1 Introduction

1.1 Tableau methods

In the book, we look into the tableau methods. These methods are used to define logical systems in which through proofs — called tableau proofs — it is possible to show the occurrence of logical relations between sets of premises and conclusions.

Tableau methods in many ways constitute an interesting alternative to other methods of constructing logical systems. They are also interesting because the tableau systems have many advantages over other types of systems. Unfortunately, they also have their drawbacks. The aim of the book is to define tableau methods in such a way that for a certain class of tableau systems these drawbacks are minimized or, where possible, completely eliminated.

Let us briefly expose some features of tableau systems, comparing them with axiomatic systems.

One of the advantages of tableau systems is a fairly intuitive and simple mechanism of theorem proving. In most cases, knowing the tableau rules and the way they work, we can mechanically search for answers to the question whether a given formula is a logical consequence of a given set of premises. Such action does not require any particular ingenuity. Another advantage is the fact that if the answer is negative, most often — also intuitively — on the basis of an unsuccessful tableau proof we can build so-called countermodel, i.e. a model in which the premises are true, but the conclusion is false.

Unfortunately, the disadvantages of tableau systems are the complications that arise when trying to construct them precisely — there are many complex theoretical concepts. Obviously, we can use intuitive concepts of branch/track of proof, tableau/tree, open or closed tableau, complete tableau, etc. However, firstly, we are not dealing with a formal system, but with a preformal one, and thus potentially more or less burdened with serious logical errors. Secondly, it is difficult, if at all possible, to generalize our results by looking for metalogical dependencies between different tableau systems, as well as dependencies between classes of systems as for such actions we need general concepts whose specific cases occur within the construction of specific tableau systems.

In turn, axiomatic systems feature precisely defined basic concepts and these concepts can be generalized. For example, it can be assumed that each axiomatic system consists of a decidable set of formulas of a certain language For and a set of rules of proving **R**. Since the system axioms can be described as zero-premiss

rules, the general concept of *rule of proving* in the axiomatic system can be, for a given set of formulas For, defined as follows:

$R = \{\langle X, A\rangle : X \text{ is a subset of } \mathsf{For},\ A \text{ is a formula}\}$.

An axiomatic system is most often a ordered pair $\langle \mathsf{For}, \mathbf{R}\rangle$, where For is a decidable set of formulas, whereas **R** is non-empty set of rules of argumentation. When a rule is an axiomatic rule, it contains pairs $\langle X, A\rangle$, where X is an empty set, whereas A is a formula being introduced to the proof without premises.

With an axiomatic system $\langle \mathsf{For}, \mathbf{R}\rangle$ we can now define the general concept of proof of formula A on the basis of set of premises X. We shall state that A is *provable* on the basis of premises X iff there exists finite sequence of formulas $B_1, \ldots, B_n$ such that:

1. $B_n = A$
2. for any $1 \leq i \leq n$ at least one of two cases occurs:
   - $B_i \in X$
   - there exist: rule $R \in \mathbf{R}$ and such pair $\langle Y, C\rangle \in R$ that
     - $C = B_i$
     - Y is an empty set or for certain $m > 0$ and certain $0 < k_1, \ldots, k_m < i$, $Y = \{B_{k_1}, \ldots, B_{k_m}\}$.

With a defined axiomatic system and the concept of provability, we can proceed to the argumentation. However, unlike the tableau systems, in most cases it is not easy as it requires intuition, ingenuity and other skills. Moreover, it is usually difficult to move from an unsuccessful proof to a countermodel construction.

The above comparison could be summarised by saying that:

- axiomatic systems are simpler to define (which does not imply that a system with the required properties is easy to define, but the general definition of system and proof is simple) and can be defined precisely, however producing proofs on their ground and finding a countermodel is usually difficult
- tableau systems are more complicated to define and are usually not defined with sufficient precision (which makes it impossible to generalize tableau concepts and create metatheories of whole classes of tableau systems), but producing proofs in tableau systems is most often simple and in many cases enables finding a countermodel.

