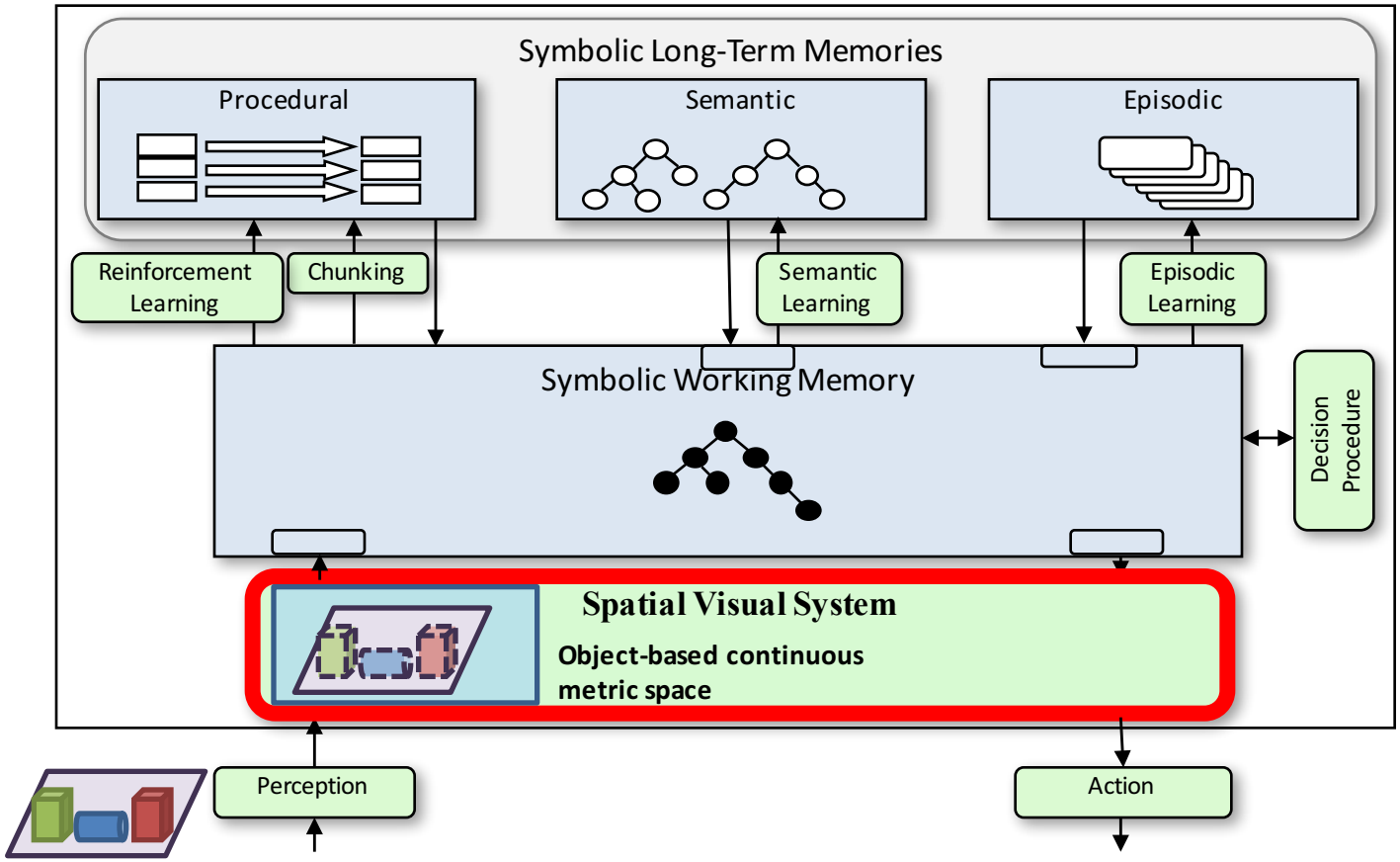
Spatial Visual System (SVS)
IJCAI 2016

Nate Derbinsky

Agenda

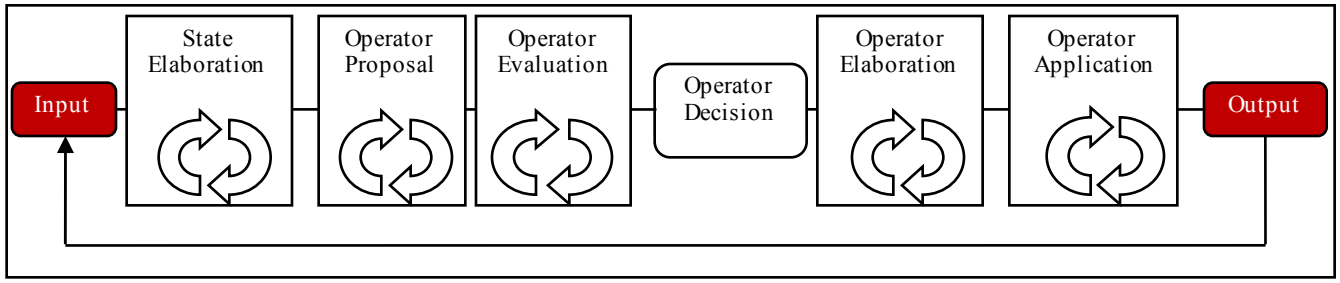
- Motivation & overview
- System components

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

Soar Basic Functions

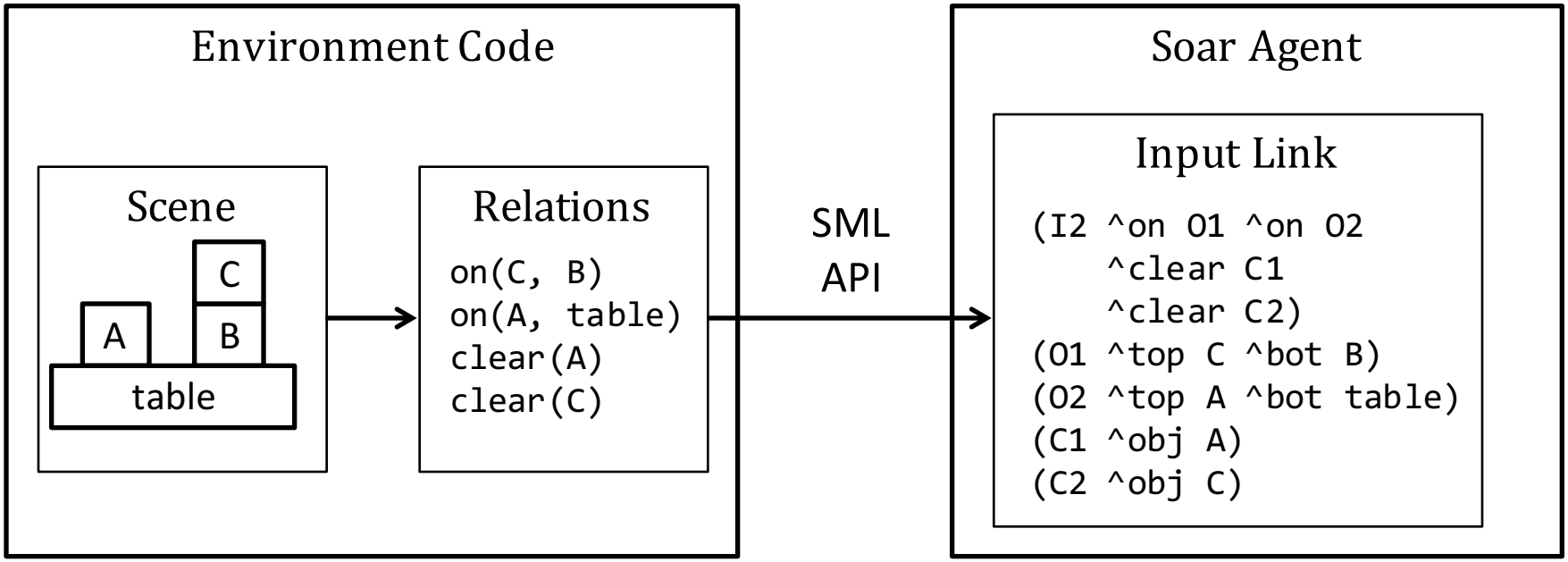
1. Input from environment
2. Elaborate current situation: *parallel rules*
3. Propose operators via acceptable preferences
4. Evaluate operators via *preferences: Numeric indifferent preference*
5. Select operator
6. Apply operator: Modify internal data structures: *parallel rules*
7. Output to motor system [and access to long-term memories]



