An Introduction to Classical Propositional Calculus (CPC)

- ♣ Formal (Object) Language (Syntax) of **CPC**
- ♣ Principles of Structural Induction and Structural Recursion
- ♣ Semantics (Model Theory) of **CPC**
- ♣ Semantic (Model-theoretical, Logical) Consequence Relation
- ♣ Normal Forms and Uniform Notation of Formulas
- ♣ Deduction System (Proof Theory) of **CPC**
- ♣ Syntactic (Proof-theoretical, Deductive) Consequence Relation
- ♣ Hilbert Style Formal Logic Systems for **CPC**
- ♣ Gentzen's Natural Deduction System for **CPC**
- ♣ Gentzen's Sequent Calculus System for **CPC**
- ♣ Semantic Tableau System for **CPC**
- ♣ Resolution System for **CPC**
- ♣ Forward Deduction and Backward Deduction

The Basic Idea of Semantic Tableau System

♣ Semantic tableau and the method of analytic tableaux

- ***Semantic tableaux***: [E. Beth, 1959] [Z. Lis, 1960] [R. Smullyan, 1968]
- ***The method of analytic tableaux***: semantic tableaux together with its variants

♣ ***Backward deduction principle*** (***Proof by refutation***)

- To prove a formula in a formal logic system, we starts with the formula, negate it, and then break the result down into simpler and simpler parts until we arrive at an obvious contradiction.

♣ The basic idea of semantic tableau system

- Proving a formula C by showing the unsatisfiability of $\neg C$
- Let $\Gamma = \{A_1, ..., A_n\}$. In proving $\Gamma \models C$, we want to show that no model for $\Gamma\cup\{\neg C\}$ exists, i.e., to show that $(\bigwedge_i A_i)\wedge(\neg C)$ is unsatisfiable.

The Basic Idea of Semantic Tableau System

♣ The deduction theorem

- $\Gamma \models_{\mathbf{CPC}} C$ IFF $\models_{\mathbf{CPC}} (\bigwedge_i A_i)\rightarrow C$ $(\Gamma = \{A_1, ..., A_n\})$

♣ The basic idea of semantic tableau system

- Because of the definition of implication, for any model, if $(\wedge_i A_i)$ is false then $(\wedge_i A_i)\rightarrow C$ is true; if $(\wedge_i A_i)$ is true then the truth-value of $(\wedge_i A_i)\rightarrow C$ depends on the truth of C. Therefore, for any model, if $(\wedge_i A_i)\wedge(\neg C)$ is false, the $(\wedge_i A_i)\rightarrow C$ is true, i.e., if $(\wedge_i A_i)\wedge(\neg C)$ is unsatisfiable, then $(\wedge_i A_i)\rightarrow C$ is valid.
- Because of the definition of implication, if $(\wedge_i A_i)\rightarrow C$ is valid, then for any model, (1) if $(\wedge_i A_i)$ is true then C must be true, $(\wedge_i A_i)\wedge(\neg C)$ must be false, (2) if $(\wedge_i A_i)$ is false then $(\wedge_i A_i)\wedge(\neg C)$ must be false, and (3) either (1) or (2) holds, since $(\wedge_i A_i)$ can only be true or false. Therefore, if $(\wedge_i A_i)\rightarrow C$ is valid, then $(\wedge_i A_i)\wedge(\neg C)$ is unsatisfiable.
- Theorem: $\models_{\mathbf{CPC}} (\bigwedge_i A_i)\rightarrow C$ IFF $(\bigwedge_i A_i)\wedge(\neg C)$ is unsatisfiable.

The Basic Idea of Representing Tableaux Proofs

♣ Representing semantic tableau proofs

- Construct a binary tree where vertexes are labeled with formulas such that: (1) each path should be thought of as representing the conjunction of the formulas appearing on it;
  (2) the binary tree itself as representing the disjunction of its paths.

♣ The semantic tableau system **ST**

- The ***semantic tableau system*** **ST** is closely connected with the notion of disjunctive normal form (DNF) or dual clause form.

An Example of Deduction in ST

$\{A\rightarrow B, B\rightarrow C\} \vdash_{\mathbf{ST}} A\rightarrow C$?

1. $A\rightarrow B$ — 1 is a premise
2. $B\rightarrow C$ — 2 is a premise
3. $\neg(A\rightarrow C)$ — 3 is the negation of the target

4. $\neg A$ 5. B — 4 and 5 are from 1 by β

6. $\neg B$ 7. C 8. $\neg B$ 9. C — 6 (8) and 7 (9) is from 2 by β

+ — +(closed) by 5 and 8

10. A 12. A 14. A — 10 (12,14) and 11 (13,15) are

11. $\neg C$ 13. $\neg C$ 15. $\neg C$ — from 3 by α

+ + +

by 4 and 10 by 7 and 13 by 9 and 15

An Example of Theorem Proof in ST

$\vdash_{\mathbf{ST}} ((A\rightarrow B)\rightarrow((B\rightarrow C)\rightarrow(A\rightarrow C)))$?

1. $\neg((A\rightarrow B)\rightarrow((B\rightarrow C)\rightarrow(A\rightarrow C)))$ — 1 is the negation of the target
2. $A\rightarrow B$ — 2 is from 1 by α
3. $\neg((B\rightarrow C)\rightarrow(A\rightarrow C))$ — 3 is from 1 by α
4. $B\rightarrow C$ — 4 is from 3 by α
5. $\neg(A\rightarrow C)$ — 5 is from 3 by α

6. $\neg A$ 7. B — 6 and 7 are from 2 by β

8. $\neg B$ 9. C 10. $\neg B$ 11. C — 8 (10) and 9 (11) are from 4 by β

+ — +(closed) by 7 and 10

12. A 14. A 16. A — 12 (14,16) and 13 (15,17) are

13. $\neg C$ 15. $\neg C$ 17. $\neg C$ — from 5 by α

+ + +

by 6 and 12 by 9 and 15 by 11 and 17

α-Formulas and β-Formulas and Their Components

Conjunctive			Disjunctive		
α	α_1	α_2	β	β_1	β_2
$X\wedge Y$	X	Y	$\neg(X\wedge Y)$	$\neg X$	$\neg Y$
$\neg(X\vee Y)$	$\neg X$	$\neg Y$	$X\vee Y$	X	Y
$\neg(X\rightarrow Y)$	X	$\neg Y$	$X\rightarrow Y$	$\neg X$	Y
$\neg(X\leftarrow Y)$	$\neg X$	Y	$X\leftarrow Y$	X	$\neg Y$
$\neg(X\neg\wedge Y)$	X	Y	$X\neg\wedge Y$	$\neg X$	$\neg Y$
$X\neg\vee Y$	$\neg X$	$\neg Y$	$\neg(X\neg\vee Y)$	X	Y
$X\neg\rightarrow Y$	X	$\neg Y$	$\neg(X\neg\rightarrow Y)$	$\neg X$	Y
$X\neg\leftarrow Y$	$\neg X$	Y	$\neg(X\neg\leftarrow Y)$	X	$\neg Y$