One of the primary goals of this study is to define the tableau concepts — concepts that seem necessary — that occur when defining various tableau systems precisely, and then try to generalize them within the scope determined by the use of tableau methods to construct propositional logic and term logic specified by bivalent semantics.

Therefore, we would like the tableau methods — similarly to axiomatic methods — to be based on certain constants and general concepts, and the construction of a given tableau system boiled down only to specifying those components that distinguish this tableau system from other tableau systems.

Such an approach will result not only in general and precise tableau concepts that can be used in the construction of different systems but also in proving metatheorems not of individual systems, but of entire classes of tableau systems. These metatheorems will apply to those properties of tableau systems that are independent of their specific features and concern all systems in a given class, constructed on the basis of general, developed tableau concepts.

By a tableau system — by analogy to the axiomatic system — we can understand a certain ordered triple of sets $\langle \mathsf{For}, \mathsf{Te}, \mathbf{Rt} \rangle$ where:

- For is a set of formulas of a given language
- Te is a set of tableau expressions — proof expressions of a tableau system being defined
- $\mathbf{Rt}$ is a set of tableau rules designed to produce tableau proofs.[1]

As distinct from the axiomatic system, the tableau system therefore features an additional element — a set of tableau expressions Te. In the specific case — such a case will be described in the next chapter — it may be so that $\mathsf{Te} = \mathsf{For}$. Most often, however, these sets are different because although we look for a logical relationship between a set of formulas and a given formula, we carry out a tableau proof based on tableau expressions Te for which a set of formulas alone is often insufficient.

Every axiomatic system $\langle \mathsf{For}, \mathbf{R} \rangle$ unambiguously determines a relation of derivability $\vdash_{\mathbf{R}}$ which occurs between such sets of premises and such formulas that on the grounds of this system the latter are derivable from the former. The axiomatic system can therefore be understood as pair $\langle \mathsf{For}, \vdash_{\mathbf{R}} \rangle$.

It should be similar for any tableau system $\langle \mathsf{For}, \mathsf{Te}, \mathbf{Rt} \rangle$, and thus it should unambiguously determine the tableau derivability relationship $\rhd_{\mathbf{Rt}}$ which contains such pairs of premises and conclusions $\langle X, A \rangle$ that in the system on grounds X we can tableau prove A. We call relationship $\rhd$ a tableau consequence, and more

1 It is an initial comprehension of a tableau system. We will not refer to it in further considerations. The proposed definition of a tableau rule will show that set Te is unnecessary for the characteristics of a tableau system since argumentation in that system does not go beyond the application of the tableau rules defined on the set of tableau expressions. In the last chapter, using the previously developed concepts, we will enunciate the concept of a tableau system, which will be more appropriate to the theory described in the book.

often *branch consequence*, showing that its occurrence can be identified with the existence of a tableau with specific properties.

We should therefore aim at such definitions of tableau concepts so that we can understand a tableau system as pair $\langle \mathsf{For}, \rhd_{\mathbf{Rt}} \rangle$. We have deliberately left aside the set of tableau expressions here Te as it will only serve as a domain for defining tableau rules, and as a consequence of creating proofs — since the tableau expressions are included in tableau rules.

Therefore, the problem of defining a tableau system should be reduced to defining a set of formulas, defining a set of tableau rules and the relationship between formulas and rules. The remaining tableau concepts — as in the case of the concept of provability in the axiomatic system — should be special cases of more general tableau concepts. All the more so as the concept of tableau proof is not a simple one, but — as we will show — dependent on many simpler concepts.

Hence, we want to define tableau concepts in such a way that tableau systems are simpler to define and at the same time more precise (which will enable the generalization of tableau concepts and the work on metatheories of entire classes of tableau systems), while maintaining the intuitiveness and automaticity present in producing proofs in tableau systems and, where possible, retaining the property of finding a countermodel.

The logic can also be perceived from the semantic side, related to the interpretation of expressions. Axiomatic systems are customarily treated as an expression of a syntactic approach to logic. Without referring to the meaning of the inscriptions used, axiomatic systems specify how to decide the occurrence of relation $\vdash$ by transforming expressions solely in terms of their structure and shape.

