Introduction to Logic

Relational Logic

Michael Genesereth
Computer Science Department
Stanford University
Premises:
   *If Jack knows Jill, then Jill knows Jack.*
   *Jack knows Jill.*

Conclusion:
   *Is it the case that Jill knows Jack?*
Premises:

If one person knows another, then the second person knows the first.

Jack knows Jill.

Conclusion:

Is it the case that Jill knows Jack?

How do we represent the first premise in a way that allows us to derive the desired conclusion?
New Linguistic Features:
  Variables
  Quantifiers

Sample Sentence:

$$\forall x. \forall y. (knows(x,y) \Rightarrow knows(y,x))$$
Syntax

Semantics

Examples, Examples, Examples

Properties of Sentences

Logical Entailment

Decidability
Syntax
Components of Language

Words

\[ a, b, g, p \]

Terms

\[ g(a,a) \]

Sentences

\[ \forall x. (p(x) \Rightarrow p(x,g(x,x))) \]
Words are strings of letters, digits, and occurrences of the underscore character.

Variables begin with characters from the end of the alphabet (from $u$ through $z$).

\[ u, v, w, x, y, z \]

Constants begin with digits or letters from the beginning of the alphabet (from $a$ through $t$).

\[ a, b, c, 123, comp225, barack\_obama \]
Constants

Object constants represent objects.

joe, stanford, usa, 2345

Relation constants represent relations.

knows, loves
The *arity* of a relation constant is the number of arguments it takes.

*Unary* relation constant - 1 argument

*Binary* relation constant - 2 arguments

*Ternary* relation constant - 3 arguments

*n*-ary relation constant - *n* arguments
Signatures

A signature consist of a set of object constants and a set of relation constants together with a specification of arity for the relation constants.

Object Constants: \(a, b\)

Unary Relation Constant: \(p\)
Binary Relation Constant: \(q\)
A term is either a variable or an object constant.

Terms represent objects.

Terms are analogous to noun phrases in natural language.
Three types of sentences in Relational Logic:

Relational sentences - analogous to the simple sentences in natural language

Logical sentences - analogous to the logical sentences in natural language

Quantified sentences - sentences that express the significance of variables
A *relational sentence* is an expression formed from an $n$-ary relation constant and $n$ terms enclosed in parentheses and separated by commas.

$$q(a,y)$$

Relational sentences are *not* terms and *cannot* be nested in relational sentences.

No! $q(a,q(a,y))$  No!
Logical sentences in Herbrand Logic are analogous to those in Propositional Logic.

\[
\begin{align*}
(\neg q(a,b)) \\
(p(a) \land p(b)) \\
(p(a) \lor p(b)) \\
(q(x,y) \Rightarrow q(y,x)) \\
(q(x,y) \Leftrightarrow q(y,x))
\end{align*}
\]
Quantified Sentences

Universal sentences assert facts about all objects.

\[
(\forall x. (p(x) \Rightarrow q(x,x)))
\]

Existential sentence assert the existence of objects with given properties.

\[
(\exists x. (p(x) \land q(x,x)))
\]

Quantified sentences can be nested within other sentences.

\[
(\forall x. p(x)) \lor (\exists x. q(x,x)) \\
(\forall x. (\exists y. q(x,y)))
\]
Parentheses can be removed when precedence allows us to reconstruct sentences correctly.

Precedence relations same as in Propositional Logic with quantifiers being of *higher* precedence than logical operators.

\[
\forall x. p(x) \Rightarrow q(x,x) \rightarrow (\forall x. p(x)) \Rightarrow q(x,x)
\]

\[
\exists x. p(x) \land q(x,x) \rightarrow (\exists x. p(x)) \land q(x,x)
\]
An expression is *ground* if and only if it contains no variables.

Ground sentence: 

\[ p(a) \]

Non-Ground Sentence: 

\[ \forall x. p(x) \]
An occurrence of a variable is **bound** if and only if it lies in the scope of a quantifier of that variable. Otherwise, it is **free**.

\[ \exists y. q(x, y) \]

In this example, \( x \) is free and \( y \) is bound.
A sentence is **open** if and only if it has free variables. Otherwise, it is **closed**.

