Propositional Logic

Dr. Son P. Nguyen

UEL
VNU-HCMC

January 25, 2016

Table of Contents

Formal Nature of Logical Reasoning

Assumption 1: All humans are mortal.

Assumption 2: Sokrates is a human.

Conclusion: Therefore, Sokrates is mortal.

This argument is valid by pure formal reasons. Only its form is relevant:

Assumption 1: All A are B.

Assumption 2: C is A.

Conclusion: Therefore, C is B.

Examples

Assumption 1: All substitution ciphers are insecure.

Assumption 2: The Caesar cipher is a substitution cipher.

Conclusion: Therefore, the Caesar cipher is insecure.

Assumption 1: All humans have four eyes.

Assumption 2: I am a human.

Conclusion: Therefore, I have four eyes.

Propositional Logic

- Precise language:
  - Symbols $p, q, r, \ldots$ for basic propositions
  - Symbols can be connected by logical operators for “and”, “or”, “not”, “implies”, “equivalent” with precise meaning to form complex propositions
- Reasoning via algebraic manipulation of formulas
- Nowadays standard tool in automated reasoning, hardware verification, configuration problems, etc.

Why First-Order Logic ?

How to formalise this argument in propositional logic ?

Assumption 1: All humans are mortal.

Assumption 2: Sokrates is a human.

Conclusion: Therefore, Sokrates is mortal.

More natural in first-order logic:

$\forall x(\text{Human}(x) \rightarrow \text{Mortal}(x))$
Human(Sokrates)

Mortal(Sokrates)

Another example

Every print job is eventually printed always.

$$\forall \text{job}\ \forall \text{time}\Big(\text{Submitted}(\textit{job}, \textit{time}) \rightarrow \exists \text{time}'\ \text{Printed}(\textit{job}, \textit{time}')\Big)$$

Logic as a Foundation for Matehmatics

- Frege proposed logic as a foundation for mathematics.
- Invented the basics for first-order logic:
  - Constants: $\pi, \ldots$
  - Predicates: $<, >$ to assert $4 < 8$
  - Functions: $+, -$
  - Logical connectives: $\wedge$, $\vee, \neg$
  - Quantifiers: $\exists$, $\forall$

Inconsistency

- Frege's system is inconsistent.
- It allows to form the "set":

$$Y := \{X \mid X \text{ is a set with } X \notin X\}$$

- But: Y is not a set!

Is Mathematics Automatable ?

- Can we prove that mathematics (Principia Mathematica) is consistent?
- Can we develop a complete formal system for mathematics?
- Is mathematics decidable: Is there a mechanical way to determine whether a given mathematical statement is true ?
- Gdel (1931): Consistency of mathematics is not provable.
- Church, Turing (1930s): No mechanical procedure that can decide whether a statement in first-order logic is valid.

Beginning of Computer Science

Hilberts questions required to answer the following ones:

- What is a mechanical procedure ?
- What is a solvable problem?
- What is an algorithm?

Table of Contents

Arguments

If the microwave oven is on, then the door is closed.
The door is not closed.
Therefore, the microwave oven is not on.

Propositions

- Propositions: statements where it makes sense to say they are true or false
- Examples:
  - The microwave oven is on.
  - The sum of the numbers 3 and 5 equals 8.
  - All Martians like pepperoni on their pizza.
- No propositions:
  - Could you please pass me the salt?
  - Welcome!

Proposition Symbol

- We fix an infinite number of proposition symbols: $p_1, p_2, p_3, \ldots$. (think of variables for propositions)
- We also use letters $p, q, r, \ldots$ (sometimes with indices etc.) to denote proposition symbols.

Interpretations

Formulae are just syntactic objects without meaning.
They get a meaning if we interpret the proposition symbols and connectives.

Definition

An interpretation I is a function from the set of all proposition symbols to $\{0, 1\}$.

