Preface

Following the *Stanford Encyclopedia of Philosophy*,

> "the term *temporal logic* has been broadly used to cover all approaches to the representation of temporal information within a logical framework".

Applications of temporal logic include philosophical issues about time, the semantics of tenses in natural languages, and its use as a formal framework for the treatment of behavioural aspects of computerized systems.

In a more narrow sense, temporal logic is understood as a *modal-logic* type of approach: temporal relationships between different assertions are expressed by applying particular temporal logic operators to them. This book focuses on this type of temporal logic and we will study computer science applications to what we call *state systems*: systems which involve "states" and exhibit "behaviours" by "running" through sequences of such states.

One of the most challenging problems facing today's software engineers and computer scientists is to find ways and establish techniques to reduce the number of errors in the systems they build. It is widely acknowledged that formal methods may contribute to solving this challenge with significant success. In particular, temporal logic is a well-established and successfully used formal tool for the *specification* and *verification* of state systems. Its formulas are interpreted over "runs" of such systems and can thus express their behavioural properties. The means induced by the (semantical and deductive) logical apparatus provide methods to formally prove such properties.

This monograph is written in the tradition of the first author's textbook [83]

> *Temporal Logic of Programs*

and the two volumes

> *The Temporal Logic of Reactive and Concurrent Systems – Specification* and
> *The Temporal Logic of Reactive and Concurrent Systems – Safety*

of Manna and Pnueli [102, 104]. This means that we will present the "mathematics" of temporal logic in considerable detail and we will then systematically study

specification and verification methods, which will be illustrated by fully elaborated examples.

Compared with those books, however, the topics and their presentation are rearranged and we have included significant new material and approaches. In particular, branching time logics, expressiveness issues of temporal logic, aspects related to Lamport's *Temporal Logic of Actions* (TLA), and model checking methods are additionally presented.

There is a wealth of relevant and interesting material in the field. The "main text" of this book presents topics that – in our opinion – constitute a "canonical" exposition of the field. In additional *Second Reading* paragraphs we have occasionally inserted short "excursions" that expand on related or advanced themes or that present interesting complements. These paragraphs can be skipped without loss of continuity in the main presentation.

The first chapter of this book gives a short overview of basic concepts and notions of (mathematical) logic. This is not only to introduce the reader not familiar with logic into that world, it also defines basic terminology and notation that we use throughout the remaining text.

Chapters 2–5 and 10 form the purely logical part of the book. Even when restricted to the modal-logic type as mentioned above, there are many different versions and variants of temporal logic. We start in Chap. 2 with the basic propositional linear temporal logic and study in Chap. 3 some important propositional extensions. It should be mentioned that even the borderline between temporal logic(s) and modal logics is not really well defined. Some relationships concerning this are briefly discussed in Second Reading paragraphs.

Chapter 4 is devoted to the expressiveness of propositional linear temporal logics. In particular, the logics are compared with other description formalisms: classical predicate logic and ω-automata.

Chapter 5 introduces first-order linear temporal logic together with some additional useful extensions. Chapter 10 discusses some other temporal logics and, particularly, introduces branching time logics.

The remaining Chaps. 6–9 and 11 deal with applications of temporal logics to state systems. Various versions of *transition systems* – as formal representations of such systems – are introduced in Chap. 6, and Chap. 7 gives a general systematic presentation of (deductive) temporal logic verification methods for them. Chapter 8 applies the methods to the special ("classical") case of the verification of concurrent programs.

Chapter 9 addresses aspects that arise when system specifications are "structured". Particularly, the refinement of specifications is considered and we study how this can be described in the logic TLA.

The "semantical" model checking approach to system verification offers an alternative to deductive methods. It has attracted much interest, largely because it can be fully automated in a way that scales to systems of interesting complexity. Chapter 11 presents the essential concepts and techniques underlying this approach.

Every chapter ends with some bibliographical notes referring to the relevant literature. After the last chapter we include an extensive list of formal laws of the various temporal logics studied in this book.

We have used drafts of this book as supports for courses at the advanced undergraduate and the graduate level. Different selections of the material are possible, depending on the audience and the orientation of the course. The book is also intended as an introduction and reference for scientists and practicing software engineers who want to familiarize themselves with the field. We have aimed to make the presentation as self-contained as possible.

We are indebted to P. Fontaine, M. Hammer, A. Knapp, and H. Störrle for helpful remarks during the preparation of this text.

Finally, we thank Springer-Verlag for the interest in publishing this book and the overall support during its completion.

Munich and Nancy, January 2008

Fred Kröger
Stephan Merz

1

Basic Concepts and Notions of Logics

In this book various temporal logics will be studied. In preparation, we first introduce some basic concepts, notions, and terminology of logics in general by means of a short overview of *classical logic*. Particularly, items are addressed which will be of relevance in subsequent considerations. This includes some well-known results from classical logic which we list here without any proofs.

1.1 Logical Languages, Semantics, and Formal Systems

A logic formalizes the reasoning about "statements" within some area of application. For this purpose, it provides formal languages containing *formulas* for the representation of the statements in question and formal concepts of reasoning like *consequence* and *derivability* relations between formulas.

Classical (mathematical) logic applies to mathematical systems: number systems such as the natural or real numbers, algebraic systems such as groups or vector spaces, etc. In a separable "nucleus" of this logic, called *propositional logic* PL, the effect of building formulas with boolean operators like *and*, *or*, *implies*, etc. is studied, while the atomic building blocks of such formulas are viewed as "black boxes" without any further internal structure.

Generally, a logical language is given by an alphabet of different symbols and the definition of the set of formulas which are strings over the alphabet. Given a set $\mathbf{V}$ whose elements are called *propositional constants*, a *language* $\mathcal{L}_{\mathrm{PL}}(\mathbf{V})$ (also shortly: $\mathcal{L}_{\mathrm{PL}}$) *of propositional logic* can be defined as follows.

Alphabet

- All propositional constants of $\mathbf{V}$,
- the symbols **false** $\mid \rightarrow \mid (\mid)$.

(The stroke $\mid$ is not a symbol but only used for separating the symbols in the list.)

