

Introduction to Logic

Relational Analysis

Michael Genesereth
Computer Science Department
Stanford University

Truth Assignment

A *truth assignment / interpretation* is an association between ground relational sentences and the truth values *true* and *false* or, equivalently 1 and 0.

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

As with Propositional Logic, we start with truth assignments for the ground relational sentences of a language. Given truth assignments 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

$$(\neg \varphi)^i = 1 \quad \text{if and only if} \quad \varphi^i = 0$$

$$(\varphi \wedge \psi)^i = 1 \quad \text{if and only if} \quad \varphi^i = 1 \text{ and } \psi^i = 1$$

$$(\varphi \vee \psi)^i = 1 \quad \text{if and only if} \quad \varphi^i = 1 \text{ or } \psi^i = 1$$

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

$$(\varphi \Leftrightarrow \psi)^i = 1 \quad \text{if and only if} \quad \varphi^i = \psi^i$$

The semantics of logical sentences is the same as in Propositional Logic.

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.

Properties of Sentences

Valid

A sentence is *valid* if and only if *every* interpretation satisfies it.

Contingent

A sentence is *contingent* if and only if *some* interpretation satisfies it and *some* interpretation falsifies it.

Unsatisfiable

A sentence is *unsatisfiable* if and only if *no* interpretation satisfies it.

A sentence is valid if and only if it is satisfied by every truth assignment. A sentence is unsatisfiable if and only if it is not satisfied by any truth assignment. A sentence is contingent if and only if there is some truth assignment that satisfies it and some truth assignment that falsifies it.

Properties of Sentences

Valid

} A sentence is *satisfiable* if and only if it is either valid or contingent.

Contingent

} A sentence is *falsifiable* if and only if it is contingent or unsatisfiable.

Unsatisfiable

A sentence is satisfiable if and only if it is valid or contingent, i.e. if and only if there is at least one truth assignment that satisfies it. A sentence is falsifiable if and only if it is either contingent or unsatisfiable, i.e. if and only if there is at least one truth assignment that falsifies it.

Relationships

A sentence ϕ is *logically equivalent* to a sentence ψ if and only if ϕ and ψ have the *same value* for every truth assignment.

A sentence ϕ is *consistent with* a sentence ψ if and only if there is a truth assignment that satisfies both ϕ and ψ .

A sentence ϕ *logically entails* a sentence ψ (written $\phi \models \psi$) if and only if every truth assignment that satisfies ϕ also satisfies ψ .

Sets of Sentences

A set of sentences Γ *logically entails* a set of sentences Δ (written as $\Gamma \models \Delta$) if and only if every truth assignment that satisfies *all* of the sentences in Γ satisfies *all* of the sentences in Δ . (Think of sets as conjunctions.)

Ditto for equivalence and consistency.

Metatheorems

Metatheorems

Propositional Metatheorems:

Monotonicity Theorem (*More premises mean more conclusions.*)

Ramification Theorem (*If many conclusions, then few conclusions.*)

Equivalence Theorem (φ equivalent to ψ iff $\varphi \leftrightarrow \psi$ is valid.)

Substitution Theorem (*If $\varphi \leftrightarrow \psi$ is valid, $\chi_{\varphi \rightarrow \psi}$ equivalent to χ .*)

Deduction Theorem ($\varphi \models \psi$ iff $(\varphi \Rightarrow \psi)$ is valid.)

Unsatisfiability Theorem ($\Delta \models \varphi$ iff $\Delta \cup \{\neg \varphi\}$ is unsatisfiable.)

Consistency Theorem (φ is consistent ψ iff $(\varphi \wedge \psi)$ is satisfiable.)

*These theorems also hold in Relational Logic provided that all sentences are **closed** (i.e. they have no **free** variables).*

Distributing Quantifiers over Quantifiers

Common Quantifier Reversal:

$$\forall x. \forall y. q(x, y) \models \forall y. \forall x. q(x, y)$$

$$\exists x. \exists y. q(x, y) \models \exists y. \exists x. q(x, y)$$

Distributing Existentials over Universals:

$$\exists y. \forall x. q(x, y) \models \forall x. \exists y. q(x, y)$$

Distributing Universals over Existentials *not cool*:

No! No!! No!!! $\forall x. \exists y. q(x, y) \models \exists y. \forall x. q(x, y)$ No! No!! No!!!

Of course, not all sentences in Relational Logic are ground. Here are some valid non-ground sentences of Relational Logic for which there are no corresponding sentences in Propositional Logic. The Common Quantifier Reversal axioms tell us that reversing quantifiers of the same type has no effect on truth assignment. Existential Distribution allows us to move an existential quantifier inside of a universal quantifier. (Note, however, that the reverse is not true, as we shall see later.) Finally, Negation Distribution allows us to distribute negation of quantification of either type by flipping the quantifier and negating the scope of the quantified sentence.

