2 Tableau system for Classical Propositional Logic

2.1 Introductory remarks

In this chapter, we will define the tableau system for Classical Propositional Logic (for short **CPL**)[1], treating this case as a basic one.

This system can be considered basic because in the definition of tableau system for **CPL** we will use the propositional logic formulas themselves as tableau expressions. In the case of **CPL** it is possible, in other cases it may not be possible.

The defined system also features the property that any branch that begins with a finite set of tableau expressions can be extended to a branch of finite length to which the tableau rule can no longer be applied. We will call this feature *a finite branch property*. However, it is not a feature specific to all the tableau systems described in the book. This feature also makes the presented system for **CPL** a basic case, although for many logics the tableau systems feature the same property.

At the end of the chapter, we show that we actually have defined a tableau system for the **CPL**, because the relation of tableau derivability it determines coincides with the consequence relation determined by the valuation of formulas.

Throughout this chapter, we establish certain conventions of notation and order of defining tableau concepts and proving facts, which conventions and order will guide us within the book.

2.2 Language and semantics

The construction of a tableau system for the classical logic will start with the basic definitions. First, we will take up the language of **CPL**.

Definition 2.1 (Alphabet of **CPL**)**.** *Alphabet of the Classical Propositional Logic* is the union of the following sets:

- set of logical constants: $\mathsf{Lc} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$
- set of propositional letters: $\mathsf{Var} = \{p_1, q_1, r_1, p_2, q_2, r_2, \dots\}$
- set of brackets: $\{\,), (\,\}$.

1 In the chapter we develop the approach that appeared for the first time in an English-language article [6]. In that study, both the concepts of tableau rules and the branches arising from the application of the rules were defined in a rather complicated way. In the meantime, these concepts have been simplified, so we present them here in an improved version. Of course, these changes affect the construction of the entire system.

Although the set of propositional letters is infinite and includes indexed propositional letters, in practice we will use a finite number of the following letters: p, q, r, s.

Definition 2.2 (Formula **CPL**)**.** *Set of formulas* of **CPL** is the smallest set X which meets conditions:

1. $\mathsf{Var} \subseteq X$
2. if $A, B \in X$, then
   a. $\neg A \in X$
   b. $(A \wedge B) \in X$
   c. $(A \vee B) \in X$
   d. $(A \to B) \in X$
   e. $(A \leftrightarrow B) \in X$.

We specify this set as $\mathsf{For}_{\mathbf{CPL}}$, and its elements will be called *formulas*.

Remark 2.3. When considering different languages, we will use separate denotations for the sets of their formulas — though in a given context, we will use formulas of only one logic — so there will be no risk of mistake. However, in the chapter devoted to the generalisation of our considerations, we will use a denotation with no index. Additionally, sometimes, if possible, we will omit external brackets.

Let us proceed now to the semantics of classical logic. *Valuation of propositional letters* Var is any function $v : \mathsf{Var} \longrightarrow \{1,0\}$, that to each propositional letter assigns a value of truth or false. With function v, we can define valuation of formulas of **CPL**.

Definition 2.4 (Valuation of formulas of **CPL**)**.** *Valuation of formulas* is function $V : \mathsf{For}_{\mathbf{CPL}} \longrightarrow \{1,0\}$, which for any $A, B \in \mathsf{For}_{\mathbf{CPL}}$ meets the following conditions:

1. $V(\neg A) = 1$ iff $V(A) = 0$
2. $V(A \wedge B) = 1$ iff $V(A) = 1$ and $V(B) = 1$
3. $V(A \vee B) = 1$ iff $V(A) = 1$ or $V(B) = 1$
4. $V(A \to B) = 1$ iff $V(A) = 0$ or $V(B) = 1$
5. $V(A \leftrightarrow B) = 1$ iff $V(A) = V(B)$.

We also call function V a *valuation of formulas of* **CPL** or shortly *valuation*.

Each function v, through conditions of definition 2.4, is unambiguously extendible to the valuation of formulas, i.e. function $V : \mathsf{For}_{\mathbf{CPL}} \longrightarrow \{1,0\}$ that to each formula assigns a value of truth or false.

Denotation 2.5. Let us adopt some abbreviations. Let '$V(X) = 1$' mean $V(X) \subseteq \{1\}$ (i.e. if $X \neq \varnothing$, then $V(X) = \{1\}$), while '$V(X) \neq 1$' means $V(X) \nsubseteq \{1\}$, (i.e. if $X \neq \varnothing$, then $V(X) \neq \{1\}$) for any set of formulas X and for any valuation V.

Using the concept of valuation of formulas, we can now define the semantic consequence of **CPL**.

Definition 2.6 (Semantic consequence of **CPL**)**.** Let set $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$. Formula A *follows from* set X (for short: $X \vDash A$) iff for any valuation V, if $V(X) = 1$, then $V(A) = 1$. Relation $\vDash$ will be called *relation of classical semantic consequence* or shortly *relation of semantic consequence*.

Thus, classical semantic consequence $\vDash$ is such a relation that $\vDash \subseteq P(\mathsf{For}_{\mathbf{CPL}}) \times \mathsf{For}_{\mathbf{CPL}}$.

Remark 2.7. When considering different relations of consequences in the following chapters, we will not use separate denotations for them. In a given context, we will only examine one relation, so there will be no risk of mistake.

Denotation 2.8. For any set of formulas X and any formula A notation $X \nvDash A$ will mean that it is not the case that $X \vDash A$.

The last concept related to **CPL**, we will apply when constructing the tableau system is the concept of contradictory set of formulas.

Definition 2.9 (Contradictory set of formulas)**.** Let $X \subseteq \mathsf{For}_{\mathbf{CPL}}$. Set X will be called *contradictory* iff there is no valuation V such that $V(X) = 1$. Set X will be called *non-contradictory* iff it is not contradictory.

2.3 Basic concepts of the tableau system for CPL

One of the basic concepts used to describe a tableau system, due to the nature of tableau proofs, is the concept of a tableau inconsistent set of proof expressions. In the case of a defined system for **CPL**, the proof expressions are the formulas themselves, so a set of tableau inconsistent expressions boils down to a certain set of formulas.

Definition 2.10 (Tableau inconsistent set of formulas)**.** Set $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ will be called *tableau inconsistent* (for short: *t-inconsistent*) iff there exists such formula $A \in \mathsf{For}_{\mathbf{CPL}}$ that $A, \neg A \in X$. Set X will be called *tableau consistent* (for short: *t-consistent*) iff it is not t-inconsistent.

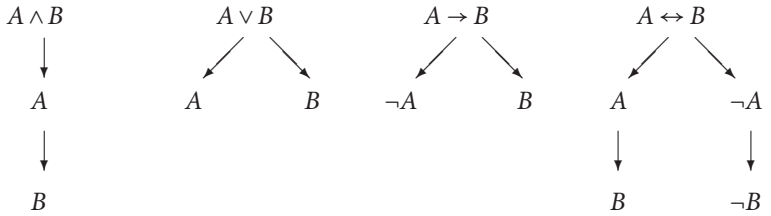
Corollary 2.11. *For any $X \subseteq \mathsf{For}_{\mathbf{CPL}}$, if X is t-inconsistent, then X is contradictory.*

Proof. By definition 2.10, 2.4, 2.9. □

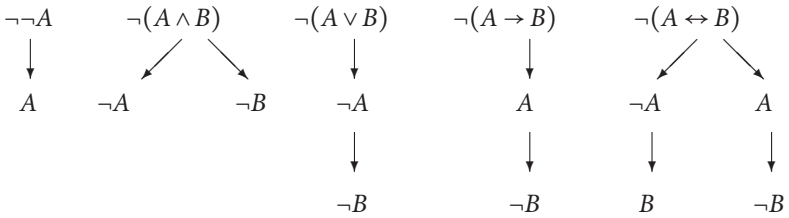
2.3.1 Tableau rules for CPL

We will now proceed to defining the proof tools. The central concept is tableau rules. Usually, in an intuitive approach, tableau rules are presented as graphs illustrating the way in which proof expressions are distributed. Rules for **CPL** are most often presented in the following way.[2]

First, we present positive rules — for formulas that do not begin with the negation functor.



The negative rules, on the other hand, for formulas preceded by negation, are presented in the following drawings.



In all of these rules, the arrows show the formulas we obtain by applying given rule, while some rules provide alternative formulas. Such a concept of rule is figurative and very intuitive, but at the same time not very formal. As a result, it is difficult to base on it to define the concept of tableau or tableau proof more formally than the concept of rule itself.

The starting point for the construction of a tableau system can therefore be a precise definition of the concept of tableau rule. Let us start with the general concept of rule.

2 See e.g. Priest G., [23], p. 5-6.

Definition 2.12 (Rule)**.** Let $P(\mathsf{For_{CPL}})$ be the set of all subsets of set of formulas. Let $P(\mathsf{For_{CPL}})^n$ be n-ry Cartesian product $\underbrace{P(\mathsf{For_{CPL}}) \times \ldots \times P(\mathsf{For_{CPL}})}_{n}$, for some $n \in \mathbb{N}$.

- By a *rule* we understand any such subset $R \subseteq P(\mathsf{For_{CPL}})^n$ that if $\langle X_1, \ldots, X_n \rangle \in R$, then $X_1 \subset X_i$, for each $1 < i \leq n$.
- If $n \geq 2$, each element R will be called *ordered n-tuple* (pair, triple, etc., respectively).
- The first element of each n-tuple will be called an *input set* (*set of premises*) while the other elements *output sets* (*sets of conclusions*).[3]

Thus, according to the above definition, each rule is for some $n \in \mathbb{N}$ a set of n-tuples in the form of $\langle X_1, \ldots, X_n \rangle$, where $X_1, \ldots, X_n \subseteq \mathsf{For_{CPL}}$. Not every rule is of course a tableau rule for **CPL**. A set of tableau rules for our tableau system for **CPL** shall be introduced by means of the following definition.

Definition 2.13 (Tableau rules for **CPL**)**.** *Tableau rules for* **CPL** are the following rules:

$R_{\wedge} = \{\langle X \cup \{(A \wedge B)\}, X \cup \{(A \wedge B), A, B\}\rangle : X \subseteq \mathsf{For_{CPL}}, A, B \in \mathsf{For_{CPL}}, X \cup \{(A \wedge B)\}$ is t-consistent$\}$

$R_{\vee} = \{\langle X \cup \{(A \vee B)\}, X \cup \{(A \vee B), A\}, X \cup \{(A \vee B), B\}\rangle : X \subseteq \mathsf{For_{CPL}}, A, B \in \mathsf{For_{CPL}}, X \cup \{(A \vee B)\}$ is t-consistent$\}$

$R_{\rightarrow} = \{\langle X \cup \{(A \rightarrow B)\}, X \cup \{(A \rightarrow B), \neg A\}, X \cup \{(A \rightarrow B), B\}\rangle : X \subseteq \mathsf{For_{CPL}}, A, B \in \mathsf{For_{CPL}}, X \cup \{(A \rightarrow B)\}$ is t-consistent$\}$

$R_{\leftrightarrow} = \{\langle X \cup \{(A \leftrightarrow B)\}, X \cup \{(A \leftrightarrow B), A, B\}, X \cup \{(A \leftrightarrow B), \neg A, \neg B\}\rangle : X \subseteq \mathsf{For_{CPL}}, A, B \in \mathsf{For_{CPL}}, X \cup \{(A \leftrightarrow B)\}$ is t-consistent$\}$

$R_{\neg\neg} = \{\langle X \cup \{\neg\neg A\}, X \cup \{\neg\neg A, A\}\rangle : X \subseteq \mathsf{For_{CPL}}, A \in \mathsf{For_{CPL}}, X \cup \{\neg\neg A\}$ is t-consistent$\}$

$R_{\neg\wedge} = \{\langle X \cup \{\neg(A \wedge B)\}, X \cup \{\neg(A \wedge B), \neg A\}, X \cup \{\neg(A \wedge B), \neg B\}\rangle : X \subseteq \mathsf{For_{CPL}}, A, B \in \mathsf{For_{CPL}}, X \cup \{\neg(A \wedge B)\}$ is t-consistent$\}$

3 The property of rule that the first set is contained properly in every subsequent set seems to correspond well to the name proposed in the literature *expansion rule* (p. 60, [4]).