Formulas

1. Every propositional constant of $\mathbf{V}$ is a formula.
2. **false** is a formula.
3. If A and B are formulas then $(A \to B)$ is a formula.

The clauses 1–3 (also called *formation rules*) constitute an *inductive* definition which may be understood to work like a set of production rules of a formal grammar: a string over the alphabet is a formula if and only if it can be "produced" by finitely many applications of the rules 1–3.

The set $\mathbf{V}$ is a parameter in this definition. For concrete applications, $\mathbf{V}$ has to be fixed yielding some particular language tailored for the "universe of discourse" in question. There are no assumptions on how many elements $\mathbf{V}$ may have. This most general setting may sometimes cause some technical complications. In applications studied in this book we do not need the full generality, so we actually may restrict $\mathbf{V}$ to be finite or "at most" denumerable.

In general, a logical language is called *countable* if its alphabet is finite or denumerable. We will tacitly assume that all the languages still to be defined subsequently will be countable in this sense.

The symbol $\to$ is a (binary) *logical operator*, called *implication*; **false** is a special formula. Further logical operators and another distinguished formula **true** can be introduced to abbreviate particular formulas.

Abbreviations

$$\begin{aligned}
&\neg A \;\equiv\; A \to \mathbf{false},\\
&A \vee B \;\equiv\; \neg A \to B,\\
&A \wedge B \;\equiv\; \neg(A \to \neg B),\\
&A \leftrightarrow B \;\equiv\; (A \to B) \wedge (B \to A),\\
&\mathbf{true} \;\equiv\; \neg\,\mathbf{false}.
\end{aligned}$$

(We have omitted surrounding parentheses and will do so also in the following. By $\equiv$ we denote equality of strings.) The operators $\neg$, $\vee$, $\wedge$, and $\leftrightarrow$ are called *negation*, *disjunction*, *conjunction*, and *equivalence*, respectively.

The symbols A and B in such definitions are not formulas (of some $\mathcal{L}_{\mathrm{PL}}$) themselves but *syntactic variables* ranging over the set of formulas. Accordingly, a string like $\neg A \to B$ is not a formula either. It yields a formula by substituting proper formulas for A and B. Nevertheless, we freely use wordings like "formula A" or "formula $\neg A \to B$" to avoid more precise but complicated formulations like "formula of the form $\neg A \to B$ where A and B stand for formulas". Moreover, we speak of formulas "of PL" since the concrete language $\mathcal{L}_{\mathrm{PL}}$ is not relevant in this notation. In all the other logics developed subsequently, we will adopt these conventions accordingly.

The language definition of a logic constitutes its *syntax*. Its *semantics* is based on formal *interpretations* J of the syntactical elements together with a notion of *validity in* (or *satisfaction by*) J.

In the case of PL, interpretations are provided by (*boolean*) *valuations*. Given two distinct *truth values*, denoted by ff ("false") and tt ("true"), a valuation B for a

set $\mathbf{V}$ of propositional constants is a mapping

$$\mathsf{B} : \mathbf{V} \ \rightarrow \ \{\mathsf{ff}, \mathsf{tt}\}.$$

Every such B can be inductively extended to the set of all formulas of $\mathcal{L}_{\mathrm{PL}}(\mathbf{V})$:

1. $\mathsf{B}(v)$ for $v \in \mathbf{V}$ is given.
2. $\mathsf{B}(\textbf{false}) = \mathsf{ff}$.
3. $\mathsf{B}(A \rightarrow B) = \mathsf{tt} \ \Leftrightarrow \ \mathsf{B}(A) = \mathsf{ff}$ or $\mathsf{B}(B) = \mathsf{tt}$.

(We use $\Leftrightarrow$ as an abbreviation for "if and only if"; later we will also use $\Rightarrow$ for "if…then…".) This also defines B for the formula abbreviations above:

4. $\mathsf{B}(\neg A) = \mathsf{tt} \ \Leftrightarrow \ \mathsf{B}(A) = \mathsf{ff}$.
5. $\mathsf{B}(A \vee B) = \mathsf{tt} \ \Leftrightarrow \ \mathsf{B}(A) = \mathsf{tt}$ or $\mathsf{B}(B) = \mathsf{tt}$.
6. $\mathsf{B}(A \wedge B) = \mathsf{tt} \ \Leftrightarrow \ \mathsf{B}(A) = \mathsf{tt}$ and $\mathsf{B}(B) = \mathsf{tt}$.
7. $\mathsf{B}(A \leftrightarrow B) = \mathsf{tt} \ \Leftrightarrow \ \mathsf{B}(A) = \mathsf{B}(B)$.
8. $\mathsf{B}(\textbf{true}) = \mathsf{tt}$.

A formula A of $\mathcal{L}_{\mathrm{PL}}$ is called *valid in* B (or B *satisfies* A), denoted by $\models_{\mathsf{B}} A$, if $\mathsf{B}(A) = \mathsf{tt}$.

Based on this notion, the *consequence relation* and *(universal) validity* in PL are defined. Let A be a formula, $\mathcal{F}$ a set of formulas of $\mathcal{L}_{\mathrm{PL}}$.

- A is called a *consequence of* $\mathcal{F}$ if $\models_{\mathsf{B}} A$ holds for every valuation B with $\models_{\mathsf{B}} B$ for all $B \in \mathcal{F}$.
- A is called *(universally) valid* or a *tautology* if it is a consequence of the empty set of formulas, i.e., if $\models_{\mathsf{B}} A$ holds for every B.

The pattern of this definition will occur analogously for all other subsequent logics with other interpretations. For any logic,

$$\mathcal{F} \models A, \ \text{also written} \ B_1, \ldots, B_n \models A \ \text{if} \ \mathcal{F} = \{B_1, \ldots, B_n\}, n \geq 1$$

will denote that A is a consequence of $\mathcal{F}$, and

$$\models A$$

will denote that A is valid.