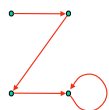
Distributing Quantifiers over Quantifiers

Distributing Existentials over Universals:

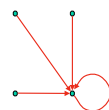
$$\exists y. \forall x. q(x, y) \models \forall x. \exists y. q(x, y)$$

Distributing Universals over Existentials *not cool*:

$$\forall x. \exists y. q(x, y) \models \exists y. \forall x. q(x, y)$$



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



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

Of course, not all sentences in Relational Logic are ground. Here are some valid non-ground sentences of Relational Logic for which there are no corresponding sentences in Propositional Logic. The Common Quantifier Reversal axioms tell us that reversing quantifiers of the same type has no effect on truth assignment. Existential Distribution allows us to move an existential quantifier inside of a universal quantifier. (Note, however, that the reverse is not true, as we shall see later.) Finally, Negation Distribution allows us to distribute negation of quantification of either type by flipping the quantifier and negating the scope of the quantified sentence.

Distributing Quantifiers over Operators

Distributing Quantifiers over Negations:

$$\exists x. \neg p(x) \models \neg \forall x. p(x)$$

$$\forall x. \neg p(x) \models \neg \exists x. p(x)$$

Distributing Quantifiers over Conjunctions:

$$\forall x. (p(x) \wedge q(x)) \models \forall x. p(x) \wedge \forall x. q(x)$$

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

Distributing Quantifiers over Disjunctions:

$$\exists x. (p(x) \vee q(x)) \models \exists x. p(x) \vee \exists x. q(x)$$

$$\text{No! No!! No!!! } \forall x. (p(x) \vee q(x)) \models \forall x. p(x) \vee \forall x. q(x) \text{ No! No!! No!!!}$$

$$\forall x. (p(x) \vee q(b)) \models \forall x. p(x) \vee q(b)$$

Of course, not all sentences in Relational Logic are ground. Here are some valid non-ground sentences of Relational Logic for which there are no corresponding sentences in Propositional Logic. The Common Quantifier Reversal axioms tell us that reversing quantifiers of the same type has no effect on truth assignment. Existential Distribution allows us to move an existential quantifier inside of a universal quantifier. (Note, however, that the reverse is not true, as we shall see later.) Finally, Negation Distribution allows us to distribute negation of quantification of either type by flipping the quantifier and negating the scope of the quantified sentence.

Distributing Quantifiers over Operators

Object Constants: a, b

$$\text{No! No!! No!!! } \forall x. (p(x) \vee q(x)) \models \forall x. p(x) \vee \forall x. q(x) \text{ No! No!! No!!!}$$

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

In this case:

$$\forall x. (p(x) \vee q(x)) \text{ is true but}$$

$$\forall x. p(x) \text{ is false and } \forall x. q(x) \text{ is false}$$

Distributing Quantifiers over Implications

Implication Distribution:

$$\forall y. (p(a) \Rightarrow q(y)) \models (p(a) \Rightarrow \forall y. q(y))$$

$$\forall x. (p(x) \Rightarrow q(b)) \models (\exists x. p(x) \Rightarrow q(b))$$

$$\forall x. \forall y. (p(x) \Rightarrow q(y)) \models (\exists x. p(x) \Rightarrow \forall y. q(y))$$

Distributing Quantifiers over Implications

Implication Distribution:

$$\forall y.(p(a) \Rightarrow q(y)) \models (p(a) \Rightarrow \forall y.q(y))$$

$$\forall x.(p(x) \Rightarrow q(b)) \models (\exists x.p(x) \Rightarrow q(b))$$

$$\forall x.\forall y.(p(x) \Rightarrow q(y)) \models (\exists x.p(x) \Rightarrow \forall y.q(y))$$

Derivation:

$$\forall x.(p(x) \Rightarrow q(b)) \models \forall x.(\neg p(x) \vee q(b))$$

$$\models (\forall x.\neg p(x) \vee q(b))$$

$$\models (\neg \exists x.p(x) \vee q(b))$$

$$\models (\exists x.p(x) \Rightarrow q(b))$$

Common, very useful distribution.

Distributing Operators over Quantifiers

Distributing Negations over Quantifiers:

$$\neg \forall x.p(x) \models \exists x.\neg p(x)$$

$$\neg \exists x.p(x) \models \forall x.\neg p(x)$$

Distributing Conjunctions over Quantifiers:

$$\forall x.p(x) \wedge \forall x.q(x) \models \forall x.(p(x) \wedge q(x))$$

$$\text{No! No!! No!!! } \exists x.p(x) \wedge \exists x.q(x) \models \exists x.(p(x) \wedge q(x)) \text{ No! No!! No!!!}$$

$$\exists x.p(x) \wedge q(b) \models \exists x.(p(x) \wedge q(b))$$

Distributing Disjunctions over Quantifiers:

$$\exists x.p(x) \vee \exists x.q(x) \models \exists x.(p(x) \vee q(x))$$

$$\forall x.p(x) \vee \forall x.q(x) \models \forall x.(p(x) \vee q(x))$$

Distributing Operators over Quantifiers

$$\forall x.p(x) \wedge \forall x.q(x) \models \forall x.(p(x) \wedge q(x))$$

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

$$\text{No! No!! No!!! } \exists x.p(x) \wedge \exists x.q(x) \models \exists x.(p(x) \wedge q(x)) \text{ No! No!! No!!!}$$

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

Truth Table Method

Truth tables and semantic trees are good ways of explicitly representing multiple models for a set of sentences. In some cases, there is just one model.

