Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Checking validity, equivalence, entailment can be reduced to satisfiability checking.

Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Checking validity, equivalence, entailment can be reduced to satisfiability checking.

Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Checking validity, equivalence, entailment can be reduced to satisfiability checking.

Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Checking validity, equivalence, entailment can be reduced to satisfiability checking.

Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Checking validity, equivalence, entailment can be reduced to satisfiability checking.

Propositional Satisfiability Problem

Given a propositional formula A, check whether it is satisfiable or unsatisfiable.

If A is satisfiable, we also want to find a satisfying assignment for A, that is, a model of A.

It is one of the most famous combinatorial problems in computer science.

It is a very hard problem with a surprisingly large number of practical applications.

It is also the first ever problem to be proved NP-complete.

Checking validity, equivalence, entailment can be reduced to satisfiability checking.

Russian spy puzzle

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.

We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Russian spy puzzle

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.

We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Russian spy puzzle

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.

We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Russian spy puzzle

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans. Moreover, every Russian must be a spy.

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”. It is known that Stirlitz always tells the truth when he is joking.

We have to show that Eismann is not a Russian spy.

How can we solve problems of this kind?

Formalisation in propositional logic

Introduce nine propositional variables as in the following table:

	Stirlitz	Müller	Eismann
Russian	RS	RM	RE
German	GS	GM	GE
Spy	SS	SM	SE

For example,

SE : Eismann is a Spy
RS : Stirlitz is Russian

Formalisation in propositional logic

Introduce nine propositional variables as in the following table:

	Stirlitz	Müller	Eismann
Russian	RS	RM	RE
German	GS	GM	GE
Spy	SS	SM	SE

For example,

SE : Eismann is a Spy
RS : Stirlitz is Russian

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \wedge GM \wedge GE) \vee (GS \wedge RM \wedge GE) \vee (GS \wedge GM \wedge RE).$$

Moreover, every Russian must be a spy.

$$(RS \rightarrow SS) \wedge (RM \rightarrow SM) \wedge (RE \rightarrow SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: "you know, Müller, you are as German as I am Russian".

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \wedge (RM \leftrightarrow \neg GM) \wedge (RE \leftrightarrow \neg GE).$$

We have to show that Eismann is not a Russian spy.

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \land GM \land GE) \lor (GS \land RM \land GE) \lor (GS \land GM \land RE).$$

Moreover, every Russian must be a spy.

$$(RS \to SS) \land (RM \to SM) \land (RE \to SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: "you know, Müller, you are as German as I am Russian".

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \land (RM \leftrightarrow \neg GM) \land (RE \leftrightarrow \neg GE).$$

We have to show that Eismann is not a Russian spy.

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \land GM \land GE) \lor (GS \land RM \land GE) \lor (GS \land GM \land RE).$$

Moreover, every Russian must be a spy.

$$(RS \to SS) \land (RM \to SM) \land (RE \to SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: "you know, Müller, you are as German as I am Russian".

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \land (RM \leftrightarrow \neg GM) \land (RE \leftrightarrow \neg GE).$$

We have to show that Eismann is not a Russian spy.

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \land GM \land GE) \lor (GS \land RM \land GE) \lor (GS \land GM \land RE).$$

Moreover, every Russian must be a spy.

$$(RS \to SS) \land (RM \to SM) \land (RE \to SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”.

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \land (RM \leftrightarrow \neg GM) \land (RE \leftrightarrow \neg GE).$$

We have to show that Eismann is not a Russian spy.

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \land GM \land GE) \lor (GS \land RM \land GE) \lor (GS \land GM \land RE).$$

Moreover, every Russian must be a spy.

$$(RS \to SS) \land (RM \to SM) \land (RE \to SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”.

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \land (RM \leftrightarrow \neg GM) \land (RE \leftrightarrow \neg GE).$$

We have to show that Eismann is not a Russian spy.

Formalisation in propositional logic

There are three persons: Stirlitz, Müller, and Eismann. It is known that exactly one of them is Russian, while the other two are Germans.

$$(RS \land GM \land GE) \lor (GS \land RM \land GE) \lor (GS \land GM \land RE).$$

Moreover, every Russian must be a spy.

$$(RS \to SS) \land (RM \to SM) \land (RE \to SE).$$

When Stirlitz meets Müller in a corridor, he makes the following joke: “you know, Müller, you are as German as I am Russian”.

$$RS \leftrightarrow GM.$$

Hidden: Russians are not Germans.

$$(RS \leftrightarrow \neg GS) \land (RM \leftrightarrow \neg GM) \land (RE \leftrightarrow \neg GE).$$

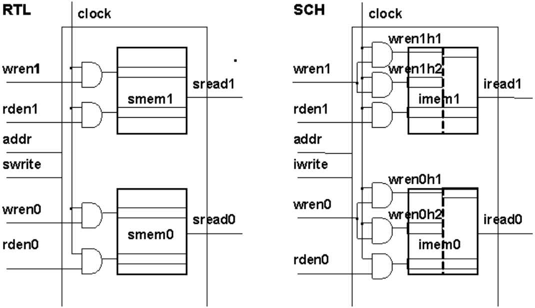
We have to show that Eismann is not a Russian spy.
To this end, we add the following formula

$$RE \land SE.$$

and check whether the resulting set of formulas is satisfiable. If it is unsatisfiable, then Eismann cannot be a Russian spy.

Circuit Equivalence

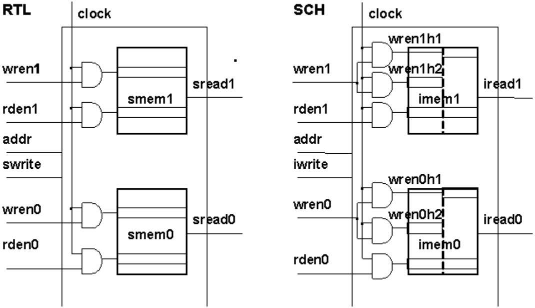
Given two circuits, check if they are equivalent. For example:



We know that equivalence-checking for propositional formulas can be reduced to unsatisfiability-checking.

Circuit Equivalence

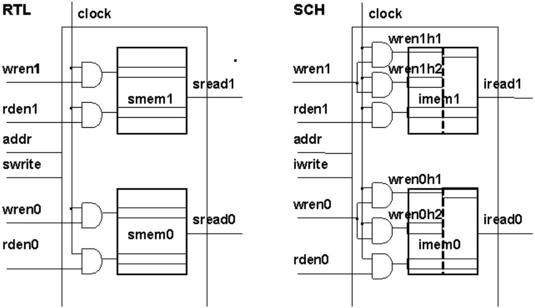
Given two circuits, check if they are equivalent. For example:



We know that equivalence-checking for propositional formulas can be reduced to unsatisfiability-checking.

Circuit Equivalence

Given two circuits, check if they are equivalent. For example:



We know that equivalence-checking for propositional formulas can be reduced to unsatisfiability-checking.

Truth tables

$\neg((p \to q) \land (p \land q \to r) \to (p \to r)$.

Likewise, we can evaluate it in all interpretations:

	subformula	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
1	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0	0	0	0	0
2	$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1	1	1	1	1
3	$p \to r$	1	1	1	1	0	1	0	1
4	$(p \to q) \land (p \land q \to r)$	1	1	1	1	0	0	0	1
5	$p \land q \to r$	1	1	1	1	1	1	0	1
6	$p \to q$	1	1	1	1	0	0	1	1
7	$p \land q$	0	0	0	0	0	0	1	1
8	p p p	0	0	0	0	1	1	1	1
9	q q	0	0	1	1	0	0	1	1
10	r r	0	1	0	1	0	1	0	1

The formula is unsatisfiable since it is false in every interpretation.

Problem: a formula with n propositional variables has 2^n different interpretations.

Truth tables

$\neg((p \to q) \land (p \land q \to r) \to (p \to r)$.

