Ontolingua Tutorial

Adam Farquhar
Knowledge Systems Lab
Stanford University
<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-10:30</td>
<td>Introduction; what is an ontology? Promise of ontologies. How can we describe ontologies? How can we use and reuse ontologies? What sorts of ontologies are currently available?</td>
</tr>
<tr>
<td>10:30-11:00</td>
<td>break</td>
</tr>
<tr>
<td>12:00-13:00</td>
<td>lunch</td>
</tr>
<tr>
<td>13:00-14:30</td>
<td>Hands-on I: Ontology Building</td>
</tr>
<tr>
<td>14:30-15:00</td>
<td>break</td>
</tr>
<tr>
<td>15:00-15:30</td>
<td>Review Progress. Tool use and modeling.</td>
</tr>
<tr>
<td>15:30-16:30</td>
<td>Hands-on II: Ontology Building</td>
</tr>
<tr>
<td>16:30-17:30</td>
<td>Review Progress. Tool use and modeling.</td>
</tr>
</tbody>
</table>
## Schedule - Day 2

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00-10:30</td>
<td>Supporting technologies: ontology servers, GFP, translation.</td>
</tr>
<tr>
<td>10:30-11:00</td>
<td><em>break</em></td>
</tr>
<tr>
<td>11:00-12:00</td>
<td>Research and development directions.</td>
</tr>
<tr>
<td>12:00-13:00</td>
<td><em>lunch</em></td>
</tr>
</tbody>
</table>
To communicate, plan, think we need a conceptualization of the world

» What kinds of things are there? What are their properties? What are their relationships?

» These things define our ontology

We all have ontologies (e.g., of organizations, computers, animals)

» Some are very idiosyncratic. Some are shared!

Communication and interaction require common shared ontologies
### Ontology - Problems in Communication

- **People, organizations, software programs must communicate**
  - Different needs and backgrounds imply different viewpoints, assumptions, jargon
  - This divergence is natural and valuable
  - But leads to problems in communication, interaction, and understanding

- **Explicit ontologies are crucial for**
  - Communication
  - Education
  - Interoperation
  - Integration
  - Adaptive agents
Researchers in molecular biology need to share results and check consistency between their models, data, and reported models and data.

The Riboweb project (Stanford, SMI)
  » Building an ontology for ribosomes, models, data, reports
    – Molecular structure, experimental data, tests, …
  » Encoding (by hand) relevant literature
Ontology - Example

- Doctors, clinics, hospitals, insurance companies, government agencies need to share information
  - Clinical guidelines, drug interactions, covered procedures, best practices
- Several efforts are addressing aspects of this problem
  - UMLS (unified medical language system)
  - SNOMED (standard nomenclature for medicine)
There are many workflow management systems available.

In order to share information across them and support interoperation, we need to define an integrated ontology that covers:

- Processes, resources, products, services, organizations

Several groups are involved in such an effort:

- NIST, WfMC, PIF, TOVE
Collaborative engineering projects need to communicate across discipline boundaries.

Several projects (e.g., PACT, Boeing) have worked to build ontologies for the subdisciplines and span them.

Goals include:

- Automated notifications on design modifications
- Cross-disciplinary simulation
- Improved design process
Natural language understanding and machine translation need large ontologies of

- Linguistic categories
- Roles
- Common-sense objects

Several projects (WordNet, Pangloss, CyC) have built large, sparse ontologies covering a large portion of common vocabulary.
Large sparse ontologies seem to be widely reusable
  » They represent a large investment
  » They make very few commitments

Small detailed ontologies seem to be most useful
  » The may represent a large investment of careful research
  » They make key commitments
Ontology - Example

- What is your ontology for documents?
- What sorts of things does it include?
- What important subfields does it touch?
Our ontologies are mostly implicit, hidden
» This works when they are already shared within a community

An explicit ontology is an artifact that must be constructed, structure, manipulated

Many artifacts have a strong ontological flavor
» Glossary, dictionary, encyclopedia
» Data dictionary, class library, database schema
» Knowledge base
Ontology As Artifact

- An explicit ontology becomes an artifact itself
- Artifacts have incidental properties
  - Syntax
  - Design decisions
  - Scoping decisions
- A good choice of representation language can reduce the incidental properties, but they can never be eliminated
- Do conceptualizations have the same sorts of incidental properties?
Ontology - Summary

