CSE

Efficiently Implementing Episodic Memory

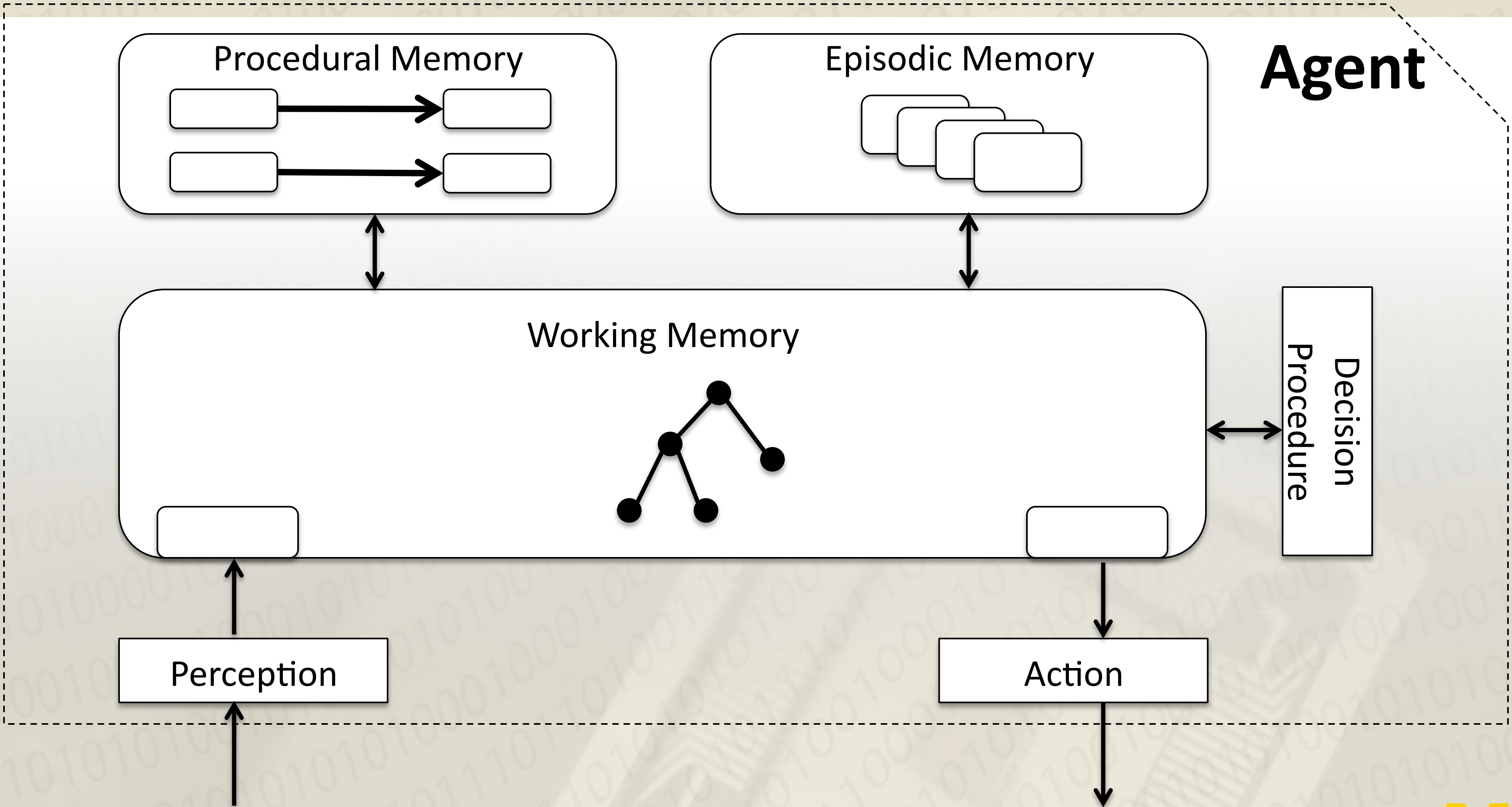
Nate Derbinsky and John E. Laird

University of Michigan

What is Episodic Memory?

- Long-term, contextualized store of specific events
  - Tulving, E.: Elements of Episodic Memory (1983)
- Functionally
  - Architectural
  - Automatic
  - Autonoetic
  - Temporally indexed

Architectural Integration



Comparison to CBR

CBR

<u>Cases</u>

- Contain problems and solutions
- Fields pre-specified

<u>Case Base</u>

- Fixed or slowly growing
- Deliberate updates
- No temporal relation between cases

EpMem

<u>Episodes</u>

- Structure and content reflect agent's experiences
  - Potentially fine-grain

<u>Episodic Store</u>

- Grows with experience
- Architectural & automatic storage
- Temporally structured

The Promise of EpMem

- Virtual Sensing
- Action Modeling
- Retroactive Learning

…

Nuxoll, A.: Enhancing Intelligent Agents with Episodic Memory. (2007)

Efficient Implementation

Goals

- Develop a system that is practical for real-world tasks
- Establish baseline results for graph-based, task-independent EpMem implementations

Assumptions

- Stored episodes do <u>not</u> change over time
- Qualitative Nearest Neighbor (NN) cue matching

Performance Challenges

- Consider a year of episodic memories…
  - 16 hours/day -> 42M to 420M episodes
  - 100 – 1000 features/episode (10-100 bytes/feature)
    - **42GB to 42TB**
    - 2GHz CPU -> **20 seconds/scan**

