Logic for Applications

Yuxin Deng

Shanghai Jiao Tong University

http://basics.sjtu.edu.cn/∼yuxin/

June 4, 2015

Instructor

Instructor: Yuxin Deng

Office: Room 3-327, SEIEE building

Email: deng-yx@cs.sjtu.edu.cn

Homepage: http://basics.sjtu.edu.cn/∼yuxin

Teaching assistant: TBA

将来无论你是做科学家，是做政治家，还是做一个成功的商人，都需要有系统的逻辑训练。

----丘成桐

Textbook and references

Textbook

Anil Nerode and Richard A. Shore. Logic for Applications (2nd Edition), Springer, 1997.

References

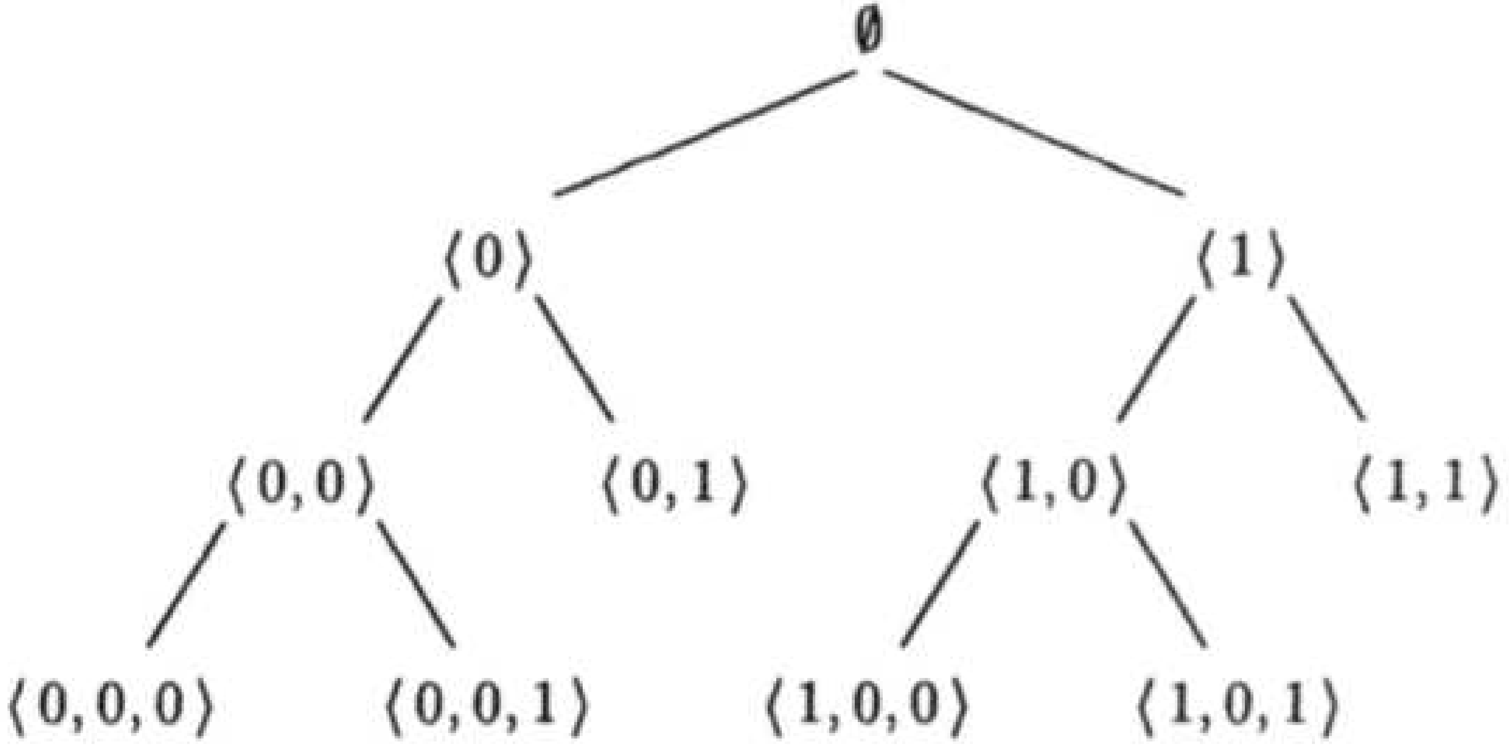
- Melvin Fitting. First-Order Logic and Automated Theorem Proving (2nd Edition), Springer 1995.
- http://www.cs.rice.edu/∼vardi/comp409/index.html

Grade

- Attendance records and quizzes: 10%
- Assignments: 20%
- Final exam: 70%

Chapter I. Propositional Logic

1 Orders and Trees



Orders

Definition 1.1 (i) A partial order is a set S with a binary relation called "less than", and written $<$, on S which is transitive and irreflexive:

$$x < y \text{ and } y < z \ \Rightarrow \ x < z \text{ and}$$

$$x \text{ is not less than } x \text{ for any } x.$$

(ii) The partial order $<$ is a linear order (or simply an order) if it also satisfies the trichotomy law:

$$x < y \text{ or } x = y \text{ or } y < x.$$

(iii) A linear order S is well ordered if every nonempty subset A of S has a least element, i.e., there is an $x \in A$ such that for no $y \in A$ is $y < x$. This property easily implies that the order has no infinite descending chain, i.e., there is no set of elements $x_0, x_1, \ldots$ of S such that

$$\ldots < x_2 < x_1 < x_0$$

(iv) We use the usual notational conventions for orderings:

$$x \leq y \;\Leftrightarrow\; x < y \text{ or } x = y.$$

$$x > y \;\Leftrightarrow\; y < x.$$

Note: We only consider finite or countably infinite sets. We may assume that we have a listing of a set as either $\{a_i \mid i < n\}$ for some $n \in \mathcal{N}$ or as $\{a_i \mid i \in \mathcal{N}\}$.

Trees (1/2)

Definition 1.2 A tree is a set T (whose elements are called nodes) partially ordered by $<_T$, with a unique least element called the *root*, in which the predecessors of every node are well ordered by $<_T$.

A path on a tree T is a maximal linearly ordered subset of T.

Trees (2/2)

Definition 1.3 (i) The levels of a tree T are defined by induction. The 0^{th} level of T consists precisely of the root of T. The $k+1^{st}$ level of T consists of the immediate successors of the nodes on the k^{th} level of T. (We say that x is a successor of y in T if $y <_T x$. It is an immediate successor if $y <_T x$ and there is no z such that $y <_T z <_T x$.)

(ii) The depth of a tree T is the maximum n such that there is a node of level n in T. If there are nodes of level n for every natural number n, we say the depth of T is infinite or ω.

(iii) If each node has at most n immediate successors, the tree is n-ary or n-branching. If each node has finitely many immediate successors, we say that the tree is finitely branching. A node with no successors is called a leaf or a terminal node.

König's lemma

Theorem 1.4 [König's lemma] If a finitely branching tree T is infinite, it has an infinite path.

Proof: Define the sequence $x_0, x_1, ...,$ constituting a path P on T by induction. The first element x_0 is the root of T. It has infinitely many successors by the assumption that T is infinite. Suppose that we have defined the first n elements of P to be $x_0, x_1, ..., x_{n-1}$ on levels $0, 1, ..., n-1$ of T, respectively, so that each x_i has infinitely many successors in T. By hypothesis, x_{n-1} has only finitely many immediate successors. As it has infinitely many successors all together, (at least) one of its immediate successors, say y, also has infinitely many successors. We now set $x_n = y$. x_n is on level n of T and has infinitely many successors in T and so we may continue our definition of P. □

Labeled trees

Frequently it is just the shape of the tree that is important and not the nodes themselves. To facilitate talking about the arrangement of different materials into the same shape and to allow the same component to be used at different places in this assemblage, we attach labels to the nodes of the tree.

Definition 1.5 A labeled tree T is a tree T with a function (the labeling function) that associates some object with every node. This object is called the label of the node.

Lexicographic ordering

Consider finite sequences of 0's and 1's. Consider a sequence or string σ of length n as a map from $\{0, 1, ..., n-1\}$ into $\{0, 1\}$. The tree ordering $<_T$ is given by extension as functions $\sigma < \tau \Leftrightarrow \sigma \subset \tau$, i.e., $\sigma(n) = \tau(n)$ for every n at which σ is defined.

Lexicographic ordering on sequences: For two sequences σ and τ we say that $\sigma <_L \tau$ if $\sigma \subset \tau$ or if $\sigma(n)$, the n^{th} entry in σ, is less than $\tau(n)$ where n is the first entry at which the sequences differ.

Left to right ordering

Left to right ordering: First use $<_L$ as a linear order on each level of the tree, then extend it to a linear ordering (also designated $<_T$) of all the nodes of the tree. Given two nodes x and y, we say that $x <_L y$ if $x <_T y$. If x and y are incomparable in the tree ordering, we find the largest predecessors x' and y' of x and y, respectively, that are on the same level of T and let $x <_L y$ iff $x' <_L y'$. Any such total ordering of the nodes of a tree is also referred to as the lexicographic ordering of the nodes.

2 Propositions, Connectives and Truth Tables

Propositions are just statements and propositional logic describes and studies the ways in which statements are combined to form other statements.

The syntactic part of logic deals with statements as just strings (i.e., sequences) of symbols. The semantic part of logic (semantics) ascribes meaning to the symbols in various ways.

Connectives are operations that combine propositions to form new ones.

Connectives

Formal symbols for some commonly used connectives:

$\vee$	or	(disjunction)
$\wedge$	and	(conjunction)
$\neg$	not	(negation)
$\rightarrow$	implies	(conditional)
$\leftrightarrow$	if and only if	(biconditional)

Propositional logic

The language of propositional logic consists of the following symbols:

(i) Connectives: $\vee$, $\wedge$, $\neg$, $\rightarrow$, $\leftrightarrow$

(ii) Parentheses:), (

(iii) Propositional letters: $A, A_1, A_2, \ldots, B, B_1, B_2 \ldots, \ldots$.

Propositions

Definition 2.1 [Propositions]

(i) Propositional letters are propositions.

(ii) If α and β are propositions, then $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\neg\alpha)$, $(\alpha \to \beta)$ and $(\alpha \leftrightarrow \beta)$ are propositions.

(iii) A string of symbols is a proposition if and only if it can be obtained by starting with propositional letters (i) and repeatedly applying (ii).

E.g. $(A \lor B)$, C, $((A \land B) \to C)$, $(\neg(A \land B) \to C)$ are propositions while $A \land \neg$, $(A \lor B$, $(\neg \to A)$ are not.

Formation tree

Definition 2.2 A formation tree is a finite tree T of binary sequences (with root $\emptyset$ and a left to right ordering given by the ordinary lexicographic ordering of sequences) whose nodes are all labeled with propositions. The labeling satisfies the following conditions:

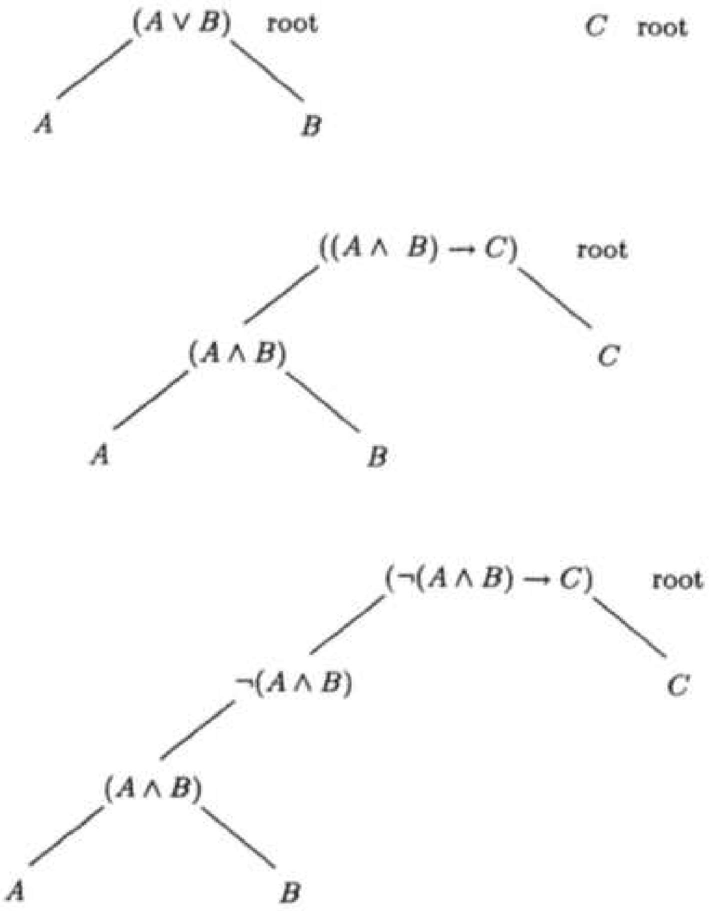
(i) The leaves are labeled with propositional letters.

(ii) If a node σ is labeled with a proposition of the form $(\alpha \wedge \beta)$, $(\alpha \vee \beta), (\alpha \rightarrow \beta)$ or $(\alpha \leftrightarrow \beta)$, its immediate successors, $\sigma \cdot 0$ and $\sigma \cdot 1$, are labeled with α and β (in that order).

(iii) If a node σ is labeled with a proposition of the form $\neg\alpha$, its unique immediate successor, $\sigma \cdot 0$, is labeled with α.

The formation tree T represents or is associated with the proposition with which its root is labeled.

Example: formation tree



Example: uniqueness of formation tree

Theorem 2.3 Each proposition has a unique formation tree associated with it.

Proof: First show by induction that each proposition has a formation tree associated with it.

Base case: for a propositional letter A, define a tree consisting of $\emptyset$ labeled with A.

Inductive case: for the proposition $\alpha \to \beta$, by induction there are formation trees T_α and T_β. The formation tree $T_{(\alpha\to\beta)}$ for $(\alpha \to \beta)$ has as its root $\emptyset$ labeled with $(\alpha \to \beta)$. The other nodes are all sequences $0{\cdot}\sigma$ for every σ on T_α and $1{\cdot}\tau$ for every τ on T_β. The labels are the same as they were in the original trees. Other cases are similar.

Then show uniqueness. For propositional letters this is clear. For the inductive case caonsider $\alpha \to \beta$. Note that its root $\emptyset$ must be labeled with $\alpha \to \beta$. Every node on must be of the form $0{\cdot}\sigma$ or $1{\cdot}\tau$, respectively, for some binary sequence σ. For $n = 0, 1$ let $T_n = \{\sigma \mid n{\cdot}\sigma \in T\}$ have the standard ordering and be labeled as in T. It is clear that T_0 is a formation tree for α and T_1 for β. They are unique by induction and so T has been uniquely determined as required. □

Depth and support of a proposition

Definition 2.4 (i) The depth of a proposition is the depth of the associated formation tree.

(ii) The support of a proposition is the set of propositional letters that occur as labels of the leaves of the associated formation tree.

Closure operation

A set S is closed under a single (for example n-ary) operation $f(s_1, ..., s_n)$ iff for every $s_1, ..., s_n \in S,\ f(s_1, ..., s_n) \in S$. The closure of a set S under (all) the operations in a set T is the smallest set C such that 1. $S \subseteq C$ aand 2. if $f \in T$ is n-ary and $s_l, ., s_n \in C$, then $f(s_1, ..., s_n) \in C$.

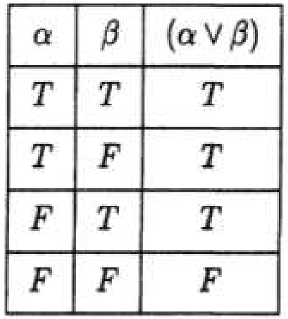
$C = \cap\{D \mid S \subseteq D \text{ \& } D \text{ is closed under the operation of } T\}$

Obviously $S \subseteq C$. Then show C is closed under the operations of T. So C is the smallest such set as it is contained in every set $D \supseteq S$ that is closed under the operations of T.

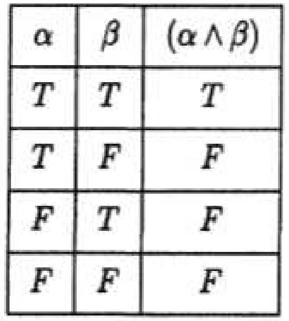
The set of propositions is the closure of the set of propositional letters under the operations $\wedge, \vee, \neg, \rightarrow$ and $\leftrightarrow$.

Truth tables

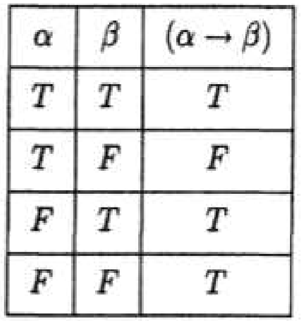
For the semantics, we view the meaning of a propositional letter is simply its truth value, that is, its truth or falsity. Each proposition then has a unique truth value (T, for true or F, for false). The truth value of a compound proposition is determined from the truth values of its parts in accordance with the truth tables below:



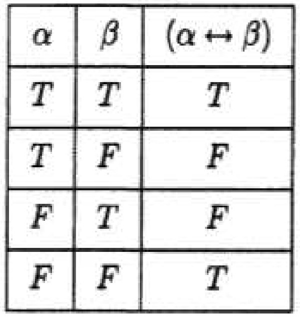
α	β	$(\alpha \vee \beta)$
T	T	T
T	F	T
F	T	T
F	F	F



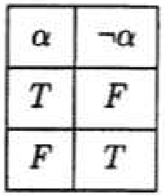
α	β	$(\alpha \wedge \beta)$
T	T	T
T	F	F
F	T	F
F	F	F



α	β	$(\alpha \rightarrow \beta)$
T	T	T
T	F	F
F	T	T
F	F	T

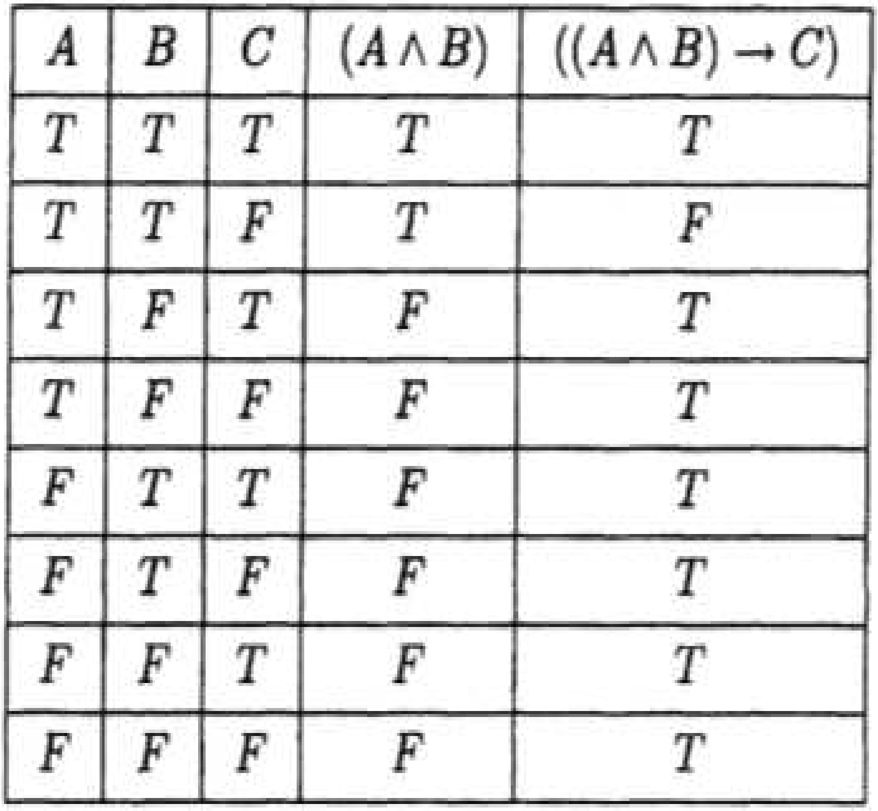


α	β	$(\alpha \leftrightarrow \beta)$
T	T	T
T	F	F
F	T	F
F	F	T



α	$\neg\alpha$
T	F
F	T

Example: Truth tables



A	B	C	$(A \wedge B)$	$((A \wedge B) \to C)$
T	T	T	T	T
T	T	F	T	F
T	F	T	F	T
T	F	F	F	T
F	T	T	F	T
F	T	F	F	T
F	F	T	F	T
F	F	F	F	T

The column for $(A \wedge B)$ is auxiliary and could be eliminated. The result would be the abbreviated truth table for $(A \wedge B) \to C)$.

Truth tables

In general, an n-ary connective is any function σ that assigns a proposition $\sigma(A_1 ..., A_n)$ to every n-tuple of propositions $A_1, ..., A_n$. An n-ary connective is truth junctional if the truth value for $\sigma(A_1, ..., A_n)$ is uniquely determined by the truth values for $A_1, ..., A_n$. Our five connectives are truth functional since their meaning was defined by truth tables. A connective like "because" is not. For let A symbolize "I had prune juice for breakfast" and B "there was an earthquake at noon". Even in the event that both A and B have truth values T it is at least debatable whether $(B\ because\ A)$ should have truth value T. An n-ary connective that is truth functional can be completely described by means of a truth table.

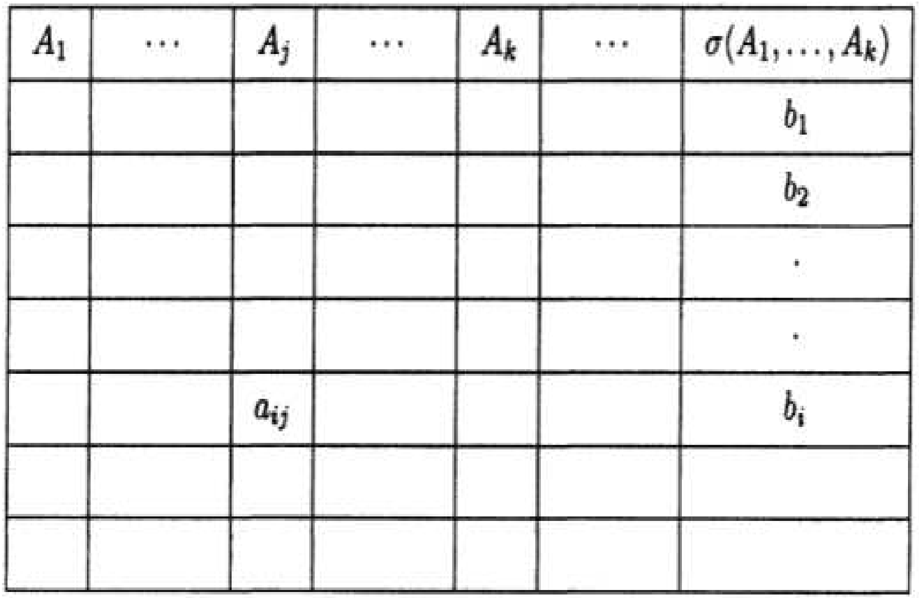
Conversely, two distinct abbreviated truth tables (with the conventional listing of truth values for $A_1, ..., A_n$) correspond to distinct truth functional connectives. By counting we see there are 2^{2^n} distinct n-ary truth functional connectives.

Adequacy

Definition 2.5 A set S of truth functional connectives is adequate if, given any truth functional connective σ, we can find a proposition built up from the connectives in S with the same abbreviated truth table as σ.

Theorem 2.6 [Adequacy] $\{\neg, \wedge, \vee\}$ is adequate.

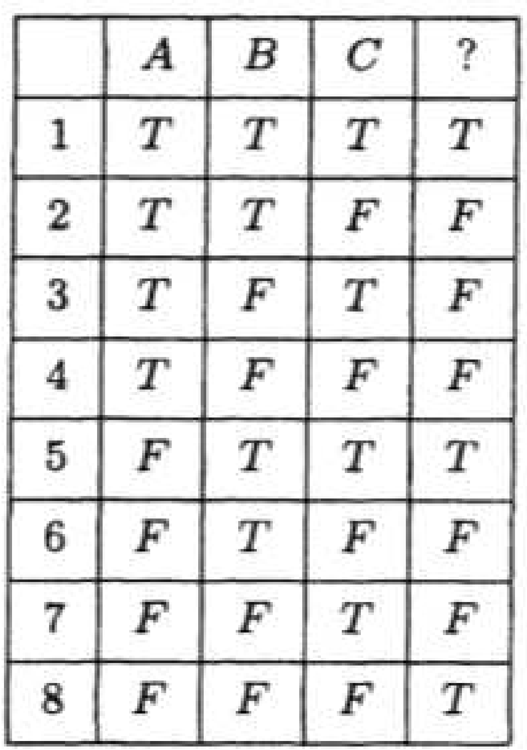
Proof: Let $A_1, ..., A_k$ be distinct propositional letters and let a_{ij} denote the entry (T or F) corresponding to the i^{th} row and j^{th} column of the truth table for $\sigma(A_1, , A_k)$. Suppose that at least one T appears in the last column.



A_1	$\cdots$	A_j	$\cdots$	A_k	$\cdots$	$\sigma(A_1,\ldots,A_k)$
						b_1
						b_2
						.
						.
		a_{ij}				b_i

For any proposition α, let α^T be α and α^F be $(\neg\alpha)$. For the i^{th} row denote the conjunction $(A_1^{a_{i1}} \wedge ... A_k^{a_{ik}})$ by a_i. Let $i_1, ..., i_m$ be the rows with a T in the last column. The desired proposition is the disjunction $(a_{i_1} \vee ... \vee a_{i_m})$ The proof that this proposition has the given truth table is left as an exercise. □

Example: From truth table to proposition



	A	B	C	?
1	T	T	T	T
2	T	T	F	F
3	T	F	T	F
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	F
8	F	F	F	T

Take the disjunction of the propositions we obtain for all relevant rows (rows 1, 5, 8 in this case), we obtain

$$(A \wedge B \wedge C) \vee ((\neg A) \wedge B \wedge C) \vee ((\neg A) \wedge (\neg B) \wedge (\neg C))$$

DNF and CNF

From any proposition we can construct its truth table, then find a disjunctive normal form (DNF) or a conjunctive normal form (CNF) .

Adequacy

Corollary 2.7 $\{\neg,\ \vee\}$ is adequate.

Proof: Note that $(A_1 \wedge A_2)$ has the same truth table as $\neg((\neg(A_1)) \vee (\neg(A_2)))$. Thus, given any proposition α we can find a DNF of α and then eliminate any use of $\wedge$ by this substitution. The resulting proposition will still have the same truth table. □

The sets $\{\neg,\ \wedge\}$ and $\{\neg,\ \rightarrow\}$ are also adequate.

Exercises

1. Show that each proposition has a CNF.
2. Exercise 8 in page 22: the binary connective *Sheffer stroke* $\alpha|\beta$ ("not both ... and") is adequate.
3. Exercise 9 in page 22: joint denial $\alpha \downarrow \beta$ (neither α nor β) is also adequate.
4. Exercise 12: the set of connectives $\{\vee, \rightarrow, \leftrightarrow\}$ is not adequate.

3 Truth Assignments and Valuations

Definition 3.1 A truth assignment $\mathcal{A}$ is a function that assigns to each propositional letter A a unique truth value $\mathcal{A}(A) \in \{T, F\}$.

Definition 3.2 A truth valuation $\mathcal{V}$ is a function that assigns to each proposition α a unique truth value $\mathcal{V}(\alpha)$ so that its value on a compound proposition (that is, one with a connective) is determined in accordance with the appropriate truth tables.

E.g. $\mathcal{V}((\neg\alpha)) = T$ iff $\mathcal{V}(\alpha) = F$ and $\mathcal{V}((\alpha \vee \beta)) = T$ iff $\mathcal{V}(\alpha) = T$ or $\mathcal{V}(\beta) = T$. We say that $\mathcal{V}$ makes α true if $\mathcal{V}(\alpha) = T$.

Unique Truth Valuation

Theorem 3.3 Given a truth assignment $\mathcal{A}$ there is a unique truth valuation $\mathcal{V}$ such that $\mathcal{V}(\alpha) = \mathcal{A}(\alpha)$ for every propositional letter α.

Proof: Given a truth assignment $\mathcal{A}$, define (by induction on the depth of the associated formation tree) a valuation $\mathcal{V}$ on all propositions by first setting $\mathcal{V}(\alpha) = \mathcal{A}(\alpha)$ for all propositional letters α. This takes care of all formation trees (propositions) of depth 0. Assuming that $\mathcal{V}$ has been defined on all propositions with depth at most n, the inductive steps are simply given by the truth tables associated with each connective. E.g. $\mathcal{V}((\alpha \to \beta))$ is defined to be F iff $\mathcal{V}(\alpha) = T$ and $\mathcal{V}(\beta) = F$.

Clearly $\mathcal{V}$ has been defined so as to be a valuation and it does extend $\mathcal{A}$. It remains to show that any two valuations $\mathcal{V}_1, \mathcal{V}_2$ both extending $\mathcal{A}$ must coincide. We prove this by induction on the depth of propositions:

(i) $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$ for all propositional letters α (depth 0) since $\mathcal{V}_1, \mathcal{V}_2$ both extend $\mathcal{A}$.

(ii) Suppose $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$ for all propositions α of depth at most n and that α and β have depth at most n. Thus, $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$ and $\mathcal{V}_1(\beta) = \mathcal{V}_2(\beta)$ by induction. $\mathcal{V}_1((\alpha \wedge \beta))$ and $\mathcal{V}_2((\alpha \wedge \beta))$ are then both given by the truth table for $\wedge$ and so are equal. The same argument works for all the other connectives and so $\mathcal{V}_1$ and $\mathcal{V}_2$ agree on every proposition. □

Corollary 3.4 If $\mathcal{V}_1$ and $\mathcal{V}_2$ are two valuations that agree on the support of α, the finite set of propositional letters used in the construction of the proposition α, then $\mathcal{V}_1(\alpha) = \mathcal{V}_2(\alpha)$.

Tautology

Definition 3.5 A proposition σ of propositional logic is said to be valid if for any valuation $\mathcal{V},\ \mathcal{V}(\alpha) = T$. Such a proposition is also called a tautology.

Definition 3.6 Two propositions α and β such that, for every valuation $\mathcal{V},\ \mathcal{V}(\alpha) = \mathcal{V}(\beta)$ are called logically equivalent. We denote this by $\alpha \equiv \beta$.

Example: Logical Equivalence

(i) $(A \vee (\neg A))$ and $(((A \rightarrow B) \rightarrow A) \rightarrow A)$ (Law of the excluded middle and Peirce's law) are tautologies.

(ii) For any proposition α and any DNF β of α, $\alpha \equiv \beta$.

(iii) Rephrasing the adequacy theorem: given any proposition α, we can find a β that uses only $\neg$, $\vee$, $\wedge$ and such that $\alpha \equiv \beta$.

Consequence

Definition 3.7 Let Σ be a (possibly infinite) set of propositions. We say that σ is a consequence of Σ (and write $\Sigma \models \sigma$) if, for any valuation $\mathcal{V}$,

$(\mathcal{V}(\tau) = T$ for all $\tau \in \Sigma) \Rightarrow \mathcal{V}(\sigma) = T$.

Note that, if Σ is empty, $\Sigma \models \sigma$ (or just $\models \sigma$) iff σ is valid. We also write this as $\models \sigma$. This definition gives a semantic notion of consequence.

Model

Definition 3.8 A valuation $\mathcal{V}$ is a model of Σ if $\mathcal{V}(\sigma) = T$ for every $\sigma \in \Sigma$. We denote by $\mathcal{M}(\Sigma)$ the set of all models of Σ.

Proposition 3.9 Let $\Sigma, \Sigma_1, \Sigma_2$ be sets of propositions. Let Cn denote the set of consequences of Σ and $Taut$ the set of all tautologies.

(i) $\Sigma_1 \subseteq \Sigma_2 \ \Rightarrow \ Cn(\Sigma_1) \subseteq Cn(\Sigma_2)$

(ii) $\Sigma \subseteq Cn(\Sigma)$

(iii) $Taut \subseteq Cn(\Sigma)$ for all Σ

(iv) $Cn(\Sigma) = Cn(Cn(\Sigma))$

(v) $\Sigma_1 \subseteq \Sigma_2 \ \Rightarrow \ \mathcal{M}(\Sigma_2) \subseteq \mathcal{M}(\Sigma_1)$

(vi) $Cn(\Sigma) = \{\sigma \mid \mathcal{V}(\sigma) = T \text{ for all } \mathcal{V} \in \mathcal{M}(\Sigma)\}$

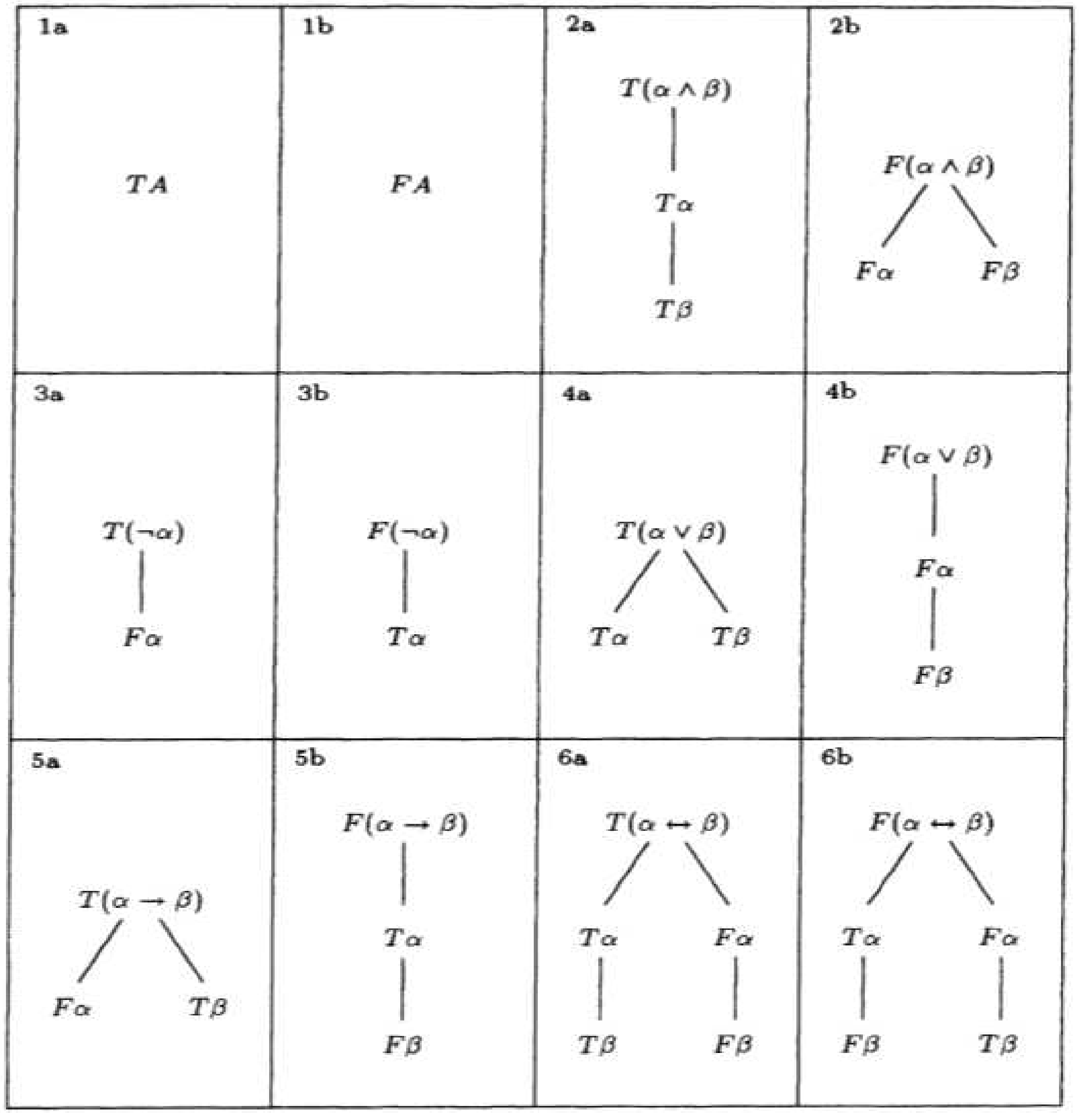
(vii) $\sigma \in Cn(\{\sigma_1, ..., \sigma_n\}) \ \Leftrightarrow \ \sigma_1 \to (\sigma_2 ... \to (\sigma_n \to \sigma)...) \in Taut$

4 Tableau Proofs in Propositional Calculus

Tableaux are labeled binary trees for representing proofs. The labels on the trees are signed propositions, i.e., a proposition preceded by either a T or an F (indicating an assumed truth value for the proposition). The labels of the nodes are the entries of the tableau.

Formally, we define (or describe how to build) tableaux for propositions inductively by first specifying certain (labeled binary) trees as tableaux (the so-called atomic tableaux) and then giving a development rule defining tableaux for compound propositions from tableaux for simple propositions.

Atomic Tableaux



Tableaux

Definition 4.1 A finite tableau is a binary tree, labeled with signed propositions called entries, that satisfies the following inductive definition:

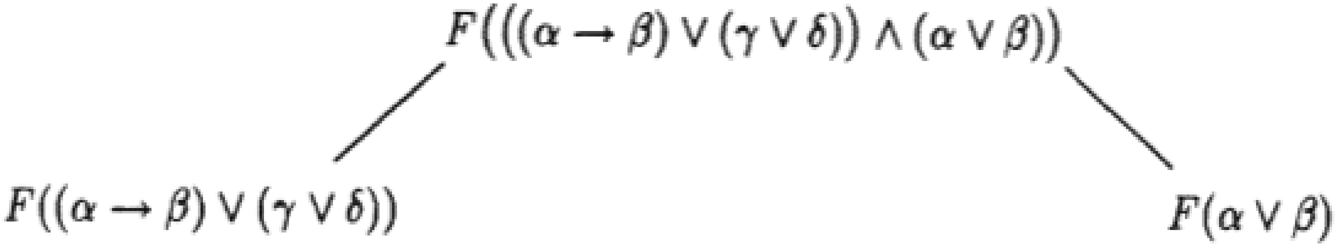
(i) All atomic tableaux are finite tableaux.

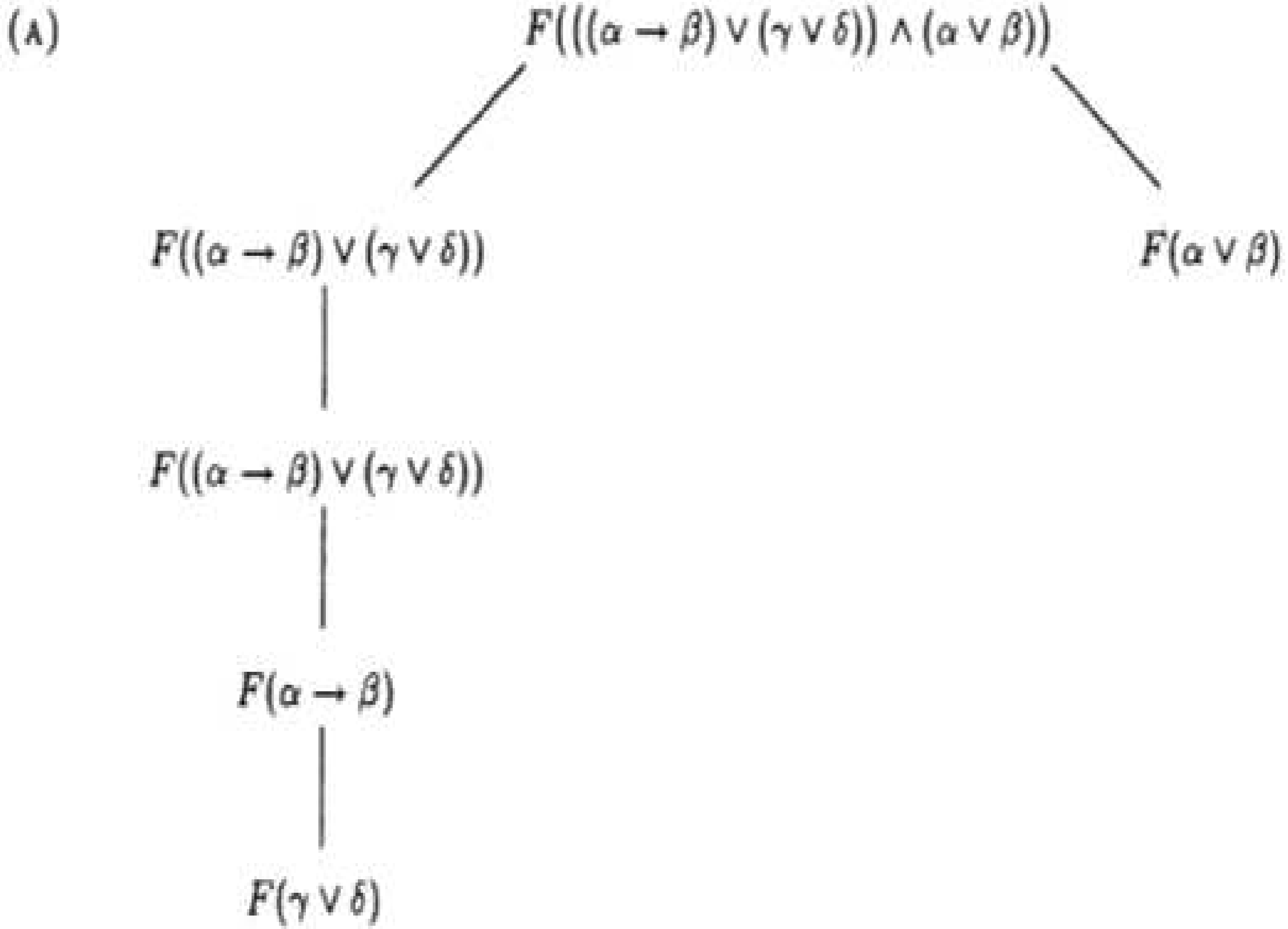
(ii) If τ is a finite tableau, P a path on τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining the unique atomic tableau with root entry E to τ at the end of the path P, then τ' is also a finite tableau.

If $\tau_0, \tau_1, ..., \tau_n, ...$ is a (finite or infinite) sequence of finite tableaux such that, for each $n \geq 0$, τ_{n+1} is constructed from τ_n by an application of (ii), then $\tau = \cup\tau_n$ is a tableau.

Example: Tableaux

We wish to begin a tableau with the signed proposition $F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta)$.


$F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$
$F((\alpha \to \beta) \vee (\gamma \vee \delta))$
$F(\alpha \vee \beta)$



(∧)

$F(((\alpha \rightarrow \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$

$F((\alpha \rightarrow \beta) \vee (\gamma \vee \delta))$

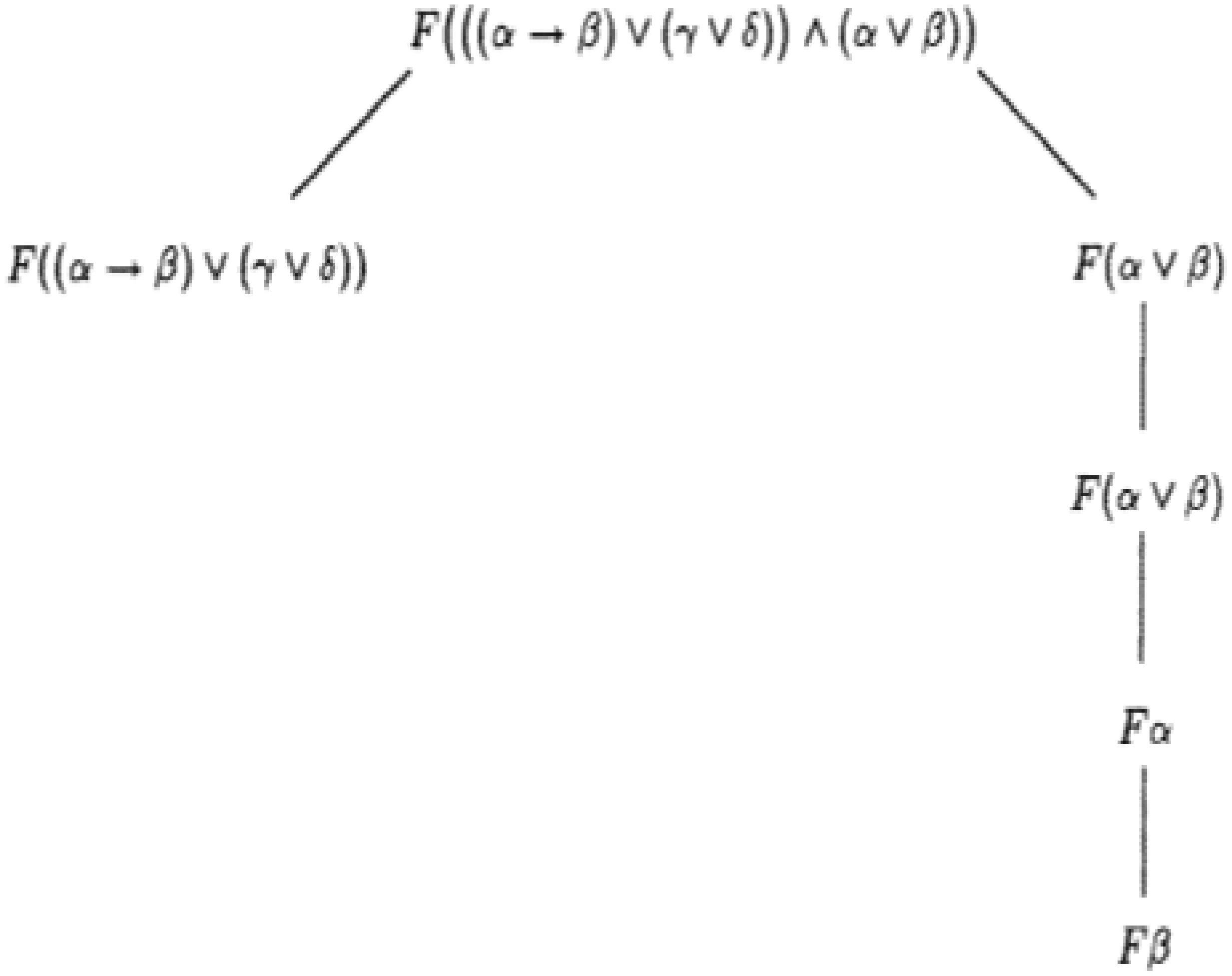
$F(\alpha \vee \beta)$

$F((\alpha \rightarrow \beta) \vee (\gamma \vee \delta))$

$F(\alpha \rightarrow \beta)$

$F(\gamma \vee \delta)$

(B)



$F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$

$F((\alpha \to \beta) \vee (\gamma \vee \delta))$

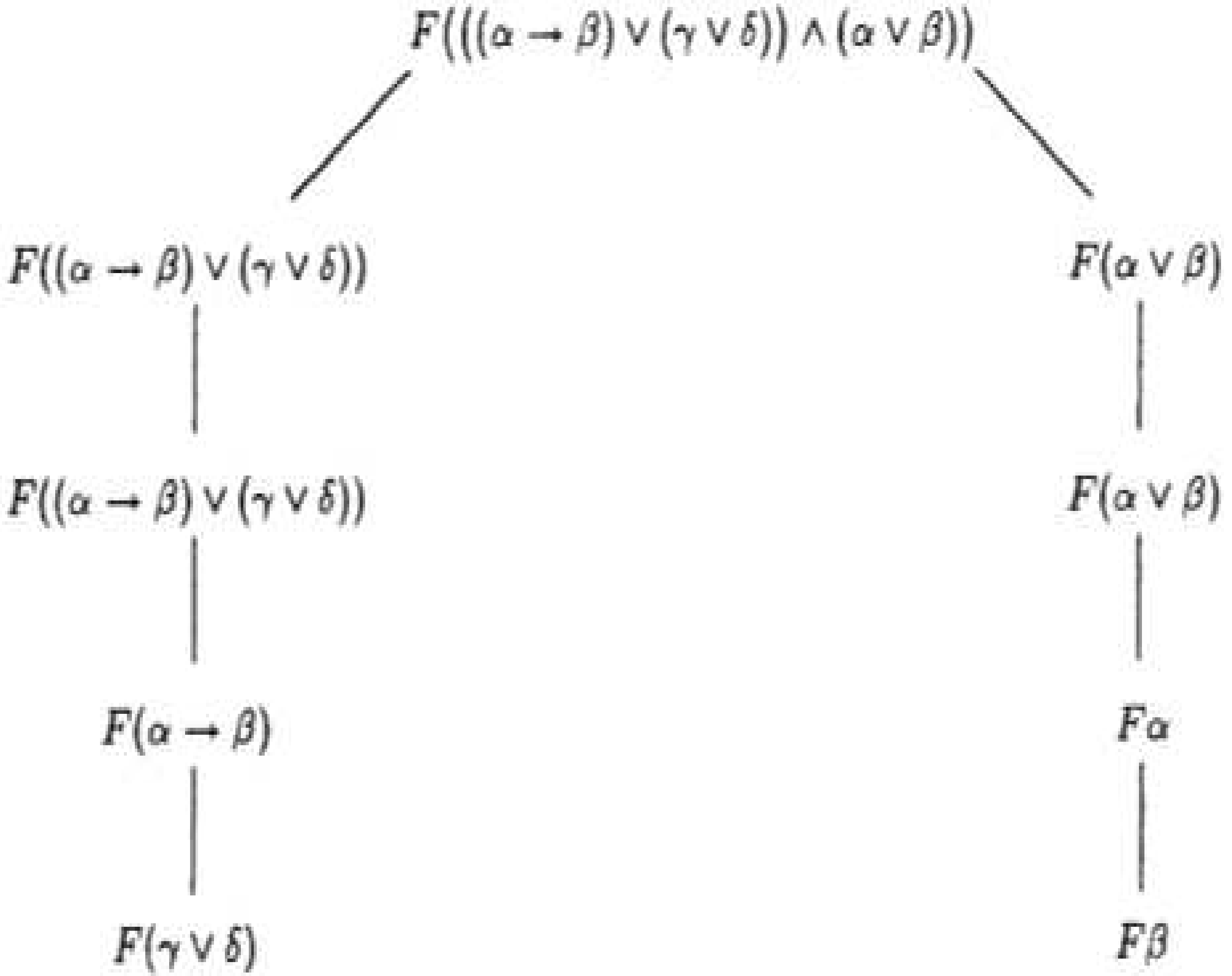
$F(\alpha \vee \beta)$

$F(\alpha \vee \beta)$

$F\alpha$

$F\beta$

(c)



$F(((\alpha \to \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$

$F((\alpha \to \beta) \vee (\gamma \vee \delta))$

$F((\alpha \to \beta) \vee (\gamma \vee \delta))$

$F(\alpha \to \beta)$

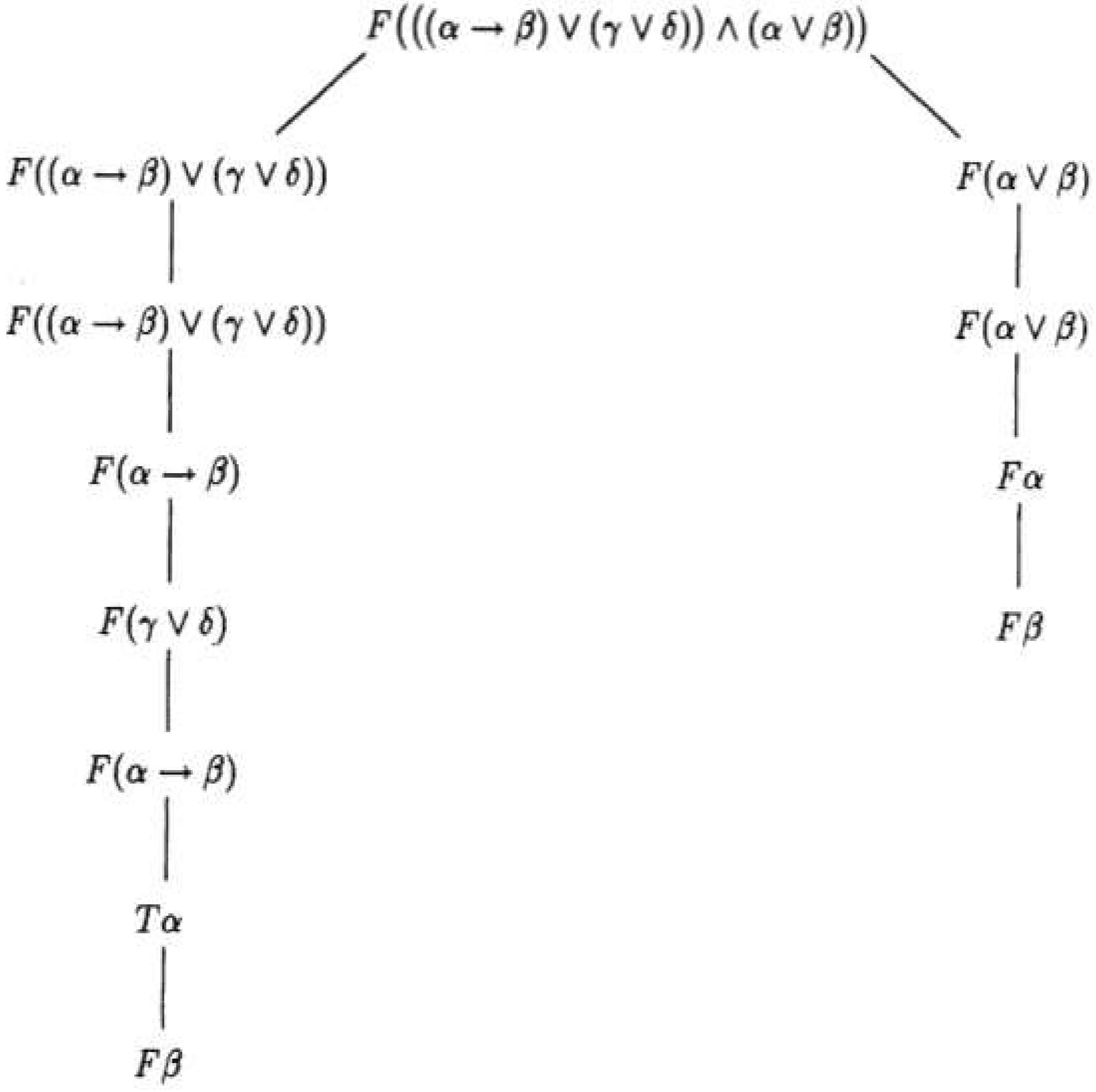
$F(\gamma \vee \delta)$

$F(\alpha \vee \beta)$

$F(\alpha \vee \beta)$

$F\alpha$

$F\beta$


$F(((\alpha \rightarrow \beta) \vee (\gamma \vee \delta)) \wedge (\alpha \vee \beta))$
$F((\alpha \rightarrow \beta) \vee (\gamma \vee \delta))$
$F((\alpha \rightarrow \beta) \vee (\gamma \vee \delta))$
$F(\alpha \rightarrow \beta)$
$F(\gamma \vee \delta)$
$F(\alpha \rightarrow \beta)$
$T\alpha$
$F\beta$
$F(\alpha \vee \beta)$
$F(\alpha \vee \beta)$
$F\alpha$
$F\beta$

Tableaux

Definition 4.2 Let τ be a tableau, P a path on τ and E an entry occurring on P.

