

Introduction to Logic

Relational Logic

Michael Genesereth
Computer Science Department
Stanford University

Welcome back. We have reached the end of our treatment of Propositional Logic, and the first midterm is out of the way. Propositional Logic is simple and elegant. It has limitations, but it also has many applications. It is pedagogically valuable in that it provides us with a simple setting in which to look at the main concepts of Logic - syntax and semantics, validity and contingency and unsatisfiability, logical equivalence and entailment and consistency, and proofs. In the weeks ahead, we will be revisiting these concepts in the context of a more expressive logical language, and it will be good that we have already examined these concepts in this simpler setting.

Propositional Logic

Premises:

*If Abby likes Bess, then Bess likes Abby.
Abby likes Bess.*

Conclusion:

Bess likes Abby.

Propositional Logic does a good job of allowing us to talk about propositions and relationships among those propositions, and it gives us the machinery to derive logical conclusions based on these relationships.

Suppose, for example, we believe that, if Jack knows Jill, then Jill knows Jack. Suppose we also believe that Jack knows Jill. From these two facts, we can conclude that Jill knows Jack using a simple application of Implication Elimination.

Symmetry of Affection

Propositional Logic:

*If Abby likes Bess, then Bess likes Abby.
If Abby likes Cody, then Cody likes Abby.
If Abby likes Dana, then Dana likes Abby.*

...

*If Bess likes Abby, then Abby likes Bess.
If Cody likes Abby, then Abby likes Cody.
If Dana likes Abby, then Abby likes Dana.*

Relational Logic:

If X likes Y , then Y likes X .

Unfortunately, when we want to say things more generally, we find that Propositional Logic is inadequate. Suppose, for example, that we wanted to say that, in general, if one person likes a second person, then the second person likes the first. How do we express this general fact? So long as there are only finitely many people, we can do this in Propositional Logic, but it is annoyingly verbose.

Relational Logic

Natural Language Sentence:

If X likes Y , then Y likes X .

For every X and for every Y , if X likes Y , then Y likes X .

New Linguistic Features:

Variables

Quantifiers

Relational Logic Sentence:

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

Relational Logic is an extension of Propositional Logic that solves this problem. The trick is to augment our language with two new linguistic features, viz. variables and quantifiers. With these new features, we can express information about multiple objects without enumerating those objects; and we can express the existence of objects that satisfy specified conditions without saying which objects they are.

In this lesson, we proceed through the same stages as in our treatment of Propositional Logic. We start with syntax and semantics of the language. We present a series of examples of increasing complexity that illustrate various features of the language. In our next lecture we will discuss the properties of Relational Logic sentences, and we define logical equivalence, logical entailment, and logical consistency. Next week, we will look at various proofs systems for Relational Logic.

Syntax

Components of Language

Words

$a \ b \ c \ p \ q \ r \ x \ y \ z$

Sentences

$\forall x. (p(x, a) \wedge q(x, b) \Rightarrow r(a, b))$

We begin our discussion of syntax with a look at the basic words of our language. We then talk about how these words can be combined to form sentences.

Words

Words are strings of letters, digits, and occurrences of the underscore character.

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

a, b, c, 123, cs157, barack_obama

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

u, v, w, x, y, z

Note that, in the online tools, we use lower case and capital letters to distinguish constants and variables.

In Propositional Logic, sentences are constructed from a basic vocabulary of propositional constants. In Herbrand Logic, there are no propositional constants; instead we have object variables and constants.

In our examples here, we write both variables and constants as strings of letters, digits, and a few non-alphanumeric characters (such as "_"). By convention, variables begin with letters from the end of the alphabet. Examples include *u, v, w, x, y, z*. By convention, all constants begin with either letters (other than *u, v, w, x, y, z*) or digits. Examples include *a, b, c, 123, comp225, and barack_obama*.

Constants

Object constants represent objects.

joe stanford canada 2345

Relation constants represent properties or relationships.

isaperson isacountry knows likes between

There are two types of constants in Relational Logic - object constants and relation constants. Object constants represent objects, e.g. *joe, stanford, usa, 2345*. Relation constants represent properties of individual objects, e.g. *isaperson* and *isacountry*, an relationships among objects, e.g. *knows* and *likes*.

Note that there is no distinction in spelling between object constants and relation constants. The type of each such word is determined by its usage or, in some cases, in an explicit specification.

Arity

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

Unary relation constant - 1 argument
e.g. *isaperson, isacountry*

Binary relation constant - 2 arguments
e.g. *knows, likes*

Ternary relation constant - 3 arguments
e.g. *between*

n-ary relation constant - *n* arguments

As we shall see, relation constants are used in forming complex expressions by combining them with an appropriate number of arguments. Accordingly, each relation constant has an associated arity, i.e. the number of arguments with which that relation constant can be combined. A relation constant that can be combined with a single argument is said to be unary; one that can be combined with two arguments is said to be binary; one that can be combined with three arguments is said to be ternary; more generally, a relation constant that can be combined with *n* arguments is said to be *n*-ary.