Motivation

Typical Soar Environment

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



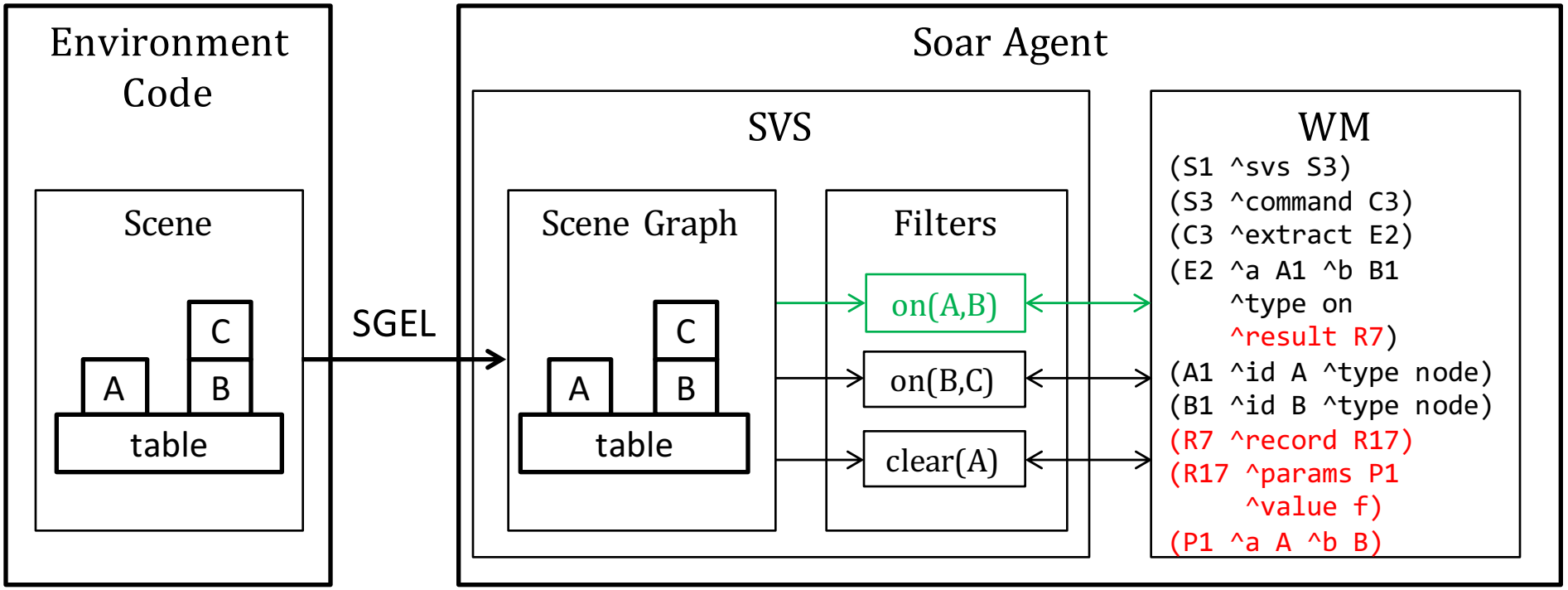
SVS Overview

- 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

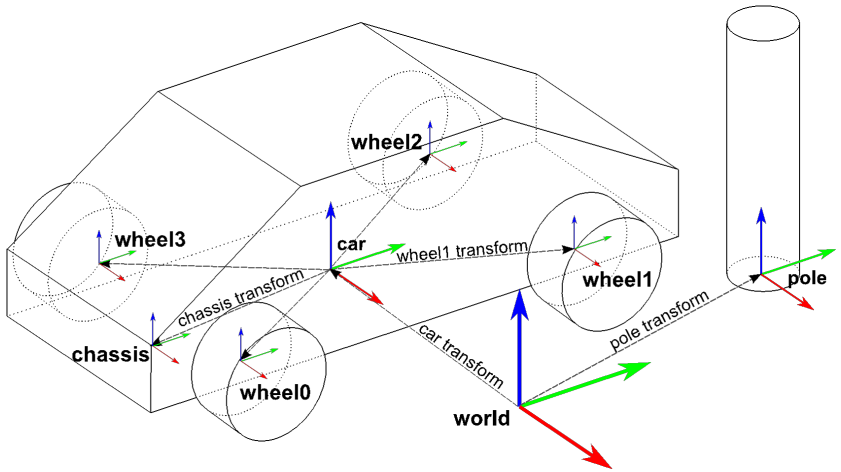
- Environment reports state with task-independent language (Scene Graph Edit Language; SGEL)
- Agent queries only relations pertinent to reasoning