**Open sentence:**

\[ \exists y. q(x, y) \]

**Closed Sentence:**

\[ \forall x. \exists y. q(x, y) \]
Semantics
The *Herbrand base* for a Relational language is the set of all ground relational sentences that can be formed from the vocabulary of the language.
Object Constants: $a, b$
Unary Relation Constant: $p$
Binary Relation Constant: $q$

Herbrand Base:

$$\{p(a), p(b), q(a,a), q(a,b), q(b,a), q(b,b)\}$$
A truth assignment is an association between ground atomic sentences and the truth values true or false. As with Propositional Logic, we use 1 as a synonym for true and 0 as a synonym for false.

\[
\begin{align*}
p(a)^i &= 1 \\
p(b)^i &= 0 \\
q(a,a)^i &= 1 \\
q(a,b)^i &= 0 \\
q(b,a)^i &= 1 \\
q(b,b)^i &= 0
\end{align*}
\]
A *sentential truth assignment* is an association between arbitrary sentences in a Herbrand language and the truth values 1 and 0.

<table>
<thead>
<tr>
<th>Truth Assignment</th>
<th>Sentential Truth Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(a)^i = 1$</td>
<td>$(p(a) \lor p(b))^i = 1$</td>
</tr>
<tr>
<td>$p(b)^i = 0$</td>
<td>$(p(a) \land \neg p(b))^i = 1$</td>
</tr>
</tbody>
</table>

Each base truth assignment leads to a particular sentential truth assignment based on the type of sentence.
Logical Sentences

$(-\varphi)^i = 1$ if and only if $\varphi^i = 0$

$(\varphi \land \psi)^i = 1$ if and only if $\varphi^i = 1$ and $\psi^i = 1$

$(\varphi \lor \psi)^i = 1$ if and only if $\varphi^i = 1$ or $\psi^i = 1$

$(\varphi \Rightarrow \psi)^i = 1$ if and only if $\varphi^i = 0$ or $\psi^i = 1$

$(\varphi \Leftrightarrow \psi)^i = 1$ if and only if $\varphi^i = \psi^i$
An instance of an expression is an expression in which all free variables have been consistently replaced by ground terms.

Consistent replacement here means that, if one occurrence of a variable is replaced by a ground term, then all occurrences of that variable are replaced by the same ground term.
A *universally quantified sentence* is true for a truth assignment if and only if every instance of the scope of the quantified sentence is true for that assignment.

An *existentially quantified sentence* is true for a truth assignment if and only if some instance of the scope of the quantified sentence is true for that assignment.
Truth Assignment:

\[ p(a)^i = 1 \quad q(a,a)^i = 1 \]
\[ p(b)^i = 0 \quad q(a,b)^i = 0 \]
\[ q(b,a)^i = 1 \]
\[ q(b,b)^i = 0 \]

Sentence:

\[ \forall x. (p(x) \Rightarrow q(x,x)) \]

Instances:

\[ p(a) \Rightarrow q(a,a) \]
\[ p(b) \Rightarrow q(b,b) \]
Truth Assignment:
\[
p(a)^i = 1 \quad q(a,a)^i = 1
\]
\[
p(b)^i = 0 \quad q(a,b)^i = 0
\]
\[
p(b)^i = 0 \quad q(b,a)^i = 1
\]
\[
q(b,b)^i = 0
\]

Sentence:
\[
\forall x. (p(x) \Rightarrow q(x,x))
\]

Instances:
\[
p(a) \Rightarrow q(a,a) \checkmark
\]
\[
p(b) \Rightarrow q(b,b)
\]
Example

Truth Assignment:
\[ p(a)^i = 1 \quad q(a,a)^i = 1 \]
\[ p(b)^i = 0 \quad q(a,b)^i = 0 \]
\[ q(b,a)^i = 1 \]
\[ q(b,b)^i = 0 \]

Sentence:
\[ \forall x. (p(x) \implies q(x,x)) \]

Instances:
\[ p(a) \implies q(a,a) \checkmark \]
\[ p(b) \implies q(b,b) \checkmark \]
Truth Assignment:

\[ p(a)^i = 1 \]
\[ q(a,a)^i = 1 \]
\[ p(b)^i = 0 \]
\[ q(a,b)^i = 0 \]
\[ q(b,a)^i = 1 \]
\[ q(b,b)^i = 0 \]

Sentence:

\[ \forall x. (p(x) \Rightarrow q(x,x)) \]

Instances:

\[ p(a) \Rightarrow q(a,a) \]
\[ p(b) \Rightarrow q(b,b) \]