In this approach, we write out an empty table for each relation and then fill in values based on the constraints of the problem. For example, for any unit constraint, we can immediately enter the corresponding truth value in the appropriate box. Given these partial assignments, we then simplify the constraints, possibly leading to new unit constraints. We continue until there are no more unit constraints.

Determining Logical Entailment

$$\{m \Rightarrow p \vee q, p \Rightarrow q\} \models m \Rightarrow q?$$

m	p	q	$m \Rightarrow p \vee q$	$p \Rightarrow q$	$m \Rightarrow q$
1	1	1	1	1	1
1	1	0	1	0	0
1	0	1	1	1	1
1	0	0	0	1	0
0	1	1	1	1	1
0	1	0	1	0	1
0	0	1	1	1	1
0	0	0	1	1	1

As in Propositional Logic, it is in principle possible to build a truth table for any set of sentences in Relational Logic. This truth table can then be used to determine validity, satisfiability, and so forth or to determine logical entailment and logical equivalence.

As an example, let us assume we have a language with just two object constants a and b and two relation constants p and q . Now consider the sentences shown below, and assume we want to know whether these sentences logically entail $\exists x.q(x)$.

$p(a) \vee p(b)$

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

A truth table for this problem is shown below. Each of the first four columns represents one of the elements of the Herbrand base for this language. The two middle columns represent our premises, and the final column represents the conclusion.

Looking at the table, we see that there are 12 truth assignments that make the first premise true and nine that make the second premise true and five that make them both true (rows 1, 5, 6, 9, and 11). Note that every truth assignment that makes both premises true also makes the conclusion true. Hence, the premises logically entail the conclusion.

Determining Logical Entailment

Question:

$$\{p(a) \vee p(b), \forall x.(p(x) \Rightarrow q(x))\} \models \exists x.q(x)?$$

As in Propositional Logic, it is in principle possible to build a truth table for any set of sentences in Relational Logic. This truth table can then be used to determine validity, satisfiability, and so forth or to determine logical entailment and logical equivalence.

As an example, let us assume we have a language with just two object constants a and b and two relation constants p and q . Now consider the sentences shown below, and assume we want to know whether these sentences logically entail $\exists x.q(x)$.

$$p(a) \vee p(b)$$

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

A truth table for this problem is shown below. Each of the first four columns represents one of the elements of the Herbrand base for this language. The two middle columns represent our premises, and the final column represents the conclusion.

Looking at the table, we see that there are 12 truth assignments that make the first premise true and nine that make the second premise true and five that make them both true (rows 1, 5, 6, 9, and 11). Note that every truth assignment that makes both premises true also makes the conclusion true. Hence, the premises logically entail the conclusion.

Determining Logical Entailment

Question:

$$\{p(a) \vee p(b), \forall x.(p(x) \Rightarrow q(x))\} \models \exists x.q(x)?$$

Object Constants: a, b

Unary Relation Constants: p, q

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

As in Propositional Logic, it is in principle possible to build a truth table for any set of sentences in Relational Logic. This truth table can then be used to determine validity, satisfiability, and so forth or to determine logical entailment and logical equivalence.

As an example, let us assume we have a language with just two object constants a and b and two relation constants p and q . Now consider the sentences shown below, and assume we want to know whether these sentences logically entail $\exists x.q(x)$.

$$p(a) \vee p(b)$$

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

A truth table for this problem is shown below. Each of the first four columns represents one

of the elements of the Herbrand base for this language. The two middle columns represent our premises, and the final column represents the conclusion.

Looking at the table, we see that there are 12 truth assignments that make the first premise true and nine that make the second premise true and five that make them both true (rows 1, 5, 6, 9, and 11). Note that every truth assignment that makes both premises true also makes the conclusion true. Hence, the premises logically entail the conclusion.

Determining Logical Entailment

$\{p(a) \vee p(b), \forall x.(p(x) \Rightarrow q(x))\} \models \exists x.q(x)?$

$p(a)$	$p(b)$	$q(a)$	$q(b)$	$p(a) \vee p(b)$	$\forall x.(p(x) \Rightarrow q(x))$	$\exists x.q(x)$
1	1	1	1	1	1	1
1	1	1	0	1	0	1
1	1	0	1	1	0	1
1	1	0	0	1	0	0
1	0	1	1	1	1	1
1	0	1	0	1	1	1
1	0	0	1	1	0	1
1	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	0	1	0	0
0	0	1	1	0	1	1
0	0	1	0	0	1	1
0	0	0	1	0	1	1
0	0	0	0	0	1	0

As in Propositional Logic, it is in principle possible to build a truth table for any set of sentences in Relational Logic. This truth table can then be used to determine validity, satisfiability, and so forth or to determine logical entailment and logical equivalence.

