COMP0009 Logic

November 1, 2022

Me

Robin Hirsch, r.hirsch@ucl.ac.uk, Room 310, 66-72 Gower Street

Me

Robin Hirsch, r.hirsch@ucl.ac.uk, Room 310, 66-72 Gower Street
Office hour: Mondays 11am-12.

What are we going to do?

- Languages and decision problems.
- Revision: the language of all propositional formulas; the first order language $L(C, F, R)$.
- Revision: the language of all propositional tautologies. Truth Tables.
- Automated theorem proving.
- First order structures. Model Checking $S \models \phi$.
- Validities of $L(C, F, R)$ and theorem proving.
- Axiomatic Hilbert systems (propositional and predicate)
- Tableau (propositional and predicate)
- Main results, first-order logic
  - Soundness
  - Completeness
  - Compactness
  - Gödel's incompleteness theorem for arithmetic
- Modal Logic
- Temporal Logic
- Epistemic Logic
- Propositional Dynamic Logic
- Algebras of Relations

Reading

- "Logic: an introduction to elementary logic" by W Hodges, Penguin, 1977.
- Logic and Discrete Mathematics: A Concise Introduction by Willem Conradie and Valentin Goranko, and published by John Wiley, 2015)
- "A friendly introduction to mathematical logic" by C Leary, Prentice-Hall, 2000.
- "Logic for Computer Scientists" by S Reeves and M Clarke, Addison-Wesley, 1999.
- See lecture notes for more reading.

What is Science?

What is Science?

Evidence

What is Science?

Evidence + Logic.

Hilbert's Programme (1900)

Formalise mathematics.

1. Find a set of axioms for mathematics and show that the axioms do not contradict themselves (consistency)

Hilbert's Programme (1900)

Formalise mathematics.

1. Find a set of axioms for mathematics and show that the axioms do not contradict themselves (consistency)
2. Show that these axioms are complete, i.e. they prove all the true statements of mathematics

Hilbert's Programme (1900)

Formalise mathematics.

1. Find a set of axioms for mathematics and show that the axioms do not contradict themselves (consistency)
2. Show that these axioms are complete, i.e. they prove all the true statements of mathematics
3. Find an algorithm that determines whether a formula is true or not.

Logic in Computer Science

- For programme specification, e.g. Z.

Logic in Computer Science

- For programme specification, e.g. Z.
- for programming languages, e.g. PROLOG.

Logic in Computer Science

- For programme specification, e.g. Z.
- for programming languages, e.g. PROLOG.
- for programme verification

Logic in Computer Science

- For programme specification, e.g. Z.
- for programming languages, e.g. PROLOG.
- for programme verification
- for programme design

Logic in Computer Science

- For programme specification, e.g. Z.
- for programming languages, e.g. PROLOG.
- for programme verification
- for programme design
- very important in Artificial Intelligence

Logic in Computer Science

- For programme specification, e.g. Z.
- for programming languages, e.g. PROLOG.
- for programme verification
- for programme design
- very important in Artificial Intelligence
- knowledge representation

Logic in Computer Science

- For programme specification, e.g. Z.
- for programming languages, e.g. PROLOG.
- for programme verification
- for programme design
- very important in Artificial Intelligence
- knowledge representation
- databases (SQL)

Formal Logic

Three parts.

- **Syntax:** A language. Grammar. How you **write** things down.
- **Semantics:** Interpreting the language. What you have written down **means**.
- **Inference or proof system:** Deduction. Proofs. A syntactic device for proving true statements. How to **reason** with what you have.

Propositional Logic

$$prop ::= p|q|r|\ldots$$

$$fm ::= prop|\neg fm|(fm \circ fm), \text{ where } \circ \text{ is } \wedge, \vee \text{ or } \rightarrow$$

In other words, using negation, conjunction and disjunction we build formulas from propositions such as p, q, r, etc.

Literal: A proposition or its negation (e.g. p, q, $\neg r$, but not $p \wedge r$, nor $\neg\neg p$)

Main Connective

The connective with the largest scope (i.e. one that is not in the scope of any other connective), the connective we would start our evaluation with

$$((p \wedge q) \ \bigcirc\!\!\!\!\vee \ \neg(q \rightarrow r))$$

Propositional Parser

Given a string $((p \wedge q) \ \vee \ \neg(q \rightarrow r))$ your parser has to work out that the main connective ($\vee$) is the character with index 7, and that the two parts are $(p \wedge q)$ and $\neg(q \rightarrow r)$.

Propositional Logic. Semantics

A valuation v maps propositions to $\{\top, \bot\}$. v extends to a unique truth-function (also called v) satisfying

$$
\begin{aligned}
v(\neg\phi) = \top &\iff v(\phi) = \bot \\
v(\phi \wedge \psi) = \top &\iff v(\phi) = v(\psi) = \top \\
v(\phi \vee \psi) = \top &\iff v(\phi) = \top \text{ or } v(\psi) = \top \\
v(\phi \to \psi) = \top &\iff v(\psi) = \top \text{ or } v(\phi) = \bot
\end{aligned}
$$

Validity, Satisfiability, Equivalence

We say that:

- ϕ is **valid** if $v(\phi) = \top$ for all possible valuations v (i.e. it is **always** true).
- ϕ is **satisfiable** if $v(\phi) = \top$ for at least one valuation v (i.e. it is **true** at least once).
- ϕ and ψ are **logically equivalent**, written down as $\phi \equiv \psi$, if and only if for every v, $v(\phi) = v(\psi)$

Every **valid** formula is **satisfiable**, but not the other way around!

Predicate Logic. Syntax, Revision

Language $L(C, F, P)$.

- C is a set of constant symbols
- F is a set of function symbols. We may write f^n to indicate that f is an n-ary function symbol.
- P is a non-empty set of predicate symbols. We may write p^n to indicate an n-ary predicate symbol.

$$
\begin{array}{rcl}
tm & ::= & \mathrm{v} \in \mathrm{Var} \mid \mathrm{c} \in \mathrm{C} \mid \mathrm{f}^{\mathrm{n}}(\mathrm{tm}, \mathrm{tm}, \ldots, \mathrm{tm}) \ : \mathrm{f}^{\mathrm{n}} \in \mathrm{F} \\
\mathrm{atom} & ::= & \mathrm{p}^{\mathrm{n}}(\mathrm{tm}_0, \mathrm{tm}_1, \ldots, \mathrm{tm}_{\mathrm{n}-1}) \ : \mathrm{p}^{\mathrm{n}} \in \mathrm{P} \\
\mathrm{fm} & ::= & \mathrm{atom} | \neg \mathrm{fm} | (\mathrm{fm}_0 \vee \mathrm{fm}_1) | \exists \mathrm{vfm} \ : \mathrm{v} \in \mathrm{Var}
\end{array}
$$

Var is a set of variable symbols.

Predicate Logic

Signature $L(C, F, P)$, where C is a set of constants, F is a set of function symbols and P is a set of predicate symbols. Each function and predicate symbol has an arity. Var is a countable set of variables.

- Term

$$\tau ::= c\ (\in C) | v\ (\in Var) | f(\tau_0, \ldots, \tau_{n-1})\ (f \in F \text{ is n-ary})$$

e.g. $3 + (x * 2)$, where $2, 3 \in C$, $+, * \in F$ (both binary) and $x \in Var$.

Predicate Formula

- Atomic

$$R(\tau_0, \ldots, \tau_{n-1})$$

  Where τ_i is a term ($i < n$) and $R \in P$ is an n-ary predicate.
  E.g. $x + y < 2 * y - 1$, where $<$ is a binary predicate (written infix).

- Formula

$$\phi ::= Atom | \neg\phi | (\phi \vee \phi') | \exists x \phi$$

  Write $\forall x \phi$ as an abbreviation of $\neg\exists x \neg\phi$.
  e.g. $(\neg\exists x R(x) \rightarrow \forall y(S(x, y) \vee S(y, x)))$.

L-structure

(D, I) where D is any non-empty set (the domain) and I interprets constants, functions and predicates

- $I(c) \in D$ (if $c \in C$)
- $I(f) : D^n \to D$ (if $f \in F$ is n-ary)
- $I(R) \subseteq D^n$ (if $R \in P$ is n-ary).

Variable Assignment

$$A : Var \to D$$

Example L-structure $\mathbf{N}$

- Let $C = \{0, 1\}$, $F = \{+, \times\}$, $P = \{=, <\}$ (functions and predicates are binary, infix). Let $L = L(C, F, P)$.

Example L-structure $\mathbf{N}$

- Let $C = \{0, 1\}$, $F = \{+, \times\}$, $P = \{=, <\}$ (functions and predicates are binary, infix). Let $L = L(C, F, P)$.
- L-structure could be $\mathbf{N} = (\mathbb{N}, I)$ where
  - $\mathbb{N} = \{0, 1, 2, \ldots\}$ is set of natural numbers.

Example L-structure $\mathbf{N}$

- Let $C = \{0, 1\}$, $F = \{+, \times\}$, $P = \{=, <\}$ (functions and predicates are binary, infix). Let $L = L(C, F, P)$.
- L-structure could be $\mathbf{N} = (\mathbb{N}, I)$ where
  - $\mathbb{N} = \{0, 1, 2, \ldots\}$ is set of natural numbers.
  - $I(0) = 0$, $I(1) = 1$,
  - $I(+) : (m, n) \mapsto m + n$, $I(\times) : (m, n) \mapsto m \times n$,
  - $I(=) = \{(n, n) : n \in \mathbb{N}\}$, $I(<) = \{(m, n) \in \mathbb{N}^2 : m < n\}$.

Example L-structure $\mathbf{N}$

- Let $C = \{0, 1\}$, $F = \{+, \times\}$, $P = \{=, <\}$ (functions and predicates are binary, infix). Let $L = L(C, F, P)$.
- L-structure could be $\mathbf{N} = (\mathbb{N}, I)$ where
  - $\mathbb{N} = \{0, 1, 2, \ldots\}$ is set of natural numbers.
  - $I(0) = 0$, $I(1) = 1$,
  - $I(+) : (m, n) \mapsto m + n$, $I(\times) : (m, n) \mapsto m \times n$,
  - $I(=) = \{(n, n) : n \in \mathbb{N}\}$, $I(<) = \{(m, n) \in \mathbb{N}^2 : m < n\}$.
- Denote this structure as $\mathbf{N}$, the structure of ordinary arithmetic.
- Another L-structure $\mathbf{R} = (\mathbb{R}, I)$ is similar, but domain is $\mathbb{R}$, the set of all real numbers.

Another example L-structure

- $L = L(\emptyset, \emptyset, \{E\})$, where E is a binary predicate symbol.
- Any directed graph $G = (V, R)$ is an L-structure, the domain is V (set of vertices) and $I(E) = R$.

Evaluating Terms

Let $\mathcal{S} = (D, I)$ be a structure, $A : Var \to D$ be a variable assignment.

$$[c]^{\mathcal{S},A} = I(c)$$
$$[x]^{\mathcal{S},A} = A(x)$$
$$[f(\tau_0, \ldots, \tau_{n-1})]^{\mathcal{S},A} = I(f)([\tau_0]^{\mathcal{S},A}, \ldots, [\tau_{n-1}]^{\mathcal{S},A})$$

Evaluating Terms, example

In the structure $\mathbf{N}$, let $A : x \mapsto 3,\ y \mapsto 5$

$$[(x + 1) \times y]^{\mathbf{N},A} =$$

Evaluating Terms, example

In the structure $\mathbf{N}$, let $A : x \mapsto 3,\ y \mapsto 5$

$$[(x + 1) \times y]^{\mathbf{N},A} = 20$$

Formulas

$$
\begin{aligned}
\mathcal{S} \models_A R(\tau_0, \ldots, \tau_{n-1}) &\iff ([\tau_0]^{\mathcal{S},A}, \ldots, [\tau_{n-1}]^{\mathcal{S},A}) \in I(R) \\
\mathcal{S} \models_A \neg\phi &\iff \mathcal{S} \not\models_A \phi \\
\mathcal{S} \models_A (\phi \vee \phi') &\iff \mathcal{S} \models_A \phi \text{ or } \mathcal{S} \models_A \phi' \\
\mathcal{S} \models_A \exists x \phi &\iff \mathcal{S} \models_{A[x \mapsto d]} \phi \text{ for some } d \in D
\end{aligned}
$$

Example formula evaluation

- $\mathbf{N} \models_A \exists x((0 < x) \wedge \forall y(y = (x \times y)))$?

Example formula evaluation

- $\mathbf{N} \models_A \exists x((0 < x) \wedge \forall y(y = (x \times y)))$?
- $\mathbf{N} \models_{A[x \mapsto d]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (some $d \in \mathbb{N}$)

Example formula evaluation

- $\mathbf{N} \models_A \exists x((0 < x) \wedge \forall y(y = (x \times y)))$?
- $\mathbf{N} \models_{A[x \mapsto d]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (some $d \in \mathbb{N}$)
- $\mathbf{N} \models_{A[x \mapsto 1]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (take $d = 1$)

Example formula evaluation

- $\mathbf{N} \models_A \exists x((0 < x) \wedge \forall y(y = (x \times y)))$?
- $\mathbf{N} \models_{A[x \mapsto d]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (some $d \in \mathbb{N}$)
- $\mathbf{N} \models_{A[x \mapsto 1]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (take $d = 1$)
- $\mathbf{N} \models ((0 < 1) \wedge \forall y(y = 1 \times y))$

Example formula evaluation

- $\mathbf{N} \models_A \exists x((0 < x) \wedge \forall y(y = (x \times y)))$?
- $\mathbf{N} \models_{A[x \mapsto d]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (some $d \in \mathbb{N}$)
- $\mathbf{N} \models_{A[x \mapsto 1]} ((0 < x) \wedge \forall y(y = (x \times y)))$, (take $d = 1$)
- $\mathbf{N} \models ((0 < 1) \wedge \forall y(y = 1 \times y))$
- True

Validity

Let $\mathcal{S} = (D, I)$ be an L-structure, ϕ a formula.

Statement	Def.	Written
ϕ is valid in $\mathcal{S}$	for all $A : Var \to D$ we have $\mathcal{S} \models_A \phi$	$\mathcal{S} \models \phi$
ϕ is valid	for all L-structures $\mathcal{S}$ we have $\mathcal{S} \models \phi$	$\models \phi$

Similarly, ϕ is satisfiable in $\mathcal{S}$ if there is some $A : Var \to D$ such that $\mathcal{S} \models_A \phi$, and ϕ is simply satisfiable if it is satisfiable in some structure.
ϕ is not valid if and only if $\neg\phi$ is satisfiable.

Mentimeter Question

$$\mathbf{N}, A \models \forall x \exists y((x < y) \wedge \forall w \forall v((y = w \times v) \rightarrow ((y = w) \vee (y = v))))?$$

Problem

Let E be a binary relation denoting the edges of a graph, let $=$ denote equality. Write down a first order formula $\phi(x, y)$ with two free variables x, y, meaning

- There is a path of length 1 from node x to node y,
- There is a path of length 2 from x to y
- There is a path of length 3 from x to y
- There is a path of length k from x to y, where $k \geq 1$ is fixed.

Evaluating first-order formulas, examples

Which are true?

- $\mathbf{N} \models \forall x \exists y (x < y)$?

Evaluating first-order formulas, examples

Which are true?

- $\mathbf{N} \models \forall x \exists y (x < y)$? Yes
- $\mathbf{N} \models \exists x \forall y (x < y)$?

Evaluating first-order formulas, examples

Which are true?

- $\mathbf{N} \models \forall x \exists y (x < y)$? Yes
- $\mathbf{N} \models \exists x \forall y (x < y)$? No, even $x \mapsto 0$ does not work
- $\mathbf{N} \models \exists x \forall y ((x < y) \vee (x = y))$?

Evaluating first-order formulas, examples

Which are true?

- $\mathbf{N} \models \forall x \exists y (x < y)$? Yes
- $\mathbf{N} \models \exists x \forall y (x < y)$? No, even $x \mapsto 0$ does not work
- $\mathbf{N} \models \exists x \forall y ((x < y) \vee (x = y))$? Yes, $x \mapsto 0$.
- $\mathbf{N} \models \exists x \forall y ((y < x) \vee (y = x))$?

Evaluating first-order formulas, examples

Which are true?

- $\mathbf{N} \models \forall x \exists y (x < y)$? Yes
- $\mathbf{N} \models \exists x \forall y (x < y)$? No, even $x \mapsto 0$ does not work
- $\mathbf{N} \models \exists x \forall y ((x < y) \vee (x = y))$? Yes, $x \mapsto 0$.
- $\mathbf{N} \models \exists x \forall y ((y < x) \vee (y = x))$? No, there is no greatest natural number.
- $\mathbf{N} \models \forall x \forall y ((x < y) \rightarrow \exists z ((x < z) \wedge (z < y)))$

Evaluating first-order formulas, examples

Which are true?

- $\mathbf{N} \models \forall x \exists y (x < y)$? Yes
- $\mathbf{N} \models \exists x \forall y (x < y)$? No, even $x \mapsto 0$ does not work
- $\mathbf{N} \models \exists x \forall y ((x < y) \vee (x = y))$? Yes, $x \mapsto 0$.
- $\mathbf{N} \models \exists x \forall y ((y < x) \vee (y = x))$? No, there is no greatest natural number.
- $\mathbf{N} \models \forall x \forall y ((x < y) \rightarrow \exists z ((x < z) \wedge (z < y)))$No, e.g. $x \mapsto 3,\ y \mapsto 4$, $\nexists z$...

Propositional Proof Systems

Proof system: A system for determining the validity of formulas.

Propositional Proof Systems

Proof system: A system for determining the validity of formulas.

Obvious system: Write down the truth table for ϕ and check that all rows give value $\top$.

Propositional Proof Systems

Problem: Takes a long time (exponential time)

Propositional Proof Systems

Problem: Takes a long time (exponential time)

$$\phi : p_1 \vee p_2 \vee \ldots \vee p_{49} \vee p_{50}$$

Propositional Proof Systems

Problem: Takes a long time (exponential time)

$$\phi : p_1 \vee p_2 \vee \ldots \vee p_{49} \vee p_{50}$$

p_1	p_2	$\ldots$	p_{49}	p_{50}	ϕ
$\top$	$\top$	$\ldots$	$\top$	$\top$	$\top$
$\top$	$\top$	$\ldots$	$\top$	$\bot$	$\top$
		$\ldots$			

Propositional Proof Systems

Problem: Takes a long time (exponential time)

$\phi : p_1 \vee p_2 \vee \ldots \vee p_{49} \vee p_{50}$

p_1	p_2	$\ldots$	p_{49}	p_{50}	ϕ
$\top$	$\top$	$\ldots$	$\top$	$\top$	$\top$
$\top$	$\top$	$\ldots$	$\top$	$\bot$	$\top$
		$\ldots$			
$\bot$	$\bot$	$\ldots$	$\bot$	$\bot$	$\bot$

Only one out of 1,125,899,906,842,624 valuations makes ϕ false

Propositional Proof Systems

Proof system: A system for determining the validity of formulas.

