Soar-SMem: A Public Pilot

Nate Derbinsky
University of Michigan

What is Semantic Memory?

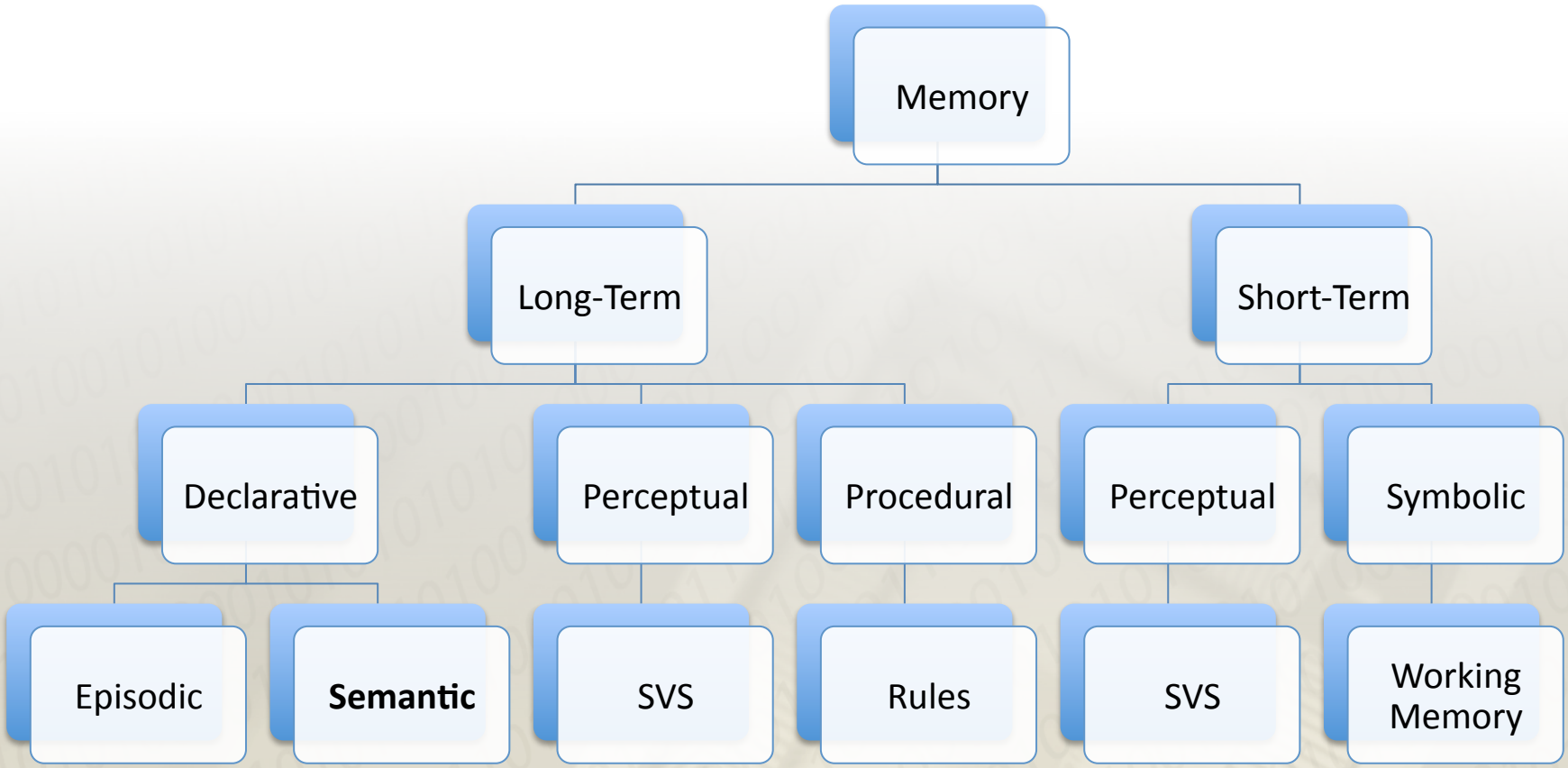
"Semantic memory refers to a person's knowledge about the world.

It encompasses a wide range of organized information, including facts, concepts, and vocabulary.

Semantic memory can be distinguished from episodic memory by virtue of its lack of association with a specific learning context."

-Tulving

Memory Organization



Semantic Memory in ACT-R

- Declarative memory module serving as semantic store for tasks
  - Used frequently by procedural knowledge to populate limited working memory (buffers)
- Foundational application of rational analysis
  - Used to model rich psychological phenomena

Semantic Memory in Soar

- Wang, Y., and Laird, J.E. 2007. Integrating Semantic Memory into a Cognitive Architecture.
  - Technical report & research branch

- Episodic Memory
  - Thinking about the effects of a large working memory over long agent lifetimes
  - Developing efficient, re-usable LTM code

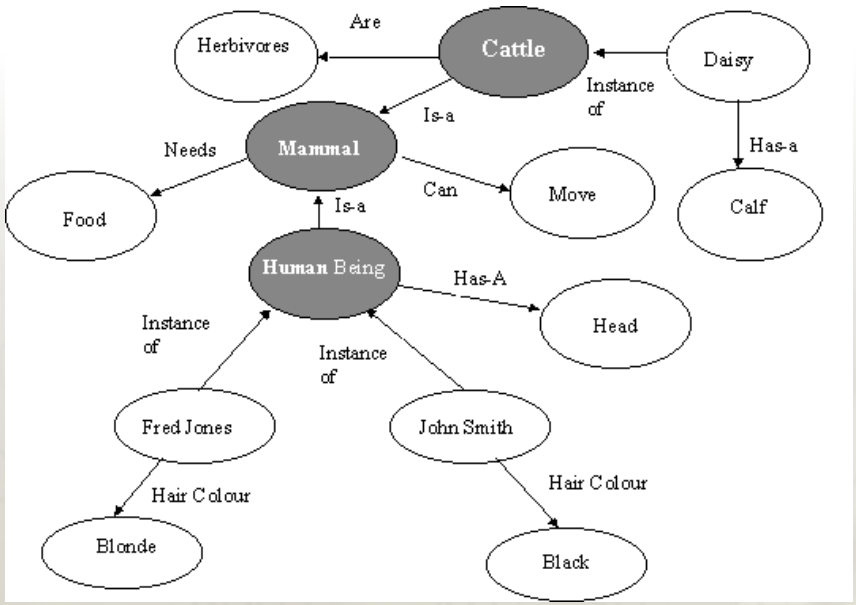
- Chicken-and-Egg
  - Without a *working system*, hard to develop motivating *tasks* (especially with Soar's "limitless" working memory)
  - Without *tasks*, hard to develop *principled requirements* for a working system

Pilot: Guiding Principles

- Performance
  - Fast
  - Scalable
- Implementation
  - Simple (for fast prototyping)
  - Flexible (to explore variants)

"Make it faster and easier to use than hand-coded working memory solutions."
–Bob Marinier

