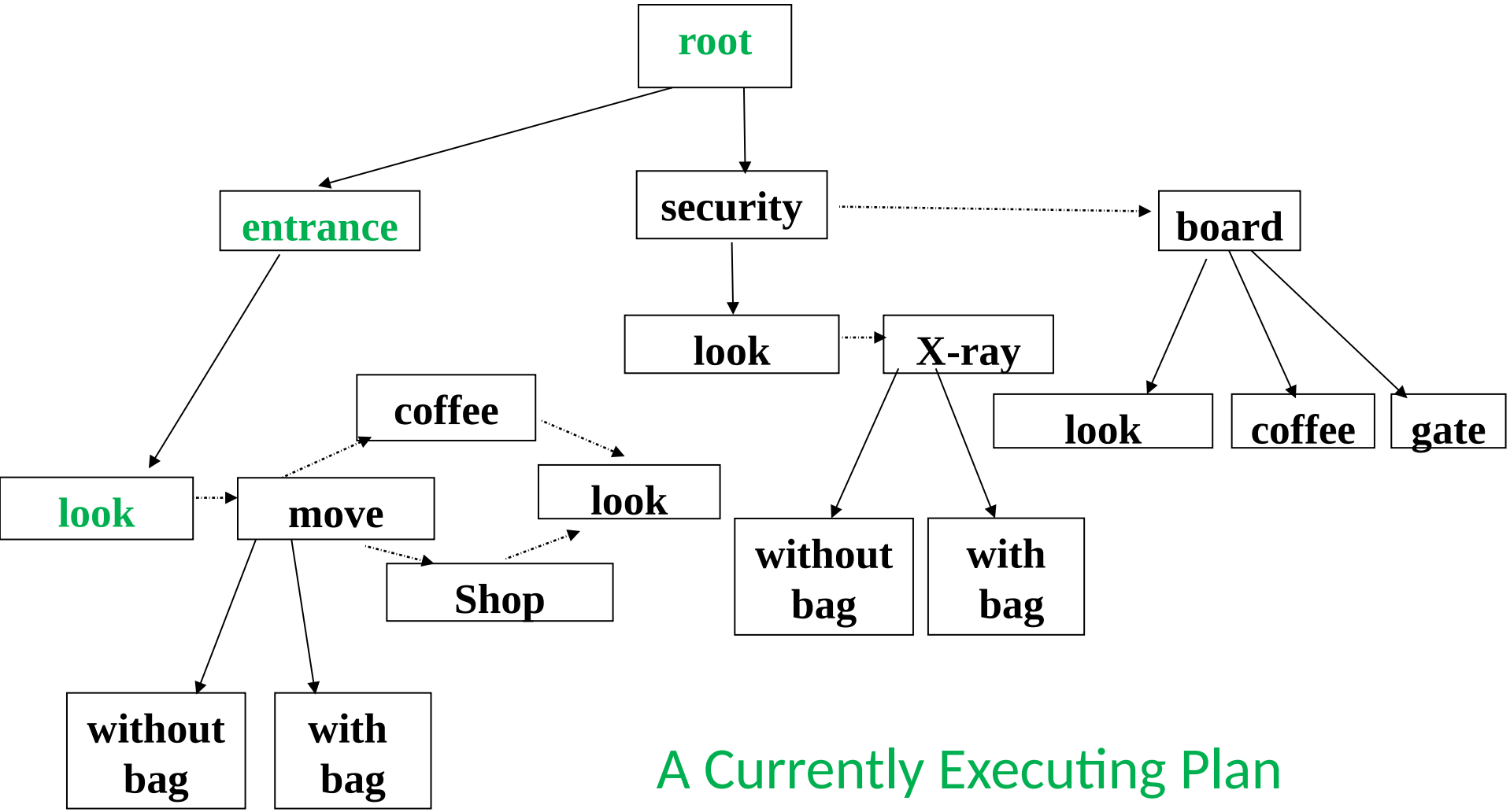


- <u>Every</u> vertex may generate obs, have duration (cycles)
- "Or"-graph (edges denote ordered choices)

Example: Plans and Hypotheses

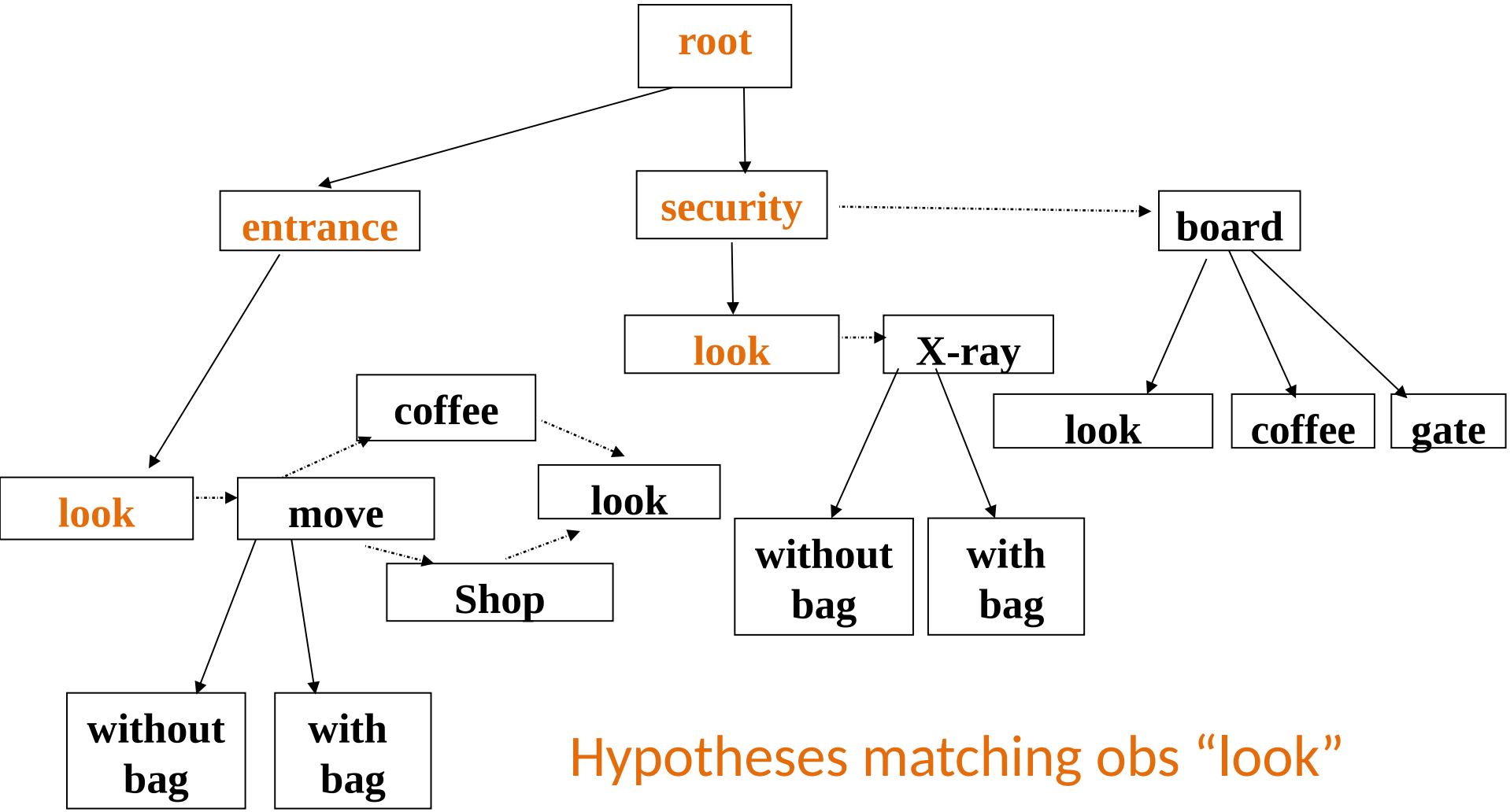


Example: Plans and Hypotheses



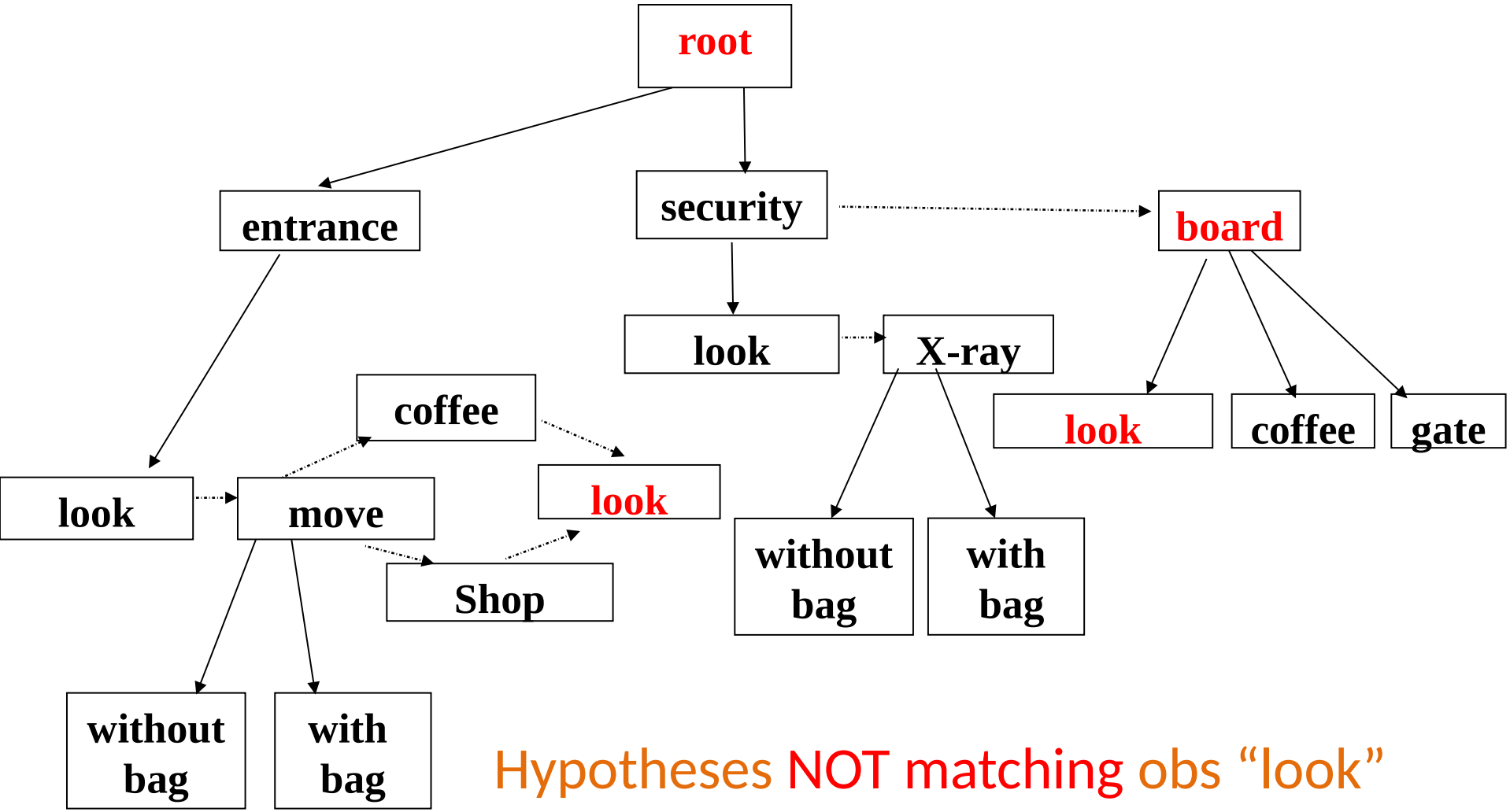
A Currently Executing Plan

Example: Plans and Hypotheses



Hypotheses matching obs “look”

Example: Plans and Hypotheses



Hypotheses NOT matching obs “look”