Tableau Expansion Rules of ST

♣ ***Tableau expansion rules***

- $\dfrac{\neg\neg A}{A}$ $\dfrac{\neg\top}{\bot}$ $\dfrac{\neg\bot}{\top}$ $\dfrac{\alpha}{\begin{matrix}\alpha_1\\ \alpha_2\end{matrix}}$ $\dfrac{\beta}{\beta_1 \mid \beta_2}$
- Let st be a finite tree with vertexes labeled by formulas. Select a branch θ and a non-literal formula occurrence X on θ.
- If X is $\neg\neg A$, lengthen θ by adding a vertex labeled A to its end.
- Similarly, if X is $\neg\top$, add $\bot$, and if X is $\neg\bot$, add $\top$.
- If X is α, add a vertex to the end of θ labeled α_1 and another vertex after that labeled α_2.
- If X is β, add left and right children to the final vertex of θ, and label one β_1 and another β_2.
- Definition: In each case, we call the resulting tree ***st**** and say ***st* results from st by the application of a tableau expansion rule***.

ST: A Semantic Tableau System for CPC

♣ Definition: ***Finite tableaux***

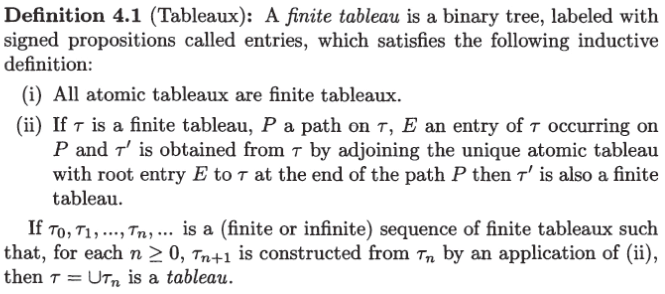
- Let $\{A_1, ..., A_n\}$ be a finite set of formulas.
  (1) The following one branch binary tree is a ***tableau*** for $\{A_1, ..., A_n\}$:
  A_1
  :
  A_n
  (2) If *st* is a tableau for $\{A_1, ..., A_n\}$ and *st** results from st by the application of a tableau expansion rule, then *st** is a ***tableau*** for $\{A_1, ..., A_n\}$.

♣ Definition: ***Closed tableaux***

- A branch (path) θ of a tableau is said to be ***closed*** if both A and $\neg A$ occur on θ for some formula A, or $\bot$ occurs on θ.
- If both A and $\neg A$ occur on branch (path) θ where A is an atomic formula, or if $\bot$ occurs on θ, then θ is said to be ***atomically closed***.
- A tableau is said to be ***(atomically) closed*** if all its branches (paths) are ***(atomically) closed***.

ST: A Semantic Tableau System for CPC

♣ Definition: ***Tableaux*** [N&S]



Definition 4.1 (Tableaux): A *finite tableau* is a binary tree, labeled with signed propositions called entries, which 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*.

- Note: This definition describes all possible tableaux.

ST: A Semantic Tableau System for CPC

♣ Definition: Semantic (Model-theoretical, Logical) consequence relation (entailments) and tableaux

- Let $\Gamma = \{A_1, ..., A_n\}$. $(\bigwedge_i A_i)\rightarrow C$ is said to be ***provable*** in **ST** IFF there is a closed tableau for $\Gamma\cup\{\neg C\}$.
- $\Gamma \models_{\mathrm{CPC}} C$ IFF $(\bigwedge_i A_i)\rightarrow C$ is provable in **ST**.

♣ Definition: ***Tableaux proofs*** and ***theorems***

- A ***tableau proof*** of a formula A in **ST** is a closed tableau for $\{\neg A\}$.
- A formula A is called a ***theorem*** of **ST** IFF A has a tableau proof in **ST**.
- We can write $\vdash_{\mathrm{ST}} A$ to indicate that A has a tableau proof in **ST**.

ST: A Semantic Tableau System for CPC

♣ Some features of tableaux proofs

- Tableau proofs can be very much shorter than truth-table verifications.
- The tableau expansion rules are non-deterministic, i.e., they say what we may do, not what we must do. They allow us to choose which formula to work with next, on which branches. They allow us to skip formulas or use them more than once. And they allow us to close branches at the atomic level or at a more complex level if we can.
- On the other hand, when it comes to incorporating the tableau system in a deterministic computer program, some standardized order of rule applications must be imposed, and various limitations on the basic tableau freedom will be necessary.

♣ Definition: ***Strict tableaux***

- A tableau is ***strict*** if in its construction no formula has had a tableau expansion rule applied to it twice on the same branch.

An Example of Theorem Proof in ST

$\vdash_{\mathrm{ST}} ((A\to(B\to C))\to((A\vee D)\to((B\to C)\vee D)))$?

1. $\neg((A\to(B\to C))\to((A\vee D)\to((B\to C)\vee D)))$		1 is the negation of the target
2. $A\to(B\to C)$		2 is from 1 by α
3. $\neg((A\vee D)\to((B\to C)\vee D))$		3 is from 1 by α
4. $A\vee D$		4 is from 3 by α
5. $\neg((B\to C)\vee D)$		5 is from 3 by α
6. $\neg(B\to C)$		6 is from 5 by α
7. $\neg D$		7 is from 5 by α
8. $\neg A$	9. $B\to C$	8 and 9 are from 2 by β
	+	+(closed) by 6 and 9
10. A	11. D	10 and 11 are from 4
+	+	
by 8 and 10	by 7 and 11	

An Example of Theorem Proof in ST [Fitting]

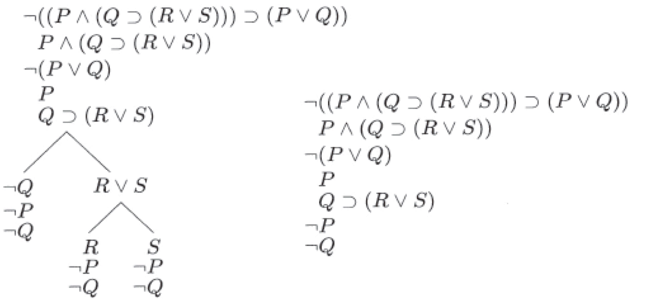


FIGURE 3.3. Two Tableau Proofs of $(P\wedge(Q\supset(R\vee S)))\supset(P\vee Q)$

Properties of ST

♣ The soundness of **ST**

- If $\vdash_{\mathrm{ST}} A$, then $\models_{\mathrm{CPC}} A$, for any $A \in$ **WFF**.

