Rules Fest 2010

International Conference on Reasoning Technologies

October 11-14 • San Jose, CA USA

Grindwork Corporation
Intelligent Automation

JBoss by Red Hat

IBM

Production Systems Technologies

morris technical solutions
engineering knowledge for all businesses

VisionArts communications
strategic thinking • creative action

Soar Rules

The Role of Production Rules in a General Cognitive Architecture

John E. Laird

John L. Tishman Professor of Engineering

University of Michigan

Research funded by TARDEC, ONR, AFOSR

Origins

"The production system was one of those happy events, though in minor key, that historians of science often talk about: a rather well-prepared formalism, sitting in wait for a scientific mission. Production systems have a long and diverse history. Their use of symbolic logic starts with Post (1943), from whom the name is taken. They also show up as Markov algorithms (1954). Their use in linguistics, where they are also called rewrite rules, dates from Chomsky (1957). As with so many other notions in computer science, they really entered into wide currency when they became operationalized in programming languages."

(Newell and Simon, 1972, p. 889).

Emil Post 1943

- Post production systems
  - P1: $$ -> *
  - P2: *$ -> *
  - P3: *x -> x*
  - P4: * -> null & halt
  - P5: $xy -> y$x
  - P6: null -> $

Newell & Simon (1965)

- Proposed post-production systems as basis for programming languages
- Outgrowth of research on human problem solving
  - Logic, Chess, Crypta-arithmetic: SEND + MORE = MONEY
- Strengths of rule-based systems
  1. *Computational generality (Turing Equivalent)*
  2. *Independent units/components of behavior – add/remove easily*
  3. *Models human-short-term memory and human procedural memory*
- PSG (Production System, version G) - 1973
  - First "widely" used rule language

Descendants of PSG

PSG
Newell, '73

ACT
Anderson '76

PRISM
Langley '80

CAPS
Just, Carpenter '87

Used for Cognitive Modeling

OPS1-5
Forgy, McDermott, Newell, 1976-1981

Soar
Laird et al, '82

Application Oriented

OPS83
Forgy

ART
'84

CLIPS
Riley '86

JESS
Friedman-Hill, '95

Drools
2001

Jrules
1996

OPSJ
Forgy

Cognitive Architecture

- Fixed structures that support cognitive functions
  - Representations of information
  - Memories
  - Processing units
  - Interfaces to perception and motor system

- Functional requirements
  - Supports end-to-end behavior, reactivity, decision making, planning, learning, …
  - Relevant knowledge is brought to bear when needed
  - No “escape” to arbitrary programming

Different Goals of Cognitive Architecture Research

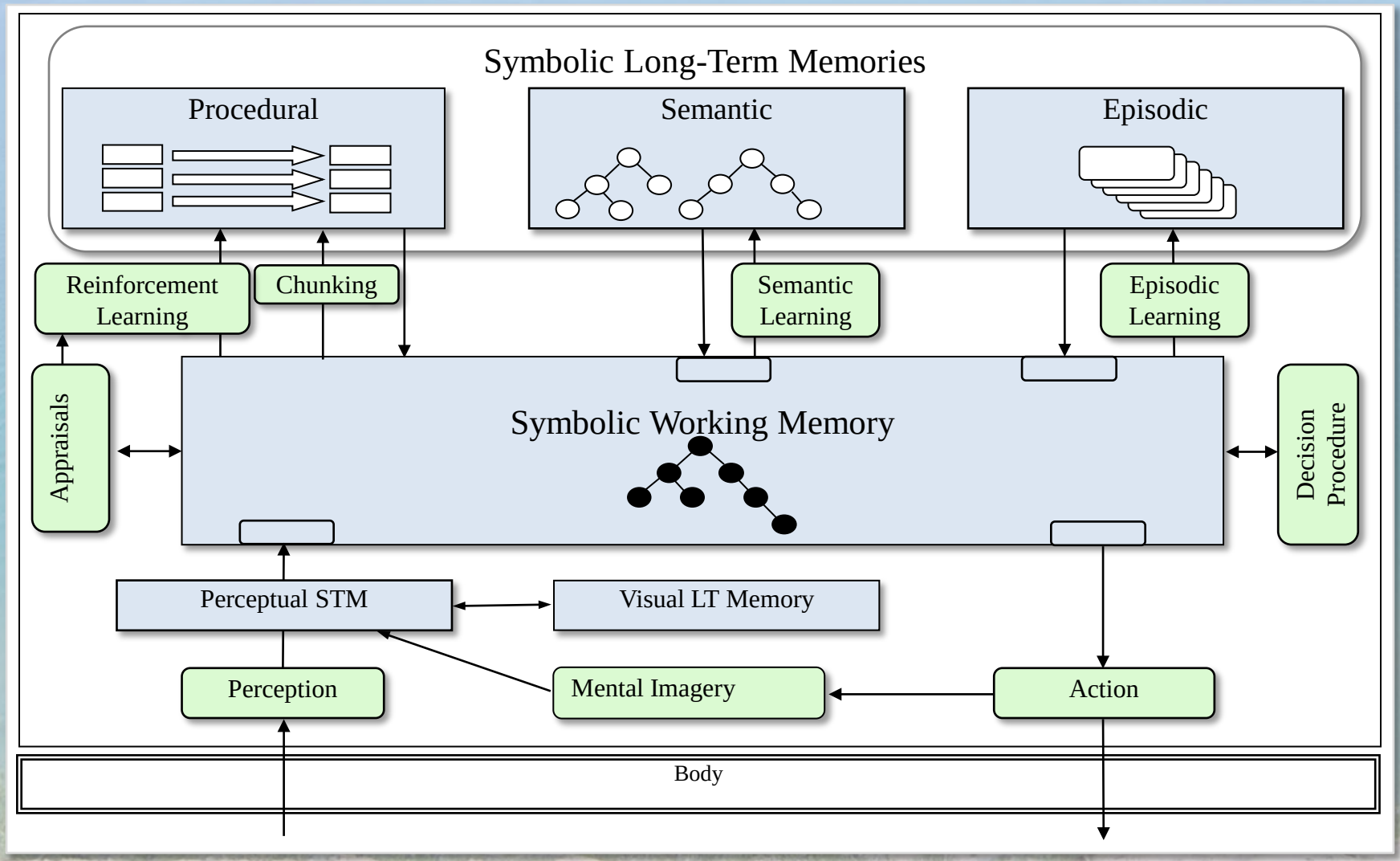
- **Biological Modeling:** Capture what we know about the brain
  - LEABRA
- **Psychological Modeling:** Capture what we know about the mind – the details of human performance in a wide range of cognitive tasks
  - ACT-R, CAPS, CLARION, EPIC, Soar, …
- **Functionality:** Support the broad range of human cognitive capabilities across wide range of tasks
  - Companions, ICARUS, LIDA, Soar, …
- *Different requirements than in commercial rule applications!*

Soar

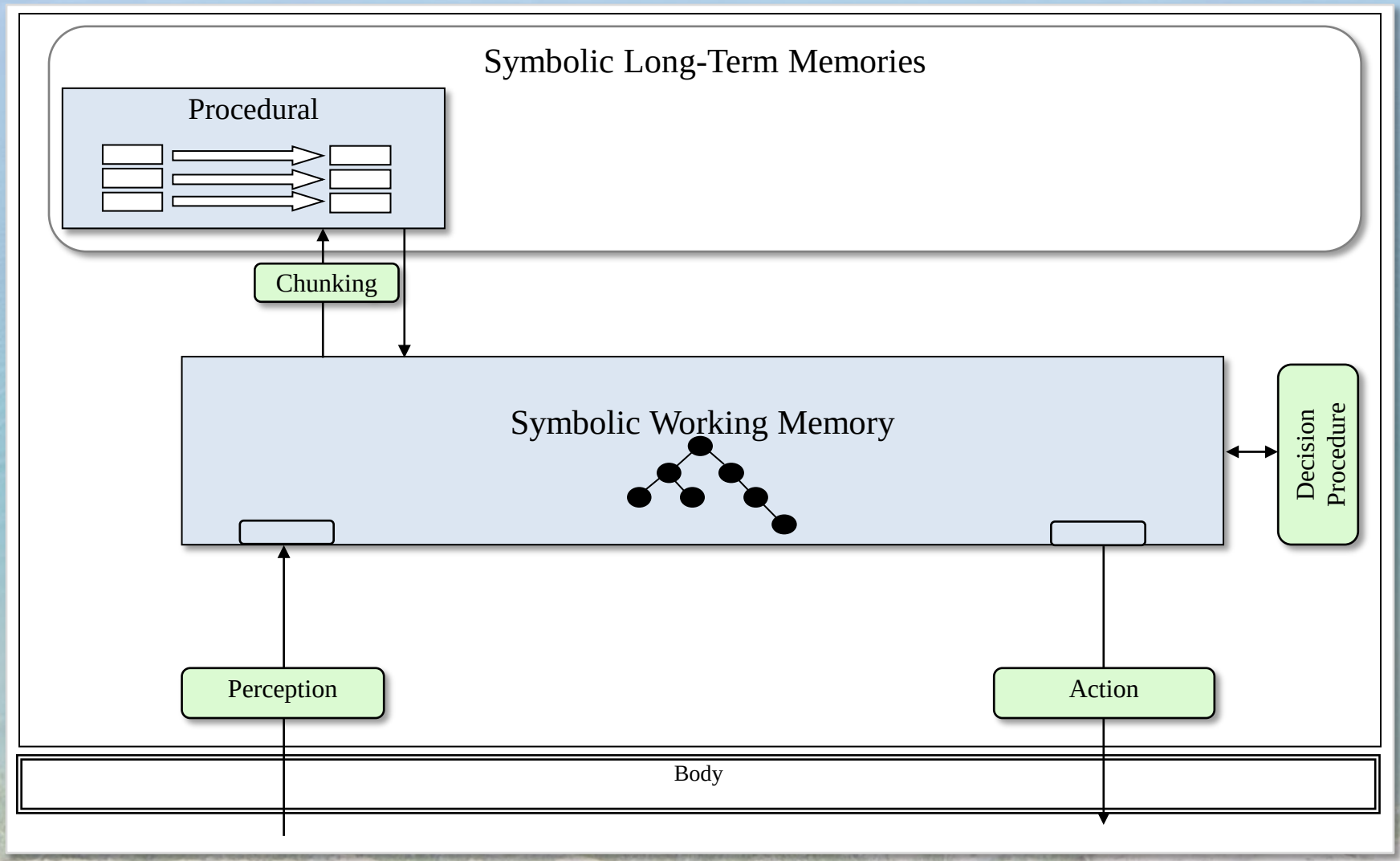
(Laird, Newell & Rosenbloom, 1982)

- Cognitive architecture for human-level intelligence
- Multi-method, multi-task problem solving
- Combined problem spaces & production systems
- Direct Lineage: OPS5, XAPS2, GPS, LT