Vocabularies

A *vocabulary / signature* consists of a finite, non-empty set of object constants and a finite, non-empty 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 vocabulary / signature consists of a finite, non-empty set of object constants, a finite, non-empty set of relation constants, and an assignment of arities for each of the relation constants. For example, we might have a signature consisting of object constants a and b , the binary function constant g , the unary relation constant p , and the binary relation constant q .

Terms

A *term* is either (1) a variable or (2) an object constant.

Terms represent objects.

Terms are analogous to pronouns and nouns in English.

A term is defined to be a variable or an object constant. Terms typically represent objects presumed or hypothesized to exist in the world; and, as such, they are analogous to pronouns and nouns in natural language.

Sentences

Three types of sentences in Relational Logic:

Relational sentences - analogous to the proposition constants in Propositional Logic

Logical sentences - analogous to logical sentences in Propositional Logic

Quantified sentences - sentences that express the significance of variables

There are three types of sentences in Relational Logic. There are relational sentences, which are analogous to propositions in Propositional Logic. There are logical sentences, which are analogous to the logical sentences in Propositional Logic. And there are quantified sentences, which 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.

$$q(a,y)$$

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

No! $q(a,q(a,y))$ No!

Relational sentences are also called atoms or atomic sentences.

A relational sentence is an expression formed from an n -ary relation constant and n terms. For example, if q is a relation constant with arity 2 and if a and y are terms, then the expression shown below is a syntactically legal relational sentence.

Note that, unlike functional terms, relational sentences may not be nested within other relational sentences.

Logical Sentences

Logical sentences in Relational Logic are analogous to those in Propositional Logic (except with relational sentences in place of propositional constants)

$$\begin{aligned} &(\neg q(a,b)) \\ &(p(a) \wedge p(b)) \\ &(p(a) \vee p(b)) \\ &(q(x,y) \Rightarrow q(y,x)) \\ &(q(x,y) \Leftrightarrow q(y,x)) \end{aligned}$$

Logical sentences in Relational Logic are defined the same as in Propositional Logic. There are negations, conjunctions, disjunctions, implications, and equivalences. The syntax is exactly the same, except that the elementary components are relational sentences and equations rather than proposition constants.

Quantified Sentences

Universal sentences assert facts about all objects.

$$(\forall x.(p(x) \Rightarrow q(x,x)))$$

Existential sentences assert the existence of objects with given properties.

$$(\exists x.(p(x) \wedge q(x,x)))$$

The sentence contained *within* a quantified sentence is called the *scope* of that sentence.

There are two types of quantified sentences - universally quantified sentences and existentially quantified sentences.

A universally quantified sentence is used to assert that all objects have a certain property. For example, the top expression here is a universally quantified sentence asserting that p holds of an object, then q holds of that object and itself. The universal quantifier is notated with this upside down A.

An existentially quantified sentence is used to assert that there are objects having a certain property. For example, the middle expression is an existentially quantified sentence asserting that there is an object that satisfies p and, when paired with itself satisfies q as well. The existential quantifier is notated by an E rotated through 180 degrees.

Nesting

Quantified sentences can be nested within other sentences.

$$(\forall x.p(x)) \vee (\exists x.q(x,x)) \\ (\forall x.(\exists y.(q(x,y) \wedge q(y,x))))$$

The sentence contained *inside* a quantified sentence is called the *scope* of that sentence.

Note that quantified sentences can be nested within other sentences. For example, in the first sentence at the bottom, we have quantified sentences inside of a disjunction. In the second sentence at the bottom, we have a quantified sentence nested inside of another quantified sentence.

Parentheses

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.

$$\forall x.p(x) \Rightarrow q(x,x) \rightarrow (\forall x.p(x)) \Rightarrow q(x,x) \\ \exists x.p(x) \wedge q(x,x) \rightarrow (\exists x.p(x)) \wedge q(x,x)$$

As with Propositional Logic, we can drop unneeded parentheses in Relational Logic, relying on precedence to disambiguate the structure of unparenthesized sentences. In Relational Logic, the precedence relations of the logical operators are the same as in Propositional Logic, and quantifiers have higher precedence than logical operators.

The examples here show how to parenthesize sentences with both quantifiers and logical operators. The sentences on the right are partially parenthesized versions of the sentences on the left. (To be fully parenthesized, we would need to add parentheses around each of the sentences as a whole.) Notice that in each case the quantifier applies only to the sentences to its right, even though the other sentence also contains occurrences of the quantified variable.

Ground and Non-Ground Expressions

An expression is *ground* if and only if it contains no variables.

Ground sentence:

$$p(a)$$

Non-Ground Sentences:

$$q(a,x) \\ \forall x.p(x)$$

Before we leave our look at syntax, we need to define some useful concepts. An expression in the language of Relational Logic is ground if and only if it contains no variables. For example, the sentence $p(a)$ is ground, whereas the sentence $\forall x.p(x)$ is not.

Bound and Free Variables

An *occurrence of a variable* is **bound** if and only if it is in the scope of a quantifier of that variable. Else, **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: $\forall x.\exists y.q(x,y)$

An occurrence of a variable is bound if and only if it appears in the scope of a quantifier of that variable. Otherwise, it is free. For example, x is free and y is bound in the sentence shown here.