With these definitions there are two possible formal statements (in PL) of what informally is expressed by a phrase like "B follows from A". The first one is asserted by implication within the language:

$$A \rightarrow B.$$

The second one is given by the consequence relation:

$$A \models B.$$

A fundamental fact of classical (propositional) logic is that these notions are equivalent:

$$A \models B \;\Leftrightarrow\; \models A \to B$$

or more generally ($\mathcal{F}$ being an arbitrary set of formulas):

$$\mathcal{F} \cup \{A\} \models B \;\Leftrightarrow\; \mathcal{F} \models A \to B$$

which can be "unfolded" for finite $\mathcal{F}$ to

$$A_1, \ldots, A_n \models B \;\Leftrightarrow\; \models A_1 \to (A_2 \to \ldots \to (A_n \to B)\ldots)$$

or, equivalently, to the more readable

$$A_1, \ldots, A_n \models B \;\Leftrightarrow\; \models (A_1 \wedge \ldots \wedge A_n) \to B.$$

Note that we write $(A_1 \wedge \ldots \wedge A_n)$ without inner parentheses, which is syntactically not correct but justified as a shortcut by the fact that the real bracketing is of no relevance (more formally: the operator $\wedge$ is associative). The analogous notation will be used for disjunctions.

Validity and consequence are key notions of any logic. Besides their semantical definitions they can (usually) be described in a *proof-theoretical* way by *formal systems*. A formal system Σ for a logical language consists of

- a set of formulas of the language, called *axioms*,
- a set of (*derivation*) *rules* of the form $A_1, \ldots, A_n \vdash B \;\; (n \geq 1)$.

The formulas $A_1, \ldots, A_n$ are called the *premises*, the formula B is the *conclusion* of the rule. To distinguish it from other existing forms, a formal system of this kind is called *Hilbert-like*. Throughout this book we will use only this form.

The *derivability* (in formal system Σ) of a formula A *from* a set $\mathcal{F}$ of formulas (*assumptions*), denoted by $\mathcal{F} \vdash_{\Sigma} A$ or $\mathcal{F} \vdash A$ when Σ is understood from the context, is defined inductively:

1. $\mathcal{F} \vdash A$ for every axiom.
2. $\mathcal{F} \vdash A$ for every $A \in \mathcal{F}$.
3. If $\mathcal{F} \vdash A$ for all premises A of a rule then $\mathcal{F} \vdash B$ for the conclusion of this rule.

A formula A is called *derivable*, denoted by $\vdash_{\Sigma} A$ or $\vdash A$, if $\emptyset \vdash A$. If A is derivable from some $A_1, \ldots, A_n$ then the "relation" $A_1, \ldots, A_n \vdash A$ can itself be used as a *derived rule* in other derivations.

For languages of PL there are many possible formal systems. One of them, denoted by Σ_{PL}, is the following.

Axioms

- $A \to (B \to A)$,
- $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$,
- $((A \to \mathbf{false}) \to \mathbf{false}) \to A$.

Rule

- $A, A \to B \vdash B$ (*modus ponens*).

We remark once more that the strings written down are not formulas. So, for example, $A \to (B \to A)$ is not one axiom but an *axiom scheme* which yields infinitely many axioms when formulas are substituted for A and B. Actually, Σ_{PL} is written in a form which is independent of the concrete language of PL. So we may call Σ_{PL} a formal system "for PL". In the same sense we will subsequently give formal systems for other logics. A formal system for a logic is also called its *axiomatization*.

Like the semantical consequence relation $\models$, derivability is related to implication in the following sense:

$$\mathcal{F} \cup \{A\} \vdash B \;\Leftrightarrow\; \mathcal{F} \vdash A \to B.$$

The only if part of this fact is called the *Deduction Theorem* (of PL).

An indispensable requirement of any reasonable formal system is its *soundness* with respect to the semantical notions of the logic. Σ_{PL} is in fact sound, which means that

$$\mathcal{F} \vdash_{\Sigma_{\mathrm{PL}}} A \;\Rightarrow\; \mathcal{F} \models A$$

holds for every $\mathcal{F}$ and A. Moreover, it can also be shown that

$$\mathcal{F} \models A \;\Rightarrow\; \mathcal{F} \vdash_{\Sigma_{\mathrm{PL}}} A$$

which states the *completeness* of Σ_{PL}. As a special case both facts imply

$$\vdash_{\Sigma_{\mathrm{PL}}} A \;\Leftrightarrow\; \models A$$

for every formula A.

A formal system allows for "producing" formulas by applying rules in a "mechanical" way. So, a particular effect of the latter relationship is that the set of valid formulas of (any language of) PL can be "mechanically generated" (in technical terms: it is *recursively enumerable*). Moreover, this set is *decidable* (shortly: PL is decidable), i.e., there is an algorithmic procedure to decide for any formula whether it is valid.

We illustrate the main concepts and notions of this section by an example. A simple logical principle of reasoning is informally expressed by

"If B follows from A and C follows from B then C follows from A".

This *chaining rule* can formally be stated and verified in several ways: we can establish the formula

$$F \;\equiv\; ((A \to B) \wedge (B \to C)) \to (A \to C)$$

as valid, i.e., $\models F$ (speaking semantically), or as derivable, i.e., $\vdash F$ (speaking proof-theoretically), or we can express it by

$$A \to B, B \to C \vDash A \to C \quad (\text{or } A \to B, B \to C \vdash A \to C).$$

The proofs for the semantical formulations are straightforward from the definitions. As an example of a formal derivation within the formal system Σ_{PL} and in order to introduce our standard format of such proofs we derive $A \to C$ from $A \to B$ and $B \to C$, i.e., we show that $A \to B, B \to C \vdash A \to C$:

(1)	$A \to B$	assumption
(2)	$B \to C$	assumption
(3)	$(B \to C) \to (A \to (B \to C))$	axiom
(4)	$A \to (B \to C)$	modus ponens,(2),(3)
(5)	$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$	axiom
(6)	$(A \to B) \to (A \to C)$	modus ponens,(4),(5)
(7)	$A \to C$	modus ponens,(1),(6)

In each of the numbered steps (lines) we list some derivable formula and indicate on the right-hand side if it is an axiom or an assumption or by what rule applied to previous lines it is found.

We add a selection of some more valid formulas. These and other tautologies will (very often implicitly) be used in the subsequent chapters.