- If $I(p_i) = 1$, then p_i is called true under I.
- If $I(p_i) = 0$, then p_i is called false under I.

Truth tables

Given an interpretation I, we can compute the

truth value of a formula ϕ under I

step by step using truth tables.

Truth Table: Negation

Definition

Let I be an interpretation. Suppose we know the truth value $I(\phi)$ of ϕ already. The truth value of ϕ under I is:

$$I(\neg\phi) = \begin{cases} 1 & \text{if } I(\phi) = 0 \\ 0 & \text{otherwise} \end{cases}$$

Corresponding truth table:

ϕ	$\neg\phi$
0	1
1	0

Table of Contents

Satisfiability

Definition

A formula ϕ is satisfiable if there exists an interpretation I such that $I(\phi) = 1$.

Examples:

- Each proposition symbol p is satisfiable.
- $\neg p$ is satisfiable.
- $p \wedge \neg p$ is not satisfiable.
- $p \wedge \neg q$ is satisfiable.

A puzzle

Isaac and Albert were excitedly describing the result of the Third Annual International Science Fair Extravaganza in Sweden. There were three contestants, Louis, Rene, and Johannes.

Isaac reported that Louis won the fair, while Rene came in second. Albert, on the other hand, reported that Johannes won the fair, while Louis came in second.

In fact, neither Isaac nor Albert had given a correct report of the results of the science fair. Each of them had given one true statement and one false statement.

What was the actual placing of the three contestants?

Modelling using Propositional Logic

Proposition symbols with the following intuitive meaning:

- l_1: Louis came in 1st; l_2: Louis came in 2nd; l_3: Louis came in 3rd
- r_1: Rene came in 1st; r_2: Rene came in 2nd; r_3: Rene came in 3rd
- j_1: Joh. came in 1st; j_2: Joh. came in 2nd; j_3: Joh. came in 3rd

Translate the puzzle into a logic formula and find the possible placings (if any) ?

Checking satisfiability

Goal: Algorithm that solves the following problem.

Satisfiability Problem for Propositional Formulas (SAT)

Input: a propositional formula ϕ

Task: If ϕ is satisfiable, output “yes”, otherwise “no”

Applications of SAT

Some applications of SAT in computer science:

- Hardware design Checking equivalence of circuits, circuit minimisation, etc.
- Verification Does a system (e.g., a circuit) have a certain property?
- Planning, scheduling, and configuration
- General purpose framework for combinatorial problems 51

Basic Algorithm for SAT

Input: propositional formula ϕ

1. Construct the truth table for .
2. If the truth table contains a 1 in the column for , then output "yes".
3. Otherwise, output "no".

- Drawback: exponential running time (the truth table has 2n rows, where n is the number of proposition symbols in)
- Major open problem in computer science: Does there exist an efficient algorithm for SAT?

The tableau method

Basic Idea
Input: formula ϕ

1. Infer formulae that have to be true *simultaneously* in order to satisfy ϕ.
2. If *no inconsistent* pairs of formulae are inferred, ϕ is satisfiable.

Example: $((\neg p \vee q) \wedge \neg q) \wedge p$

Constraints

Definition

A constraint C is a finite set of propositional formulae

- It is satisfiable if there is an interpretation I with $I(\phi) = 1$ for all $\phi \in C$.
- It contains a clash if there is a formula $\phi \in C$ with $\neg\phi \in C$.

Examples

- $C_1 = \{\neg(p \wedge q) \vee r,\ p \wedge q\}$ is satisfiable
- $C_2 = \{\neg(p \wedge q) \vee r,\ p \wedge q,\ \neg r\}$ is not satisfiable
- $C_3 = \{\neg(p \wedge q) \vee r,\ p \wedge q,\ \neg(p \wedge q)\}$ contains a clash.