In the book, we approach the problem of tableau systems in an analogous manner, but the starting point is usually (although not necessarily so) a logic defined in a purely semantic way. It can be understood as pair $\langle \mathsf{For}, \models \rangle$ where For is a set of formulas of given language, and $\models$ is the relation between sets of formulas and individual formulas, defined in a standard way based on the interpretation of set For.

Thus, having such a semantically understood logic $\langle \mathsf{For}, \models \rangle$, we want to define an tableau system (often for the same purpose we define an axiomatic system) $\langle \mathsf{For}, \rhd \rangle$[2] which, by transforming the tableau rules themselves, would allow us — where possible — to determine the occurrence for a given pair of relationships $\models$. To demonstrate that relations $\models$ and $\rhd$ coincide (i.e. $\rhd = \models$) would thus be not

2 In the designation, we skip set of rules **Rt** since definition $\rhd$ will depend on a set of tableau rules.

only a correctness measure for the constructed tableau system, but it also is the purpose of defining the tableau system.

Our approach to tableau systems is therefore syntactic. We treat them as systems defining the code of transforming expressions without direct reference to the semantic concepts. This may seem controversial, especially in those cases (i.e. in the vast majority) where $\mathsf{For} \neq \mathsf{Te}$; hence, when we use expressions that are different from the formulas in the proofs, by conjecture they encode some semantic data/privileges. This controversy can be dismissed by referring to two arguments.

The first one is hinges upon the following fact. When defining, in some tableau system, the relation of branch consequence $\rhd$, with semantically determined logic $\langle \mathsf{For}, \vDash \rangle$, it is not at all self-evident that these approaches coincide. This requires additional proofs for two theorems:

- completeness theorem $\vDash \subseteq \rhd$
- soundness theorem $\rhd \subseteq \vDash$.

So despite different definitions of logic and the tableau system and mutually independent definitions of both relations, we are seeking a proof that these relations are extensionally equal.

Finally, it is also possible to define a tableau system in which relation $\rhd$ does not coincide with the output semantic relation $\vDash$. Relation $\rhd$ and, consequently, the tableau system that determines it, are therefore something different than the logic and relations of semantically defined consequences. We can then try to improve the defined tableau system, or look for a different semantics — both cases show that we are dealing with something different than the initial, semantically defined system.

Let us put forward the second argument proving that the fact that tableau expressions Te encode certain semantic properties does not necessarily lead to the conclusion that the tableau systems are not syntactic ones. After all, in the case of axiomatic systems, the axioms and rules also encode various semantic properties of functors in a sense, even though it is not directly visible. Of course, we can say that in many cases there exist various axiomatic systems (defined by disjoint sets of rules) corresponding to the same semantic relation $\vDash$, which means that the semantic intuitions present in the rules and axioms do not have to be identical. But it is also the case — as we will show — in many cases of the tableau systems. There may be more than one tableau systems that are equivalent to the same semantically determined logic, which would have to imply that also in this case the semantic intuitions present in their construction are not identical.

Regardless of how we evaluate tableau systems, we define here the relations of tableau provability in a different way than is the case in the semantic

determination of consequence relations. For we do it in an autonomous way, although we undoubtedly transfer some semantic intuitions from semantically defined logic.

In this study, we will proof the theorems of completeness and soudness, leaning towards the view that we are dealing with a syntactic approach. However, a different orientation in this matter will not change the fact that we are considering something different than a purely semantically defined logic, nor will it invalidate the formal concepts presented in the paper.

1.2 Terminology and problems in the book

In each chapter of the study, we consider a case of a tableau system or make generalisations. The selected systems have different properties in some aspects which allows us to seek sufficiently general definitions of the tableau concepts so that all the systems described and all tableau systems similar to them — the systems described represent, due to their unique features, the classes of similar systems — are special cases of these definitions.