- $A \vee \neg A$,
- $\neg\neg A \leftrightarrow A$,
- $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$,
- $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$,
- $((A \wedge B) \to C) \leftrightarrow (A \to (B \to C))$,
- $(A \to B) \leftrightarrow (\neg B \to \neg A)$,
- $(A \wedge \mathbf{true}) \leftrightarrow A$,
- $(A \vee \mathbf{false}) \leftrightarrow A$,
- $(A \to C) \to ((A \wedge B) \to C)$,
- $((A \vee B) \to C) \to (A \to C)$.

As mentioned at the beginning, PL formalizes (a part of) reasoning about statements in mathematical systems. Its semantics formalizes the natural basic point of view of "usual" mathematics that a statement is something which is either "false" or "true". We still remark that, for specific applications or propagated by philosophical considerations, there are other "non-standard" semantical concepts as well. Examples are *three-valued logic* (a statement can have three different truth values which may be understood as "false", "possible", or "true"), *probabilistic logic* (truth is given with a certain probability), or *intuitionistic logic* (statements are interpreted *constructively* which, e.g., means that the *tertium non datur* formula $A \vee \neg A$ is no longer valid since it might be that neither the truth nor the falsity of A can be found in a constructive way).

1.2 Classical First-Order Logic

Mathematical statements and argumentations usually need more means than can be represented in propositional logic. These are provided by extending PL to *predicate logic* which investigates a more detailed structure of formulas dealing with *objects*, *functions*, and *predicates*, and includes the concept of *quantification* with operators like *for some* and *for all*.

The standard form of predicate logic is *first-order logic* FOL which we describe in its *many-sorted* version as follows.

A *signature* $SIG = (\mathbf{S}, \mathbf{F}, \mathbf{P})$ is given by

- a set $\mathbf{S}$ of *sorts*,
- $\mathbf{F} = \bigcup_{\vec{s}\in\mathbf{S}^*, s\in\mathbf{S}} \mathbf{F}^{(\vec{s},s)}$ where $\mathbf{F}^{(\vec{s},s)}$, for every $\vec{s} \in \mathbf{S}^*$ and $s \in \mathbf{S}$, is a set of *function symbols* (also called *individual constants* in the case of $\vec{s} = \varepsilon$),
- $\mathbf{P} = \bigcup_{\vec{s}\in\mathbf{S}^*} \mathbf{P}^{(\vec{s})}$ where $\mathbf{P}^{(\vec{s})}$, for every $\vec{s} \in \mathbf{S}^*$, is a set of *predicate symbols* (also called *propositional constants* in the case of $\vec{s} = \varepsilon$).

($\mathbf{S}^*$ denotes the set of finite strings over $\mathbf{S}$; ε is the empty string.) For $f \in \mathbf{F}$ we will often write $f^{(\vec{s},s)}$ to indicate that f belongs to $\mathbf{F}^{(\vec{s},s)}$, and analogously for $p \in \mathbf{P}$.

Given a signature $SIG = (\mathbf{S}, \mathbf{F}, \mathbf{P})$, a *first-order language* $\mathcal{L}_{\mathrm{FOL}}(SIG)$ (also shortly: $\mathcal{L}_{\mathrm{FOL}}$) is given by the following syntax.

Alphabet

- All symbols of $\mathbf{F}$ and $\mathbf{P}$,
- for every $s \in \mathbf{S}$ denumerably many (*individual*) *variables*,
- the *equality symbol* $=$,
- the symbols **false** $\mid \rightarrow \mid \exists \mid , \mid (\mid)$.

We will denote the set of variables for $s \in \mathbf{S}$ by $\mathcal{X}_s$ and define $\mathcal{X} = \bigcup_{s\in\mathbf{S}} \mathcal{X}_s$. Strictly speaking, $\mathcal{L}_{\mathrm{FOL}}(SIG)$ does not only depend on the given signature SIG but also on the choice of (the notations for) all these variables. We do not display this dependence since $\mathcal{X}$ could also be fixed for all languages. Note that requesting each $\mathcal{X}_s$ to be denumerable is only for having "enough" variables available.

Terms and *their sorts* (inductively defined):

1. Every variable of $\mathcal{X}_s$ is a term of sort s.
2. If $f \in \mathbf{F}^{(s_1\cdots s_n, s)}$ is a function symbol and t_i are terms of sorts s_i for $1 \leq i \leq n$ then $f(t_1, \ldots, t_n)$ is a term of sort s.

An *atomic formula* is a string of the form

- $p(t_1, \ldots, t_n)$, where $p \in \mathbf{P}^{(s_1\cdots s_n)}$ is a predicate symbol and t_i are terms of sorts s_i for $1 \leq i \leq n$, or
- $t_1 = t_2$, where t_1 and t_2 are terms of the same sort.

Formulas (inductively defined):

1. Every atomic formula is a formula.

2. **false** is a formula, and if A and B are formulas then $(A \to B)$ is a formula.
3. If A is a formula and x is a variable then $\exists x A$ is a formula.

We reuse the abbreviations from $\mathcal{L}_{\mathrm{PL}}$ and introduce two more:

$$\forall x A \;\equiv\; \neg\exists x \neg A,$$
$$t_1 \neq t_2 \;\equiv\; \neg\, t_1 = t_2.$$

Furthermore, we will write f instead of $f()$ for individual constants $f \in \mathbf{F}^{(\varepsilon,s)}$ and p instead of $p()$ for propositional constants $p \in \mathbf{P}^{(\varepsilon)}$; x, y and the like will be used to denote variables.

A variable x (more precisely: an occurrence of x) in a formula A is called *bound* if it appears in some part $\exists x B$ of A; otherwise it is called *free*. If t is a term of the same sort as x then $A_x(t)$ denotes the result of substituting t for every free occurrence of x in A. When writing $A_x(t)$ we always assume implicitly that t does not contain variables which occur bound in A. (This can always be achieved by replacing the bound variables of A by others.) A formula without any free variables is called *closed*. If A is a formula that contains no free occurrences of variables other than $x_1, \ldots, x_n$ then the (closed) formula $\forall x_1 \ldots \forall x_n A$ is called the *universal closure* of A.