A sentence is open if and only if it has free variables. Otherwise, it is closed. For example, the first sentence here is open and the second sentence is closed.

Exercise

Object Constants: *jim, molly*

Unary Relation Constant: *person*

Binary Relation Constant: *parent*

parent(jim, molly)

parent(molly, molly)

$\neg person(jim)$

person(jim, molly)

parent(molly, z)

$\exists x.parent(molly, x)$

$\forall y.parent(molly, jim)$

$\exists z.(z(jim, molly) \vee z(molly, jim))$

Semantics

Herbrand Base

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.

The Herbrand base for a Relational language is the set of all ground atomic sentences that can be formed from the vocabulary of the language. Let's look at some examples.

Example

Object Constants: a, b
Unary Relation Constant: p
Binary Relation Constant: q

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

Questions:
How large is the Herbrand base for a vocabulary with n object constants and 2 unary relation constants?

How large is the Herbrand base for a vocabulary with n object constants and 1 binary relation constant?

For a schema with object constants a and b , no function constants, and relation constants p and q where p has arity 1 and q has arity 2, the Herbrand base is shown here. We have one p sentence for each object constant, and we have one q sentence for each of the four possible pairs of object constants.

It is worthwhile to note that, for a given relation constant and a finite set of terms, there is an upper bound on the number of possible ground relational sentences that can be formed using that relation constant. In particular, for a domain of size b , there are b^n distinct n -tuples of object constants; and hence there are b^n ground relational sentences for each n -ary relation constant. Since the number of relation constants in a signature is finite, this means that the Herbrand base is also finite.

Truth Assignment

A *truth assignment / interpretation* 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*.

$$\begin{aligned} p(a)^i &= 1 \\ p(b)^i &= 0 \\ q(a,a)^i &= 1 \\ q(a,b)^i &= 0 \\ q(b,a)^i &= 1 \\ q(b,b)^i &= 0 \end{aligned}$$

How many truth assignments are there for a language with n object constants and 1 binary relation constant?

A truth assignment for a Relational Logic language is a function that maps each ground relational sentence in the Herbrand base to a truth value. As in Propositional Logic, we use the digit 1 as a synonym for true and 0 as a synonym for false; and we refer to the value assigned to a ground atom by writing the atom with the name of the truth assignment as a superscript. The truth assignment i defined here is an example for the case of the language mentioned earlier.

Sentential Truth Assignment

A *sentential truth assignment* is an association between arbitrary sentences in a Relational language and the truth values 1 and 0.

Truth Assignment	Sentential Truth Assignment
$p(a)^i = 1$	$(p(a) \vee p(b))^i = 1$
$p(b)^i = 0$	$(p(a) \wedge \neg p(b))^i = 1$

Each truth assignment gives rise to a unique sentential truth assignment based on the type of sentence.

As with Propositional Logic, once we have a truth assignment for the ground relational sentences of a language, the semantics of our language prescribes a unique extension of that assignment to the other sentences of the language.

Logical Sentences

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

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

$(\varphi \vee \psi)^i = 1$ 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 $\varphi^i = \psi^i$

The rules for logical sentences in Relational Logic are the same as those for logical sentences in Propositional Logic. A truth assignment satisfies a negation if and only if does not satisfy target of the negation. The truth value of a conjunction is true under a truth assignment if and only if the truth value of its conjuncts are both true; otherwise, the truth value is false. The truth value of a disjunction is true if and only if the truth value of at least one its disjuncts is true; otherwise, the truth value is false. A truth assignment satisfies an implication if and only if it does not satisfy the antecedent or it does satisfy consequent. A truth assignment satisfies an equivalence if and only if the truth values of the component sentences are the same.

Instances

An *instance* of an expression is an expression in which all **free** variables have been consistently replaced by ground terms.

Example:

$p(x) \Rightarrow q(x,x)$

$p(a) \Rightarrow q(a,a)$

$p(b) \Rightarrow q(b,b)$

Example:

$p(x) \Rightarrow \exists y.q(x,y)$

$p(a) \Rightarrow \exists y.q(a,y)$

$p(b) \Rightarrow \exists y.q(b,y)$

Consistent replacement here means that, if one occurrence of a variable is replaced by a ground term, then all occurrences are replaced by the same ground term.

In order to define truth assignments for quantified sentences, we need the notion of instances. An instance of an expression is an expression in which all 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.

Quantified Sentences

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.

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.

Example

Truth Assignment:

$$\begin{array}{ll} p(a)^i = 1 & q(a,a)^i = 1 \\ p(b)^i = 0 & q(a,b)^i = 0 \\ & q(b,a)^i = 1 \\ & q(b,b)^i = 0 \end{array}$$

Sentence:

$$\forall x.(p(x) \Rightarrow q(x,x))$$

Instances:

$$\begin{array}{l} p(a) \Rightarrow q(a,a) \\ p(b) \Rightarrow q(b,b) \end{array}$$

As an example of these definitions, consider the sentence $(\forall x.p(x) \Rightarrow q(x,x))$. What is the truth value under the truth assignment shown here? According to our definition, a universally quantified sentence is true if and only every instance of its scope is true. For this language, there are just two instances, a shown here. We know that $p(a)$ is true and $q(a,a)$ is true, so the first instance is true. [Check mark.] $q(b,b)$ is false, but so is $p(b)$ so the second instance is true as well. [Check mark.] Since both instances are true, the quantified sentence is true. [Check mark.]