SBR Key Ideas

- Sacrifice memory to gain speed
  - Entire grounded library in memory, in advance
  - Auxiliary data structures to support fast queries
- Distinguish recognition queries
  - Given last observation, what might be true now
  - Given history of observations, what might have been true

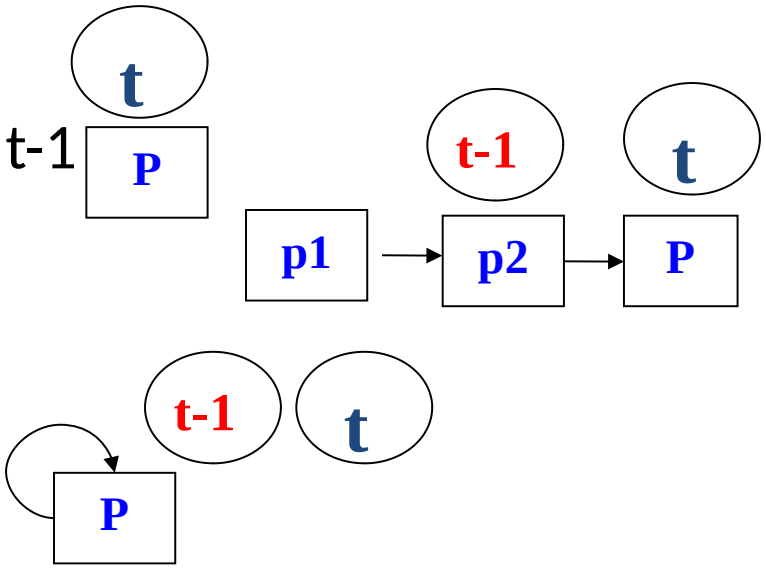
SBR Key Ideas

- Sacrifice memory to gain speed
  - Entire grounded library in memory, in advance
  - Auxiliary data structures to support fast queries
- **Distinguish recognition queries**
  - **Given last observation, what might be true now**
  - **Given history of observations, what might have been true**

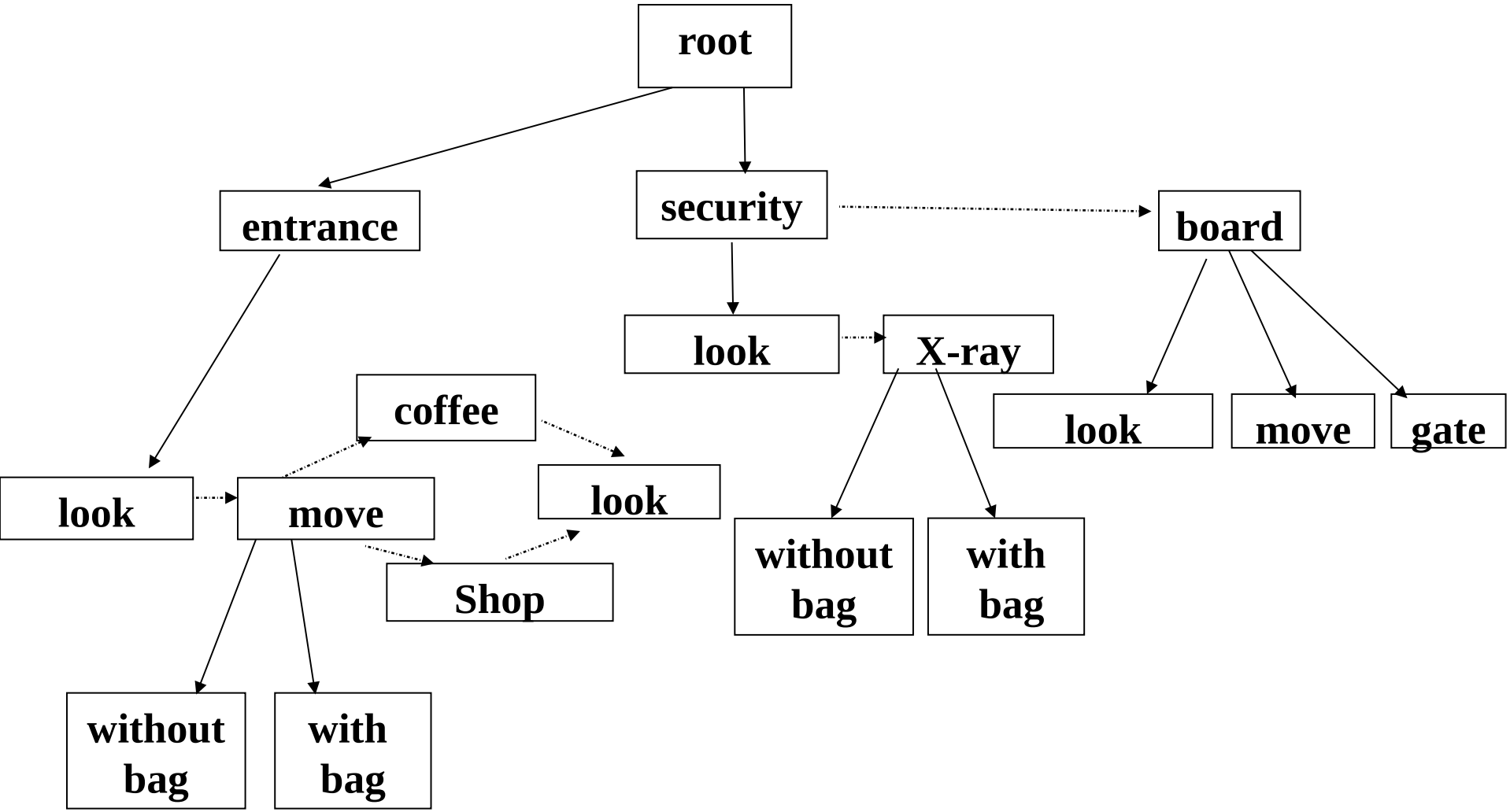
This allows computing only what is needed!

Current State Query
What <u>is</u> True

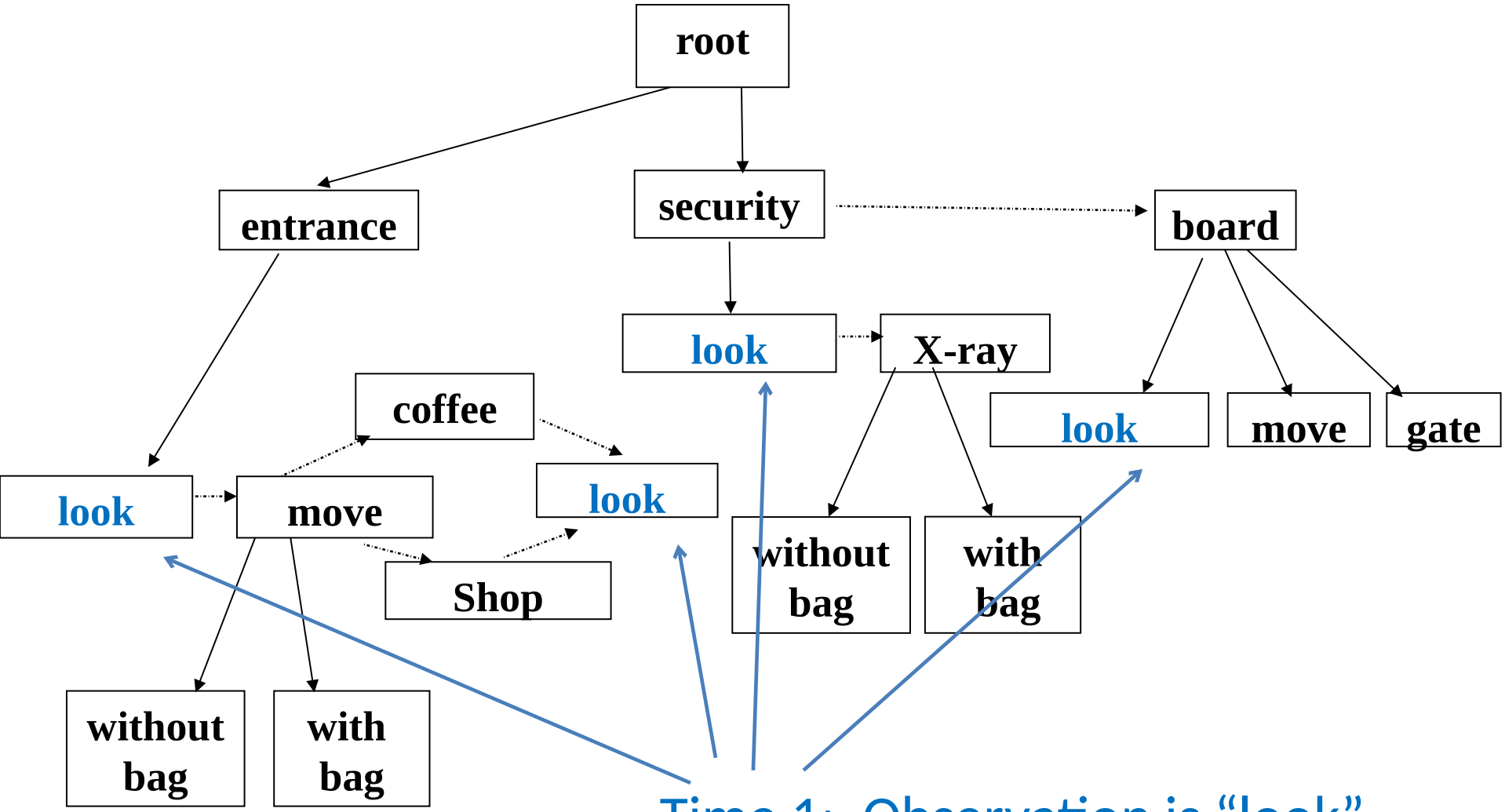
- Generating plan-path hypotheses:
  - Tag matching plans by observation time
  - Propagate tags up/down according to <u>temporal consistency</u>
- Plan P is <u>temporally consistent</u> in time-stamp t if:
  - No incoming sequential edges
  - OR: Exist previous plan tagged with t-1
  - OR: Tagged with t-1
- Efficient: O(M log L)
  - M number of matching plan steps
  - L size of library