♣ The completeness of **ST**

- If $\models_{\mathrm{CPC}} A$, then $\vdash_{\mathrm{ST}} A$, for any $A \in$ **WFF**.

♣ The relationship between **ST** and **L**

- The theorem set of **ST** is the same as that of **L**.
- **Th**(**ST**) = **Th**(**CPC**) = **Th**(**L**)

♣ Note

- **ST** and **L** just are two different representations of the same propositional logic system **CPC**.

Semantic Tableau System for CPC [R&C]: The (Schemas for) Splitting Rules for Connectives

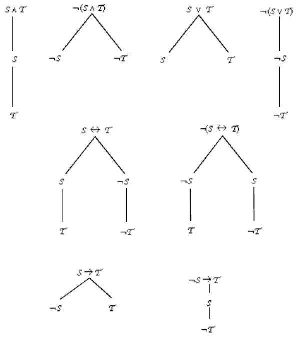


Figure 4.8

Semantic Tableau System for CPC [R&C]: An Example

(A ∧ B) → C, ¬A → D ⊨ B → (C ∨ D) the tableau for which is given in figure 4.9.

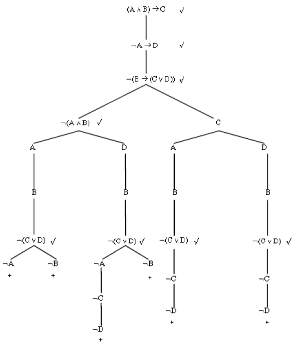


Figure 4.9

Semantic Tableau System for CPC [Smullyan]: Eight Basic Facts Behind the Tableau Method for CPC

♣ The eight basic facts

- Under any interpretation, the following eight facts hold for any formulas X and Y:
  (1) If $\neg X$ is true, then X is false. If $\neg X$ is false, then X is true.
  (2) If $X\wedge Y$ is true, then X and Y are both true. If $X\wedge Y$ is false, then either X is false or Y is false.
  (3) If $X\vee Y$ is true, then either X is true or Y is true. If $X\vee Y$ is false, then X and Y are both false.
  (4) If $X\to Y$ is true, then either X is false or Y is true. If $X\to Y$ is false, then X is true and Y is false.
- Note: The eight basic facts are nothing more than the semantic (logical) interpretations of the logical connectives '¬', '∧', '∨', and '→'.

♣ The eight basic facts as the foundation

- The above eight basic facts underlie the tableau method for **CPC**.

Semantic Tableau System for CPC [Smullyan]: Signed Formulas

♣ Signed formulas

- For several reasons, it will be useful to incorporate the symbols **T** and **F** into our formal language, and define a ***signed formula*** as an expression **T** *X* or **F** *X*, where *X* is a formula (and which will now be called an ***unsigned formula***).
- Informally we read "**T** *X*" as "*X* is true," and "**F** *X*" as "*X* is false."
- Under any interpretation of the propositional variables, a signed formula **T** *X* is called true if *X* is true, and false if *X* is false. A signed formula **F** *X* is called true if *X* is false, and false if *X* is true.

♣ The conjugate of a signed formula

- By the ***conjugate*** of a signed formula we mean the result of changing "**T**" to "**F**" and "**F**" to "**T**."
- Thus **T** *X* and **F** *X* are conjugates of each other.

Semantic Tableau System for CPC [Smullyan]: Illustration of the Tableaux Method

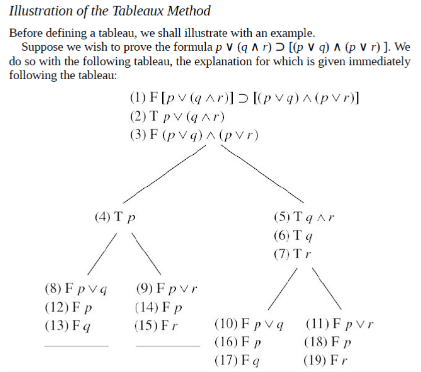


Illustration of the Tableaux Method

Before defining a tableau, we shall illustrate with an example.

Suppose we wish to prove the formula $p \vee (q \wedge r) \supset [(p \vee q) \wedge (p \vee r)]$. We do so with the following tableau, the explanation for which is given immediately following the tableau:

(1) F $[p \vee (q \wedge r)] \supset [(p \vee q) \wedge (p \vee r)]$
(2) T $p \vee (q \wedge r)$
(3) F $(p \vee q) \wedge (p \vee r)$

(4) T p

(8) F $p \vee q$
(12) F p
(13) F q

(9) F $p \vee r$
(14) F p
(15) F r

(5) T $q \wedge r$
(6) T q
(7) T r

(10) F $p \vee q$
(16) F p
(17) F q

(11) F $p \vee r$
(18) F p
(19) F r

Note: Here '⊃' is '→' in our notation.

Semantic Tableau System for CPC [Smullyan]: Illustration of the Tableaux Method

- As we construct this tableau we are attempting to see if we can derive a contradiction from the assumption that the formula $(p\vee(q\wedge r)) \rightarrow ((p\vee q)\wedge(p\vee r))$ is false.
- The first line consists of this formula preceded by the letter '**F**'. Now, a formula of the form $X \rightarrow Y$ can be false only if *X* is true and *Y* is false. Thus in the language of tableaux, both **T** *X* and **F** *Y* are direct consequences of the signed formula F $X \rightarrow Y$. Hence we write the lines (2) and (3) as direct consequences of line (1).
- Next, we look at line (2), which is of the form **T** $X\vee Y$, where $X = p$ and $Y = q\wedge r$. We cannot draw any conclusion about the truth of *X*, nor of the truth of *Y*. All we can infer is that either **T** *X* (*X* is true) or **T** *Y* (*Y* is true). Hence the tableau branches into the two possibilities – lines (4) and (5).
- Line (5), which is **T** $q\wedge r$, immediately yields **T** *q* and **T** *r* as direct consequences, and we thus have lines (6) and (7).
- Now we look at line (3). It is of the form **F** $X\wedge Y$, where $X = p\vee q$ and $Y = p\vee r$. No direct conclusion can be drawn; all we can infer is that either **F** *X* or **F** *Y*. We also know that either (4) or (5) holds, and so for each of the possibilities (4), (5), we have the two possibilities **F** *X*, **F** *Y*.