$R_{\neg\vee} = \{\langle X \cup \{\neg(A \vee B)\}, X \cup \{\neg(A \vee B), \neg A, \neg B\}\rangle : X \subseteq \mathsf{For}_{\mathbf{CPL}}, A, B \in \mathsf{For}_{\mathbf{CPL}}, X \cup \{\neg(A \vee B)\} \text{ is t-consistent}\}$

$R_{\neg\to} = \{\langle X \cup \{\neg(A \to B)\}, X \cup \{\neg(A \to B), A, \neg B\}\rangle : X \subseteq \mathsf{For}_{\mathbf{CPL}}, A, B \in \mathsf{For}_{\mathbf{CPL}}, X \cup \{\neg(A \to B)\} \text{ is t-consistent}\}$

$R_{\neg\leftrightarrow} = \{\langle X \cup \{\neg(A \leftrightarrow B)\}, X \cup \{\neg(A \leftrightarrow B), \neg A, B\}, X \cup \{\neg(A \leftrightarrow B), A, \neg B\}\rangle : X \subseteq \mathsf{For}_{\mathbf{CPL}}, A, B \in \mathsf{For}_{\mathbf{CPL}}, X \cup \{\neg(A \leftrightarrow B)\} \text{ is t-consistent}\}$.

Set of tableau rules for **CPL** will be denoted as $\mathbf{R}_{\mathbf{CPL}}$.

By definition 2.13 it follows that we have nine tableau rules for the tableau system being defined for **CPL**. Let us consider one of them, e.g. $R_{\neg\wedge}$. Take any x and assume that $x \in R_{\neg\wedge}$. By definitions 2.12 and 2.13, we get $\exists_{X \subseteq \mathsf{For}_{\mathbf{CPL}}}, \exists_{A,B \in \mathsf{For}_{\mathbf{CPL}}}$ that:

1. $x = \langle X \cup \{\neg(A \wedge B)\}, X \cup \{\neg(A \wedge B), \neg A\}, X \cup \{\neg(A \wedge B), \neg B\}\rangle$
2. $X \cup \{\neg(A \wedge B)\} \subset X \cup \{\neg(A \wedge B), \neg A\}$
3. $X \cup \{\neg(A \wedge B)\} \subset X \cup \{\neg(A \wedge B), \neg B\}$
4. $X \cup \{\neg(A \wedge B)\}$ is t-consistent.

Therefore, there is such a set and such formulas that the rule contains the ordered triple of sets which was constructed on them and has the following properties:

(a) the first element — the input set — is a proper subset of both the second and third element — the output sets
(b) the first element is a t-consistent set.

Both these properties are important for the method of constructing tableau systems presented in the book. They also form an element which distinguishes this method from other ones. But before we proceed to a more detailed discussion of the properties, we will look at the notation of the rules itself. The provided method of notation is quite precise, but at the same time complicated. Thus, despite the fact that the case of **CPL** is the simplest of those considered in the book, the description of sets — rules is somewhat complicated.

In order to simplify the notation, we will propose a fractional one. With rule $R = \{\langle X_1, \ldots, X_n\rangle : X_1 \text{ is t-consistent}\}$, where $n \geq 2$, we will write:

$$\frac{X_1}{X_2 | \ldots | X_n}$$

Heuristically speaking, the fraction bar in the above notation will tell us that from the input set X_1 we can proceed to one of the output sets $X_2, \ldots, X_n$, described under the bar. In the fractional notation, the individual sets will also be described structurally, i.e. the fraction will be a scheme of infinitely many n-tuples, which are elements of rule R.

Remark 2.14. Further in the book, we will capture specific tableau rules with fractional notation only, but we will have introduced a general definition of the rule for a given type of tableau systems beforehand.

Now, using the above method of notation, we will once again present set of rules $\mathbf{R_{CPL}}$. Set of rules $\mathbf{R_{CPL}}$ contains only rules defined by the following schemes in which the input sets are t-consistent:

$$R_{\wedge}\colon \frac{X\cup\{(A\wedge B)\}}{X\cup\{(A\wedge B),A,B\}} \qquad R_{\vee}\colon \frac{X\cup\{(A\vee B)\}}{X\cup\{(A\vee B),A\}\,|\,X\cup\{(A\vee B),B\}}$$

$$R_{\rightarrow}\colon \frac{X\cup\{(A\rightarrow B)\}}{X\cup\{(A\rightarrow B),\neg A\}\,|\,X\cup\{(A\rightarrow B),B\}}$$

$$R_{\leftrightarrow}\colon \frac{X\cup\{(A\leftrightarrow B)\}}{X\cup\{(A\leftrightarrow B),A,B\}\,|\,X\cup\{(A\leftrightarrow B),\neg A,\neg B\}}$$

$$R_{\neg\neg}\colon \frac{X\cup\{\neg\neg A\}}{X\cup\{\neg\neg A,A\}}$$

$$R_{\neg\wedge}\colon \frac{X\cup\{\neg(A\wedge B)\}}{X\cup\{\neg(A\wedge B),\neg A\}\,|\,X\cup\{\neg(A\wedge B),\neg B\}}$$

$$R_{\neg\vee}\colon \frac{X\cup\{\neg(A\vee B)\}}{X\cup\{\neg(A\vee B),\neg A,\neg B\}} \qquad R_{\neg\rightarrow}\colon \frac{X\cup\{\neg(A\rightarrow B)\}}{X\cup\{\neg(A\rightarrow B),A,\neg B\}}$$

$$R_{\neg\leftrightarrow}\colon \frac{X\cup\{\neg(A\leftrightarrow B)\}}{X\cup\{\neg(A\leftrightarrow B),\neg A,B\}\,|\,X\cup\{\neg(A\leftrightarrow B),A,\neg B\}}$$

Remark 2.15. The method of tableau rules defining we have presented — no matter what notation method is adopted — carries some benefits. The first is elimination of redundant applications of rule that may occur in the case of intuitive formulation of a tableau system. An redundant application of rule takes place when application of rule does not cause a new expression to appear on

every branch it generates. In extreme cases, theoretically even intuitive rules can be applied without limitations, never receiving anything new. This is illustrated by the following example 2.16.

The rules defined according to our recipe exclude this type of situation, and the responsible factor is the condition that requires the input set to be a proper subset of each of the output sets. As per definition 2.12, for each rule R and $n \in \mathbb{N}$, if $\langle X_1, \ldots, X_n \rangle \in R$, then $X_1 \subset X_i$, for each $1 < i \leq n$. Therefore, no rule $R \in \mathbf{R_{CPL}}$ can be applied idly.

Example 2.16. Take formulas $(p \wedge q), \neg\neg r, p, q$. Using the intuitive rules described at the beginning of this subchapter 2.3.1, we can apply to these formulas many times the rule for conjunction to formula $(p \wedge q)$, although it does not bring anything new — in the meantime we have applied the rule for double negation to formula $\neg\neg r$.

$$
\begin{array}{c}
(p \wedge q), p, q, \neg\neg r \\
| \\
p \\
| \\
q \\
| \\
r \\
| \\
p \\
| \\
q
\end{array}
$$

It is difficult to formally limit such redundant use, since the rule is illustrated in the formula scheme, whereas the proof consists of many elements. Therefore, it is necessary to refer to all previous elements in a limitation. However, some of these elements may have appeared after the earlier application of other rules. Hence, it is unclear at which level in this approach to formally exclude such application of the tableau rules.

Remark 2.17. Another benefit of the way the rules are defined is the exclusion of their applicability to the sets that are t-inconsistent. Within the classic approach to rules, there is nothing to prevent us from continuing to apply the rule, even if there is an explicit tableau inconsistency in the proof tree. Of course, one way to avoid this problem is to add a blocking rule to the set of tableau rules, which introduces a new element which closes the branch. Again, however, nothing formally excludes the application of further rules, even if there is a blocking element in the proof.

We illustrate this problem with example 2.18. In our approach, the informal "no use" directive makes technical sense: it is simply not possible — in the case of

a t-inconsistent set, there are no rules that can be applied. Again, this is because of the rule definition. By definition 2.13 for each rule R and $n \in \mathbb{N}$, if $\langle X_1, \ldots, X_n \rangle \in R$, then X_1 is t-consistent, thus X_1 contains no pair of formulas A, $\neg A$, by definition 2.10. Consequently, no rule $R \in \mathbf{R_{CPL}}$ can be applied to a tableau inconsistent set.

Example 2.18. Take formulas $(p \wedge q)$, $\neg\neg\neg r$, p, r. Using the intuitive rules described at the beginning of this subchapter 2.3.1, we can apply the rule for conjunction to formula $(p \wedge q)$, even though previously we applied the rule for double negation to formula $\neg\neg\neg r$, which eventually produced a t-inconsistent set as we already had formula r.

$$
\begin{array}{c}
(p \wedge q),\ \neg\neg\neg r, p, r \\
| \\
\neg r \\
| \\
p \\
| \\
q
\end{array}
$$

Again, it is difficult to limit such application of rules if we do not have a precise concept of proof and rule.

Remark 2.19. Instead of set $\mathbf{R_{CPL}}$, we could use a different axiomatization to construct a tableau system. For instance, instead of rule $R_{\neg\wedge}$, we could consider the following tableau rule:

$$R'_{\neg\wedge}: \frac{X \cup \{\neg(A \wedge B)\}}{X \cup \{\neg(A \wedge B), (\neg A \vee \neg B)\}}$$

Similar proposals could be made for some other rules from set $\mathbf{R_{CPL}}$. Among many ideas, one of the simpler ones seems to be the replacement of rule $R_\vee$ e.g., the following one:

$$R'_\vee: \frac{X \cup \{(A \vee B)\}}{X \cup \{(A \vee B), (A \wedge B)\} | X \cup \{(A \vee B), A\} | X \cup \{(A \vee B), B\}}$$

Of course, the new rules — according to definition of tableau rule 2.13 — would also have t-consistent input sets, plus the input set would be contained properly in the output sets. Various axiomatizations can be examined for dependencies between sets of tableau rules, lengths of proofs and intuitiveness of rules.

For three reasons, however, we will not be researching such alternative tableau axiomatizations.

First of all, we want to describe the method of constructing tableau systems on the example of selected cases and then try to generalize this method on different cases. So in each case of the tableau systems considered in the book, it is sufficient to describe a single case, e.g. the most typical one. If we were to investigate an alternative axiomatization for a tableau system for **CPL**, e.g. using set $(\mathbf{R_{CPL}} \setminus \{R_{\neg\wedge}\}) \cup \{R'_{\neg\wedge}\}$, then in order to formally demonstrate that both systems yield the same logical consequences, it would require defining both systems and demonstrating that the sets of their consequences are identical. Meanwhile, we want to define a sample system for set of rules $\mathbf{R_{CPL}}$ and sample systems for logics specified in other languages, and then look for a general pattern for the construction of a tableau system that would simplify the construction of tableau systems and also help to study the equivalence of different sets of rules that axiomatize the same logic. So instead of describing consecutive tableau systems for the same semantically determined logic, we will try to seek common attributes of tableau systems for various semantically determined logic.

Secondly, the specified set of tableau rules $\mathbf{R_{CPL}}$ corresponds to the intuitive and normally adopted tableau rules for **CPL**. Thus, they are typical, because in the simplest way they correspond to the semantic content related to the interpretation of the classical connectives.

Thirdly, rules such as $R'_{\neg\wedge}$ do not seem good candidates for axiomatization. For example, the adoption of set $(\mathbf{R_{CPL}} \setminus \{R_{\neg\wedge}\}) \cup \{R'_{\neg\wedge}\}$, which is to replace rule $R_{\neg\wedge}$ with rule $R'_{\neg\wedge}$ would extend some of proofs. Because rule $R_{\neg\wedge}$ allows to proceed from formula $\neg(A \wedge B)$ to formulas $\neg A$ or $\neg B$. Meanwhile, in the case of rule $R'_{\neg\wedge}$ we proceed from formula $\neg(A \wedge B)$ to formula $(\neg A \vee \neg B)$. Transition to formulas $\neg A$ or $\neg B$ still requires the use of rule $R_{\vee}$. Thus, introduction of rule $R'_{\neg\wedge}$ in lieu of $R_{\neg\wedge}$ seems unnatural. It is not certain, however, that for the cases considered in the book there are no alternative rules that are not unnatural, or maybe even in some way they are better than the ones considered. In the next chapter, when describing the tableau system for term logic, we will show an example of alternative rules that seem at least equally good as the ones we will finally investigate there (see note 3.20).

However, among the reasons mentioned, the first one is conclusive. Therefore, although we will not examine alternative axiomatizations of tableau systems, it is worth stressing that the introduction of the general concept of rule 2.12 for a given set of tableau expressions Te is meaningful in overall. Most often it is possible to define different sets of tableau rules that define tableau systems corresponding to the same semantically determined logic.

2.3.2 Branches for CPL

Another thing in our theory that needs discussion is the concept of branch. It is a concept that depends on the tableau rule because branches are created by applying rules. Although we are not currently introducing any formal concept of *rule application*, intuitively the point is that a given rule R is applied to a given set Y when among the elements contained in R there is such n-tuple $\langle X_1, \ldots, X_n \rangle$ that $X_1 = Y$. The result of application of R is some set X_i, where $1 < i \leq n$, and consequently also sequence $\langle X_1, X_i \rangle$, which forms precisely a branch. Branches are therefore setwise objects consisting of sets. Let us now proceed to the formal definition of branch in the tableau system for **CPL**.