Likewise, we can evaluate it in all interpretations:

	subformula	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
1	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0	0	0	0	0
2	$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1	1	1	1	1
3	$p \to r$	1	1	1	1	0	1	0	1
4	$(p \to q) \land (p \land q \to r)$	1	1	1	1	0	0	0	1
5	$p \land q \to r$	1	1	1	1	1	1	0	1
6	$p \to q$	1	1	1	1	0	0	1	1
7	$p \land q$	0	0	0	0	0	0	1	1
8	p p p	0	0	0	0	1	1	1	1
9	q q	0	0	1	1	0	0	1	1
10	r r	0	1	0	1	0	1	0	1

The formula is unsatisfiable since it is false in every interpretation.

Problem: a formula with n propositional variables has 2^n different interpretations.

Truth tables

$\neg((p \to q) \land (p \land q \to r) \to (p \to r)$.

Likewise, we can evaluate it in all interpretations:

	subformula	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8
1	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0	0	0	0	0
2	$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1	1	1	1	1
3	$p \to r$	1	1	1	1	0	1	0	1
4	$(p \to q) \land (p \land q \to r)$	1	1	1	1	0	0	0	1
5	$p \land q \to r$	1	1	1	1	1	1	0	1
6	$p \to q$	1	1	1	1	0	0	1	1
7	$p \land q$	0	0	0	0	0	0	1	1
8	p p p	0	0	0	0	1	1	1	1
9	q q	0	0	1	1	0	0	1	1
10	r r	0	1	0	1	0	1	0	1

The formula is unsatisfiable since it is false in every interpretation.

Problem: a formula with n propositional variables has 2^n different interpretations.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
$p \quad p \quad p$	0	1	1	
$q \quad q$		0	1	
$r \quad r$	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
$p \quad p \quad p$	0	1	1	
$q \quad q$		0	1	
$r \quad r$	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \wedge (p \wedge q \to r)$		0	0	
$p \wedge q \to r$		1	0	1
$p \to q$		0	1	
$p \wedge q$		0	1	
p	0	1	1	
q		0	1	
r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p	0	1	1	
q		0	1	
r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p $\quad$ p $\quad$ p	0	1	1	
q $\quad$ q		0	1	
r $\quad$ r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p	0	1	1	
q		0	1	
r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r))$	0	0	0	0
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \rightarrow r)$	1	1	1	1
$p \rightarrow r$	1	0	0	1
$(p \rightarrow q) \wedge (p \wedge q \rightarrow r)$		0	0	
$p \wedge q \rightarrow r$		1	0	1
$p \rightarrow q$		0	1	
$p \wedge q$		0	1	
p p p	0	1	1	
q q		0	1	
r r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p p p	0	1	1	
q q		0	1	
r r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
$p \quad p \quad p$	0	1	1	
$q \quad q$		0	1	
$r \quad r$	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p	0	1	1	
q		0	1	
r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p $\quad p$ $\quad p$	0	1	1	
q $\quad q$		0	1	
r $\quad r$	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
$p \quad p \quad p$	0	1	1	
$q \quad q$		0	1	
$r \quad r$	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p p p	0	1	1	
q q		0	1	
r r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p p p	0	1	1	
q q		0	1	
r r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Compact truth table

Idea: we can sometimes evaluate a formula based on values of only a subset of all variables.

subformula	I_2	I_3	I_4	I_1
$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	0	0	0	0
$(p \to q) \land (p \land q \to r) \to (p \to r)$	1	1	1	1
$p \to r$	1	0	0	1
$(p \to q) \land (p \land q \to r)$		0	0	
$p \land q \to r$		1	0	1
$p \to q$		0	1	
$p \land q$		0	1	
p p p	0	1	1	
q q		0	1	
r r	0	0	0	1

The formula is unsatisfiable.

Note: the size of the compact table (but not the result) depends on the order of atoms!

The ideas of guessing variable values (or case analysis) and propagation are the key ideas in nearly all propositional satisfiability algorithms.

Splitting: idea

$A_p^\perp$ and $A_p^\top$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma

Let p be an atom, A be a formula, and I be an interpretation.

1. *If $I \not\models p$, then A is equivalent to $A_p^\perp$ in I.*
2. *If $I \models p$, then A is equivalent to $A_p^\top$ in I.*

- Pick a variable p and perform case analysis on this variable:
  - If p is false, replace p by $\perp$;
  - If p is true, replace p by $\top$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it.

Splitting: idea

$A_p^\perp$ and $A_p^\top$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma

Let p be an atom, A be a formula, and I be an interpretation.

1. *If $I \not\models p$, then A is equivalent to $A_p^\perp$ in I.*
2. *If $I \models p$, then A is equivalent to $A_p^\top$ in I.*

- Pick a variable p and perform case analysis on this variable:
  - If p is false, replace p by $\perp$;
  - If p is true, replace p by $\top$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it.

Splitting: idea

$A_p^{\perp}$ and $A_p^{\top}$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma

Let p be an atom, A be a formula, and I be an interpretation.

1. *If $I \not\models p$, then A is equivalent to $A_p^{\perp}$ in I.*
2. *If $I \models p$, then A is equivalent to $A_p^{\top}$ in I.*

- Pick a variable p and perform case analysis on this variable:
  - If p is false, replace p by $\perp$;
  - If p is true, replace p by $\top$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it.

Splitting: idea

$A_p^{\perp}$ and $A_p^{\top}$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma

Let p be an atom, A be a formula, and I be an interpretation.

1. *If $I \not\models p$, then A is equivalent to $A_p^{\perp}$ in I.*
2. *If $I \models p$, then A is equivalent to $A_p^{\top}$ in I.*

- Pick a variable p and perform case analysis on this variable:
  - If p is false, replace p by $\perp$;
  - If p is true, replace p by $\top$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it.

Splitting: idea

$A_p^{\perp}$ and $A_p^{\top}$: the formulas obtained by replacing in A all occurrences of p by $\perp$ and $\top$, respectively.

Lemma

Let p be an atom, A be a formula, and I be an interpretation.

1. *If $I \not\models p$, then A is equivalent to $A_p^{\perp}$ in I.*
2. *If $I \models p$, then A is equivalent to $A_p^{\top}$ in I.*

- Pick a variable p and perform case analysis on this variable:
  - If p is false, replace p by $\perp$;
  - If p is true, replace p by $\top$.
- When a formula contains occurrences of $\top$ or $\perp$, simplify it.

Simplification rules for $\top$ and $\bot$

Simplification rules for $\top$:

$$
\begin{array}{c}
\neg\top \;\Rightarrow\; \bot \\
\top \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; A_1 \wedge \ldots \wedge A_n \\
\top \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; \top \\
A \rightarrow \top \;\Rightarrow\; \top \qquad \top \rightarrow A \;\Rightarrow\; A \\
A \leftrightarrow \top \;\Rightarrow\; A \qquad \top \leftrightarrow A \;\Rightarrow\; A
\end{array}
$$

Simplification rules for $\bot$:

$$
\begin{array}{c}
\neg\bot \;\Rightarrow\; \top \\
\bot \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; \bot \\
\bot \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; A_1 \vee \ldots \vee A_n \\
A \rightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \rightarrow A \;\Rightarrow\; \top \\
A \leftrightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \leftrightarrow A \;\Rightarrow\; \neg A
\end{array}
$$

Note that they cover all cases when $\bot$ or $\top$ occurs in the formula apart from the trivial ones.

Thus, if we apply these rules until they are no more applicable we obtain either $\bot$, or $\top$, or a formula containing neither $\bot$ nor $\top$.

Simplification rules for $\top$ and $\bot$

Simplification rules for $\top$:

$$
\begin{array}{c}
\neg\top \;\Rightarrow\; \bot \\
\top \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; A_1 \wedge \ldots \wedge A_n \\
\top \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; \top \\
A \rightarrow \top \;\Rightarrow\; \top \qquad \top \rightarrow A \;\Rightarrow\; A \\
A \leftrightarrow \top \;\Rightarrow\; A \qquad \top \leftrightarrow A \;\Rightarrow\; A
\end{array}
$$

Simplification rules for $\bot$:

$$
\begin{array}{c}
\neg\bot \;\Rightarrow\; \top \\
\bot \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; \bot \\
\bot \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; A_1 \vee \ldots \vee A_n \\
A \rightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \rightarrow A \;\Rightarrow\; \top \\
A \leftrightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \leftrightarrow A \;\Rightarrow\; \neg A
\end{array}
$$

Note that they cover all cases when $\bot$ or $\top$ occurs in the formula apart from the trivial ones.

