JessTab: Using Jess together with Protégé

Henrik Eriksson



Background

Problems:

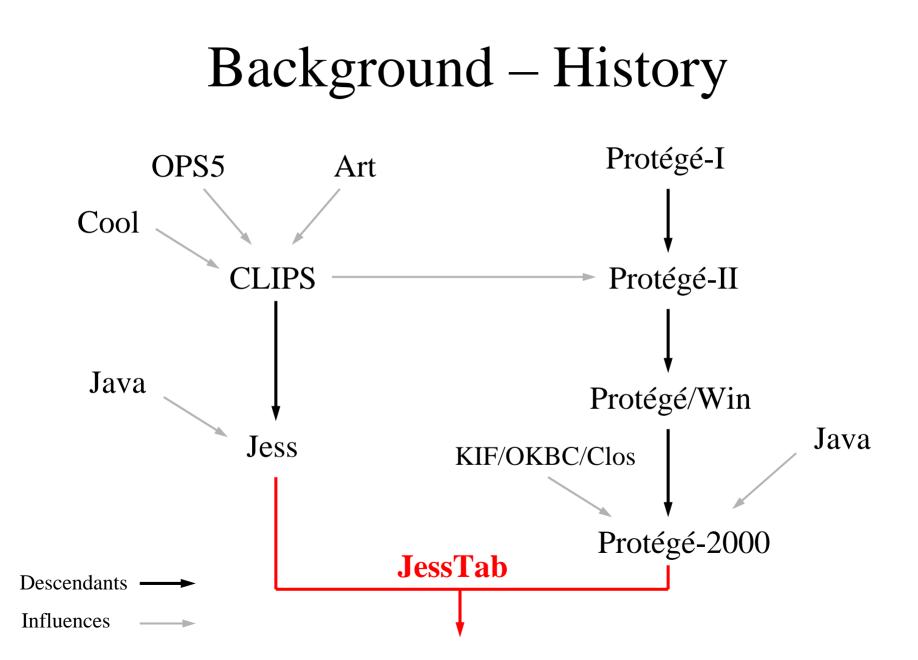
• Difficult to directly integrate problem solving and ontology development in Protégé

- Languages/shells need direct access to Protégé

Difficult manage large/complex ontologies
 – Ontology editors should be programmable

Background: What is Jess?

- Java Expert System Shell; based on CLIPS
- Forward chaining; production rules
- Fact-base and pattern matching
- Lisp-like syntax
- No support for object orientation
 The Cool subsystem of CLIPS not implemented
- Developed by Sandia Laboratories
 - http://herzberg.ca.sandia.gov/jess/



Integration – Two possibilities

- Loose integration
 - No changes to each representation model
 - Translators between formats
 - Independent software
- Tight integration
 - Changes to representation models when needed
 - Integrated software (e.g., same Java VM)
 - Unified user interface

Approach – JessTab plug-in for Protégé

- Jess console window in Protégé
- Mapping instances to Jess facts
- Functions for knowledge-base operations
- Mirroring Jess definitions in Protégé knowledge bases
- Support for metalevel objects
- Support for methods and message handlers (coming)