Definition 2.20 (Branch)**.** Let $K = \mathbb{N}$ or $K = \{1,2,\ldots,n\}$, where $n \in \mathbb{N}$. Let X be any set of formulas. *A branch* (or *a branch beginning with* X) will be called any sequence $\phi : K \longrightarrow P(\mathsf{For}_{\mathbf{CPL}})$ that meets the following conditions:

1. $\phi(1) = X$
2. for any $i \in K$: if $i+1 \in K$, then there exists such rule $R \in \mathbf{R}_{\mathbf{CPL}}$ and such n-tuple $\langle Y_1, \ldots, Y_n \rangle \in R$ that $\phi(i) = Y_1$ and $\phi(i+1) = Y_k$, for some $1 < k \leq n$.

Having two branches ϕ, ψ such that $\phi \subset \psi$ we shall state that:

- ϕ is a *sub-branch* of ψ
- ψ is a *super-branch* of ϕ.

Denotation 2.21. From now on — when speaking of branches — for convenience, we will use the following notations or designations:

1. $X_1, \ldots, X_n$, where $n \geq 1$
2. $\langle X_1, \ldots, X_n \rangle$, where $n \geq 1$
3. abbreviations: ϕ_M (where M is a domain ϕ, i.e. $\phi : M \longrightarrow P(\mathsf{For}_{\mathbf{CPL}})$)
4. or — to denote branches — small Greek letters: ϕ, ψ, etc.

The sets of branches, in turn, we shall denote with capital Greek letters: Φ, Ψ, etc. Furthermore, the domain cardinality of a given branch K we shall sometimes call a *length* of that branch.

Remark 2.22. As we can see, the concept of branch depends on some set of rules. In the case under consideration, the branch structure is based on the rules from set $\mathbf{R}_{\mathbf{CPL}}$. Further described complex tableau concepts will also depend on some sets of rules. Because in this chapter we are studying tableau system for **CPL**, based on rules from set $\mathbf{R}_{\mathbf{CPL}}$, so we are not going to make it any more complicated. In practice, however, the tableau concepts of systems constructed according to the presented idea always hinge upon some set of rules. In one of the further chapters,

in the general description of the construction method itself, the set of rules will be a certain variable. In this chapter, it is specified as: $\mathbf{R_{CPL}}$, and the complex tableau concepts defined here depend on it.

By definition of rules 2.12, through the fact that the rules are defined by proper containing of the output set in each of the output sets, in any n-tuple, there is a conclusion.

Corollary 2.23. *Each branch is an injective sequence.*

We will now look at the issue of branch length. To investigate this problem, we need a function to measure the formula complexity. But, this is not about a syntactic complexity, but branch complexity. In practice, the branch complexity of the formula boils down to the maximal length of the branch, which is obtained by applying the tableau rules to a single formula only. First, we will give a definition of the function that measures this property and show that the function has been well defined, and then we will present an example. Assume that for any two natural numbers n, m, $max\{n, m\} = n$ iff $n \geq m$.

Definition 2.24 (Measure of the branch complexity of formula of **CPL**)**.** *The measure of the branch complexity* is function $* : \mathsf{For}_{\mathbf{CPL}} \longrightarrow \mathbb{N}$, defined for each $x \in \mathsf{Var}$ and $A, B \in \mathsf{For}_{\mathbf{CPL}}$ with the below conditions:

1. $*(x) = 1$
2. $*(\neg x) = 1$
3. $*((A \wedge B)) = *(A) + *(B)$
4. $*((A \vee B)) = max\{*(A), *(B)\} + 1$
5. $*((A \rightarrow B)) = max\{*(\neg A), *(B)\} + 1$
6. $*((A \leftrightarrow B)) = max\{*(A) + *(B), *(\neg(A)) + *(\neg(B))\}$
7. $*(\neg\neg A) = *(A) + 1$
8. $*(\neg(A \wedge B)) = max\{*(\neg A), *(\neg B)\} + 1$
9. $*(\neg(A \vee B)) = *(\neg A) + *(\neg B)$
10. $*(\neg(A \rightarrow B)) = *(A) + *(\neg B)$
11. $*(\neg(A \leftrightarrow B)) = max\{*(\neg A) + *(B), *(A) + *(\neg B)\}$.

Function $*$, tells us how at most long a branch can be obtained by applying the rules from $\mathbf{R_{CPL}}$ to a given formula and its components. The following few facts clarify the problem. From the above definition and the definition of formula 2.2, it follows that:

Corollary 2.25. *For any $A \in \mathsf{For}_{\mathbf{CPL}}$: $*(A) \in \mathbb{N}$.*

We will now formulate a fact that describes the relationship between the branches and the measure of the branch complexity of formula.

Proposition 2.26. *Let $A \in \mathsf{For}_{\mathbf{CPL}}$. Let $X_1 = X \cup \{A\}, X_2, \ldots, X_n$ be such a branch that $Y_1, \ldots, Y_n$, where:*

- $Y_1 = \{A\}$
- $Y_2 = X_2 \setminus X$
- $\ldots$
- $Y_n = X_n \setminus X$

is a branch. Then $n \leq *(A)$.

Before we proceed to the proof of fact, let us try to explain its assumption. For each $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$ branch $X_1 = X \cup \{A\}, X_2, \ldots, X_n$ is constructed so that for each $1 < i \leq n$, $X_i = X \cup Y_i$. This means that X_2 was created by applying a certain rule to formula A, X_3 was created by applying a certain rule to formula from $X_2 \setminus X$, whereas X_{i+1} was created by applying a certain rule to formula from $X_i \setminus X$. So, to put it figuratively, the branch under consideration was created by applying the rules to formula A and the consequences of applying the rules to what was created previously. It is of course possible that at some point in time a certain rule may have been applied to a element of set X, but the effect must have been the same as applying the rule to A or its consequences.

Therefore, fact 2.26 tells us that a branch constructed through the application of rules to a single formula A and further effects of application of rules is not longer than $*(A)$, thus the measure of the branch complexity is properly defined.

Let us now move on to the proof of fact. It has an inductive nature and is based on the branch complexity of the formulas found in branches. For the proof, we will use the definition of branch 2.20 the definition of measure of branch complexity 2.24.

Proof. Take any branch $X_1 = X \cup \{A\}, X_2, \ldots, X_n$ that meets the theorem assumptions.

Initial step. Let $A = x$, for some $x \in \mathsf{Var}$. There is then no rule to apply to X_1, thus $n = 1$. Since $*(x) = 1$, so $n \leq *(x)$. If $A = \neg(x)$, for some $x \in \mathsf{Var}$, then there is either no rule to apply to X_1, so $n = 1$. Since $*(\neg x) = 1$, thus $n \leq *(\neg x)$.

Induction step. Assume that a fact thesis holds for each formula B such that $*(A) > *(B)$. So, for each branch beginning with $\{B\}$, m long, it is the case that $m \leq *(B)$. The branches will be called *B-branches*, *C-branches*, etc., depending on the formula they start with.

Cases:

i) Let $A := \neg\neg B$. Then, by definition of function $*$, $*(A) > *(B)$. Under the induction hypothesis $*(B) \geq m$, where m is the length of any B–branch. Since $n = m_1 + 1$, for some $m_1 \leq *(B)$, by definition of branch 2.20, so $*(A) = *(B) + 1 \geq m_1 + 1 = n$, by definition 2.24.

ii) Let $A := (B \wedge C)$. Then, by definition of function $*$, $*(A) > *(B), *(C)$. Under the induction hypothesis, $*(B) \geq m$, where m is the length of any B-branch, and $*(C) \geq k$, where k is the length of any C-branch. Consequently, since $n \leq m_1 + k_1$, for some $m_1 \leq *(B)$, $k_1 \leq *(C)$, by definition of branch 2.20, so $*(A) = *((B \wedge C)) = *(B) + *(C) \geq m_1 + k_1 \geq n$, by definition 2.24.

iii) Let $A := (B \vee C)$. Then, by definition of function $*$, $*(A) > *(B)$, $*(C)$. Under the induction hypothesis, $*(B) \geq m$, where m is the length of any B-branch, and $*(C) \geq k$, where k is the length of any C-branch. Consequently, since $n = m_1 + 1$ or $n = k_1 + 1$, for some $m_1 \leq *(B)$, $k_1 \leq *(C)$, by definition of branch 2.20, so:

1. if $max\{*(B), *(C)\} = *(B)$, then $*(A) = *((B \vee C)) = *(B) + 1 \geq m_1 + 1 = n$, by definition 2.24
2. if $max\{*(B), *(C)\} = *(C)$, then $*(A) = *((B \vee C)) = *(C) + 1 \geq k_1 + 1 = n$, by definition 2.24.

iv) Let $A := (B \rightarrow C)$. Then, by definition of function $*$, $*(A) > *(\neg B), *(C)$. Under the induction hypothesis, $*(\neg B) \geq m$, where m is the length of any $\neg B$-branch, and $*(C) \geq k$, where k is the length of any C-branch. Consequently, since $n = m_1 + 1$ or $n = k_1 + 1$, for some $m_1 \leq *(\neg(B))$, $k_1 \leq *(C)$, by definition of branch 2.20, so:

1. if $max\{*(\neg B), *(C)\} = *(\neg B)$, then $*(A) = *((B \rightarrow C)) = *(\neg(B)) + 1 \geq m_1 + 1 = n$, by definition 2.24
2. if $max\{*(\neg B), *(C)\} = *(C)$, then $*(A) = *((B \rightarrow C)) = *(C) + 1 \geq k_1 + 1 = n$, by definition 2.24.

v) Let $A := (B \leftrightarrow C)$. Then, by definition of function $*$, $*(A) > *(B)$, $*(C)$, $*(\neg B), *(\neg C)$. Under the induction hypothesis, $*(B) \geq m$, $*(C) \geq k$, $*(\neg B) \geq l$, $*(\neg C) \geq o$, where m, k, l, o are respectively lengths of any B, C, $\neg B$ and $\neg C$-branch. Consequently, since $n \leq m_1 + k_1$ or $n \leq l_1 + o_1$, for some $m_1 \leq *(B)$, $k_1 \leq *(C)$, $l_1 \leq *(\neg B)$, $o_1 \leq *(\neg C)$, by definition of branch 2.20, so:

1. if $max\{*(B) + *(C), *(\neg B) + *(\neg C)\} = *(B) + *(C)$, then $*(A) = *((B \leftrightarrow C)) = *(B) + *(C) \geq m_1 + k_1 \geq n$, by definition 2.24

2. if $max\{*(B)+*(C), *(\neg B)+*(\neg C)\} = *(\neg B)+*(\neg C)$, then $*(A) = *(B \leftrightarrow C) = *(\neg B)+*(\neg C) \geq l_1 + o_1 \geq n$, by definition 2.24.

vi) Let $A := \neg(B \wedge C)$. Then, by definition of function $*$, $*(A) > *(\neg B), *(\neg C)$. Under the induction hypothesis, $*(\neg B) \geq m$, where m is the length of any $\neg B$-branch, and $*(\neg C) \geq k$, where k is the length of any $\neg C$-branch. Consequently, since $n = m_1 + 1$ or $n = k_1 + 1$, for some $m_1 \leq *(\neg B)$, $k_1 \leq *(\neg C)$, by definition of branch 2.20, so:

1. if $max\{*(\neg B), *(\neg C)\} = *(\neg B)$, then $*(A) = *(\neg(B \wedge C)) = *(\neg B) + 1 \geq m_1 + 1 = n$, by definition 2.24
2. if $max\{*(\neg B), *(\neg C)\} = *(\neg C)$, then $*(A) = *(\neg(B \wedge C)) = *(\neg C) + 1 \geq k_1 + 1 = n$, by definition 2.24.

vii) Let $A := \neg(B \vee C)$. Then, by definition of function $*$, $*(A) > *(\neg B), *(\neg C)$. Under the induction hypothesis, $*(\neg B) \geq m$, where m is the length of any $\neg B$-branch, and $*(\neg C) \geq k$, where k is the length of any $\neg C$-branch. Consequently, since $n \leq m_1 + k_1$, for some $m_1 \leq *(\neg B)$, $k_1 \leq *(\neg C)$, by definition of branch 2.20, so $*(A) = *(\neg(B \vee C)) = *(\neg B) + *(\neg C) \geq m_1 + k_1 \geq n$, by definition 2.24.

viii) Let $A := \neg(B \rightarrow C)$. Then, by definition of function $*$, $*(A) > *(B), *(\neg C)$. Under the induction hypothesis, $*(B) \geq m$, where m is the length of any B-branch, and $*(\neg C) \geq k$, where k is the length of any $\neg C$-branch. Consequently, since $n \leq m_1 + k_1$, for some $m_1 \leq *(B)$, $k_1 \leq *(\neg C)$, by definition of branch 2.20, so $*(A) = *(\neg(B \rightarrow C)) = *(B) + *(\neg C) \geq m_1 + k_1 \geq n$, by definition 2.24.

ix) Let $A := \neg(B \leftrightarrow C)$. Then, by definition of function $*$, $*(A) > *(B), *(C), *(\neg B), *(\neg C)$. Under the induction hypothesis, $*(B) \geq m$, $*(C) \geq k$, $*(\neg B) \geq l$, $*(\neg C) \geq o$, where m, k, l, o are respectively lengths of any B, C, $\neg B$ and $\neg C$-branch. Consequently, since $n \leq m_1 + o_1$ or $n \leq l_1 + k_1$, for some $m_1 \leq *(B)$, $k_1 \leq *(C), l_1 \leq *(\neg B)$, $o_1 \leq *(\neg C)$, by definition of branch 2.20, so:

1. if $max\{*(\neg B)+*(C), *(B)+*(\neg C)\} = *(\neg B)+*(C)$, then $*(A) = *(\neg(B \leftrightarrow C)) = *(\neg B)+*(C) \geq l_1 + k_1 \geq n$, by definition 2.24
2. if $max\{*(\neg B)+*(C), *(B)+*(\neg C)\} = *(B)+*(\neg C)$, then $*(A) = *(\neg(B \leftrightarrow C)) = *(B)+*(\neg C) \geq m_1 + o_1 \geq n$, by definition 2.24. □

So the fact 2.26 tells us that if we create a branch based on a single A and make use of tableau rules in any order, then each produced branch will at most have a length of $*(A)$. See the example below.