Example

Truth Assignment:

$$\begin{array}{ll} p(a)^i = 1 & q(a,a)^i = 1 \\ p(b)^i = 0 & q(a,b)^i = 0 \\ & q(b,a)^i = 1 \\ & q(b,b)^i = 0 \end{array}$$

Sentence:

$$\forall x.(p(x) \Rightarrow q(x,x))$$

Instances:

$$\begin{array}{l} p(a) \Rightarrow q(a,a) \checkmark \\ p(b) \Rightarrow q(b,b) \end{array}$$

As an example of these definitions, consider the sentence $(\forall x.p(x) \Rightarrow q(x,x))$. What is the truth value under the truth assignment shown here? According to our definition, a universally quantified sentence is true if and only every instance of its scope is true. For this language, there are just two instances, a shown here. We know that $p(a)$ is true and $q(a,a)$ is true, so the first instance is true. [Check mark.] $q(b,b)$ is false, but so is $p(b)$ so the second instance is true as well. [Check mark.] Since both instances are true, the quantified sentence is true. [Check mark.]

Example	
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:	
	$p(a) \Rightarrow q(a,a) \checkmark$
	$p(b) \Rightarrow q(b,b) \checkmark$

As an example of these definitions, consider the sentence $(\forall x.p(x) \Rightarrow q(x,x))$. What is the truth value under the truth assignment shown here? According to our definition, a universally quantified sentence is true if and only every instance of its scope is true. For this language, there are just two instances, a shown here. We know that $p(a)$ is true and $q(a,a)$ is true, so the first instance is true. [Check mark.] $q(b,b)$ is false, but so is $p(b)$ so the second instance is true as well. [Check mark.] Since both instances are true, the quantified sentence is true. [Check mark.]

Example	
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:	$\forall x.(p(x) \Rightarrow q(x,x)) \checkmark$
Instances:	
	$p(a) \Rightarrow q(a,a) \checkmark$
	$p(b) \Rightarrow q(b,b) \checkmark$

As an example of these definitions, consider the sentence $(\forall x.p(x) \Rightarrow q(x,x))$. What is the truth value under the truth assignment shown here? According to our definition, a universally quantified sentence is true if and only every instance of its scope is true. For this language, there are just two instances, a shown here. We know that $p(a)$ is true and $q(a,a)$ is true, so the first instance is true. [Check mark.] $q(b,b)$ is false, but so is $p(b)$ so the second instance is true as well. [Check mark.] Since both instances are true, the quantified sentence is true. [Check mark.]

Example	
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:	$\forall x.\exists y.q(x,y) \checkmark$
Instances:	
$\exists y.q(a,y) \checkmark$	$\exists y.q(b,y) \checkmark$
$q(a,a) \checkmark$	$q(b,a) \checkmark$
$q(a,b) \times$	$q(b,b) \times$

Now let's consider a case with nested quantifiers. Is $\forall x.\exists y.q(x,y)$ true or false for the truth assignment shown here? As before, we know that this sentence is true if every instance of its scope is true. The two possible instances are shown here.

To determine the truth of the first of these existential sentences, we must find at least one instance of its scope that is true. The possibilities are shown here on the left. Of these, the first is true; and so the first existential sentence is true.

Now, we do the same for the second existentially quantified. The possible instances are shown here on the right. Of these, again the first is true; and so the second existential sentence is true.

Since both existential sentences are true, the original quantified sentence must be true as

well.

Open Sentences

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.)

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

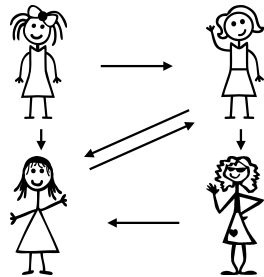
A truth assignment satisfies *a set of sentences* if and only if it satisfies every sentence in the set.

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.)

A truth assignment satisfies a set of sentences if and only if it satisfies every sentence in the set.

Example - Friends

Friends



Consider the small group of friends we met in the first lecture. There are just four people - Abby, Bess, Cody, and Dana.

One Possible State

	Abby	Bess	Cody	Dana
Abby			✓	
Bess			✓	
Cody	✓	✓		✓
Dana			✓	

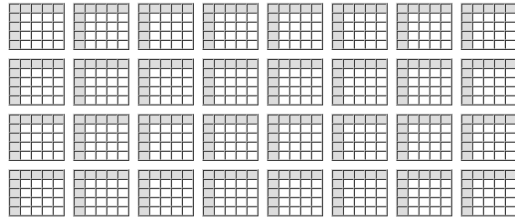
Some of the girls like each other, but some do not. The checkmark in the first row here means that Abby likes Cody, while the absence of a checkmark means that Abby does not like the other girls (including herself). Bess likes Cody too. Cody likes everyone but herself. And Dana also likes the popular Cody.