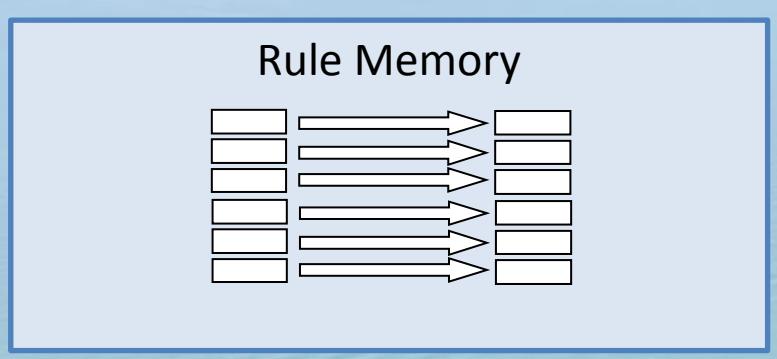
Soar 9 Structure

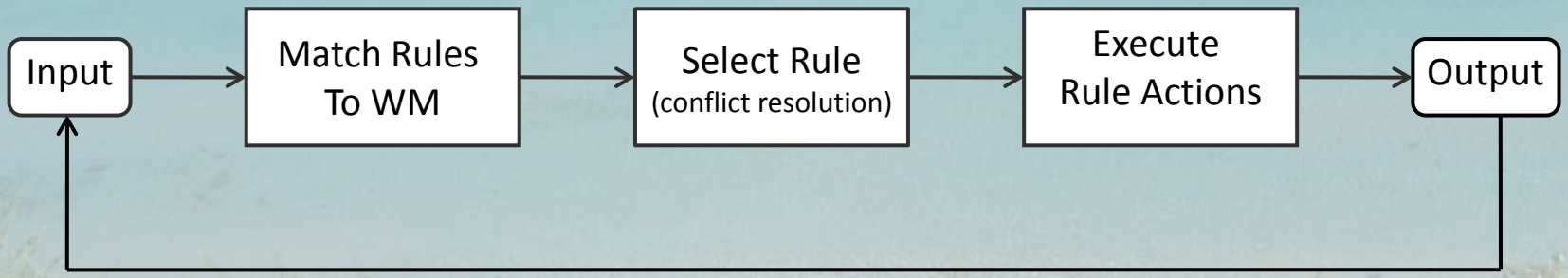


Original Soar Structure



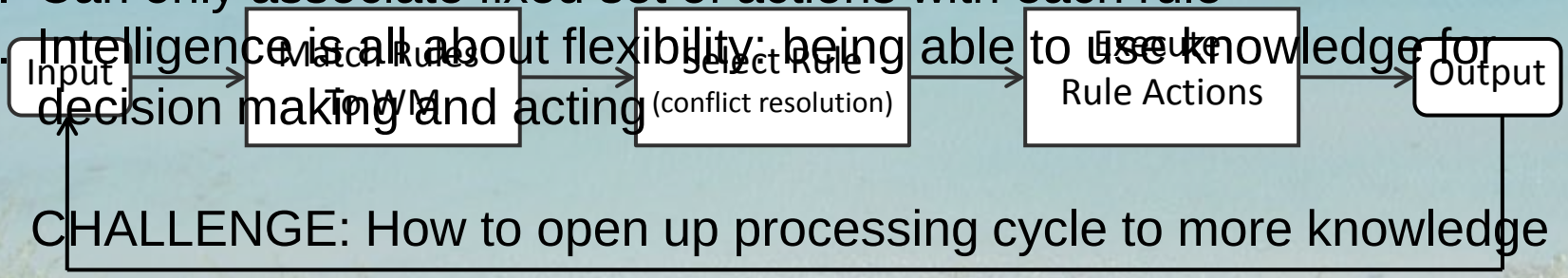
Rule-based System Processing Cycle



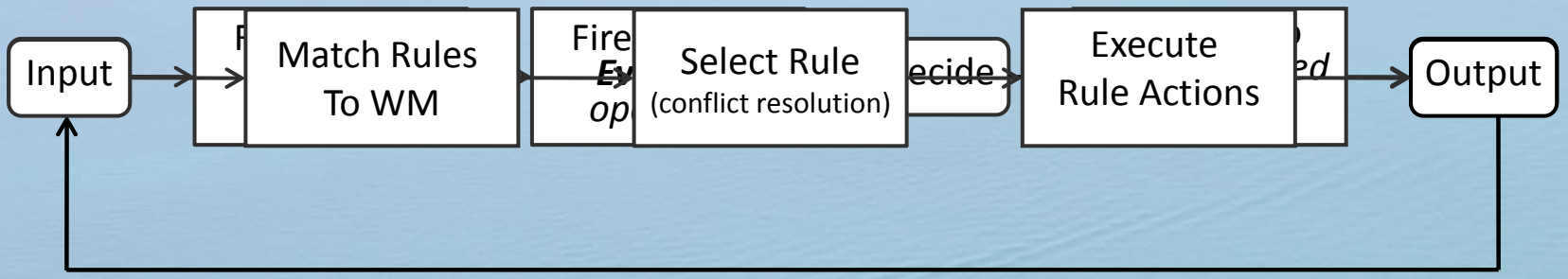


Rule-based System Processing Cycle

1. Conditionality is limited to determining which rules match.
2. Cannot use additional task knowledge to *select* one rule
   - Rely on fixed conflict resolution schemes using syntactic features
   - Works about 80%
   - Forced to “engineer” conditions to avoid conflicts
3. Can only associate fixed set of actions with each rule
4. Intelligence is all about flexibility: being able to use knowledge for decision making and acting

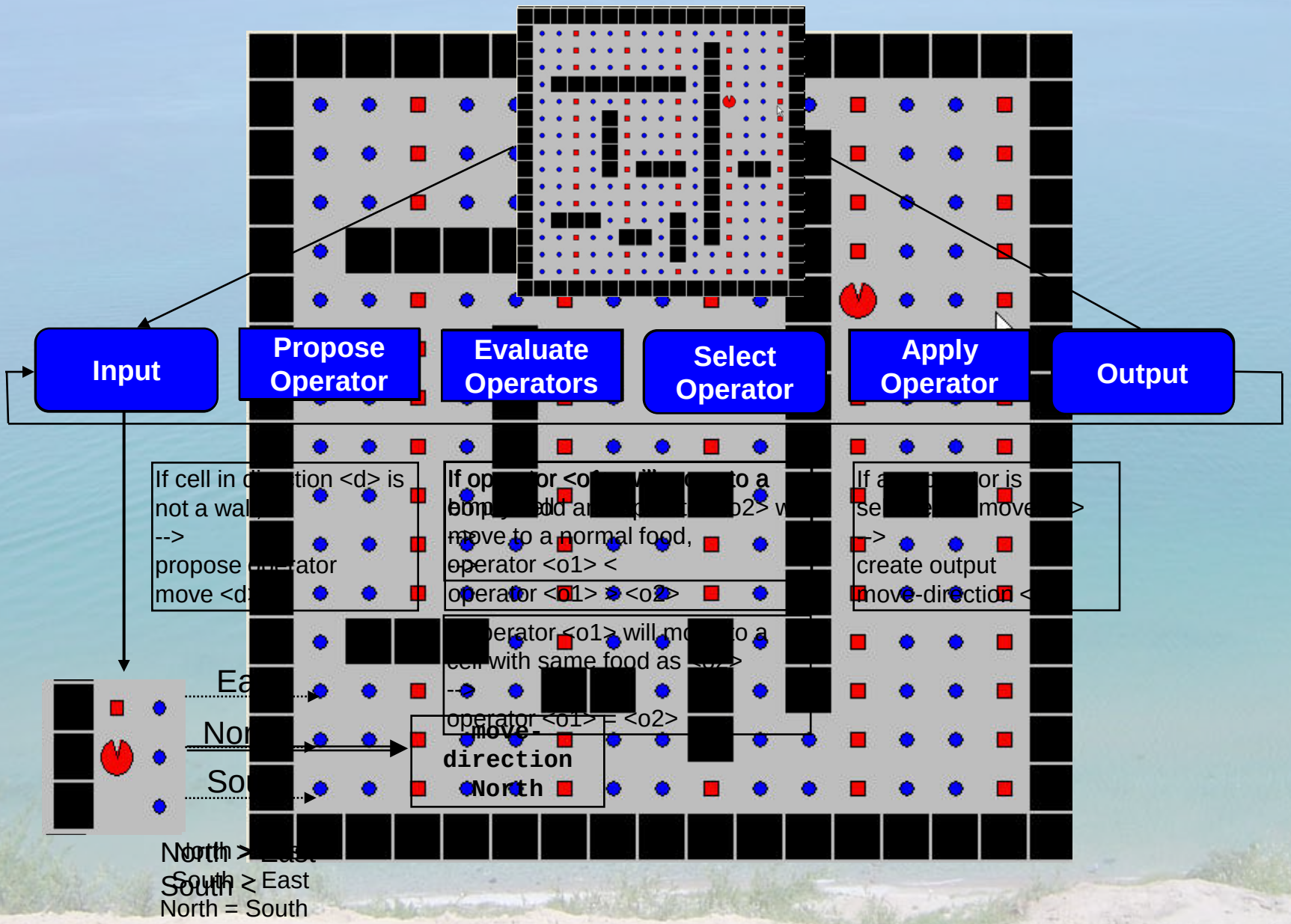


Rule-based System Processing Cycle



- Introduce **OPERATORS** as basic unit of deliberation.
  - Explicitly represent current operator
- Rules contain knowledge that
  - **Propose Operators**: create *acceptable preferences*
  - **Evaluate Operators**: create *preferences*
  - **Apply Operator**: modify working memory
- All rules that match fire in parallel

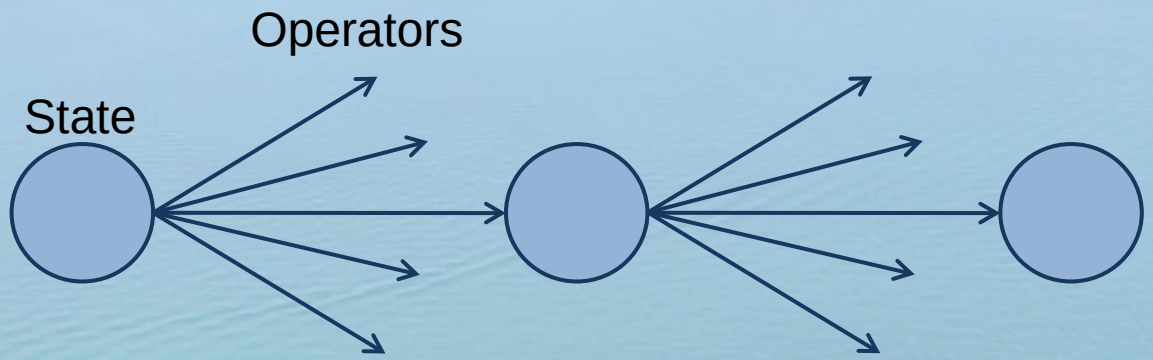
Soar 101: Eaters

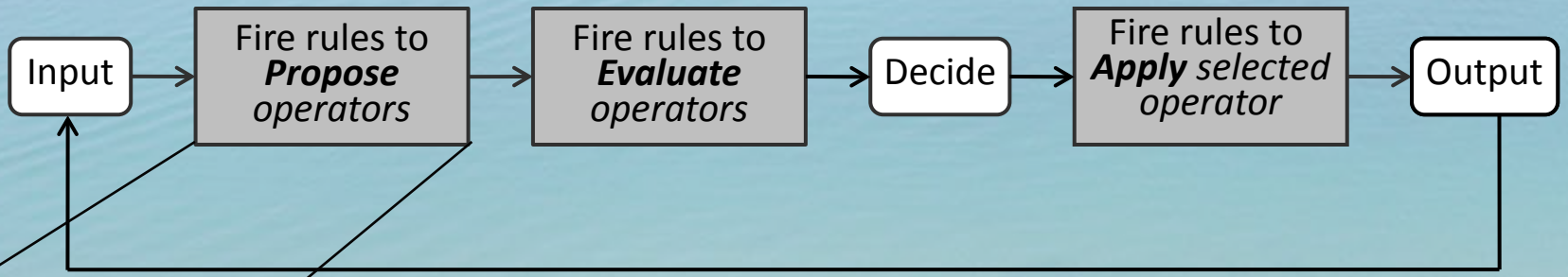


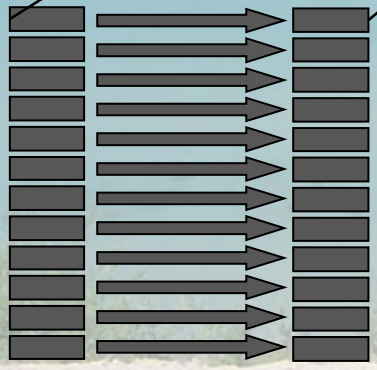
Problem Search & Knowledge Search

Problem Search

- Propose legal operators
- Select best for situation
- Generates new structures



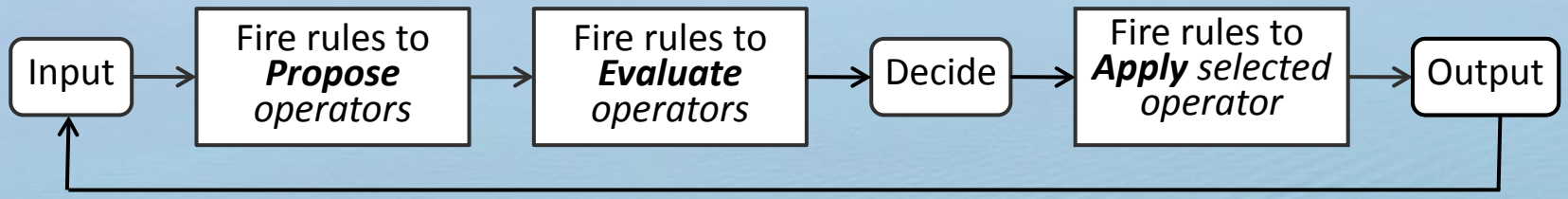




Knowledge Search

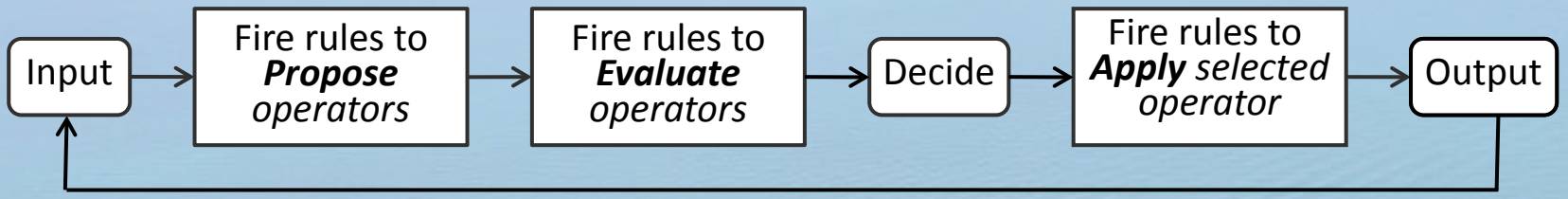
- Find relevant knowledge for operator proposal, selection, application
- Searches through known structures (rule conditions)
- Inner loop of behavior: must be bounded to maintain reactivity