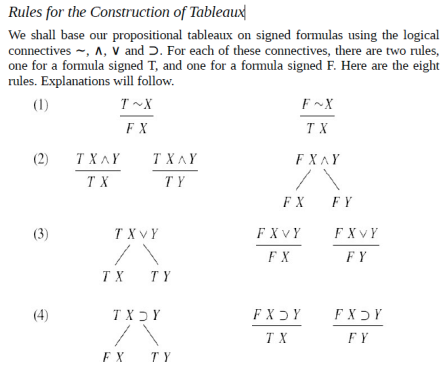
Semantic Tableau System for CPC [Smullyan]: Illustration of the Tableaux Method

- Thus there are now four possibilities. And so each of the branches (4), (5) branches again into the possibilities **F** *X*, **F** *Y*. Thus (4) branches to (8), (9) and (5) branches to (10), (11).
- Lines (12), (13) are direct consequences of (8), while lines (14), (15) are direct consequences of (9).
- Similarly, lines (16) and (17) are direct consequences of (10), and lines (18) and (19) are direct consequences of (11).
- The tableau is now completed, in the sense that the consequence or consequences of any formula on any line have also been entered into the tableau appropriately. Every line of the tableau that can be used has now been used.
- Examining this tableau, let us look at the leftmost branch, which ends with line (13). We can see that lines (4) and (12), two lines on this branch, are direct contradictions of each other [(12) is the conjugate of (4)]. Hence they can not both be true, and so we close the branch by putting bar under (13), signifying that the branch leads to a contradiction, and so is not a real possibility.
- Similarly, (14) and (4) contradict each other, and so we can close the branch ending with (15). The next branch (going from left to right) is closed by virtue of (17) and (6). Finally, the rightmost branch is closed by virtue of (19) and (7).

Semantic Tableau System for CPC [Smullyan]: Illustration of the Tableaux Method

- Thus all branches lead to a contradiction, so line (1) is untenable. Thus the formula $(p\vee(q\wedge r)) \rightarrow ((p\vee q)\wedge(p\vee r))$ cannot be false under any interpretation, and so is a tautology.
- What we do in the tableau method is explore all the possible scenarios in which a formula *X* could be false. Each branch of a completed tableau shows the situation in one of the possible scenarios.
- And the reason we could conclude that the formula $(p\vee(q\wedge r)) \rightarrow ((p\vee q)\wedge(p\vee r))$ of our example could not ever be false is that we always reached contradictions in following out every single possible scenario.
- We can define a ***completed tableau*** as one such that on all the open branches (i.e. branches that are not closed), all formulas of the branch that can be used have been used.

Semantic Tableau System for CPC [Smullyan]: Rules for the Construction of Tableaux Using Signed Formulas



Rules for the Construction of Tableaux

We shall base our propositional tableaux on signed formulas using the logical connectives ~, ∧, ∨ and ⊃. For each of these connectives, there are two rules, one for a formula signed T, and one for a formula signed F. Here are the eight rules. Explanations will follow.

(1) $\dfrac{T \sim X}{F\ X}$ $\dfrac{F \sim X}{T\ X}$

(2) $\dfrac{T\ X \wedge Y}{T\ X}$ $\dfrac{T\ X \wedge Y}{T\ Y}$ $F\ X \wedge Y$: $F\ X$ $F\ Y$

(3) $T\ X \vee Y$: $T\ X$ $T\ Y$ $\dfrac{F\ X \vee Y}{F\ X}$ $\dfrac{F\ X \vee Y}{F\ Y}$

(4) $T\ X \supset Y$: $F\ X$ $T\ Y$ $\dfrac{F\ X \supset Y}{T\ X}$ $\dfrac{F\ X \supset Y}{F\ Y}$

Note: Here '⊃' is '→' in our notation.

Semantic Tableau System for CPC [Smullyan]: Rules for the Construction of Tableaux Using Signed Formulas

♣ Explanations on the rules

- Rule (1) means that from **T** $\neg X$ we can directly infer **F** X, in the sense that we can subjoin **F** X to any branch through **T** $\neg X$. (To "subjoin" here means to add on at the end of the branch.) And from **F** $\neg X$ we can directly infer **T** X (again by subjoining it at the end of a branch containing the formula **F** $\neg X$).
- Rule (2) means that **T** $X \wedge Y$ directly yields both **T** X and **T** Y, in the sense that on any branch that contains **T** $X \wedge Y$ we can subjoin either of the two formulas **T** X and **T** Y. On the other hand, Rule (2) also says that if we wish to draw a conclusion from **F** $X \wedge Y$ on a branch containing that formula, the branch must divide, branching at its end to both **F** X and **F** Y. (In other words, in the case of the rule for **F** $X \wedge Y$, if θ is a branch containing **F** $X \wedge Y$, when we apply Rule (2) to **F** $X \wedge Y$, we divide θ into two branches at its end, with **F** X as the last formula of one of the new branches, and **F** Y as the last formula of the other branch).

Semantic Tableau System for CPC [Smullyan]: Rules for the Construction of Tableaux Using Signed Formulas

♣ Explanations on the rules

- Rules (3) and (4) are similarly understood. Note that what we conclude directly from a signed formula according to the tableau rules is precisely everything we can conclude from our assumption about the truth or falsity of the signed formula based on the meaning of the sign and of the highest level connective in the formula (of course they can be other connectives in X and Y that went into their formation at an earlier stage).

♣ The way to complete a tableau

- One way of being sure to complete a tableau is to systematically work downward through the tree, i.e., to never use a line until all lines above it (on the same branch) have been used.

Semantic Tableau System for CPC [Smullyan]: Rules for the Construction of Tableaux Using Signed Formulas

♣ Rules for the construction of tableaux using signed formulas [N&S]

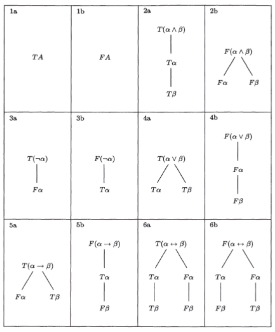


FIGURE 9

Semantic Tableau System for CPC [Smullyan]: Completed Tableau

♣ Closed branch and closed tableau

- A branch on a tableau is considered ***closed*** if it contains a formula and its conjugate, and is considered ***open*** otherwise. A tableau is considered ***closed*** if all its branches are closed, and is considered ***open*** otherwise.
- A ***completed tableau*** is one such that on all the open branches, all formulas of the branch that can be used have been used.