- Explicit ontologies support
  - Shared understanding among people
  - Interoperability between tools
  - Systems engineering
  - Reusability
  - Declarative specification
Example: Ontology of Organizations

- **Concepts**
  - Authority, empowerment, commitment, cooperation, achievement

- **Competency questions**
  - What resources does X have authority to assign?
  - Is X allowed to perform activity Y in any situation?
  - What goals is person X committed to achieving?
  - Is a goal achievable by an agent given its current commitments and the commitments of other agents?
  - What goals are solitarily unachievable for a given agent?
Example: Organization Ontology (U. Toronto)

- Organization unit
- Organization position
- Agent
- Goal
- Policy
- Ability
- Authority
- Commitment
- Empowered
- Achievable goal
- Solitarily unachievable goal
Example: Process Interchange Format

- PIF is a bridge among heterogeneous process representations
- The core PIF ontology is supported by all targets
- Partially shared views support interoperation between similar systems
A standard terminology is evolving which can serve as a common framework for different workflow management system vendors.

The wfmc glossary

- Contains technical definitions for terms to be used in the workflow management coalition specifications and discussions.
- Helps to establish consistent usage.
- For each term the following is provided:
  - Definition, discussion of usage, set of synonyms.
- The glossary serves as an informal ontology for shared understanding.
Name: process activity

- Definition: A logical step or description of a piece of work that contributes toward the accomplishment of a process. A process activity may be a manual process activity and/or an automated workflow process activity.

- Usage: A process definition generally consists of many process activities that are connected for the purpose of defining a process flow or state transition network.

- Synonyms: step, node, task, work element, process element
The best understood tool for writing down knowledge is logic

» Every sentence has an unambiguous meaning

» Combining sentences also has a clear meaning

» Of course, we may not anticipate the consequences of a set of sentences!

But which sentences do we write? What style do we use? What idioms do we use?

The field of Knowledge Representation (KR) studies methods for writing down knowledge in ways that are computationally useful.
A class can be defined by *any* set of objects

- The set of people in this room
- The set of left-front wheels

But some sets are worth naming

- The class of countries, the class of presidents

A class is a distinguished named set

The members of the set are the instances of the class

- Bill Clinton is an instance of President
- Germany is an instance of Country
A subclass is nothing but a subset

- Graduate student is a subclass of student
  - Is every graduate student a student?

- Ellipse is a subclass of circle
  - Is every ellipse a circle?

- A class can have many subclasses
  - Animal
    - Mammal, Reptile, Fish, ...
    - Male, Female
    - Living, Dead
Some classes are *completely* defined

- A polygon is a triangle iff it has three sides
- A vehicle is a bicycle iff it has pedals and has exactly two wheels

- Mathematical objects, data structures, human artifacts are often completely defined

Some classes are *partially* defined

- Dog, chair, bridge, house, ship, organization
- Sometimes called *primitive concepts*
- Partially defined *basic* concepts characterize a large portion of our everyday world
T-Rex is a subclass of Dinosaur. The T-Rex is extinct. Alice is a T-Rex. Is Alice extinct?

No! Alice is Dead. The species is extinct.

Some classes have instances that are classes

- The class Extinct-species has species for instances. A species is a class of animals.

Classes whose instances are classes are sometimes called meta-classes

Meta-classes are (ontologically) common!
A slot is a relation between two things

- The age of Alan is 25. A son of Alan is Bob. A son of Alan is Conner.
- The average height of Americans is 5’9”. The number of buffalo is 5000.
- Age-of, son-of, average-height, number are slots

Slots that belong to individual objects are called own slots

A class can have an own slot, too!
Some slots apply to every instance of a class. We call these **template slots**. They are also known as instance slots, or attributes.

- Age-of is a template slot of Physical-Thing
- Son-of is a template slot of Animal
- Cardinality-of is a template slot of Set

Template slots summarize information about every instance.

Attributes in programming languages are a kind of template slots.
A facet says something about a slot on a class. Facets can *constrain* the meaning of a slot.

**Common facets**

- The value-type of age on people is non-negative
- The reverse of son-of for animals is parent-of
- The cardinality of biological parents for mammals is 2

**Most programming languages define a small fixed set of facets**
The arity of a relation is the number of arguments it takes.