~~Obvious system: Write down the truth table for ϕ and check that all rows give value $\top$.~~

Propositional Proof Systems

Proof system: A system for determining the validity of formulas.

~~Obvious system: Write down the truth table for ϕ and check that all rows give value $\top$.~~

Another way: Manipulate and analyze the syntax of the formula in order to find out whether there is something that can falsify it.

Propositional Proof Systems

$\models \phi$ means ϕ is valid

$\vdash \phi$ means there is a proof of ϕ

Propositional Proof Systems

$\models \phi$ means ϕ is valid
$\vdash \phi$ means there is a proof of ϕ

- Soundness: the system can prove only valid things:

$$\vdash \phi \;\; \Rightarrow \;\; \models \phi$$

Propositional Proof Systems

$\models \phi$ means ϕ is valid
$\vdash \phi$ means there is a proof of ϕ

- Soundness: the system can prove only valid things:

$$\vdash \phi \;\Rightarrow\; \models \phi$$

- Completeness: if something is valid that the system can prove it:

$$\models \phi \;\Rightarrow\; \vdash \phi$$

Axiomatic Proof Systems

Fix a propositional language with only $\rightarrow$ and $\neg$, and no double negations $\neg\neg$.

Axiom Schema

I $(p \rightarrow (q \rightarrow p))$

II $((p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r)))$

III $((\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p))$.

An axiom may be obtained by substituting any formulas in place of p, q, r above.

Inference Rule

Modus Ponens.

$$\frac{\phi \qquad (\phi \to \psi)}{\psi}$$

If you have proved ϕ and you have proved $(\phi \to \psi)$ then you may deduce ψ.

Proofs

A proof is a sequence of formulas

$$\phi_0, \phi_1, \phi_2, \ldots, \phi_n$$

such that for $i \leq n$, ϕ_i is either an instance of an axiom or it is obtained by modus ponens from ϕ_j, ϕ_k (some $j, k < i$), e.g., if $\phi_k = (\phi_j \to \phi_i)$.
If such a proof exists, ϕ_n is called a theorem and we may write

$$\vdash \phi_n$$

Proof Example. $\vdash (p \to p)$

1. $((p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p)))$
   (Ax. II, replace p, q, r by $p, (p \to p)$ and p).

Proof Example. $\vdash (p \to p)$

1. $((p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p)))$
   (Ax. II, replace p, q, r by $p, (p \to p)$ and p).
2. $(p \to ((p \to p) \to p))$ (Ax. I, replace p, q by $p, (p \to p)$).

Proof Example. $\vdash (p \to p)$

1. $((p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p)))$
   (Ax. II, replace p, q, r by $p, (p \to p)$ and p).
2. $(p \to ((p \to p) \to p))$ (Ax. I, replace p, q by $p, (p \to p)$).
3. $((p \to (p \to p)) \to (p \to p))$ (M.P., 1, 2).

Proof Example. $\vdash (p \to p)$

1. $((p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p)))$
   (Ax. II, replace p, q, r by $p, (p \to p)$ and p).
2. $(p \to ((p \to p) \to p))$ (Ax. I, replace p, q by $p, (p \to p)$).
3. $((p \to (p \to p)) \to (p \to p))$ (M.P., 1, 2).
4. $(p \to (p \to p))$ (Ax. I).

Proof Example. $\vdash (p \to p)$

1. $((p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p)))$ (Ax. II, replace p, q, r by $p, (p \to p)$ and p).
2. $(p \to ((p \to p) \to p))$ (Ax. I, replace p, q by $p, (p \to p)$).
3. $((p \to (p \to p)) \to (p \to p))$ (M.P., 1, 2).
4. $(p \to (p \to p))$ (Ax. I).
5. $(p \to p)$ (M.P., 3, 4).

Proofs with other connectives

To include double negations, add axioms

IV $(p \to \neg\neg p)$ and $(\neg\neg p \to p)$.

For disjunction and conjunctions, add axioms

V $((p \vee q) \leftrightarrow (\neg p \to q))$,

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

Proofs with assumptions

Write

$$\Gamma \vdash \phi$$

if there is a proof of ϕ using assumptions from Γ. I.e. there is a sequence

$$\phi_0, \phi_1, \ldots, \phi_n$$

where $\phi = \phi_n$ and for each $i \leq n$, ϕ_i is either

- an instance of an axiom schema,
- an assumption, $\phi_i \in \Gamma$, or
- obtained from ϕ_j, ϕ_k (some $j, k < i$) by modus ponens.

Proof example 2: $\{\bot\} \vdash p$

or, rather $\{\neg(q \to q)\} \vdash p$.

5 $(q \to q)$ (previous example)

6 $((q \to q) \to \neg\neg(q \to q))$ (Ax. IV)

7 $\neg\neg(q \to q)$ (M.P. 5,6)

8 $(\neg\neg(q \to q) \to (\neg p \to \neg\neg(q \to q)))$ (Ax. I)

9 $(\neg p \to \neg\neg(q \to q))$ (M.P. 7,8)

10 $(\neg p \to \neg\neg(q \to q)) \to (\neg(q \to q) \to p)$ (Ax. III)

11 $(\neg(q \to q) \to p)$ (M.P. 9, 10)

12 $\neg(q \to q)$ (assumption)

13 p (M.P. 11,12).

Proof from assumptions example: $\{p\} \vdash (q \to p)$

Proof from assumptions example: $\{p\} \vdash (q \to p)$

- $(p \to (q \to p))$ (Ax I)

Proof from assumptions example: $\{p\} \vdash (q \to p)$

- $(p \to (q \to p))$ (Ax I)
- p (assumption)

Proof from assumptions example: $\{p\} \vdash (q \to p)$

- $(p \to (q \to p))$ (Ax I)
- p (assumption)
- $(q \to p)$ (MP)

Soundness and Completeness

Soundness Check all axioms are valid. Check that if ϕ is valid and $(\phi \to \psi)$ is valid then ψ is valid. Deduce (by induction on length of proof) that all provable formulas are valid,

$$\vdash \phi \quad \Rightarrow \quad \models \phi$$

Soundness and Completeness

Soundness Check all axioms are valid. Check that if ϕ is valid and $(\phi \to \psi)$ is valid then ψ is valid. Deduce (by induction on length of proof) that all provable formulas are valid,

$$\vdash \phi \quad \Rightarrow \quad \models \phi$$

Completeness

$$\models \phi \quad \Rightarrow \vdash \phi$$

A bit harder.

Soundness and Completeness

Soundness Check all axioms are valid. Check that if ϕ is valid and $(\phi \to \psi)$ is valid then ψ is valid. Deduce (by induction on length of proof) that all provable formulas are valid,

$$\vdash \phi \quad \Rightarrow \quad \models \phi$$

Completeness

$$\models \phi \quad \Rightarrow \vdash \phi$$

A bit harder.

Termination Not necessarily.

Propositional Parser Implementation

- Alphabet:$\Sigma = \{p, q, r, -, v, \hat{}, >, (,)\}$. AND is 'shift 6'.
- Input: a string $s \in \Sigma^*$ of characters in Σ.
- Output yes if s is a propositional formula, no otherwise.
- If yes, is s (a) a proposition, (b) a negated formula, or (c) a binary formula?
- If (b), a negation of what formula? If (c), what is the main binary connective, what are the two subformulas?

Mentimeter Question

Go to www.menti.com

Proof by Contradiction

- To prove a formula ϕ,

Proof by Contradiction

- To prove a formula ϕ, first
- Assume $\neg\phi$,

Proof by Contradiction

- To prove a formula ϕ, first
- Assume $\neg\phi$, then
- Deduce a contradiction $q \wedge \neg q$.

Proof by Contradiction

- To prove a formula ϕ, first
- Assume $\neg\phi$, then
- Deduce a contradiction $q \wedge \neg q$. Hence,
- Conclude ϕ

Propositional Tableaux

- Given the formula ϕ, a tableau can tell us whether it is **satisfiable or not**

Propositional Tableaux

- Given the formula ϕ, a tableau can tell us whether it is **satisfiable or not**
- It does so by **decomposing** the formulae according to certain rules and growing a labelled tree to the point that only literals are left

Propositional Tableaux

- Given the formula ϕ, a tableau can tell us whether it is **satisfiable or not**
- It does so by **decomposing** the formulae according to certain rules and growing a labelled tree to the point that only literals are left
- A branch is closed if it has one node labelled by a proposition and another node labelled by its negation.

Propositional Tableaux

- Given the formula ϕ, a tableau can tell us whether it is **satisfiable or not**
- It does so by **decomposing** the formulae according to certain rules and growing a labelled tree to the point that only literals are left
- A branch is closed if it has one node labelled by a proposition and another node labelled by its negation.
- A tableau is closed if every branch is closed.

Propositional Tableaux

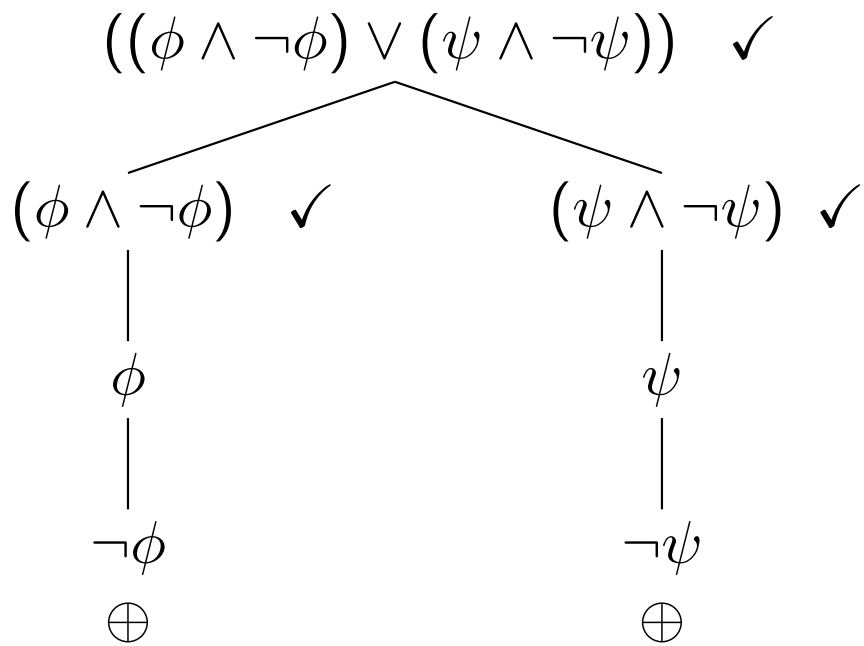
- Given the formula ϕ, a tableau can tell us whether it is **satisfiable or not**
- It does so by **decomposing** the formulae according to certain rules and growing a labelled tree to the point that only literals are left
- A branch is closed if it has one node labelled by a proposition and another node labelled by its negation.
- A tableau is closed if every branch is closed.
- If the tableau closes then ϕ is **unsatisfiable**

Propositional Tableaux

- Given the formula ϕ, a tableau can tell us whether it is **satisfiable or not**
- It does so by **decomposing** the formulae according to certain rules and growing a labelled tree to the point that only literals are left
- A branch is closed if it has one node labelled by a proposition and another node labelled by its negation.
- A tableau is closed if every branch is closed.
- If the tableau closes then ϕ is **unsatisfiable**
- If the tableau never closes then ϕ is **satisfiable**

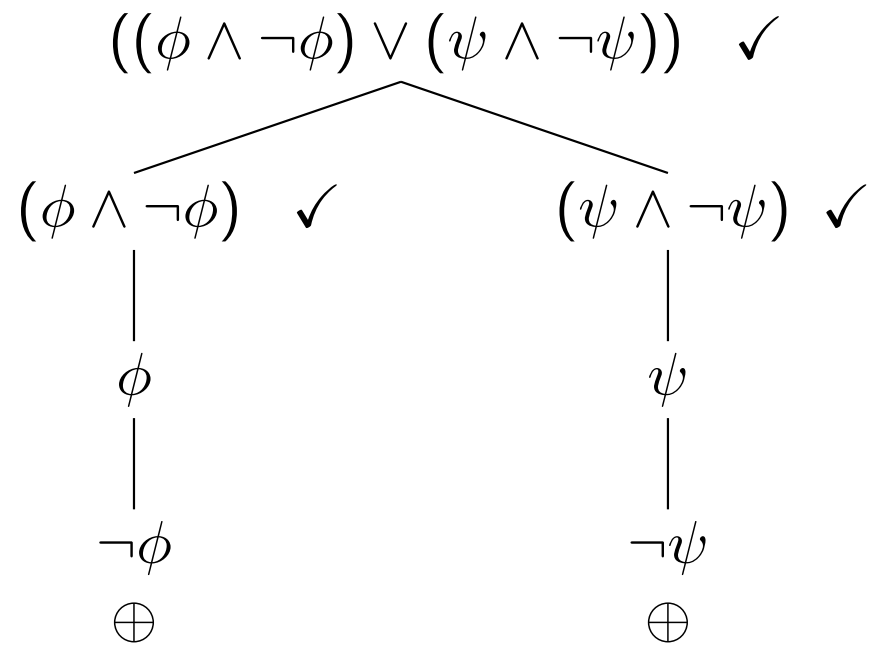
The idea

To test whether $((\phi \wedge \neg\phi) \vee (\psi \wedge \neg\psi))$ is satisfiable



The idea

To test whether $((\phi \wedge \neg\phi) \vee (\psi \wedge \neg\psi))$ is satisfiable



Both branches close $\Rightarrow$ not satisfiable.

Tableaux

- Formally, a tableau T is a type of binary tree where every node is labelled by a formula.

Tableaux

- Formally, a tableau T is a type of binary tree where every node is labelled by a formula.
- At first, T has only one node, containing the formula ϕ whose satisfiability we are investigating.

Tableaux

- Formally, a tableau T is a type of binary tree where every node is labelled by a formula.
- At first, T has only one node, containing the formula ϕ whose satisfiability we are investigating.
- Every formula in the tableau, except literals, gets "expanded" and ticked.
- If a branch of T contains p and $\neg p$ then it is **closed**. Otherwise it is **open**.
- If every branch of T is closed then T is **closed**. Otherwise it is **open**.

Tableau Expansion Rules

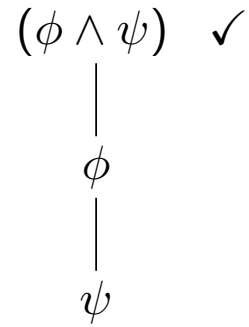
α formulas

$$(\phi \wedge \psi)$$

Observe that $(\phi \wedge \psi)$ is true if and only if ϕ is true and ψ is true.

Tableau Expansion Rules

α formulas



Observe that $(\phi \wedge \psi)$ is true if and only if ϕ is true and ψ is true.

Means that nodes corresponding to ϕ and ψ are added at every leaf of T below the current node $(\phi \wedge \psi)$.

Tableau Expansion Rules

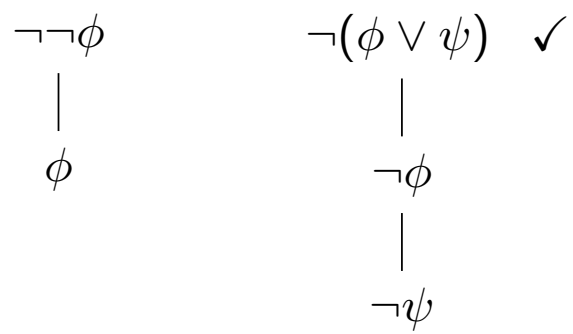
α formulas

Three other kinds of α formulas.

Tableau Expansion Rules

α formulas

Three other kinds of α formulas.

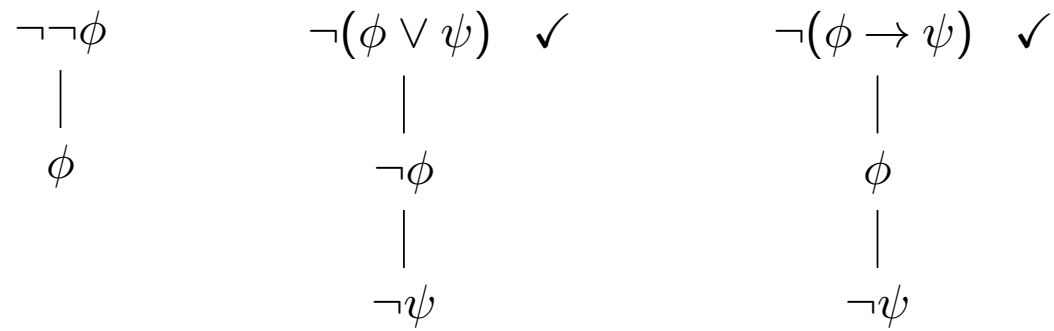


Observe that $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$

Tableau Expansion Rules

α formulas

Three other kinds of α formulas.



Observe that $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$

Observe that $\neg(\phi \rightarrow \psi) \equiv \neg(\neg\phi \vee \psi) \equiv \phi \wedge \neg\psi$

Tableau Expansion Rules

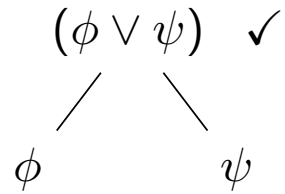
β formulas

$$(\phi \vee \psi)$$

For this case, at each leaf of the tableau below the current node, we create two new leaves, one labelled ϕ the other labelled ψ.

Tableau Expansion Rules

β formulas

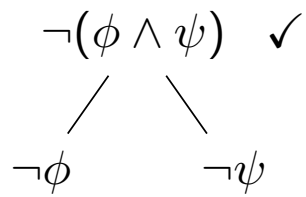


For this case, at each leaf of the tableau below the current node, we create two new leaves, one labelled ϕ the other labelled ψ.
Observe that $(\phi \vee \psi)$ is true if and only if ϕ is true or ψ is true.

Tableau Expansion Rules

β formulas

There are two other kinds of β formulas.



Observe that $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$

Tableau Expansion Rules

β formulas

There are two other kinds of β formulas.