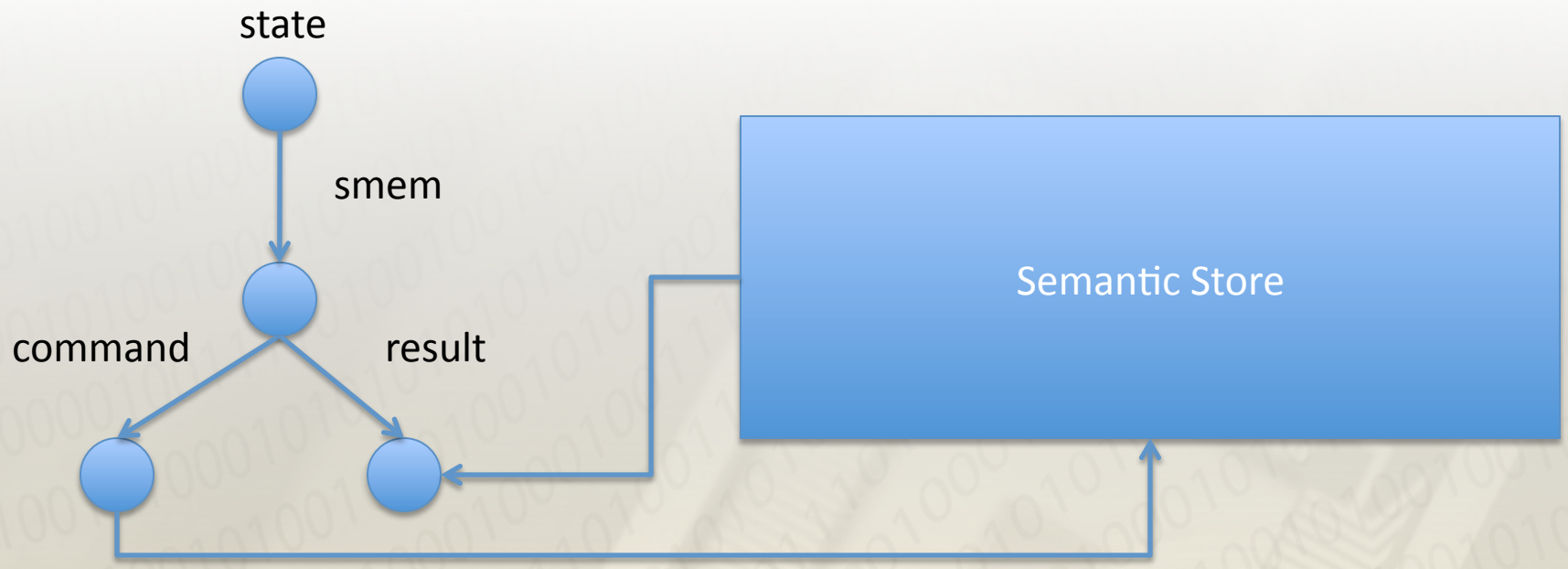
Semantic Store



- Hierarchical association of concepts (or “chunks,” ala ACT)
- Very similar arrangement to WMEs in Working Memory

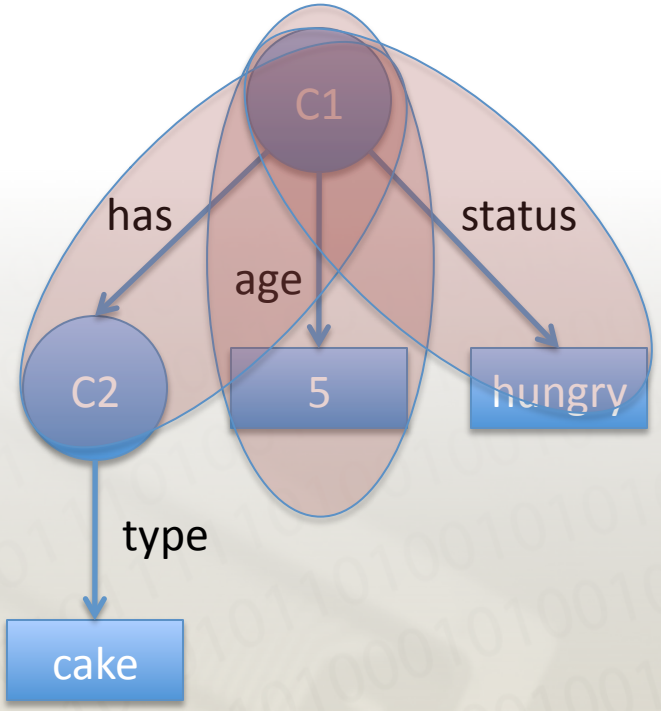
Soar-SMem

- Agents interact with an **smem** structure on each state (ala Soar-EpMem)



Agent Storage

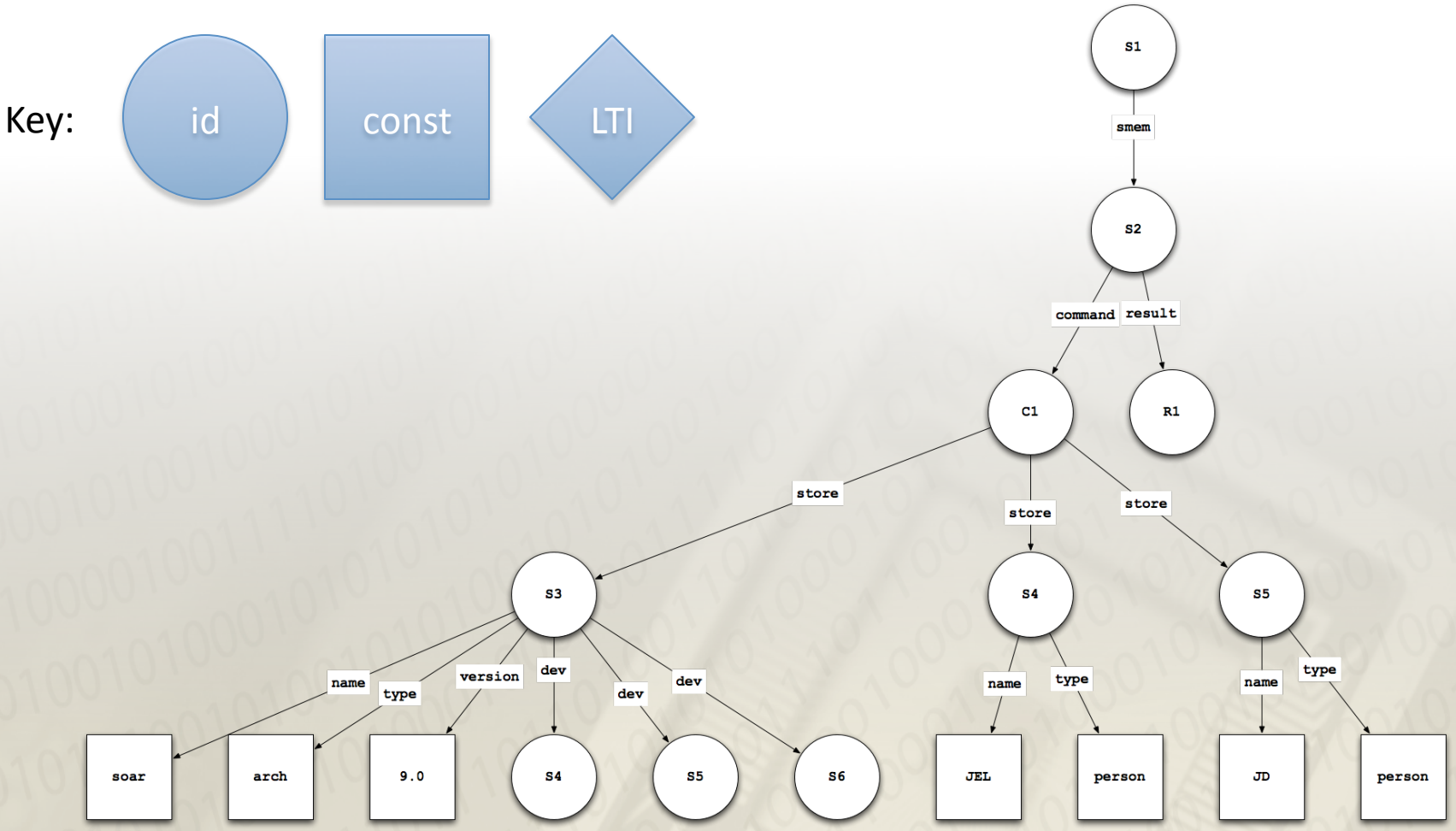
- Deliberate
  - `state.smem.command.store <id>`
- Multiple **store** commands can be issued in a single cycle
  - Storage takes place during Output phase and is guaranteed to succeed
- Stores WMEs rooted at `id`
  - Supports multi-valued attributes