Current State Query: Example

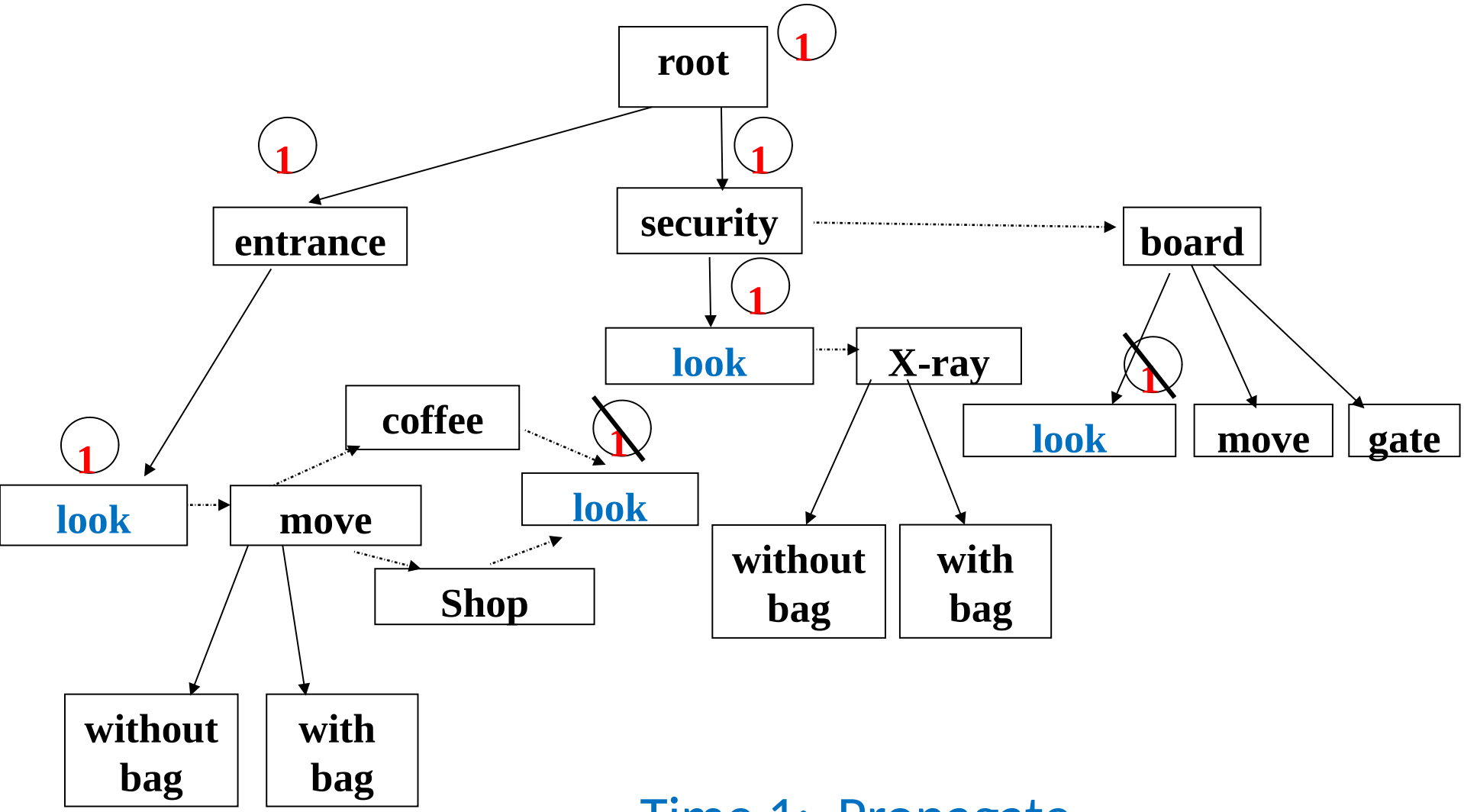


Current State Query: Example



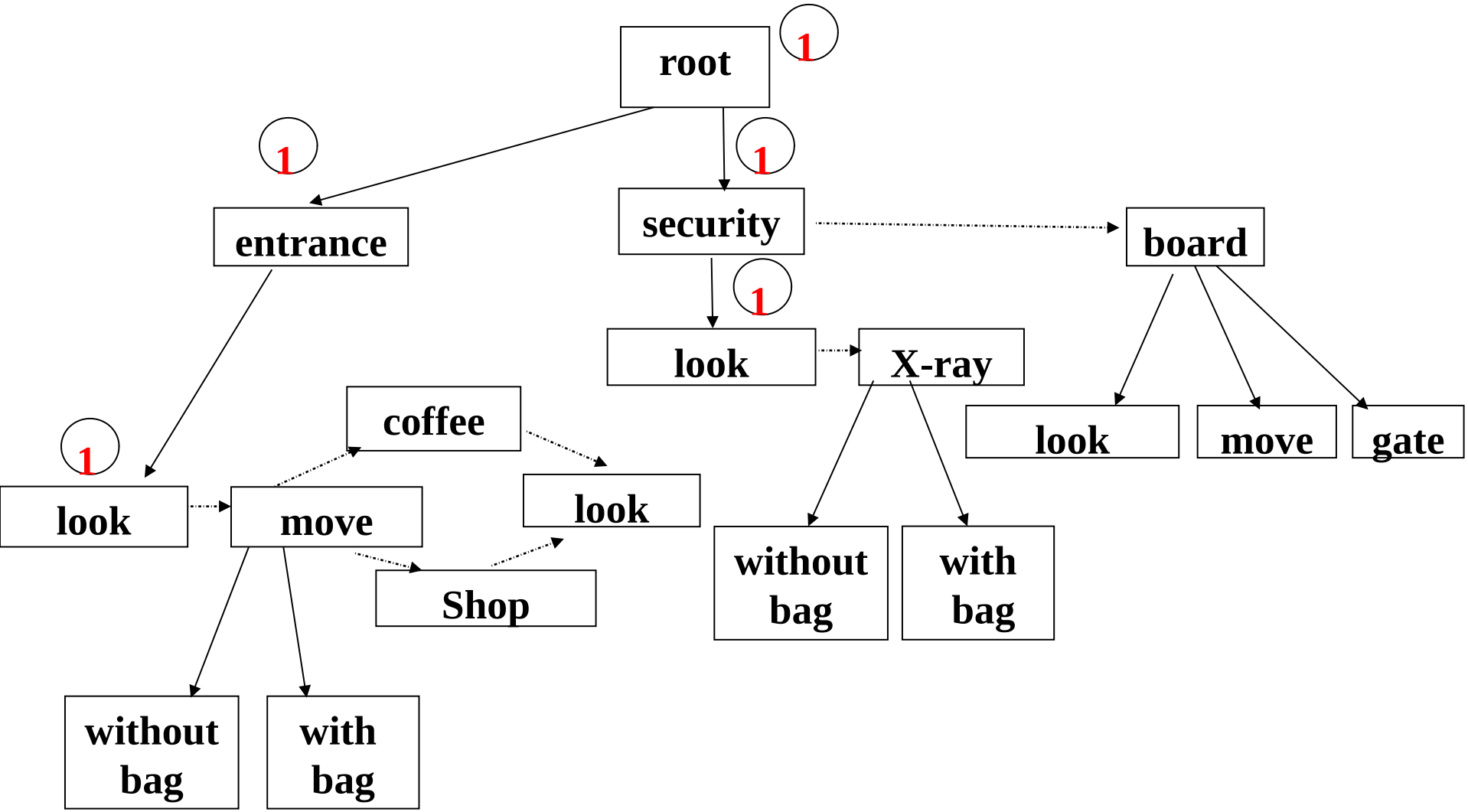
Time 1: Observation is “look”