Another Possible State

	Abby	Bess	Cody	Dana
Abby	✓		✓	
Bess		✓		✓
Cody	✓		✓	
Dana		✓		✓

Of course, that is not the only possible state of affairs. This table shows another possible world. In this world, every girl likes exactly two other girls, and every girl is liked by just two girls.

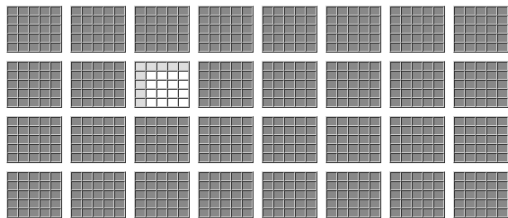
Possible States



2^{16} (65,536) possible worlds.

As it turns out, there are quite a few possibilities. Given four girls, there are sixteen possible instances of the likes relation - Abby likes Abby, Abby likes Bess, Abby likes Cody, Abby likes Dana, Bess likes Abby, and so forth. Each of these sixteen can be either true or false; there are 2^{16} (65,536) combinations of these true-false possibilities; and so there are 2^{16} possible worlds.

Actual State



Ideally, we would like to know exactly how things stand, i.e. we want to know which of these possible worlds is correct.

Signature

Object Constants: *abby, bess, cody, dana*

Binary Relation Constant: *likes*

Herbrand base has 16 ground relational sentences.

In order to encode information in Relational Logic, we adopt a signature with four object constants (*abby, bess, cody, and dana*) and one binary relation constant (*likes*). The Herbrand base in this case contains 16 ground relational sentences, one for each pair of object constants.

Herbrand Base

likes(abby,abby) *likes(bess,abby)*
likes(abby,bess) *likes(bess,bess)*
likes(abby,cody) *likes(bess,cody)*
likes(abby,dana) *likes(bess,dana)*

likes(cody,abby) *likes(dana,abby)*
likes(cody,bess) *likes(dana,bess)*
likes(cody,cody) *likes(dana,cody)*
likes(cody,dana) *likes(dana,dana)*

Given complete information about the likes and dislikes of the girls, we could completely characterize the state of affairs as a set of ground relational sentences or negations of ground relational sentences, like the ones shown below, with one sentence for each member of the Herbrand base.

To make things more interesting, let's assume that we do not have complete information, only fragments of information about the girls' likes and dislikes. Let's see how we can encode such fragments in the language of Relational Logic.

State of Friends World

	Abby	Bess	Cody	Dana
Abby			✓	
Bess			✓	
Cody	✓	✓		✓
Dana			✓	

Let's look at one of the possible worlds we saw earlier.

Ground Data

¬likes(abby,abby) *¬likes(bess,abby)*
¬likes(abby,bess) *¬likes(bess,bess)*
likes(abby,cody) *likes(bess,cody)*
¬likes(abby,dana) *¬likes(bess,dana)*

likes(cody,abby) *¬likes(dana,abby)*
likes(cody,bess) *¬likes(dana,bess)*
¬likes(cody,cody) *likes(dana,cody)*
likes(cody,dana) *¬likes(dana,dana)*

Given complete information about the likes and dislikes of the girls, we could completely characterize the state of affairs with a truth assignment like the one shown here.

To make things more interesting, let's assume that we do not have complete information, only fragments of information about the girls' likes and dislikes. Let's see how we can encode such fragments in the language of Relational Logic.

Sentences

Abby likes everyone Bess likes.

Let's start with a simple case. Abby likes everyone Bess likes. Paraphrasing this sentence helps in the translation. If Bess likes a girl, then Abby also likes her. Since this is a fact about everyone, we use a universal quantifier. For all y, if bess likes y, then abby likes y.

Next case. Cody likes everyone who likes her. In other words, if some girl likes Cody, then Cody likes that girl. Again, we use a universal quantifier. For all x, if x likes cody, then cody likes x.

Sentences

Abby likes everyone Bess likes.
If Bess likes a person, then Abby also likes her.

$\forall y.(\text{likes}(\text{bess}, y) \Rightarrow \text{likes}(\text{abby}, y))$

Let's start with a simple case. Abby likes everyone Bess likes. Paraphrasing this sentence helps in the translation. If Bess likes a girl, then Abby also likes her. Since this is a fact about everyone, we use a universal quantifier. For all y, if bess likes y, then abby likes y.

Next case. Cody likes everyone who likes her. In other words, if some girl likes Cody, then Cody likes that girl. Again, we use a universal quantifier. For all x, if x likes cody, then cody likes x.

Sentences

Abby likes everyone Bess likes.
If Bess likes someone, then Abby also likes her.

$\forall y.(\text{likes}(\text{bess}, y) \Rightarrow \text{likes}(\text{abby}, y))$

Cody likes everyone who likes her.

Let's start with a simple case. Abby likes everyone Bess likes. Paraphrasing this sentence helps in the translation. If Bess likes a girl, then Abby also likes her. Since this is a fact about everyone, we use a universal quantifier. For all y, if bess likes y, then abby likes y.