A class is a unary relation!

- Person(alan)

A slot is a binary relation

- son-of(alan, bob), age-of(alan, 25)

A facet is a ternary relation

- slot-value-type(person, age-of, number)

A function is just a relation whose last argument is uniquely determined by the others (a.k.a. single valued slot)
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame</td>
<td>Any object, including classes, instances, and relations</td>
</tr>
<tr>
<td>Relation</td>
<td>A relation over one or more objects</td>
</tr>
<tr>
<td>Class</td>
<td>A distinguished set of objects</td>
</tr>
<tr>
<td>Instance</td>
<td>A member of a class</td>
</tr>
<tr>
<td>Function</td>
<td>A relation where the last argument is uniquely determined by the others</td>
</tr>
<tr>
<td>Slot</td>
<td>A binary relation</td>
</tr>
<tr>
<td>Own Slot</td>
<td>A slot belonging to a frame</td>
</tr>
<tr>
<td>Template Slot</td>
<td>A slot associated with a class, but with values for instances</td>
</tr>
<tr>
<td>Facet</td>
<td>A ternary relation on a frame, slot, value</td>
</tr>
<tr>
<td>Constraint</td>
<td>Any assertion that constrains the possible interpretation of frames</td>
</tr>
<tr>
<td>Axiom</td>
<td>Any statement taken to be true without proof</td>
</tr>
</tbody>
</table>
Building an ontology appears similar to building an object oriented program, but there are profound differences:

» Classes and objects in a program are about data structures
» Classes and objects in ontologies are about the world

There is often a correspondence between data structures in programs and definitions in an ontology.
Modeling - Reusing Knowledge or Code

- OOP encourages code reuse
- Sometimes code reuse violates ontological principles:
  - Is an ellipse really a subclass of circle?
- Classes in an ontology must reflect the structure of the world, not the structure of the data

```java
class circle {
    int x; int y; int height;
}
class ellipse extends circle {
    int width;
}
```
Modeling - How Many Chars in a Name?

- In a program, it makes sense to define a name as a string of 20 characters
- In an ontology, this is a **bad** idea
- A name is an object with many properties. It is not a string
- Avoid using primitive types as long as possible

```c
char[20] name;
```
In a program, it makes sense to allocate a fixed number of bytes to represent an instance of a class.

In an ontology, this is a bad idea.

The ontology is prior.

It is not advisable to assume:
- Attributes will be stored or computed
- The order of allocation
- Classes will correspond to native classes

The mapping from ontology to application is neither simple nor obvious.
Modeling - Choosing a Language

- The best candidate for an unambiguous language for writing down knowledge is First Order Logic
- But syntax is not important!
  - We can present FOL as an object oriented model
  - Anything which doesn’t fit in the object model can still be said in FOL
- An expressive language makes it easier (possible) to say what we need to say
  - I’ll get you an answer in one hundredth the time, but it will probably be wrong
Formal does not mean precise

Formal does mean that you can understand what you said!

It is completely possible to make imprecise statements in a formal language

- There is a relation between cigarettes and cancer
- The English description of vehicle is “A machine that transports stuff from one place to another”
- Dog and canine are synonyms

An imprecise statement may not mean much, but you know what it means!
Modeling - Style

- Names should not encode meaning
  - bears-of-little-brain, faster-than-a-speeding-bullet
- Try to identify subclasses that *partition* a class
- Try to identify *exhaustive partitions* of a class
- Try to identify multiple exhaustive partitions
- Bundle co-occurring attributes into a class
- Make polymorphic relations and functions
Ontolingua - Getting Started

- The public server is available at http://Ontolingua.Stanford.edu

- Logon
  - Interact with your web browser
  - You must have an account to use the system
    - New accounts are issued immediately

- All work takes place within a session, which has a
  - Duration (logoff cannot be guaranteed)
  - Description (you can have several)
Ontolingua - Getting Started

- User preferences
- Types of pages
  - The Library page
  - The Ontology page
  - The Frame page
    - A frame is a class, instance, relation, or function
    - The browser page
- What’s in a link?
  - Almost everything is a hyperlink
  - All words are hyperlinked to their definition page
## Library of Ontologies

