Taxonomic Reasoning
Let us meet RBG

https://youtu.be/vRRwM95aRj0
Example

SC: Is a hot dog a sandwich?
RBG: Are you asking me? Or, you tell me what a sandwich is and I will tell you if a hot dog is a sandwich?
SC: A sandwich is two pieces of bread with almost any type of filling in between as long as it is not more bread.
RBG: You said two pieces of bread. Does that include a roll that has been cut open, but still not completely?
SC: See, that is the crux, and you have gotten it immediately. See this is why you are on the supreme court. That immediately gets to the question does a roll need to be separated into two parts. Because a sub sandwich ... a sub is not split, and yet it is a sandwich.
RBG: Yes.
SC: Is hot dog a sandwich?
RBG: On your definition it is.
Example

**SC:** Is a hot dog a sandwich?

**RBG:** Are you asking me? Or, you tell me what a sandwich is and I will tell you if a hot dog is a sandwich?

**SC:** A sandwich is two pieces of bread with almost any type of filling in between as long as it is not more bread.

**RBG:** You said two pieces of bread. Does that include a roll that has been cut open, but still not completely?

**SC:** See, that is the crux, and you have gotten it immediately. See this is why you are on the supreme court. That immediately gets to the question does a roll need to be separated into two parts. Because a sub sandwich ... a sub is not split, and yet it is a sandwich.

**RBG:**

**SC:** Is hot dog a sandwich?

**RBG:** On your definition it is.
Taxonomic Reasoning

• Useful when we can organize knowledge into classes
  • Class membership
  • Class specialization
  • Disjoint classes
  • Value restriction
  • Inheritance
### Example of classes

<table>
<thead>
<tr>
<th>Male</th>
<th>Female</th>
<th>Class</th>
<th>Instance of</th>
<th>Class</th>
<th>Instance of</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>bea</td>
<td>male</td>
<td>(art, male)</td>
<td>female</td>
<td>(bea, female)</td>
</tr>
<tr>
<td>bob</td>
<td>coe</td>
<td>male</td>
<td>(bob, male)</td>
<td>female</td>
<td>(coe, female)</td>
</tr>
<tr>
<td>cal</td>
<td>cory</td>
<td>male</td>
<td>(cal, male)</td>
<td>female</td>
<td>(cory, female)</td>
</tr>
<tr>
<td>cam</td>
<td></td>
<td>male</td>
<td>(cam, male)</td>
<td>female</td>
<td></td>
</tr>
</tbody>
</table>
Class specialization

• We can organize classes into a hierarchy
  • e. g., male and female are subclasses of person
    subclass-of(female, person)
    subclass-of(male, person)
  • subclass relationship is transitive
    subclass_of(A,C) :- subclass_of(A,B) & subclass_of(B,C)
  • subclass and instance-of relationships are related
    instance_of(I,B) :- subclass_of(A,B) & instance_of(I,A)
Disjoint classes

• Classes can be declared to be disjoint
  • e.g, disjoint(male,female)
  • i.e., they do not have any instances in common
    illegal :- disjoint(A,B) & instance_of(I,A) & instance_of(I,B)
    or
    illegal(“Disjoint classes cannot have an instance in common”) :-
    disjoint(A,B) & instance_of(I,A) & instance_of(I,B)
Class Definition

• Necessary properties of a class
  • Will have instance-of in the body of the rule
    has_hair_color(X,brown) :-
    instance_of(X,brown_haired_person)

• Sufficient properties of a class
  • Will have instance-of in the head of the rule
    instance_of(X,brown_haired_person) :-
    instance_of(X,person) &
    has_hair_color(X,brown)
Value Restriction

• We can restrict the arguments of a relation to be instances of a specific class
  • domain is the restriction on the first argument
    illegal :- domain(parent,person) &
    parent(X,Y) &
    ~instance_of(X,person)
  • range is the restriction on the second argument
    illegal :- range(parent,person) &
    parent(X,Y) &
    ~instance_of(Y,person)
Cardinality and Number Restrictions

• We can further restrict the values of relations by specifying cardinality and number restrictions
  • A cardinality restriction limits the number of values of a relation
    illegal :- instance_of(X,person) & ~countofall(P,parent(P,X),2)
  • A numeric range restriction limits the minimum and maximum value
    illegal :- instance_of(X,person) & age(X,Y) & min(0,Y)
    illegal :- instance_of(X,person) & age(X,Y)& min(125,Y,100)
Inheritance

• The relation values are said to inherit to the instances of a class
  • If art is an instance of the class brown-haired-person, we can conclude that art has brown hair
Taxonomic Inference

- Given two classes A and B, whether A is a subclass of B?
- Given a class A and an instance I, whether I is an instance of A?
- Given a ground relation atom determine whether it is true or false?
- Given a relation atom, determine values which values make it true?
Is hot dog a sandwich?