(i) E has been reduced on P if all the entries on one path through the atomic tableau with root E occur on P. (E.g., TA and FA are reduced for every propositional letter A. $T\neg\alpha$ and $F\neg\alpha$ are reduced (on P) if $F\alpha$ and α, respectively, appear on P. $T(\alpha \vee \beta)$ is reduced if either $T\alpha$ or $T\beta$ appears on P. $F(\alpha \vee \beta)$ is reduced if both $F\alpha$ and $F\beta$ appear on P.)

(ii) P is contradictory if, for some proposition α, $T\alpha$ and $F\alpha$ are both entries on P. P is finished if it is contradictory or every entry on P is reduced on P.

(iii) τ is finished if every path through τ is finished.

(iv) τ is contradictory if every path through τ is contradictory. (It is, of course, then finished as well.)

Example: Tableaux

A finished tableau with three paths. The leftmost path is contradictory; the other two are not.

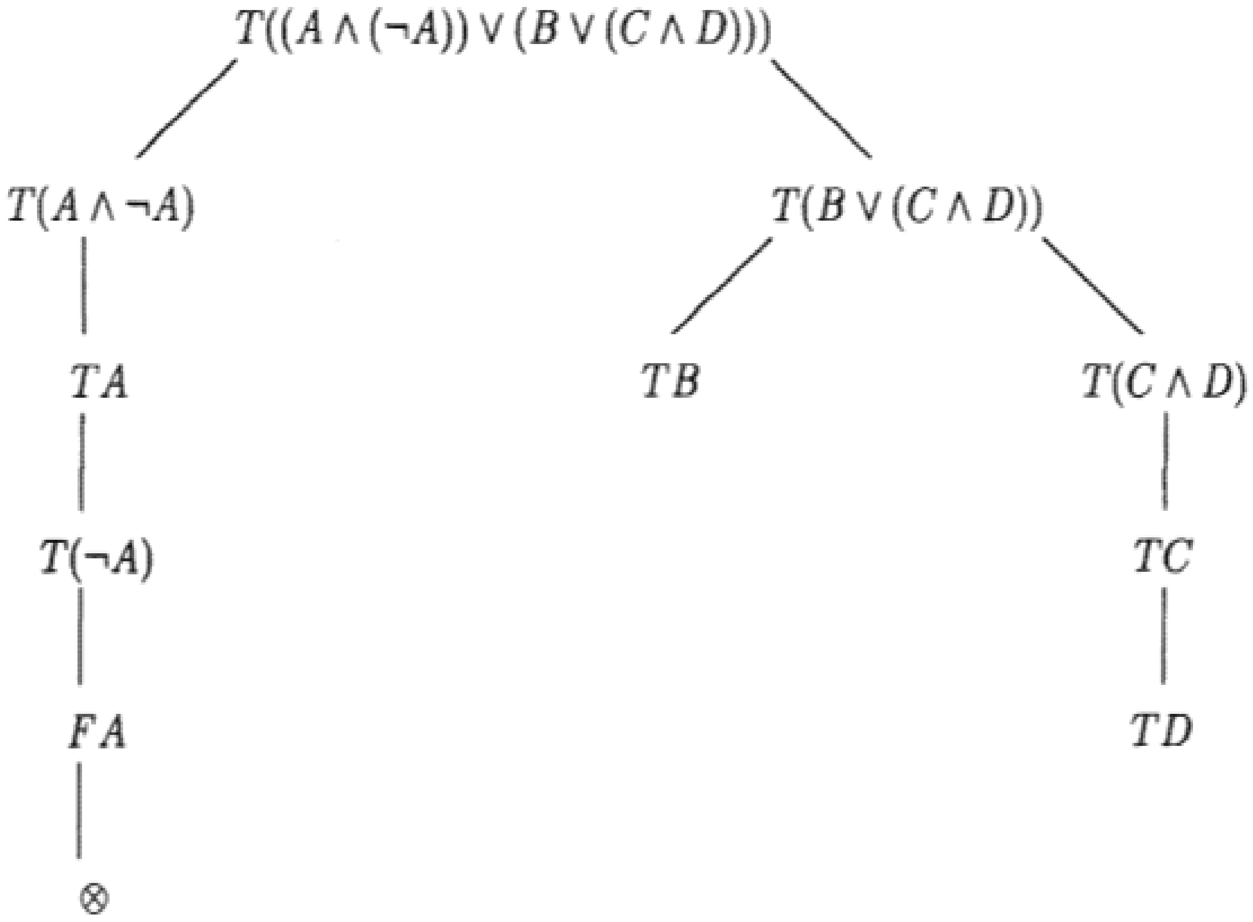


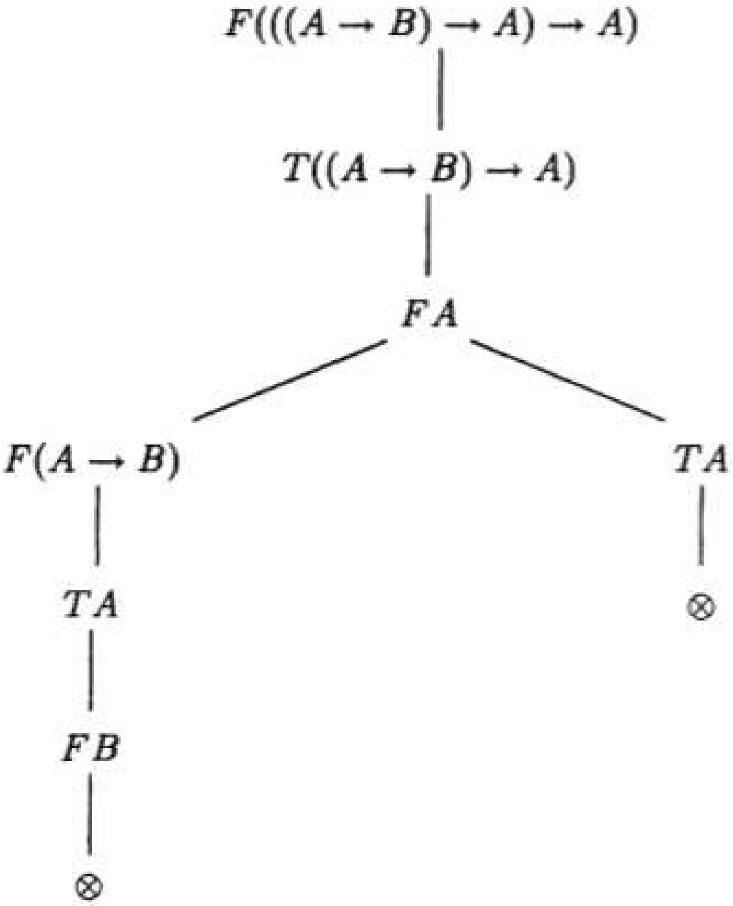
Tableau Proof

Definition 4.3 A tableau proof of a proposition α is a contradictory tableau with root entry $F\alpha$. A proposition is tableau provable, written $\vdash \alpha$, if it has a tableau proof.

A tableau refutation for a proposition α is a contradictory tableau starting with $T\alpha$. A proposition is tableau refutable if it has a tableau refutation.

Example: Tableau Proof

A tableau proof of an instance of Peirce's law.



Complete systematic tableaux (CST)

Definition 4.4 Let R be a signed proposition. We define the complete systematic tableau (CST) with root entry R by induction.

- Let τ_0 be the unique atomic tableau with R at its root.
- Assume that τ_m has been defined. Let n be the smallest level of τ_m containing an entry that is unreduced on some noncontradictory path in τ_m and let E be the leftmost such entry of level n. We now let τ_{m+l} be the tableau gotten by adjoining the unique atomic tableau with root E to the end of every noncontradictory path of τ_m on which E is unreduced. The union of the sequence τ_m is our desired complete systematic tableau.

CST are finished

Theorem 4.5 Every CST is finished.

Proof: Consider any entry E that occurs at some level n of the CST τ and lies on a noncontradictory path P in T. There are at most finitely many entries on T at or above level n. Thus, all the entries at level n or above on T must be in place by some point of the construction. That is, there is an m_0 such that for every $m \geq m_0$, τ_m through level n is the same as T through level n. Now, for $m \geq m_0$, the restriction of P to τ_m is a path in τ_m containing E. At each step $m \geq m_0$ in the construction of the CST we reduce the entry on the lexicographically least node labeled with an unreduced entry that is on some noncontradictory path in the tableau τ_m. If E is not already reduced on P by stage m_0, we can proceed for at most finitely many steps in this construction before E would become the lexicographically least unreduced entry. At this point in the construction we would reduce E. □

CST as a proof

Theorem 4.6 If $\tau = \cup\tau_n$ is a contradictory tableau, then for some m, τ_m is a finite contradictory tableau. Thus, in particular, if a CST is a proof, it is a finite tableau.

Proof: τ is a finitely branching tree. Consider the subset of all nodes of τ with no contradiction above them. If this set is infinite, it has an infinite path by Konig's lemma. As this contradicts the assumption that every path in τ is contradictory, there are only finitely many such nodes. They must all appear by some level n of τ. Thus, every node at level $n+1$ of τ has a contradiction above it. Once again, as τ through level $n+1$ is finite, there is an m such that τ_m is the same as τ through level $n+1$. Now every path P in τ_m is either a path in τ (ending with a leaf of level $\leq n$) or a path containing a node of level $n+1$. In the first case, P is contradictory by our assumption that τ is contradictory. In the second, P is contradictory by our choice of n and m. Thus, τ_m is the desired contradictory tableau.

Note that if $\tau = \cup \tau_n$ is as in the definition of a CST and m is the least such that τ_m is contradictory, then we cannot extend τ_m in the construction of τ. In this case $\tau = \tau_m$. □

Every CST is finite

Definition 4.7 Define the degree of a proposition α, $d(\alpha)$ by induction:

(i) If α is a propositional letter, then $d(\alpha) = 0$.

(ii) If α is $\neg\beta$, then $d(\alpha) = d(\beta) + 1$.

(iii) If α is $\beta \vee \gamma$, $\beta \wedge \gamma$, $\beta \rightarrow \gamma$ or $\beta \leftrightarrow \gamma$, then $d(\alpha) = d(\beta) + d(\gamma) + 1$. The degree of a signed proposition $T\alpha$ or $F\alpha$ is the degree of α. If P is a path in a tableau τ, then $d(P)$ the degree of P is the sum of the signed propositions on P that are not reduced on P.

Theorem 4.8 Every CST is finite

Proof: Let $\tau = \cup\tau_m$ be any CST. We prove that every path on τ is finite (indeed has length at most the degree of the root of τ) and so, by König's lemma, τ itself is finite. Consider any path P on τ. It is the union of paths P_m on τ_m. A change occurs between P_m and P_{m+1} when we add the atomic tableau σ with root E to the end of P_m for some entry E that is unreduced on a path in τ_m. We claim that $d(P_{m+1}) < d(P_m)$. Of course, this immediately implies that we can add an atomic tableau to the end of P_m at most finitely often (indeed at most $d(\alpha)$ many times where α is the proposition at the root of τ). Thus, P is finite as desired. To verify the claim first note that adding σ to the end of our path reduces the entry E on P. This subtracts $d(E)$ from the degree of the path while adding on the degrees of the signed propositions other than E occurring on the path of σ that is added on to P_m to form P_{m+1}. Thus, it suffices to check that the sum of the degrees of the signed formulas (excluding the root) on each path through any atomic tableau σ is less than the degree of the root of σ. This is immediate from the definition of degree and the list of atomic tableaux. □

Exercises

1. Exercise 4 in page 26
2. Exercise 3 in page 36
3. Exercise 10 in page 38

5 Soundness and Completeness of Tableaux

We prove the equivalence of the semantic notion of validity ($\models$) and the syntactic notion of provability ($\vdash$). Thus, we show that all tableau provable propositions are valid (soundness of the proof method) and that all valid propositions are tableau provable (completeness of the method).

Soundness

Theorem 5.1 [Soundness] If α is tableau provable, then α is valid, i.e., $\vdash \alpha \;\Rightarrow\; \models \alpha$.

Proof: We prove the contrapositive. Suppose α is not valid. There is a valuation $\mathcal{V}$ assigning F to α. We say that the valuation $\mathcal{V}$ agrees with a signed proposition E in two situations: if E is $T\alpha$ and $\mathcal{V}(\alpha) = T$ or if E is $F\alpha$ and $\mathcal{V}(\alpha) = F$. We show below that if any valuation $\mathcal{V}$ agrees with the root node of a tableau, then there is a path P in the tableau such that $\mathcal{V}$ agrees with every entry on P. As no valuation can agree with any path on a contradictory tableau there can be no tableau proof of α. □

Lemma 5.2 If $\mathcal{V}$ is a valuation that agrees with the root entry of a given tableau τ given as in Definition 4.1 as $\cup\tau_n$, then τ has a path P every entry of which agrees with V.

Proof: Prove by induction that there is a sequence $\langle P_n \rangle$ such that, for every n, P_n is contained in P_{n+1} and P_n is a path through τ_n such that $\mathcal{V}$ agrees with every entry on P_n. The desired path P through τ will then simply be the union of the P_n. The base case of the induction is easily seen to be true by the assumption that $\mathcal{V}$ agrees with the root of τ. As an example, consider (6a) with root entry $T(\alpha \leftrightarrow \beta)$. If $\mathcal{V}(\alpha \leftrightarrow \beta) = T$,then either $\mathcal{V}(\alpha) = T$ and $\mathcal{V}(\beta) = T$ or $\mathcal{V}(\alpha) = F$ and $\mathcal{V}(\beta) = F$ by the truth table definition for $\leftrightarrow$. Similar for other atomic tableaux.

For the induction step, suppose that we have constructed a path P_n in τ_n every entry of which agrees with $\mathcal{V}$. If τ_{n+1} is gotten from τ_n without extending P_n, then we let $P_{n+1} = P_n$. If P_n is extended in τ_{n+1}, then it is extended by adding on to its end an atomic tableau with root E for some entry E appearing on P_n. By induction $\mathcal{V}$ agrees with E, the same analysis as used in the base case shows that $\mathcal{V}$ agrees with one of the extensions of P_n to a path P_{n+1} in τ_{n+1}. □

Completeness

Theorem 5.3 If α is valid, then α is tableau provable, i.e., $\models \alpha \;\Rightarrow\; \vdash \alpha$. In fact, any finished tableau with root entry $F\alpha$ is a proof of α and so, in particular, the complete systematic tableaux with root $F\alpha$ is such a proof.

Key Lemma

Lemma 5.4 Let P be a noncontradictory path of a finished tableau τ. Define a truth assignment $\mathcal{A}$ on all propositional letters A as follows:
$\mathcal{A}(A) = T$ if TA is an entry on P.
$\mathcal{A}(A) = F$ otherwise.

If $\mathcal{V}$ is the unique valuation (Theorem 3.3) extending the truth assignment $\mathcal{A}$, then $\mathcal{V}$ agrees with all entries of P.

Proof: By induction on the depth of propositions on P.

(i) If α is a propositional letter and $T\alpha$ occurs on P, then $\mathcal{V}(\alpha) = T$ by definition and we are done. If $F\alpha$ occurs on P, then, as P is noncontradictory, $T\alpha$ does not and $\mathcal{V}(\alpha) = F$.

(ii) Suppose $T(\alpha \wedge \beta)$ occurs on the noncontradictory path P. Since τ is a finished tableau, both $T(\alpha)$ and $T(\beta)$ occur on P. By the induction hypothesis $\mathcal{V}(\alpha) = T = \mathcal{V}(\beta)$ and so $\mathcal{V}(\alpha \wedge \beta) = T$ as required.

(iii) Suppose $F(\alpha \wedge \beta)$ occurs on the noncontradictory path P. Again by the definition of a finished tableau, either $F\alpha$ or $F\beta$ must occur on P. Whichever it is, the induction hypothesis tells us that it agrees with $\mathcal{V}$ and so either $\mathcal{V}(\alpha) = F$ or $\mathcal{V}(\beta) = F$. In either case $\mathcal{V}(\alpha \wedge \beta) = F$ as required.

Other connectives are treated likewise. □

Proof: (of Theorem 5.3): Suppose that α is valid and so $\mathcal{V}(\alpha) = T$ for every valuation $\mathcal{V}$. Consider any finished tableau τ with root $F\alpha$. (The CST with root $F\alpha$ is one by Theorem 4.5) If τ had a noncontradictory path P, there would be, by Lemma 5.4, a valuation $\mathcal{V}$ that agrees with all its entries and so in particular with $F\alpha$. This would give us a valuation with $\mathcal{V}(\alpha) = F$ contradicting the validity of α. Thus, every path on τ is contradictory and τ is a tableau proof of α. □

6 Deductions from Premises

Definition 6.1 [Tableaux from premises] Let Σ be a (possibly infinite) set of propositions. A finite tableau with premises from Σ is a binary tree that satisfies the following inductive definition:

(i) All atomic tableaux are finite tableaux from Σ.

(ii) If τ is a finite tableau from Σ and $\alpha \in \Sigma$, then the tableau formed by putting $T\alpha$ at the end of every noncontradictory path not containing it is also a finite tableau from Σ.

(iii) If τ is a finite tableau from Σ, P a path on τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining the unique atomic tableau with root entry E to τ at the end of the path P, then τ' is also a finite tableau from Σ.

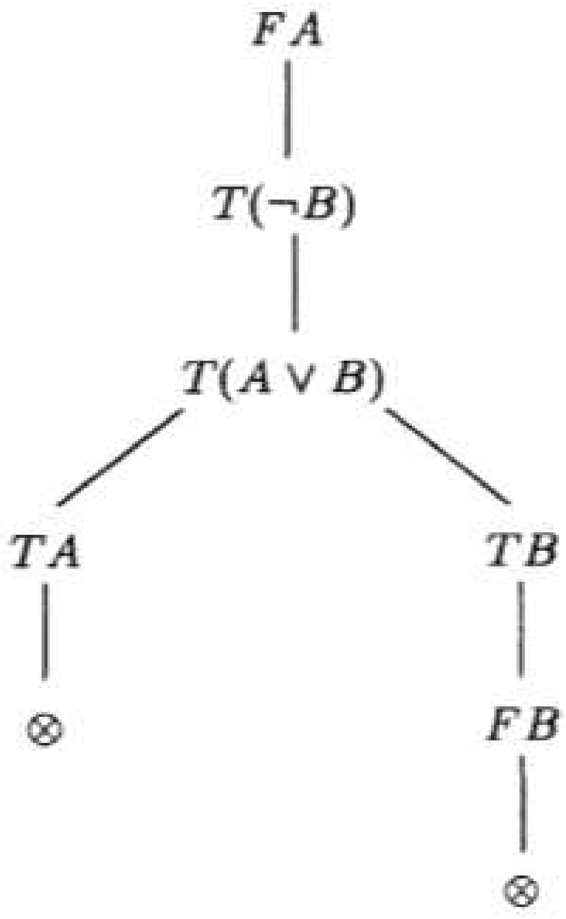
If $\tau_0, \tau_1, ..., \tau_n, ...$ is a (finite or infinite) sequence of finite tableaux from Σ such that, for each $n \geq 0$, τ_{n+1} is constructed from τ_n by an application of (ii) or (iii), then $\tau = \cup \tau_n$ is a tableau from Σ.

Tableau Proof of a Proposition from Σ

Definition 6.2 A tableau proof of a proposition α from Σ (or with premises from Σ) is a tableau from Σ with root entry $F\alpha$ that is contradictory, that is, one in which every path is contradictory. If there is such a proof we say that α is provable from Σ and write it as $\Sigma \vdash \alpha$.

Example: Tableau Proof

A tableau proof of A from $\{\neg B,\ (A \vee B)\}$.



CST from a set of premises

A finished tableau from Σ is a tableau from Σ that is a finished tableau in the sense of Definition 4.2 and has an occurrence of $T\alpha$ on every noncontradictory path for every $\alpha \in \Sigma$.

To construct a CST from Σ, we list the elements of Σ as $\alpha_m, \ m \in \mathcal{N}$, and revise the definition of the CST by simply adding on one step to the definition of τ_{m+1}. If our new construction has produced τ_m we let τ'_{m+1} be the next tableau that would be defined by the standard CST procedure. (If that procedure would now terminate because every path is contradictory, we also terminate the current construction.) We now add on $T\alpha_m$ to the end of every noncontradictory path in τ'_{m+1} that does not already contain $T\alpha$ to form our new τ_{m+1}.

Theorem 6.3 Every CST from a set of premises is finished.

Soundness of deduction from premises

Lemma 6.4 If a valuation $\mathcal{V}$ makes every $\alpha \in \Sigma$ true and agrees with the root of a tableau τ from Σ, then there is a path in τ every entry of which agrees with $\mathcal{V}$.

Theorem 6.5 [Soundness of deduction from premises] If there is a tableau proof of α from a set of premises Σ, then α is a consequence of Σ, $\Sigma \vdash \alpha \ \Rightarrow \Sigma \models \alpha$.

Proof: If not, there is a valuation that makes β true for every $\beta \in \Sigma$ but makes α false. Continue now as in the proof of Theorem 5.1. □

Completeness of deduction from premises

Lemma 6.6 Let P be a noncontradictory path in a finished tableau τ from Σ. Define a valuation $\mathcal{V}$ as in Lemma 5.4. $\mathcal{V}$ then agrees with all entries on P and so in particular makes every proposition $\beta \in \Sigma$ true (as $T\beta$ must appear on P for every $\beta \in \Sigma$ by definition of a finished tableau from Σ).

Theorem 6.7 [Completeness of deduction from premises] If α is a consequence of a set Σ of premises, then there is a tableau deduction of α from Σ, i.e., $\Sigma \models \alpha \ \Rightarrow \ \Sigma \vdash \alpha$.

Proof: If $\Sigma \models \alpha$, every valuation $\mathcal{V}$ that makes every proposition in Σ true also makes α true. Consider the CST from Σ with root $F\alpha$. It is finished by Theorem 6.3. Now apply Lemma 6.6. □

Compactness

Theorem 6.8 If $\tau = \cup\tau_n$ is a contradictory tableau from Σ, then, for some m, τ_m is a finite contradictory tableau from Σ. In particular, if a CST from Σ is a proof, it is finite.

Theorem 6.9 α is a consequence of Σ iff α is a consequence of some finite subset of Σ.

Compactness

Definition 6.10 A set Σ of propositions is called satisfiable if it has a model, i.e., there is a valuation $\mathcal{V}$ such that $\mathcal{V}(\alpha) = T$ for every $\alpha \in \Sigma$. We also say that such a valuation satisfies Σ.

E.g.

(i) $\{A_1, A_2, (A_1 \wedge A_2), A_3, (A_1 \wedge A_3), A_4, (A_1 \wedge A_4), ...\}$ is a satisfiable infinite set of propositions.

(ii) $\{A_1, A_2, (A_1 \to A_3), (\neg A_3)\}$ is a finite set of propositions that is not satisfiable nor is any set containing it.

Compactness

Theorem 6.11 [Compactness] Let $\Sigma = \{\alpha_i \mid i \in \omega\}$ be an infinite set of propositions. Σ is satisfiable if and only if every finite subset of Σ is satisfiable.

Proof: $(\Rightarrow)$ Trivial.

$(\Leftarrow)$ Let $\langle C_i \mid i \in \omega \rangle$ be a list of all the propositional letters. We define a tree T whose nodes are binary sequences ordered by extension. We use $lth(\alpha)$ to denote the length of a sequence σ and set $T = \{\sigma \mid \text{ there exists a valuation } \mathcal{V} \text{ s.t., for } i \leq lth(\sigma),\ \mathcal{V}(\alpha_i) = T \text{ and } \mathcal{V}(C_i) = T \text{ iff } \sigma(i) = 1\}$. What this definition says is that we put σ on the tree unless interpreting it as an assignment of truth values to the propositional letters $C_i (i \leq lth(\sigma))$ already forces one of the α_i to be false for $i \leq lth(\sigma)$.

Claim: There is an infinite path in T if and only if Σ is satisfiable.

Proof of Claim: If $\mathcal{V}$ satisfies Σ, then, by definition, the set of all σ such that $\sigma(i) = 1$ iff $\mathcal{V}(C_i) = T$ is a path on T. On the other hand, suppose that $\langle \sigma_j \mid j \in \mathcal{N} \rangle$ is an infinite path on T. Let $\mathcal{V}$ be the unique valuation extending the assignment determined by the σ_j, i.e., the one for which C_i is true iff $\sigma_j(i) = 1$ for some j (or equivalently, as the σ_i are linearly ordered by extension, iff $\sigma_j(i) = 1$ for every i such that $i \leq lth(\sigma_j)$) If $\mathcal{V} \not\models \Sigma$, then there is some $\alpha_j \in \Sigma$ such that $\mathcal{V}(\alpha_j) = F$. Now by Corollary 3.4 this last fact depends on the truth values assigned by $\mathcal{V}$ to only finitely many propositional letters. Let us suppose it depends only on those C_i, with $i \leq n$. It is then clear from the definition of T that no σ with length $\geq n$ can be on T at all. As there are only finitely many binary sequences σ with length $\leq n$, we have contradicted the assumption that the sequence $\langle \sigma_j \rangle$ is an infinite path on T and so $\mathcal{V} \models \Sigma$ as claimed.

The next **claim** is that there is, for every n, a σ of length n in T. By assumption every finite subset of Σ is satisfiable. Thus, for each n, there is a valuation $\mathcal{V}_n$ that makes α_i true for each $i \leq n$. The string σ given by $\sigma(i) = 1$ iff $\mathcal{V}_n(C_i) = T$ for $i \leq n$ is then on T by definition.

By König's lemma (Theorem 1.4) there is an infinite path in T and so Σ is satisfiable as required. □

Exercises

- Exercise 6 in page 45
- Exercise 7 in page 46
- Exercise 8 in page 46

7 An Axiomatic Approach

The axioms of propositional logic are certain valid propositions. A rule of inference, R, in general, "infers" a proposition α from certain n-tuples $\alpha_1, ..., \alpha_n$ of propositions in a way that is expected to preserve validity.

Axioms: The axioms of our system are all propositions of the following forms:

(i) $(\alpha \to (\beta \to \alpha))$

(ii) $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$

(iii) $(\neg\beta \to \neg\alpha) \to ((\neg\beta \to \alpha) \to \beta)$

where α, β and γ can be any propositions. The forms in this list are often called axiom schemes. The axioms are all instances of these schemes as α, β and γ vary over all propositions.

The rule of inference (Modus Ponens)

$$\frac{\begin{array}{c}\alpha \\ \alpha \to \beta\end{array}}{\beta}$$

Systems based on axioms and rules in the above style are called Hilbert-style proof systems. Denote provability by $\vdash_H$.

Proofs

Definition 7.1 (i) A proof from Σ is a finite sequence $\alpha_1, \alpha_2, ..., \alpha_n$ such that for each $i \leq n$ either:

(1) α_i is a member of Σ;

(2) α_i is an axiom; or

(3) α_i can be inferred from some of the previous a α_j by an application of a rule of inference.

(ii) α is provable from Σ, $\Sigma \vdash_E \alpha$, if there is a proof $\alpha_1, ..., \alpha_n$ from Σ where $\alpha_n = \alpha$.

(iii) A proof of α is simply a proof from the empty set $\emptyset$; α is provable if it is provable from $\emptyset$.

Example: Proofs

Here is a proof of $((\neg\beta \to \alpha) \to \beta)$ from $\Sigma = \{\neg\alpha\}$:

$\neg\alpha$	from Σ
$(\neg\alpha \to (\neg\beta \to \neg\alpha))$	from axiom (i)
$(\neg\beta \to \neg\alpha)$	modus ponens
$((\neg\beta \to \neg\alpha) \to ((\neg\beta \to \alpha) \to \beta))$	axiom (iii)
$((\neg\beta \to \alpha) \to \beta)$	modus ponens

Note that although the set of premises Σ may be infinite, if α is provable from Σ, then α is provable from a finite subset of Σ. Proofs are always finite!

Soundness and completeness from premises

Theorem 7.2 α is provable from a set of propositions Σ if and only if α is a consequence of Σ, i.e., $\Sigma \vdash_H \alpha \ \Leftrightarrow \Sigma \models \alpha$.

Corollary 7.3 A proposition α is provable if and only if it is valid, i.e., $\vdash_H \alpha \ \Leftrightarrow \ \models \alpha$.

8 Resolution

Definition 8.1 (i) A literal l is a propositional letter p or its negation $\neg p$. If l is p or $\neg p$, we write $\bar{l}$ for $\neg p$ or p, respectively. The propositional letters are also called positive literals and their negations negative literals.

(ii) A clause C is a finite set of literals (which you should think of as the disjunction of its elements). As we think of C as being true iff one of its elements is true, the empty clause $\Box$ is always false - it has no true element.

(iii) A formula S is a (not necessarily finite) set of clauses (which you should think of as the conjunction of its elements). As we think of a formula S as being true if everyone of its elements is true, the empty formula $\emptyset$ is always true - it has no false element.

(iv) An assignment $\mathcal{A}$ is a consistent set of literals, i.e., one not containing both p and $\neg p$ for any propositional letter p. (This, of course, is just the (partial) truth assignment in which those $p \in \mathcal{A}$ are assigned T and those q with $\bar{q} \in \mathcal{A}$ are assigned F.) A complete assignment is one containing p or $\neg p$ for every propositional letter p. It corresponds to what we called a truth assignment in Definition 3.1.

(v) $\mathcal{A}$ satisfies S, $\mathcal{A} \models S$, iff $\forall C \in S(C \cap \mathcal{A} \neq \emptyset)$, i.e., the valuation induced by $\mathcal{A}$ makes every clause in S true.

(vi) A formula S is (un)satisfiable if there is an (no) assignment $\mathcal{A}$ that satisfies it.

Example: literals, clauses and formulas

Definition 8.2 (i) $p, q, r, \neg p, \bar{q}(= \neg q), \bar{r}$ and $\neg\bar{q}(= q)$ are literals.

(ii) $\{p, r\}$, $\{\neg q\}$ and $\{q, \neg r\}$ are clauses.

(iii) $S = \{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{s, \neg t\}\}$ is a formula that, in our original notation system, would be written as $((p \vee r) \wedge (q \vee \neg r) \wedge (\neg q) \wedge (\neg p \vee t) \wedge (s \vee \neg t))$.

(iv) If $\mathcal{A}$ is given by $\{p, q, r, s, t\}$, i.e., the (partial) assignment such that $\mathcal{A}(p) = T = \mathcal{A}(q) = \mathcal{A}(r) = \mathcal{A}(s) = \mathcal{A}(t)$, then $\mathcal{A}$ is an assignment not satisfying the formula S in (iii). S is, however, satisfiable.

Resolution rule

The resolution rule is much like a version of modus ponens called cut. Modus ponens says that from α and $\alpha \to \beta$ one can infer β. In this format, the cut rule says that from $\alpha \vee \gamma$ and $\neg\alpha \vee \beta$ infer $\gamma \vee \beta$. Thus, cut is somewhat more general than modus ponens in that it allows one to carry along the extra proposition γ. Resolution is a restricted version of cut in which α must be a literal while β and γ must be clauses.

Resolution

Definition 8.3 [Resolution] From clauses C_1 and C_2 of the form $\{l\} \sqcup C_1'$ and $\{\bar{l}\} \sqcup C_2'$, infer $C = C_1' \cup C_2'$ which is called a resolvent of C_1 and C_2. (Here l is any literal and $\sqcup$ means that we are taking a union of disjoint sets.) We may also call C_1 and C_2 the parent and C their child and say that we resolved on (the literal) l.

Resolution deduction

Definition 8.4 A (resolution) deduction or proof of C from a given formula S is a finite sequence $C_1, C_2, ..., C_n = C$ of clauses such that each C_i is either a member of S or a resolvent of clauses C_j, C_k for $j, k < i$. If there is such a deduction, we say that C is (resolution) provable from S and write $S \vdash_{\mathcal{R}} C$. A deduction of $\Box$ from S is called a (resolution) refutation of S. If there is such a deduction we say that S is (resolution) refutable and write $S \vdash_{\mathcal{R}} \Box$.

Example: resolution

(i) From $\{p, r\}$ and $\{\neg q, \neg r\}$ conclude $\{p, \neg q\}$ by resolution (on r).

(ii) From $\{p, q, \neg r, s\}$ and $\{\neg p, q, r, t\}$ we could conclude either $\{q, \neg r, s, r, t\}$ or $\{p, q, s, \neg p, t\}$ by resolution (on p or r), respectively. Of course, both of these clauses are valid and are equivalent to the empty formula.

Resolution tree proof

Definition 8.5 A resolution tree proof of C from S is a labeled binary tree T with the following properties:

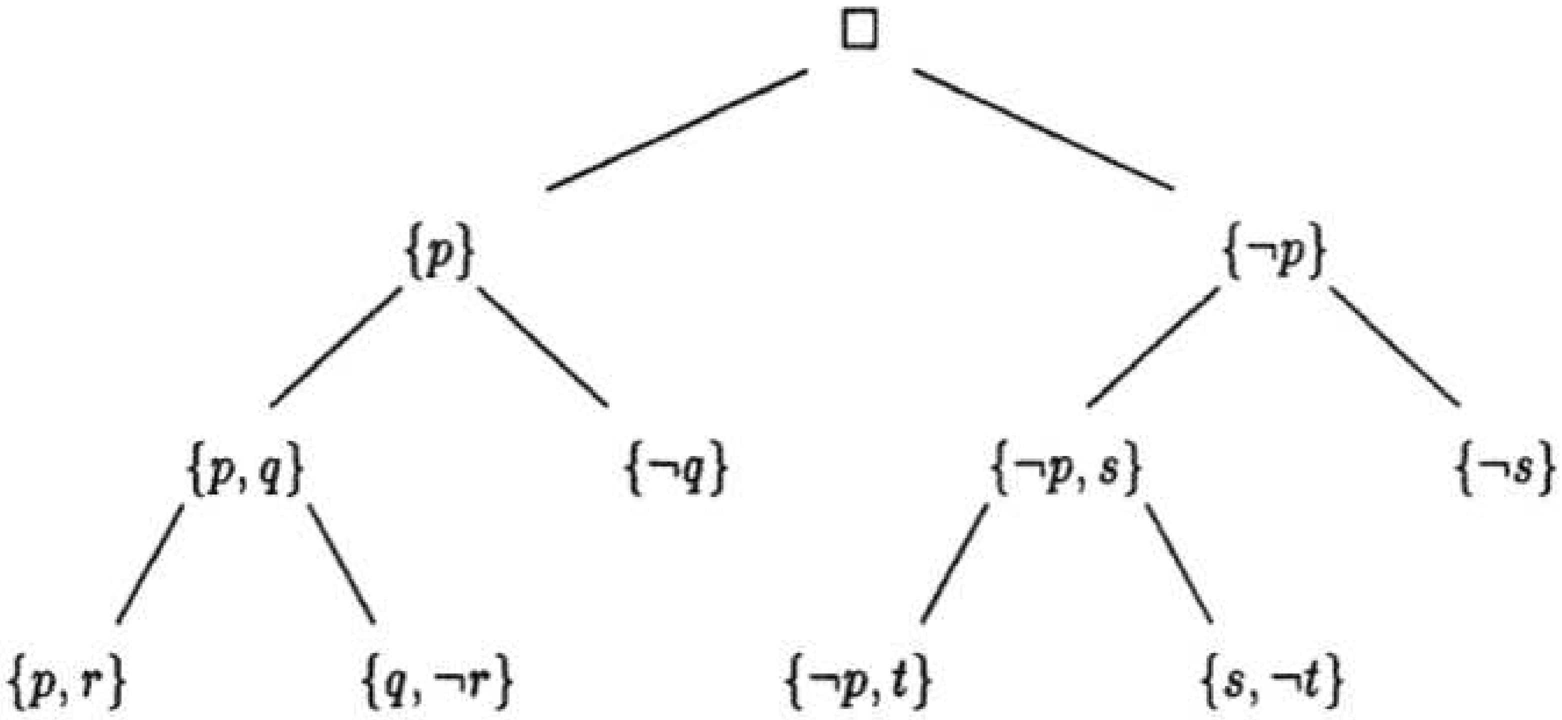
(i) The root of T is labeled C.

(ii) The leaves of T are labeled with elements of S.

(iii) If any nonleaf node σ is labeled with C_2 and its immediate successors σ_0, σ_1 are labeled with C_0, C_1, respectively, then C_2 is a resolvent of C_0 and C_1.

Example: resolution tree proof

A resolution tree refutation of the formula $S = \{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{\neg s\}, \{s, \neg t\}\}$, i.e., a resolution tree proof of $\Box$ from S:



Resolution tree proof

Lemma 8.6 C has a resolution tree proof from S if and only if there is a resolution deduction of C from S.

Proof: ($\Rightarrow$) List all the labels of the nodes σ of the tree proof of C from S in any order that reverses the $<$ ordering of the tree (so leaves are listed first and the root last). This sequence can be seen to be a resolution deduction of C from S by simply checking the definitions.

($\Leftarrow$) We proceed by induction on the length of the resolution deduction of C from S. Suppose we can get tree proofs for any deduction of length $< n$ and $C_1, ..., C_n$ is one of length n from S. If $C_n \in S$,there is nothing to prove. If not, then C_n is the resolvent of C_i and C_j for some i and j less than n. By induction, we have tree proofs T_i and T_j of C_i and C_j. Let T_n be the tree whose root is labeled C and to whose immediate successors we attach T_i and T_j. Again, by definition, this is the desired tree proof. □

Closure

Definition 8.7 $\mathcal{R}(S)$ is the closure of S under resolution, i.e., the set determined by the following inductive definition:

1. If $C \in S, C \in \mathcal{R}(S)$.
2. If $C_1, C_2 \in \mathcal{R}(S)$ and C is a resolvent of C_1 and C_2, then $C \in \mathcal{R}(S)$.

Proposition 8.8 For any clause C and formula S, there is a resolution deduction of C from S iff $C \in \mathcal{R}(S)$. In particular, there is a resolution refutation of S iff $\Box \in \mathcal{R}(S)$.

Soundness (1/2)

Lemma 8.9 If the formula (i.e., set of clauses) $S = \{C_1, C_2\}$ is satisfiable and C is a resolvent of C_1 and C_2, then C is satisfiable. Indeed, any assignment $\mathcal{A}$ satisfying S satisfies C.

Proof: As C is a resolvent of C_1 and C_2, there are l, C_1' and C_2' such that $C_1 = \{l\} \sqcup C_1'$, $C_2 = \{\bar{l}\} \sqcup C_2'$ and $C = C_1' \cup C_2'$. As $\mathcal{A}$ satisfies $\{C_1, C_2\}$, it satisfies (that is, it contains an element of) each of C_1 and C_2. As $\mathcal{A}$ is an assignment, it cannot be the case that both $l \in \mathcal{A}$ and $\bar{l} \in \mathcal{A}$. Say $\bar{l} \notin \mathcal{A}$. As $\mathcal{A} \models C_2$ and $\bar{l} \notin \mathcal{A}$, $\mathcal{A} \models C_2'$ and so $\mathcal{A} \models C$. The proof for $l \notin \mathcal{A}$ just replaces C_2 by C_1 and $\bar{l}$ by l. □

Soundness (2/2)

Theorem 8.10 [Soundness of resolution] If there is a resolution refutation of S, then S is unsatisfiable.

Proof: If $C_1, ..., C_n$ is a resolution deduction from S, then the lemma shows by induction (on n) that any assignment satisfying S satisfies every C_i. If the deduction is in fact a refutation of S, then $C_n = \Box$. As no assignment can satisfy $\Box$, S is unsatisfiable. $\Box$

Completeness (1/2)

Lemma 8.11 For any formula T and any literal l, let $T(l) = \{C \in \mathcal{R}(T) \mid l, \bar{l} \notin C\}$. If T is unsatisfiable, then so is $T(l)$.

Proof: Assume T is unsatisfiable and suppose, for the sake of a contradiction, that $\mathcal{A}$ is any assignment that satisfies $T(l)$ and is defined on all the literals (of T) other than l. Let $\mathcal{A}_1 = \mathcal{A} \cup \{l\}$ and $\mathcal{A}_2 = \mathcal{A} \cup \{\bar{l}\}$. As T is unsatisfiable, there are clauses C_1 and C_2 in T such that $\mathcal{A}_1 \not\models C_1$ and $\mathcal{A}_2 \not\models C_2$. Now as $l \in \mathcal{A}_1$ and $\mathcal{A}_1 \not\models C_1$, $l \notin C_1$. If $\bar{l}$ is also not in C_1, then $C_1 \in T(l)$ by definition. As this would contradict our assumption that $\mathcal{A} \models T(l)$, $\bar{l} \in C_1$. Similarly, $l \in C_2$. Thus, we may resolve C_1 and C_2 on l to get a clause D not containing l and hence in $T(l)$. (As a resolvent of two clauses in T, D is certainly in $\mathcal{R}(T)$). Then, by our choice of $\mathcal{A}$, $\mathcal{A} \models D$. If $\mathcal{A}$ satisfies the resolvent D, however, it must satisfy one of the parents C_1 or C_2. Thus, we have the desired contradiction. □

Completeness (2/2)

Theorem 8.12 [Completeness of resolution] If S is unsatisfiable, then there is a resolution refutation of S.

Proof: By the compactness theorem (Theorem 6.13), there is a finite subset S' of S that is unsatisfiable. As any refutation deduction from S' is one from S, we may assume that S is finite, i.e., it contains only finitely many clauses. If there are only finitely many clauses in S and each clause is finite, there are only finitely many literals, say $l_1, l_2, ..., l_n$ which are in any clause in S.Then we consider only clauses and formulas based on these n literals.

We wish to consider the set of clauses $C \in \mathcal{R}(S)$ and prove that it contains $\Box$. We proceed by eliminating each literal in turn by applying Lemma 8.11. We begin with $S_n = S(l_n) = \{C \in \mathcal{R}(S) \mid l_n, \bar{l}_n \notin C\}$. By definition, it is a collection of resolution consequences of S none of which contain l_n or $\bar{l}_n$. By Lemma 8.11 it is unsatisfiable. Next we let $S_{n-1} = S_n(l_{n-1})$. It is an unsatisfiable collection of resolution consequences of S_n (and hence of S) none of which contain $l_{n-1}, \bar{l}_{n-1}, l_n, \bar{l}_n$. Continuing in this way we define $S_{n-2}, ..., S_0$. By repeated applications of the definitions and Lemma 8.11, we see that S_0 is an unsatisfiable set of resolution consequences of S containing no literals at all. As the only formulas with no literals are $\emptyset$ and $\{\Box\}$ and $\emptyset$ is satisfiable, $\Box \in S_0$. Thus, $\Box$ is a resolution consequence of S as required. □

An abstract formulation of completeness

Definition 8.13 If S is a formula and l a literal, we let

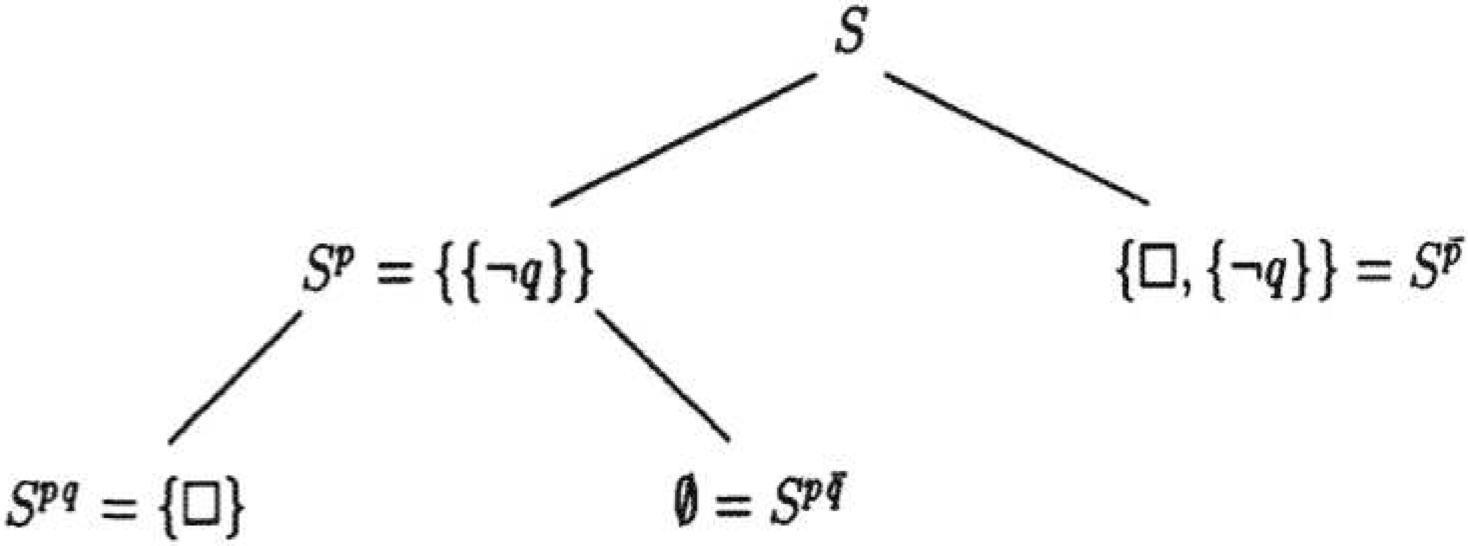
$S^l = \{C - \{\bar{l}\} \mid C \in S \wedge l \notin C\}$

So S^l consists of those clauses C of S containing neither l nor $\bar{l}$, plus those clauses (not containing l) such that $C \cup \{\bar{l}\} \in S$. Note that if the singleton clause $\{\bar{l}\}$ is in S, then $\Box$ is in S^l.

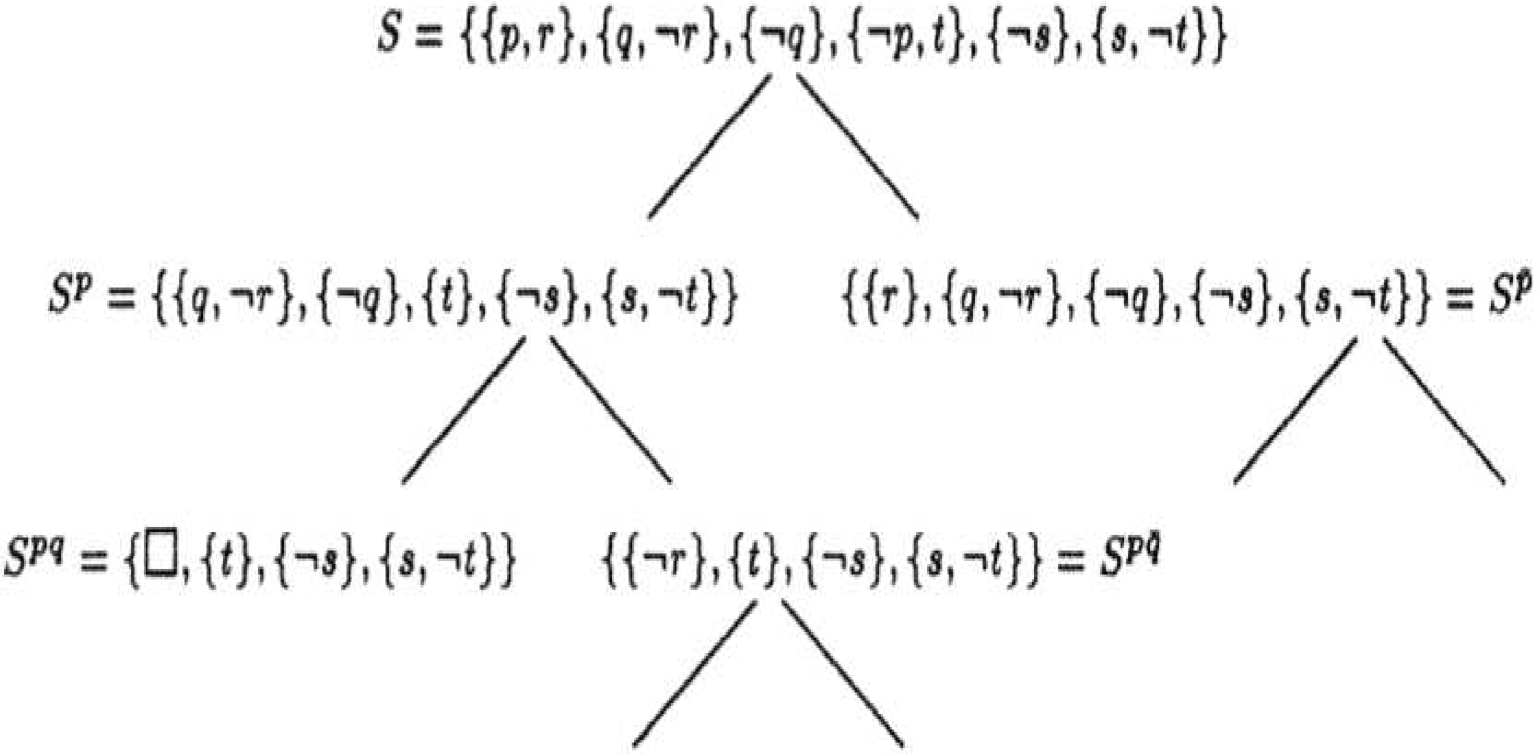
Intuition: assuming l to be true, we may omit any clause containing l from S as far as satisfiability is concerned, $\bar{l}$ can be omitted from any clause C containing it without changing the satisfiability of C. If l is false, then $\bar{l}$ is true and the same analysis applies to $S^{\bar{l}}$. As one of l and $\bar{l}$ must be true, S is satisfiable if and only if one of S^l and $S^{\bar{l}}$ is satisfiable.

Example: constructing S^l

Let $S = \{\{p\}, \{\neg q\}, \{\neg p, \neg q\}\}$. The analysis in which we eliminate first p and then q can be represented below:


S
$S^p = \{\{\neg q\}\}$
$\{\Box, \{\neg q\}\} = S^{\bar{p}}$
$S^{pq} = \{\Box\}$
$\emptyset = S^{p\bar{q}}$

Example: constructing S^l



$S = \{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{\neg s\}, \{s, \neg t\}\}$

$S^p = \{\{q, \neg r\}, \{\neg q\}, \{t\}, \{\neg s\}, \{s, \neg t\}\}$

$\{\{r\}, \{q, \neg r\}, \{\neg q\}, \{\neg s\}, \{s, \neg t\}\} = S^{\bar{p}}$

$S^{pq} = \{\Box, \{t\}, \{\neg s\}, \{s, \neg t\}\}$

$\{\{\neg r\}, \{t\}, \{\neg s\}, \{s, \neg t\}\} = S^{p\bar{q}}$

Satisfiability of S

Lemma 8.14 S is satisfiable if and only if either S^l or $S^{\bar{l}}$ is satisfiable.

Proof: ($\Rightarrow$) Suppose that $\mathcal{A} \models S$. If $\mathcal{A}$ were a complete assignment, we could conclude that it must make one of $l, \bar{l}$ true, say l. We could then show that $\mathcal{A} \models S^l$. If we do not wish to make this assumption on $\mathcal{A}$, we instead start with the fact that, by definition, one of l or $\bar{l}$ does not belong to $\mathcal{A}$. For the sake of definiteness assume that $\bar{l} \notin \mathcal{A}$. We now also claim that $\mathcal{A} \models S^l$. We must show that $\mathcal{A}$ satisfies every clause in S^l. Consider any $C \in S^l$. By the definition of S^l, either $C \cup \{\bar{l}\} \in S$ or $C \in S$ (depending on whether or not $\bar{l}$ is in the clause of S which "puts" C into S^l). Thus, by hypothesis, $\mathcal{A} \models C$ or $\mathcal{A} \models C \cup \{\bar{l}\}$. As an assignment satisfies a clause only if it contains one of its literals, there is a literal k such that either $k \in C \cap \mathcal{A}$ or $k \in (C \cup \{\bar{l}\}) \cap \mathcal{A}$. As $\bar{l} \notin \mathcal{A}$ by our assumption, in either case we must have $k \in C \cap \mathcal{A}$, i.e., $\mathcal{A} \models C$ as required. The case that $l \notin \mathcal{A}$ is handled similarly.

($\Leftarrow$) Suppose for definiteness that $\mathcal{A} \models S^l$. Now neither l nor $\bar{l}$ appear in any clause of S^l and so we may adjust $\mathcal{A}$ on l as we choose without disturbing the satisfiability of S^l. More precisely, if we let $\mathcal{A}' = (\mathcal{A} - \{\bar{l}\}) \cup \{l\}$, then $\mathcal{A}' \models S^l$ as well. We claim that $\mathcal{A}' \models S$. Consider any $C \in S$. If $l \in C$, then $\mathcal{A}' \models C$ as $l \in \mathcal{A}'$. If $l \notin C$ then $C - \{\bar{l}\} \in S^l$ by definition of S^l. As $\mathcal{A} \models S^l$, there is some literal $k \in (C - \{\bar{l}\}) \cap \mathcal{A}$. Now $\mathcal{A}$ and $\mathcal{A}'$ differ at most at l and $\bar{l}$. As $k \neq l$ or $\bar{l}$, we see that $k \in \mathcal{A}' \cap C$ as required. □

Corollary 8.15 S is unsatisfiable iff both S^l and $S^{\bar{l}}$ are.

