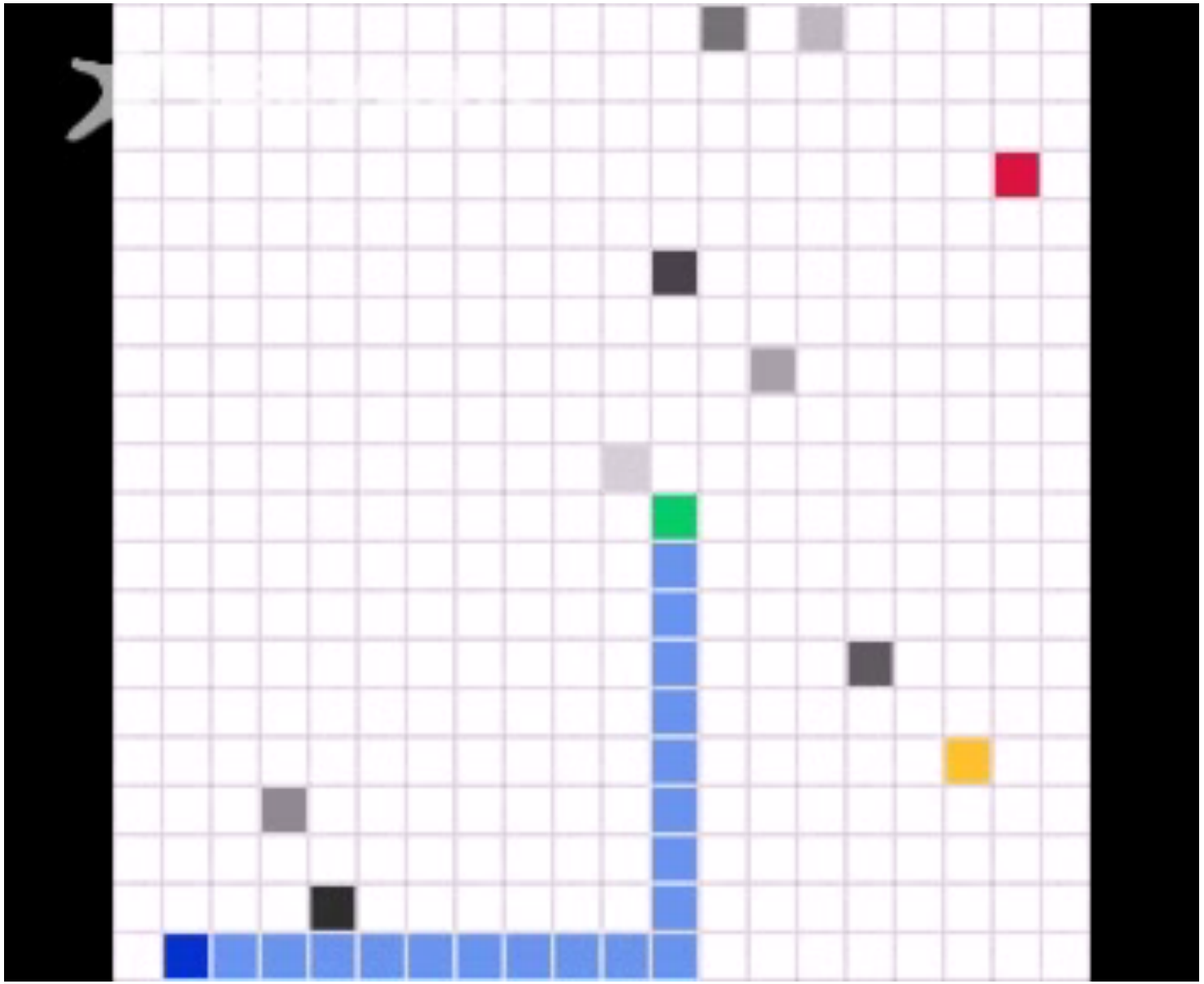
Learning to make Predictions in High Dimensional, Partially Observable Domains

Erik Talvitie

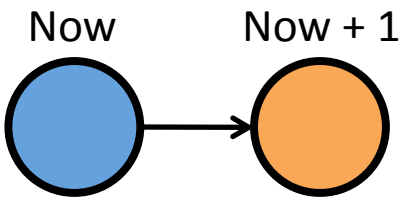
Franklin & Marshall College



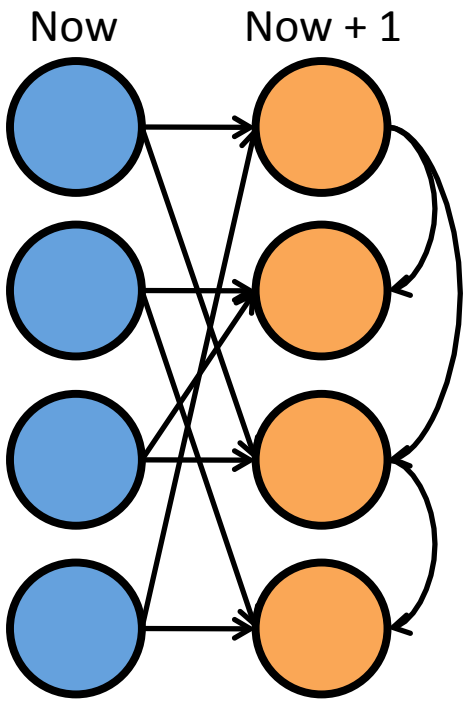
- Requires arbitrarily long-range memory
- Over 10^{30} observations (more underlying states)

Structured Models

MDP

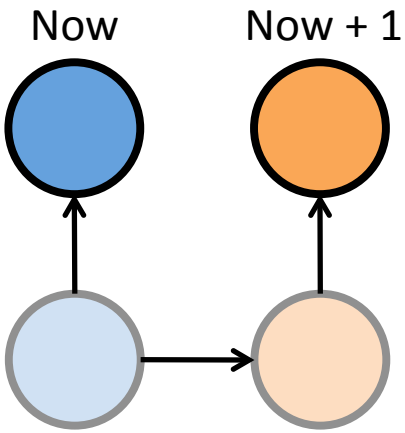


Factored MDP

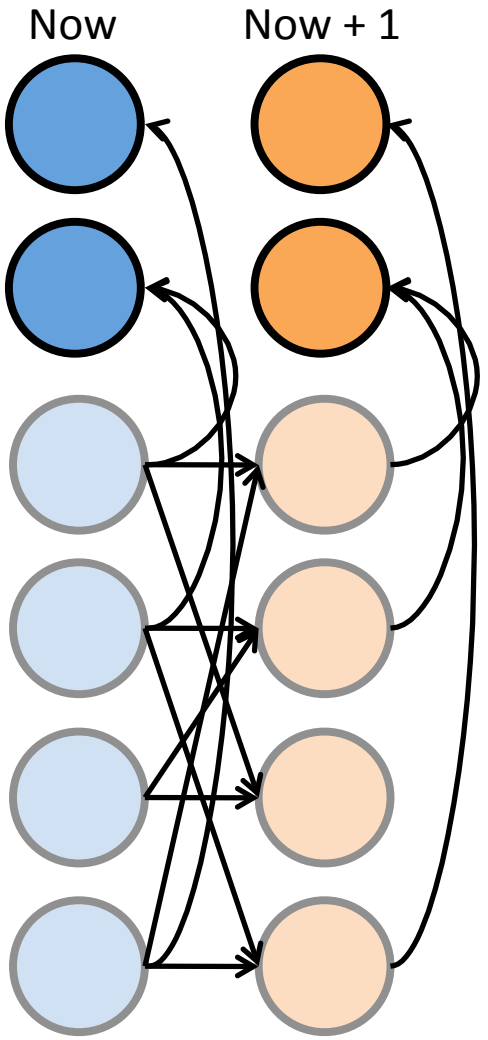


Structured Models

POMDP



Dynamic Bayes Net (DBN)