Example 2.27. Let us consider set of formulas $Y \cup \{\neg(p \leftrightarrow \neg q)\}$. We decompose the highlighted formula using rule $R_{\neg\leftrightarrow}$ to this set. Due to two initial sets in the rule, we get two branches:

1. $X_1' = Y \cup \{\neg(p \leftrightarrow \neg q)\}, X_2' = Y \cup \{\neg(p \leftrightarrow \neg q), \neg p, \neg q\}$
2. $X_1'' = Y \cup \{\neg(p \leftrightarrow \neg q)\}, X_2'' = Y \cup \{\neg(p \leftrightarrow \neg q), p, \neg\neg q\}$.

In the first case, we can no longer decompose the components of the initial formula, but in the case of X_2'' we can still apply rule $R_{\neg\neg}$, which produces: $X_3'' = Y \cup \{\neg(p \leftrightarrow \neg q), p, \neg\neg q, q\}$. (Of course, we assume that X_1'', X_2'', are t-consistent. Otherwise, we could not use the tableau rules.) The last sequence — X_1'', X_2'', X_3'' — is the longest among the sequences based on the decomposition of formula $\neg(p \leftrightarrow \neg q)$. At most, is has length of $*(\neg(p \leftrightarrow \neg q))$. Let us calculate: $*(p) = *(\neg q) = *(\neg p) = 1$, $*(\neg\neg q) = 1 + *(q) = 2$. Thus $max\{*(\neg p) + *(\neg q), *(p) + *(\neg\neg q)\} = max\{2,3\}$. Hence $*(\neg(p \leftrightarrow \neg q)) = 3$.

Another fact generalizes our observations on the relationship between the length of the branch and the branch complexity of the formula on the finite sets of formulas.

Proposition 2.28. *If X is a finite set of formulas, then each such branch $\phi : K \longrightarrow P(For_{CPL})$ that $\phi(1) = X$ is finite.*

We will carry out an inductive proof with respect to the number of elements X by applying the previous fact 2.26.

Proof. Let X be a finite set of formulas, and $\phi : K \longrightarrow P(\mathsf{For_{CPL}})$ any such branch that $\phi(1) = X$.

Initial step. Assume that $|X| = 1$, so some formula $A \in X$. From the previous fact 2.26, we know that each branch based on decomposition of A has at most the length of $*(A)$. Hence, $|K| \leq *(A)$, so branch ϕ is finite, under 2.25.

Induction step. Assume that the fact thesis is true for each such set of formulas Y that $|Y| = n$. Let us consider a situation where $|X| = n + 1$. So $X = Y' \cup \{A\}$ for certain set of formulas Y' such that $|Y'| = n$, and some new formula A.

From fact 2.26, we know that each branch based on the decomposition of A has at most the length of $*(A)$, which means it is finite. From the inductive assumption, we also know that each branch beginning with set of formulas Y' is finite.

Consequently, each branch ϕ beginning with $X = Y' \cup \{A\}$ is finite as it is composed of elements of some branch which begins with A and elements of some branch which begins with Y', by definition of branch 2.20, hence $|K| \leq n + *(A)$. □

By virtue of the last fact, we know that there are no infinite branches beginning with finite sets of formulas, which is expressed by the following conclusion.

Corollary 2.29. *Let X be a finite set of formulas. Then there is no such branch $\phi : K \longrightarrow P(\mathsf{For}_{\mathbf{CPL}})$ that:*

- $K = \mathbb{N}$
- $\phi(1) = X$.

Proof. From the previous fact 2.28 and by definition of branch 2.20. □

2.3.3 Maximal branches

Another important concept in the construction of a tableau system is the concept of a maximal branch. Intuitively, the maximal branch is a branch to which no rule can be applied anymore, extending it to some super-branch. The definition of the maximal branch for the currently defined system is as follows.

Definition 2.30 (Maximal branch)**.** Let $\phi : K \longrightarrow P(\mathsf{For}_{\mathbf{CPL}})$ be a branch. We shall state that ϕ is *maximal* iff

1. $K = \{1, 2, 3, \ldots, n\}$, for some $n \in \mathbb{N}$
2. there is no branch ψ such that $\phi \subset \psi$.

Before we discuss this definition, let us consider an example.

Example 2.31. Consider a branch beginning with set $X_1 = \{\underbrace{\neg \ldots \neg}_{n} r : n = m \cdot 3,$ where $m \in \mathbb{N}\}$. Set X_1 contains an infinite number of formulas in the form of $\underbrace{\neg \ldots \neg}_{n=m\cdot 3} r$, i.e. an infinite number of instances of propositional letter r, preceded in each instance by such number of instances of negation functor that is a multiple of 3.

We now define a branch based on set X_1 and rule $R_{\neg\neg}$, according to the following algorithm: for any $m \in \mathbb{N}$,

$$X_{m+1} = X_m \cup \{\underbrace{\neg \ldots \neg}_{m\cdot 3-2} r\}.$$

So, transition from set X_i to its superset X_{i+1} is an effect of addition through rule $R_{\neg\neg}$ to set X_i of formula $\underbrace{\neg \ldots \neg}_{i\cdot 3-2} r$.

The branch can be illustrated as follows:

$$
\begin{array}{c}
R_{\neg\neg} X_1 \\
| \\
R_{\neg\neg} X_2 = X_1 \cup \{\neg r\} \\
| \\
R_{\neg\neg} X_3 = X_2 \cup \{\neg\neg\neg r\} \\
| \\
R_{\neg\neg} X_4 = X_3 \cup \{\neg\neg\neg\neg\neg r\} \\
| \\
R_{\neg\neg} \ldots
\end{array}
$$

The defined branch is infinite. There is no super-branch to contain it. However, there is still an infinite number of formulas to which we might apply rule $R_{\neg\neg}$. As a matter of fact, already in the area of set X_4 we could obtain t-contradictory set, thereby closing the door on further extension of branches, defining $X_4 = X_3 \cup \{\neg\neg r\}$, with respect to $R_{\neg\neg}$, since $\neg r \in X_2 \subseteq X_3$.

Seemingly, we could limit the definition of maximal branch 2.30 to the second condition, i.e. to the non-existence of super-branch. However, we would then allow branches that contain non-decomposed expressions to which the tableau rules can still be applied.

Leaving out the first condition of the definition of maximal branch 2.30, would not change anything with respect to the cases of finite sets, but example 2.31 proves that we would allow cases of infinite branch that:

- begin with an infinite set
- meet the second condition of definition because they are infinite branches
- but not all expressions contained were decomposed, in particular those responsible for the emergence of t-inconistent subset.

Remark 2.32. The definition of maximal branch 2.30 is suitable for those tableau systems where only finite branches are obtained from finite sets of expressions. So it is i.a. good for the tableau system we construct for **CPL**.

For other systems, including modal logics, the definition is too narrow, because it does not include cases of branches that are not finite even though they start with a finite set of expressions.

We have now deliberately adopted definition 2.30, as sufficient for **CPL**. When we move on to defining the system for modal logic, we will generalize this definition. So it is going to describe special cases of infinite maximal branches which appear in the construction of tableau systems using the described method.

Extending the concept of branch onto infinite sets is problematic for many reasons, and what is more, it is unnecessary in practice, because important and sufficient concepts for our metatheory — the concept of branch consequences and tableau — we apply in practice to cases of finite sets.

Besides, we showed that a finite sets of expressions, using the rules from set of rules $\mathbf{R_{CPL}}$ we get branches of finite length (conclusion 2.29), so for the case of **CPL** the definition of maximal branch 2.30 is good enough.

The definition of maximal branch 2.30 in practice says that branch $X_1, \ldots, X_n$, for some $n \geq 1$, we shall call *maximal* iff there is no branch $X_1, \ldots, X_n, X_{n+1}$.

In the subsequent fact, we state that each finite set of formulas is the first element of some maximal branch.

Proposition 2.33. *Let X be a finite set of formulas. Then, there exists such maximal branch $X_1, \ldots, X_n$ that $X_1 = X$ and $n \geq 1$.*

Proof. Take any finite set of formulas X, and then indirectly assume that there is no maximal branch $X_1, \ldots, X_n$, where $X_1 = X$ and $n \geq 1$.

However, by definition of branch 2.20 we know that there exists at least one branch that starts with set X, i.e. $\langle X_1 \rangle$, where $X_1 = X$. From the indirect assumption and from the definition of maximal branch 2.30 it follows, however, that $\langle X_1 \rangle$ is not a maximal branch and it has some super-branch, so by definition of branch 2.20 there is some branch: X_1, X_2.

Let us now consider branch $X_1, \ldots, X_m$ m long, for some $m \in \mathbb{N}$. From the indirect assumption and from the definition of maximal branch 2.30 it follows, however, that $X_1, \ldots, X_m$ is not a maximal branch and it has some super-branch, so by definition of branch 2.20 there is some branch: $X_1, \ldots, X_m, X_{m+1}$.

So, from the indirect assumption and from the definition of maximal branch 2.30 results in a conclusion that (†) for any $n \in \mathbb{N}$ there exists branch $X_1, \ldots, X_n$, X_{n+1} such that $X_1 = X$ and $n \geq 1$.

Let Φ be such a minimal set of branches that:

1. $\langle X_1 \rangle \in \Phi$
2. if $\langle X_1, \ldots, X_n \rangle \in \Phi$, then $\langle X_1, \ldots, X_n, X_{n+1} \rangle \in \Phi$, for any $n \in \mathbb{N}$.

Since Φ is a minimal set that meets conditions 1 and 2, then from (†) it follows that for any $n \in \mathbb{N}$, there exists precisely one branch $\langle X_1, \ldots, X_n \rangle \in \Phi$, where $X_1 = X$.

K_i will now denote a domain of such branch ϕ_K contained in Φ that $|K| = i \in \mathbb{N}$. From the above considerations it follows that for each $i \in \mathbb{N}$ there exists precisely one set K_i that constitutes a domain of some branch which belongs to Φ. Then let $\overline{K} = \bigcup \{K_i : \phi_K \in \Phi\}$. Since $1 \in \overline{K}$ and $n+1 \in \overline{K}$, if $n \in \overline{K}$, so $\overline{K} = \mathbb{N}$.

Let us now define sequence $\psi : \overline{K} \longrightarrow P(\mathsf{For_{CPL}})$ such that for any $i \in K$: $\psi(i) = \phi_{K_i}(i)$. From the definition of branch 2.20, that sequence is an infinite branch beginning with a finite set of formulas $X_1 = X$, which contradicts conclusion 2.29. □

Using the above fact, we can prove another fact important for our theory.