Thus, if we apply these rules until they are no more applicable we obtain either $\bot$, or $\top$, or a formula containing neither $\bot$ nor $\top$.

Simplification rules for $\top$ and $\bot$

Simplification rules for $\top$:

$$
\begin{array}{c}
\neg\top \;\Rightarrow\; \bot \\
\top \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; A_1 \wedge \ldots \wedge A_n \\
\top \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; \top \\
A \rightarrow \top \;\Rightarrow\; \top \qquad \top \rightarrow A \;\Rightarrow\; A \\
A \leftrightarrow \top \;\Rightarrow\; A \qquad \top \leftrightarrow A \;\Rightarrow\; A
\end{array}
$$

Simplification rules for $\bot$:

$$
\begin{array}{c}
\neg\bot \;\Rightarrow\; \top \\
\bot \wedge A_1 \wedge \ldots \wedge A_n \;\Rightarrow\; \bot \\
\bot \vee A_1 \vee \ldots \vee A_n \;\Rightarrow\; A_1 \vee \ldots \vee A_n \\
A \rightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \rightarrow A \;\Rightarrow\; \top \\
A \leftrightarrow \bot \;\Rightarrow\; \neg A \qquad \bot \leftrightarrow A \;\Rightarrow\; \neg A
\end{array}
$$

Note that they cover all cases when $\bot$ or $\top$ occurs in the formula apart from the trivial ones.

Thus, if we apply these rules until they are no more applicable we obtain either $\bot$, or $\top$, or a formula containing neither $\bot$ nor $\top$.

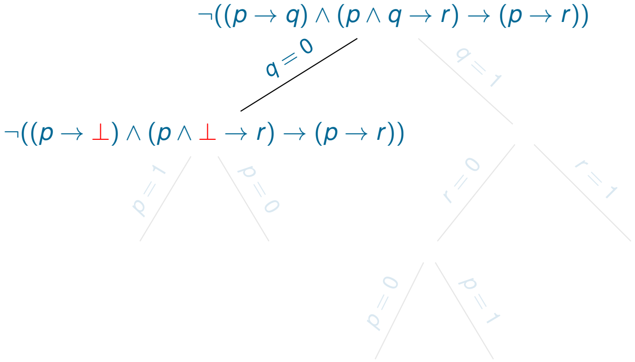
Splitting algorithm

procedure $split(G)$
parameters: function $select$
input: formula G
output: "satisfiable" or "unsatisfiable"
begin
  $G := simplify(G)$
  if $G = \top$ **then** **return** "satisfiable"
  if $G = \bot$ **then** **return** "unsatisfiable"
  $(p, b) := select(G)$
  case b **of**
  $1 \Rightarrow$
    if $split(G_p^\top) =$"satisfiable"
      then **return** "satisfiable"
      else **return** $split(G_p^\bot)$
  $0 \Rightarrow$
    if $split(G_p^\bot) =$"satisfiable"
      then **return** "satisfiable"
      else **return** $split(G_p^\top)$
end