### Preferred Ontologies

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Status</th>
<th>Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compartmental-Modeling</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Federal-Supply-Classification-System</td>
<td>Not loaded</td>
<td>JUST-ME, JTFLOCAL</td>
</tr>
<tr>
<td>Genesereth-Example</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Ih-Base-Ontology</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Ih-Vehicle-Ads</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Ih-Vehicles</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Nomenclator-Ontology</td>
<td>Not loaded</td>
<td>JUST-ME, NOMEN</td>
</tr>
<tr>
<td>Oed-Discussion</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Onto-Standard-Base.96.09.30</td>
<td>Not loaded</td>
<td>JUST-ME, ONTOSTD-BETA</td>
</tr>
<tr>
<td>Onto-Standard-Object-Base.96.09.30</td>
<td>Not loaded</td>
<td>JUST-ME, ONTOSTD-BETA</td>
</tr>
<tr>
<td>Pangloss-Tcp-Level</td>
<td>Not loaded</td>
<td>JUST-ME, ONTOSTD-STD</td>
</tr>
<tr>
<td>Bibliographic-Data-In-Verification</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Car-Rental</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Classified-Advertisements</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Clocks</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Compartmental-Modeling</td>
<td>Not loaded</td>
<td>JUST-ME</td>
</tr>
<tr>
<td>Computer-World</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Cyc-Interface</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Cyc-Top-level-9.20.96</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Cyc-Upper-Level</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Database-Metadata-Terror</td>
<td>Not loaded</td>
<td></td>
</tr>
<tr>
<td>Database-Schema-Terror</td>
<td>Not loaded</td>
<td></td>
</tr>
</tbody>
</table>

Please try to find the time to fill in our User Interface Survey.
An ontology can include other ontologies from the library

Reuse ontological modules!

You can always augment a definition from a module
  » But you can never say less!

Different ontologies may make incompatible extensions
Ontolingua - Viewing an Ontology

Ontology HP-PRODUCT-ONTOLOGY

- Generality: Low
- Maturity: Low
- I/O Syntax: Case Insensitive
- Source code: hp-product-ontology.lisp

There are 10 classes (browser, alphabetical list, indented list)
There are 6 relations
There are 32 functions
There are 76 instances
There are no axioms
View Hp-Product-Ontology in detail

Ontology documentation:

A ontology that contains information about Hewlett Packard test & measurement products. This ontology focuses on oscilloscopes.
Class Tangible-Product

- Defined in Ontology: Product-ontology
- Source code: product-ontology.hsp
- Source pathname: /tmp_mnt/vol/qhitw/cms/frame-editor/ontology-lib

**Axioms for Tangible-Product:**

**Frame References to Tangible-Product:**

In class Product:

Subclass-Partition: \( \circ \) \{Service, Tangible-Product\}
Ontolingua - Viewing a Relation

**Relation Has-Model-Number**
- Defined in Ontology: *Product-ontology*
- Source code: *product-ontology.lisp*
- Source pathname: /tmp_mnt/vol/q/htw/cms/frame-editor/ontology

**Arity:** 2

**Documentation:** A relation that indicates the model number of a product.

**Domain:** Product

**Instance-Of:** Binary-Relation, Relation, Set

**Range:** Model-Number

**Function Maximum-Storage-Temperature**
- Defined in Ontology: *Product-ontology*
- Source code: *product-ontology.lisp*
- Source pathname: /tmp_mnt/vol/q/htw/cms/frame-editor/ontology-library/ontologies

**Arity:** 2

**Documentation:** The maximum temperature at which this object should be stored.

**Domain:** Tangible-Product

**Instance-Of:** Binary-Relation, Function, Relation, Set

**Range:** Temperature-Quantity
Ontolingua - Modifying a Definition

Make sure that you are in edit mode!

- Edit pens allow you to modify values
- Addition gadgets allow you to add slots, facets, values

Class Product

- Defined in Ontology: Product-ontology
- Source code: product-ontology.lisp
- Source pathname: /tmp/mnt/vol/q/hlw/cms/frame-editor/ontology-library/ont...