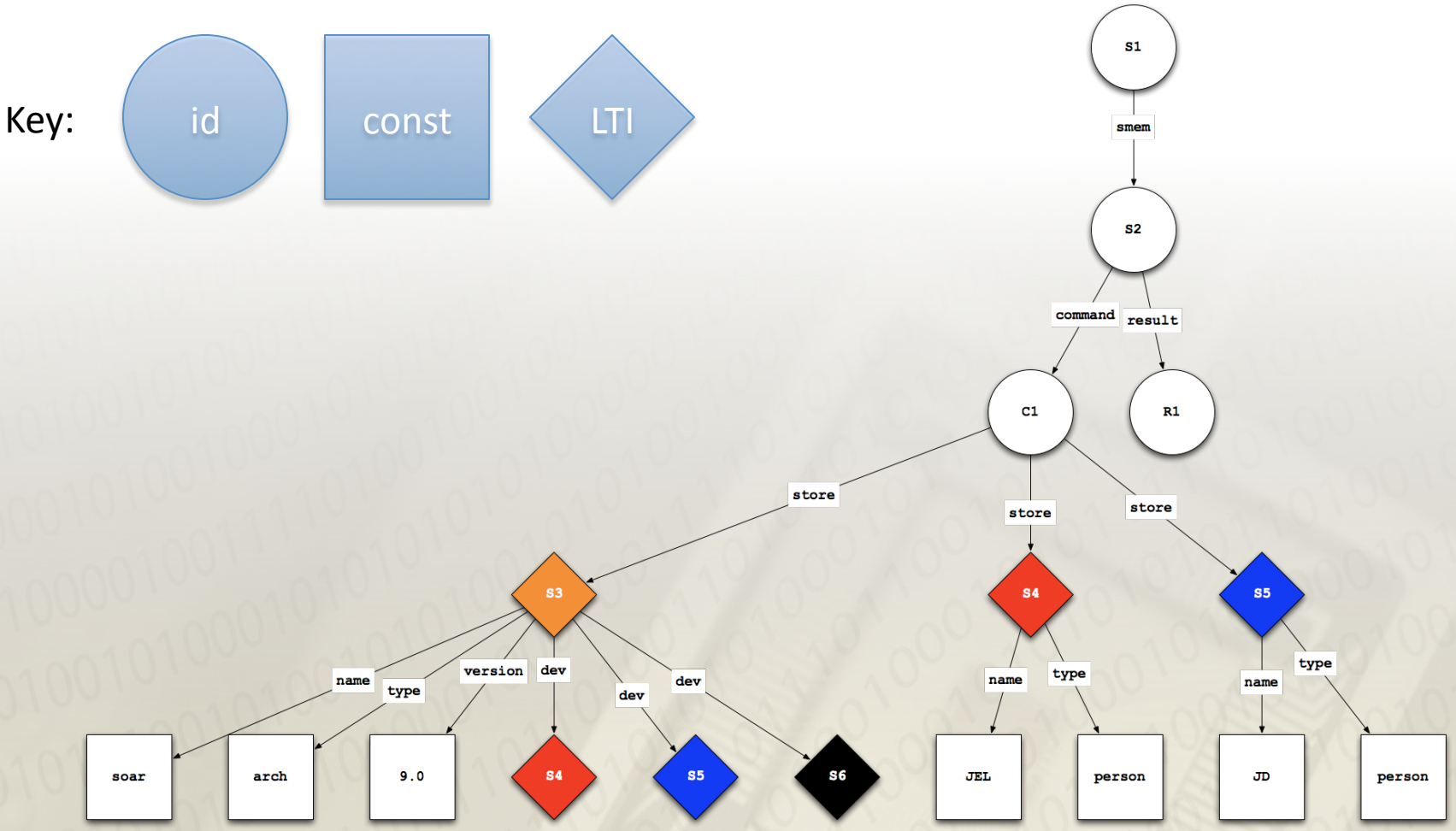
Long-Term Identifiers

- When an identifier is stored in semantic memory, it is converted into a *long-term identifier* (LTI)
  - Letter-number combination (i.e. C1 or C2) is permanently associated with the stored concept
- Subsequent storage of an LTI will overwrite previous contents within semantic memory
- Between *store* commands, it is possible to have the augmentations of an LTI in *working* memory be inconsistent those in *semantic* memory
  - Simple implementation
  - Allows for "imagining" concept variations

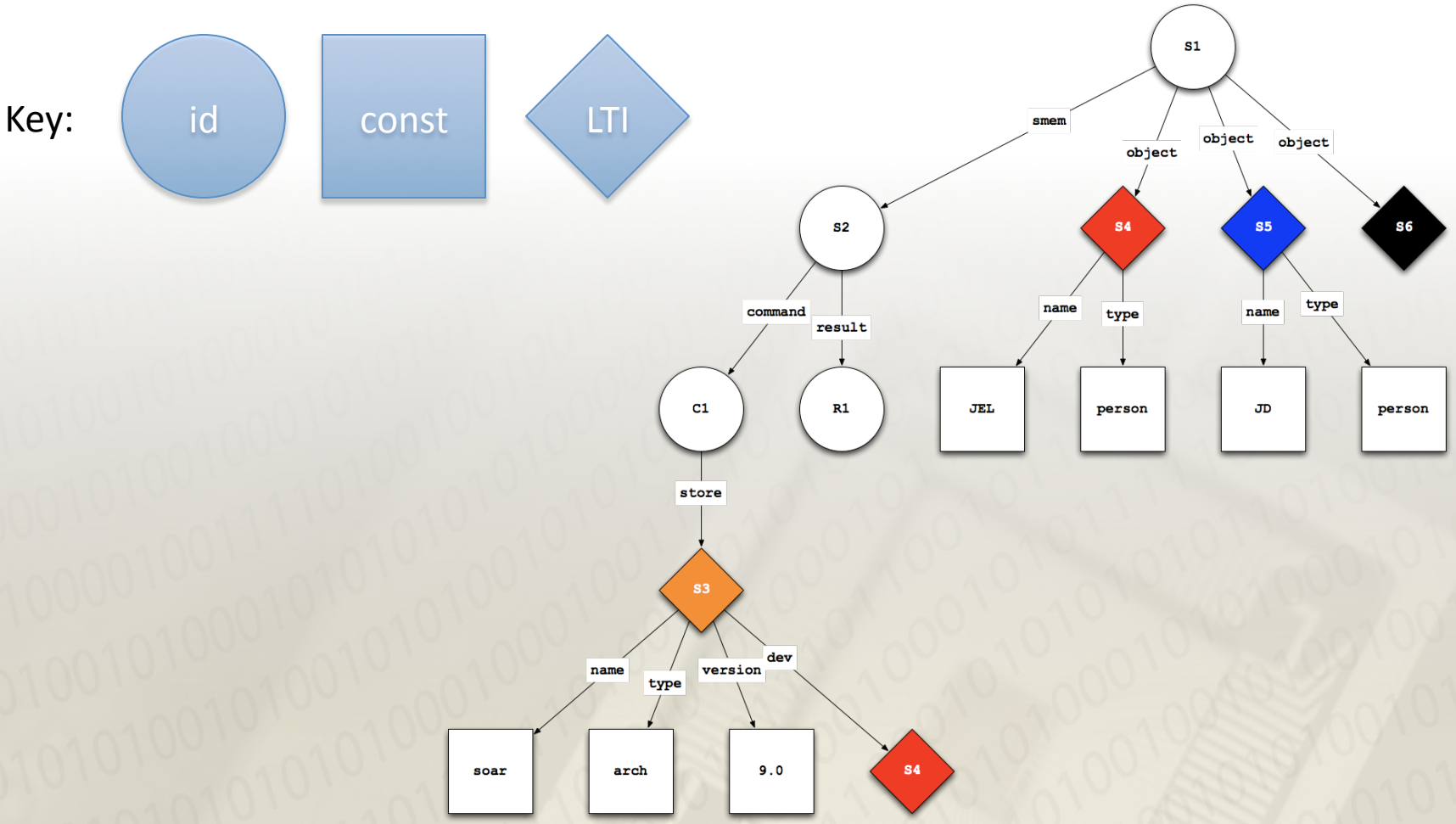
Storage Example



Storage Example: Result



Storage Example: Later…



Manual Storage

- Concepts can be easily loaded into the semantic store using the **add** switch of the **smem** command:

```
smem –add {
   (<arithmetic> ^add10-facts <a01> <a02> <a03>)
   (<a01> ^digit1 1 ^digit-10 11)
   (<a02> ^digit1 2 ^digit-10 12)
   (<a03> ^digit1 3 ^digit-10 13)
}
```

Uses syntax nearly identical to rule RHS, including dot notation (but no RHS functions)