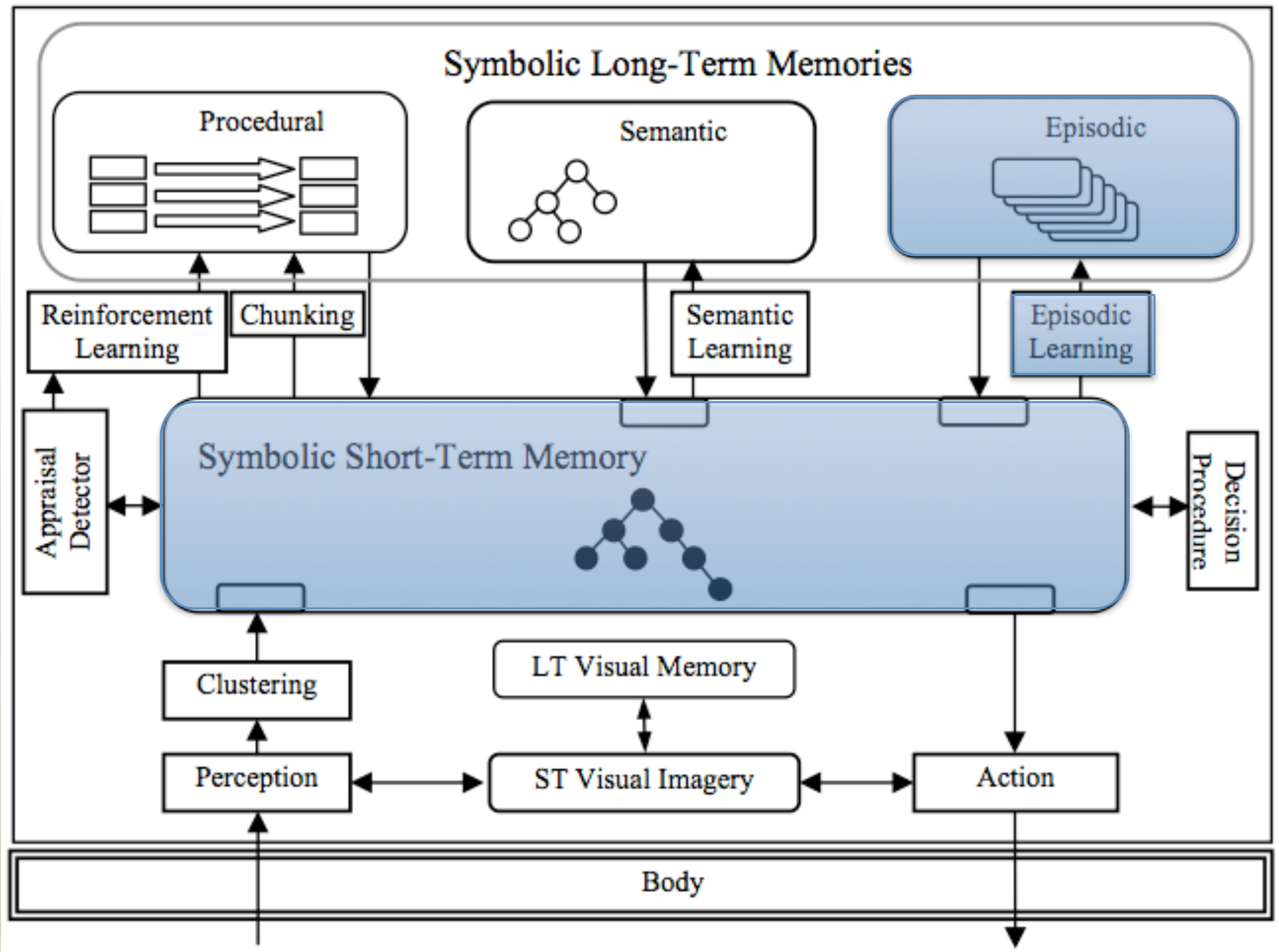
- Agents in real-world, dynamic environments have real-time constraints on reactivity
  - 50-100ms per deliberate decision
  - 20 decisions for utility
  - **1 second per episodic retrieval**

Laird, J.E., Derbinsky, N.: A Year of Episodic Memory (2009)

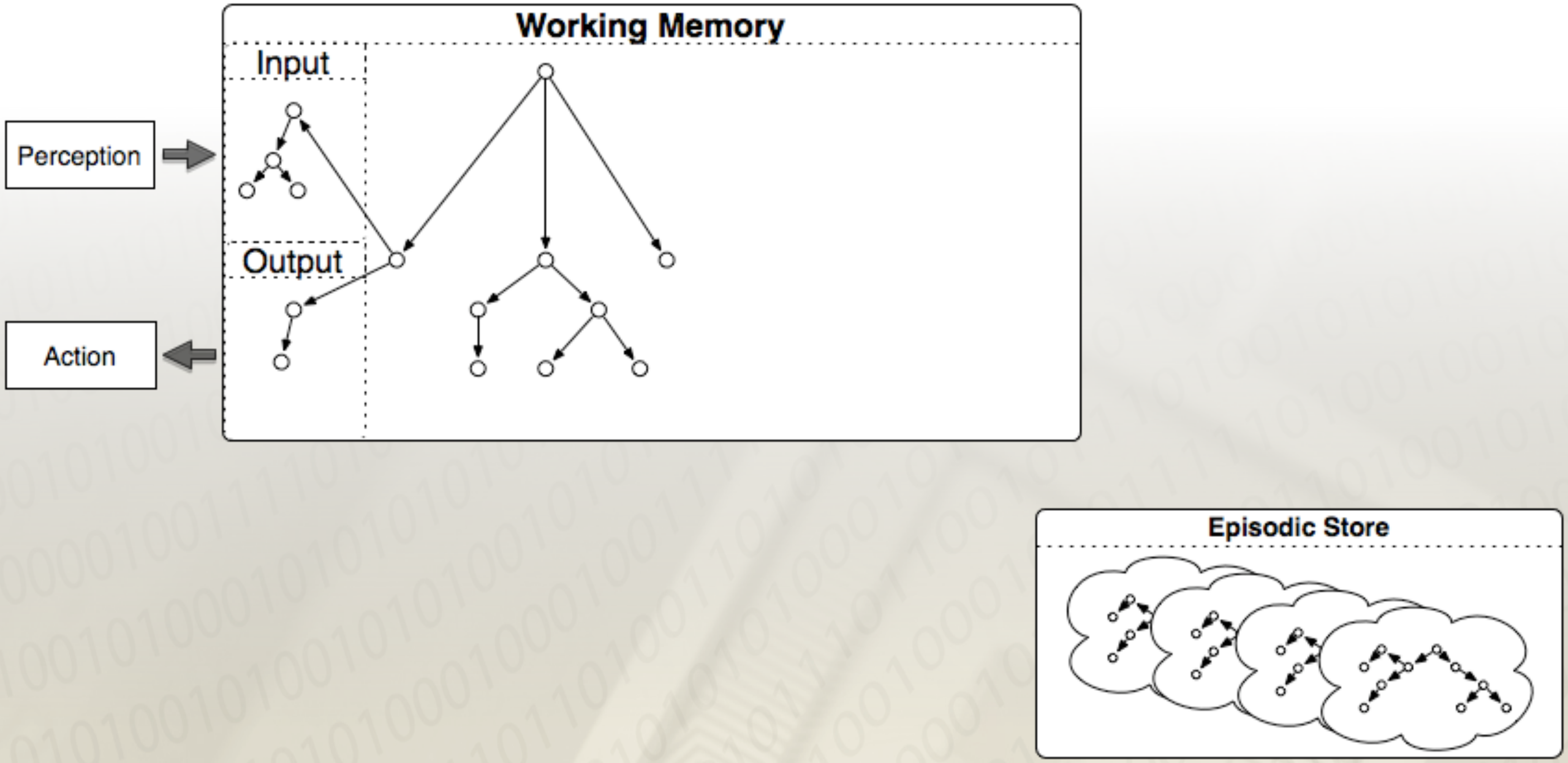
Related CBR Work

- Efficient NN qualitative/quantitative algorithms
- Heuristic retrieval algorithms
  - Refined indexing, storage reduction, case deletion
- Two-stage cue matching
- Temporal CBR
  - Time-dependent case attributes
  - Temporal case sequences