Arity: 1 +Facet
Documentation: A class of objects that are typically bought and sold. +Value +Facet
Domain-Of: Has-Model-Number, Has-Special-Discount, Has-Warranty, List-
+Facet
Instance-Of: Class, Relation, Set +Value +Facet
Range-Of: Object-Sold, Product-With-This-Warranty +Value +Facet
Subclass-Of: Individual-Thing +Value +Facet
Subclass-Partition: (Service, Tangible-Product) +Value +Facet
Superclass-Of: Product-Previously-Owned, Service, Service-Agreement, Tang-
+Class Slot

Slots:

- Associated-Documents: +Value +Facet
- Has-Model-Number: +Value +Facet
- Has-Special-Discount: +Value +Facet
- Has-Warranty: +Value +Facet
- List-Price: +Value +Facet
- Net-Weight: +Value +Facet
- Instance Slot
No pull-down menus, so we use a selection menu together with a submit button

There are separate editing and viewing modes

Toggle using the browse/edit toggle button to switch between them

Typically, you only edit one object at a time

The edit mode looks like the browse mode, but with type-in fields snipped into the page
Ontolingua - Basic Principles

- Presentation versus representation
  - Object-Oriented presentation, full logical representation
- Creating a new object is different from saying new things about existing objects
- Include existing ontologies from the library
- Augment, but do not edit included definitions
- Measure twice, cut once!
### Ontolingua - the Frame Ontology

- **The Frame Ontology** defines the basic concepts for a rich object-oriented representation.

- **Almost all ontologies** should include the Frame Ontology.

- **Things**
  - A thing can be anything, including a set.
  - An individual-thing is not a set.
  - Most common-sense things are individual-things according to the frame ontology.

<table>
<thead>
<tr>
<th>Class</th>
<th>All-Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arity</td>
<td>Subclass-Of</td>
</tr>
<tr>
<td>Subclass-Partition</td>
<td>Class-Partition</td>
</tr>
<tr>
<td>Facet</td>
<td>Thing</td>
</tr>
<tr>
<td>Individual-Thing</td>
<td></td>
</tr>
</tbody>
</table>
Ontolingua - How do I Save My Work

- Your work is stored on the server
- You may download your work to a local file system using your browser
- You may email your work to any email address
- Any ontology may be translated into several different target languages
Once you have created a subclass, you can use frame>copy to create siblings.

Do not reference a frame before creation.

Completion will help with long names.

Find will help you find frames in any loaded ontology.

You can edit anything except the basic type of an object after creation.

Rename a frame or ontology by editing its name.
Values may include strings containing html

Strings are automatically hyperlinked
- Use a frame name with dashes, or put it in all caps
- Handle plurals in text with ‘s (tangible-product’s)
- To hyperlink against a definition use the word tag in an anchor `<a word=individual-thing>Object</a>`

You can hyperlink a subtree against an ontology
- `<a href="http://a.b.com/helpfile.html" usingontology>Help</a>`
The web interface handles domain differently in different contexts

- When creating a slot, multiple values for domain are interpreted as a union.
- When creating a general relation, domain values are assigned to arguments in turn.
- Adding a value to the domain slot of a relation is simple conjunction (as are all multiple values).
It may appear that arity is handled inconsistently between relations and functions:

- $R(x,y) - R$ has arity 2
- $F(x,y) - F$ has arity 3!

The term $F(x,y)$ is equivalent to the predicate $z = F(x,y)$, which can be written in relational form:

- $F(x,y,z) - F$ does have arity 3
Design - Basic Procedure

- Sketch out objectives
- Identify example scenarios
- Identify core competency questions

- Sketch out the space
- Incorporate knowledge sources
- Identify possible design decisions
- Iterate, increasing coverage and fidelity

- This is a pragmatic, effective procedure with little methodological overhead
Design - Sketch out Objectives

- How much coverage is required?
- What tasks are anticipated?
- Do existing sources need to be considered?
- What depth of axiomatization is needed?
Design - Identify Example Scenarios

- Good scenarios will
  - Appeal to a sufficiently broad community
  - Include a variety of tasks
  - Not be too detailed - narrow focus will produce a narrow brittle ontology
  - Be useful for documentation and understanding in addition to guiding design
  - Help readers and developers understand the scope and intentions of the authors

- Story problems with intuitively understandable solutions
Design - Competency of an Ontology

- Identify competency questions
- The ontology must define the knowledge necessary to answer the questions
- Don’t worry about how the questions will be answered!