A sandwich is two pieces of bread with almost any type of filling in between as long as it is not more bread
a roll that has been cut open, but still not completely?
a sub is not split, and yet it is a sandwich

Constants

hotdog, sandwich, bread, filling, sub, roll, half_cut_roll
Defining the taxonomy

class(hotdog) subclass_of(roll, bread)
class(sandwich) subclass_of(half_cut_roll, roll)
class(bread) disjoint(bread, filling)
class(slice) subclass_of(half_cut_roll, roll)
class(filling) subclass_of(half_cut_roll, roll)
class(sub) subclass_of(half_cut_roll, roll)
class(roll) subclass_of(half_cut_roll, roll)
class(half_cut_roll)
Defining a Sandwich

A sandwich is two pieces of bread with almost any type of filling in between as long as it is not more bread

```
instance_of(X,sandwich_with_slice) :-
    has_part(X,B1) & instance_of(B1,slice) &
    has_part(X,B2) & instance_of(B2,slice) & distinct(B1,B2) &
    has_part(X,Z) & instance_of(Z,filling)

subclass_of(sandwich_with_slice,sandwich)
```
Extending the definition of Sandwich

a sub is not split, and yet it is a sandwich

instance_of(X, sandwich_with_roll) :-
    has_part(X, B1) & instance_of(B1, half_cut_roll) &
    has_part(X, Z) & instance_of(Z, filling)

subclass_of(sandwich_with_roll, sandwich)
subclass_of(sub, sandwich_with_roll)
Is hot dog a sandwich?

has_part(s1,B1)
instance_of(B1,half_cut_roll)
has_part(s1,Z)
instance_of(Z,filling)

Is instance_of(s1,sandwich) true?
class(filling)
class(hotdog)
class(sandwich)
class(sandwich_with_roll)
class(sandwich_with_slice)
class(bread)
class(slice)
disjoint(filling, bread)

class(hotdog)
class(sandwich)
class(sandwich_with_roll)
class(sandwich_with_slice)
class(bread)
class(slice)
disjoint(filling, bread)

all_subclass_of(X, Y) :- subclass_of(X, Y)
all_subclass_of(X, Y) :- all_subclass_of(X, Z) & subclass_of(Z, Y)
all_instance_of(I, B) :- instance_of(I, B)
all_instance_of(I, B) :- all_instance_of(I, A) & all_subclass_of(A, B)
subclass_of(roll, bread)
subclass_of(half_cut_roll, roll)
subclass_of(sandwich_with_roll, sandwich)
subclass_of(sandwich_with_slice, sandwich)
subclass_of(sub, sandwich_with_roll)
subclass_of(hotdog, sandwich_with_roll)
instance_of(s1, hotdog)

instance_of(X, sandwich_with_slice) :-
    instance_of(B1, slice) & has_part(X, B1) &
    instance_of(B2, slice) & has_part(X, B2) &
    instance_of(B, filling) & has_part(X, B)

instance_of(X, sandwich_with_roll) :-
    has_part(X, B1) & instance_of(B1, half_cut_roll)
    has_part(X, B2) & instance_of(B2, filling)
Project Suggestions

- Create a Smart Document for Internal Revenue Service
- Build a comprehensive taxonomic reasoner on top of Epilog
Smart Documents

Smart Document = Traditional Document + Logic Program

Demo:
Example Definitions

1. Mortgages you took out on or before October 13, 1987 (called grandfathered debt).
2. Mortgages you (or your spouse if married filing a joint return) took out after October 13, 1987, to buy, build, or improve your home (called home acquisition debt), but only if throughout 2016 these mortgages plus any grandfathered debt totaled $1 million or less ($500,000 or less if married filing separately).
3. Mortgages you (or your spouse if married filing a joint return) took out after October 13, 1987, that are home equity debt but that are not home acquisition debt, but only if throughout 2016 these mortgages totaled $100,000 or less ($50,000 or less if married filing separately) and totaled no more than the fair market value of your home reduced by (1) and (2).

The above definitions can be captured as logic programming rules as shown below.

```
view(loan_type(X), "Grandfathered Debt") :- loan(X) & has_value(before_Oct_13_1987(X))
view(loan_type(X), "Home acquisition Debt") :- loan(X) & ~has_value(before_Oct_12_1987(X), true) & value(loan_purpose(X), buy_home) | value(loan_purpose(X), build_home) | value(loan_purpose(X), improve_home)
view(loan_type(X), "Home equity debt") :- loan(X) & ~has_value(before_Oct_12_1987(X)) & value(loan_purpose(X), other)
```
Example Taxonomy

- Money
  - Debt
    - Home equity debt
    - Home acquisition debt
    - Grandfathered debt
  - Loan
    - Home equiv loan
    - Line of credit
    - Mortgage
      - Second mortgage
  - Deduction
  - Premium
    - Insurance premium
      - Mortgage insurance premium
  - Interest
    - Deductible interest
    - Interest point
    - Mortgage interest
      - Home mortgage interest
A comprehensive taxonomic reasoner has over 200 different operations
• Computing subclasses, instances, disjoint classes
• Computing relation values
• Checking constraint violations
• Updating the rule base