As an example, let us assume we have a language with just two object constants a and b and two relation constants p and q . Now consider the sentences shown below, and assume we want to know whether these sentences logically entail $\exists x.q(x)$.

$p(a) \vee p(b)$

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

A truth table for this problem is shown below. Each of the first four columns represents one of the elements of the Herbrand base for this language. The two middle columns represent our premises, and the final column represents the conclusion.

Looking at the table, we see that there are 12 truth assignments that make the first premise true and nine that make the second premise true and five that make them both true (rows 1, 5, 6, 9, and 11). Note that every truth assignment that makes both premises true also makes the conclusion true. Hence, the premises logically entail the conclusion.

Determining Logical Entailment

$$\{p(a) \vee p(b), \forall x.(p(x) \Rightarrow q(x))\} \models \exists x.q(x)?$$

$p(a)$	$p(b)$	$q(a)$	$q(b)$	$p(a) \vee p(b)$	$\forall x.(p(x) \Rightarrow q(x))$	$\exists x.q(x)$
1	1	1	1	1	1	1
1	1	1	0	1	0	1
1	1	0	1	1	0	1
1	1	0	0	1	0	0
1	0	1	1	1	1	1
1	0	1	0	1	1	1
1	0	0	1	1	0	1
1	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	0	1	0	0
0	0	1	1	0	1	1
0	0	1	0	0	1	1
0	0	0	1	0	1	1
0	0	0	0	0	1	0

As in Propositional Logic, it is in principle possible to build a truth table for any set of sentences in Relational Logic. This truth table can then be used to determine validity, satisfiability, and so forth or to determine logical entailment and logical equivalence.

As an example, let us assume we have a language with just two object constants a and b and two relation constants p and q . Now consider the sentences shown below, and assume we want to know whether these sentences logically entail $\exists x.q(x)$.

$$p(a) \vee p(b)$$

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

A truth table for this problem is shown below. Each of the first four columns represents one of the elements of the Herbrand base for this language. The two middle columns represent our premises, and the final column represents the conclusion.

Looking at the table, we see that there are 12 truth assignments that make the first premise true and nine that make the second premise true and five that make them both true (rows 1, 5, 6, 9, and 11). Note that every truth assignment that makes both premises true also makes the conclusion true. Hence, the premises logically entail the conclusion.

Analysis

Object constants: n
 Binary relation constants: k
 Factoids in Herbrand Base: $k \cdot n^2$
 Interpretations: $2^{k \cdot n^2}$

Object constants: 4
 Binary relation constants: 4
 Factoids in Herbrand Base: 64
 Interpretations: $2^{64} = 18,446,744,073,709,551,616$

Methods

Truth Tables / Models

Guaranteed

Often Impractical

Constraint Satisfaction (model creation + proofs)

Boolean Grids (aka Logic Grids)

Non-Boolean Grids

Proofs

Guaranteed

Usually smaller than models

Next time

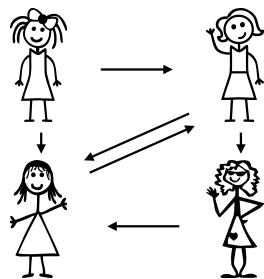
Sometimes not as intuitive as models

Boolean Grids / Logic Grids

Truth tables and semantic trees are good ways of explicitly representing multiple models for a set of sentences. In some cases, there is just one model.

In this approach, we write out an empty table for each relation and then fill in values based on the constraints of the problem. For example, for any unit constraint, we can immediately enter the corresponding truth value in the appropriate box. Given these partial assignments, we then simplify the constraints, possibly leading to new unit constraints. We continue until there are no more unit constraints.

Friends



There are just four members - Abby, Bess, Cody, and Dana. Some girls like other girls and some do not.

Logical Sentences

Dana likes Cody.
Abby does not like Dana.
Dana does not like Abby.
Abby likes everyone that Bess likes.
Bess likes Cody or Dana.
Abby and Dana both dislike Bess.
Cody likes everyone who likes her.
Nobody likes herself.

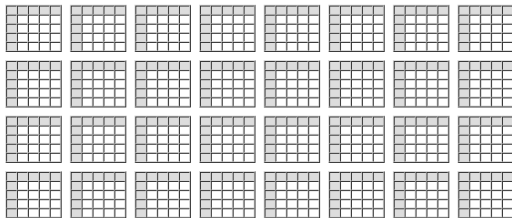
Here we see one such collection of sentences about these likes and dislikes. The first sentence is straightforward; it tells us directly that Dana likes Cody. The second and third sentences tell us what is not true, without saying what is true. The fourth sentence says that one condition holds or another but does not say which. The fifth sentence gives a general fact about the girls Abby likes. The sixth sentence expresses a general fact about Cody's likes. The last sentence says something about everyone.