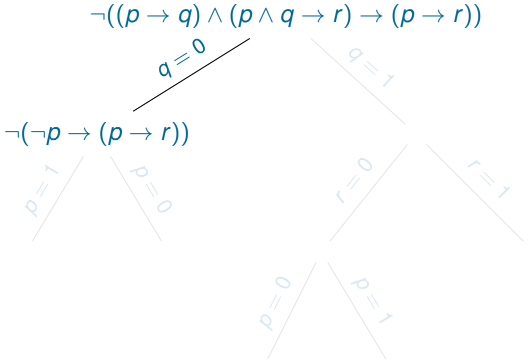
Splitting algorithm, example

$$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$$

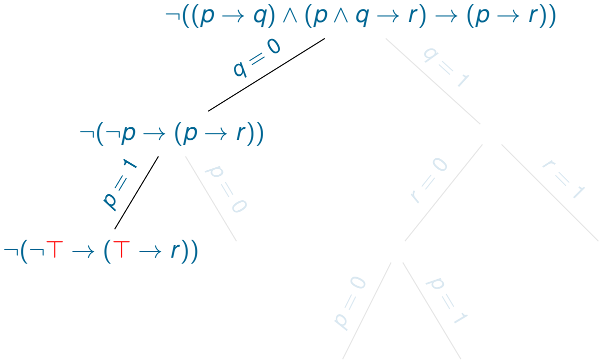
Splitting algorithm, example



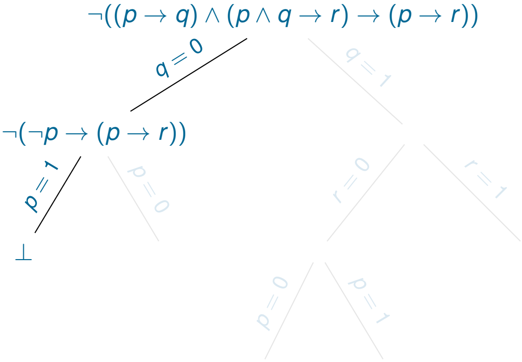
Splitting algorithm, example



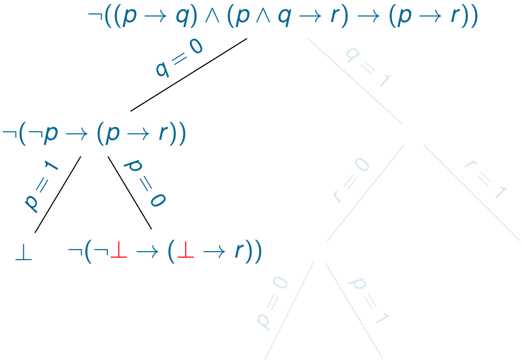
Splitting algorithm, example



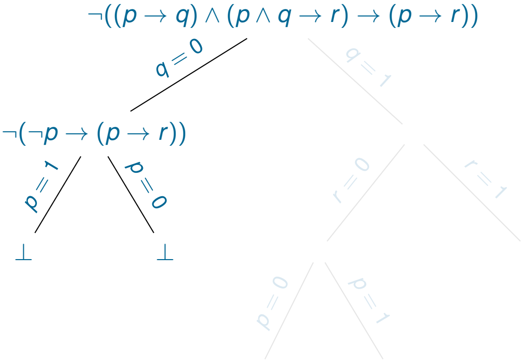
Splitting algorithm, example



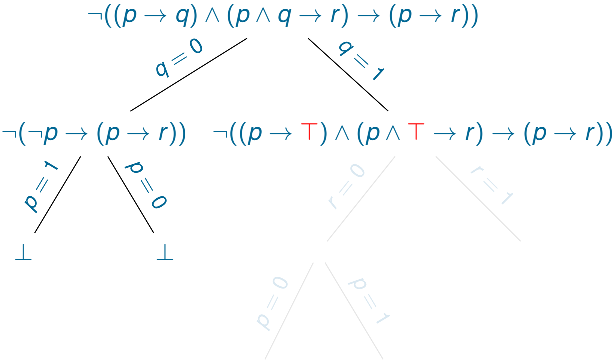
Splitting algorithm, example



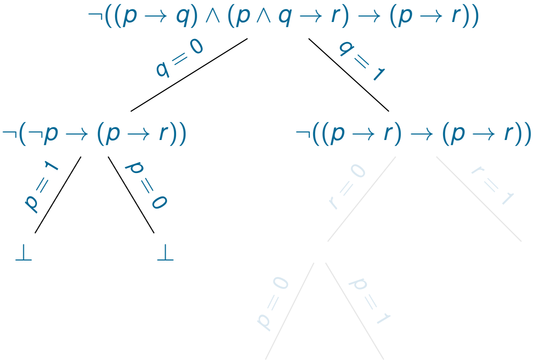
Splitting algorithm, example



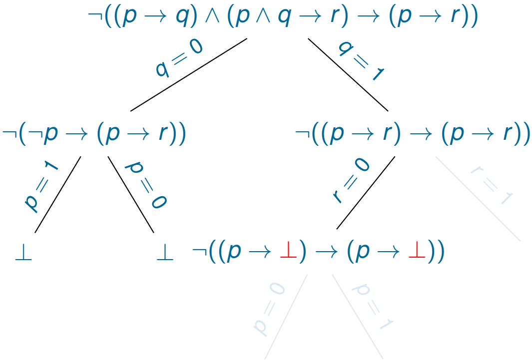
Splitting algorithm, example



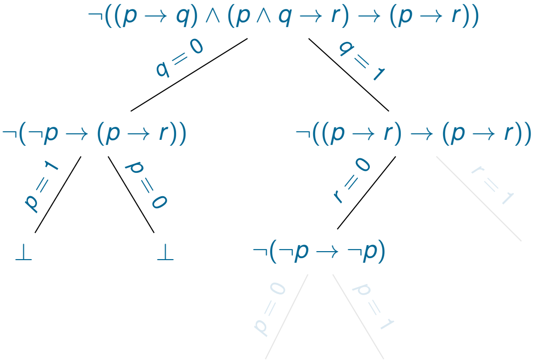
Splitting algorithm, example



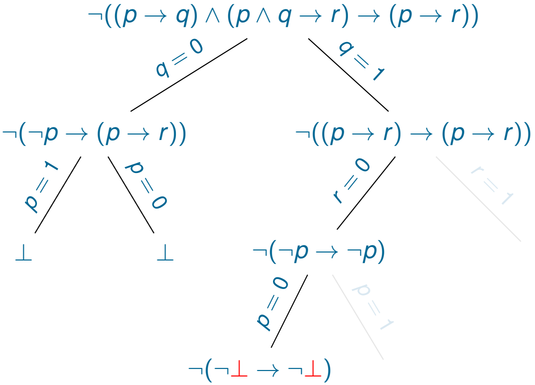
Splitting algorithm, example



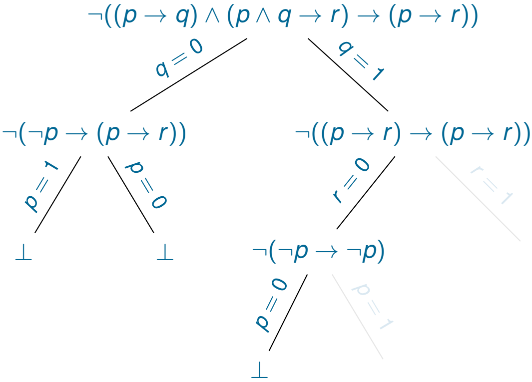
Splitting algorithm, example



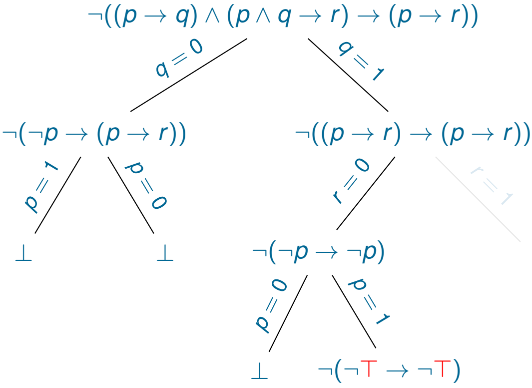
Splitting algorithm, example



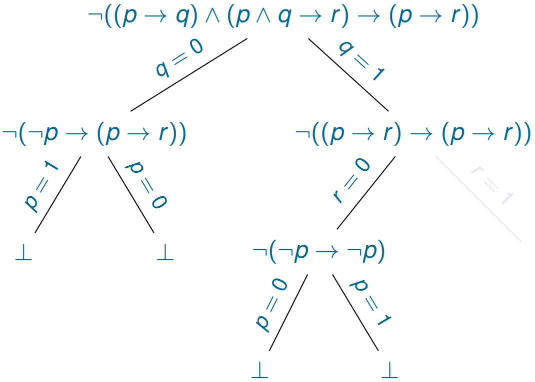
Splitting algorithm, example



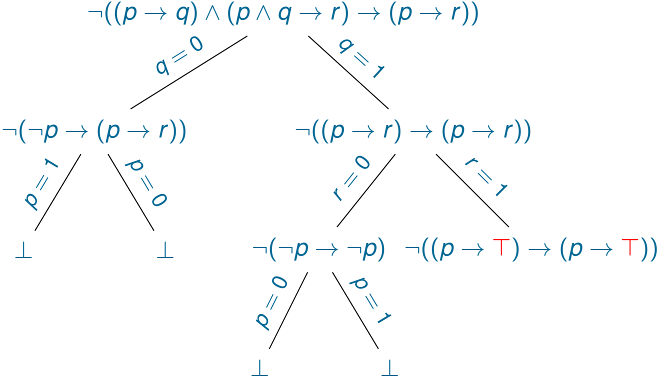
Splitting algorithm, example



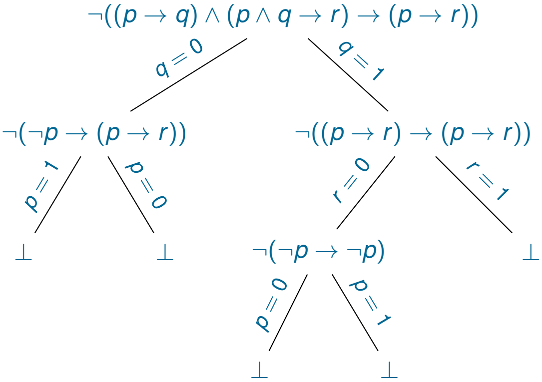
Splitting algorithm, example



Splitting algorithm, example



Splitting algorithm, example

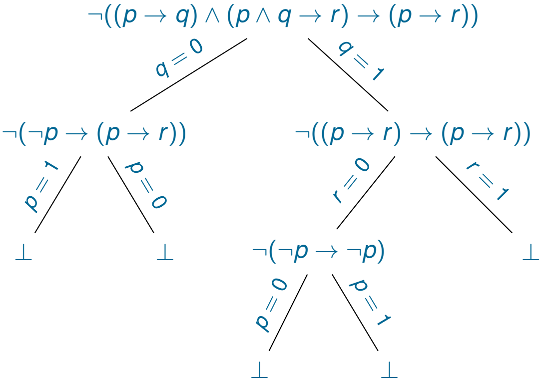


Splitting algorithm, example

$$\begin{tikzcd}[column sep=small]
 & & & & \neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r)) \arrow[dlll, dash, "q = 0"'] \arrow[drr, dash, "q = 1"] & & & \\
 & \neg(\neg p \to (p \to r)) \arrow[dl, dash, "p = 1"'] \arrow[dr, dash, "p = 0"] & & & & & \neg((p \to r) \to (p \to r)) \arrow[dll, dash, "r = 0"'] \arrow[dr, dash, "r = 1"] & \\
\bot & & \bot & & \neg(\neg p \to \neg p) \arrow[dl, dash, "p = 0"'] \arrow[dr, dash, "p = 1"] & & & \bot \\
 & & & \bot & & \bot & &
\end{tikzcd}$$

The formula is unsatisfiable.

Splitting algorithm, example



The formula is unsatisfiable.

What is going on here is very similar to using compact truth tables, but on the syntactic level.

