Introduction to Logic
Propositional Logic

Michael Genesereth
Computer Science Department
Stanford University

Multiple Logics

→ Propositional Logic (logical operators)

*If **raining** and **cold**, then **wet**.*

Relational Logic (variables and quantifiers)

*If **abby** likes **x**, then **bess** likes **x**.*

Functional Logic (functional terms)

*{**a**, **b**} is a subset of {**a**, **b**, **c**}.*

As mentioned in our last lesson, we will be covering three types of Logic in this course - propositional logic, relational logic, and term logic. In this first section of the course, we concentrate on Propositional Logic.

Propositional Logic is the logic of propositions. Symbols in the language represent monolithic "conditions" in the world, and complex sentences in the language express interrelationships among these conditions. The primary operators are Boolean connectives, such as and, or, not, if, and if and only if.

Example

If Mary loves Pat, then Mary loves Quincy.

If it is Monday, then Mary loves Pat or Quincy.

If it is Monday, does Mary love Quincy?

If it is Monday, does Mary love Pat?

Here is a simple propositional logic problem. If Mary loves Pat, then Mary loves Quincy. If it is Monday, then Mary loves Pat or Quincy. If it is Monday, does Mary love Quincy? Yes. If it is Monday, does Mary love Pat?

Example

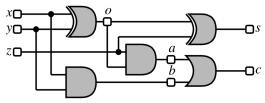
Victor has been murdered, and Art, Bob, and Carl are suspects. Art says he did not do it. He says that Bob was the victim's friend but that Carl hated the victim. Bob says he was out of town the day of the murder, and besides he didn't even know the guy. Carl says he is innocent and he saw Art and Bob with the victim just before the murder. You can assume that everyone is telling the truth - except for the murderer.

Whodunnit? (Answer: Bob)

http://intrologic.stanford.edu/extras/whodunnit.html

Here is another example, drawn from the course website. Victor has been murdered, and Art, Bob, and Carl are suspects. Art says he did not do it. He says that Bob was the victim's friend but that Carl hated the victim. Bob says he was out of town the day of the murder, and besides he didn't even know the guy. Carl says he is innocent and he saw Art and Bob with the victim just before the murder. You can assume that everyone is telling the truth - except for the murderer. Whodunnit? This problem is more complicated than the case of Mary, Pat, and Quincy; but it can be solved using the same techniques.

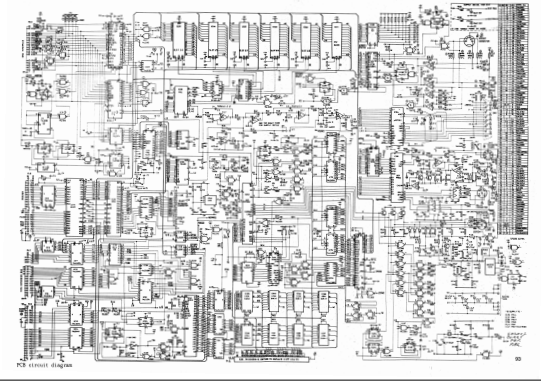
Digital Circuits



http://intrologic.stanford.edu/extras/circuits.html

The analysis of Boolean circuits is practical application for Propositional Logic. The little squares here represent boolean values 1 or 0 (representing true or false). The grey elements are gates that produce output values based on their inputs. The output of an and gate is true iff both inputs are true. The output of an or gate is true iff at least one of the inputs are true. The output of an xor gate is true iff the inputs disagree. If the inputs are 101, what is the first output? What is the second output? How do we test whether or not the circuit has a fault? If there is a fault, how do we figure out which gate is broken?

Digital Circuits Example



That circuit may not be so hard to analyze. But consider how you might proceed if you have this circuit instead.

Programme

Basics
- syntactically legal sentences
- meaning of syntactically legal sentences

Evaluation
- Given truth values for simple sentences, find truth values of complex sentences.

Satisfaction
- Given truth values for complex sentences, find truth values of simple sentences.

Examples
- Natural Language
- Digital Circuits

In this session, we first look at the syntactic rules that define the language of Propositional Logic, and we introduce the notion of a truth assignment and use it to define the meaning of Propositional Logic sentences. After that, we present a mechanical method for evaluating complex sentences for given values of simple sentences. We then present a mechanical method for finding truth assignments for simple sentences that satisfy given values for complex sentences. We conclude with some applications of Propositional Logic in Natural Language understanding and Digital Circuit analysis.