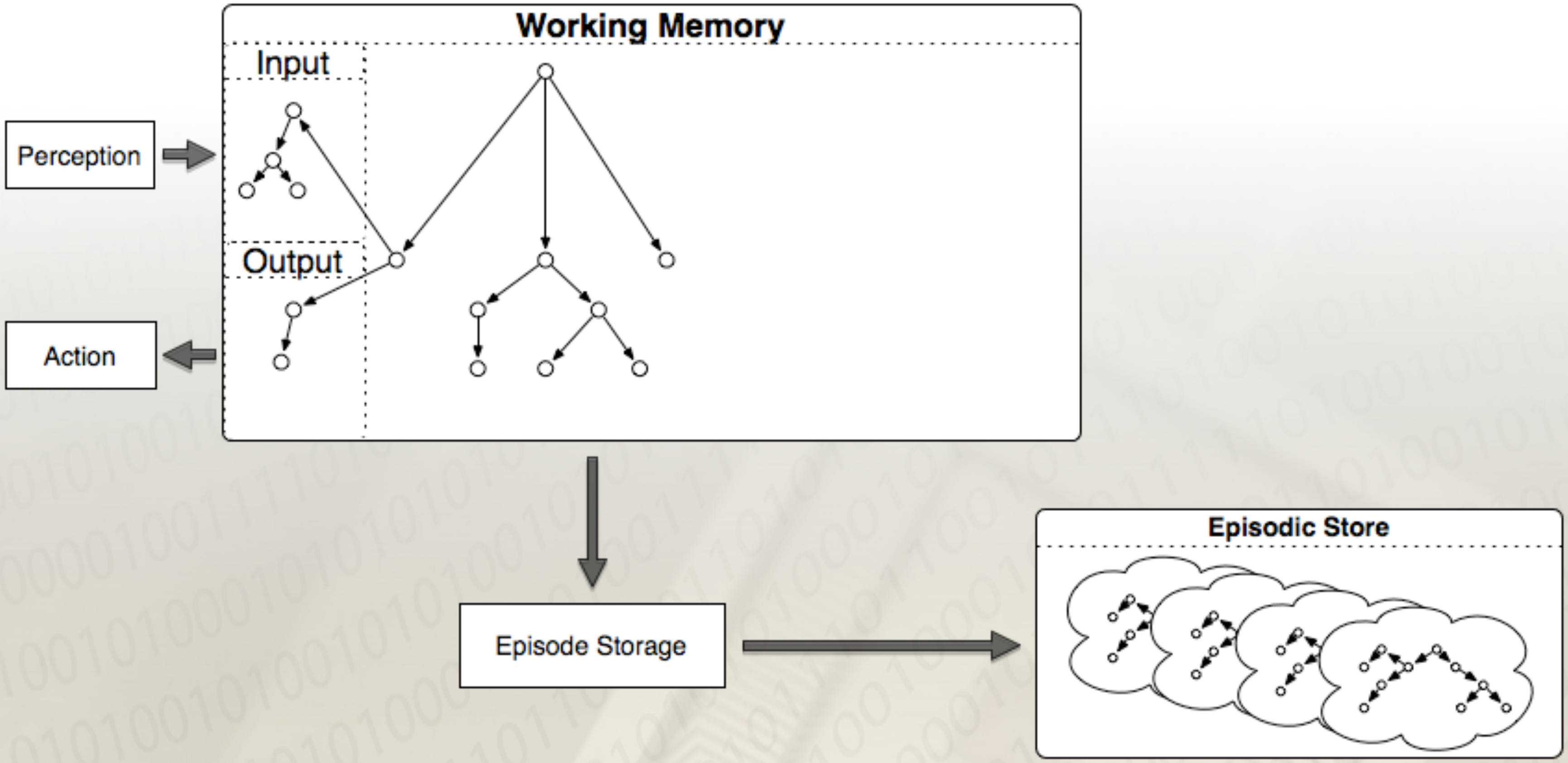
Soar 9


Symbolic Long-Term Memories
Procedural
Semantic
Episodic
Reinforcement Learning
Chunking
Semantic Learning
Episodic Learning
Appraisal Detector
Symbolic Short-Term Memory
Decision Procedure
Clustering
LT Visual Memory
Perception
ST Visual Imagery
Action
Body