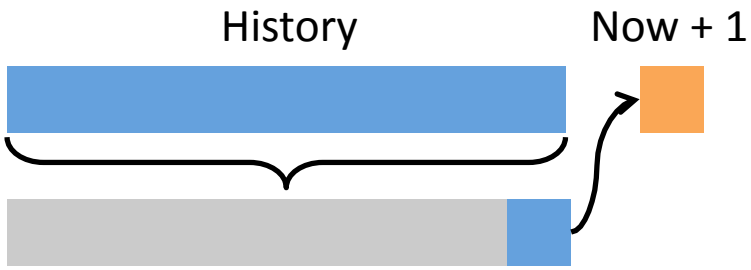
DBN Structure:

Conditional independence of

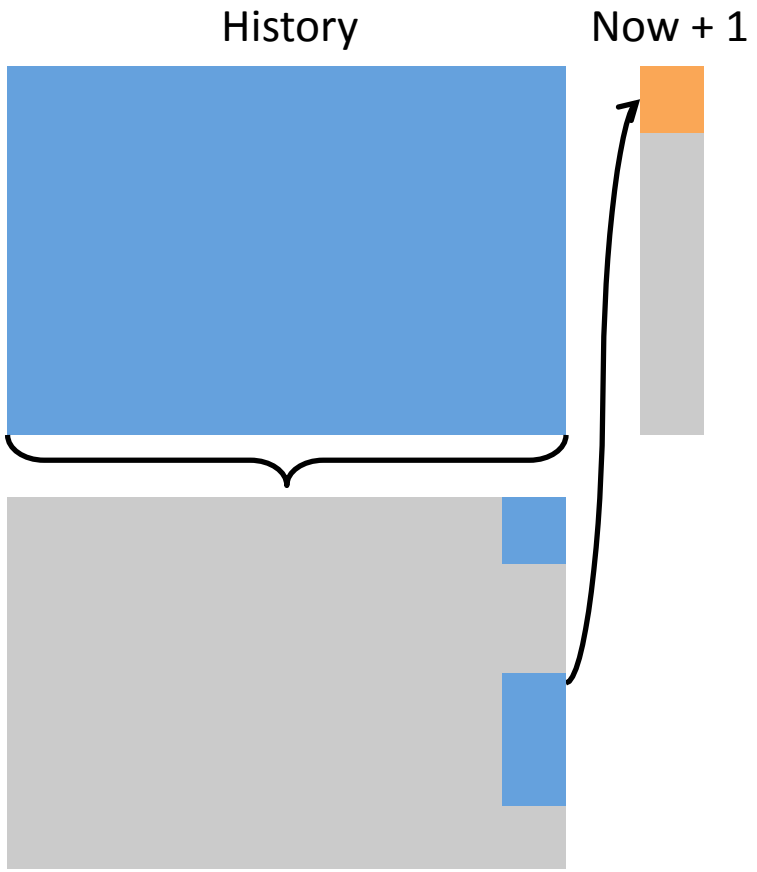
- Hidden variables from one timestep to the next
- Hidden variables and observed variables within a timestep

