

More on Description

Dan Rochowiak  
[drochowi@cs.uah.edu](mailto:drochowi@cs.uah.edu)

---

---

---

---

---

---

---

---

### The Jess deftemplate Construct

```
(deftemplate template-name
  ["Documentation comment"]
  [(declare (slot-specific TRUE | FALSE)
            (backchain-reactive TRUE | FALSE)
            (from-class class name)
            (include-variables TRUE | FALSE)
            (ordered TRUE | FALSE))]
  (slot | multislot slot-name
    [(type ANY | INTEGER | FLOAT | NUMBER |
          SYMBOL | STRING |
          LEXEME | OBJECT | LONG)]
    [(default default value)]
    [(default-dynamic default value)]*))
```

---

---

---

---

---

---

---

---

### The Jess defquery Construct

```
(defquery query-name
  ["Documentation comment"]
  [(declare (variables variable+)
            (node-index-hash value)
            (max-background-rules value))]
  (conditional element*) )
```

---

---

---

---

---

---

---

---

## Parts of a Jess Database

- Each Jess engine holds a collection of knowledge items called facts. This collection is known as the working memory.
- Every fact has a template. The template has a name and a set of slots, and each fact gets these things from its template. The template is like the class of a Java object, or like a relational database table. The slots are like the properties of the JavaBean, or the columns of a table. A fact is therefore like a single JavaBean, or like a row in a database table. You can think of it either way.
- The defquery construct is used to create a special kind of rule with no right-hand-side. Queries are used to search the working memory under direct program control.

---

---

---

---

---

---

---

---

## A Jess Database Example

```
Jess> (deftemplate person (slot firstName) (slot
lastName) (slot age))
Jess> (defquery search-by-name
"Finds people with a given last name"
(declare (variables ?ln))
(person (lastName ?ln) (firstName ?fn) (age ?age)))
Jess> (deffacts data
(person (firstName Fred) (lastName Smith) (age 12))
(person (firstName Fred) (lastName Jones) (age 9))
(person (firstName Bob) (lastName Thomas) (age 32))
(person (firstName Bob) (lastName Smith) (age 22))
(person (firstName Pete) (lastName Best) (age 21))
(person (firstName Pete) (lastName Smith) (age 44))
(person (firstName George) (lastName Smithson) (age 1))
)
```

---

---

---

---

---

---

---

---

## A Jess Database Example

```
Jess> (reset)
Jess> (bind ?result (run-query* search-by-name
Smith))

Jess> (while (?result next)
(printout t (?result getString fn) " " (?result
getString ln)
" , age " (?result getInt age) crlf))

Fred Smith, age 12
Bob Smith, age 22
Pete Smith, age 44
FALSE
```

---

---

---

---

---

---

---

---

### SQL Recipe

- Create database for the domain
- Create table for the concept with definitions of columns (fields)
  - Ideally you might create another table that represents the concepts and their relations.
- Insert data creating an instance of the concept.
- Use select and where to find desired instances

---

---

---

---

---

---

---

---

### About Description and Ontology

Based on Ontology 101

- Sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontologies.
  - For example, suppose several different Web sites contain medical information. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.
  - Note that the Web is simply a distribution device. Similar considerations might apply in attaching to a database or even in a specialized single user mode

---

---

---

---

---

---

---

---

### About Description and Ontology

- Enabling reuse of domain knowledge is one of the driving forces behind recent ontology research.
  - For example, models for many different domains need to represent the notion of time. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops such an ontology in detail, others can simply reuse it for their domains.
  - This is a very high-level example. You might also consider an ontology for a computer program or car manufacturing. Reuse is the key.

---

---

---

---

---

---

---

---

### About Description and Ontology

- Making explicit domain assumptions underlying an implementation makes it possible to change these assumptions easily if our knowledge about the domain changes. Hard-coding assumptions about the world in programming-language code makes these assumptions not only hard to find and understand but also hard to change.
  - This is a common concern for all information processing. The things being modeled may change and the implicit assumptions may become explicit and change.

---

---

---

---

---

---

---

---

### About Description and Ontology

- Separating the domain knowledge from the operational knowledge is another common use of ontologies.
  - One can describe the task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves.
    - Develop an ontology of PC-components and characteristics and apply the algorithm to configure made-to-order PCs.
  - A similar idea can be found in the development of a computer program; the data may change but if the program is well parameterized the operations may remain the same. The reverse is also possible.

---

---

---

---

---

---

---

---

### About Description and Ontology

- Analyzing domain knowledge is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable when both attempting to reuse existing ontologies and extending them.
  - It should be noted that this idea is also important for the testing, validation and configuration of any product, including a software product.

---

---

---

---

---

---

---

---

### About Description and Ontology

- An ontology of a domain is rarely a goal in itself. Developing an ontology is akin to defining a set of data and their structure for other programs to use. Problem-solving methods, domain-independent applications, and software agents use ontologies and knowledge bases built from ontologies as data.
  - A research task may emphasize the development of a general ontology, say, an ontology of time. However, there is always present the idea that the ontology will be used by some program

---

---

---

---

---

---

---

---

### Practical Concerns

- In practical terms, developing an ontology includes:
  - defining classes in the ontology,
  - arranging the classes in a taxonomic (subclass–superclass) hierarchy,
  - defining slots and describing allowed values for these slots,
  - filling in the values for slots for instances.

---

---

---

---

---

---

---

---

### Practical Concerns

- There is no one correct way to model a domain - there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.
- Ontology development is necessarily an iterative process.
- Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

---

---

---

---

---

---

---

---

Current Assignment

- The tool that use for building the ontology is less important than the ontology you build.
  - A database tool (Derby, Access) or a specific ontology building tool (Protégé) is fine. In fact a standard word processing document with tables will do. Do not get overly excited about the tool.
  - As noted at the beginning of this class Jess itself may furnish the tool that you can use.

---

---

---

---

---

---

---

Current Assignment

- The domain of programming or software development must be narrowed.
- The narrowing will be done by making assumptions and specifying constraints.
  - This is part of the assignment since it requires you to make these items explicit.
  - The narrowing may be done in many ways. You might look at the function of the software, the parts of a program, the structuring of the code, the process of building the code and so on.

---

---

---

---

---

---

---

Current Assignment

- The keys in building your descriptions is to use the notion of a hierarchy and the notion of a composition of parts.
  - This means that there will be links of the form is-a, has-a and has-part.
  - There may be several hierarchies. There may not be one inheritance root for all of the things you will describe. (Yes you can make one formally, but it might not be practical.)

---

---

---

---

---

---

---

### Current Assignment

- Examine the notions of deftemplate and defquery in Jess to see what the notion of “computable” will mean in the course.
  - No matter what tool you use eventually we will need create templates so that we can create facts and reason about them.
- Since these constructs are available we will assume that the use of these constructs will produce “computable” descriptions.
  - As we will see many things can be added to the evolving assignment as we go along.
  - You might want to think of ways that your term project can use this assignment and expand on it.

---

---

---

---

---

---

---

---