Soar's Spatial Visual System (SVS)

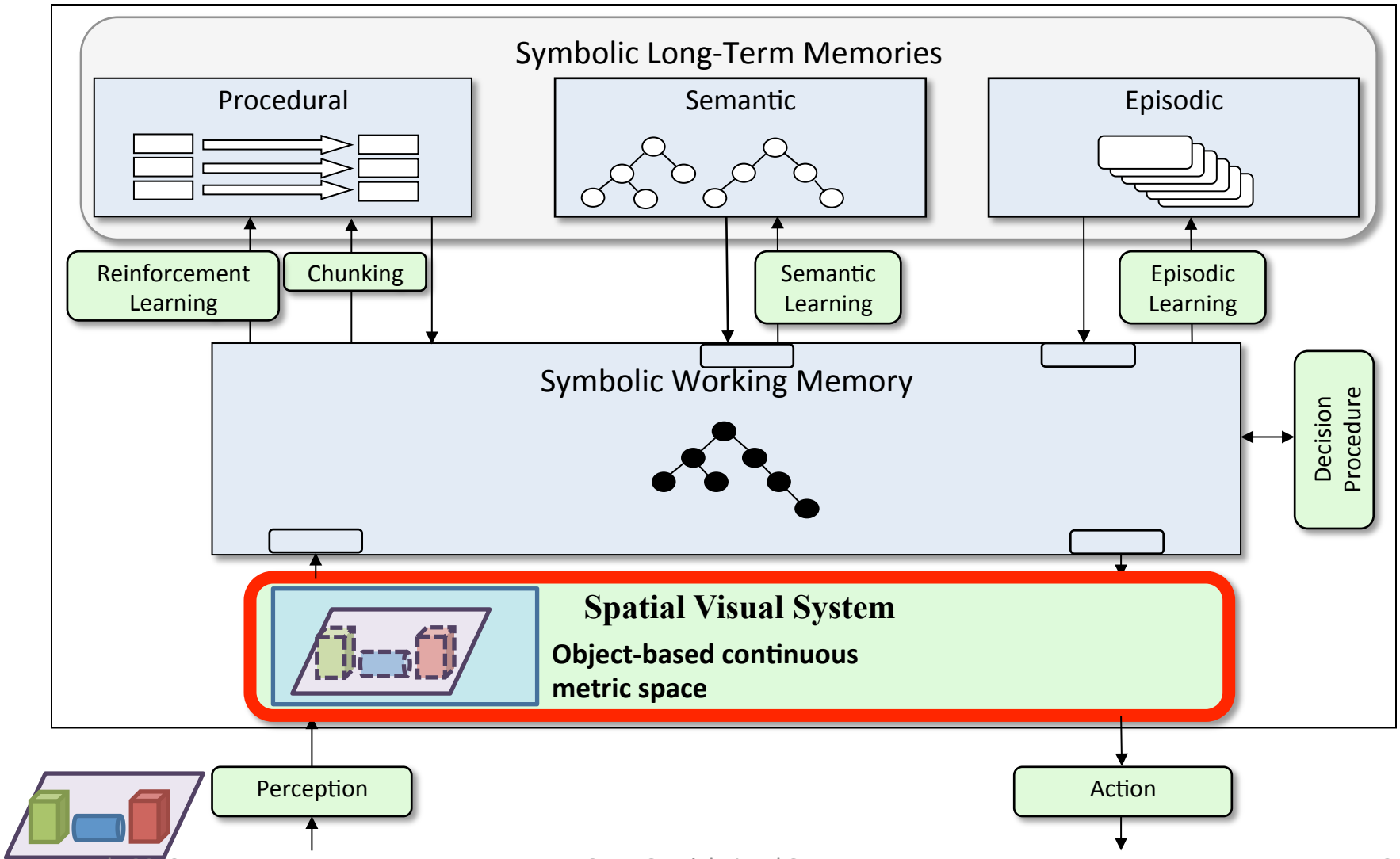
Nate Derbinsky
Disney Research, Boston

Slides and guidance from
Joseph Xu, University of Michigan

Topics

- Motivation & overview
- System components
- Example domains
  - Blocks world, Frogger
- History and future plans
- Comparison to ACT-R