Programme

Basics
- syntactically legal sentences
- meaning of syntactically legal sentences

Evaluation
- Given truth values for simple sentences, find truth values of complex sentences.

Satisfaction
- Given truth values for complex sentences, find truth values of simple sentences.

Examples
- Natural Language
- Digital Circuits

In this session, we first look at the syntactic rules that define the language of Propositional Logic, and we introduce the notion of a truth assignment and use it to define the meaning of Propositional Logic sentences. After that, we present a mechanical method for evaluating complex sentences for given values of simple sentences. We then present a mechanical method for finding truth assignments for simple sentences that satisfy given values for complex sentences. We conclude with some applications of Propositional Logic in Natural Language understanding and Digital Circuit analysis.

Syntax

Propositional Sentences

Proposition Constants
express simple facts about the world
In English - *It is raining*.
In our language - *raining*

Compound Sentences
express relationships between sentences
In English - *It is raining* ***or*** *it is snowing*.
In our language - *raining* ∨ *snowing*

In Propositional Logic, there are two types of sentences -- simple sentences and compound sentences. Simple sentences express simple facts about the world. For example, we would use the propositional constant raining to encode the fact that it is raining. Compound sentences express logical relationships between the simpler sentences of which they are composed. For example, we would use the compound sentence raining l snowing to encode the fact that it is raining *or* it is snowing.

Proposition Constants

By convention (in this course), proposition constants are written as strings of letters, digits, and underscores ("_") beginning with a lower case letter.

Examples:
raining
rAiNiNg
raining_or_snowing

Non-Examples:
Raining
324567
raining-or-snowing

In what follows, we write proposition constants as strings of letters, digits, and underscores ("_"), where the first character is a lower case letter. For example, raining is a proposition constant, as are rAiNiNg (with some letters capitalized) and raining_or_snowing. Raining is not a proposition constant because it begins with an upper case character. 324567 fails because it begins with a number. raining-or-snowing fails because it contains hyphens (instead of underscores).

Compound Sentences (part I)

Negations:

(¬*raining*)

The argument of a negation is called the *target*.

Compound sentences are formed from simpler sentences and express relationships among the constituent sentences. There are five types of compound sentences, viz. negations, conjunctions, disjunctions, implications, and biconditionals.

A negation consists of the negation operator ¬ and an arbitrary sentence, called the target. For example, given the sentence p, we can form the negation of p as shown in the first example here.

Compound Sentences (part I)

Negations:

$(\neg raining)$

The argument of a negation is called the *target*.

Conjunctions:

$(raining \wedge snowing)$

The arguments of a conjunction are called *conjuncts*.

A conjunction is a sequence of sentences separated by occurrences of the $\wedge$ operator and enclosed in parentheses, as shown in the second example. The constituent sentences are called conjuncts.

Compound Sentences (part I)

Negations:

$(\neg raining)$

The argument of a negation is called the *target*.

Conjunctions:

$(raining \wedge snowing)$

The arguments of a conjunction are called *conjuncts*.

Disjunctions:

$(raining \vee snowing)$

The arguments of a disjunction are called *disjuncts*.

A disjunction is a sequence of sentences separated by occurrences of the $\vee$ operator and enclosed in parentheses. The constituent sentences are called disjuncts. For example, we can form the disjunction of p and q as shown in the third example.

Compound Sentences (part II)

Implications:

$(raining \Rightarrow cloudy)$

The left argument of an implication is the *antecedent*.
The right argument is the *consequent*.

Biconditionals:

$(cloudy \Leftrightarrow raining)$

An implication consists of a pair of sentences separated by the => operator and enclosed in parentheses. The implication of p and q is shown in the first example here. The sentence to the left of the operator is called the antecedent, and the sentence to the right is called the consequent.

A biconditional is a combination of an implication and a reverse implication. For example, we can express the biconditional of p and q as shown in the second example.

Nested Compound Sentences

$(\neg raining)$
$(raining \wedge snowing)$
$(raining \vee snowing)$
$(raining \Rightarrow cloudy)$
$(cloudy \Leftrightarrow raining)$

$\neg(raining \wedge snowing)$
$((raining \wedge snowing) \Rightarrow cloudy)$
$(cloudy \Rightarrow (raining \wedge snowing))$
$((cloudy \wedge wet) \Leftrightarrow (raining \vee snowing))$
$(\neg raining \Rightarrow (cloudy \Rightarrow snowing))$

Note that the constituent sentences within any compound sentence can be either simple sentences (as illustrated by the examples in the first group here) or they can themselves be compound sentences (as shown in the examples in the second group).

