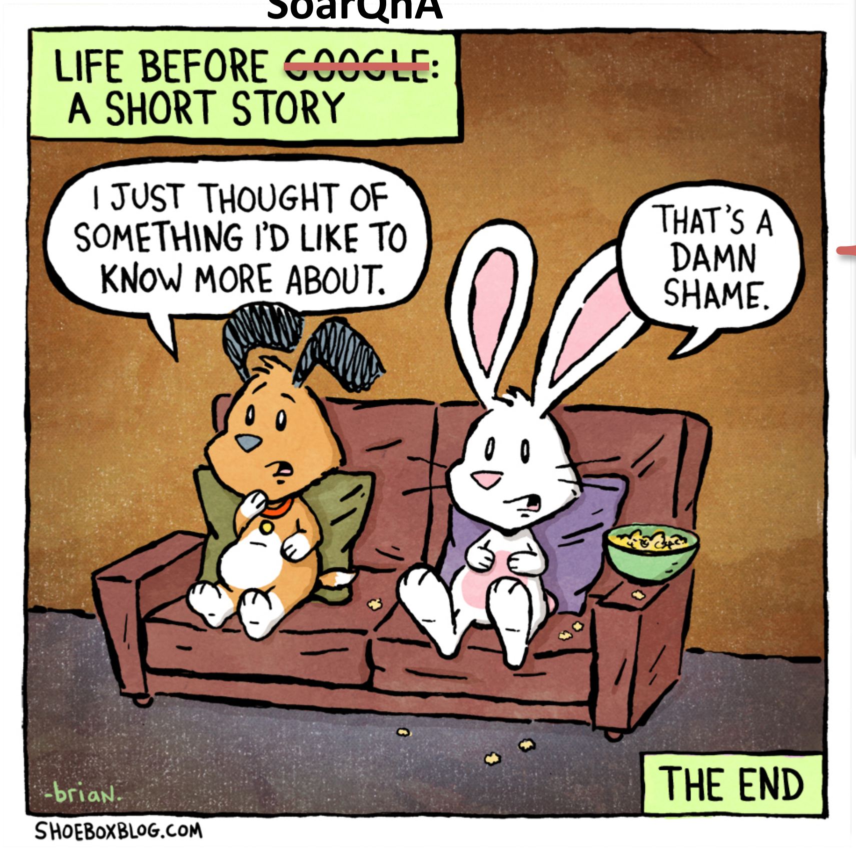
SoarQnA
Standardized Access to External Knowledge

