Logic Programming

Datasets

Michael Genesereth
Computer Science Department
Stanford University
Datasets

**Dataset** - collection of simple facts about state of "world"
- Facts in dataset are assumed to be true
- Facts not in dataset are assumed to be false

**Role #1 - Datasets as logic programs**
- used by themselves as standalone databases
- used in combination with rules to form complex programs

**Role #2 - Datasets as basis for semantics of logic programs**
Basics
**Conceptualization**

**Objects** - e.g. people, companies, cities
  - concrete (*person*) or abstract (*number, set, justice*)
  - primitive (*computer chip*) or composite (*car*)
  - real (*earth*) or fictitious (*Sherlock Holmes*)

**Relationships**
  - properties of objects or relationships among objects
  - e.g. Joe *is a person*
  - e.g. Joe *is the parent of Bill*
  - e.g. Joe *likes Bill more than Harry*
Graphical Representation

```
  art
 /   \
/     /
 bob   bea
 /     \
cal    cam
     /  \
    cat  coe
```
<table>
<thead>
<tr>
<th>parent</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>bob</td>
</tr>
<tr>
<td>art</td>
<td>bea</td>
</tr>
<tr>
<td>bob</td>
<td>cal</td>
</tr>
<tr>
<td>bob</td>
<td>cam</td>
</tr>
<tr>
<td>bea</td>
<td>cat</td>
</tr>
<tr>
<td>bea</td>
<td>coe</td>
</tr>
</tbody>
</table>
Art is the parent of Bob.
Art is the parent of Bea.
Bob is the parent of Cal.
Bob is the parent of Cam.
Bea is the parent of Cat.
Bea is the parent of Coe.
parent(art, bob)
parent(art, bea)
parent(bob, cal)
parent(bob, cam)
parent(bea, cat)
parent(bea, coe)
Constants are strings of lower case letters, digits, underscores, and periods or strings of arbitrary ascii characters within double quotes.

Examples:

```
  joe, bill, cs151, 3.14159
  person, worksfor, office.occupant
  the_house_that_jack_built,
  "Mind your p’s & q’s!"
```

Non-examples:

```
  Art, p&q, the-house-that-jack-built
```

A set of constants is called a vocabulary.
Types of Constants

Symbols / object constants represent objects.
   joe, bill, harry, a23, 3.14159
   the_house_that_jack_built
   “Mind your p’s & q’s!”

Constructors / function constants represent functions.
   cell, pair, triple, set

Predicates / relation constants represent relations.
   person, parent, prefers
The **arity** of a constructor or a predicate is the number of arguments that can be associated with the constructor or predicate in writing complex expressions in the language.

**Unary** predicate (1 argument): \texttt{person(joe)}

**Binary** predicate (2 arguments): \texttt{parent(art,bob)}

**Ternary** predicate (3 arguments): \texttt{prefers(art,bob,bea)}

In talking about vocabulary, we sometimes notate the arity of a constructor or predicate by annotating with a slash and the arity, e.g. \texttt{male/1}, \texttt{parent/2}, and \texttt{prefers/3}. 

In some logic programming languages (e.g. Prolog), types and arities determine syntactic legality; and they are enforced by interpreters and compilers.

In other languages (e.g. Epilog), types and arities suggest their intended use. However, they do not determine syntactic legality, and they are not enforced by interpreters and compilers.

In our examples, we use Epilog; but, in this course, we specify types and arities where appropriate and we try to adhere to them.
A datum / factoid is an expression formed from an $n$-ary predicate and $n$ symbols enclosed in parentheses and separated by commas.

Symbols: $a, b$
Predicate: $p/2, q/1$

Sample Datum: $p(a, a)$
Sample Datum: $p(a, b)$
Sample Datum: $q(a)$
Sample Datum: $q(b)$
The **Herbrand base** for a vocabulary is the set of all factoids that can be formed from the vocabulary.

Symbols:  \( a, b \)
Predicate: \( p/2, q/1 \)

Herbrand Base:
\[
\{p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b)\}
\]
A **dataset** is any *set of factoids* that can be formed from a vocabulary, i.e. a subset of the Herbrand base.