- Relations fixed across environments

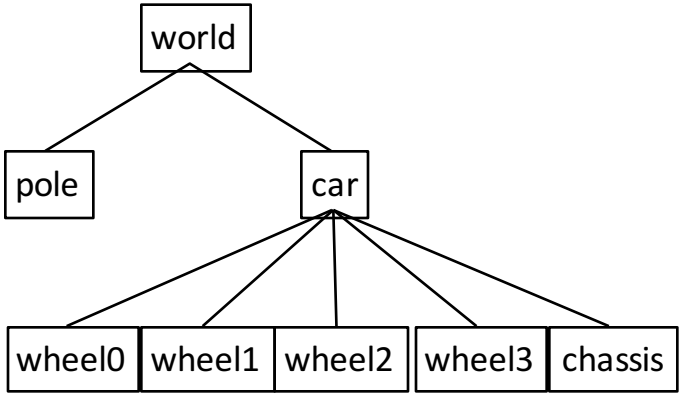


Scene Graph Representation

- 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

```
(S1 ^svs S3)
(S3 ^command C3 ^spatial-scene S4)
(S4 ^id world ^child C1 ^child C2)
(C1 ^id pole)
(C2 ^id car ^child C3 ^child C4
    ^child C5 ^child C6 ^child C7)
(C3 ^id wheel0)
(C4 ^id wheel1)
(C5 ^id wheel2)
(C6 ^id wheel3)
(C7 ^id chassis)
```