Parentheses Removal

Dropping Parentheses is good:

$$(p \wedge q) \rightarrow p \wedge q$$

But it can lead to ambiguities:

$$((p \vee q) \wedge r) \rightarrow p \vee q \wedge r$$

$$(p \vee (q \wedge r)) \rightarrow p \vee q \wedge r$$

One disadvantage of our notation, as written, is that the parentheses tend to build up and need to be matched correctly. It would be nice if we could dispense with parentheses, e.g. simplifying the sentence on the left here to the one on the right.

Unfortunately, we cannot do without parentheses entirely, since then we would be unable to render certain sentences unambiguously. For example, the sentence shown on the left could have resulted from dropping parentheses from either of the following sentences.

Precedence

Parentheses can be dropped when the structure of an expression can be determined by precedence.

$\neg$
$\wedge$
$\vee$
$\Rightarrow$
$\Leftrightarrow$

An operand surrounded by operators associates with operator of higher precedence.

$$\neg p \vee q \rightarrow ((\neg p) \vee q)$$
$$p \vee q \wedge r \rightarrow (p \vee (q \wedge r))$$
$$p \wedge q \Rightarrow r \rightarrow ((p \wedge q) \Rightarrow r)$$
$$p \Rightarrow q \Leftrightarrow r \rightarrow ((p \Rightarrow q) \Leftrightarrow r)$$

The solution to this problem is the use of operator precedence. The table here gives a hierarchy of precedences for our operators. The $\neg$ operator has higher precedence than $\wedge$; $\wedge$ has higher precedence than $\vee$; $\vee$ has higher precedence than $\Rightarrow$; and $\Rightarrow$ has higher precedence than $\Leftrightarrow$.

In sentences without parentheses, it is often the case that an expression is flanked by operators, one on either side. In interpreting such sentences, the question is whether the expression associates with the operator on its left or the one on its right. We can use precedence to make this determination. In particular, we agree that an operand in such a situation always associates with the operator of higher precedence. The examples here show how these rules work in various cases. The expressions on the right are the fully parenthesized versions of the expressions on the left.

Precedence (continued)

If surrounded by occurrences of $\wedge$ or $\vee$, the operand associates with the operator to the left.

$$p \wedge q \wedge r \rightarrow ((p \wedge q) \wedge r)$$
$$p \vee q \vee r \rightarrow ((p \vee q) \vee r)$$

If surrounded by two occurrences of $\Rightarrow$ or $\Leftrightarrow$, the operand associates with the operator to the right.

$$p \Rightarrow q \Rightarrow r \rightarrow (p \Rightarrow (q \Rightarrow r))$$
$$p \Leftrightarrow q \Leftrightarrow r \rightarrow (p \Leftrightarrow (q \Leftrightarrow r))$$

When an operand is surrounded by two $\wedge$ operators or by two $\vee$ operators, the operand associates to the left. When an operand is surrounded by two $\Rightarrow$ operators or by two $\Leftrightarrow$ operators, the operand associates to the right.

Useful Definitions

A *propositional vocabulary* is a set of *proposition constants*.

Given a propositional vocabulary, a *propositional sentence* is either
(1) a proposition constant or
(2) a compound sentence.

A *propositional language* is the set of *all* propositional sentences that can be formed from a propositional vocabulary.

Before leaving our discussion of Syntax, let's look at the definitions of a few concepts that are useful in discussing Propositional Logic. A propositional vocabulary is a set of proposition constants. A propositional sentence is either a proposition constant in teh vocabulary or a compound sentence that can be formed from the constants in the vocabulary. A propositional language is the set of all propositional sentences that can be formed from a propositional vocabulary.

Semantics

The treatment of semantics in Logic is similar to its treatment in Algebra. Algebra is unconcerned with the real-world significance of variables. What is interesting are the relationships among the values of the variables expressed in the equations we write. Algebraic methods are designed to respect these relationships, independent of what the variables represent.