Unsatisfiability

Theorem 8.16 If $\mathtt{UNSAT} = \{S \mid S \text{ is an unsatisfiable formula}\}$, then $\mathtt{UNSAT}$ is the collection $\mathcal{U}$ of formulas defined inductively by the following clauses:

(i) $\Box \in S \;\Rightarrow\; S \in \mathcal{U}$ and

(ii) $S^{l} \in \mathcal{U} \wedge S^{\bar{l}} \in \mathcal{U} \;\Rightarrow\; S \in \mathcal{U}$

Proof: As $\Box$ is unsatisfiable, $\mathtt{UNSAT}$ satisfies (i). By the above corollary it also satisfies (ii). Thus, $\mathcal{U} \subseteq \mathtt{UNSAT}$. We must show that $\mathtt{UNSAT} \subseteq \mathcal{U}$. We prove the contrapositive by showing that if $S \notin \mathcal{U}$, then S is satisfiable. Let $\{p_i\}$ list the propositional letters such that p_i or $\bar{p}_i$ occurs in a clause of S. Define by induction the sequence $\{l_i\}$ such that $l_i = p_i$ or $\bar{p}_i$ and $S^{l_1,\ldots,l_i} \notin \mathcal{U}$. (Property (ii) guarantees that we can always find such an l_i.) Now let $\mathcal{A} = \{l_i \mid i \in \mathcal{N}\}$. We claim that $\mathcal{A}$ satisfies S. Suppose $C \in S$. We must show that $C \cap \mathcal{A} \neq \emptyset$. As C is finite, there is an n such that for all propositional letters p_i occurring in C, $i < n$. If $C \cap \mathcal{A} = \emptyset$, then $\forall i < n(l_i \notin C)$ and so a clause corresponding to C is passed on to each $S^{l_1,\ldots,l_i}$, for $i < n$. At each such transfer, say to $S^{l_1,\ldots,l_n}$, we remove $\bar{l}_i$ from the clause. As all literals in C are among the $\bar{l}_i$, the clause deriving from C becomes $\Box$ in $S^{l_1,\ldots,l_n}$. By our choice of the l_i, $S^{l_1,\ldots,l_n} \notin \mathcal{U}$. On the other hand, any S containing $\Box$ is in $\mathcal{U}$ by Clause (i) and we have our desired contradiction. $\Box$

Completeness of the resolution method

Theorem 8.17 [Completeness] If S is unsatisfiable, then there is a resolution refutation of S (equivalently, $\Box \in \mathcal{R}(S)$).

Proof: We proceed by induction according to the characterization of `UNSAT` provided by Theorem 8.16. Of course, if $\Box \in S$, then $\Box \in \mathcal{R}(S)$. For the inductive step, suppose that, for some l and S, $\Box \in \mathcal{R}(S^l)$ and $\Box \in \mathcal{R}(S^{\bar{l}})$. We must show that $\Box \in \mathcal{R}(S)$. By assumption, we have tree proofs T_0 and T_1 of $\Box$ from S^l and $S^{\bar{l}}$. Consider T_0. If every leaf in T_0 is labeled with a clause in S, then T_0 is already a proof of $\Box$ from S. If not, we define a tree T_0' by changing every label C on T_0 that is above a leaf labeled with a clause not in S to $C \cup \{\bar{l}\}$. We claim that T_0' is a tree proof of $\{\bar{l}\}$ from S. Clearly, by the definition of S^l, every leaf of T_0' is in S. We must now check that every nonleaf node of T_0' is labeled with a resolvent C' of its immediate successors C_0' and C_1'. Suppose they correspond to clauses C, C_0 and C_1, respectively, on T_0. As T_0 is a resolution tree proof, C is a resolvent of C_0 and C_1. Note first that no resolution in T_0 is on l or $\bar{l}$ as neither appear in any label on T_0 (by the definition of S^l). Next, consider the possible forms of clauses C'_0, C'_1 and C' on T'_0. If, for example, both C_0 and C_1 (and hence certainly C) are above leaves labeled with clauses not in S, then $C' = C \cup \{\bar{l}\}$ is the resolvent of $C'_0 = C_0 \cup \{\bar{l}\}$ and $C'_1 = C_1 \cup \{\bar{l}\}$, as is required for T'_0 to be a resolution tree proof. The other cases to consider either keep all three clauses the same in T'_0 as they were in T_0 or change C and precisely one of C_0 and C_1 by adding on $\{\bar{l}\}$. In all these cases C' is still clearly the resolvent of C'_0 and C'_1 and we again verify that T'_0 is a tree proof. Similarly, if we replace every label C on a node of T_1 above a leaf labeled with a clause not in S by $C \cup \{l\}$, we get T'_1, a tree proof of $\{l\}$ from S (or, if all leaves were in S, one of $\Box$). We can now define a tree proof T of $\Box$ from S by simply attaching T'_0 and T'_1 to the immediate successors of the root node of T which we label with $\Box$. As $\Box$ is a resolvent of $\{l\}$ and $\{\bar{l}\}$, the resulting tree T is a proof of $\Box$ from S. $\Box$

Compactness revisited

Theorem 8.18 If S is unsatisfiable, so is some finite subset of S.

Proof: Let $\mathcal{T} = \{S \mid \exists S_1 \subseteq S[S_1 \text{ is finite } \wedge S_1 \text{ is unsatisfiable }]\}$. If we can show that $\mathcal{T}$ satisfies (i) and (ii) of Theorem 8.16, then we are done for it will then contain all unsatisfiable formulas.

(i) If $\Box \in S$, then $S_1 = \{\Box\} \subseteq S$ shows that $S \in \mathcal{T}$ as required.

(ii) Suppose $S^l, S^{\bar{l}} \in \mathcal{T}$. We must show that $S \in \mathcal{T}$. By definition of $\mathcal{T}$, S^l and $S^{\bar{l}}$, there are finite formulas $S_1, S_2 \subseteq S$ such that $S_1^l \subseteq S^l, S_2^{\bar{l}} \subseteq S^{\bar{l}}$, and S_1^l and $S_2^{\bar{l}}$ are unsatisfiable. Let $S_3 = S_1 \cup S_2$. S_3 is a finite subset of S. It suffices to show that it is unsatisfiable. If not, then there would be an assignment $\mathcal{A}$ satisfying S_3. Now $\mathcal{A}$ must omit either l or $\bar{l}$. Thus, $\mathcal{A}$ would satisfy either $S_3^{\bar{l}}$ or S_3^l, respectively. As it would then satisfy $S_2^{\bar{l}}$ or S_1^l (as $S_3 \supset S_2, S_1$), we have the desired contradiction. □

Exercises

1. Exercise 9 in page 61
2. Exercise 10 in page 61
3. Exercise 11 in page 62
4. Exercise 12 in page 62

9 Refining Resolution

Resolution means to search systematically for a resolution refutation of a given S. A major concern is then developing ways to limit the search space (preferably without giving up soundness or completeness although in actual applications both are often sacrificed).

Note that $SAT = \{S \mid S \text{ is satisfiable}\}$ is NP-complete in the sense of complexity theory. Nonetheless, in practice smaller search spaces tend to correspond to faster run times.

T-Resolutions

Definition 9.1 T-resolutions are resolutions in which neither of the parent clauses is a tautology. $\mathcal{R}^T(S)$ is the closure of S under T-resolutions.

Lemma 9.2 Any restriction of a sound method, i.e., one that allows fewer deductions than the sound method, is itself sound. In particular, as resolution is sound, so is $\mathcal{R}^T$, i.e., if $\Box \in \mathcal{R}^T(S)$, S is unsatisfiable.

Proof: As any deduction in the restricted system is one in the original system and by soundness there is no deduction of $\Box$ in the original one, there is none in the restricted system. □

Completeness of T-Resolution

Theorem 9.3 If S is unsatisfiable, then $\Box \in \mathcal{R}^T(S)$.

Proof: The proof of the completeness of resolution given in Theorem 8.17 remains correct for $\mathcal{R}^T$. The only remark needed is that if T_0 and T_1 have no tautologies on them, then neither do the trees T_0' and T_1' gotten by adding $\bar{l}$ and l, respectively, to the appropriate clauses. The point here is that no clause on T_0 (T_1) contains l ($\bar{l}$) by assumption as T_0 (T_1) is a proof from S^l ($S^{\bar{l}}$). □

$\mathcal{A}$-resolutions

Definition 9.4 Let $\mathcal{A}$ be an assignment. An $\mathcal{A}$-resolution is a resolution in which at least one of the parents is false in $\mathcal{A}$. $\mathcal{R}^{\mathcal{A}}$ is the closure of S under $\mathcal{A}$-resolutions. This procedure is often called semantic resolution.

Completeness of $\mathcal{A}$-resolutions

Theorem 9.5 For any $\mathcal{A}$ and S, if $S \in \mathtt{UNSAT}$, then $\Box \in \mathcal{R}^{\mathcal{A}}(S)$.

Proof: Fix an assignment $\mathcal{A}$ and let $\mathcal{T}^{\mathcal{A}} = \{S \mid \Box \in \mathcal{R}^{\mathcal{A}}(S)\}$. We must show that $\mathtt{UNSAT} \subseteq \mathcal{T}^{\mathcal{A}}$. By the characterization of $\mathtt{UNSAT}$ of Theorem 8.16 it suffices to prove that

(i) $\Box \in S \;\Rightarrow\; S \in \mathcal{T}^{\mathcal{A}}$ and

(ii) For any S and l, if $S^{l} \in \mathcal{T}^{\mathcal{A}}$ and $S^{\bar{l}} \in \mathcal{T}^{\mathcal{A}}$, then $S \in \mathcal{T}^{\mathcal{A}}$.

(i) is immediate. For (ii) consider the $\mathcal{A}$-resolution proofs T_0 and T_1 of $\Box$ from S^{l} and $S^{\bar{l}}$, respectively. We can form T_0' (T_1') as in the proof of Theorem 9.3 before by adding $\bar{l}(l)$ to the appropriate clauses of T_0 (T_1). The resulting trees are, of course, resolution proofs of $\{\bar{l}\}$ and $\{l\}$, respectively (or perhaps of $\Box$). They may not, however, be $\mathcal{A}$-resolutions since one of $\bar{l}, l$ may be true in $\mathcal{A}$. On the other hand, as at most one of $l, \bar{l}$

is true in $\mathcal{A}$, at least one of T_0' and T_1' is an $\mathcal{A}$-resolution proof. For definiteness say that $l \notin \mathcal{A}$ and so T_1' is an $\mathcal{A}$-resolution proof of $\{l\}$ or $\Box$ from S. In the latter case we are done. In the former, we can combine this proof of $\{l\}$ with T_0 to get the desired $\mathcal{A}$-resolution proof of $\Box$ as follows: To each leaf C of T_0 that is not in S attach as children $C \cup \{\bar{l}\}$ and $\{l\}$. As $l \notin \mathcal{A}$, this is an $\mathcal{A}$-resolution. Since $C \notin S$, $C \cup \{\bar{l}\}$ is in S. Thus, except for the fact that $\{l\}$ may not be in S, we have the desired $\mathcal{A}$-resolution proof of $\Box$ from S. We finish the construction of the required proof by attaching a copy of the tree T_1' below each leaf labeled with $\{l\}$. The resulting tree is now easily seen to represent an $\mathcal{A}$-resolution deduction of $\Box$ from S. Other than the resolutions of $\{l\}$ and nodes of the form $C \cup \{\bar{l}\}$ that we have just considered, all the resolutions appearing in this new proof appear in one of the $\mathcal{A}$-resolution deduction trees T_0 or T_1'. Thus, every resolution appearing on the tree is an $\mathcal{A}$-resolution. $\Box$

Ordered resolution

Definition 9.6 Assume that we have indexed all the propositional letters. We define $\mathcal{R}^{<}(S)$, for ordered resolution, as usual except that we only allow resolutions of $C_1 \sqcup \{p\}$ and $C_2 \sqcup \{\bar{p}\}$ when p has a higher index than any propositional letter in C_1 or C_2.

Completeness of ordered resolution

Theorem 9.7 `UNSAT` is equal to the class of formulas $\mathcal{U}^{<}$ defined inductively by the following clauses:

(i) $\Box \in S \;\Rightarrow\; S \in \mathcal{U}^{<}$ and

(ii$^{<}$) If no propositional letter with index strictly smaller than that of p occurs in S, $S^{p} \in \mathcal{U}^{<}$ and $S^{\bar{p}} \in \mathcal{U}^{<}$, then $S \in \mathcal{U}^{<}$.

Proof: As the inductive clause (ii$^{<}$) is weaker than (ii) of Theorem 8.16, $\mathcal{U}^{<}$ is surely contained in $\mathcal{U} = \mathtt{UNSAT}$. On the other hand, if we list the $\{p_i\}$ occurring in S in ascending order of their indices, then the original proof of the characterization of `UNSAT` (Theorem 8.16) actually shows that any $S \notin \mathcal{U}^{<}$ is satisfiable and so `UNSAT` is also contained in $\mathcal{U}^{<}$. □

Theorem 9.8 [Completeness of ordered resolution] If S is unsatisfiable, then there is an ordered resolution refutation of S, i.e., $\Box \in \mathcal{R}^{<}(S)$.

10 Linear Resolution, Horn Clauses and PROLOG

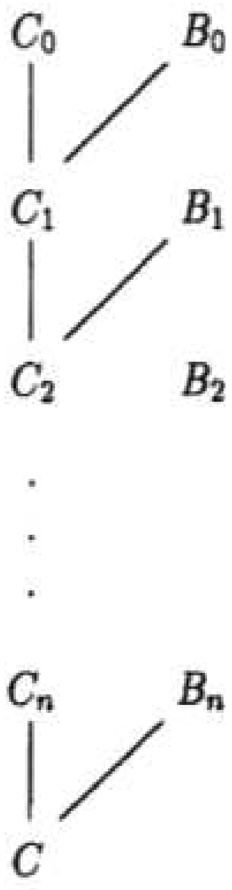
Definition 10.1 (i) A linear (resolution) deduction or proof of C from S is a sequence of pairs $\langle C_0, B_0\rangle, ..., \langle C_n.B_n\rangle$ such that $C = C_{n+1}$ and

(1) C_0 and each B_i are elements of S or some C_j with $j < i$,

(2) each C_{i+l}, $i \leq n$, is a resolvent of C_i, and B_i.

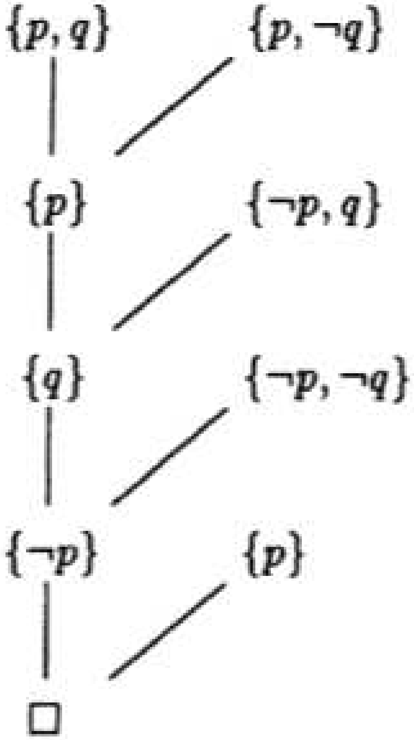
(ii) As usual we say that C is linearly deducible (or provable) from S, $S \vdash_{\mathcal{L}} C$, if there is a linear deduction of C from S. There is a linear refutation of S if $S \vdash_{\mathcal{L}} \Box$. $\mathcal{L}(S)$ is the set of all clauses linearly deducible from S.

Linear Resolution



Example: Linear Resolution

Let $S = \{A_1, A_2, A_3, A_4\}$, $A_1 = \{p, q\}$, $A_2 = \{p, \neg q\}$, $A_3 = \{\neg p, q\}$, $A_4 = \{\neg p, \neg q\}$. Below is a linear refutation of S.



Linear Resolution

Definition 10.2 In the context of linear resolution, the elements of the set S from which we are making our deductions are frequently called input clauses. The C_i are called center clauses and the B_i side clauses. C_0 is called the starting clause of the deduction.

Horn Clause

Definition 10.3 (i) A Horn clause is a clause that contains at most one positive literal.

(ii) A program clause is one that contains exactly one positive literal. (In PROLOG notation it looks like $A : -B_1, B_2, ..., B_n.$)

(iii) If a program clause contains some negative literals it is called a rule ($n > 0$ in the notation of (ii)).

(iv) A fact (or unit clause) is one that consists of exactly one positive literal (Notation: $A.$ or $A : -.$).

(v) A goal clause is one that contains no positive literals. (Thus, in PROLOG it is entered as a question with the symbol $?-.$)

(vi) A PROLOG program is a set of clauses containing only program clauses (rules or facts).

Horn clauses are either program or goal clauses while program clauses are either rules or facts.

Horn Clause

Lemma 10.4 If a set of Horn clauses S is unsatisfiable, then S must contain at least one fact and one goal clause.

Proof: The assignment that makes every propositional letter true satisfies every program clause. The assignment that makes every propositional letter false satisfies every goal clause and every rule. Thus, any unsatisfiable set of Horn clauses must contain both a fact and a goal clause. □

Horn Clause

Lemma 10.5 If P is a PROLOG program and $G = \{\neg q_1, \neg q_2, ..., \neg q_n\}$ a goal clause, then all of the q_i are consequences of P if and only if $P \cup \{G\}$ is unsatisfiable.

Proof: The proof simply consists of tracing through the definitions. First note that $P \cup \{G\}$ is unsatisfiable if and only if any assignment satisfying P makes G false. Next note that the goal clause G is false iff none of the $\neg q_i$ are true, i.e., G is false iff all the q_i are true. Thus, our desired conjunction of facts is a consequence of our assumptions P just in case $P \cup \{G\}$ is unsatisfiable. □

Completeness of linear resolution for Horn Clause

Theorem 10.6 If S is an unsatisfiable set of Horn clauses, then there is a linear resolution deduction of $\Box$ from S, i.e., $\Box \in \mathcal{L}(S)$.

Proof: By the compactness theorem (Theorem 8.18) we may assume that S is finite. We proceed by induction on the number of literals in S. By Lemma 10.4 we know that there is at least one positive literal p occurring as a fact $\{p\}$ in S. Consider the formula S^p as described in Definition 8.16. Each clause in S^p is a subset of one in S and so is Horn by definition. We claim that S^p is unsatisfiable. The point here is that, if $\mathcal{A} \models S^p$, then $\mathcal{A} \cup \{p\} \models S$ contradicting the unsatisfiability of S. As S^p contains fewer literals than S (we omit any clause containing p and remove $\bar{p}$ from every other clause), we may apply the induction hypothesis to S^p to get a linear resolution deduction of $\Box$ from S^p. As in the inductive step of the proof of the completeness theorem for the general resolution method given for

Theorem 8.17, either this is already a linear proof of $\Box$ from S or we can convert it into one of $\{\bar{p}\}$ from S by adding $\bar{p}$ to every clause below one not in S. We can now extend this proof one step by adding on $\{p\} \in S$ as a new side clause and resolving against the last center clause $\{\bar{p}\}$ to get $\Box$ as required. $\Box$

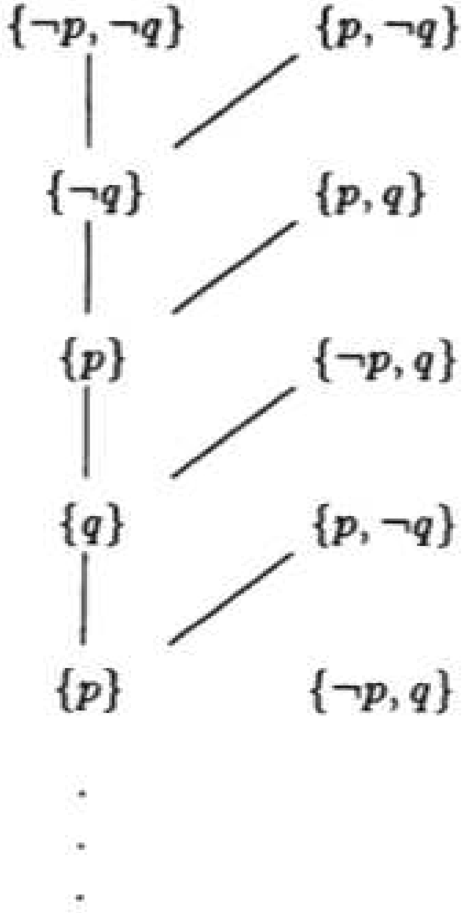
Linear input (LI) resolution

Definition 10.7 Let P be a set of program clauses and G a goal clause. A linear input (LI) resolution refutation of $S = P \cup \{G\}$ is a linear resolution refutation of S that starts with G and in which all the side clauses are from P (input clauses).

LI-resolution

The method of LI-resolution is not complete in general as may be seen from the following example.

Recall the clauses of Example 10.2:
$S = \{A_1, A_2, A_3, A_4\}, A_1 = \{p, q\}, A_2 = \{p, \neg q\}, A_3 = \{\neg p, q\}, A_4 = \{\neg p, \neg q\}$.
The only goal clause here is A_4 which we set equal to G. The remaining clauses are, however, not all program clauses. If we set $P = \{A_1, A_2, A_3\}$ and try to produce a linear input resolution refutation of $S = P \cup \{G\}$ beginning with G, we are always thwarted.



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

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

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

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

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

$\vdots$

LI-resolution

Lemma 10.8 If T is a set of Horn clauses, G a goal clause such that $T \cup \{G\} \in \mathtt{UNSAT}$ but $T \in \mathtt{SAT}$, then there is a linear resolution deduction of $\Box$ from $T \cup \{G\}$ starting with G.

Proof: As before, we may assume that T is finite by the compactness theorem. We proceed by induction on the number of literals in T. As in the proof of Theorem 10.6, we know that T contains a fact $\{p\}$ for some positive literal p and that $T' = (T \cup \{G\})^p = T^p \cup \{G\}^p$ is an unsatisfiable set of Horn clauses. (As G is a goal clause, it contains no positive literals and so $\{G\}^p$ is just $\{G - \{\bar{p}\}\}$.) As T was satisfiable and contained $\{p\}$, T^p is satisfiable by the same assignment that satisfied T (by the proof of the "only if" direction of Lemma 8.14). Thus, we may apply the induction hypothesis to T' to get a linear proof of $\Box$ from T' starting with $G - \{\bar{p}\}$. If this proof is not already the desired one of $\Box$ from T starting with G, we may, as in the proofs of Theorem 8.17 or 10.6, convert it into a proof of $\{\bar{p}\}$ from T starting with G. We can again extend this proof one step by adding on $\{p\} \in T$ as a new side clause at the end to do one more resolution to get $\Box$ as desired. $\Box$

LI-resolution

Theorem 10.9 Let P be a set of program clauses and G be a goal clause. If $S = P \cup \{G\} \in \mathtt{UNSAT}$, there is a linear input resolution refutation of S.

Proof: As any set of program clauses is satisfiable by Lemma 10.5, the above lemma suffices to prove this theorem. □

LD-resolution refutation

Definition 10.10 Let $P \cup \{G\}$ be a set of program clauses, then an LD-resolution refutation of $P \cup \{G\}$ is a sequence $\langle G_0, C_0 \rangle, ..., \langle G_n, C_n \rangle$ of ordered clauses G_i, C_i such that $G_0 = G$, $G_{n+1} = \Box$, and

(i) Each $G_i, i \leq n$, is an ordered goal clause $\{\neg A_{i,0}, ..., \neg A_{i,n(i)}\}$ of length $n(i) + 1$.

(ii) Each $C_i = \{B_i, \neg B_{i,0}, ..., \neg B_{i,m(i)}\}$ is an ordered program clause of length $m(i) + 2$ from P. (We include the possibility that $C_i = \{B_i\}$, i.e., $m(i) = -1$.)

(iii) For each $i < n$, there is a resolution of G_i and C_i as ordered clauses with resolvent the ordered clause G_{i+1} (of length $n(i) + m(i) + 1$) given by $\{\neg A_{i,0}, ..., \neg A_{i,k-1}, \neg B_{i,0}, ..., \neg B_{i,m(i)}, \neg A_{i,k+1}, ..., \neg A_{i,n(i)}\}$. (In this resolution we resolve on $B_i = A_{i,k}$.)

LD-resolution refutation

Lemma 10.11 If $P \cup \{G\} \in \texttt{UNSAT}$, then there is an LD-resolution refutation of $P \cup \{G\}$ starting with G.

Proof: Proceed by induction on the length of the LI-resolution refutation of $P \cup \{G\}$. (Note that we can only resolve a program clause and a goal clause at each step of the resolution. Each center clause must be a goal clause and each side one a program clause.) □

SLD-resolution refutation

Definition 10.12 An SLD-resolution refutation of $P \cup \{G\}$ via (the selection rule) R is an LD-resolution proof $\langle G_0, C_0\rangle, ..., \langle G_n, C_n\rangle$ with $G_0 = G$ and $G_{n+1} = \Box$ in which $R(G_i)$ is the literal resolved on at the $(i+1)$ step of the proof. (If no R is mentioned we assume that the standard one of choosing the leftmost literal is intended.)

Completeness of SLD-refutation

Theorem 10.13 If $P \cup \{G\} \in \texttt{UNSAT}$ and R is any selection rule, then there is an SLD-resolution refutation of $P \cup \{G\}$ via R.

Proof: By Lemma 10.11, there is an LD-resolution refutation of $P \cup \{G\}$ starting with G. We prove by induction on the length n of such proofs (for any P and G) that there is an SLD one via R. For $n = 1$ there is nothing to prove as $G = G_0$ is a unit clause and so every R makes the same choice from G_0. Let $\langle G_0, C_0\rangle, ..., \langle G_n, C_n\rangle$, with the notation for these clauses as in Definition 10.12, be an LD-resolution refutation of length n of $\{G_0\} \cup P$. Suppose that the selection rule R chooses the clause $\neg A_{0,k}$ from G_0. As $G_{n+1} = \Box$ there must be a $j < n$ at which we resolve on $\neg A_{0,k}$. If $j = 0$, we are done by induction. Suppose then that $j \geq 1$. Consider the result C of resolving G_0 and $C_j = \{B_j, \neg B_{j,0}, ..., \neg B_{j,m(j)}\}$ on $B_j = A_{0,k}$:

$$C = \{\neg A_{0,0}, ..., \neg A_{0,k-1}, \neg B_{j,0}, ..., \neg B_{j,m(j)}, \neg A_{0,k+1}, ..., \neg A_{0,n(0)}\}$$

We claim that there is an LD-resolution refutation of length $n-1$ from $P \cup \{C\}$ that begins with C. One simply resolves in turn with $C_0, ..., C_{j-1}$ on the same literals as in the original proof that started with G. The only change is that we carry along the sequence of clauses $\neg B_{j,0}, ..., \neg B_{j,m(j)}$ in place of $\neg A_{0,k}$ in the center clauses of the resolution. After resolving with each side clause $C_0, ..., C_{j-1}$, we have precisely the same result G_{j+1} as we had in the original resolution after resolving with C_j. We can then continue the resolution deduction exactly as in the original resolution with $C_{j+1}, ..., C_n$. This procedure produces an LD-resolution refutation of length $n-1$ beginning with C. By induction, it can be replaced by an SLD-resolution refutation via R. Adding this SLD-resolution via R onto the single step resolution of G_0 with C_j described above produces the desired SLD-resolution refutation from $P \cup \{G\}$ via R starting with $G = G_0$. □

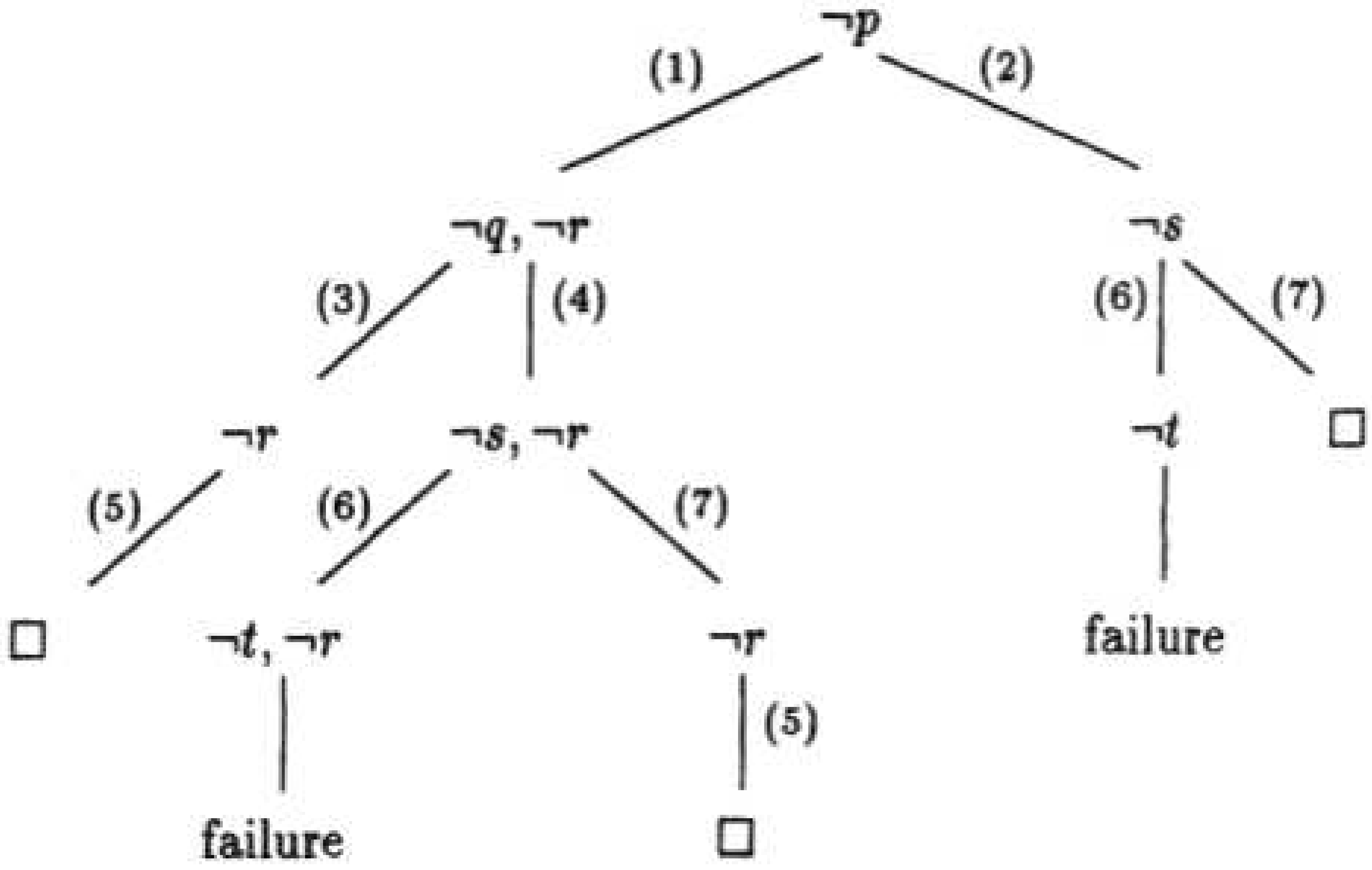
SLD-Trees

We can display the space of all possible SLD-derivations as a labeled tree T. The root of T is labeled G. If any node of T is labeled G', then its immediate successors are labeled with the results of resolving on the leftmost literal of G' with the various possible choices of clauses in P. We call such trees SLD-trees for P and G.

Example: SLD-Trees

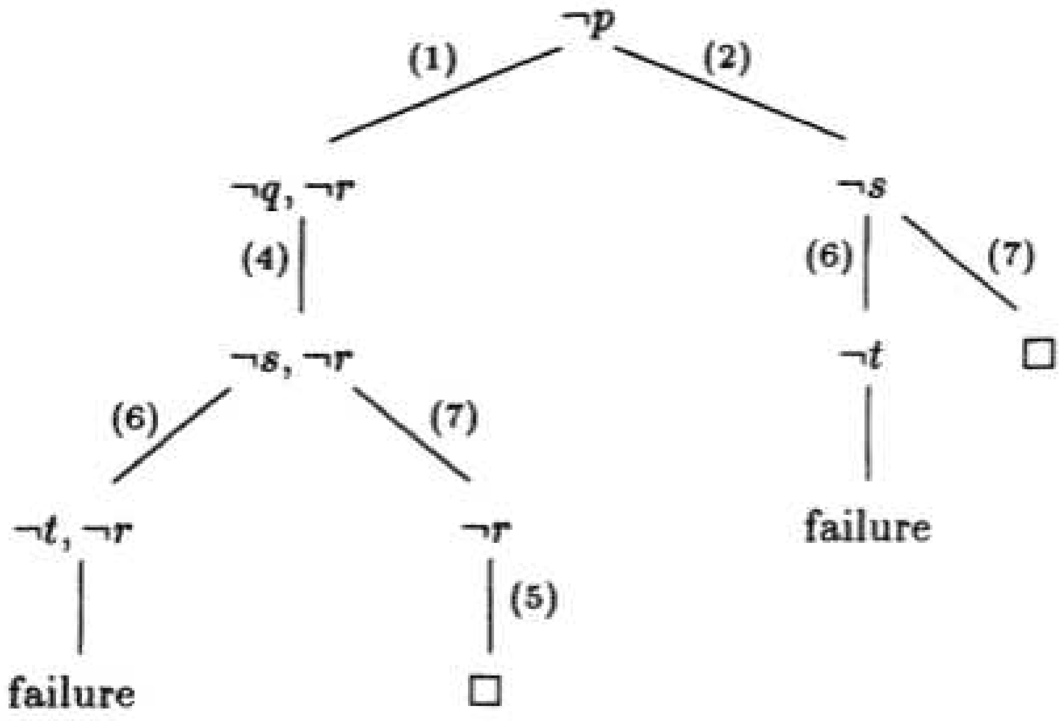
Consider the program P_0:

$$
\begin{aligned}
p :- q, r. &\quad (1)\\
p :- s. &\quad (2)\\
q. &\quad (3)\\
q :- s. &\quad (4)\\
r. &\quad (5)\\
s :- t. &\quad (6)\\
s. &\quad (7)
\end{aligned}
$$


¬p
(1)
(2)
¬q, ¬r
¬s
(3)
(4)
(6)
(7)
¬r
¬s, ¬r
¬t
□
(5)
(6)
(7)
failure
□
¬t, ¬r
¬r
(5)
failure
□

Backtracking

If we omit clause (3) from the above program P_0 to produce P_1 we get a new SLD-tree.



In this case, the theorem prover first tries the path (1), (4), (6), failure. It then backtracks to $\neg s, \neg r$ and tries (7), (5), success, to give the answer yes.

Failure of depth-first searching

Consider the following simple program:

$$q :- r. \quad (1)$$
$$r :-q. \quad (2)$$
$$q. \quad (3)$$

The usual search procedure applied to the starting clause $\neg q$ will loop back and forth between $\neg q$ and $\neg r$. It will never find the contradiction supplied by (3).

Exercises

1. Exercise 3 in page 77
2. Exercise 6 in page 78

Chapter II. Predicate Logic

1. Predicates and Quantifiers

A predicate represents a property holding of an object or a relation holding between objects.

Let $\varphi(x, y)$ denote the relation (predicate) "x is less than y". Then $\varphi(3, y)$ denotes the property (unary predicate) of y "3 is less than y" and $\varphi(3, 4)$ denotes the proposition (0-ary predicate) "3 is less than 4".

Terms stand for the symbols generated by the function symbols, constants and variables such as $f(x, g(y, y))$.

The universal quantifier, $\forall$

The existential quantifier, $\exists$

0-ary predicates are usually called sentences rather than propositions.

Predicates and Quantifiers

Let the domain of discourse consist of all rational numbers Q. $\varphi(x, y)$ denotes $x < y$, $f(x, y)$ represents addition $(x + y)$, $g(x, y)$ division (x/y) and c is the constant representing 2.

$(\forall x)(\forall y)(\varphi(x, y) \to (\varphi(x, g(f(x, y), c)) \wedge \varphi(g(f(x, y), c), y))))$ is a sentence saying that for every x and y, if $x < y$ then $x < \frac{x+y}{c} < y$.

2. The Language: Terms and Formulas

Definition 2.1: A language $\mathcal{L}$ consists of the following disjoint sets of distinct primitive symbols:

(i) Variables: $x, y, z, v, x_0, x_1, ..., y_0, y_1, ..., ...$ (an infinite set)

(ii) Constants: $c, d, c_0, d_0, ...$ (any set of them)

(iii) Connectives: $\wedge, \neg, \vee, \rightarrow, \leftrightarrow$

(iv) Quantifiers: $\forall, \exists$

(v) Predicate symbols: $P, Q, R, P_1, P_2, ...$ (some set of them for each arity $n = 1, 2, ...$. There must be at least one predicate symbol in the language but otherwise there are no restrictions on the number of them for each arity).

(vi) Function symbols: $f, g, h, f_0.f_1, ..., g_0, ...$ (any set of them for each arity

$n = 1, 2, \ldots$. The 0-ary function symbols are simply the constants listed by convention separately in (ii). The set of constant symbols may also be empty, finite or infinite).

(vii) Punctuation: the comma , and (right and left) parentheses) , (.

Terms

Definition 2.2: Terms.

(i) Every variable is a term.

(ii) Every constant symbol is a term.

(iii) If f is an n-ary function symbol $(n = 1, 2, ...\ )$ and $t_1, ..., t_n$ are terms, then $f(t_1, ..., t_n)$ is also a term.

Definition 2.3: Terms with no variables are called variable-free terms or ground terms.

Ground terms are the constants and the terms built up from the constants by applications of function symbols as in (iii) above.

Formulas

Definition 2.4: An atomic formula is an expression of the form $R(t_1, ..., t_n)$ where R is an n-ary predicate symbol and $t_1, ..., t_n$ are terms.

Definition 2.5: Formulas.

(i) Every atomic formula is a formula.

(ii) If α, β are formulas, then so are $(\alpha \wedge \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$, $(\neg\alpha)$ and $(\alpha \vee \beta)$.

(iii) If v is a variable and α is a formula, then $((\exists v)\alpha)$ and $((\forall v)\alpha)$ are also formulas.

Subformulas

Definition 2.6:

(i) A subformula of a formula φ is a consecutive sequence of symbols from φ which is itself a formula.

(ii) An occurrence of a variable v in a formula φ is bound if there is a subformula ψ of φ containing that occurrence of v such that ψ begins with $(\forall v)$ or $(\exists v)$. (This includes the v in $\forall v$ or $\exists v$ that are bound by this definition.) An occurrence of v in φ is free if it is not bound.

(iii) A variable v is said to occur free in φ if it has at least one free occurrence there.

(iv) A sentence of predicate logic is a formula with no free occurrences of any variable, i.e., one in which all occurrences of all variables are bound.

(v) An open formula is a formula without quantifiers.

Substitution

Definition 2.7: Substitution (or Instantiation) If φ is a formula and v a variable, we write $\varphi(v)$ to denote the fact that v occurs free in φ. If t is a term, then $\varphi(t)$, or $\varphi(v/t)$, is the result of substituting (or instantiating) t for all free occurrences of v in φ. We call $\varphi(t)$ an instance of φ. If $\varphi(t)$ contains no free variables, we call it a ground instance of φ.

Definition 2.8: If the term t contains an occurrence of some variable x (which is necessarily free in t) we say that t is substitutable for the free variable v in $\varphi(v)$ if all occurrences of x in t remain free in $\varphi(v/t)$.

Example: substitution

(i) $((\forall x)R(x,y))$ is a formula in which y occurs free but x does not. The formula $((\exists y)((\forall x)R(x,y)))$ has no free variables; it is a sentence.

(ii) A variable may have both a free and a bound occurrence in a single formula as do both x and y in $(((\forall x)R(x,y)) \vee ((\exists y)R(x,y)))$.

(iii) If $\varphi(x)$ is $(((\exists y)R(x,y)) \wedge ((\forall z)\neg Q(x,z)))$ and t is $f(w,u)$, then $\varphi(t) = \varphi(x/t)$ is $(((\exists y)R(f(w,u),y)) \wedge ((\forall z)\neg Q(f(w,u),z)))$. The term $g(y,s(y))$ would, however, not be substitutable for x in $\varphi(x)$.

Unique readability for terms

Proposition 2.11: If a term s is an initial segment of a term t, $s \subseteq t$, then $s = t$.

Proof: If s is a variable or constant symbol, then the proposition is clear. Otherwise s must be of the form $f(s_1, ..., s_n)$ and so of length at least two. Now if $s \neq t$, then s is a proper initial segment of t, $s \subset t$, and we would contradict the properties of parentheses in terms (equal number of left and right parentheses). □

Theorem 2.12: (Unique readability for terms): Every term s is either a variable or constant symbol or of the form $f(s_1, ..., s_n)$ in which case f, n and the s_i for $1 \leq i \leq n$ are all uniquely determined.

Unique readability for formulas

Proposition 2.13: If a formula α is an initial segment of a formula γ, $\alpha \subseteq \gamma$, then $\alpha = \gamma$.

Theorem 2.12: (Unique readability for formulas): Each formula φ is of precisely one of the following forms: an atomic formula (i.e., of the form $R(t_1, ..., t_n)$ for an n-ary predicate symbol R and terms $t_1, ..., t_n$), $(\alpha \wedge \beta), (\alpha \rightarrow \beta), (a \leftrightarrow \beta), (\neg\alpha), (\alpha \vee \beta), ((\exists v)\alpha)$ or $((\forall v)\alpha)$ (where α and β are formulas and v a variable). Moreover, the relevant "components" of φ as displayed in each of these forms are uniquely determined (i.e., R, n and the t_i for $1 \leq i \leq n$ for an atomic formula φ and the formulas α, β and variable v as appropriate to the other possible forms for φ).

Proof: Exercise. □

3. Formation Trees, Structures and Lists

Definition 3.1:

(i) Term formation trees are ordered, finitely branching trees T labeled with terms satisfying the following conditions:

(1) The leaves of T are labeled with variables or constant symbols.

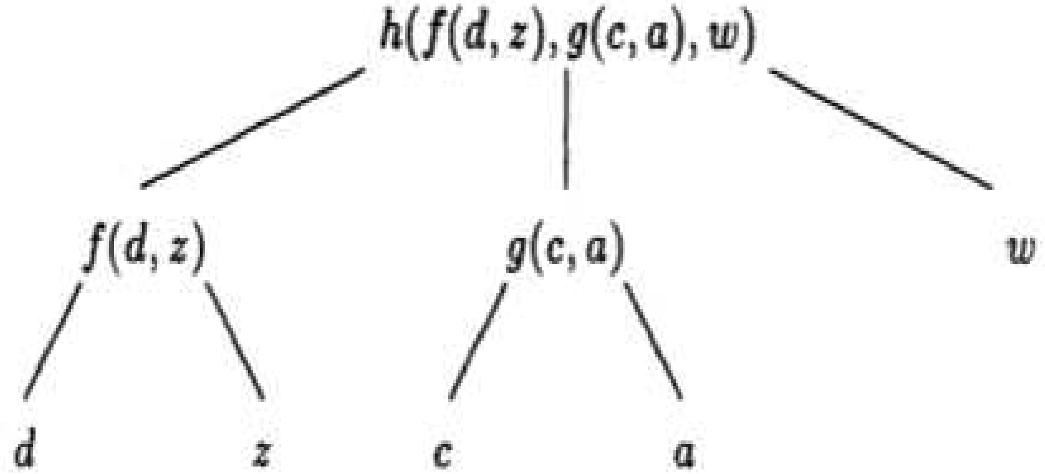
(2) Each nonleaf node of T is labeled with a term of the form $f(t_1, ..., t_n)$.

(3) A node of T that is labeled with a term of the form $f(t_1, ..., t_n)$ has exactly n immediate successors in the tree. They are labeled in (lexicographic) order with $t_1, ..., t_n$.

(ii) A term formation tree is associated with the term with which its root node is labeled.

Example: Formation Tree

Associated with $h(f(d,z),g(c,a),w)$ we have the term formation tree



Formation Trees

Proposition 3.3: Every term t has a unique formation tree associated with it.

Proposition 3.4: The ground terms are those terms whose formation trees have no variables on their leaves.

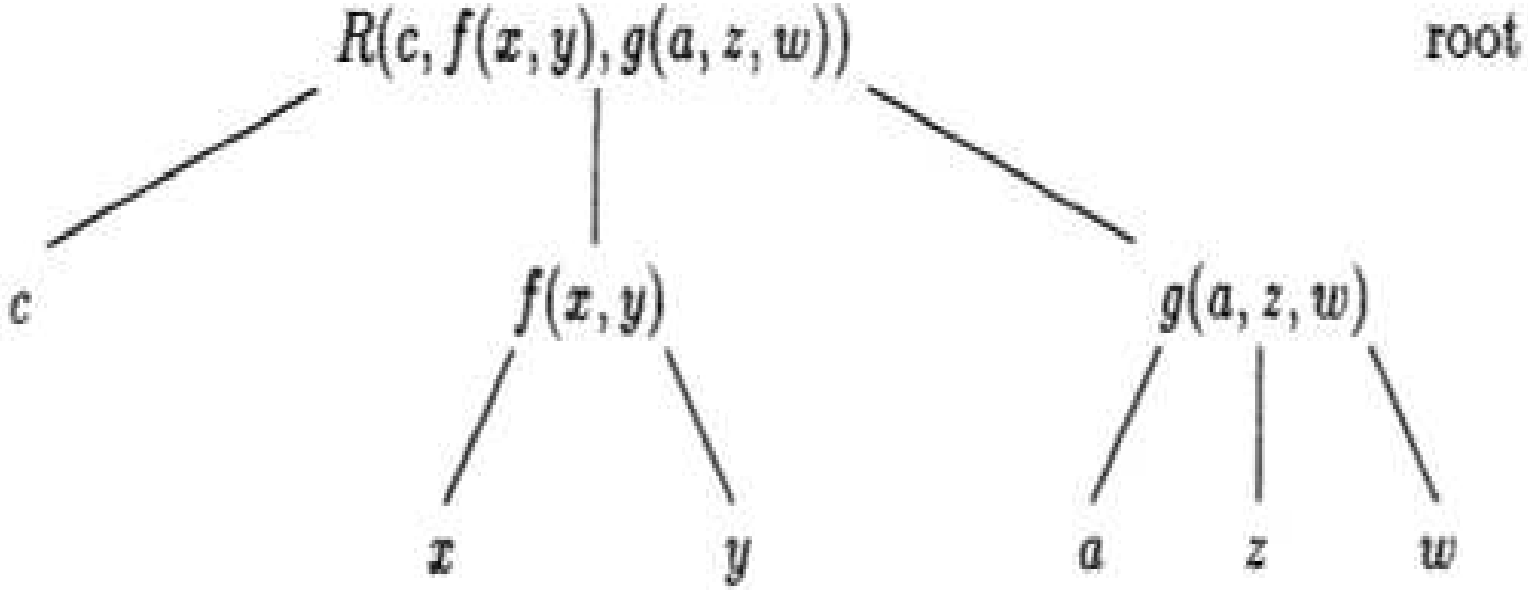
Atomic Formula Formation Trees

Definition 3.5:

(i) The atomic formula auxiliary formation trees are the labeled, ordered, finitely branching trees of depth one whose root node is labeled with an atomic formula. If the root node of such a tree is labeled with an n-ary relation $R(t_1, ..., t_n)$, then it has n immediate successors which are labeled in order with the terms $t_1, ..., t_n$.

(ii) The atomic formula formation trees are the finitely branching, labeled, ordered trees gotten from the auxiliary trees by attaching at each leaf labeled with a term t the rest of the formation tree associated with t. Such a tree is associated with the atomic formula with which its root is labeled.

Example: Atomic Formula Formation Trees



Proposition 3.7: Every atomic formula is associated with a unique formation tree.

Formula Formation Trees

Definition 3.5:

(i) The formula auxiliary formation trees are the labeled, ordered, binary branching trees T such that

(1) The leaves of T are labeled with atomic formulas.

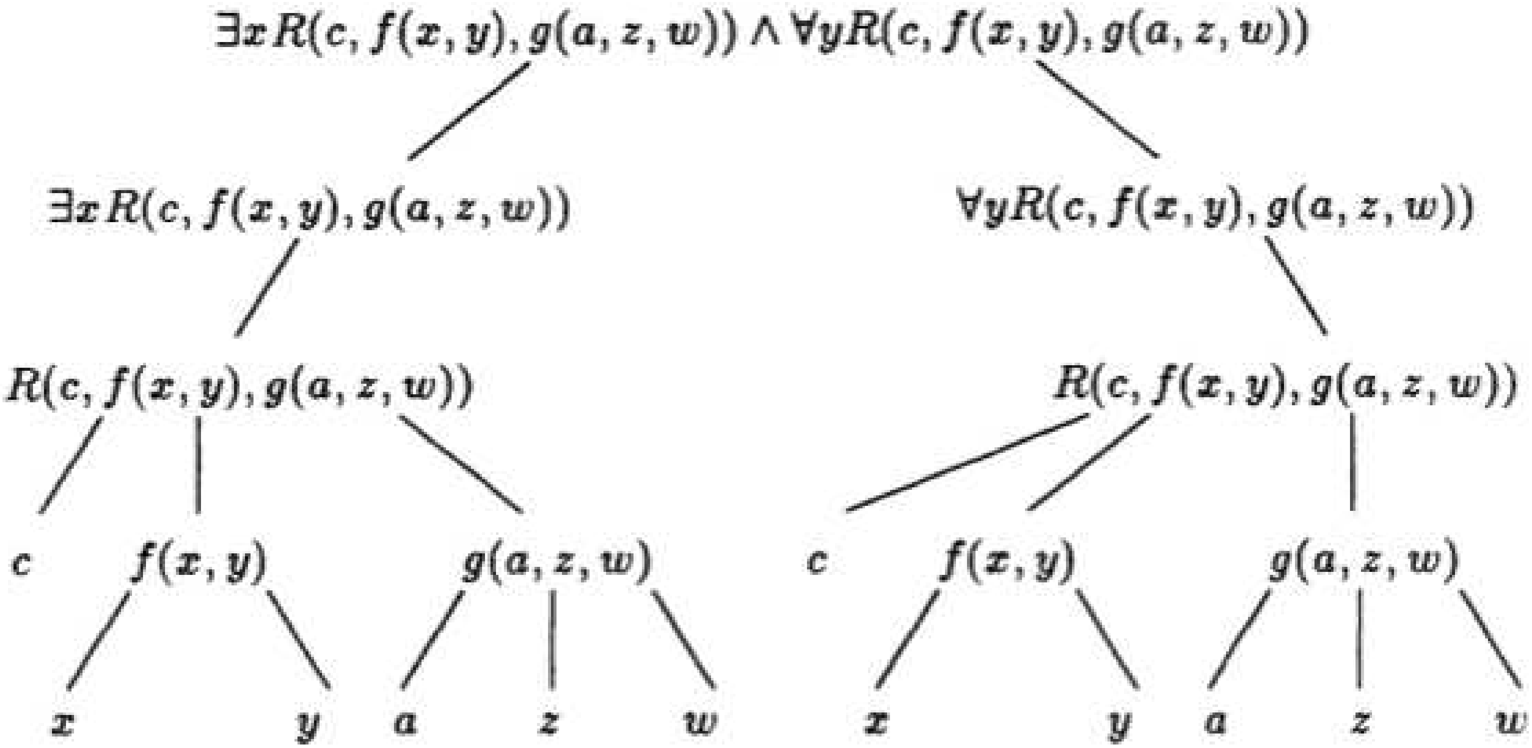
(2) If σ is a nonleaf node of T with one immediate successor $\sigma \cdot 0$ which is labeled with a formula φ, then σ is labeled with $\neg\varphi, \exists v\varphi$ or $\forall v\varphi$ for some variable v.

(3) If σ is a nonleaf node with two immediate successors, $\sigma \cdot 0$ and $\sigma \cdot 1$, which are labeled with formulas φ and ψ, then σ is labeled with $\varphi \wedge \psi$, $\varphi \vee \psi, \varphi \to \psi$ or $\varphi \leftrightarrow \psi$.

(ii) The formula formation trees are the ordered, labeled trees gotten from the auxiliary ones by attaching to each leaf labeled with an atomic formula the rest of its associated formation tree. Each such tree is again associated with the formula with which its root is labeled.

(iii) The depth of a formula: depth of the associated auxiliary formation tree.

Example: Formula Formation Trees


$\exists x R(c, f(x,y), g(a,z,w)) \wedge \forall y R(c, f(x,y), g(a,z,w))$
$\exists x R(c, f(x,y), g(a,z,w))$
$\forall y R(c, f(x,y), g(a,z,w))$
$R(c, f(x,y), g(a,z,w))$
$R(c, f(x,y), g(a,z,w))$
c
$f(x,y)$
$g(a,z,w)$
c
$f(x,y)$
$g(a,z,w)$
x
y
a
z
w
x
y
a
z
w

Proposition 3.10: Every formula is associated with a unique (auxiliary) formation tree.

Subformula

Proposition 3.11: The subformulas of a formula φ are the labels of the nodes of the auxiliary formation tree associated with φ.

Proposition 3.12:

(i) The occurrences of a variable v in a formula φ are in one-one correspondence with the leaves of the associated formation tree that are labeled with v. (The correspondence is given by matching the typographical ordering of the occurrences of v in φ with the left-right ordering given by the tree to the leaves labeled with v.) We may also refer to the appropriate leaf labeled with v as the occurrence of v in φ.

(ii) An occurrence of the variable v in φ is bound if there is a formula ψ beginning with $(\forall v)$ or $(\exists v)$ which is the label of a node above the corresponding leaf of the formation tree for φ labeled with v.

Substitution

Proposition 3.13: If φ is a formula and v a variable, then $\varphi(v/t)$ is the formula associated with the formation tree gotten by replacing each leaf in the tree for $\varphi(v)$ which is labeled with a free occurrence of v with the formation tree associated with t and propagating this change through the tree.

Proposition 3.14: The term t is substitutable for v in $\varphi(v)$ if all occurrences of x in t remain free in $\varphi(t)$, i.e., any leaf in the formation tree for t which is a free occurrence of a variable x remains free in every location in which it appears in the formation tree described in Proposition 3.9.