Structured Models

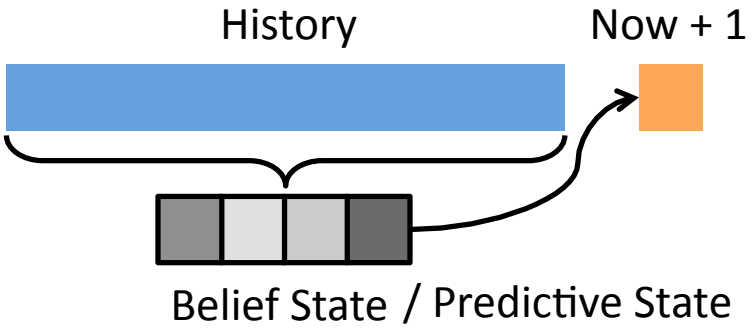
MDP



Factored MDP

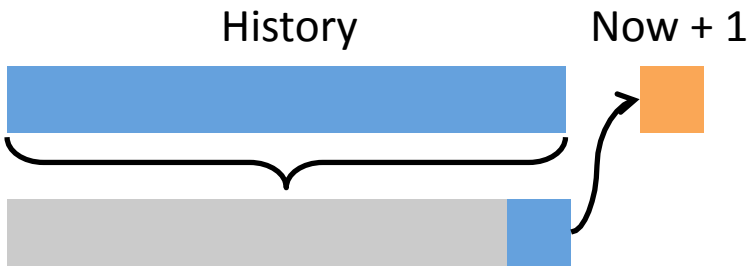


POMDP/PSR

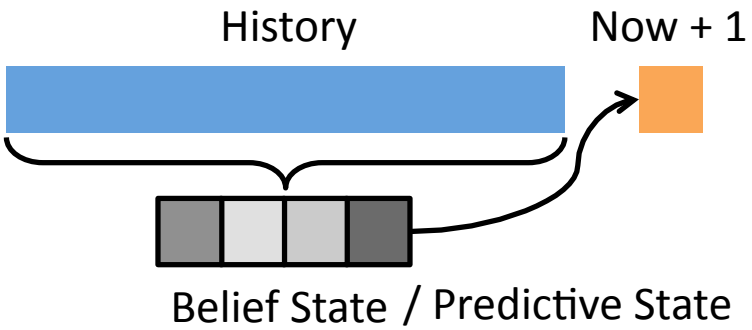


Structured Models

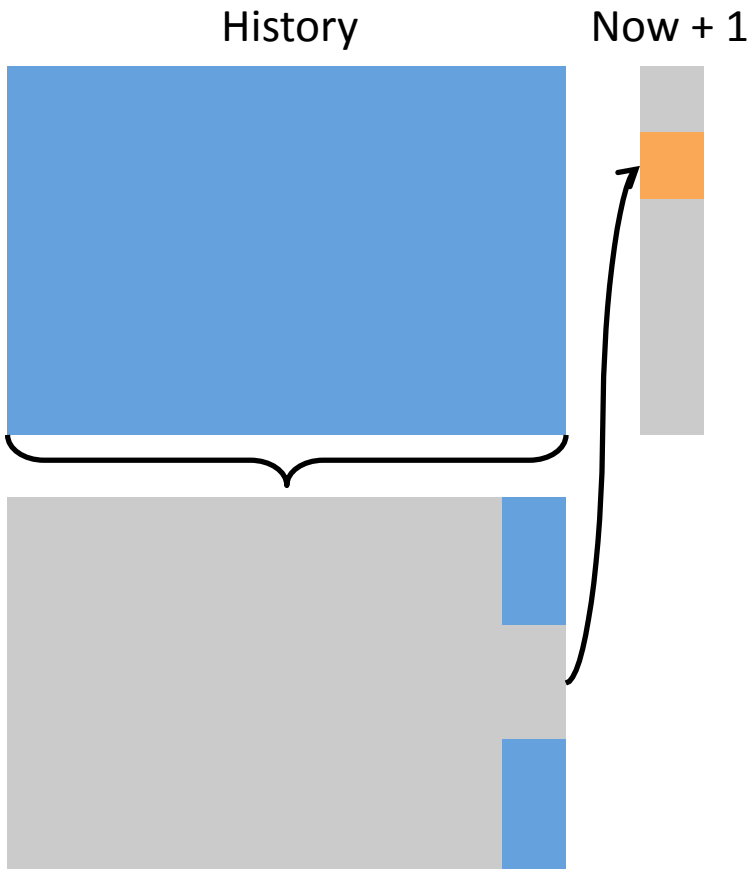
MDP



POMDP/PSR

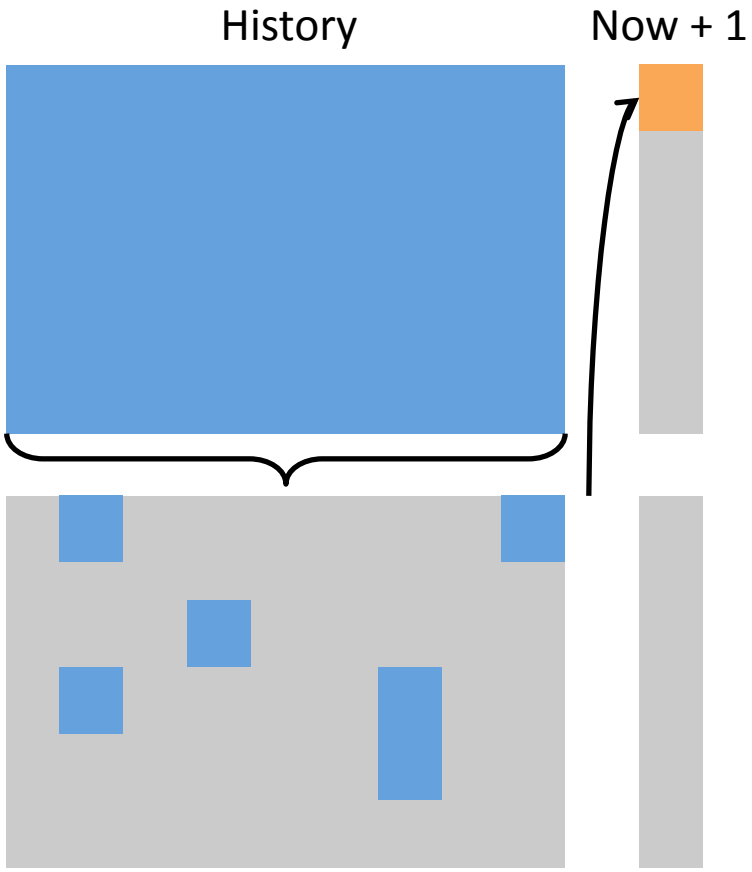


Factored MDP



Structured Models

Collection of Partial Models (CPM)



CPM Structure:

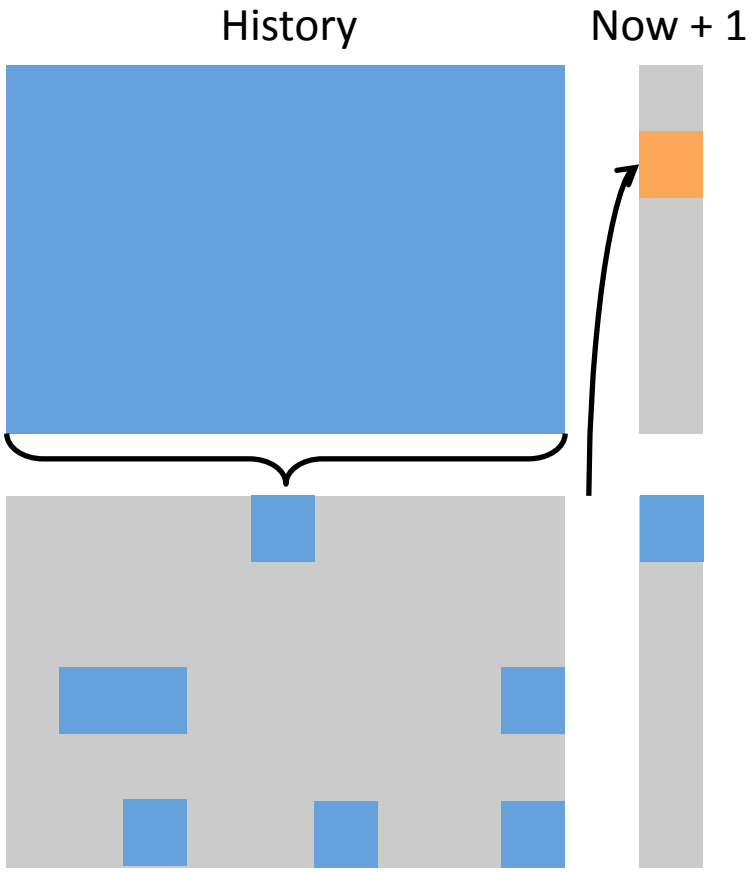
Conditional independence of

- History features and future events
- Future events, given history

(Also: when to apply each model)

Structured Models

Collection of Partial Models (CPM)



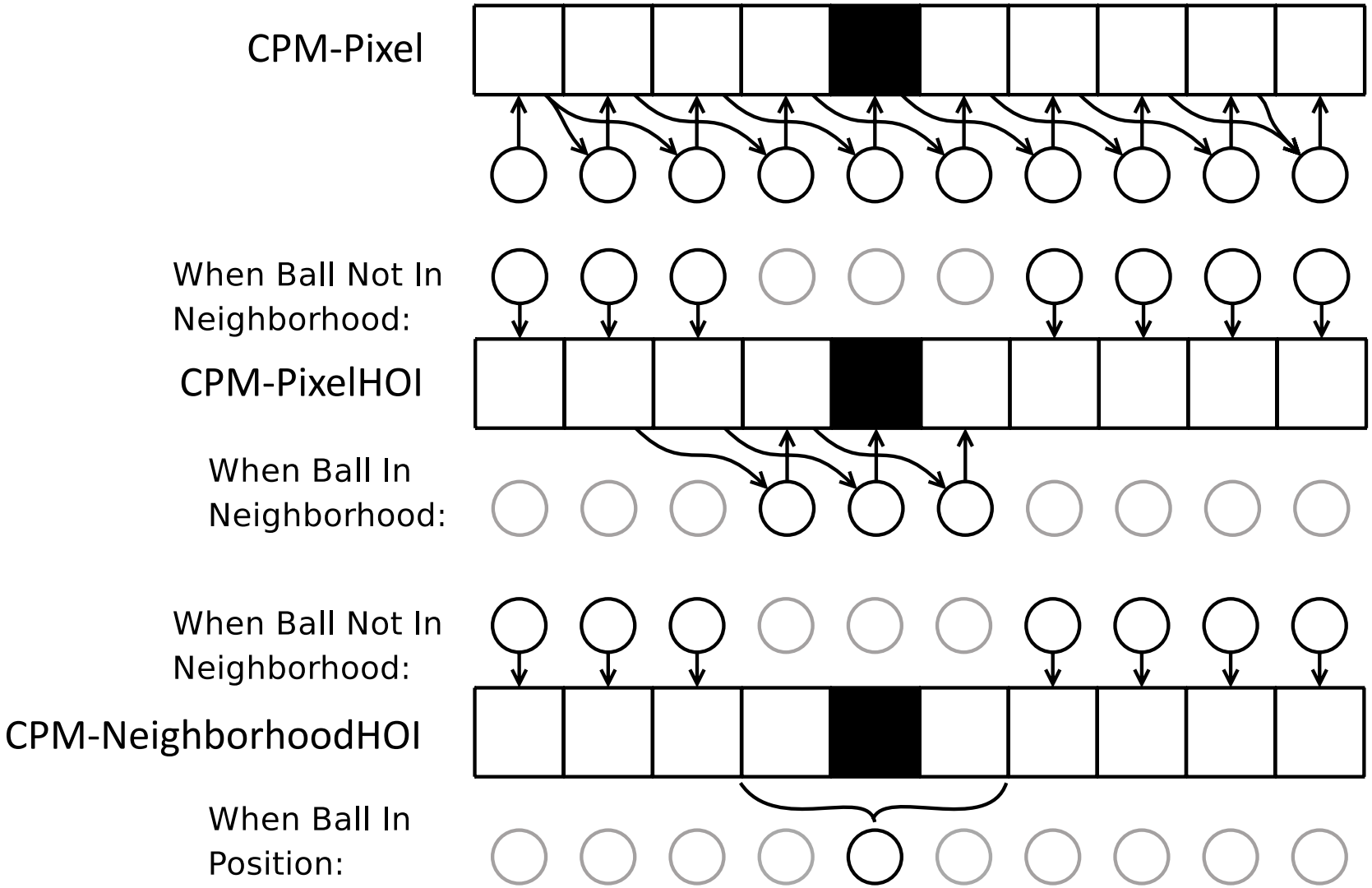
CPM Structure:

Conditional independence of

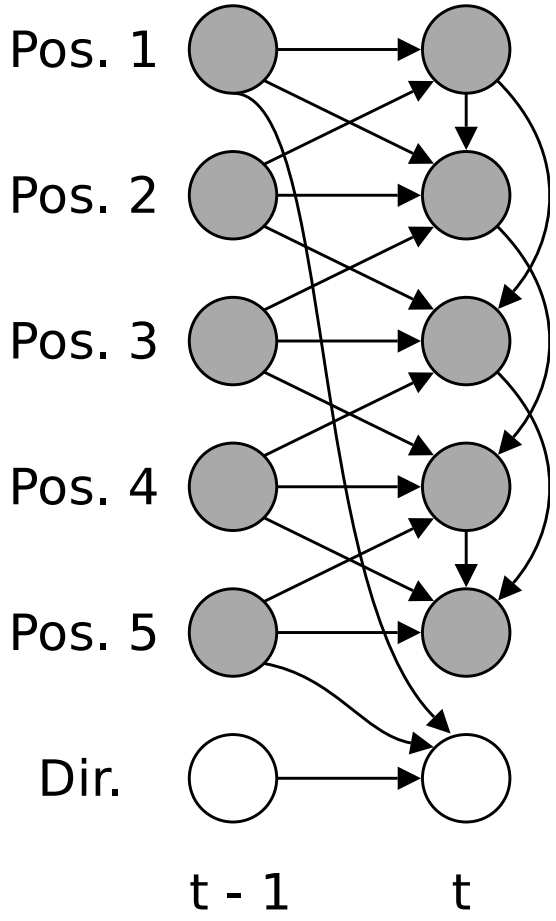
- History features and future events
- Future events, given history

(Also: when to apply each model)

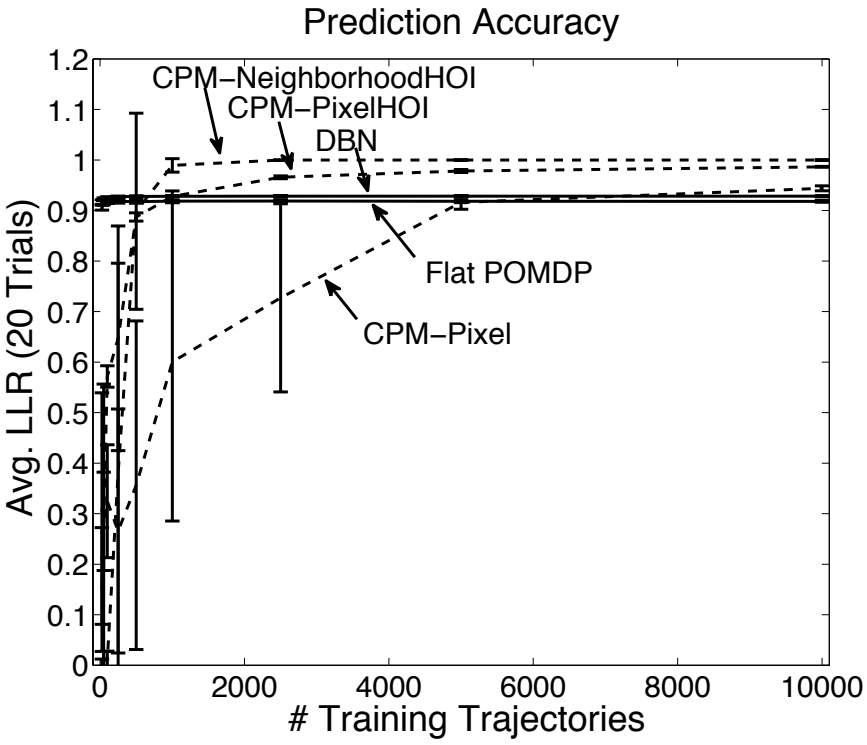
Example: 1D Ball Bounce

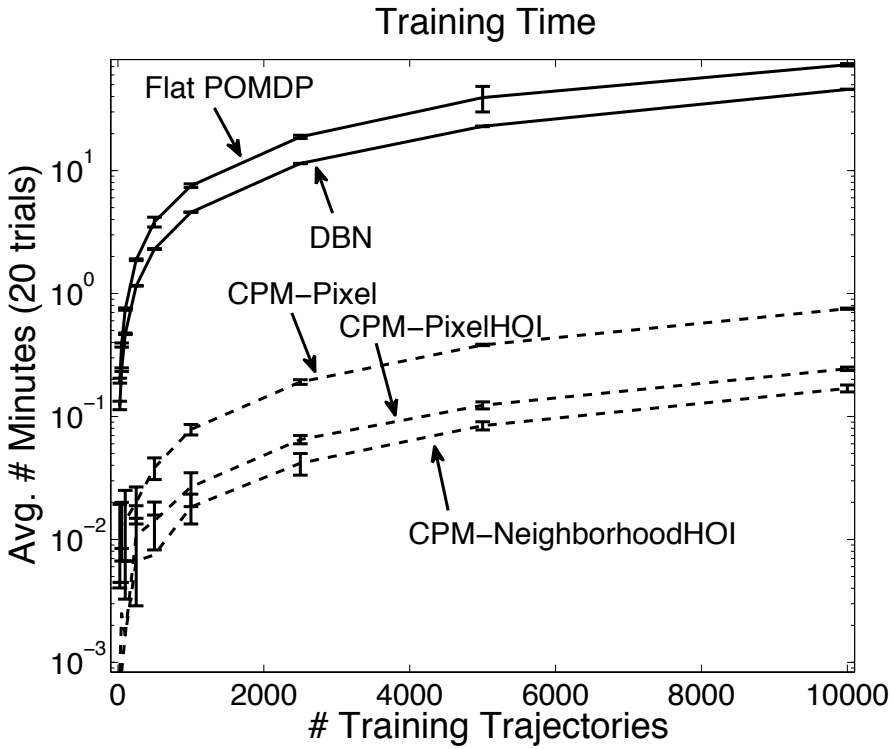


Variant 1 (Simple Hidden State)