Proposition 2.34. *If X is a finite set of formulas, then for each branch $Y_1, \ldots, Y_n$ such that $Y_1 = X$ and $n \geq 1$, there exists maximal branch $Z_1, \ldots, Z_n, \ldots, Z_{n+m}$, where for any $i \leq n$ $Y_i = Z_i$ and $m \geq 0$.*

Proof. Let X be a finite set of formulas. Now, take any branch $Y_1, \ldots, Y_n$ such that $Y_1 = X$ and $n \geq 1$. From the definition of branch 2.20 and definition of tableau rules for **CPL** 2.13, we know that Y_n is a finite set. So, due to the last fact 2.33, for some $m \geq 0$ there exists such maximal branch $Z_n^1, \ldots, Z_{n+m}^{1+m}$ that $Z_n^1 = Y_n$, hence by virtue of definition of branch 2.20 there also exists maximal branch $Z_1, \ldots, Z_n, \ldots, Z_{n+m}$, where for any $i \leq n$ $Y_i = Z_i$ and $m \geq 0$. □

This fact tells us that any branch beginning with a finite set of formulas can be extended to the maximal branch in which it is included as a sequence.

For further consideration, the concept of branch which is maximal in a given set of branches, will be useful.

Definition 2.35 (Maximal branch in the set of branches)**.** Let Φ be a set of branches and let branch $\psi \in \Phi$. Branch ψ will be called *maximal in* Φ (or Φ–*maximal*) iff there is no such branch $\phi \in \Phi$ that $\psi \subset \phi$. Having some set of formulas X, $B(X)$ will denote the set of all branches $X_1, \ldots, X_n$ such that $X_1 = X$ and $n \geq 1$, while $MB(X)$ will denote the set of all maximal branches in $B(X)$.

Remark 2.36. The above concept of a maximal branch in a certain set of branches could be a starting point instead of the concept described in the definition of maximal branch 2.30. The latter is its special case in a situation when set Φ is identical to set $B(X)$, for some finite set of formulas X, considering fact 2.33.

However, we deliberately separated these concepts, because in one of the subsequent chapters we will change the definition of maximal branch in such a way that its scope will not be identical to the scope of definition 2.35 in relation to finite sets. This new concept of a maximal branch will also include cases where branches can be infinitely long, even though they start with a finite set of expressions. So, consequently, it can be the case that for a given branch ϕ there is no such branch ψ that $\phi \subset \psi$, and yet ϕ it will not be considered maximal for other reasons. Although we will continue to use the concept of a maximal branch in a given set, we will eventually extend the concept of a maximal branch.

Corollary 2.37. *Let X be a finite set of formulas. Then $B(X)$ contains a non-empty subset $MB(X)$.*

Proof. Take any finite set of formulas X. Due to fact 2.34, there exists maximal branch ψ beginning with X. Of course, branch ψ belongs to $MB(X)$, so set $MB(X)$ is non-empty. Since each branch contained in $MB(X)$ belongs to $B(X)$, so non-empty set $MB(X)$ is contained in $B(X)$. □

2.3.4 Closed and open branches

Another concept important for the tableau theory is the concept of a closed branch and an open branch. Intuitively, a branch is closed when we have reached a t-inconsistency set by decomposing formulas.

Definition 2.38 (Closed/open branch)**.** Branch $\phi : K \longrightarrow P(\mathsf{For}_{\mathbf{CPL}})$ will be called *closed* iff $\phi(i)$ is a t-inconsistent set, for some $i \in K$. A branch will be called *open* iff it is not closed.

From the above definition, the definition of tableau rules for **CPL** 2.13 and the definition of branch 2.20, the following conclusion follows.

Corollary 2.39. *If branch* $\phi : K \longrightarrow P(\mathsf{For}_{\boldsymbol{CPL}})$ *is closed, then* $|K| \in \mathbb{N}$.

Proof. Let branch $\phi : K \longrightarrow P(\mathsf{For}_{\mathbf{CPL}})$ be closed. From the definition of closed branch 2.38 we know that there exists such $i \in K$ that $\phi(i)$ is a t-inconsistent set. From the definition of tableau rules for **CPL** it follows that there is no branch element $\phi(i+1)$, since the rules do not contain n-tuples with t-inconsistent input sets, hence none of the rules can be applied to set $\phi(i)$. So, from the definition of branch 2.20 we get: $K = \{1, 2, 3, \ldots i\}$, which means that $|K| \in \mathbb{N}$. □

In the case of a closed branch, the t-inconsistent element of sequence is therefore the last element. It is so because no more rule can be applied for branch extensions, because the tableau rules are defined in such a way that they cannot be applied to t-inconsistent sets. Therefore, from the definition of maximal branch 2.30 another conclusion follows.

Corollary 2.40. *If branch* $\phi : K \longrightarrow P(\mathsf{For}_{\boldsymbol{CPL}})$ *is closed, then it is maximal.*

2.3.5 Branch consequence relation

We will now proceed to the metalogical concept which occurs in each of the systems defined in the book. This concept seems to be a novelty, seemingly so far not defined in the studies on tableau systems.

As we wrote, the tableau method is treated and defined in the book in a purely syntactic manner, i.e. as a method of transforming the notations of a given language in order to answer the question whether the considered inference is correct.

A concept that corresponds to this question is the concept of branch consequence relation, specified with another definition.

Definition 2.41 (Branch consequence relation of **CPL**)**.** Let $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$. We shall state that A is a *branch consequence* of X (or short: $X \rhd A$) iff there

exists such finite set $Y \subseteq X$ that each maximal branch beginning with $Y \cup \{\neg A\}$ is closed. Relation $\triangleright$ will be called *branch consequence relation* of **CPL**.

Remark 2.42. When considering different branch consequence relations in the following chapters, we will not use separate denotations for them. In a given context, we will only examine one relation, so there will be no risk of mistake.

Denotation 2.43. For any set of formulas X and any formula A, notation $X \not\triangleright A$ shall mean that it is not the case that $X \triangleright A$.

In order to explain what is meant by definition 2.41, we will refer to an example.

Example 2.44. Let us consider the following example of occurrence of the relation of branch consequence. Take set $\{p\}$ and formula $(p \vee q)$. Relation $\{p\} \triangleright (p \vee q)$ occurs since each maximal branch in the form $X_1 = \{p, \neg(p \vee q)\}, \ldots, X_n$ is closed. In fact, there is only one maximal branch in this form. It is branch: $X_1 = \{p, \neg(p \vee q)\}$, $X_2 = \{p, \neg(p \vee q), \neg p, \neg q\}$, which originates through rule $R_{\neg\vee}$. It is closed because $p \in X_2$ and $\neg p \in X_2$.

The above example shows the mechanism of the relation of branch consequence. It is based on the impossibility of finding a branch that would be both maximal and open. However, this example is somewhat misleading. We deliberately chose a set of formulas and a formula so that the branches beginning with this set and negation of the formula are neither too long nor too many. In practice, none of these features may actualize and most often they do not.

2.4 Relations of semantic consequence and branch consequence

Before we proceed to the issue of how to reduce as much as possible the number of branches, which suffice to consider to show the occurrence of the relation of branch consequence, we will first show that the concept of the branch consequence relation defines the same set of objects as the concept of the semantic consequence relation — it is thus the branch, syntactic counterpart of the semantic consequence relation.

2.4.1 Soundness theorem

In this subchapter, we will show that the relation of branch consequence is contained in the relation of semantic consequence, hence for any set of formulas X and any formula A it is the case that if $X \triangleright A$, then $X \vDash A$.

We will use abbreviations concerning the valuations of the formulas adopted in denotation 2.5.

We will start with a lemma that tells us about the relationship between valuations of formulas and tableau rules.

Lemma 2.45. *Let X be a set of formulas and V be any valuation such that $V(X) = 1$. For any rule $R \in \boldsymbol{R_{CPL}}$:*

- *if $\langle X, Y\rangle \in R$, then $V(Y) = 1$*
- *if $\langle X, Y, Z\rangle \in R$, then $V(Y) = 1$ or $V(Z) = 1$.*

Proof. We carry out the proof by checking the rules one by one. We will use the definition of valuation of **CPL** formulas, 2.4.

Take any set of formulas X and any valuation V such that $V(X) = 1$. Take any rule R from set $\mathbf{R_{CPL}}$. If set X is an input set of some *n – tuple* contained in R, there must exist: certain set of formulas X' and formulas A, B such that there occurs at least on of the below cases.

- $X = X' \cup \{(A \wedge B)\}$, $V(X' \cup \{(A \wedge B)\}) = 1$, $R = R_{\wedge}$ and $\langle X' \cup \{(A \wedge B)\}, X' \cup \{(A \wedge B), A, B\}\rangle \in R_{\wedge}$. Since $V((A \wedge B)) = 1$, so $V(A) = V(B) = 1$. Thus $V(X' \cup \{(A \wedge B), A, B\}) = 1$.
- $X = X' \cup \{(A \vee B)\}$, $V(X' \cup \{(A \vee B)\}) = 1$, $R = R_{\vee}$ and $\langle X' \cup \{(A \vee B)\}, X' \cup \{(A \vee B), A\}, X' \cup \{(A \vee B), B\}\rangle \in R_{\vee}$. Since $V((A \vee B)) = 1$, so $V(A) = 1$ or $V(B) = 1$. Thus $V(X' \cup \{(A \vee B), A\}) = 1$ or $V(X' \cup \{(A \vee B), A\}) = 1$.
- $X = X' \cup \{(A \to B)\}$, $V(X' \cup \{(A \to B)\}) = 1$, $R = R_{\to}$ and $\langle X' \cup \{(A \to B)\}, X' \cup \{(A \to B), \neg A\}, X' \cup \{(A \to B), B\}\rangle \in R_{\to}$. Since $V((A \to B)) = 1$, so $V(A) = 0$ or $V(B) = 1$. Hence $V(\neg A) = 1$ or $V(B) = 1$. Thus $V(X' \cup \{(A \to B), \neg A\}) = 1$ or $V(X' \cup \{(A \to B), B\}) = 1$.
- $X = X' \cup \{(A \leftrightarrow B)\}$, $V(X' \cup \{(A \leftrightarrow B)\}) = 1$, $R = R_{\leftrightarrow}$ and $\langle X' \cup \{(A \leftrightarrow B)\}, X' \cup \{(A \leftrightarrow B), A, B\}, X' \cup \{(A \leftrightarrow B), \neg A, \neg B\}\rangle \in R_{\leftrightarrow}$. Since $V((A \leftrightarrow B)) = 1$, so $V(A) = V(B)$. Hence $V(A) = V(B) = 1$ or $V(A) = V(B) = 0$. Thus $V(A) = V(B) = 1$ or $V(\neg A) = V(\neg B) = 1$. Consequently $V(X' \cup \{(A \leftrightarrow B), A, B\}) = 1$ or $V(X' \cup \{(A \leftrightarrow B), \neg A, \neg B\}) = 1$.
- $X = X' \cup \{\neg\neg A\}$, $V(X' \cup \{\neg\neg A\}) = 1$, $R = R_{\neg\neg}$ and $\langle X' \cup \{\neg\neg A\}, X' \cup \{\neg\neg A, A\}\rangle \in R_{\neg\neg}$. Since $V(\neg\neg A) = 1$, so $V(\neg A) = 0$, and $V(A) = 1$. Thus $V(X' \cup \{\neg\neg A, A\}) = 1$.
- $X = X' \cup \{\neg(A \wedge B)\}$, $V(X' \cup \{\neg(A \wedge B)\}) = 1$, $R = R_{\neg\wedge}$ and $\langle X' \cup \{\neg(A \wedge B)\}, X' \cup \{\neg(A \wedge B), \neg A\}, X' \cup \{\neg(A \wedge B), \neg B\}\rangle \in R_{\neg\wedge}$. Since $V(\neg(A \wedge B)) = 1$, so $V((A \wedge B)) = 0$. Hence $V(A) = 0$ or $V(B) = 0$, and consequently $V(\neg A) = 1$ or $V(\neg B) = 1$. Thus $V(X' \cup \{(A \wedge B), \neg A\}) = 1$ or $V(X' \cup \{(A \wedge B), \neg B\}) = 1$.
- $X = X' \cup \{\neg(A \vee B)\}$, $V(X' \cup \{\neg(A \vee B)\}) = 1$, $R = R_{\neg\vee}$ and $\langle X' \cup \{\neg(A \vee B)\}, X' \cup \{\neg(A \vee B), \neg A, \neg B\}\rangle \in R_{\neg\vee}$. Since $V(\neg(A \vee B)) = 1$, so $V((A \vee B)) = 0$, thus $V(A) = V(B) = 0$, and consequently $V(\neg A) = V(\neg B) = 1$. Thus $V(X' \cup \{\neg(A \vee B), \neg A, \neg B\}) = 1$.