1.2.1 Study schedule and goals

In this study, we deal with the formalization of tableau methods. The result is a certain method of tableau systems construction, which corresponds to an intuitive approach, but apart from the precision we have already mentioned, it gives us additional benefits.

In Chapter Two, we will define the tableau system for Classical Propositional Logic as the basic case. As in work we focus on the economy principle when applying technical measures, we shall define this system based on set of tableau expressions Te equal to set of Classical Propositional Logic formulas For, although the tableau system for the Classical Propositional Logic could be defined based on set $\mathsf{Te} \neq \mathsf{For}$. Formally, this would be of course a different tableau system.

Hence, the case of a tableau system for the Classical Propositional Logic described in that chapter seems to be a borderline case. The system also features the property that any branch that begins with a finite set of tableau expressions can be extended to a branch that can no longer be governed by tableau rules and which is a branch of a finite length. 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.

At the end of the chapter, we show that we actually have defined a tableau system for the Classical Propositional Logic because the tableau provability, which it

determines, coincides with the consequence relation determined by the valuation of formulas.

In Chapter Three, we consider the tableau system for the simplest term logic (which we call Term Logic), i.e. the logic of classical and non-modal categorical propositions, allowing empty names. In the case of this tableau system for Term Logic, the set of formulas is actually contained in the set of tableau expressions. The proof language is therefore more complicated than in the previous case.

The issue of tableau concepts looks analogous, especially the key for proofs — as we will see — concept of the maximal branch. Also in this case, a maximal branch is always obtainable from a finite set of tableau expressions.

At the end of Chapter Three, we show the completeness and consistency of the tableau system with the normally and semantically defined consequence relations in Term Logic.

Chapter Four deals with another borderline case — but this time it is a borderline case on the opposite side to the case of Classical Propositional Logic. This is because we consider modal logic **S5**, by defining a set of tableau expressions in such a way that it neither coincides with the set of formulas of logic **S5** nor is it its proper superset — both sets are therefore disjoint. Also for this reason, the case is the most general one as the previously considered sets of expressions could also be defined, somewhat artificially, in such a way that the set of formulas would be disjoint from the set of tableau expressions. In the tableau system for logic **S5** those two sets are naturally disjoint.

We put *also* because in Chapter Four there emerges a problem with infinite branches. The systems described previously featured the property of a finite branch, which is not the case for the presented tableau system for **S5**. Therefore, it may happen that when constructing a branch and consequently a tableau which start with a finite set of expressions, it is not possible to finish them as some sequences of application of the rules become cyclical.

The lack of a finite branch property forces changes in some tableau concepts. This applies in particular to the concept of maximal branch and derived concepts. So, the tableau concepts defined in previous chapters become special cases of tableau concepts for systems that do not feature the property of a finite branch. The leading change is the generalisation of the concept of maximal branch. Intuitively, the maximal branch is the one to which we can no longer apply any rules. In previous cases, however, this meant that the maximal branch was of a finite length. It does not have to be the case this time. A maximal branch can be infinite, although not every infinite branch is a maximal one.

Despite the fact that, in our study, we do not focus on the issue of decidability in the tableau system, it is worth noting that the systems that feature the property of a finite branch, are decidible. For theoretically, always in a finite number of steps it is possible to construct a complete tableau for them — closed or open one, thus answering the question whether a given formula is or is not tableau provable on the grounds of given premises.

In the case of tableau systems that do not feature a finite branch property, these systems may not be decidable. So, although we prove that they are complete and sound in relation to the initial, semantically defined consequence relation, there does not have to exist a way of constructing an infinite tableau. Hence, starting from branches and tableaux as finite and potentially always constructible objects, and generalizing the tableau concepts, we come to theoretical branches or tableaux that are likely to exist, despite the fact that as setwise infinite objects, they cannot be defined by calculating their elements. However, as we said, in our study we do not deal directly with the problem of decidability in tableau systems, although examination of this problem is partly conditioned by the precision of the tableau concepts that we are working on.