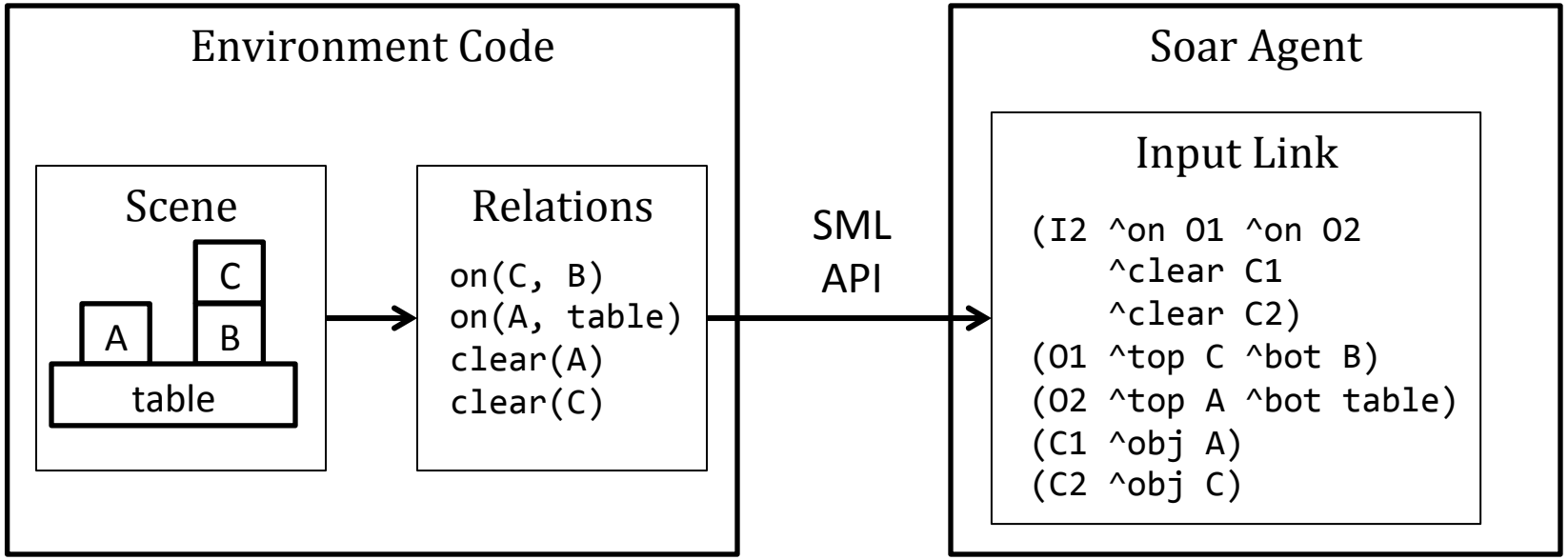
Soar 9


Symbolic Long-Term Memories
Procedural
Semantic
Episodic
Reinforcement Learning
Chunking
Semantic Learning
Episodic Learning
Symbolic Working Memory
Decision Procedure
Spatial Visual System
Object-based continuous metric space
Perception
Action

Motivation

Typical Soar Environment

- Environment reports state with task-specific representation (possibly metric)
- All available relations are reported all the time



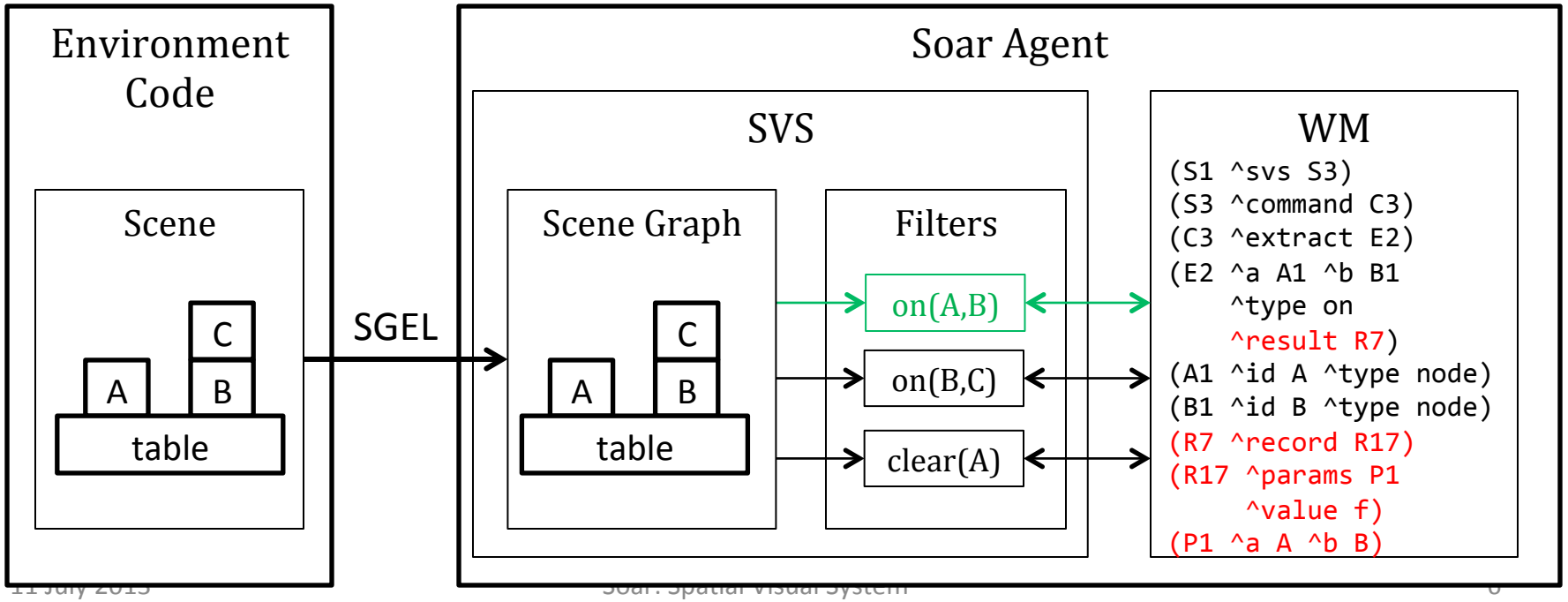
SVS Overview

- Provides a general framework for Soar to reason about continuous environments
- Environment state is represented as 3D scene
- Agent queries for spatial relationships in scene using *filters*
- Supports a working-memory interface similar to EpMem and Smem
- Supports *imagery*: hypothetical manipulations to and queries of the scene graph in substates