Splitting algorithm, example 2

$$\neg((p \to q) \wedge (p \wedge q \to r) \to (\neg p \to r))$$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p = 0$

$$\neg((\bot \to q) \land (\bot \land \neg q \to r) \to (\neg \bot \to r))$$

$r = 0$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$

$p = 0$

$$\neg r$$

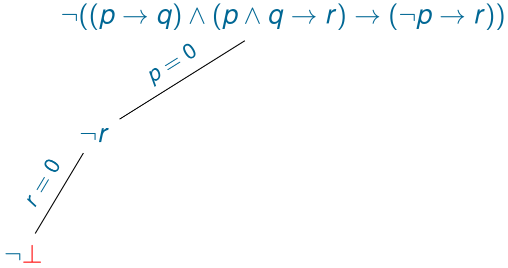
$r = 0$

The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2

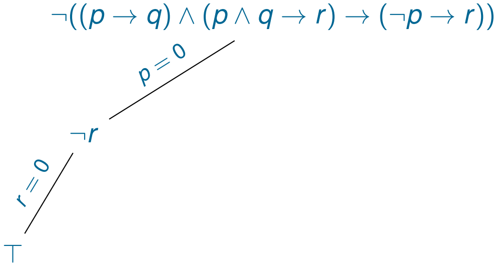


The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2

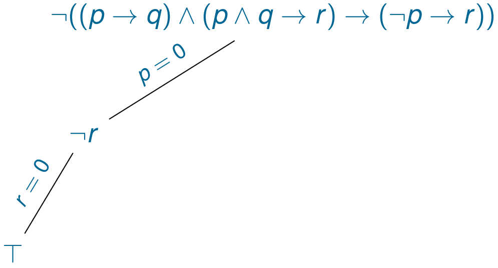


The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2

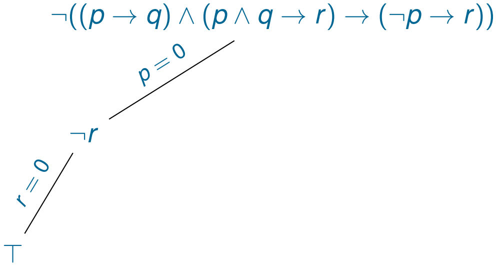


The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2

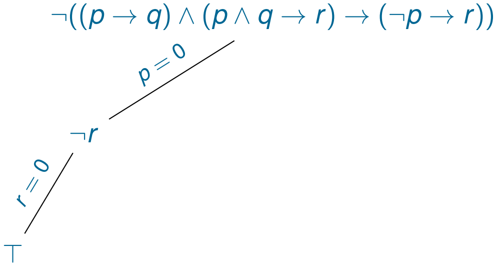


The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Splitting algorithm, example 2



The formula is satisfiable.

To find a model of this formula, we should simply collect choices made on the branch terminating at $\top$.

Any interpretation I such that $I(p) = I(r) = 0$ satisfies the formula, for example the interpretation $\{p \mapsto 0, q \mapsto 0, r \mapsto 0\}$.

Next:

- monotonicity
- position of a subformula occurrence,
- polarity of a subformula occurrence,
- monotonic replacement based on polarity,
- optimizations based on monotonic replacement: pure atom rule.

Monotonicity

- Introduce an order $<$ on truth values by defining $0 < 1$ and
- A function $f(x_1, \ldots, x_n)$ is called monotonic on its k-th argument (w.r.t. an order $<$) if $a_k \leq a'_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- A function $f(x_1, \ldots, x_n)$ is called anti-monotonic on its k-th argument if $a'_k \leq a_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- consider the behaviour of the logical connectives w.r.t. monotonicity.

Monotonicity

- Introduce an order $<$ on truth values by defining $0 < 1$ and
- A function $f(x_1, \ldots, x_n)$ is called monotonic on its k-th argument (w.r.t. an order $<$) if $a_k \leq a'_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- A function $f(x_1, \ldots, x_n)$ is called anti-monotonic on its k-th argument if $a'_k \leq a_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- consider the behaviour of the logical connectives w.r.t. monotonicity.

Monotonicity

- Introduce an order $<$ on truth values by defining $0 < 1$ and
- A function $f(x_1, \ldots, x_n)$ is called monotonic on its k-th argument (w.r.t. an order $<$) if $a_k \leq a'_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- A function $f(x_1, \ldots, x_n)$ is called anti-monotonic on its k-th argument if $a'_k \leq a_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- consider the behaviour of the logical connectives w.r.t. monotonicity.

Monotonicity

- Introduce an order $<$ on truth values by defining $0 < 1$ and
- A function $f(x_1, \ldots, x_n)$ is called monotonic on its k-th argument (w.r.t. an order $<$) if $a_k \leq a'_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- A function $f(x_1, \ldots, x_n)$ is called anti-monotonic on its k-th argument if $a'_k \leq a_k$ implies $f(a_1, \ldots, a_k, \ldots, a_n) \leq f(a_1, \ldots, a'_k, \ldots, a_n)$.
- consider the behaviour of the logical connectives w.r.t. monotonicity.

Monotonicity and logical connectives

- The connectives $\wedge$ and $\vee$ are monotonic on all of their arguments.
- The negation $\neg$ is anti-monotonic.
- The implication $\rightarrow$ is monotonic on its second argument, but anti-monotonic on its first argument.
- The equivalence $\leftrightarrow$ is neither monotonic nor anti-monotonic on either of its arguments.

Monotonicity and logical connectives

- The connectives $\wedge$ and $\vee$ are monotonic on all of their arguments.
- The negation $\neg$ is anti-monotonic.
- The implication $\rightarrow$ is monotonic on its second argument, but anti-monotonic on its first argument.
- The equivalence $\leftrightarrow$ is neither monotonic nor anti-monotonic on either of its arguments.

Monotonicity and logical connectives

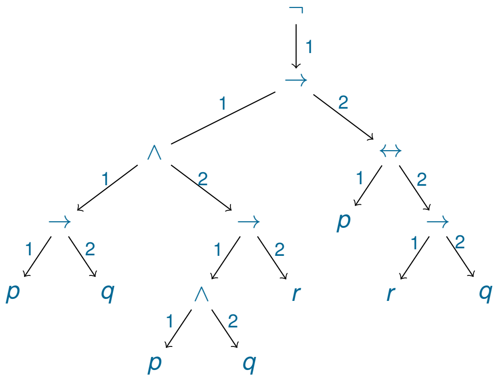
- The connectives $\wedge$ and $\vee$ are monotonic on all of their arguments.
- The negation $\neg$ is anti-monotonic.
- The implication $\rightarrow$ is monotonic on its second argument, but anti-monotonic on its first argument.
- The equivalence $\leftrightarrow$ is neither monotonic nor anti-monotonic on either of its arguments.

Monotonicity and logical connectives

- The connectives $\wedge$ and $\vee$ are monotonic on all of their arguments.
- The negation $\neg$ is anti-monotonic.
- The implication $\rightarrow$ is monotonic on its second argument, but anti-monotonic on its first argument.
- The equivalence $\leftrightarrow$ is neither monotonic nor anti-monotonic on either of its arguments.

Parse tree

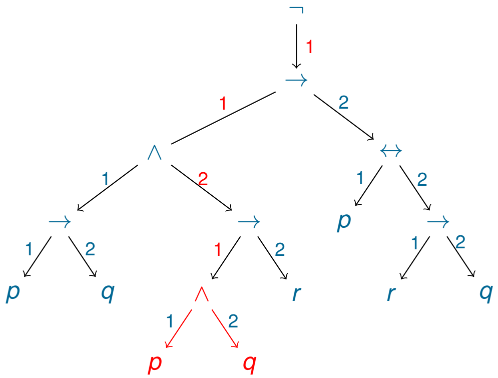
$A \stackrel{\text{def}}{=} \neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (p \leftrightarrow (r \rightarrow q)))$.



- Position in the formula: 1.1.2.1;
- Subformula at this position: $p \wedge q$; denoted $A|_{1.1.2.1} = p \wedge q$.
- Position of A is ϵ.

Parse tree

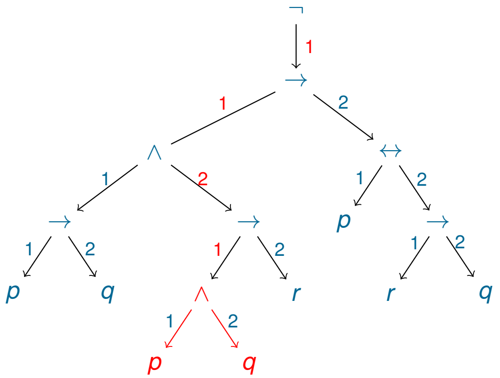
$A \stackrel{\text{def}}{=} \neg((p \to q) \wedge (p \wedge q \to r) \to (p \leftrightarrow (r \to q)))$.



- Position in the formula: $1.1.2.1$;
- Subformula at this position: $p \wedge q$; denoted $A|_{1.1.2.1} = p \wedge q$.
- Position of A is ϵ.