Completion Rules

Constraints $\mathcal{C}$ without clashes can be extended via the following completion rules:

$\wedge$-rule: Applicable to $\mathcal{C}$ if $\varphi \wedge \psi \in \mathcal{C}$ and $\{\varphi, \psi\} \not\subseteq \mathcal{C}$.
Result: $\mathcal{C} \cup \{\varphi, \psi\}$.

$\neg\neg$-rule: Applicable to $\mathcal{C}$ if $\neg\neg\varphi \in \mathcal{C}$ and $\varphi \notin \mathcal{C}$.
Result: $\mathcal{C} \cup \{\varphi\}$.

$\neg\vee$-rule: Applicable to $\mathcal{C}$ if $\neg(\varphi \vee \psi) \in \mathcal{C}$ and $\{\neg\varphi, \neg\psi\} \not\subseteq \mathcal{C}$.
Result: $\mathcal{C} \cup \{\neg\varphi, \neg\psi\}$.

$\vee$-rule: Applicable to $\mathcal{C}$ if $\varphi \vee \psi \in \mathcal{C}$, $\varphi \notin \mathcal{C}$, and $\psi \notin \mathcal{C}$.
Result: $\mathcal{C} \cup \{\varphi\}$ **or** $\mathcal{C} \cup \{\psi\}$.

$\neg\wedge$-rule: Applicable to $\mathcal{C}$ if $\neg(\varphi \wedge \psi) \in \mathcal{C}$, $\neg\varphi \notin \mathcal{C}$, and $\neg\psi \notin \mathcal{C}$.
Result: $\mathcal{C} \cup \{\neg\varphi\}$ **or** $\mathcal{C} \cup \{\neg\psi\}$.

Example 1

1. Start with:
$$\mathcal{C}_0 = \{\, \neg(\neg p \vee q) \wedge \neg r \,\}.$$
2. Applying the $\wedge$-rule to $\mathcal{C}_0$ yields:
$$\mathcal{C}_1 = \{\, \neg(\neg p \vee q) \wedge \neg r,\ \neg(\neg p \vee q),\ \neg r \,\}.$$
3. Applying the $\neg\vee$-rule to $\mathcal{C}_1$ yields:
$$\mathcal{C}_2 = \{\, \neg(\neg p \vee q) \wedge \neg r,\ \neg(\neg p \vee q),\ \neg r,\ \neg\neg p,\ \neg q \,\}.$$
4. Applying the $\neg\neg$-rule to $\mathcal{C}_2$ yields:
$$\mathcal{C}_3 = \{\, \neg(\neg p \vee q) \wedge \neg r,\ \neg(\neg p \vee q),\ \neg r,\ \neg\neg p,\ \neg q,\ p \,\}.$$

Example 1

We have obtained the constraint

$$\mathcal{C}_3 = \{ \neg(\neg p \vee q) \wedge \neg r,\ \neg(\neg p \vee q),\ \neg r,\ \neg\neg p,\ \neg q,\ p \}.$$

Note:

- No completion rule can be applied to $\mathcal{C}_3$.
- $\mathcal{C}_3$ contains no clash.
- $\mathcal{C}_3$ yields an interpretation $\mathcal{I}$ under which all $\varphi \in \mathcal{C}_3$ are true: Namely, let $\mathcal{I}(p_i) = 1$ if and only if $p_i \in \mathcal{C}_3$. Thus, $\mathcal{I}(p) = 1$ and $\mathcal{I}(q) = \mathcal{I}(r) = 0$.