Result

Environment with SVS

- Environment reports state with task-independent language (Scene Graph Edit Language; SGEL)
- Agent queries only relations pertinent to reasoning
- Relations fixed across environments

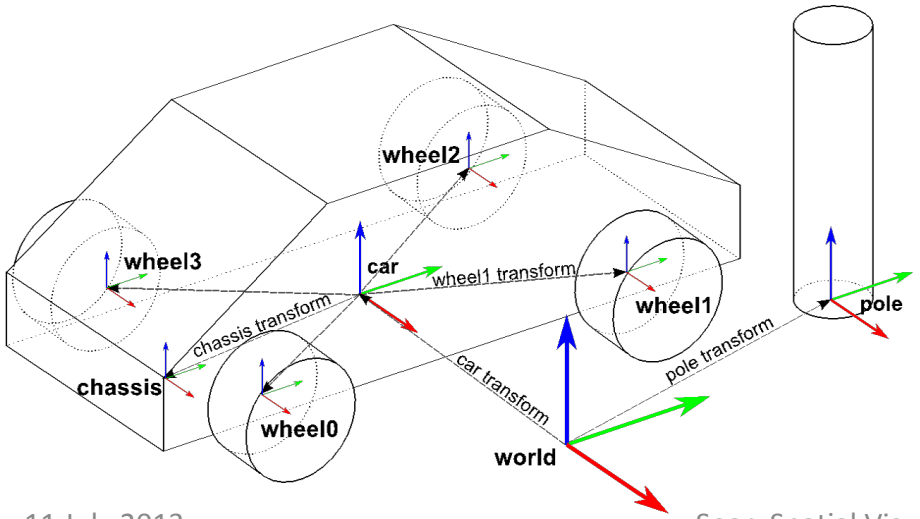


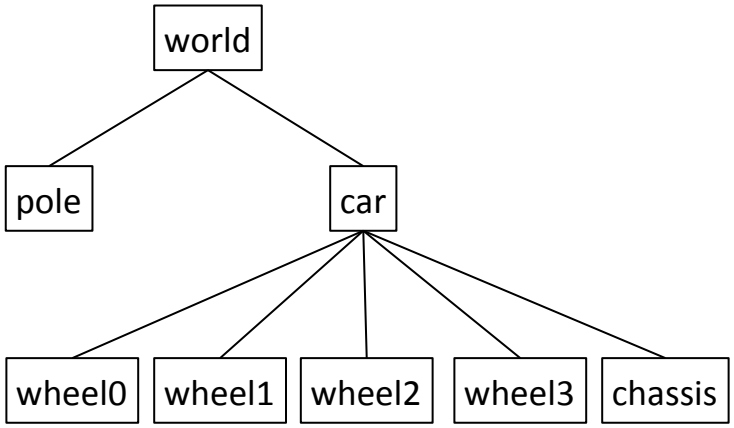
System Components

- Scene Graph
  - Edit language (SGEL)
- SVS Viewer
- Filters
  - Pipelines
- Agent interface [in Working Memory]
- Imagery

Scene Graph

- Organizes objects as tree of nodes
- Child nodes are part of the parent node
  - Group nodes
  - Geometry nodes (leaves)
- Each node has position, rotation, transform
- Transforms are accumulated from parent to child



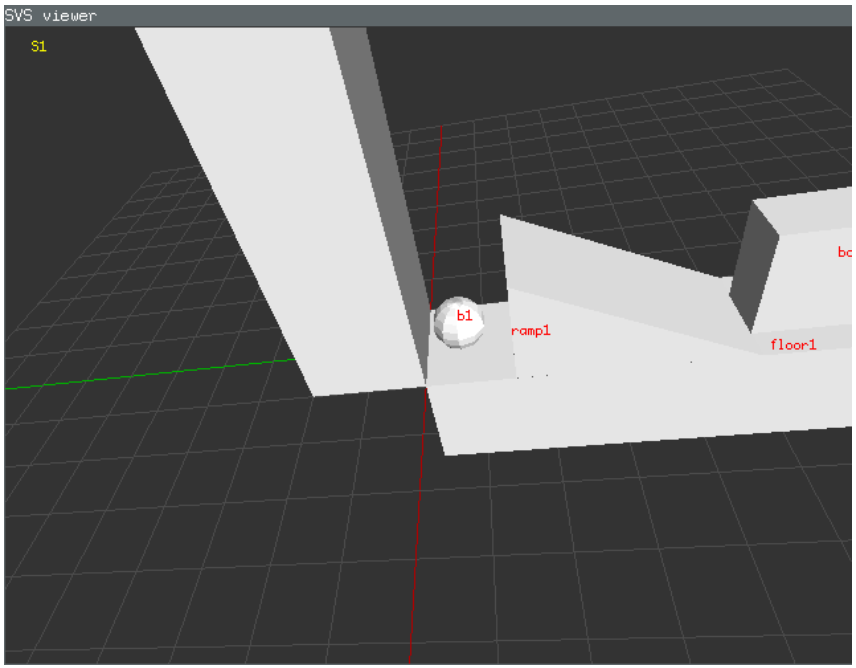


Scene Graph Edit Language (SGEL)