As an example of a first-order language consider the signature

$$SIG_{gr} = (\{GR\}, \{NEL^{(\varepsilon,GR)}, \circ^{(GR\,GR,GR)}, INV^{(GR,GR)}\}, \emptyset).$$

The terms of $\mathcal{L}_{\mathrm{FOL}}(SIG_{gr})$ are the variables $x \in \mathcal{X}_{GR} = \mathcal{X}$ of the language, the individual constant NEL, and expressions of the form $\circ(t_1, t_2)$ or $INV(t)$ with terms t, t_1, t_2. All terms are of the sole sort GR. Since SIG_{gr} contains no predicate symbols the only atomic formulas are "equalities" $t_1 = t_2$ with terms t_1, t_2. The string

$$\forall x \circ(NEL, x) = x$$

is an example of a formula.

For the semantics of FOL, interpretations are given by *structures* which generalize the valuations of PL to the new situation. A structure S for a signature $SIG = (\mathbf{S}, \mathbf{F}, \mathbf{P})$ consists of

- $|\mathsf{S}| = \bigcup_{s \in \mathbf{S}} |\mathsf{S}|_s$ where $|\mathsf{S}|_s$ is a non-empty set (called *domain*) for every $s \in \mathbf{S}$,
- mappings $f^{\mathsf{S}} : |\mathsf{S}|_{s_1} \times \ldots \times |\mathsf{S}|_{s_n} \to |\mathsf{S}|_s$ for all function symbols $f \in \mathbf{F}^{(s_1 \ldots s_n, s)}$,
- mappings $p^{\mathsf{S}} : |\mathsf{S}|_{s_1} \times \ldots \times |\mathsf{S}|_{s_n} \to \{\mathsf{ff}, \mathsf{tt}\}$ for all predicate symbols $p \in \mathbf{P}^{(s_1 \ldots s_n)}$.

Note that for individual constants $f \in \mathbf{F}^{(\varepsilon,s)}$ we obtain $f^{\mathsf{S}} \in |\mathsf{S}|_s$. For $p \in \mathbf{P}^{(\varepsilon)}$ we have $p^{\mathsf{S}} \in \{\mathsf{ff}, \mathsf{tt}\}$ which justifies these p again being called propositional constants.

A *variable valuation* ξ (with respect to S) assigns some $\xi(x) \in |\mathsf{S}|_s$ to every variable $x \in \mathcal{X}_s$ (for all $s \in \mathbf{S}$). A structure together with a variable valuation ξ defines inductively a value $\mathsf{S}^{(\xi)}(t) \in |\mathsf{S}|$ for every term t:

1. $\mathsf{S}^{(\xi)}(x) = \xi(x)$ for $x \in \mathcal{X}$.
2. $\mathsf{S}^{(\xi)}(f(t_1, \ldots, t_n)) = f^{\mathsf{S}}(\mathsf{S}^{(\xi)}(t_1), \ldots, \mathsf{S}^{(\xi)}(t_n))$.

Furthermore, we can define $\mathsf{S}^{(\xi)}(A) \in \{\mathsf{ff}, \mathsf{tt}\}$ for every atomic formula:

1. $\mathsf{S}^{(\xi)}(p(t_1, \ldots, t_n)) = p^{\mathsf{S}}(\mathsf{S}^{(\xi)}(t_1), \ldots, \mathsf{S}^{(\xi)}(t_n))$.
2. $\mathsf{S}^{(\xi)}(t_1 = t_2) = \mathsf{tt} \;\Leftrightarrow\; \mathsf{S}^{(\xi)}(t_1)$ and $\mathsf{S}^{(\xi)}(t_2)$ are equal values in $|\mathsf{S}|_s$ (where s is the sort of t_1 and t_2).

Analogously to the valuations B in PL, $\mathsf{S}^{(\xi)}$ can be inductively extended to all formulas of $\mathcal{L}_{\mathrm{FOL}}$. Defining the relation $\sim_x$ for $x \in \mathcal{X}$ between variable valuations by

$$\xi \sim_x \xi' \;\Leftrightarrow\; \xi(y) = \xi'(y) \text{ for all } y \in \mathcal{X} \text{ other than } x,$$

the inductive clauses are:

1. $\mathsf{S}^{(\xi)}(A)$ for atomic formulas is already defined.
2. $\mathsf{S}^{(\xi)}(\mathbf{false}) = \mathsf{ff}$.
3. $\mathsf{S}^{(\xi)}(A \to B) = \mathsf{tt} \;\Leftrightarrow\; \mathsf{S}^{(\xi)}(A) = \mathsf{ff}$ or $\mathsf{S}^{(\xi)}(B) = \mathsf{tt}$.
4. $\mathsf{S}^{(\xi)}(\exists x A) = \mathsf{tt} \;\Leftrightarrow$ there is a ξ' such that $\xi \sim_x \xi'$ and $\mathsf{S}^{(\xi')}(A) = \mathsf{tt}$.

The truth values for formulas like $\neg A$, $A \vee B$, etc. result from these definitions as in PL, and for $\forall x A$ we obtain:

5. $\mathsf{S}^{(\xi)}(\forall x A) = \mathsf{tt} \;\Leftrightarrow\; \mathsf{S}^{(\xi')}(A) = \mathsf{tt}$ for all ξ' with $\xi \sim_x \xi'$.

The value $\mathsf{S}^{(\xi)}(A)$ depends only on the valuation of variables that have free occurrences in A. In particular, $\mathsf{S}^{(\xi)}(A)$ does not depend on the variable valuation ξ when A is a closed formula. This observation justifies our convention that bound variables are suitably renamed before a substitution $A_x(t)$ is performed.

A formula A of $\mathcal{L}_{\mathrm{FOL}}$ is called *valid in* S (or S *satisfies* A), denoted by $\models_{\mathsf{S}} A$, if $\mathsf{S}^{(\xi)}(A) = \mathsf{tt}$ for every variable valuation ξ. Following the general pattern from Sect. 1.1, A is called a *consequence of* a set $\mathcal{F}$ of formulas ($\mathcal{F} \models A$) if $\models_{\mathsf{S}} A$ holds for every S with $\models_{\mathsf{S}} B$ for all $B \in \mathcal{F}$. A is called (*universally*) *valid* ($\models A$) if $\emptyset \models A$.