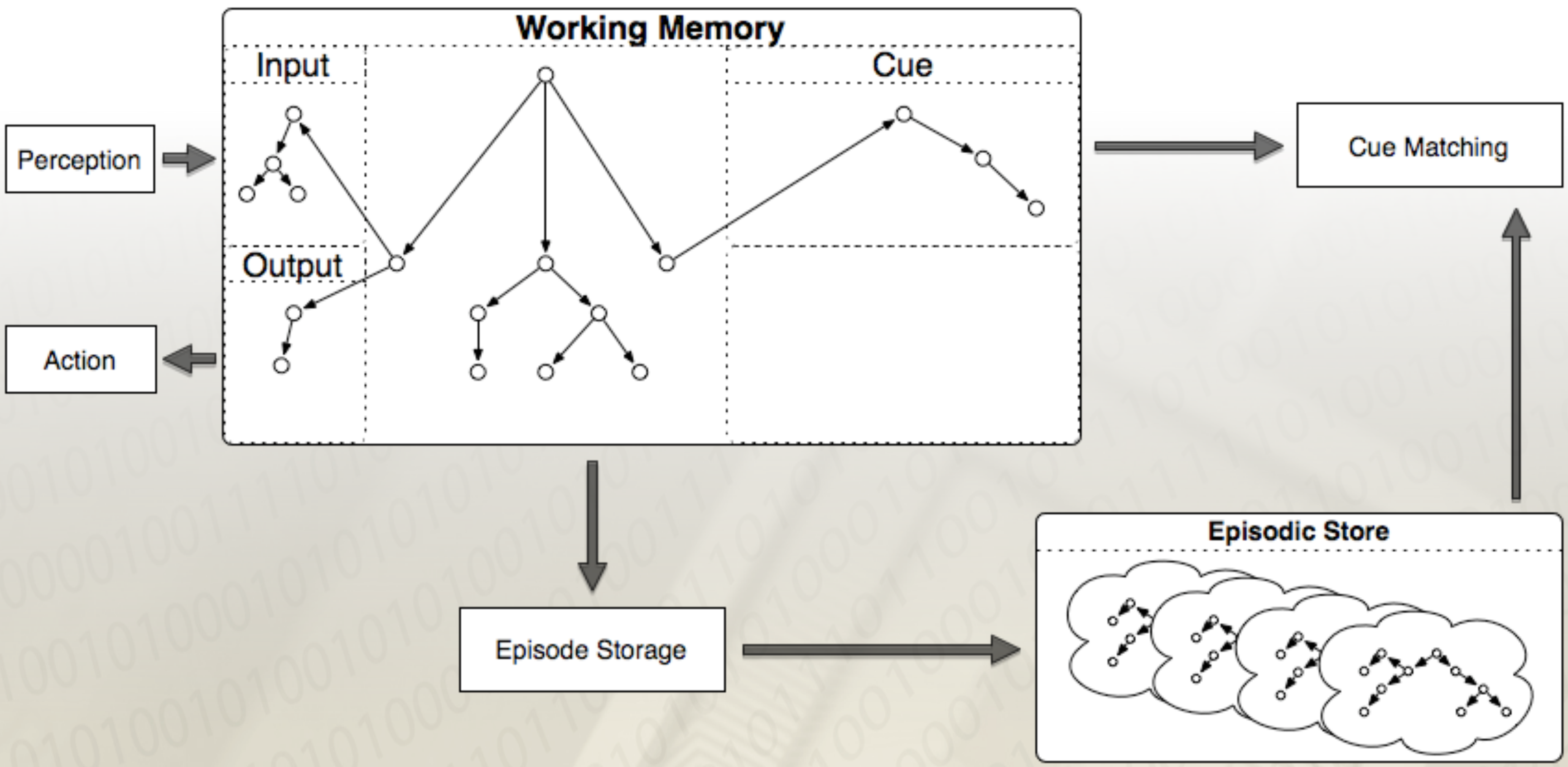
Integrating EpMem with Soar 9



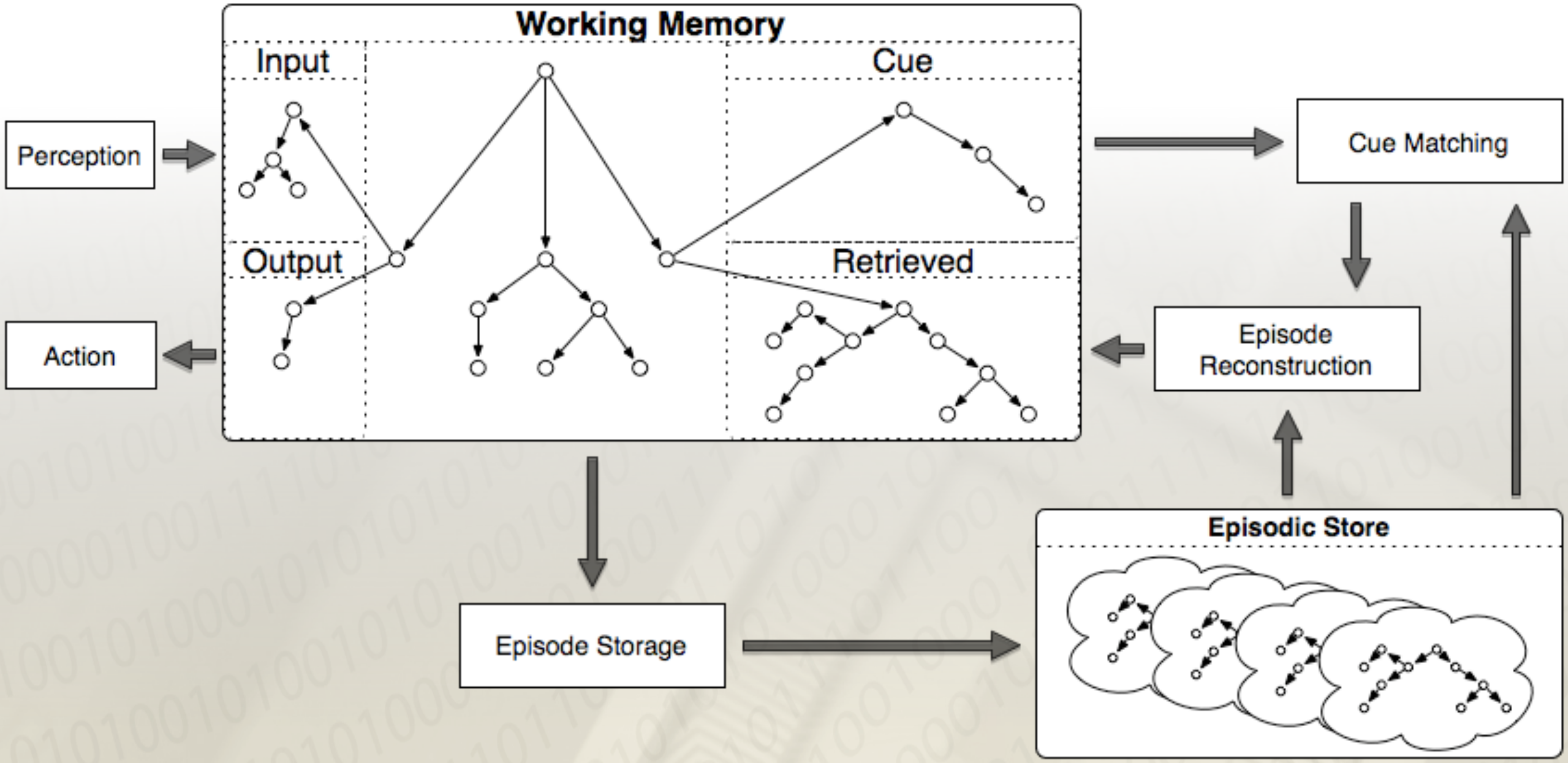
Integrating EpMem with Soar 9



Integrating EpMem with Soar 9



Integrating EpMem with Soar 9

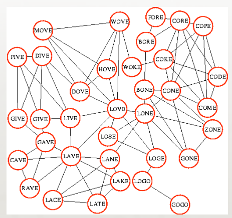
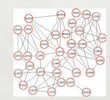
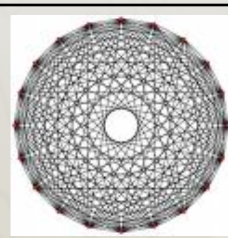
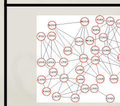
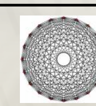
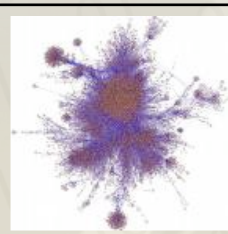
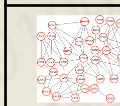
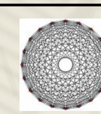
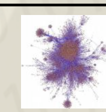