- `a NAME TYPE PARENT [GEOMETRY] [TRANSFORM]`
  Add object to the scene graph
- `d NAME`
  Delete object from scene graph
- `c NAME [GEOMETRY] [TRANSFORM]`
  Change object geometry and/or transform
- `p NAME PROPERTY VALUE`
  Set custom property
- Geometries
  - `b RADIUS`
  - `v X1 Y1 Z1 X2 Y2 Z2 ...`
- `Transforms`
  - `[p X Y Z] [r X Y Z] [s X Y Z]`

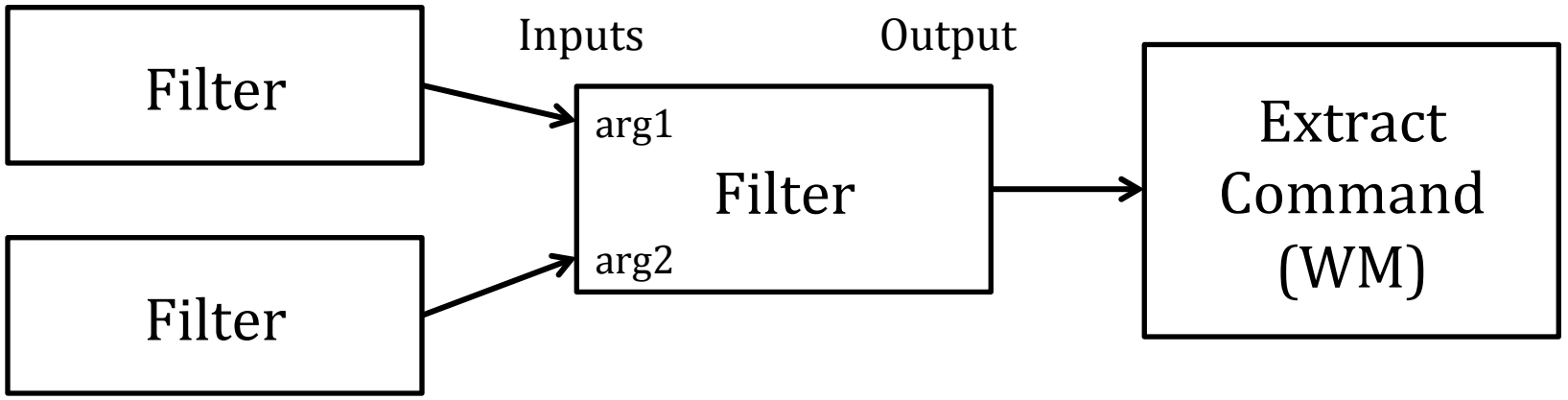
SVS Viewer

- Displays scene graph contents
- Separate debugging program that communicates with SVS via TCP sockets