Continuing the example considered above, a structure Z for the signature SIG_{gr} could be given by

$$|\mathsf{Z}| = |\mathsf{Z}|_{GR} = \mathbb{Z} \text{ (the set of integers)},$$
$$NEL^{\mathsf{Z}} = 0 \in \mathbb{Z},\ \circ^{\mathsf{Z}}(k, l) = k + l,\ INV^{\mathsf{Z}}(k) = -k \text{ (for } k, l \in \mathbb{Z}).$$

The formula $\forall x \circ(NEL, x) = x$ is valid in Z but not universally valid: consider the structure S that differs from Z by defining $NEL^{\mathsf{S}} = 1$. An example of a valid formula is

$$\circ(x, y) = INV(y) \to INV(y) = \circ(x, y).$$

More generally,

$$t_1 = t_2 \to t_2 = t_1$$

is a valid formula ("scheme") for arbitrary terms t_1, t_2 (of the same sort), and this formulation is in fact independent of the concrete signature SIG in the sense that it is valid in every $\mathcal{L}_{\mathrm{FOL}}(SIG)$ for terms t_1, t_2 that can be built within SIG.

The fundamental relationship

$$\mathcal{F} \cup \{A\} \models B \quad \Leftrightarrow \quad \mathcal{F} \models A \to B$$

between implication and consequence stated in Sect. 1.1 for PL has to be modified slightly in FOL. It holds if A does not contain free variables.

In contrast to PL, FOL is not decidable (for arbitrary first-order languages), but there exist again sound and complete axiomatizations of FOL. An example is given by the following formal system Σ_{FOL} (which uses x and y to denote variables).

Axioms

- All axioms of Σ_{PL},
- $A_x(t) \to \exists x A$,
- $x = x$,
- $x = y \to (A \to A_x(y))$.

Rules

- $A, A \to B \vdash B$,
- $A \to B \vdash \exists x A \to B$ if there is no free occurrence of x in B (*particularization*).

According to the remark above, the Deduction Theorem for FOL must be formulated with somewhat more care than in PL. A possible formulation is:

$$\mathcal{F} \cup \{A\} \vdash B \quad \Rightarrow \quad \mathcal{F} \vdash A \to B$$ if the derivation of B from $\mathcal{F} \cup \{A\}$ contains no application of the particularization rule involving a variable that occurs free in A.

Note in particular that the condition for the applicability of this rule is trivially fulfilled if A is a closed formula.

The converse connection holds without any restrictions (as in PL).

Again we list some valid formulas (now of FOL) in order to give an impression what kinds of such predicate logical facts may be used subsequently.

- $t_1 = t_2 \leftrightarrow t_2 = t_1$,
- $t_1 = t_2 \wedge t_2 = t_3 \to t_1 = t_3$,
- $\forall x A \to A_x(t)$,
- $\forall x(A \to B) \to (\forall x A \to \forall x B)$,
- $\exists x(A \vee B) \leftrightarrow (\exists x A \vee \exists x B)$,
- $\exists x(A \to B) \leftrightarrow (\forall x A \to B)$, x not free in B,
- $\exists x \forall y A \to \forall y \exists x A$.

Finally we note a derivable rule, called *generalization*, which is "dual" to the above particularization rule:

- $A \to B \vdash A \to \forall x B$ if there is no free occurrence of x in A.

1.3 Theories and Models

Languages of predicate logic provide a general linguistic framework for the description of mathematical systems. This framework is instantiated to specific systems by fixing an according signature. For example, the language $\mathcal{L}_{\mathrm{FOL}}(SIG_{gr})$ with

$$SIG_{gr} = (\{GR\}, \{NEL^{(\varepsilon,GR)}, \circ^{(GR\,GR,GR)}, INV^{(GR,GR)}\}, \emptyset),$$

mentioned in the previous section, is an appropriate language for formalizing groups: GR represents the underlying set of the group, NEL should be interpreted as the neutral element, $\circ$ as the group product, and INV as the inverse operation. This interpretation is formally performed by a structure, and in fact, the sample structure Z for SIG_{gr} of Sect. 1.2 is a group.

Valid formulas hold in all structures. Structures that "fit" the mathematical system in question can be distinguished by a set of formulas that are valid in those structures, though not necessarily universally valid. Formalizing this concept, a (*first-order*) *theory* $Th = (\mathcal{L}_{\mathrm{FOL}}(SIG), \mathcal{A})$ is given by a language $\mathcal{L}_{\mathrm{FOL}}(SIG)$ and a set $\mathcal{A}$ of formulas of $\mathcal{L}_{\mathrm{FOL}}(SIG)$, called the *non-logical axioms* of Th. A structure S for SIG satisfying all formulas of $\mathcal{A}$ is called a *model* of the theory Th. Given a class $\mathcal{C}$ of structures for a signature SIG, a *$\mathcal{C}$-theory* is a theory $Th = (\mathcal{L}_{\mathrm{FOL}}(SIG), \mathcal{A}_{\mathcal{C}})$ such that all structures of $\mathcal{C}$ are models of Th. A formula F of $\mathcal{L}_{\mathrm{FOL}}(SIG)$ is valid in all structures of $\mathcal{C}$ if

$$\mathcal{A}_{\mathcal{C}} \vdash_{\Sigma_{\mathrm{FOL}}} F$$

since Σ_{FOL} is sound and therefore $\mathcal{A}_{\mathcal{C}} \vdash_{\Sigma_{\mathrm{FOL}}} F$ implies $\mathcal{A}_{\mathcal{C}} \models F$.