Episodic Storage

- Faithfully capture architecturally defined subset of working memory
- Incrementally update indexing structures to facilitate efficient cue matching
- Minimize
  - Memory (monotonically increasing store)
  - Time (relatively frequent operation)

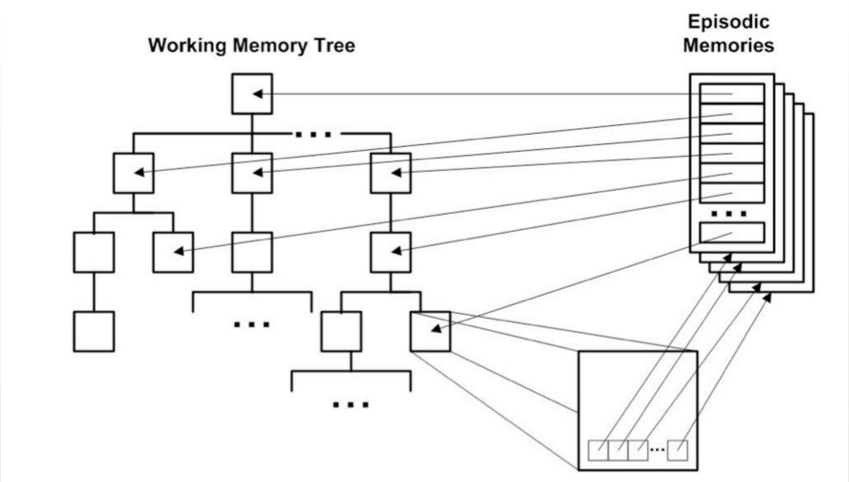
CSE

Episodic Storage: Naïve Implementation

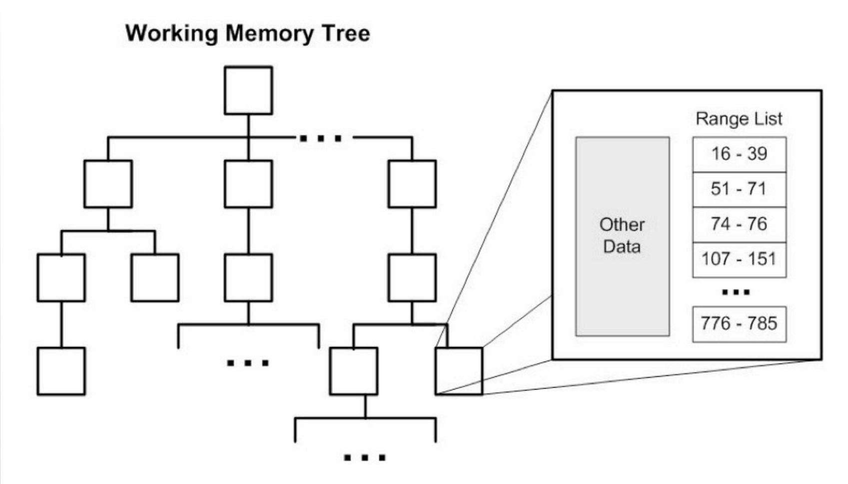
Time	Working Memory		Episodic Store		
1			1		
					
2			1	2	
					
3			1	2	3
					

Compression via Global Memory Structure

- Observation
  - Agents tend to re-use WM structures
- Result
  - Maintain a global record of unique structures
  - Define episodes as “bag of pointers”



Gains via Interval Representation



- Observation
  - An episode will differ from the previous (and next) only in a relatively small number of features
- Result
  - Define episodes implicitly as temporal *changes*

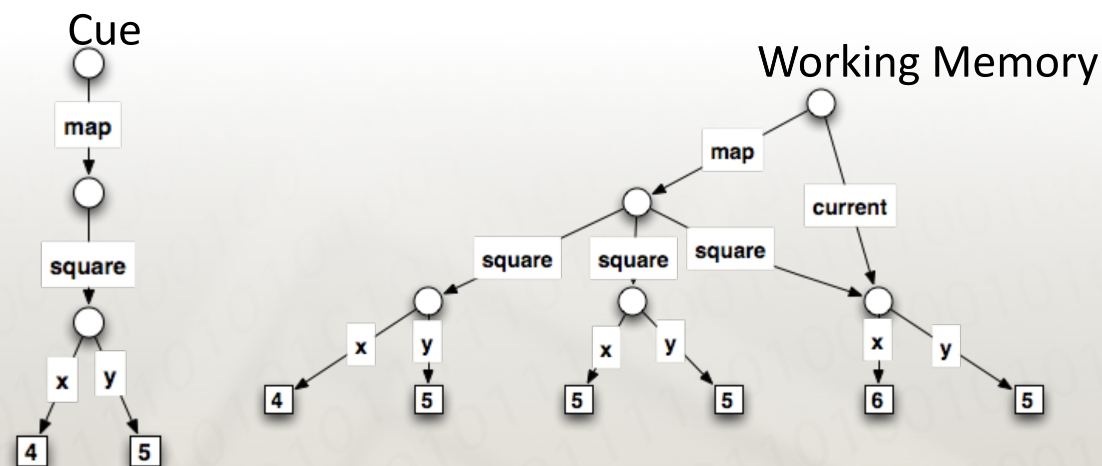
Episodic Storage Summary

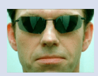
- Maintain record of unique structures
- Maintain associated intervals on node addition/removal
  - *Only process changes!*

Episodic storage performs in time/space <u>linear</u> in the *changes* in working memory.