Logical Sentences

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$
 $likes(bess, cody) \vee likes(bess, dana)$
 $\neg likes(abby, bess) \wedge \neg likes(dana, bess)$
 $\forall x. (likes(x, cody) \Rightarrow likes(cody, x))$
 $\neg \exists x. likes(x, x)$

Here the sentences encoded in the language of Relational Logic.

All Possible Truth Assignments



2^{16} (65,536) truth assignments.

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.

Only One Truth Assignment Works

	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.

Logic Grid

	Abby	Bess	Cody	Dana
Abby				
Bess				
Cody				
Dana				

In this particular case, it turns out that there is just one model that satisfies all of these sentences. One way to create this model is to use a logic grid like the one we saw in the first lecture and to use constraint propagation rather than exhaustive search of all 65 thousand possible rows.

Logic Grid

	Abby	Bess	Cody	Dana
Abby				0
Bess				
Cody				
Dana	0		1	

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$

The data we are given has one atomic sentence and two negated atomic sentences - the fact that Dana likes Cody and the facts that Abby does not like Dana and Dana does not like Abby. Using this information we can refine our model by putting a one into the third box in the fourth row and putting zeros in the fourth box of the first row and the first box of the fourth row.

Logic Grid

	Abby	Bess	Cody	Dana
Abby				0
Bess				0
Cody				
Dana	0		1	

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$

Logic Grid

	Abby	Bess	Cody	Dana
Abby			1	0
Bess			1	0
Cody				
Dana	0		1	

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$
 $likes(bess, cody) \vee likes(bess, dana)$

Logic Grid

	Abby	Bess	Cody	Dana
Abby		0	1	0
Bess			1	0
Cody				
Dana	0	0	1	

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$
 $likes(bess, cody) \vee likes(bess, dana)$
 $\neg likes(abby, bess) \wedge \neg likes(dana, bess)$

Logic Grid

	Abby	Bess	Cody	Dana
Abby		0	1	0
Bess			1	0
Cody	1	1		1
Dana	0	0	1	

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$
 $likes(bess, cody) \vee likes(bess, dana)$
 $\neg likes(abby, bess) \wedge \neg likes(dana, bess)$
 $\forall x. (likes(x, cody) \Rightarrow likes(cody, x))$

Logic Grid

	Abby	Bess	Cody	Dana
Abby	0	0	1	0
Bess		0	1	0
Cody	1	1	0	1
Dana	0	0	1	0

$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$
 $likes(bess, cody) \vee likes(bess, dana)$
 $\neg likes(abby, bess) \wedge \neg likes(dana, bess)$
 $\forall x. (likes(x, cody) \Rightarrow likes(cody, x))$
 $\neg \exists x. likes(x, x)$

Logic Grid

	Abby	Bess	Cody	Dana
Abby	0	0	1	0
Bess	0	0	1	0
Cody	1	1	0	1
Dana	0	0	1	0

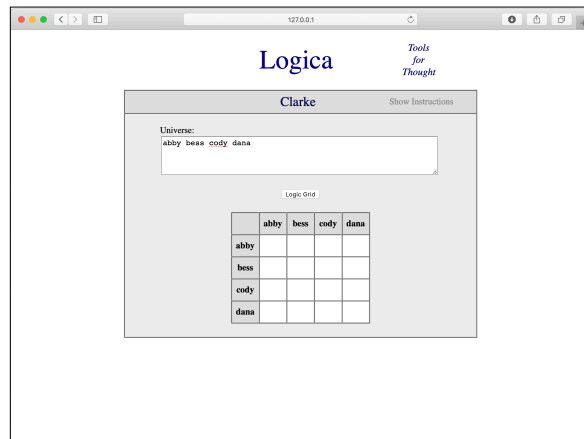
$likes(dana, cody)$
 $\neg likes(abby, dana)$
 $\neg likes(dana, abby)$
 $\forall y. (likes(bess, y) \Rightarrow likes(abby, y))$
 $likes(bess, cody) \vee likes(bess, dana)$
 $\neg likes(abby, bess) \wedge \neg likes(dana, bess)$
 $\forall x. (likes(x, cody) \Rightarrow likes(cody, x))$
 $\neg \exists x. likes(x, x)$

At this point, we have a complete model, and we can check our conclusion to see that this model satisfies the desired conclusion. In this case, it is easy to see that Dana indeed does like everyone that Bess likes.

We motivated this method by talking about cases where the given sentences have a unique model, as in this case. However, the method can also be of value even when there are multiple possible models. For example, if we had left out the belief that Cody likes everyone who likes her, we would still have eight models (corresponding to the eight possible combinations of feelings Cody has for Abby, Bess, and Dana). Yet, even with this ambiguity, it would be possible to determine whether Dana likes everyone Bess likes using just the portion of the table already filled in.