Observe that $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$

Observe that $(\phi \rightarrow \psi) \equiv \neg\phi \vee \psi$

Tableaux

- A tableau is **complete** if every node is either ticked (already expanded) or a literal.
- If ϕ is at the root of a **complete open** tableau, then ϕ is **satisfiable**.

Propositional Tableaux

Remember: In a proof system we are interested in validity, not just satisfiability. Fortunately, we know that:

$$\phi \text{ is satisfiable} \iff \neg\phi \text{ is not valid}$$
$$\phi \text{ is valid} \iff \neg\phi \text{ is not satisfiable}$$

Propositional Tableaux

Remember: In a proof system we are interested in validity, not just satisfiability. Fortunately, we know that:

$$\phi \text{ is satisfiable} \iff \neg\phi \text{ is not valid}$$
$$\phi \text{ is valid} \iff \neg\phi \text{ is not satisfiable}$$

Solution: In order to test whether a formula ϕ is valid we see whether the tableau for $\neg\phi$ closes

Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?

$$(1)\ \neg(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$$

Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?

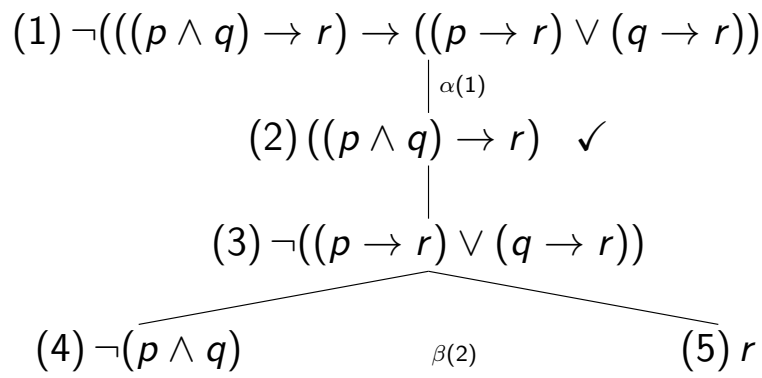
$(1)\ \neg(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ ✓

$\alpha(1)$

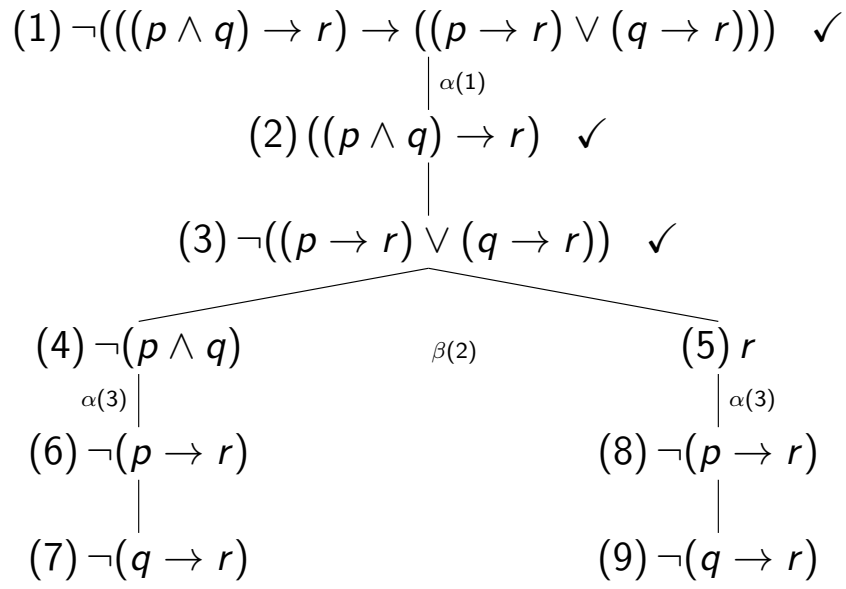
$(2)\ ((p \wedge q) \rightarrow r)$

$(3)\ \neg((p \rightarrow r) \vee (q \rightarrow r))$

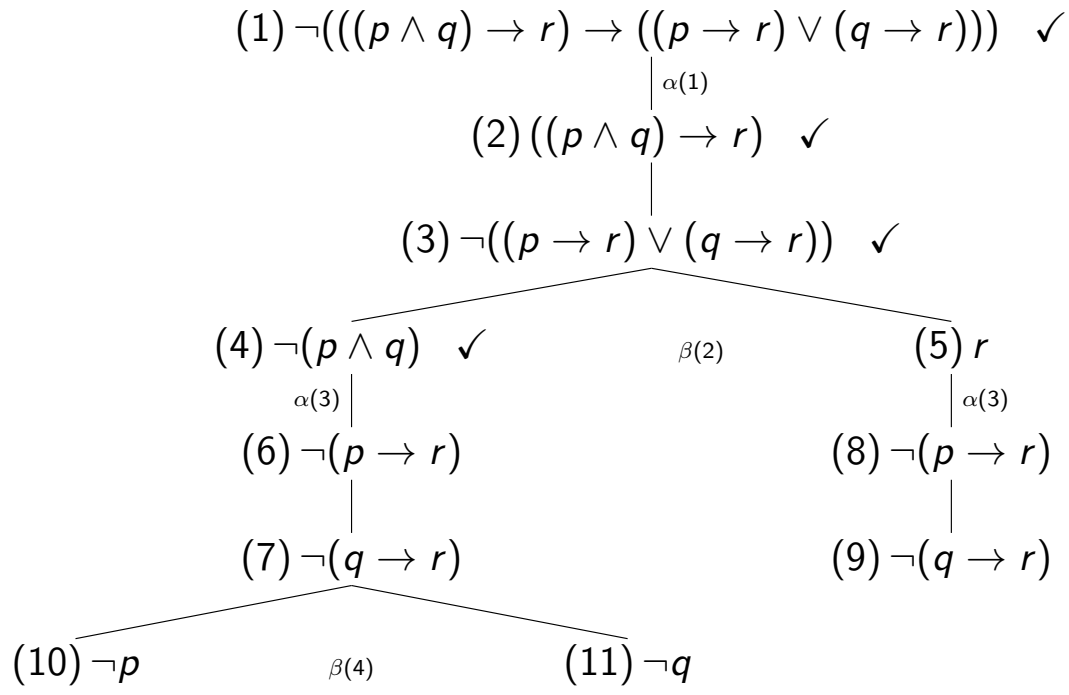
Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?



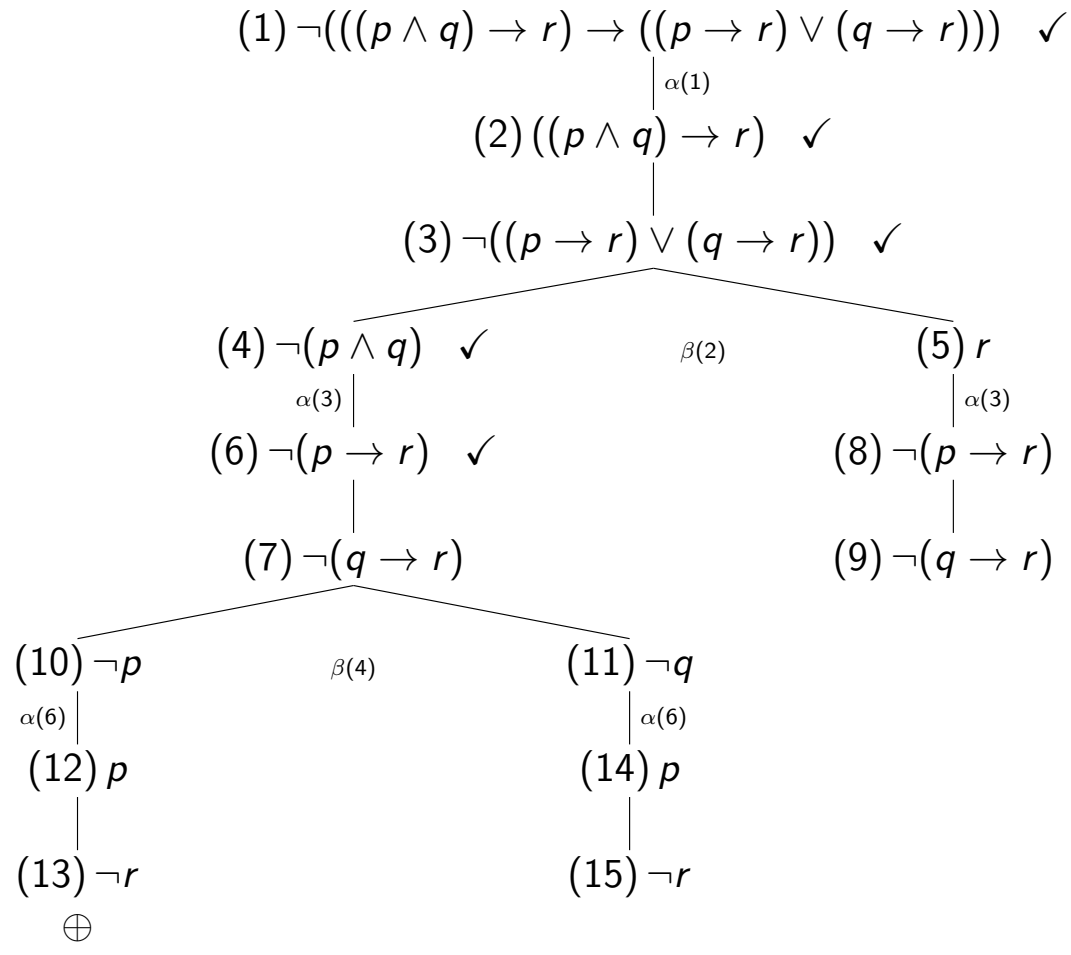
Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?



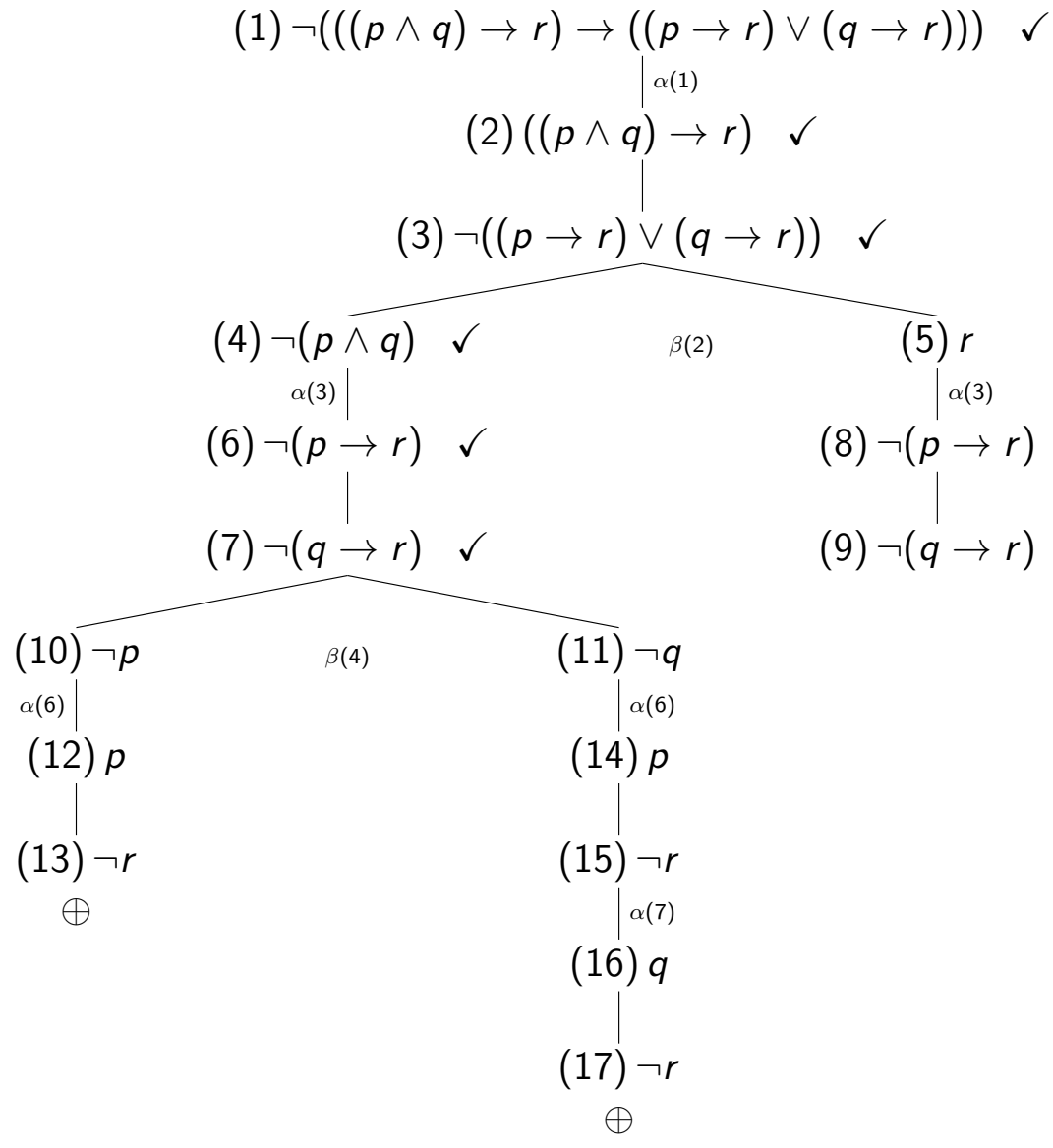
Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?



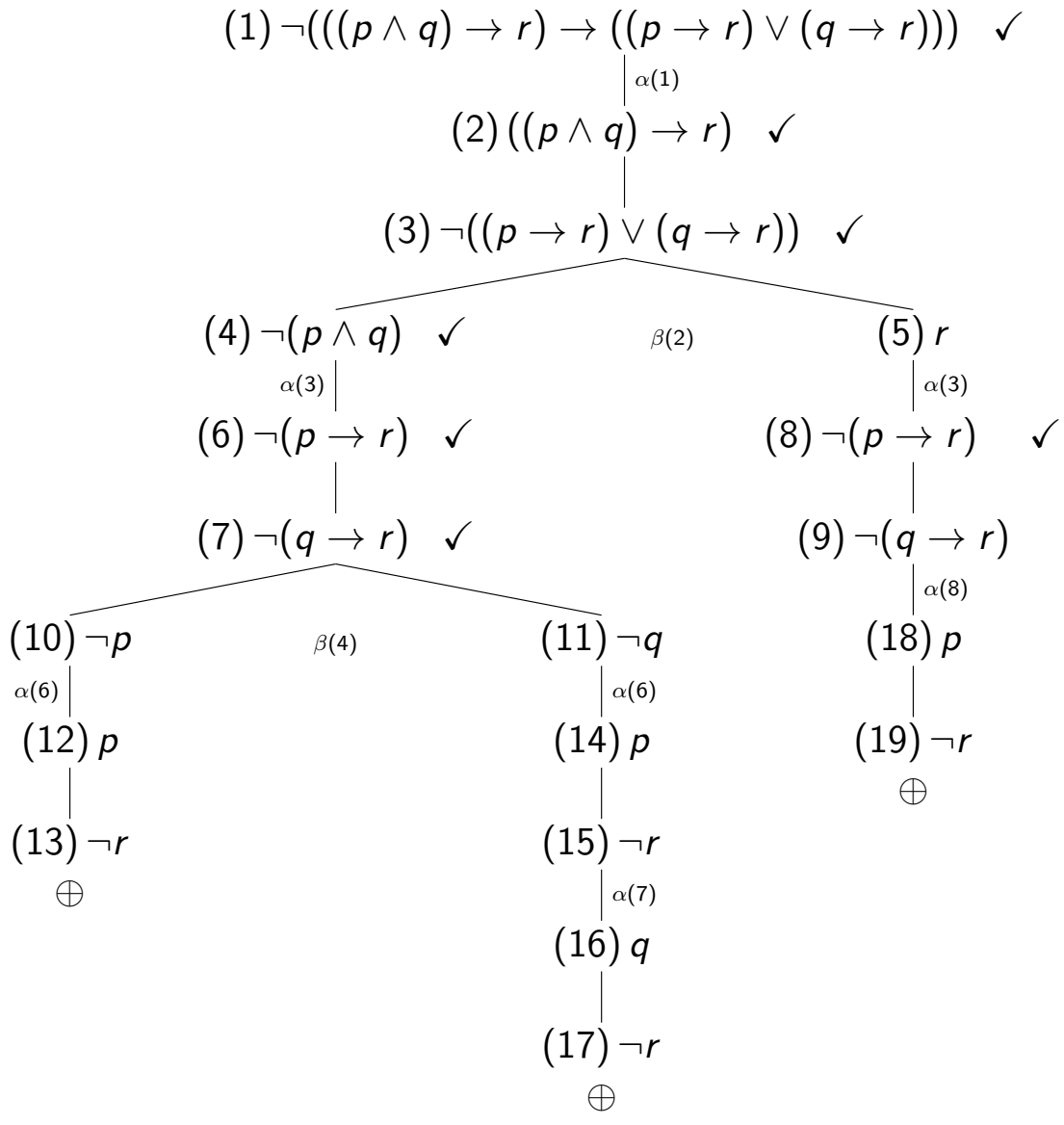
Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?