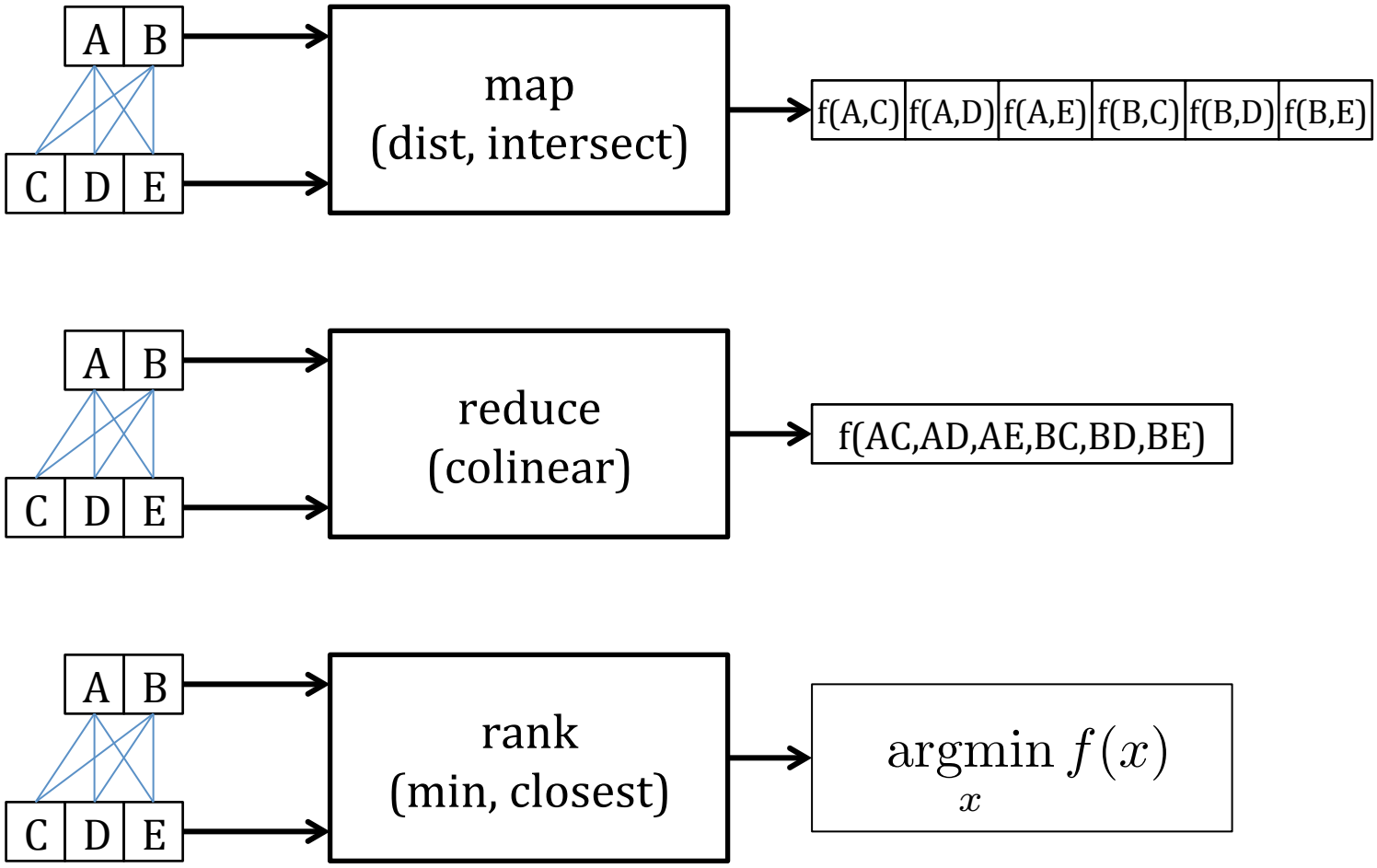


Filters

- Transforms continuous information from scene graph into symbolic information in working memory
  - Implements spatial relations, amongst other things
- Can be combined into pipeline
- Caches results and avoids re-computation when possible
- Extensible via C++ subclassing



Filter Types



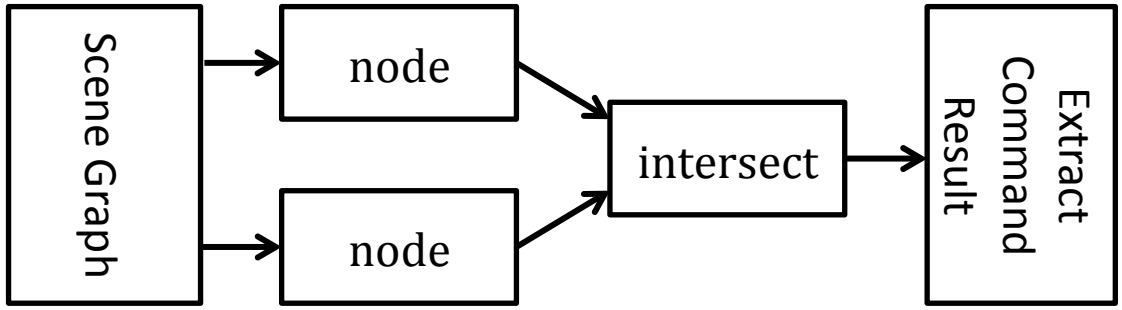
Example Filters

Filter	Parameters	Type	Output type
node	id	map	node
all_nodes	none	special	node
[xyz]-greater-than	a, b	map	boolean
[xyz]-less-than	a, b	map	boolean
[xyz]-aligned-than	a, b	map	boolean
on-top	a, b	map	boolean
intersect	a, b	map	boolean
distance	a, b	map	float
closest	a, b	rank	node
smaller-than	a, b	map	boolean

Agent Interface in Working Memory

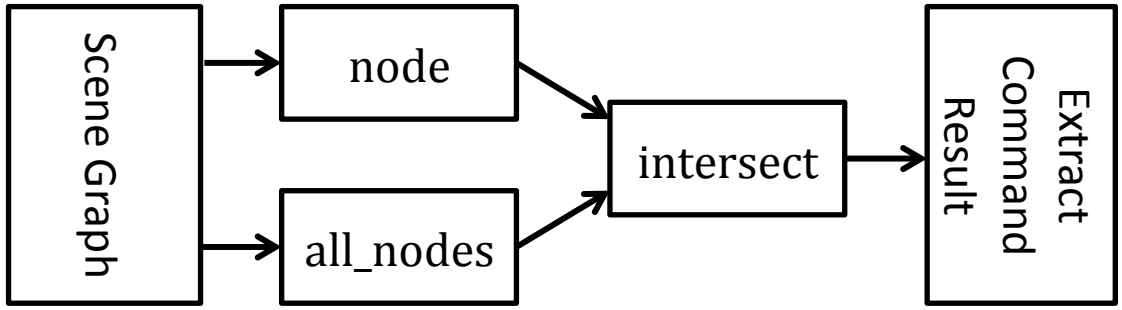
```
(S1 ^svs S3)
(S3 ^command C3 ^spatial-scene S4)
(C3 ^extract E2)
(E2 ^a A1 ^b B1 ^type intersect)
(A1 ^id b1 ^type node)
(B1 ^id b2 ^type node)
```

```
(S1 ^svs S3)
(S3 ^command C3 ^spatial-scene S4)
(C3 ^extract E2)
(E2 ^a A1 ^b B1 ^type intersect ^result R7
    ^status success)
(A1 ^id b1 ^type node ^status success)
(B1 ^id b2 ^type node ^status success)
(R7 ^record R17)
(R17 ^params P1 ^value f)
(P1 ^a b1 ^b b2)
```