Example 2

Try $\neg(\neg p \wedge q) \wedge \neg p$

Example 2

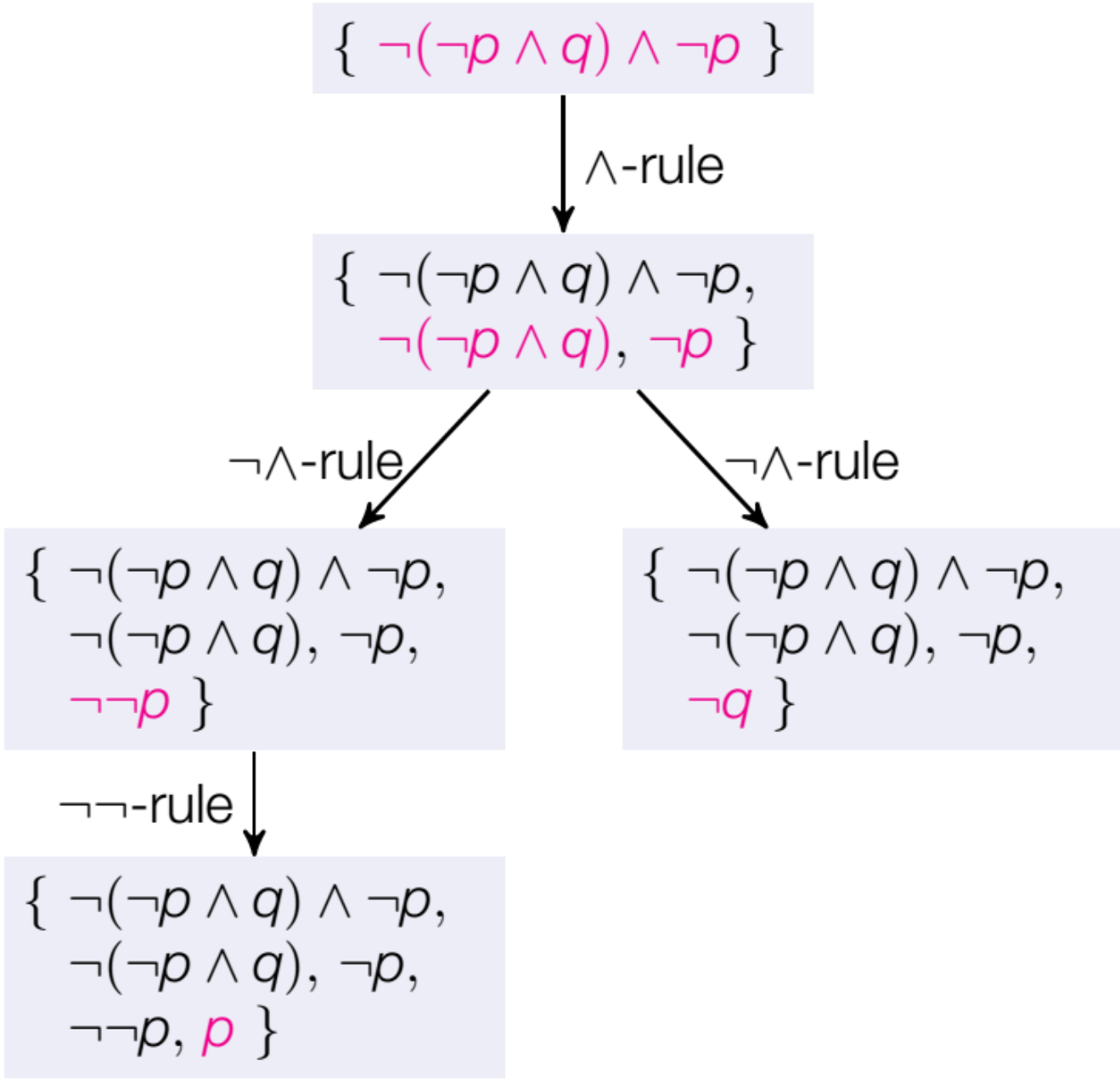


Tableau paths

Definition

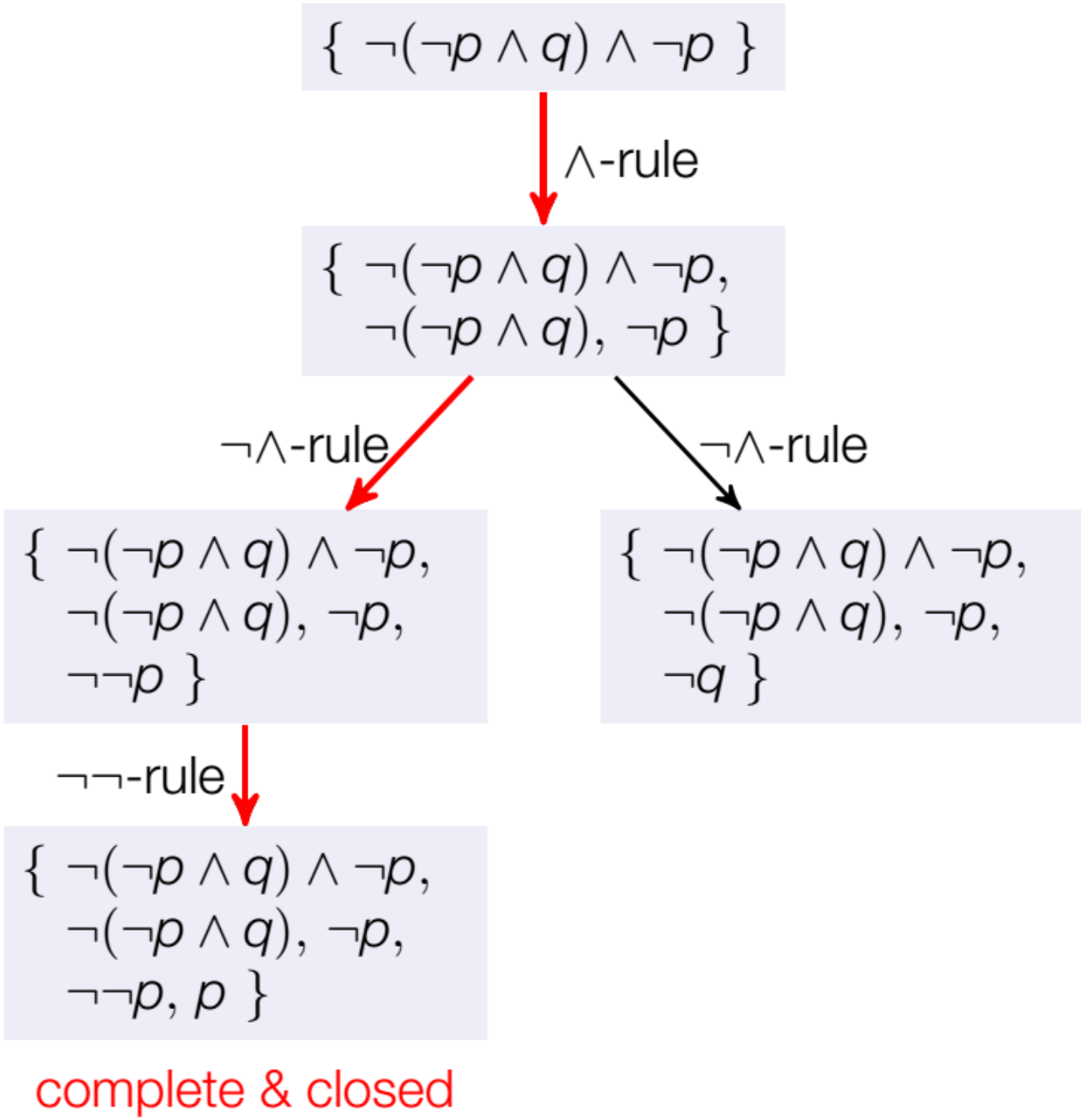
A tableau path is a sequence $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$ of constraints such that for all $i \in \{1, \ldots, n\}$:

- $\mathcal{C}_{i-1}$ does not contain a clash,
- $\mathcal{C}_i$ is the result of applying a completion rule to $\mathcal{C}_{i-1}$.