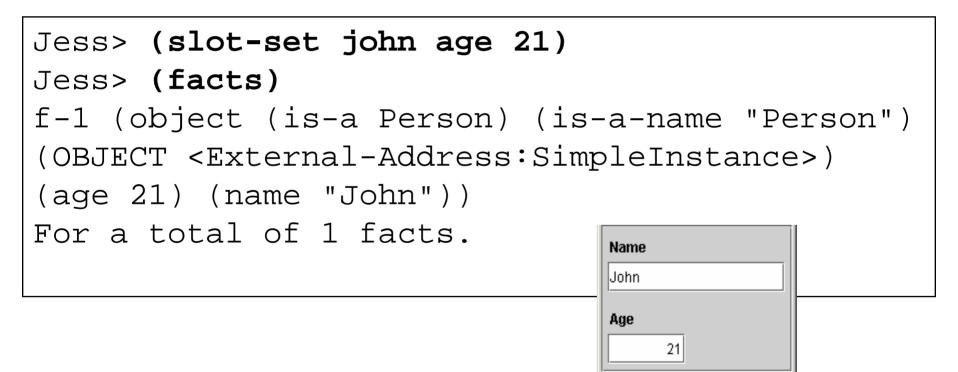
Jess console window in Protégé

🐕 <new> Protégé</new>	é				
Project Edit Window	w Help				
🎽 🖆 😭 🗠	🖷 🖪 🛃	AR			
C)) Classes	Slots Forms	tii Instances	🚧 Queries	🗸 Jess]
Console Facts	Rules Functions	Defglobals De	eftemplates	Deffacts	Settings
Jess, the Java Expert 8 Copyright (C) 2001 E.J Jess Version 6.1p3 6/2 Jess>	J. Friedman Hill and the	e Sandia Corporatio	en Enter	য় হ	ear Window

Defining classes and instantiating them

```
Jess> (defclass Person (is-a :THING)
                                                                                                                                                                                                                                                                                                  S Slots
                                                                                                                                                                                                                                                            C)) Classes
(slot name (type string))
                                                                                                                                                                                                                                                        Relationship Superc... 💌
                                                                                                                                                                                                                                                          🕒 :THING 🗛
 (slot age (type integer)))
                                                                                                                                                                                                                                                          Our Contended Contende
                                                                                                                                                                                                                                                                   C) Person
TRUE
Jess> (make-instance john of Person (name "John") (age 20))
<External-Address:SimpleInstance>
                                                                                                                                                                                                                                                                                   (IIII) Instances
                                                                                                                                                                                                                                                               Forms
Jess> (mapclass Person)
                                                                                                                                                                                                                                                                                                        VC
                                                                                                                                                                                                                                                         Direct Instances
                                                                                                                                                                                                                                                          🛈 iohn
Person
Jess> (facts)
f-0 (object (is-a Person) (is-a-name "Person")
 (OBJECT <External-Address:SimpleInstance>)
 (age 20) (name "John"))
For a total of 1 facts.
```

Modifying slots



Creating a second instance

```
Jess> (make-instance sue of Person (name "Sue") (age 22))
<External-Address:SimpleInstance>
Jess> (facts)
f-1 (object (is-a Person) (is-a-name "Person")
(OBJECT <External-Address:SimpleInstance>)
(age 21) (name "John"))
f-4 (object (is-a Person) (is-a-name "Person")
(OBJECT <External-Address:SimpleInstance>)
(age 22) (name "Sue"))
For a total of 2 facts.
```

Adding a Jess rule

```
Jess> (defrule twentyone
  (object (is-a Person)
  (name ?n) (age ?a&:(>= ?a 21)))
=>
(printout t "The person " ?n
            " is 21 or older" crlf))
TRUE
Jess> (run)
The person John is 21 or older
The person Sue is 21 or older
2
Jess>
```

Functions for knowledge-base operations

mapclass mapinstance unmapinstance defclass make-instance initialize-instance modify-instance duplicate-instance definstances unmake-instance slot-get slot-set slot-replace\$ slot-insert\$ slot-delete\$ slot-facets slot-types slot-cardinality

slot-range slot-allowed-values slot-allowed-classes slot-allowed-parents slot-documentation slot-sources facet-get facet-set class class-existp class-abstractp class-reactivep superclassp subclassp class-superclasses class-subclasses get-defclass-list class-slots

instancep instance-existp instance-name instance-address instance-addressp instance-namep slot-existp slot-default-value set-kb-save get-kb-save load-kb-definitions load-project include-project save-project jesstab-version-number jesstab-version-string get-knowledge-base qet-tabs