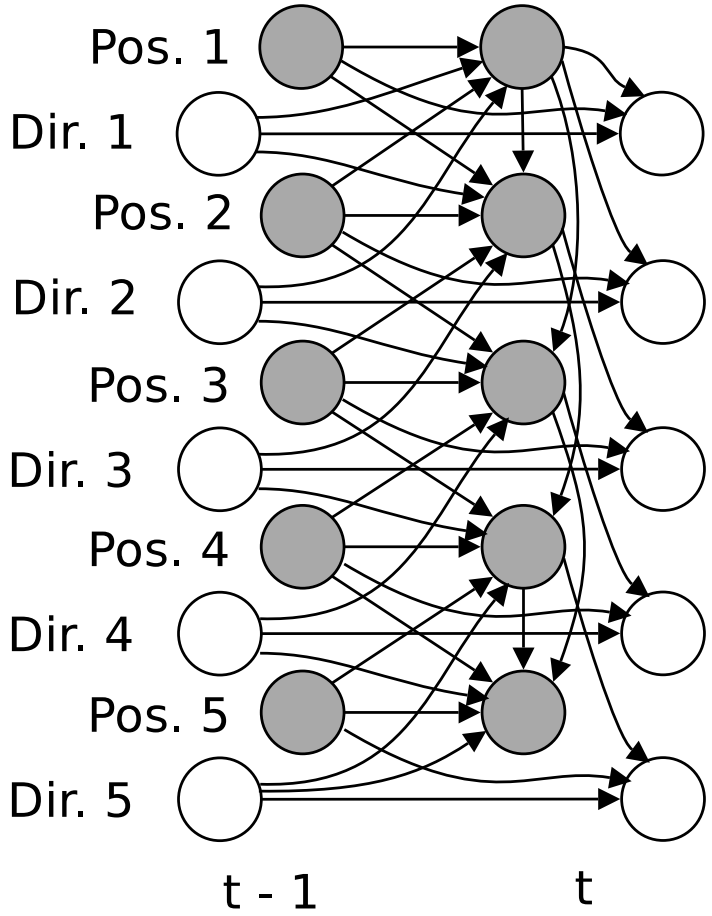


Variant 1 (Simple Hidden State)

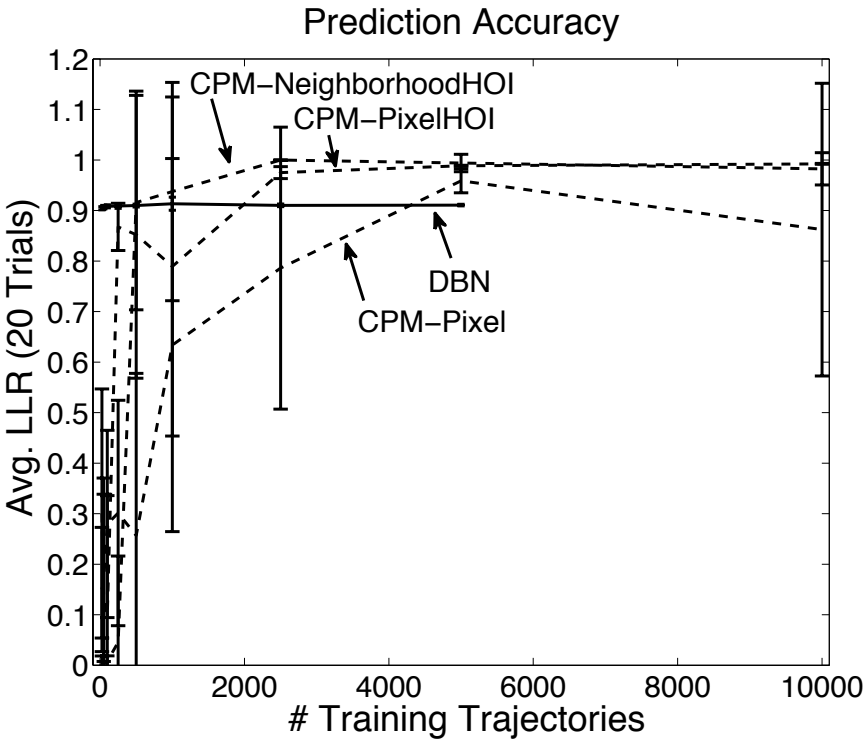


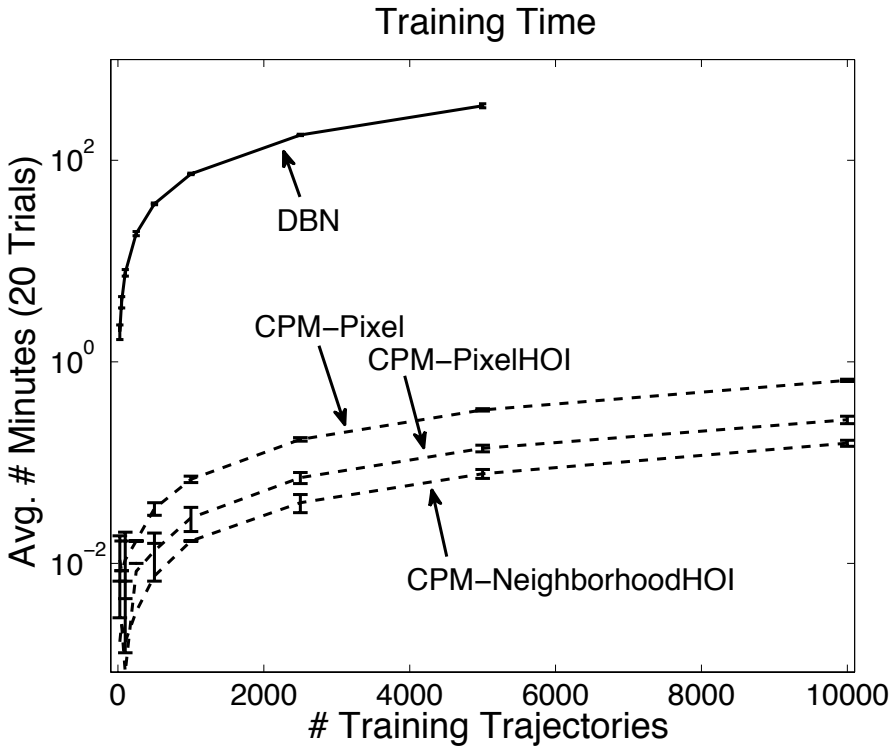


Variant 2 (Complex Hidden State)



Variant 2 (Complex Hidden State)





CPMs and DBNs

- DBN training (EM) requires joint inference over all state variables
  - Partial models in a CPM are trained separately
  - (May result in duplicated effort)
- EM suffers from local maxima
  - Partial models need not be based on hidden variables (more consistent learning algorithms)
- DBN structure is unverifiable
  - Structural assertions of CPM can be tested
  - CPM structure is more learnable (?)
  - DBNs may capture some structure more easily