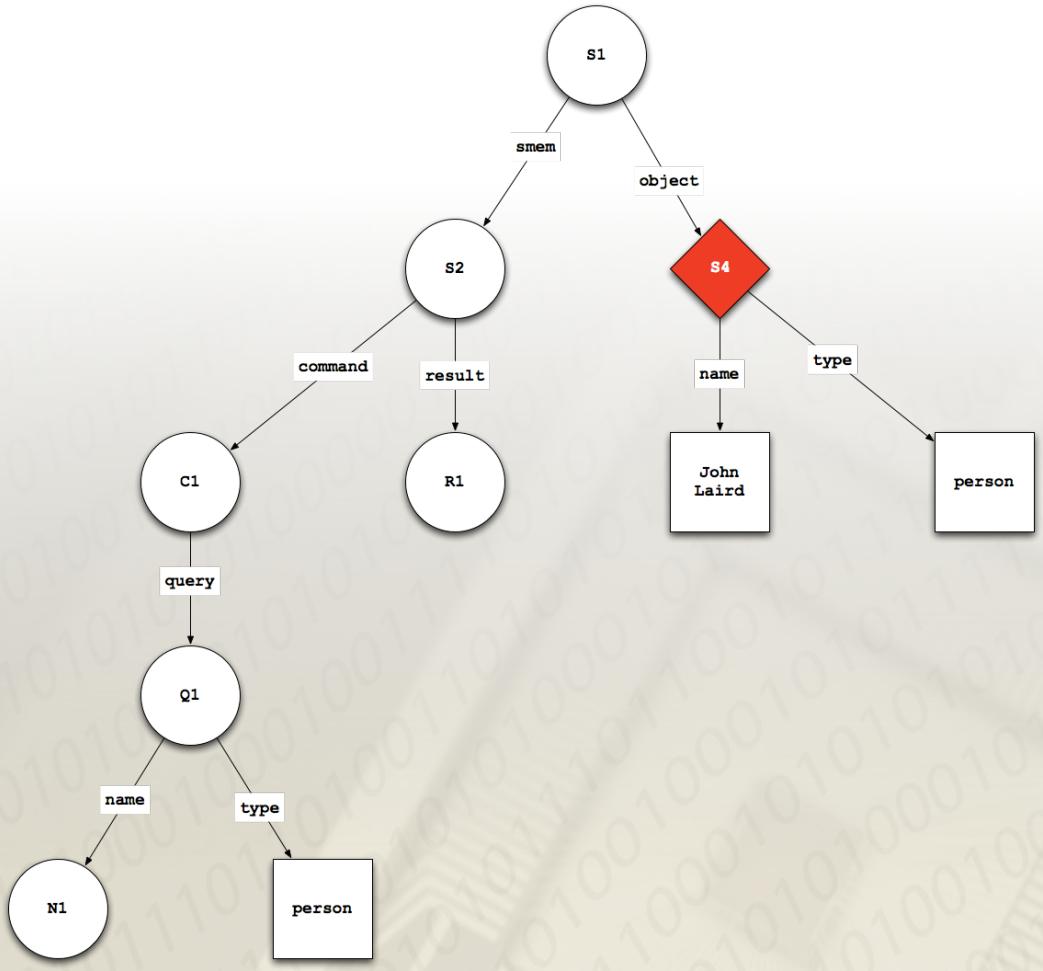
Semantic Store

- Re-uses EpMem database code
- Currently SQLite
  - Allows inspection/queries if stored to disk

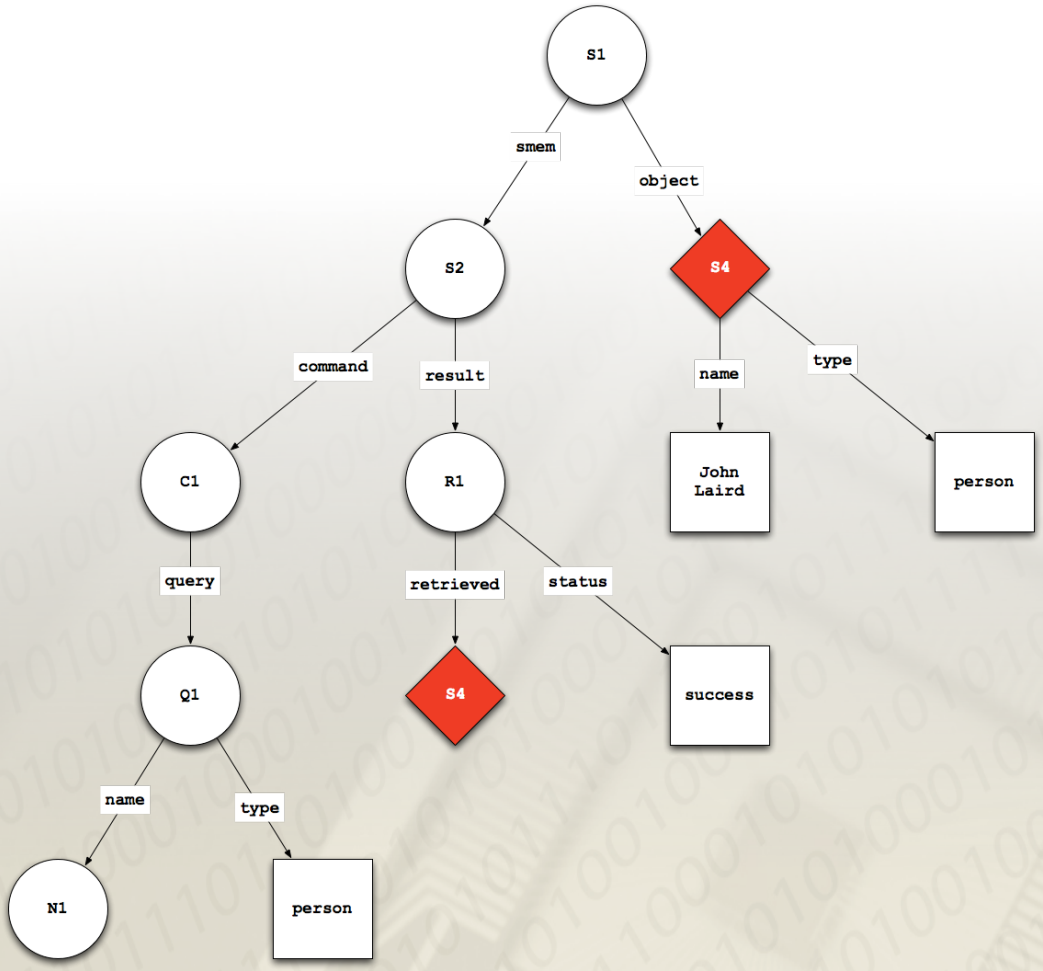
Cue-Based Retrievals

- Deliberate, one per state
  - `state.smem.command.query <cue>`
- Cue semantics
  - Constant symbol: exact match of attribute and value
  - LTI: exact match of attribute and value
  - Identifier: exact match of attribute, anything for value
- Matches
  - Must match ALL cue elements
  - Most recently stored/retrieved on tie (can optionally **prohibit**)
- Return
  - Status
  - LTI (if successful)
    - Direct augmentations (if LTI was not in WM or did not have augmentations in WM)