- $X = X' \cup \{\neg(A \to B)\}$, $V(X' \cup \{\neg(A \to B)\}) = 1$, $R = R_{\neg\to}$ and $\langle X' \cup \{\neg(A \to B)\}, X' \cup \{\neg(A \to B), A, \neg B\}\rangle \in R_{\neg\to}$. Since $V(\neg(A \to B)) = 1$, so $V((A \to B)) = 0$, so $V(A) = 1$ and $V(B) = 0$. Thus $V(A) = 1$ and $V(\neg B) = 1$. Consequently $V(X' \cup \{\neg(A \to B), A, \neg B\}) = 1$.
- $X = X' \cup \{\neg(A \leftrightarrow B)\}$, $V(X' \cup \{\neg(A \leftrightarrow B)\}) = 1$, $R = R_{\neg\leftrightarrow}$ and $\langle X' \cup \{\neg(A \leftrightarrow B)\}, X' \cup \{\neg(A \leftrightarrow B), \neg A, B\}, X' \cup \{\neg(A \leftrightarrow B), A, \neg B\}\rangle \in R_{\neg\leftrightarrow}$. Since $V(\neg(A \leftrightarrow B)) = 1$, so $V(A) \neq V(B)$. Hence $V(A) = 0$ and $V(B) = 1$ or $V(A) = 1$ and $V(B) = 0$. Thus $V(\neg A) = V(B) = 1$ or $V(A) = V(\neg B) = 1$. Consequently, $V(X' \cup \{(A \leftrightarrow B), \neg A, B\}) = 1$ or $V(X' \cup \{(A \leftrightarrow B), A, \neg B\}) = 1$. □

Another lemma describes the relationship between finite, non-contradictory sets of formulas and the branches that originate from them. In this lemma we state that for a finite and non-contradictory set of formulas there is always at least one branch, beginning with this set, which is open and maximal.

Lemma 2.46 (On the existence of maximal and open branch)**.** *Let X be a finite set of formulas, and V be a valuation. If $V(X) = 1$, then there exists at least one maximal and open branch $X_1, \ldots, X_n$ such that $X_1 = X$ and $n \geq 1$.*

Proof. Take any finite set of formulas X and valuation V such that $V(X) = 1$. Based on conclusion 2.37, we know that set of all maximal branches $MB(X)$ is non-empty. Indirectly assume that none of the branches contained in $MB(X)$ is open.

Now, consider the branches beginning with set X, taking accounts of their lengths. Through inductive proof, we will show that for any $n \in \mathbb{N}$ there exists such open branch $X_1, \ldots, X_n$ that $X_1 = X$ and there exists such set of formulas Y that sequence $X_1, \ldots, X_n, X_{n+1}$, where $X_{n+1} = Y$, is also an open branch.

Initial step. There exists an open branch with the length of 1 beginning with set X. It is branch $X_1 = X$. Since $V(X) = 1$, by definition 2.9, set X is not contradictory, while by virtue of conclusion 2.11, set X is not t-inconsistent. So, by definition of open branch 2.38 branch X_1 is open. Hence, by virtue of the indirect assumption, X_1 is not a maximal branch. Thus, by definition of maximal branch 2.30 and lemma 2.45, there exists branch X_1, X_2 such that $V(X_2) = 1$. By definition 2.9, set X_2 is not contradictory, while by virtue of conclusion 2.11, set X_2 is not t-inconsistent. So, by definition of open branch 2.38, branch X_1, X_2 is open.

Induction step. Assume that for some $n \in \mathbb{N}$, there exists such open branch X_1, …, X_n that $X_1 = X$, $V(X_n) = 1$ and $n \geq 2$. Hence, by the indirect assumption, X_1, …, X_n is not a maximal branch.

Since $V(X_n) = 1$, by definition 2.9, set X_n is not contradictory, while by virtue of conclusion 2.11, set X_n is not t-inconsistent. Thus, by definition of maximal branch

2.30 and lemma 2.45, there exists branch $X_1, \ldots, X_n, X_{n+1}$ such that $V(X_{n+1}) = 1$. By definition 2.9, set X_{n+1} is not contradictory, while by virtue of conclusion 2.11, set X_{n+1} is not t-inconsistent. So, by definition of open branch 2.38, branch $X_1, \ldots, X_n, X_{n+1}$ is open.

Consequently, for any $n \in \mathbb{N}$, there exists such open branch $X_1, \ldots, X_n$ that $X_1 = X$ and there exists such set of formulas Y that sequence $X_1, \ldots, X_n, X_{n+1}$, where $X_{n+1} = Y$, is also an open branch and is not a maximal branch.

From this it follows that there exists an infinite branch $Y_1, Y_2, \ldots$ such that

1. $Y_1 = X$
2. $Y_{n+1} = X_{n+1}$, where X_{n+1} is such set that sequence $X_1, \ldots, X_n, X_{n+1}$, is also an open branch.

But, since X is a finite set of formulas, so the fact of existence of infinite branch $Y_1, Y_2, \ldots$ contradicts fact 2.28. So, the indirect assumption that each branch contained in $MB(X)$ is closed, is false. So there exists a branch beginning with X, which is maximal and open. □

With a lemma on the existence of maximal and open branch, we can now prove the soundness theorem.

Theorem 2.47 (Soundness). *For any* $X \subseteq \mathsf{For_{CPL}}$, $A \in \mathsf{For_{CPL}}$, *if* $X \rhd A$, *then* $X \vDash A$.

Proof. Take any $X \subseteq \mathsf{For_{CPL}}$, $A \in \mathsf{For_{CPL}}$ and assume that $X \nvDash A$. So by definition 2.6 there exists such valuation V that $V(X \cup \{\neg A\}) = 1$. Hence for any finite set $Y \subseteq X$, $V(Y \cup \{\neg A\}) = 1$. From the previous lemma 2.46 it follows that for any finite $Y \cup \{\neg A\}$ there exists at least one maximal and open branch $X_1, \ldots, X_n$ such that $X_1 = Y \cup \{\neg A\}$ i $n \geq 1$. So there is no such finite set $Z \subseteq X$ that each maximal branch beginning with $Z \cup \{\neg A\}$ is closed. Hence by definition 2.41 $X \not\rhd A$. □

2.4.2 Completeness theorem

In this section, we will show that the relation of semantic consequence is contained in the relation of branch consequence, hence for any set of formulas X and any formula A it is the case that if $X \vDash A$, then $X \rhd A$.

Also here, we will use abbreviations concerning the valuations of the formulas adopted in denotation 2.5.

In our proofs, we will use the fact that the classical consequence relation $\vDash$ is compact. However, let us first recall the general definition of compactness of a binary relation, because the concept of a compact relation will be useful in the next chapter.

Definition 2.48 (Compact relation)**.** Take any set of objects X and binary relation R, specified as follows $R \subseteq P(X) \times X$. We shall state that relation R is *compact* iff for any $Y \subseteq X$ and for any $x \in X$: $YRx \Leftrightarrow$ there exists a finite set Y' such that $Y' \subseteq Y$ and $Y'Rx$.

The well-known fact of the compactness of semantic consequence relation of **CPL** is expressed in the next fact.

Proposition 2.49 (Compactness)**.** *For any $X \subseteq \mathsf{For_{CPL}}$, $A \in \mathsf{For_{CPL}}$, $X \vDash A$ iff there exists such finite set $Y \subseteq X$ that $Y \vDash A$.*

We will start with a lemma which says about the relationship between the valuation of propositional letters and the negation of propositional letters in a maximal branch and the valuation of formulas from some initial set.

Lemma 2.50. *Let $X_1, \ldots, X_n$, for some $n \geq 1$, be a maximal and open branch. Let $L(X_n) = \{x \in X_n : x = y \text{ or } x = \neg y, \text{ for some } y \in \mathsf{Var}\}$. Then for any valuation V, if $V(L(X_n)) = 1$, then $V(X_n) = 1$.*

Proof. We will carry out an inductive proof, taking account of the complexity of formulas in a branch.

Take any maximal and open branch $X_1, \ldots, X_n$, for some $n \geq 1$, and valuation V such that $V(L(X_n)) = 1$, where $L(X_n) = \{x \in X_n : x = y \text{ or } x = \neg y, \text{ for some } y \in \mathsf{Var}\}$. Let $A \in X_n$.

Initial step. If $A = y$ or $A = \neg y$, for some $y \in \mathsf{Var}$, then $V(A) = 1$, since $A \in L(X_n)$.

Induction step. Assume that for all formulas B, $C \in X_n$ that have the lesser complexity than A, $V(B) = V(C) = 1$. Our proof will be based on possible cases of constructing formula A and on the initial assumption that $X_1, \ldots, X_n$ is a maximal branch. Since the branch is maximal, so if A could be decomposed by some of the tableau rules, then its component (components) are already in X_n (as for all $i \leq n$, $X_i \subseteq X_n$).

Now, for some formulas B, C occurs one of the cases:

- $A := (B \wedge C)$, and so $B, C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(B) = V(C) = 1 = V((B \wedge C))$
- $A := (B \vee C)$, and so B or $C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(B) = 1$ or $V(C) = 1$, therefore $V((B \vee C)) = 1$
- $A := (B \to C)$, and so $\neg B$ or $C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(\neg B) = 1$ or $V(C) = 1$, therefore $V((B \to C)) = 1$

- $A := (B \leftrightarrow C)$, and so B, C or $\neg B$, $\neg C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(B) = 1 = V(C)$ or $V(\neg B) = 1 = V(\neg C)$, therefore $V((B \leftrightarrow C)) = 1$
- $A := \neg\neg B$, and so $B \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(B) = 1 = V(\neg\neg B)$
- $A := \neg(B \wedge C)$, and so $\neg B$ or $\neg C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(\neg B) = 1$ or $V(\neg C) = 1$, therefore $V(\neg(B \wedge C)) = 1$
- $A := \neg(B \vee C)$, and so $\neg B, \neg C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(\neg B) = V(\neg C) = 1 = V(\neg(B \vee C))$
- $A := \neg(B \rightarrow C)$, and so $B, \neg C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(B) = V(\neg C) = 1 = V(\neg(B \rightarrow C))$
- $A := \neg(B \leftrightarrow C)$, and so $\neg B$, C or B, $\neg C \in X_n$, on the inductive assumption and by definition of valuation of formulas 2.4 we then get that $V(\neg B) = 1 = V(C)$ or $V(B) = 1 = V(\neg C)$, therefore $V(\neg(B \leftrightarrow C)) = 1$.

Consequently, $V(X_n) = 1$, since $V(A) = 1$, for all $A \in X_n$. □

The next lemma will show that a maximal and open branch allows to define a valuation that assigns the value of truth to each formula contained in the first element of branch.

Lemma 2.51 (Lemma on the existence of valuation)**.** *Let $X_1, \ldots, X_n$, for some $n \geq 1$, be a maximal and open branch. Then, there exists valuation V such that $V(X_1) = 1$.*

Proof. Take any maximal and open branch $X_1, \ldots, X_n$, for some $n \geq 1$. Now, we define $L(X_n) = \{x \in X_n : x = y \text{ or } x = \neg y, \text{ for some } y \in \mathsf{Var}\}$. Since $L(X_n)$ is t-consistent, we define valuation $v : \mathsf{Var} \longrightarrow \{1, 0\}$ such that for any $x \in \mathsf{Var}$:

- $v(x) = 1$, if $x \in L(X_n)$
- $v(x) = 0$, if $x \notin L(X_n)$.

We see that both $v(L(X_n)) = 1$ and $V(L(X_n)) = 1$, where V is an extension of function v to the set of all formulas.

Now, using lemma 2.50, we get thesis $V(X_1) = 1$, since $X_1 \subseteq X_n$. □

The lemma on the existence of valuation allows us to prove the completeness theorem.

Theorem 2.52 (Completeness)**.** *For any* $X \subseteq \mathsf{For_{CPL}}$, $A \in \mathsf{For_{CPL}}$, *if* $X \models A$, *then* $X \rhd A$.

Proof. Take any $X \subseteq \mathsf{For_{CPL}}$, $A \in \mathsf{For_{CPL}}$ and assume that $X \not\rhd A$. Thus, by definition of branch consequence relation 2.41 for each finite subset $Y \subseteq X$ there exists a maximal branch beginning with $Y \cup \{\neg A\}$ which is open. By virtue of the lemma on the existence of valuation 2.51 for each finite set $Y \subseteq X$ there exists valuation V such that $V(Y \cup \{\neg A\}) = 1$. Thus, by definition of valuation of formulas 2.4 and consequence classical relation 2.6, for each finite set $Y \subseteq X$, $Y \not\models A$. Hence and by virtue of the compactness property of relation of consequence $\models$, fact 2.49, we get thesis $X \not\models A$. □

2.5 Tableaux for CPL vs. semantic consequence relation

So we can see that the concepts of relation of semantic consequence and relation of branch consequence denote exactly the same set of pairs $\langle X, A\rangle$, where X is a set of formulas, while A is a formula. In practice, however, it is not easy to determine whether a pair belongs to relation $\rhd$. According to definition 2.41, in order to achieve this, we need to select from X its finite subset Y such that each maximal branch beginning with set $Y \cup \{\neg A\}$ is closed. The first stage of this activity, i.e. the selection of an appropriate set, is difficult to study in general, since further we will deal with various different logics, and the relation of branch consequence has been defined for any sets, particularly infinite ones. However, if the set of premises is a finite set, we can proceed straight to the second stage, i.e. reviewing all maximal branches and checking if they are closed branches.