Two Related Questions



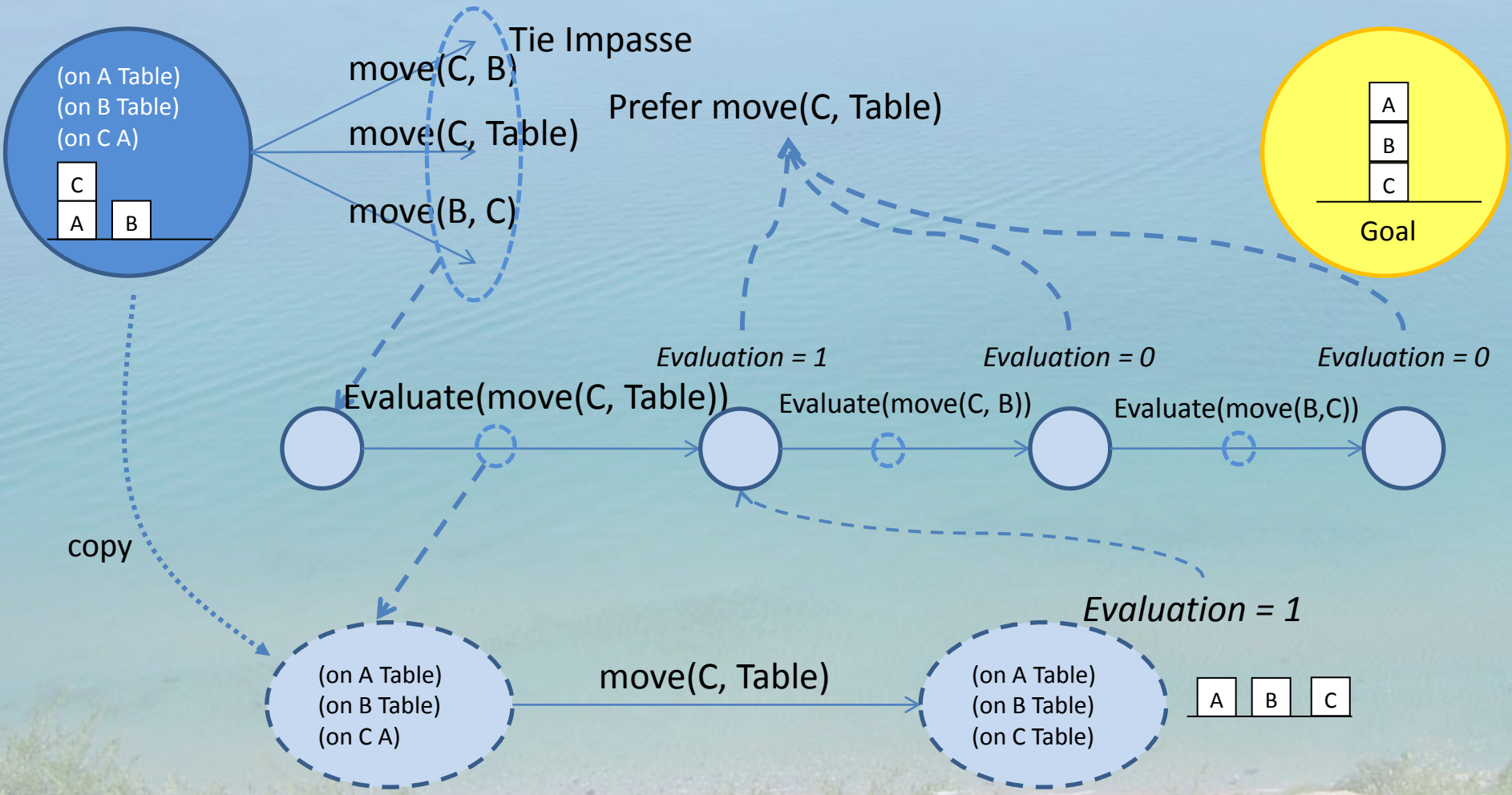
1. How use complex reasoning/knowledge for operator selection and application knowledge?
   - Without introducing a new formalism
   - While still maintaining reactivity
2. What happens when knowledge search fails:
   - Insufficient knowledge (encoded as rules) to propose, evaluate, or apply operators?

Impasses and Substates

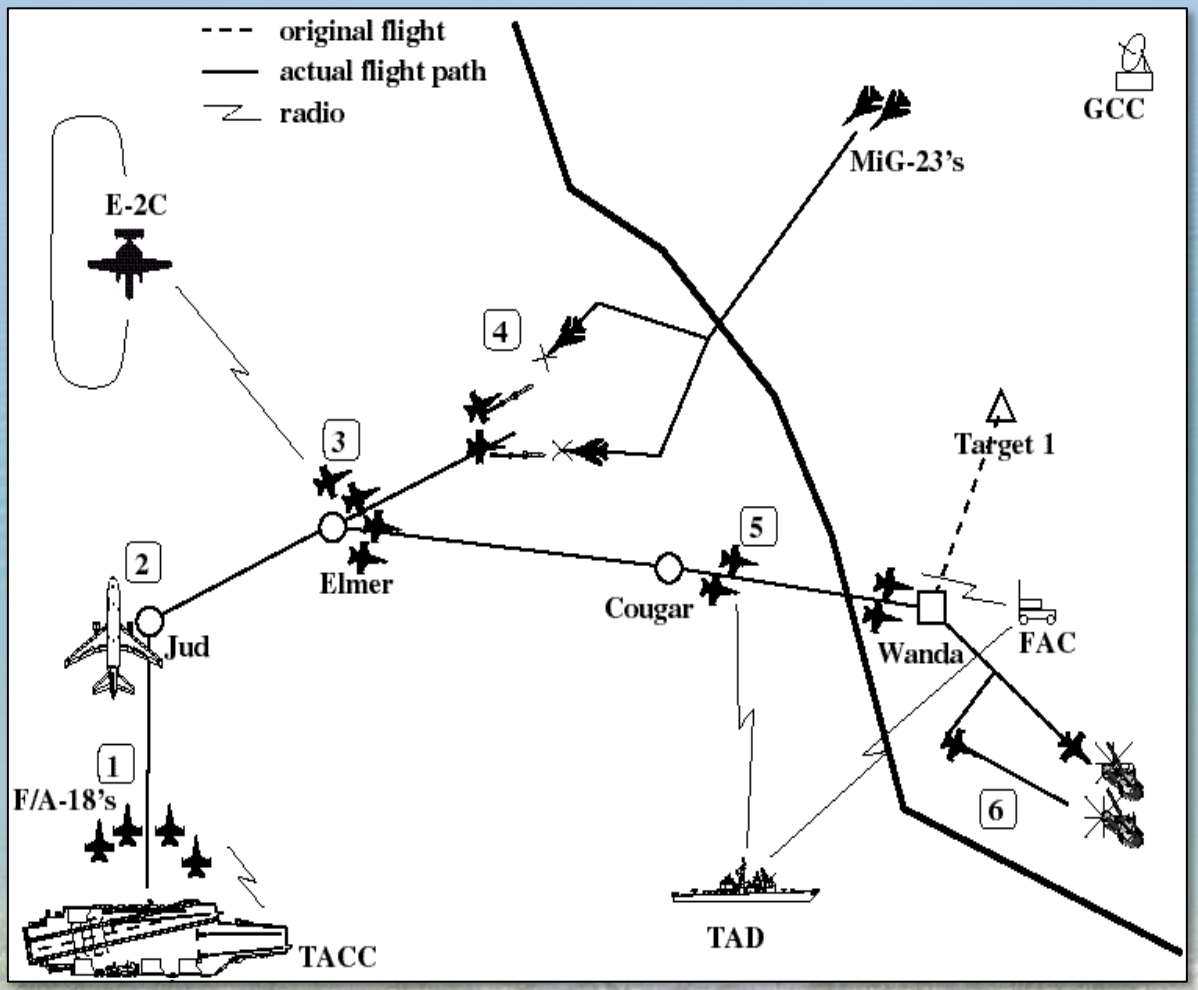


- If knowledge search fails (incomplete or conflicting knowledge is retrieved)
  - Generate a substate
  - Recursively use operators to reason about impasse
- Leads to internal planning, hierarchical task decomposition, …
- One form of meta-cognitive processing

One-step Look-ahead



TacAir-Soar: 1997



Controls simulated aircraft in real-time training exercises (>3000 entities)

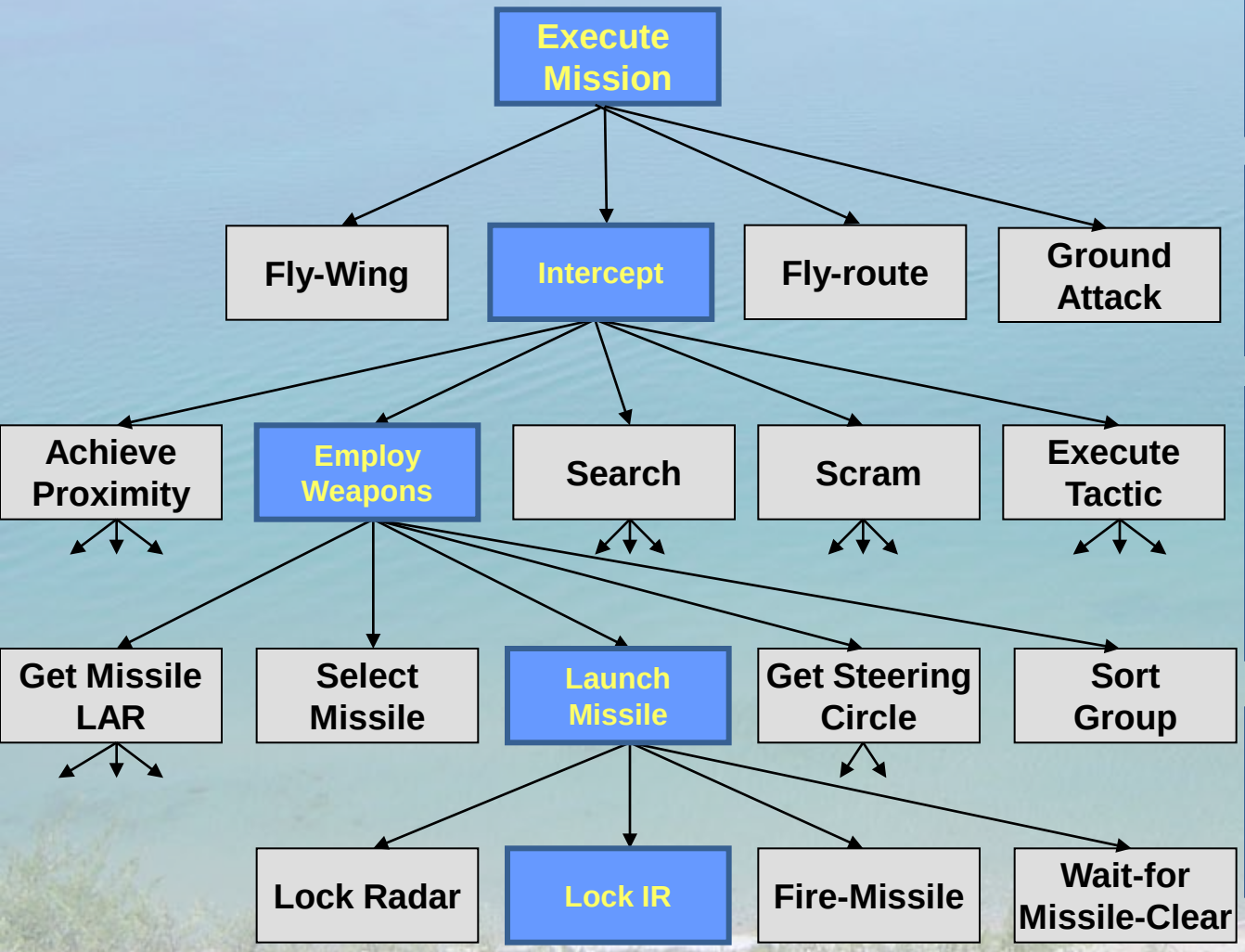
Flies all U.S. air missions

Dynamically changes missions as appropriate

Communicates and coordinates with computer and human controlled planes

>8000 rules

TacAir-Soar Task Decomposition

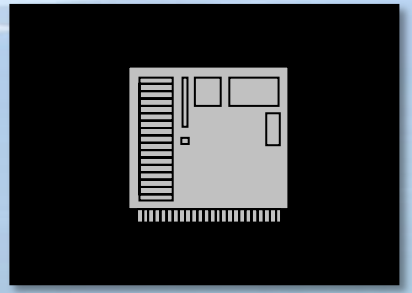


If instructed to intercept an enemy then
propose intercept

If intercepting an enemy and the enemy is within range ROE are met then
propose employ-weapons