Current State Query: Example

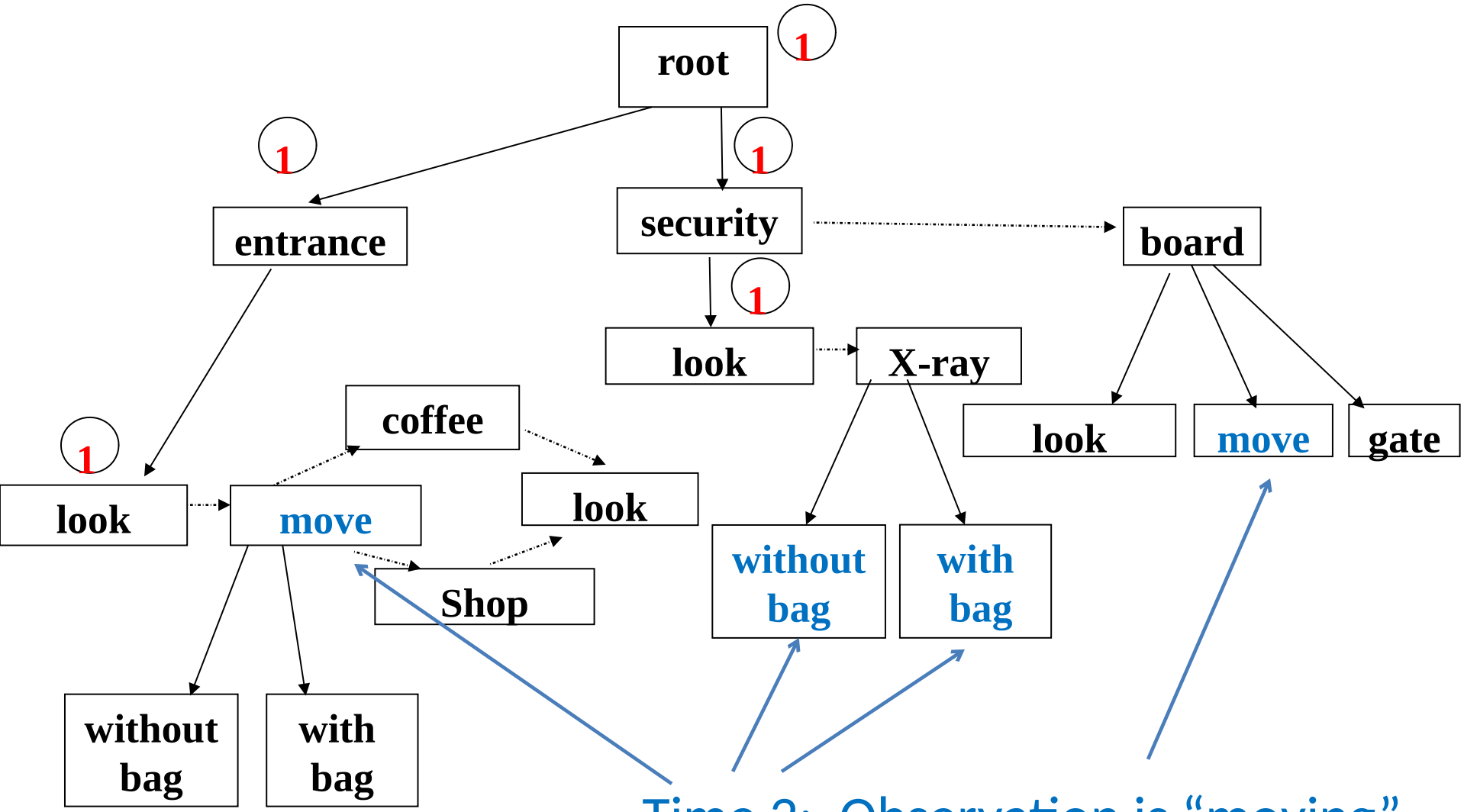


Time 1: Propagate

Current State Query: Example

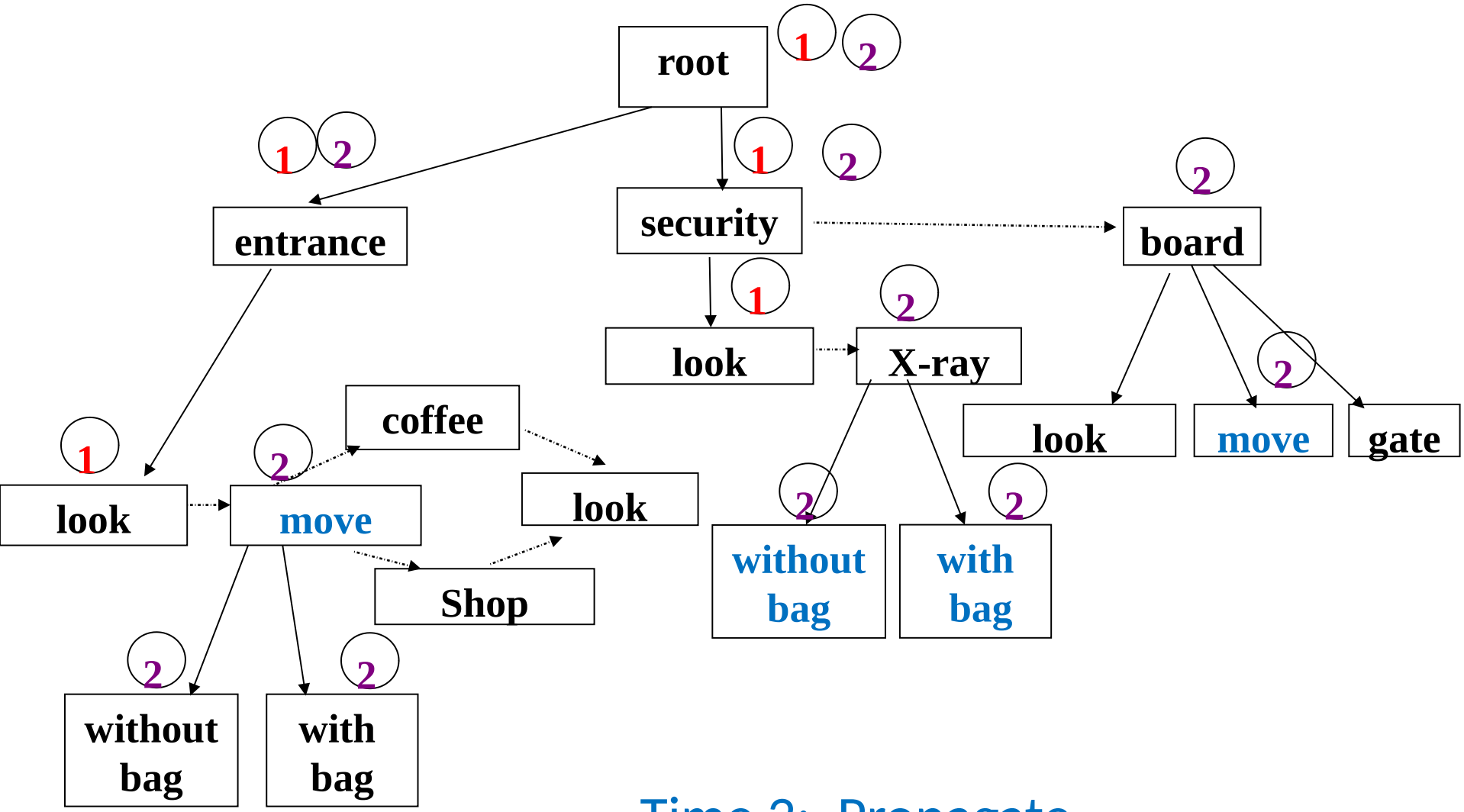


Current State Query: Example



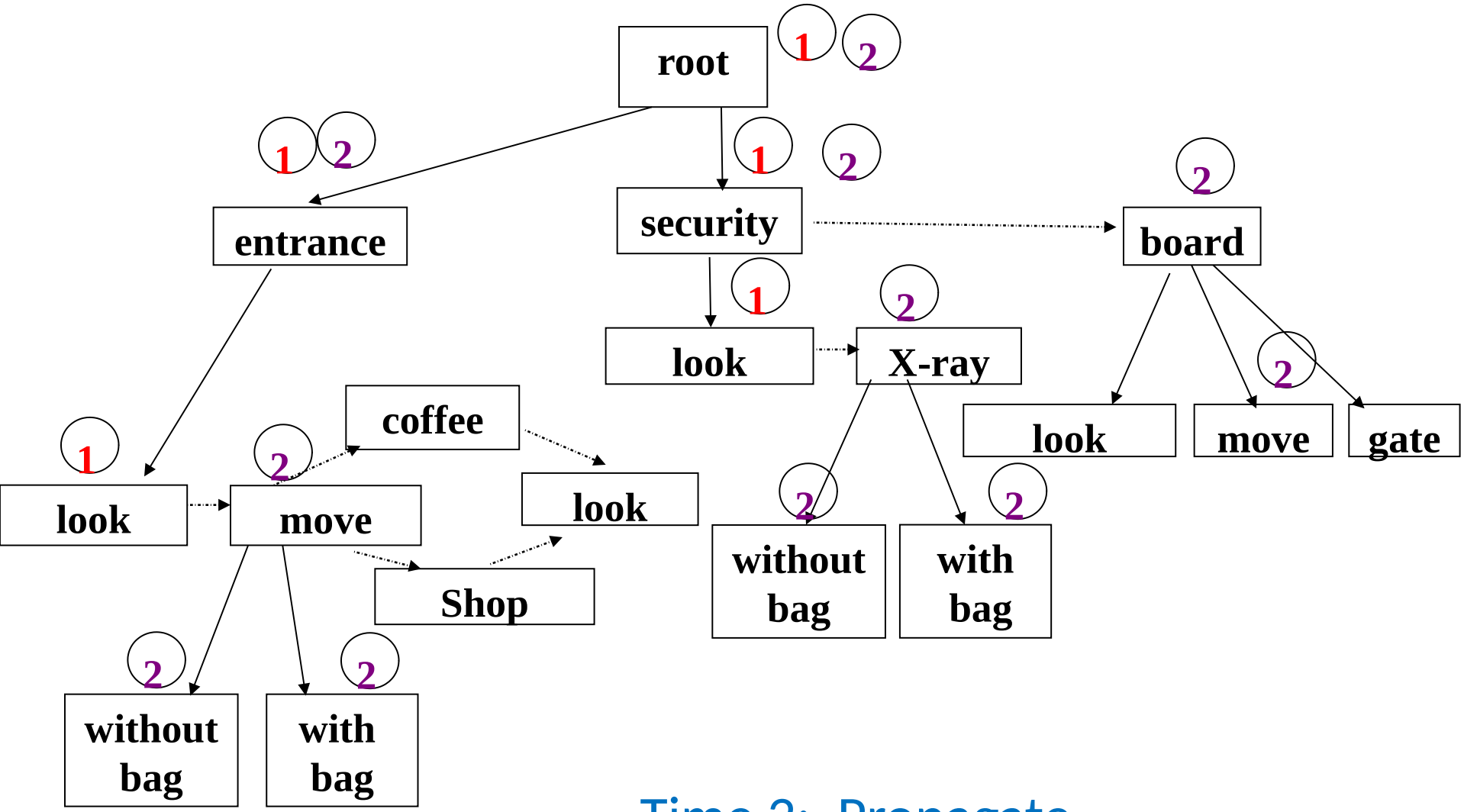
Time 2: Observation is “moving”

Current State Query: Example



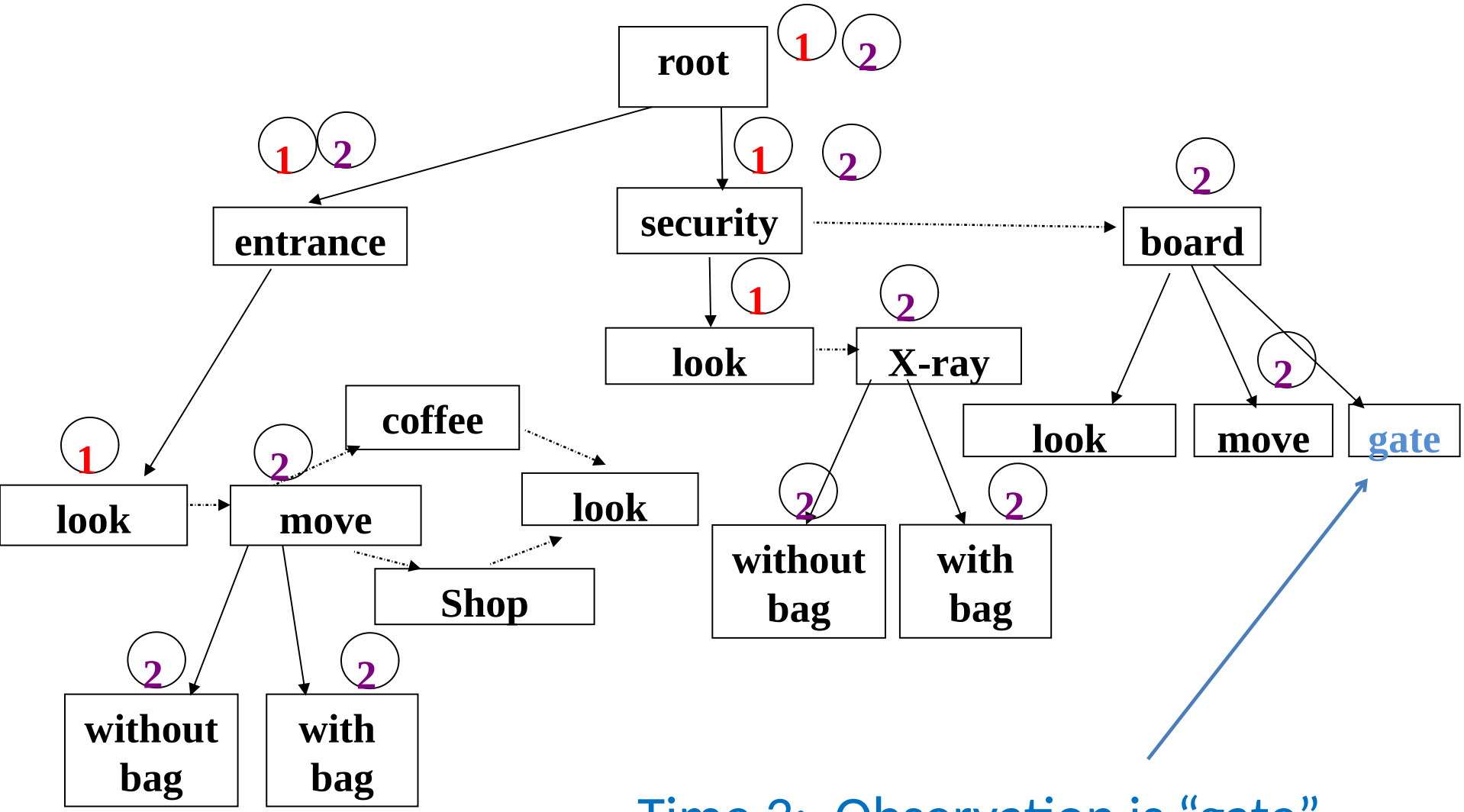
Time 2: Propagate

Current State Query: Example



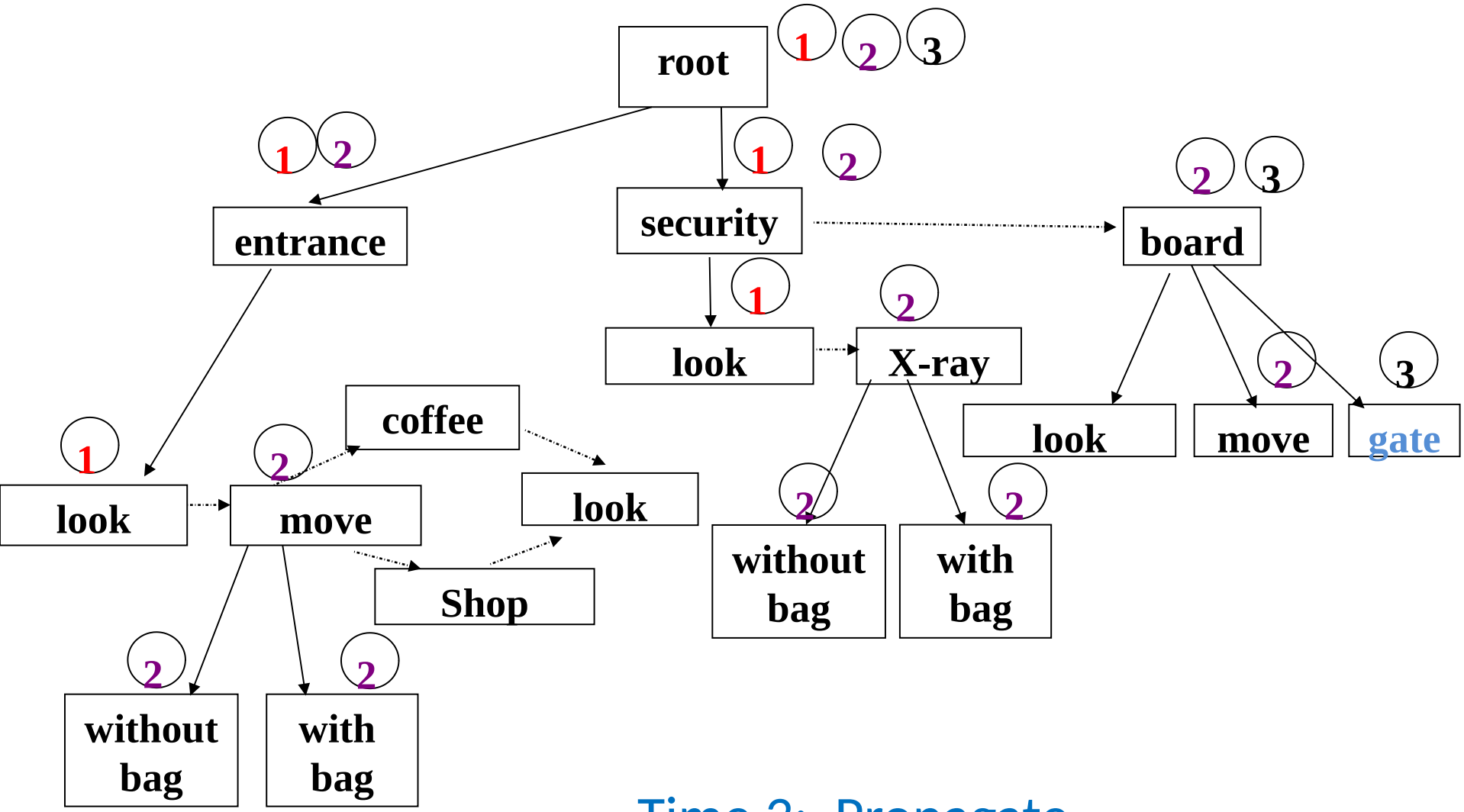
Time 2: Propagate

Current State Query: Example



Time 3: Observation is “gate”