Symbols: $a, b$
Predicates: $p/2, q/1$
Herbrand Base:

$$\{p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b)\}$$

Dataset: $\{p(a,b), p(b,a), q(a)\}$
Dataset: $\{\}$
Dataset: $\{p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b)\}$

We use datasets to characterize states of the world. The *facts in a dataset are assumed to be true* and those that are *not in the dataset are assumed to be false*. 
Exercise

Vocabulary
Symbols: a, b
Predicates: p/2, q/1

Questions
How many symbols in our vocabulary?
How many elements in the Herbrand base?
How many possible datasets?
Spelling carries no meaning in logic programming (except as informal documentation for programmers).

\[
\begin{align*}
&\text{parent(art,bob)} \\
&\text{parent(bob,cal)} \\
&\text{p(a,b)} \\
&\text{p(b,c)} \\
&\text{coulish(widget,gadget)} \\
&\text{coulish(gadget,framis)}
\end{align*}
\]

The meaning of a constant in logic programming is determined solely by the sentences that mention it.
The order of arguments in an instance of a relation is determined by one’s understanding of the relation.

Example:

\[ \text{prefers(art,bea,bob)} \]

For me, this sentence means that Art prefers Bea to Bob. Other interpretations are possible; the important thing is to be consistent - once you choose, stick with it.
Kinship
Parentage

\[
\begin{array}{c}
\text{art} \\
\text{bob} & \text{bea} \\
\text{cal} & \text{cam} & \text{cat} & \text{coe}
\end{array}
\]
Kinship Relations

art

bob

cal cam cat coe

bea

art

bob

cal cam cat coe

bea

art

bob bea

cal cam cat coe

bea

art

bob bea

cal cam cat coe

bea
Degenerate Relations

\[ \text{art} \quad \text{bob} \quad \text{bea} \]

\[ \text{cal} \quad \text{cam} \quad \text{cat} \quad \text{coe} \]
parent(art,bob)
parent(art,bud)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
grandparent(art, cal)
grandparent(art, cam)
grandparent(art, cat)
grandparent(art, coe)
sibling(bob, bea)
sibling(bea, bob)
sibling(cal, cam)
sibling(cam, cal)
sibling(cat, coe)
sibling(coe, cat)
ancestor(art,bob)
ancestor(art,bea)
ancestor(art,cal)
ancestor(art,cam)
ancestor(art,cat)
ancestor(art,coe)
ancestor(bob,cal)
ancestor(bob,cat)
ancestor(bea,cat)
ancestor(bea,coe)
Other Relations

Unary Relations:

- male(art)
- male(bob)
- male(cal)
- male(cam)
- female(bea)
- female(cat)
- female(coe)

Ternary Relations:

- prefers(art, bob, bea)
- prefers(bob, cam, cal)
- prefers(bea, cat, coe)
Some relations definable in terms of others
e.g. we can define grandparent in terms of parent
e.g. we can define sibling in terms of parent
e.g. we can define ancestor in terms of parent
e.g. we can define parent in terms of ancestor
See upcoming material on view definitions

Some combinations of arguments do not make sense
e.g. parent(art,art)
e.g. parent(art,bob) and parent(bob,art)
e.g. old(art) and young(art)
See upcoming material on constraints
Blocks World
Blocks World

- A
- B
- C
- D
- E

Diagram showing a configuration where A is on top of B, which is on top of C, and D is on top of E.
Symbols: $a, b, c, d, e$

Unary Predicates:
- clear - blocks with no blocks on top.
- table - blocks on the table.

Binary Predicates:
- on - pairs of blocks in which first is on the second.
- above - pairs in which first block is above the second.

Ternary Predicates:
- stack - triples of blocks arranged in a stack.
clear(a)
clear(d)
table(c)
table(e)
on(a,b)
on(b,c)
on(d,e)
above(a,b)
above(b,c)
above(a,c)
above(d,e)
stack(a,b,c)
University
### University

<table>
<thead>
<tr>
<th>Students:</th>
<th>Departments:</th>
<th>Faculty:</th>
<th>Years:</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaron</td>
<td>architecture</td>
<td>alan</td>
<td>freshman</td>
</tr>
<tr>
<td>belinda</td>
<td>computers</td>
<td>cathy</td>
<td>sophomore</td>
</tr>
<tr>
<td>calvin</td>
<td>english</td>
<td>donna</td>
<td>junior</td>
</tr>
<tr>
<td>george</td>
<td>physics</td>
<td>frank</td>
<td>senior</td>
</tr>
</tbody>
</table>

**Predicate:**

\[
\text{student}(\text{Student}, \text{Department}, \text{Advisor}, \text{Year})
\]

**Dataset:**

- student(aaron,architecture,alan,freshman)
- student(belinda,computers,cathy,sophomore)
- student(calvin,english,donna,junior)
- student(george,physics,frank,senior)
Suppose a student has not declared a major. What if a student does not have an advisor?