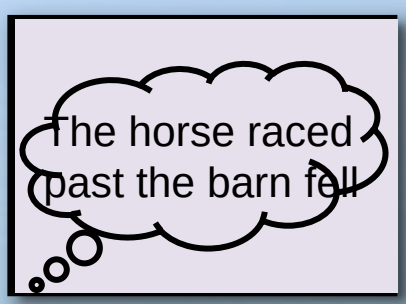
If employing-weapons and missile has been selected and the enemy is in the steering circle and LAR has been achieved,
then propose launch-missile

If launching a missile and it is an IR missile and there is currently no IR lock
then propose lock-IR



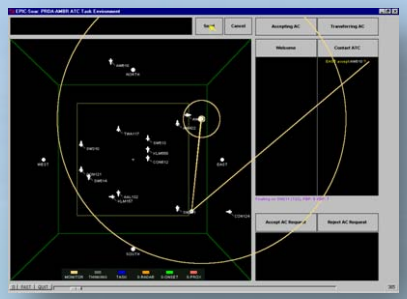
R1-Soar

Computer Configuration



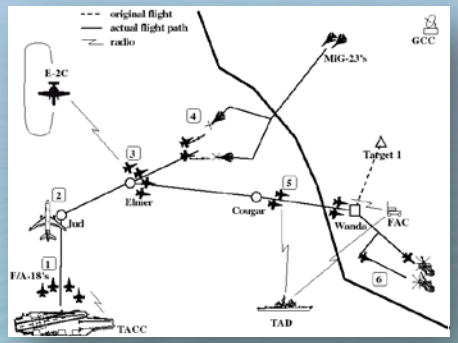
NL-Soar

Natural Language Processing



Amber EPIC-Soar

Modeling Human-Computer Interaction



TacAir-Soar

Complex Doctrine & Tactics Execution

Urban Combat

Transfer Learning

Soar Quakebot

Anticipation of Enemy Actions

Haunt

Actors and Automated Direction

MOUTbot

Team Tactics and Unpredictable Behavior

SORTS

Spatial Reasoning & Real-time Strategy

Simulated Scout

Mental Imagery

Splinter-Soar

Robot Control

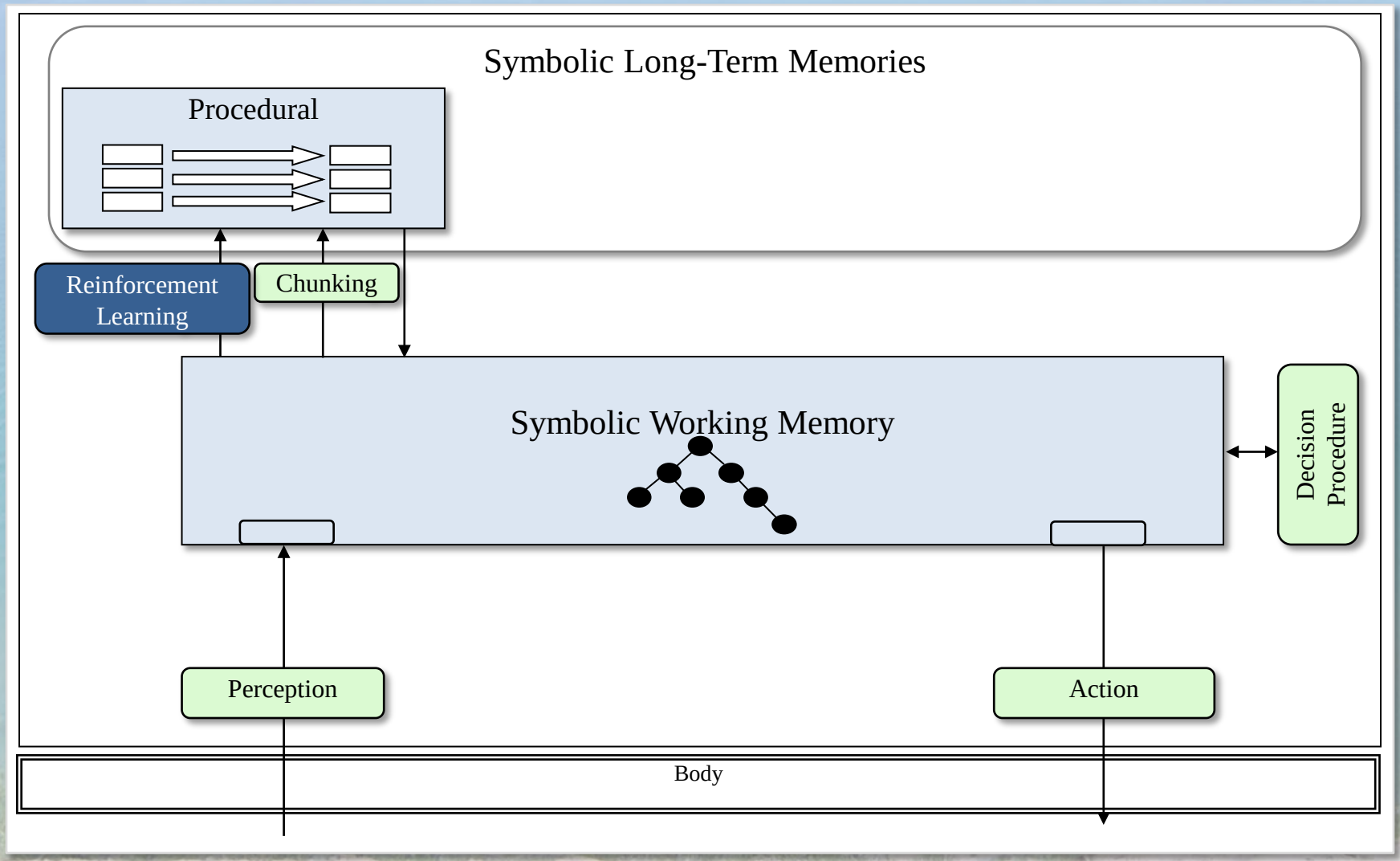
Beyond Rules: How to Use and Learn Other Forms of Knowledge?

- Statistical regularities related to success/failure
  - Reinforcement Learning
- Factual Knowledge
  - Semantic Memory
- Memory of Experiences
  - Episodic Memory
- Spatial and Visual Knowledge
  - Mental Imagery

Reinforcement Learning in Soar

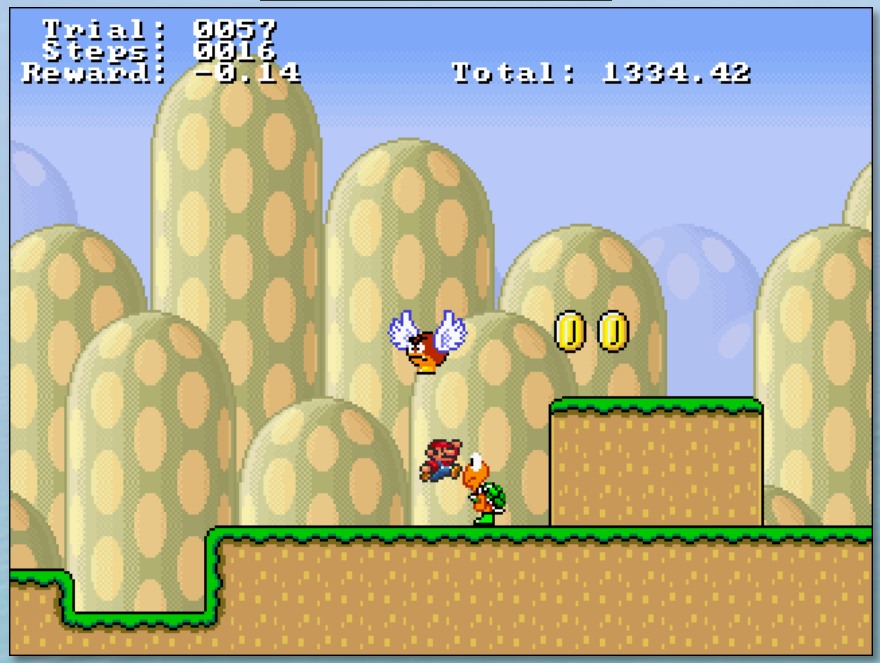
- Get reward from environment
  - Scalar value for good/bad results
- Operator evaluation rules encode expected value of operator
  - If operator A is selected in situation B, expected value is X.
- Each time an operator is applied, update evaluation rules based on reward and reward expected for next state.

Soar 9 Structural Diagram



Interactive Hierarchical Reinforcement Learning: Shiwali Mohan

Infinite Mario World



Agent Design

Input

- type of tiles in the visual scene (brick, coin etc.)
- monsters - location, horizontal/vertical velocity, type, wings

Output

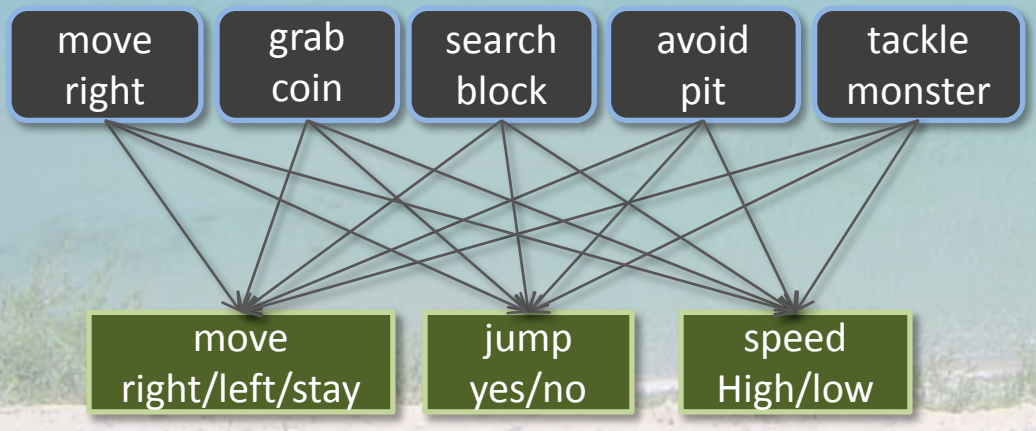
- primitive actions: left or right movement, jump, high/low speed

Relational State Representation

- of the form - 'there exists a pit at a distance of three tiles in horizontal direction'
- qualitative attributes – 'isthreat' for monsters and pits, 'isreachable' for coins and blocks