♣ Two types of signed formulas

- Assuming the formula we are drawing our consequences from is true.
- ***Type A***: Those formulas that have one or two direct consequences which we know must be true (**T** $X \wedge Y$, **F** $X \vee Y$, **F** $X \rightarrow Y$, **T** $\neg X$, **F** $\neg X$).
- ***Type B***: Those formulas that branch to two possible consequences, one of which must be true (**F** $X \wedge Y$, **T** $X \vee Y$, **T** $X \rightarrow Y$, **T** $\neg X$, **F** $\neg X$).
- To complete a tableau, it is more efficient to give priority to lines containing formulas of type A, i.e., to use up all such lines at hand before using lines containing formulas of type B.

Semantic Tableau System for CPC [Smullyan]: Signed α-β Formulas

α	α_1	α_2
T $(X \wedge Y)$	T X	T Y
F $(X \vee Y)$	F X	F Y
F $(X \supset Y)$	T X	F Y
T $\sim X$	F X	F X
F $\sim X$	T X	T X

We note that under any interpretation, any α is true if and only if its components α_1 and α_2 are both true. Accordingly, we refer to α-formulas as ones of *conjunctive type*.

- Note: Here '⊃' is '→' in our notation.

Semantic Tableau System for CPC [Smullyan]: Signed α-β Formulas

β	β_1	β_2
F $(X \wedge Y)$	F X	F Y
T $(X \vee Y)$	T X	T Y
T $(X \supset Y)$	F X	T Y
T $\sim X$	F X	F X
F $\sim X$	T X	T X

We note that under any interpretation, β is true if and only if at least one of its components β_1 and β_2 is true. Accordingly, we refer to a β-formula as a formula of *disjunctive type*.

- Note: Here '⊃' is '→' in our notation.

Semantic Tableau System for CPC [Smullyan]: Logical Consequence

♣ Representing logical consequence

- The method of tableaux can be used to show that a formula X is a logical consequence of a finite set S of formulas.
- For example, suppose we wish to show that $p\rightarrow r$ is a logical consequence of the two formulas $p\rightarrow q$, $q\rightarrow r$. One way, of course, is to construct a closed tableau starting with the formula **F** $((p\rightarrow q)\wedge(q\rightarrow r))\rightarrow(p\rightarrow r)$. But it is more economical (and understandable!) to instead start a tableau with the three lines: **T** $p\rightarrow q$, **T** $q\rightarrow r$, **F** $p\rightarrow r$ and show that all branches close.
- In general, to show that a formula Y is a logical consequence of formulas $X_1, ..., X_n$, we can either construct a closed tableau starting with $(X_1\wedge ... \wedge X_n)\rightarrow$Y, or (and preferably, in terms of the number of times rules must be applied) construct a closed tableau starting with the lines: **T** X_1, **T** X_2, …, **F** Y.

Semantic Tableau System for CPC [Smullyan]: Tableaux Using Unsigned Formulas

♣ The tableaux using unsigned formulas

- For certain purposes, we will also have use for tableaux using unsigned formulas.
- These differ from those using signed formulas in that '**T**' is deleted and '**F**' is replaced by '¬'.
- Thus the rules for constructing tableaus for unsigned formulas are the following [note that the left half of Rule (1) for signed formulas is now superfluous].

Semantic Tableau System for CPC [Smullyan]: Rules for the Construction of Tableaux Using Unsigned Formulas

(1) $$\frac{\sim\sim X}{X}$$

(2) $$\frac{X\wedge Y}{X} \qquad \frac{X\wedge Y}{Y} \qquad \begin{array}{c}\sim(X\wedge Y)\\ /\quad\backslash\\ \sim X\quad\ \sim Y\end{array}$$

(3) $$\begin{array}{c}X\vee Y\\ /\quad\backslash\\ X\quad\ Y\end{array} \qquad \frac{\sim(X\vee Y)}{\sim X} \qquad \frac{\sim(X\vee Y)}{\sim Y}$$

(4) $$\begin{array}{c}(X\supset Y)\\ /\quad\backslash\\ \sim X\quad\ Y\end{array} \qquad \frac{\sim(X\supset Y)}{X} \qquad \frac{\sim(X\supset Y)}{\sim Y}$$

Note: Here '⊃' is '→' in our notation.

Semantic Tableau System for CPC [Smullyan]: The Construction of Tableaux Using Unsigned Formulas

♣ The construction of tableaux using unsigned formulas

- For tableaux using unsigned formulas, closing a branch means terminating the branch whenever some formula and its negation are both on it.
- For any unsigned formula Z, it will prove convenient to define $!Z$ as follows:
  (1) If Z is of one of the forms $(X\wedge Y)$, $(X\vee Y)$, $(X\rightarrow Y)$, or $(X\leftrightarrow Y)$, we take $!Z$ to be $\neg Z$.
  (2)If Z is of the form p, where p is a propositional variable, we take $!Z$ to be ¬p.
  (3) If Z is of one of the forms $\neg X$, $\neg(X\wedge Y)$, $\neg(X\vee Y)$, $\neg(X\rightarrow Y)$, or $\neg(X\leftrightarrow Y)$ we take $!Z$ to be X, $(X\wedge Y)$, $(X\vee Y)$, $(X\rightarrow Y)$ or $(X\leftrightarrow Y)$, respectively.
- $!Z$ is logically equivalent to $\neg Z$, and is called the conjugate of Z.

Semantic Tableau System for CPC [Smullyan]: Proof in the Context of Tableaux for CPC

♣ Proof in the context of tableaux

- We say that we have proved that a formula X is a tautology, or simply say we have proved the formula X, if we have found a closed tableau starting with **F** X (in the case of tableaux constructed with signed formulas) or starting with $\neg X$ (in the case of tableaux constructed with unsigned formulas).

♣ The correctness (soundness) and completeness

(1) *Correctness*. The tableau method is *correct* in the sense that if an unsigned formula X is provable by the tableau method (e.g. if there is a closed tableau starting with FX), then X really is a tautology.

