Relational Logic

Propositional Logic

Proposition constants refer to atomic propositions.

\[ raining \quad snowing \quad wet \]

Compound sentences capture relationships among propositions.

\[ raining \lor snowing \Rightarrow wet \]
Relational Logic

Object constants refer to objects.

\[ john \quad mary \]

Relation constants refer to relationships.

\[ loves \quad happy \]

Sentences express information about objects and their relationships.

\[ loves(joe,mary) \]
\[ loves(x,y) \Rightarrow loves(y,x) \]
\[ loves(x,y) \land loves(y,x) \Rightarrow happy(x) \]

Plan of Action

Relational Logic Syntax and Informal Semantics
Relational Logic Semantics
Relational Proofs
Properties of Relational Logic
Resolution Preliminaries
Resolution
Applications
Strategies
Words

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

\[a, b, c, arthur, betty, cathy, 1, 2, \ldots\]

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

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

Constants

Object constants refer to objects.

\[joe, stanford, usa, 2345\]

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, function, 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, mother
Binary Function constants: plus, times
Ternary Function constants: price

Unary Relation constants: person, happy
Binary Relation constants: parent, loves
Ternary Relation constants: between

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

Relational Languages

A relational signature is a set/sequence of constants.

Given a relational signature and an arity function, a relational sentence is a compound expression formed from members of the signature in a way that respects the arity function. (Details to follow.)

A relational language is the set of all relational sentences that can be formed from a relational signature and an arity function.
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.

\[
\begin{align*}
\text{father}(\text{joe}) \\
\text{age}(\text{joe}) \\
\text{plus}(x,2)
\end{align*}
\]

Functional terms are terms and so can be nested.

\[
\text{plus}(\text{age}((\text{father}(\text{joe})),\text{age}(\text{mother}(\text{joe}))))
\]
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(art)
loves(art,cathy)

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

No! happy(person(joe)) No!

happy(joe)
person(joe)
Logical Sentences

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

\[\neg loves(art,cathy)\]

\[(loves(art,betty) \land loves(betty,art))\]

\[(loves(art,betty) \lor loves(art,cathy))\]

\[(loves(x,y) \Rightarrow loves(y,x))\]

\[(loves(x,y) \Leftrightarrow loves(y,x))\]

Parenthesization rules are the same as for Propositional Logic.

Quantified Sentences

Universal sentences assert facts about all objects.

\[\forall x.(person(x) \Rightarrow mammal(x))\]

Existential sentence assert the existence of an object with given properties.

\[\exists x.(person(x) \land happy(x))\]

Quantified sentences can be nested within other sentences.

\[\forall x.apple(x) \lor \exists x.pear(x)\]

\[\forall x.\exists y.loves(x,y)\]
Syntax Test

Object Constants: art, betty, cathy
Unary Function Constants: father, mother, age
Binary Function Constant: plus
Unary Relation Constants: person, happy
Binary Relation Constants: reflexive, parent, loves

loves(father(art),mother(art))
plus(father(art),betty)
happy(person(cathy))
∀x.∀y.(loves(x,y) ⇒ loves(y,x))
loves(x,y) ⇒ loves(y,x)
∀x.∀y.loves(art,cathy)
∀z.∀x.reflexive(z) ⇒ 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

\[
\begin{align*}
\text{plus}(2,3) & \iff 2 + 3 \\
\text{minus}(2,3) & \iff 2 - 3 \\
\text{times}(2,3) & \iff 2 \times 3 \\
\text{quotient}(2,3) & \iff 2 \div 3 \\
\text{expt}(2,3) & \iff 2 \uparrow 3 \\
\text{union}(s,t) & \iff s \cup t \\
\text{intersection}(s,t) & \iff s \cap t
\end{align*}
\]

Infix Syntax for Relations

\[
\begin{align*}
\text{eq}(2,3) & \iff 2 = 3 \\
\text{nq}(2,3) & \iff 2 \neq 3 \\
\text{lt}(2,3) & \iff 2 < 3 \\
\text{gt}(2,3) & \iff 2 > 3 \\
\text{leq}(2,3) & \iff 2 \leq 3 \\
\text{geq}(2,3) & \iff 2 \geq 3 \\
\text{member}(s,t) & \iff s \in t \\
\text{subset}(s,t) & \iff s \subset t \\
\text{subsetq}(s,t) & \iff s \subseteq t
\end{align*}
\]
Parenthesis Removal

Dropping Parentheses is good:

\[(p(x) \land q(x)) \rightarrow p(x) \land q(x)\]