Next case. Cody likes everyone who likes her. In other words, if some girl likes Cody, then Cody likes that girl. Again, we use a universal quantifier. For all x, if x likes cody, then cody likes x.

Sentences

*Abby likes everyone Bess likes.
If Bess likes someone, then Abby also likes her.*

$$\forall y.(\text{likes}(\text{bess}, y) \Rightarrow \text{likes}(\text{abby}, y))$$

*Cody likes everyone who likes her.
If a person likes Cody, then Cody likes that person.*

$$\forall x.(\text{likes}(x, \text{cody}) \Rightarrow \text{likes}(\text{cody}, x))$$

Let's start with a simple case. Abby likes everyone Bess likes. Paraphrasing this sentence helps in the translation. If Bess likes a girl, then Abby also likes her. Since this is a fact about everyone, we use a universal quantifier. For all y, if bess likes y, then abby likes y.

Next case. Cody likes everyone who likes her. In other words, if some girl likes Cody, then Cody likes that girl. Again, we use a universal quantifier. For all x, if x likes cody, then cody likes x.

Sentences

Cody likes somebody who likes her.

$$\exists x.(\text{likes}(x, \text{cody}) \Rightarrow \text{likes}(\text{cody}, x)) \quad \text{Wrong!}$$

$\text{likes}(\text{abby}, \text{cody}) \Rightarrow \text{likes}(\text{cody}, \text{abby})$
 $\text{likes}(\text{bess}, \text{cody}) \Rightarrow \text{likes}(\text{cody}, \text{bess})$
 $\text{likes}(\text{cody}, \text{cody}) \Rightarrow \text{likes}(\text{cody}, \text{cody})$
 $\text{likes}(\text{dana}, \text{cody}) \Rightarrow \text{likes}(\text{cody}, \text{dana})$

Suppose no one likes Cody. All of these sentences are true!

Cody likes somebody who likes her. The word somebody here is a tip-off that we need to use an existential quantifier. There is a y such that cody likes y and y likes cody.

Nobody likes herself. The use of the word nobody here suggests a negation. A good technique in such cases is to rewrite the English sentence as the negation of a positive version of the sentence before translating to Relational Logic. There does not exist an x such that x likes x.

Sentences

*Cody likes somebody who likes her.
There is someone who likes cody and is liked by Cody.*

$$\exists y.(\text{likes}(\text{cody}, y) \wedge \text{likes}(y, \text{cody}))$$

Cody likes somebody who likes her. The word somebody here is a tip-off that we need to use an existential quantifier. There is a y such that cody likes y and y likes cody.

Nobody likes herself. The use of the word nobody here suggests a negation. A good technique in such cases is to rewrite the English sentence as the negation of a positive version of the sentence before translating to Relational Logic. There does not exist an x such that x likes x.

Sentences

*Cody likes somebody who likes her.
There is someone who likes cody and is liked by Cody.*

$$\exists y.(likes(cody,y) \wedge likes(y,cody))$$

Nobody likes herself.

Cody likes somebody who likes her. The word somebody here is a tip-off that we need to use an existential quantifier. There is a y such that cody likes y and y likes cody.

Nobody likes herself. The use of the word nobody here suggests a negation. A good technique in such cases is to rewrite the English sentence as the negation of a positive version of the sentence before translating to Relational Logic. There does not exist an x such that x likes x.

Sentences

*Cody likes somebody who likes her.
There is someone who likes cody and is liked by Cody.*

$$\exists y.(likes(cody,y) \wedge likes(y,cody))$$

Nobody likes herself.

It is not the case that there is someone who likes herself.

$$\neg \exists x.likes(x,x)$$
$$\forall x.\neg likes(x,x)$$

Cody likes somebody who likes her. The word somebody here is a tip-off that we need to use an existential quantifier. There is a y such that cody likes y and y likes cody.

Nobody likes herself. The use of the word nobody here suggests a negation. A good technique in such cases is to rewrite the English sentence as the negation of a positive version of the sentence before translating to Relational Logic. There does not exist an x such that x likes x.

Sentences

Everybody likes somebody.

$$\forall x.\exists y.likes(x,y)$$

There is somebody whom everybody likes.

$$\exists y.\forall x.likes(x,y)$$

Everybody likes somebody. Here we have a case requiring two quantifiers, one universal and one existential. The key to this case is getting the quantifiers in the right order. For all x, there is a y such that x likes y.

There is someone everyone likes. The preceding sentence tells us that everyone likes someone, but that someone can be different for different people. This sentence tells us that everybody likes the same person.

To understand the difference between these two sentences, let's diagram the situation. We have 4 girls - abby, bess, cody, and dana. The first sentence tells us that everyone likes somebody. For example, it could be the case that abby likes bess and bess likes abby. Cody likes dana and dana likes cody. [Draw it out.] [Pause.] Okay, that's a little too idealistic. Let's be a little more realistic. [Erase and draw again.] We have abby, bess,

cody, and dana. [Draw.] Abby likes bess; bess likes cody; cody likes dana; and dana...
Now, consider the second sentence. Again we have abby, bess, cody, and dana. [Draw.]
All like the same person. [Draw.] And of course we have the narcissistic Dana.