Parse tree

$A \stackrel{\text{def}}{=} \neg((p \to q) \wedge (p \wedge q \to r) \to (p \leftrightarrow (r \to q)))$.



- Position in the formula: $1.1.2.1$;
- Subformula at this position: $p \wedge q$; denoted $A|_{1.1.2.1} = p \wedge q$.
- Position of A is ϵ.

Positions and Subformulas

- **Position** is any sequence of positive integers $a_1, \ldots, a_n$, where $n \geq 0$, written as $a_1.a_2.\ \cdots\ .a_n$.
- **Empty position**, denoted by ϵ: when $n = 0$.
- **Position π in a formula A, subformula at a position**, denoted $A|_\pi$.

1. For every formula A, ϵ is a position in A and $A|_\epsilon \stackrel{\text{def}}{=} A$.
2. Let $A|_\pi = B$.

   2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$.

   2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$.

   2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$;

   2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$.

If $A|_\pi = B$, we also say that B **occurs in A at the position π**.

Positions and Subformulas

- Position is any sequence of positive integers $a_1, \ldots, a_n$, where $n \geq 0$, written as $a_1.a_2.\;\cdots\;.a_n$.
- Empty position, denoted by ϵ: when $n = 0$.
- Position π in a formula A, subformula at a position, denoted $A|_\pi$.

1. For every formula A, ϵ is a position in A and $A|_\epsilon \stackrel{\text{def}}{=} A$.
2. Let $A|_\pi = B$.
   - 2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$.
   - 2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$.
   - 2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$;
   - 2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$.

If $A|_\pi = B$, we also say that B occurs in A at the position π.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_{\epsilon} \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_{\pi} = B$.
   - 2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.
   - 2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_{\pi} = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.

   2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.

   2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.
   2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.

   2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.

   2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.

   2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.
   - 2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.
   - 2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.
   - 2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.
   - 2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.

   2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.

   2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.

   2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.
   - 2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.
   - 2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.
   - 2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive respectively.

Polarity

Polarity of subformula at a position. Notation: $pol(A, \pi)$.

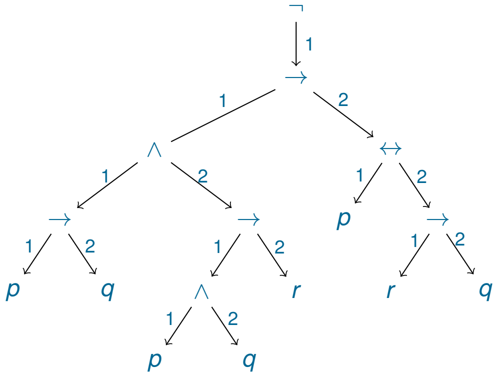
1. For every formula A, ϵ is a position in A, $A|_\epsilon \stackrel{\text{def}}{=} A$ and $pol(A, \epsilon) \stackrel{\text{def}}{=} 1$.
2. Let $A|_\pi = B$.
   2.1 If B has the form $B_1 \wedge \ldots \wedge B_n$ or $B_1 \vee \ldots \vee B_n$, then for all $i \in \{1, \ldots, n\}$ the position $\pi.i$ is a position in A, $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$, and $pol(A, \pi.i) \stackrel{\text{def}}{=} pol(A, \pi)$.
   2.2 If B has the form $\neg B_1$, then $\pi.1$ is a position in A, $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$ and $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$.
   2.3 If B has the form $B_1 \rightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and we have $A|_{\pi.1} \stackrel{\text{def}}{=} B_1$, $A|_{\pi.2} \stackrel{\text{def}}{=} B_2$, $pol(A, \pi.1) \stackrel{\text{def}}{=} -pol(A, \pi)$, $pol(A, \pi.2) \stackrel{\text{def}}{=} pol(A, \pi)$.
   2.4 If B has the form $B_1 \leftrightarrow B_2$, then $\pi.1$ and $\pi.2$ are positions in A and $A|_{\pi.i} \stackrel{\text{def}}{=} B_i$ and $pol(A, \pi.i) \stackrel{\text{def}}{=} 0$ for $i = 1, 2$.

- If $pol(A, \pi) = 1; -1; 0$ and $A|_\pi = B$, then we call the occurrence of B at the position π in A positive; negative; neutral respectively.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

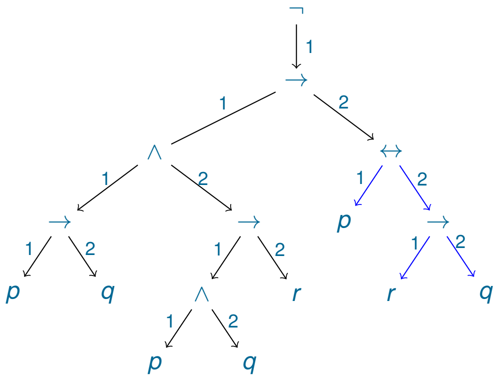


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

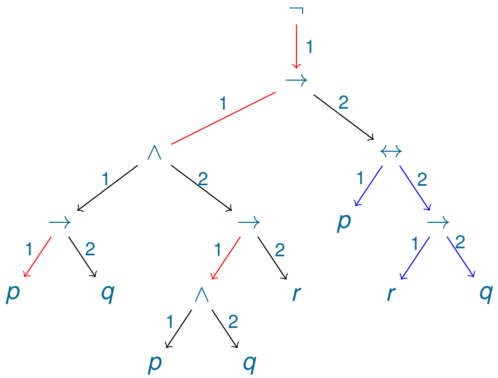


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

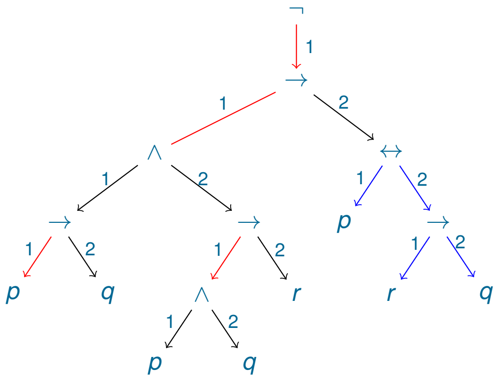


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

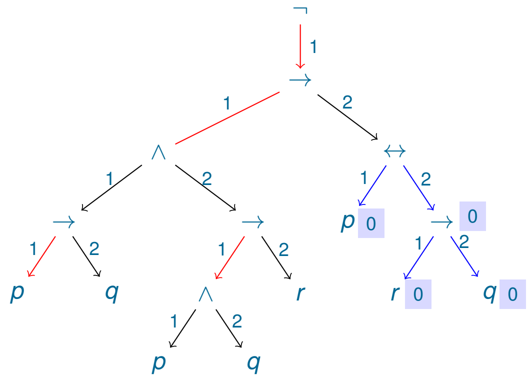


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

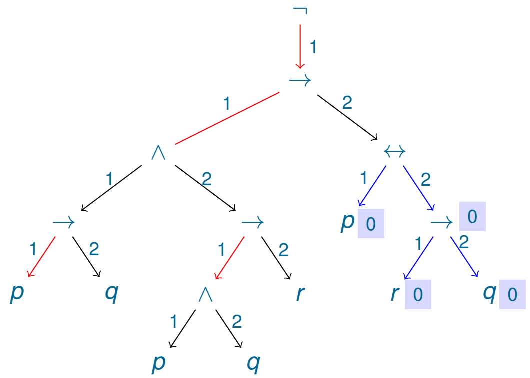


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.

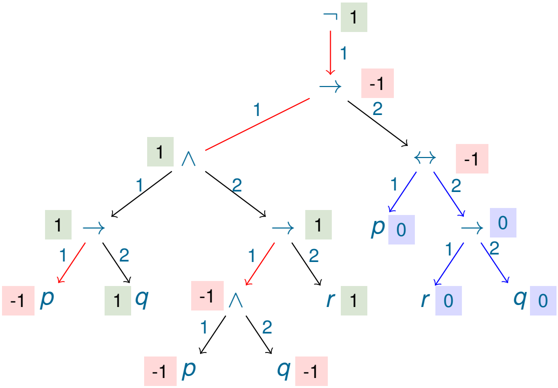


- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

The coloring algorithm for determining polarity

$\neg((p \to q) \land (p \land q \to r) \to (p \leftrightarrow (r \to q)))$.

