A Pure Java Implementation of Soar

Dave Ray

daveray@gmail.com

Agenda

- What and why?
- What's (not) there?
- What's new?
- Performance
- Nuggets and Coal
- Demo
- Conclusion

What is JSoar?

- 100% Java implementation of Soar
- Open Source, BSD licensed
- Runs on Java SE versions 1.6 and higher
- Started in Fall of 2008 based on Soar 9.0.1
- http://jsoar.googlecode.com

Why JSoar?

- It's a fun and educational personal project

"It's like someone who doesn't know how to drive building a car."

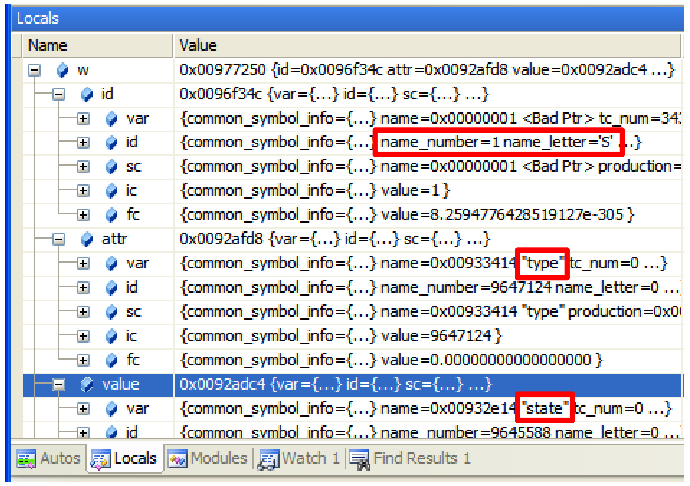
- Anonymous

Why JSoar?

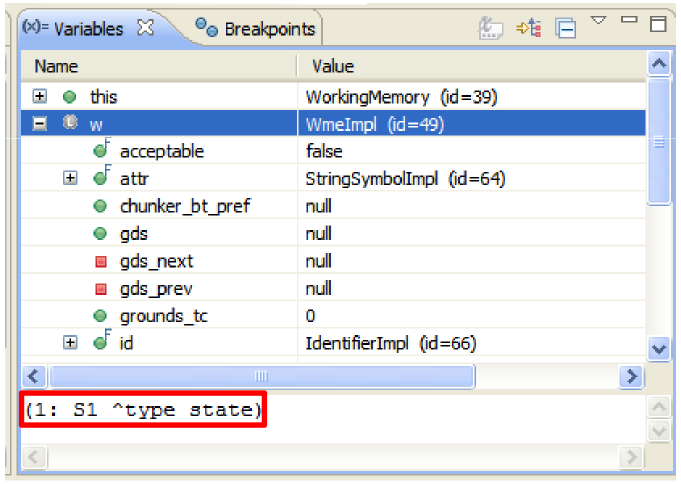
- Native dependencies are a frequent topic on Soar mailing lists
  - System.DllNotFoundException: Unable to load DLL 'CSharp_sml_ClientInterface': The specified module could not be found.
  - Native code library failed to load. java.lang.UnsatisifiedLinkError: libJava_sml_ClientInterface.jnilib
  - etc

Why JSoar?

- Debugging the Soar kernel in C is painful
- Unions, pointer tricks, poor tool support



Name	Value
w	0x00977250 {id=0x0096f34c attr=0x0092afd8 value=0x0092adc4 ...}
id	0x0096f34c {var={...} id={...} sc={...} ...}
var	{common_symbol_info={...} name=0x00000001 <Bad Ptr> tc_num=34
id	{common_symbol_info={...} name_number=1 name_letter='S' ...}
sc	{common_symbol_info={...} name=0x00000001 <Bad Ptr> production=
ic	{common_symbol_info={...} value=1 }
fc	{common_symbol_info={...} value=8.2594776428519127e-305 }
attr	0x0092afd8 {var={...} id={...} sc={...} ...}
var	{common_symbol_info={...} name=0x00933414 "type" tc_num=0 ...}
id	{common_symbol_info={...} name_number=9647124 name_letter=0 ...
sc	{common_symbol_info={...} name=0x00933414 "type" production=0x0
ic	{common_symbol_info={...} value=9647124 }
fc	{common_symbol_info={...} value=0.00000000000000000 }
value	0x0092adc4 {var={...} id={...} sc={...} ...}
var	{common_symbol_info={...} name=0x00932e14 "state" tc_num=0 ...}
id	{common_symbol_info={...} name_number=9645588 name_letter=0 ...



Name	Value
this	WorkingMemory (id=39)
w	WmeImpl (id=49)
acceptable	false
attr	StringSymbolImpl (id=64)
chunker_bt_pref	null
gds	null
gds_next	null
gds_prev	null
grounds_tc	0
id	IdentifierImpl (id=66)

```
(1: S1 ^type state)
```

- Tests are easier to write and run
- Simpler crash diagnosis

Why JSoar?