Structure

Except for the distinction we have made in our alphabet between function symbols and predicate symbols, the formation trees for terms and atomic formulas are indistinguishable. Terms and atomic formulas are all lumped together and called structures.

4. Semantics: Meaning and Truth

Definition 4.1: A structure $\mathcal{A}$ for a language $\mathcal{L}$ consists of a nonempty domain A, an assignment, to each n-ary predicate symbol R of $\mathcal{L}$, of an actual predicate (i.e., a relation) $R^{\mathcal{A}}$ on the n-tuples $(a_1, ..., a_n)$ from A, an assignment, to each constant symbol c of $\mathcal{L}$, of an element $c^{\mathcal{A}}$ of A and, to each n-ary function symbol f of $\mathcal{L}$, an n-ary function $f^{\mathcal{A}}$ from A^n to A.

The interpretation of ground terms

Definition 4.2:

(i) Each constant term c names the element $c^{\mathcal{A}}$.

(ii) If the terms $t_1, ..., t_n$ of $\mathcal{L}$ name the elements $t_1^{\mathcal{A}}, ..., t_n^{\mathcal{A}}$ of A and f is an n-ary function symbol of $\mathcal{L}$, then the term $f(t_1, ..., t_n)$ names the element $f(t_1, ..., t_n)^{\mathcal{A}} = f^{\mathcal{A}}(t_1^{\mathcal{A}}, ..., t_n^{\mathcal{A}})$ of A. (Remember that $f^{\mathcal{A}}$ is an n-ary function on A and that $t_1^{\mathcal{A}}, ..., t_n^{\mathcal{A}}$ are elements of A so that $f^{\mathcal{A}}(t_1^{\mathcal{A}}, ..., t_n^{\mathcal{A}})$ is in fact an element of A.)

E.g. if f is interpreted as multiplication of integers and $c^{\mathcal{A}} = 1$, then $(f(c, f(c, c)))^{\mathcal{A}} = 1$.

The interpretation of ground terms

Definition 4.3: The truth (satisfaction) of a sentence φ of $\mathcal{L}$ in a structure $\mathcal{A}$ in which every $a \in A$ is named by a ground term of $\mathcal{L}$ is defined by induction.

(i) For an atomic sentence $R(t_1, ..., t_n)$, $\mathcal{A} \models R(t_1, ..., t_n)$ iff $R^{\mathcal{A}}(t_1^{\mathcal{A}}, ..., t_n^{\mathcal{A}})$, i.e., the relation $R^{\mathcal{A}}$ on A^n assigned to R holds of the elements named by the terms $t_1, ..., t_n$. Note that, as $R(t_1, ..., t_n)$ is a sentence, the t_i are all ground terms and so name particular elements of A.

(ii) $\mathcal{A} \models \neg\varphi \;\Leftrightarrow$ it is not the case that $\mathcal{A} \models \varphi$ (We also write this as $\mathcal{A} \not\models \varphi$).

(iii) $\mathcal{A} \models (\varphi \vee \psi) \;\Leftrightarrow\; \mathcal{A} \models \varphi$ or $\mathcal{A} \models \psi$.

(iv) $\mathcal{A} \models (\varphi \wedge \psi) \;\Leftrightarrow\; \mathcal{A} \models \varphi$ and $\mathcal{A} \models \psi$.

(v) $\mathcal{A} \models (\varphi \to \psi) \;\Leftrightarrow\; \mathcal{A} \not\models \varphi$ or $\mathcal{A} \models \psi$.

(vi) $\mathcal{A} \models (\varphi \leftrightarrow \psi) \;\Leftrightarrow\;$ ($\mathcal{A} \models \varphi$ and $\mathcal{A} \models \psi$) or ($\mathcal{A} \not\models \varphi$ and $\mathcal{A} \not\models \psi$).

(vii) $\mathcal{A} \models \exists v \varphi(v) \;\Leftrightarrow\;$ for some ground term $t, \mathcal{A} \models \varphi(t)$.

(viii) $\mathcal{A} \models \forall v \varphi(v) \;\Leftrightarrow\;$ for all ground terms $t, \mathcal{A} \models \varphi(t)$.

Satisfaction of sentences

Definition 4.4: Fix some language $\mathcal{L}$.

(i) A sentence φ of $\mathcal{L}$ is valid, $\models \varphi$, if it is true in all structures for $\mathcal{L}$.

(ii) Given a set of sentences $\Sigma = \{\alpha_1, ...\}$, we say that α is a logical consequence of $\Sigma, \Sigma \models \alpha$, if α is true in every structure in which all of the members of Σ are true.

(iii) A set of sentences $\Sigma = \{\alpha_1, ...\}$ is satisfiable if there is a structure $\mathcal{A}$ in which all the members of Σ are true. Such a structure is called a model of Σ. If Σ has no model it is unsatisfiable.

Validity of formulas

Definition 4.5: A formula φ of a language $\mathcal{L}$ with free variables $v_1, ..., v_n$ is valid in a structure $\mathcal{A}$ for $\mathcal{L}$ (also written $\mathcal{A} \models \varphi$) if the universal closure of φ, i.e., the sentence $\forall v_1 \forall v_2, ... \forall v_n \varphi$ gotten by putting $\forall v_i$ in front of φ for every free variable v_i in φ, is true in $\mathcal{A}$. The formula φ of $\mathcal{L}$ is valid if it is valid in every structure for $\mathcal{L}$.

As long as we are in a situation in which every element of the structure $\mathcal{A}$ is named by a ground term, this definition of validity in $\mathcal{A}$ is equivalent to saying that every ground instance of φ is true in $\mathcal{A}$, i.e., $\mathcal{A} \models \varphi(t_1, ..., t_n)$ for all ground terms $t_1, ..., t_n$ of $\mathcal{L}$. Also note that as sentences have no free variables, a sentence is true in a structure iff it is valid in the structure.

Validity of formulas

Warning: For a sentence φ and structure $\mathcal{A}$ either φ or $\neg\varphi$ is true in $\mathcal{A}$ (and the other false). It is not true, however, for an arbitrary formula ψ that ψ or $\neg\psi$ must be valid in $\mathcal{A}$. It may well be that some ground instances of ψ are true while others are false. Similarly, one can have a sentence such that neither it nor its negation is valid. It is true in some structures but not in others.

Definition 4.6: A set Σ of formulas with free variables is satisfiable if there is a structure in which all of the formulas in Σ are valid (i.e., their universal closures are true). Again such a structure is called a model of Σ. If Σ has no models it is unsatisfiable.

Example: Validity of formulas

Consider a language $\mathcal{L}$ specified by a binary relation symbol R and constants $c_0, c_1, c_2, \ldots$. Here are two possible structures for $\mathcal{L}$ corresponding to two different interpretations of the language.

(i) Let the domain A consist of the natural numbers, let $R^{\mathcal{A}}$ be the usual relation $<$, and $c_0^{\mathcal{A}} = 0, c_1^{\mathcal{A}} = 1, \ldots$. The sentence $(\forall x)(\exists y)R(x, y)$ says that for every natural number there is a larger one, so it is true in this structure.

(ii) Let the domain of $\mathcal{A}$ consist of the rational numbers $\mathcal{Q} = \{q_0, q_1, \ldots\}$; let $R^{\mathcal{A}}$ again be $<$, and let $c_0^{\mathcal{A}} = q_0, c_1^{\mathcal{A}} = q_1, \ldots$. The sentence $(\forall x)(\forall y)(R(x, y) \rightarrow (\exists z)(R(x, z) \wedge R(z, y)))$ is true in this structure. (It says that the rationals are dense.) It is not, however, valid as it is false in the structure of (i) for the natural numbers.

Embedding of propositional logic

Theorem 4.8: Let φ be an open (i.e., quantifier-free) formula of predicate logic. We may view φ as a formula φ' of propositional logic by regarding every atomic subformula of φ as a propositional letter. With this correspondence, φ is a valid formula of predicate logic if and only if φ' is valid in propositional logic.

5. Interpretations of PROLOG Programs

Definition 5.1 (Clausal notation):

(i) Literals are atomic formulas or their negations. The atomic formulas are called positive literals and their negations, negative literals.

(ii) A clause is a finite set of literals.

(iii) A clause is a Horn clause if it contains at most one positive literal.

(iv) A program clause is a clause with exactly one positive literal. If a program clause contains some negative literals it is a rule; otherwise, it is a fact.

(v) A goal clause is a clause with no positive literals.

(vi) A formula is a not necessarily finite set of clauses.

PROLOG notation

Definition 5.1 (Clausal notation):

(i) In PROLOG, the fact $\{p(\vec{X})\}$ consisting of the single positive literal $p(\vec{X})$ appears in PROLOG programs as follows:

$$p(\vec{X}).$$

(ii) The rule $C = \{p(\vec{X}), \neg q_1(\vec{X}, \vec{Y}), ..., \neg q_n(\vec{X}, \vec{Y})\}$ appears in PROLOG programs as follows:

$$p(\vec{X}) :- q_1(\vec{X}, \vec{Y}), ..., q_n(\vec{X}, \vec{Y}).$$

(iii) For a rule C as in (ii), we call $p(\vec{X})$ the goal or head of C. We call the $q_1(\vec{X}, \vec{Y}), ..., q_n(\vec{X}, \vec{Y})$ the subgoals or body of C. When the head-body terminology is used, the symbol :- which connects the head and body of C is called the neck.

(iv) A (PROLOG) program is a formula (set of clauses) containing only program clauses (i.e., rules and facts).

Meaning of clauses and formulas

Each clause is interpreted as the universal closure of the disjunction of its elements. Thus the intended meaning of $C_1 = \{q(X, Y), r(Y)\}$ is $\forall X \forall Y [q(X, Y) \vee r(Y)]$. In this vein the intended meaning of the rule C given by $p(X) : -q_1(X, Y), ..., q_n(X, Y)$ (in clausal notation $C = \{p(X), \neg q_1(X, Y), ..., \neg q_n(X, Y)\}$) is $\forall X \forall Y [p(X) \vee \neg q_1(X, Y) \vee ... \vee ..., \neg q_n(X, Y)]$. This is equivalent to $\forall X [\exists Y (q_1(X, Y) \wedge ... \wedge q_n(X, Y)) \to p(X)]$.

A formula S is interpreted as the conjunction of its clauses. Thus if $S = \{C_1, C_2\}$ where C_1 is as above and $C_2 = \{q(X, Y), m(Y)\}$, then S has the same meaning as $\forall X \forall Y [q(X, Y) \vee r(Y)] \wedge \forall X \forall Y [q(X, Y) \vee m(Y)]$.

Meaning of goal clauses

The intended meaning of, e.g., "$? - p(X_1, X_2), q(X_2, X_3).$" is "are there objects a_1, a_2, a_3 such that $p(a_1, a_2)$ and $q(a_2, a_3)$".

As in the propositional case, PROLOG implements the search for such witnesses a_1, a_2 and a_3 by adding the goal clause $G = \{\neg p(X_1, X2), \neg q(X_2, X_3)\}$ to the current program P and then deciding if the result is an unsatisfiable formula.

The meaning of the clause G is $\forall X_1 \forall X_2 \forall X_3 [\neg p(X_1, X_2) \vee \neg q(X_2, X_3)]$. If adding it to the program P produces an unsatisfiable formula $P \cup \{G\}$, then $P \models \neg \forall X_1 \forall X_2 \forall X_3 [\neg p(X_1, X_2) \vee \neg q(X_2, X_3)]$ which is equivalent to $P \models \exists X_1 \exists X_2 \exists X_3 [p(X_1, X_2) \wedge q(X_2, X_3)]$

The implementation of PROLOG tries to establish this consequence relation by producing a resolution refutation of $P \cup \{G\}$.

Exercises

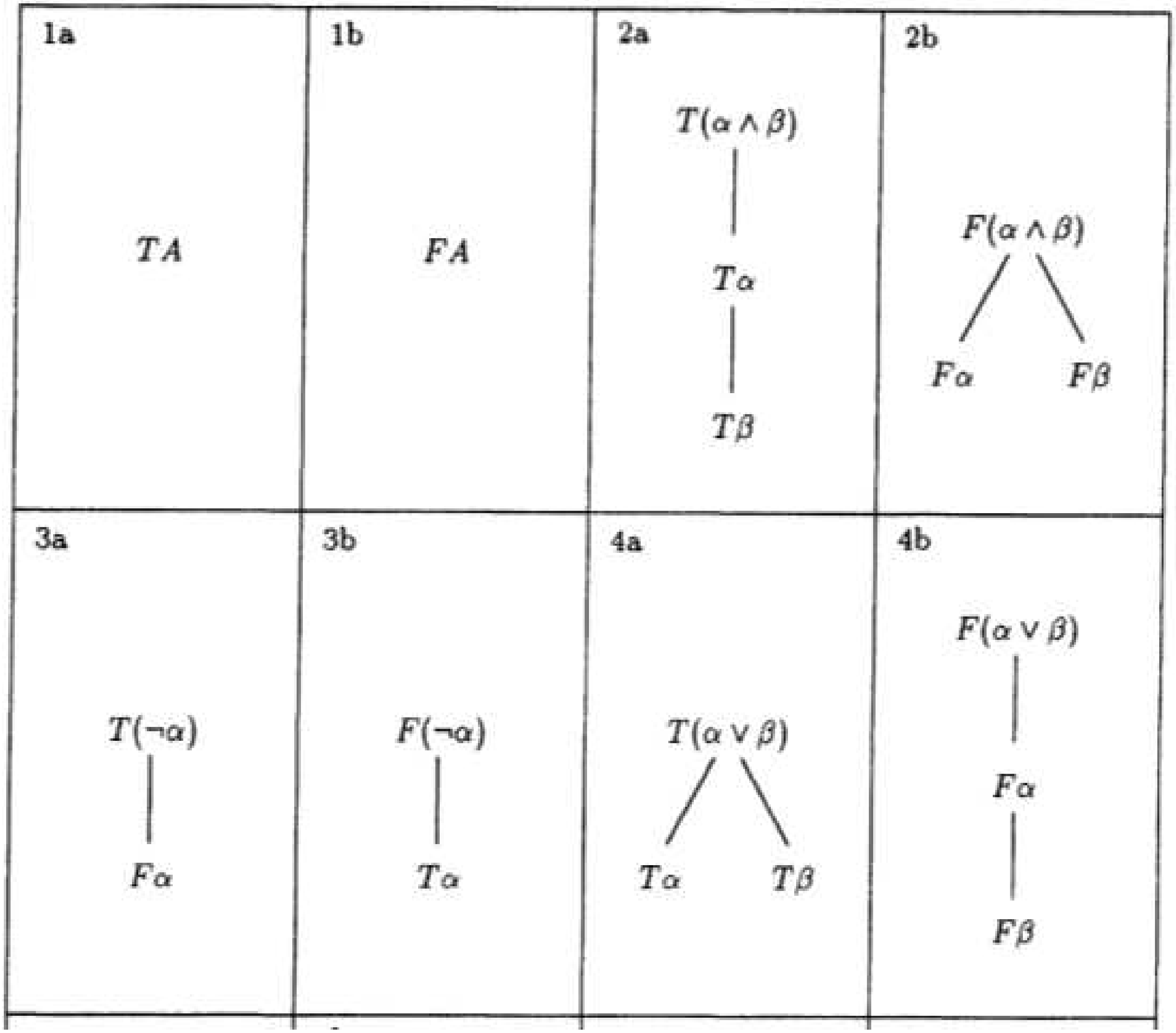
1. Exercise 2 in page 94
2. Exercises 6,7 in page 99
3. Exercise 10 in page 100
4. Exercise 7 in page 107
5. Exercise 8 in page 108

6. Proofs: Complete Systematic Tableaux

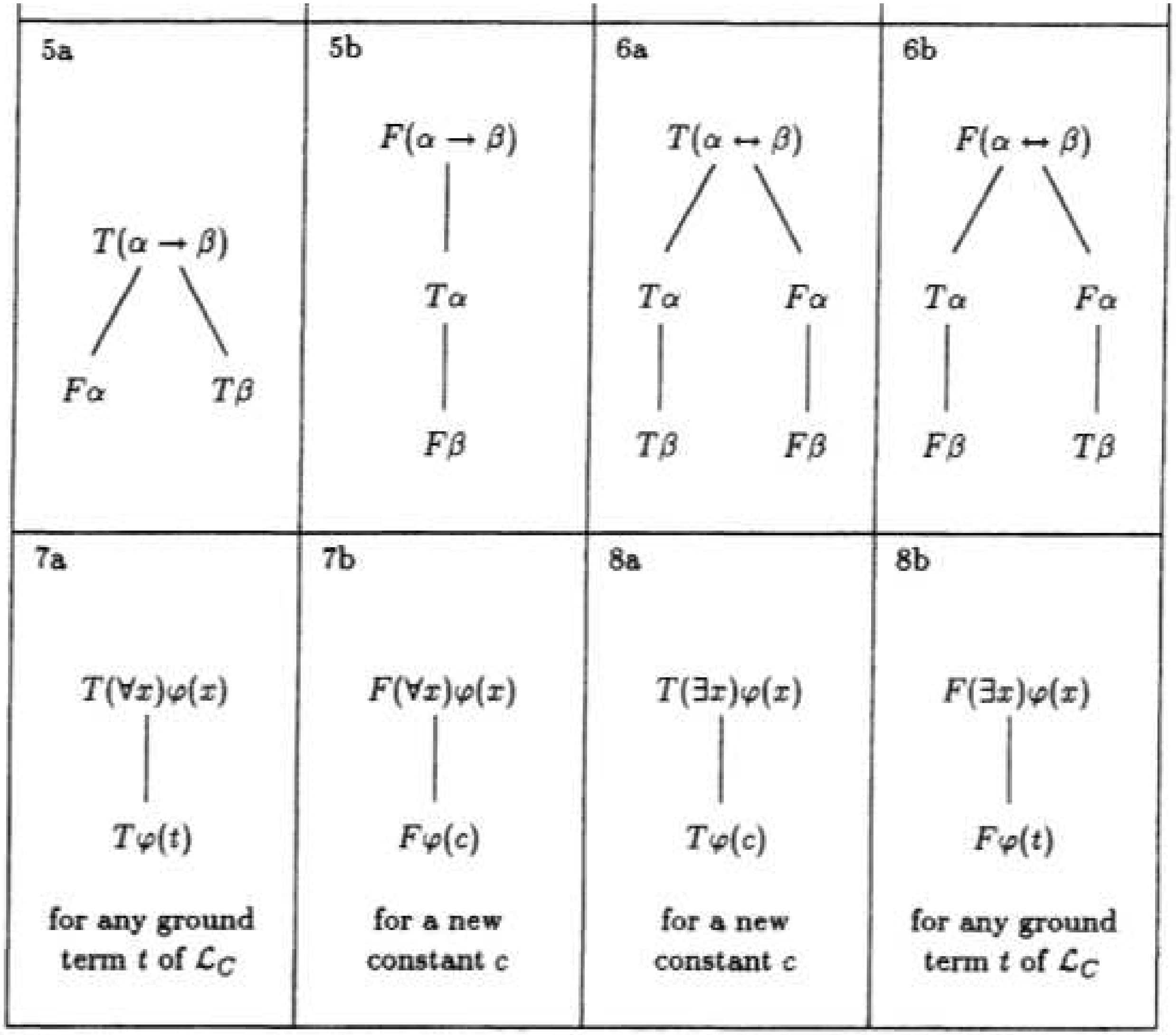
The proofs are labeled binary trees called tableaux. The labels on the trees are signed sentences (i.e., sentences preceded by T or F to indicate that, for the sake of the analysis, we are assuming them true or false, respectively).

For a language $\mathcal{L}$, we expand it to $\mathcal{L}_C$ by adding on a set of constant symbols $c_0, c_1, c_2, ...$ not used in $\mathcal{L}$.

Atomic tableaux



Atomic tableaux



5a $T(\alpha \to \beta)$ $F\alpha \quad T\beta$	5b $F(\alpha \to \beta)$ $T\alpha$ $F\beta$	6a $T(\alpha \leftrightarrow \beta)$ $T\alpha \quad F\alpha$ $T\beta \quad F\beta$	6b $F(\alpha \leftrightarrow \beta)$ $T\alpha \quad F\alpha$ $F\beta \quad T\beta$
7a $T(\forall x)\varphi(x)$ $T\varphi(t)$ for any ground term t of $\mathcal{L}_C$	7b $F(\forall x)\varphi(x)$ $F\varphi(c)$ for a new constant c	8a $T(\exists x)\varphi(x)$ $T\varphi(c)$ for a new constant c	8b $F(\exists x)\varphi(x)$ $F\varphi(t)$ for any ground term t of $\mathcal{L}_C$

Tableaux

Definition 6.1: We define tableaux as binary trees labeled with signed sentences (of $\mathcal{L}_C$) called entries by induction:

(i) All atomic tableaux are tableaux. The requirement that c be new in Cases 7b and 8a here simply means that c is one of the constants c_i added on to $\mathcal{L}$ to get $\mathcal{L}_C$ (which therefore does not appear in φ).

(ii) If τ is a finite tableau, P a path on τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining an atomic tableau with root entry E to τ at the end of the path P, then τ' is also a tableau. Here the requirement that c be new in Cases 7b and 8a means that it is one of the c_i that do not appear in any entries on P. (In actual practice it is simpler in terms of bookkeeping to choose one not appearing at any node of τ.)

(iii) If τ_0 is a finite tableau and $\tau_0, \tau_1, ..., \tau_n, ...$ is a sequence of tableaux such that, for every $n \geq 0$, τ_{n+1} is constructed from τ_n by an application of (ii), then $\tau = \cup \tau_n$ is also a tableau.

Tableaux

Definition 6.1 (Continued): Tableaux from S. The definition for tableaux from S is the same as for ordinary tableaux except that we include an additional formation rule:

(ii') If τ is a finite tableau from S, φ a sentence from S, P a path on τ and τ' is obtained from τ by adjoining $T\varphi$ to the end of the path P, then τ' is also a tableau from S.

Note: It is clear from the definition that every tableau τ (from S) is the union of a finite or infinite sequence $\tau_0, \tau_1, ..., \tau_n, ...$ of tableaux (from S) in which τ_0 is an atomic tableau and each τ_{n+1} is gotten from τ_n by an application of (ii) (or (ii')).

Tableau proofs

Definition 6.2 Let τ be a tableau and P a path in τ.

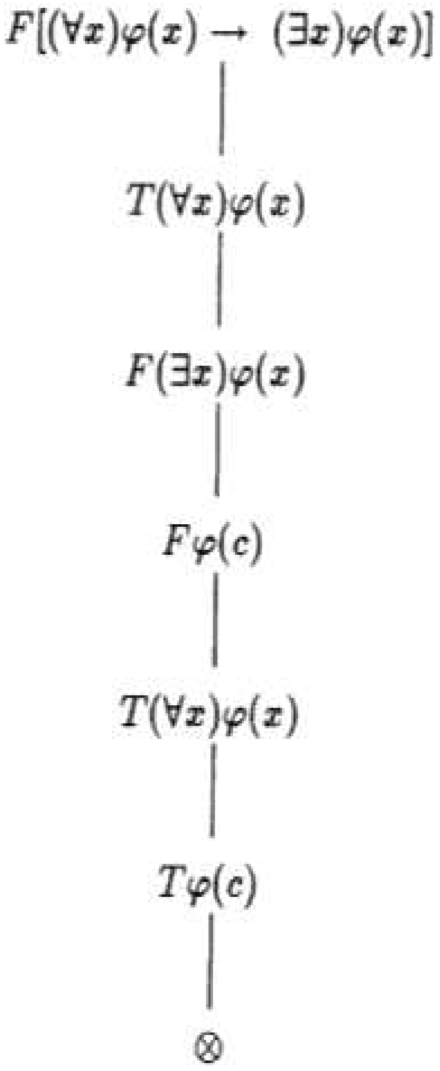
(i) P is contradictory if, for some sentence α, $T\alpha$ and $F\alpha$ both appear as labels of nodes of P.

(ii) τ is contradictory if every path on τ is contradictory.

(iii) τ is a proof of α (from S) if τ is a finite contradictory tableau (from S) with its root node labeled $F\alpha$. If there is proof τ of α (from S), we say α is provable (from S) and write $\vdash \alpha(S \vdash \alpha)$.

(iv) S is inconsistent if there is a proof of $\alpha \wedge \neg\alpha$ from S for some sentence α.

Example: Tableau proof



$F[(\forall x)\varphi(x) \to (\exists x)\varphi(x)]$

$T(\forall x)\varphi(x)$

$F(\exists x)\varphi(x)$

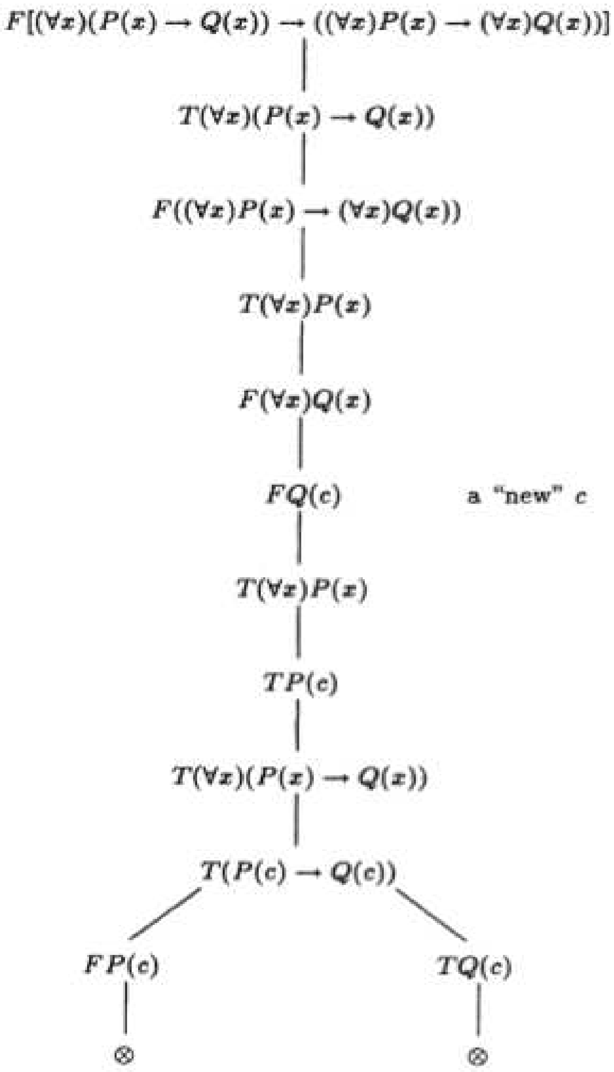
$F\varphi(c)$

$T(\forall x)\varphi(x)$

$T\varphi(c)$

$\otimes$

Example: Tableau proof



$F[(\forall x)(P(x) \to Q(x)) \to ((\forall x)P(x) \to (\forall x)Q(x))]$

$T(\forall x)(P(x) \to Q(x))$

$F((\forall x)P(x) \to (\forall x)Q(x))$

$T(\forall x)P(x)$

$F(\forall x)Q(x)$

$FQ(c)$ a "new" c

$T(\forall x)P(x)$

$TP(c)$

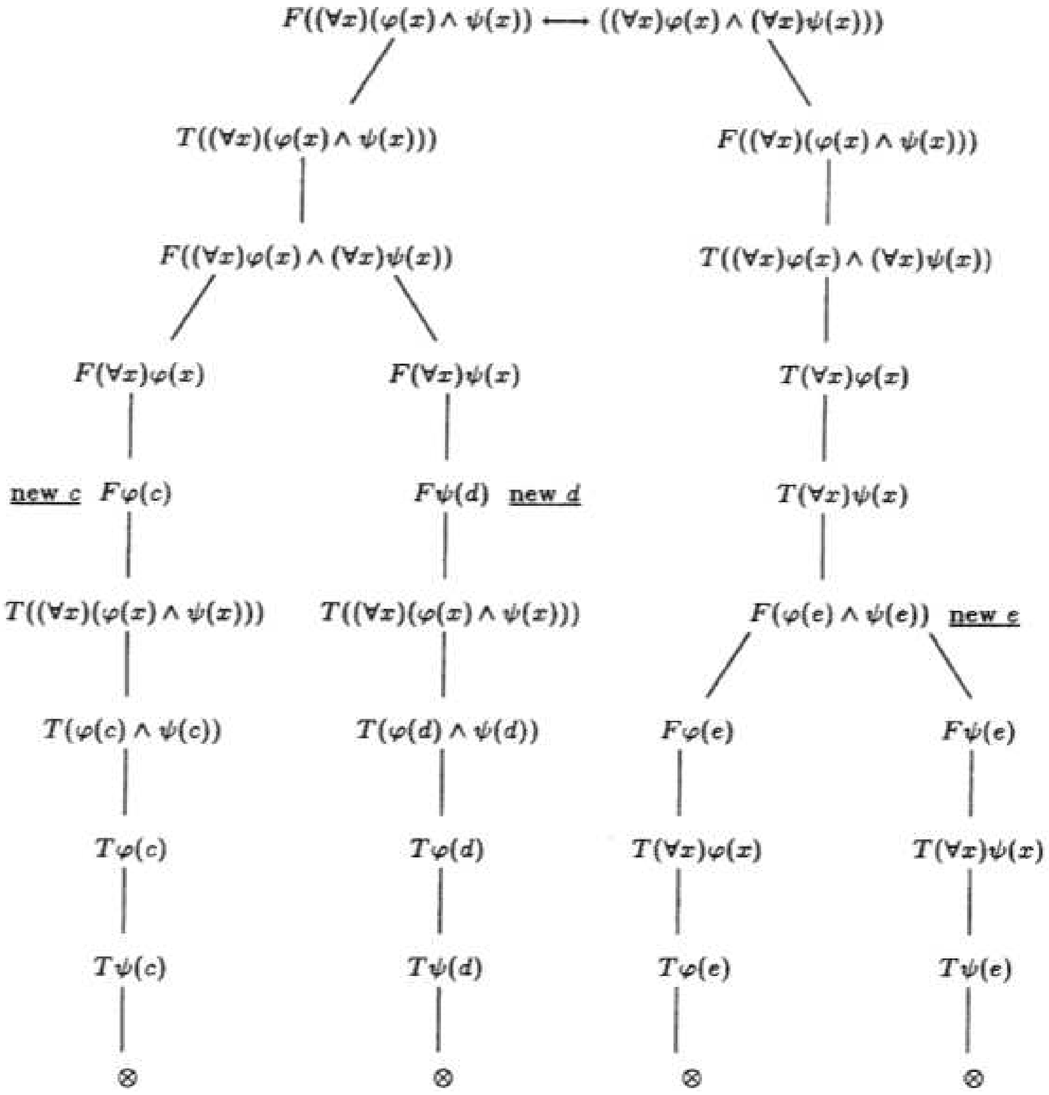
$T(\forall x)(P(x) \to Q(x))$

$T(P(c) \to Q(c))$

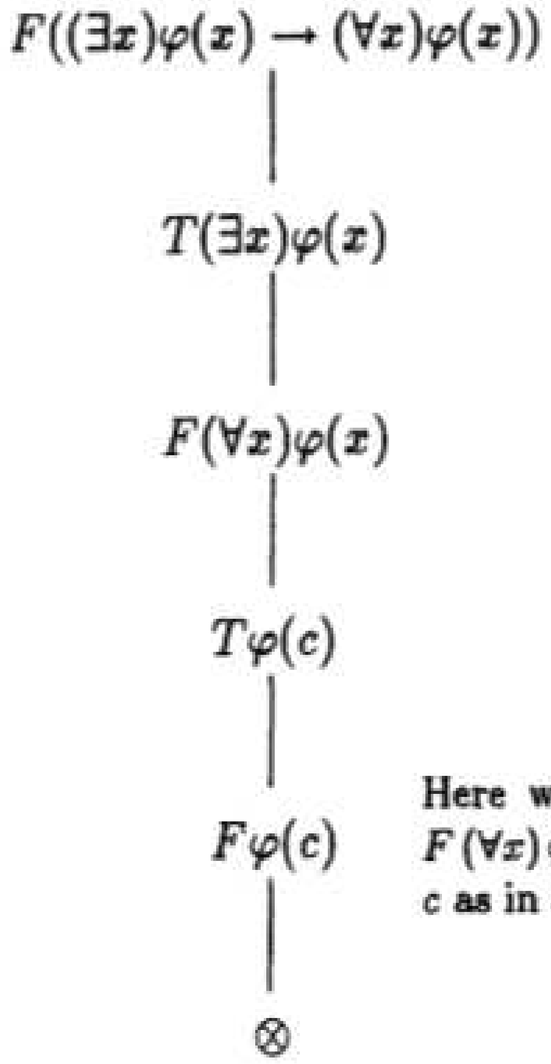
$FP(c)$ $TQ(c)$

$\otimes$ $\otimes$

Example: Tableau proof


$F((\forall x)(\varphi(x) \wedge \psi(x)) \longleftrightarrow ((\forall x)\varphi(x) \wedge (\forall x)\psi(x)))$
$T((\forall x)(\varphi(x) \wedge \psi(x)))$
$F((\forall x)(\varphi(x) \wedge \psi(x)))$
$F((\forall x)\varphi(x) \wedge (\forall x)\psi(x))$
$T((\forall x)\varphi(x) \wedge (\forall x)\psi(x))$
$F(\forall x)\varphi(x)$
$F(\forall x)\psi(x)$
$T(\forall x)\varphi(x)$
new c $F\varphi(c)$
$F\psi(d)$ new d
$T(\forall x)\psi(x)$
$T((\forall x)(\varphi(x) \wedge \psi(x)))$
$T((\forall x)(\varphi(x) \wedge \psi(x)))$
$F(\varphi(e) \wedge \psi(e))$ new e
$T(\varphi(c) \wedge \psi(c))$
$T(\varphi(d) \wedge \psi(d))$
$F\varphi(e)$
$F\psi(e)$
$T\varphi(c)$
$T\varphi(d)$
$T(\forall x)\varphi(x)$
$T(\forall x)\psi(x)$
$T\psi(c)$
$T\psi(d)$
$T\varphi(e)$
$T\psi(e)$
⊗
⊗
⊗
⊗

Example: A bad tableau proof



$F((\exists x)\varphi(x) \rightarrow (\forall x)\varphi(x))$

$T(\exists x)\varphi(x)$

$F(\forall x)\varphi(x)$

$T\varphi(c)$

$F\varphi(c)$

$\otimes$

Here we have developed the entry $F(\forall x)\varphi(x)$, illegally using the same c as in a previous entry.

Reduced entry

Definition 6.7: Let $\tau = \cup\tau_n$ be a tableau (from S), P a path in τ, E an entry on P and w the i^{th} occurrence of E on P (i.e., the i^{th} node on P labeled with E).

(i) w is reduced on P if

(1) E is neither of the form $T(\forall x)\varphi(x)$ nor $F(\exists x)\varphi(x)$ and, for some j, τ_{j+1} is gotten from τ_j by an application of Rule (ii) of Definition 6.1 to E and a path on τ_j which is an initial segment of P. (In this case we say that E occurs on P as the root entry of an atomic tableau.) or

(2) E is of the form $T(\forall x)\varphi(x)$ or $F(\exists x)\varphi(x)$, $T\varphi(t_i)$ or $F\varphi(t_i)$, respectively, is an entry on P and there is an $(i+1)^{st}$ occurrence of E on P.

(ii) τ is finished if every occurrence of every entry on τ is reduced on every noncontradictory path containing it (and $T\varphi$ appears on every noncontradictory path of τ for every φ in S). It is unfinished otherwise.

Level-lexicographic ordering

Definition 6.8: Suppose T is a tree with a left-right ordering on the nodes at each of its levels. We define the level-lexicographic ordering $\leq_{LL}$ on the nodes ν, μ of T as follows:

$\nu \leq_{LL} \mu \;\Leftrightarrow\;$ the level of ν in T is less than that of μ or ν and μ are on the same level of T and ν is to the left of μ.

Complete systematic tableau

Definition 6.9: We construct the CST, the complete systematic tableau, with any given signed sentence as the label of its root, by induction.

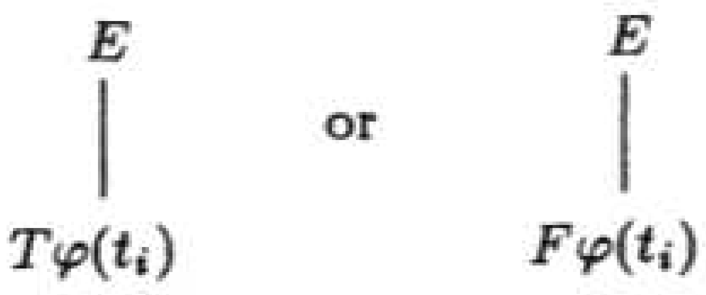
(i) We begin with τ_0 an atomic tableau with root the given signed sentence. This atomic tableau is uniquely specified by requiring that in Cases 7a and 8b we use the term t_1 and that in Cases 7b and 8a we use c_i for the least allowable i.

Let τ_n is a (finite, labeled) binary tree. If every occurrence of every entry on T is reduced, we terminate the construction. Otherwise, let w be the level-lexicographically least node of τ_n that contains an occurrence of an entry E which is unreduced on some noncontradictory path P of τ_m. We now proceed according to one of the following cases:

(ii) If E is not of the form $T(\forall x)\varphi(x)$ or $F(\exists x)\varphi(x)$, we adjoin the atomic tableau with apex E to the end of every noncontradictory path in τ

that contains w. For E of the form $T(\exists x)\varphi(x)$ or $F(\forall x)\varphi(x)$, we use the least constant c_j not yet appearing in the tableau.

(iii) If E is of the form $T(\forall x)\varphi(x)$ or $F(\exists x)\varphi(x)$ and w is the i^{th} occurrence of E on P we adjoin



respectively, to the end of every noncontradictory path in T containing w.

CST from premises

The CST from a set of premises S with a given root is defined like the ordinary CST above with one change to introduce the elements of S. At even stages ($n = 2k$) we proceed as in (i), (ii) and (iii) above. At odd stages ($n = 2k + 1$) we adjoin $T\alpha_k$ for α_k the k^{th} element of S to every noncontradictory path in τ_n to get τ_{n+1}. We do not terminate the construction of the CST from S unless all elements of S have been put on every noncontradictory path in this way and every occurrence of every entry is reduced on every path containing it.

In general, a CST will be an infinite tableau (even if S is finite).

Every CST is finished

Proposition 6.10: Every CST is finished.

Proof: Consider any unreduced occurrence w of an entry E in $\tau_k \subseteq \tau$ that is on a noncontradictory path P of the given CST T. (If there is none, τ is finished by definition.) Suppose there are n nodes of τ that are level-lexicographically less than w. It is clear from the definition of the CST that we must reduce w on P by the time we form τ_{k+n+1} Thus, every occurrence of each entry on a noncontradictory path in τ is reduced as required.

If we consider the CST from S, the same considerations apply to show that every entry is reduced on every path. (It just takes twice as many steps to get there.) The procedure of adding on the k^{th} member of S at stage $2k+1$ guarantees that every element of S is put on every path of the CST from S. It is therefore a finished tableau from S. □

Exercises

1. Exercises 10,11,12,13 in page 118

7. Soundness and Completeness of Tableau Proofs

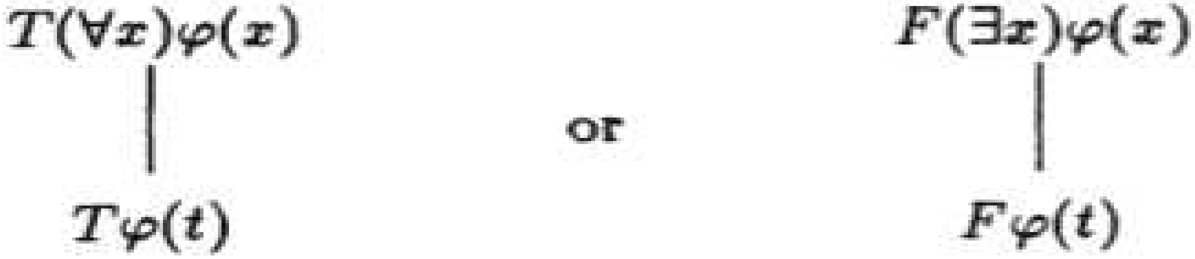
Lemma 7.1: If $\tau = \cup\tau_n$ is a tableau from a set of sentences S with root $F\alpha$, then any $\mathcal{L}$-structure $\mathcal{A}$ that is a model of $S \cup \{\neg\alpha\}$ can be extended to one agreeing with every entry on some path P through τ. (Recall that $\mathcal{A}$ agrees with $T\alpha\,(F\alpha)$ if α is true (false) in $\mathcal{A}$.)

Proof: The only expansion of $\mathcal{A}$ that is necessary to make it a structure for all the sentences appearing in τ is to define $c_i^{\mathcal{A}}$ for the constants c_i in $\mathcal{L}_C - \mathcal{L}$ appearing on P.

We define P and $c_i^{\mathcal{A}}$ by an induction on the sequence τ_n giving the construction of τ. At each step n we have a path P_n through τ_n and an extension $\mathcal{A}_n$ of $\mathcal{A}$ (with the same domain) which interprets all the c_i on P_n and agrees with P_n. This clearly suffices to prove the lemma. When τ_{n+1} is gotten from τ_n by extending some path other than P_n we need make no changes in P_n or $\mathcal{A}_n$. Suppose then that τ_{n+1} is gotten by adding on to the end of P_n either an atomic tableau with root E an entry on P_n or an element α_k of S. In the latter case we extend P_n in the only way possible by attaching α_k to its end. No extension of $\mathcal{A}_n$ is necessary and it agrees with α_k (and hence P_{n+1}) by hypothesis. We consider then the case of extending τ_n by adding on an atomic tableau τ' with root E. By induction we may assume that $\mathcal{A}_n$ agrees with E. We wish to extend $\mathcal{A}_n$ to $\mathcal{A}_{n+1}$ and find a path P_{n+1} extending P_n through τ_{n+1} agreeing with $\mathcal{A}_{n+1}$. (The base case of our induction is the atomic tableau τ_0 whose root $F\alpha$ agrees with $\mathcal{A}$ by hypothesis. The analysis of the base case is then exactly as in the inductive step: We wish to extend $\mathcal{A}$ to $\mathcal{A}_0$ and find a path P_0 through τ_0 agreeing with $\mathcal{A}_0$.) We consider each type of atomic tableau τ'.

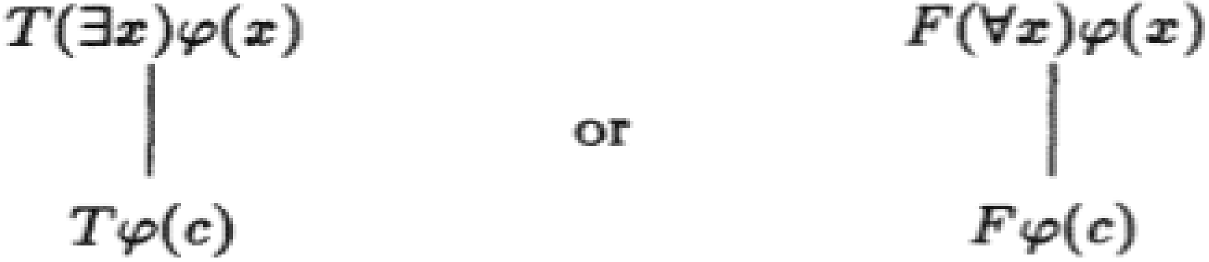
(i) The situation for the propositional connectives is the same as in the proof of soundness for propositional logic (Lemma 1.5.4). In particular, no extension of $\mathcal{A}_n$ is necessary. E.g. if we added on the atomic tableau with root $T(\alpha \vee \beta)$ then we know by induction that $\mathcal{A}_n \models \alpha \vee \beta$ and so $\mathcal{A}_n \models \alpha$ or $\mathcal{A}_n \models \beta$. We choose to extend P_n accordingly.

(ii) If we added on



we again have no problem. $\mathcal{A}_n \models \forall x\varphi(x)$ (or $\mathcal{A}_n \models \neg\exists\varphi(x)$) and so $\mathcal{A}_n \models \varphi(t)$) ($\mathcal{A}_n \models \neg\varphi(t)$). (Note that if $t^{\mathcal{A}}$ is not yet defined by our inductive procedure we can now define it arbitrarily and still maintain our inductive hypothesis as we know that $\mathcal{A}_n \models \forall x\varphi(x)$ (or $\mathcal{A}_n \models \neg\exists x\varphi(x)$).)

(iii) Finally, if we added on



for some new constant symbol c (i.e., one not appearing either in S or in an entry on P_n), we must define $c^{\mathcal{A}}$. By induction, we know that $\mathcal{A}_n \models \exists x\varphi(x)$

$(\mathcal{A}_n \models \neg\forall x\varphi(x))$ and so we may choose an element $a \in A$ $(= \mathcal{A}_n$ by construction) such that, if we extend $\mathcal{A}_n$ to $\mathcal{A}_{n+1}$ by letting $c^{\mathcal{A}} = a$, we have $\mathcal{A}_{n+1} \models \varphi(c)$ $(\mathcal{A}_{n+1} \models \neg\varphi(c))$ as required.

□

Soundness

Theorem 7.2: If there is a tableau proof τ of α from S, then $S \models \alpha$.

Proof: If not, then there is a structure $\mathcal{A} \models \neg\alpha$ in which every α_k in S is true. Lemma 7.1 then tells us that there is a path P through τ and an expansion $\mathcal{A}'$ of $\mathcal{A}$ that agrees with every node on P. As P is contradictory by assumption, we have our desired contradiction. □

Structure from noncontradictory path

Theorem 7.3: Suppose P is a noncontradictory path through a complete systematic tableau τ from S with root $F\alpha$. There is then a structure $\mathcal{A}$ in which α is false and every sentence in S is true.

Proof: Let the domain of this structure be the set A of ground terms t_i on the master list of ground terms of our expanded language $\mathcal{L}_C$. We define the functions $f^{\mathcal{A}}$ associated with the n-ary function symbols f of our language in the natural way corresponding to the syntax of $\mathcal{L}_C$: $f^{\mathcal{A}}(t_{i_1,t_{i_2},...,t_{i_n}}) = f(t_{i_1}, ..., t_{i_2})$

Remember that the elements of our structure are the ground terms and so the t_i appearing on the left-hand side of this equation are being viewed as elements of our structure to which we apply the function $f^{\mathcal{A}}$. On the right-hand side we have another term, and so an element of our structure, which we declare to be the value of this function. If R is an n-ary predicate letter, we define $R^{\mathcal{A}}$ as dictated by the path P: $R^{\mathcal{A}}(t_{i_1}, t_{i_2}, ..., t_{i_n}) \Leftrightarrow TR(t_{i_1}, ..., t_{i_2})$ is an entry on P.

We now prove the theorem by establishing a slightly stronger assertion by induction. □

Structure from noncontradictory path

Lemma 7.4: Let the notation be in Theorem 7.3.

(i) If $F\beta$ occurs on P, then β is false in $\mathcal{A}$.

(ii) If $T\beta$ occurs on P, then β is true in $\mathcal{A}$.

Proof: First recall that, by Proposition 6.10, every occurrence of every entry on P is reduced on P. We now proceed by induction on the depth of β (more precisely, on the depth of the associated auxiliary formation tree as given in Definition 3.8).

(i) If β is an atomic sentence, then β is of the form $R(t_{i_1}, ..., t_{i_n})$. If $T\beta$ occurs on P, then $R^{\mathcal{A}}$ has been declared true of $t_{i_1}, ..., t_{i_n}$. If $F\beta$ occurs on P, then, as P is noncontradictory, $T\beta$ does not occur on P and $R^{\mathcal{A}}$ has been declared false of $t_{i_1}, ..., t_{i_n}$.

(ii) Suppose β is built using a connective, e.g., β is $(\beta_1 \vee \beta_2)$. As τ is finished, we know that if $T\beta$ occurs on P, then either $T\beta_1$ or $T\beta_2$ occurs on P. By the induction hypothesis, if $T\beta_1$ occurs on P, then β_1 is true in $\mathcal{A}$ (and similarly for β_2). Thus, one of β_1, β_2 is true so $(\beta_1 \vee \beta_2)$ is true in $\mathcal{A}$ (by the inductive definition of truth). On the other hand, if $F(\beta_1 \vee \beta_2)$ appears on P, then we know that both $F\beta_1$ and $F\beta_2$ appear on P. Our inductive hypothesis then tells us that both β_1 and β_2 are false in $\mathcal{A}$. We then have that $(\beta_1 \vee \beta_2)$ is false in $\mathcal{A}$ as required. The cases for the other connectives are similar.

(iii) Suppose β is of the form $(\forall v)\varphi(v)$. If w is the i^{th} occurrence of $T((\forall v)\varphi(v))$ on P, then $T\varphi(t_i)$ occurs on P and there is an $i+1^{st}$ occurrence of $T((\forall v)\varphi(v))$ on P. Thus, if $T((\forall v)\varphi(v))$ appears on P , then $\varphi(t)$ appears on P for every ground term t. As the depth of $\varphi(t)$ is less than that of $(\forall v)\varphi(v)$, the inductive hypothesis tells us that $\varphi(t)$ is true in $\mathcal{A}$ for every ground term t. As these terms constitute the universe of our structure $\mathcal{A}$, $(\forall v)\varphi(v)$ is true in $\mathcal{A}$ as required.

If $F(\forall v)\varphi(v)$ occurs on P, then, again as τ is finished, $F\varphi(t)$ occurs on P for some t. By induction hypothesis $\varphi(t)$ is false in $\mathcal{A}$. So $(\forall v)\varphi(v)$ is false in $\mathcal{A}$.

(iv) The case for $\exists v\varphi(v)$ is similar.

□

This also completes the proof of Theorem 7.3. □

Contradictory CST is finite

Proposition 7.5: If every path of a complete systematic tableau is contradictory, then it is a finite tableau.

Proof: By construction, we never extend a path on a CST once it is contradictory. Thus, every contradictory path on a CST is finite. The theorem then follows from König's lemma (Theorem I.1.4). □

Corollary 7.6: For every sentence α and set of sentences S of $\mathcal{L}$, either

(i) the CST from S with root $F\alpha$ is a tableau proof of α from S or

(ii) there is a noncontradictory branch through the complete systematic tableau that yields a structure in that α is false and every element of S is true.

Skolem-Löwenheim theorem and Completeness

Theorem 7.7: (Skolem-Löwenheim): If a countable set of sentences S is satisfiable (that is, it has some model), then it has a countable model.

Proof: Consider the CST from S that starts with a contradiction $\alpha \wedge \neg\alpha$ at its root. By the soundness theorem (Theorem 7.2) it cannot be a tableau proof of $\alpha \wedge \neg\alpha$ from S. Thus, it must have a noncontradictory path P. As there are only countably many ground terms in $\mathcal{L}_C$, the structure defined in the proof of Theorem 7.4 is the desired countable model of S. □

Theorem 7.8: (Completeness and Soundness):

(i) α is tableau provable from S $\Leftrightarrow$ α is a logical consequence of S.

(ii) If we take α to be any contradiction such as $\beta \wedge \neg\beta$ in (i), we see that S is inconsistent if and only if S is unsatisfiable.

Compactness

Theorem 7.9: Let $S = \{\alpha_1, \alpha_2, ...\}$ be a set of sentences of predicate logic. S is satisfiable if and only if every finite subset of S is satisfiable.

Proof: The only if direction is immediate. For the if direction consider the CST from S with root entry $F(\alpha \wedge \neg\alpha)$. If the CST is contradictory, it is finite by Proposition 7.5. If it is infinite, it has a noncontradictory path and so by Corollary 7.6 there is a structure in which every α_i is true. If it is contradictory and finite, then $\alpha \wedge \neg\alpha$ is a logical consequence of the finite subset of S whose elements are those appearing on this tableau. This finite subset can have no model as $\alpha \wedge \neg\alpha$ has no model. □

Exercises

1. Exercises 1, 2, 3 in page 125

2. Exercises 10, 11 in page 126

8. An Axiomatic Approach

Axioms: Let α, β and γ be any formulas of $\mathcal{L}$. The axioms of our system are all formulas of $\mathcal{L}$ of the following forms:

(i) $(\alpha \to (\beta \to \alpha))$

(ii) $((\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma)))$

(iii) $((\neg\alpha) \to (\alpha \to \beta))$

(iv) $(\forall x)\alpha(x) \to \alpha(t)$ for any term t that is substitutable for x in α

(v) $(\forall x)(\alpha \to \beta) \to (\alpha \to (\forall x)\beta)$ if α contains no free occurrences of x.

The rules of inference:

(i) Modus Ponens: From α and $\alpha \to \beta$, we can infer β for any formulas α and β.

(ii) Generalization: From $\forall x\alpha$ infer α.

Proofs from premises

Definition 8.3: Let Σ be a set of formulas of $\mathcal{L}$.

(i) A proof from Σ is a finite sequence $\alpha_1, \alpha_2, , \alpha_n$ of formulas of $\mathcal{L}$ such that, for each $i \leq n$, one of the following is true:

(1) α_i is a member of Σ

(2) α_i is an axiom

(3) α_i can be inferred from some of the previous α_j by an application of a rule of inference.

(ii) α is provable (a theorem) from Σ if there is a proof $\alpha_1, ..., \alpha_n$ from Σ with $\alpha_n = \alpha$.

(iii) A proof of α is simply a proof from $\emptyset$. α is provable if it is provable from $\emptyset$.

The standard soundness, completeness and compactness theorems can be proven for the system presented here.

9. Prenex Normal Form and Skolemization

Basic ideas: to eliminate quantifiers by introducing new function symbols and terms. E.g. the formula

$\varphi = \forall x_1 ... \forall x_n \exists y_1 ... \exists y_m R(x_1, ..., x_n, y_1, ..., y_m)$

will be replaced by

$\psi = \forall x_1 ... \forall x_n R(x_1, ..., x_n,\ f_1(x_1, ..., x_n), f_2(x_1, ..., x_n), ..., f_n(x_1, ..., x_n))$

Here f_i is a new function that chooses, for any given $x_1, ..., x_n$, a y_i that makes the formula true if one exists.

The ultimate goal is to get a universal formula ψ (i.e., one with only universal quantifiers which all occur as the initial symbols of ψ) that is equisatisfiable with the original φ. (Two formulas are equisatisfiable if both are satisfiable or if neither is.)