In a similar way, Logic is unconcerned with the real world significance of proposition constants. What is interesting is the relationship among the truth values of simple sentences and the truth values of compound sentences within which the simple sentences are contained. As with Algebra, logical reasoning methods are independent of the significance of proposition constants; all that matter is the form of sentences.

Propositional Interpretation

A *propositional interpretation* is an association between the propositional constants in a propositional language and the values T or F.

$p \xrightarrow{i} \text{T}$ $\quad p^i = \text{T}$
$q \xrightarrow{i} \text{F}$ $\quad q^i = \text{F}$
$r \xrightarrow{i} \text{T}$ $\quad r^i = \text{T}$

We sometimes write 1 and 0 in place of T and F.

$p^i = 1$
$q^i = 0$
$r^i = 1$

Although the values assigned to proposition constants are not crucial in the sense just described, in talking about Logic, it is sometimes useful to make truth assignments explicit and to consider various assignments or all assignments and so forth. Such an assignment is called a truth assignment.

Formally, a truth assignment for a propositional vocabulary is a function assigning a truth value to each of the proposition constants of the vocabulary. In what follows, we use the digit 1 as a synonym for true and 0 as a synonym for false; and we refer to the value of a constant or expression under a truth assignment i by superscripting the constant or expression with i as the superscript.

Multiple Interpretations

Logic does not prescribe which interpretation is "correct". In the absence of additional information, one interpretation is as good as another.

Interpretation *i*	Interpretation *j*
$p^i = \text{T}$	$p^j = \text{F}$
$q^i = \text{F}$	$q^j = \text{F}$
$r^i = \text{T}$	$r^j = \text{T}$

Multiple Interpretations

Logic does not prescribe which interpretation is "correct". In the absence of additional information, one interpretation is as good as another.

Interpretation *i*	Interpretation *j*
$p^i = \text{T}$	$p^j = \text{F}$
$q^i = \text{F}$	$q^j = \text{F}$
$r^i = \text{T}$	$r^j = \text{T}$

Examples:

- Different days of the week
- Different locations
- Beliefs of different people

Sentential Interpretation

A *sentential interpretation* is an association between the sentences in a propositional language and truth values.

$p^i = \mathrm{T}$ $\quad$ $(p \vee q)^i = \mathrm{T}$
$q^i = \mathrm{F}$ $\quad$ $(\neg q \vee r)^i = \mathrm{T}$
$r^i = \mathrm{T}$ $\quad$ $((p \vee q) \wedge (\neg q \vee r))^i = \mathrm{T}$

NB: Each distinct *propositional interpretation* gives rise to a *unique sentential interpretation* due to operator semantics.

Semantics of Negations

A *negation* is true if and only if the target is false.

ϕ	$\neg\phi$
T	F
F	T

For example, if the interpretation of p is F, then the interpretation of $\neg p$ is T.

For example, if the interpretation of $(p \wedge q)$ is T, then the interpretation of $\neg(p \wedge q)$ is F.

Semantics of Conjunctions

A *conjunction* is true if and only if both conjuncts are true.

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

For example, if the interpretation of p is true and q is true, then $(p \wedge q)$ is true.

Semantics of Disjunctions

A *disjunction* is true if and only if at least one of the disjuncts is true.

ϕ	ψ	$\phi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

The type of disjunction here is called *inclusive or*. This contrasts with *exclusive or*, which says that a disjunction is true if and only if an *odd number* of disjuncts are true.

Semantics of Biconditionals

A *biconditional* is true if and only if the truth values of its two constituents are the same.

ϕ	ψ	$\phi \Leftrightarrow \psi$
T	T	T
T	F	F
F	T	F
F	F	T

Semantics of Implications

An *implication* is true if and only if the antecedent is false *or* the consequent is true.

ϕ	ψ	$\phi \Rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

The semantics of implication here is called *material implication*.

What Choice Do We Have?

ϕ	ψ	$\phi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

ϕ	ψ	$\phi \Leftrightarrow \psi$
T	T	T
T	F	F
F	T	F
F	F	T

ϕ	ψ	ψ
T	T	T
T	F	F
F	T	T
F	F	F

ϕ	ψ	$\phi \Rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

Implications and Biconditionals

ϕ	ψ	$\phi \Rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

ϕ	ψ	$\psi \Rightarrow \varphi$
T	T	T
T	F	T
F	T	F
F	F	T

$\varphi \Leftrightarrow \psi$ is true if and only if $\varphi \Rightarrow \psi$ and $\psi \Rightarrow \varphi$ are true.