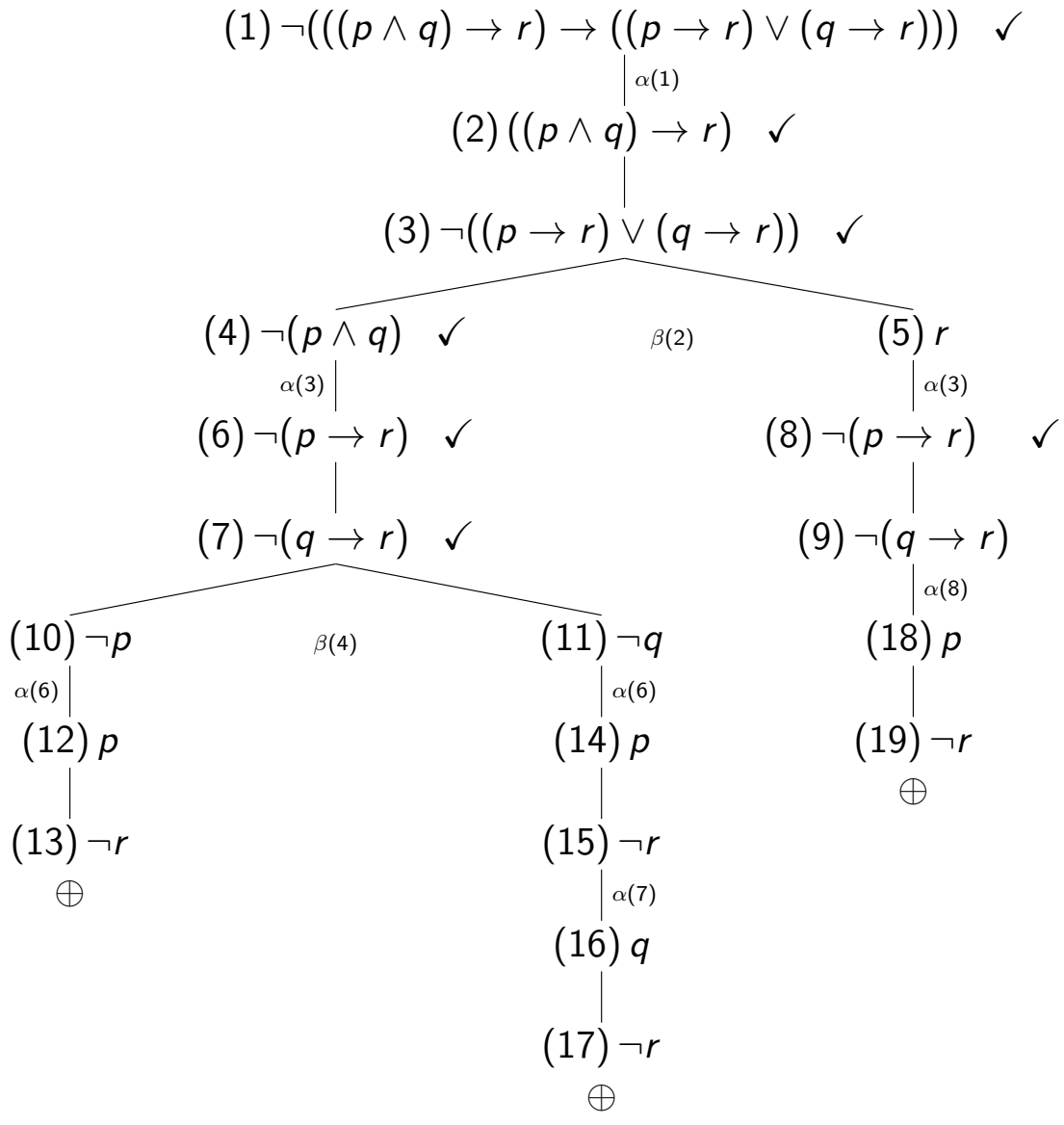
Example 1: Is formula $(((p \wedge q) \to r) \to ((p \to r) \vee (q \to r)))$ valid?



Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?



Example 1: Is formula $(((p \wedge q) \rightarrow r) \rightarrow ((p \rightarrow r) \vee (q \rightarrow r)))$ valid?



Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?

(1) $\neg(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$

Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?

$(1)\ \neg(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ ✓

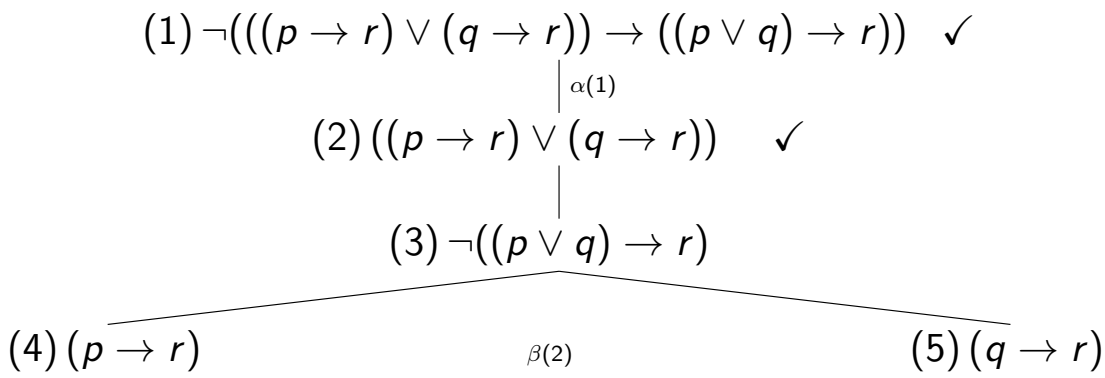
$\Big|\ \alpha(1)$

$(2)\ ((p \to r) \vee (q \to r))$

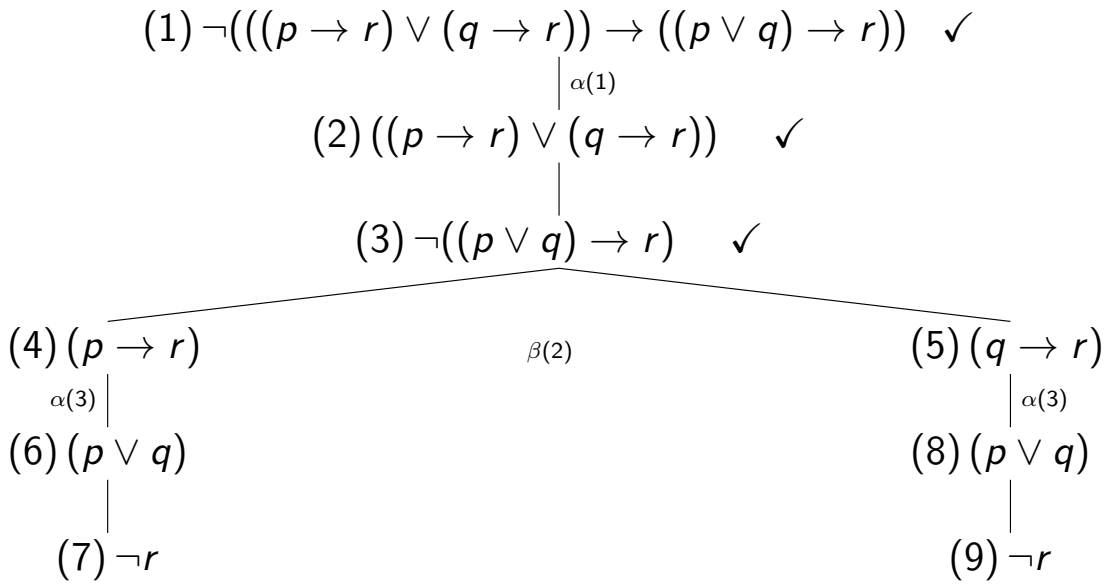
$\Big|$

$(3)\ \neg((p \vee q) \to r)$

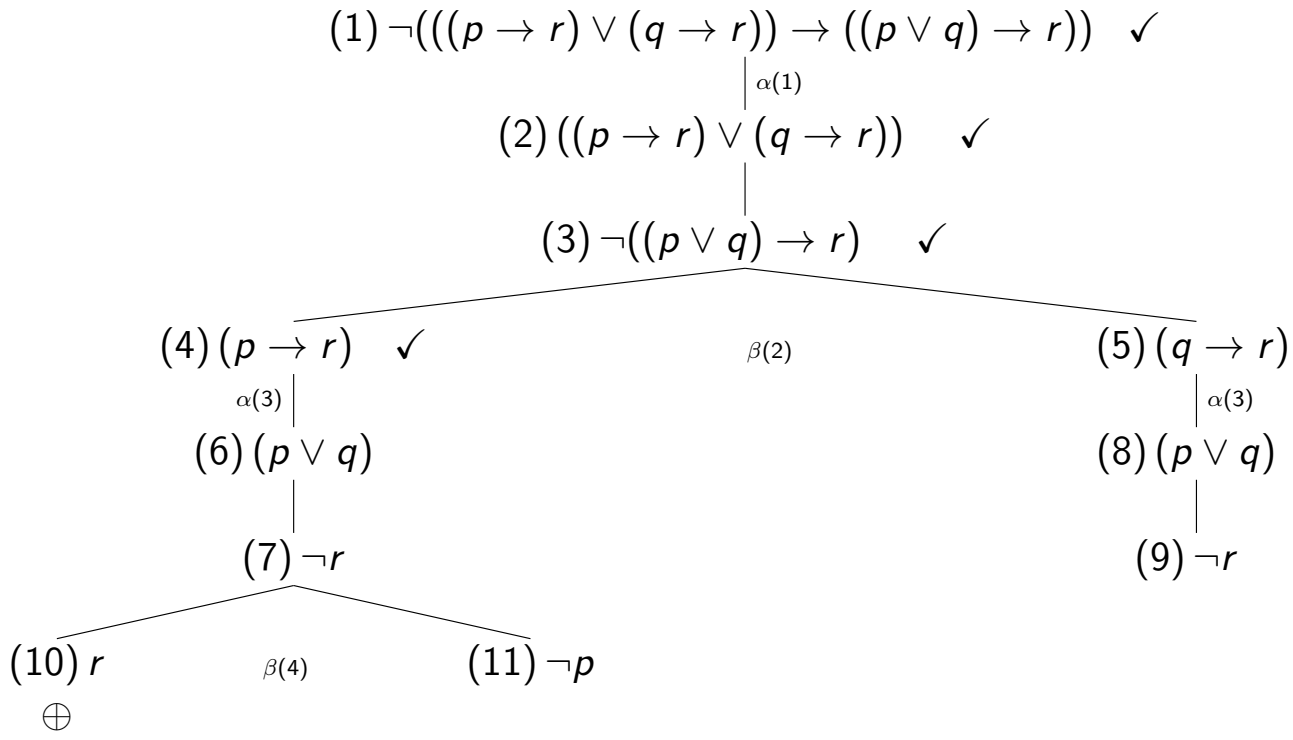
Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



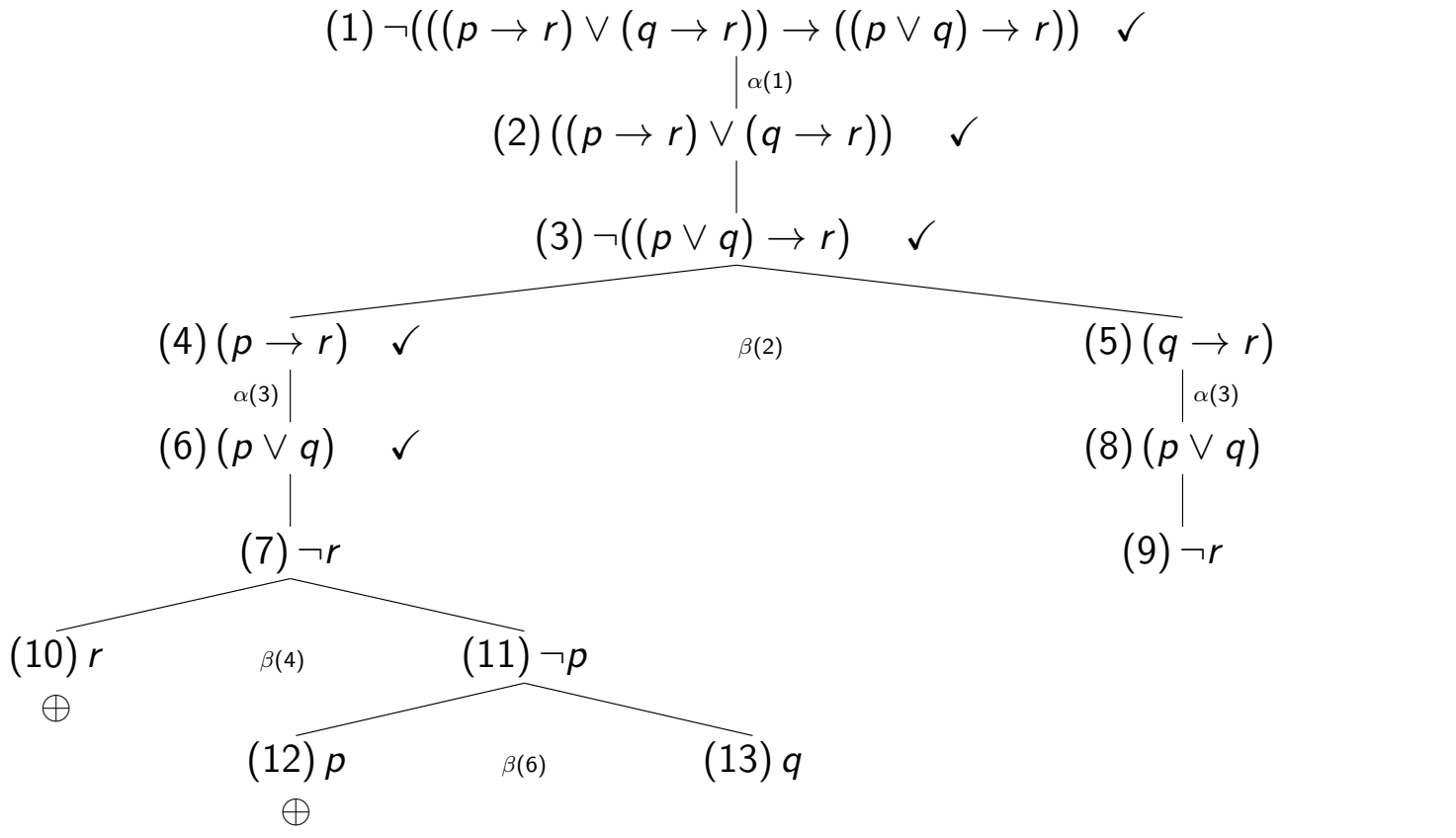
Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



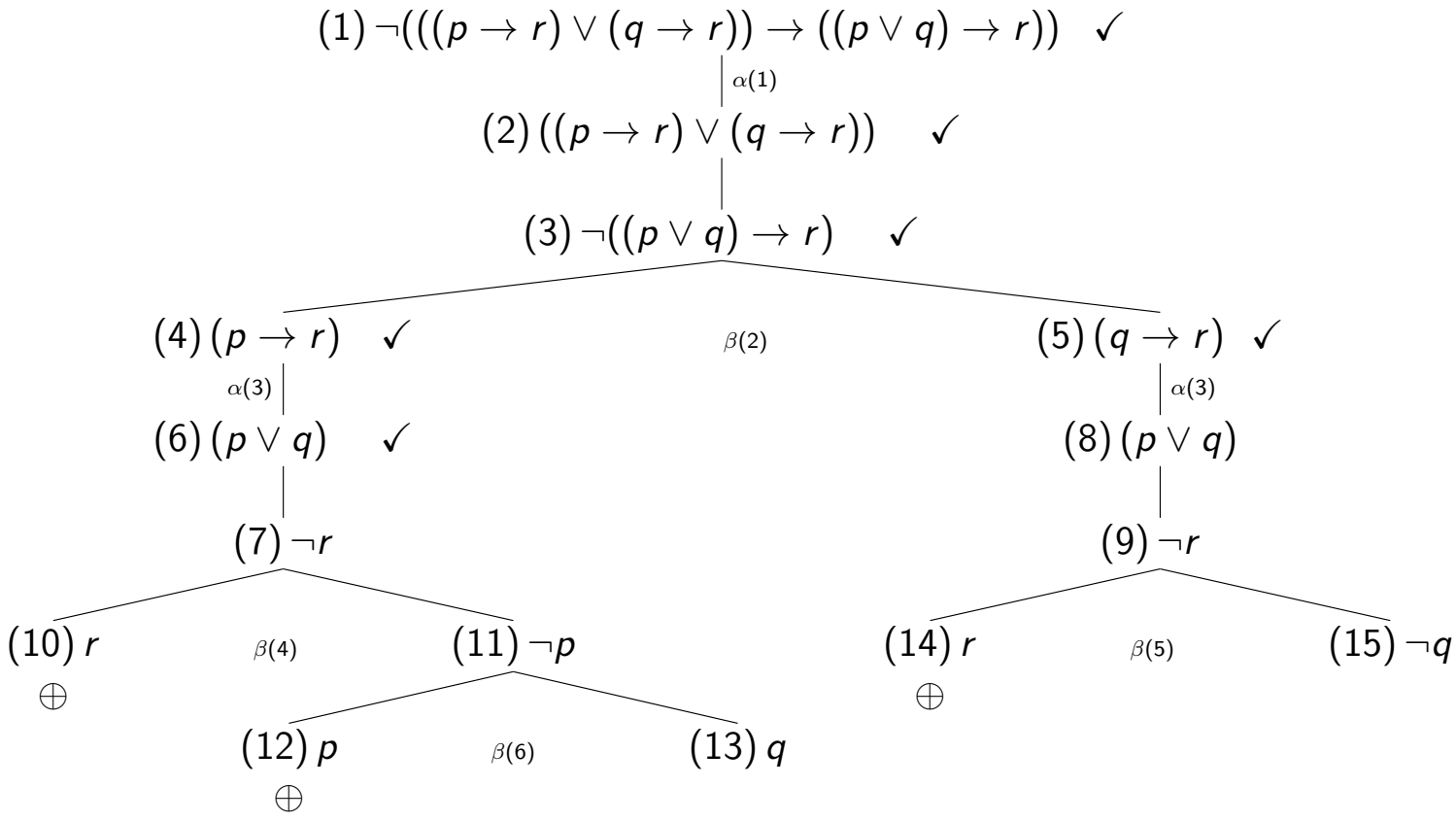
Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