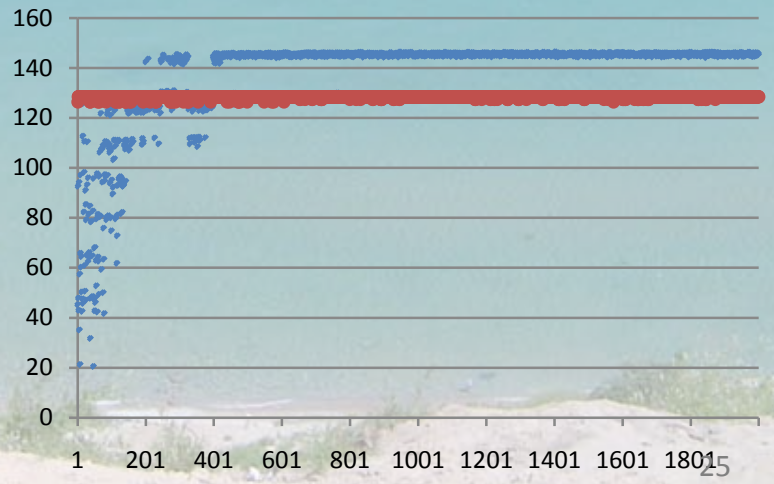
Operator Hierarchy

Functional Level Operators



Key-stroke Level Operators

Performance of Soar RL agent and Sample Agent



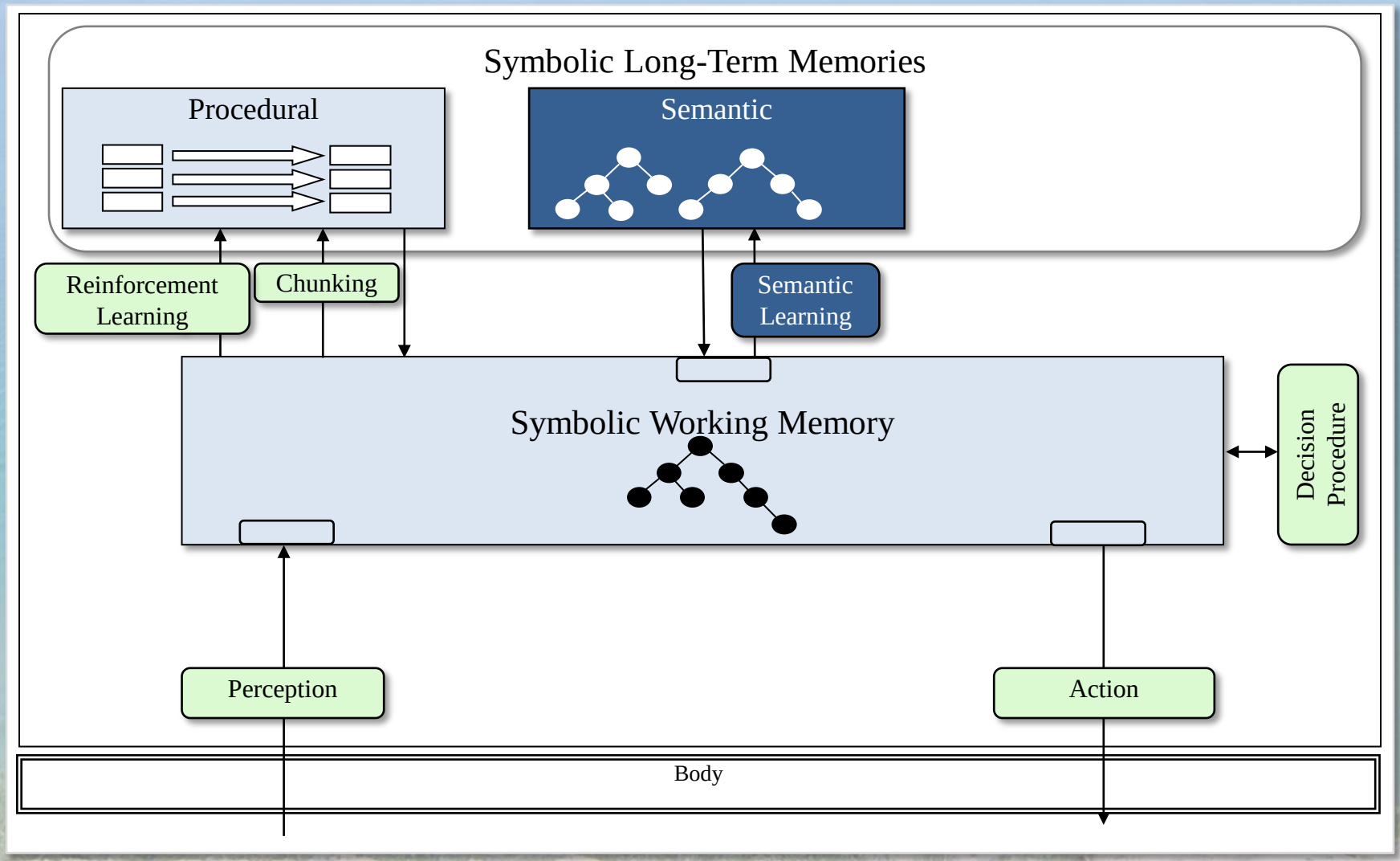
Which card is higher?

There is a King on the card with the name of a movie in which this state's governor starred. The other is a six.

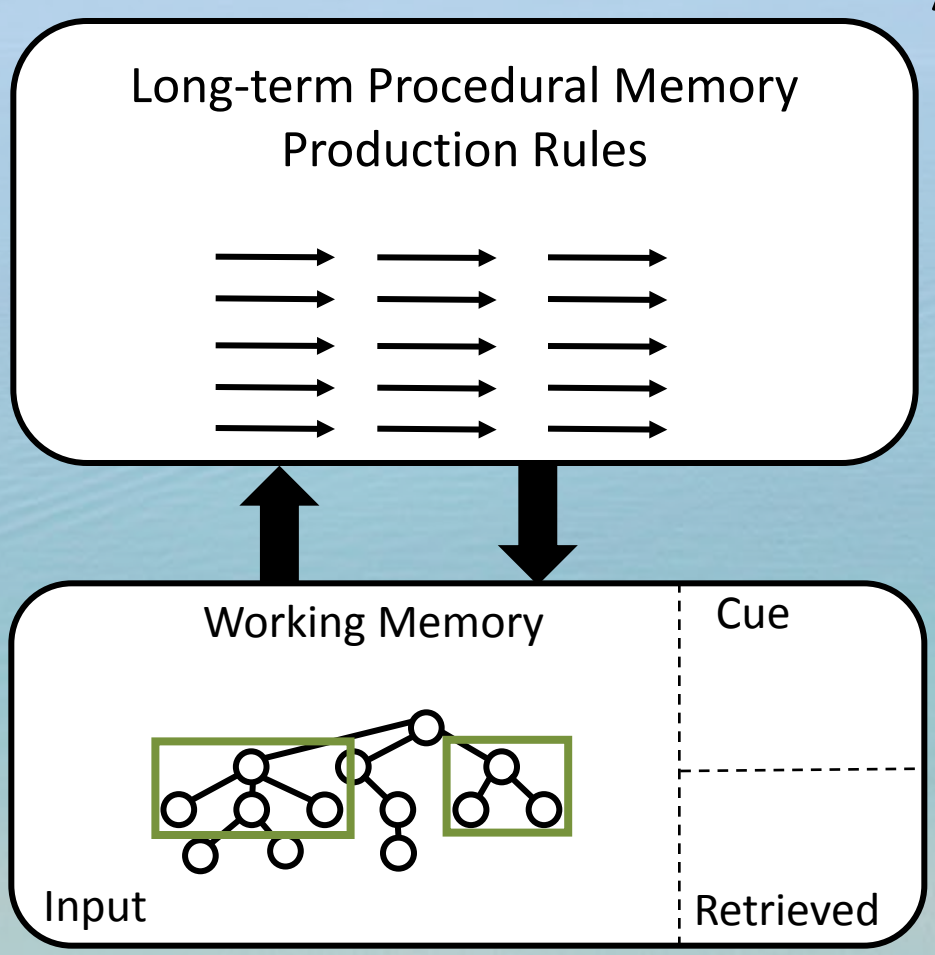
Semantic Memory

- Long-term declarative store of facts
  - Knowledge acquired by agent
  - Preload from external databases (Cyc, Wordnet, …)
- Why not just use working memory?
  - Slows down significantly as amount of data increases
- Why not rules?
  - Requires a rule for every way data might be accessed
- Our implementation
  - Statistically-based query optimization
  - SQL-Lite backend

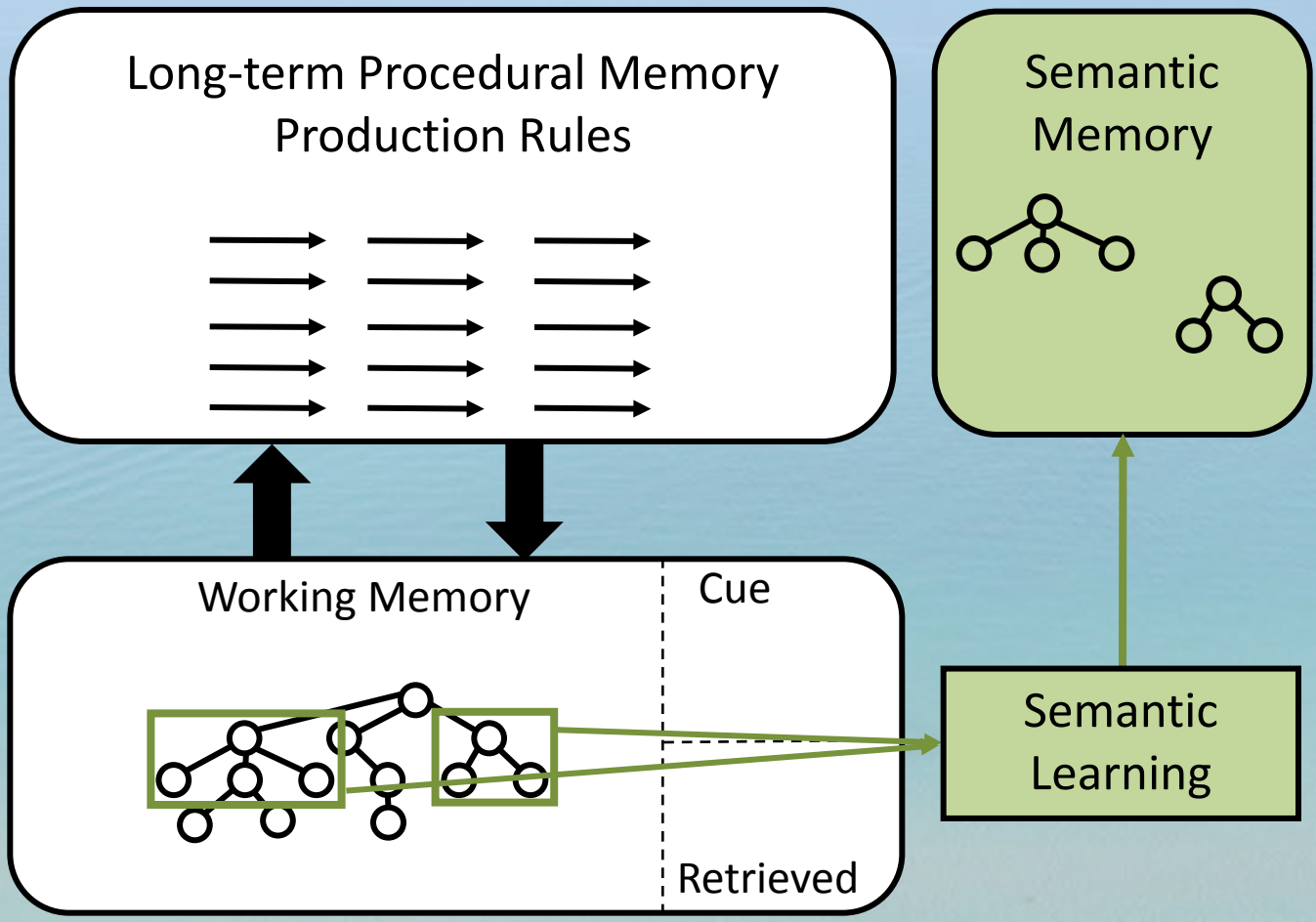
Soar 9 Structure



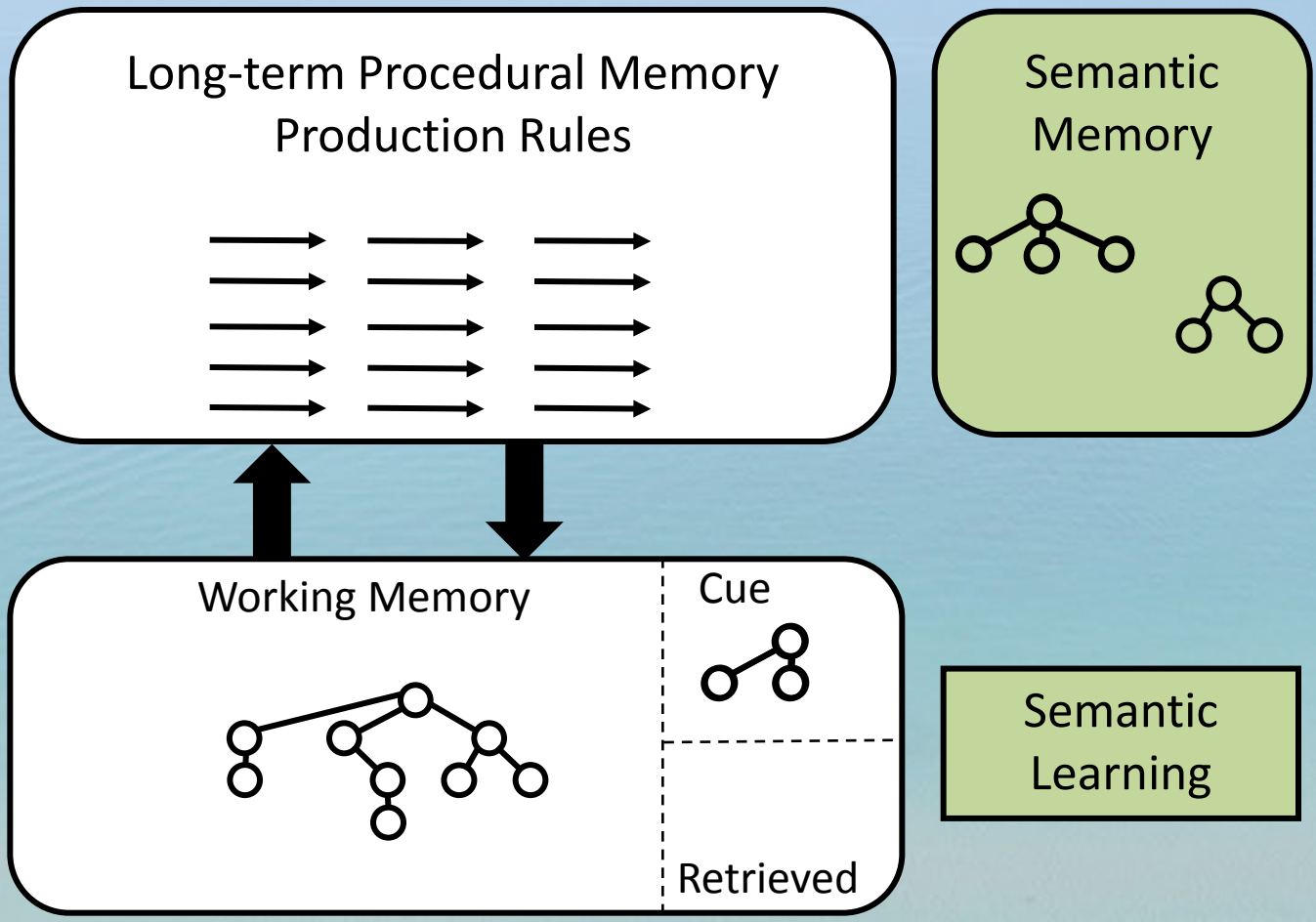
Soar Semantic Memory



Soar Semantic Memory



Soar Semantic Memory



WordNet Results
(Nate Derbinsky)