Example Filter Pipeline

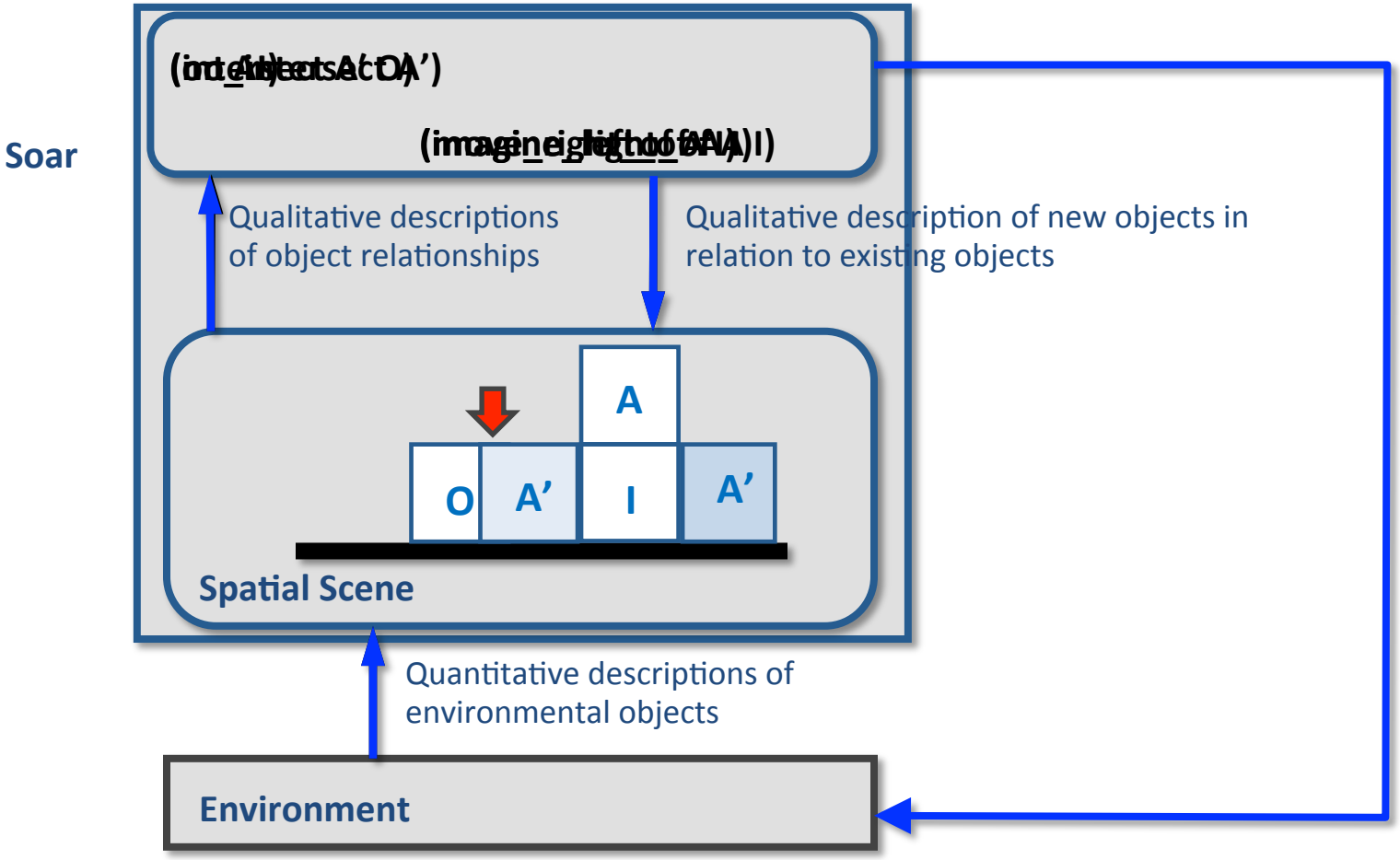
```
sp {make-filter2
   (state <s> ^superstate nil
              ^svs.command <c>)
-->
   (<c> ^extract <e>)
   (<e> ^type intersect ^a <a> ^b <b>)
   (<a> ^type node ^id b1)
   (<b> ^type all_nodes)}
```



Imagery

- Each substate contains an independent copy of the superstate scene graph
- The `add_node` command adds nodes to the scene graph
- The `property` command changes the properties of a node, such as its position