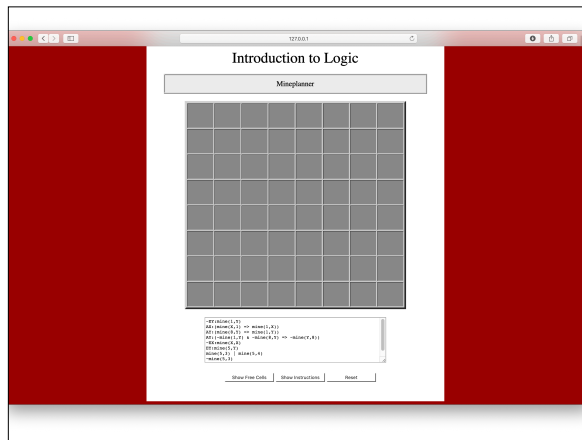
Course Website

<http://logica.stanford.edu>



Course Website

<http://intrologic.stanford.edu>



Show without last three.

$AX:AY:(\text{mine}(X,Y) \Rightarrow \text{mine}(Y,X))$ leaves only the diagonal.

Non-Boolean Grids

As defined in Chapter 6, a model in Relational Logic is an assignment of truth values to the ground atoms of our language. We treat each ground atom in our language as a variable and assign it a single truth value (1 or 0). In general, this is a good way to proceed. However, we can sometimes do better.

Consider, for example, a theory with four object constants and two unary relation constants. In this case, there would be eight elements in the Herbrand base and 28 (256) possible truth assignments. Now, suppose we had the constraint that each relation is true of at most a single object. Most of these assignments would not satisfy the single value constraint and thus considering them is wasteful.

Luckily, in cases like this, there is a representation for truth assignments that allows us to eliminate such possibilities and thereby save work. Rather than treating each ground atom as a separate variable with its own Boolean value, we can think of each relation as a variable with 4 possible values. In order to analyze sentences in such a theory, we would need to consider only 42 (16) possibilities.

Even if we search the entire space of assignments, this is a significant saving over the pure truth table method. Moreover, we can combine this representation with the techniques described earlier to find assignments for these non-Boolean variables in an even more efficient manner.

Sudoku

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
	6					3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

Applying these updates again leads to an overall solution to the problem. Since neither Ruby Red nor Betty Blue is wearing blue, Willa White must be wearing blue.

This problem is special in that we can solve it solely by inferring values from other values. In constraint satisfaction problems like this one, some search is often necessary. That said, constraint propagation techniques, like the one used here, we can often cut down on this search even when they cannot be used to solve the problem altogether.

This case is also special in that it is easy to express all of the update rules necessary to solve the problem. For some problems, such as solving Sudoku puzzles, it is impractical to write update rules using our limited update language. Luckily, as we shall see in future lessons, we can easily express more complicated rules once we have the ability to write view definitions and action definitions.

Sukoshi

	4		1
2			
			3
		4	

The game of Sukoshi illustrates this technique and its benefits. (Sukoshi is similar to Sudoku, but it is smaller and simpler.) A typical Sukoshi puzzle is played on a 4x4 board. In a typical instance of Sukoshi, several of the squares are already filled, as in the example below. The goal of the game is to place the numerals 1 through 4 in the remaining squares of the board in such a way that no numeral is repeated in any row or column.

47

Axiomatization

cell(1,2,4)
cell(1,4,1)
cell(2,1,2)
cell(3,4,3)
cell(4,3,4)

same(1,1) $\neg$ *same*(2,1) $\neg$ *same*(3,1) $\neg$ *same*(4,1)
 $\neg$ *same*(1,2) *same*(2,2) $\neg$ *same*(3,2) $\neg$ *same*(4,2)
 $\neg$ *same*(1,3) $\neg$ *same*(2,3) *same*(3,3) $\neg$ *same*(4,3)
 $\neg$ *same*(1,4) $\neg$ *same*(2,4) $\neg$ *same*(3,4) *same*(4,4)

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

$\forall x. \forall y. \forall z. \forall w. (\text{cell}(x,y,w) \wedge \text{cell}(x,y,z) \Rightarrow \text{same}(w,z))$

$\forall x. \forall y. \forall z. \forall w. (\text{cell}(x,y,w) \wedge \text{cell}(x,z,w) \Rightarrow \text{same}(y,z))$

$\forall x. \forall y. \forall z. \forall w. (\text{cell}(x,z,w) \wedge \text{cell}(y,z,w) \Rightarrow \text{same}(x,y))$

We can formalize the rules of this puzzle in the language of Logic. Once we have done that, we can use the techniques described here to find a solution. In our formalization, we use the expression *cell*(1,2,3) to express the fact that the cell in the first row and the second column contains the numeral 3. For example, we can describe the initial board shown above with the following sentences.

We use the expression *same*(x,y) to say that x is the same as y. We can axiomatize *same* by simply stating when it is true and where it is false. An abbreviated axiomatization is shown here.

Using this vocabulary, we can write the rules defining Sukoshi as shown here. The first sentence expresses the constraint that two cells in the same row cannot contain the same

value. The second sentence expresses the constraint that two cells in that same column cannot contain the same value. The third constraint expresses the fact that every cell must contain at least one value.