(2) *Completeness*. The tableau method is *complete*, in the sense that every tautology is provable by the tableau method. Better still, if X is a tautoloty, then not only is there a closed tableau starting with FX, but *every* completed tableau starting with FX will be closed.

Semantic Tableau System for CPC [Smullyan]: Unsigned α-β Formulas

α	α_1	α_2	β	β_1	β_2
$X\wedge Y$	X	Y	$\sim(X\wedge Y)$	$\sim X$	$\sim Y$
$\sim(X\vee Y)$	$\sim X$	$\sim Y$	$X\vee Y$	X	Y
$\sim(X\supset Y)$	X	$\sim Y$	$X\supset Y$	$\sim X$	Y
$\sim\sim X$	X	X	$\sim\sim X$	X	X

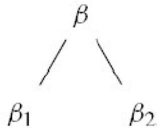
- Note: Here '⊃' is '→' in our notation.

Semantic Tableau System for CPC [Smullyan]]: Rules for the Construction of Tableaux Using α-β Notation

- Using our α – β notation, the eight tableau rules can now be collapsed to two:

Rule A $\frac{\alpha}{\alpha_1}$ $\frac{\alpha}{\alpha_2}$

Rule B



An Introduction to Classical Propositional Calculus (CPC)

- Formal (Object) Language (Syntax) of **CPC**
- Principles of Structural Induction and Structural Recursion
- Semantics (Model Theory) of **CPC**
- Semantic (Model-theoretical, Logical) Consequence Relation
- Normal Forms and Uniform Notation of Formulas
- Deduction System (Proof Theory) of **CPC**
- Syntactic (Proof-theoretical, Deductive) Consequence Relation
- Hilbert Style Formal Logic Systems for **CPC**
- Gentzen's Natural Deduction System for **CPC**
- Gentzen's Sequent Calculus System for **CPC**
- Semantic Tableau System for **CPC**
- Resolution System for **CPC**
- Forward Deduction and Backward Deduction

The Basic Idea of Resolution System

- ***Resolution*** [J. A. Robinson, 1965]
  - Modus Ponens: From A and $\neg A \vee B$ ($A \rightarrow B$) to infer B
  - Cut: From $A \vee C$ and $\neg A \vee B$ to infer $C \vee B$
  - These two inference rules underlie the resolution method.
  - The method of resolution, invented by J.A. Robinson in 1965, is an efficient method for searching for a proof of theorem.
- ***Backward deduction principle*** (***Proof by refutation***)
  - To prove a formula in a formal logic system, we starts with the formula, negate it, and then break the result down into simpler and simpler parts until we arrive at an obvious contradiction.
  - Note: This is the same as to that of semantic tableau method.

The Basic Idea of Resolution System

- The basic idea of resolution
  - Proving a formula C by showing the unsatisfiability of $\neg C$
  - Let $\Gamma = \{A_1, ..., A_n\}$. In proving $\Gamma \models C$, we want to show that no model for $\Gamma \cup \{\neg C\}$ exists, i.e., to show that $(\bigwedge_i A_i) \wedge (\neg C)$ is unsatisfiable.
  - Note: This is the same as to that of semantic tableau method.
- Representation of resolution proofs
  - List disjuncts within square brackets, and represent a conjunction of disjunctions by listing its members in a sequence, one disjunction to a line.
- ***Resolution system***
  - The ***resolution system*** is closely connected with the notion of conjunctive normal form (CNF) or clause form.
  - Note: The semantic tableau system is closely connected with the notion of disjunctive normal form (DNF) or dual clause form.

An Example of Deduction in R

$\{A \rightarrow B, B \rightarrow C\} \vdash_{\mathbf{R}} A \rightarrow C$?

1. $[A \rightarrow B]$	1 is a premise
2. $[B \rightarrow C]$	2 is a premise
3. $[\neg(A \rightarrow C)]$	3 is the negation of the target
4. $[\neg A, B]$	4 is from 1 by β
5. $[\neg B, C]$	5 is from 2 by β
6. $[A]$	6 is from 3 by α
7. $[\neg C]$	7 is from 3 by α
8. $[\neg A, C]$	8 is the resolution rule on B in 4 and 5
9. $[C]$	9 is the resolution rule on A in 6 and 8
10. []	10 is the resolution rule on C in 7 and 9

An Example of Theorem Proof in R

$\vdash_{\mathbf{R}} ((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$?

1. $[\neg((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))]$	1 is the negation of the target
2. $[A \rightarrow B]$	2 is from 1 by α
3. $[\neg((B \rightarrow C) \rightarrow (A \rightarrow C))]$	3 is from 1 by α
4. $[\neg A, B]$	4 is from 2 by β
5. $[B \rightarrow C]$	5 is from 3 by α
6. $[\neg(A \rightarrow C)]$	6 is from 3 by α
7. $[\neg B, C]$	7 is from 5 by β
8. $[A]$	8 is from 6 by α
9. $[\neg C]$	9 is from 6 by α
10. $[\neg A, C]$	10 is the resolution rule on B in 4 and 7
11. $[C]$	11 is the resolution rule on A in 8 and 10
12. []	12 is the resolution rule on C in 9 and 11

α-Formulas and β-Formulas and Their Components

Conjunctive			Disjunctive		
α	α_1	α_2	β	β_1	β_2
$X \land Y$	X	Y	$\neg(X \land Y)$	$\neg X$	$\neg Y$
$\neg(X \lor Y)$	$\neg X$	$\neg Y$	$X \lor Y$	X	Y
$\neg(X \rightarrow Y)$	X	$\neg Y$	$X \rightarrow Y$	$\neg X$	Y
$\neg(X \leftarrow Y)$	$\neg X$	Y	$X \leftarrow Y$	X	$\neg Y$
$\neg(X \neg\land Y)$	X	Y	$X \neg\land Y$	$\neg X$	$\neg Y$
$X \neg\lor Y$	$\neg X$	$\neg Y$	$\neg(X \neg\lor Y)$	X	Y
$X \neg\rightarrow Y$	X	$\neg Y$	$\neg(X \neg\rightarrow Y)$	$\neg \mathrm{X}$	Y
$X \neg\leftarrow Y$	$\neg X$	Y	$\neg(X \neg\leftarrow Y)$	X	$\neg Y$

Resolution Expansion Rules of R

♣ ***Resolution expansion rules***

$$\frac{\neg\neg A}{A} \qquad \frac{\neg \mathrm{T}}{\bot} \qquad \frac{\neg \bot}{\mathrm{T}} \qquad \frac{\beta}{\begin{array}{c}\beta_1 \\ \beta_2\end{array}} \qquad \frac{\alpha}{\alpha_1 \mid \alpha_2}$$

- Let gd be a generalized disjunction containing a non-literal formula occurrence X.
- If X is $\neg\neg A$, then a disjunction follows that is like gd except that it contains an occurrence of A where gd contained $\neg\neg A$.
- Similarly, if X is $\neg\mathrm{T}$, add $\bot$, and if X is $\neg\bot$, add T.
- If X is β, then a disjunction follows that is like gd except that it contains occurrences of both β_1 and β_2 where gd contained β.
- If X is α, then two disjunctions follow, one like gd but with α replaced by α_1, the other with α replaced by α_2.