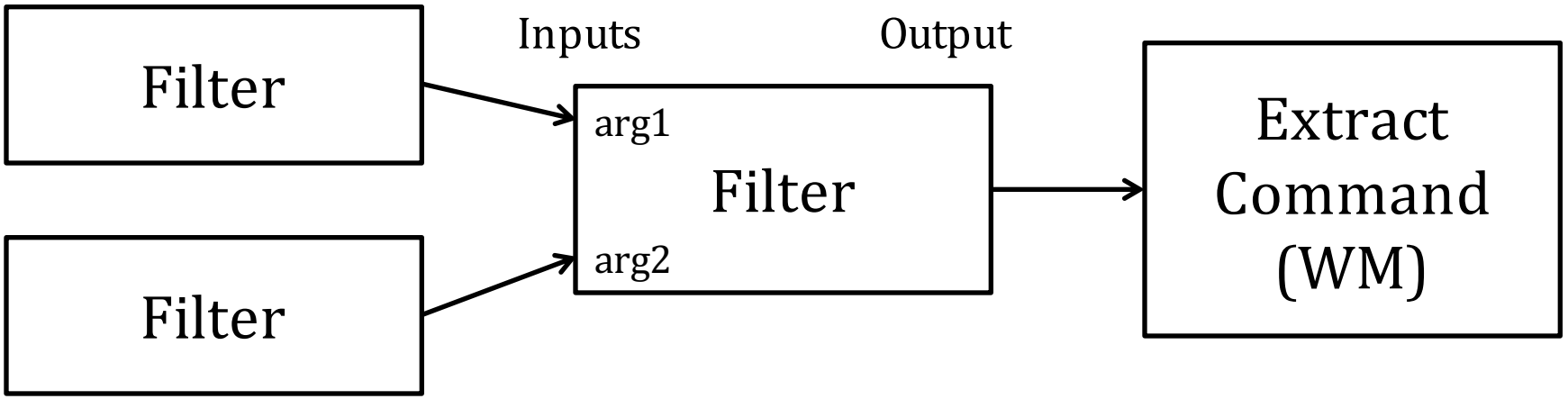
Scene Graph Edit Language

```
add <id> <parent> [GEOM] [TRANS]
change <id> [TRANS]
delete <id>
tag add|change|delete <id> <tag_name> <tag_val>

GEOMETRY:
   ball <rad>
   vertices x1 y1 z1 x2 y2 z2 ...
TRANSFORM
   pos x y z, rot x y z, scale x y z
```

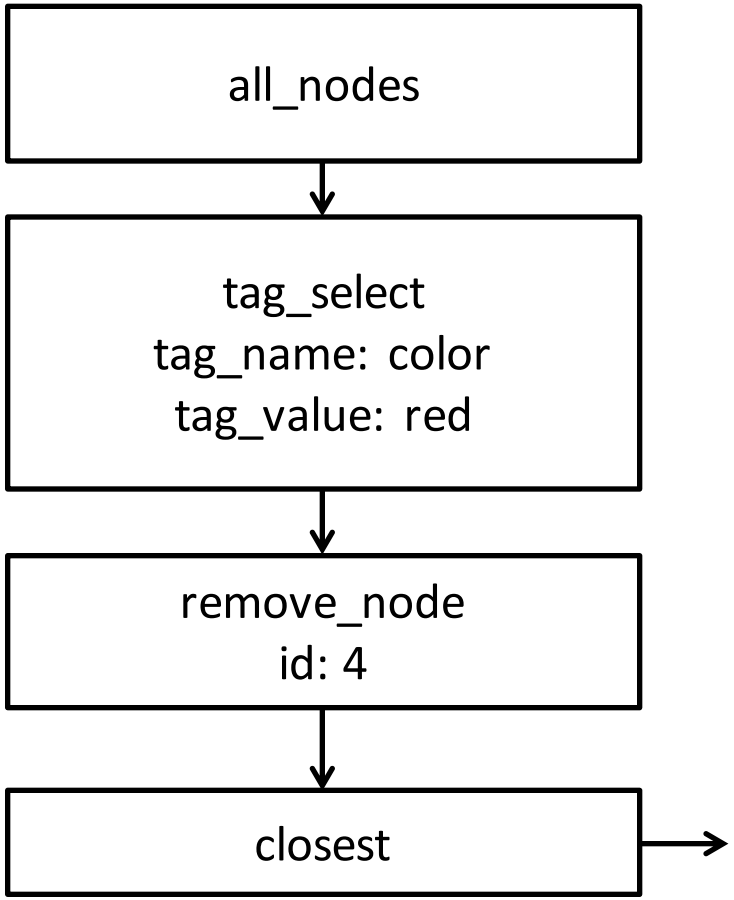
Filters

- 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



SVS Examples

Other than object 4, what is the closest red object?




```
(state <s> ^svs.command <cmd>)
(<cmd> ^extract <e>)
  (<e> ^type closest
       ^a <rem>)
    (<rem> ^type remove_node
           ^id 4
           ^a <tsel>)
      (<tsel> ^type tag_select
              ^tag_name color
              ^tag_value red
              ^a <all>)
          (<all> ^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