Provable equivalences

Lemma 9.1: For any string of quantifiers $\overrightarrow{Qx} = Q_1x_1Q_2x_2...Q_nx_n$ (each Q_i is ∀ or ∃) and any formulas φ, ψ we have the following provable equivalences:

(1a) $\vdash \overrightarrow{Qx}\neg\forall y\varphi \leftrightarrow \overrightarrow{Qx}\exists y\neg\varphi$

(1a) $\vdash \overrightarrow{Qx}\neg\exists y\varphi \leftrightarrow \overrightarrow{Qx}\forall y\neg\varphi$

(2a) $\vdash \overrightarrow{Qx}(\forall y\varphi \vee \psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi(y/z) \vee \psi)$

(2a') $\vdash \overrightarrow{Qx}(\varphi \vee \forall y\psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi \vee \psi(y/z))$

(2b) $\vdash \overrightarrow{Qx}(\exists y\varphi \vee \psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi(y/z) \vee \psi)$

(2b') $\vdash \overrightarrow{Qx}(\varphi \vee \exists y\psi) \leftrightarrow \overrightarrow{Qx}\forall\exists z(\varphi \vee \psi(y/z))$

where z is a variable not occurring in φ or ψ or among the x_i.

The practice of renaming variables as in $(2a)$ and $(2b)$ to avoid possible conflicts is often called standardizing the variables apart.

Prenex normal form

Theorem 9.2: For every formula φ there is an equivalent formula φ' with the same free variables in which all quantifiers appear at the beginning. Such an equivalent of φ is called a prenex normal form (PNF) of φ.

Proof: By induction on the depth of φ. Remember that, by Corollary I.2.11, we may assume that the only propositional connectives occurring in φ are $\neg$ and $\vee$. If φ is atomic, there is nothing to prove. If φ is $\forall y\psi$ or $\exists y\psi$ and ψ' is a PNF of ψ, then $\forall y\psi'$ or $\exists y\psi'$ is one for ψ. If $\varphi = \neg\psi$ and ψ' is a PNF of ψ, then repeated applications of the clauses (1a) and (1b) of the lemma will produce the desired PNF for φ. If $\varphi = \psi \vee \theta$, then repeated applications of the clauses (2a), (2a'), (2b) and (2b') will give the result for φ. □

Prenex normal form

One can easily introduce prenexing rules that deal directly with the other connectives.

(3a) $\vdash \overrightarrow{Qx}(\forall y\varphi \wedge \psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi(y/z) \wedge \psi)$

(3a') $\vdash \overrightarrow{Qx}(\varphi \wedge \forall y\psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi \wedge \psi(y/z))$

(3b) $\vdash \overrightarrow{Qx}(\exists y\varphi \wedge \psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi(y/z) \wedge \psi)$

(3b') $\vdash \overrightarrow{Qx}(\varphi \wedge \exists y\psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi \wedge \psi(y/z))$

(4a) $\vdash \overrightarrow{Qx}(\forall y\varphi \rightarrow \psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi(y/z) \rightarrow \psi)$

(4a') $\vdash \overrightarrow{Qx}(\varphi \rightarrow \forall y\psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi \rightarrow \psi(y/z))$

(4b) $\vdash \overrightarrow{Qx}(\exists y\varphi \rightarrow \psi) \leftrightarrow \overrightarrow{Qx}\forall z(\varphi(y/z) \rightarrow \psi)$

(4b') $\vdash \overrightarrow{Qx}(\varphi \rightarrow \exists y\psi) \leftrightarrow \overrightarrow{Qx}\exists z(\varphi \rightarrow \psi(y/z))$

where z is a variable not occurring on the left-hand side of the equivalences.

Example: PNF's

$$\forall x \forall y [(\exists z)(P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)]$$
$$\forall x \forall y \forall w [P(x,w) \wedge P(y,w) \rightarrow \exists u Q(x,y,u)]$$
$$\forall x \forall y \forall w \exists z [P(x,w) \wedge P(y,w) \rightarrow Q(x,y,z)]$$

Example: PNF's

$\forall x \exists y P(x, y) \vee \neg \exists x \forall y Q(x, y):$

$\forall u[\exists y P(u, y) \vee \neg \exists x \forall y Q(x, y)]$

$\forall u \exists v[P(u, v) \vee \neg \exists x \forall y Q(x, y)]$

$\forall u \exists v[P(u, v) \vee \forall x \neg \forall y Q(x, y)]$

$\forall u \exists v[P(u, v) \vee \forall x \exists y \neg Q(x, y)]$

$\forall u \exists v \forall w[P(u, v) \vee \exists y \neg Q(w, y)]$

$\forall u \exists v \forall w \exists z[P(u, v) \vee \neg Q(w, z)].$

Alternatively

$\forall x \exists y P(x, y) \vee \neg \exists x \forall y Q(x, y):$

$\forall u[\exists y P(u, y) \vee \neg \exists x \forall y Q(x, y)]$

$\forall u[\exists y P(u, y) \vee \forall x \neg \forall y Q(x, y)]$

$\forall u \forall w[\exists y P(u, y) \vee \neg \forall y Q(w, y)]$

$\forall u \forall w \exists v[P(u, v) \vee \neg \forall y Q(w, y)]$

$\forall u \forall w \exists v[P(u, v) \vee \exists y \neg Q(w, y)]$

$\forall u \forall w \exists v \exists z[P(u, v) \vee \neg Q(w, z)]$

Skolemization

Theorem 9.4: For every sentence φ in a given language $\mathcal{L}$ there is a universal formula φ' in an expanded language $\mathcal{L}'$ gotten by the addition of new function symbols such that φ and φ' are equisatisfiable.

(Note that the formulas are not necessarily equivalent. The procedure will always produce a φ' such that $\varphi' \to \varphi$ is valid but $\varphi \to \varphi'$ need not always hold.)

Proof: By Theorem 9.2 we may assume that φ is in prenex normal form. Let $y_1, ..., y_n$ be the existentially quantified variables of φ in the order in which they appear in φ from left to right and, for each $i \leq n$, let $x_1, ..., x_{n_i}$ be all the universally quantified variables preceding y_i. We expand $\mathcal{L}$ to $\mathcal{L}'$ by adding new n_i-ary function symbols f_i for each $i \leq n$. We now form φ' by first deleting each $\exists y_i$ and then replacing each remaining occurrence of y_i by $f_i(x_1, ..., x_{n_i})$. We claim that φ' is the desired sentence equisatisfiable with φ. To verify this claim it suffices to apply the next lemma n times. □

Skolemization

Lemma 9.5: For any sentence $\varphi = \forall x_1 ... \forall x_n \exists y \psi$ of a language $\mathcal{L}$, φ and $\varphi' = \forall x_1 ... \forall x_n \psi(y/f(x_1, ..., x_n))$ are equisatisfiable when f is a function symbol not in $\mathcal{L}$.

Proof: Let $\mathcal{L}'$ be the language obtained from $\mathcal{L}$, by adding the function symbol f. It is clear that if $\mathcal{A}'$ is a structure for $\mathcal{L}'$, $\mathcal{A}$ is the structure obtained from $\mathcal{A}'$ by omitting the function interpreting f and $\mathcal{A}' \models \varphi'$, then $\mathcal{A} \models \varphi$. On the other hand, if $\mathcal{A}$ is a structure for $\mathcal{L}$, and $\mathcal{A} \models \varphi$, we can extend $\mathcal{A}$ to a structure $\mathcal{A}'$ by defining $f^{\mathcal{A}'}$ so that for every $a_1, ..., a_n \in A = A', \mathcal{A} \models \psi(y/f(a_1, ..., a_n))$. Of course. $\mathcal{A}' \models \varphi'$. Note that n may be 0; that is, f may be a constant symbol. □

Skolemization

Corollary 9.6: For any set S of sentences of a language $\mathcal{L}$, we can construct a set S' of universal sentences of a language $\mathcal{L}'$ which is an expansion of $\mathcal{L}$ gotten by adding on new function symbols such that S and S' are equisatisfiable.

Proof: Apply the construction supplied by Theorem 9.4 to each sentence φ of S separately to introduce new function symbols f_φ for each sentence φ of S and form the corresponding universal sentence φ'. Let S' be the collection of all of these sentences φ' and $\mathcal{L}'$ the corresponding expansion of $\mathcal{L}$. As in the proof of the theorem it is clear that, if a structure $\mathcal{A}'$ for $\mathcal{L}'$ is a model of S'. then it is one of S. The proof also shows how to extend any model of S to one of S' by defining each new function symbol f_φ independently of what is done for the others. □

Example: Skolemization

Possible Skolemizations corresponding to $\forall u \exists v \forall w \exists z [P(u, v) \vee \neg Q(w, z)]$ and $\forall u \forall w \exists v \exists z [P(u, v) \vee \neg Q(w, z)]$ are $\forall u \forall w [P(u, f_1(u)) \vee \neg Q(w, f_2(u, w))]$ and $\forall u \forall w [P(u, f_1(u, w)) \vee \neg Q(w, f_2(u, w))]$ respectively.

Skolemization

Corollary 9.9: For any set S of sentences of $\mathcal{L}$, there is a set T of clauses in a language $\mathcal{L}'$ gotten by adding new function symbols to $\mathcal{L}$ such that S and T are equisatisfiable.

Proof: Consider the set S' of universal sentences $\forall\vec{x}\varphi'(\vec{x})$ equisatisfiable with S given by Corollary 9.6. Let T' consist of the equivalent open formulas $\varphi'(\vec{x})$ gotten by dropping the initial universal quantifiers from the elements of S'. If we view each atomic formula of $\mathcal{L}'$ as a propositional letter and form the CNF equivalent $\psi_\varphi = \wedge\psi_{\varphi,i}$ of each formula $\varphi' \in T'$, we get a set of formulas T'' each in CNF and each equivalent to the one of $T' : \wedge\psi_{\varphi,i} = \psi_\varphi \equiv \varphi' \equiv \varphi$ for each $\varphi \in S$. (For each φ, ψ_φ is equivalent to φ' by Theorem 4.8.) The desired set T of clauses then consists precisely of the set of all conjuncts from all of the formulas φ in $T'' : T = \{\psi_{\varphi,i} \mid \varphi \in S\}$. □

10. Herbrand's Theorem

Definition 10.1: The set of ground (i.e., variable-free) terms of a language $\mathcal{L}$, is called the Herbrand universe of $\mathcal{L}$. A structure $\mathcal{A}$ for $\mathcal{L}$ is an Herbrand structure if its universe A is the Herbrand universe of $\mathcal{L}$ and, for every function symbol f of $\mathcal{L}$ and elements $t_1, ..., t_n$ of A, $f^{\mathcal{A}}(t_1, ..., t_n) = f(t_1, ..., t_n)$. (We include here the requirement that $c^{\mathcal{A}} = c$ for each constant symbol c of $\mathcal{L}$.)

Definition 10.2: If S is a set of sentences of $\mathcal{L}$, then an Herbrand model $\mathcal{M}$ of S is an Herbrand structure for $\mathcal{L}$ which is a model of S, i.e., every sentence of S is true in $\mathcal{M}$.

Example: Herbrand universe

If our language $\mathcal{L}$ contains the constants a and c, a unary function symbol f and a binary one g and predicates P, Q, R, then the Herbrand universe H for $\mathcal{L}$ is
$\{a, c, f(a), f(c), g(a, c), ff(a), ff(c), f(g(a, c)), g(a, f(a)), g(a, f(c)),$
$...g(a, g(a, c)), ..., g(f(a), f(c)), ..., fff(a), ...\}.$

Herbrand's theorem

Theorem 10.4: Let $S = \{\varphi_i(x_1, ..., x_{n_i})\}$ be a set of open formulas of a language $\mathcal{L}$. Either

(i) S has an Herbrand model or

(ii) S is unsatisfiable and, in particular, there are finitely many ground instances of elements of S whose conjunction is unsatisfiable.

The latter case, (ii), is equivalent to

(ii') There are finitely many ground instances of the negations of formulas of S whose disjunction is valid. (As we may view these ground instances as built from propositional letters, the disjunction being valid is equivalent to its being a truth-functional tautology.)

Proof: Let S' consist of all ground instances from $\mathcal{L}$ of formulas from S. Consider the CST from S' (in the language $\mathcal{L}$ alone, i.e., with no additional constant symbols added on) starting with $F(\alpha \wedge \neg\alpha)$ for any sentence α. There are two possible outcomes. First, there might be a (possibly infinite) noncontradictory path in the tableau. In this case, the proof of Theorem 7.3

supplies us with a model $\mathcal{A}$ of S' whose elements are the ground terms of $\mathcal{L}$, i.e., an Herbrand model for S'. By definition of S' and of tableau proofs from S', $\varphi(t_1, ..., t_n)$ is true in $\mathcal{A}$ for every $\varphi \in S$ and every $t_1, , t_n$ in the Herbrand universe. Thus the structure $\mathcal{A}$ defined on the Herbrand universe by the path is a model for S.

The other possibility is that the tableau is finite and contradictory. In this case, the tableau is, by definition, a proof of the unsatisfiability of the set of elements of S' appearing in the tableau and so we have the unsatisfiable conjunction required in (ii). Moreover, S cannot be satisfiable: A model for S is one in which $\varphi_i(x_1, ..., x_{n_i})$ is valid, i.e., true for every instance of the free variables $x_1, ..., x_{n_i}$ for every $\varphi_i \in S$. Any example of (ii), however, directly exhibits a set of such instances that cannot be simultaneously satisfied in any model.

Finally, by Theorem 4.8 we may manipulate the variable-free formulas as propositional letters. The unsatisfiability of the conjunction as required in (ii) is then equivalent by propositional rules to the disjunction of their negations being valid or a tautology. Thus, (ii) and (ii') are equivalent. □

Variations on Herbrand's theorem

Corollary 10.5: If $\varphi(\vec{x})$ is a quantifier-free formula in a language $\mathcal{L}$ with at least one constant symbol, then $\exists\vec{x}\varphi(\vec{x})$ is valid if and only if there are ground terms $\vec{t_i}$ of $\mathcal{L}$ such that $\varphi(\vec{t_1}) \vee ... \vee \varphi(\vec{t_n})$ is a tautology.

Proof: First, note that $\exists\vec{x}\varphi(\vec{x})$ is valid $\Leftrightarrow$ $\forall\vec{x}\neg\varphi(\vec{x})$ is unsatisfiable $\Leftrightarrow$ $\neg\varphi(\vec{x})$ is unsatisfiable. By Theorem 10.4 (ii), $\neg\varphi(\vec{x})$ is unsatisfiable iff there are finitely many ground terms $\vec{t_i}$ of $\mathcal{L}$ such that $\varphi(\vec{t_1}) \vee ... \vee \varphi(\vec{t_n})$ is a tautology. □

Variations on Herbrand's theorem

Theorem 10.6: A set S of clauses is unsatisfiable if and only if the set S' of all ground instances from the Herbrand universe of the clauses in S is unsatisfiable.

Proof: If some set of instances of elements of S (instantiated with terms from the Herbrand universe) is unsatisfiable, then S, which asserts the validity of its member clauses, is surely unsatisfiable. In the other direction, if S is unsatisfiable, then, by Herbrand's theorem (ii), there is, in fact, a finite set of instances of clauses of S that is unsatisfiable. □

Variations on Herbrand's theorem

Theorem 10.7: Let φ be a sentence in prenex normal form in a language $\mathcal{L}$, ψ a prenex equivalent of $\neg\varphi$ and $\theta(\vec{x})$ an open Skolemization of ψ in the language $\mathcal{L}'$ as in Theorem 9.4. (Note that the free variables in ψ are precisely the existentially quantified ones of φ.) Then φ is valid if and only if there are terms $\vec{t_1}, ..., \vec{t_n}$ of $\mathcal{L}'$ such that $\neg\theta(\vec{t_1}) \vee ... \vee \neg\theta(\vec{t_n})$ is a tautology.

Proof: By Corollary 10.5, it suffices to prove that φ is valid if and only if $\exists\vec{x}\neg\theta(\vec{x})$ is valid. Now φ is valid iff $\neg\varphi$ is not satisfiable. On the other hand, Theorem 9.4 says that $\neg\varphi$ is satisfiable if and only if $\theta(\vec{x})$ is satisfiable. Thus, φ is valid iff $\theta(\vec{x})$ is not satisfiable. Finally, note that $\theta(\vec{x})$ (or, equivalently, $\forall\vec{x}\theta$) is not satisfiable iff $\exists\vec{x}\neg\theta(\vec{x})$ is valid. □

11. Unification

Consider how to resolve the two clauses in resolution theorem proving: $C_1 = \{P(f(x), y), \neg Q(a, b, x)\}$ and $C_2 = \{\neg P(f(g(c)), g(d)\}$. We can resolve C_1 and C_2 by directly substituting $g(c)$ for x and $g(d)$ for y to get $\{\neg Q(a, b, g(c))\}$.

Substitution

Definition 11.1: A substitution θ is a finite set of the form $\{x_1/t_1, x_2/t_2, ..., x_n/t_n\}$ where the x_i are distinct variables and each t_i is a term other than x_i. If the t_i are all ground terms, we call θ a ground substitution. If the t_i are distinct variables, we call θ a renaming substitution.

Definition 11.2: An expression is any term or literal. Given a substitution θ and an expression E (or a set of expressions S) we write $E\theta$ $(S\theta)$ for the result of replacing each occurrence of x_i in E (in every element of S), by t_i for every $i \leq n$. If the resulting expression $E\theta$ (set of expressions $S\theta$) is ground, i.e., variable-free, then the substitution is called a ground instance of E (S).

Example: Substitution

(i) Let $S = \{f(x, g(y)), P(a, x), Q(y, z, b), \neg P(y, x)\}$, $\theta = \{x/h(a), y/g(b), z/c\}$. Then $S\theta = \{f(h(a), g(g(b))), P(a, h(a)), Q(g(b), c, b), \neg P(g(b), h(a))\}$. Here θ is a ground substitution and $S\theta$ is a ground instance of S.

(ii) Let S be as in (i) and let $\sigma = \{x/h(y), y/g(z), z/c\}$. Then $S\sigma = \{f(h(y), g(g(z))), P(a, h(y)), Q(g(z), c, b), \neg P(g(z), h(y))\}$.

Example: Composition of substitutions

Let $E = P(x, y, w, u)$ and consider the two substitutions $\theta = \{x/f(y), y/g(z), w/v\}$ and $\sigma = \{x/a, y/b, z/f(y), v/w, u/c\}$. Then $E\theta = P(f(y), g(z), v, u)$ and $(E\theta)\sigma = P(f(b), g(f(y)), w, c)$. We have $\theta\sigma = \{x/f(b), y/g(f(y)), u/c, z/f(y), v/w\}$

Composition of substitutions

(i) If $\theta = \{x_1/t_1, ..., x_n/t_n\}$ and $\sigma = \{y_1/s_1, ..., y_m/s_m\}$, then $\theta\sigma$ is the substitution $\{x_1/t_1\sigma, ..., x_n/t_n\sigma, y_1/s_1, ..., y_m/s_m\}$ with any $x_i/t_i\sigma$ for which $x_i = t_i\sigma$ and any y_j/s_j for which $y_j \in \{x_1, ..., x_n\}$ removed.

(ii) The empty substitution ϵ (which does nothing to any expression) is an identity for this operation, i.e., $\theta\epsilon = \epsilon\theta = \theta$ for every substitution θ.

Composition of substitutions

Proposition 11.6: For any expression E and substitutions θ, ψ and σ:

(i) $(E\theta)\sigma = E(\theta\sigma)$ and

(ii) $(\psi\theta)\sigma = \psi(\theta\sigma)$

Proof: Let θ and σ be as in Definition 11.5 and let $\psi = \{z_1/r_1, ..., z_k/r_k\}$. As the result of a substitution consists simply of replacing each variable in an expression by some term, it suffices to consider the case in which E is a variable, say v, in (i) and the result of applying $(\psi\theta)\sigma$ and $\psi(\theta\sigma)$ to v in (ii).

(i) We divide the argument into two cases.

Case 1: $v \notin \{x_1, ..., x_n\}$. In this case $v\theta = v$ and $(v\theta)\sigma = v\sigma$. If $v \notin \{y_1, ..., y_m\}$, then $v\sigma = v = v(\theta\sigma)$ as $v \notin \{x_1, ..., x_n, y_1, ..., y_m\}$ and so no substitution is made. If, on the other hand, $v = y_j$ for some $j \leq n$, then $y_j \notin \{x_1, ..., x_n\}$, $(v\theta)\sigma = v\sigma = s_j = v(\theta\sigma)$.

Case 2: $v = x_i$ for some $i \leq n$. In this case $v\theta = t_i$ and $(v\theta)\sigma = t_i\sigma$ but this is exactly $v(\theta\sigma)$ by definition.

(ii) The result follows from several applications of (i):

$$\begin{aligned} v((\psi\theta)\sigma) &= (v(\psi\theta))\sigma \\ &= ((v\psi)\theta)\sigma \\ &= (v\psi)(\theta\sigma) \\ &= v(\psi(\theta\sigma)) \end{aligned}$$

□

Unifier

Definition 11.7: If $S = \{E_1, , E_n\}$ is a set of expressions, we say a substitution θ is a unifier for S if $E_1\theta = E_2\theta = ... = E_n\theta$, i.e., $S\theta$ is a singleton. S is said to be unifiable if it has a unifier.

(i) Neither $\{P(x, a), P(b, c)\}$ nor $\{P(f(x), z), P(a, w)\}$ is unifiable.

(ii) $S_1 = \{P(x, c), P(b, c)\}$ and $S_2 = \{P(f(x), y), P(f(a), w)\}$ are both unifiable. The first can be unified by $\{x/b\}$ and only by this substitution. For S_2, $\theta = \{x/a, y/w\}$ unifies S_2 but so do $\sigma = \{x/a, y/a, w/a\}$ and $\psi = \{x/a, y/b, w/b\}$ as well as many others.

Most general unifier

Definition 11.9: A unifier θ for S is a most general unifier (mgu) for S if, for every unifier σ for S, there is a substitution λ such that $\theta\lambda = \sigma$.

Up to renaming variables there is only one result of applying an mgu:

Theorem 11.10: If θ and ψ are both mgu's for S, then there are renaming substitutions σ and λ (i.e., ones that consist solely of replacements of distinct variables by other distinct variables) such that $S\theta\sigma = S\psi$ and $S\theta = S\psi\lambda$.

Proof: By the definition of an mgu there are σ and λ such that $S\theta\sigma = S\psi$ and $S\psi\lambda = S\theta$. Clearly, we may assume that σ and λ make substitutions only for variables occurring in $S\theta$ and $S\psi$, respectively. (They consist of the single terms $E\theta$ and $E\psi$, respectively, as θ and ψ both unify S.) Suppose σ makes some substitution x_i/t_i where t_i is not a variable or a constant. In this case the complexity (e.g., length) of the expression $E\theta\sigma$ in $S\theta\sigma = \{E\theta\sigma\}$ must be strictly larger than that of $E\theta$ in $S\theta$. As no substitutions of terms for variables (e.g., λ) can decrease the length of an expression we could not then have $S\psi\lambda = S\theta\sigma\lambda = S\theta$ as required. If there were in σ a substitution x_i/c, for some constant c, then no further substitution (e.g., λ) could return the resulting instances of c in an expression $E\theta\sigma$ in $S\theta\sigma$ back to instances of the variable x_i in $E\theta \in S\theta$. Thus, once again, we could not have $S\theta\sigma\lambda = S\theta$ for any λ. We now know that σ can contain only substitutions of one variable by another. If σ identified distinct variables by such a substitution, then λ could not distinguish them again. Thus σ (and similarly λ) is simply a renaming substitution. □

Exercises

1. Exercises 4, 5 in page 133
2. Exercise 2, 4 in page 136
3. Exercises 1, 3 in page 141

The Unification Algorithm

Consider the example $S_1 = \{f(x, g(x)), f(h(y), g(h(z)))\}$. The unification of S_1 requries to unify $T_1 = \{x, h(y)\}$ and $T_2 = \{g(x), g(h(z))\}$. The former gives the substitution $\{x/h(y)\}$, then the elements in T_2 becomes $g(h(y))$ and $g(h(z))$. Then we can unify them by applying $\{y/z\}$. Thus the composition $\{x/h(y)\}\{y/z\} = \{x/h(z), y/z\}$ is our desired unifier.

However, the example $S_2 = \{f(h(x), g(x)), f(g(x), h(x))\}$ is not unifiable.

Disagreement set

Definition 12.1: Let S be a finite nonempty set of expressions. To define the disagreement set of S find the first (i.e., leftmost) position at which not all elements E of S have the same symbol. The set of subexpressions of each $E \in S$ that begin at this position is the disagreement set $D(S)$ of S. (In terms of formation trees, we find the lexicographically least node of the formation trees associated with each expression such that not all the labels of these nodes begin with the same symbol. $D(S)$ is then the set of labels of these nodes.)

E.g., For the sets of expressions $S_1 = \{f(x, g(x)), f(h(y), g(h(z)))\}$ and $S_2 = \{f(h(x), g(x)), f(g(x), h(x))\}$ considered above, the disagreement sets are $D(S_1) = \{x, h(y)\}$ and $D(S_2) = \{h(x), g(x)\}$. For $T_1 = S_1\{x/h(y)\} = \{f(h(y), g(h(y))), f(h(y), g(h(z)))\}$ the disagreement set is $\{y, z\}$.

The Unification Algorithm

Let S be a set of expressions. We unify it as follows:

Step 0. Set $S_0 = S, \ \sigma_0 = \epsilon$.

Step $k+1$. If S_k is a singleton, terminate the algorithm with the announcement that $\sigma_0\sigma_1...\sigma_k$ is an mgu for S. Otherwise, see if there is a variable v and a term t not containing v both of which are in $D(S_k)$. If not, terminate the algorithm with the announcement that S has no mgu. (Note that, in this case, it is at least clear that S_k is not unifiable.) If so, let v and t be the least such pair (in any fixed ordering of terms). (Indeed, we could nondeterministically choose any such t and v as will become clear from the proof that the algorithm succeeds.) Set $\sigma_{k+1} = \{v/t\}$ and $S_{k+1} = S_k\sigma_{k+1}$ and go on to step $k+2$.

Example: the unification algorithm

Let $S = \{P(f(y, g(z)), h(b)), P(f(h(w), g(a)), t), P(f(h(b), g(z)), y)\}$.

Step 1. $S = S\epsilon = S_0\sigma_0$ is not a singleton. $D(S_0) = \{y, h(w), h(b)\}$. Depending on the ordering of terms, there are two possibilities for $\sigma_1 : \{y/h(w)\}$ and $\{y/h(b)\}$. It is better to choose the second (see Step 2) but suppose we are not so clever and blindly set $\sigma_1 = \{y/h(w)\}$. We then get $S_1 = S_0\sigma_1$ which is $\{P(f(h(w), g(z)), h(b)), P(f(h(w), g(a)), t), P(f(h(b), g(z)), h(w))\}$.

Step 2. $D(S_1) = \{w, b\}, \sigma_2 = \{w/b\}$ (so we get to $\{y/h(b)\}$ after all). Then S_2 is $\{P(f(h(b), g(z)), h(b)), P(f(h(b), g(a)), t), P(f(h(b), g(z)), h(b))\}$.

Step 3. $D(S_2) = \{z, a\}, \sigma_3 = \{z/a\}$. Then S_3 is $\{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), t), P(f(h(b), g(a)), h(b))\}$.

Step 4. $D(S_3) = \{h(b), t\}, \sigma_4 = \{t/h(b)\}$. Then S_4 is $\{P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b)), P(f(h(b), g(a)), h(b))\}$.

Step 5. S_4 is a singleton and the mgu for S is $\{y/h(w)\}\{w/b\}\{z/a\}\{t/h(b)\} = \{y/h(b), w/b, z/a, t/h(b)\}$.

Correctness of the unification algorithm

Theorem 12.5: For any S, the unification algorithm terminates at some step $k+1$ with a correct solution, i.e., either S is not unifiable as announced or $\psi = \sigma_0\sigma_1...\sigma_k$ is in fact an mgu for S. Moreover, ψ has the special property that for any unifier θ of S, $\theta = \psi\theta$.

Proof: First of all, the algorithm always terminates as each nonterminal step eliminates all occurrences of one of the finitely many variables in S. It is obvious that if the algorithm terminates with an announcement that there is no unifier, then S is not unifiable. On the other hand, if the algorithm terminates with the announcement that $\psi = \sigma_0...\sigma_n$ is an mgu for S, then it is at least clear that ψ is a unifier for S. Suppose then that θ is any unifier for S. We must show that $\theta = \psi\theta$. We prove by induction that, for every i, $\theta = \sigma_0...\sigma_i\theta$.

For $i = 0$, the claim clearly holds. Suppose we have $\theta = \sigma_0\sigma_1...\sigma_i\theta$ and $\sigma_{i+1} = \{v/t\}$. It suffices to show that the substitutions $\sigma_{i+1}\theta$ and θ are equal. We show that their actions on each variable are the same. For $x \neq v,\ x\sigma_{i+1}\theta$ is clearly the same as $x\theta$. For v itself $v\sigma_{i+1} = t\theta$. As θ unifies $S\sigma_0...\sigma_i$ and v and t belong to $D(S\sigma_0...\sigma_i),\ \theta$ must unify v and t as well, i.e., $t\theta = v\theta$ as required. □

The unification algorithm

The unification algorithm given here is simple but inefficient. The search for a v and t with v not occurring in t can take excessive amount of time.

Let $S = \{P(x_1, ..., x_n), P(f(x_0, x_0), ..., f(x_{n-1}, x_{n-l}))\}$:

$$\begin{aligned}
D(S_0) &= \{x_1, f(x_0, x_0)\};\ \sigma_1 = \{x_1/f(x_0, x_0)\}; \\
S_1 &= \{P(f(x_0, x_0), x_2, ..., x_n), P(f(x_0, x_0), \\
&\qquad f(f(x_0, x_0), f(x_0, x_0)), f(x2, x2), ..., f(x_{n-1}, x_{n-1}))\} \\
D(S_1) &= \{x_2, f(f(x_0, x_0), f(x_0, x_0))\};\ \sigma_2 = \{x_2/f(f(x_0, x_0), f(x_0, x_0))\}; \\
&\vdots
\end{aligned}$$

Before announcing σ_1 we had to check that x_1 was not either of the two occurrences of variables in $f(x_0, x_0)$. For σ_2 there were four occurrences to check. In general $D(S_{i+1})$ will have twice as many occurrences of variables as $D(S_i)$ and so the “occurs check” takes exponential time.

Resolution

Consider formulas in clausal form. Each clause is understood as its universal closure. There are no connections between the variables of distinct clauses. To reflect this syntactically, we generally rename variables when using two clauses together so that they have no variables in common. (This procedure is called standardizing the variables apart.)

Definition 13.2: Suppose that we can rename the variables of C_1 and C_2 so that they have no variables in common and are of the form $C_1' \sqcup \{P\vec{t_1}, ..., P\vec{t_n}\}$ and $C_2' \sqcup \{\neg P\vec{s_1}, ..., \neg P\vec{s_m}\}$, respectively. If σ is an mgu for $\{P\vec{t_1}, ..., P\vec{t_n}, P\vec{s_1}, ..., P\vec{s_m}\}$, then $C_1'\sigma \cup C_2'\sigma$ is a resolvent of C_1 and C_2. ($C_1'\sigma \cup C_2'\sigma$ is also called the child of the parents C_1 and C_2.)

Resolution proofs

Resolution proofs of C from S and resolution refutations of S in both linear and tree form are defined as in the propositional case (Definitions I.8.4 and I.8.6) except that we use the version of the resolution rule given above and allow the premises inserted from S, or equivalently the leaves of the tree proof, to be $C\sigma$ for any renaming substitution σ and any $C \in S$. Similarly, we define $\mathcal{R}(S)$ as the closure under resolution of the set of all renamings of elements of S.

Resolution proofs

Note that the renaming of variables is necessary. For example,the sentence $\{\{P(x)\}, \{\neg P(f(x))\}\}$ is (unsatisfiable and) resolution refutable but the clauses cannot be unified without renaming the variables.

We cannot assume in Definition 13.2 that n or m are equal to 1 as we did in propositional logic. We must be able to eliminate several literals at once. (This aspect of the procedure is often called factoring.) For example, $S = \{\{P(x), P(y)\}, \{\neg P(x)\}\}$ is (unsatisfiable and) resolution refutable but no resolution proof from S that eliminates only one literal at a time can produce $\square$.

Exampple: resolution proofs

We can resolve

$C_1 = \{Q(x), \neg R(y), P(x, y), P(f(z), f(z))\}$ and

$C_2 = \{\neg N(u), \neg R(w), \neg P(f(a), f(a)), \neg P(f(w), f(w))\}$ to get

$C_3 = \{Q(f(a)), \neg R(f(a)), \neg N(u), \neg R(a)\}$.

To do this we unify

$\{P(x, y), P(f(z), f(z)), P(f(a), f(a)), P(f(w), f(w))\}$

via the mgu $\{x/f(a), y/f(a), z/a, w/a\}$ and perform the appropriate substitutions and union on C_1 and C_2.

Exampple: resolution proofs

From (a) and (b) below we wish to conclude (c):

(a) $\forall x \forall y \forall z[P(x,y) \wedge P(y,z) \rightarrow P(x,z)]$ (transitivity)

(b) $\forall x \forall y[P(x,y) \rightarrow P(y,x)]$ (symmetry)

(c) $\forall x \forall y \forall z[P(x,y) \wedge P(z,y) \rightarrow P(x,z)]$.

In clausal form, we can derive C_3 from $S = \{C_1, C_2\}$ where

$C_1 = \{\neg P(x,y), \neg P(y,z), P(x,z)\}$,

$C_2 = \{\neg P(u,v), P(v,u)\}$ and

$C_3 = \{\neg P(x,y), \neg P(z,y), P(x,z)\}$.

(Note that we have standardized the clauses of S apart.)

Correct answer substitution

Definition 13.5: If P is a program and $G = \{\neg A_1, ..., \neg A_n\}$ a goal clause, we say that the substitution θ (for the variables of G) is a correct answer substitution if $(A_1 \wedge A_2 \wedge ... \wedge A_n)\theta$ is a logical consequence of P (that is, of its universal closure).

Soundness of resolution

Theorem 13.6 If $\Box \in \mathcal{R}(S)$, then S is unsatisfiable.

Proof: Suppose, for the sake of a contradiction, that $\mathcal{A} \models S$. It suffices to show that if $\mathcal{A} \models C_1, C_2$ and C is a resolvent of C_1, C_2 then $\mathcal{A} \models C$, i.e., $\mathcal{A} \models C\tau$ for every ground substitution τ. (If so, we could show by induction that $\mathcal{A} \models C$ for every $C \in \mathcal{R}(S)$. As $\mathcal{R}(S)$ contains $\Box$, we would have the desired contradiction.) The only point to notice here is that if $\mathcal{A} \models C_i$, then $\mathcal{A} \models C_i\sigma_i$ for any C_i as the C_i are open. For every ground instantiation τ of the variables of $C = C_1'\sigma \cup C_2'\sigma$ we can argue as in the propositional case. (See Lemma I.8.12 and Theorem I.8.11.) As, for each ground instantiation τ, either $C_1'\sigma\tau$ or $C_2'\sigma\tau$ is true in $\mathcal{A}$ (depending on whether the literal resolved on is true in $\mathcal{A}$ or not and in which of the $C_i'\tau$ it appears positively), then so is their union $C\tau$. □

Completeness of resolution

Lemma 13.7 If C_1' and C_2' are ground instances (via the substitutions θ_1 and θ_2) of C_1 and C_2, respectively, and C' is a resolvent of C_1' and C_2', then there is a resolvent C of C_1 and C_2 such that C' is a ground instance of C (via $\theta_1\theta_2$ if C_1 and C_2 have no variables in common).

Proof: As the resolution rule allows us to rename the variables in C_1 and C_2 as part of the resolution, we may as well assume that they (and so also θ_1 and θ_2) have no variables in common. As $C_1' = C_1\theta_1$ and $C_2' = C_2\theta_2$ are resolvable, say on the ground literal $P(t_1, ..., t_n)$, there are sets of literals

$A_1 = \{P(\vec{s}_{1,1}), ..., P(\vec{s}_{1,n_1})\} \subseteq C_1\}$ and

$A_2 = \{\neg P(\vec{s}_{2,1}), ..., P(\vec{s}_{2,n_2})\} \subseteq C_2\}$

which become unified to $\{P(t_1, ..., t_n)\}$ and $\{\neg P(t_1, ..., t_n)\}$ by θ_1 and θ_2, respectively. As the sets of variables in θ_1 and θ_2 are disjoint, $\theta_1\theta_2$ unifies

both sets of literals A_1 and A_2 simultaneously. Thus, by the definition of resolution for the predicate calculus (Definition 13.2),
$C = ((C_1 - A_1) \cup (C_2 - A_2))\sigma$ is a resolvent of C_1 and C_2 where σ is the mgu for

$$\{\neg P(\vec{s}_{1,1}), ..., \neg P(\vec{s}_{1,n_1})\} \cup \{\neg P(\vec{s}_{2,1}), ..., \neg P(\vec{s}_{2,n_2})\}$$

given by the unification algorithm. The only point left to verify is that C' is an instance of C. We claim that $C' = C\theta_1\theta_2$. Note that as $\theta_1\theta_2$ unifies $\neg A_1 \cup A_2$, the special property of the mgu given by our algorithm (Theorem 12.5) guarantees that $\sigma\theta_1\theta_2 = \theta_1\theta_2$. Thus

$$\begin{aligned}
C\theta_1\theta_2 &= ((C_1 - A_1) \cup (C_2 - A_2))\sigma\theta_1\theta_2 \\
&= ((C_1 - A_1) \cup (C_2 - A_2))\theta_1\theta_2 \\
&= (C_1\theta_1 - A_1\theta_1) \cup (C_2\theta_2 - A_2\theta_2) \qquad \text{(by disjointness of variables)} \\
&= (C_1' - \{P(t_1, ..., t_n)\}) \cup (C_2' - \{\neg P(t_1, ..., t_n)\}) \\
&= C' \qquad \text{(by definition)}
\end{aligned}$$

□

Lifting Lemma

Lemma 13.8 Let S be a formula in a language $\mathcal{L}$ and let S' be the set of all ground instances of clauses in S in the Herbrand universe for $\mathcal{L}$. If T' is a resolution tree proof of C' from S', then there is a clause C of $\mathcal{L}$, a resolution tree proof T of C from S and a substitution θ such that $T\theta = T'$ (i.e., T and T' are labelings of the same tree and $C_i\theta = C_i'$ for C_i, C_i' the respective labels of each node of the common tree underlying T and T'. Thus, in particular, $C' = C\theta$). Moreover, if the leaves of T' are labeled R_i and each R_i is an instance of an S_i in S, then we may arrange it so that the corresponding leaves of T are labeled with renamings of the appropriate S_i.

Proof: We proceed by induction on the depth of resolution tree proofs from S'. For the base case of elements R_i of S', the lemma is immediate as each such R_i is a substitution instance of an element of S. Consider now a proof of C' from S' of depth $n + 1$. It consists of two proofs, T_1' and T_2' (of depth $\leq n$) of ground clauses C_1', C_2' from S' and a final resolution of C_1', C_2' to get C'. Suppose that $P(t_1, ..., t_n) \in C_1'$, $\neg P(t_1, ..., t_n) \in C_2'$ and that we resolved on this literal to get

$$C' = C_1' \cup C_2' - \{P(t_1, ..., t_n), \neg P(t_1, ..., t_n)\}.$$

By induction, we have predicate clauses C_1 and C_2, proof trees T_1 and T_2 of C_1 and C_2 and substitutions θ_1 and θ_2 such that $T_i\theta_i = T_i'$. (The leaves of T_i are also labeled appropriately by induction.) At the cost perhaps of renaming variables in T_1 and T_2, we may assume that θ_1 and θ_2 have no variables in common. (As the resolution rule allows for arbitrary renamings of the parents, the T_i remain resolution proofs. As our lemma only calls for the leaves to be labeled with some renamings of the given clauses from S, this renaming does not alter the fact that we have the leaves appropriately labeled.) We now apply Lemma 13.7 to get a resolvent C of C_1 and C_2 with $C' = C\theta_1\theta_2$. We can now form a resolution tree proof T from S of C by combining T_1 and T_2. As θ_1 and θ_2 are disjoint, $T\theta_1\theta_2$ restricted to T_1 and T_2 simply gives us back $T_1\theta_1$ and $T_2\theta_2$. Of course, on the remaining node C of T we have $C\theta_1\theta_2 = C'$. Thus T is the required predicate logic resolution proof from S of C and $\theta_1\theta_2$ is the substitution required in our lemma. □

Corollary 13.9: If T' is a resolution tree proof of $\Box$ each of whose leaves L_i is labeled with a ground instance R_i of the clause S_i, then there is a relabeling T of the underlying tree of T' that gives a resolution proof of $\Box$ each of whose leaves L_i is labeled with (a renaming) of S_i.

Proof: This is simply the special case of the theorem with $C' = \Box$. The only point to notice is that the only clause C that can have $\Box$ as a substitution instance is $\Box$ itself. □

Completeness of resolution

Theorem 13.10 If S is unsatisfiable, then $\Box \in \mathcal{R}(s)$.

Proof: Let S' be the set of all ground instances of clauses in S in the Herbrand universe for the language $\mathcal{L}$ of S. By one of the consequences (Theorem 10.6) of Herbrand's theorem, S and S' are equisatisfiable. Thus if we assume that S is unsatisfiable, then so is S'. By the completeness of resolution for propositional logic (Theorem I.8.15 or I.8.22) we then know that $\Box \in \mathcal{R}_p(S')$ where we use $\mathcal{R}_p$ to represent the resolution procedure in propositional logic. (As usual we consider the atomic formulas as propositional letters in this situation.) The completeness of resolution for predicate logic (i.e., $\Box \in \mathcal{R}(s)$ if S is unsatisfiable) is now immediate from Corollary 13.9. □

Exercises

1. Exercise 2 in page 144

2. Exercises 5, 6 in page 152

14. Refining resolution: Linear resolution

Definition 14.1: Let C be a clause and S a formula.

(i) A linear deduction of C from S is a sequence $\langle C_0, B_0\rangle, ..., \langle C_n, B_n\rangle$ of pairs of clauses such that C_0 and each B_i are either renaming substitutions of elements of S or some C_j for $j < i$; each C_{i+1}, $i \leq n$, is a resolvent of C_i and B_i and $C_{n+1} = C$.

(ii) C is linearly deducible from S, $S \vdash_{\mathcal{L}} C$, if there is a linear deduction of C from S. There is a linear resolution refutation of S if $\Box$ is linearly deducible from S. $\mathcal{L}(S)$ is the set of all clauses linearly deducible from S.

The elements of S are frequently called input clauses. The C_i are called center clauses and the B_i side clauses.

Supports of formula

Definition 14.3: $U \subseteq S$ is a set of support for S if $S - U$ is satisfiable. We say that a linear resolution proof $\langle C_i, B_i \rangle$, $i \leq n$, of C from S has support U if $C_0 \in U$.

The intuition here is that we consider a formula $S \in \mathtt{UNSAT}$. In this case the "cause" of the unsatisfiability has been isolated in U (which "supports" the fact that $S \in \mathtt{UNSAT}$).

Minimal unsatisfiability

Definition 14.5: S is minimally unsatisfiable if it is unsatisfiable but every proper subset is satisfiable, i.e., $\{C\}$ is a set of support for S for every $C \in S$.

Lemma 14.6: If $S \in \mathtt{UNSAT}$, then there is a minimally unsatisfiable $S' \subseteq S$. Moreover, if U is a set of support for S, $U \cap S'$ is one for S'.

Proof: By compactness, some finite subset of S is unsatisfiable. If S' is an unsatisfiable subset of S with the least possible number of clauses, S' is certainly a minimally unsatisfiable subset of S. Let U be any set of support for S. If $U \cap S' = \emptyset$, S' would be contained in the satisfiable set $S - U$ for a contradiction. Thus $S' - (S' \cap U)$ is a proper subset of S' and so, by the minimality of S', is satisfiable. □

Completeness of linear resolution

Theorem 14.4: If $S \in \texttt{UNSAT}$ and U is a set of support for S, then there is a linear refutation of S with support U.

Proof: Our plan now is once again to reduce the proof to the case of propositional logic. As in the case of general resolution, we apply Herbrand's theorem. If S is unsatisfiable and has support U, so is S', the set of all ground instances of elements of S, and it has support U', the set of all ground instances of elements of U. We wish to show that any linear resolution proof T' of $\Box$ from S' with support U' lifts to one from S with support U. This is immediate from Corollary 13.9 to the lifting lemma. The lifting lemma preserves the shape of the resolution tree and so lifts linear proofs to linear proofs. It also lifts instances R_i of clauses S_i on the leaves of the tree to (renamings of) S_i. Thus if the clause C'_0 of the proof T' is in U' and so is an instance of a clause C in U, then it lifts to a (renaming of) the same clause C. □

Completeness of linear resolution

Proof: (of the propositional version of Theorem 14.4): By Lemma 14.6, it suffices to consider only those S that are minimally unsatisfiable. (Any linear resolution refutation of $S' \subseteq S$ with support $U \cap S'$ is one of S with support U by definition.) We proceed by induction on $E(S) =$ the excess literal number of S, that is, the number of occurrences of literals in all clauses of S minus the number of clauses in S. (Note that we need S to be finite to even define the excess literal number.) We in fact prove by induction that, for any $C \in S$, there is a linear refutation of S that begins with C, i.e. $C = C_0$ in the proof tree. At the bottom, if $E(S) = -1$, $\Box \in S$ and there is nothing to prove. Suppose now that $E(S) \geq 0$.

Case 1. C is a unit clause, i.e., it contains exactly one literal l. There must be a clause $C' \in S$ with $\bar{l} \in C'$ as otherwise any assignment satisfying $S - C$ (which is satisfiable by the minimality of S) could be extended to one satisfying S by adding on l. Note that $l \notin C'$ for if it did, C' would be a tautology and S would not be minimally unsatisfiable contrary to our assumption. Thus $C' - \{\bar{l}\}$ is in S^l by the definition of S^l (Definition I.8.16). If $C' = \{\bar{l}\}$, we are done as we can simply resolve C and C' to get $\square$. Suppose then that $C' = \{\bar{l}, ...\}$ has more than one literal. As $S \in \mathtt{UNSAT}$, $S^l \in \mathtt{UNSAT}$ by Lemma I.8.19. Each clause removed from S in forming S^l has at least one literal (l) (again by definition). Thus their removal cannot increase the excess literal number. On the other hand, at least C' loses one literal ($\bar{l}$) in its transition to S^l. Thus $E(S^l) < E(S)$.

We next claim that S^l is also minimally unsatisfiable: Suppose $D \in S^l$ but $S^l - \{D\}$ is unsatisfiable. Now, by the definition of S^l, $D \in S$ or $D \cup \{\bar{l}\} \in S$ and in either case $l \notin D$. Let D' represent whichever clause belongs to S. We know, by the minimal unsatisfiability of S, that $S - \{D'\}$ is satisfiable. Let $\mathcal{A}$ be an assignment satisfying it. As $C = \{l\} \in S - \{D'\}$, $\mathcal{A} \models l$. Consider now any $F \in S^l - \{D\}$ and the associated $F' \in S - \{D'\}$. As $\mathcal{A} \models l$ and $\mathcal{A} \models F'$, $\mathcal{A} \models F$ in either case of the definition of F'. (F' is defined from F as D' was defined from D.) Thus $\mathcal{A} \models S^l - \{D\}$ contrary to our assumption.

Our induction hypothesis now gives us a linear resolution deduction of $\square$ from S^l starting with $C' - \{\bar{l}\} : \langle C_0, B_1\rangle, ..., \langle C_n, B_n\rangle$ with $C_0 = C' - \{\bar{l}\}$. Each B_i is a member of S^l or is C_j for some $j < i$ and C_n, B_n resolve to $\square$. We construct a new proof $\langle D_j, A_j\rangle$ in segments with the i^{th} one ending with $D_k = C_i$. We begin by setting $D_0 = \{l\} = C$ and $A_0 = C'$. Of course, they can be resolved to get $D_1 = C_0$. Now we proceed by induction. Suppose we have $A_j, j < k$ and $D_k = C_i$. If $B_i = C_j$ for some $j < i$, we let $A_k = C_j$ (which by induction is a previous D_m) and resolve to get $D_{k+1} = C_{k+1}$. Otherwise, $B_i \in S^l$ and we have two cases to consider. If $B_i \in S$ we set $A_k = B_i$ and resolve to get $D_{k+1} = C_{i+1}$. If $B_i \notin S$, then $B_i \cup \{\bar{l}\} \in S$. We set $A_k = B_i \cup \{\bar{l}\}$ and resolve to get $D_{k+1} = C_{i+1} \cup \{\bar{l}\}$. In this case, we set $A_{k+1} = \{l\}$ and resolve to get $D_{k+2} = C_{i+1}$ and so continue the induction. As $\{l\} = D_0$, we now have a linear resolution refutation of S as required.

Case 2. $C = \{l, ...\}$ has more than one literal. Now consider $S^{\bar{l}}$. As above, it is minimally unsatisfiable and has lower excess literal number than S. We thus have, by induction, a linear resolution deduction of $\Box$ from $S^{\bar{l}}$ starting with $C - \{l\}$. If we add on l to every center clause and to any side clause which is in $S^{\bar{l}}$ but not S, we get a linear proof of $\{l\}$ from S starting with C. Consider now $S' = S - \{C\} \cup \{\{l\}\}$. It too is unsatisfiable. (Any assignment satisfying it satisfies S.) As C has more than one literal, $E(S') < E(S)$. Now as $\Box \notin S'$, for any $S'' \subseteq S'$, $E(S'') \leq E(S')$. If we take $S'' \subseteq S'$ to be minimally unsatisfiable we have, by induction, a linear resolution proof of $\Box$ from $S'' \subseteq S \cup \{\{l\}\}$ beginning with $\{l\}$. (Note that $S' - \{l\} = S - \{C\}$ is satisfiable by the minimal unsatisfiability of S. Thus, any unsatisfiable subset S'' of S' must contain $\{l\}$.) Attaching this proof to the end of the one of $\{l\}$ from S gives the desired linear refutation of S starting with C. □

Chapter IV. Modal Logic

1. Possibility and Necessity; Knowledge or Belief

Formal modal logics were developed to make precise the mathematical properties of differing conceptions of such notions as possibility, necessity, belief, knowledge and temporal progression which arise in philosophy and natural languages.

Definition 1.1: If $\mathcal{L}$ is a language for (classical) predicate logic, we extend it to a modal language $\mathcal{L}_{\Box,\Diamond}$ by adding (to Definition II.2.1) two new primitive symbols $\Box$ and $\Diamond$. We add a new clause to the definition (II.2.5) of formulas:

(iv) If φ is a formula, then so are $\Box\varphi$ and $\Diamond\varphi$.

We write $\mathcal{L}$ for $\mathcal{L}_{\Box,\Diamond}$ if no confusion arises.

Possibility and Necessity

- $\Box$: “it is necessary that”, “it will always be true that”, “ I know” or “I believe”
- $\Diamond$: “it is possible that”, “it will eventually be true that”,

The semantics for a modal language $\mathcal{L}_{\Box,\Diamond}$ is based on a generalization of the structures for classical predicate logic of II.4 known as Kripke frames. Consider a collection W of “possible worlds”. Each world $w \in W$ constitutes a view of reality as represented by a structure $\mathcal{C}(w)$ associated with it. $w \Vdash \varphi$ means φ is true in the possible world w, read “w forces φ” or “φ is true at w”.

Possibility and Necessity

In dynamic logic of sequential programs, the “possible worlds” are the states of the machine.

- $s \Vdash_\alpha \Box_\alpha \varphi$ asserts that φ is true at any state s' s.t. there exists a legal execution sequence for α which starts in state s and eventually reaches state s'.

- $s \Vdash_\alpha \Diamond_\alpha \varphi$ asserts that φ is true at (at least) one state s' such that there exists α legal execution sequence for a which starts in state s and eventually reaches state s'.

The intended accessibility relation, S_α, is that s' is accessible from s, $sS_\alpha s'$, if and only if some execution of program α starting in state s ends in state s'.

2. Frames and Forcing

Fix a modal language $\mathcal{L}$ and assume that it has at least one constant symbol but no function symbols other than constants.

The elimination of function symbols does not result in a serious loss of expressiveness. We can systematically replace function symbols with relations. The work of a binary function symbol $f(x, y)$, for example, can be taken over by a ternary relation symbol $R_f(x, y, z)$ whose intended interpretation is that $f(x, y) = z$. A formula $\varphi(f(x, y))$ can then be systematically replaced by the formula $\exists z(R_f(x, y, z) \wedge \varphi(z))$.

Frames

Definition 2.1: Let $\mathcal{C} = (W, S, \{\mathcal{C}(p)\}_{p \in W})$ consist of a set W, a binary relation S on W and a function that assigns to each p in W a (classical) structure $\mathcal{C}(p)$ for $\mathcal{L}$ (in the sense of Definition II.4.1). To simplify the notation we write $\mathcal{C} = (W, S, \mathcal{C}(p))$ instead of the more formally precise version, $\mathcal{C} = (W, S, \{\mathcal{C}(P)\}_{p \in W})$. As usual, we let $C(p)$ denote the domain of the structure $\mathcal{C}(p)$. We also let $\mathcal{L}(p)$ denote the extension of $\mathcal{L}$ gotten by adding on a name c_a for each element a of $C(p)$ in the style of the definition of truth in II.4. We write either pSq or $(p, q) \in S$ to denote the fact that the relation S holds between p and q. We also describe this state by saying that q is accessible from (or a successor of) p. We say that $\mathcal{C}$ is a frame for the language $\mathcal{L}$, or simply an $\mathcal{L}$-frame if, for every p and q in W, pSq implies that $C(p) \subseteq C(q)$ and the interpretations of the constants in $\mathcal{L}(p) \subseteq \mathcal{L}(q)$ are the same in $\mathcal{C}(p)$ as in $\mathcal{C}(q)$.

Forcing for frames

Definition 2.2: Let $\mathcal{C} = (W, S, \mathcal{C}(P))$ be a frame for a language $\mathcal{L}$, p be in W and φ be a sentence of the language $\mathcal{L}(p)$. We give a definition of p forces φ, written $p \Vdash \varphi$ by induction on sentences φ.