- Color in blue all arcs below an equivalence.
- Color in red all uncolored arcs going down from a negation or left-hand side of an implication.



- If a position has at least one blue arc above it, its polarity is 0.
- Otherwise, its polarity is -1 if it has an odd number of red arcs above it and 1 if even.

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
	$(p \to q) \land (p \land q \to r) \to (p \to r)$	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
	$(p \to q) \land (p \land q \to r)$	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	1
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \wedge (p \wedge q \to r)$	1
	$p \to q$	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	1
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \wedge (p \wedge q \to r)$	1
1.1.1	$p \to q$	1
	p	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
	q	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
	$p \land q \to r$	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \land (p \land q \to r) \to (p \to r))$	1
1	$(p \to q) \land (p \land q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \land (p \land q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \land q \to r$	1
	$p \land q$	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	1
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \wedge (p \wedge q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \wedge q \to r$	1
1.1.2.1	$p \wedge q$	-1
	p	

Position and polarity, again

position	subformula	polarity
ϵ	$\neg((p \to q) \wedge (p \wedge q \to r) \to (p \to r))$	1
1	$(p \to q) \wedge (p \wedge q \to r) \to (p \to r)$	-1
1.1	$(p \to q) \wedge (p \wedge q \to r)$	1
1.1.1	$p \to q$	1
1.1.1.1	p	-1
1.1.1.2	q	1
1.1.2	$p \wedge q \to r$	1
1.1.2.1	$p \wedge q$	-1
1.1.2.1.1	p	-1
1.1.2.1.2	q	-1
1.1.2.2	r	1
1.2	$p \to r$	-1
1.2.1	p	1
1.2.2	r	-1

Monotonic replacement lemma

Notation:

- $A[B]_\pi$ denotes a formula A with the subformula B at the position π;
- $A[B']_\pi$ denotes A with the subformula at the position π replaced by B'.

Monotonic Replacement Lemma.
Let I be an interpretation,

Monotonic replacement lemma

Notation:

- $A[B]_\pi$ denotes a formula A with the subformula B at the position π;
- $A[B']_\pi$ denotes A with the subformula at the position π replaced by B'.

Monotonic Replacement Lemma.
Let I be an interpretation,

Monotonic replacement lemma

Notation:

- $A[B]_\pi$ denotes a formula A with the subformula B at the position π;
- $A[B']_\pi$ denotes A with the subformula at the position π replaced by B'.

Monotonic Replacement Lemma.
Let I be an interpretation,

Monotonic replacement lemma

Notation:

- $A[B]_\pi$ denotes a formula A with the subformula B at the position π;
- $A[B']_\pi$ denotes A with the subformula at the position π replaced by B'.

Monotonic Replacement Lemma.
Let I be an interpretation,

- If $pol(A, \pi) = 1$, then $I(B) \leq I(B')$ implies $I(A[B]_\pi) \leq I(A[B']_\pi)$

Monotonic replacement lemma

Notation:

- $A[B]_\pi$ denotes a formula A with the subformula B at the position π;
- $A[B']_\pi$ denotes A with the subformula at the position π replaced by B'.

Monotonic Replacement Lemma.
Let I be an interpretation,

- If $pol(A, \pi) = 1$, then $I(B) \leq I(B')$ implies $I(A[B]_\pi) \leq I(A[B']_\pi)$
- if $pol(A, \pi) = -1$, then $I(B') \leq I(B)$ implies $I(A[B]_\pi) \leq I(A[B']_\pi)$

Monotonic replacement lemma

Notation:

- $A[B]_\pi$ denotes a formula A with the subformula B at the position π;
- $A[B']_\pi$ denotes A with the subformula at the position π replaced by B'.

Monotonic Replacement Lemma.
Let I be an interpretation,

- If $pol(A, \pi) = 1$, then $I(B) \leq I(B')$ implies $I(A[B]_\pi) \leq I(A[B']_\pi)$
- if $pol(A, \pi) = -1$, then $I(B') \leq I(B)$ implies $I(A[B]_\pi) \leq I(A[B']_\pi)$

A positive (negative) occurrence is a sufficient syntactic condition for monotonicity (anti-monotonicity).

Monotonic replacement theorem

Lemma. For any interpretation I:
$I(A) \leq I(B)$ if and only if $I \models A \to B$.

Monotonic Replacement Theorem.

Let $B \to B'$ be valid.

- If $pol(A, \pi) = 1$, then for every interpretation I:
  $I(A[B]_\pi) \leq I(A[B']_\pi)$.

Let $B' \to B$ be valid.

- If $pol(A, \pi) = -1$, then for every interpretation I:
  $I(A[B]_\pi) \leq I(A[B']_\pi)$.

Monotonic replacement theorem

Lemma. For any interpretation I:
$I(A) \leq I(B)$ if and only if $I \models A \rightarrow B$.

Monotonic Replacement Theorem.

Let $B \rightarrow B'$ be valid.

- If $pol(A, \pi) = 1$, then for every interpretation I:
  $I(A[B]_\pi) \leq I(A[B']_\pi)$.

Let $B' \rightarrow B$ be valid.

- If $pol(A, \pi) = -1$, then for every interpretation I:
  $I(A[B]_\pi) \leq I(A[B']_\pi)$.

Monotonic replacement theorem

Lemma. For any interpretation I:
$I(A) \leq I(B)$ if and only if $I \models A \rightarrow B$.

Monotonic Replacement Theorem.

Let $B \rightarrow B'$ be valid.

- If $pol(A, \pi) = 1$, then for every interpretation I:
  $I(A[B]_\pi) \leq I(A[B']_\pi)$.

Let $B' \rightarrow B$ be valid.

- If $pol(A, \pi) = -1$, then for every interpretation I:
  $I(A[B]_\pi) \leq I(A[B']_\pi)$.

Pure Atom

Atom p is pure in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.

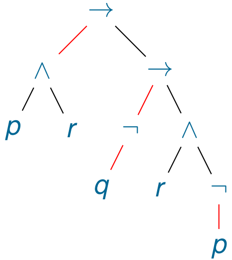
$$p \wedge r \rightarrow (\neg q \rightarrow (r \wedge \neg p))$$

- Both occurrences of p are negative, so p is pure.
- The only occurrence of q is positive, so q is pure.
- r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is pure in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.

$$p \land r \to (\neg q \to (r \land \neg p))$$

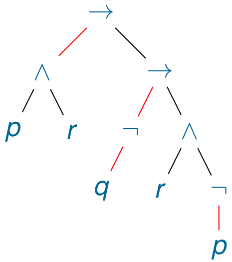


- Both occurrences of p are negative, so p is pure.
- The only occurrence of q is positive, so q is pure.
- r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is pure in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.

$$p \land r \to (\neg q \to (r \land \neg p))$$

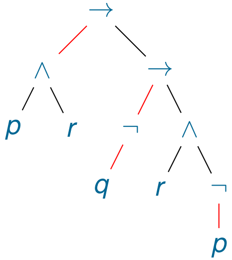


- Both occurrences of p are negative, so p is pure.
- The only occurrence of q is positive, so q is pure.
- r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is pure in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.

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

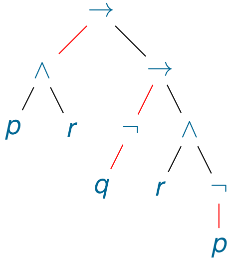


- Both occurrences of p are negative, so p is pure.
- The only occurrence of q is positive, so q is pure.
- r is not pure, since it has both negative and positive occurrences.

Pure Atom

Atom p is pure in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.

$$p \wedge r \to (\neg q \to (r \wedge \neg p))$$



- Both occurrences of p are negative, so p is pure.
- The only occurrence of q is positive, so q is pure.
- r is not pure, since it has both negative and positive occurrences.