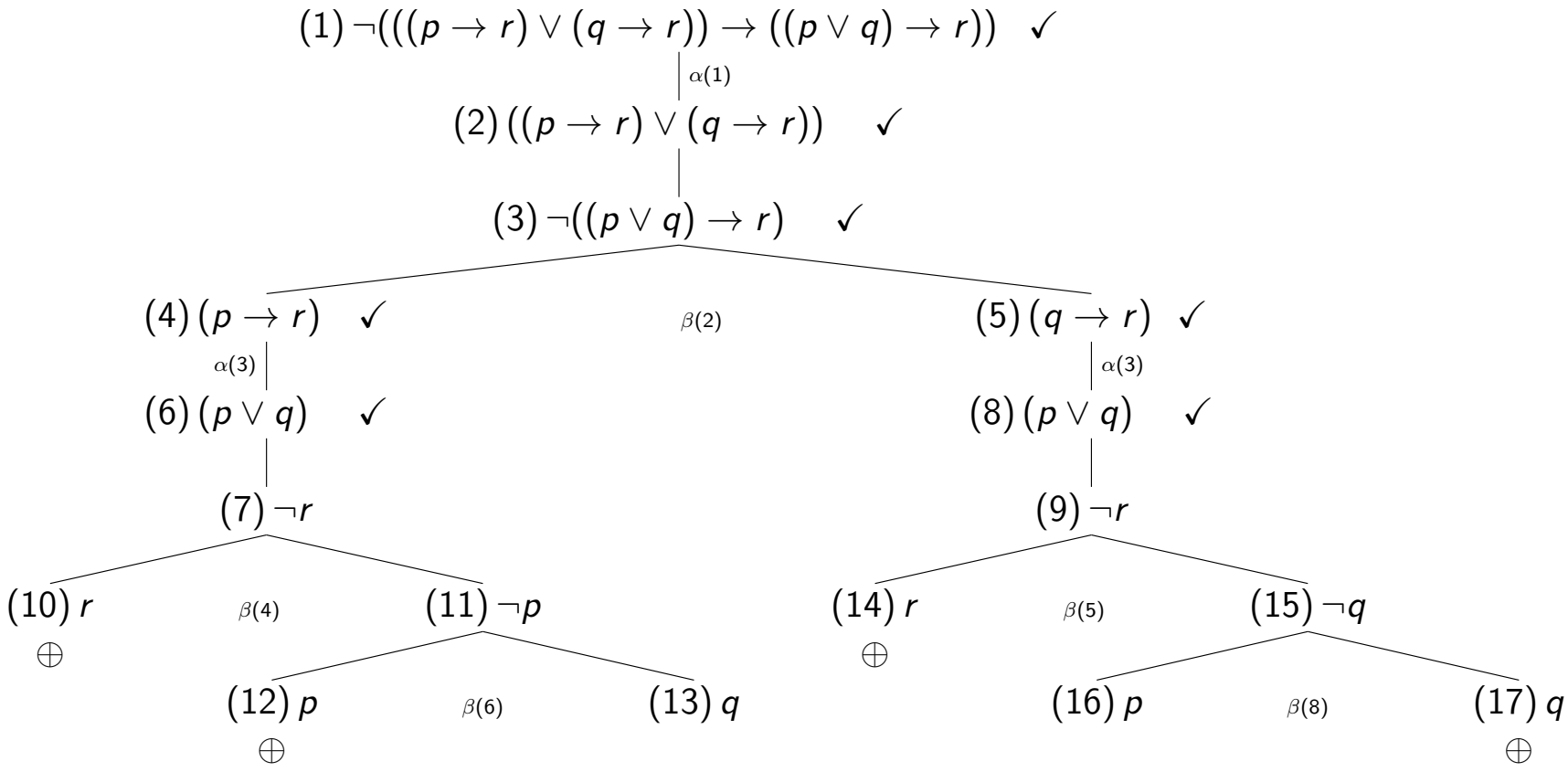
Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



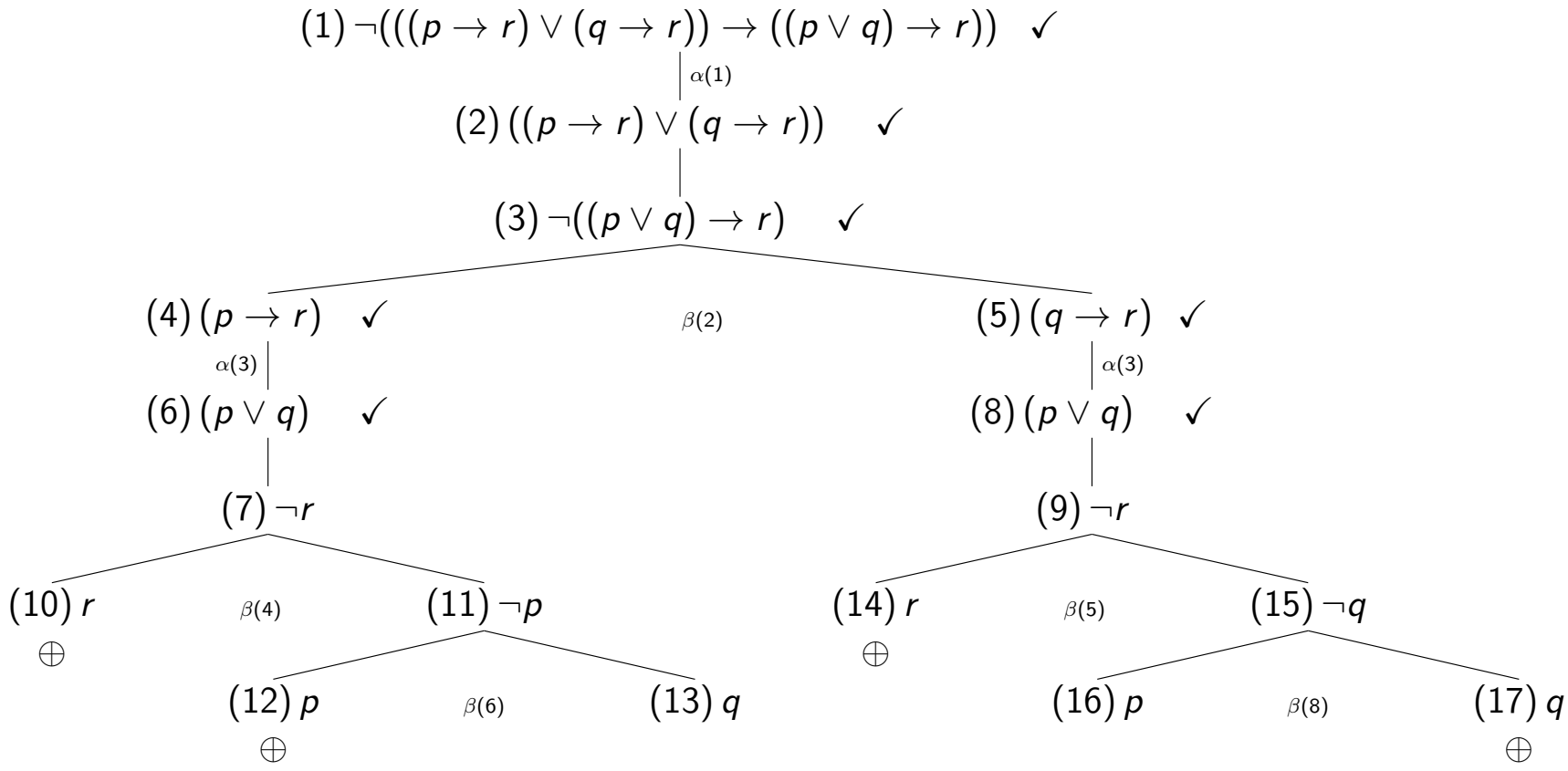
Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



Two open branches — **root formula is satisfiable, original formula is not valid.**

Disjunctive Normal Form (DNF)

A **DNF** formula is a disjunctions of clauses where each clause is a conjunction of literals. E.g.

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

Also

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

Every propositional formula has an equivalent formula in DNF. A **CNF** formula is a conjunction of clauses where each clause is disjunction of literals. E.g.

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

Converting a formula to DNF

1. By tableau
2. By truth table
3. By logical equivalences (De Morgan's laws, distribution laws etc., later)

Using tableau to find DNF equivalent

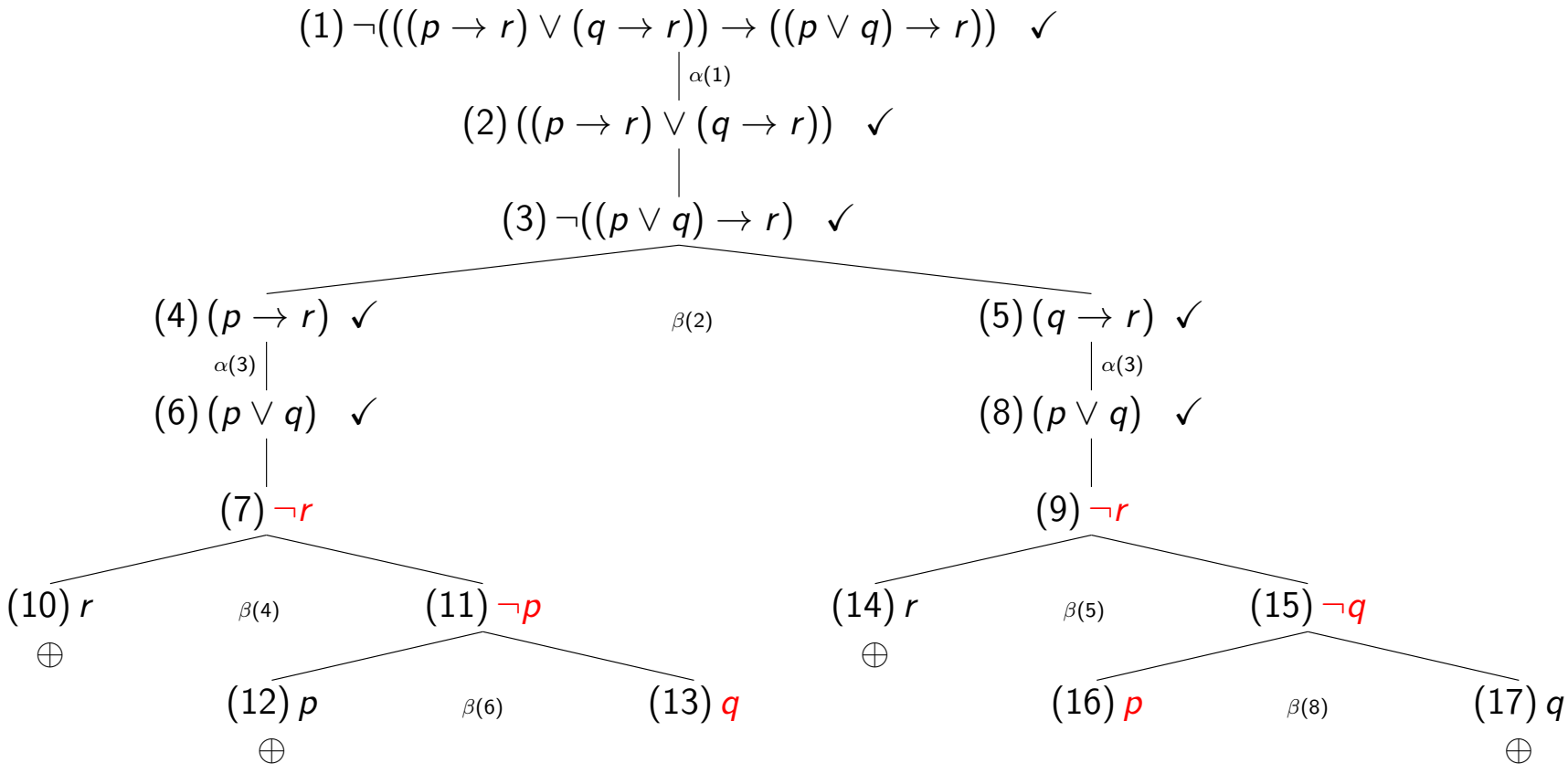
- Place ϕ at root of new tableau.
- Expand until tableau T is completed (no nodes left to expand).
- For each open branch Θ of T let

$$C_\Theta = \bigwedge \{\text{literals in } \Theta\}$$

- Then

$$\phi \equiv \bigvee_{\text{open branches } \Theta} C_\Theta$$

Example 2: Is formula $(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ valid?



The **DNF** for $\neg(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$ is $(\neg r \wedge \neg p \wedge q) \vee (\neg r \wedge \neg q \wedge p)$

DNF by truth table

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

p	q	r	$\neg(((p \to r) \vee (q \to r)) \to ((p \vee q) \to r))$	
0	0	0	0	
0	0	1	0	
0	1	0	1	$(\neg p \wedge q \wedge \neg r)$
0	1	1	0	$\vee$
1	0	0	1	$(p \wedge \neg q \wedge \neg r)$
1	0	1	0	
1	1	1	0	

Problem

- How do you convert a propositional formula into an equivalent CNF?
- E.g.

$$((p \wedge \neg q) \vee (q \wedge \neg r)) \equiv ??$$

First Order Tableaus

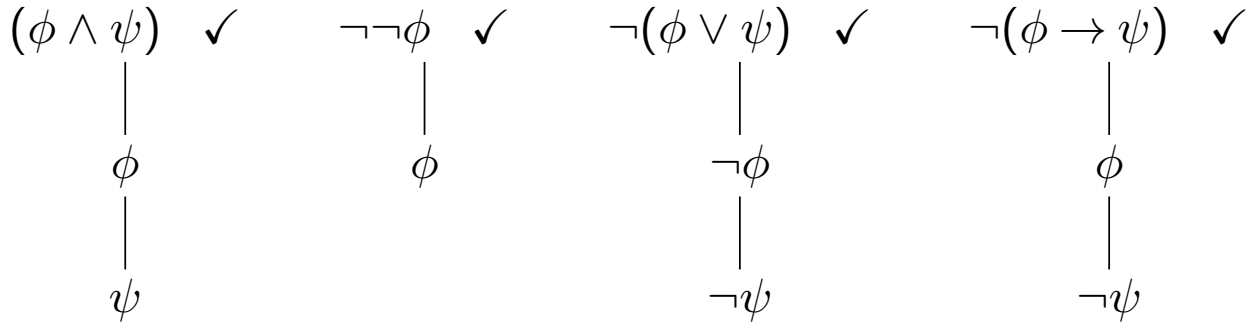
- A **literal** is an atom or its negation (i.e. $r^n(t_1, \ldots, t_n)$ or $\neg r^n(t_1, \ldots, t_n)$ where r is a predicate and t_i is a term)
- A **closed term** is a term that contains no variables (e.g. constants, functions over constants, etc.)
- We use the same kind of tableau construction that we used for propositional logic...

First Order Tableaus

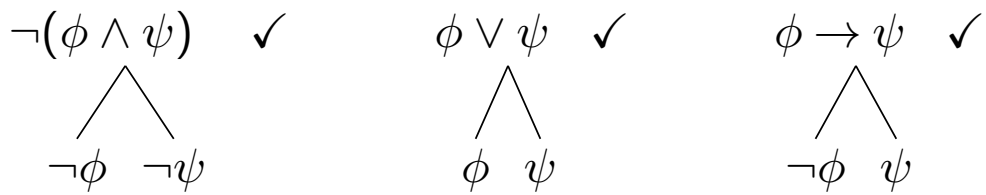
- A **literal** is an atom or its negation (i.e. $r^n(t_1, \ldots, t_n)$ or $\neg r^n(t_1, \ldots, t_n)$ where r is a predicate and t_i is a term)
- A **closed term** is a term that contains no variables (e.g. constants, functions over constants, etc.)
- We use the same kind of tableau construction that we used for propositional logic...
- ...but we need new expansion rules to deal with the quantifiers!

Expansion Rules

α formulas Add both formulas in one branch at each leaf below current node. Tick current node.



β formulas Make two separate branches (one with each formula) at each leaf below current node. Tick current node.



δ formulas

Choose new constant c (not included in tableau so far). Add formula at each leaf below current node. Tick current node.

$$\begin{array}{c} \exists x\phi \ \checkmark \\ | \\ \phi(c/x) \end{array}$$

δ formulas

Choose new constant c (not included in tableau so far). Add formula at each leaf below current node. Tick current node.

$$\begin{array}{cc} \exists x\phi \checkmark & \neg\forall x\phi \checkmark \\ | & | \\ \phi(c/x) & \neg\phi(c/x) \end{array}$$

Observe: $\neg\forall x\phi \equiv \exists x\neg\phi$

γ formulas

Pick any closed term t. Add formula at each leaf below current node. Do not tick the node.

$$\begin{array}{c} \forall x \phi \\ | \\ \phi(t/x) \end{array}$$

γ formulas

Pick any closed term t. Add formula at each leaf below current node. Do not tick the node.

$$\begin{array}{cc} \forall x\phi & \neg\exists x\phi \\ | & | \\ \phi(t/x) & \neg\phi(t/x) \end{array}$$

Observe: $\neg\exists x\phi \equiv \forall x\neg\phi$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$
- $H(a) \rightarrow F(a)$
- $\neg\neg\exists y\ p(y)$
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$
- $G(a) \rightarrow H(a)$
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \to F(a)$
- $\neg\neg\exists y\ p(y)$
- $\neg(\forall x\ \neg p(x) \to \neg\exists y\ p(y))$
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$
- $G(a) \to H(a)$
- $\neg\neg(\forall x\,(G(x) \to H(x)) \wedge \forall x\,(H(x) \to F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \rightarrow F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$
- $G(a) \rightarrow H(a)$
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \rightarrow F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$
- $G(a) \rightarrow H(a)$
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \rightarrow F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$ **(α rule)**
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$
- $G(a) \rightarrow H(a)$
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \rightarrow F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$ **(α rule)**
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$ **(α rule)**
- $G(a) \rightarrow H(a)$
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \to F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \to \neg\exists y\ p(y))$ **(α rule)**
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$ **(α rule)**
- $G(a) \to H(a)$ **(β rule)**
- $\neg\neg(\forall x\,(G(x) \to H(x)) \wedge \forall x\,(H(x) \to F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \rightarrow F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$ **(α rule)**
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$ **(α rule)**
- $G(a) \rightarrow H(a)$ **(β rule)**
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$ **(α rule)**
- $\neg\exists x\,(G(x) \wedge F(x))$
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \rightarrow F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \rightarrow \neg\exists y\ p(y))$ **(α rule)**
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$ **(α rule)**
- $G(a) \rightarrow H(a)$ **(β rule)**
- $\neg\neg(\forall x\,(G(x) \rightarrow H(x)) \wedge \forall x\,(H(x) \rightarrow F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$ **(α rule)**
- $\neg\exists x\,(G(x) \wedge F(x))$ **(γ rule)**
- $\neg\forall y \neg(c < y)$

Which expansion rule should be applied to the first-order formulae below?