Example

Abby ● ● Bess

Cody ● ● Dana

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

To understand the difference between these two sentences, lets; diagram the situation. We have 4 girls - abby, bess, cody, and dana.

Everybody Likes Somebody

Abby ● ● Bess

Cody ● ● Dana

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

The first sentence tells us that everyone likes somebody. For example, it could be the case that abby likes bess and bess likes abby. Cody likes dana and dana likes cody.

Everybody Likes Somebody

Abby → Bess

Cody Dana

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

Okay, that's a little too idealistic. Let's be a little more realistic. We have abby, bess, cody, and dana. Abby likes bess.

Everybody Likes Somebody

Abby → Bess

Cody → Bess

Dana

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

Bess likes cody.

Everybody Likes Somebody

Abby → Bess

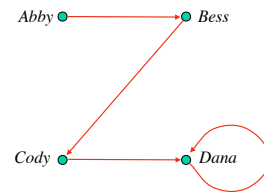
Cody → Bess

Dana

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

Cody likes Dana.

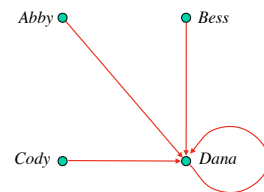
Everybody Likes Somebody



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

And then there is narcissistic Dana...

There is Somebody Whom Everyone Likes

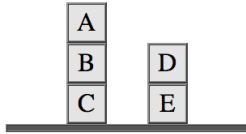


$$\exists y. \forall x. \text{likes}(x, y)$$

Now, consider the second sentence. Again we have abby, bess, cody, and dana. All like the same person, say Dana. Abby likes Dana. Bess likes Dana. Cody likes Dana. And, of course, we have the narcissistic Dana.

Example - Blocks World

Blocks World



The Blocks World is a popular application area for illustrating ideas in the field of Artificial Intelligence. A typical Blocks World scene is shown here. Most people looking at this figure interpret it as a configuration of five toy blocks. Some people conceptualize the table on which the blocks are resting as an object as well; but, for simplicity, we ignore it here.

Signature

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.

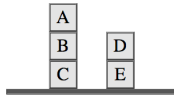
Ternary Relation Constant:

stack - triples of blocks arranged in a stack.

In order to describe this scene, we adopt a signature with five object constants - *a, b, c, d, e* - with one object constant for each of the five blocks in the scene. The intent here is for each of these object constants to represent the block marked with the corresponding capital letter in the scene.

In a spatial conceptualization of the Blocks World, there are numerous meaningful relations. For example, it makes sense to think about the relation that holds between two blocks if and only if one is resting on the other. In what follows, we use the relation constant *on* to refer to this relation. We might also think about the relation that holds between two blocks if and only if one is anywhere above the other, i.e. the first is resting on the second or is resting on a block that is resting on the second, and so forth. In what follows, we use the relation constant *above* to talk about this relation. There is the relation that holds of three blocks that are stacked one on top of the other. We use the relation constant *stack* as a name for this relation. We use the unary relation constant *clear* to denote the relation that holds of a block if and only if there is no block on top of it. We use the unary relation constant *table* to denote the relation that holds of a block if and only if that block is resting on the table.

Ground Data



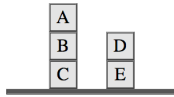
$clear(a)$ $\neg table(a)$
 $\neg clear(b)$ $\neg table(b)$
 $\neg clear(c)$ **$table(c)$**
 $clear(d)$ $\neg table(d)$
 $\neg clear(e)$ **$table(e)$**

Here is a dataset recording some of the information about this scene. Question: How do we write definitions for various definable relations?

Now go to editor and do everything there.

Make safety mistake and see that or maybe show after talk about issues.

Ground Data



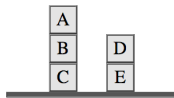
$\neg on(a,a)$ $\neg on(b,a)$ $\neg on(c,a)$ $\neg on(d,a)$ $\neg on(e,a)$
 $on(a,b)$ $\neg on(b,b)$ $\neg on(c,b)$ $\neg on(d,b)$ $\neg on(e,b)$
 $\neg on(a,c)$ **$on(b,c)$** $\neg on(c,c)$ $\neg on(d,c)$ $\neg on(e,c)$
 $\neg on(a,d)$ $\neg on(b,d)$ $\neg on(c,d)$ $\neg on(d,d)$ $\neg on(e,d)$
 $\neg on(a,e)$ $\neg on(b,e)$ $\neg on(c,e)$ **$on(d,e)$** $\neg on(e,e)$

Here is a dataset recording some of the information about this scene. Question: How do we write definitions for various definable relations?

Now go to editor and do everything there.

Make safety mistake and see that or maybe show after talk about issues.