Unfortunately, although their number in the case of a finite set of premises will also be finite, it may be so large that their construction and examination are only theoretically possible. Therefore, we need a method that allows to select and study a sufficiently small number of maximal branches, the closure of which guarantees the occurrence of branch consequence relation.

In our theory, this method is based on the concept of a tableau. By tableau we mean a finite and minimal set of branches beginning with the same set. Of course, such a concept of tableau is far from its graphic presentation or one based on graphs. Due to its formal nature, it specifies the standard concept of tableau, plus its scope at least includes a set of those objects which are traditionally called semantic or analytical trees or tableaux. The problem of this relation will be dealt with in the last chapter of the book.

Let us now proceed to the definition of tableau.

Definition 2.53 (Tableau)**.** Let $X \subseteq \mathsf{For}_{\mathbf{CPL}}$, $A \in \mathsf{For}_{\mathbf{CPL}}$, while Φ will be a set of branches. Ordered triple $\langle X, A, \Phi \rangle$ will be called a *tableau for* $\langle X, A \rangle$ (or shortly *tableau*) iff the below conditions are met:

1. Φ is a non-empty subset of set of branches beginning with $X \cup \{\neg A\}$ (i.e. if $\psi \in \Phi$, then $\psi(1) = X \cup \{\neg A\}$)
2. each branch contained in Φ is Φ-maximal
3. for any $n, i \in \mathbb{N}$ and any branches $\psi_1, \ldots, \psi_n \in \Phi$, if:
   - i and $i+1$ belong to the domains of functions $\psi_1, \ldots, \psi_n$
   - for any $1 < k \leq n$ and any $o \leq i$, $\psi_1(o) = \psi_k(o)$

   then there exists such rule $R \in \mathbf{R}_{\mathbf{CPL}}$ and such ordered m-tuple $\langle Y_1, \ldots, Y_m \rangle \in R$, where $1 < m \leq 3$, that for any $1 \leq k \leq n$:
   - $\psi_k(i) = Y_1$
   - and there exists such $1 < l \leq m$ that $\psi_k(i+1) = Y_l$.

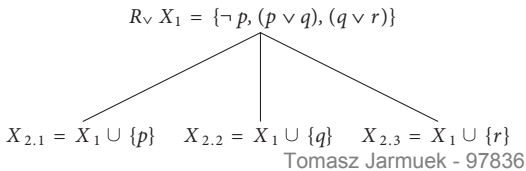
Remark 2.54. The first two conditions in the tableau definition are standard ones. Each branch in the tableau begins with a set which is the sum of the set of premises and the negation of the presumed conclusion. In addition, the tableau contains only Φ-maximal branches, so it does not include sub-branches of the branches belonging to Φ.

However, the third condition is particularly worth discussing. This condition says that branching can occur in a tableau only if there is a suitable rule and its ordered triple which contains two output sets corresponding to the branching.

We have deliberately reduced number m to range $\{2,3\}$, since the rules in set $\mathbf{R}_{\mathbf{CPL}}$ have at least two elements each, but not more than three. In the case of tableau systems for other logics, however, it may happen that m will be any number greater than 1 which will allow for several branchings at a given stage of the tableau construction. In general, the upper range of number m minus 1 means the number of branchings that can appear in the tableau at a given stage of construction.

Therefore, definition of tableau 2.53 excludes such sets of branches from being considered tableaux as in example 2.55.

Example 2.55. We consider set $\{\neg p, (p \vee q), (q \vee r)\}$ and create three branches.



Although there exists a rule — it is rule $R_\vee$ — which contains ordered triples $\langle X_1, X_{2.1}, X_{2.2}\rangle$ and $\langle X_1, X_{2.2}, X_{2.3}\rangle$, set $\mathbf{R_{CPL}}$ includes no rule to comprise a ordered quadruple $\langle X_1, X_{2.1}, X_{2.2}, X_{2.3}\rangle$, therefore the presented set of branches $\{\langle X_1, X_{2.1}\rangle, \langle X_1, X_{2.2}\rangle, \langle X_1, X_{2.3}\rangle\}$ does not meet the third condition of the definition of tableau 2.53.

As we can see, the concept of tableaux has been defined in such a way that tableaux can also begin with infinite sets.

In practice, the construction of tableau is to show that a given formula is a branch consequence of a given finite set of premises. To this end, we must construct tableaux that have all the elements necessary to solve the problem. Tableaux with these properties are called complete tableaux. But, before we proceed to the definition of complete tableau, we will consider one more issue.

When constructing a tableau, it may happen that branchings and branches are formed which are redundant variants of the already existing branches. Let us consider the following two examples.

Example 2.56. Consider set $\{p \vee q, \neg\neg p\}$ and create a branch using a rule $R_{\neg\neg}$ to this set. We get the following branch.

$$\begin{array}{c} R_{\neg\neg}\ X_1 = \{p \vee q, \neg\neg p\} \\ | \\ X_2 = X_1 \cup \{p\} \end{array}$$

Branches X_1, X_2 cannot be any more extended since ordered triple $\langle X_1 \cup \{p\}, X_1 \cup \{p\}, X_1 \cup \{p, q\}\rangle$ does not belong to rule $R_\vee$ due to the fact that $X_1 \cup \{p\} \notin X_1 \cup \{p\}$, whereas the input set should be contained in each output set.

We can, however, starting from set $\{p \vee q, \neg\neg p\}$, through the use of rule $R_\vee$, produce two branches, and than apply rule $R_{\neg\neg}$ to set $X_{2.2}$. Then we get the following branches.

$$\begin{array}{ccc} & R_\vee\ X_1 = \{(p \vee q), \neg\neg p\} & \\ \swarrow & & \searrow \\ X_{2.1} = X_1 \cup \{p\} & & R_{\neg\neg}\ X_{2.2} = X_1 \cup \{q\} \\ & & | \\ & & X_3 = X_{2.2} \cup \{p\} \end{array}$$

In the light of definition of tableau 2.53, the set of these two branches is, obviously, a tableau. However, from the viewpoint of a tableau complexity and information it provides, the branch on the right seems unnecessary. This is because if we fail to get t-inconsistent set in the right-hand branch, we will also fail

to get t-inconsistent set in the left-hand branch. On each set of formulas Y we generally know from the definition of tableau inconsistent set of formulas 2.10 that Y is not t-inconsistent iff each of its subsets is not t-inconsistent, and the point of constructing a tableau is precisely finding a t-inconsistency. Therefore, the branch on the right seems superfluous, and since it brings nothing important, it can be called redundant.

Such branchings are not a formal obstacle, hence they can be accepted. However, due to the economy of a tableau construction, they can be disregarded. Further concepts will be defined so that the tableau with or without redundant branches can be considered a complete tableau. Practically, we know that when we try to write or draw out a tableau proof, we endeavour to take account of all possibilities. So there is no reason to further restrict this process — and we allow both options. Let us now proceed to the formal concept of a redundant variant of branch.

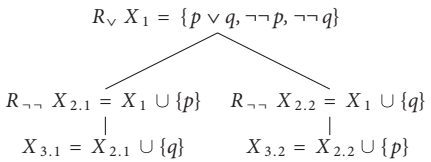
Definition 2.57 (Redundant variant of branch)**.** Let ϕ and ψ be such branches that if there exists number i that i and $i+1$ belong to their domains, then for any $j \leq i$, $\phi(j) = \psi(j)$, but $\phi(i+1) \neq \psi(i+1)$. We shall state that branch ψ is a *redundant variant* of branch ϕ iff:

1. there exist such rule $R \in \mathbf{R_{CPL}}$ and such pair $\langle X, Y\rangle \in R$ that $X = \phi(i)$ and $Y = \phi(i+1)$
2. there exist such rule $R \in \mathbf{R_{CPL}}$ and such triple $\langle X, W, Z\rangle \in R$ that $X = \phi(i)$ and:
   a. $W = \phi(i+1)$ and $Z = \psi(i+1)$
      or
   b. $Z = \phi(i+1)$ and $W = \psi(i+1)$.

Let Φ, Ψ be sets of branches and $\Phi \subset \Psi$. We shall state that Ψ is an *redundant superset* of Φ iff for any branch $\psi \in \Psi \setminus \Phi$ there exists such branch $\phi \in \Phi$ that ψ is a redundant variant of ϕ.

Let us consider another example of tableau with redundant branches.

Example 2.58. Take set $\{\neg\neg p,\ \neg\neg q,\ p \vee q\}$. By applying rules $R_{\vee}$ and $R_{\neg\neg}$ successively, we will get an interesting case of tableau.



Sets $X_{3.1}$ and $X_{3.2}$ are identical. So, we face an interesting situation where both branches "diverged" (sets $X_{2.1}$ and $X_{2.2}$ are different), and then "converged" (sets $X_{3.1}$ and $X_{3.2}$ are identical). Due to the definition of redundant variant of branch 2.57, each of those two branches is a redundant variant of the other.

Remark 2.59. The concept of a redundant variant of branch can be extended onto other cases of branchings where one of branches brings no expressions into the tableau that are necessary to obtain the answer to the question whether given inference is correct. Nevertheless, we will not expand this concept. After all, we are not interested in the economy of tableau proof. We introduced this concept in order to show that the adopted concepts of tableau and branch allows us to distinguish certain types of branches, and consequently, to distinguish certain sets of tableaux that are less complex.

Now, we will move on to the definition of complete tableau.

Definition 2.60 (Complete tableau)**.** Let $\langle X, A, \Phi \rangle$ be a tableau. We shall state that $\langle X, A, \Phi \rangle$ is *complete* iff:

1. each branch contained in Φ is maximal
2. any set of branches Ψ such that:
   a. $\Phi \subset \Psi$
   b. $\langle X, A, \Psi \rangle$ is a tableau

   is a redundant superset of Φ.

A tableau is *incomplete* iff it is not complete.

In a complete tableau, all branches are maximal, not only the maximal ones in a given set. In addition, a complete tableau is such set of branches that adding a new branch to it at most gives us a redundant superset or the set ceases to be a tableau. In other words, a complete tableau is such set of maximal branches that any of its supersets ceases to be a tableau or is a redundant superset.

When constructing a complete tableau, we can come across a situation in which all the branches are closed, meaning each branch ends with a t-inconsistent set. Such a tableau is called closed tableau. Let us first define a closed/open tableau, and then discuss the definition.

Definition 2.61 (Closed/open tableau)**.** Let $\langle X, A, \Phi \rangle$ be a tableau. We shall state that $\langle X, A, \Phi \rangle$ is *closed* iff the below conditions are met:

1. each branch contained in Φ is closed
2. any set of branches Ψ such that:
   a. $\Phi \subset \Psi$

b. $\langle X, A, \Psi \rangle$ is a tableau

is a redundant superset of Φ.

A branch is *open* iff it is not closed.

As we have said, by the above definitions — the definition of complete tableau 2.60 and the definition of closed tableau 2.61 — and the conclusion 2.40, we have another conclusion.

Corollary 2.62. *Each closed tableau is a complete tableau.*

Moreover, among complete tableaux, those tableaux that only contain sets of closed branches are closed tableaux. So we have another conclusion.

Corollary 2.63. *Each complete tableau in which the set of branches only contains closed branches is a closed tableau.*

Making use of conclusions 2.62 and 2.63, we can, therefore, simplify the definition of closed/open tableau by formulating another conclusion.

Corollary 2.64 (Closed/open tableau)**.** *Let* $\langle X, A, \Phi \rangle$ *be a tableau. Tableau* $\langle X, A, \Phi \rangle$ *is closed iff the below conditions are met:*

1. $\langle X, A, \Phi \rangle$ *is a complete tableau*
2. *each branch contained in* Φ *is closed.*

A branch is open iff it is not closed.

Conclusion 2.64 can be adopted as an equivalent version of the definition of closed/open tableau.

Further, we will show that the concept of tableau is significantly helpful in determining the occurrence of relation $\triangleright$.

Now, we will focus on the fact that the initial, finite set of formulas allows to construct a complete tableau.

Proposition 2.65. *Let X be a finite subset of set* $\mathsf{For}_{\boldsymbol{CPL}}$ *and let* $A \in \mathsf{For}$*. Then, there exists at least one complete tableau* $\langle X, A, \Phi \rangle$*.*

Proof. Consider a set of all branches $B(X \cup \{\neg A\})$. We know that the set of maximal branches $MB(X \cup \{\neg A\})$ is non-empty (fact 2.37). In addition, for each branch $\phi \in B(X \cup \{\neg A\})$ there exists branch $\psi \in MB(X \cup \{\neg A\})$ such that $\phi \subseteq \psi$, by 2.34.