Resolution Expansion Rules of R

♣ Definition: ***Applications of resolution expansion rules***

- In each case of resolution expansion rules, we say *the new disjunction (disjunctions) follows from gd by the application of a resolution expansion rule*.

♣ Definition: ***Strict applications of resolution expansion rules***

- A sequence of applications of resolution expansion rules is ***strict*** if every disjunction has at most one resolution expansion rule applied to it.

The Propositional Resolution Rule

♣ Definition: ***Resolvent***

- Let D_1 and D_2 be two generalized disjunctions, with X occurring as a member of D_1 and $\neg X$ as a member of D_2. Let D be the result of the following:
  (1) deleting all occurrences of X from D_1,
  (2) deleting all occurrences of $\neg X$ from D_2, and
  (3) combining the resulting two disjunctions as one.
  D is called the result of ***resolving D_1 and D_2 on X***, or the ***resolvent*** of D_1 and D_2, with X being the formula ***resolved on***.
- If X is a generalized disjunction containing $\bot$ and D is the result of deleting all occurrences of $\bot$ from X, then D is called the ***trivial resolvent*** of X.
- Theorem: If D is the resolvent of D_1 and D_2, then $\{D_1, D_2\} \models_{\mathbf{CPC}} D$.

The Propositional Resolution Rule

♣ ***Propositional resolution rule***

- If D is the result of resolving D_1 and D_2 on X, then from D_1 and D_2 to infer D.
- ***Trivial application of the propositional resolution rule***: If D is the trivial resolvent of D_1, then from D_1 to infer D.
- An application of the propositional resolution rule is called an ***atomic application*** if the formula resolved on is atomic.

R: A Resolution System for CPC

♣ Definition: ***Resolution expansion***

- Let $\{A_1, A_2, \ldots, A_n\}$ be a finite set of formulas.
  (1) The following sequence is a resolution expansion for $\{A_1, \ldots, A_n\}$:
  $[A_1]$
  $[A_2]$
  :
  $[A_n]$
  (2) If r is a resolution expansion for $\{A_1, \ldots, A_n\}$ and D results from r some line or lines of r by the application of a resolution expansion rule or the resolution rule, then r with D added as a new last line is also a resolution expansion for $\{A_1, \ldots, A_n\}$.

♣ Definition: ***Closed resolution expansion***

- A resolution expansion is ***closed*** if it contains the empty clause.

R: A Resolution System for CPC

- ♣ Definition: Semantic (Model-theoretical, Logical) consequence relation (entailments) and resolution expansion
  - Let $\Gamma = \{A_1, ..., A_n\}$. $(\bigwedge_i A_i) \rightarrow C$ is said to be ***provable*** in **R** IFF there is a closed resolution expansion for $\{A_1, ..., A_n, \neg C\}$.
  - $\Gamma \models_{\mathbf{CPC}} C$ IFF $(\bigwedge_i A_i) \rightarrow C$ is provable in **R**.
- ♣ Definition: Resolution proofs and theorems
  - A ***resolution proof*** of a formula A in **R** is a closed resolution expansion for $\{\neg A\}$.
  - A formula A is called a ***theorem*** of **R** IFF A has a resolution proof in **R**.
  - We can write $\vdash_{\mathbf{R}} A$ to indicate that A has a resolution proof in **R**.

An Example of Theorem Proof in R

A strict resolution proof of $((A \wedge B) \vee (C \rightarrow D)) \rightarrow ((A \vee (C \rightarrow D)) \wedge (B \vee (C \rightarrow D)))$

1. $[\neg(((A \wedge B) \vee (C \rightarrow D)) \rightarrow ((A \vee (C \rightarrow D)) \wedge (B \vee (C \rightarrow D))))]$	
2. $[(A \wedge B) \vee (C \rightarrow D)]$	2 is from 1 by α
3. $[\neg((A \vee (C \rightarrow D)) \wedge (B \vee (C \rightarrow D)))]$	3 is from 1 by α
4. $[A \wedge B, C \rightarrow D]$	4 is from 2 by β
5. $[A, C \rightarrow D]$	5 is from 4 by α
6. $[B, C \rightarrow D]$	6 is from 4 by α
7. $[\neg(A \vee (C \rightarrow D)), \neg(B \vee (C \rightarrow D))]$	7 is from 3 by β
8. $[\neg A, \neg(B \vee (C \rightarrow D))]$	8 is from 7 by α
9. $[\neg(C \rightarrow D), \neg(B \vee (C \rightarrow D))]$	9 is from 7 by α
10. $[\neg A, \neg B]$	10 is from 8 by α
11. $[\neg A, \neg(C \rightarrow D)]$	11 is from 8 by α
12. $[\neg(C \rightarrow D), \neg B]$	12 is from 9 by α
13. $[\neg(C \rightarrow D), \neg(C \rightarrow D)]$	13 is from 9 by α
14. $[A, \neg B]$	14 is by the resolution rule on $R \rightarrow S$ in 5 and 12
15. $[\neg B]$	15 is by the resolution rule on P in 10 and 14
16. $[C \rightarrow D]$	16 is by the resolution rule on Q in 6 and 15
17. $[]$	17 is by the resolution rule on $R \rightarrow S$ in 13 and 16

Properties of R

- ♣ The soundness of **R**
  - If $\vdash_{\mathbf{R}} A$, then $\models_{\mathbf{CPC}} A$, for any $A \in$ **WFF**.
- ♣ The completeness of **R**
  - If $\models_{\mathbf{CPC}} A$, then $\vdash_{\mathbf{R}} A$, for any $A \in$ **WFF**.
- ♣ The relationship between **R** and **L**
  - The theorem set of **R** is the same as that of **L**.
  - **Th**(**R**) = **Th**(**CPC**) = **Th**(**L**)
- ♣ Note
  - **R** and **L** just are two different representations of the same propositional logic system **CPC**.