Current State Query: Example

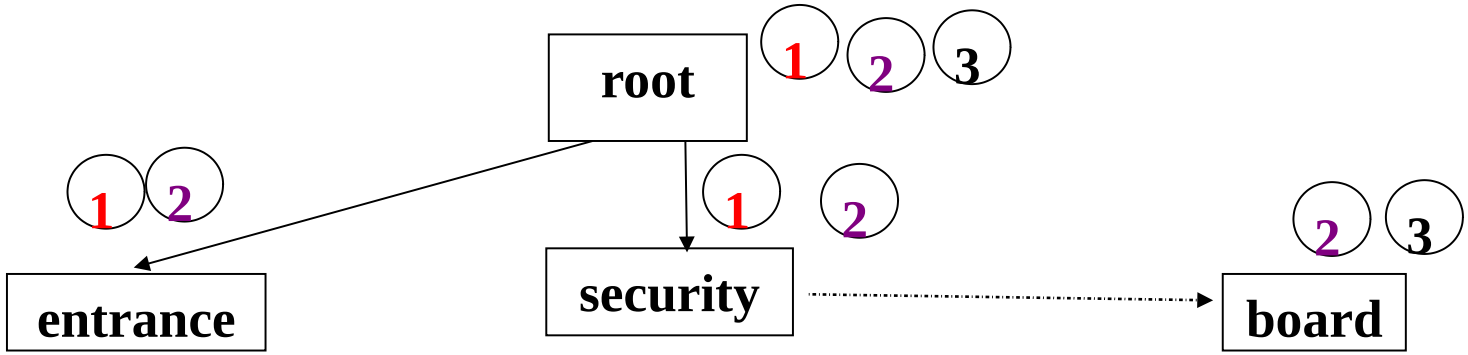


Time 3: Propagate

History of States Query

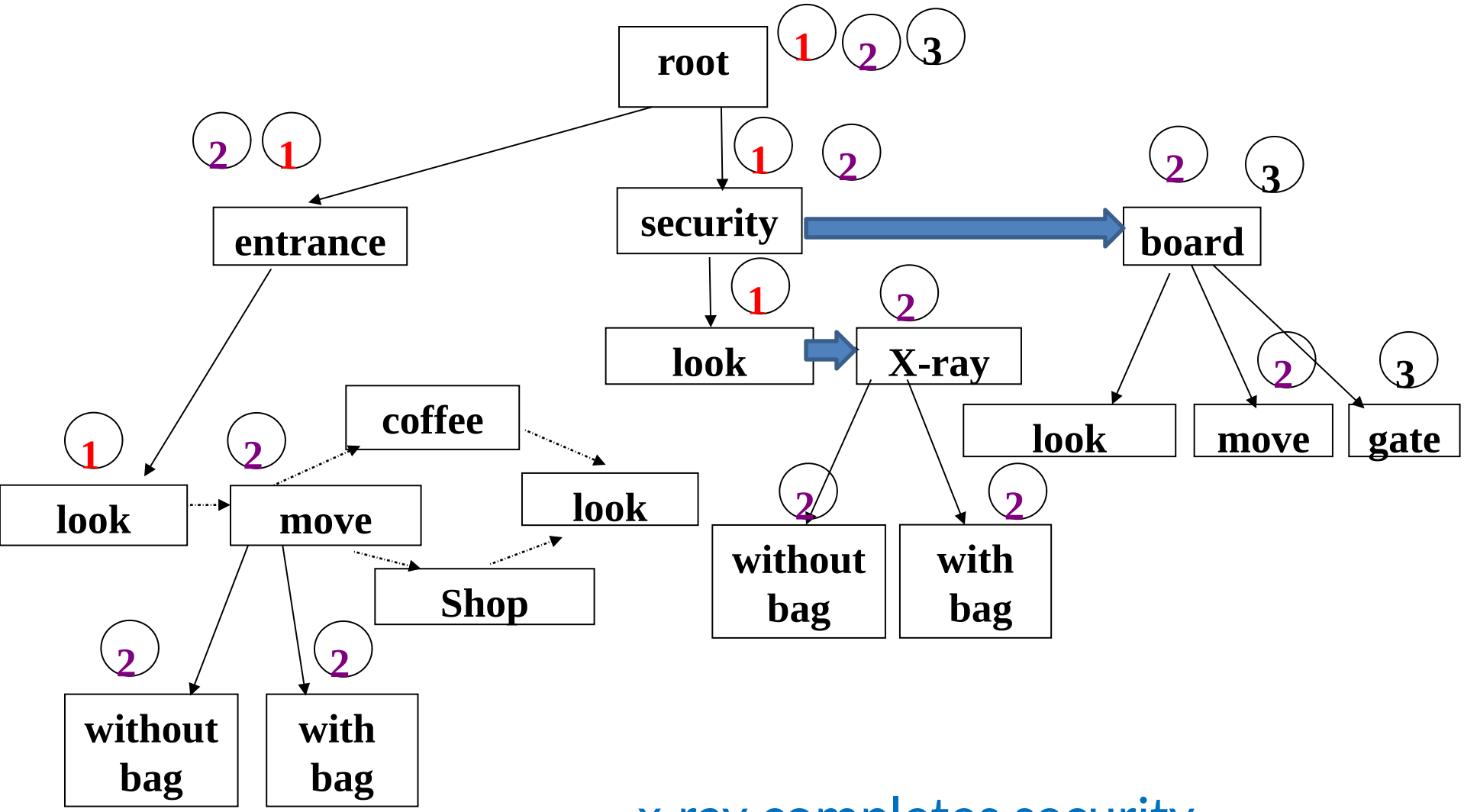
What <u>was</u> True

- CSQ is <u>situated</u>: only hypothesizes as to current state
- Does not explain complete sequences