- Java is a kitchen sink. Easy access to extensive, often built-in, libraries
- Seamless (no SWIG) integration with a growing list of alternative languages
  - JRuby, Jython, JavaScript, Scala, Groovy, Clojure, …
  - And you thought Tcl was dead ☺
- .NET interoperability with IKVM.NET

What's there?

- Base Soar kernel (9.0.1)
  - Extensive, automated, functional tests to ensure compatibility and prevent regressions
- Chunking
- Reinforcement learning
- Java API
  - Encapsulates kernel while still providing access to Soar internals if needed

What's not there?

- SML
  - No remote debugging
- Rete network save/load
- Episodic Memory
- Semantic Memory
- Several small, recent fixes and changes

What's new?

- JSoar debugger
- Improved concurrency
  - One thread per agent
  - **wait** RHS function: no busy waiting for input
    - *Give your CPU fan a break*
- "Real" RHS functions
  - Multiple args, as symbols
  - Return structures as well as primitives

What's new?

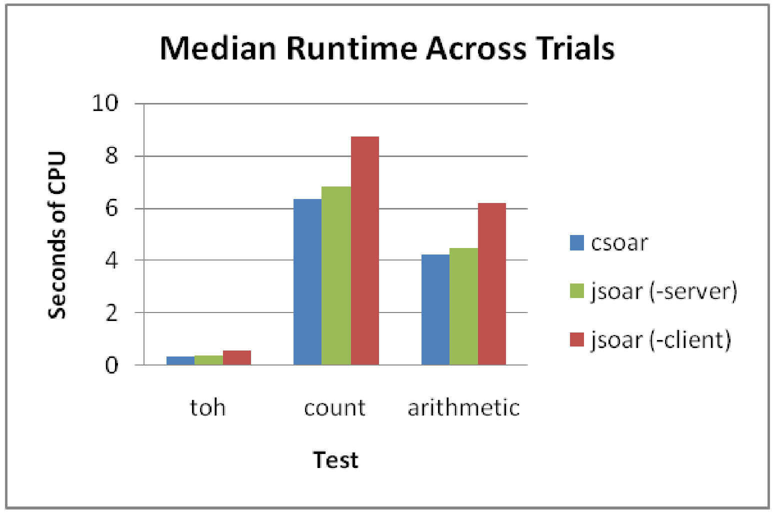
- Additional RHS functions
  - **java** – Call Java code
  - **debug** – Open the debugger
  - **wait** – Pause the agent's thread until new input arrives
  - **get-url** – Read from a URL
  - **to/from-xml** – Convert XML (e.g., from a URL) to working memory
  - All **java.lang.Math** functions

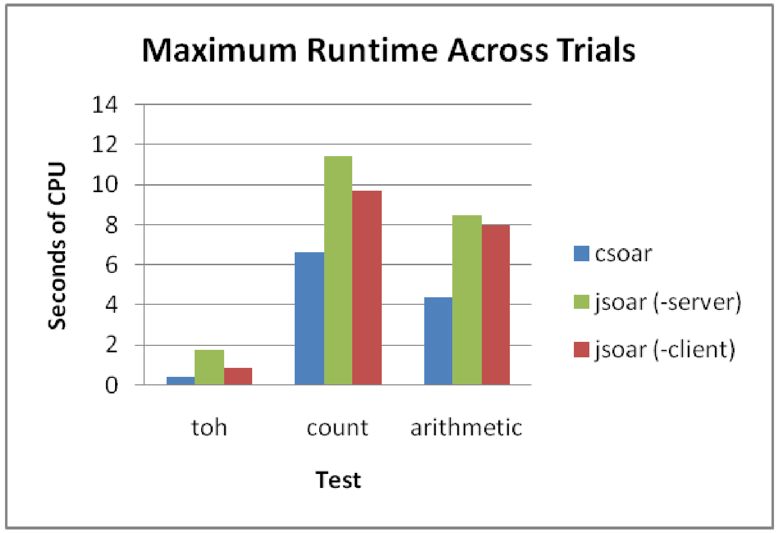
What (else) is new?

- **source** command works on URLs
  - source http://darevay.com/jsoar/waterjugs.soar
- **SoarBeans** – Auto-convert working memory to Java objects
- **QMemory** – Dead-simple, thread-safe input generation
- **RSoar** – Ruby API (w/ example Twitter interface)
  - *Yet another wrapper for Soar*

Performance

- JSoar fairs pretty well in CPU performance
- -server option of JVM provides significant speedup
- Significant startup cost seen in "max" graph