Normal Forms of Formulas: Literal and Clause

- ♣ Generalized disjunction and conjunction

  Let $X_1, X_2, ..., X_n$ be a list of formulas.
  - ***Generalized disjunction***: $[X_1, X_2, ..., X_n] =_{\mathrm{df}} X_1 \vee X_2 \vee ... \vee X_n$.
  - ***Generalized conjunction***: $<X_1, X_2, ..., X_n> =_{\mathrm{df}} X_1 \wedge X_2 \wedge ... \wedge X_n$.
- ♣ Literal
  - A formula is called a ***literal***, if it is a propositional variable or the negation of a propositional variable, or a constant, ⊤ or ⊥.
- ♣ Clause
  - A ***clause*** is a generalized disjunction $[X_1, X_2, ..., X_n]$ in which each member is a literal.
  - A ***dual clause*** is a generalized conjunction $<X_1, X_2, ..., X_n>$ in which each member is a literal.

Resolution System for CPC [Ben-Ari]: Clausal Form of Formula

- ♣ Clausal form of formula
  - The clausal form of formula is a notational variant of CNF.
- ♣ Definition: ***Clausal form of formula***
  - A ***clause*** is a set of literals.
  - A clause is considered to be an implicit disjunction of its literals.
  - A ***unit clause*** is a clause consisting of exactly one literal.
  - The empty set of literals is the ***empty clause***, denoted by [].
  - A formula in ***clausal form*** is a set of clauses.
  - A formula is considered to be an implicit conjunction of its clauses.
  - The formula that is the ***empty set of clauses*** is denoted by ∅.
- ♣ An example
  - The CNF formula $(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$ is logically equivalent to its clausal form:
    $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$.

Resolution System for CPC [Ben-Ari]: Clausal Form of Formula

- ♣ Corollary
  - Every formula φ in **CPC** can be transformed into a logically equivalent formula in clausal form.
- ♣ Definition: ***Trivial clauses***
  - A clause if ***trivial*** if it contains a pair of clashing literals $(p, \neg p)$.
  - Since a trivial clause is valid $(p \vee \neg p \equiv \text{true})$, it can be removed from a set of clauses without changing the truth-value of the formula.
- ♣ Lemma
  - Let S be a set of clauses and let $C \in S$ be a trivial clause. Then $S - \{C\}$ is logically equivalent to S.
- ♣ Lemma
  - [], the empty clause, is unsatisfiable.
  - ∅, the empty set of clauses, is valid.

Resolution System for CPC [Ben-Ari]: Resolution Rule

♣ ***Resolution rule***

- Let C_1, C_2 be clauses such that $l \in C_1$, $\neg l \in C_2$.
- The clauses C_1, C_2 are said to be ***clashing clauses*** and to clash on the complementary pair of literals l, $\neg l$.
- C, the ***resolvent*** of C_1 and C_2, is the clause: $C = \mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{\neg l\})$.
- C_1 and C_2 are the parent clauses of C.
- Resolution is only performed if the pair of clauses clash on exactly one pair of complementary literals.
- If two clauses clash on more than one literal, their resolvent is a trivial clause.

♣ An example

- The pair of clauses $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$ clash on the pair of complementary literals c, ¬c. The resolvent is: $C = (\{a, b, \neg c\} - \{\neg c\}) \cup (\{b, c, \neg e\} - \{c\}) = \{a, b\} \cup \{b, \neg e\} = \{a, b, \neg e\}$.

Resolution System for CPC [Ben-Ari]: Resolution Rule

♣ Theorem

- The resolvent C is satisfiable IFF the parent clauses C_1 and C_2 are both satisfiable.

♣ Resolution procedure

Algorithm 4.18 (Resolution procedure)
Input: A set of clauses S.
Output: S is satisfiable or unsatisfiable.

Let S be a set of clauses and define $S_0 = S$.

Repeat the following steps to obtain S_{i+1} from S_i until the procedure terminates as defined below:

- *Choose* a pair of clashing clauses $\{C_1, C_2\} \subseteq S_i$ that has not been chosen before.
- Compute $C = Res(C_1, C_2)$ according to the resolution rule.
- If C is not a trivial clause, let $S_{i+1} = S_i \cup \{C\}$; otherwise, $S_{i+1} = S_i$.

Terminate the procedure if:

- $C = \square$.
- All pairs of clashing clauses have be resolved. ■

An Introduction to Classical Propositional Calculus (CPC)

♣ Formal (Object) Language (Syntax) of **CPC**
♣ Principles of Structural Induction and Structural Recursion
♣ Semantics (Model Theory) of **CPC**
♣ Semantic (Model-theoretical, Logical) Consequence Relation
♣ Normal Forms and Uniform Notation of Formulas
♣ Deduction System (Proof Theory) of **CPC**
♣ Syntactic (Proof-theoretical, Deductive) Consequence Relation
♣ Hilbert Style Formal Logic Systems for **CPC**
♣ Gentzen's Natural Deduction System for **CPC**
♣ Gentzen's Sequent Calculus System for **CPC**
♣ Semantic Tableau System for **CPC**
♣ Resolution System for **CPC**
♣ Forward Deduction and Backward Deduction

Forward Deduction and Backward Deduction

♣ ***Forward deduction systems***

- Hilbert Style Formal Systems
- Natural Deduction Systems
- Sequent Calculus Systems

♣ ***Backward deduction systems***

- Semantic Tableau Systems
- Resolution Systems

♣ The features of forward deduction and backward deduction

- Forward deduction: suitable for reasoning, ATF
- Backward deduction: suitable for proving, ATP
- Note: Recall the differences between reasoning and proving.

An Introduction to Classical Propositional Calculus (CPC)

♣ Formal (Object) Language (Syntax) of **CPC**
♣ Principles of Structural Induction and Structural Recursion
♣ Semantics (Model Theory) of **CPC**
♣ Semantic (Model-theoretical, Logical) Consequence Relation
♣ Normal Forms and Uniform Notation of Formulas
♣ Deduction System (Proof Theory) of **CPC**
♣ Syntactic (Proof-theoretical, Deductive) Consequence Relation
♣ Hilbert Style Formal Logic Systems for **CPC**
♣ Gentzen's Natural Deduction System for **CPC**
♣ Gentzen's Sequent Calculus System for **CPC**
♣ Semantic Tableau System for **CPC**
♣ Resolution System for **CPC**
♣ Forward Deduction and Backward Deduction