Such a path is:

- complete if no completion rule can be applied to $\mathcal{C}_n$;
- closed if $\mathcal{C}_n$ contains a clash.

Example 2


{ ¬(¬p ∧ q) ∧ ¬p }
∧-rule
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p }
¬∧-rule
¬∧-rule
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p,
¬¬p }
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p,
¬q }
¬¬-rule
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p,
¬¬p, p }
complete & closed

Example 2

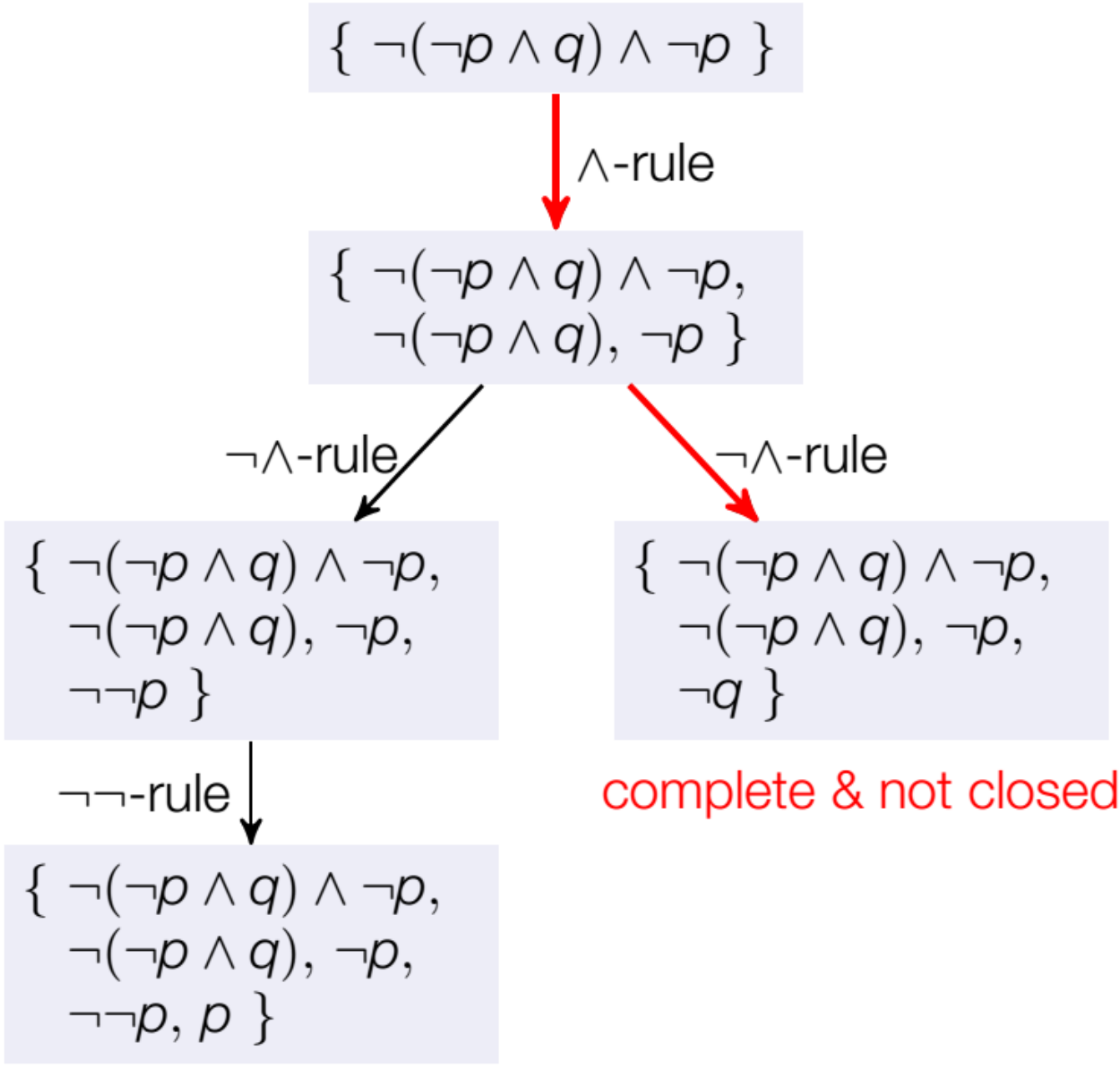

{ ¬(¬p ∧ q) ∧ ¬p }
∧-rule
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p }
¬∧-rule
¬∧-rule
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p,
¬¬p }
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p,
¬q }
¬¬-rule
complete & not closed
{ ¬(¬p ∧ q) ∧ ¬p,
¬(¬p ∧ q), ¬p,
¬¬p, p }

Tableau algorithm

Input: A propositional formula φ
Output: "satisfiable" if φ is satisfiable; otherwise "unsatisfiable"

1. Generate tableau paths starting with the constraint $\{\varphi\}$.
2. If there is a complete tableau path that is not closed, output "satisfiable".
3. If all complete tableau paths are closed, output "unsatisfiable".

Examples

3 $\phi = (\neg p \wedge q) \wedge \neg\neg r$

Examples

3. $\phi = (\neg p \wedge q) \wedge \neg\neg r$
4. $\phi = (p \wedge q) \wedge \neg(p \vee q)$

Examples

3. $\phi = (\neg p \wedge q) \wedge \neg\neg r$
4. $\phi = (p \wedge q) \wedge \neg(p \vee q)$
5. $\phi = ((p \rightarrow q) \wedge p) \wedge \neg\neg q$

Examples

3 $\phi = (\neg p \wedge q) \wedge \neg\neg r$

4 $\phi = (p \wedge q) \wedge \neg(p \vee q)$

5 $\phi = ((p \rightarrow q) \wedge p) \wedge \neg\neg q$