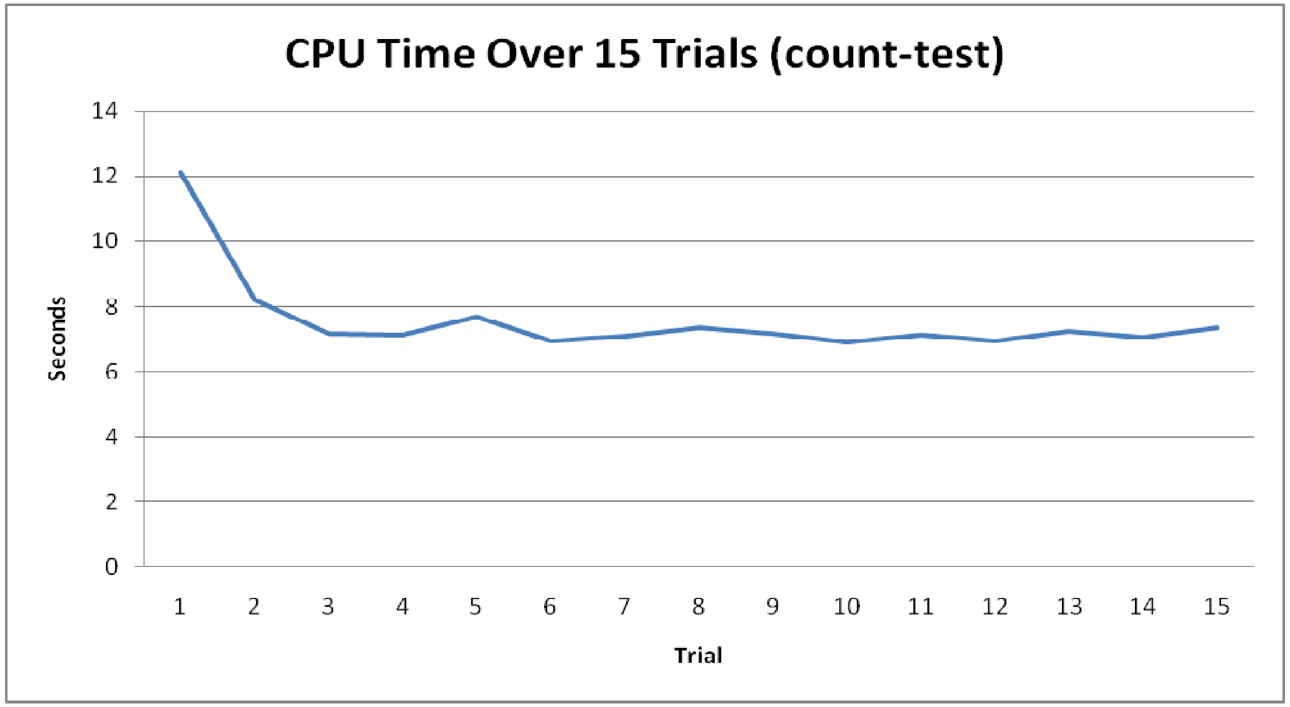




See http://code.google.com/p/jsoar/wiki/PerformanceTestNotes20090621

Performance

- JVM optimizes at run-time



- JSoar gets faster the longer it runs

Future Plans

- Port Episodic and Semantic Memories
- Finish SML Port (volunteers?)
- Investigate memory usage issues
- Build a cool demo with a cluster of agents handling requests behind a JRuby on Rails web app (or something)

Nuggets

- It works!
- Performs better than expected
- Jon Voigt: "Develop and debug in JSoar, backport to CSoar"
- In use on at least one SoarTech project
- Symbol reference counts are gone

Coal

- Large memory footprint
- Lack of SML support means rewriting interface code
- Symbol reference counts are gone, but counts remain on other structures

JSoar Debugger Demo

Conclusion

- Please give it a try!
- Home: http://jsoar.googlecode.com
- Applet: http://darevay.com/jsoar
- Blog: http://blog.darevay.com/category/soar/
- Thanks
  - to SoarTech for supporting this talk
  - to Bob Marinier for early, painful debugging and performance testing
  - to Jon Voigt for ongoing bug fixes

OVERFLOW

What's new

- Quick Memory
- Simple, thread-safe input generation
- Programmatic, or command-line

```
> qset block(red).x-location 1
> qset block(red).y-location 0
> qset block(red).color red
> qset block(blue).x-location 2
> qset block(blue).y-location 0
> qset block(blue).color blue
...
> qset block(red).x-location 2
> qset block(red).y-location 1
```

```
^io.input-link
  ^block
    ^x-location 2
    ^y-location 1
    ^color red
  ^block
    ^x-location 2
    ^y-location 0
    ^color blue
```

- See *org.jsoar.kernel.io.quick*

What's new?

- Convert any working memory to Java beans

```
^output-link
   ^move-along-route
      ^route-name |alpha|
      ^desired-speed 100.0
      ^altitude 35.0
```

```
class MoveAlongRoute {
  public String routeName;
  public double altitude;
  private double speed;

  public double getDesiredSpeed(){
    return speed;
  }

  public void setDesiredSpeed(double s){
    speed = s;
  }
}
```

- See *org.jsoar.kernel.io.beans*