With these definitions, the theory $Group = (\mathcal{L}_{\mathrm{FOL}}(SIG_{gr}), \mathcal{G})$ with $\mathcal{G}$ consisting of the formulas

$$(x_1 \circ x_2) \circ x_3 = x_1 \circ (x_2 \circ x_3),$$
$$NEL \circ x = x,$$
$$INV(x) \circ x = NEL$$

– where we write $x_1 \circ x_2$ instead of $\circ(x_1, x_2)$ – is a (first-order) group theory. More precisely, $Group$ is a $\mathcal{C}_{gr}$-theory where $\mathcal{C}_{gr}$ is the class of all structures G for SIG_{gr} such that the set $|\mathsf{G}| = |\mathsf{G}|_{GR}$ together with the interpretations NEL^{G}, $\circ^{\mathsf{G}}$, and INV^{G} form a group. The non-logical axioms of $\mathcal{G}$ are just well-known group axioms. Formulas F valid in groups can be obtained within the logical framework by derivations

$$\mathcal{G} \vdash_{\Sigma_{\mathrm{FOL}}} F.$$

The axioms of $\mathcal{G}$ contain free variables. Sometimes it might be convenient to write such axioms in "closed form" by taking their universal closures, e.g.,

$$\forall x_1 \forall x_2 \forall x_3 ((x_1 \circ x_2) \circ x_3 = x_1 \circ (x_2 \circ x_3))$$

instead of the first axiom above. The definition of validity in a structure implies that any structure satisfies a formula A if and only if it satisfies the universal closure of A; therefore the two versions of how to write the axioms are in fact equivalent.

We give some more examples of theories: let

$$SIG_{lo} = (\{ORD\}, \emptyset, \{\prec^{(ORD\ ORD)}\}),$$
$$LinOrd = (\mathcal{L}_{\mathrm{FOL}}(SIG_{lo}), \mathcal{O})$$

with $\mathcal{O}$ consisting of the axioms (in non-closed form)

$$\neg\, x \prec x,$$
$$(x_1 \prec x_2 \wedge x_2 \prec x_3) \rightarrow x_1 \prec x_3,$$
$$x_1 \neq x_2 \rightarrow (x_1 \prec x_2 \vee x_2 \prec x_1).$$

(As for $\circ$ in $\mathcal{L}_{\mathrm{FOL}}(SIG_{gr})$, we mostly use infix notation – here and in the following – for "binary" function and predicate symbols.) Every structure O where $|\mathsf{O}| = |\mathsf{O}|_{ORD}$ is a non-empty set and $\prec^{\mathsf{O}}$ is a (strict) linear order on $|\mathsf{O}|$ is a model of $LinOrd$ which, hence, may be called a linear order theory.

A natural number theory Nat is based on a signature

$$SIG_{Nat} = (\{NAT\}, \mathbf{F}, \emptyset)$$

with

$$\mathbf{F} = \{0^{(\varepsilon, NAT)}, SUCC^{(NAT,NAT)}, +^{(NAT\ NAT,NAT)}, *^{(NAT\ NAT,NAT)}\}.$$

In the intended structure N for SIG_{Nat}, $|\mathsf{N}| = |\mathsf{N}|_{NAT}$ is the set $\mathbb{N}$ of natural numbers (including zero), 0^{N} is the number zero, $SUCC^{\mathsf{N}}$ is the successor function on natural numbers, and $+^{\mathsf{N}}$ and $*^{\mathsf{N}}$ are addition and multiplication, respectively. $Nat = (\mathcal{L}_{\mathrm{FOL}}(SIG_{Nat}), \mathcal{N})$ is an $\{\mathsf{N}\}$-theory if we let $\mathcal{N}$ contain the following axioms:

$$SUCC(x) \neq 0,$$
$$SUCC(x) = SUCC(y) \rightarrow x = y,$$
$$x + 0 = x,$$
$$x + SUCC(y) = SUCC(x + y),$$
$$x * 0 = 0,$$
$$x * SUCC(y) = (x * y) + x,$$
$$(A_x(0) \wedge \forall x(A \rightarrow A_x(SUCC(x)))) \rightarrow \forall x A.$$

The notion of $\mathcal{C}$-theories is not very sharp. If $Th = (\mathcal{L}_{\mathrm{FOL}}, \mathcal{A})$ is a $\mathcal{C}$-theory then, by definition, so is every $(\mathcal{L}_{\mathrm{FOL}}, \mathcal{A}')$ where $\mathcal{A}' \subseteq \mathcal{A}$. Hence, in general, a $\mathcal{C}$-theory does not really "characterize" the class $\mathcal{C}$ of structures. The reason is that in the definition we only required that all structures of $\mathcal{C}$ satisfy the non-logical axioms, but not that these structures be the only models of the theory.

The theories $Group$ and $LinOrd$ actually satisfy this stronger requirement: somewhat roughly speaking, a structure is a model of $Group$ or $LinOrd$ if and only if it is a group or a linearly ordered set, respectively. The theories characterize these mathematical systems and we may call them *theory of groups* and *theory of linear orders* (instead of "$\mathcal{C}$-theories").

In the case of Nat, however, the situation is fundamentally different. The structure N cannot be characterized in this way: every first-order theory which has N as a model has also other (even "essentially" different) models. This is a consequence of the famous *(First) Gödel Incompleteness Theorem* which (particularly) says that N cannot be completely axiomatized in first-order logic. More precisely: for any first-order $\{\mathsf{N}\}$-theory whose non-logical axiom set $\mathcal{A}$ is decidable there are formulas F of SIG_{Nat} such that $\models_{\mathsf{N}} F$ but F is not derivable from $\mathcal{A}$ in Σ_{FOL}.

In the presence of different models for natural number theories, N is usually called the *standard model*. This model, together with its underlying signature SIG_{Nat}, will frequently occur in subsequent sections. If necessary, we will feel free to assume (without explicitly mentioning) that the signature may also be enriched by more symbols than shown above (e.g., symbols for other individual constants like $1, 2, \ldots$, for subtraction, division, order relations, etc.) together with their standard interpretations in N. Furthermore, we will overload notation by denoting the interpretations of syntactic symbols by these same symbols (e.g., $+^{\mathsf{N}}, *^{\mathsf{N}}, 1^{\mathsf{N}}, 2^{\mathsf{N}}, \ldots$ will be denoted by $+, *, 1, 2, \ldots$).