(i) For atomic sentences φ, $p \Vdash \varphi \;\Leftrightarrow\; \varphi$ is true in $\mathcal{C}(p)$.

(ii) $p \Vdash (\varphi \rightarrow \psi) \;\Leftrightarrow\; p \Vdash \varphi$ implies $p \Vdash \psi$.

(iii) $p \Vdash \neg\varphi \;\Leftrightarrow\; p$ does not force φ (written $p \nVdash \varphi$).

(iv) $p \Vdash (\forall x)\varphi(x) \;\Leftrightarrow\;$ for every constant c in $\mathcal{L}(p)$, $p \Vdash \varphi(c)$.

(v) $p \Vdash (\exists x)\varphi(x) \;\Leftrightarrow\;$ there is a constant c in $\mathcal{L}(p)$ such that $p \Vdash \varphi(c)$.

(vi) $p \Vdash (\varphi \wedge \psi) \;\Leftrightarrow\; p \Vdash \varphi$ and $p \Vdash \psi$.

(vii) $p \Vdash (\varphi \vee \psi) \;\Leftrightarrow\; p \Vdash \varphi$ or $p \Vdash \psi$.

(viii) $p \Vdash \Box\varphi \;\Leftrightarrow\;$ for all $q \in W$ such that pSq, $q \Vdash \varphi$.

(ix) $p \Vdash \Diamond\varphi \;\Leftrightarrow\;$ there is a $q \in W$ such that pSq and $q \Vdash \varphi$.

If we need to make the frame explicit, we say that p forces φ in $\mathcal{C}$ and write $p \Vdash_{\mathcal{C}} \varphi$.

Definition 2.3: Let φ be a sentence of the language $\mathcal{L}$. We say that φ is forced in the $\mathcal{L}$-frame $\mathcal{C}$, $\Vdash_{\mathcal{C}} \varphi$, if every p in W forces φ. We say φ is valid, $\models \varphi$, if φ is forced in every $\mathcal{L}$-frame $\mathcal{C}$.

Example: Forcing for frames

For any sentence φ, the sentence $\Box\varphi \rightarrow \neg\Diamond\neg\varphi$ is valid: Consider any frame $\mathcal{C} = (W, S, \mathcal{C}(p))$ and any $p \in W$. We must verify that $p \Vdash \Box\varphi \rightarrow \neg\Diamond\neg\varphi$ in accordance with Clause (ii) of Definition 2.2. Suppose then that $p \Vdash \Box\varphi$. If $p \nVdash \neg\Diamond\neg\varphi$, then $p \Vdash \Diamond\neg\varphi$ (by (iii)). By Clause (ix), there is a $q \in W$ such that pSq and $q \Vdash \neg\varphi$. Our assumption that $p \Vdash \Box\varphi$ and Clause (viii) then tell us that $q \Vdash \varphi$, contradicting Clause (iii).

Example: Forcing for frames

We claim that $\Box\forall x\varphi(x) \to \forall x\Box\varphi(x)$ is valid. If not, there is a frame $\mathcal{C}$ and a p such that $p \Vdash \Box\forall x\varphi(x)$ but $p \nVdash \forall x\Box\varphi(x)$. If $p \nVdash \forall x\Box\varphi(x)$, there is, by Clause (iv), a $c \in \mathcal{L}(p)$ such that $p \nVdash \Box\varphi(c)$. There is then, by Clause (ix), a $q \in W$ such that pSq and $q \nVdash \varphi(c)$. As $p \Vdash \Box\forall x\varphi(x)$, $q \Vdash \forall x\varphi(x)$ by (ix). Finally, $q \Vdash \varphi(c)$ by (iv) for the desired contradiction. Note that the assumption that the domains $C(p)$ are monotonic, in the sense that $pSq \;\Rightarrow\; C(p) \subseteq C(q)$, plays a key role in this argument.

Example: Forcing for frames

$\Box\varphi(c) \to \varphi(c)$ is not valid: Consider any frame in which the atomic sentence $\varphi(c)$ is not true in some $\mathcal{C}(p)$ and there is no q such that pSq. In such a frame $p \Vdash \Box\varphi(c)$ but $p \nVdash \varphi(c)$.

$\forall x\varphi(x) \to \Box\forall x\varphi(x)$ is not valid: Let $\mathcal{C}$ be the frame in which $W = \{p, q\}, S = \{(p, q)\}, C(p) = \{c\}, C(q) = \{c, d\}, \mathcal{C}(p) \Vdash \varphi(c)$ and $\mathcal{C}(q) \Vdash \varphi(c) \wedge \neg\varphi(d)$. Now $p \Vdash \forall x\varphi(x)$ but $p \nVdash \Box\forall x\varphi(x)$ as $q \nVdash \varphi(d)$. It is crucial in this example that the domains $C(p)$ of a frame $\mathcal{C}$ are not assumed to all be the same.

Logical consequence

Definition 2.8: Let Σ be a set of sentences in a modal language $\mathcal{L}$ and φ a single sentence of $\mathcal{L}$. φ is a logical consequence of Σ, $\Sigma \models \varphi$, if φ is forced in every $\mathcal{L}$-frame $\mathcal{C}$ in which every $\psi \in \Sigma$ is forced.

Warning: This notion of logical consequence is not the same as requiring that, in every $\mathcal{L}$-frame $\mathcal{C}$, φ is true (forced) at every world w at which every $\psi \in \Sigma$ is forced (Exercise 11). In particular, the deduction theorem ($\Sigma \models \varphi \;\Rightarrow\; \models \wedge\Sigma \rightarrow \varphi$) fails, as witnessed by Examples 2.7 and 2.9.

Example: Logical consequence

$\forall x\varphi(x) \models \Box\forall x\varphi(x)$: Suppose $\mathcal{C}$ is a frame in which $p \Vdash \forall x\varphi(x)$ for every possible world $p \in W$. If $q \in W$, we claim that $q \Vdash \Box\forall x\varphi(x)$. If not, there would be a $p \in W$ such that qSp and $p \nVdash \forall x\varphi(x)$ contradicting our assumption.

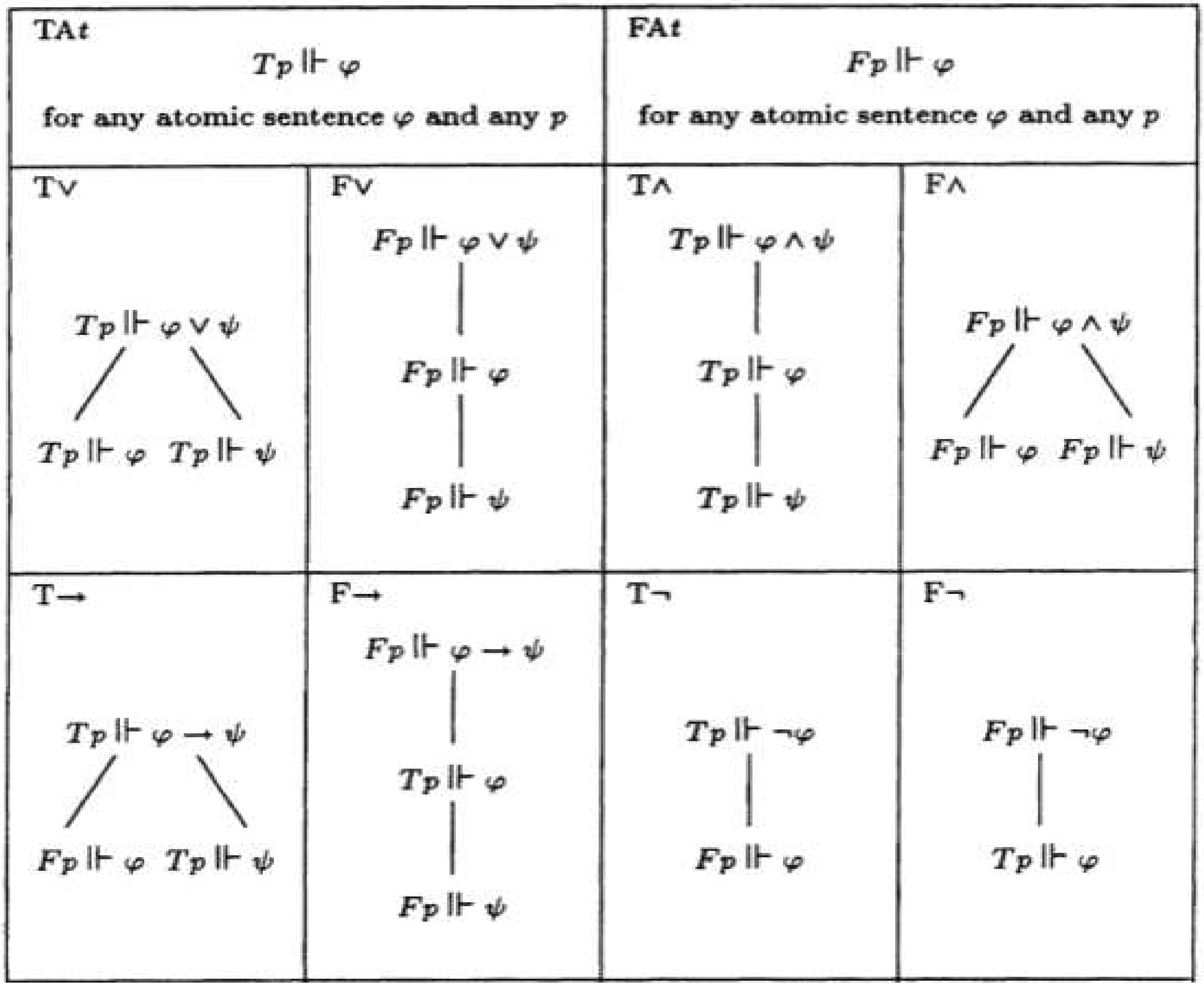
If φ is an atomic unary predicate, $\Box\varphi(c) \not\models \Diamond\varphi(c)$: Consider a frame $\mathcal{C}$ in which $S = \emptyset$ and in which $\mathcal{C}(p) \not\models \varphi(c)$ and so $p \nVdash \varphi(c)$ for every p. In $\mathcal{C}$, every p forces $\Box\varphi(c)$ but none forces $\varphi(c)$ and so none forces $\Diamond\varphi(c)$.

3. Modal Tableaux

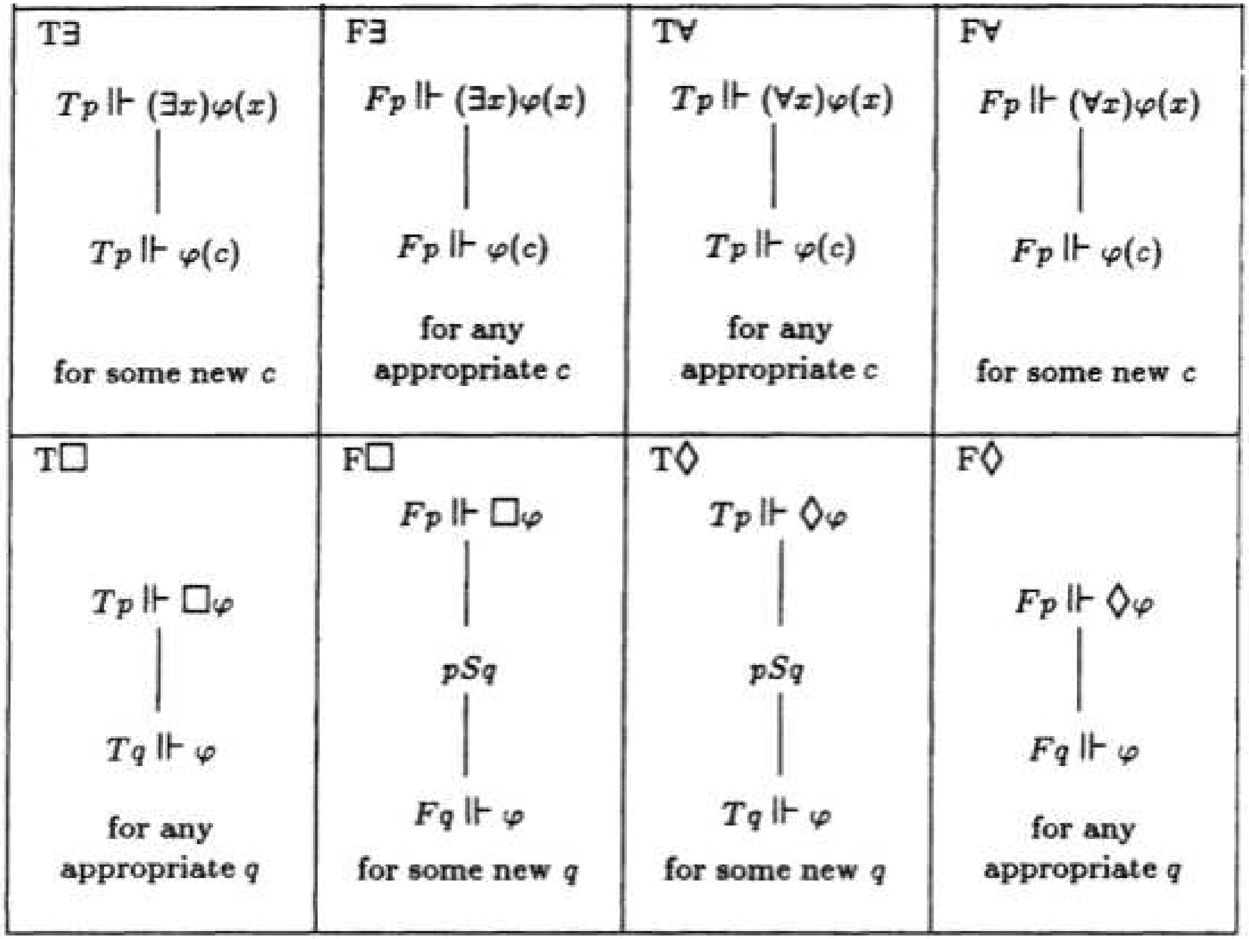
The labels (the entries of the tableau) are either signed forcing assertions (i.e., labels of the form $Tp \Vdash \varphi$ or $Fq \Vdash \varphi$ for φ a sentence of any given appropriate language) or accessibility assertions pSq. We read $Tp \Vdash \varphi$ as p forces φ and $Fp \Vdash \varphi$ as p does not force φ.

Atomic tableaux

We first fixing a modal language $\mathcal{L}$ and an expansion to $\mathcal{L}_C$ given by adding new constant symbols c_i for $i \in \mathcal{N}$.



Atomic tableaux



T∃ $Tp \Vdash (\exists x)\varphi(x)$ $Tp \Vdash \varphi(c)$ for some new c	F∃ $Fp \Vdash (\exists x)\varphi(x)$ $Fp \Vdash \varphi(c)$ for any appropriate c	T∀ $Tp \Vdash (\forall x)\varphi(x)$ $Tp \Vdash \varphi(c)$ for any appropriate c	F∀ $Fp \Vdash (\forall x)\varphi(x)$ $Fp \Vdash \varphi(c)$ for some new c
T□ $Tp \Vdash \Box\varphi$ $Tq \Vdash \varphi$ for any appropriate q	F□ $Fp \Vdash \Box\varphi$ pSq $Fq \Vdash \varphi$ for some new q	T◊ $Tp \Vdash \Diamond\varphi$ pSq $Tq \Vdash \varphi$ for some new q	F◊ $Fp \Vdash \Diamond\varphi$ $Fq \Vdash \varphi$ for any appropriate q

Warning: In $(T\Box)$ and $(F\Diamond)$ we allow for the possibility that there is no appropriate q by admitting $Tp \Vdash \Box\varphi$ and $Fp \Vdash \Diamond\varphi$ as instances of $(T\Box)$ and $(F\Diamond)$, respectively.

Modal tableaux

Definition 3.2: We continue to use our fixed modal language $\mathcal{L}$ and its extension by constants $\mathcal{L}_C$. We also fix a set $\{p_i \mid i \in \mathcal{N}\}$ of potential candidates for the p's and q's in our forcing assertions. A modal tableau (for $\mathcal{L}$) is a binary tree labeled with signed forcing assertions or accessibility assertions; both sorts of labels are called entries of the tableau. The class of modal tableaux (for $\mathcal{L}$) is defined inductively as follows.

(i) Each atomic tableau τ is a tableau. The requirement that c be new in cases $(T\exists)$ and $(F\forall)$ here means that c is one of the constants c_i added on to $\mathcal{L}$ to get $\mathcal{L}_c$ which does not appear in φ. The phrase "any appropriate c" in $(F\exists)$ and $(T\forall)$ means any constant in $\mathcal{L}$ or in φ. The requirement that q be new in $(F\Box)$ and $(T\Diamond)$ here means that q is any of the p_i other than p. The phrase "any appropriate q" in $(T\Box)$ and $(F\Diamond)$ in this case simply means that the tableau is just $Tp \Vdash \Box\varphi$ or $Fp \Vdash \Diamond\varphi$ as there is no appropriate q.

(ii) If τ is a finite tableau, P a path on τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining an atomic tableau with root entry E to τ at the end of the path P, then τ' is also a tableau.

The requirement that c be new in cases $(T\exists)$ and $(F\forall)$ here means that it is one of the c_i (and so not in $\mathcal{L}$) that do not appear in any entry on τ. The phrase "any appropriate c" in $(F\exists)$ and $(T\forall)$ here means any c in $\mathcal{L}$ or appearing in an entry on P of the form $Tq \Vdash \psi$ or $Fq \Vdash \psi$ such that qSp also appears on P.

In $(F\Box)$ and $(T\Diamond)$ the requirement that q be new means that we choose a p_i not appearing in τ as q. The phrase "any appropriate q" in $(T\Box)$ and $(F\Diamond)$ means we can choose any q such that pSq is an entry on P.

(iii) If $\tau_0, \tau_1, ..., \tau_n, ...$ is a sequence of finite tableaux such that, for every $n \geq 0,\ \tau_{n+1}$ is constructed from τ_n by an application of (ii), then $\tau = \cup\tau_n$ is also a tableau.

Tableau proofs

Definition 3.3: Let τ be a modal tableau and P a path in τ.

(i) P is contradictory if, for some forcing assertion $p \Vdash \varphi$, both $Tp \Vdash \varphi$ and $Fp \Vdash \varphi$ appear as entries on P.

(ii) τ is contradictory if every path through τ is contradictory.

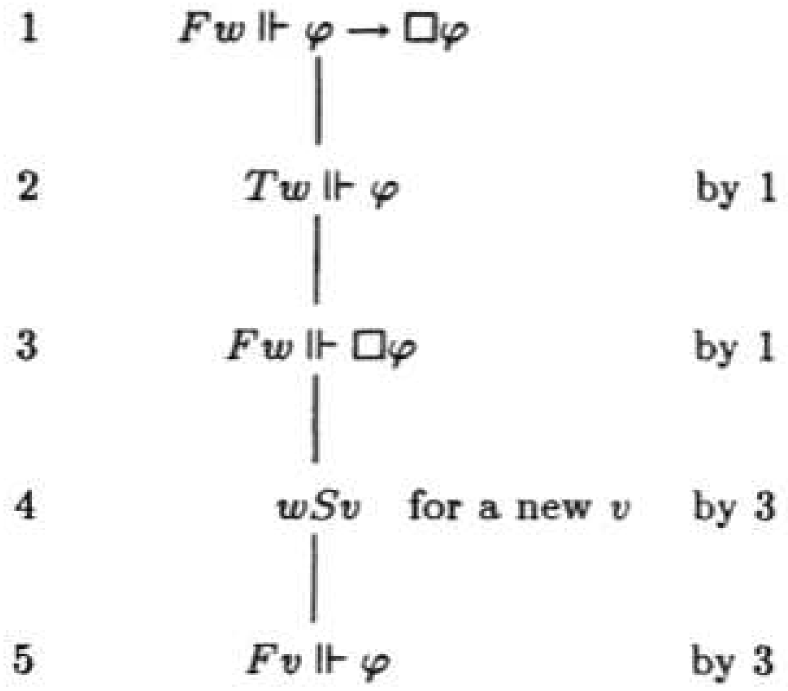
(iii) τ is a proof of φ if τ is a finite contradictory modal tableau with its root node labeled $Fp \Vdash \varphi$ for some p. φ is provable, $\vdash \varphi$, if there is a proof of φ.

Example: tableau proofs

There is a natural correspondence between the tableaux of classical predicate logic and those of modal logic beginning with sentences without modal operators. One goes from the modal tableau to the classical one by replacing signed forcing assertions $Tp \Vdash \varphi$ and $Fp \Vdash \varphi$ by the corresponding signed sentences $T\varphi$ and $F\varphi$, respectively.

Example: tableau proofs

$\varphi \to \Box\varphi$ is not valid.



1 $\quad F w \Vdash \varphi \to \Box\varphi$

2 $\quad T w \Vdash \varphi$ $\quad$ by 1

3 $\quad F w \Vdash \Box\varphi$ $\quad$ by 1

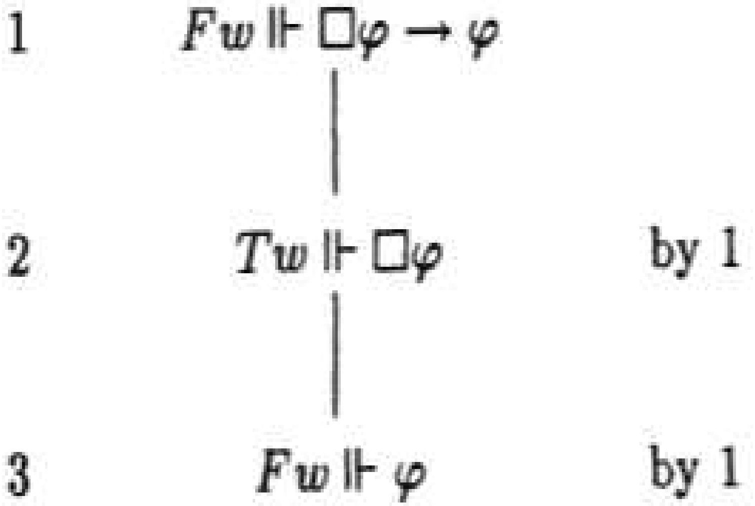
4 $\quad wSv$ $\quad$ for a new v $\quad$ by 3

5 $\quad F v \Vdash \varphi$ $\quad$ by 3

This failed attempt at a proof suggests a frame counterexample $\mathcal{C}$ for which $W = \{w, v\}, S = \{(w, v)\}$ and structures such that φ is true at w but not at v. Such a frame demonstrates that $\varphi \to \Box\varphi$ is not valid as in this frame, w does not force $\varphi \to \Box\varphi$.

Example: tableau proofs

$\Box\varphi \to \varphi$ is not valid.



1 $F w \Vdash \Box\varphi \to \varphi$

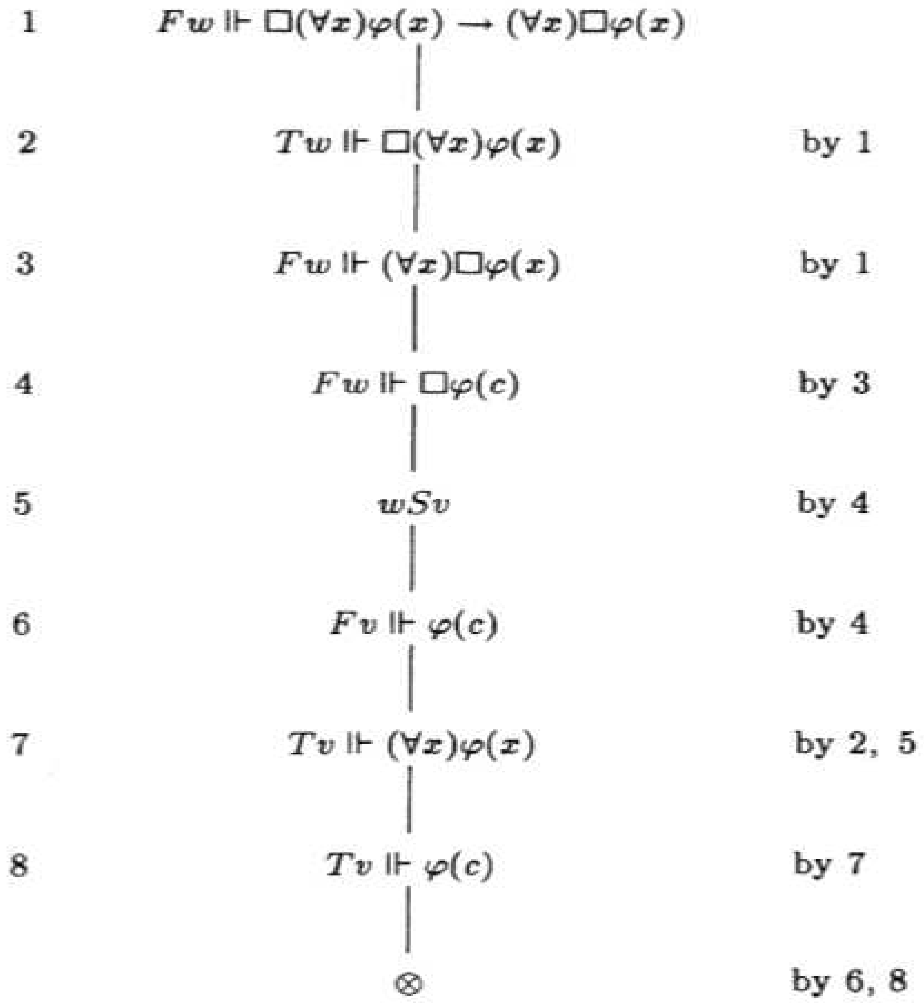
2 $T w \Vdash \Box\varphi$ by 1

3 $F w \Vdash \varphi$ by 1

The frame counterexample suggested here consists of a one world $W = \{w\}$ with empty accessibility relation S and φ false at w. It shows that $\Box\varphi \to \varphi$ is not valid.

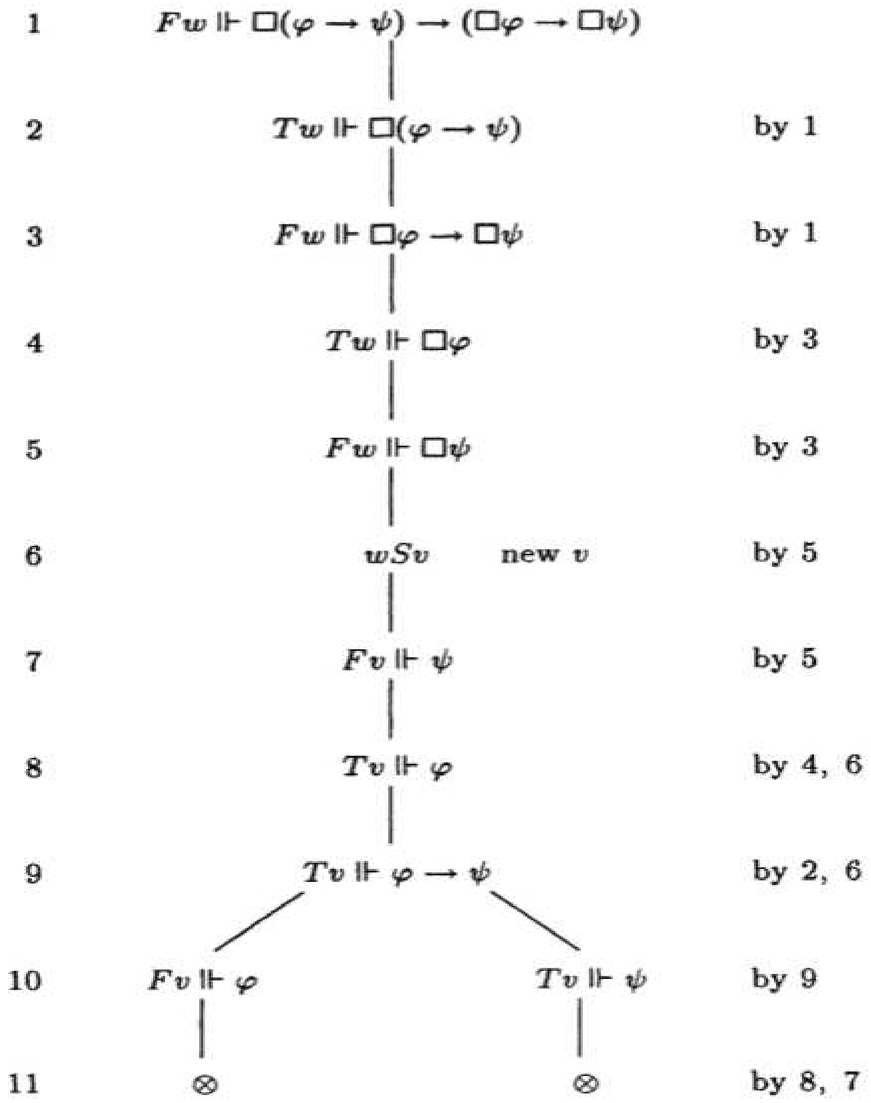
Example: tableau proofs

$\Box(\forall x)\varphi(x) \to (\forall x)\Box\varphi(x)$ is provable.



1	$Fw \Vdash \Box(\forall x)\varphi(x) \to (\forall x)\Box\varphi(x)$	
2	$Tw \Vdash \Box(\forall x)\varphi(x)$	by 1
3	$Fw \Vdash (\forall x)\Box\varphi(x)$	by 1
4	$Fw \Vdash \Box\varphi(c)$	by 3
5	wSv	by 4
6	$Fv \Vdash \varphi(c)$	by 4
7	$Tv \Vdash (\forall x)\varphi(x)$	by 2, 5
8	$Tv \Vdash \varphi(c)$	by 7
	$\otimes$	by 6, 8

Example: tableau proofs



1 $Fw \Vdash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$

2 $Tw \Vdash \Box(\varphi \to \psi)$ by 1

3 $Fw \Vdash \Box\varphi \to \Box\psi$ by 1

4 $Tw \Vdash \Box\varphi$ by 3

5 $Fw \Vdash \Box\psi$ by 3

6 wSv new v by 5

7 $Fv \Vdash \psi$ by 5

8 $Tv \Vdash \varphi$ by 4, 6

9 $Tv \Vdash \varphi \to \psi$ by 2, 6

10 $Fv \Vdash \varphi$ $Tv \Vdash \psi$ by 9

11 $\otimes$ $\otimes$ by 8, 7

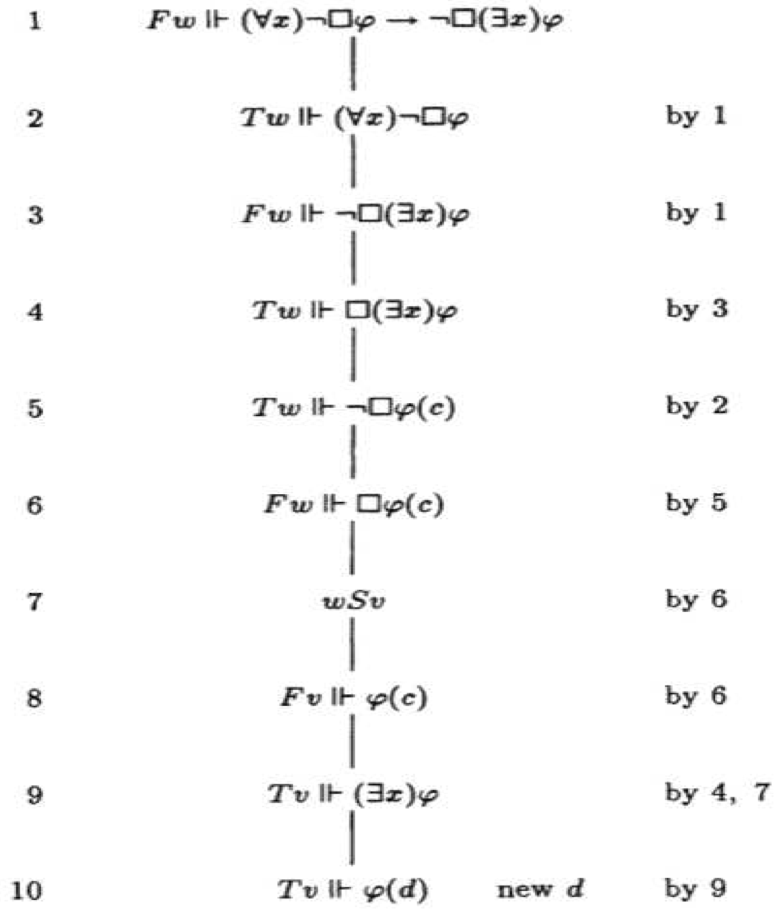
Example: An incorrect proof

1	$Fw \Vdash (\forall x)\Box\varphi(x) \to \Box(\forall x)\varphi(x)$		
2	$Tw \Vdash (\forall x)\Box\varphi(x)$		by 1
3	$Fw \Vdash \Box(\forall x)\varphi(x)$		by 1
4	wSv		by 3
5	$Fv \Vdash (\forall x)\varphi(x)$		by 3
6	$Fv \Vdash \varphi(c)$	new c	by 5
7	$Tw \Vdash \Box\varphi(c)$		by 2
8	$Tv \Vdash \varphi(c)$		by 7
	$\otimes$		

The false step occurs at line 7.

Example: tableau proof

$(\forall x)\neg\Box\varphi \rightarrow \neg\Box(\exists x)\varphi$ is not valid. We have the counterexample with constant domain $C = \{c, d\}$; two worlds w and v with v accessible from w; no atomic sentences true at w and the sentence $\varphi(d)$ true at v.



1	$Fw \Vdash (\forall x)\neg\Box\varphi \rightarrow \neg\Box(\exists x)\varphi$		
2	$Tw \Vdash (\forall x)\neg\Box\varphi$		by 1
3	$Fw \Vdash \neg\Box(\exists x)\varphi$		by 1
4	$Tw \Vdash \Box(\exists x)\varphi$		by 3
5	$Tw \Vdash \neg\Box\varphi(c)$		by 2
6	$Fw \Vdash \Box\varphi(c)$		by 5
7	wSv		by 6
8	$Fv \Vdash \varphi(c)$		by 6
9	$Tv \Vdash (\exists x)\varphi$		by 4, 7
10	$Tv \Vdash \varphi(d)$	new d	by 9

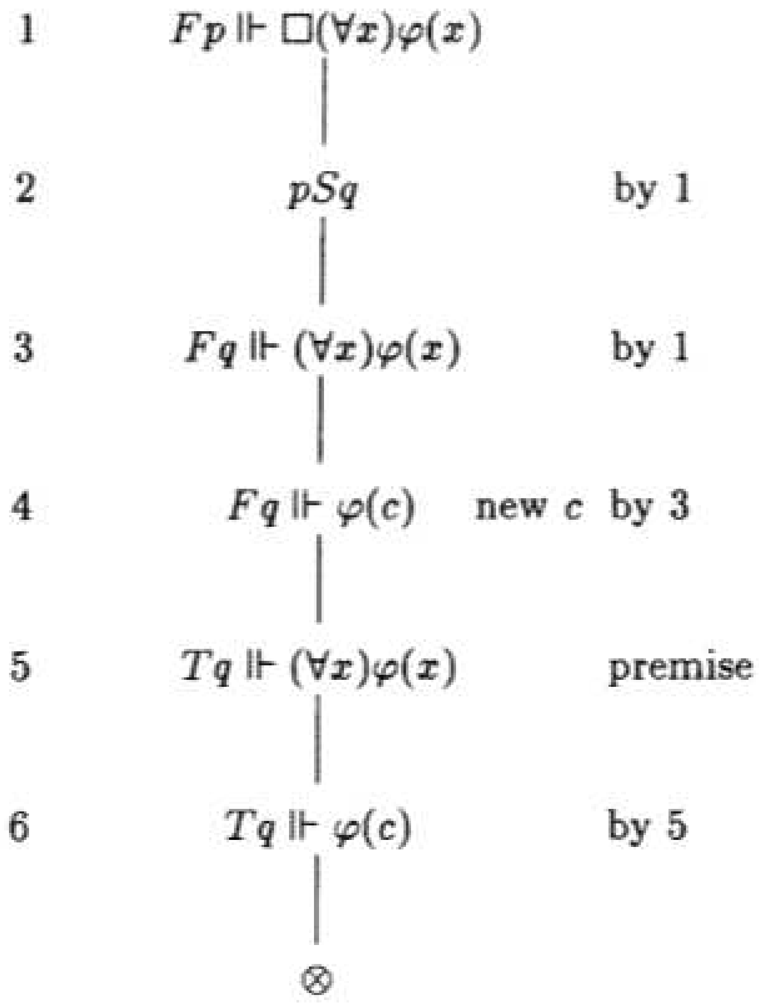
Tableaux from premises

The definition of modal tableaux from Σ, a set of sentences of a modal language called premises, is the same as for simple modal tableaux in Definition 3.2 except that we allow one additional formation rule:

(ii') If τ is a finite tableau from $\Sigma, \varphi \in \Sigma, P$ a path in τ and p a possible world appearing in some signed forcing assertion on P, then appending $Tp \Vdash \varphi$ to the end of P produces a tableau τ' from Σ.

Example: Tableaux from premises

A tableau proof of $\Box\forall\varphi(x)$ from the premise $\forall x\varphi(x)$.



1	$Fp \Vdash \Box(\forall x)\varphi(x)$	
2	pSq	by 1
3	$Fq \Vdash (\forall x)\varphi(x)$	by 1
4	$Fq \Vdash \varphi(c)$	new c by 3
5	$Tq \Vdash (\forall x)\varphi(x)$	premise
6	$Tq \Vdash \varphi(c)$	by 5
	$\otimes$	

Exercises

1. Exercise 7, 9, 11 in page 228
2. Exercise 4, 5, 7, 10 in page 239

4. Soundness and Completeness

Definition 4.1: Suppose $\mathcal{C} = (V, T, \mathcal{C}(p))$ is a frame for a modal language $\mathcal{L}$, τ is a tableau whose root is labeled with a forcing assertion about a sentence φ of $\mathcal{L}$ and P is a path through τ. Let W be the set of p's appearing in forcing assertions on P and let S be the accessibility relation on W determined by the assertions pSq occurring on P . We say that $\mathcal{C}$ agrees with P if there are maps f and g such that

(i) f is a map from W into V that preserves the accessibility relation, i.e., $pSq \;\Rightarrow\; f(p)Tf(q)$. (Note that f is not assumed to be one-one.)

(ii) g sends each constant c occurring in any sentence ψ of a forcing assertion $Tp \Vdash \psi$ or $Fp \Vdash \psi$ on P to a constant in $\mathcal{L}(f(p))$. Moreover, g is the identity on constants of $\mathcal{L}$. We also extend g to be a map on formulas in the obvious way: To get $g(\psi)$ simply replace every constant c in ψ by $g(c)$.

(iii) If $Tp \Vdash \psi$ is on P, then $f(p)$ forces $g(\psi)$ in $\mathcal{C}$ and if $Fp \Vdash \psi$ is on P, then $f(p)$ does not force $g(\psi)$ in $\mathcal{C}$.

Agreement of a frame with a tableau

Lemma 4.5: If f and g are witnesses that a path P of a tableau τ agrees with $\mathcal{C}$ and τ' is gotten from τ by an application of Clause (ii) of Definition 3.2, then there are extensions P', f' and g' of P, f and g, respectively, such that f' and g' are witnesses that the path P' through τ' also agrees with $\mathcal{C}$.

Proof: First note that if τ' is gotten by extending τ somewhere other than at the end of P, then the witnesses for τ work for τ' as well. Thus, we may assume that we form τ' by adjoining one of the atomic tableaux at the end of P in τ. We now consider the sixteen cases given by the atomic tableaux of Definition 3.1.

The cases other than $(T\exists)$, $(F\forall)$, $(F\Box)$ and $(T\Diamond)$ require no extension of f or g. In each of these cases it is obvious from the induction hypothesis and the corresponding case of the definition of forcing (Definition 2.2) (and the monotonicity assumption on the domains $C(p)$ for cases $(F\exists)$ and $(T\forall)$) that one of the extensions of P to a path through τ' satisfies the requirements of the lemma.

We present cases $(T\exists)$ and $(T\Diamond)$ in detail. Cases $(F\forall)$ and $(F\Box)$ are similar. In case ($T\exists$) the entry of P being developed is $Tp \Vdash (\exists x)\varphi(x)$. Then P', the required extension of P, is the only one possible. It is determined by adding $Tp \Vdash \varphi(c)$ to the end of P. By our induction hypothesis, $f(p) \Vdash_{\mathcal{C}} g((\exists)\varphi(x))$. By the definition of forcing an existential sentence (2.2(v)), there is a $c' \in \mathcal{L}(p)$ such that $f(p) \Vdash g(\varphi(c'))$. Fix such a c' and extend g to g' by setting $g'(c) = c'$. It is now obvious that $P', f' = f$ and g' satisfy the requirements of the lemma, i.e., f' and g' witness that P' agrees with $\mathcal{C}$.

Finally, in case ($T\Diamond$) the entry of P being developed is $Tp \Vdash \Diamond\varphi$. The required extension of P to P' is the only possible one. It is determined by adding both pSq and $Tq \Vdash \varphi$ onto the end of P. By our induction hypothesis, $f(p) \Vdash_{\mathcal{C}} g(\Diamond\varphi)$. As $g(\Diamond\varphi) = \Diamond g(\varphi)$, there is, by the definition of forcing for $\Diamond$ (2.2(ix)), a $q' \in V$ such that $f(p)Tq'$ and $q' \Vdash g(\varphi)$. Fix such a q' and extend f to f' by setting $f'(q) = q'$. It is now obvious that P', f' and $g' = g$ satisfy the requirements of the lemma, i.e., f' and g' witness that P' agrees with $\mathcal{C}$. □

Agreement of a frame with an atomic tableau

Lemma 4.4: Suppose $\mathcal{C} = (V, T, \mathcal{C}(p))$ is a frame for a language $\mathcal{L}$ and τ is an atomic tableau whose root is labeled with a forcing assertion about a sentence φ of $\mathcal{L}$. If $q \in V$ and either

(i) $Fr \Vdash \varphi$ is the root of τ and q does not force φ in $\mathcal{C}$, or

(ii) $Tr \Vdash \varphi$ is the root of τ and q does force φ in $\mathcal{C}$,

then there is a path P through τ that agrees with $\mathcal{C}$ with a witness function f (as required in Definition 4.1) that sends r to q.

Proof: We begin by defining $f(r) = q$ and g to be the identity on the constants of $\mathcal{L}_{\mathcal{C}}$. With this choice of f and g, the root itself agrees with $\mathcal{C}$ by the hypothesis of the theorem. This completes the argument for $T\,At$ and $F\,At$. The argument needed for each of the other atomic tableaux is precisely the same as the one for the corresponding case of Lemma 4.5. The inductive argument applied to the rest of the atomic tableau then provides the required extensions. (The degenerate cases of $(T\Box)$ and $(F\Diamond)$ are exceptions but in those cases the conclusion is precisely the hypothesis.) □

Agreement of a frame with a tableau

Theorem 4.2: Suppose $\mathcal{C} = (V, T, \mathcal{C}(p))$ is a frame for a language $\mathcal{L}$ and τ is a tableau whose root is labeled with a forcing assertion about a sentence φ of $\mathcal{L}$. If $q \in V$ and either

(i) $Fr \Vdash \varphi$ is the root of τ and q does not force φ in $\mathcal{C}$, or

(ii) $Tr \Vdash \varphi$ is the root of τ and q does force φ in $\mathcal{C}$,

then there is a path P through τ that agrees with $\mathcal{C}$ with a witness function f (as required in Definition 4.1) that sends r to q.

Proof: Lemma 4.4 establishes the theorem for atomic tableau. Lemma 4.5 then proves the theorem for all finite tableaux by induction. In fact, it proves it for infinite tableaux as well: Suppose $\tau = \cup\tau_n$ is an infinite tableau as defined by Clause (iii) of Definition 3.2. We begin by applying the appropriate case of Lemma 4.4 to τ_0 to get suitable P_0, f_0 and g_0. We then apply Lemma 4.5 to each τ_n in turn to construct P_n, f_n and g_n. The required P, f and g for τ are then simply the unions of the P_n, f_n and g_n, respectively. □

Soundness

Theorem 4.3 ($\vdash \varphi \;\Rightarrow\; \models \varphi$): If there is a (modal) tableau proof of a sentence φ (of modal logic), then φ is (modally) valid.

Proof: A modal tableau proof of φ is a tableau τ with a root of the form $Fr \Vdash \varphi$ in which every path is contradictory. If φ is not valid, then there is a frame $\mathcal{C} = (V, T, \mathcal{C}(p))$ and a $q \in V$ such that q does not force φ in $\mathcal{C}$. Now apply Theorem 4.2 to get a path P through τ and functions f and g with the properties listed in Definition 4.1. As τ is contradictory, there is a p and a sentence ψ such that both $Tp \Vdash \psi$ and $Fp \Vdash \psi$ occur on P. Definition 4.1(iii) then provides an immediate contradiction. □

Reduced entries

Recall that $c_1, ..., c_n, ...$ is a list of all the constants of our expanded language $\mathcal{L}_C$ and $p_1, p_2, ...$ a list of our stock of possible worlds. For convenience we assume that c_1 is in $\mathcal{L}$.

Definition 4.6: Let $\tau = \cup \tau_n$ be a tableau, P a path in τ, E an entry on P and w the i^{th} occurrence of E on P (i.e., the i^{th} node on P labeled with E).

(i) w is reduced on P if one of the following situations hold:

(1) E is not of the form of the root of an atomic tableau of type $(F\exists)$, $(T\forall)$, $(T\Box)$ or $(F\Diamond)$ (of Definition 3.1) and, for some j, τ_{j+1} is gotten from τ_j by an application of Rule (ii) of Definition 3.2 to E and a path on τ_j which is an initial segment of P. [In this case we say that E occurs on P as the root entry of an atomic tableau.]

(2) E is of the form $Fp \Vdash (\exists x)\varphi(x)$ or $Tp \Vdash (\forall x)\varphi(x)$ (cases $(F\exists)$ and $(T\forall)$, respectively); there is an $i+1^{st}$ occurrence of E on P and either

(a) c_i does not occur in any assertion on P about a possible world q such that qSp occurs on P or

(b) $Fp \Vdash \varphi(c_i)$ or $Tp \Vdash \varphi(c_i)$ is an entry on P.

(3) E is of the form $Tp \Vdash \Box\varphi$ or $Fp \Vdash \Diamond\varphi$ (Cases ($T\Box$) and ($F\Diamond$), respectively); there is an $i + 1^{st}$ occurrence of E on P and either

(a) pSp_i is not an entry on P or

(b) $Tp_i \Vdash \varphi$ or $Fp_i \Vdash \varphi$ is an entry on P.

(4) E is of the form pSq.

(ii) τ is finished if every occurrence of every entry on τ is reduced on every noncontradictory path containing it. It is unfinished otherwise.

Unreduced entries

Lemma 4.7: Suppose w is the i^{th} occurrence of an entry E on a path P of a tableau τ and is reduced on P in τ. If τ' is an extension of τ and P' is an extension of P to a path in τ', the only way w could fail to be reduced on P' in τ' is if

(i) E is of the form $Fp \Vdash \exists x\varphi(x)$ $(Tp \Vdash \forall x\varphi(x))$ and c_i does not occur in any assertion on P about a possible world q such that qSp occurs on P but c_i does occur in such an assertion on P' and $Fp \Vdash \varphi(c_i)$ $(Tp \Vdash \varphi(c_i))$ is not an entry on P'; or

(ii) E is of the form $Tp \Vdash \Box\varphi$ $(Fp \Vdash \Diamond p)$ and pSp_i occurs on P' but not on P and $Fp_i \Vdash \varphi$ $(Tp_i \Vdash \varphi)$ does not occur on P'.

Proof: Obvious from the definitions. $\Box$

Complete systematic modal tableau (CSMT)

Definition 4.8: We define the complete systematic modal tableau (the CSMT) starting with a sentence φ by induction as follows.

(i) τ_0 is the atomic tableau with root $Fp_1 \Vdash \varphi$. This atomic tableau is uniquely specified by requiring that in cases $(F\exists)$ and $(T\forall)$ we use the constant c_1, in cases $(T\exists)$ and $(F\forall)$ we use c_i for the least allowable i and in cases $(F\Box)$ and $(T\Diamond)$ we use the least p_i not occurring in the root. (Note that in Cases $(T\Box)$ and $(F\Diamond)$ the tableau consists of just the root entry. It is finished and constitutes our CSMT.)

At stage n we have, by induction, a tableau τ_n. If τ_n is finished, we terminate the construction. Otherwise, we let w be the level-lexicographically least node of τ_n that contains an occurrence of an entry E which is unreduced on some noncontradictory path P of τ_n. We now extend τ_n to a tableau τ_{n+1} by applying one of the following procedures:

(ii) If E is not of the form occurring in the root node of case $(F\exists)$, $(T\forall)$, $(T\Box)$ or $(F\Diamond)$, we adjoin the atomic tableau with root E to the end of every noncontradictory path in τ that contains w. For E of type $(T\exists)$ or $(F\forall)$ we use the least constant c_j not yet appearing in the tableau. If E is of type $(F\Box)$ or $(T\Diamond)$, we choose p_j for q where j is least such that p_j does not appear in the tableau.

(iii) If E is of type $(F\exists)$ or $(T\forall)$ and w is the i^{th} occurrence of E on P, we adjoin the corresponding atomic tableau with c_j as the required c, where j is least such that c_j is appropriate and $Fp \Vdash \varphi(c_j)$ or $Tp \Vdash \varphi(c_j)$, respectively, does not appear as an entry on P. We take c to be c_1 if there is no such c_j.

(iv) If E is of type $(T\Box)$ or $(F\Diamond)$ and w is the i^{th} occurrence of E on P, we adjoin the corresponding atomic tableau with q_j as the required q where j is least such that q_j is appropriate and $Tq_j \Vdash \varphi$ or $Fq_j \Vdash \varphi$, respectively, does not appear as an entry on P. If $Tq_j \Vdash \varphi$ (or $Fq_j \Vdash \varphi$) already appears on P for every appropriate q_j, we simply repeat the assertion $Tq_j \Vdash \varphi$ (or $Fq_j \Vdash \varphi$), where j is least such that q_j is appropriate. (There is at least one appropriate q_j by the assumption that E is not reduced on P.)

The union τ of the sequence of tableaux τ_n is the CSMT starting with φ.

CSMT

Lemma 4.9: If $p_i S p_j$ appears as an entry on a CSMT, then $i < j$.

Proof: New possible worlds p_j and new instances of the accessibility relation appear on a CSMT $\tau = \cup \tau_n$ only when an entry of the form $p_i S p_j$ is put on the tableau by an application of Clause (ii) of Definition 4.8 for Cases $(F\Box)$ or $(T\Diamond)$. Thus they are put on the CSMT in numerical order and so when $p_i S p_j$ is put on the tree, $i < j$ as required. $\Box$

Every CSMT is finished

Proposition 4.10: Every CSMT is finished.

Proof: Consider any entry E and any unreduced occurrence w of E on a noncontradictory path P of the given CSMT τ. (If there is no such w, τ is finished by definition.) Suppose that E makes a forcing assertion about some possible world p_m, w is the i^{th} occurrence of E on P and that there are n nodes of τ that are level-lexicographically less than w. Let k be large enough so that

(i) The occurrence w of E is in τ_k.

(ii) $p_m S p_i$ is on P in τ_k if it is on P at all.

(iii) If any assertion about a possible world p_j (for which $p_j S p_m$ occurs on P) and the constant c_i appear on P, then $p_j S p_m$ and some occurrence of an assertion involving both p_j and c_i occurs on P in τ_k.

Note that by Lemma 4.9 there are only finitely many p_j that are relevant to (iii) and so we can find a k large enough to accommodate them all.

It is clear from the definition of the CSMT that we must reduce w on P by the time we form τ_{k+n+1}. □

Frames from noncontradictory path

Theorem 4.11: Suppose that $\tau = \cup \tau_n$ is a CSMT and P is a noncontradictory path in τ. We define a frame $\mathcal{C} = (W, S, \mathcal{C}(p))$ associated with P as follows:

> W is the set of all p_i appearing in forcing assertions on P. S is the set of all pairs (p_i, p_j) such that $p_i S p_j$ appears on P.
>
> For each $p_i \in W, C(p_i)$ is defined by induction on i as the set consisting of all the constants of $\mathcal{L}$, and all other constants appearing in forcing assertions $Tq \Vdash \psi$ or $Fq \Vdash \psi$ on P such that qSp_i. (Note that by Lemma 4.10, if $p_j S p_i$ appears on P, then $j < i$. Thus $C(p_i)$ is well defined by induction.)
>
> For each $p \in W, \mathcal{C}(p)$ is defined by setting each atomic sentence ψ true in $\mathcal{C}(p)$ if and only if $Tp \Vdash \psi$ occurs on P. (Warning: We are using the convention that every $c \in C(p)$ is named by itself in $\mathcal{L}(p)$.)

If we let f and g be the identity functions on W and on the set of constants appearing on P, respectively, then they are witnesses that $\mathcal{C}$ agrees with P.

Proof: First note that the clauses of the definition of $\mathcal{C}$ are designed to guarantee that $\mathcal{C}$ is a frame for $\mathcal{L}$, according to Definition 2.1. Just remember that every constant c in $\mathcal{L}(p)$ names itself.

We now wish to prove that (f and g witness that) P agrees with $\mathcal{C}$. We use induction on the depth of sentences φ appearing in forcing assertions on P. The key point in the induction is that, by Proposition 4.10, every occurrence of every entry is reduced on P.

(i) Atomic φ: If $Tp \Vdash \varphi$ appears on P, then φ is true in $\mathcal{C}(p)$, and so forced by p. If $Fp \Vdash \varphi$ appears on P, then $Tp \Vdash \varphi$ does not appear on P as P is noncontradictory. As this is the only way that p could come to force φ in $\mathcal{C}$, we can conclude that p does not force φ, as required.

The inductive cases are each handled by the corresponding clauses of Definition 4.6 and the definition of forcing (Definition 2.2) together with the induction hypothesis for the theorem. We consider some representative cases.