Mirroring Jess definitions in Protégé knowledge bases

👺 <new> Protégé</new>			
Project Edit Window Help			
C) Classes S Slots Forms	A R Instances A Queries	Jess	
Classes Slots Forms			
Relationship Superc ▼ V C ♪ ×	C:JESS-RULE (type=:STANDAR	D-CLASS)	
	Name	Documentation	Constraints 🛛 🗸 C 🕇 –
	JESS-RULE		
	Role		
	Concrete -		
		- I	
C JESS-RULE™	Template Slots		
- © :JESS-FUNCTION ^M	Name	Type Cardinality	Other Facets
USA DEFGLOBAL	S :DOCUMENTATION	String multiple String single	
- C :JESS-DEFTEMPLATE - C :JESS-DEFFACTS	S :DEFINITION-NAME	String single	
- C :JESS-DEFMESSAGE-HANDLE	S :KB-SAVE	Boolean single	default={false}
SI SS-DEFMETHOD			
C FONCTION*			
		Your Jess de	finitions
		I OUT JESS de	ennuons as
		C ¹ 1 1 1 1 1 1	Contraction Description
		TITST-CLASS C1	tizen in Protégé
Superclasses + -			
©: JESS-DEFINITION A			
©:RULE ^A			
	<u> </u>		

Editing Jess definitions in Protégé

<pre>> Protégé-2000</pre>	
roject Edit Window Help	
ற Classes Sill Slots 🔝 Forms 🕬 Instances 🆓 Queries 🗸	Jess
	Jess MAIN::print-abstract (:JESS-RULE) C × Name Documentation MAIN::print-abstract
	✓ Parse! X Revert
<u><u></u></u>	Save definition in know Jess rule editor in
	Protégé

Editing Jess definitions in Protégé (cont.)

test1 Protégé-2000	(C:\Program Files\Protege-2000\test1.pprj)		
roject Window Help			
) 🖻 🕼 🗧 🧏			
C)) Classes SII Slots	📰 Forms 🔅 Instances 🏘 Queries 🗸 Jes	s	
Console Facts Rule	Functions Defglobals Deftemplates Deffacts	Settings	
Rule List		V	
MAIN::print-abstract	Name	Documentation	
	MAIN::print-abstract		
	Jess definition		
	(defrule MAIN::print-abstract (MAIN::object (:NAME ?n) (:ROLE Abstract))		
	=> (printout t "The class " ?n " is abstract." crlf))		
	✓ Parse!	× Revert	
	Save definition in knowledge base	Rule-editor subtab in	
		IessTab	

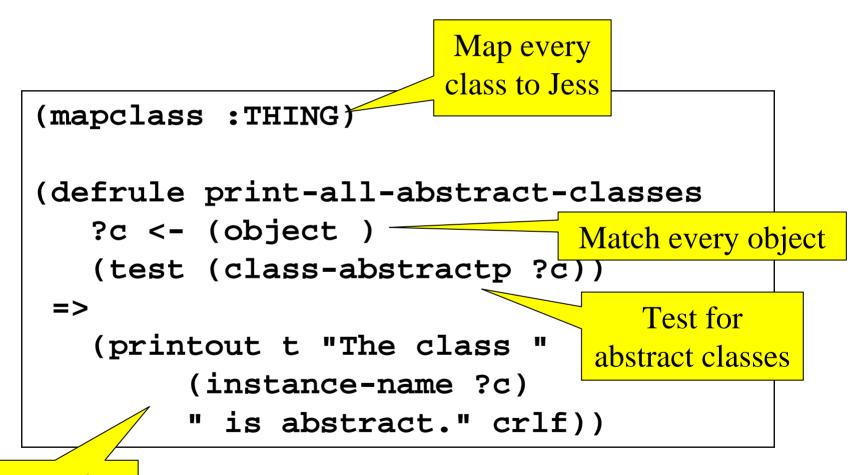
Support for Protégé metalevel objects

- JessTab support for metaclasses, metaslots, and metafacets
- Functions for instances work for classes too

 and for slots and facets
- Defining classes by instantiating metaclasses:

(make-instance Person of :STANDARD-CLASS
 (:DIRECT-SUPERCLASSES :THING))