- $\forall x\ \neg p(x)$ **(γ rule)**
- $H(a) \to F(a)$ **(β rule)**
- $\neg\neg\exists y\ p(y)$ **(α rule)**
- $\neg(\forall x\ \neg p(x) \to \neg\exists y\ p(y))$ **(α rule)**
- $\neg(\forall x \neg q(x) \vee \exists x \forall y \neg(x < y))$ **(α rule)**
- $G(a) \to H(a)$ **(β rule)**
- $\neg\neg(\forall x\,(G(x) \to H(x)) \wedge \forall x\,(H(x) \to F(x)) \wedge G(a) \wedge \neg\exists x\,(G(x) \wedge F(x)))$ **(α rule)**
- $\neg\exists x\,(G(x) \wedge F(x))$ **(γ rule)**
- $\neg\forall y \neg(c < y)$ **(δ rule)**

Example 3: Is formula $(\forall x \neg p(x) \rightarrow \neg \exists y\ p(y))$ valid?

$$(1)\ \neg(\forall x\ \neg p(x) \rightarrow \neg \exists y\ p(y))$$

Example 3: Is formula $(\forall x \neg p(x) \rightarrow \neg \exists y\ p(y))$ valid?

$$(1)\ \neg(\forall x\ \neg p(x) \rightarrow \neg \exists y\ p(y)) \quad \checkmark$$

$$\Big|\ \alpha(1)$$

$$(2)\ \forall x\ \neg p(x)$$

$$\Big|$$

$$(3)\ \neg\neg \exists y\ p(y)$$

Example 3: Is formula $(\forall x \neg p(x) \rightarrow \neg \exists y\ p(y))$ valid?

$$(1)\ \neg(\forall x\ \neg p(x) \rightarrow \neg \exists y\ p(y)) \quad \checkmark$$

$$\Big|\ \alpha(1)$$

$$(2)\ \forall x\ \neg p(x)$$

$$\Big|$$

$$(3)\ \neg\neg \exists y\ p(y) \quad \checkmark$$

$$\Big|\ \alpha(3)$$

$$(4)\ \exists y\ p(y)$$

Example 3: Is formula $(\forall x \neg p(x) \rightarrow \neg \exists y\ p(y))$ valid?

$$(1)\ \neg(\forall x\ \neg p(x) \rightarrow \neg \exists y\ p(y)) \quad \checkmark$$

$$\Big|\ \alpha(1)$$

$$(2)\ \forall x\ \neg p(x)$$

$$\Big|$$

$$(3)\ \neg\neg \exists y\ p(y) \quad \checkmark$$

$$\Big|\ \alpha(3)$$

$$(4)\ \exists y\ p(y) \quad \checkmark$$

$$\Big|\ \delta(4, c)$$

$$(5)\ p(c)$$

Example 3: Is formula $(\forall x \neg p(x) \rightarrow \neg \exists y\ p(y))$ valid?

$$(1)\ \neg(\forall x\ \neg p(x) \rightarrow \neg \exists y\ p(y)) \quad \checkmark$$

$$\Big|\ \alpha(1)$$

$$(2)\ \forall x\ \neg p(x) \quad \checkmark$$

$$\Big|$$

$$(3)\ \neg\neg \exists y\ p(y) \quad \checkmark$$

$$\Big|\ \alpha(3)$$

$$(4)\ \exists y\ p(y) \quad \checkmark$$

$$\Big|\ \delta(4, c)$$

$$(5)\ p(c)$$

$$\Big|\ \gamma(2, c)$$

$$(6)\ \neg p(c)$$

$$\oplus$$

Closed tableau — **root formula is unsatisfiable, original formula is valid.**

Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?

(1) $\neg\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$

Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \wedge \forall x\,(Hx \to Fx) \wedge Ga \wedge \neg\exists x\,(Gx \wedge Fx))$ valid?

$$(1)\ \neg\neg(\forall x\,(Gx \to Hx) \wedge \forall x\,(Hx \to Fx) \wedge Ga \wedge \neg\exists x\,(Gx \wedge Fx)) \quad \checkmark$$

$$\overset{\alpha(1)}{\big|}$$

$$(2)\ \forall x\,(Gx \to Hx) \wedge \forall x\,(Hx \to Fx) \wedge Ga \wedge \neg\exists x\,(Gx \wedge Fx)$$

Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?

$$(1)\ \neg\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx)) \quad \checkmark$$

$$\overset{\alpha(1)}{\big|}$$

$$(2)\ \forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx) \quad \checkmark$$

$$\overset{\alpha(2)}{\big|}$$

$$(3)\ \forall x\,(Gx \to Hx)$$

$$\big|$$

$$(4)\ \forall x\,(Hx \to Fx)$$

$$\big|$$

$$(5)\ Ga$$

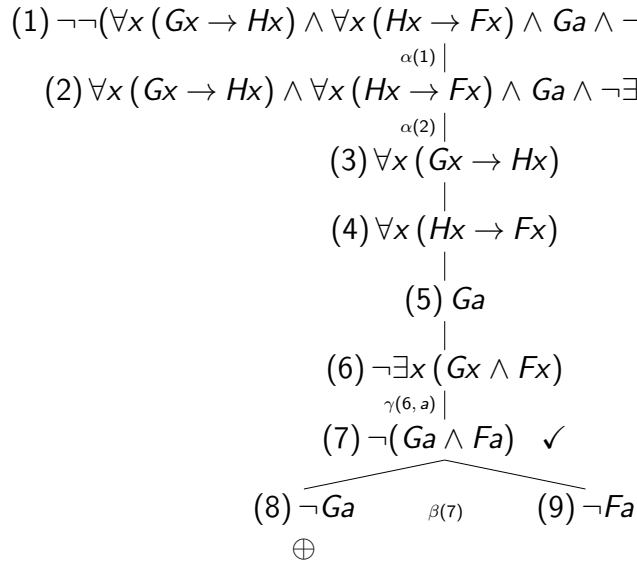
$$\big|$$

$$(6)\ \neg\exists x\,(Gx \land Fx)$$

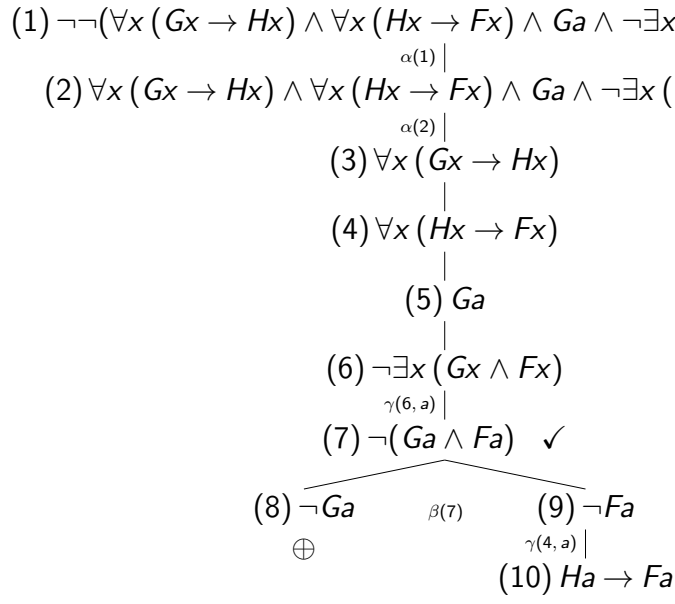
Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?

$$(1)\ \neg\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx)) \quad \checkmark$$
$$\overset{\alpha(1)}{\big|}$$
$$(2)\ \forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx) \quad \checkmark$$
$$\overset{\alpha(2)}{\big|}$$
$$(3)\ \forall x\,(Gx \to Hx)$$
$$\big|$$
$$(4)\ \forall x\,(Hx \to Fx)$$
$$\big|$$
$$(5)\ Ga$$
$$\big|$$
$$(6)\ \neg\exists x\,(Gx \land Fx)$$
$$\overset{\gamma(6,\,a)}{\big|}$$
$$(7)\ \neg(Ga \land Fa)$$

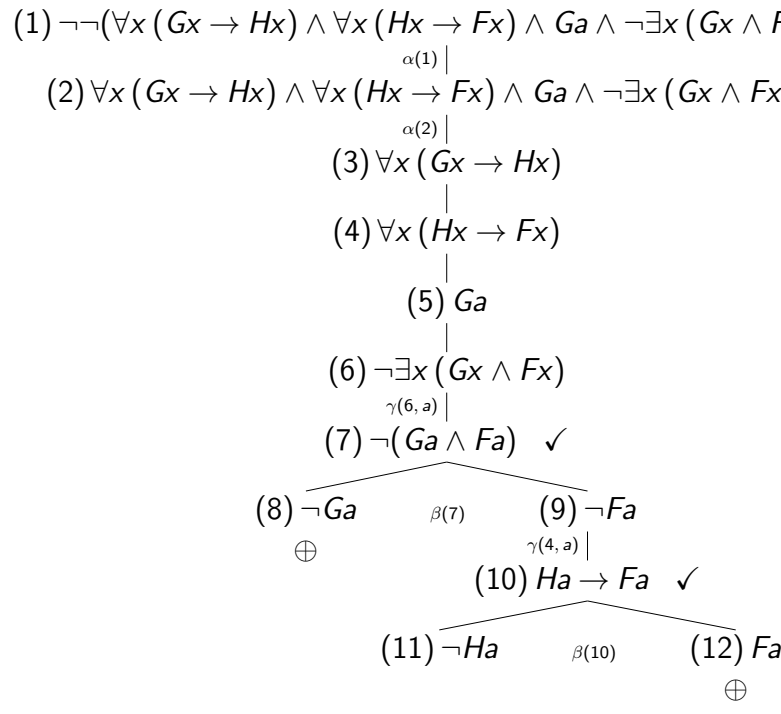
Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?



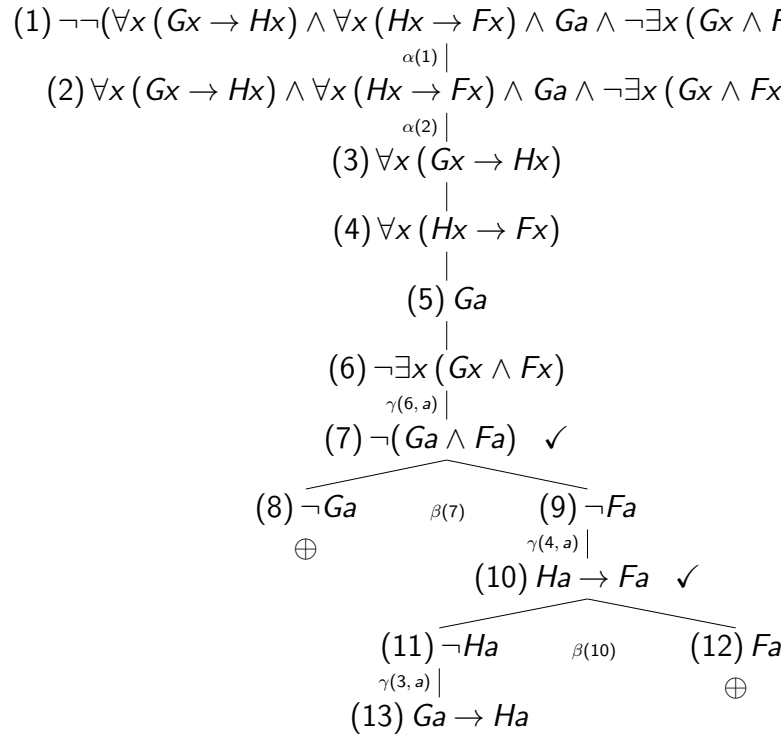
Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?



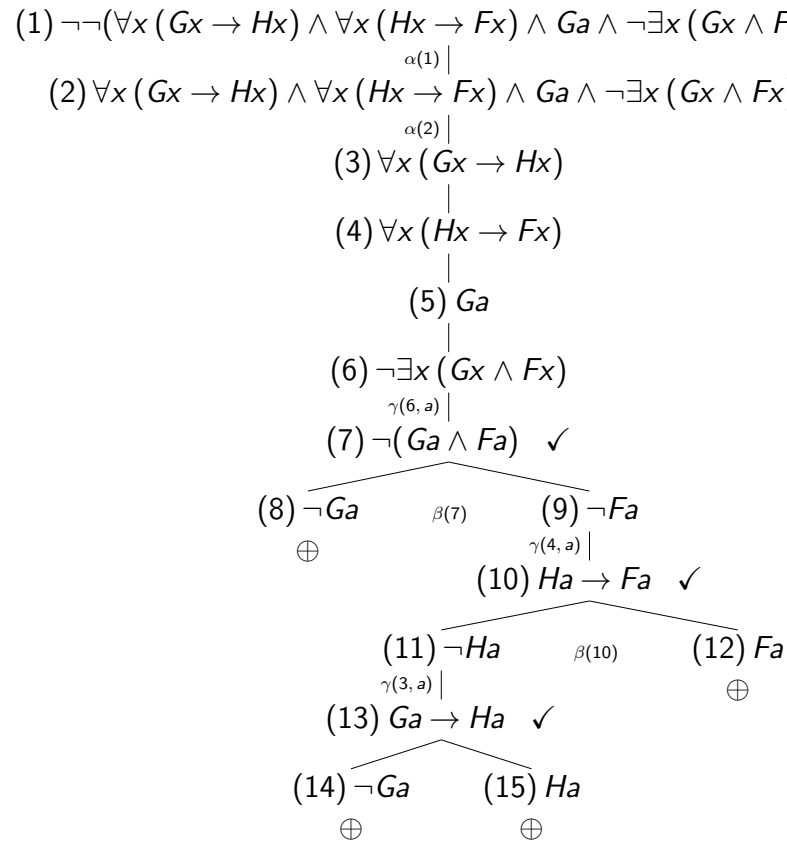
Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?



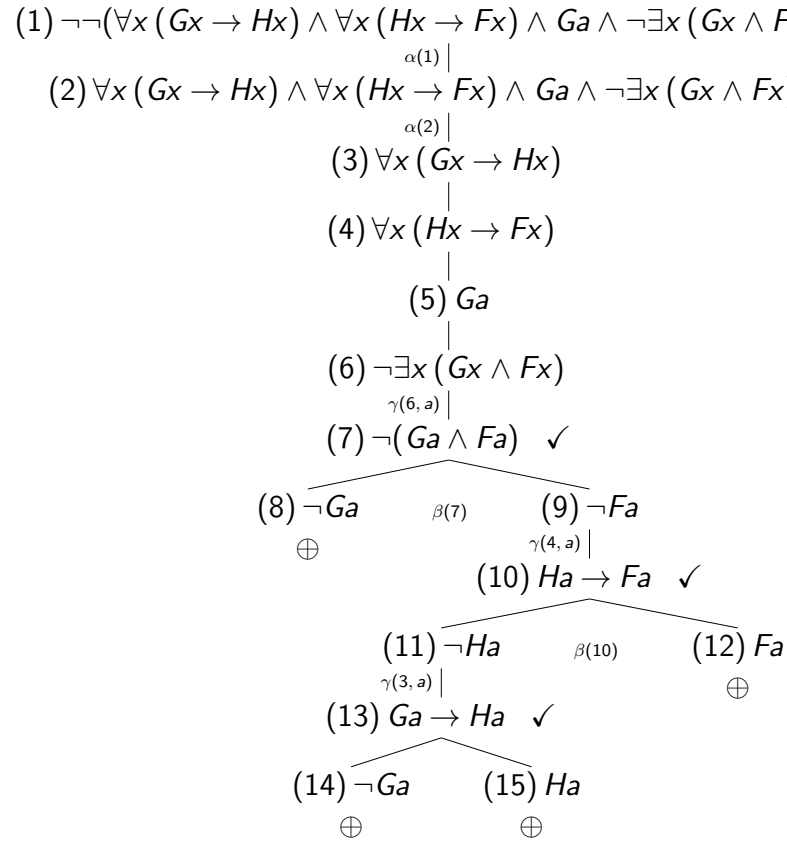
Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?



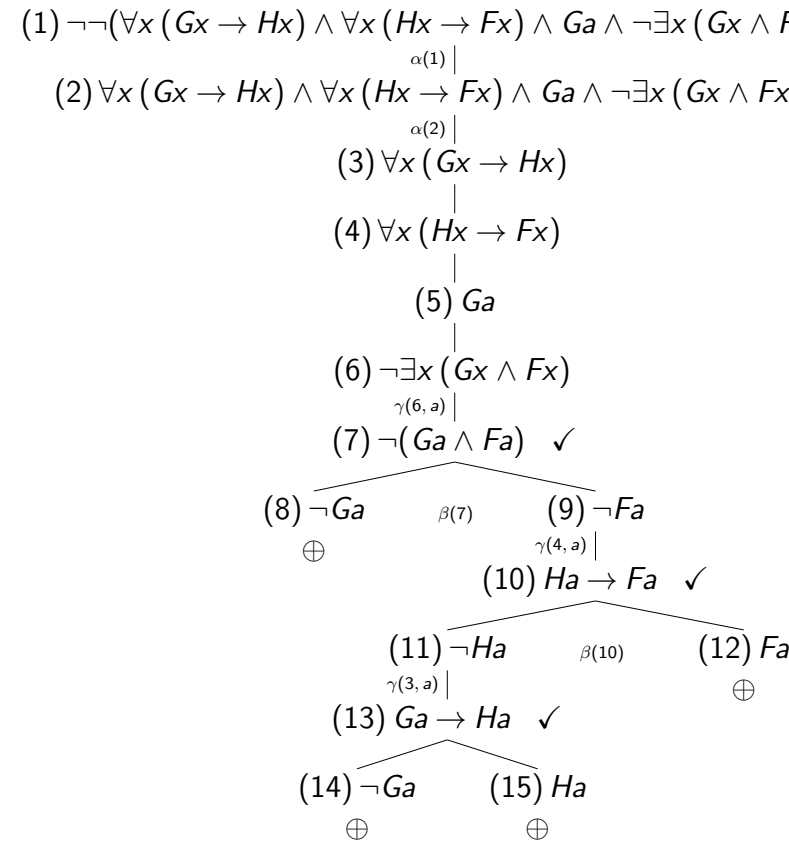
Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?



Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?

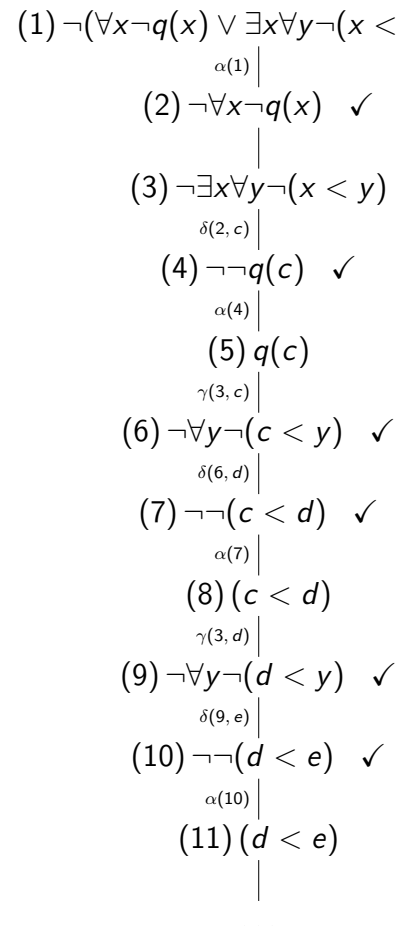


Example 4: Is formula $\neg(\forall x\,(Gx \to Hx) \land \forall x\,(Hx \to Fx) \land Ga \land \neg\exists x\,(Gx \land Fx))$ valid?