Nate Derbinsky
University of Michigan

Access to External Knowledge in Soar

SoarQnA

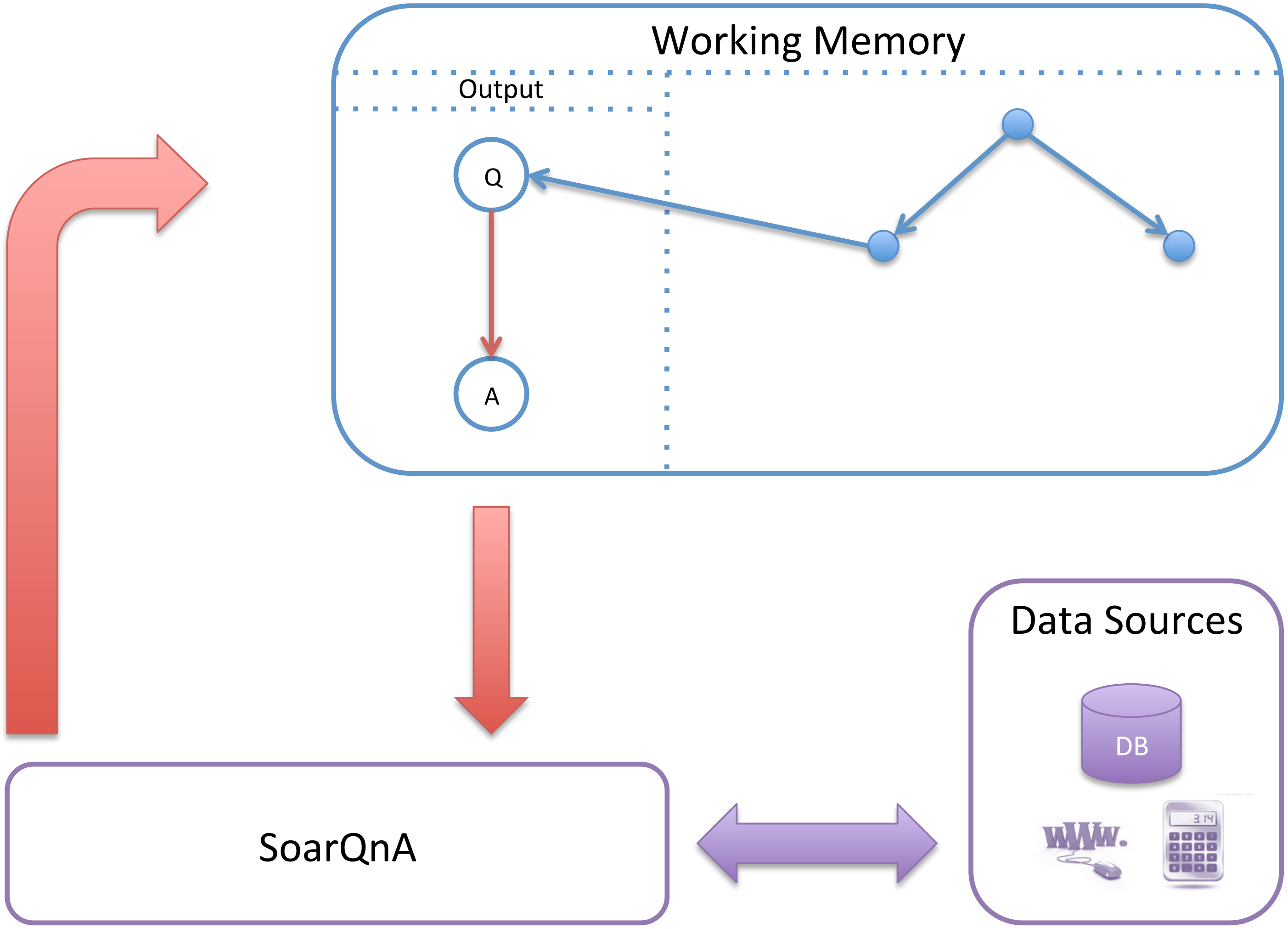

LIFE BEFORE ~~GOOGLE~~: A SHORT STORY
I JUST THOUGHT OF SOMETHING I'D LIKE TO KNOW MORE ABOUT.
THAT'S A DAMN SHAME.
THE END
-brian.
SHOEBOXBLOG.COM

- Data chunking
- Large working memory structures
- Custom RHS functions
- Custom SML

SoarQnA: Objectives

- Efficient
- Quick development/prototyping
- Simple, unified agent interface independent of
  - Data format/source
  - Query type/complexity
- Broadly applicable

SoarQnA: Overview



SoarQnA: Components

- Data source driver
- Data source instance descriptor
- Agent interface
- IO Hooks

Data Source Driver

Interface	**Example**
<u>Driver</u> • Create new connections	• Database Driver
<u>Connection</u> • Executes queries	• Database Connection
<u>QueryState</u> • Maintains query state and provides incremental result access	• Query cursor

Data Source Instance Descriptor

Interface	**Example**
• Instance name – Agent will use this	• “my_database”
• Connection information – Driver, instance parameters	• “host=localhost, port=…”
• Available queries – Name, definition	• “all=select * from…”

Agent Interface

Input

```
qna-registry
    ^|instance name|
        ^query |query name|
```

Output

```
qna-query
    ^source |instance name|
    ^query |query name|
    ^parameters
        ^name value
    ^results << all incremental >>

    ^status
    ^id #
    ^result
        ^num #
        ^features
            ^name value
        ^next

qna-next
    ^id #

    ^status
```