6 $\phi = ((\neg p \vee q) \wedge p) \wedge \neg q$

Answers

3, 5: satisfiable

4, 6: unsatisfiable

Example 3

We use the tableau algorithm to check if $\varphi = (\neg p \wedge q) \wedge \neg\neg r$ is satisfiable:

1. Start with:
   $$\mathcal{C}_0 = \{\ (\neg p \wedge q) \wedge \neg\neg r\ \}.$$
2. Apply the $\wedge$-rule:
   $$\mathcal{C}_1 = \{\ (\neg p \wedge q) \wedge \neg\neg r,\ \neg p \wedge q,\ \neg\neg r\ \}.$$
3. Apply the $\wedge$-rule again:
   $$\mathcal{C}_2 = \{\ (\neg p \wedge q) \wedge \neg\neg r,\ \neg p \wedge q,\ \neg\neg r,\ \neg p,\ q\ \}.$$
4. Apply the $\neg\neg$-rule:
   $$\mathcal{C}_3 = \{\ (\neg p \wedge q) \wedge \neg\neg r,\ \neg p \wedge q,\ \neg\neg r,\ \neg p,\ q,\ r\ \}.$$

The tableau path $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ is complete and not closed.
The algorithm outputs “satisfiable”.

Example 4

Tableau algorithm on input: $\varphi = (p \wedge q) \wedge \neg(p \vee q)$

1. Start with:
   $$\mathcal{C}_0 = \{\ (p \wedge q) \wedge \neg(p \vee q)\ \}$$
2. Apply the $\wedge$-rule:
   $$\mathcal{C}_1 = \{\ (p \wedge q) \wedge \neg(p \vee q),\ p \wedge q,\ \neg(p \vee q)\ \}$$
3. Apply the $\wedge$-rule again:
   $$\mathcal{C}_2 = \{\ (p \wedge q) \wedge \neg(p \vee q),\ p \wedge q,\ \neg(p \vee q),\ p,\ q\ \}$$
4. Apply the $\neg\vee$-rule:
   $$\mathcal{C}_3 = \{\ (p \wedge q) \wedge \neg(p \vee q),\ p \wedge q,\ \neg(p \vee q),\ p,\ q,\ \neg p,\ \neg q\ \}$$

$\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ is the only complete path. Since it is closed, the algorithm outputs “unsatisfiable”.