In Chapter Five, we summarize the considerations of the previous chapters, defining general tableau concepts for propositional logic and term logic which cover all previous cases. We also show certain properties which, by virtue of these concepts, occur for the tableau systems defined by the presented method.

Finally, in Chapter Six, we apply general tableau concepts to a new case, showing how much their use simplifies the construction of a tableau system compared to previous cases where the tableau systems were defined from the simplest concepts. In this chapter we consider some modal term logic with the interpretation of modality *de re*. Since we already have tableau concepts for this type of systems, the construction of the tableau system and demonstrating that it is complete and consistent in relation to the semantics presented boils down to the statement of a few basic facts. We also present the application of general concepts to the case of modal logics determined by the semantics of possible worlds, showing how tableau concepts allow us to express conditions sufficient for a given tableau system for some modal logic to be sound and complete in relation to its semantics.

The formalisation of tableau concepts, which we propose, and its effects on the construction of tableau systems produce the main result of our study.

1.2.2 Terminology and issues in the book

Every time, before we move on to the construction of a tableau system, we start from a semantically defined logic, i.e. from ordered pair $\langle \mathsf{For}, \models \rangle$, where For is a

set of formulas, and $\vDash$ is a relation of semantic consequence, defined in a standard way on the basis of set of all interpretations of set For.

For such understood logic, we construct a tableau system which — as mentioned before — by analogy to the axiomatic system — can be understood as a ordered triple of sets $\langle \mathsf{For}, \mathsf{Te}, \mathbf{Rt} \rangle$.

Such a triple, through the application of the definitions proposed by us further, unambiguously determines ordered pair $\langle \mathsf{For}, \rhd \rangle$, where $\rhd$ is a tableau provability/branch consequence relation, defined by a general definition, but in any case based on a predefined set of tableau rules **Rt**.

Now, let us notice that having initial pair $\langle \mathsf{For}, \vDash \rangle$ and observing the rules of defining a tableau system, which we will present below, we can usually define at least several different tableau systems which differ in terms of set of expressions Te or set of rules **Rt**, or both sets at the same time. Despite the fact that these are formally different systems, they can define the relations of tableau provability identical in scope to relation $\vDash$.

The above remark explains why we will be writing e.g. a tableau system for Classical Propositional Logic rather than a tableau system of Classical Propositional Logic. Potentially, there are many tableau systems for Classical Propositional Logic, whose relations $\rhd$ coincide with consequence relation $\vDash$ of Classical Propositional Logic.

So, in each case, we write about a tableau system for given logic rather than a tableau system of given logic, taking account of the multitude of possible but equivalent tableau systems.

Although in the following chapters the starting point will always be some semantically defined logic $\langle \mathsf{For}, \vDash \rangle$, the construction principles for the tableau system presented by us can be used for the construction of a tableau system regardless of any pair $\langle \mathsf{For}, \vDash \rangle$. For a tableau system defined in this way, it is only then possible to search for the appropriate semantics.

Within the said method, a set of formulas For is an essential initial ingredient for the construction of a tableau system.

With a fixed set For, we define a set of tableau expressions Te, that is a set of expressions on the basis of which we carry out the tableau proofs. Of course, to each formula corresponds at least one tableau expression that represents it in the tableau proof. In the chapters where we present a full description of tableau systems for selected logics, we define set Te based on the previously mentioned principle of economy — we choose the option that seems to be the simplest. Then we define the concept of a tableau inconsistent set of expressions, i.e. a set which should be searched for in each track of proof/branch of the tableau proof.

Next, on at least two-element Cartesian products of set of all subsets of the set of expressions, we define the tableau rules. The tableau rules of a given system are therefore sets of two or more element, ordered n-tuples which comprise the subsets of set Te.

In this approach, we apply tableau rules to sets, and consequently we get one or possibly more sets, if the application of rule to a given set is possible at all. Already in Chapter Two we present the mechanisms that we generally impose on the rules to avoid redundant — for several reasons — and unproductive applications of the tableau rules. We consider these mechanisms to be an important feature of the formalisation adopted.