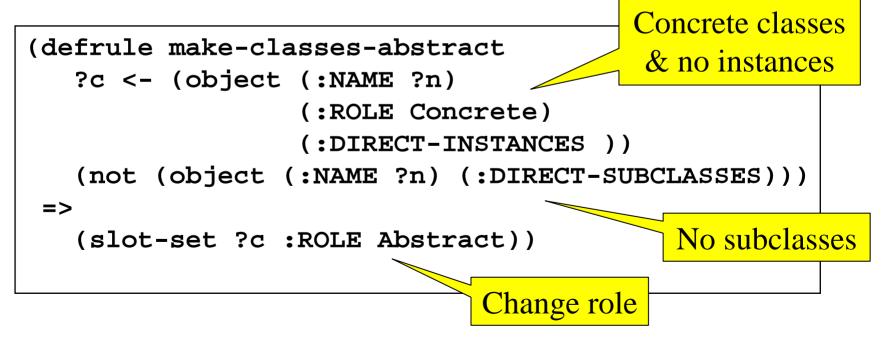
Printing abstract classes in Protégé



Print matches

Modifying ontologies

Change the role to *abstract* for classes that have subclasses, but do not have any instances:



Support for methods and message handlers

(defmethod add ((?a STRING) (?b STRING))

```
(defmethod add ((?a MyClass) (?b MyClass))
...)
```

(defmessage-handler MyClass get-foo ()
 ?self:foo)

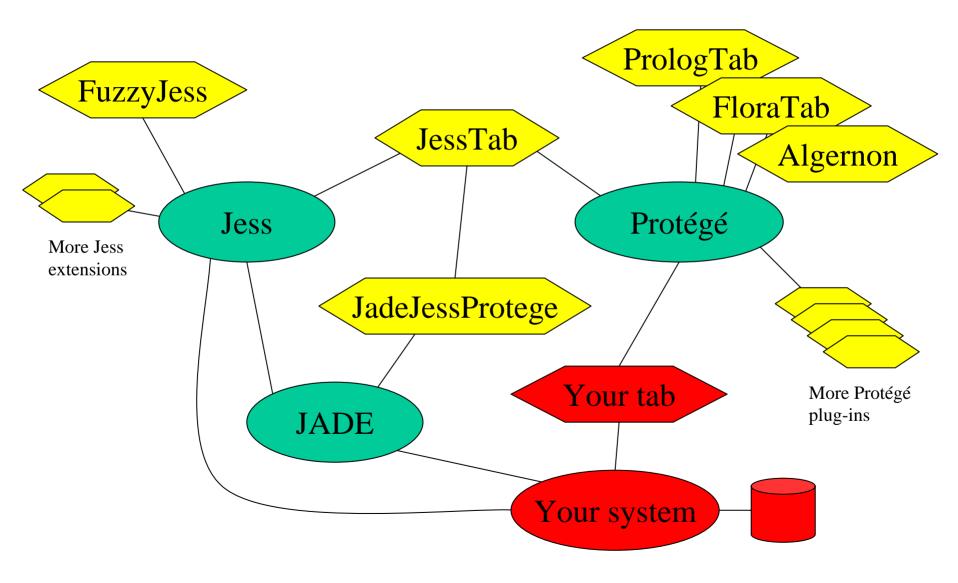
(defmessage-handler rectangle find-area ()
 (* ?self:side-a ?self:side-b))

Examples of applications

- Ontology engineering and reengineering

 Jess as macro/scripting for ontologies
- Importing ontologies – Jess as input filter
- Semantic Web
- Problem-solving methods
- Agent frameworks
 - JadeJessProtege

Tool Web/Library



Ideas for future work

- Support for managing Protégé forms
- Improved GUI
- Multiple Jess engines
- Multiple knowledge bases
- Aspect-oriented functionality (e.g., pointcut for message-handlers)
- ???

Trying JessTab

- Obtain Protégé
 - Download from <u>http://protege.stanford.edu/</u>
- Obtain Jess
 - Download from <u>http://herzberg.ca.sandia.gov/jess/</u>
 - License required (commercial or free academic)
 - Compilation required
- Get JessTab
 - Download from http://www.ida.liu.se/~her/JessTab/

Summary

- JessTab: Protégé Jess integration
- Manage Protégé ontologies and knowledge bases from Jess
- Rule-based reasoning in Protégé
- Protégé as graphical, object-oriented extension to Jess