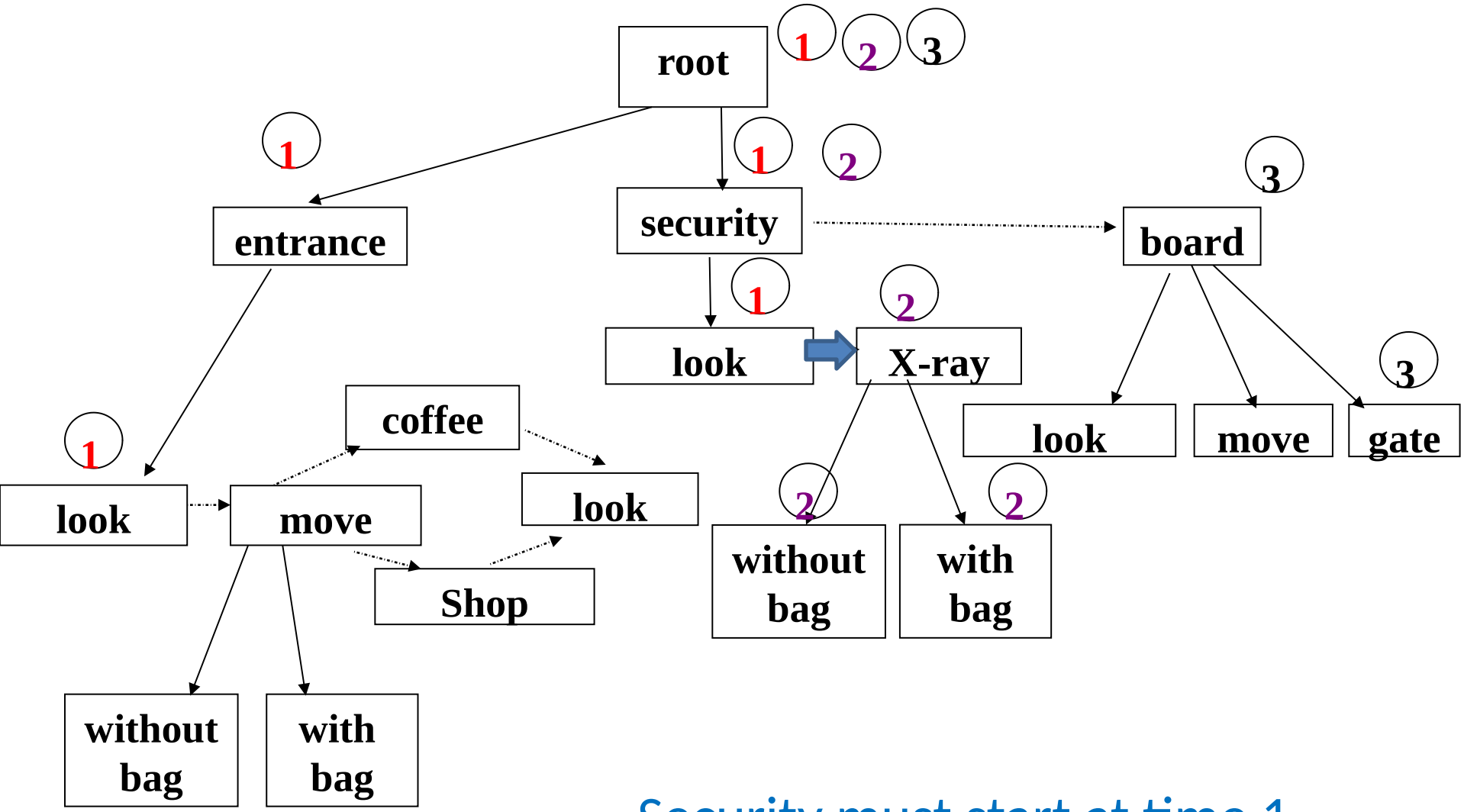
All possible sequences (1,2,3)? **No.**

History of States



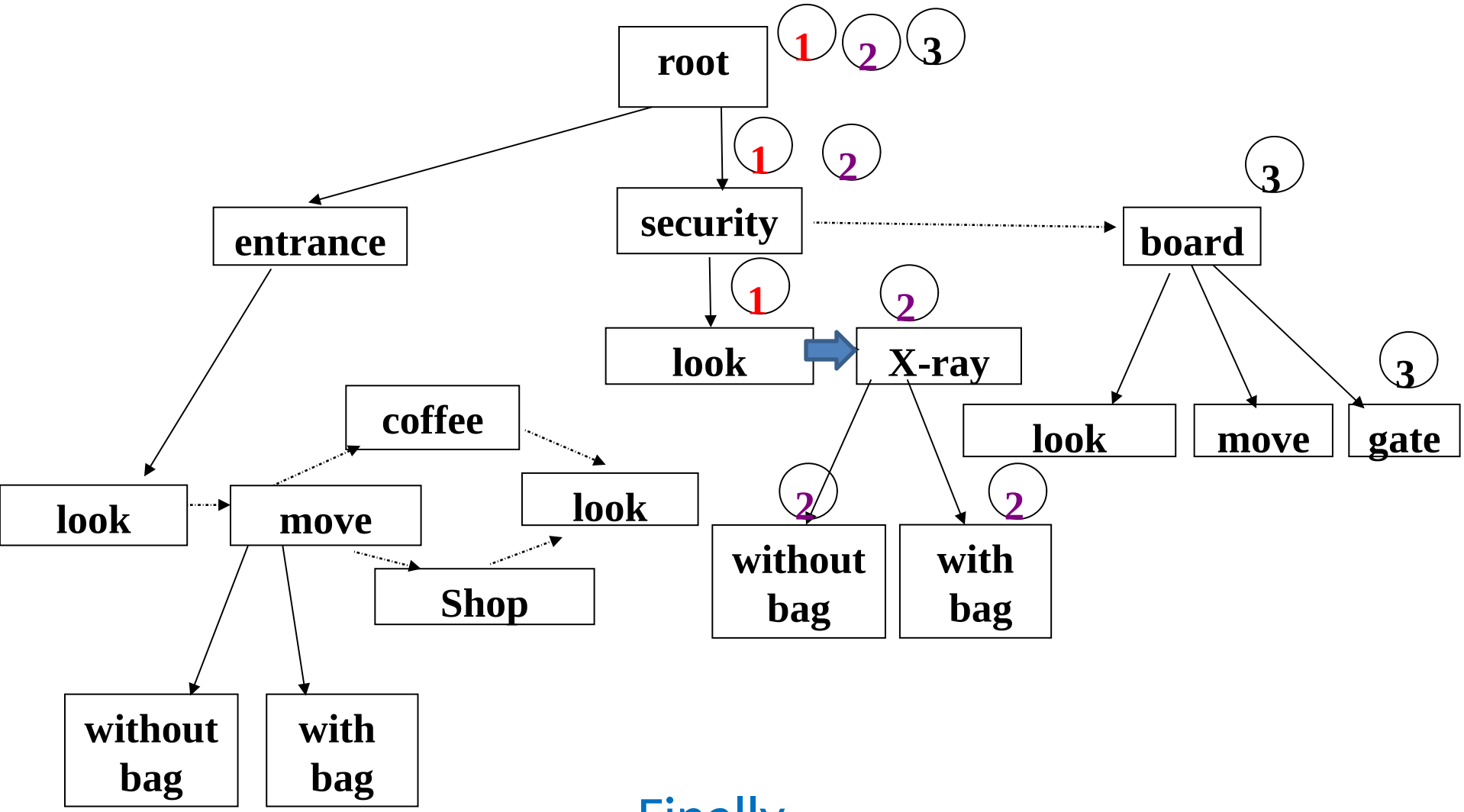
x-ray completes security
Security terminates at time 2

History of States



Security must start at time 1

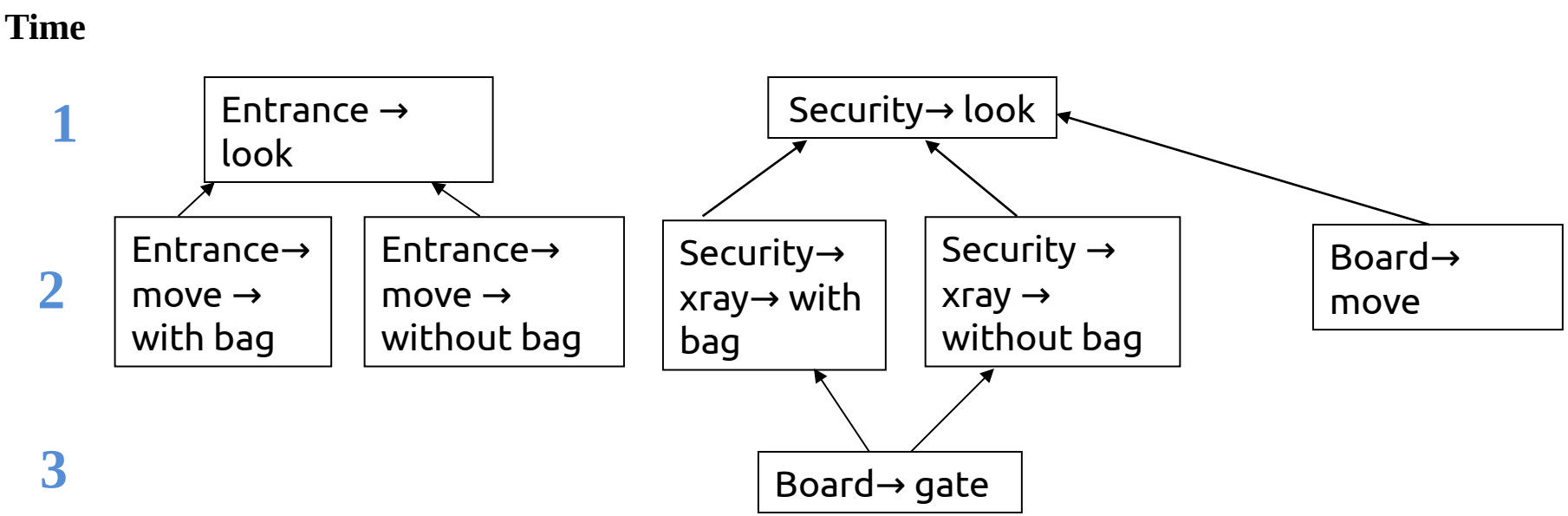
History of States



Finally

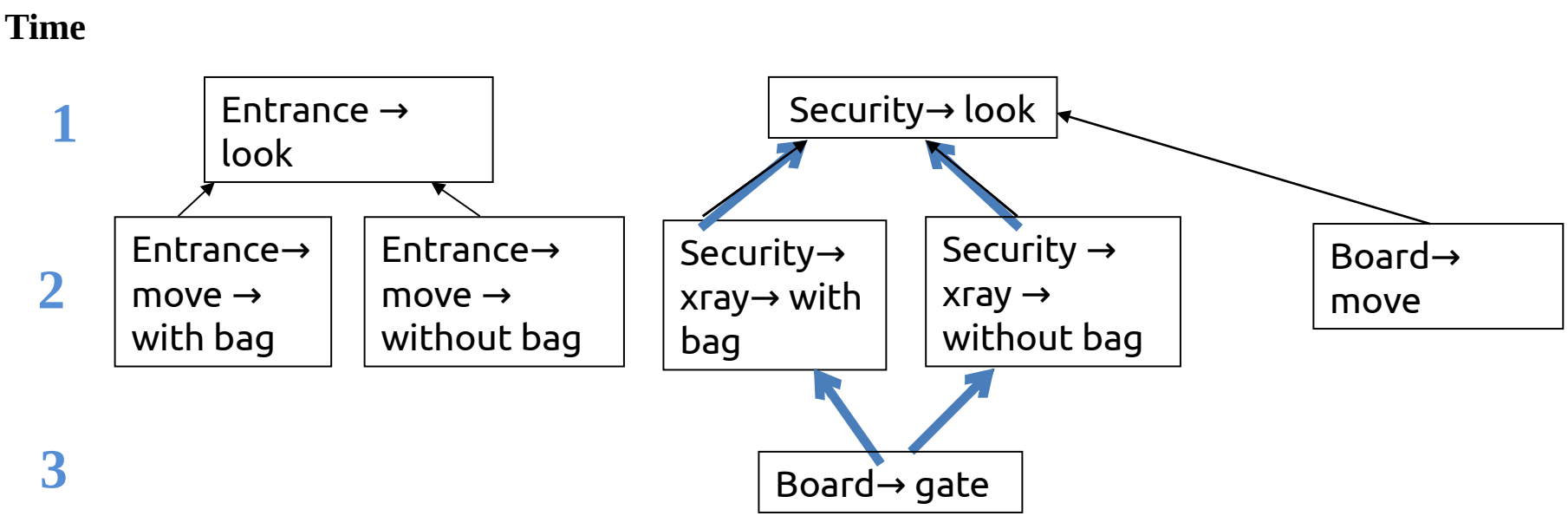
History of States Query

- Generate (or incrementally build) <u>hypotheses graph</u>
  - Vertices: denoting hypotheses at time *t*
  - Edges: connect valid continuations from *t* to *t+1*



History of States Query

- Generate (or incrementally build) hypotheses graph
  - Vertices: denoting hypotheses at time *t*
  - Edges: connect valid continuations from *t* to *t+1*
- Valid hypotheses: paths from vertices in time *t* to time *1*



SBR Highlights
[IJCAI 2005, MOO 2004-2005]

- <u>Highly efficient, Complete:</u>
  - Match observations to plan library vertices in O(1)
  - CSQ is O(M log L), M # of matches, L size of library
  - History query is polynomial:
    - Graph construction is $O(M^2)$ for each observation
    - Hypotheses extraction is $O(TM^2)$
- Extensions for interleaving, interrupting, …
- Limitation: grounded library, must fit in memory

Lazy Commitment in SBR

- Book keeping allows delaying inference
- <u>Compute only if queried</u>
  - e.g, hypotheses graph can be built only on history query
- No commitment to ranking
  - Probabilistic or decision-theoretic ranking is separate
  - SBR as filter [AAAI 2007]; PHATT, SLIM use similar approach

Mirroring
[ACS 2016, IJCAI 2017, AAAI 2018]

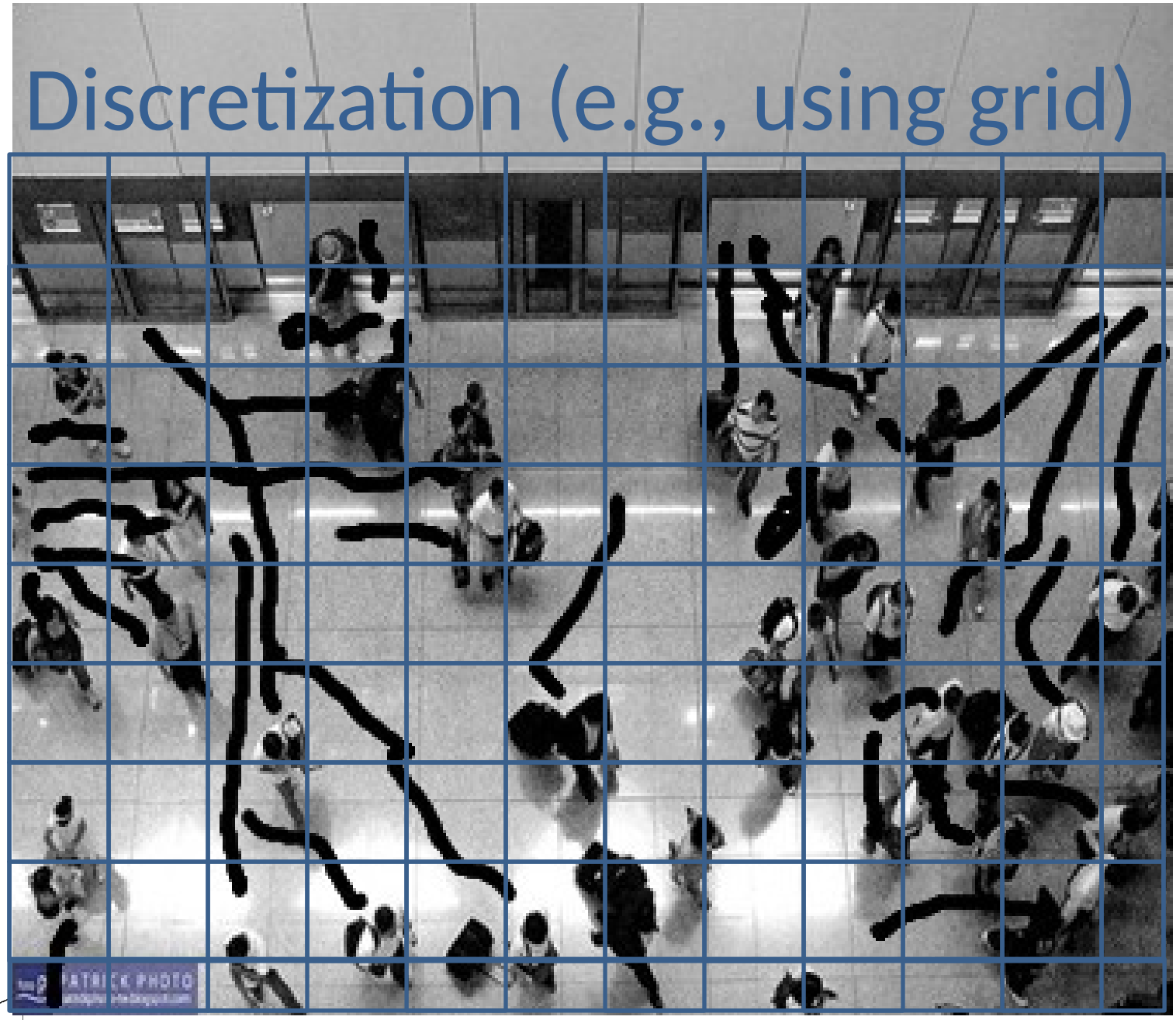
Observations and Plans are in Continuous Space