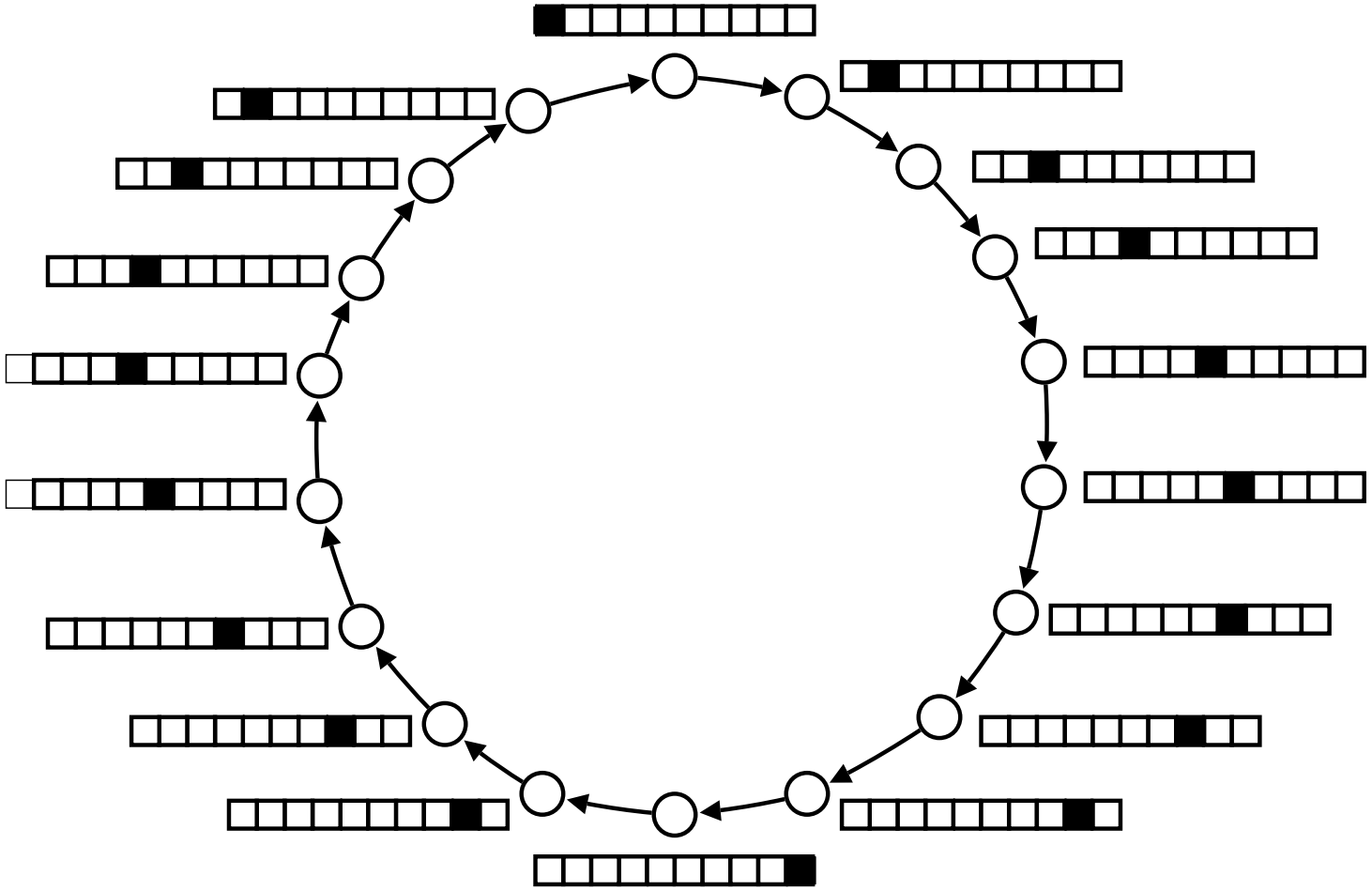
Learning Partial Models

- The partial model learning problem
  - Given:
    - a feature of the future, *F*, to predict
    - and possibly a feature of the future, *C*, to condition on
      - e.g. value of pixels to the left
  - Learn a function $\phi(h) = \Pr(F \mid \mathrm{H} = h, C)$ from histories to (conditional) predictions about *F*.

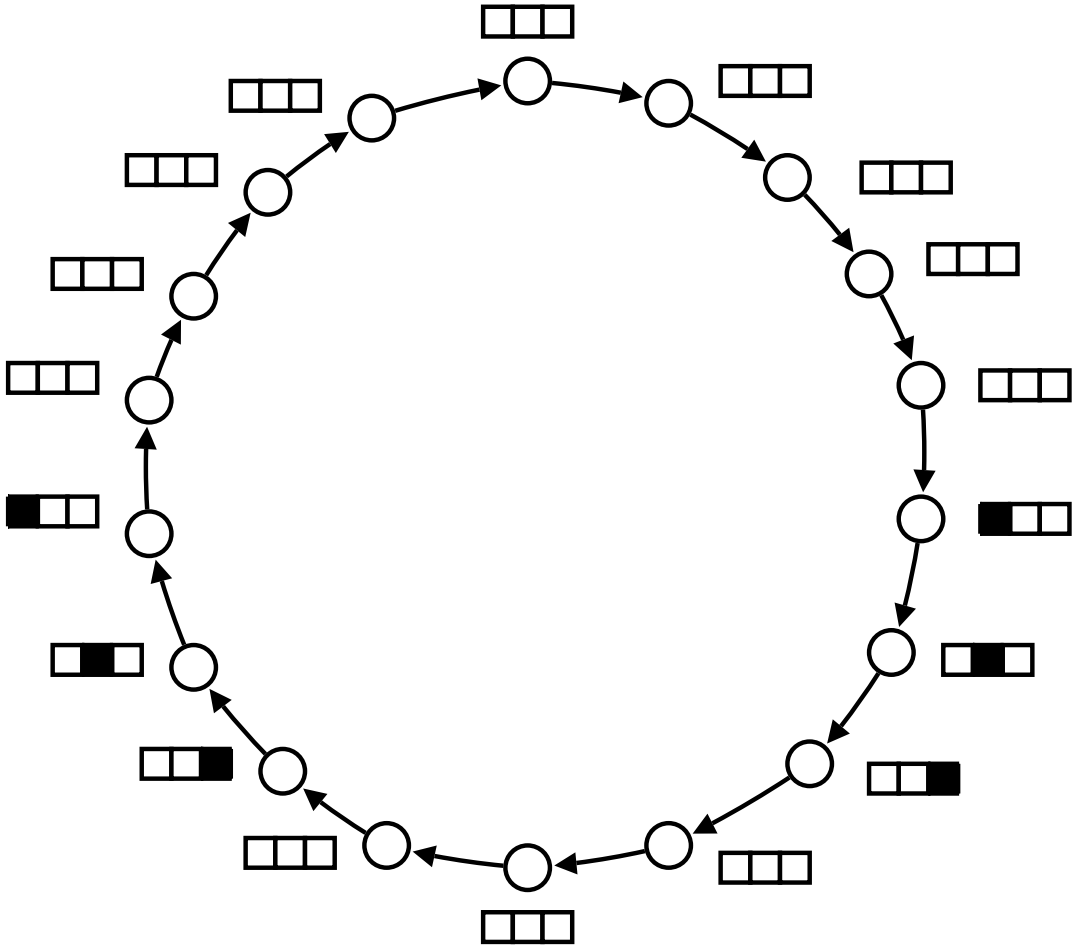
(Why Can't We Use POMDPs/PSRs?)

- We can, but…
  - They are *generative models*
    - Represent a joint distribution of future and past
  - They *can* be used to get a conditional distribution
    - $\Pr(X \mid Y) = \Pr(X, Y)/\Pr(Y)$
  - But anything you want to condition on, you must also predict

(Why Can’t We Use POMDPs/PSRs?)



(Why Can’t We Use POMDPs/PSRs?)