For documents:

» What is the difference between a particular book, the first edition of a book, the second edition of a book, and a translation into another language?
» Find all books in a collection that have been written by McCarthy
» Find all books whose publisher was in San Francisco, but is no longer in business
Design - Sketch out the Space

- What are fifty key concepts?
- What are some of their properties?
- What are some of the relations between them?
## Design - Find and Incorporate Sources

- Almost all interesting domains already have extensive existing descriptions
  - Text books
  - Professional societies
  - Data dictionaries
  - Standards documents
  - Encyclopedias, thesauri, dictionaries
  - Earlier attempts at formalization (data dictionaries, metadata, standard codes)

- Due diligence pays off in buy-in, acceptance, completeness

- Examples: Standard Industry Codes (SICS), Federal Classification System (FSC), ASME publications
Design - Triad

- Users contribute tasks
  - Cover 80%
- Domains are infinite
  - Cover standard terms
- Data provides lower bound
  - Minimal complete coverage
Practicum - Getting Started

- Customize browser settings
  - Check document every time
  - Small, readable font
  - Minimal menubar to maximize vertical space

- Create an account

- Customize user preferences
  - Menus>Full
  - Ontology Library>accessible in header
  - Images>Show unbordered icons
Practicum - Basic Use

- Look at an ontology in the library
  - HP-Product-Ontology
  - Use class browser/slot browser
  - Try focusing on Characterization-Oscilloscope

- Create a new ontology
  - Return to library page first!

- Create a class
- Create a subclass (undo/redo)
- Create a slot on the class
- Assert a facet on the slot
Practicum - Getting Serious

- Follow ontology design steps
- Sketch out (on paper) an ontology of documents for your task
- Enter document ontology
- Refine and extend ontology
USE - Ontology Server Architecture

- **Three modes of use**
  - Runtime query
  - Translation
  - Development
Remote applications may query an ontology server
  » To determine if a term is defined
  » To determine the relationship between terms
  » To manipulate the contents of an ontology

Typical applications include
  » Browsing and editing tools
  » Ontology analysis tools
  » Knowledge based applications
  » Mediators
An AIDS treatment protocol advisor developed at Stanford’s SMI

Matches patients with treatment protocols

Queries the ontology server to determine relationships between terms
  » Is zidovine an antiretroviral drug?

Uses a substantial ontology of drugs and treatments

Uses the generic frame protocol (GFP) to access the server
The Generic Frame Protocol (GFP) provides a common object oriented API to knowledge representation systems

- CLIPS, Classic, Theo, Loom, Ocelot, Sipe, Ontolingua, Clos, ...

Features

- Queries, updates, network access
- Multiple language bindings (C, Lisp, Java)
- Function specifier language for server-side execution
- Layered architecture
Use - GFP Knowledge Model

- Similar to the representation defined earlier
- Frames can be class, slot, facet, instance, value
- Assertions are conjunctive
- Multiple inheritance
- Open set of facets for defining constraints
## Use - GFP Operations

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get-kbs</td>
<td>GFP defines a wide range of operations</td>
</tr>
<tr>
<td>find-kb-of-type</td>
<td>Access to data and schema</td>
</tr>
<tr>
<td>get-frame-matching</td>
<td>Access to classes, instances, and metaclasses</td>
</tr>
<tr>
<td>get-class-direct-supers</td>
<td>The underlying system need not be a KB!</td>
</tr>
<tr>
<td>get-class-supers</td>
<td>The underlying system need not be object oriented</td>
</tr>
<tr>
<td>get-instance-direct-types</td>
<td></td>
</tr>
<tr>
<td>get-slot-value</td>
<td></td>
</tr>
<tr>
<td>get-slot-values</td>
<td></td>
</tr>
<tr>
<td>get-frame-slots</td>
<td></td>
</tr>
<tr>
<td>get-frame-facets</td>
<td></td>
</tr>
<tr>
<td>get-facet-values</td>
<td></td>
</tr>
<tr>
<td>put-slot-values</td>
<td></td>
</tr>
<tr>
<td>put-facet-values</td>
<td></td>
</tr>
<tr>
<td>add-slot-value</td>
<td></td>
</tr>
<tr>
<td>delete-frame</td>
<td></td>
</tr>
<tr>
<td>create-kb</td>
<td></td>
</tr>
</tbody>
</table>
Use - GFP Layered Architecture