The effect of defining tableau rules is always set of tableau rules **Rt**. By analogy to axiomatic systems, we treat determination of set **Rt** as an axiomatization of some logic by means of tableau rules. On given set Te it is often possible — as we mentioned before — to define different sets of rules that produce the same effects in terms of consequences of the tableau system. In this study we will provide examples of different axiomatizations by means of tableau rules of the same semantically determined logic. However, since the proof that two axiomatizations are equivalent without reference to the metatheory of tableau systems seems complicated, we examine only one axiomatization — the one that we consider natural and the simplest.

Although the tableau rules are intended to determine the tableau system unambiguously, without further tableau concepts it is either not clear how the rules determine further concepts or it is only intuitive. Therefore, in three subsequent chapters we define all tableau concepts, and in the next one we define general tableau concepts, which in combination with specific tableau rules sets automatically determine the tableau system. If the tableau rules have certain properties, the tableau system they determine is complete and sound with respect to the initial and semantically defined relationship of consequence $\vDash$.

In the presented approach, a set of tableau rules is a starting point in the process of defining more complex concepts of the tableau system. From a heuristic point of view, a set of rules is the most important component of the tableau system, but in order to precisely describe the concept of the tableau proof, we need further concepts that are to be general and, in individual cases, to define the initial set of tableau rules.

By applying the rules, we create proof tracks/branches. In the book, we use the term of *branch*. A branch is such a sequence of sets that each set is basically contained in its successor (except the last one — if it exists). Branches are created by using tableau rules that allow the last set belonging to a branch to be expanded

in at least one way, so each branch element contains all the expressions present in the earlier elements of the branch.

Branches the last element of which contains a tableau inconsistent set are called *closed branches*. Such branches can no longer be extended because in our construction there are no rules whose premises would contain tableau inconsistent sets. Normally, branches that are not closed are called *open branches*.

There is one more, important from the viewpoint of the theory being developed, type of branches — maximal branches. Intuitively, the *maximal branch* is the one to which no more tableau rule can be adapted. Closed branches form a special type of maximal branches. However, maximal branches can also be open, even infinitely long, although, as we will see, the infinity of branches is neither a necessary nor a sufficient condition for its maximality.

A tableau proof is made up of branches that begin with the same set of expressions. We want to check whether formula A in a given tableau system is a consequence of set of formulas X. To this end, we construct branches which begin with expressions that are tableau equivalents of the formulas from set X and with the tableau equivalent of a formula contrary to formula A.

Hence, when constructing branches, we start from an assumption which is an indirect assumption. We develop branches in any way we want since the rules do not allow us to expand the initial sets in a way that would make the sequence of expressions cease to be a branch. Each non-maximal branch, which begins with a finite set of tableau expressions, can be extended — at least theoretically in the case of infinite branches — to a maximal branch, and each maximal branch is closed or open. In systems that do not have the property of a finite branch, we show that each branch beginning with a finite set of expressions can be extended to a maximal branch. In general, we also indicate conditions which, when met by set of tableau rules **Rt**, are sufficient to always obtain a maximal branch from a finite set of expressions. If now each maximal branch is closed, we conclude that formula A is *a branch consequence* (or *a tableau consequence*) of set of formulas X — for short $X \triangleright A$.

Sometimes we may be interested in the occurrence of branch consequence relation between infinite set of formulas Y and formula A. Then, an intuitive component appears in the proof — we have to show that there is such finite subset X of set Y that $X \triangleright A$. Although choosing the right set X is a non-formalised activity, we want the demonstration that $X \triangleright A$, where the property of a finite branch occurs, to be mechanical. This also distinguishes our approach to the tableau systems from other approaches. We do not restrict the branch consequences only to the finite sets.

It may even be practically impossible to check if there is a branch consequence relation in a given tableau system and in a given case, due to the number and complexity of branches. In order to reduce this complexity and make checking more practical, we introduce the concept of tableau/tree proof. In the study, we will use term *tableau*, although our concept of a tableau is far from the original one, i.e. a tableau with rows and columns. However, as it is customary to use this term, we will call the tableau proof a tableau.