Ground Data



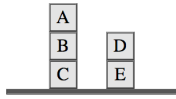
$\neg above(a,a)$ $\neg above(b,a)$ $\neg above(c,a)$ $\neg above(d,a)$ $\neg above(e,a)$
 $above(a,b)$ $\neg above(b,b)$ $\neg above(c,b)$ $\neg above(d,b)$ $\neg above(e,b)$
 $above(a,c)$ **$above(b,c)$** $\neg above(c,c)$ $\neg above(d,c)$ $\neg above(e,c)$
 $\neg above(a,d)$ $\neg above(b,d)$ $\neg above(c,d)$ $\neg above(d,d)$ $\neg above(e,d)$
 $\neg above(a,e)$ $\neg above(b,e)$ $\neg above(c,e)$ **$above(d,e)$** $\neg above(e,e)$

Here is a dataset recording some of the information about this scene. Question: How do we write definitions for various definable relations?

Now go to editor and do everything there.

Make safety mistake and see that or maybe show after talk about issues.

Constraints



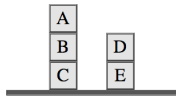
Constraints on the *on* relation:

$$\neg \exists x. on(x, x) \\ \forall x. \forall y. (on(x, y) \Rightarrow \neg on(y, x))$$

The above relation is a bit tricky to define correctly. One block is above another block if and only if the first block is on the second block or it is on another block that is above the second block. Also, no block can be above itself. Given a complete definition for the *on* relation, these two axioms determine a unique above relation.

The main lesson to be taken from this example is the way that logical sentences can be used to define new relations in terms of existing relations.

Definitions

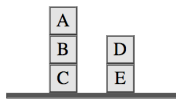


Definition of *clear*:

$$\forall y. (clear(y) \Leftrightarrow \neg \exists x. on(x, y))$$

A block satisfies the *clear* relation if and only if there is nothing on it.

Definitions

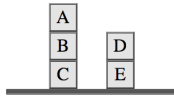


Definition of *table*:

$$\forall x. (table(x) \Leftrightarrow \neg \exists y. on(x, y))$$

A block satisfies the *table* relation if and only if it is not on some block.

Definitions

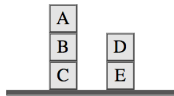


Definition of *stack*:

$$\forall x. \forall y. \forall z. (stack(x, y, z) \Leftrightarrow on(x, y) \wedge on(y, z))$$

Three blocks satisfy the stack relation if and only if the first is on the second and the second is on the third.

Constraints



Constraint on *above*:

$$\forall x. \forall z. (above(x, z) \Leftrightarrow on(x, z) \vee \exists y. (above(x, y) \wedge above(y, z)))$$

Almost but not quite a definition!

The above relation is a bit tricky to define correctly. One block is above another block if and only if the first block is on the second block or it is on another block that is above the second block. Also, no block can be above itself. Given a complete definition for the on relation, these two axioms determine a unique above relation.

Exercise 8.3



Introduction to Logic

Tools
for
Thought

Exercise 8.3 - Consistency

Consider a version of the Blocks World with just three blocks - *a*, *b*, and *c*. The *on* relation is axiomatized below.

$$\begin{array}{lll} \neg on(a, a) & on(a, b) & \neg on(a, c) \\ \neg on(b, a) & \neg on(b, b) & on(b, c) \\ \neg on(c, a) & \neg on(c, b) & \neg on(c, c) \end{array}$$

Let's suppose that the *above* relation is defined as follows. This is *almost* the same as in Section 7.7 except that we have replaced an occurrence of *on* with *above*.

$$\forall x. \forall z. (above(x, z) \Leftrightarrow on(x, z) \vee \exists y. (above(x, y) \wedge above(y, z)))$$

A sentence ϕ is consistent with a set Δ of sentences if and only if there is a truth assignment that satisfies all of the sentences in $\Delta \cup \{\phi\}$. Say whether each of the following sentences is consistent with the sentences about *on* and *above* shown above. Be careful. It's tricky.

- a. *above(a, c)* ☐
- b. *above(a, a)* ☐
- c. *above(c, a)* ☐

Show Answers | Reset Answers

The main lesson to be taken from these examples is the way that logical sentences can be used to define new relations in terms of existing relations. But we need to be careful.

Example - Minefinder

Course Website



Introduction to Logic

Tools for Thought

Help Lessons Chapters Dictionary Tools

Lesson 7 - Relational Logic

- ☐ Lesson 7.1 - Introduction
- ☐ Lesson 7.2 - Syntax
- ☐ Lesson 7.3 - Semantics
- ☐ Lesson 7.4 - Evaluation
- ☐ Lesson 7.5 - Satisfaction
- ☐ Lesson 7.6 - Surety World
- ☐ Lesson 7.7 - Blocks World
- ☐ Lesson 7.8 - Modular Arithmetic
- ☐ Exercise 7.1
- ☐ Exercise 7.2
- ☐ Exercise 7.3
- ☐ Exercise 7.4
- ☐ Extra - Surety Life
- ☐ Extra - Minefinder
- ☐ Extra - Minefield
- ☐ Extra - Logicians
- ☐ Puzzle - Cards

Use the arrow keys to navigate.

Previous	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Next
----------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	------

Press the mouse key to toggle all icons.



Introduction to Logic

Tools for Thought

Minefinder

$\neg \exists x \text{ mine}(1, x)$

	1	2	3	4	5	6	7	8
1								
2								
3								
4								
5								
6								
7								
8								

Show Answer Show Instructions Reset Game