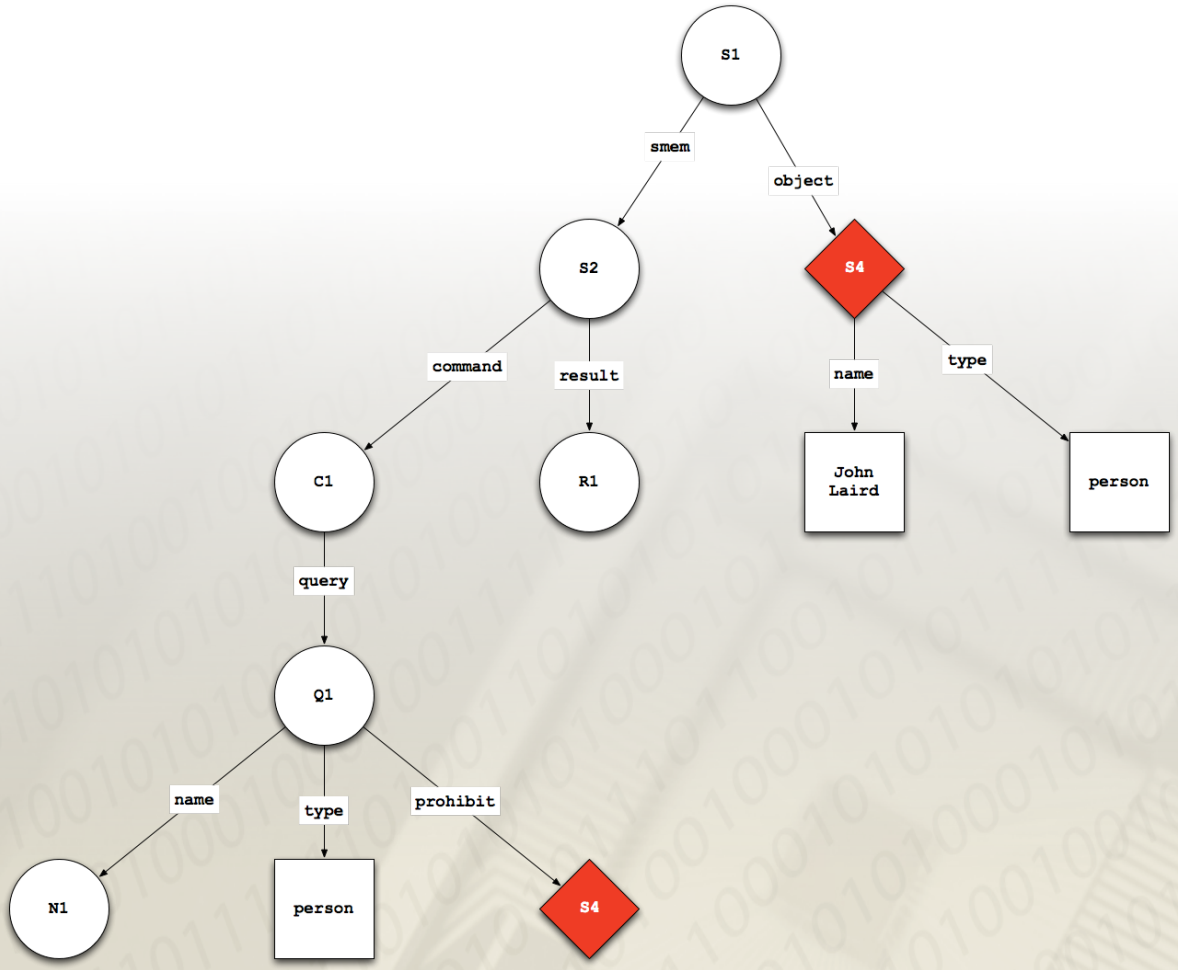
Cue Example


S1
smem
object
S2
S4
command
result
name
type
C1
R1
John
Laird
person
query
Q1
name
type
N1
person