Analysis

Object constants: 4
Ternary relation constants: 1
Factoids in Herbrand Base: 64
Truth Assignments: $2^{64} = 18,446,744,073,709,551,616$

Analysis

Object constants: 4
Ternary relation constants: 1
Factoids in Herbrand Base: 64
Truth Assignments: $2^{64} = 18,446,744,073,709,551,616$

Non-Boolean Grids: $4^{16} = 4,294,967,296$

Sukoshi

	4		1
2			
			3
		4	

51

We start with our initial board.

Sukoshi

	4		1
2			4
			3
		4	

52

As a first step in solving this problem, we start by focussing on the fourth column, since two of the cells in that column are already filled. We know that there must be a 4 in one of the cells. It cannot be the first, since that cell contains a 1, and it cannot be the third since that cell contains a 3. It also cannot be the fourth, since there is already a 4 in the third cell of the fourth row. By process of elimination, the 4 must go in the fourth cell of the second row, leading to the board shown here.

Sukoshi

	4		1
2			4
4			3
		4	

At this point, there is a four in every row and every column except for the first column in the third row. So, we can safely place a four in that cell.

Sukoshi

	4		1
2			4
4			3
		4	2

54

Since there is just one empty cell in the fourth column, we know it must be filled with the single remaining value, viz. 2. After adding this value, we have the following board.

Sukoshi

	4		1
2			4
4			3
1		4	2

55

Now, let's turn our attention to the first column. We know that there must be a 1 in one of the cells. It cannot be the first, since there is already a 1 in that row, and it cannot be the second or third since those cells already contain values. Consequently, the 1 must go in the first cell of the fourth row.

Sukoshi

3	4		1
2			4
4			3
1		4	2

56

Once again, we have a column with all but one cell filled. Column 1 has a 2 in the second cell, a 4 in the third, and a 1 in the fourth. So, we can place a 3 in the first cell of that column.

Sukoshi

3	4	2	1
2			4
4			3
1		4	2

57

At this point, we can fill in the single empty cell in the first row, leading to the following board.

Sukoshi

3	4	2	1
2			4
4			3
1	3	4	2

58

And we can fill in the single empty cell in the fourth row as well.

Sukoshi

3	4	2	1
2			4
4	2		3
1	3	4	2

59

Now, let's consider the second column. We cannot put a 2 in the second cell, since there is already a 2 in that row. Since the first and last cells are already full, the only option is to put the 2 into the third cell.

Sukoshi

3	4	2	1
2			4
4	2	1	3
1	3	4	2

Finishing off the third row leads to the board below.

Sukoshi

3	4	2	1
2		3	4
4	2	1	3
1	3	4	2

Finishing off the third column leads to the following board.

Sukoshi

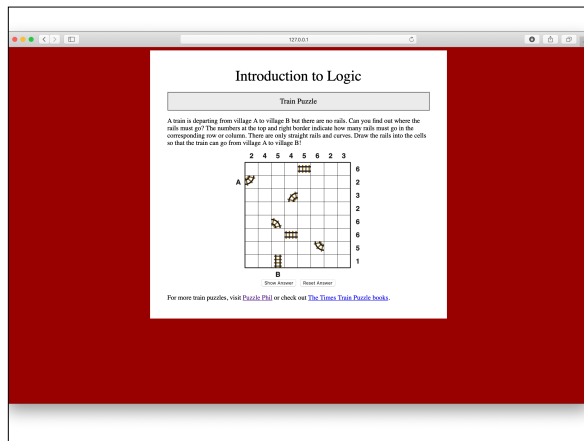
3	4	2	1
2	1	3	4
4	2	1	3
1	3	4	2

Finally, we can place a 1 in the second cell of the second row. And, with that, the board is full. We have a distinct numeral in every row and every column, as required by the rules.

Given the initial assignment in this case, it is fairly easy to find a complete assignment that satisfies the Sukoshi constraints. For other initial assignments, solving the problem is more difficult. However, the techniques described here still work to cut down on the amount of work necessary. In fact, virtually all Sukoshi puzzles can be solved using these techniques without any form of trial and error.

Course Website

<http://intrologic.stanford.edu>



Zebra Puzzle

There is a row of five houses.
The Englishman lives in the red house.
The Spaniard owns the dog.
Coffee is drunk in the green house.
The Ukrainian drinks tea.
The green house is immediately to the right of the ivory house.
The Old Gold smoker owns snails.
Kools are smoked in the yellow house.
Milk is drunk in the middle house.
The Norwegian lives in the first house.
The man who smokes Chesterfields lives in the house next to the man with the fox.
Kools are smoked in the house next to the house where the horse is kept.
The Lucky Strike smoker drinks orange juice.
The Japanese smokes Parliaments.
The Norwegian lives next to the blue house.

Credit to Einstein or Carroll? Kools did not exist then. Life magazine in 1962.