- Soar (WN-Lexical, v3)
  - 821K nodes
  - 3.8M edges
  - 398.1MB on disk
- ACT-R
  - 240k nodes
  - 2.2M edges
  - ~40 msec. retrieval
    - 1-4 cue size

Query Type	Cue Size	Avg. Retrieval Time (msec.)	Std. Deviation
100 random	1	0.211	0.0135
5 word senses	2	0.222	0.0107
“soar”	7	0.277	0.0017

*Intel 2.8GHz Core 2 Duo, 4GB RAM, Mac OS 10.6, SQLite v3

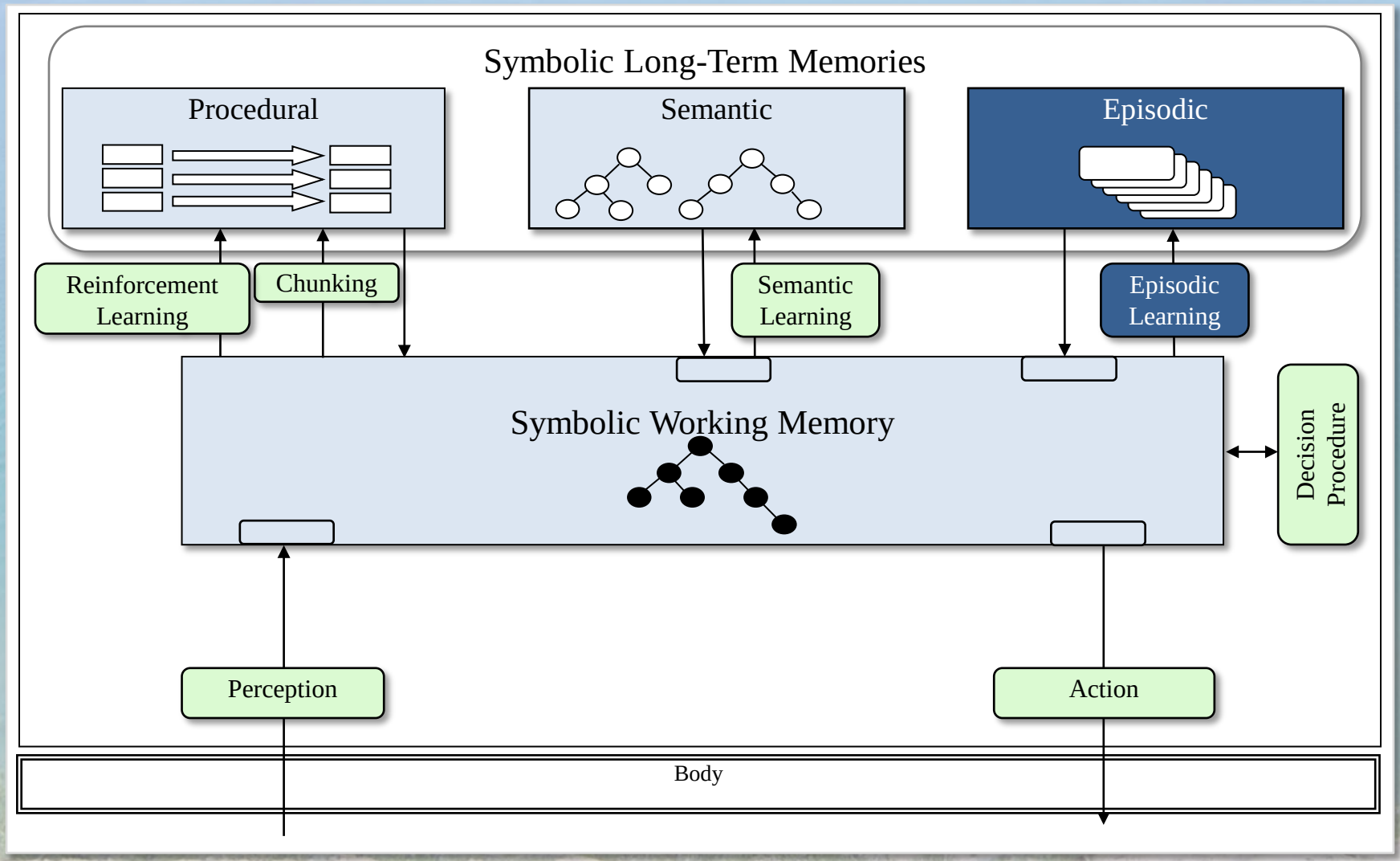
Which card should I pick to get the highest value?

The cards are the same as last time.

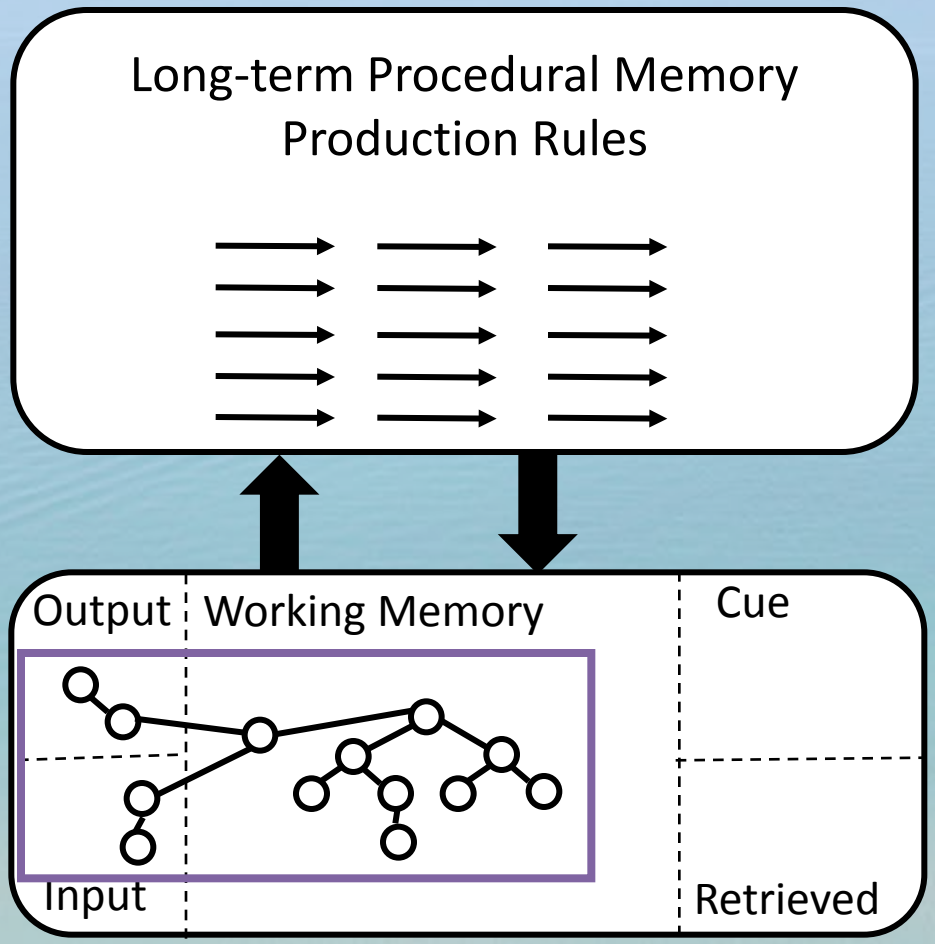
Episodic Memory

- Long-term declarative store of experiences
  - What you “remember”
- Why not just use working memory?
  - Significant slow down as experiences accumulate
  - Difficult to support biased partial match retrievals
- Our implementation
  - Separate Memory
    - Store only WM changes to episodic memory
    - Reconstruct episodes
  - B+ trees provide access to SQL-Lite backend