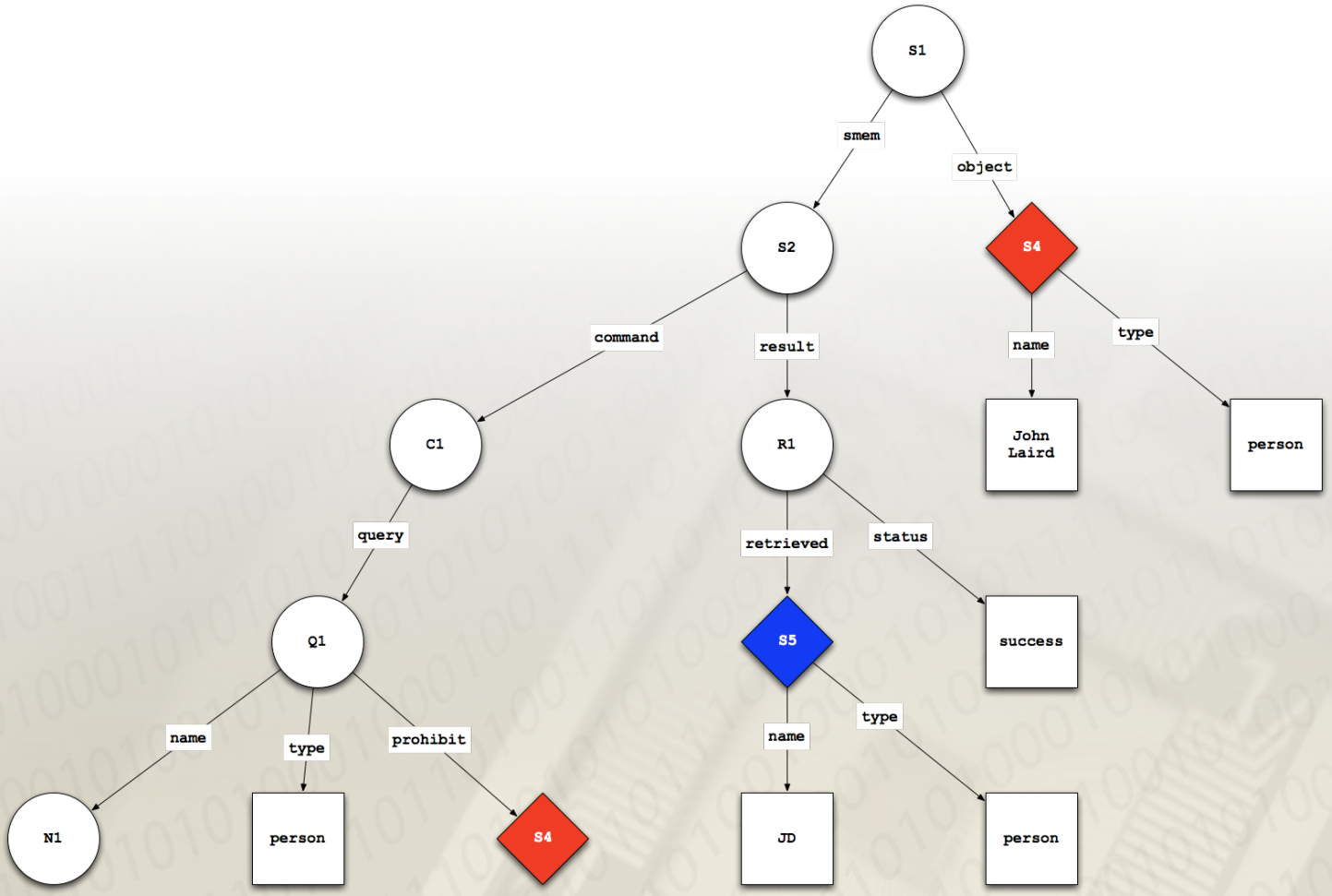
Cue Example: Result



Prohibit Example



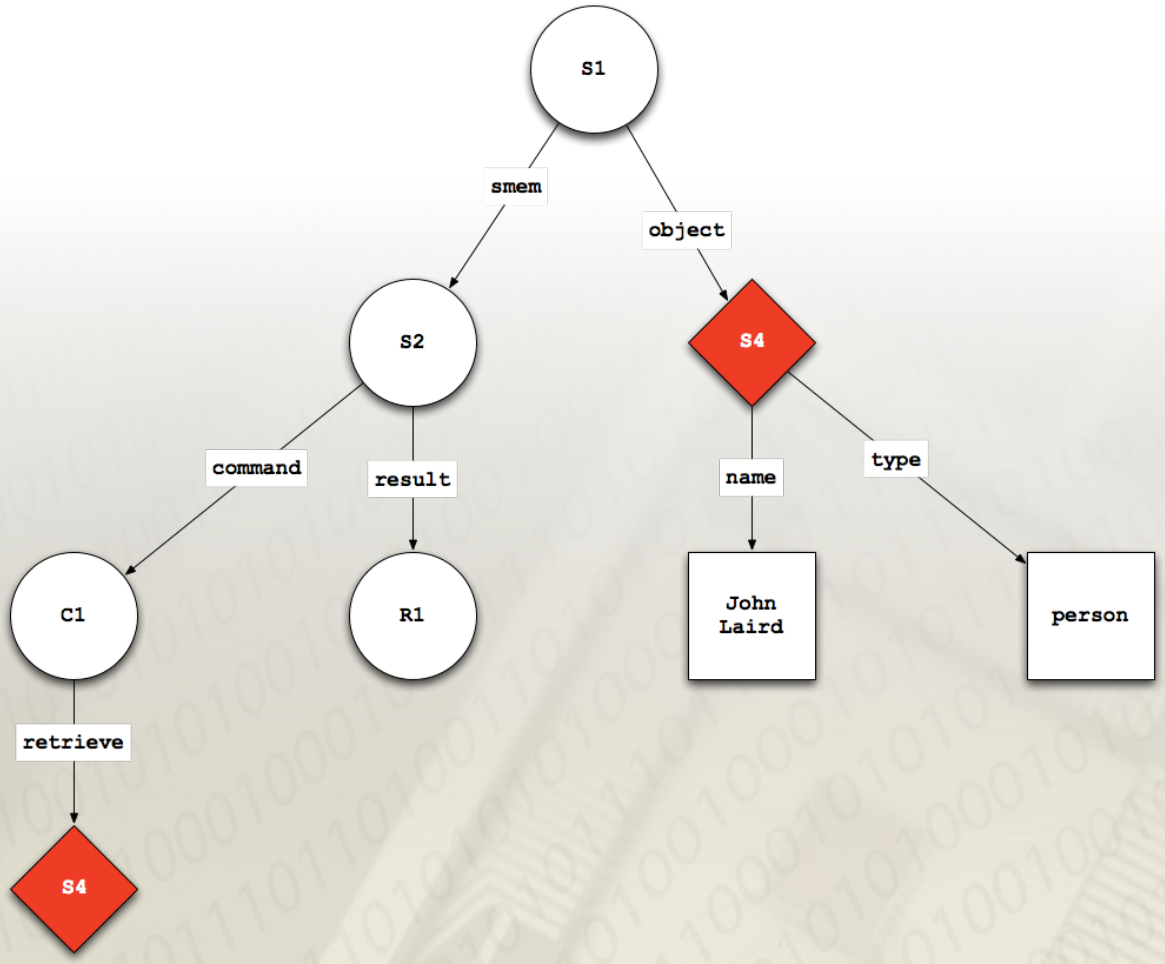
Prohibit Example: Result



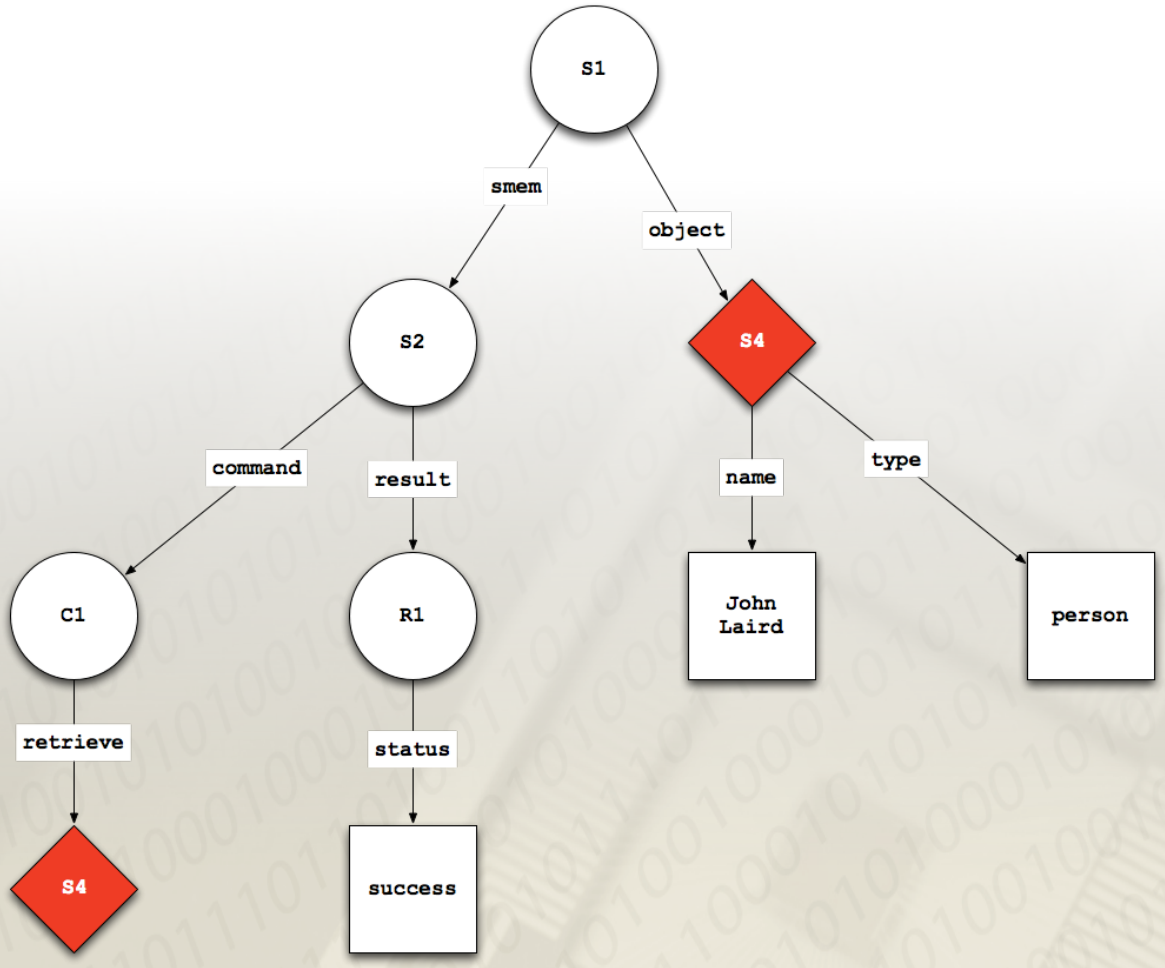
Non-Cue-Based Retrievals

- Deliberate, one-per-state
  - `state.smem.command.retrieve <lti>`
- Return
  - Status
  - Direct augmentations of LTI (if does not contain augmentations in WM)