ϕ	ψ	$\phi \Leftrightarrow \psi$
T	T	T
T	F	F
F	T	F
F	F	T

Counterfactuals

An *implication* is true if and only if the antecedent is false or the consequent is true.

ϕ	ψ	$\phi \Rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

A *counterfactual* is an implication in which the antecedent is false.

Counterfactuals are Weird

A *counterfactual* is an implication in which the antecedent is false.

NB: Counterfactuals are always *true!* The truth value of the consequent does not matter.

Examples:

It is raining ⇒ *I am a billionaire*
It is raining ⇒ *I am a pauper*
Shakespeare is alive ⇒ *Shakespeare is dead*
2+2=5 ⇒ 2+2=7

Evaluation

Evaluation is the process of determining the truth values of compound sentences given a truth assignment for the truth values of proposition constants.

Evaluation

Interpretation *i*:

p^i = T
q^i = T
r^i = F

Compound Sentence

$(p \vee q) \wedge (\neg q \vee r)$
(T ∨ T) ∧ (¬T ∨ F)
(T ∨ T) ∧ (F ∨ F)
T ∧ F
F

As it turns out, there is a simple technique for evaluating complex sentences. We substitute true and false values for the proposition constants in our sentence, forming an expression with 1s and 0s and logical operators. We use our operator semantics to evaluate subexpressions with these truth values as arguments. We then repeat, working from the inside out, until we have a truth value for the sentence as a whole.

Evaluation

Interpretation i:

$$p^i = \text{T}$$
$$q^i = \text{F}$$
$$r^i = \text{T}$$

Compound Sentence

$$(p \vee q) \wedge (\neg q \vee r)$$
$$(\text{T} \vee \text{F}) \wedge (\neg \text{F} \vee \text{T})$$
$$(\text{T} \vee \text{F}) \wedge (\text{T} \vee \text{T})$$
$$\text{T} \quad \wedge \quad \text{T}$$
$$\text{T}$$

Evaluation

Interpretation i:

$$p^i = \text{T}$$
$$q^i = \text{F}$$
$$r^i = \text{T}$$

Compound Sentence

$$(p \wedge q) \vee (\neg q \wedge r)$$
$$(\text{T} \wedge \text{F}) \vee (\neg \text{F} \wedge \text{T})$$
$$(\text{T} \wedge \text{F}) \vee (\text{T} \wedge \text{T})$$
$$\text{F} \quad \vee \quad \text{T}$$
$$\text{T}$$

Using this technique, we can evaluate the truth of arbitrary sentences in our language. The cost is proportional to the size of the sentence. Of course, in some cases, it is possible to economize and do even better. For example, when evaluating a conjunction, if we discover that the first conjunct is false, then there is no need to evaluate the second conjunct since the sentence as a whole must be false.

Satisfaction

Evaluation versus Satisfaction

Evaluation:

$p^i = \text{T}$, $q^i = \text{F}$ $\longrightarrow$ $(p \vee q)^i = \text{T}$, $(\neg q)^i = \text{T}$

Satisfaction:

$(p \vee q)^i = \text{T}$, $(\neg q)^i = \text{T}$ $\longrightarrow$ $p^i = \text{T}$, $q^i = \text{F}$

Satisfaction is the opposite of evaluation. We begin with one or more compound sentences and try to figure out which truth assignments satisfy those sentences. One nice feature of Propositional Logic is that there are effective procedures for finding truth assignments that satisfy Propositional Logic sentences. In this section, we look at a method based on truth tables.

Propositional Interpretation

A *propositional interpretation* is an association between the propositional constants in a propositional language and the values T or F.

$p \xrightarrow{i} \text{T}$ $\quad p^i = \text{T}$

$q \xrightarrow{i} \text{F}$ $\quad q^i = \text{F}$

$r \xrightarrow{i} \text{T}$ $\quad r^i = \text{T}$

Reminder.

Truth Tables

A *truth table* is a table of all possible interpretations for the propositional constants in a language.

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

One column per constant.

One row per interpretation.

For a language with n constants, there are 2^n interpretations.

A truth table for a propositional language is a table showing all of the possible truth assignments for the proposition constants in the language. The columns of the table correspond to the proposition constants of the language, and the rows correspond to different truth assignments for those constants.