- Full protocol defines about 200 operations
- For a simple read-only back end, about 15 operations must be defined
- For a simple read-write back end, about 35 operations must be defined
- All other operations have default implementations
  - get-kb-classes can be defined using get-kb-frames and class-p
  - add-slot-value can be defined using get-slot-values and put-slot-values
Use - GFP Optimization

- Network protocols need to provide methods for achieving reasonable performance
- **Caching**
  - The Lisp and Java clients implement caching-network-kbs that cache the results of all GFP operations
  - Prefetching is currently being developed
- **Server-side Execution**
  - Function specifiers allow for compound operations to be executed on the server
if (kb = find_ontology("FRAME-ONTOLOGY", connection)) {
    frames = gfp_get_class_direct_subs(frame, kb, gfp_true);
    if (frames) {
        slots = gfp_get_frame_slots(frame, kb,
                                   gfp_all, gfp_all, gfp_true);
    }
    if (slots) {
        print_frame(frame, slots, frames);
    }
} else if (glast_error_code!=GFP_NO_ERROR)
    explain_error();
Use - GFP Java Code

- GFP KB is the primary class
  » CachingKB, NetworkKB, TupleKB
- Currently developing Java editing apps using

```java
kb.get_slot_values (frame, slot,
    GfpKb._T,    // local-only-p
    GfpKb._auto, // slot-type
    GfpKb._all); // #values
```
Translation is a challenging problem

- Semantics - ensure that the meaning is preserved
- Syntax - ensure that target syntax is correct
- Style - ensure that target idioms are preserved
Use - Current Translation Technology

Class c2RailBridge

- Defined in Ontology: Target-schema_1.0
- Source code: target-schema_1.0.lisp
- Source pathname: /tmp_mnt/vol/q/htw/cms

SUBCLASS-OF: c2RailLink
SUPERCLASS-OF: c2RRBrdg
INSTANCE-OF: CLASS, INTERFACE, root
ARITY: 1

Slots:

location:
  SLOT-VALUE-TYPE: string
root_name:
  SLOT-VALUE-TYPE: string
root_version:

IDL Translation of c2RailBridge

interface c2RailBridge : c2RailLink {
};

Class c2Entity

- Defined in Ontology: Target-schema_1.0
- Source code: target-schema_1.0.lisp
- Source pathname: /tmp_mnt/vol/q/htw/cms

SUBCLASS-OF: c2Root
SUPERCLASS-OF: c2Computer, c2Countermeasure
INSTANCE-OF: CLASS, INTERFACE, root
ARITY: 1

Has-ATTRIBUTE: location

Slots:

location:
  SLOT-VALUE-TYPE: string
root_name:
  SLOT-VALUE-TYPE: string
root_version:

IDL Translation of c2Entity

interface c2Entity : c2Root {
  attribute string location;
};
Three modes of use

- Runtime query
- Translation
- Development
Remote applications may query an ontology server
  » To determine if a term is defined
  » To determine the relationship between terms
  » To manipulate the contents of an ontology

Typical applications include
  » Browsing and editing tools
  » Ontology analysis tools
  » Knowledge based applications
  » Mediators
An AIDS treatment protocol advisor developed at Stanford’s SMI
Matches patients with treatment protocols
Queries the ontology server to determine relationships between terms
   » Is zidovine an antiretroviral drug?
Uses a substantial ontology of drugs and treatments
Uses the generic frame protocol (GFP) to access the server
The Generic Frame Protocol (GFP) provides a common object oriented API to knowledge representation systems

- CLIPS, Classic, Theo, Loom, Ocelot, Sipe, Ontolingua, Clos, ...