Zebra Puzzle

There is a row of five houses.
The Englishman lives in the red house.
The Spaniard owns the dog.
Coffee is drunk in the green house.
The Ukrainian drinks tea.
The green house is immediately to the right of the ivory house.
The Old Gold smoker owns snails.
Kools are smoked in the yellow house.
Milk is drunk in the middle house.
The Norwegian lives in the first house.
The man who smokes Chesterfields lives in the house next to the man with the fox.
Kools are smoked in the house next to the house where the horse is kept.
The Lucky Strike smoker drinks orange juice.
The Japanese smokes Parliaments.
The Norwegian lives next to the blue house.

Who owns the Zebra?

Credit to Einstein or Carroll? Kools did not exist then. Life magazine in 1962.

Relational Logic and Propositional Logic

Mapping

There is a simple procedure for mapping RL sentences to equivalent PL sentences.

- (1) Convert to Prenex form.
- (2) Compute the grounding.
- (3) Rewrite from RL in PL.

The procedure for transforming our RL sentences to PL has multiple steps but each step is easy. First, we convert to prenex form, then we ground the result, and we rewrite in Propositional Logic. Let's look at these steps in turn.

Prenex Form

A sentence is in *prenex form* if and only if (1) it is closed and (2) all of the quantifiers are outside of all logical operators.

Sentence in Prenex Form:

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

Sentences *not* in Prenex Form:

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

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

A sentence is in prenex form if and only if it is closed and all of the quantifiers are on the outside of all logical operators. For example, the first sentence here is in prenex form. The two sentences at the bottom are not. The first fails because the quantifiers are inside of logical operators. The second fails because the sentence contains a free occurrence of the variable y .

Conversion to Prenex Form

Rename duplicate variables.

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

Distribute logical operators over quantifiers.

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

Quantify any free variables.

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

Converting a set of RL sentences to a logically equivalent set in prenex form is simple. First, we rename variables in different quantified sentences to eliminate any duplicates. In the example shown here, we rename the variable y in the second disjunct. In the second step, we apply quantifier distribution rules to move quantifiers outside of logical operators. Here, we move the two quantifiers out. Finally, we universally quantify any free variables in our sentences. In this case, the variable x is free in the sentence, and so we add a universal quantifier.

Grounding

Instantiate all quantified sentences.

(1) Leave all ground sentences as is.

(2) Replace every universally quantified sentence by all instances of its scope.

(3) Replace every existentially quantified sentence by a disjunction of instances of its scope.

Once we have a set of sentences in prenex form, we compute the **grounding**. The procedure is described here. We start with our initial set Delta of sentences and we incrementally build up our grounding Gamma. On each step we process a sentence in Delta, using the rules shown here. The procedure terminates when Delta becomes empty. The set Gamma at that point is the grounding of the input. The first rule covers the case when the sentence being processed is ground. In this case, we remove the sentence from Delta and add it to Gamma.

Grounding

Object constants: a, b

Unary Relations constants: p, q

$\{p(a), \forall x.(p(x) \Rightarrow q(x)), \exists x.q(x)\}$

$p(a)$

$p(a)$

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

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

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

$\exists x.q(x)$

$q(a) \vee q(b)$

Once we have a set of sentences in prenex form, we compute the **grounding**. The procedure is described here. We start with our initial set Delta of sentences and we incrementally build up our grounding Gamma. On each step we process a sentence in Delta, using the rules shown here. The procedure terminates when Delta becomes empty. The set Gamma at that point is the grounding of the input. The first rule covers the case when the sentence being processed is ground. In this case, we remove the sentence from Delta and add it to Gamma.

Renaming RL to PL

Select a proposition for each ground relational sentence and rewrite the grounding from RL to PL.

RL Grounding:

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

Corresponding PL:

$p(a) \leftrightarrow pa$

$q(a) \leftrightarrow qa$

$p(b) \leftrightarrow pb$

$q(b) \leftrightarrow qb$

Corresponding PL:

$\{pa, pa \Rightarrow qa, pb \Rightarrow qb, qa \vee qb\}$

Once we have a grounding Gamma, we replace each ground relational sentence in Gamma by a proposition constant. The resulting sentences are all in Propositional Logic; and the set is equivalent to the sentences in Delta* in that any RL truth assignment that satisfies our RL sentences agrees with the corresponding Propositional Logic truth assignment applied to the Propositional Logic sentences.

For example, let's represent the RL sentence $p(a)$ with the proposition pa ; let's represent $p(b)$ with pb ; let's represent $q(a)$ with qa ; and let's represent $q(b)$ with qb . With this correspondence, we can represent the sentences in our grounding with the Propositional Logic sentences shown below.

Decidability

Unsatisfiability and logical entailment for Propositional Logic (PL) is decidable.

Given our mapping, we also know that unsatisfiability and logical entailment for Relational Logic (RL) is also decidable.

Since the question of unsatisfiability for PL is decidable, then the question of unsatisfiability for RL is also decidable. Since logical entailment and unsatisfiability are correlated, we also know that the question of logical entailment for RL is decidable.