The figure here shows a truth table for a propositional language with just three proposition constants (p, q, and r). Each column corresponds to one proposition constant, and each row corresponds to a single truth assignment. The truth assignments i and j defined in the preceding section correspond to the third and sixth rows of this table, respectively.

Note that, for a language with n proposition constants, there are n columns in the truth table and 2^n rows.

Truth Table Method

Method to find all propositional interpretations that satisfy a given set of sentences:

(1) Form a truth table for the propositional constants.

(2) For each sentence in the set and each row in the truth table, check whether the row satisfies the sentence. If not, cross out the row.

(3) Any row remaining satisfies all sentences in the set. (Note that there might be more than one.)

In solving satisfaction problems, we start with a truth table for the proposition constants of our language. We then process our sentences in turn, for each sentence placing an x next to rows in the truth table corresponding to truth assignments that do not satisfy the sentence. The truth assignments remaining at the end of this process are all possible truth assignments of the input sentences.

Are these sentences satisfiable?

$$q \Rightarrow r$$
$$p \Rightarrow q \wedge r$$
$$\neg r$$

Satisfaction Example

$p \Rightarrow q \wedge r$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

Satisfaction Example

$q \Rightarrow r$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

$p \Rightarrow q \wedge r$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

Satisfaction Example

$q \Rightarrow r$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

$p \Rightarrow q \wedge r$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

$\neg r$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
F	F	F

Satisfaction Example

$\{q \Rightarrow r, p \Rightarrow q \wedge r, \neg r\}$

p	q	r
T	T	T
T	T	F
T	F	T
T	F	F
F	T	T
F	T	F
F	F	T
→ F	F	F ←

The disadvantage of the truth table method is computational complexity. The size of a truth table for a language grows exponentially with the number of proposition constants in the language. When the number of constants is small, the method works well. When the number is large, the method becomes impractical. Even for moderate sized problems, it can be tedious. Even for an application like Sorority World, where there are only 16 proposition constants, there are 65,536 truth assignments.

Over the years, researchers have proposed ways to improve the performance of truth table checking. However, the best approach to dealing with large vocabularies is to use symbolic manipulation (i.e. logical reasoning and proofs) in place of truth table checking.

Logica

Course Website

http://logica.stanford.edu

Logica

Stickel
Clausal Form Converter

Wegman
Unifier

Babbage
Truth Table Generator

Clarke
Logic Grid Editor

Quine
Sentence Evaluator

Russell
Constraint Satisfier

Herbrand
Sentence Analyzer

Hilbert
Hilbert-style Proof Editor

Fitch
Fitch-style Proof Editor

Robinson
Resolution Proof Editor

Babbage Show Instructions

Premises:

```
q=>r
p => q&r
¬r
```

Truth Table

Constants			Premises		
q	r	p	q => r	p => q & r	¬r
1	1	1	1	1	0
1	1	0	1	1	0
1	0	1	0	0	1
1	0	0	0	1	1
0	1	1	1	0	0
0	1	0	1	1	0
0	0	1	1	0	1
0	0	0	1	1	1

Computing Satisfiability

Truth Table Method

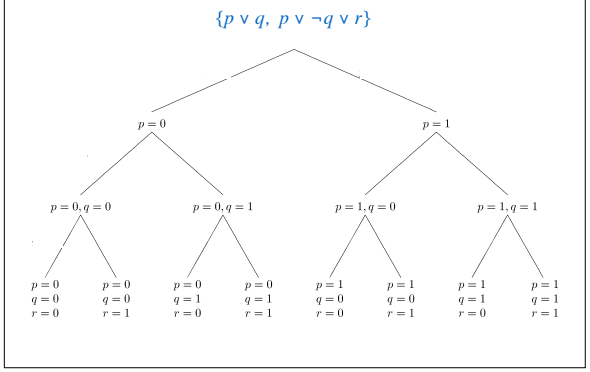
$\{p \vee q,\ p \vee \neg q \vee r\}$

p	q	r	$p \vee q$	$p \vee \neg q \vee \neg r$	Δ
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

We can determine whether a set of sentences is satisfiable by building a truth table and checking whether each row satisfies the sentences. Let Δ = {p∨q, p∨¬q∨r}. There is one row for each possible truth assignment. For each truth assignment, each of the sentences in Δ is evaluated. If any sentence evaluates to 0, then Δ as a whole is not satisfied by the truth assignment. If a satisfying truth assignment is found, then Δ is determined to be satisfiable. If no satisfying truth assignment is found, then Δ is unsatisfiable. In this example, there are several rows where Δ not satisfied. So Δ is satisfiable.