Closed tableau — **root formula is not satisfiable and the original formula is valid.**

Example 5: Is formula $\forall x \neg q(x) \vee \exists x \forall y \neg (x < y)$ valid?



Open tableau — the tableau will never close, hence the **root formula is satisfiable and the original formula is not valid.**

Alternative: tableaux as lists

Recall

literal	$p, \neg p$
α	$(\phi_1 \wedge \phi_2), \neg(\phi_1 \vee \phi_2), \neg(\phi_1 \rightarrow \phi_2), \neg\neg\phi$
β	$(\phi_1 \vee \phi_2), (\phi_1 \rightarrow \phi_2), \neg(\phi_1 \wedge \phi_2)$

Formula expansions

α	α_1	α_2
$(A \wedge B)$	A	B
$\neg(A \vee B)$	$\neg A$	$\neg B$
$\neg(A \to B)$	A	$\neg B$
$\neg\neg A$	A	—

β	β_1	β_2
$(A \vee B)$	A	B
$(A \to B)$	$\neg A$	B
$\neg(A \wedge B)$	$\neg A$	$\neg B$

Propositional Tableaux

- A **theory** Σ is a set of propositional formulas.
- If $p, \neg p \in \Sigma$ theory is **contradictory**, write $C(\Sigma)$
- If each formula in Σ is a literal, theory is fully expanded, write $Exp(\Sigma)$
- A **tableau** is a list of theories. Think of these as alternative theories.

Tableau(ϕ)

Initialise $Tab = [\{\phi\}]$
while Not empty Tab **do**
  $\Sigma = Dequeue(Tab)$
  if $Exp(\Sigma)$ **and** NOT $C(\Sigma)$ **then**
    Output SATISFIABLE
  else
    Pick non-literal $\psi \in \Sigma$
    switch (ψ)
    case α**:**
      $\Sigma = \Sigma[\alpha/\{\alpha_1, \alpha_2\}]$, **if** NOT $C(\Sigma)$ **and** $\Sigma \notin Tab$ **then** enqueue Σ
    case β**:**
      $\Sigma_1 = \Sigma[\beta/\beta_1]$, **if** $\Sigma_1 \notin Tab$ **and** NOT $C(\Sigma_1)$ **then** enqueue Σ_1
      $\Sigma_2 = \Sigma[\beta/\beta_2]$, **if** $\Sigma_2 \notin Tab$ **and** NOT $C(\Sigma_2)$ **then** enqueue Σ_2
    end switch
  end if
end while
(Empty Tab) Output UNSATISFIABLE

Predicate Tableaux

Under switch statement add cases δ and γ.

switch (ψ)
case $\delta = \exists x\theta(x)$**:**
  $\Sigma = \Sigma[\exists x\theta(x)/\theta(c)]$ (new const c),
case $\delta = \neg\forall x\theta(x)$**:**
  fill in
case $\gamma = \forall x\theta(x)$**:**
  pick closed term t occurring in Σ, $\Sigma = \Sigma \cup \{\theta(t)\}$.
case $\gamma = \neg\exists x\theta(x)$**:**
  fill in
end switch
If $\Sigma \notin \mathit{Tab}$ and NOT $C(\Sigma)$ enqueue Σ.

Menti.com

Go to www.menti.com

Termination of propositional tableau algorithm

Termination of propositional tableau algorithm

- When running tableau algorithm for ϕ, only new theories are enqueued.

Termination of propositional tableau algorithm

- When running tableau algorithm for ϕ, only new theories are enqueued.
- Let X be set of subformulas of ϕ and single negations of subformulas of ϕ — at most $2|\phi|$ of these.

Termination of propositional tableau algorithm

- When running tableau algorithm for ϕ, only new theories are enqueued.
- Let X be set of subformulas of ϕ and single negations of subformulas of ϕ — at most $2|\phi|$ of these.
- A theory is a subset of X — at most $2^{2|\phi|}$ of these.

Termination of propositional tableau algorithm

- When running tableau algorithm for ϕ, only new theories are enqueued.
- Let X be set of subformulas of ϕ and single negations of subformulas of ϕ — at most $2|\phi|$ of these.
- A theory is a subset of X — at most $2^{2|\phi|}$ of these.
- Algorithm terminates in $2^{2|\phi|}$ steps (at most).

Soundness for Tableaus

$$\left.\begin{array}{l} \vdash \phi \\ \text{Tab}(\neg\phi) \text{ closes} \\ \text{Tab}(\psi) \text{ closes} \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \models \phi \\ \neg\phi \text{ unsat} \\ \psi \text{ unsat} \end{array}\right.$$

Soundness for Tableaus

$$\frac{\left.\begin{array}{l}\vdash \phi \\ \text{Tab}(\neg\phi) \text{ closes} \\ \text{Tab}(\psi) \text{ closes}\end{array}\right\} \Rightarrow \left\{\begin{array}{l}\models \phi \\ \neg\phi \text{ unsat} \\ \psi \text{ unsat}\end{array}\right.}{\text{Tab}(\psi) \text{ never closes} \Leftarrow \psi \text{ is sat}}$$

Soundness of Propositional Tableau Algorithm

If ϕ is satisfiable then tableau for ϕ cannot close, so algorithm outputs SATISFIABLE eventually.

Proof.

Assume $v(\phi) = \top$. Prove by induction on number n of iterations of while statement that there is $\Sigma \in \mathit{Tab}$ where $\theta \in \Sigma \to v(\theta) = \top$.
True for $n = 0$ by initialisation and assumption.
Assume after n iterations that there is $\Sigma \in \mathit{Tab}$ such that $\theta \in \Sigma \to v(\theta) = \top$.
If Σ is dequeued and $\psi \in \Sigma$ is picked we have $v(\psi) = \top$ (by IH).
If ψ is an α then $v(\alpha_1) = v(\alpha_2) = \top$ so $\theta \in \Sigma[\alpha/\{\alpha_1, \alpha_2\}] \to v(\theta) = \top$ still true.
If ψ is a β then either $v(\beta_1) = \top$ or $v(\beta_2) = \top$, so either $\theta \in \Sigma_1 \to v(\theta) = \top$ or $\theta \in \Sigma_2 \to v(\theta) = \top$.
Result follows by induction over n.

Soundness of Predicate Tableau Algorithm

Assume ϕ has a model.
Prove by induction on number of iterations of while loop that there is $\Sigma \in \mathit{Tab}$ and $S \models_A \Sigma$ (some S, A).
True for $n = 0$ by assumption.
Assume $\Sigma \in \mathit{Tab}$ and $S \models_A \Sigma$.
When Σ is dequeued and ψ is picked, cases where ψ is α or β are same as propositional case, no change to S, A needed.
If ψ is δ, say $\psi = \exists x \theta(x)$ then $S \models_A \exists x \theta(x)$ (by IH).
So there is $s \in S$ such that $S_{A[x \to s]} \models \theta(x)$.
$\exists x \theta(x)$ gets replaced by $\theta(c)$ in Σ (new constant c).
Let S' be same as S except $I(c) = s$. Then $S' \models_A \Sigma[\exists x \theta(x)/\theta(c)]$.
If ψ is a γ, say $\psi = \forall x \theta(x)$ then $S \models_A \forall x \theta(x)$ (I.H.)
Follows that $S \models_A \theta(t)$ for any closed term t.
$S \models_A \Sigma[\forall x \theta(x)/\theta(t)]$ still true (no need to change S, A in this case).

Tableau proof from hypothesis

As with axiomatic proofs, you can use tableau to prove from hypotheses.

Tableau proof from hypothesis

As with axiomatic proofs, you can use tableau to prove from hypotheses.

Let Γ be a set of hypotheses, let ϕ be a formula.

Tableau proof from hypothesis

As with axiomatic proofs, you can use tableau to prove from hypotheses.

Let Γ be a set of hypotheses, let ϕ be a formula.

For tableau as list of theories, initialise as $[\Gamma \cup \{\neg\phi\}]$.

If list becomes empty, write $\Gamma \vdash \phi$

Tableau proof from hypothesis

As with axiomatic proofs, you can use tableau to prove from hypotheses.

Let Γ be a set of hypotheses, let ϕ be a formula.

For tableau as list of theories, initialise as $[\Gamma \cup \{\neg\phi\}]$.

If list becomes empty, write $\Gamma \vdash \phi$

For tableau as tree, start with $\neg\phi$ at root.

Expand as before, but add new rule — at any stage you can pick $\gamma \in \Gamma$ and add node labelled γ at any leaf.

If tableau eventually closes, we may write

$$\Gamma \vdash \phi$$

Tableau proof from hypothesis

As with axiomatic proofs, you can use tableau to prove from hypotheses.
Let Γ be a set of hypotheses, let ϕ be a formula.
For tableau as list of theories, initialise as $[\Gamma \cup \{\neg\phi\}]$.
If list becomes empty, write $\Gamma \vdash \phi$
For tableau as tree, start with $\neg\phi$ at root.
Expand as before, but add new rule — at any stage you can pick $\gamma \in \Gamma$ and add node labelled γ at any leaf.
If tableau eventually closes, we may write

$$\Gamma \vdash \phi$$

If Γ is a finite set, say $\{\gamma_0, \gamma_1 \ldots, \gamma_{n-1}\}$ you may as well just start with $\neg\phi, \gamma_0, \ldots, \gamma_{n-1}$ in a single branch of new tableau and try to close it.

Ancestors

If $\Sigma \in \mathit{Tab}$ is dequeued and Σ_1 (Σ_2) are enqueued, then Σ is parent of Σ_1 (Σ_2)

$$P(\Sigma) = \Sigma' \text{ if } \Sigma' \text{ is the parent of } \Sigma$$
$$P^0(\Sigma) = \Sigma$$
$$P^{n+1}(\Sigma) = P(P^n(\Sigma))$$

Say Σ is ancestor of Σ' if there is $n \geq 0$ and $P^n(\Sigma') = \Sigma$. Initial tableau $[\{\phi\}]$ and $\{\phi\}$ is ancestor of every theory in the tableau.

Completeness for Tableaus

$$\left.\begin{array}{l} \vdash \phi \\ \text{Tab}(\neg\phi) \text{ closes} \\ \text{Tab}(\psi) \text{ closes} \end{array}\right\} \Leftarrow \left\{\begin{array}{l} \models \phi \\ \neg\phi \text{ unsat} \\ \psi \text{ unsat} \end{array}\right.$$

Completeness for Tableaus

$$\frac{\left.\begin{array}{l}\vdash \phi \\ \text{Tab}(\neg\phi) \text{ closes} \\ \text{Tab}(\psi) \text{ closes}\end{array}\right\} \Leftarrow \left\{\begin{array}{l}\models \phi \\ \neg\phi \text{ unsat} \\ \psi \text{ unsat}\end{array}\right.}{\text{Tab}(\psi) \text{ never closes } \Rightarrow \psi \text{ is sat}}$$

Completeness for Tableaus

$$\left.\begin{array}{l} \vdash \phi \\ \text{Tab}(\neg\phi) \text{ closes} \\ \text{Tab}(\psi) \text{ closes} \end{array}\right\} \Leftarrow \left\{\begin{array}{l} \models \phi \\ \neg\phi \text{ unsat} \\ \psi \text{ unsat} \end{array}\right.$$

$$\text{Tab}(\psi) \text{ never closes} \quad \text{(with fair schedule)} \Rightarrow \psi \text{ is sat}$$

Completeness of Propositional Tableau Algorithm

Start tableau with ϕ. If algorithm outputs SATISFIABLE then there is $v : \mathit{Props} \to \{\top, \bot\}$ such that $v(\phi) = \top$, so ϕ is satisfiable.

Proof.

Output SATISFIABLE $\Rightarrow \Sigma \in \mathit{Tab}$ dequeued $\wedge\ \mathit{Exp}(\Sigma) \wedge \neg C(\Sigma)$
Define v by $v(p) = \top \iff p \in \Sigma$.
We have $\phi \in \Sigma \Rightarrow v(\phi) = \top$
We prove, if $(\phi \in \Sigma' \Rightarrow v(\phi) = \top)$ then $(\phi \in P(\Sigma') \Rightarrow v(\phi) = \top)$.
For this, one formula is replaced in $P(\Sigma')$ by expansion formula(s) in Σ'.
If it is an α, then $\alpha_1, \alpha_2 \in \Sigma'$, so $v(\alpha_1) = v(\alpha_2) = \top$, hence $v(\alpha) = \top$
If it is a β, then either $\beta_1 \in \Sigma'$ or $\beta_2 \in \Sigma'$, so either $v(\beta_1) = \top$ or $v(\beta_2) = \top$, either way $v(\beta) = \top$.
Other formulas in $P(\Sigma')$ don't change in Σ'.
It follows (by induction) that all formulas in the original ancestor theory are true under v.
Hence $v(\phi) = \top$, so ϕ is satisfiable.

Herbrand Structures

A closed term t is built up from constants and function symbols only — no variables.

A Herbrand structure $H = (D, I)$ has

Domain

$$D = \{\text{closed terms}\}$$

Interpretation $I = (I_c, I_f, I_p)$.

$$\begin{aligned} I_c(c) &= c \\ I_f(f^n) : (d_1, \ldots, d_n) &\mapsto f^n(d_1, \ldots, d_n) \end{aligned}$$

I_p can be chosen freely.

It follows, for any closed term t, that

$$[t]^{H,A} = t$$

Herbrand Theorem

Let L be a language with ∞ many constant symbols (and no equality predicate in this version of the theorem).
If ϕ is satisfiable (i.e. $S \models_A \phi$, some S some A) then ϕ is satisfiable in a Herbrand model H, i.e. $H \models_A \phi$ (some A).

Fairness

Suppose you have several (countably many) processes $P_1, P_2, \ldots, P_k, \ldots$ and each of them is waiting for some input. It might be that when you give P_i some input, it creates a new process P_{k+1}, so the list can grow, but it will always be countable. In what order should you supply inputs to the various processes? You could simply supply input to P_1 again and again, but that would be unfair to all the other processes. In a fair schedule, if any process P_i is waiting for input at time t then eventually (at some time $t' > t$) P_i will get some input. If a process is always waiting for input, then it will get input infinitely often. Since the total number of requests for input is countable, it is possible to find a fair schedule.

Rank

$$
\begin{aligned}
Rk(P(t_0, \ldots, t_{k-1})) &= 1 \\
Rk(\neg\phi) &= 1 + Rk(\phi) \\
Rk((\phi \circ \psi)) &= 1 + Rk(\phi) + Rk(\psi) \\
Rk(\exists x \phi) &= 1 + Rk(\phi) \\
Rk(\forall x \phi) &= 1 + Rk(\phi)
\end{aligned}
$$

So $Rk(\phi)$ is number of nodes in parse tree.

Formula Induction

Base case Prove $Rk(\phi) \leq 2 \Rightarrow \pi(\phi)$

Induction Hypothesis Assume (for some $k \geq 2$) that
$Rk(\phi) \leq k \Rightarrow \pi(\phi)$

Induction Step Prove that $Rk(\phi) \leq k + 1 \Rightarrow \pi(\phi)$.

If you prove the base case, and prove the induction step using induction hypothesis, then you can conclude

$$\pi(\phi)$$

for all FO formulas ϕ.

Completeness of algorithm for predicate tableau

If tableau for ϕ never closes and expanded by a fair schedule then ϕ is satisfiable.

Proof: If tableau never closes, by König's tree lemma there is a sequence $\Sigma_0, \Sigma_1, \Sigma_2, \ldots \in \mathit{Tab}$ where $\Sigma_n = P(\Sigma_{n+1})$. Let $\Sigma = \bigcup_{n<\infty} \Sigma_n$.
Since fair schedule was used,

$$\begin{aligned}
\alpha \in \Sigma &\Rightarrow \alpha_1 \in \Sigma \text{ and } \alpha_2 \in \Sigma \\
\beta \in \Sigma &\Rightarrow \beta_1 \in \Sigma \text{ or } \beta_2 \in \Sigma \\
\exists x \theta(x) \in \Sigma &\Rightarrow \theta(c) \in \Sigma (\text{some } c) \\
\neg \forall x \theta(x) \in \Sigma &\Rightarrow \neg \theta(c) \in \Sigma \text{ (some } c) \\
\forall x \theta(x) \in \Sigma &\Rightarrow \theta(t) \in \Sigma \text{ (all closed terms } t) \\
\neg \exists x \theta(x) &\Rightarrow
\end{aligned}$$

Completeness, continued

Let H be Herbrandt structure, base $\{$closed terms of $\Sigma\}$, $I(t) = t$ (closed term t) and
$(t_0, t_1, \ldots, t_{k-1}) \in I(R^n) \iff R^n(t_0, \ldots, t_{n-1}) \in \Sigma$.
Note that rank of expansion formulas is strictly less than rank of formula.
Let θ be any sentence. Prove by induction on $Rk(\theta)$ that

$$\theta \in \Sigma \Rightarrow H \models \theta.$$

Hence $H \models \phi$.

Equality Rules

$$\frac{A(t) \quad t = s}{A(s)} \qquad \frac{A(t) \quad s = t}{A(s)} \qquad \frac{\neg(t = t)}{x}$$

Example $s = t \vdash t = s$

1. $s = t$ (hypothesis)
2. $\neg(t = s)$ (to prove $t = s$)

Example $s = t \vdash t = s$

1. $s = t$ (hypothesis)
2. $\neg(t = s)$ (to prove $t = s$)
3. $\neg(s = s)$ (from (2), equality rule)

Example $s = t \vdash t = s$

1. $s = t$ (hypothesis)
2. $\neg(t = s)$ (to prove $t = s$)
3. $\neg(s = s)$ (from (2), equality rule)
4. — closed, by 3rd equality rule

Tableau Summary

- Tableau method is sound and complete for first order logic (this is, essentially, Gödel's completeness theorem).
- If ϕ is not satisfiable its tableau will close finitely, provided a fair sequence is used (completeness).
- If ϕ is satisfiable its tableau will never close (soundness).
- But a tableau construction may never terminate.

Go to www.menti.com

Theorem Proving for Predicate Logic. Axiomatic Proofs

Take axioms schemas for propositional logic.

Quantifier Axioms:

VIII. $(\forall x \neg A \leftrightarrow \neg \exists x A)$

IX. $(\forall x A(x) \rightarrow A(t/x))$ if t is substitutable for x in A.

X. $(\forall x(A \rightarrow B) \rightarrow (\forall x A \rightarrow \forall x B))$.

Equality Axioms:

XI. $(x = x)$

XII. $(x = y) \rightarrow (y = x)$

XIII. $((x = y) \rightarrow (t(x) = t(y/x)))$

XIV. $((x = y) \rightarrow (A(x) \rightarrow A(y/x)))$ if y is substitutable for x in A.

An instance of any of the axioms above is obtained by replacing A, B, C etc. by arbitrary formulas.

