Computational Logic Lecture 5

Relational Logic

Michael Genesereth Autumn 2002

Propositional Logic

Constants refer to atomic propositions.

raining *snowing* *wet*

Compound sentences capture relationships among propositions.

$$raining \vee snowing \Rightarrow wet$$

Relational Logic

Constants refer to objects and relationships.

joe *mary* *loves* *happy*

Simple sentences express relationships among objects.

loves(joe,mary)

Compound sentences capture relationships among relations.

$$loves(x,y) \Rightarrow loves(y,x)$$

$$loves(x,y) \wedge loves(y,x) \Rightarrow happy(x)$$

Plan of Action

Relational Logic Syntax and Informal Semantics
Formal Semantics
Herbrand Method
Relational Proofs

Unification
Relational Resolution
Applications
Strategies

Model Elimination
Epilog
Equality

Words

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*, *arthur*, *betty*, *cathy*, 1, 2, ...

Constants

Object constants refer to objects in the universe of discourse.

Function constants denote functions.

father, *mother*, *age*, *plus*, *times*

Relation constants refer to relations.

person, *happy*, *parent*, *loves*

There is no syntactic distinction between object constants, function constants, and relation constants. The type of each such word is determined from context.

Arity

The arity of a function constant or a relation constant is the number of arguments it takes.

Unary Function constants: $father_1$, $mother$
Binary Function constants: $plus_2$, $times_2$
Ternary Function constants: $price_3$

Unary Relation constants: $person_1$, $happy_1$
Binary Relation constants: $parent_2$, $loves_2$
Ternary Relation constants: $between_3$

The arity of a function constant or a relation constant is optionally notated as a subscript on the constant.

Terms

A *term* is either a variable, an object constant, or a functional term.

Terms refer to items in the universe of discourse.

Terms are analogous to noun phrases in natural language.

Functional Terms

A *functional term* is an expression formed from an *n*-ary function constant and *n* terms enclosed in parentheses and separated by commas.

$$father_1(joe)$$
$$age_1(joe)$$
$$plus_2(x,2)$$

Functional terms are terms and, as such, can be nested.

$$plus_2(age_1(father_1(joe)),age_1(mother_1(joe)))$$

Sentences

There are three types of sentences.

Relational sentences - analogous to the simple sentences in natural language

Logical sentences - analogous to the compound sentences in natural language

Quantified sentences - sentences that express the significance of variables

Relational Sentences

A *relational sentence* is an expression formed from an *n*-ary relation constant and *n* terms enclosed in parentheses and separated by commas.

$$happy_1(art)$$
$$loves_2(art,cathy)$$

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

$$happy_1(person_1(joe))$$

$$happy_1(joe)$$
$$person_1(joe)$$

Logical Sentences

Logical sentences in Relational Logic are analogous to those in Propositional Logic.

$\neg loves(art,cathy)$
$(loves(art,betty) \wedge loves(betty,art))$
$(loves(art,betty) \vee loves(art,cathy))$
$(loves(x,y) \Rightarrow loves(y,x))$
$(loves(x,y) \Leftarrow loves(y,x))$
$(loves(x,y) \Leftrightarrow loves(y,x))$

Parenthesization rules are the same as for Propositional Logic.

Quantified Sentences

Quantified sentences can be nested within other sentences.

$$\forall x.apple(x) \vee \exists x.pear(x))$$
$$\forall x.\forall y.loves(x,y)$$

Syntax Test

Object Constants: *art*, *betty*, *cathy*, 1, 2, …
Function Constants: $father_1$, $mother_1$, age_1, $plus_2$, $times_2$
Relation Constants: $person_1$, $happy_1$, $reflexive_2$,
$parent_2$, $loves_2$, lt_2

$lt(father(art),mother(betty))$

$plus(father(art),betty)$

$happy(person(cathy))$

$loves(x,y) \Rightarrow loves(y,x)$

$reflexive(z) \Rightarrow z(x,x)$

Reminder

Functional terms and relational sentences look similar.
However, they are not the same.

Functional terms *may* be used within other functional terms.
Functional terms *may* be used within relational sentences.

Relational sentences may *not* be used in functional terms.
Relational sentences may *not* be used in relational sentences.