(ii) The propositional connectives: Suppose φ is built using a connective, e.g., φ is $(\varphi_1 \vee \varphi_2)$. As τ is finished, we know that if $Tp \Vdash \varphi$ occurs on P, then either $Tp \Vdash \varphi_1$ or $Tp \Vdash \varphi_2$ occurs on P. By the induction hypothesis if, say, $Tp \Vdash \varphi_1$ occurs on P, then p forces φ_1 and so, by the definition of forcing

(Definition 2.2(vii)), p forces φ, as required. Similarly, if $Fp \Vdash \varphi$ occurs on P, then both $Fp \Vdash \varphi_1$ and $Fp \Vdash \varphi_2$ appear on P. Thus, by induction and Definition 2.2(vii), p does not force φ, as required. The other classical propositional connectives are treated similarly.

(iii) Quantifiers: Suppose φ is of the form $(\forall v)\psi(v)$. If w is the i^{th} occurrence of $Tp \Vdash (\forall v)\psi(v)$ on P, then there is an $i+1^{st}$ occurrence of $Tp \Vdash (\forall v)\psi(v)$ on P. Moreover, if $c_i \in C(p)$, then $Tp \Vdash \psi(c_i)$ occurs on P. Thus, if $Tp \Vdash (\forall v)\psi(v)$ appears on P, then $Tp \Vdash \psi(c)$ appears on P for every constant $c \in C(p)$. As the depth of $\psi(c)$ is less than that of $(\forall v)\psi(v)$, p forces $\psi(c)$ for every $c \in C(p)$. Thus, p forces $\forall v \psi(v)$ by Definition 2.2(iv).

If $Fp \Vdash (\forall v)\psi(v)$ occurs on P, then again as τ is finished, $Fp \Vdash \psi(c)$ occurs on P for some c. By induction hypothesis, p does not force $\psi(c)$. Thus, by Definition 2.2(iv), p does not force $\forall v \psi(v)$ as required.

The analysis for the existential quantifier is similar.

(iv) The modal operators: If $Tp \Vdash \Box\varphi$ and pSq appear on P, then $Tq \Vdash \varphi$ appears on P as the tableau is finished. (Note that being finished guarantees that if there is one occurrence of $Tp \Vdash \Box\varphi$, there are infinitely many and so in particular a j^{th} one where $q = p_j$.) Thus, q forces φ by induction and p forces $\Box\varphi$ by Definition 2.2(viii).

If $Fp \Vdash \varphi$ appears on P, then both pSq and $Fq \Vdash \varphi$ appear on P for some q. Thus, q does not force φ by induction and q is a successor of p by definition. So by Definition 2.2(ix), p does not force $\Box\varphi$.

The cases for $\Diamond$ are similar.

□

Completeness

Theorem 4.12 ($\models \varphi \;\Rightarrow\; \vdash \varphi$): If a sentence φ of modal logic is valid (in the frame semantics), then it has a (modal) tableau proof.

Proof: Suppose φ is valid. Consider the CSMT τ starting with root $Fp_1 \Vdash \varphi$. By definition, every contradictory path of τ is finite. Thus, if every path of τ is contradictory, τ is finite by König's lemma. In particular, if τ is not a tableau proof of φ, it has a noncontradictory path P. Theorem 4.9 then provides a frame $\mathcal{C}$ in which p_1 does not force φ. Thus, φ is not valid and we have the desired contradiction. □

Tableaux from premises

Theorem 4.13 ($\Sigma \vdash \varphi \ \Rightarrow\ \Sigma \models \varphi$): If there is a modal tableau proof of φ from a set Σ of sentences, then φ is a logical consequence of Σ.

Proof: The proof of the basic ingredient (Theorem 4.2) of the soundness theorem (Theorem 4.3) shows that if τ is a tableau from Σ and $\mathcal{C}$ is a frame that forces every $\varphi \in \Sigma$ which agrees with the root of τ, then $\mathcal{C}$ agrees with some path P of τ. The only new point is that a tableau can be extended by adding, for any $\varphi \in \Sigma$, the assertion $Tp \Vdash \psi$ to the end of any path mentioning p. The proof of Lemma 4.5 is easily modified to incorporate this difference. As $q \Vdash_{\mathcal{C}} \psi$ for every possible world q of $\mathcal{C}$ (by assumption), the inductive hypothesis of the proof of Lemma 4.5 can be immediately verified in this new case as well. The deduction of the theorem from this result is now the same as that of Theorem 4.3 from Theorem 4.2. □

CSMT from premises

Definition 4.14: We define the complete systematic modal tableau (CSMT) from a set Σ of sentences starting with a sentence φ by induction as follows. τ_0 is the atomic tableaux with root $Fp_1 \Vdash \varphi$. For the inductive step, we modify Definition 4.8 in much the same way that the notion of a CST was modified in Definition II.6.9 to accommodate premises. Let $\Sigma = \{\psi_j \mid j \in \mathcal{N}\}$. At even stages of the construction we proceed as in (i)-(iv) of Definition 4.8 as appropriate. At odd stages $n = 2k + 1$, we adjoin $Tp_i \Vdash \psi_j$ for every $i, j < k$ to every noncontradictory path P in τ_n on which p_i occurs. We do not terminate the construction unless, for every $\psi \in \Sigma$, $Tp \Vdash \psi$ appears on every noncontradictory path P on which p is mentioned.

Completeness of tableaux from premises

Theorem 4.15 ($\Sigma \models \varphi \;\Rightarrow\; \Sigma \vdash \varphi$)**:** If φ is a logical consequence of a set Σ of sentences of modal logic, then there is a modal tableau proof of φ from Σ.

Proof: Suppose φ is a logical consequence of Σ. The argument for Proposition 4.10 still shows that the CSMT from Σ is finished. Its definition also guarantees that, for every $\psi \in \Sigma$, $Tp \Vdash \psi$ appears on any noncontradictory path P on which p is mentioned. The argument for Theorem 4.11 now shows that if the CSMT from Σ with root node $Fp_1 \Vdash \varphi$ is not a tableau proof of φ from Σ, then there is a frame $\mathcal{C}$ in which every $\psi \in \Sigma$ is forced but in which φ is not forced. So we have the desired contradiction. □

Exercise

1. Exercise 5 in page 248

Chapter V. Intuitionistic Logic

1. Intuitionism and Constructivism

As a variety of constructive mathematics, intuitionism is essentially a philosophy of the foundations of mathematics. It is sometimes and rather simplistically characterized by saying that its adherents refuse to use the law of excluded middle (for every sentence A, $A \vee \neg A$ is true) in mathematical reasoning.

Example: Nonconstructive proof

We wish to prove that there are two irrational numbers a and b such that a^b is rational. Let $c = \sqrt{2}^{\sqrt{2}}$. If c is rational, then we may take $a = \sqrt{2} = b$. On the other hand, if c is not rational, then $c^{\sqrt{2}} = 2$ is rational and we may take $a = c$ and $b = \sqrt{2}$. Thus, in either case, we have two irrational numbers a and b such that a^b is rational. This proof depends on the law of the excluded middle in that we assume that either c is rational or it is not. It gives us no clue as to which of the two pairs contains the desired numbers.

Intuitionistic logic

L. E. J. Brouwer: Proponent of an extreme constructivist point of view. He rejected much of early twentieth century mathematics on the grounds that it did not provide acceptable existence proofs.

A formal logic that attempts to capture Brouwer's philosophical position was developed by his student Heyting. This logic is called intuitionistic logic. It is an important attempt at capturing constructive reasoning. In particular, the law of the excluded middle is not valid in intuitionistic logic.

Intuitionistic logic

Theorem 2.20: If $\varphi \vee \psi$ is intuitionistically valid, then either φ or ψ is intuitionistically valid.

Theorem 2.21: If $\exists x \varphi(x)$ is intuitionistically valid, then so is $\varphi(c)$ for some constant c.

2. Frames and Forcing

Our language is the same as that for classical predicate logic in Chapter II except that we make one modification and two restrictions that simplify the technical details in the development of forcing.

Modification: omit the logical connective $\leftrightarrow$ and view $\varphi \leftrightarrow \psi$ as an abbreviation for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

Restrictions: assume throughout this chapter that (i) every language $\mathcal{L}$ has at least one constant symbol but (ii) no function symbols other than constants.

Frames

Definition 2.1: Let $\mathcal{C} = (R, \leq, \{\mathcal{C}(p)\}_{p\in R})$ consist of a partially ordered set $(R, \leq)$ together with an assignment, to each p in R, of a structure $\mathcal{C}(p)$ for $\mathcal{L}$ (in the sense of Definition II.4.1). To simplify the notation, we write $\mathcal{C} = (R, \leq, \mathcal{C}(p))$ instead of the more formally precise version, $\mathcal{C} = (R, \leq, \{\mathcal{C}(p)\}_{p\in R})$. As usual, we let $C(p)$ denote the domain of the structure $\mathcal{C}(p)$. We also let $\mathcal{L}(p)$ denote the extension of $\mathcal{L}$ gotten by adding on a name c_a for each element a of $C(p)$ in the style of the definition of truth in II.4. $A(p)$ denotes the set of atomic formulas of $\mathcal{L}(p)$ true in $\mathcal{C}(p)$. We say that $\mathcal{C}$ is a frame for the language $\mathcal{L}$, or simply an $\mathcal{L}$-frame if, for every p and q in R, $p \leq q$ implies that $C(p) \subseteq C(q)$, the interpretations of the constants in $\mathcal{L}(p) \subseteq \mathcal{L}(q)$ are the same in $\mathcal{C}(p)$ as in $\mathcal{C}(q)$ and $A(p) \subseteq A(q)$.

Often $p \leq q$ is read "q extends p", or "q is a future of p". The elements of R are called forcing conditions, possible worlds or states of knowledge.

Forcing for rames

Definition 2.2: Let $\mathcal{C} = (R, \leq, \{\mathcal{C}(p)\})$ be a frame for a language $\mathcal{L}$, p be in R and φ be a sentence of the language $\varphi(p)$. We give a definition of p forces φ , written $p \Vdash \varphi$ by induction on sentences φ.

(i) For atomic sentences φ, $p \Vdash \varphi \;\Leftrightarrow\; \varphi$ is in $A(p)$.

(ii) $p \Vdash (\varphi \to \psi) \;\Leftrightarrow\;$ for all $q \geq p$, $q \Vdash \varphi$ implies $q \Vdash \psi$.

(iii) $p \Vdash \neg\varphi \;\Leftrightarrow\;$ for all $q \geq p$, q does not force φ.

(iv) $p \Vdash (\forall x)\varphi(x) \;\Leftrightarrow\;$ for every $q \geq p$ and for every constant c in $\mathcal{L}(q)$, $q \Vdash \varphi(c)$.

(v) $p \Vdash (\exists x)\varphi(x) \;\Leftrightarrow\;$ there is a constant c in $\mathcal{L}(p)$ such that $p \Vdash \varphi(c)$.

(vi) $p \Vdash (\varphi \wedge \psi) \;\Leftrightarrow\; p \Vdash \varphi$ and $p \Vdash \psi$.

(vii) $p \Vdash (\varphi \vee \psi) \;\Leftrightarrow\; p \Vdash \varphi$ or $p \Vdash \psi$.

If we need to make the frame explicit, we say that p forces φ in $\mathcal{C}$ and write $p \Vdash_{\mathcal{C}} \varphi$.

Forcing for rames

Clause (ii) describes a sort of permanence of implication in the face of more knowledge.

Clause (iii) says that $\neg\varphi$ is forced if φ cannot be forced by supplying more knowledge than p supplies.

Clause (iv) describes a permanence of forcing universal sentences in the face of any new knowledge beyond that supplied by p.

Definition 2.3: Let φ be a sentence of the language $\mathcal{L}$. We say that φ is forced in the $\mathcal{L}$-frame $\mathcal{C}$ if every p in R forces φ. We say φ is intuitionistically valid if it is forced in every $\mathcal{L}$-frame.

Restriction lemma

Another aspect of the permanence of forcing that says the past does not count in forcing, only the future counts.

Lemma 2.4: Let $\mathcal{C} = (R, \leq, \{\mathcal{C}(p)\}_{p\in R})$ be a frame, let q be in R and let $R_q = \{r \in R \mid r \geq q\}$. Then

$$\mathcal{C}_q = (R_q, \leq, \mathcal{C}(p))$$

is a frame, where $\leq$ and the function $\mathcal{C}(p)$ are restricted to R_q. Moreover, for r in R_q, r forces φ in $\mathcal{C}$ iff r forces φ in $\mathcal{C}_q$.

Degeneracy lemma

Lemma 2.5: Let $\mathcal{C}$ be a frame for a language $\mathcal{L}$ and φ a sentence of $\mathcal{L}$. If p is a maximal element of the partial ordering R associated with $\mathcal{C}$, then φ is classically true in $\mathcal{C}(p)$, i.e., $\mathcal{C}(p) \models \varphi$, if and only if $p \Vdash \varphi$. In particular, if there is only one state of knowledge p in R, then $\mathcal{C}(p) \models \varphi$ if and only if $p \Vdash \varphi$.

Intuitionistic validity

Theorem 2.6: Any intuitionistically valid sentence is classically valid.

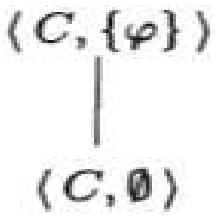
Proof: By the degeneracy lemma (Lemma 2.5), every classical model is a frame model with a one-element partially ordered set in which forcing and classical truth are equivalent. As a sentence is classically valid if true in all classical models, it is valid if forced in every frame. □

Notational conventions

Examples 2.7-2.11 below have orderings that are suborderings of the full binary tree. We can therefore view the associated frames as labeled binary trees with the label of a node p being the structure $\mathcal{C}(p)$, or equivalently, the pair consisting of $C(p)$ and $A(p)$. We thus draw frames as labeled binary trees in our usual style and display the labels in the form $\langle C(p), A(p)\rangle$. $\mathcal{C}(\emptyset)$ is C with all the constants of $\mathcal{L}$ interpreted as c.

Intuitionistic invalidity

Example 2.7: The sentence $\varphi \vee \neg\varphi$ is not intuitionistically valid. Let the frame $\mathcal{C}$ be

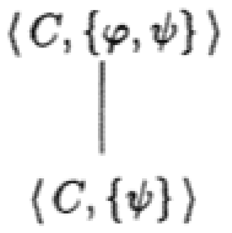


(Thus we have taken C as the domain at both nodes, $\emptyset$ and 0, of the frame.) At the bottom node, no atomic facts are true, i.e., $A(\emptyset)$ is empty. At the upper node 0, we have made the single atomic fact φ true by setting $A(0) = \{\varphi\}$.

Consider now whether or not $\emptyset \Vdash \varphi \vee \neg\varphi$. Certainly $\emptyset$ does not force φ since φ is atomic and not true in $\mathcal{C}(\emptyset)$, i.e., not in $A(\emptyset)$. On the other hand, $0 \Vdash \varphi$ since $\varphi \in A(0)$. Thus $\emptyset$ does not force $\neg\varphi$ since it has an extension 0 forcing φ. So by definition, $\emptyset$ does not force $\varphi \vee \neg\varphi$ and this sentence is not intuitionistically valid.

Intuitionistic invalidity

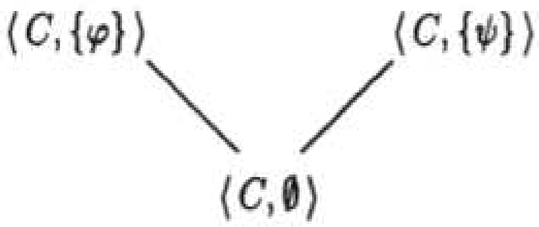
Example 2.8: The sentence $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$ is not intuitionistically valid. Let the frame $\mathcal{C}$ be


$\langle C, \{\varphi, \psi\}\rangle$
$\langle C, \{\psi\}\rangle$

Suppose, for the sake of a contradiction, that $\emptyset \Vdash (\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$. Then $\emptyset \Vdash (\neg\varphi \to \neg\psi)$ would imply $\emptyset \Vdash (\psi \to \varphi)$ by Clause (ii) of the definition of forcing (Definition 2.2). Now by Clause (iii) of the definition, neither $\emptyset$ nor 0 forces $\neg\varphi$ since φ is in $A(0)$ and so forced at 0. Thus we see that $\emptyset$ does in fact force $(\neg\varphi \to \neg\psi)$ by applying Clause (ii) again and the fact that $\emptyset$ and 0 are the only elements $\geq \emptyset$. On the other hand, $\emptyset$ does not force $(\psi \to \varphi)$ because $\emptyset$ forces ψ but not φ and so we have our desired contradiction.

Intuitionistic invalidity

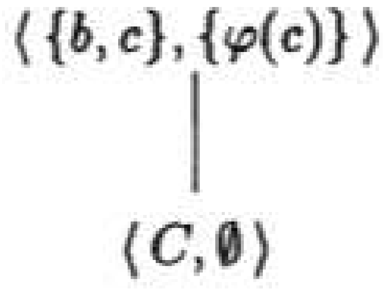
Example 2.9: The sentence $(\varphi \to \psi) \vee (\psi \to \varphi)$ is not intuitionistically valid. Let the frame $\mathcal{C}$ be


$\langle C, \{\varphi\} \rangle$
$\langle C, \{\psi\} \rangle$
$\langle C, \emptyset \rangle$

In this frame, $\emptyset$ forces neither φ nor ψ, 0 forces φ but not ψ and 1 forces ψ but not φ. Since there is a node above $\emptyset$, namely 0, which forces φ but not ψ, 0 does not force $\varphi \to \psi$. Similarly, $\emptyset$ does not force $\psi \to \varphi$. So $\emptyset$ does not force $(\varphi \to \psi) \vee (\psi \to \varphi)$.

Intuitionistic invalidity

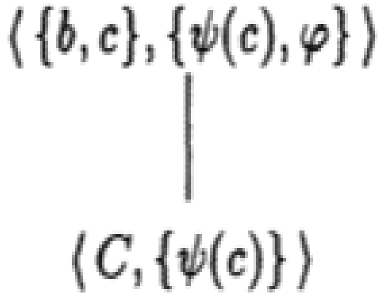
Example 2.10: The sentence $\neg(\forall x)\varphi(x) \rightarrow (\exists x)\neg\varphi(x)$ is not intuitionistically valid. Let the frame $\mathcal{C}$ be



Now by Clause (iv) of Definition 2.2, neither $\emptyset$ nor 0 forces $(\forall x)\varphi(x)$ since $b \in C(0)$ but 0 does not force $\varphi(b)$. Thus $\emptyset \Vdash \neg(\forall x)\varphi(x)$. If $\emptyset \Vdash \neg(\forall x)\varphi(x) \rightarrow (\exists x)\neg\varphi(x)$, as it would were our given sentence valid, then $\emptyset$ would also force $(\exists x)\neg\varphi(x)$. By Clause (v) of the definition this can happen only if there is a $c \in C$ such that $\emptyset \Vdash \neg\varphi(c)$. As c is the only element of C and $0 \Vdash \varphi(c)$, $\emptyset$ does not force $(\exists x)\neg\varphi(x)$.

Intuitionistic invalidity

Example 2.11: The sentence $(\forall x)(\varphi \vee \psi(x)) \rightarrow \varphi \vee (\forall x)\psi(x)$ is not intuitionistically valid. The required frame is



We first claim that $\emptyset \Vdash (\forall x)(\varphi \vee \psi(x))$. As $\emptyset \Vdash \psi(c)$ and $0 \Vdash \varphi$, combining the clauses for disjunction (vii) and universal quantification (iv) we see that $\emptyset \Vdash (\forall x)(\varphi \vee \psi(x))$ as claimed. Suppose now for the sake of a contradiction that $\emptyset \Vdash (\forall x)(\varphi \vee \psi(x)) \rightarrow \varphi \vee (\forall x)\psi(x)$. We would then have that $\emptyset \Vdash \varphi \vee (\forall x)\psi(x)$. However, $\emptyset$ does not force φ and, as 0 does not force $\psi(b)$, $\emptyset$ does not force $(\forall x)\psi(x)$ either. Thus $\emptyset$ does not force the disjunction $\varphi \vee (\forall x)\psi(x)$, so we have the desired contradiction.

Monotonicity lemma

Lemma 2.12: For every sentence φ of $\mathcal{L}$ and every $p, q \in R$, if $p \Vdash \varphi$ and $q \geq p$, then $q \Vdash \varphi$.

Proof: By induction on the logical complexity of φ.

(i) If φ is atomic and $p \Vdash \varphi$, then φ is in $A(p)$. By the definition of a frame, however, $A(p) \subseteq A(q)$, and so φ is in $A(q)$. Thus, by definition, $q \Vdash \varphi$.

(ii) Suppose $p \Vdash \varphi \to \psi$ and $q \geq p$. We show that $q \Vdash \varphi \to \psi$ by showing that if $r \geq q$ and $r \Vdash \varphi$, then $r \Vdash \psi$. Now $r \geq p$ by transitivity and so our assumptions that $p \Vdash \varphi \to \psi$ and $r \Vdash \varphi$ imply that $r \Vdash \psi$, as required.

(iii) Suppose $p \Vdash \neg\varphi$ and $q \geq p$. We show that $q \Vdash \neg p$ by showing that if $r \geq q$, then r does not force φ. Again by transitivity, $r \geq p$. The definition of $p \Vdash \neg\varphi$ then implies that r does not force φ.

(iv) Suppose $p \Vdash (\forall x)\varphi(x)$ and $q \geq p$. We show that $q \Vdash (\forall x)\varphi(x)$ by showing that, for any $r \geq q$ and any $c \in C(r),\ r \Vdash \varphi(c)$. Again, $r \geq p$ by transitivity. The definition of $p \Vdash (\forall x)\varphi(x)$ then implies that for any c in $C(r),\ r \Vdash \varphi(c)$.

(v) Suppose $p \Vdash (\exists x)\varphi(x)$ and $q \geq p$. Then by the definition of forcing there is a c in $C(p)$ such that $p \Vdash \varphi(c)$. By the inductive hypothesis, $q \geq p$ and $p \Vdash \varphi(c)$ imply that $q \Vdash \varphi(c)$. Thus $q \Vdash (\exists x)\varphi(x)$.

(vi) Suppose $p \Vdash (\varphi \wedge \psi)$ and $q \geq p$. Then by the definition of forcing $p \Vdash \varphi$ and $p \Vdash \psi$. By the inductive hypothesis, $q \Vdash \varphi$ and $q \Vdash \psi$. Thus $q \Vdash (\varphi \wedge \psi)$.

(vii) Suppose $p \Vdash (\varphi \vee \psi)$, and $q \geq p$. Then by the definition of forcing either $p \Vdash \varphi$ or $p \Vdash \psi$. By the inductive hypothesis, we get that either $q \Vdash \varphi$ or $q \Vdash \psi$. By the definition of forcing a disjunction, this says that $q \Vdash (\varphi \vee \psi)$.

□

Double negation lemma

Lemma 2.13: $p \Vdash \neg\neg\varphi$ if and only if for any $q \geq p$ there is an $r \geq q$ such that $r \Vdash \varphi$.

Proof: $p \Vdash \neg\neg\varphi$ if and only if every $q \geq p$ fails to force $\neg\varphi$, or equivalently, if and only if every $q \geq p$ has an $r \geq q$ forcing φ. □

Weak quantifier lemma

Lemma 2.14:

(i) $p \Vdash \neg(\exists x)\neg\varphi(x)$ if and only if for all $q \geq p$ and for all $c \in C(q)$, there is an $r \geq q$ such that $r \Vdash \varphi(c)$.

(ii) $p \Vdash \neg(\forall x)\neg\varphi(x)$ if and only if for all $q \geq p$, there exists an $s \geq q$ and a $c \in C(s)$ such that $s \Vdash \varphi(c)$.

Proof: (i) This claim follows immediately from the definition.

(ii) $q \Vdash (\forall x)\neg\varphi(x)$ if and only if for all $r \geq q$ and all $c \in C(r)$ there is no $s \geq r$ such that $s \Vdash \varphi(c)$. Thus q does not force $(\forall x)\neg\varphi(x)$ if and only if there is an $r \geq q$ and a $c \in C(r)$ such that for some $s \geq r,\ s \Vdash \varphi(c)$. So $p \Vdash \neg(\forall x)\neg\varphi(x)$ if and only if for all $q \geq p$, there is an $r \geq q$ and a $c \in C(r)$ such that for some $s \geq r,\ s \Vdash \varphi(c)$. By transitivity $s \geq q$ and c is in $C(s)$ as required in the claim.

□

Intuitionistic validity

Example 2.15: $\varphi \to \neg\neg\varphi$ is intuitionistically valid. To see that any p forces $\varphi \to \neg\neg\varphi$ we assume that $q \geq p$ and $q \Vdash \varphi$. We must show that $q \Vdash \neg\neg\varphi$. By the double negation lemma, it suffices to show that for every $r \geq q$ there is an $s \geq r$ such that $s \Vdash \varphi$. By the monotonicity lemma $r \Vdash \varphi$, and so r is the required s.

Intuitionistic validity

Example 2.16: $\neg(\varphi \wedge \neg\varphi)$ is intuitionistically valid. To show that any p forces $\neg(\varphi \wedge \neg\varphi)$ we need to show that no $q \geq p$ forces $\varphi \wedge \neg\varphi$, or equivalently no $q \geq p$ forces both φ and $\neg\varphi$. Suppose then that q forces both φ and $\neg\varphi$. Now $q \Vdash \neg\varphi$ means no $r \geq q$ forces φ. Since $q \geq q$, we have both q forces φ and q does not force φ for the desired contradiction.

Intuitionistic validity

Example 2.17: $(\exists x)\neg\varphi(x) \rightarrow \neg(\forall x)\varphi(x)$ is intuitionistically valid. To see that any p forces $(\exists x)\neg\varphi(x) \rightarrow \neg(\forall x)\varphi(x)$, we need to show that if $q \geq p$ and $q \Vdash (\exists x)\neg\varphi(x)$, then $q \Vdash \neg(\forall x)\varphi(x)$. Now $q \Vdash (\exists x)\neg\varphi(x)$ says there is a c in $C(q)$ such that $q \Vdash \neg\varphi(c)$. By monotonicity, any $r \geq q$ forces $\neg\varphi(c)$ as well, so no such r forces $(\forall x)\varphi(x)$, thus $q \Vdash \neg(\forall x)\varphi(x)$.

Note that the contrapositive of this example (Example 2.10) is classically but not intuitionistically valid.

Intuitionistic validity

Example 2.18: $\neg(\exists x)\varphi(x) \rightarrow (\forall x)\neg\varphi(x)$ is intuitionistically valid. To see that any p forces $\neg(\exists x)\varphi(x) \rightarrow (\forall x)\neg\varphi(x)$ we have to show that for any $q \geq p$, if $q \Vdash \neg(\exists x)\varphi(x)$, then $q \Vdash (\forall x)\neg\varphi(x)$. Now $q \Vdash \neg(\exists x)\varphi(x)$ says that, for every $r \geq q$ and every c in $C(r),$ r does not force $\varphi(c)$. By transitivity $s \geq r$ implies $s \geq q$. So for every $r \geq q$ and every c in $C(r)$, no $s \geq r$ forces $\varphi(c)$. This says $q \Vdash (\forall x)\neg\varphi(x)$.

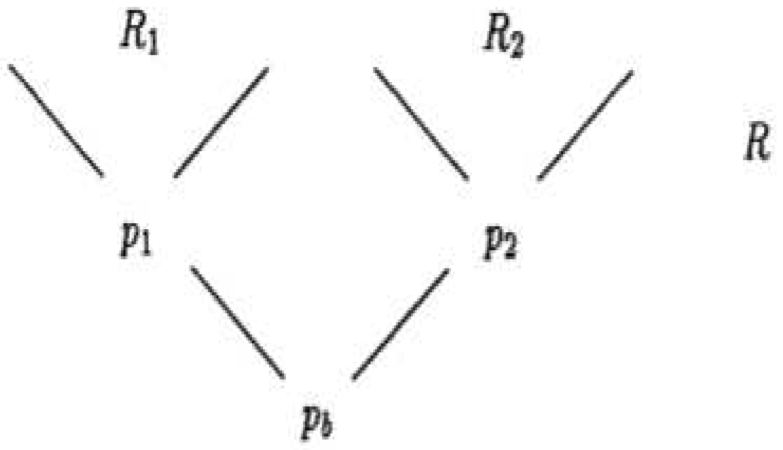
Intuitionistic validity

Example 2.19: If x is not free in φ, then $\varphi \vee (\forall x)\psi(x) \to (\forall x)(\varphi \vee \psi(x))$ is intuitionistically valid. To see that any p forces $\varphi \vee (\forall x)\psi(x) \to (\forall x)(\varphi \vee \psi(x))$ we must show that, for any $q \geq p$, if $q \Vdash \varphi$ or $q \Vdash (\forall x)\psi(x)$, then $q \Vdash (\forall x)(\varphi \vee \psi(x))$. There are two cases. If $q \Vdash \varphi$, then for any $r \geq q$ and any c in $C(r),\ r \Vdash \varphi \vee \psi(c)$, so $q \Vdash (\forall x)(\varphi \vee \psi(x))$. If $q \Vdash (\forall x)\psi(x)$, then for all $r \geq q$ and all c in $C(r),\ r \Vdash \psi(c)$, so $r \Vdash \varphi \vee \psi(c)$. This says that $q \Vdash (\forall x)(\varphi \vee \psi(x))$. Compare this example with Example 2.11.

Disjunction property

Theorem 2.20: If $(\varphi_1 \vee \varphi_2)$ is intuitionistically valid, then one of φ_1, φ_2 is intuitionistically valid.

Proof: We prove the theorem by establishing its contrapositive. So suppose neither φ_1 nor φ_2 is intuitionistically valid. Thus there are, for $i = 1, 2$, frames $\mathcal{C}_i$ and elements p_i of the associated partial orderings R_i such that φ_1 is not forced by p_1 in $\mathcal{C}_1$ and φ_2 is not forced by p_2 in $\mathcal{C}_2$. By the restriction lemma (2.4), we may assume that p_i is the least element of R_i. By Exercise 11 we may assume that no two distinct constants of the language $\mathcal{L}$ are interpreted as the same element in anyone of the structures in either frame $\mathcal{C}_i(p)$. Now simply by relabeling the elements of $\mathcal{C}_i(p)$ (and so of all the other structures in the frames $\mathcal{C}_i$) and R_i we may assume that the interpretation of each constant c of $\mathcal{L}$ is the same in the two structures $\mathcal{C}_i(p_i)$ and that the R_i are disjoint. Let R be the union of R_1, R_2, and $\{p_b\}$, with p_b not in either R_i. Make R into a partial order by ordering R_1 and R_2 as before and putting p_b below p_1 and p_2.



We define a frame $\mathcal{C}$ with this ordering on R by setting $\mathcal{C}(p)$ equal to $\mathcal{C}_i(p)$ for $p \in R_i$ and $\mathcal{C}(p_b)$ equal to the structure defined on the set of the interpretations of the constants of $\mathcal{L}$ in $\mathcal{C}_i(p_i)$ (remember these interpretations are the same for $i = 1, 2$) by setting $A(p_b) = \emptyset$. (Our standing assumption that $\mathcal{L}$ has at least one constant guarantees that this structure is nonempty.) In this frame $\mathcal{C}$, p_1 does not force φ_1 by the restriction lemma (2.4). Thus, p_b does not force φ_1 by the monotonicity lemma (2.12). Similarly, p_b does not force φ_2 as p_2 does not. Thus, p_b does not force $\varphi_1 \vee \varphi_2$; hence $\varphi_1 \vee \varphi_2$ is not intuitionistically valid: contradiction. □

Existence property

Theorem 2.20: If $(\exists x)\varphi(x)$ is an intuitionistically valid sentence of a language $\mathcal{L}$, then for some constant c in $\mathcal{L}$, $\varphi(c)$ is also intuitionistically valid. (Remember that, by convention, $\mathcal{L}$ has at least one constant.)

Proof: Suppose that, for each constant a in $\mathcal{L}$, $\varphi(a)$ is not intuitionistically valid. Then, for each such constant, there is an $\mathcal{L}$-frame $\mathcal{C}_a$ with a partially ordered set R_a containing an element p_a that does not force $\varphi(a)$. As in the previous proof, we may, without loss of generality, assume that p_a is the least element of R_a and all the R_a's are pairwise disjoint. We also assume that the interpretation of some fixed constant c of $\mathcal{L}$ is the same element d in every $\mathcal{C}(p_a)$. We now form a new partial ordering R by taking the union of all R_a and the union of the partial orders and adding on a new bottom element p_b under all the p_a. We next define an $\mathcal{L}$-frame associated with R, as in the previous proof, by letting $C(p_b) = \{d\},\ A(p_b) = \emptyset$ and $\mathcal{C}(p) = \mathcal{C}_a(p)$ for every $p \in R_a$ and every constant a of $\mathcal{L}$. We can now imitate the argument in Theorem 2.20. As we are assuming that $\exists x\varphi(x)$ is intuitionistically valid, we must have $p_b \Vdash_{\mathcal{C}} \exists x\varphi(x)$. Then, by definition, $p_b \Vdash \varphi(a)$ for some constant a in $\mathcal{L}$. Applying first the monotonicity lemma and then the restriction lemma we would have p_a forcing $\varphi(a)$ first in $\mathcal{C}$ and then in $\mathcal{C}_a$; this contradicts our initial hypothesis that p_a and $\mathcal{C}_a$ show that $\varphi(a)$ is not intuitionistically valid. □

Exercises

1. Exercises 5, 6, 11 in page 274

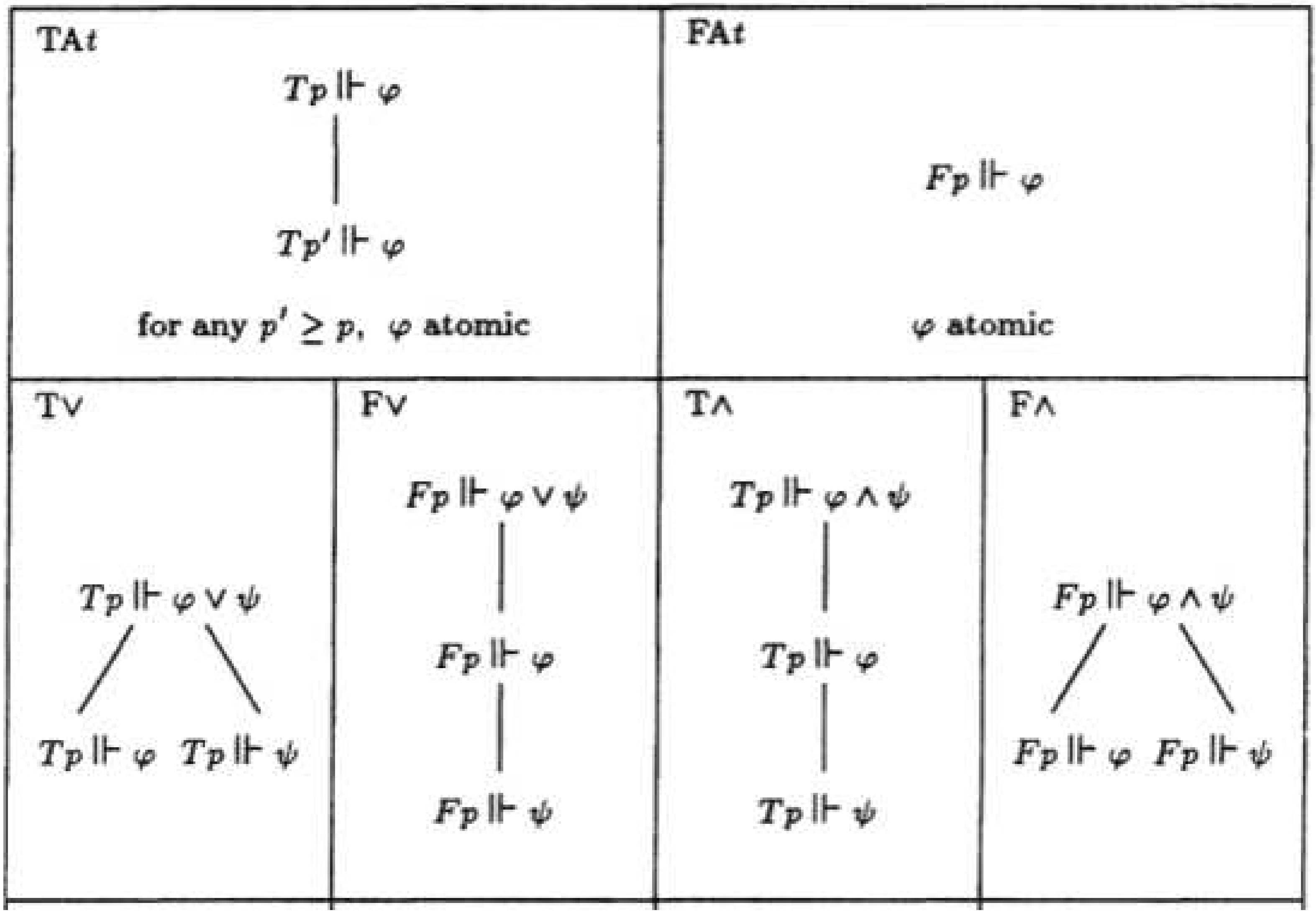
3. Intuitionistic Tableaux

Intuitionistic tableaux and tableau proofs are labeled binary trees. The labels (again called the entries of the tableau) are now signed forcing assertions, i.e., labels of the form $Tp \Vdash \varphi$ or $Fp \Vdash \varphi$ for φ a sentence of any appropriate language. We read $Tp \Vdash \varphi$ as p forces φ and $Fp \Vdash \varphi$ as p does not force φ. If we begin with $Fp \Vdash \varphi$, we either find a frame in which p does not force φ or decide that we have an intuitionistic proof of φ.

Atomic intuitionistic tableaux

Definition 3.1: We begin by fixing a language $\mathcal{L}$ and an expansion $\mathcal{L}_C$ given by adding new constant symbols c_i for $i \in \mathcal{N}$. We list below the atomic intuitionistic tableaux (for the language $\mathcal{L}$). In the tableaux in this list, φ and ψ, if unquantified, are any sentences in the language $\mathcal{L}_C$. If quantified, they are formulas in which only x is free.

Atomic intuitionistic tableaux



TAt

$T p \Vdash \varphi$

$T p' \Vdash \varphi$

for any $p' \geq p$, φ atomic

FAt

$F p \Vdash \varphi$

φ atomic

T∨

$T p \Vdash \varphi \vee \psi$

$T p \Vdash \varphi$ $\quad$ $T p \Vdash \psi$

F∨

$F p \Vdash \varphi \vee \psi$

$F p \Vdash \varphi$

$F p \Vdash \psi$

T∧

$T p \Vdash \varphi \wedge \psi$

$T p \Vdash \varphi$

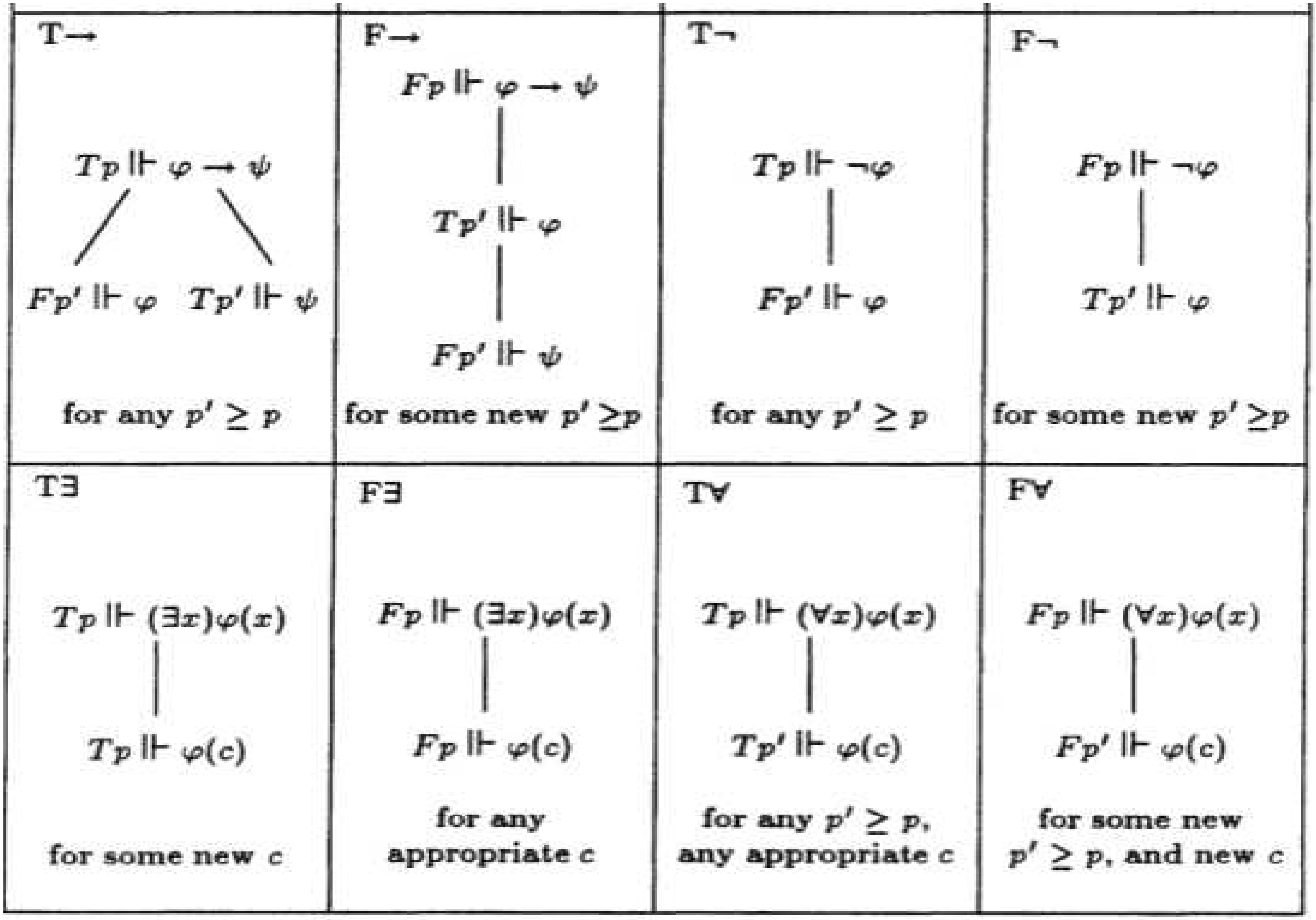
$T p \Vdash \psi$

F∧

$F p \Vdash \varphi \wedge \psi$

$F p \Vdash \varphi$ $\quad$ $F p \Vdash \psi$

Atomic intuitionistic tableaux



T→	F→	T¬	F¬
$Tp \Vdash \varphi \to \psi$ $Fp' \Vdash \varphi$ $\quad$ $Tp' \Vdash \psi$ for any $p' \geq p$	$Fp \Vdash \varphi \to \psi$ $Tp' \Vdash \varphi$ $Fp' \Vdash \psi$ for some new $p' \geq p$	$Tp \Vdash \neg\varphi$ $Fp' \Vdash \varphi$ for any $p' \geq p$	$Fp \Vdash \neg\varphi$ $Tp' \Vdash \varphi$ for some new $p' \geq p$
T∃	F∃	T∀	F∀
$Tp \Vdash (\exists x)\varphi(x)$ $Tp \Vdash \varphi(c)$ for some new c	$Fp \Vdash (\exists x)\varphi(x)$ $Fp \Vdash \varphi(c)$ for any appropriate c	$Tp \Vdash (\forall x)\varphi(x)$ $Tp' \Vdash \varphi(c)$ for any $p' \geq p$, any appropriate c	$Fp \Vdash (\forall x)\varphi(x)$ $Fp' \Vdash \varphi(c)$ for some new $p' \geq p$, and new c

Intuitionistic tableaux

Definition 3.2: We continue to use our fixed language $\mathcal{L}$ and extension by constants $\mathcal{L}_C$. We also fix a set $S = \{p_i \mid i \in \mathcal{N}\}$ of potential candidates for the p's and q's in our forcing assertions. An intuitionistic tableau (for $\mathcal{L}$) is a binary tree labeled with signed forcing assertions which are called the entries of the tableau. The class of all intuitionistic tableaux (for $\mathcal{L}$) is defined by induction. We simultaneously define, for each tableau τ, an ordering $\leq_\tau$ on the elements of S appearing in τ.

(i) Each atomic tableau τ is a tableau. The requirement that c be new in cases $(T\exists)$ and $(F\forall)$ here simply means that c is one of the constants c_i added on to $\mathcal{L}$ to get $\mathcal{L}_C$ which does not appear in φ. The phrase "any c" in $(F\exists)$ and $(T\forall)$ means any constant in $\mathcal{L}$ or in φ. The requirement that p' be new in $(F \rightarrow)$, $(F\neg)$ and $(F\forall)$ here means that p' is any of the p_i other than p. We also declare p' to be larger than p in the associated ordering. The phrase "any $p' \geq p$" in $(T \rightarrow)$, $(T\neg)$, $(T\forall)$ and (TAt) in this case simply means that p' is p. (Of course we always declare $p \leq p$ for every p in every ordering we define.)

(ii) If τ is a finite tableau, P a path on τ, E an entry of τ occurring on P and τ' is obtained from τ by adjoining an atomic tableau with root entry E to τ at the end of the path P, then τ' is also a tableau. The ordering $\leq_{\tau'}$ agrees with $\leq_\tau$ on the p_i appearing in τ. Its behavior on any new element is defined below when we explain the meaning of the restrictions on p' in the atomic tableaux for cases $(F \rightarrow)$, $(F\neg)$ and $(F\forall)$.

The requirement that c be new in cases $(T\exists)$ and $(F\forall)$ here means that it is one of the c_i (and so not in $\mathcal{L}$) that do not appear in any entry on τ. The phrase "any c" in $(F\exists)$ and $(T\forall)$ here means any c in $\mathcal{L}$ or appearing in an entry on P of the form $Tq \Vdash \psi$ or $Fq \Vdash \psi$ with $q \leq_\tau p$.

In $(F \rightarrow)$, $(F\neg)$ and $(F\forall)$ the requirement that $p' \geq p$ be new means that we choose a p_i not appearing in τ as p' and we declare that it is larger than p in $\leq_{\tau'}$. (Of course, we ensure transitivity by declaring that $q \leq_\tau p'$ for every $q \leq_\tau p$.) The phrase "any $p' \geq p$" in $(T \rightarrow)$, $(T\neg)$, $(T\forall)$ and (TAt) means we can choose any p' that appears in an entry on P and has already been declared greater than or equal to p in $\leq_\tau$.

(iii) If $\tau_0, \tau_1, ..., \tau_n, ...$ is a sequence of finite tableaux such that, for every

$n \geq 0$, τ_{n+1} is constructed from τ_n by an application of (ii), then $\tau = \cup\tau_n$ is also a tableau.

As in predicate logic, we insist that the entry E in Clause (ii) formally be repeated when the corresponding atomic tableau is added on to P.

The ordering $\leq_\tau$

Lemma 3.3: For any intuitionistic tableau τ with associated ordering $\leq_\tau$ if $p' \leq_\tau p$, then p and p' both appear on some common path through τ.

Proof: The proof proceeds by an induction on the definition of τ and $\leq_\tau$.
□

Intuitionistic tableau proofs

Definition 3.4: Let τ be a intuitionistic tableau and P a path in τ.

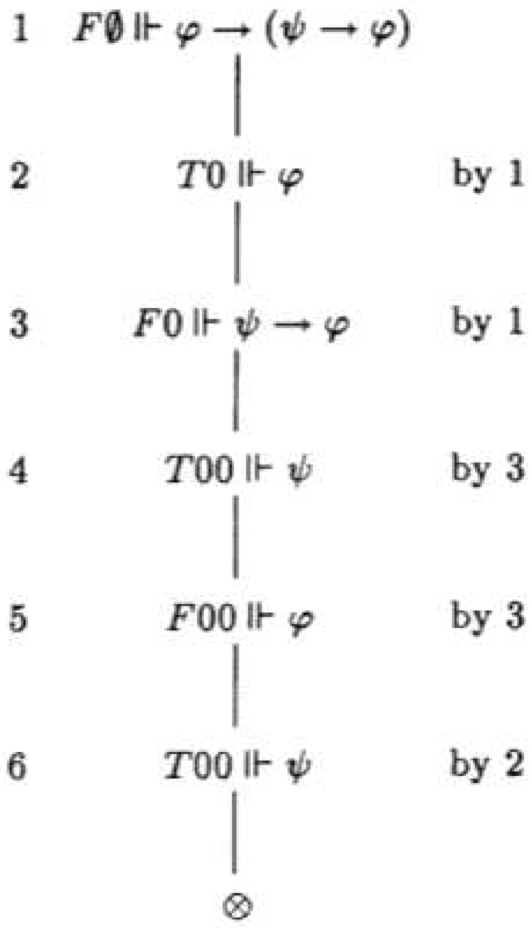
(i) P is contradictory if, for some forcing assertion $p \Vdash \varphi$, both $Tp \Vdash \varphi$ and $Fp \Vdash \varphi$ appear as entries on P.

(ii) τ is contradictory if every path through τ is contradictory.

(iii) τ is an intuitionistic proof of φ if τ is a finite contradictory intuitionistic tableau with its root node labeled $Fp \Vdash \varphi$ for some $p \in S$. φ is intuitionistically provable, $\vdash \varphi$, if there is an intuitionistic proof of φ.

Example: Intuitionistic tableau proofs

Let φ and ψ be any atomic sentences of $\mathcal{L}$.



1 $F\emptyset \Vdash \varphi \to (\psi \to \varphi)$

2 $T0 \Vdash \varphi$ by 1

3 $F0 \Vdash \psi \to \varphi$ by 1

4 $T00 \Vdash \psi$ by 3

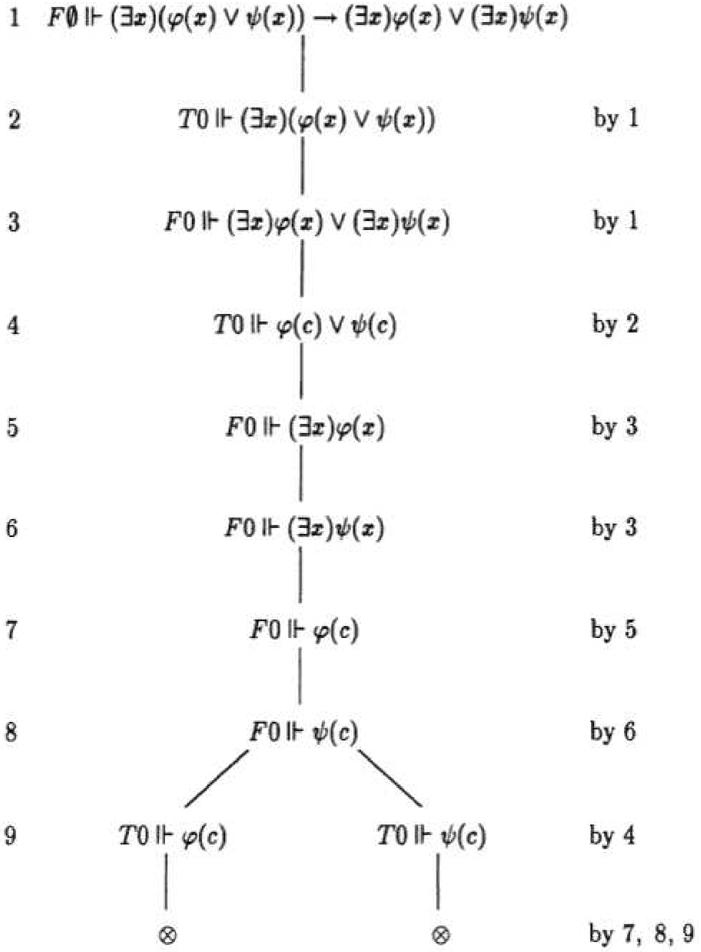
5 $F00 \Vdash \varphi$ by 3

6 $T00 \Vdash \psi$ by 2

$\otimes$

Example: Intuitionistic tableau proofs

Any sentence of $\mathcal{L}$ of the following form is intuitionistically provable: $(\exists x)(\varphi(x) \vee \psi(x)) \rightarrow (\exists x)\varphi(x) \vee (\exists x)\psi(x)$.



