Semantic Networks in Prolog

Tim Finin
University of Maryland
Baltimore County

class(thing).
class(person).
class(man).
class(woman).
class(integer).
isa(integer,thing).
isa(person,thing).
isa(man,person).
isa(woman,person).
age(person,integer).
parent(person,person).
inverse(person,child).
child(person,person).
inverse(child,parent).
sex(man,male).
isa(john,man).
age(john,25).
parent(john,mary).

class/1 is true for nodes
isa/2 captures the subclass relation
FOO/2 where FOO is the name of an arc asserts an arc between two nodes

class/1 is true for nodes
isa/2 captures the subclass relation
FOO/2 where FOO is the name of an arc asserts an arc between two nodes

Slot facets: type, cardinality, inverse, value, etc.
**Syntactic Sugar**

A person is a thing with
1. age with type integer,
2. sex with type one of (male, female),
3. parent with type person and inverse child,
   child with type person.

John is a man with
   age = 25,
   parent = Mary.

**Inheritance**

- A logical model of inheritance is easy to implement.

```
is(C, C) :- class(C).

is(C1, C2) :- isa(C1, C2).

is(C1, C2) :- isa(C1, X), is(X, C2).
```

- Characteristics: everything that is true for a class is true for all its subclasses and individual members. (i.e., no defaults, shadowing, overriding)

**Lots of issues**

- Detecting inconsistencies
- Own slots vs. inherited slots
- Instances vs. classes
- Subslots
  - e.g.: father is a subslot of parent, i.e.,
    father(X, Y) => parent(X, Y).
- Defaults
  - e.g.: hasa(person, numberArms, default, 2)
- Attached procedures
  - e.g.: if-added, if-removed, if-needed, truth maintenance
- Attached arbitrary axioms
- When to do inferencing, caching stuff