Infix Syntax for Functions

$$
\begin{aligned}
\mathit{plus}(2,3) &\leftrightarrow 2+3 \\
\mathit{minus}(2,3) &\leftrightarrow 2-3 \\
\mathit{times}(2,3) &\leftrightarrow 2\times 3 \\
\mathit{quotient}(2,3) &\leftrightarrow 2\div 3 \\
\mathit{expt}(2,3) &\leftrightarrow 2\uparrow 3
\end{aligned}
$$

$$
\begin{aligned}
\mathit{union}(s,t) &\leftrightarrow s\cup t \\
\mathit{intersection}(s,t) &\leftrightarrow s\cap t
\end{aligned}
$$

Infix Syntax for Relations

$$eq(2,3) \quad \leftrightarrow \quad 2 = 3$$
$$nq(2,3) \quad \leftrightarrow \quad 2 \neq 3$$
$$lt(2,3) \quad \leftrightarrow \quad 2 < 3$$
$$gt(2,3) \quad \leftrightarrow \quad 2 > 3$$
$$leq(2,3) \quad \leftrightarrow \quad 2 \leq 3$$
$$geq(2,3) \quad \leftrightarrow \quad 2 \geq 3$$

$$member(s,t) \quad \leftrightarrow \quad s \in t$$
$$subset(s,t) \quad \leftrightarrow \quad s \subset t$$
$$subseteq(s,t) \quad \leftrightarrow \quad s \subseteq t$$

Operator Precedence

$\uparrow$

$\times \div$

$+ -$

$\cap$

$\cup$

$= \neq < > \leq \geq$

$\in \notin \subset \supset \subseteq \supseteq$

$\neg \forall \exists$

$\wedge$

$\vee$

$\Leftarrow \Leftrightarrow \Rightarrow$

Mushrooms

Unary relation constants: *mushroom*, *purple*, *poisonous*

Purple mushrooms are poisonous.
If a thing is a purple mushroom, then it is poisonous.
If a thing is mushroom and it is purple, then it is poisonous.

$$\forall x.(\mathit{mushroom}(x) \wedge \mathit{purple}(x) \Rightarrow \mathit{poisonous}(x))$$

No purple mushroom is poisonous.
There is no thing that is a mushroom and purple and poisonous.

$$\neg\exists x.(\mathit{mushroom}(x) \wedge \mathit{purple}(x) \wedge \mathit{poisonous}(x))$$

More Mushrooms

Unary relation constants: *mushroom*, *purple*, *poisonous*

A mushroom is poisonous only if it is purple.
If a thing is a mushroom, it is poisonous only if it is purple.
If a thing is a mushroom and it is poisonous, then it is purple.

$$\forall x.(mushroom(x) \wedge poisonous(x) \Rightarrow purple(x))$$

A mushroom is not poisonous unless it is purple.
If a thing is a mushroom, it is not poisonous if it is not purple.
If a thing is a mushroom and it is poisonous, then it is purple.

$$\forall x.(mushroom(x) \wedge poisonous(x) \Rightarrow purple(x))$$

Interpersonal Relations

Object constants: *mike*, *maureen*
Binary relation constant: *loves*

Everybody loves Maureen.
$\forall x.loves(x,maureen)$

Maureen loves everyone who loves her.
$\forall x.(loves(x,maureen) \Rightarrow loves(maureen,x))$

Nobody loves Mike.
$\neg\exists x.loves(x,mike)$

Nobody who loves Maureen loves Mike.
$\forall x.(loves(x,maureen) \Rightarrow \neg loves(x,mike))$