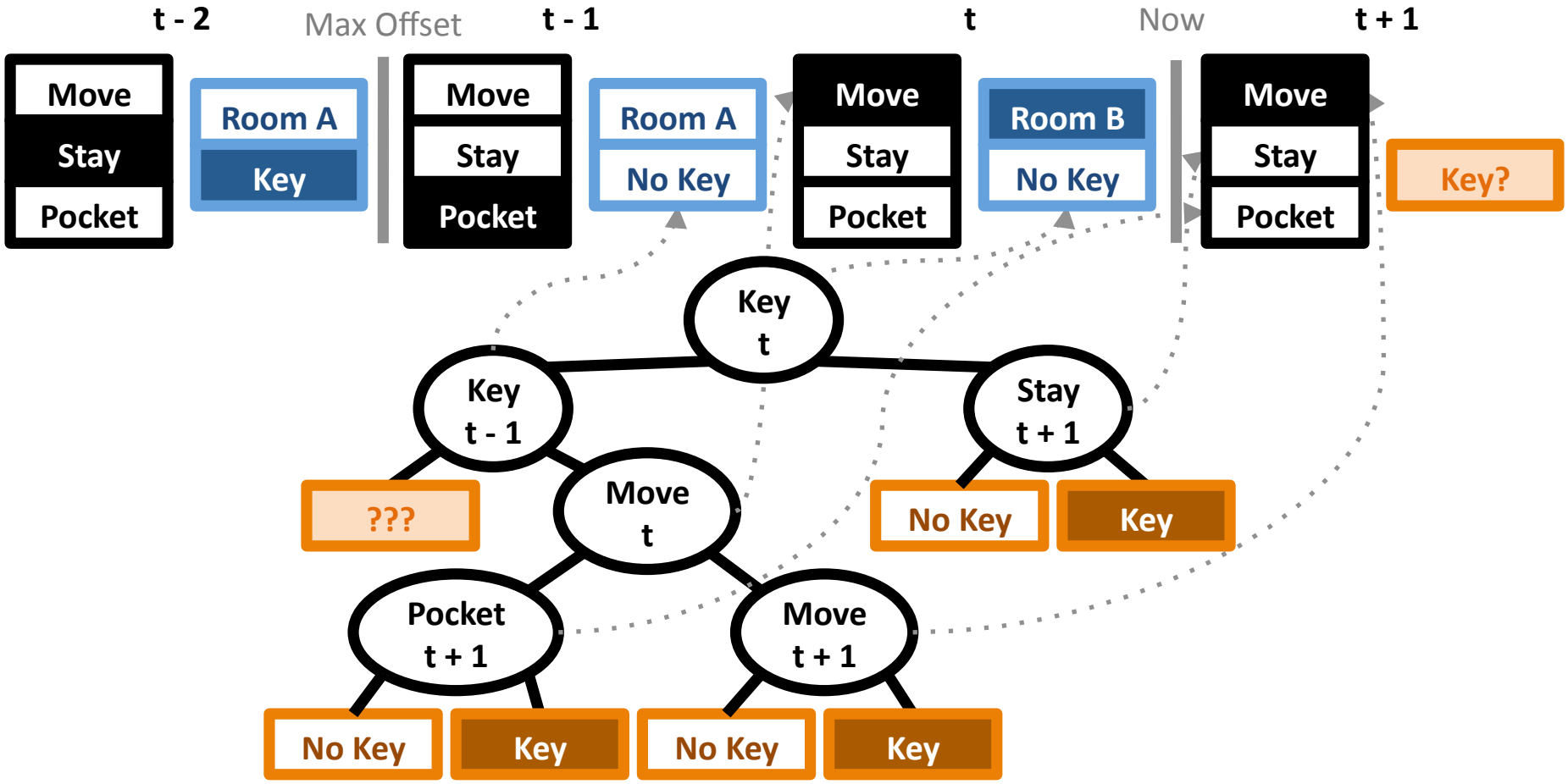


Learning Partial Models

- Some partial model learning methods
  - Utile Distinction Memory (McCallum, ICML 1993)
  - Looping Predictive Suffix Trees: (Holmes and Isbell, ICML 2006)
  - Local Agent State Representations (Dinculescu and Precup, ICML 2010)
  - Deterministic Markov Models (Mahmud, ICML 2010)
  - Prediction Profile Models (Talvitie and Singh, JAIR 2011)
- In principle:
  - Can represent arbitrarily long-term dependencies
- In practice:
  - Assume atomic observations (no observation abstraction)
  - Barriers to discovering long-range dependencies

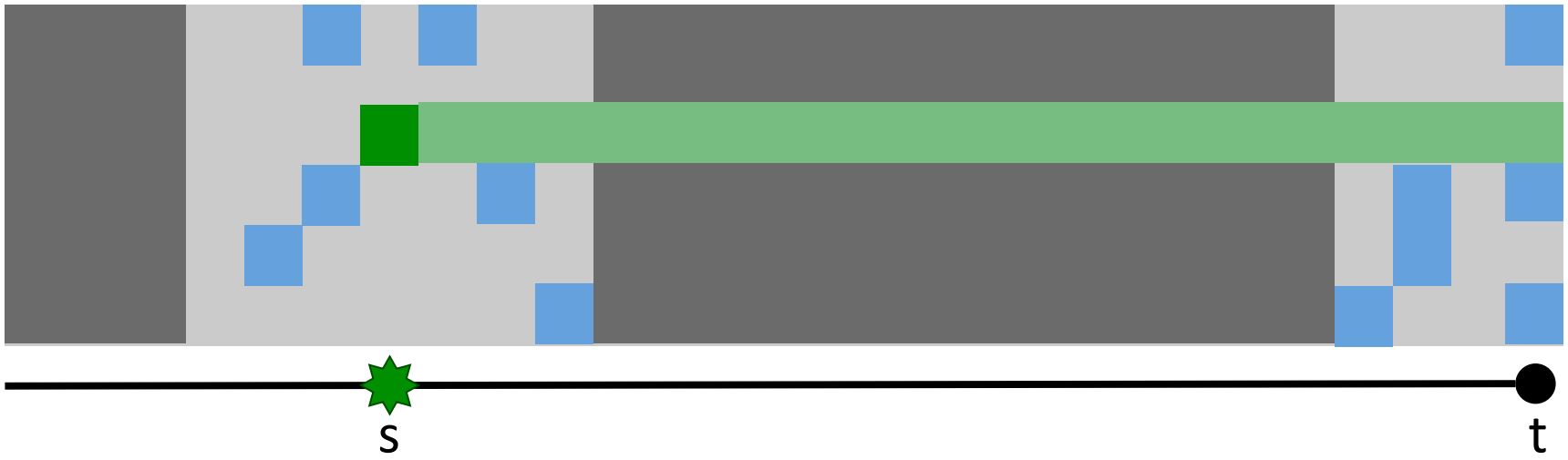
U-Tree (McCallum, 1995)

Where is your room key?