Soar 9 Structure

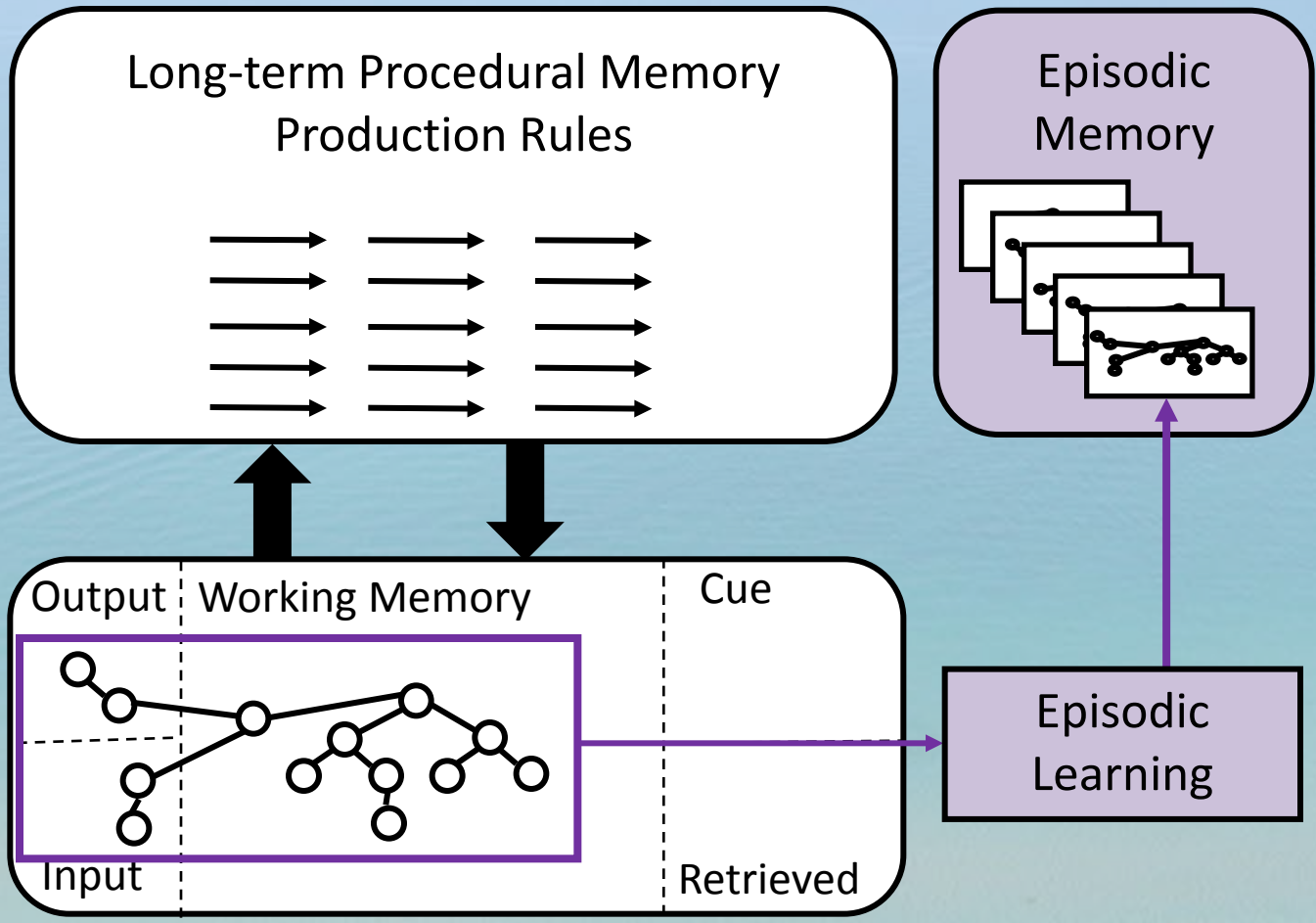


Soar Episodic Memory



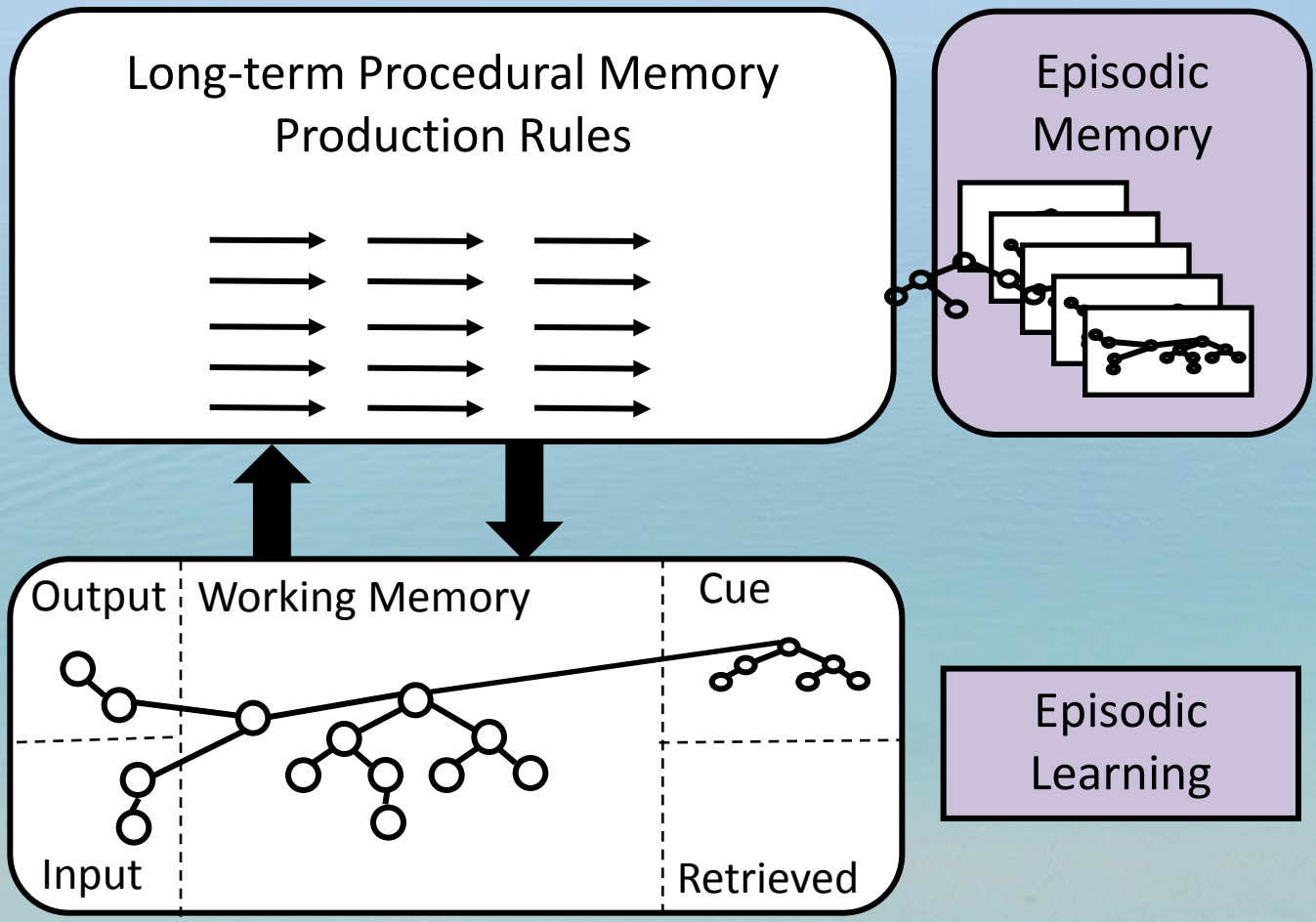
Working memory is stored in the episode

Soar Episodic Memory



Episodes are stored in a separate memory

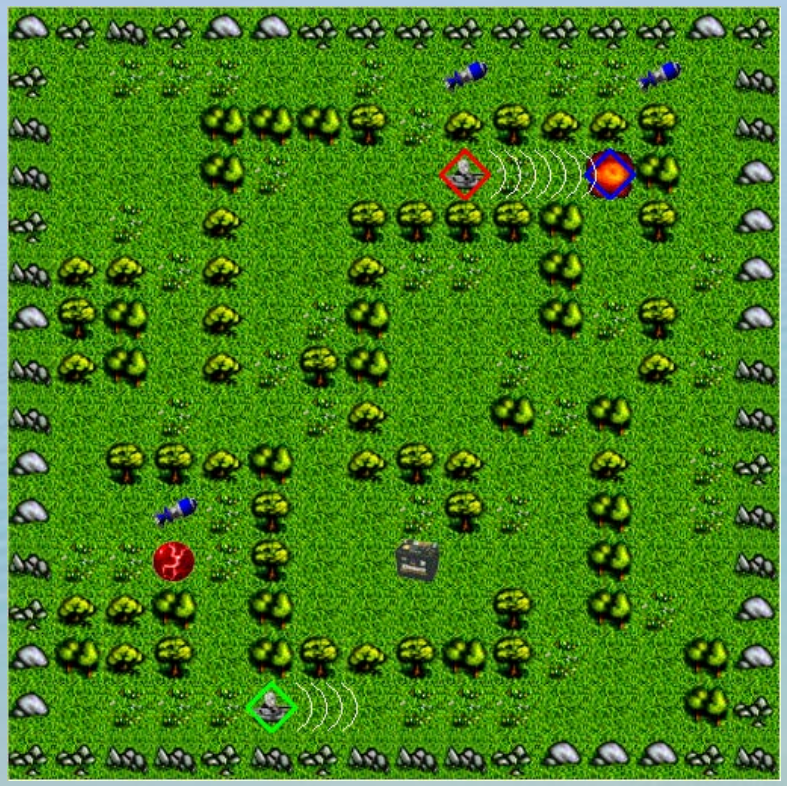
Soar Episodic Memory

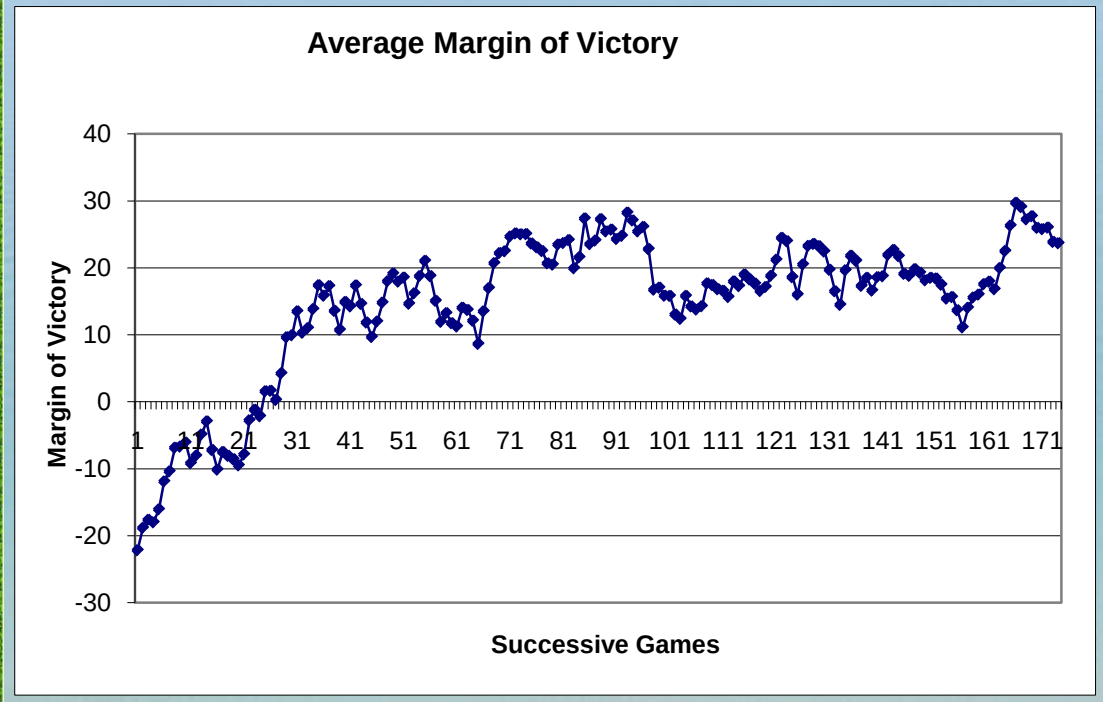


The closest partial match is retrieved.

Episodic Memory: Multi-Step Action Projection

(Andrew Nuxoll)





- Learn tactics from prior success and failure
  - Fight/flight
  - Back away from enemy (and fire)
  - Dodging

Episodic Memory Implementation
(Nate Derbinsky)

- Practical performance for real-world tasks
  - **1 million** episodes (>2,500 features/episode)

Storage	Cue Matching	Reconstruction
2.68ms, 1620MB	57.6ms	22.65ms

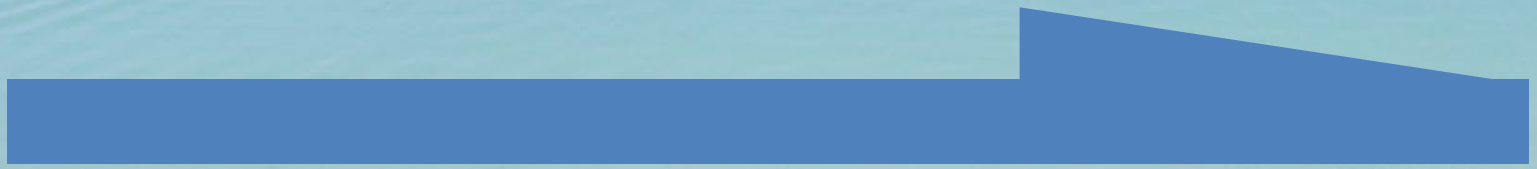
*Intel 2.8GHz Core 2 Duo, 4GB RAM, Mac OS 10.6, SQLite v3

Which ball should I drop the ball to get one in the goal?