Let us now define such set $\Phi \subseteq MB(X \cup \{\neg A\})$ that Φ is a maximal set among the sets that meet the following condition:

- for any $n, i \in \mathbb{N}$ and any branches $\phi_1, \ldots, \phi_n \in \Phi$, if:
  - i and $i+1$ belong to domains of functions $\phi_1, \ldots, \phi_n$
  - for any $1 < k \leq n$ and any $o \leq i$, $\phi_1(o) = \phi_k(o)$,

  then there exists such rule $R \in \mathbf{R_{CPL}}$ and such ordered m-tuple $\langle Y_1, \ldots, Y_m \rangle \in R$, where $1 < m \leq 3$, that for any $1 \leq k \leq n$:
  - $\phi_k(i) = Y_1$
  - and there exists such $1 < l \leq m$ that $\phi_k(i+1) = Y_l$.

By definition of tableau 2.53, $\langle X, A, \Phi \rangle$ is a tableau. What is more, since Φ is a maximal set among those meeting the given condition, then by the definition of complete tableau 2.60, $\langle X, A, \Phi \rangle$ is a complete tableau. Since $MB(X \cup \{\neg A\})$ is non-empty, there exists at least one Φ that meets the given condition. □

We will now move on to a lemma that defines the relationship between the existence of a closed tableau and complete tableaux.

Lemma 2.66. *Let X be a finite subset of set $\mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$. Then, the following two statements are equivalent:*

1. *there exists closed tableau $\langle X, A, \Phi' \rangle$*
2. *each complete tableau $\langle X, A, \Phi'' \rangle$ is closed.*

Proof. Take any finite subset X of set $\mathsf{For}_{\mathbf{CPL}}$ and any formula $A \in \mathsf{For}_{\mathbf{CPL}}$.

First, we will take up the proof of implication $(1) \Rightarrow (2)$. Indirectly assume that there exists closed tableau $\langle X, A, \Phi' \rangle$ and not every complete tableau $\langle X, A, \Phi'' \rangle$ is closed. Hence, there exists complete tableau $\langle X, A, \Phi'' \rangle$ which is not closed. By definition of open tableau 2.61, $\langle X, A, \Phi'' \rangle$ is an open tableau. And since $\langle X, A, \Phi'' \rangle$ is a complete tableau, then each branch contained in Φ'' is maximal. Hence, there exists branch $\psi \in \Phi''$ such that:

1. ψ begins with $X \cup \{\neg A\}$
2. ψ is a maximal branch
3. ψ is an open branch.

Since ψ is an open branch, then by definition of open branch 2.38, there is no such formula A that A and $\neg A$ belong to the union of all the elements of branch $\bigcup \psi$.

Note that since branch ψ is maximal and open, then it is also closed under tableau rules in this regard that for any $R \in \mathbf{R_{CPL}}$ and any n-tuple $\langle X_1, \ldots, X_n \rangle \in R$, where $n > 1$, if $X_1 \subseteq \bigcup \psi$, then for some $1 < j \leq n$, $X_j \subseteq \bigcup \psi$. For if that was not the case, there would exist rule $R \in \mathbf{R_{CPL}}$ and such n-tuple $\langle X_1, \ldots, X_n \rangle \in R$, where $n > 1$, that $X_1 \subseteq \bigcup \psi$, but for none $1 < j \leq n$, $X_j \nsubseteq \bigcup \psi$. And it would mean that branch ψ

is not maximal, because $\bigcup\psi$ is identical to the last element of open and maximal branch ψ, by definition of branch 2.20.

We assumed that there existed closed tableau $\langle X, A, \Phi'\rangle$. By virtue of conclusion 2.64, $\langle X, A, \Phi'\rangle$ is a complete tableau. Hence, each branch contained in Φ' is maximal and Φ' contains all such branches without which ordered triple $\langle X, A, \Phi'\rangle$ would not be a complete tableau. Let us adopt designation $X \cup \{\neg A\} = Y_1$.

We will carry out an inductive proof with respect to n-th elements of branches contained in Φ', showing that they enable construction of an infinite branch beginning with set Y_1.

Initial step. Consider the first element of each branch contained in Φ'. It is set Y_1. Since $Y_1 \subseteq \bigcup\psi$, and ψ is an open branch, then there must exist a rule $R \in \mathbf{R_{CPL}}$ such that $\langle Y_1, Z_1, \ldots, Z_n\rangle \in R$, where $n \geq 1$, and — since $\langle X, A, \Phi'\rangle$ is a complete tableau — for each $1 \leq j \leq n$ there exists a branch in Φ' such that it contains Z_j. Since branch ψ is also closed under tableau rules, then certain $Z_j \subseteq \bigcup\psi$. So, Φ' includes such branch that its first element is Y_1, and second $Y_2 = Z_j$, and $Y_2 \subseteq \bigcup\psi$

Induction step. Assume that set Φ' includes such branch that its first n elements $Y_1, \ldots, Y_n$ are contained in set $\bigcup\psi$. Since $Y_n \subseteq \bigcup\psi$, and ψ is an open branch, then there must exist a rule $R \in \mathbf{R_{CPL}}$ such that $\langle Y_n, Z_1, \ldots, Z_k\rangle \in R$, where $k \geq 1$, and — since $\langle X, A, \Phi'\rangle$ is a complete tableau — for each $1 \leq j \leq k$ there exists a branch in Φ' such that it contains Z_j. Since branch ψ is also closed under tableau rules, then certain $Z_j \subseteq \bigcup\psi$. So, Φ' includes such branch that its first $n+1$ elements $Y_1, \ldots, Y_n, Y_{n+1}$ are contained in set $\bigcup\psi$.

So, for each branch in Φ', the first element equals Y_1 and $Y_1 \subseteq \bigcup\psi$ and for each $n \in \mathbb{N}$ if there exists in Φ' such branch that its first n elements $Y_1, \ldots, Y_n$ are contained in set $\bigcup\psi$, then also in Φ' there exists such branch that its first n elements equal $Y_1, \ldots, Y_n$, and its $n+1$-element Y_{n+1} is contained in set $\bigcup\psi$.

Now, we take all elements Y_i, where $i \in \mathbb{N}$ and we put them in increasing sequence $Y_1, Y_2, Y_3, \ldots$. This sequence is a branch, because for any Y_j there exists a rule $R \in \mathbf{R_{CPL}}$ such that $\langle Y_j, Z_1, \ldots, Z_k\rangle \in R$, where $k \geq 1$, and $Y_{j+1} = Z_l$, for some $1 \leq l \leq k$.

There exists therefore an infinitely long branch $Y_1, \ldots, Y_n, \ldots$ such that $Y_1 = X \cup \{\neg A\}$. But, since set $X \cup \{\neg A\}$, by assumption, is finite, then it contradicts fact 2.28.

Let us now move on to the proof of implication $(2) \Rightarrow (1)$. Assume that each complete tableau $\langle X, A, \Phi''\rangle$ is closed. From fact 2.65, we know that for finite set of formulas X there exists at least one complete tableau $\langle X, A, \Phi\rangle$. Therefore, there exists closed tableau $\langle X, A, \Phi'\rangle$. □

We will now consider the relationship between complete and closed tableaux and the occurrence of branch consequence relation.

Lemma 2.67 (On relation between complete tableaux and branch consequence)**.** *Let $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$. Then, the two below statements are equivalent:*

1. *there exists such finite set $Y \subseteq X$ that each complete tableau $\langle Y, A, \Phi \rangle$ is closed*
2. *$X \rhd A$.*

Proof. The proof of the above equivalence is based on the definition of the notion tableau 2.60, notion of a closed tableau 2.61 and notion of a relation of branch consequence $\rhd$.

Take $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$, and assume there exists such finite set $Y \subseteq X$ that each complete tableau $\langle Y, A, \Phi \rangle$ is closed. Therefore, each maximal branch that begins with set $Y \cup \{\neg A\}$ is closed. So, there exists such finite set $Y \subseteq X$ that each maximal branch that begins with set $Y \cup \{\neg A\}$ is closed. Hence, $X \rhd A$.

On the other hand, if $X \rhd A$, then there exists such finite set $Y \subseteq X$ that each maximal branch that begins with set $Y \cup \{\neg A\}$ is closed. Hence, there exists such finite set $Y \subseteq X$ that each complete tableau $\langle Y, A, \Phi \rangle$ is closed. □

We now proceed to the most important theorem of this subchapter. The occurrence of this theorem, or at least of its part "from the left to the right," may be a criterion for the correctness of tableau system construction. When constructing a tableau emerging from a given set of formulas, we can usually do it in many ways. In practice, however, we construct one tableau, checking whether each branch ends with a contradictory set. Usually we also assume that it is sufficient to state that given formula belongs to the set of correct conclusions from the initial set of formulas.

Intuitively, however, it seems doubtful. Why should one closed tableau be a proof if it is potentially possible to construct more tableaux? After all, usually, we have to rule out all cases in indirect proofs, and this is what the construction of a tableau is. Another theorem removes this doubt by saying that in order to determine the occurrence of relation of branch consequence, it is sufficient to construct one closed tableaux.

Theorem 2.68 (Theorem on the tableau arbitrariness)**.** *Let $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$. Then, the two below statements are equivalent:*

1. *there exists finite set $Y \subseteq X$ and closed tableau $\langle Y, A, \Phi \rangle$*
2. *$X \rhd A$.*

Proof. Take any $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ and $A \in \mathsf{For}_{\mathbf{CPL}}$ and assume there exists such finite set $Y \subseteq X$ and closed tableau $\langle Y, A, \Phi \rangle$. By virtue of lemma 2.66, we get the conclusion that there exists such finite set $Y \subseteq X$ that each complete tableau $\langle Y, A, \Phi' \rangle$ is closed. And from the above, and from lemma 2.67, it follows that $X \triangleright A$.

Now, assume that $X \triangleright A$. By virtue of lemma 2.67, there exists such finite set $Y \subseteq X$ that each complete tableau $\langle Y, A, \Phi' \rangle$ is closed. So, from lemma 2.66, it follows that there exists finite set $Y \subseteq X$ and closed tableau $\langle Y, A, \Phi \rangle$. □

The construction of one closed tableau which begins with a finite subset of set of premises X and negation of formula A is, therefore, equivalent to the fact that $X \triangleright A$. Because previously, through the theorems on soudness and completeness, we proved that relations $\vDash$ and $\triangleright$ are identical, i.e. they define the same set of pairs, now we can put in words the semantic form of the theorem on the tableau arbitrariness.

Theorem 2.69 (Theorem on the tableau arbitrariness — semantic form)**.** *Let* $X \subseteq \mathsf{For}_{\mathbf{CPL}}$ *and* $A \in \mathsf{For}_{\mathbf{CPL}}$*. Then, the two below statements are equivalent:*

1. *there exists finite set* $Y \subseteq X$ *and closed tableau* $\langle Y, A, \Phi \rangle$
2. $X \vDash A$.

This theorem says that the logical relation occurs between the premises and conclusion if and only if it is possible to select a finite set of premises, and then by attaching to it the negation of conclusion, construct a closed tableau emerging from that set.

2.6 Summary

In this chapter we presented a theory for the construction of a tableau system for **CPL**, using the method of defining tableau rules as rules that extend sets.

For the purposes of presentation, we separately showed the relationships between the semantic consequence relation, the branch consequence relation and the existence of a closed tableau, proving the theorems on the equivalence of these concepts. In practice, however, it is easier to prove a theorem equivalent to the conjunction of the above theorems, i.e. the following theorem.

Theorem 2.70 (Theorem on the completeness of tableau system of **CPL**)**.** *For any* $X \subseteq \mathsf{For}_{\mathbf{CPL}}$*,* $A \in \mathsf{For}_{\mathbf{CPL}}$*, the below statements are equivalent.*

- $X \vDash A$
- $X \triangleright A$
- *there exists finite* $Y \subseteq X$ *and closed tableau* $\langle Y, A, \Phi \rangle$.

Although this theorem is equivalent to the conjunction of earlier theorems — as we can see — its demonstration requires a proof of three implications. Certain proof transitions may therefore be omitted.

Theorem 2.70, was named a theorem on the completeness of tableau system of **CPL**, referring to the usual name to define the relationship between the semantic characteristics of given logic and its deductive definition. Often, when speaking on the completeness of a given deductive system, we mean not only a one-way relationship, but a relationship occurring two ways, that is, both soundness and completeness in the strict sense.

In the presentation of other tableau systems constructed using the presented method, we will always strive to demonstrate that there occurs a relevant theorem on the completeness of the tableau system, formulated in an analogous way as theorem 2.70. The proof of such theorem will be a positive criterion for the good formulation of the tableau system.