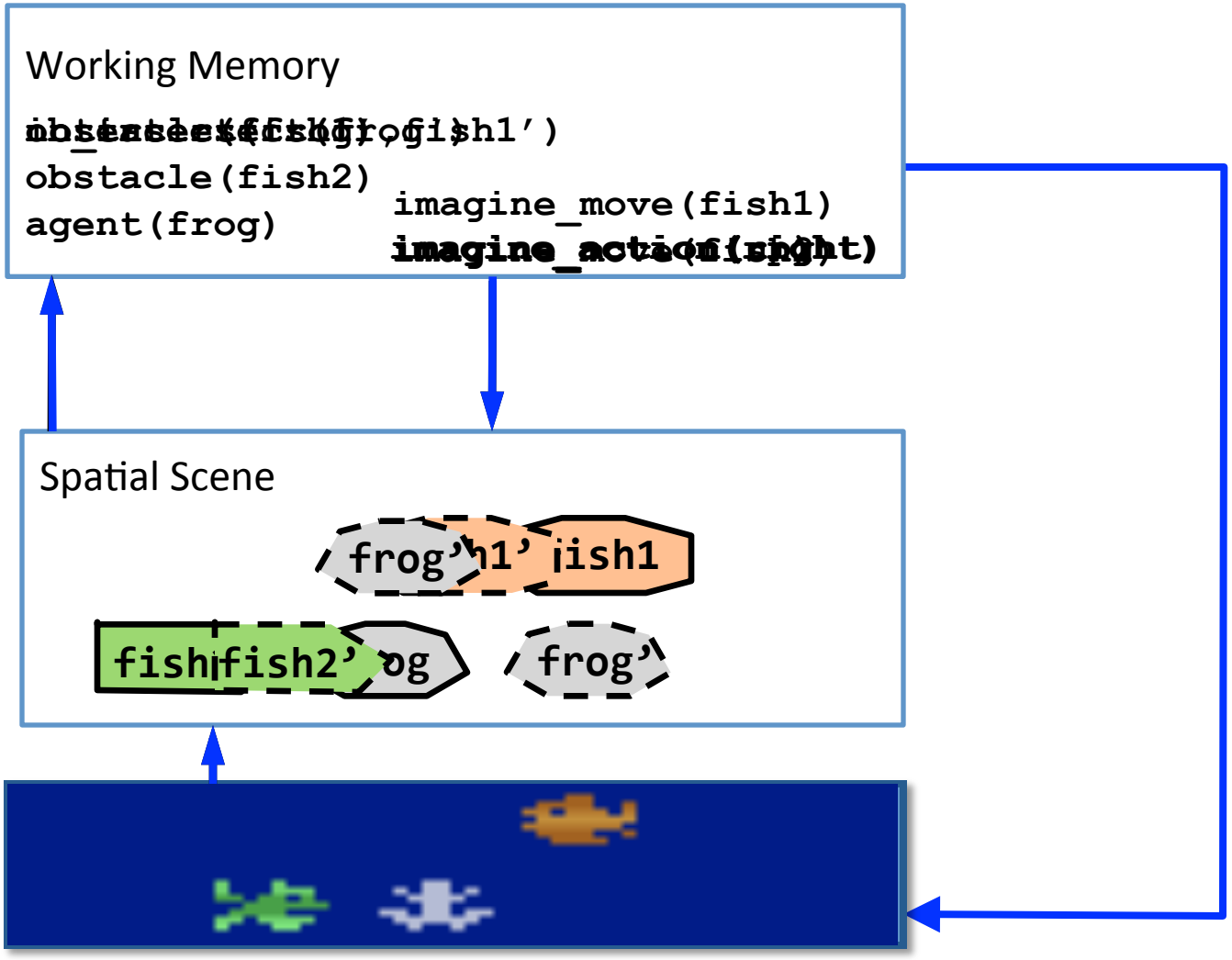
Spatial Problem Solving via Mental Imagery



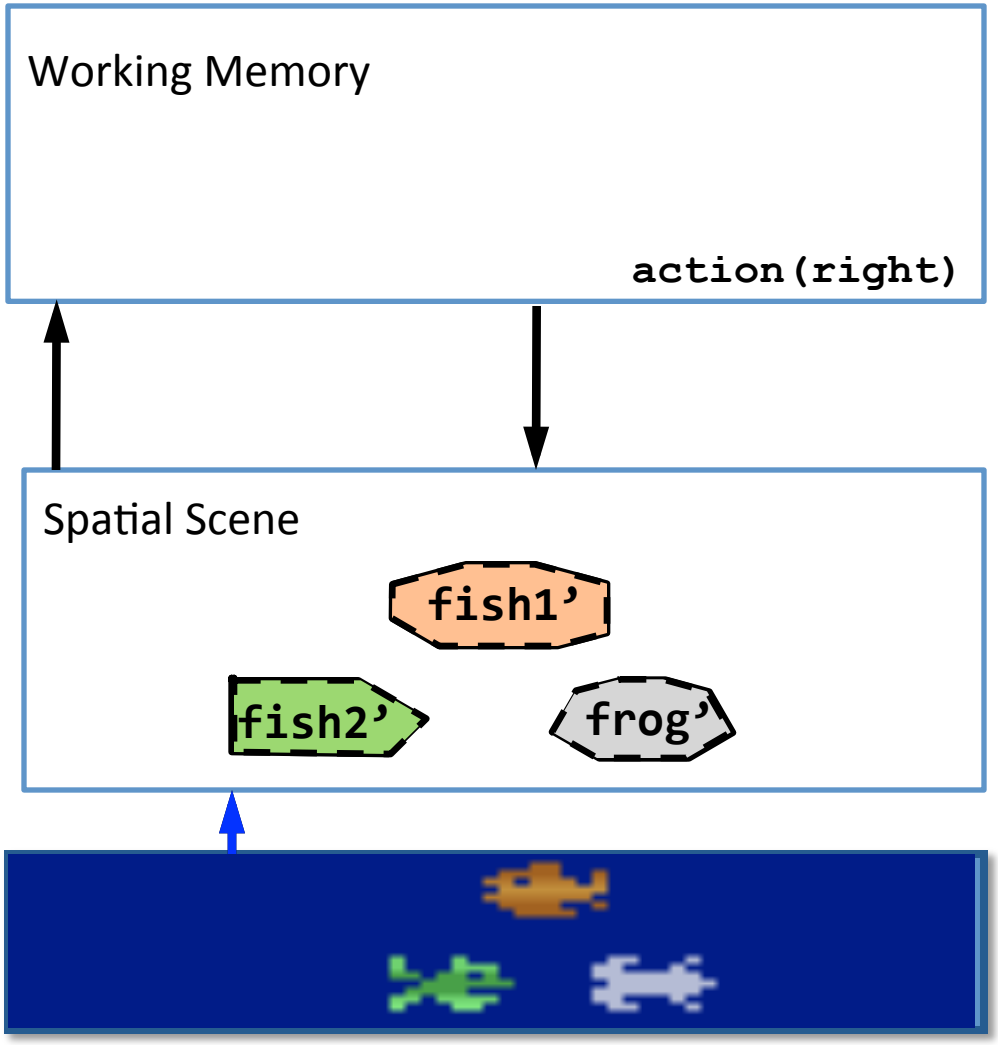
Frogger II Domain

- Imagery supports better performance with a given perceptual system
  - Dynamics are simulated in imagery
  - Perception is applied to predicted state
- Goal is to navigate from the bottom to the top
- Obstacles are moving, a current pulls the frog to the right
- Learns to control frog using RL (no task-specific control knowledge)