Cue Matching

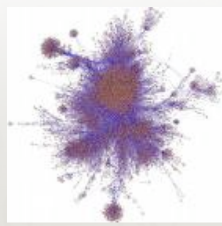
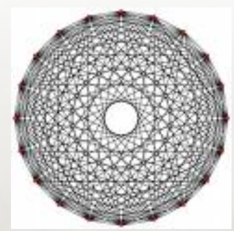
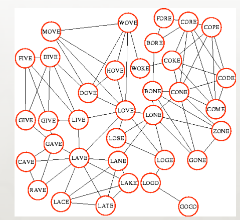
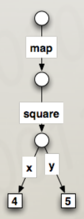
- A cue is an acyclic graph, partially specifying a subset of an episode



y \ x	4	5	6
5			

- Cue matching returns the <u>most recent</u> episode containing the greatest number of cue <u>leaf elements</u>

Cue Matching: Naïve Implementation

<table>
<tr><th>Time</th><th>N</th><th>N-1</th><th>N-2</th></tr>
<tr><td>Episode</td><td></td><td></td><td></td></tr>
<tr><td>
</td><td></td><td></td><td></td></tr>
<tr><td>Score</td><td>§</td><td>Ω </td><td>ß</td></tr>
</table>

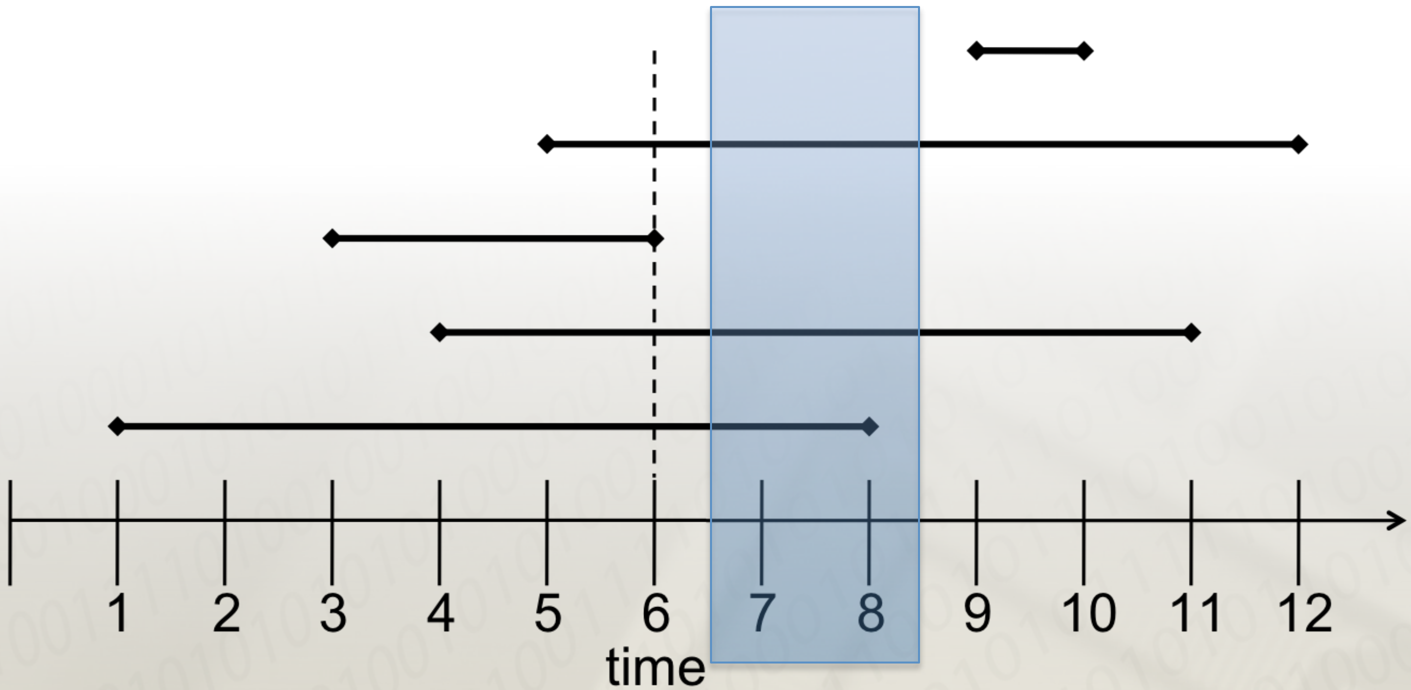
Minimizing Combinatorics via Two-Stage Matching

1. Evaluate *candidate* episodes based upon relatively inexpensive <u>surface</u> match

2. Perform combinatorial <u>structural</u> match (graph-match via CSP backtracking) ONLY on candidate episodes with a *perfect* surface score

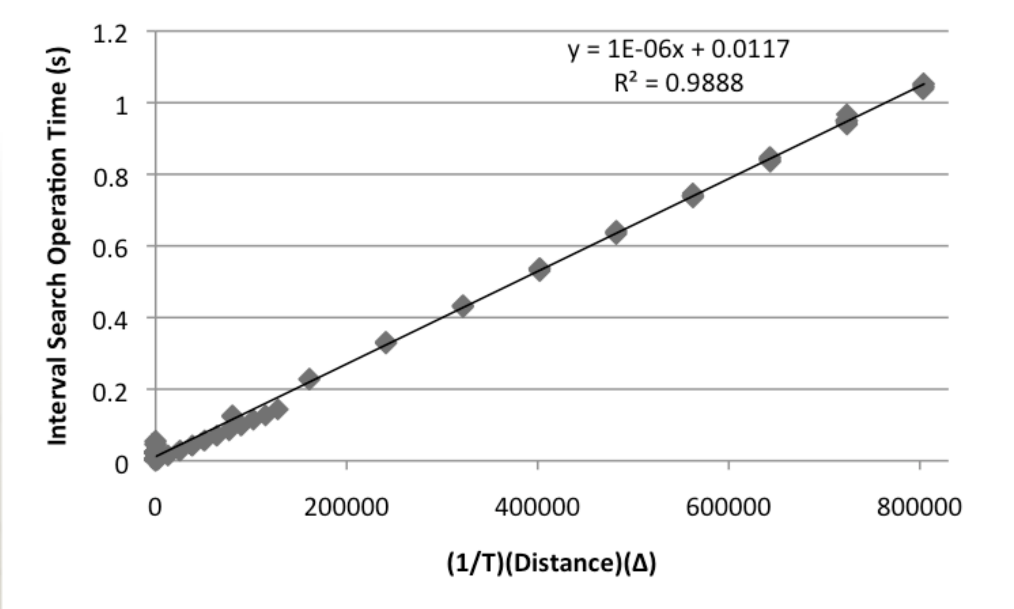
End search on perfect match or no more episodes.

Minimize Episode Evaluation via Interval Endpoint Search



Episode match score changes only at interval endpoints!