But it can lead to ambiguities:

\[((p(x) \lor q(x)) \land r(x)) \rightarrow p(x) \land q(x) \lor r(x)\]

\[(p(x) \lor (q(x) \land r(x))) \rightarrow p(x) \land q(x) \lor r(x)\]

Operator Precedence

\[\uparrow\]
\[\times +\]
\[\pm\]
\[\cap\]
\[\cup\]
\[= < > \leq \geq\]
\[\in \notin \subseteq \supseteq\]
\[\neg \forall \exists\]
\[\wedge\]
\[\lor\]
\[\iff\]
Mushrooms

Unary relation constants: mushroom, purple, poison

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

\[ \forall x. (\text{mushroom}(x) \land \text{purple}(x) \Rightarrow \text{poison}(x)) \]

No purple mushroom is poison.
There is nothing that is a mushroom and purple and poison.

\[ \neg \exists x. (\text{mushroom}(x) \land \text{purple}(x) \land \text{poisonous}(x)) \]

More Mushrooms

Unary relation constants: mushroom, purple, poison

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

\[ \forall x. (\text{mushroom}(x) \land \text{poison}(x) \Rightarrow \text{purple}(x)) \]

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

\[ \forall x. (\text{mushroom}(x) \land \neg \text{purple}(x) \Rightarrow \neg \text{poison}(x)) \]
Interpersonal Relations

Object constants: mike, maureen
Binary relation constant: loves

Everybody loves Maureen.
\( \forall x. \text{loves}(x, \text{maureen}) \)

Maureen loves everyone who loves her.
\( \forall x. (\text{loves}(x, \text{maureen}) \Rightarrow \text{loves}(\text{maureen}, x)) \)

Nobody loves Mike.
\( \neg \exists x. \text{loves}(x, \text{mike}) \)

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

More Interpersonal Relations

Object constants: mike, maureen
Binary relation constant: loves

Everybody loves somebody.
\( \forall x. \exists y. \text{loves}(x, y) \)

There is somebody whom everybody loves.
\( \exists y. \forall x. \text{loves}(x, y) \)
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
\[x < y \land y < z \Rightarrow x < z\]
Binary Trees

Representation as a term:

\[
pair(pair(a, b), pair(c, d))
\]

Membership axioms:

\[
in(x, x) \\
(in(x, pair(y, z))) \iff in(x, y) \lor in(x, z)
\]

Variable Length Lists

Example

\[ [a, b, c, d] \]

Representation as Term

\[ cons(a, cons(b, cons(c, cons(d, nil)))) \]

Language

Objconst: \( a, b, c, d, \text{nil} \)
Funconst: \( cons \)
Relconst: \( member \)

Membership axioms:

\[
member(x, cons(x, y)) \\
member(x, cons(y, z)) \iff member(x, z)
\]
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

*Every number is smaller than some number.*

Universal Logic

\[ x < y \quad \text{no!} \]
\[ x < a \quad \text{no!} \]

Existential Logic

\[ \forall x. \exists y. x < y \]

Functional Logic

\[ x < f(x) \]

Existential and Universal Quantifiers

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

iff, for some \( x \), \( p(x) \) is true.

iff it is not true that, for all \( x \), \( p(x) \) is false.

iff it is not true that, for all \( x \), \( \neg p(x) \) is true.

iff it is not true that \( \forall x. \neg p(x) \) is true.

iff \( \neg \forall x. \neg p(x) \) is true.

In general, \( \exists \nu. \phi \) is equivalent to \( \neg \forall \nu. \neg \phi \).
Need for Explicit Quantifiers

Since $\exists u. \varphi(u)$ is equivalent to $\neg \forall u. \neg \varphi(u)$ and $\varphi(u)$ in universal logic is equivalent to $\forall u. \varphi(u)$, can we express existentials in Universal Logic by negating?

How do we say that somebody loves Mike?

-loves(x,mike)

This says that everyone loves Mike. Intended:

- $\exists x. \text{loves}(x,\text{mike})$
- $\neg \forall x. \neg \text{loves}(x,\text{mike})$
- $\neg \neg \text{loves}(x,\text{mike})$ No!
- $\text{loves}(x,\text{mike})$

Existential Quantifiers and Functions

Functions Replaced by Existential Quantifiers:

-loves(x,a(x)) $\iff$ $\exists y. \text{loves}(x,y)$

Existential Quantifiers Replaced by Functions:

- $\exists y. \text{loves}(x,y) \iff \text{loves}(x,a(x))$

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