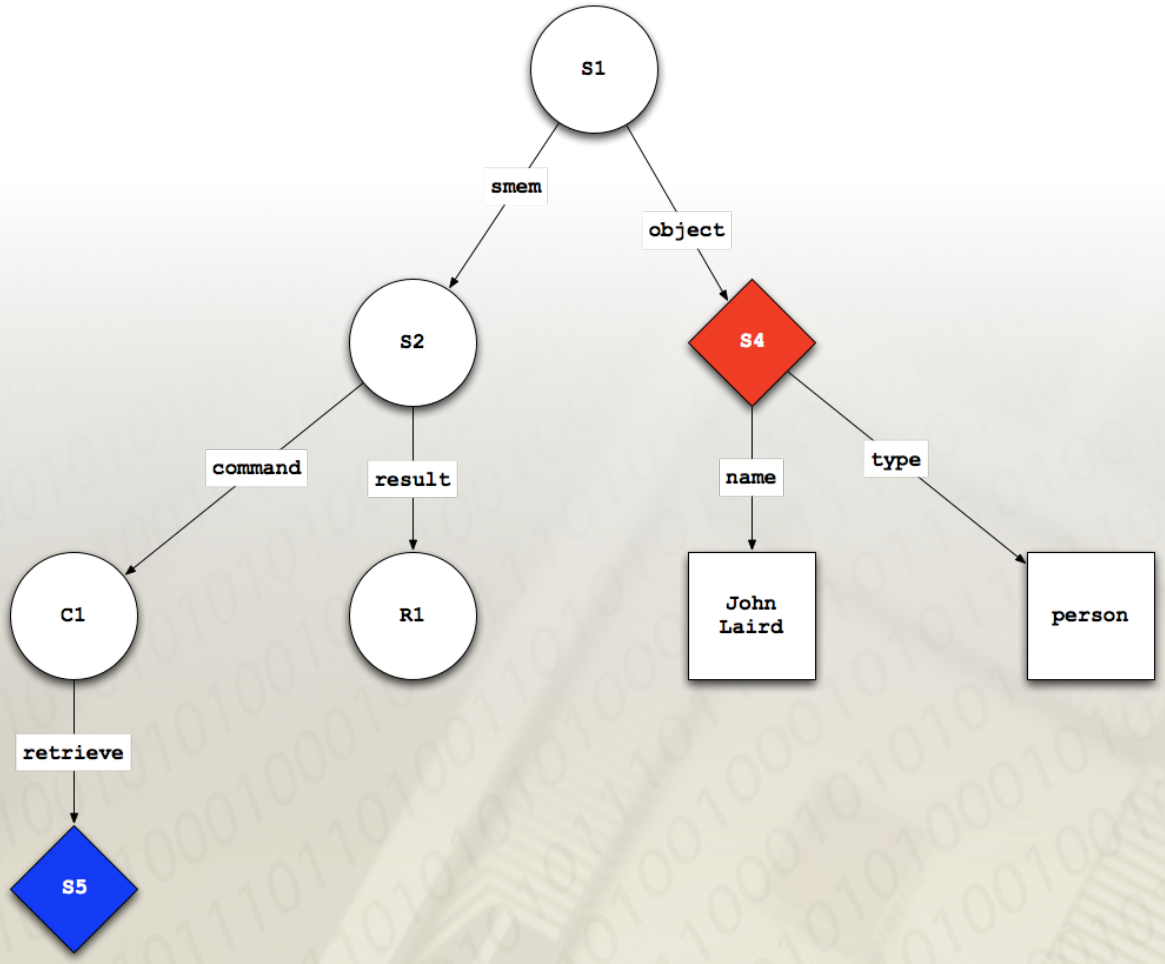
NCB Example 1



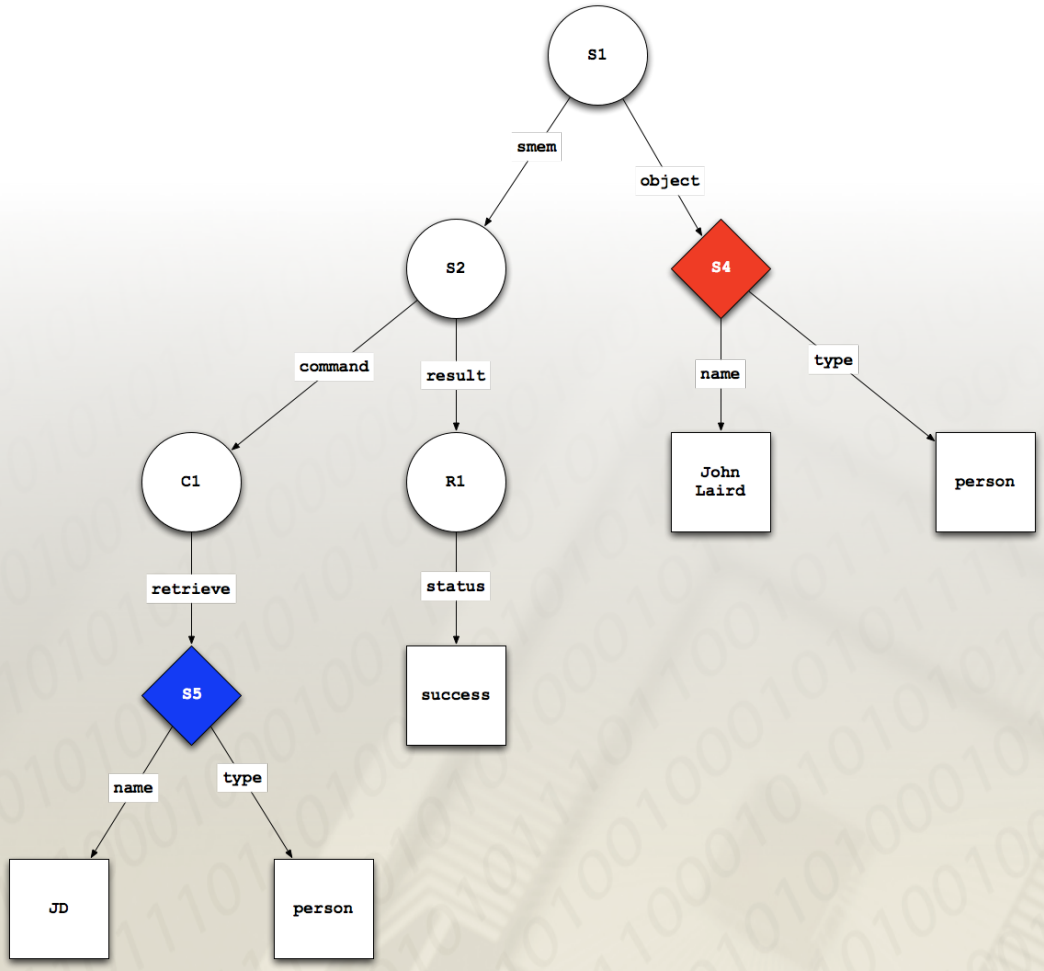
NCB Example 1: Result



NCB Example 2



NCB Example 2: Result



Integration

- Integration with other learning mechanisms is currently not implemented

- Proposed: treat LTI's as symbols instead of variables
  - Chunking: instantiate LTI's as constants
  - EpMem: LTI in cue is treated as a constant match

Arithmetic Demo

- Modified to use Working Memory or SMem (default)
  - If SMem, can use agent or manual storage (default)

- Automatically verifies results
  - Used as a basic SMem unit test

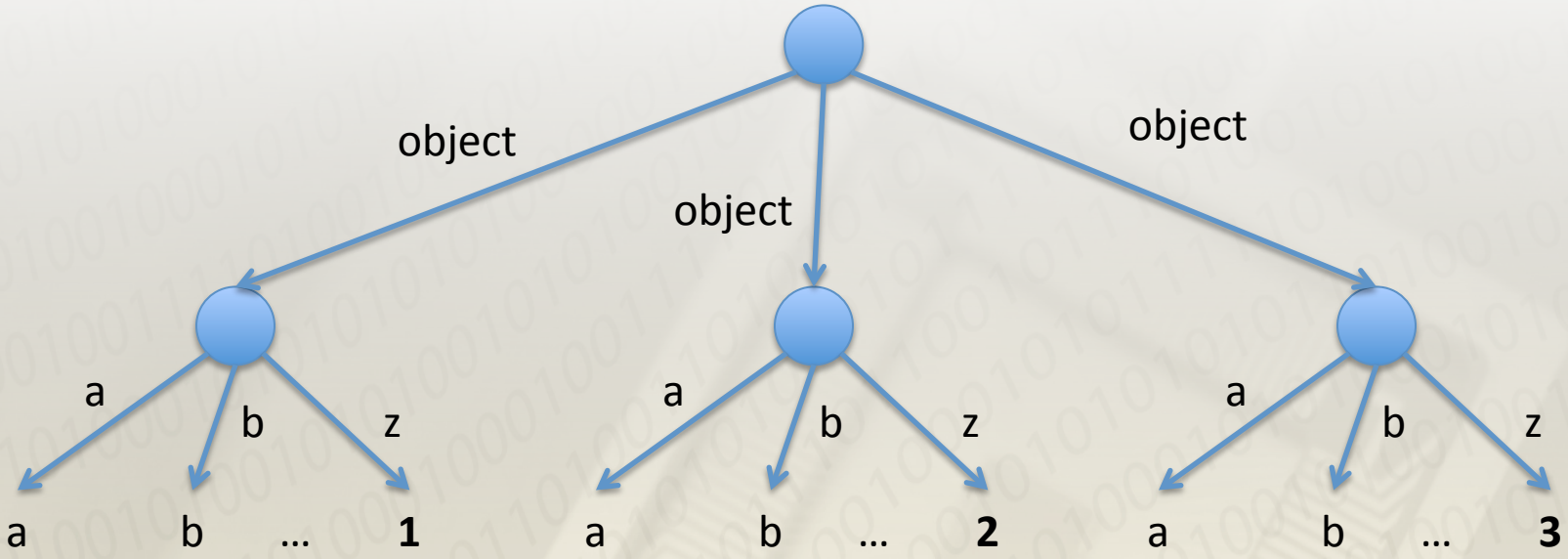
Source	Load	Decisions	Kernel Time (sec)
SMem	Agent	497,093	55.814
SMem	Manual	497,088	55.957
WM	Agent	447,629	44.401