Leave out fields (syntactically illegal):

\[
\text{student(aaron,,freshman)}
\]

Add suitable values to vocabulary (new symbol):

\[
\text{student(aaron,undeclared,orphan,freshman)}
\]

Database nulls (new linguistic feature):

\[
\text{student(aaron,null,null,freshman)}
\]
Suppose a student has two majors.

**Multiple Rows (storage, update inconsistencies):**

```plaintext
student(calvin,english,donna,junior)
student(calvin,physics,donna,junior)
```

**Multiple fields (storage, extensibility?):**

```plaintext
student(calvin,english,physics,donna,junior)
student(george,physics,physics,frank,senior)
```

**Use compound symbols:**

```plaintext
student(calvin,english_physics,donna,junior)
```
Represent wide relations as collections of binary relations.

**Wide Relation:**

```
student(Student,Department,Advisor,Year)
```

**Binary Relations:**

```
major(Student,Department)
advisor(Student,Faculty)
year(Student,Year)
```

Always works when there is a field of the wide relation (called the **key**) that uniquely specifies the values of the other elements. If none exists, possible to create one.
Triples

major(aaron,architecture)
advisor(aaron,alan)
year(aaron,freshman)

year(belinda,sophomore)

major(calvin,english)
major(calvin,physics)
advisor(calvin,donna)
year(calvin,senior)

major(george,physics)
advisor(george,frank)
year(george,senior)
Classes

student, department, faculty, year

Attributes (binary relations associated with a class):

- major(Student, Department)
- advisor(Student, Faculty)
- year(Student, Year)

Properties of Attributes:

- **domain** is class of objects in first position (*arguments*)
- **range** is class of objects in second position (*values*)
- **unique** if *at most* one value for each argument
- **total** if *at least* one value for each argument
**Subtlety**

**Missing information**
there is a value but we do not know it.
e.g. Aaron has an advisor but we do not know who it is.

**Non-existent value**
there is no value
e.g. Aaron does not have an advisor.

*For now, in talking about datasets, we assume full info.*
*If a value is missing, it means that there is no value.*
Sales
In 2015, Art sold Arborhouse to Bob for $1000000.
In 2016, Bob sold Pelicanpoint to Carl for $2000000.
In 2016, Carl sold Ravenswood to Dan in $2000000.
In 2017, Dan sold Ravenswood to Art for $3000000.
# Real Estate Ledger

<table>
<thead>
<tr>
<th>People</th>
<th>Properties</th>
<th>Years</th>
<th>Money</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>arborhouse</td>
<td>2015</td>
<td>1000000</td>
</tr>
<tr>
<td>bob</td>
<td>pelicanpoint</td>
<td>2016</td>
<td>2000000</td>
</tr>
<tr>
<td>carl</td>
<td>ravenswood</td>
<td>2017</td>
<td>3000000</td>
</tr>
<tr>
<td>dan</td>
<td>arborhouse</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relation Constant:**

\[
sale(Year, Seller, Property, Buyer, Amount)
\]

**Dataset:**

- sale(2015, art, arborhouse, bob, 1000000)
- sale(2016, art, pelicanpoint, bob, 2000000)
- sale(2016, carl, ravenswood, dan, 2000000)
- sale(2017, dan, arborhouse, art, 3000000)
In 2015, Art sold Arborhouse to Bob for $1000000.
In 2016, Bob sold Pelicanpoint to Carl for $2000000.
In 2016, Carl sold Ravenswood to Dan in $2000000.
In 2017, Dan sold Ravenswood to Art for $3000000.

In 2015, Art sold Bob a widget for $10.
In 2016, Art sold Bob a gadget for $20.
In 2016, Art sold Bob a gadget for $20.  
In 2017, Art sold Bob a framis for $30.  