More Interpersonal Relations

Object constants: *mike*, *maureen*
Binary relation constant: *loves*

Everybody loves somebody.
$\forall x.\exists y.loves(x,y)$

There is somebody whom everybody loves.
$\exists y.\forall x.loves(x,y)$

Abelian Groups

Associativity Axiom

$$(x+y)+z = x+(y+z)$$

Commutativity Axiom

$$x+y = y+x$$

Identity Axioms

$$0+y = y$$

$$y+0 = y$$

Inverse Axioms

$$x+inv(x) = 0$$

$$inv(x)+x = 0$$

Open Partial Orders

Non-reflexivity

$$\neg x<x$$

Asymmetry

$$x<y \Rightarrow \neg y<x$$

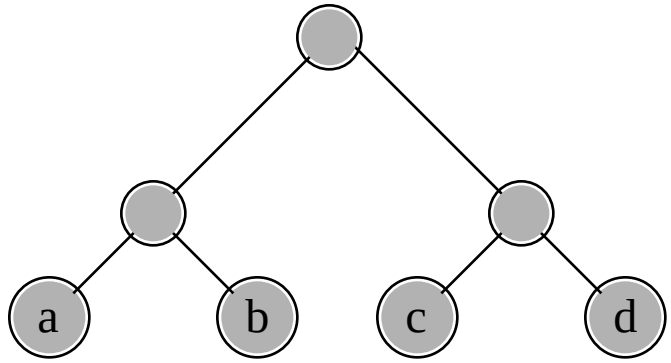
Transitivity

$$x<y \wedge y<z \Rightarrow x<z$$

Binary Trees

Representation as a term:

$$pair(pair(a,b),\ pair(c,d))$$



Membership axioms:

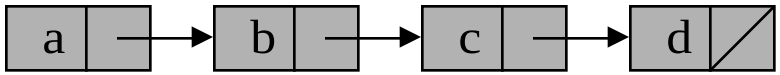
$$in(x,x)$$

$$in(x,pair(y,z)) \Leftarrow in(x,y) \vee in(x,z)$$

Variable Length Lists

Example

[*a*,*b*,*c*,*d*]



Representation as Term

cons(*a*,*cons*(*b*,*cons*(*c*,*cons*(*d*,*nil*))))

Language

Objconst: *nil*

Funconst: *cons*

Relconst: *member*

Membership axioms:

member(*x*,*cons*(*x*,*y*))

member(*x*,*cons*(*y*,*z*)) ⇐ *member*(*x*,*z*)

Special Cases of Relational Logic

Ground Logic

no variables, no functions, no quantifiers

Universal Logic

no functions, no quantifiers

free variables implicitly universally quantified

Existential Logic

no functions

Functional Logic

no quantifiers

Limitations of Ground Logic

Everybody loves somebody.

loves(joe, mary) or loves(joe,sally) or ...
loves(mary, joe) or loves(mary,bill) or ...

The sum of two natural numbers is greater than either number.

1+1>1
1+2>1
1+2>2
…

What about facts about the real numbers?

Limitations of Universal Logic

For every number, there is a number that is larger than it.

Universal Logic

$x<y$ *no!*

$x<a$ *no!*

Existential Logic

$\forall x.\exists y.x<y$

Functional Logic

$x<f(x)$

Existential and Universal Quantifiers

$\exists x.p(x)$ is true.

iff $p(x)$ is *true* for some x.
iff $\neg p(x)$ is *false* for some x.
iff $\neg p(x)$ is not *true* for some x.
iff it not *true* that $\neg p(x)$ is *true* for all x.
iff $\forall x.\neg p(x)$ is not *true*.
iff $\forall x.\neg p(x)$ is *false*.
iff $\neg\forall x.\neg p(x)$ is *true*.

In general, $\exists \upsilon.\varphi$ is equivalent to $\neg\forall \upsilon.\neg\varphi$.

Need for Explicit Quantifiers

Since $\exists \upsilon.\varphi(\upsilon)$ is equivalent to $\neg\forall\upsilon.\neg\varphi(\upsilon)$ and $\varphi(\upsilon)$ in universal logic is equivalent to $\forall\upsilon.\varphi(\upsilon)$, why can we not say existential things in Universal Logic by negating?

Example: How do we say that somebody loves Mike?

$$hates(x,y) \Leftrightarrow \neg loves(x,y)$$
$$\neg hates(x,mike)$$

What is intended: $\neg\forall x.hates(x,mike))$
What we get: $\forall x.\neg hates(x,mike))$

This says that no one hates Mike, i.e. Everyone loves Mike.
Point: In Universal Logic quantifiers are not explicit and so they cannot be negated!

Existential Quantifiers and Functions

Functions Replaced by Existential Quantifiers:

$$loves(x,a(x)) \Leftrightarrow \exists y.loves(x,y)$$

Existential Quantifiers Replaced by Functions:

$$\exists y.loves(x,y) \Leftrightarrow loves(x,a(x))$$

Theorem: An existential sentence is satisfiable iff
the corresponding functional sentence is satisfiable.