Example

Truth Assignment:

\[ p(a)^i = 1 \quad q(a,a)^i = 1 \]
\[ p(b)^i = 0 \quad q(a,b)^i = 0 \]
\[ q(b,a)^i = 1 \quad q(b,b)^i = 0 \]

Sentence:

\[ \forall x. \exists y. q(x,y) \]

Instances:

\[ \exists y. q(a,y) \quad \exists y. q(b,y) \]
\[ q(a,a) \quad q(b,a) \]
\[ q(a,b) \quad q(b,b) \]
A truth assignment satisfies *a sentence with free variables* if and only if it satisfies every instance of that sentence. (In other words, we can think of all free variables as being universally quantified.)

\[(\exists y. q(x,y))^i = (\forall x. \exists y. q(x,y))^i\]

A truth assignment satisfies *a set of sentences* if and only if it satisfies every sentence in the set.
Example - Sorority World
<table>
<thead>
<tr>
<th></th>
<th>Abby</th>
<th>Bess</th>
<th>Cody</th>
<th>Dana</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abby</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Bess</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Cody</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dana</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Object Constants: *abby, bess, cody, dana*

Binary Relation Constant: *likes*

Herbrand base has 16 ground relational sentences.
\neg \text{likes}(abby,abby) \quad \neg \text{likes}(bess,abby) \\
\neg \text{likes}(abby,bess) \quad \neg \text{likes}(bess,bess) \\
\text{likes}(abby,cody) \quad \quad \text{likes}(bess,cody) \\
\neg \text{likes}(abby,dana) \quad \neg \text{likes}(bess,dana) \\
\quad \text{likes}(cody,abby) \quad \quad \quad \neg \text{likes}(dana,abby) \\
\text{likes}(cody,bess) \quad \quad \neg \text{likes}(dana,bess) \\
\neg \text{likes}(cody,cody) \quad \quad \quad \text{likes}(dana,cody) \\
\text{likes}(cody,dana) \quad \quad \quad \quad \neg \text{likes}(dana,dana)
Abby likes everyone Bess likes.
If Bess likes a girl, then Abby also likes her.

\[ \forall y. (\text{likes}(\text{bess}, y) \implies \text{likes}(\text{abby}, y)) \]

Cody likes everyone who likes her.
If some girl likes Cody, then Cody likes that girl.

\[ \forall x. (\text{likes}(x, \text{cody}) \implies \text{likes}(\text{cody}, x)) \]
Cody likes somebody who likes her.
There is someone who likes cody and is liked by Cody.

\[ \exists y. (\text{likes}(\text{cody}, y) \land \text{likes}(y, \text{cody})) \]

Nobody likes herself.
It is not the case that someone likes herself.

\[ \neg \exists x. \text{likes}(x, x) \]
Everybody likes somebody.

$$\forall x. \exists y. \text{likes}(x,y)$$

There is somebody whom everybody likes.

$$\exists y. \forall x. \text{likes}(x,y)$$
Example

Abby

Bess

Cody

Dana
Everybody Likes Somebody
Everybody Likes Somebody

Abby  Bess

Cody  Dana
Everybody Likes Somebody

- Abby
- Bess
- Cody
- Dana

Diagram shows connections between individuals:
- Abby likes Bess
- Cody likes Dana
Everybody Likes Somebody
Everybody Likes Somebody
There is Somebody Whom Everyone Likes
Example - Blocks World
Blocks World
Object Constants: $a, b, c, d, e$

Unary Relation Constants:
- *clear* - blocks with no blocks on top.
- *table* - blocks on the table.

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

Ternary Relation Constant:
- *stack* - triples of blocks arranged in a stack.
<table>
<thead>
<tr>
<th></th>
<th>on(a,a)</th>
<th>on(a,b)</th>
<th>on(a,c)</th>
<th>on(a,d)</th>
<th>on(a,e)</th>
<th>¬on(d,a)</th>
<th>¬on(d,b)</th>
<th>¬on(d,c)</th>
<th>¬on(d,d)</th>
<th>on(d,e)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Definitions

Definition of *clear*:

\[ \forall y. (\text{clear}(y) \iff \neg \exists x. \text{on}(x,y)) \]

Definition of *table*:

\[ \forall x. (\text{table}(x) \iff \neg \exists y. \text{on}(x,y)) \]
Definition of *stack*:

\[ \forall x. \forall y. \forall z. (\text{stack}(x,y,z) \iff \text{on}(x,y) \land \text{on}(y,z)) \]