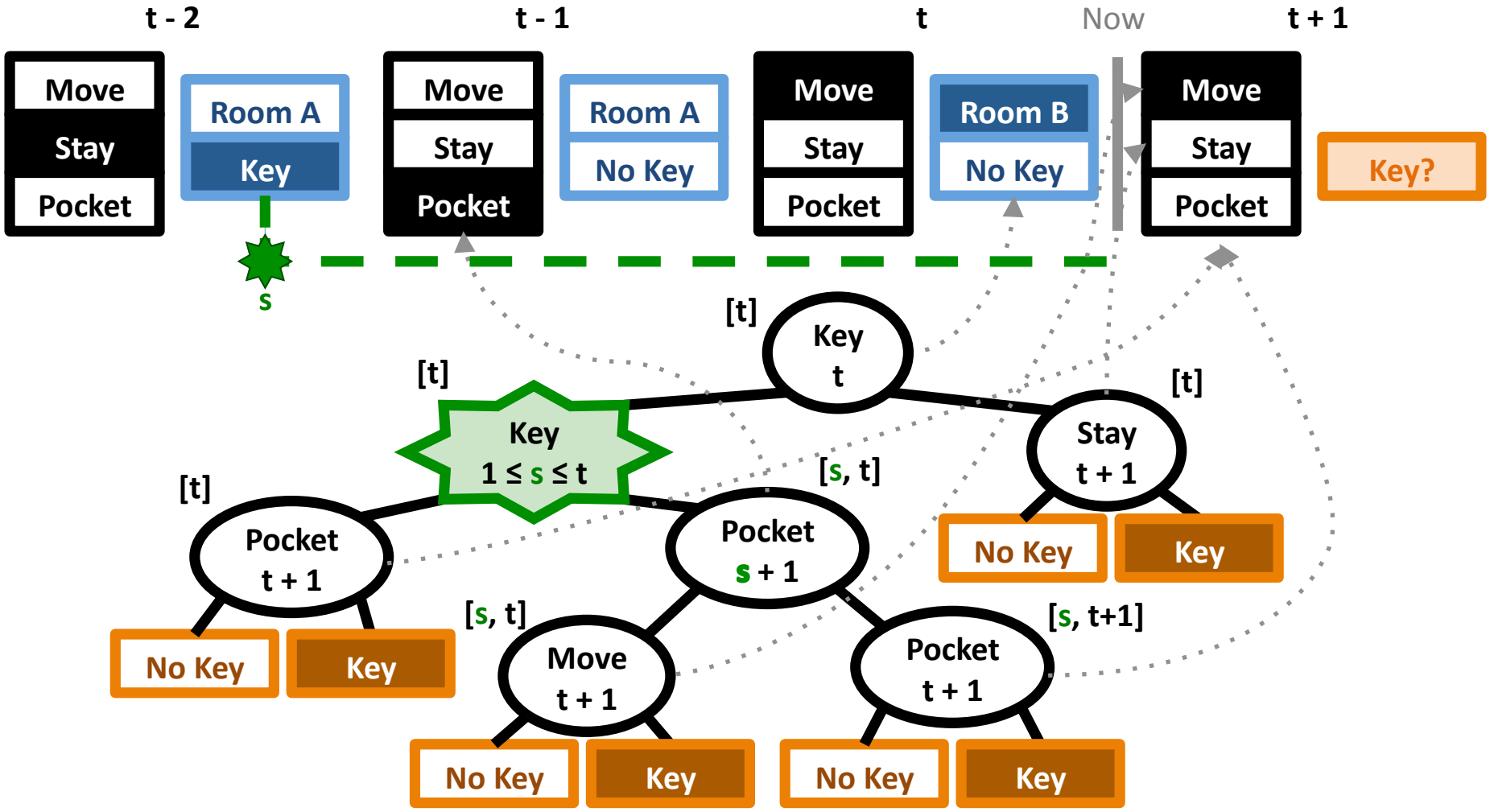
Timeline Trees (NIPS 2012)

- "Now" is a timestamp
- Binary feature *f* can associate with two types of node
  - Offset node: Value of *f* at some offset from a timestamp
  - Timestamp node:
    - Determine if *f* was ever "on" between two timestamps.
    - If so, create a new timestamp at most recent such time.



Timeline Trees (NIPS 2012)

Where is your room key?



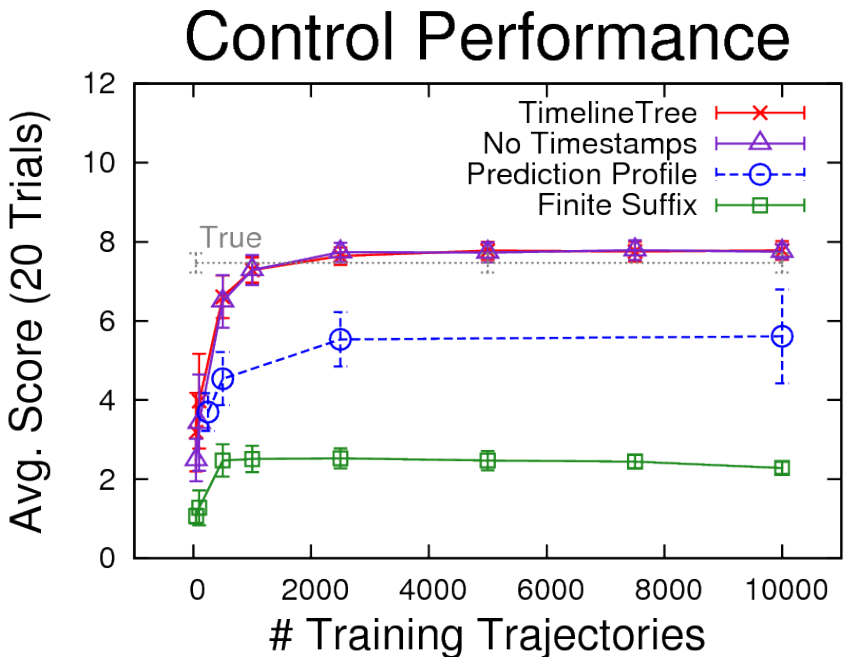
Timeline Trees (NIPS 2012)

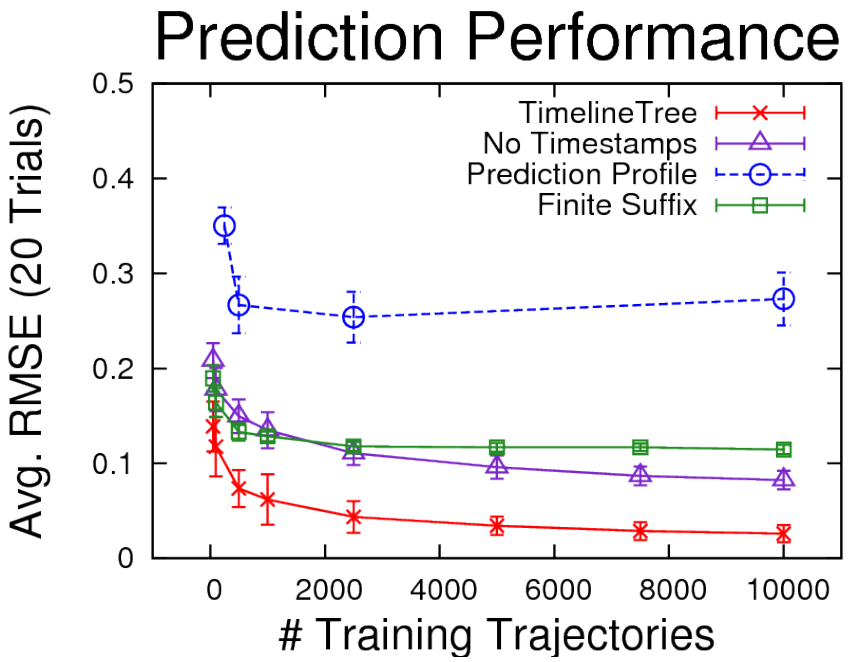
- Can learn to make predictions in high-dimensional, long-horizon systems
  - Learn an abstraction over both observation and time simultaneously
  - Can split on features spread arbitrarily far apart in history (in practice, not just in principle)
- Assume a small number of important events
  - Not always true!

Snake Performance

- ○ Prediction Profile Models
  - Highly informed structure
  - Abstractions learned as a pre-processing step.
- × Timeline Tree
  - Simple structure

- Sanity Checks
  - □ Finite Suffix: No timestamp features (similar to U-Tree)
  - △ No Timestamps: New timestamps are not created





CPMs for Planning

- Learning models means not having to accept the model you are given for planning
- Model learning and planning should be a two-way conversation
  - Planning considerations should inform what models we build
  - What models are learnable should inform what planning asks of its model
  - (Also planning could help to learn the model: exploration etc.)

CPMs for Planning

- Beyond complete models
  - Ignore details irrelevant to task at hand
    - Predict the reward signal
    - Predict the features used in that model
    - …
  - Models with limited roll forward capabilities
    - Maybe it's too hard to get a model that will accurately roll forward forever
    - Allow rollouts to become more abstract as they go deeper (apply more and more abstract models)

CPMs for Planning

- Beyond fine-grained, one-step models
  - Integrate models with many levels of abstraction
    - Model can sample both long and short term futures
    - Abstract predictions could constrain noisy rollouts
      - (e.g. “the snake has only one head”)
  - Incrementally incorporate new questions
    - Planning process may suggest questions the model should answer
    - Model may suggest planning priorities (exploration etc.)

Thanks!