A

B

C

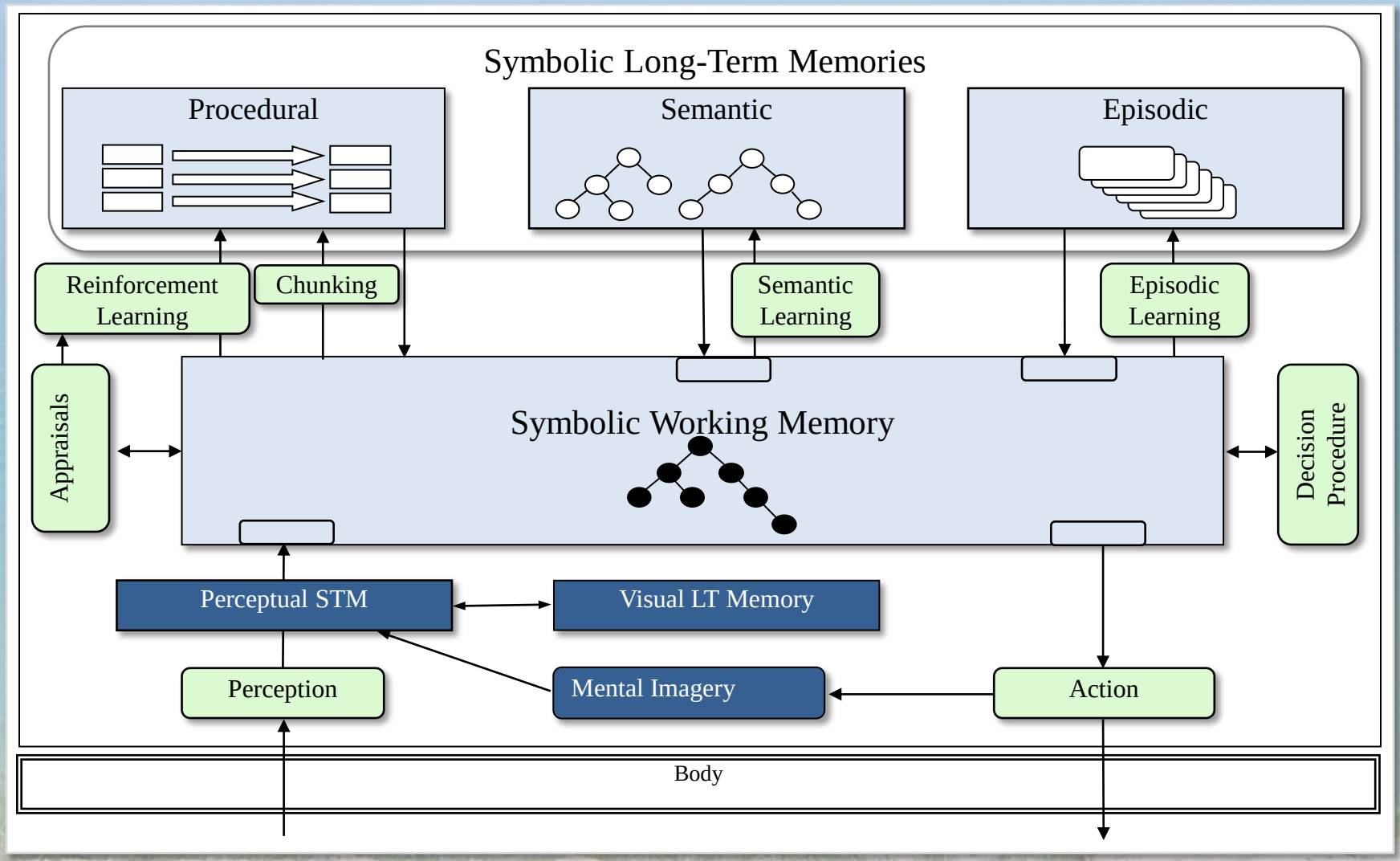


Goal

Spatial Problem Solving via Mental Imagery

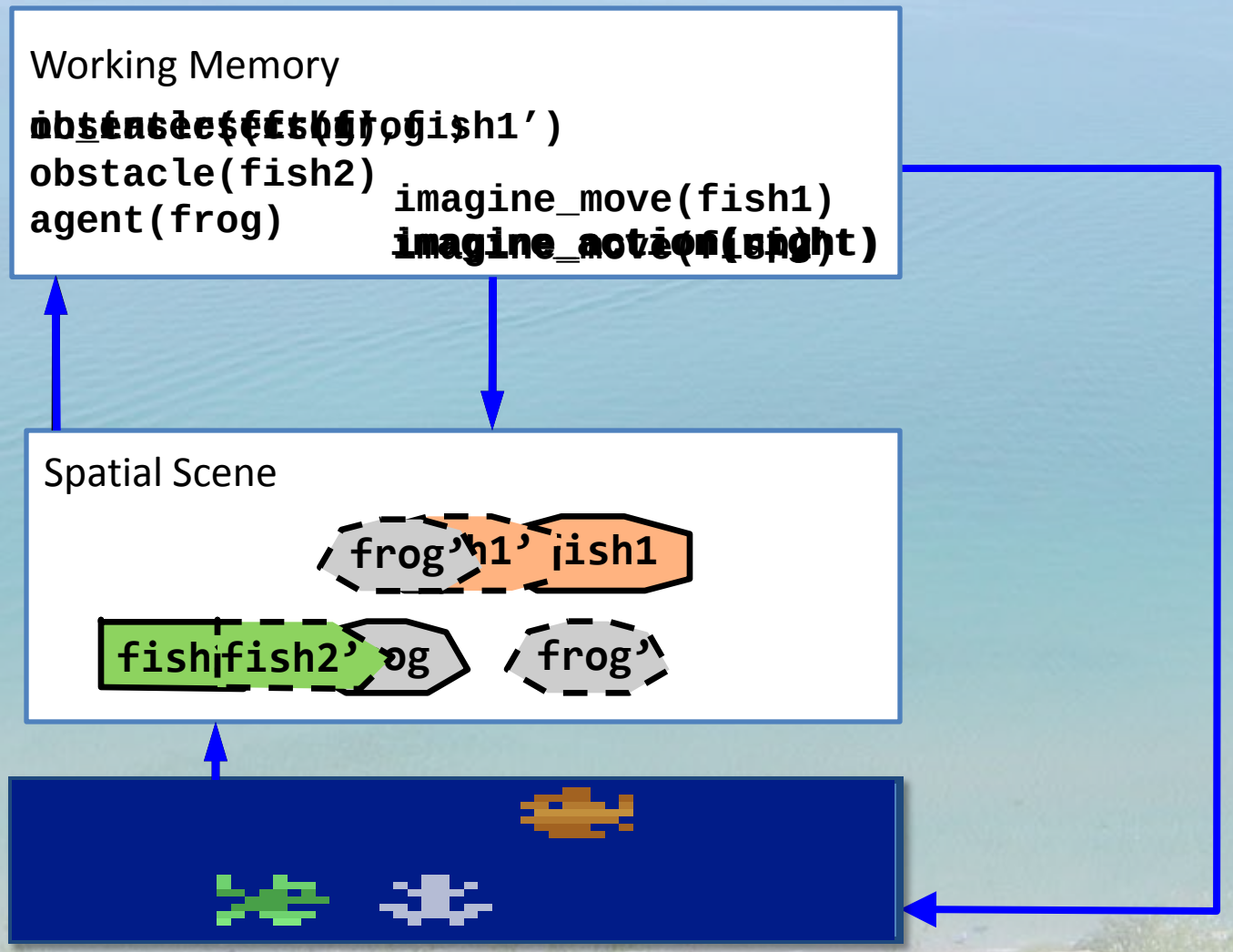
- Non-symbolic processing of images and spatial representations
- Controlled by operators (and rules) that manipulate symbolic structures

Soar 9 Structure

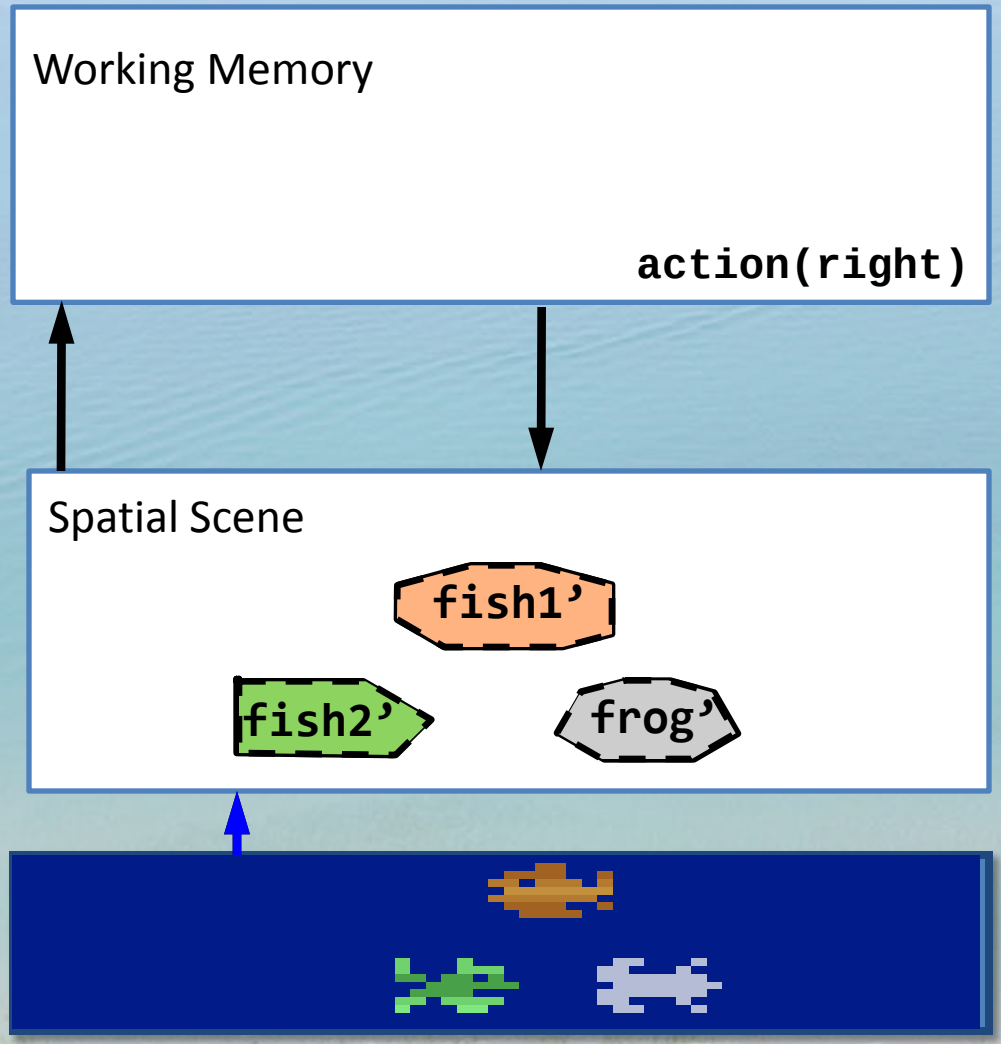


Spatial Problem Solving with Mental Imagery

[Scott Lathrop & Sam Wintermute]

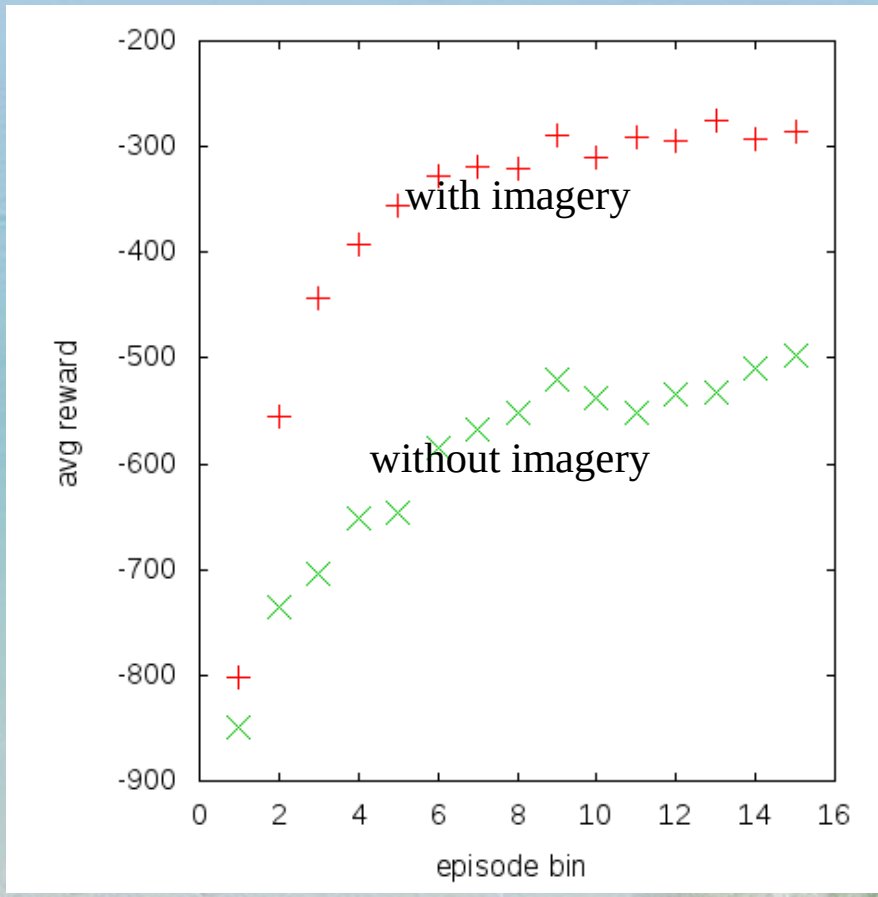


Imagery in Frogger II



Frogger II Results
[Sam Wintermute]

- Integrated with reinforcement learning using one-step look-ahead with and without imagery
- Reward function:
  - +10 for moving up a row
  - -10 for moving down a row
  - 1000 for reaching top
  - -1000 for dying
  - -1 at each time step
- Starts knowing only available operators
  - Up, down, left, right
  - Learns when to select them
- Final performance
  - Imagery agent won 70%
  - No-imagery won 45%



Summary of Knowledge/Structures

- Skills and Procedural Knowledge
  - How and when to do things
  - Rules (operators)
  - Learned by chunking and reinforcement learning
- Long-term Facts
  - What you “know”
  - Semantic memory
- Memory of past experiences
  - What you “remember”
  - Episodic memory
- Spatial and visual information processing
  - What you experience in your “mind’s eye”
  - Mental Imagery

Acknowledgments

Nate Derbinsky, Nick Gorski, Scott Lathrop, Shiwali Mohan, Shelley Nason, Andrew Nuxoll, Miller Tinkerhess, Jon Voigt, Yongjia Wang, Sam Wintermute, Joseph Xu

Soar Availability

Free: open source, BSD license

- Manuals, Tutorials, Editors, Debuggers, …
- Interfaces to C, C++, Java, Python, TCL, …

Soar (C): http://sitemaker.umich.edu/soar/home

- Supported by University of Michigan

Soar (Java): http://code.google.com/p/jsoar/

- Translated from C version and supported by David Ray