Example 5

We check if $\varphi = ((p \to q) \wedge p) \wedge \neg\neg q$ is satisfiable.

Recall: $p \to q$ is an abbreviation for $\neg p \vee q$.

Set $\mathcal{C}_0 = \{\varphi\}$.

1. Applying the $\wedge$-rule yields: $\mathcal{C}_1 = \mathcal{C}_0 \cup \{\ (\neg p \vee q) \wedge p,\ \neg\neg q\ \}$
2. Applying the $\wedge$-rule yields: $\mathcal{C}_2 = \mathcal{C}_1 \cup \{\ \neg p \vee q,\ p\ \}$
3. Applying the $\vee$-rule has two possible results:
   - $\mathcal{C}_3 = \mathcal{C}_2 \cup \{\ \neg p\ \}$
   - $\mathcal{C}_4 = \mathcal{C}_2 \cup \{\ q\ \}$
4. $\mathcal{C}_3$ contains a clash, so the tableau path $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ is complete & closed.
5. The tableau path $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4$ is complete & not closed, so the algorithm outputs “satisfiable”.

Example 5

Thus, on input $\varphi = ((p \to q) \wedge p) \wedge \neg\neg q$, the tableau algorithm finds a tableau path that is complete, not closed, and ends with:

$$\mathcal{C}_4 = \{\, \varphi,\ (\neg p \vee q) \wedge p,\ \neg\neg q,\ \neg p \vee q,\ p,\ q \,\}.$$

It therefore outputs "satisfiable".

Indeed, $\mathcal{C}_4$ described an interpretation $\mathcal{I}$ under which every formula $\psi \in \mathcal{C}_4$ (in particular, φ) is true:

$$\mathcal{I}(p_i) = 1 \iff p_i \in \mathcal{C}_4 \quad \text{for all integers } i \geq 1$$

(i.e., $\mathcal{I}(p) = \mathcal{I}(q) = 1$ because $p, q \in \mathcal{C}_4$).

Example 6

We check if $\varphi = ((\neg p \vee q) \wedge p) \wedge \neg q$ is satisfiable.

Set $\mathcal{C}_0 = \{\varphi\}$.

1. Applying the $\wedge$-rule yields: $\mathcal{C}_1 = \mathcal{C}_0 \cup \{\ ((\neg p \vee q) \wedge p),\ \neg q\ \}$
2. Applying the $\wedge$-rule yields: $\mathcal{C}_2 = \mathcal{C}_1 \cup \{\ \neg p \vee q,\ p\ \}$
3. Applying the $\vee$-rule has two possible results:
   - $\mathcal{C}_3 = \mathcal{C}_2 \cup \{\ \neg p\ \}$
   - $\mathcal{C}_4 = \mathcal{C}_2 \cup \{\ q\ \}$
4. $\mathcal{C}_3$ contains a clash, so the tableau path $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ is complete & closed.
5. $\mathcal{C}_4$ contains a clash, so the tableau path $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4$ is complete & closed.
6. Since there are no further tableau paths, the algorithm outputs “unsatisfiable”.

Analysis of the Tableau Algorithm

Theorem

Let φ be a propositional formula. The following properties hold:

1. *Termination: The tableau algorithm terminates on input φ (there are only finitely many tableau paths starting in $\{\varphi\}$).*
2. *Soundness: If the algorithms outputs "satisfiable", then φ is satisfiable.*
3. *Completeness: If φ is satisfiable, then the algorithm outputs "satisfiable".*