Properties of Pure Atoms

Theorem (Pure Atom)

Let an atom p has only positive (respectively, only negative) occurrences in A. Then A is satisfiable if and only if so is $A_p^{\top}$ (respectively, $A_p^{\perp}$).

We can prove Pure Atom Theorem by applying Monotonic Replacement Theorem.
Note: $p \rightarrow \top$ and $\perp \rightarrow p$ are valid formulas.

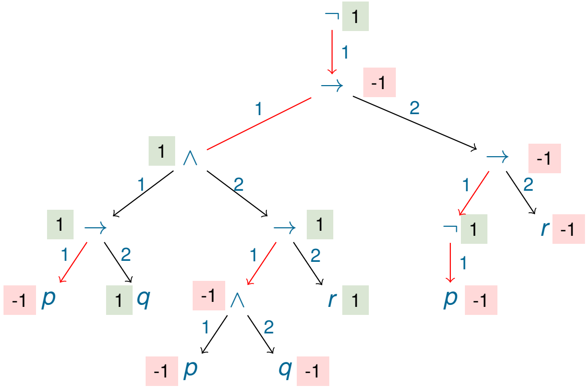
Pure atom rule, example

Consider $\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$.

All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by $\perp$.

Pure atom rule, example

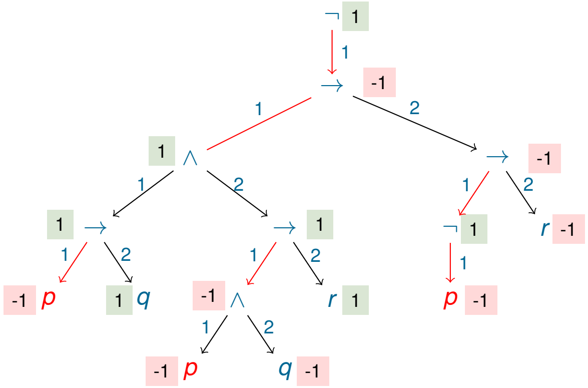
Consider $\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$.



All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by $\perp$.

Pure atom rule, example

Consider $\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$.



All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by $\perp$.

Example, continued

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r))$$
$$\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg\bot \to r))$$
$$\neg(\top \land (\bot \land q \to r) \to (\neg\bot \to r))$$
$$\neg((\bot \land q \to r) \to (\neg\bot \to r))$$
$$\neg((\bot \to r) \to (\neg\bot \to r))$$
$$\neg(\top \to (\neg\bot \to r))$$
$$\neg(\neg\bot \to r)$$
$$\neg(\top \to r)$$
$$\neg r$$
$$\neg\bot$$
$$\top$$

All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by $\bot$.

Example, continued

$$
\begin{array}{c}
\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) \qquad \Rightarrow \\
\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) \\
\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) \\
\neg((\bot \land q \to r) \to (\neg \bot \to r)) \\
\neg((\bot \to r) \to (\neg \bot \to r)) \\
\neg(\top \to (\neg \bot \to r)) \\
\neg(\neg \bot \to r) \\
\neg(\top \to r) \\
\neg r \\
\neg \bot \\
\top
\end{array}
$$

All occurrences of p are negative, so, for the purpose of checking satisfiability we can replace p by $\bot$.

Example, continued

$$
\begin{array}{cl}
\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) & \Rightarrow \\
\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) & \\
\neg((\bot \land q \to r) \to (\neg \bot \to r)) & \\
\neg((\bot \to r) \to (\neg \bot \to r)) & \\
\neg(\top \to (\neg \bot \to r)) & \\
\neg(\neg \bot \to r) & \\
\neg(\top \to r) & \\
\neg r & \\
\neg \bot & \\
\top &
\end{array}
$$

Example, continued

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) \quad \Rightarrow$$
$$\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg((\bot \land q \to r) \to (\neg \bot \to r))$$
$$\neg((\bot \to r) \to (\neg \bot \to r))$$
$$\neg(\top \to (\neg \bot \to r))$$
$$\neg(\neg \bot \to r)$$
$$\neg(\top \to r)$$
$$\neg r$$
$$\neg \bot$$
$$\top$$

Example, continued

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) \quad \Rightarrow$$
$$\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg((\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg((\bot \to r) \to (\neg \bot \to r))$$
$$\neg(\top \to (\neg \bot \to r))$$
$$\neg(\neg \bot \to r)$$
$$\neg(\top \to r)$$
$$\neg r$$
$$\neg \bot$$
$$\top$$

Example, continued

$$\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) \quad \Rightarrow$$
$$\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg((\bot \land q \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg((\bot \to r) \to (\neg \bot \to r)) \quad \Rightarrow$$
$$\neg(\top \to (\neg \bot \to r))$$
$$\neg(\neg \bot \to r)$$
$$\neg(\top \to r)$$
$$\neg r$$
$$\neg \bot$$
$$\top$$

Example, continued

$$
\begin{array}{cc}
\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) & \Rightarrow \\
\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg((\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg((\bot \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\neg \bot \to r) & \\
\neg(\top \to r) & \\
\neg r & \\
\neg \bot & \\
\top &
\end{array}
$$

Example, continued

$$
\begin{array}{cc}
\neg((p \rightarrow q) \wedge (p \wedge q \rightarrow r) \rightarrow (\neg p \rightarrow r)) & \Rightarrow \\
\neg((\bot \rightarrow q) \wedge (\bot \wedge q \rightarrow r) \rightarrow (\neg \bot \rightarrow r)) & \Rightarrow \\
\neg(\top \wedge (\bot \wedge q \rightarrow r) \rightarrow (\neg \bot \rightarrow r)) & \Rightarrow \\
\neg((\bot \wedge q \rightarrow r) \rightarrow (\neg \bot \rightarrow r)) & \Rightarrow \\
\neg((\bot \rightarrow r) \rightarrow (\neg \bot \rightarrow r)) & \Rightarrow \\
\neg(\top \rightarrow (\neg \bot \rightarrow r)) & \Rightarrow \\
\neg(\neg \bot \rightarrow r) & \Rightarrow \\
\neg(\top \rightarrow r) & \\
\neg r & \\
\neg \bot & \\
\top &
\end{array}
$$

Example, continued

$$
\begin{array}{cc}
\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) & \Rightarrow \\
\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \land (\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg((\bot \land q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg((\bot \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\neg \bot \to r) & \Rightarrow \\
\neg(\top \to r) & \Rightarrow \\
\neg r & \\
\neg \bot & \\
\top &
\end{array}
$$

All occurrences of r are negative, so, for the purpose of checking satisfiability we can replace r by $\bot$.

Example, continued

$$\begin{array}{cc}
\neg((p \to q) \land (p \land q \to r) \to (\neg p \to r)) & \Rightarrow \\
\neg((\bot \to q) \land (\bot \land q \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg(\top \land (\bot \land q \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg((\bot \land q \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg((\bot \to r) \to (\neg\bot \to r)) & \Rightarrow \\
\neg(\top \to (\neg\bot \to r)) & \Rightarrow \\
\neg(\neg\bot \to r) & \Rightarrow \\
\neg(\top \to r) & \Rightarrow \\
\neg r & \Rightarrow \\
\neg\bot & \\
\top &
\end{array}$$

All occurrences of r are negative, so, for the purpose of checking satisfiability we can replace r by $\bot$.

Example, continued

$$
\begin{array}{cc}
\neg((p \to q) \wedge (p \wedge q \to r) \to (\neg p \to r)) & \Rightarrow \\
\neg((\bot \to q) \wedge (\bot \wedge q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \wedge (\bot \wedge q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg((\bot \wedge q \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg((\bot \to r) \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\top \to (\neg \bot \to r)) & \Rightarrow \\
\neg(\neg \bot \to r) & \Rightarrow \\
\neg(\top \to r) & \Rightarrow \\
\neg r & \Rightarrow \\
\neg \bot & \Rightarrow \\
\top &
\end{array}
$$

We have shown satisfiability of this formula deterministically, using only the pure atom rule.

Summary

We have studied:

- how to formalise problems in propositonal logic,
- splitting algorithm for checking satisfiability,
- position/polarity of a subformula occurrence,
- monotonic replacement,
- pure atom rule.