Theorem Proving for Predicate Logic. Axiomatic Proofs

Take axioms schemas for propositional logic.

Quantifier Axioms:

VIII. $(\forall x \neg A \leftrightarrow \neg \exists x A)$

IX. $(\forall x A(x) \rightarrow A(t/x))$ if t is substitutable for x in A.

X. $(\forall x(A \rightarrow B) \rightarrow (\forall x A \rightarrow \forall x B))$.

Equality Axioms:

XI. $(x = x)$

XII. $(x = y) \rightarrow (y = x)$

XIII. $((x = y) \rightarrow (t(x) = t(y/x)))$

XIV. $((x = y) \rightarrow (A(x) \rightarrow A(y/x)))$ if y is substitutable for x in A.

An instance of any of the axioms above is obtained by replacing A, B, C etc. by arbitrary formulas.

Q: Why don't we need $((x = y) \wedge (y = z)) \rightarrow (x = z)$?

Inference Rules

Modus Ponens

$$\frac{A, (A \to B)}{B}$$

Universal Generalisation

$$\frac{A(x)}{\forall x A(x)}$$

(N.B. $A(x) \to \forall x A(x)$ is not an axiom, as it is not valid. Universal generalisation says that if $A(x)$ is valid then $\forall x A(x)$ is also valid. This rule is sound.)

Proofs

A proof of ϕ is a finite sequence

$$\phi_0, \phi_1, \phi_2, \ldots, \phi_n = \phi$$

such that, for each $i \leq n$, either

- ϕ_i is an instance of one of the axioms or
- ϕ_i is obtained from ϕ_j (and maybe ϕ_k) where $j, k < i$, by an inference rule.

Write

$$\vdash \phi$$

in this case.

Proving from hypotheses

So far, this is all to do with validity over arbitrary models. If you want to find validities in a particular model, or a particularly type of model, then you can add hypotheses.
These hypotheses are formulas which are valid in the type of formula you want, and they define it.
E.g. Linearly Ordered Models
Hypotheses:

$$\forall x \forall y (x < y \vee y < x \vee x = y)$$
$$\forall x \neg (x < x)$$
$$\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)$$

Proofs with hypotheses

Let Γ be a set of hypotheses. Write

$$\Gamma \vdash \phi$$

if there is a sequence

$$\phi_0, \phi_1, \ldots, \phi_n = \phi$$

such that for each $i \leq n$ either

- ϕ_i is an axiom,
- ϕ_i is obtained from ϕ_j (ϕ_k) (some $j, k < i$) by an inference rule, or
- $\phi_i \in \Gamma$.

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg (x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg (x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg (x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg(x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)
- $\forall x \neg(x < x)$ (Hypothesis)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg(x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)
- $\forall x \neg(x < x)$ (Hypothesis)
- $((x < y \wedge y < x) \rightarrow x < x) \rightarrow (\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$
  (Ax. III)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg(x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)
- $\forall x \neg(x < x)$ (Hypothesis)
- $((x < y \wedge y < x) \rightarrow x < x) \rightarrow (\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$ (Ax. III)
- $(\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$(Modus Ponens)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg(x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)
- $\forall x \neg(x < x)$ (Hypothesis)
- $((x < y \wedge y < x) \rightarrow x < x) \rightarrow (\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$
  (Ax. III)
- $(\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$(Modus Ponens)
- $\neg(x < x)$ (Ax. IX)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg(x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)
- $\forall x \neg(x < x)$ (Hypothesis)
- $((x < y \wedge y < x) \rightarrow x < x) \rightarrow (\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$ (Ax. III)
- $(\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$(Modus Ponens)
- $\neg(x < x)$ (Ax. IX)
- $\neg(x < y \wedge y < x)$(Modus Ponens)

Example Proof using Hypotheses

Linear Order $\vdash \forall x \forall y \neg(x < y \wedge y < x)$

Proof

- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow x < z)$ (Hypothesis)
- $\forall x \forall y \forall z((x < y \wedge y < z) \rightarrow (x < z)) \rightarrow$
  $((x < y \wedge y < x) \rightarrow x < x)$(Ax. IX)
- $((x < y \wedge y < x) \rightarrow x < x)$ (Modus Ponens)
- $\forall x \neg(x < x)$ (Hypothesis)
- $((x < y \wedge y < x) \rightarrow x < x) \rightarrow (\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$ (Ax. III)
- $(\neg(x < x) \rightarrow \neg(x < y \wedge y < x))$(Modus Ponens)
- $\neg(x < x)$ (Ax. IX)
- $\neg(x < y \wedge y < x)$(Modus Ponens)
- $\forall x \forall y \neg(x < y \wedge y < x)$ (Universal Generalisation)

Deduction Theorem

Let A be a sentence.

If $\Sigma \cup \{A\} \vdash B$ then $\Sigma \vdash (A \to B)$.

Entailment

Let Γ be a set of sentences and let S be an $\mathcal{L}$-structure. Write

$$S \models \Gamma$$

if $S \models \phi$ for each $\phi \in \Gamma$ (say "S is a model of Γ).
Write

$$\Gamma \models \phi$$

if every model of Γ is a model of ϕ (i.e. $S \models \Gamma \Rightarrow S \models \phi$).

$\models$ in FOL

- $(D, I), \models_A \phi$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$ ϕ is valid over all structures

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$ ϕ is valid over all structures
- $\mathcal{K} \models \phi$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$ ϕ is valid over all structures
- $\mathcal{K} \models \phi$ ϕ is valid over all $(D, I) \in \mathcal{K}$
- $(D, I) \models \Sigma$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$ ϕ is valid over all structures
- $\mathcal{K} \models \phi$ ϕ is valid over all $(D, I) \in \mathcal{K}$
- $(D, I) \models \Sigma$ For all $\phi \in \Sigma$ we have $(D, I) \models \phi$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$ ϕ is valid over all structures
- $\mathcal{K} \models \phi$ ϕ is valid over all $(D, I) \in \mathcal{K}$
- $(D, I) \models \Sigma$ For all $\phi \in \Sigma$ we have $(D, I) \models \phi$
- $\Sigma \models \phi$

$\models$ in FOL

- $(D, I), \models_A \phi$ ϕ is true in (D, I) using A
- $(D, I) \models \phi$ ϕ is valid in (D, I)
- $\models \phi$ ϕ is valid over all structures
- $\mathcal{K} \models \phi$ ϕ is valid over all $(D, I) \in \mathcal{K}$
- $(D, I) \models \Sigma$ For all $\phi \in \Sigma$ we have $(D, I) \models \phi$
- $\Sigma \models \phi$ Σ entails ϕ, i.e. every model of Σ is a model of ϕ, i.e $(D, I) \models \Sigma \Rightarrow (D, I) \models \phi$

Strong Completeness

$$\Gamma \vdash \phi \Rightarrow \Gamma \models \phi \qquad \text{(Soundness)}$$

$$\Gamma \models \phi \Rightarrow \Gamma \vdash \phi \qquad \text{(Strong Completeness)}$$

Corollary

There is an enumeration of the valid formulas.

Recursive Languages

A language L is just a set of strings over some finite alphabet Σ.
L is recursive if there is a computer program that takes an arbitrary string $s \in \Sigma^*$ as an input and outputs

$$\begin{cases} \text{“yes”} & \text{if } s \in L \\ \text{“no”} & \text{otherwise} \end{cases}$$

The program must be guaranteed to terminate, for any $s \in \Sigma^*$.
The set of all formulas of first order logic is a recursive set (a parsing program decides if a string is a well formed formula).
The valid statements of first order logic form a language, but this language is not recursive (not decidable).

Recursively Enumerable Languages

A language L is recursively enumerable (r.e.) if there is a computer program that outputs strings from L, only strings from L, and will eventually output any given string from L.
The valid statements of first-order logic form a recursively enumerable language.

First Order Logic is r.e.

Let $\phi_0, \phi_1, \ldots$ be an enumeration of all formulas.

For $(i = 0, i++, \text{forever})$
{ Start new tableau T_i with $\neg\phi_i$ at root;
For each $j < i$
{ expand T_j once, using a fair schedule;
If T_j becomes closed, output "ϕ_j is valid";
}
}

Note: for any formula ϕ_k, If ϕ_k is not valid then T_k will never close (by soundness).
if ϕ_k is valid then eventually T_k will close and the program will output "ϕ_k is valid" (this follows by completeness, though you do not know how long this will take).
So the program only outputs valid formulas, and any given valid formula will eventually get output.

Recursive and r.e. languages

The set of formulas of first-order logic is a recursive set.
The set of valid formulas of FOL is not recursive, but it is r.e.
The set of true statements of arithmetic is not even r.e.
This last statement is Gödel's incompleteness theorem.

Proving from Assumptions

Suppose you want to prove that ϕ is valid in a particular model, or type of model (e. g. linear order).
Write down assumptions Σ that define this type of model. E.g.

$$\Sigma = \left\{ \begin{array}{l} \forall x \forall y (x = y \vee x < y \vee y < x), \\ \forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z), \\ \forall x \neg (x < x) \end{array} \right\}$$

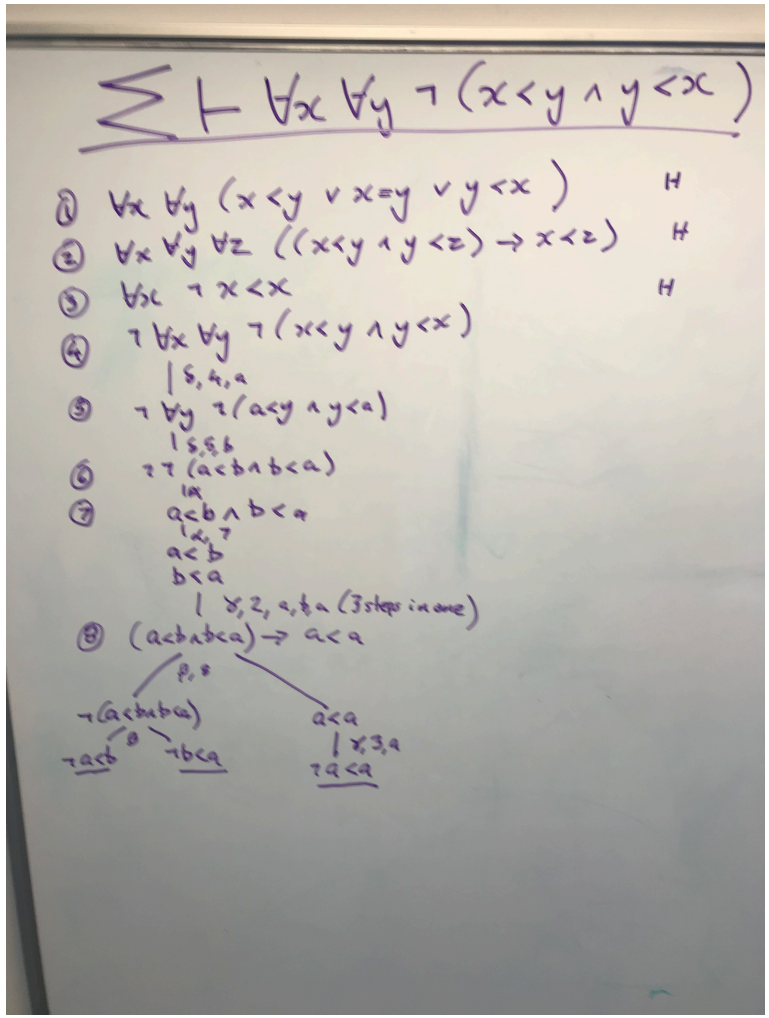
New rule: you can add any assumption in Σ at leaf of tableau at any time. A proof of ϕ using Σ is a closed tableau for $\neg\phi$, but you can use assumptions to help you close the tableau. Write

$$\Sigma \vdash \phi$$

Tableau with assumptions example (TAE)

$$\Sigma = \{\forall x \neg(x < x),\ \forall x \forall y (x < y \vee y < x \vee x = y), \forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)\}$$

$$\Sigma \vdash \forall x \forall y \neg(x < y \wedge y < x)$$


$\Sigma \vdash \forall x \forall y \neg(x<y \wedge y<x)$
① $\forall x \forall y (x<y \vee x=y \vee y<x)$ H
② $\forall x \forall y \forall z ((x<y \wedge y<z) \rightarrow x<z)$ H
③ $\forall x \neg x<x$ H
④ $\neg \forall x \forall y \neg(x<y \wedge y<x)$
| 5, 4, a
⑤ $\neg \forall y \neg(a<y \wedge y<a)$
| 5, 5, b
⑥ $\neg\neg(a<b \wedge b<a)$
| ∧
⑦ $a<b \wedge b<a$
| ∧, 7
$a<b$
$b<a$
| 8, 2, a, b, a (3 steps in one)
⑧ $(a<b \wedge b<a) \rightarrow a<a$
β, 8
$\neg(a<b \wedge b<a)$
$a<a$
θ
$\neg a<b$
$\neg b<a$
| ∀, 3, a
$\neg a<a$

Compactness

If

$$\Sigma \vdash \phi$$

then

$$\Sigma_0 \vdash \phi$$

for some finite subset Σ_0 of Σ.

Inconsistency, Compactness, Completeness.

Σ is inconsistent if

$$\Sigma \vdash (p \wedge \neg p)$$

If Σ is inconsistent then, by compactness, Σ_0 is inconsistent, for some finite subset Σ_0 of Σ.
By strong completeness theorem, every consistent set has a model.
Hence, compactness says that if every finite subset of Σ has a model then there is a model for the whole of Σ.

Problem

Let E be a binary relation denoting the edges of a graph, let $=$ denote equality. Write down a first order formula $\phi(x, y)$ with two free variables x, y, meaning

- There is a path of length 1 from node x to node y,
- There is a path of length 2 from x to y
- There is a path of length 3 from x to y
- There is a path of length k from x to y, where $k \geq 1$ is fixed.

First Order Logic Cannot Define Connectedness

Language

$$\begin{aligned} C &= \{c, d\} \\ F &= \emptyset \\ P &= \{=, E\} \text{ (both binary)} \end{aligned}$$

Suppose, for contradiction, that

$$G \models \Sigma \iff G \text{ is connected}$$

Let

$$\begin{aligned} \phi_1(x, y) &= E(x, y) \\ \phi_{n+1}(x, y) &= \exists z(\phi_n(x, z) \wedge E(z, y)) \end{aligned}$$

"there is a path of length $n + 1$ from x to y".
Consider

$$\Sigma \cup \{\neg\phi_1(c, d), \neg\phi_2(c, d), \ldots\}$$

Every finite subset has a model (what model?). By compactness, the whole set has a model, say G,

$$G \models \Sigma \cup \{\neg\phi_n(c, d) : n = 1, 2, 3, \ldots\}$$

G is therefore connected (since a model of Σ), but there is no path from c to d — a contradiction.

Compactness theorem and non-standard analysis

Let

$$\Sigma = \{\text{all valid statements about } \mathbb{N}\}$$

in a language with constants $0, 1, 2, \ldots$ functions $+, \times$ and predicate $=$.
E.g. $2 + 2 = 4 \in \Sigma$.
Also $\forall x \forall y (x \times y = y \times x) \quad \in \Sigma$.
Let c be another constant symbol.
Every finite subset of

$$\Sigma^+ = \Sigma \cup \{c \neq 0, c \neq 1, c \neq 2, \ldots, c \neq n, \ldots\}$$

has a model (what model?).
Therefore Σ^+ has a model.

Non-standard real analysis

Let L be similar but with a constant for every real number. Let

$$\Sigma = \{\text{all valid statements about } \mathbb{R}\}$$

and

$$\Sigma^+ = \Sigma \cup \{\alpha > r : r \in \mathbb{R}\}$$

Every finite subset of Σ^+ has a model (just interpret α as a sufficiently big real number), therefore Σ^+ has a model M. Then $[\alpha]^M$ is an "infinitely big" real number and $[\frac{1}{\alpha}]^M$ is and "infinitesimally small" positive real number.
Can do calculus perfectly rigorously in this way. Can show that

$$\forall x((|x| < r) \to (x = St(x) + Inf(x)))$$

where r is a constant for any positive real, $St(x)$ is a "standard real" and $Inf(x)$ is an "infinitesimal real". Then let

$$f'(x) = St\left(\frac{f(x+\delta x) - f(x)}{\delta x}\right)$$

where x is any standard real and δx is any infinitesimal, provided this does not depend on the choice of δx.

Finiteness

No first-order theory defines the class of all finite structures.

Finiteness

No first-order theory defines the class of all finite structures.
Suppose for contradiction that $(D, I) \models F$ iff D is finite.

Finiteness

No first-order theory defines the class of all finite structures.
Suppose for contradiction that $(D, I) \models F$ iff D is finite.
Let C be an infinite set of constants.
Let $\Sigma = \{\neg(c = d) : c \neq d \in C\}$.

Finiteness

No first-order theory defines the class of all finite structures.
Suppose for contradiction that $(D, I) \models F$ iff D is finite.
Let C be an infinite set of constants.
Let $\Sigma = \{\neg(c = d) : c \neq d \in C\}$.
$\Sigma \cup F$ is consistent (why?)

Finiteness

No first-order theory defines the class of all finite structures.
Suppose for contradiction that $(D, I) \models F$ iff D is finite.
Let C be an infinite set of constants.
Let $\Sigma = \{\neg(c = d) : c \neq d \in C\}$.
$\Sigma \cup F$ is consistent (why?)
By strong completeness $\Sigma \cup F$ has a model — finite and infinite — contradiction.

Finiteness

No first-order theory defines the class of all finite structures.
Suppose for contradiction that $(D, I) \models F$ iff D is finite.
Let C be an infinite set of constants.
Let $\Sigma = \{\neg(c = d) : c \neq d \in C\}$.
$\Sigma \cup F$ is consistent (why?)
By strong completeness $\Sigma \cup F$ has a model — finite and infinite — contradiction.
Therefore no such theory F exists.

Summary

- Σ is consistent iff Σ has a model (soundness and strong completeness)
- $\Sigma \vdash \phi \iff \exists \Sigma_0 \subseteq_f \Sigma,\ \Sigma_0 \vdash \phi$ (compactness)
- Σ has a model iff for all $\forall \Sigma_0 (\Sigma_0 \subseteq_f \Sigma \Rightarrow \Sigma_0$ has a model)
- No first order theory defines connected graphs
- $\mathbb{N}$ has a non-standard model.