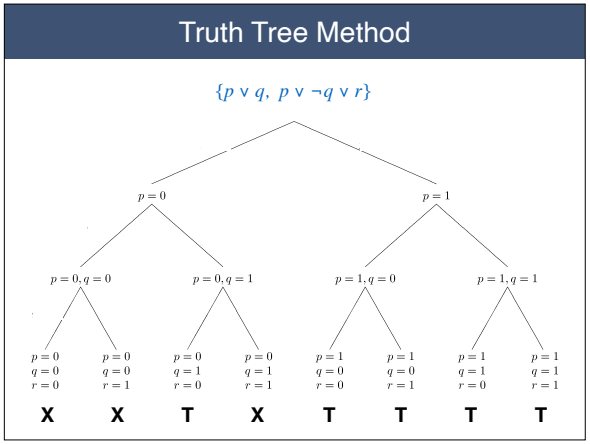
The truth table method is complete because every truth assignment is checked. However, the method is impractical for all but very small problem instances. In our example with 3 propositions, there 8 rows. For a problem instance with 10 propositions, there are 210 = 1024 rows - still quite small for a modern computer. But as the number of propositions grow, the number of rows quickly overwhelms even the fastest computers. A more efficient method is needed.

Truth Tree



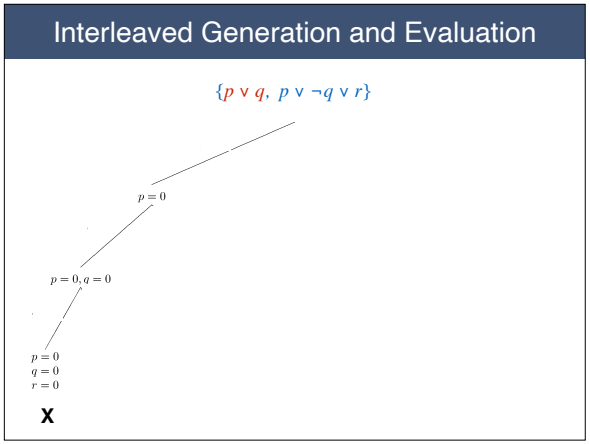
The truth tree method is an alternative to the truth table method. Instead of visualizing the possibilities as a table, we view the space of possibilities as a tree. The branches at each non-terminal node represent different values for one of the propositions in our language; and each terminal node represents a complete interpretation.

Truth Tree Method



Once we have generated the tree, we can check whether the truth assignment at each terminal node satisfies the given sentences. In this case, several of the assignments work, so the sentences are satisfiable. In this case, the cost is the same as a the truth table method. However, in some cases, we can do better time by interleaving generation and evaluation.

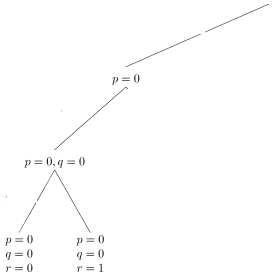
Interleaved Generation and Evaluation



As we generate each combination of values, we check whether it satisfies the sentences. If not, we just continue until the entire tree is generated. However, if we find an interpretation that satisfies the sentences, we can stop early.

Interleaved Generation and Evaluation

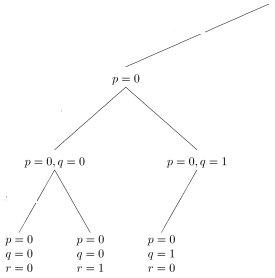
$\{p \vee q,\ p \vee \neg q \vee r\}$



X X

Interleaved Generation and Evaluation

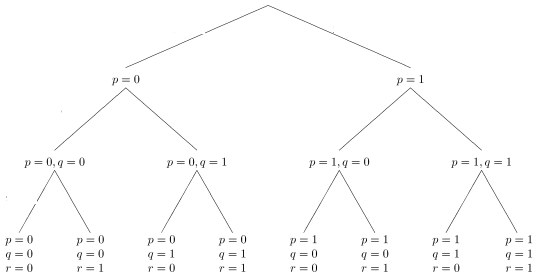
$\{p \vee q,\ p \vee \neg q \vee r\}$



X X T

Interleaved Generation and Evaluation

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



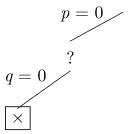
X X X X X X X X

Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$

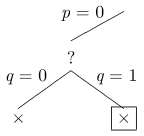
Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