As we have stated, the concept of tableau is intended to simplify checking whether there is a branch consequence in a given case. *Tableau* is always such a set of branches beginning with the same set of expressions that the existence of several different branches in a tableau implies the existence of tableau rules that led to the creation of these different branches.

Among tableaux we can distinguish *complete tableau* that, to put it intuitively, contain everything we need. Each branch belonging to such a tableau is a maximal one, and what is more, adding another branch to the complete tableau causes the obtained object not to be a tableau. We are also considering the issue of so-called *redundant branch variants*, i.e. branches that may or may not belong to a complete tableau.

A tableau that demonstrates the occurrence of branch consequence is a *closed tableau*, i.e. a complete tableau in which each branch is closed. It is the construction of a *closed tableau* that we consider to be a tableau proof, while the equivalence of a closed tableau existence with the occurrence of a branch consequence can be considered as a test of a good definition of a tableau system.

The concept of branch consequence is based on the inclusion of set of maximal branches in the set of closed branches. Most often the construction of a single closed tableau means the selection of a sufficient and appropriate subset of maximal branches that are closed. Theoretically, therefore, it is possible to construct many closed tableaux, but when constructing a tableau system, we should strive to show that one such tableau will suffice to recognise the occurrence of the tableau consequence. This postulate is our guiding principle in further studies on tableau systems.

On the other hand, it may happen that we will construct a complete tableau, which at the same time will be *open*, i.e. contain at least one maximal and open branch. The existence of such a tableau is a proof that the tableau consequence does not occur. The description of a complete and open tableau forms the basis for the construction of a counter-model, i.e. it allows to proceed to purely semantic concepts.

In the description of the method of construction of tableau systems we do not take account of the issue of effectiveness or length of proof. We do not aspire to

create systems in which the proofs will be as short as possible and whose computer implementations will be as effective as possible. Our main goal is to formalise the tableau methods, and consequently to precisely define the tableau concepts. Despite this, both in the way we formulate tableau rules, consider the redundant branch variants or treat the existence of an appropriate tableau as a measure of the branch consequence occurrence, we are guided by a certain need for economy and efficiency. However, its implementation is contained in the very tableau concepts, rather that in the strategy of applying tableau rules or the strategy of carrying out a proof — in these aspects we leave room for discretion.

Our construction of the tableau systems is fully related to the set theory:

- the tableau rules are sets of ordered n-tuples, which comprise the sets of tableau expressions
- the branches are sequences of sets that are monotonic due to the inclusion
- tableaux are sets of branches.

When in the subsequent chapters considering three different cases of logic, for which we define tableau systems with this method, we aim to determine general tableau concepts which would reduce the construction of a tableau system to define a set of tableau rules according to a given scheme and to specify several concepts in more detail. The other concepts we have described would be constant and general.

We also want to simplify the construction of the tableau system, which would be complete and sound in relation to the semantically established logic, to check if there occur several system-specific facts.

Thus, in the book we present the formalisation of tableau methods, which not only brings a certain level of precision, but also leads to generalizations which allow to simplify the construction of the tableau system for activities that require intuition. On the other hand, in the case of determining the completeness and soundness with respect to semantics, they require only those facts that are specific to the system under examination.

Due to the fact that we usually deal with tableau methods which are applied more intuitively, further considerations contained in the book may raise the question whether the formalism presented in the book fits into the tableau methods, or whether it is something different, but similar.

Since the transition from intuitively understood tableaux to formalism proposed in the book and, in some cases, the transition from the concept of tableau in the proposed approach to the standard tableau is quite straightforward, the approach described in the book seems to be no less general than the standard

approach. Apart from its already mentioned advantages, it contains all these intuitive ways of constructing tableau systems for propositional logic and term logic based on bivalent semantics.