Definition of *above*:

\[ \forall x. \forall z. (\text{above}(x,z) \iff \text{on}(x,z) \lor \exists y. (\text{on}(x,y) \land \text{above}(y,z))) \]

\[ \forall x. \neg \text{above}(x,x) \]
Example - Modular Arithmetic
In Modular Arithmetic of modulus 4 there are just 4 numbers (0, 1, 2, 3).

\[
\begin{align*}
0+0 &= 0 \\
0+1 &= 1 \\
0+2 &= 2 \\
0+3 &= 3 \\
1+0 &= 1 \\
1+1 &= 2 \\
1+2 &= 3 \\
1+3 &= 0 \\
2+0 &= 2 \\
2+1 &= 3 \\
2+2 &= 0 \\
2+3 &= 1 \\
3+0 &= 3 \\
3+1 &= 0 \\
3+2 &= 1 \\
3+3 &= 2
\end{align*}
\]
Object Constants: 0, 1, 2, 3

Binary Relation Constants:
  same - the first and second arguments are identical
  next - the second argument is number after the first

Ternary Relation Constant:
  plus - the third argument is the sum of the first two

  \( \text{plus}(1,2,3) \)
Ground Relational Data:

<table>
<thead>
<tr>
<th></th>
<th>0,0</th>
<th>1,0</th>
<th>2,0</th>
<th>3,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>same</strong></td>
<td>¬</td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>1</td>
<td>¬</td>
<td><strong>same</strong></td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>2</td>
<td>¬</td>
<td>¬</td>
<td><strong>same</strong></td>
<td>¬</td>
</tr>
<tr>
<td>3</td>
<td>¬</td>
<td>¬</td>
<td>¬</td>
<td><strong>same</strong></td>
</tr>
</tbody>
</table>
Ground Relational Data:

<table>
<thead>
<tr>
<th></th>
<th>(0,0)</th>
<th>(1,0)</th>
<th>(2,0)</th>
<th>(3,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>¬next</td>
<td></td>
<td></td>
<td>next</td>
</tr>
<tr>
<td>1</td>
<td>next</td>
<td>¬next</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>¬next</td>
<td>next</td>
<td>¬next</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>¬next</td>
<td>¬next</td>
<td>next</td>
<td>¬next</td>
</tr>
</tbody>
</table>
Ground Relational Data:

\[ next(0,1) \]
\[ next(1,2) \]
\[ next(2,3) \]
\[ next(3,0) \]

Deal with negative literals by saying that all other cases are false. How do we do this?
For every \( x \), there is just one \( y \) such that \( next(x,y) \):

\[
\forall x. \forall y. \forall z. (next(x,y) \land next(x,z) \Rightarrow same(y,z))
\]

Logically equivalent formulation:

\[
\forall x. \forall y. \forall z. (next(x,y) \land \neg same(y,z) \Rightarrow \neg next(x,z))
\]
Ground Relational Data:

\[ \text{plus}(0,0,0) \quad \text{plus}(1,0,1) \quad \text{plus}(2,0,2) \quad \text{plus}(3,0,3) \]
\[ \text{plus}(0,1,1) \quad \text{plus}(1,1,2) \quad \text{plus}(2,1,3) \quad \text{plus}(3,1,0) \]
\[ \text{plus}(0,2,2) \quad \text{plus}(1,2,3) \quad \text{plus}(2,2,0) \quad \text{plus}(3,2,1) \]
\[ \text{plus}(0,3,3) \quad \text{plus}(1,3,0) \quad \text{plus}(2,3,1) \quad \text{plus}(3,3,2) \]

Functionality Axiom:

\[ \forall x. \forall y. \forall z. \forall w. (\text{plus}(x,y,z) \land \text{plus}(x,y,w) \Rightarrow \text{same}(z,w)) \]
Alternative Definition of Addition

Identity:

\[ \forall y. \text{plus}(0, y, y) \]

Successor:

\[
\forall x. \forall y. \forall z. (\text{plus}(x, y, z) \land \text{next}(x, x2) \land \text{next}(z, z2) \\
\Rightarrow \text{plus}(x2, y, z2))
\]

Functionality:

\[
\forall x. \forall y. \forall z. \forall w. (\text{plus}(x, y, z) \land \text{plus}(x, y, w) \Rightarrow \text{same}(z, w))
\]