IO Hooks

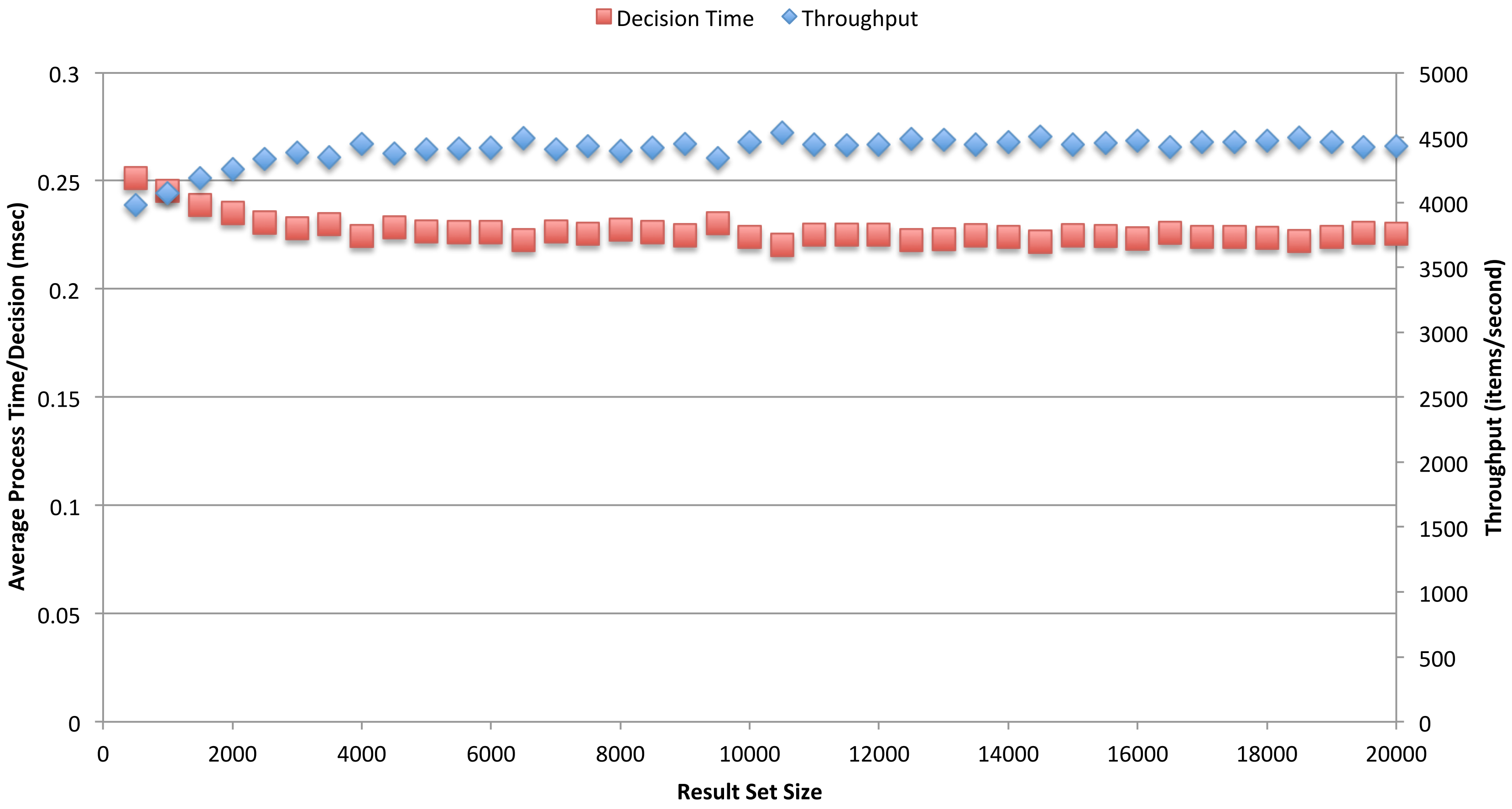
- Listens for qna- output commands, provides feedback during input-phase
- Executable can run with any existing SML client with no modification
- Exposes library calls for more direct programmatic control

Included with Soar 9.3.1

- Documentation
- Data Source Drivers
  - **Math**. Most Soar RHS functions
  - **Dice**. Liar's dice probability calculations
  - **DB**. JDBC wrapper
- Example agent
- SQLite performance evaluation

SQLite Performance Evaluation

Large Result Set Exhaustion



SoarQnA: Objectives Revisited

- ✓ Efficient
- ✓ Quick development/prototyping
- ✓ Simple, unified agent interface independent of
  - Data format/source
  - Query type/complexity
- ✓ Broadly applicable

Pragmatic Development Comparison

	Semantic Memory	SoarQnA
Implementation	C++, Kernel	Java, SML
Knowledge Representation	Directed graph via symbolic triples	Fixed per data source instance
Query Semantics	Fixed in architecture	Fixed per query
Query Composition	Dynamic	Parameterized
Retrieval Biases	Fixed in architecture	Fixed per query
Query Frequency	1 per state per decision	Arbitrary number per decision
Query Time	Generally << 1msec	Dependent upon data source instance, query
Store Dynamics	Fixed in architecture (user can only add new knowledge)	Fixed per data source instance (could include arbitrary user modification)
Learning	Integrated within architecture	Deliberate via rules

Evaluation

Nuggets

- Objectives achieved
- In use for Liar's dice
- Included in Soar 9.3.1

Coal

- Small data source driver set
  - Currently read-only
- Limited performance evaluation
- Query execution is synchronous with agent execution, which may impede reactivity
- CSoar only