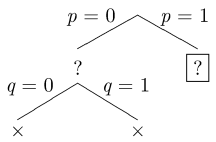
Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



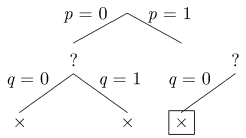
Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



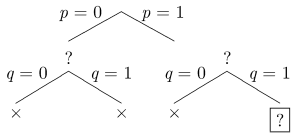
Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



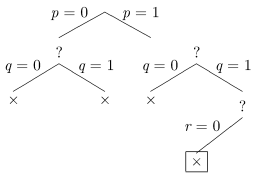
Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



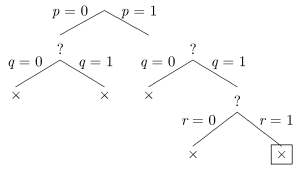
Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



Intermediate State Checking

$\{p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q \vee \neg r,\ \neg p \vee r\}$



Simplification and Unit Propagation

Constraints

$p \vee q$
$p \vee \sim q$
$\sim p \vee q$
$\sim p \vee \sim q \vee \sim r$
$\sim p \vee r$

Two additional techniques for backtracking search - simplification and unit propagation. In order for these methods to work, we assume that our sentences have been transformed into logically equivalent disjunctions (using a method like the one described in Lesson 5). As we choose the truth values of some propositions in a partial truth assignment for these disjunctions, there are opportunities to simplify the set of sentences that need to be checked.

Simplification and Unit Propagation

(1) Requires constraints to be written as disjunctions.

(2) As each proposition is assigned a value, simplify.

If $p = 0$, simplify the constraint.

$$p \vee q \longrightarrow q$$

If $p = 1$, drop the constraint.

$$p \vee q \longrightarrow \text{-}$$

(3) If the result is a "unit", set the value to 1 and repeat.

Simplification and Unit Propagation

Given $p = 1$

Original	**Simplified**
$p \vee q$	-
$p \vee \sim q$	-
$\sim p \vee q$	q
$\sim p \vee \sim q \vee \sim r$	$\sim q \vee \sim r$
$\sim p \vee r$	r

Simplification and Unit Propagation

Given $p = 1, q = 1$

Original	**Simplified**
$p \vee q$	-
$p \vee \sim q$	-
$\sim p \vee q$	-
$\sim p \vee \sim q \vee \sim r$	$\sim r$
$\sim p \vee r$	r

Simplification and Unit Propagation

Given $p = 1, q = 1, r = 1$

Original	Simplified
$p \vee q$	-
$p \vee \sim q$	-
$\sim p \vee q$	-
$\sim p \vee \sim q \vee \sim r$	x
$\sim p \vee r$	-

More on Computing Satisfaction

http://intrologic.stanford.edu/extras/satisfiability.html

The Big Game

The Big Game

Stanford people always tell the truth, and Berkeley people always lie. Unfortunately, by looking at a person, you cannot tell whether he is from Stanford or Berkeley.

You come to a fork in the road and want to get to the football stadium down one fork. However, you do not know which to take. There is a person standing there. What single question can you ask him to help you decide which fork to take?

Basic Idea

left	*su*	*Question*	*Response*
T	T		
T	F		
F	T		
F	F		

Desired Response

left	*su*	*Question*	*Response*
T	T		"T"
T	F		"T"
F	T		"F"
F	F		"F"

Desired Response

left	*su*	*Question*	*Response*
T	T	T	"T"
T	F	F	"T"
F	T	F	"F"
F	F	T	"F"

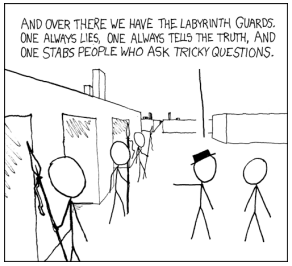
The Big Game (solved)

Question: *The left road is the way to the stadium if and only if you are from Stanford. Is that correct?*

$$left \Leftrightarrow su$$

Notice that we end up knowing the location of the stadium but we do not know whether the person is from Stanford or Berkeley. Still just one bit of information.

Let's Be Careful Out There



Tougher problems for which logic does not have answers.

xkcd (Randall Munroe)

Puzzle - rotors