Features

- Queries, updates, network access
- Multiple language bindings (C, Lisp, Java)
- Function specifier language for server-side execution
- Layered architecture
Use - GFP Knowledge Model

- Similar to the representation defined earlier
- Frames can be class, slot, facet, instance, value
- Assertions are conjunctive
- Multiple inheritance
- Open set of facets for defining constraints
## Use - GFP Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get-kbs</code></td>
<td>GFP defines a wide range of operations</td>
</tr>
<tr>
<td><code>find-kb-of-type</code></td>
<td>Access to data and schema</td>
</tr>
<tr>
<td><code>get-frame-matching</code></td>
<td>Access to classes, instances, and metaclasses</td>
</tr>
<tr>
<td><code>get-class-direct-supers</code></td>
<td>The underlying system need not be a KB!</td>
</tr>
<tr>
<td><code>get-class-supers</code></td>
<td>The underlying system need not be object oriented</td>
</tr>
<tr>
<td><code>get-instance-direct-types</code></td>
<td></td>
</tr>
<tr>
<td><code>get-slot-value</code></td>
<td></td>
</tr>
<tr>
<td><code>get-slot-values</code></td>
<td></td>
</tr>
<tr>
<td><code>get-frame-slots</code></td>
<td></td>
</tr>
<tr>
<td><code>get-frame-facets</code></td>
<td></td>
</tr>
<tr>
<td><code>get-facet-values</code></td>
<td></td>
</tr>
<tr>
<td><code>put-slot-values</code></td>
<td></td>
</tr>
<tr>
<td><code>put-facet-values</code></td>
<td></td>
</tr>
<tr>
<td><code>add-slot-value</code></td>
<td></td>
</tr>
<tr>
<td><code>delete-frame</code></td>
<td></td>
</tr>
<tr>
<td><code>create-kb</code></td>
<td></td>
</tr>
</tbody>
</table>
Use - GFP Layered Architecture

- Full protocol defines about 200 operations
- For a simple read-only back end, about 15 operations must be defined
- For a simple read-write back end, about 35 operations must be defined
- All other operations have default implementations
  - `get-kb-classes` can be defined using `get-kb-frames` and `class-p`
  - `add-slot-value` can be defined using `get-slot-values` and `put-slot-values`
Use - GFP Optimization

- Network protocols need to provide methods for achieving reasonable performance
- Caching
  - The Lisp and Java clients implement caching-network-kbs that cache the results of all GFP operations
  - Prefetching is currently being developed
- Server-side Execution
  - Function specifiers allow for compound operations to be executed on the server
Use - GFP C Code

```c
if (kb = find_ontology("FRAME-ONTOLEGY", connection)) {
    frames=gfp_get_class_direct_subs(frame, kb, gfp_true);
    if(frames){
        slots=gfp_get_frame_slots(frame, kb,
                                  gfp_all, gfp_all, gfp_true);
        if(slots){
            print_frame(frame,slots,frames);}
    }
    else {explain_error();}
}
```
Use - GFP Java Code

- GFP KB is the primary class
  - CachingKB, NetworkKB, TupleKB
- Currently developing Java editing apps using

```java
kb.get_slot_values (frame, slot,
    GfpKb._T,    // local-only-p
    GfpKb._auto, // slot-type
    GfpKb._all); // #values
```
Translation is a challenging problem

- Semantics - ensure that the meaning is preserved
- Syntax - ensure that target syntax is correct
- Style - ensure that target idioms are preserved
Class c2RailBridge

- Defined in Ontology: Target-schema_1.0
- Source code: target-schema_1.0.lisp
- Source pathname: /tmp_mnt/vol/q/htw/cms

SUBCLASS-OF: c2RailLink
SUPERCLASS-OF: c2RRBrdg
INSTANCE-OF: CLASS, INTERFACE, @, BOUN
ARITY: 1

Slots:

location:
  SLOT-VALUE-TYPE: @, string

root_name:
  SLOT-VALUE-TYPE: @, string

root_version:

IDL Translation of c2RailBridge

interface c2RailBridge : c2RailLink {
    ...
};

Class c2Entity

- Defined in Ontology: Target-schema_1.0
- Source code: target-schema_1.0.lisp
- Source pathname: /tmp_mnt/vol/q/htw/cms

SUBCLASS-OF: c2Root
SUPERCLASS-OF: c2Computer, c2Countermeasure
INSTANCE-OF: CLASS, INTERFACE, @, BOUN
HAS-ATTRIBUTE: location
ARITY: 1

Slots:

location:
  SLOT-VALUE-TYPE: string

root_name:
  SLOT-VALUE-TYPE: string

root_version:

IDL Translation of c2Entity

interface c2Entity : c2Root {
    attribute string location;
};