Interval Search Model



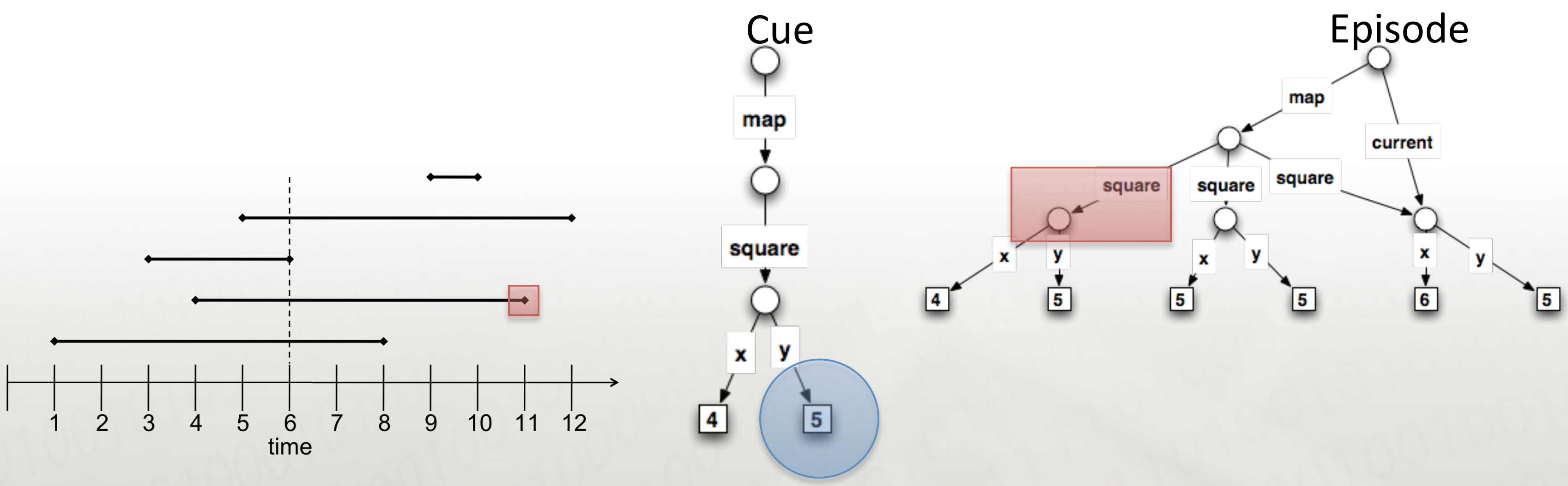
T = Total episodes

Distance = Temporal distance to best match

Δ = Cue intervals (~ WM changes)

Interval search is dependent upon the number of candidate episodes evaluated and WM *changes*.

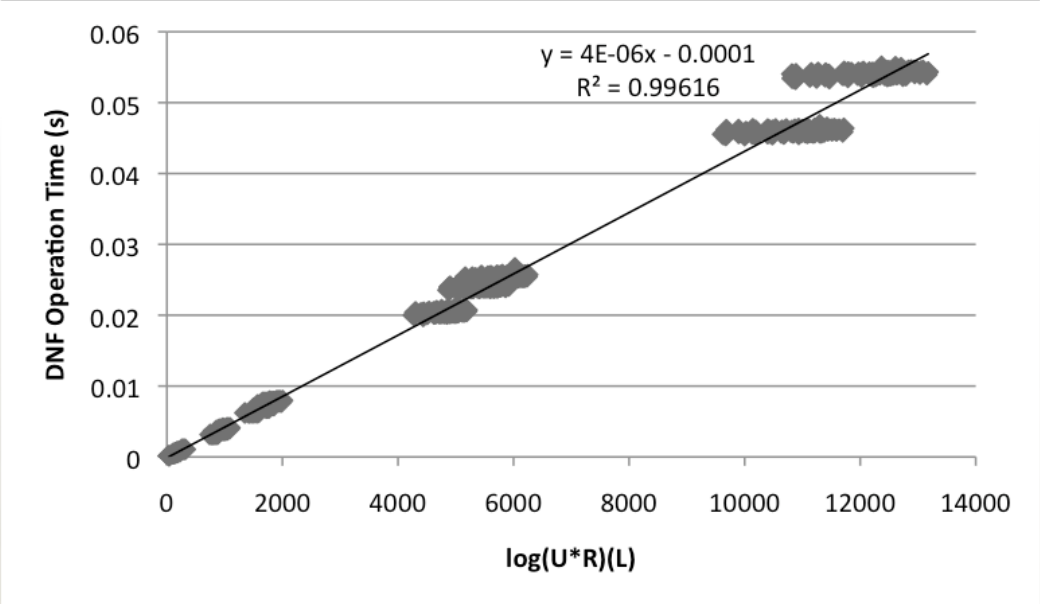
Efficient Surface Evaluation via Incremental DNF Satisfaction



- sat(y=5) := (root AND map[1] AND square[1] AND y=5[1]) OR
  (root AND map[1] AND square[2] AND y=5[2]) OR
  (root AND map[1] AND square[3] AND y=5[3])

- Surface matching can be expressed as evaluating the satisfaction of a set of disjunctive normal form (DNF) Boolean equations
  - **Each interval endpoint inverts the value of a <u>single variable</u>**

DNF Model



U = Unique nodes

R = Stored intervals (~ changes)

L = Cue node literals

DNF performance is dependent upon the changes in working memory.

CSE

Cue Matching Summary

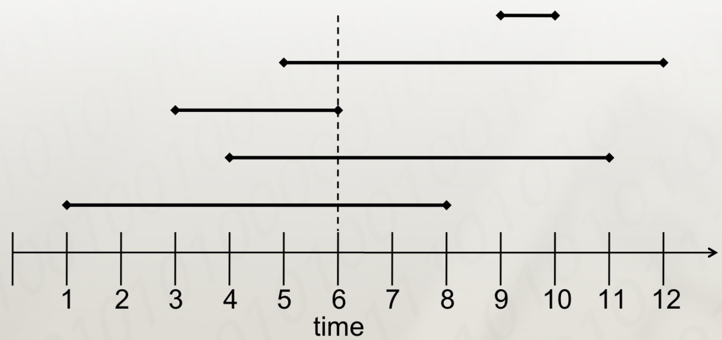
- Minimize candidates by only considering episodes with at least one cue node
- Minimize combinatorics via two-stage matching policy
  - Exponential growth in the worst case
- Minimize episode evaluation via interval endpoint search
  - Linear growth in the worst case
- Minimize surface evaluation cost by only processing cue node changes