Different sale!
### People:
<table>
<thead>
<tr>
<th>People</th>
<th>Items</th>
<th>Years</th>
<th>Money</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>widget</td>
<td>2015</td>
<td>10</td>
</tr>
<tr>
<td>bob</td>
<td>gadget</td>
<td>2016</td>
<td>20</td>
</tr>
<tr>
<td>carl</td>
<td>framis</td>
<td>2017</td>
<td>30</td>
</tr>
<tr>
<td>dan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Relation Constant:

\[
sale(\text{Year},\text{Seller},\text{Item},\text{Buyer},\text{Amount})
\]

### Dataset:

- sale(2015, art, widget, bob, 10)
- sale(2016, art, gadget, bob, 20)
- sale(2016, art, gadget, bob, 20) \textit{Duplicate factoid}!
- sale(2017, art, framis, bob, 30)
<table>
<thead>
<tr>
<th>Sales:</th>
<th>People:</th>
<th>Items:</th>
<th>Years:</th>
<th>Money:</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>art</td>
<td>widget</td>
<td>2015</td>
<td>10</td>
</tr>
<tr>
<td>t2</td>
<td>bob</td>
<td>gadget</td>
<td>2016</td>
<td>20</td>
</tr>
<tr>
<td>t3</td>
<td>carl</td>
<td>framis</td>
<td>2017</td>
<td>30</td>
</tr>
<tr>
<td>t4</td>
<td>dan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Relation Constant:**

\[
\text{sale}(Sale, Year, Seller, Item, Buyer, Amount)
\]

**Dataset:**

\[
\begin{align*}
\text{sale}(t1, 2015, art, widget, bob, 10) \\
\text{sale}(t2, 2016, art, gadget, bob, 20) \\
\text{sale}(t3, 2016, art, gadget, bob, 20) \\
\text{sale}(t4, 2017, art, framis, bob, 30)
\end{align*}
\]
Compound Names
We sometimes want to talk about complex objects made up of simpler structures.

Examples:
- the list of a, b, and c
- the cell in row 2 and column 3

Alternative 1: Symbols (structure implicit):
- the_list_of_a_b_c
- cell_2_3

Alternative 2: Compound names (structure explicit):
- [a,b,c]
- cell(2,3)
Types of Constants

Symbols / object constants represent objects.

joe, bill, harry, a23, 3.14159
the_house_that_jack_built
“Mind your p’s & q’s!”

Constructors / function constants

cell, pair, triple, set

Predicates / relation constants represent relations.

person, parent, prefers
Symbols / object constants represent objects.
  joe, bill, harry, a23, 3.14159
  the_house_that_jack_built
  “Mind your p’s & q’s!”

Constructors / function constants
  cell, pair, triple, set

Predicates / relation constants represent relations.
  person, parent, prefers
A **compound name** is an expression formed from an *n*-ary constructor and *n* symbols enclosed in parentheses and separated by commas.

Symbols:  \( a, b \)  
Constructor:  \( f/2, g/1 \)

Compound Names:  \( f(a,b), f(b,a), g(a), g(b) \)

*This allows us to refer to complex objects made up of simple objects. How do we refer to complex objects made up of other complex objects?*
A **compound name** is an expression formed from an *n*-ary *constructor* and *n* *symbols* or *compound names* enclosed in parentheses and separated by commas.

Symbols: \( a, b \)
Constructor: \( f/2, g/1 \)

- Compound Names: \( f(a,b), f(b,a), g(a), g(b) \)
- Compound Names: \( f(g(a),b), g(f(a,b)) \)
- Compound Names: \( g(g(a)), g(f(g(a),g(b))) \)
- Compound Names: \( g(g(g(a))) \)
A ground term is either a symbol or a compound name.

The adjective ground here means that the term does not contain any variables (which we discuss in later lessons).
The **Herbrand universe** for a vocabulary is the set of **all ground terms** that can be formed from the **symbols** and **constructors** in the vocabulary.
A **datum / factoid** is an expression formed from an \( n \)-ary predicate and \( n \) ground terms enclosed in parentheses and separated by commas.

Symbols:  \( a, b \)
Constructor: \( f/2, g/1 \)
Predicate: \( p/2 \)

Sample Datum: \( p(a, g(a)) \)
Sample Datum: \( p(f(a, b), g(b)) \)
The **Herbrand universe** for a vocabulary is the set of all *ground terms* that can be formed from the *symbols* and *constructors* in the vocabulary.

The **Herbrand base** for a vocabulary is the set of all factoids that can be formed from the vocabulary.