gestures, motions, goal locations: <u>trajectories</u>

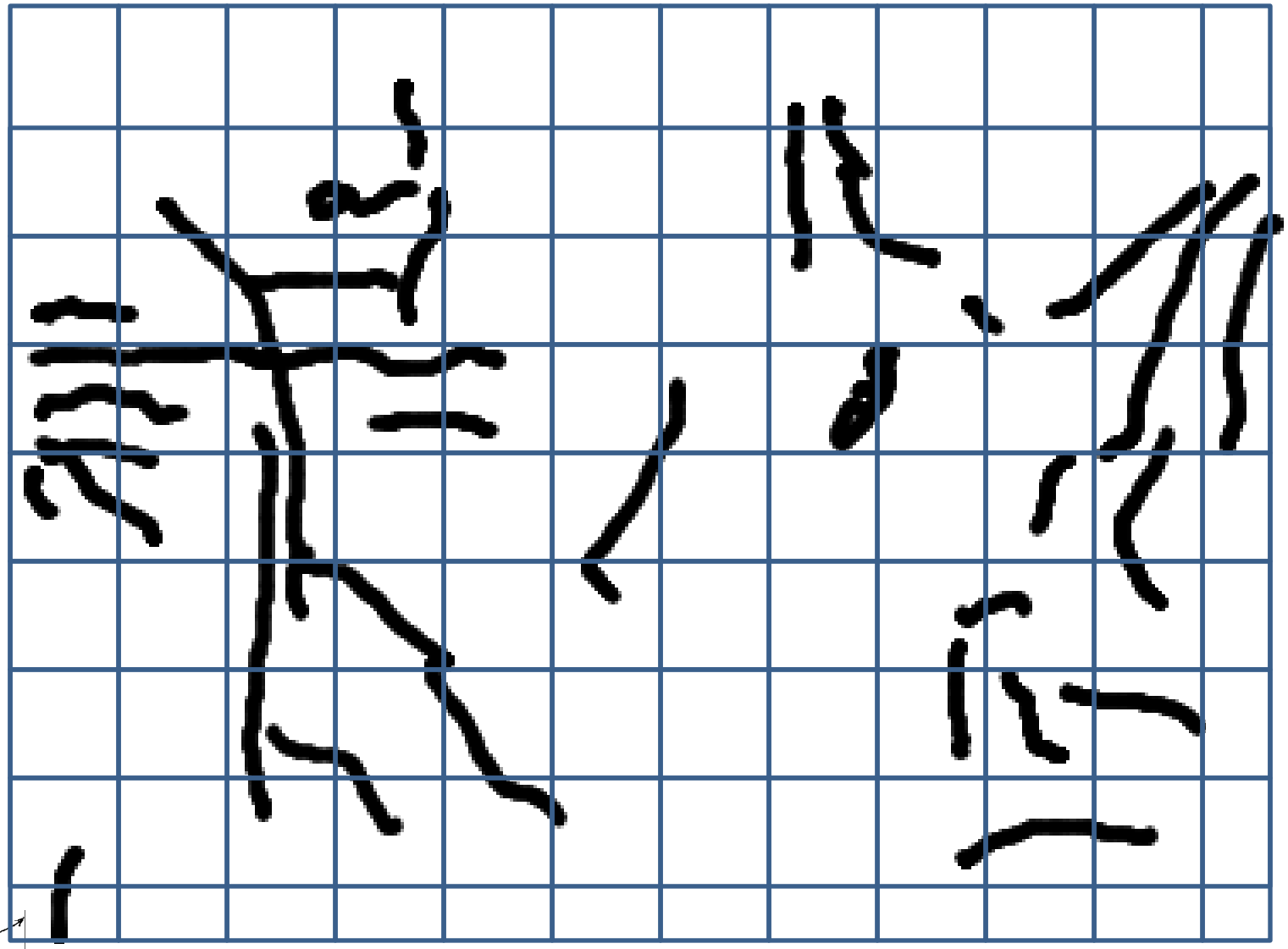
Security Camera Image

Tracked Trajectories

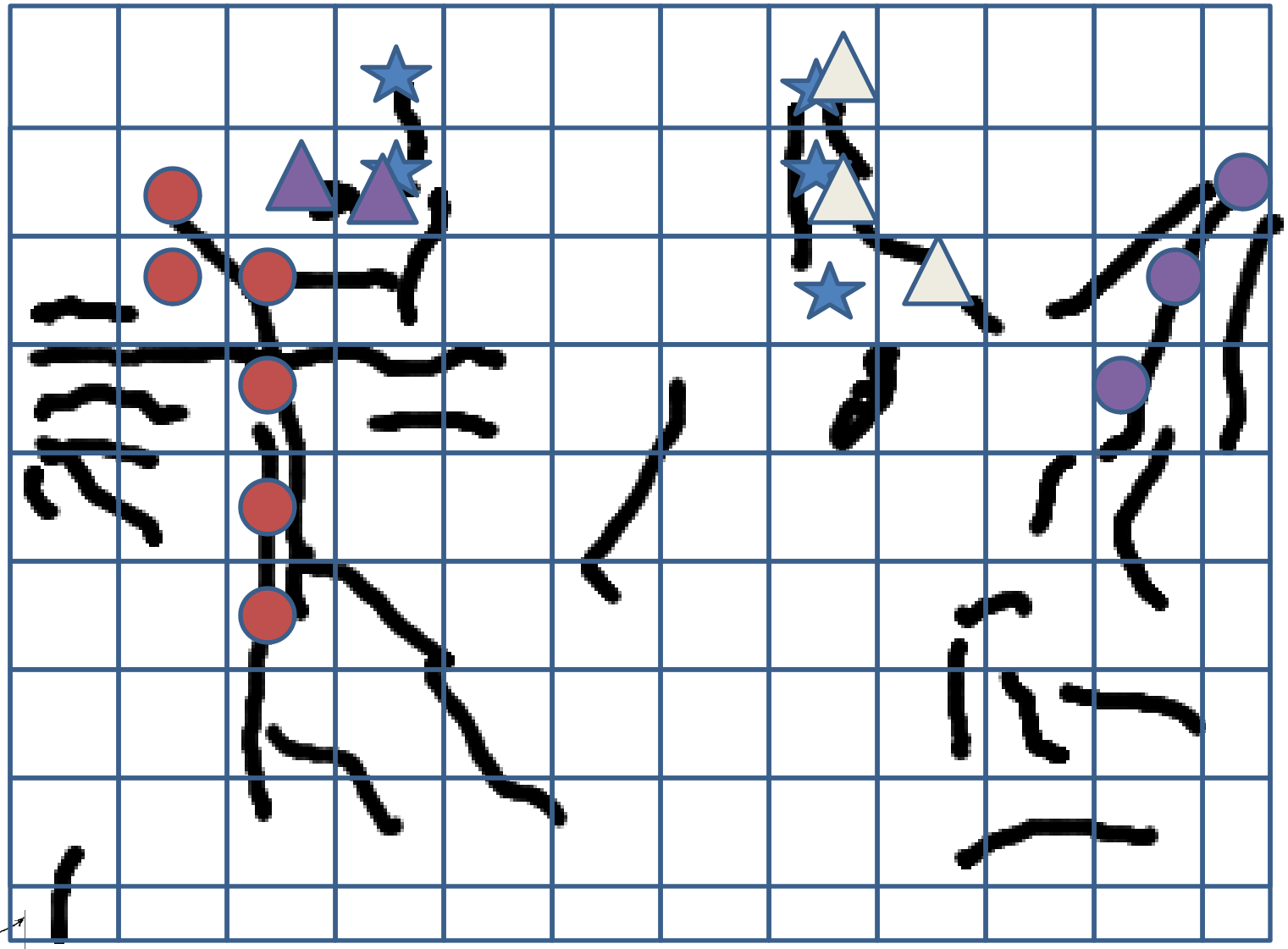
Discretization (e.g., using grid)



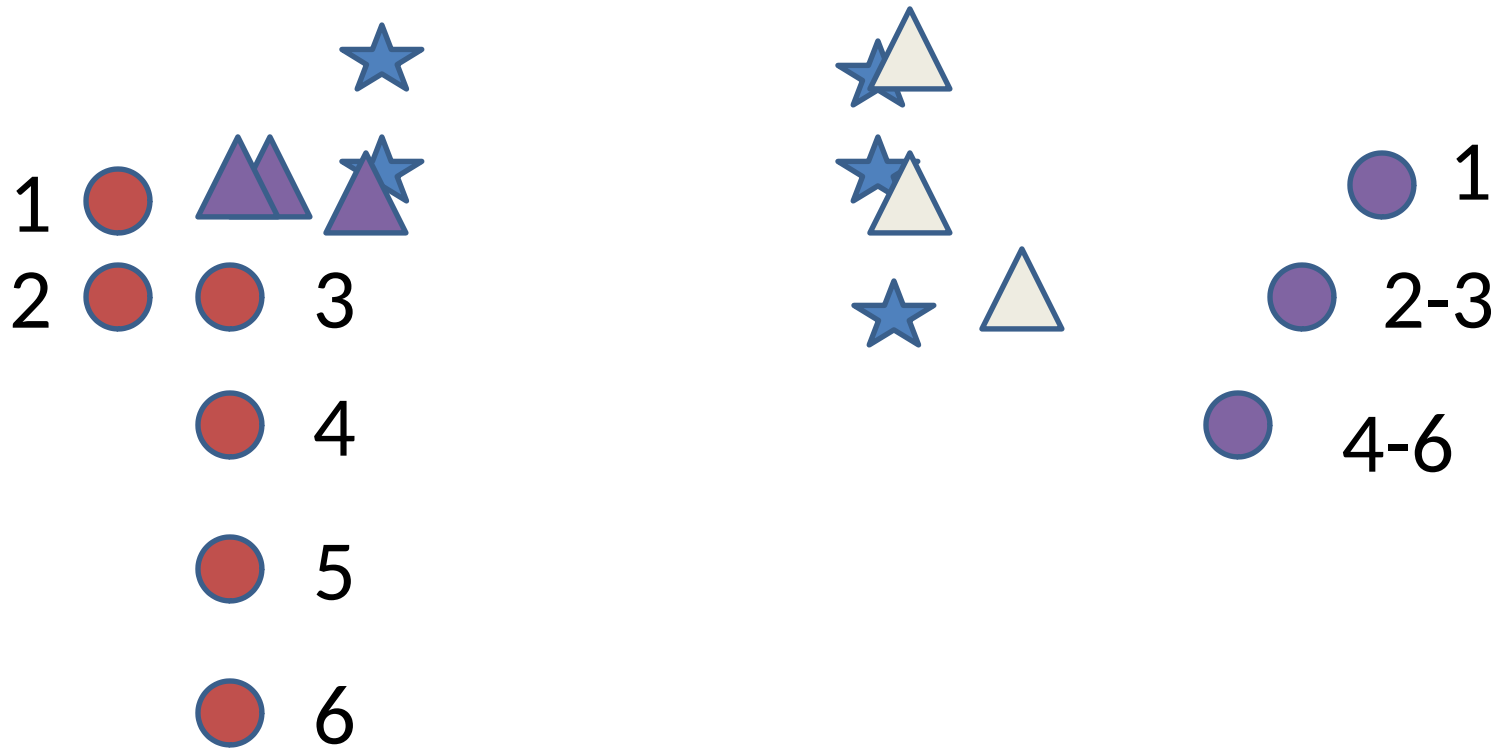
Discretization (e.g., using grid)