1 $F\emptyset \Vdash (\exists x)(\varphi(x) \vee \psi(x)) \rightarrow (\exists x)\varphi(x) \vee (\exists x)\psi(x)$

2 $T0 \Vdash (\exists x)(\varphi(x) \vee \psi(x))$ by 1

3 $F0 \Vdash (\exists x)\varphi(x) \vee (\exists x)\psi(x)$ by 1

4 $T0 \Vdash \varphi(c) \vee \psi(c)$ by 2

5 $F0 \Vdash (\exists x)\varphi(x)$ by 3

6 $F0 \Vdash (\exists x)\psi(x)$ by 3

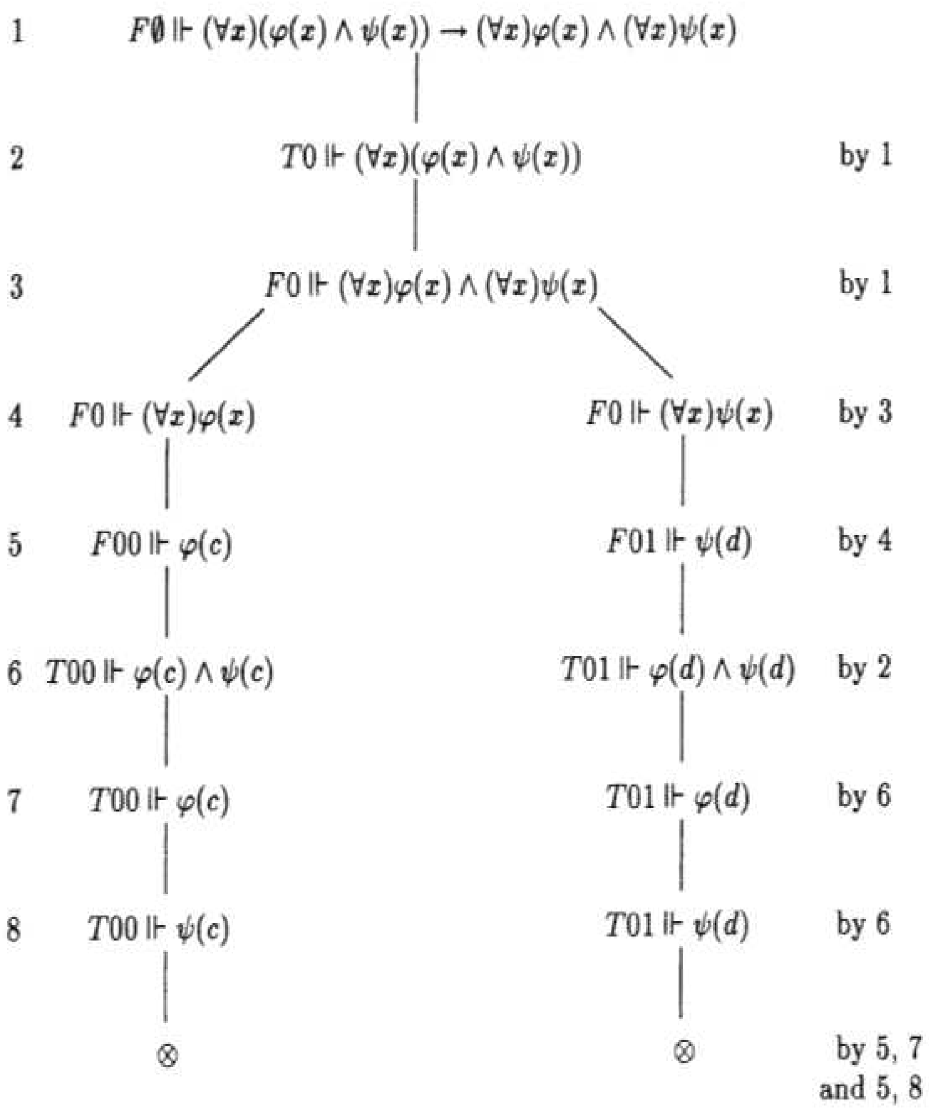
7 $F0 \Vdash \varphi(c)$ by 5

8 $F0 \Vdash \psi(c)$ by 6

9 $T0 \Vdash \varphi(c)$ $T0 \Vdash \psi(c)$ by 4

$\otimes$ $\otimes$ by 7, 8, 9

Example: Intuitionistic tableau proofs



1 $F\emptyset \Vdash (\forall x)(\varphi(x) \wedge \psi(x)) \rightarrow (\forall x)\varphi(x) \wedge (\forall x)\psi(x)$

2 $T0 \Vdash (\forall x)(\varphi(x) \wedge \psi(x))$ by 1

3 $F0 \Vdash (\forall x)\varphi(x) \wedge (\forall x)\psi(x)$ by 1

4 $F0 \Vdash (\forall x)\varphi(x)$ $F0 \Vdash (\forall x)\psi(x)$ by 3

5 $F00 \Vdash \varphi(c)$ $F01 \Vdash \psi(d)$ by 4

6 $T00 \Vdash \varphi(c) \wedge \psi(c)$ $T01 \Vdash \varphi(d) \wedge \psi(d)$ by 2

7 $T00 \Vdash \varphi(c)$ $T01 \Vdash \varphi(d)$ by 6

8 $T00 \Vdash \psi(c)$ $T01 \Vdash \psi(d)$ by 6

$\otimes$ $\otimes$ by 5, 7 and 5, 8

Exercises

1. Exercises 17, 24, 28 in page 284

4. Soundness and Completeness

Definition 4.1: Suppose $\mathcal{C} = (R, \leq_R, \mathcal{C}(p))$ is a frame for a language $\mathcal{L}$, τ is a tableau whose root is labeled with a forcing assertion about a sentence φ of $\mathcal{L}$ and P is a path through τ. Let S be the set of p's appearing in forcing assertions on P and let $\leq_S$ be the ordering on S defined in the construction of τ. We say that $\mathcal{C}$ agrees with P if there are maps f and g such that

(i) f is an order preserving (but not necessarily one-to-one) map from S into R.

(ii) g sends each constant c occurring in any sentence ψ of a forcing assertion $Tp \Vdash \psi$ or $Fp \Vdash \psi$ on P to a constant in $\mathcal{L}(f(p))$. Moreover, g is the identity on constants of $\mathcal{L}$. We also extend g to be a map on formulas in the obvious way: To get $g(\psi)$ simply replace every constant c in ψ by $g(c)$.

(iii) If $Tp \Vdash \psi$ is on P, then $f(p)$ forces $g(\psi)$ in $\mathcal{C}$ and if $Fp \Vdash \psi$ is on P, then $f(p)$ does not force $g(\psi)$ in $\mathcal{C}$.

Soundness

Lemma 4.5: If f and g witness that a path P of a tableau τ agrees with $\mathcal{C}$ and τ' is gotten from τ by an application of Clause (ii) of Definition 3.2, then there are extensions P', f' and g' of P, f and g, respectively, such that f' and g' witness that the path P' through τ' also agrees with $\mathcal{C}$.

Proof: First, note that, if τ' is gotten by extending τ somewhere other than at the end of P, then the witnesses for τ work for τ' as well. Thus, we may assume that we form τ' by adjoining one of the atomic tableaux at the end of P in τ. We now consider the fourteen cases given by the atomic tableaux of Definition 3.1.

Cases $(T\vee)$, $(F\vee)$, $(T\wedge)$, $(F\wedge)$, $(T\rightarrow)$, $(T\neg)$, $(F\exists)$, $(T\forall)$ and (FAt) require no extension of f or g. In each of these cases it is obvious from the induction hypothesis and the corresponding case of the definition of forcing (Definition 2.2) that one of the extensions of P to a path through τ' satisfies the requirements of the lemma. Note that (TAt) also requires the monotonicity assumption on $A(p)$.

The arguments for the remaining cases are all illustrated by that for Case $(F\forall)$. Here the entry of P being developed is $Fp \Vdash (\forall x)\varphi(x)$. The required extension of P to P' is the only possible one. It is determined by adding $Fp' \Vdash \varphi(c)$ to the end of P. By our induction hypothesis, $f(p) \Vdash_{\mathcal{C}} g((\forall x)\varphi(x))$. By the definition of forcing a universal sentence (2.2(iv)), there is a $q' \in R$ and a $c' \in \mathcal{L}(q')$ such that $q' \geq f(p)$ and q' does not force $g(\varphi(c'))$. Fix such q' and c' and extend f and g to f' and g' by setting $f'(p') = q'$ and $g'(c) = c'$. It is now obvious that P', f' and g' satisfy the requirements of the lemma, i.e., f' and g' witness that P' agrees with $\mathcal{C}$.

Other cases are similar. □

Soundness

Lemma 4.4: Suppose $\mathcal{C} = (R, \leq_R, \mathcal{C}(p))$ is a frame for a language $\mathcal{L}$, τ is an atomic tableau whose root is labeled with a forcing assertion about a sentence φ of $\mathcal{L}$. If either

(i) $Fr \Vdash \varphi$ is at the root of τ and $q \in R$ does not force φ in $\mathcal{C}$, or

(ii) $Tr \Vdash \varphi$ is at the root of τ and $q \in R$ does force φ in $\mathcal{C}$,

then there is a path P through τ that agrees with $\mathcal{C}$; moreover, there is a witness function f (as required in Definition 4.1) that sends r to q.

Proof: We begin by defining $f(r) = q$ and g to be the identity on the constants of $\mathcal{L}$. The argument now needed for each atomic tableau is precisely the same as the one for the corresponding case of Lemma 4.5. The point here is that, with this choice of f and g, the root itself agrees with $\mathcal{C}$ by the hypothesis of the theorem. The inductive argument applied to the rest of the atomic tableau then provides the required extensions. □

Soundness

Theorem 4.2: Suppose $\mathcal{C} = (R, \leq_R, \mathcal{C}(p))$ is a frame for a language $\mathcal{L}$, τ is a tableau whose root is labeled with a forcing assertion about a sentence φ of $\mathcal{L}$. If either

(i) $Fr \Vdash \varphi$ is at the root of τ and $q \in R$ does not force φ in $\mathcal{C}$, or

(ii) $Tr \Vdash \varphi$ is at the root of τ and $q \in R$ does force φ in $\mathcal{C}$,

then there is a path P through τ that agrees with $\mathcal{C}$; moreover, there is a witness function f (as required in Definition 4.1) that sends r to q.

Proof: Lemma 4.4 establishes the theorem for atomic tableaux. Lemma 4.5 then proves the theorem for all finite tableaux by induction. In fact, it proves it for infinite tableaux as well. Suppose $\tau = \cup\tau_n$ is an infinite tableau as defined by Clause (iii) of Definition 3.2. We begin by applying the appropriate case of Lemma 4.4 to τ_0 to get suitable P_0, f_0 and g_0. We then apply Lemma 4.5 to each τ_n in turn to construct P_n, f_n and g_n. The required P, f and g for τ are then simply the unions of the P_n, f_n and g_n, respectively. □

Soundness

Theorem 4.3: If there is an intuitionistic tableau proof of a sentence φ, then φ is intuitionistically valid.

Proof: An intuitionistic proof of φ is an intuitionistic tableau τ with a root of the form $Fr \Vdash \varphi$ in which every path is contradictory. If φ is not intuitionistically valid, then there is a frame $\mathcal{C} = (R, \leq_R, \mathcal{C}(p))$ and a $q \in R$ such that q does not force φ in $\mathcal{C}$. Now apply Theorem 4.2 to get a path P through τ and functions f and g with the properties listed in Definition 4.1. As τ is contradictory, there is a p and a sentence ψ such that both $Tp \Vdash \psi$ and $Fp \Vdash \psi$ occur on P. Definition 4.1(iii) then provides an immediate contradiction. □

Complete systematic intuitionistic tableaux

Definition 4.6: Let φ be a sentence of a language $\mathcal{L}$. Let $d_1, d_2, ..., d_n, ...$ be a listing of the set D consisting of all the constants of our standard extension $\mathcal{L}_C$ of $\mathcal{L}$ by new constants. For convenience we assume that d_1 is in $\mathcal{L}$. Let $p_1, p_2, ..., p_n, ...$ be a listing of the set S of all finite sequences of elements of $\mathcal{N}$ that we partially order by extension and let $v_1, v_2, ..., v_k, ...$ be a listing of the set V of all pairs $\langle d_i, p_j \rangle$ consisting of an element from D and one from S. From now on, when we speak of the least element of D, S or V with some property, we mean the first one in the above lists for the appropriate sets.

We define a sequence τ_n of tableaux and what it means for an occurrence w of an entry E of τ_n to be properly developed. The union τ of our sequence of tableaux τ_n is the complete systematic intuitionistic tableau (the CSIT) starting with φ.

τ_0 is the atomic tableau with root $F\emptyset \Vdash \varphi$. If this tableau requires a partial ordering element p' or a constant c we choose the least elements of S or D that make it into a tableau according to Clause (i) of the Definition 3.2.

Suppose we have constructed τ_n. Let m be the least level of τ_n containing an occurrence of an entry that has not been properly developed and let w be the leftmost such occurrence (say of entry E) on level m of τ_n. We form τ_{n+1} by adding an atomic tableau with root E to the end of every noncontradictory path P through τ_n that contains w. To be precise, we list the noncontradictory paths $P_1, P_2, ..., P_k$ of τ_n that contain w. We deal with each P_j in turn by appending an atomic tableau with root E to the end of P_j. Suppose we have reached some P_j on our list. We must now describe the atomic tableau with root E added on to the end of P_j. Cases ($T\vee$), ($F\vee$), ($T\wedge$), ($F\wedge$) and (FAt) of the list of atomic tableaux require no further information to determine the added tableau. Each of the other cases requires fixing some p' and/or some c:

($T \rightarrow$) Let p' be the least q in S that is on P_j, extends p and is such that neither $Fq \Vdash \varphi$ nor $Tq \Vdash \varphi$ occurs on P_j. If there is no such q, let $p' = p$.

($F \rightarrow$) Let $k \in \mathcal{N}$ be least such that $p\hat{\ }k$ has not occurred in the construction so far and let $p' = p\hat{\ }k$. (Note that p' is incomparable with everything that has occurred so far except those that are initial segments of p.)

($T\neg$) Let p' be the least q in S that is on P_j, extends p and is such that $Fq \Vdash \varphi$ does not occur on P_j. If there is no such q, let $p' = p$.

($F\neg$) Proceed as in Case ($F \rightarrow$).

($T\exists$) Let c be the least element of D not occurring in the construction so far.

($F\exists$) Let c be the least element d of D that is either in $\mathcal{L}$ or else occurs in a forcing assertion $Tq \Vdash \psi$ or $Fq \Vdash \psi$ on P_j for any $q \leq p$ such that $Fp \Vdash \varphi(d)$ does not appear on P_j. If there is no such $d \in D$, let $c = d_1$.

($T\forall$) Let $\langle p', c\rangle$ be the least $v = \langle r, d\rangle$ in V such that r appears on P_j, d is either in $\mathcal{L}$ or else occurs in a forcing assertion $Tq \Vdash \psi$ or $Fq \Vdash \psi$ on P_j for any $q \leq p,\ r$ extends p and $Tr \Vdash \varphi(d)$ does not appear on P_j. If there is no such pair, we let $p' = p$ and $c = d_1$.

($F\forall$) Let $k \in \mathcal{N}$ be least such that $p\hat{\ }k$ has not occurred in the construction so far. We set $p' = p\hat{\ }k$ and let c be the least element of D not occurring in the construction so far.

(TAt) Let p' be the least q in S on P_j such that $Tq \Vdash \varphi$ does not appear on P_j. If there is no such q, let $p' = p$.

In all of these cases we say that we have properly developed the occurrence w of entry E.

CSIT

Lemma 4.7: Let $\tau = \cup \tau_n$ be a CSIT as defined above and P a path through τ.

(i) If a sequence $p \in S$ occurs in an assertion at level n of P, then every initial segment q of p occurs on P at some level $m \leq n$ of τ.

(ii) τ is a tableau in accordance with Definition 3.2.

Proof: (i) We proceed by induction through the construction of τ. The only cases in which we actually introduce a new p on P are $(F \rightarrow)$, $(F\neg)$ and $(F\forall)$. In those cases we introduce some sequence $p\hat{\ }k$ for a p already on P.

(ii) The only point to verify is that, in the construction of τ_{n+1} if we add on an atomic tableau with root entry E to the end of some path P_j in τ_n, the p' and c used (if any) satisfy the conditions of Definition 3.2(ii). Otherwise, we obviously are following the prescription for building new tableaux from old ones given in that definition. A simple inspection of the cases shows that we are obeying these restrictions.

□

CSIT

Lemma 4.8: Let $\tau = \cup\tau_n$ be a CSIT as defined above and P a noncontradictory path through τ.

($T\vee$) If $Tp \Vdash \varphi \vee \psi$ appears on P, then either $Tp \Vdash \varphi$ or $Tp \Vdash \psi$ appears on P.

($F\vee$) If $Fp \Vdash \varphi \vee \psi$ appears on P, then both $Fp \Vdash \varphi$ and $Fp \Vdash \psi$ appear on P.

($T\wedge$) If $Tp \Vdash \varphi \wedge \psi$ appears on P, then both $Tp \Vdash \varphi$ and $Tp \Vdash \psi$ appear on P.

($F\wedge$) If $Fp \Vdash \varphi A \psi$ appears on P, then either $Fp \Vdash \varphi$ or $Fp \Vdash \psi$ appears on P .

($T \rightarrow$) If $Tp \Vdash \varphi \rightarrow \psi$ and p' appear on P with $p' \geq p$, then either $Fp' \Vdash \varphi$ or $Tp' \Vdash \psi$ appears on P.

($F \rightarrow$) If $Fp \Vdash \varphi \rightarrow \psi$ appears on P, then for some $p' \geq p$ both $Tp' \Vdash \varphi$ and $Fp' \Vdash \psi$ appear on P.

($T\neg$) If $Tp \Vdash \neg\varphi$ and p' appear on P with $p' \geq p$, then $Fp' \Vdash \varphi$ appears on P.

($F\neg$) If $Fp \Vdash \neg\varphi$ appears on P, then $Tp' \Vdash \varphi$ appears on P for some $p' \geq p$.

($T\exists$) If $Tp \Vdash \exists x\varphi(x)$ appears on P, then $Tp \Vdash \varphi(c)$ appears on P for some c.

($F\exists$) If $Fp \Vdash \exists x\varphi(x)$ appears on P and c is in $\mathcal{L}$ or occurs in a forcing assertion $Tq \Vdash \psi$ or $Fq \Vdash \psi$ on P for any $q \leq p$, then $Fp \Vdash \varphi(c)$ appears on P.

($T\forall$) If $Tp \Vdash \forall x\varphi(x)$ appears on P, c is in $\mathcal{L}$ or occurs in a forcing assertion $Tq \Vdash \psi$ or $Fq \Vdash \psi$ on P for any $q \leq p$ and p' appears on P with $p' \geq p$, then $Tp' \Vdash \varphi(c)$ appears on P.

($F\forall$) If $Fp \Vdash \forall x\varphi(x)$ appears on P, then $Fp' \Vdash \varphi(c)$ appears on P for some c and $p' \geq p$.

(TAt) If p and $Tq \Vdash \varphi$ appear on P for any atomic φ and $q \leq p$, then $Tq \Vdash \varphi$ appears on P.

Proof: First note that every occurrence w in τ of any entry E is properly developed at some stage of the construction. (Consider any w at level n of τ. It is clear that, by the first stage s after all w' at levels $m \leq n$ which are ever properly developed have been so developed, we would have properly developed w.)

Cases ($T\vee$), ($F\vee$), ($T\wedge$), ($F\wedge$), ($F\rightarrow$), ($F\neg$), ($T\exists$) and ($F\forall$) are now almost immediate. Let w be the occurrence of the appropriate signed forcing condition on P. Suppose we properly develop w at stage n of the construction. As w is on

P (which is noncontradictory), one of the P_j that we deal with at stage n is an initial segment of P. We add the appropriate atomic tableau to the end of this P_j as part of our construction of τ_n. Thus P must go through one of the branches of this atomic tableau. This immediately gives the desired conclusion.

Next, note that every entry E occurring on P occurs infinitely often on P. The point here is that each occurrence w of E on P is properly developed. When we properly develop w, we add on another occurrence of E to the end of every noncontradictory path in τ_n that goes through w. Thus we make sure that there there is another occurrence of E on P. As every occurrence of each entry E on P is properly developed, the entry itself is properly developed infinitely often. It is now easy to deal with the remaining cases of the lemma. We choose a few examples.

$(T \rightarrow)$ Suppose for the sake of a contradiction that p' is the least (in our master listing of S) extension of p that occurs on P such that neither $Fp' \Vdash \varphi$ nor $Tp' \Vdash \psi$ occurs on P. Let Q be the finite set of $q \geq p$ that precede p in our master listing of S. For each $q \in Q$, either $Fq \Vdash \varphi$ or $Tq \Vdash \psi$ occurs on P by our choice of p'. Let m be a stage in the construction of τ by which, for each

$q \in Q$, either $Fq \Vdash \varphi$ or $Tq \Vdash \psi$ occurs on the initial segment of P constructed so far. Consider the first stage $n \geq m$ at which we properly develop an occurrence on P of $E = Tp \Vdash \varphi \to \psi$. (As we properly develop an occurrence of E infinitely often in our construction, there must be such a stage.) The definition of the CSIT then guarantees that we add on the atomic tableau with root E using the given p' to the end of some path P_j which is an initial segment of P. As P_j is an initial segment of P, one of the branches through this atomic tableau must also be an initial segment of P as required.

($T\forall$) Suppose, for the sake of a contradiction, that $v = \langle p', e \rangle$ is the least pair (in our master listing of V) satisfying the hypotheses of ($T\forall$) but not the conclusion, i.e., $Tp' \Vdash \varphi(c)$ does not occur on P. Let Q be the finite set of pairs $\langle q, d \rangle$ that precede v and satisfy the hypotheses of $T\forall$. Let m be a stage by which, for each $\langle q, d \rangle \in Q$, we already have an occurrence of $Tq \Vdash \varphi(d)$ on the initial segment of P defined so far. Consider the first stage $n \geq m$ at which we properly develop an occurrence on P of $E = Tq \Vdash \forall x \varphi(x)$. The definition of the CSIT then guarantees that we add on the atomic tableau with root E using the given p' and c to the end of some path P_j which is an initial segment of P. As P_j is an initial segment of P, the unique branch through this atomic tableau must also be an initial segment of P as required.

All the remaining cases, $(T\neg)$, $(F\exists)$ and (TAt), are proved in a similar fashion. □

Frames from noncontradictory paths

Theorem 4.9: Suppose that $\tau = \cup\tau_n$ is a CSIT and P is a noncontradictory path in τ. We define a frame $\mathcal{C} = (R, \leq, \mathcal{C}(p))$ associated with P as follows:

1. R is the set of all sequences in S appearing in forcing assertions on P. The partial ordering on R is the same as that on S - extension.
2. For each $p \in R$, $C(p)$ is the set consisting of the constants of $\mathcal{L}$ and all other constants appearing in forcing assertions $Tq \Vdash \psi$ or $Fq \Vdash \psi$ on P with $q \leq p$.
3. For each $p \in R$, $A(p)$ is the set of all atomic sentences ψ such that $Tq \Vdash \psi$ occurs on P for some $q \leq p$. (Warning: We are using the convention that every $c \in C(p)$ is named by itself in $\mathcal{L}(p)$.)

If we set f and g to be the identity functions on R and on the set of constants appearing on P, respectively, then they are witnesses that $\mathcal{C}$ agrees with P.

Proof: First, note that the clauses of the definition of $\mathcal{C}$ are designed to guarantee that $\mathcal{C}$ is a frame for $\mathcal{L}$ according to Definition 2.1. Just remember that every constant c in $\mathcal{L}(p)$ names itself.

We now wish to prove that P agrees with $\mathcal{C}$; we use induction on the complexity of sentences φ appearing in forcing assertions on P.

Atomic φ: If $Tp \Vdash \varphi$ appears on P, then φ is in $A(p)$ and so forced by p. If $Fp \Vdash \varphi$ appears on P, then we must show that $Tq \Vdash \varphi$ does not appear on P for any $q \leq p$. (This is the only way that p could come to force φ in $\mathcal{C}$.) If there were such an occurrence of $Tq \Vdash \varphi$ on P then, by Lemma 4.8 (TAt), $Tp \Vdash \varphi$ would also occur on P contradicting the assumption that P is noncontradictory.

The inductive cases are each handled by the corresponding clauses of Lemma 4.8 and of the definition of forcing (Definition 2.2) together with the induction assumption for the theorem, i.e. the requirements for $\mathcal{C}$ to agree with P are met for sentences of lower complexity.

As a sample we consider $F\forall : Fp \Vdash \forall x \varphi(x)$ appears on P. By Lemma 4.8 ($F\forall$), $Fp' \Vdash \varphi(c)$ appears on P for some c and $p' \geq p$. The inductive hypothesis then says that p' does not force $\varphi(c)$ in $\mathcal{C}$. The definition of forcing a universal sentence (2.2(v)) then tells us that p does not force $\forall x \varphi(x)$ in $\mathcal{C}$, as required. □

Completeness

Theorem 4.10: If φ is intuitionistically valid, then it has an intuitionistic tableau proof.

Proof: Consider the CSIT τ starting with an intuitionistically valid φ. If τ is not an intuitionistic tableau proof of φ, then it has by definition a noncontradictory path P. Theorem 4.9 then provides a frame $\mathcal{C}$ in which $\emptyset$ does not force φ. Thus φ can not be intuitionistically valid for a contradiction. □

5. Decidability and Undecidability

The CSIT gives us a systematic method for searching for either an intuitionistic proof of a given sentence φ or a frame counterexample. If φ is not intuitionistically valid, the frame counterexample constructed in the proof of the completeness theorem will usually be infinite.

Intuitionistic logic, like classical logic, is undecidable: There is no algorithm that is guaranteed to terminate in a finite time and to tell us if φ is intuitionistically valid (Theorem 5.16). Nonetheless, there are special classes of sentences whose intuitionistic validity can be decided and there are ways of improving our chances of finding both proofs and finite counterexamples in many cases.

Finished tableaux

Definition 5.1: If τ is a tableau and $\leq$ is the ordering defined on the p's and q's appearing in τ , then τ is finished if every noncontradictory path P through τ has the thirteen properties listed in Lemma 4.8.

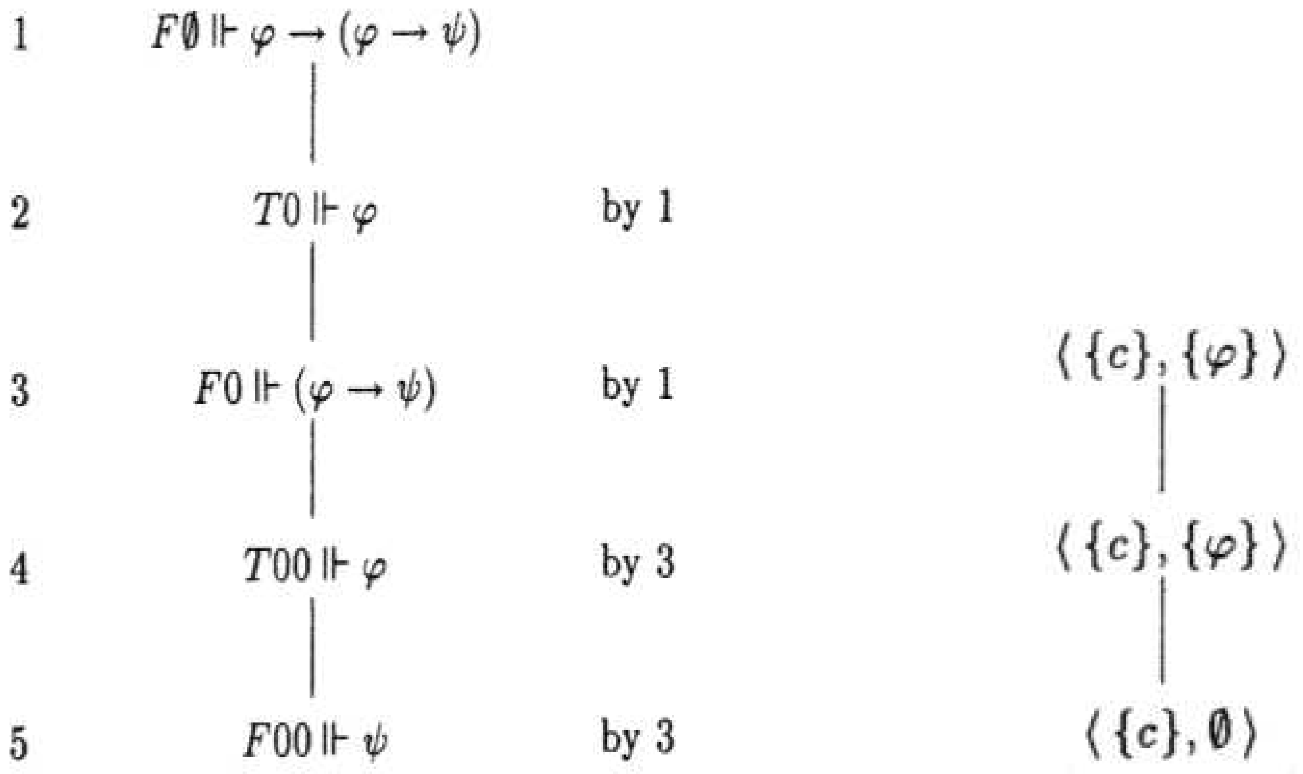
Finished tableaux

Theorem 5.2: If τ is a finished tableau with root $Fp \Vdash \varphi$ and P is a noncontradictory path through τ, then there is a frame $\mathcal{C}$ that agrees with P (and so φ is not intuitionistically valid).

Proof: We proceed exactly as in Theorem 4.9 except that the ordering on the p's occurring in forcing assertions on P is now defined by τ (rather than being given in advance by extension of sequences). The proof of Theorem 4.9 then makes no use of any properties of the CSIT other than those specified in Lemma 4.8. These properties of P are now guaranteed by the definition of τ being finished. □

Example: Counterexample from finished tableau

$\varphi \to (\varphi \to \psi)$ is not intuitionistically valid for atomic sentences φ and ψ.



1 $F\emptyset \Vdash \varphi \to (\varphi \to \psi)$

2 $T0 \Vdash \varphi$ by 1

3 $F0 \Vdash (\varphi \to \psi)$ by 1

4 $T00 \Vdash \varphi$ by 3

5 $F00 \Vdash \psi$ by 3

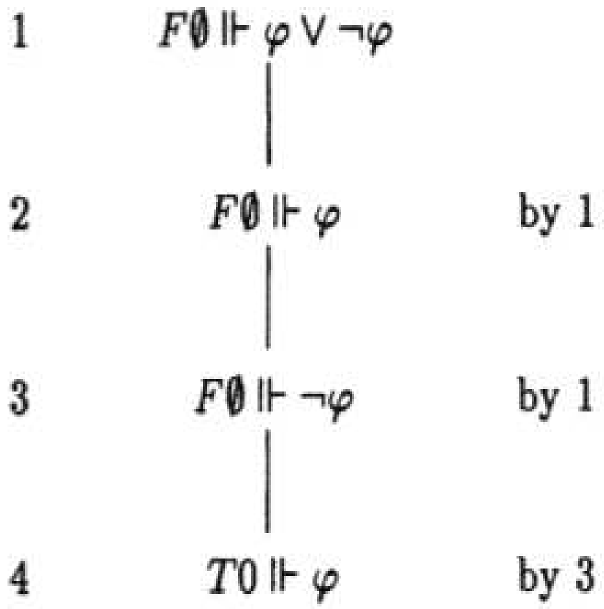
$\langle \{c\}, \{\varphi\} \rangle$

$\langle \{c\}, \{\varphi\} \rangle$

$\langle \{c\}, \emptyset \rangle$

Example: Counterexample from finished tableau

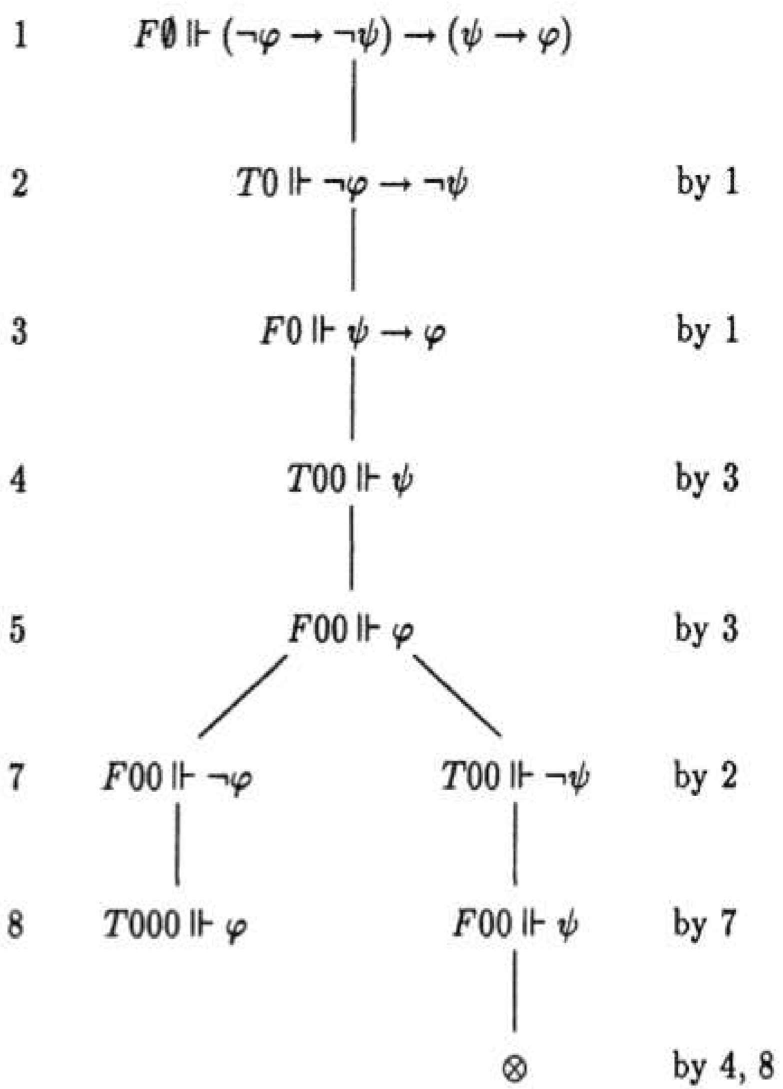
Consider $\varphi \vee \neg\varphi$ and the tableau below.



1	$F\emptyset \Vdash \varphi \vee \neg\varphi$	
2	$F\emptyset \Vdash \varphi$	by 1
3	$F\emptyset \Vdash \neg\varphi$	by 1
4	$T0 \Vdash \varphi$	by 3

Assume that φ and ψ are atomoc sentences. The frame counterexample corresponding to this finished tableau is the same one produced in Example 2.7.

Example: Counterexample from finished tableau



1 $F\emptyset \Vdash (\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$

2 $T0 \Vdash \neg\varphi \to \neg\psi$ by 1

3 $F0 \Vdash \psi \to \varphi$ by 1

4 $T00 \Vdash \psi$ by 3

5 $F00 \Vdash \varphi$ by 3

7 $F00 \Vdash \neg\varphi$ $T00 \Vdash \neg\psi$ by 2

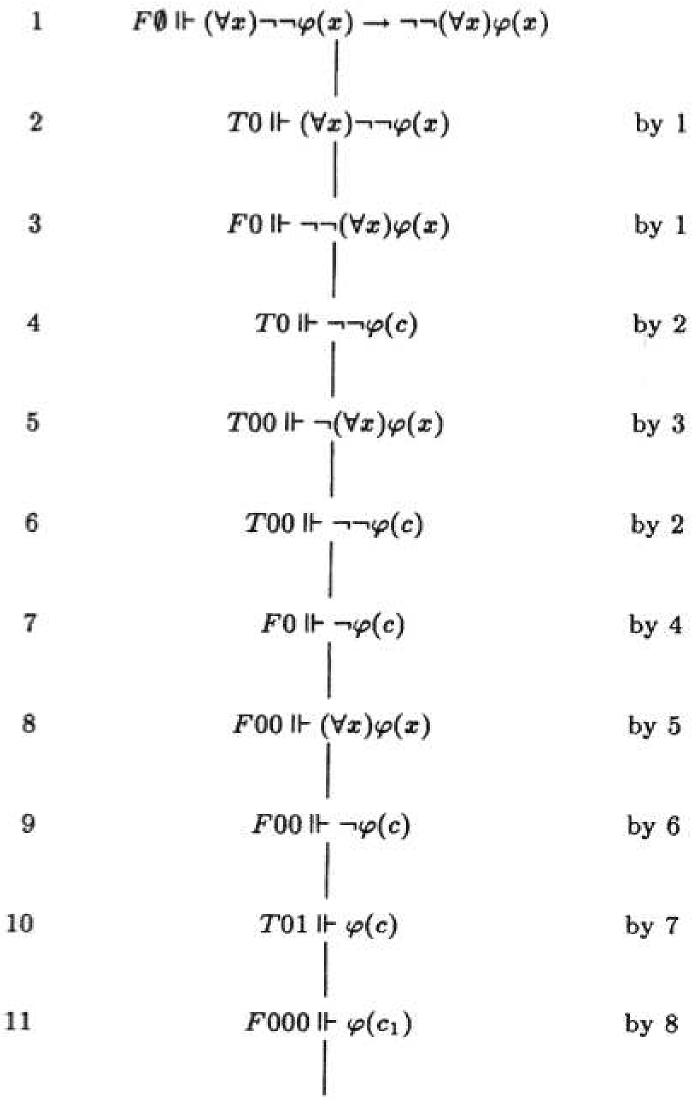
8 $T000 \Vdash \varphi$ $F00 \Vdash \psi$ by 7

$\otimes$ by 4, 8

This is not a finished tableau but has as much information as we need to build the counterexample.

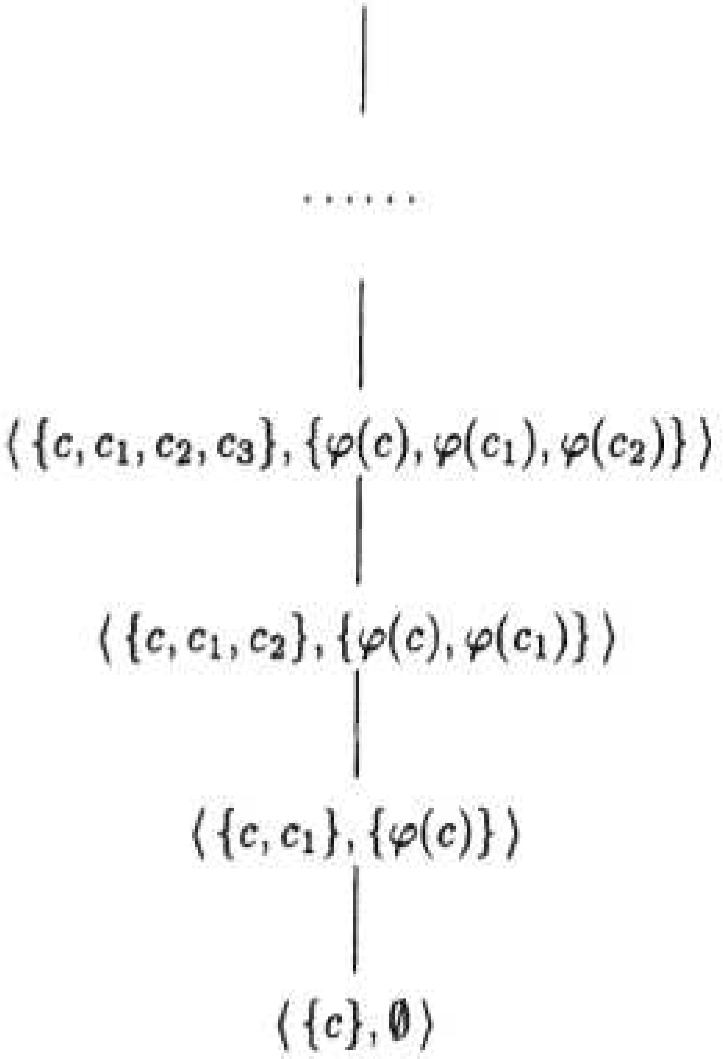
$$A(\emptyset) = A(0) = \emptyset, \qquad A(00) = \{\psi\}, \qquad A(000) = \{\varphi, \psi\}$$

Example: A sentence with no finite counterexample



1	$F\emptyset \Vdash (\forall x)\neg\neg\varphi(x) \rightarrow \neg\neg(\forall x)\varphi(x)$	
2	$T0 \Vdash (\forall x)\neg\neg\varphi(x)$	by 1
3	$F0 \Vdash \neg\neg(\forall x)\varphi(x)$	by 1
4	$T0 \Vdash \neg\neg\varphi(c)$	by 2
5	$T00 \Vdash \neg(\forall x)\varphi(x)$	by 3
6	$T00 \Vdash \neg\neg\varphi(c)$	by 2
7	$F0 \Vdash \neg\varphi(c)$	by 4
8	$F00 \Vdash (\forall x)\varphi(x)$	by 5
9	$F00 \Vdash \neg\varphi(c)$	by 6
10	$T01 \Vdash \varphi(c)$	by 7
11	$F000 \Vdash \varphi(c_1)$	by 8

A counterexample is as follows.


......
$\langle \{c, c_1, c_2, c_3\}, \{\varphi(c), \varphi(c_1), \varphi(c_2)\} \rangle$
$\langle \{c, c_1, c_2\}, \{\varphi(c), \varphi(c_1)\} \rangle$
$\langle \{c, c_1\}, \{\varphi(c)\} \rangle$
$\langle \{c\}, \emptyset \rangle$

Proposition 5.8: $\forall x \neg\neg\varphi(x) \to \neg\neg\forall x\varphi(x)$ is forced by every node p in every frame $\mathcal{C}$ with a finite partial ordering R.

Proof: Let p, $\mathcal{C}$ and R be as in the proposition. To verify that $p \Vdash \forall x \neg\neg\varphi(x) \to \neg\neg\forall x\varphi(x)$ consider any $q \geq p$ such that $q \Vdash \forall x \neg\neg\varphi(x)$. We must show that $q \Vdash \neg\neg\forall x\varphi(x)$. By Lemma 2.13, this is equivalent to the assertion that, for every $r \geq q$, there is an $s \geq r$ such that $s \Vdash \forall x\varphi(x)$. Fix any $r \geq q$. As R is finite, there is a maximal extension s of r in R. Now by monotonicity, $s \Vdash \forall x \neg\neg\varphi(x)$. Thus, for any $c \in C(s)$, $s \Vdash \neg\neg\varphi(c)$. Applying Lemma 2.13 again, as well as the maximality of s, gives us that $s \Vdash \varphi(c)$. Thus (again by the maximality of s), $s \Vdash \forall x\varphi(x)$ as required. □

Finite model property

Theorem 5.9 A quantifier-free sentence is forced in all frames if and only if it is forced in all finite frames.

Proof: Consider any quantifier-free formula φ. We must show that if it is not forced by some $p \in R$ in a frame $\mathcal{C}$, then there is some finite frame $\mathcal{C}'$ in which it is not forced. Let X be the set of all subformulas of φ. For p in R, define a class $[P]$ of elements of R that force the same elements of X as p:

$$[p] = \{q \in R \mid (\forall \psi \in X)(p \Vdash \psi \leftrightarrow q \Vdash \psi)\}$$

Let R' be the set of all such $[p]$ for p in R. Now different classes $[p] \in R'$ correspond to different subsets of X. As X, the set of subformulas of φ, is finite, so is R'. Partially order R' by $[q] \leq [p]$ if every formula in X forced by q is forced by p. (Due to the definition of $[p]$ and $[q]$, this is the same as the requirement that every formula in X forced by some r in $[q]$ is also forced by some s in $[p]$.) Define a finite frame $\mathcal{C}'$ with R' as its partially ordered set by setting $A([p]) = A(p) \cap X$ and $C'([p])$ to be the set of constants appearing in $A([p])$. We claim that for all $p \in R$ and all ψ in X, $[p] \Vdash_{\mathcal{C}'} \psi$ if and only if $p \Vdash_{\mathcal{C}} \psi$. This claim clearly suffices to prove the theorem. We proceed to prove the claim by induction on formulas:

Atomic ψ: $A([p]) = A(p) \cap X$ says that if ψ is an atomic formula in X, then $[p] \Vdash_{\mathcal{C}'} \psi$ if and only if ψ is in $A(p) \cap X$, or equivalently, if and only if $p \Vdash \psi$.

Induction Step: Suppose θ and ψ are in X. Suppose, by induction, that for all $q \in R$, $q \Vdash_{\mathcal{C}} \psi$ if and only if $[q] \Vdash_{\mathcal{C}'} \theta$ and $q \Vdash_{\mathcal{C}} \psi$ if and only if $[q] \Vdash_{\mathcal{C}'} \psi$.

(1) Disjunction: $p \Vdash_{\mathcal{C}} \theta \vee \psi \;\Leftrightarrow\; p \Vdash_{\mathcal{C}} \theta$ or $p \Vdash_{\mathcal{C}} \psi \;\Leftrightarrow\; [p] \Vdash_{\mathcal{C}'} \theta$ or $[p] \Vdash_{\mathcal{C}'} \psi$ (by induction) $\;\Leftrightarrow\; [p] \Vdash_{\mathcal{C}'} \theta \vee \psi$.

(2) Conjunction: $p \Vdash_{\mathcal{C}} \theta \wedge \psi \;\Leftrightarrow\; p \Vdash_{\mathcal{C}} \theta$ and $p \Vdash_{\mathcal{C}} \psi \;\Leftrightarrow\; [p] \Vdash_{\mathcal{C}'} \theta$ and $[p] \Vdash_{\mathcal{C}'} \psi$ (by induction) $\;\Leftrightarrow\; [p] \Vdash_{\mathcal{C}'} \theta \wedge \psi$.

(3) Implication: Suppose $[p] \Vdash_{\mathcal{C}'} \theta \to \psi$. We must show that $p \Vdash_{\mathcal{C}} \theta \to \psi$. If $q \geq p$ and $q \Vdash_{\mathcal{C}} \theta$, then by induction $[q] \Vdash_{\mathcal{C}'} \theta$, so by our assumption and the fact that $[q] \geq [p]$ follows from $q \geq p$, $[q] \Vdash_{\mathcal{C}'} \psi$. The induction hypothesis then says that $q \Vdash_{\mathcal{C}} \psi$ as required. Conversely, suppose $p \Vdash_{\mathcal{C}} \theta \to \psi$ and $\theta \to \psi$ is in X. We must prove that $[p] \Vdash_{\mathcal{C}'} \theta \to \psi$, that is, if $[q] \geq [p]$ and $[q] \Vdash_{\mathcal{C}'} \theta$, then $[q] \Vdash_{\mathcal{C}'} \psi$. Now as $\theta \to \psi$ is in X and $p \Vdash_{\mathcal{C}} \theta \to \psi$ by assumption, $[q] \geq [p]$ implies that $q \Vdash_{\mathcal{C}} \theta \to \psi$. By our assumption that $[q] \Vdash_{\mathcal{C}'} \theta$ and the induction hypothesis, $q \Vdash_{\mathcal{C}} \theta$. Thus $q \Vdash_{\mathcal{C}} \psi$ and again by

induction, $[q] \Vdash_{\mathcal{C}'} \psi$, as required.

(4) Negation is similar to implication.

$\square$

Validity of quantifier-free sentences

Theorem 5.10: The intuitionistic validity of any quantifier-free sentence can be effectively decided.

Proof: We know by the properties of the CSIT expressed in the completeness theorem (Theorems 4.9 and 4.10) that if a given sentence φ is intuitionistically valid then the CSIT will give a (necessarily) finite tableau proof of φ. On the other hand, if φ is not valid then there is, by the finite model property (5.9), a finite frame counterexample. We can thus simultaneously search for a finite frame counterexample to φ and develop the CSIT for φ. We must eventually find either a finite counterexample to φ or an intuitionistic tableau proof that φ is intuitionistically valid. □

Gödel formula

If A is an atomic formula, then $\neg\neg A$ is a Gödel formula. If φ and ψ are Gödel formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$ and $\forall x\varphi$.

Gödel formula

Lemma 5.12: If φ is a Gödel sentence and p is a forcing condition in a frame $\mathcal{C}$, then either $p \Vdash \varphi$ or $(\exists q \geq p)(q \Vdash \neg\varphi)$. In particular, if p does not force φ, then one of the following cases holds:

(1) If $\varphi = \neg\psi$, then $\exists q \geq p(q \Vdash \psi)$.

(2) If $\varphi = \psi \wedge \theta$, then $\exists q \geq p(q \Vdash \neg\psi$ or $q \Vdash \neg\theta)$.

(3) If $\varphi = \forall x\psi$, then $(\exists q \geq p)(\exists c \in C(q))(q \Vdash \neg\psi(c))$.

Proof: We proceed by induction on φ. The base case is that φ is $\neg\neg A$ for some atomic sentence A. In this case, if p does not force φ, there is, by the definition of forcing a negation (Definition 2.2 (iii)), a $q \geq p$ that forces $\neg A$ as required in (1). Note that, in general, if $\varphi = \neg\psi$ and $q \Vdash \psi$, then $q \Vdash \neg\neg\psi$ (i.e., $q \Vdash \neg\varphi$) by the intuitionistic validity of $\psi \rightarrow \neg.\neg\psi$ (Example 2.15).

If φ is $\neg\psi$ and p does not force $\neg\psi$, then, as in the base case, there is a $q \geq p$ that forces ψ and so $\neg\neg\psi$.

If φ is $\psi \wedge \theta$ and p does not force φ, then either p does not force ψ or p does not force θ. Thus by induction there is a $q \geq p$ that forces $\neg\psi$ or $\neg\theta$. (Again, note that by the basic facts about forcing this implies that $q \Vdash \neg(\psi \wedge \theta)$.)

If φ is $\forall x \psi(x)$ and p does not force φ, then, by the definition of forcing, there is an $r \geq p$ and a $c \in C(r)$ such that r does not force $\psi(c)$. The induction hypothesis then tells us that there is a $q \geq r$ such that $q \Vdash \neg\psi(c)$, as required. Of course any such q forces $\neg\varphi$ as well. □

Generic sequence

Definition 5.13: A sequence $\langle p_i \rangle$ of forcing conditions in a frame $\mathcal{C}$ is a generic sequence extending p if $p_0 = p$ and the following conditions hold:

(i) For every i, $p_i \leq p_{i+1}$.

(ii) For every atomic sentence ψ, there is an i such that p_i forces ψ or p_i forces $\neg\psi$.

(iii) For every Gödel sentence φ, there is an i such that $p_i \Vdash \varphi$ or p_{i+1} is a condition $q \geq p_i$ as required for φ in the appropriate clause of Lemma 5.12.

Generic sequence

Lemma 5.14: For every forcing condition p in a frame $\mathcal{C}$, there is a generic sequence extending p.

Proof: Let $\{\varphi_i \mid i \in \mathcal{N}\}$ be an enumeration of all the Gödel sentences. We define a sequence $\langle p_i \mid i \in \mathcal{N}\rangle$ by induction. Set $p_0 = p$. If $p_i \Vdash \varphi_i$ and φ_i is $\neg\neg\psi$ for some atomic sentence ψ, then, by the definition of forcing, there is a $q \geq p_i$ that forces ψ. Let p_{i+1} be such a q. If $p_i \Vdash \varphi_i$ but φ_i is not of this form, let $p_{i+1} = p_i$. If p_i does not force φ_i, let p_{i+1} be a condition extending p_i as guaranteed in the clause of Lemma 5.12 corresponding to φ_i. (So, in particular, if φ_i is $\neg\neg\psi$ for an atomic ψ and p_i does not force φ_i, then $p_{i+1} \Vdash \neg\psi$.) It is clear that the sequence p_i satisfies the definition of a generic sequence extending p. □

Gödel sentence

Theorem 5.15: For every Gödel sentence φ, φ is classically valid if and only if φ is intuitionistically valid.

Proof: As we remarked, if φ is intuitionistically valid, it is classically valid (Theorem 2.6). To prove the converse suppose that φ is not intuitionistically valid, i.e., there is a frame $\mathcal{C}$ and a forcing condition p such that p does not force φ. We must build a classical model $\mathcal{A}$ in which φ is false. By Lemma 5.14 we can choose an enumeration of Gödel sentences in which $\varphi_0 = \varphi$ and a generic sequence $\langle p_i \mid i \in \mathcal{N} \rangle$ in $\mathcal{C}$ extending p. Note that, by our assumption that p does not force φ and the definition of a generic sequence, $p_1 \Vdash \neg\varphi$. We let the universe A of our required classical model A be $\cup\{C(p_i) \mid i \in \mathcal{N}\}$. We define the relations of $\mathcal{A}$ by $\mathcal{A} \Vdash R(\vec{c})$ iff $\exists i(p_i \Vdash R(\vec{c}))$

We claim that, for every Gödel sentence φ, $\mathcal{A} \Vdash \varphi \;\Leftrightarrow\; \exists i(p_i \Vdash \varphi)$. As $p_1 \Vdash \neg\varphi$ (and $\neg\varphi$ is a Gödel sentence), this gives us the desired classical model of $\neg\varphi$. The proof is by induction on the formation of the sentence φ. (Note that we specify the levels of formation of formulas in the order given in Definition 5.11 so that $\forall x\psi(x)$ follows $\neg\psi(c)$ for each constant c.)

The base case is that φ is $\neg\neg\psi$ for some atomic sentence ψ. As p_i is a generic sequence, there is an i such that p_i forces ψ or p forces $\neg\psi$. In the former case, $\mathcal{A}$ satisfies ψ (and so $\neg\neg\psi$) by definition. In the latter case, no p_j can force ψ. So by the definition of $\mathcal{A}$, ψ is false in $\mathcal{A}$.

If φ is $\psi \wedge \theta$, then $\mathcal{A} \models \varphi \;\Leftrightarrow\; \mathcal{A} \models \psi$ and $\mathcal{A} \models \theta$. By induction this condition holds iff there are j and k such that $p_j \Vdash \psi$ and $p_k \Vdash \theta$. As the p_i form an increasing sequence of forcing conditions, this is equivalent to the existence of an i such that $p_i \Vdash \psi \wedge \theta$, i.e., one such that $p_i \Vdash \varphi$.

If φ is $\neg\psi$, then $\mathcal{A} \models \varphi \Leftrightarrow \mathcal{A} \not\models \psi$. By induction this is true if and only if there is no j such that $p_j \Vdash \psi$. By the definition of generic sequence, this last condition is equivalent to the existence of an i such that $p_i \Vdash \neg\psi$, i.e., $p_i \Vdash \varphi$.

If φ is $\forall x\psi(x)$, suppose first that $\mathcal{A} \Vdash \varphi$. If there is no p_i forcing φ, then by the definition of a generic sequence, there is an i and a c such that $p_i \Vdash \neg\psi(c)$. Then, by induction, $\mathcal{A} \models \neg\psi(c)$ for the desired contradiction. For the converse, suppose that there is an i such that $p_i \Vdash \forall x\psi(x)$.Then, for every $c \in A$,there is a $j \geq i$ such that $c \in C(p_j)$ and $p_j \Vdash \psi(c)$. By induction, we then know that $\mathcal{A} \models \psi(c)$ for every $c \in A$, i.e., $\mathcal{A} \models \varphi$ as required. □

Undecidability

Theorem 5.16: The validity problem for intuitionistic logic is undecidable.

Proof: If we could effectively decide if any given sentence ψ is intuitionistically valid, then we could decide if any sentence φ is classically valid by checking if φ° (as defined in 5.11) is intuitionistically valid. This would contradict the undecidability of validity for classical predicate logic (Corollary III.7.10). □

Undecidability

Corollary 5.17: Not every sentence is intuitionistically equivalent to a sentence in prenex form.

Proof: If every sentence had an intuitionistically equivalent prenex form, a systematic search for a tableau proof of such an equivalence would find one. The decision procedure for the validity of prenex sentences (Exercise 17) would then supply one for all sentences. □

Exercises

1. Exercises 11, 12, 14 in page 306