A **dataset** is any *set of factoids* that can be formed from a vocabulary, i.e. a subset of the Herbrand base.
Exercise

Vocabulary
Symbols: $a, b$
Predicates: $p/2, q/1$

Questions
How many symbols in the Herbrand universe?
How many elements in the Herbrand base?
How many possible datasets?
Vocabulary
Symbols: a, b
Constructor: f/1, g/1
Predicates: p/2, q/1

Questions
How many elements in the Herbrand universe?
How many elements in the Herbrand base?
How many possible datasets?
Sierra
Sierra is browser-based IDE (interactive development environment) for Epilog.

- Saving and loading files
- Visualization of datasets
- Querying datasets
- Transforming datasets
- Interpreter (for view definitions, action definitions)
- Trace capability (useful for debugging rules)
- Analysis tools (error checking and optimizing rules)

http://epilog.stanford.edu/sierra/sierra.html
Assignments
Required:
  Reading - Datasets

Background:
  Reading - Programs with Common Sense
  Reading - Logic Programming
The goal of this exercise is for you to familiarize yourself with the updates mechanism of Sierra. As always, go to http://epilog.stanford.edu and click on the Sierra link.

In a separate window, open the documentation for Sierra. To access the documentation, go to http://epilog.stanford.edu, click on Documentation, and then click on the Sierra item on the resulting drop-down menu.

Read Sections 1-3 of the documentation and reproduce the examples in the Sierra window you opened earlier. Read section 9 and play around with saving and loading data and configurations.
Assignment 1.2 - Teams

Composition

3 people each (2 or 4 okay with good reason)

Names:

Pansy Division
The Pumamen
Team Camembert
Mighty Bourgeoisie
Greedy Bastards
Red Hot Chili Peppers
/*v*/
X Æ A-12
Michael Genesereth
Consider a vocabulary that includes the following relations.

\texttt{movie.instance}(x) means that \( x \) is a movie.
\texttt{actor.instance}(x) means that \( x \) is an actor.
\texttt{director.instance}(x) means that \( x \) is a director.
\texttt{year.instance}(x) means that \( x \) is a year.
\texttt{title.instance}(x) means that \( x \) is a title.

\texttt{movie.star}(x,y) means that movie \( x \) stars actor \( y \).
\texttt{movie.director}(x,y) means that movie \( x \) was directed by \( y \).
\texttt{movie.year}(x,y) means that movie \( x \) was released in year \( y \).
\texttt{movie.title}(x,y) means that movie \( x \) has the title \( y \).

Choose symbols for a few movies, actors, directors, years, and titles, and encode the relevant data about these entities using this vocabulary.
Consider a vocabulary that includes the following relations.

\texttt{type.instance(x)} means that \(x\) is a type.
\texttt{type.predicate(x,y)} means that type \(x\) has predicate \(y\).
\texttt{type.attribute(x,y)} means that type \(x\) was attribute \(y\).

\texttt{predicate.instance(x)} means that \(x\) is a predicate.
\texttt{predicate.domain(x,y)} means that predicate \(x\) has domain \(y\).

\texttt{attribute.instance(x)} means that \(x\) is an attribute.
\texttt{predicate.domain(x,y)} means that attribute \(x\) has domain \(y\).
\texttt{predicate.codomain(x,y)} means that \(x\) has codomain \(y\).
\texttt{predicate.total(x,yes)} whether \(x\) has at least one value.
\texttt{predicate.unique(x,yes)} whether \(x\) has at most one value.

Use this vocabulary to encode types and relations in movie vocabulary.
Use the vocabulary in Assignment 1.4 to describe itself.

Factoids describing type below. Your job is to do other types, predicates, attributes, and booleans.

\[
\text{type}\cdot\text{instance}(\text{type}) \\
\text{type}\cdot\text{predicate}(\text{type},\text{type}\cdot\text{instance}) \\
\text{type}\cdot\text{attribute}(\text{type},\text{type}\cdot\text{predicate}) \\
\text{type}\cdot\text{attribute}(\text{type},\text{type}\cdot\text{attribute}) \\
\ldots
\]

Yes, the *predicates* in our vocabulary are *symbols* in this vocabulary as well as *predicates*!
Term Project

Criteria:
Inherent interest of application (25%)
Difficulty (25%)
Appropriate use of Logic Programming (50%)

Topics:
May be same as others or coordinate with others
May be same as example in class or new and different

Deliverables:
running code / Sierra configuration / etc.
project presentation (last two class meetings)
final report (end of quarter)

http://logicprogramming.stanford.edu/assignments/project/index.html