Discretization (e.g., using grid)



Resulting Observations



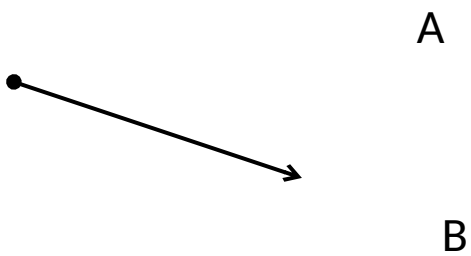
Early Commitment to Discretization

<u>Theorem **1**:</u>

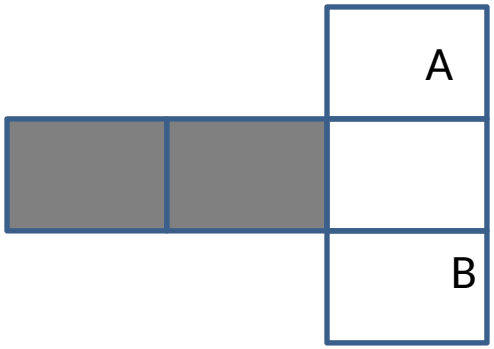
For any discretization, can find case where it fails

What is the goal of the agent?

A or B?



Continuous Version



Discrete Version

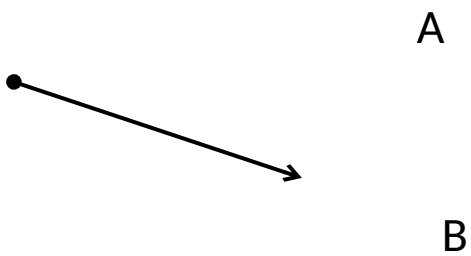
Late Commitment to Discretization

<u>Theorem **2**:</u>

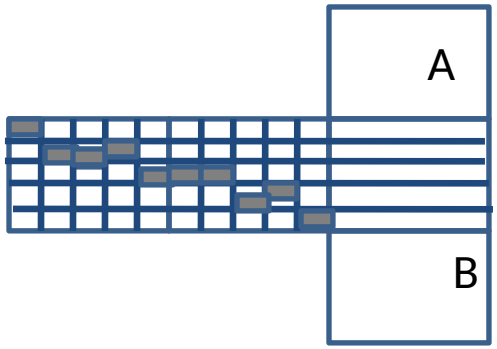
For any case, can find discretization where it succeeds

What is the goal of the agent?

A or B?



Continuous Version



Discrete Version

Challenge:
Allow Late Commitment to Discretization

- Plan recognition libraries <u>require</u> early commitment
  - Hierarchical structures, grammars, HMMs and variants

- Plan recognition by planning (PRP) has potential
  - Generates hypotheses ad-hoc, <u>after</u> getting observations
  - Expensive for online recognition

PRP in Continuous Domains

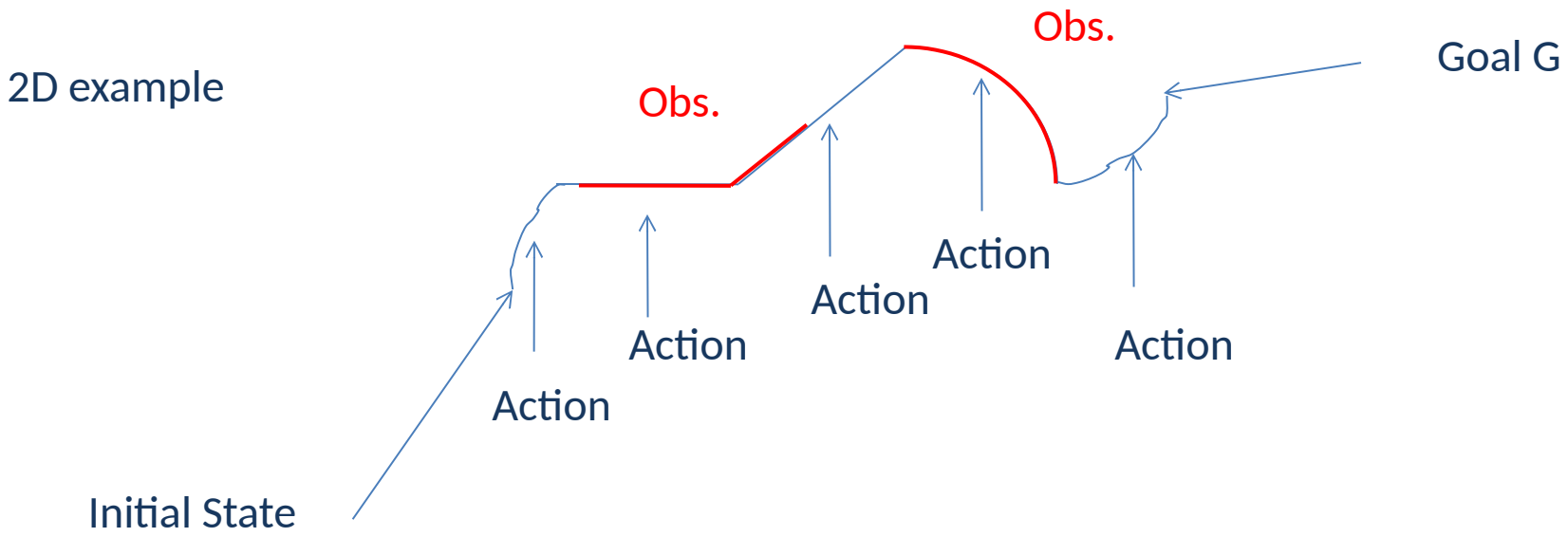
- Observations are of continuous actions?
  - Requires domain theory describing continuous domains
- Requires planner that can work in continuous domains
  - Lots of these in OMPL (Open Motion Planning Lib)
  - **But cannot compute path that "deviates from O"**

Mirroring [IJCAI 17, AAAI 18]

- Revised procedure for ranking hypotheses
  - Optimal plan (plan(G)) vs observed plan (plan(G+O))
  - Closer to formulation in [R&G 2009] (abandoned?)
- Generalize planning domain theories
  - Actions generate trajectories, not single final state
- Online recognition
  - 2T planner calls, each goal ✉ (T+1) planner calls each
  - Heuristics can improve performance (guaranteed)

Plan Recognition Problems

- Given $R = \langle W, I, G, O \rangle$
  - W domain; I initial state; G set of possible goals; O observations
- Find plans in W, from I to a goal in G, <u>that match O</u>



Plans in Domain W

<u>Plan: sequence of actions</u>

- States defined using fluents (numerical values allowed)
  - e.g., *on(A,B)=true*, *fuel-remaining(robot)=50.33*, *pos-x(r)=4.5*

- *Actions: <u>trajectories</u> of state-changes*
  - $\delta(s_{BEG},a) = (S_{BEG},\ldots,S_{END})$
  - $S_i = (S_{i-1} \backslash DEL_a(S_i)) \oplus ADD_a(S_i)$
  - Proper generalization of STRIPS actions

Finding good plan hypotheses

- Want: plan hypothesis $\pi_R = \text{argmax } P(\pi|O)$
  - Intuitively – "best matches the observations"

- We want matching that maximizes

$$P(\pi|O) = \beta\, P(O|\pi)\, P(\pi)$$

$$= \beta\, \underbrace{P(O|\pi)}_{\text{Obs., given plan}}\, \underbrace{P(\pi|g)}_{\text{Plan, given goal}}\, \underbrace{P(g)}_{\text{Prior on goal}}$$

Focus: Maximize $P(O|\pi)$, $P(\pi|g)$

How to maximize $P(O|\pi)$, $P(\pi|g)$:

<u>Two principles:</u>

- $P(O|\pi)$ [obs. given plan]: prefer <u>plans matching obs.</u>
  - Minimize *Error*(π, O): Accumulated distance between π and O
  - <u>$P(O|\pi)$</u> = 1 / (1+Error(π, O))

- $P(\pi|g)$ [plan given goal]: <u>prefer optimal plans</u>
  - Assume rationality of observed
  - Higher <u>$P(\pi|g)$</u> when π <u>**closer**</u> to ideal plan π^*
  - Defined as (normalized) ratio between costs of π and π^*

How to maximize $P(O|\pi)$, $P(\pi|g)$: Shortcut

<u>Two principles:</u>

- $P(O|\pi)$ [obs. given plan]: prefer <u>plans matching obs.</u>
  - Minimize *Error*(π, O): Accumulated distance between π and O
  - <u>$P(O|\pi)$</u> = 1 / (1+Error(π, O))

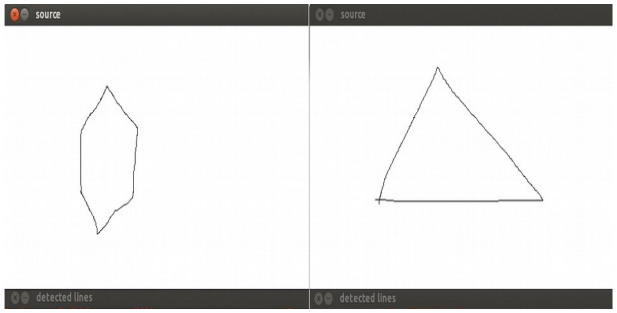
    = 1 when perfect = plan that goes through obs.

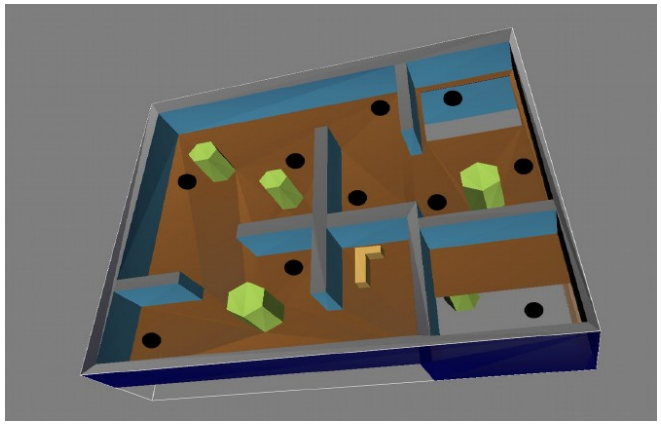
- $P(\pi|g)$ [plan given goal]: <u>prefer optimal plans</u>
  - Assume rationality of observed
  - Higher <u>$P(\pi|g)$</u> when π **<u>closer</u>** to ideal plan π^*
  - Defined as (normalized) ratio between costs of π and π^*

Calls to planner

Domains

- Motion planners in OMPL [Şucan et al. 2012]
- Polygon drawing planner [Vered et al. 2016]
- ROS MoveBase standard navigation package





Summary: Lazy is Good

- Late commitment to discretization: a MUST
  - Remember the theorems (and shown experimentally)
  - Mirroring: a novel form of PRP allowed this
- Late commitment to computing queries: Efficient
- Promising: distinguish queries in PRP
  - Ignore (most of) the past [Masters & Sardina 2017]
  - Pre-computing [Marting et al. 2015, Pereira et al. 2016, 2017]
- Thanks: Friendly organizers and atmosphere at PAIR

galk@cs.biu.ac.il
http://www.cs.biu.ac.il/~galk/