50K Decisions
26%

* 10K addition problems

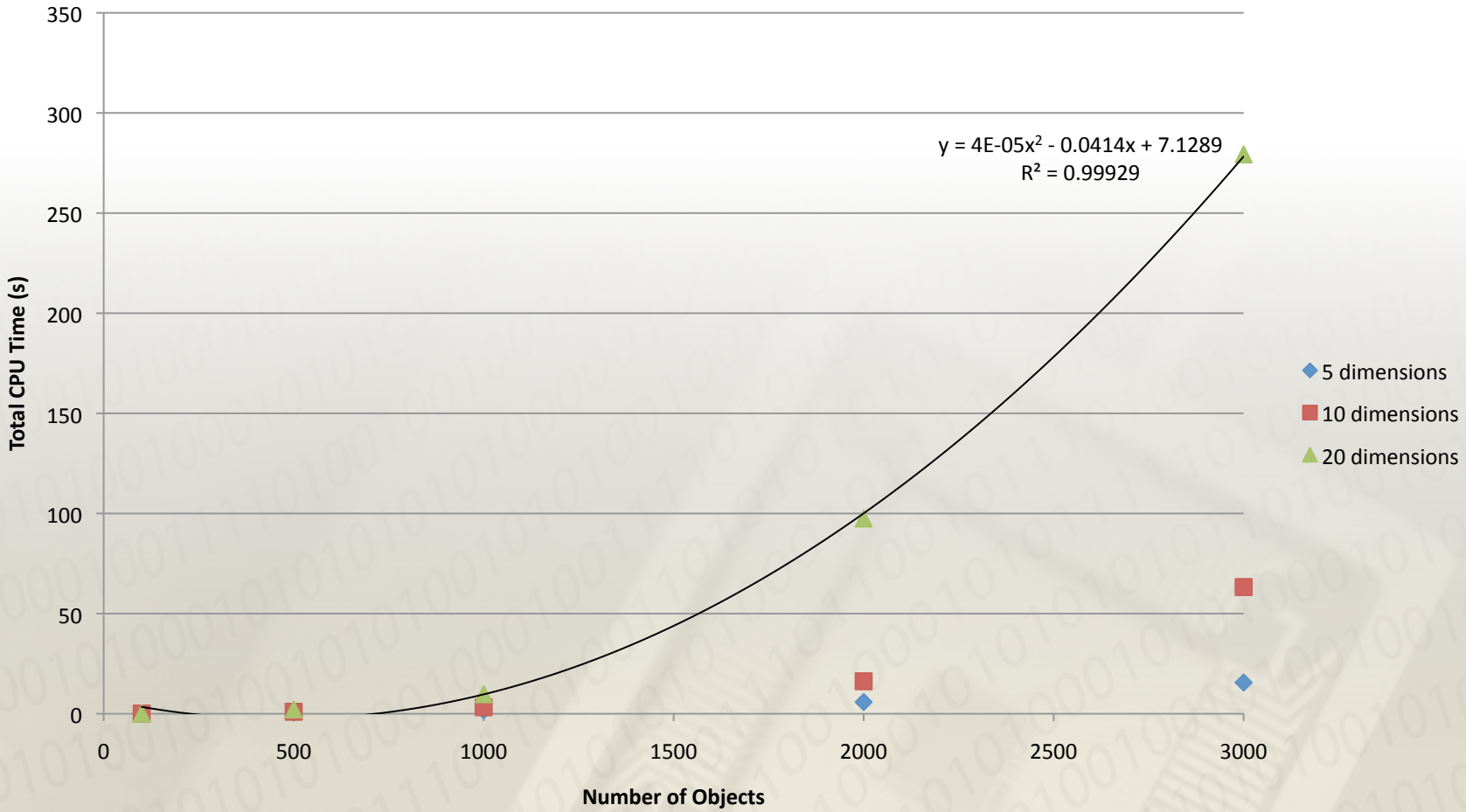
Rete Defeat

- Contrived agent to expose a situation in which SMem achieves better performance than WM+rete
  - Intuition: run-time vs. compile-time optimization

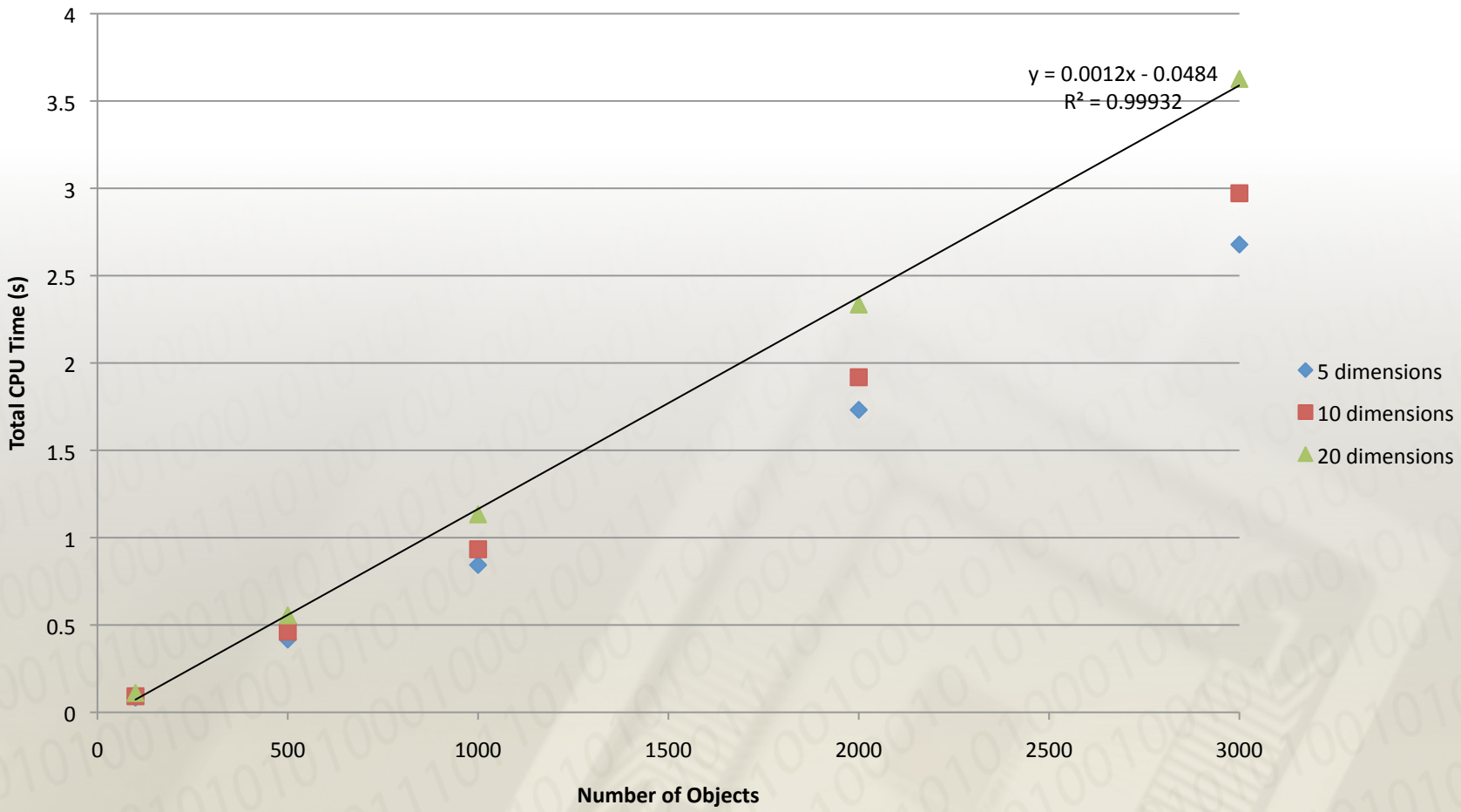


- Task: for i=1 to #objects, retrieve (given all features)

WM: Rete Defeat



SMem: Rete Defeat



Future Work

- Extended Use
  - Need feedback from “customers”
  - Scaling study to large semantic store: Cyc, WordNet
- Exploration of motivating tasks
  - Perhaps limiting Working Memory (# WMEs, multi-valued attributes, activation+forgetting, etc)
- Iterative development and improvement of software

Evaluation

Nuggets

- Pilot release (9.2.0)
- Re-uses Soar-EpMem database codebase

Coal

- Limited use & evaluation