1.3 Notations and concepts of the set theory

In this book we define all tableau concepts using the concepts of the set theory. We use standard denotations on the relations between sets or their elements: $\in$, $\subseteq$, $\supseteq$, $\subset$, $\supset$, $=$, $\neq$, etc. Also in a standard way we denote operations on sets $\cap$, $\bigcap$, $\cup$, $\bigcup$, $\setminus$, and $P(X)$, where X is a set, while $P(X)$ is a set of all subsets contained in set X. An empty set is denoted as $\varnothing$.

Sometimes, in metalanguage, we also use quantifiers $\exists$, $\forall$ and classical constants: $\Rightarrow$, $\Leftrightarrow$, &, etc.

With established set X, we shall state that its subset $Y \subseteq X$, which meets certain established condition (a), is a *maximal* set among subsets X which meet condition (a) iff there is no subset $Z \subseteq X$ such that Z meets condition (a) and $Y \subset Z$. On the other hand, with established set X, we shall state that its subset $Y \subseteq X$, which meets certain established condition (a), is a *minimal* set among subsets X which meet condition (a) iff there is no subset $Z \subseteq X$ such that Z meets condition (a) and $Z \subset Y$.

In the study we will very often use numbers from the set of natural numbers as well as the set of natural numbers itself $\mathbb{N}$ (without zero). Sometimes we will refer to the concept of set cardinality, i.e. its cardinal number, defining it with the use of notation $|X|$, where X is set. However, the cardinalities of the sets under consideration will never be greater than $|\mathbb{N}|$ cardinality of the set of natural numbers. When saying that given set X is *finite*, we will mean that $|X| < |\mathbb{N}|$.

We also often define multifold Cartesian products $X_1 \times \cdots \times X_n$, where $n \in \mathbb{N}$, $n \geq 2$ and $X_1, \ldots, X_n$ are sets. Elements of product $X_1 \times \cdots \times X_n$ are ordered n-tuples $\langle a_1, \ldots, a_n \rangle$, where $a_1 \in X_1, \ldots, a_n \in X_n$.

We also use the concept of function, using different letters or symbols, depending on the needs, to denote a function, e.g. $f : X \longrightarrow Y$.

We shall state that function f is *injective* or it is *injection* iff for any $x, y \in X$ it is the case that if $x \neq y$, then $f(x) \neq f(y)$. In turn, f is function *onto* iff for each $y \in Y$ there exists such $x \in X$ that $f(x) = y$. A function which is injective and *onto*, shall be called *bijection*.

We know that when f is a bijection, then there exists precisely one *inverse function* $g : Y \longrightarrow X$, i.e. such that for any $x \in X$, $g(f(x)) = x$. Function g will be denoted f^{-1}.

One of the fundamental concepts defined and used in the study is the concept of branch. With an established set of tableau expressions Te, a branch will take the form of function $\phi : K \longrightarrow P(\mathsf{Te})$, where $K = \{1, 2, 3, \ldots, n\}$ or $K = \mathbb{N}$, and ϕ meets

certain additional conditions that are specified in the tableau theory, discussed further. Assuming that the cardinality of set K equals n, for certain $n \in \mathbb{N}$, we shall state that branch ϕ has *length of* n or is n *long*. In turn, if $|K| = |\mathbb{N}|$, we shall state that branch ϕ is *infinite* or is *infinitely long*.

Hence, branches are special types of sequences that to each natural number belonging to its domain assign a certain set of tableau expressions. Branches can be denoted in all possible ways in which we denote sequences, so we can list all the elements of branches with indices, describe a branch as a ordered set or an ordered n-tuple, if $|K| = n$, where $n \in \mathbb{N}$.

Branches are also sets and therefore we can define different relations on them, similarly as on sets. Assume we have two branches $\phi : K \longrightarrow P(\mathsf{Te})$ and $\psi : M \longrightarrow P(\mathsf{Te})$. Thus $\phi \subseteq \psi$ iff $K \subseteq M$ and for any number $i \in K$, $\phi(i) = \psi(i)$. In turn, $\phi = \psi$ iff $\phi \subseteq \psi$ and $\psi \subseteq \phi$. And finally, $\phi \subset \psi$ iff $\phi \subseteq \psi$ and $\phi \neq \psi$.