Imagery in Frogger II



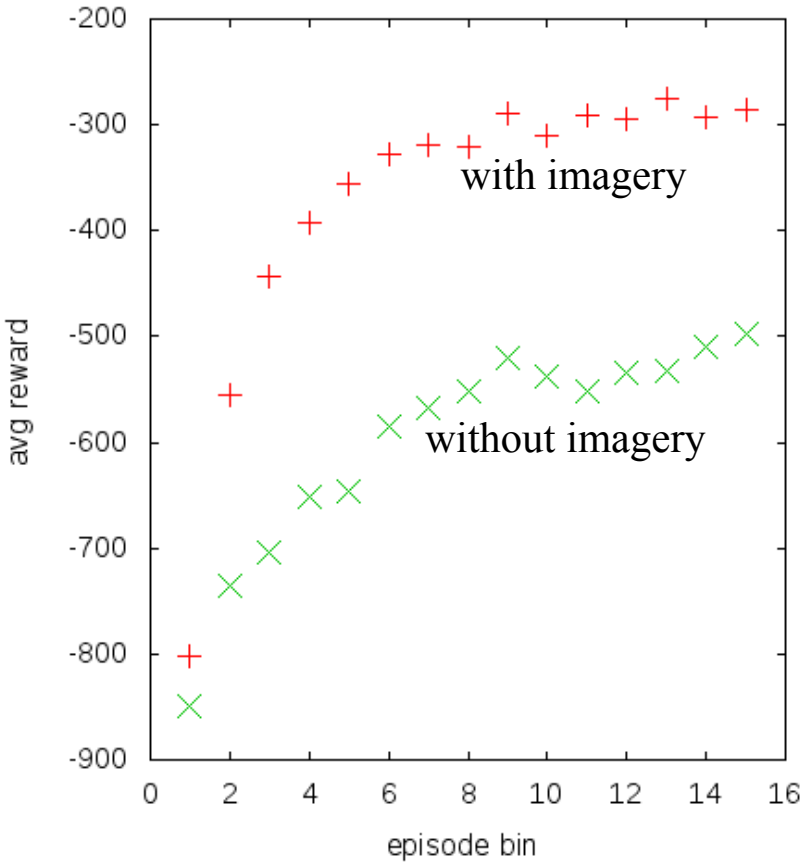
Imagery in Frogger II



Frogger II Results

- Integrated with RL using one-step look-ahead with and without imagery
- Same abstract state representation
- 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

- Final performance (no exploration)
  - Imagery agent won 70%
  - No-imagery won 45%

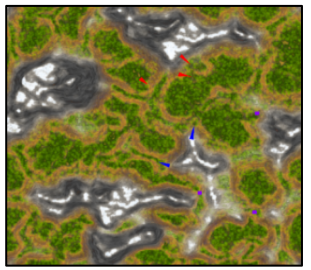


Wintermute, S. (2010). Using Imagery to Simplify Perceptual Abstraction in Reinforcement Learning Agents. Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10), Atlanta, Georgia

Some History

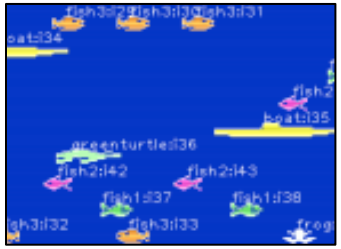
Soar Visual Imagery (SVI). Performed operations on pixel representations.

Lathrop, S.D., and Laird, J.E. (2007). Towards Incorporating Visual Imagery into a Cognitive Architecture. Proceedings of the Eighth International Conference on Cognitive Modeling. Ann Arbor, MI.



Scott Lathrop

Sam Wintermute



SVS Theory

Wintermute, S. Imagery in Cognitive Architecture: Representation and Control at Multiple Levels of Abstraction. Cognitive Systems Research, 19-20,1-29.

Future Plans

- Implemented the current version of SVS
- Dissertation on learning integrated symbolic and continuous action models for continuous domains

Xu, J. Z., Laird, J. E. (2013). Learning Integrated Symbolic and Continuous Action Models for Continuous Domains. Proceedings of the 27th AAAI Conference on Artificial Intelligence. Bellevue, WA.

Joseph Xu

SVS Beta release coming later this year!

Comparison to Perception/Motor Modules in ACT-R

- Does not model timing/details of human sensing (e.g. saccades, attention, tracking) or movement (e.g. Fitts's law)
- Does not impose an abstraction layer of "devices" (e.g. keyboard/mouse; could be done via environment)
- Does not interact directly with long-term memory (e.g. SMem, EpMem)
- Is not limited/constrained to human-specific spatial/visual features (more concerned with functionality, real-time performance)

Thank You :)

Questions?