Episode Reconstruction

- The process of faithfully reproducing all episode content and structure within the agent's working memory
  - Collect contributing episode elements
  - Add elements to working memory

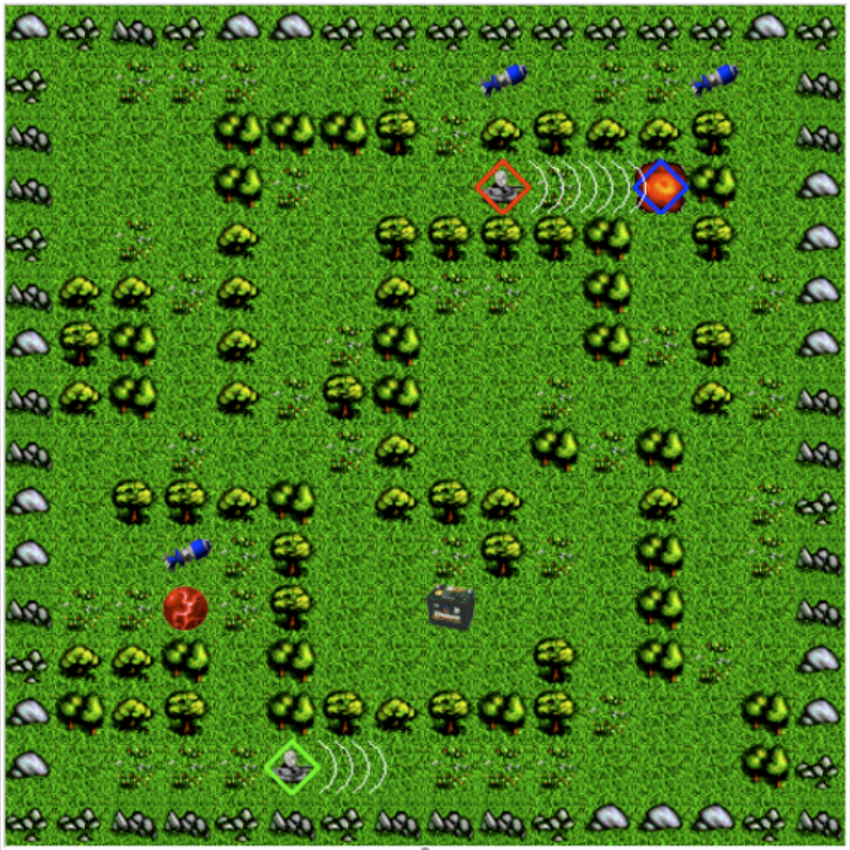
Logarithmic Interval Query via Relational Interval Tree

- Collecting episode elements in an Interval representation is tantamount to an interval intersection query:
  - Collect all elements that started before and ended after time *t*



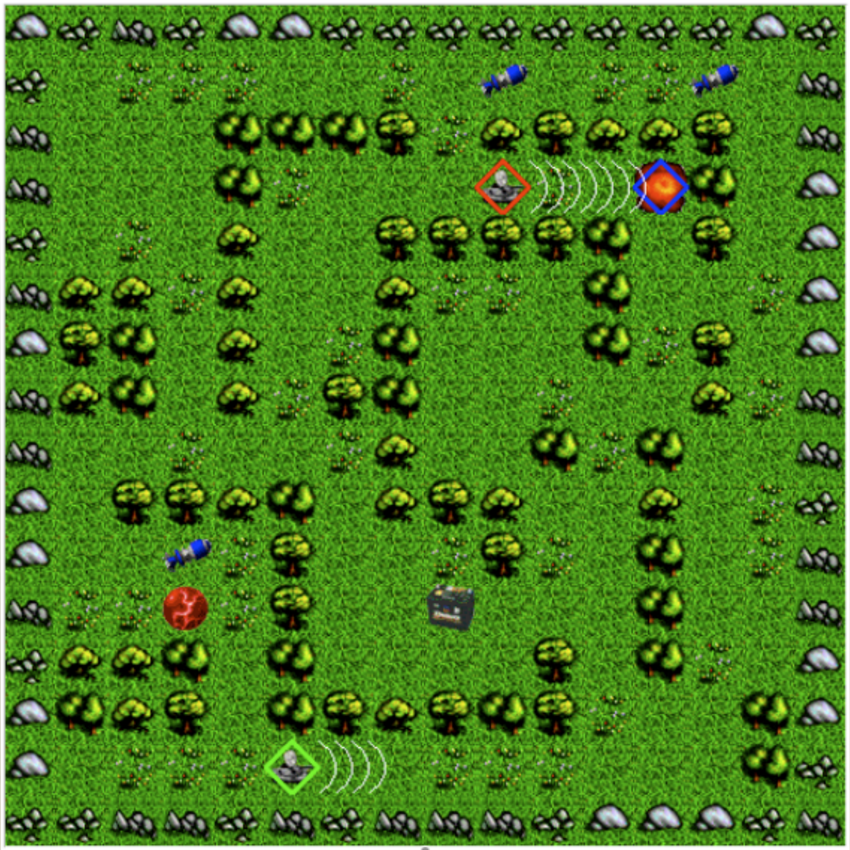
- By implementing an interval tree, intersection queries are answered in time <u>logarithmic</u> with respect to the changes in working memory

Empirical Domain: TankSoar



- Discrete 15x15 grid
  - Turn-based
- Turning, moving, firing missiles, raising shields, radar
- Smell, hearing, path blockage, radar, incoming missiles

Empirical Evaluation



- *Mapping-Bot*
  - 2500 features
  - 70-90% of perceptual inputs change each episode
- 2.8GHz, 4GB RAM
- SQLite3

Empirical Results

1 million episodes (~1 episode/decision), 10 trials

Storage	Cue Matching*	Reconstruction**	Total
2.68ms 625-1620MB (0.64-1.66KB/ep)	57.6ms	22.65ms	**82.93ms**

* 15 cues

** 50 random times

Future Work

- Better Evaluation
  - Characterize architecture performance with respect to properties of the environment, agent, cues, and task
  - Longer and multi-task runs

- Bound Cue Matching
  - Fast familiarity
  - Heuristic graph-match

- Algorithmic Variants
  - Selection bias: activation, arousal via appraisals
  - Stored episode dynamics
  - Characterize efficiency vs. proficiency

CSE

Summary

- Implemented graph-based, task-independent episodic memory
  - Released as Soar 9.1.1 beta
  - Gorski, N.A., Laird, J.E.: Learning to Use Episodic Memory (2009)
- Characterized computational challenges
  - Formal models of costs related to episodic operations
  - Initial empirical study for 1 million episodes