Besides formalizing mathematical systems such as groups and linear orders, the concept of logical theories also finds applications in computer science. For example, the theory Nat (perhaps presented with some extra "syntactic sugar") would typically be called an *algebraic specification* of the natural numbers. In general, an algebraic specification of an *abstract data type* (in a *functional* setting) is just the same as a theory.

A further example is given by the signature

$$SIG_{st} = (\{OBJ, STACK\}, \mathbf{F}, \emptyset)$$

with

$$\mathbf{F} = \{EMPTY^{(\varepsilon, STACK)}, PUSH^{(STACK\ OBJ, STACK)}, \\ POP^{(STACK, STACK)}, TOP^{(STACK, OBJ)}\}$$

and the theory (or algebraic specification)

$$Stack = (\mathcal{L}_{\mathrm{FOL}}(SIG_{st}), \mathcal{S})$$

with $\mathcal{S}$ consisting of the axioms

$$PUSH(x, y) \neq EMPTY,$$
$$POP(PUSH(x, y)) = x,$$
$$TOP(PUSH(x, y)) = y$$

(where $x \in \mathcal{X}_{STACK}, y \in \mathcal{X}_{OBJ}$). Clearly, $Stack$ is a theory of stacks: the domain $|\mathsf{S}|_{STACK}$ of its (standard) models consists of stacks of objects from $|\mathsf{S}|_{OBJ}$ and $EMPTY^{\mathsf{S}}$, $PUSH^{\mathsf{S}}$, POP^{S}, and TOP^{S} are functions implementing the usual stack operations.

Note that the semantical definitions in the previous section obviously assume that the mappings which interpret function and predicate symbols are total since

otherwise the evaluation $\mathsf{S}^{(\xi)}$ would not always be defined. In this example, on the other hand, pop and top are usually understood to be partial (not defined on the empty stack). To solve this technical problem in a trivial way, we assume that pop and top deliver some arbitrary values when applied to the empty stack and, hence, are total. In subsequent similar situations we will always tacitly make corresponding assumptions.

In some computer science texts, specifications like Nat or $Stack$ additionally contain (or "use") an explicit specification of a data type $Boolean$ containing the boolean values and the usual boolean operators. Because $Boolean$ is implicitly contained in the propositional fragment of first-order logic, it does not have to be an explicit part of a first-order theory.

We have introduced the concept of theories in the framework of classical first-order logic. Of course, it can be defined in the same way for any other logic as well. For example, a theory could also be based on propositional logic. Such *propositional theories* are of minor interest in mathematics. In computer science, however, this changes, particularly because of the decidability of PL. A typical situation arises by first-order theories of structures with finite domains. These can be encoded as propositional theories (essentially by expressing a quantification $\exists x A$ by a disjunction of all instantiations of A with the finitely many possible values $\xi(x)$ of x) and can then be accessible to appropriate algorithmic treatments. We will investigate this aspect in the context of temporal logic in Chap. 11 and content ourselves here with a toy example of the kind which typically serves as a measure for automatic proof systems. Consider the following criminal story:

> Lady Agatha was found dead in her home where she lived together with her butler and with uncle Charles. After some investigations of the detective, the following facts are assured:
>
> 1. Agatha was killed by one of the inhabitants.
> 2. Nobody kills somebody without hating him or her.
> 3. The perpetrator is never richer than the victim.
> 4. Charles hates nobody whom Agatha was hating.
> 5. Agatha hated all inhabitants except perhaps the butler.
> 6. The butler hates everybody not richer than Agatha or hated by Agatha.
> 7. No inhabitant hates (or hated) all inhabitants.
>
> Who killed Agatha?

In order to fix a language of propositional logic we have to determine the set $\mathbf{V}$ of propositional constants. For our story we represent the persons Agatha, the butler, and Charles by a, b, and c, respectively, and let $\mathbb{P} = \{a, b, c\}$ and

$$\mathbf{V}_{murder} = \{kill_{ij}, hate_{ij}, richer_{ij} \mid i, j \in \mathbb{P}\}.$$

The elements of $\mathbf{V}_{murder}$ represent the propositions "i killed j", "i hates (or hated) j", and "i is (was) richer than j" for $i, j \in \mathbb{P}$, respectively. With this in mind we get a propositional theory $(\mathcal{L}_{\mathrm{PL}}(\mathbf{V}_{murder}), \mathcal{M})$ by collecting in $\mathcal{M}$ the following formulas which formally express the above facts:

1. $kill_{aa} \vee kill_{ba} \vee kill_{ca}$,
2. $kill_{ij} \rightarrow hate_{ij}$ for all $i, j \in \mathbb{P}$,
3. $kill_{ij} \rightarrow \neg richer_{ij}$ for all $i, j \in \mathbb{P}$,
4. $hate_{aj} \rightarrow \neg hate_{cj}$ for all $j \in \mathbb{P}$,
5. $hate_{aa} \wedge hate_{ac}$,
6. $(\neg richer_{ja} \vee hate_{aj}) \rightarrow hate_{bj}$ for all $j \in \mathbb{P}$,
7. $\neg hate_{ia} \vee \neg hate_{ib} \vee \neg hate_{ic}$ for all $i \in \mathbb{P}$.

The case can be solved semantically by showing that for any model M of the theory, which in this propositional situation is just a valuation $\mathsf{M} : \mathbf{V}_{murder} \rightarrow \{\mathsf{ff}, \mathsf{tt}\}$, $\mathsf{M}(kill_{aa}) = \mathsf{tt}$ must hold whereas $\mathsf{M}(kill_{ba}) = \mathsf{M}(kill_{ca}) = \mathsf{ff}$, or proof-theoretically by showing that

$$\mathcal{M} \models_{\Sigma_{\mathrm{PL}}} kill_{aa} \wedge \neg kill_{ba} \wedge \neg kill_{ca} \,.$$

Anyway, the conclusion is that Agatha committed suicide. One should also convince oneself by exhibiting a model M for $\mathcal{M}$ that the assumed facts are not contradictory: otherwise, the conclusion would hold trivially.

1.4 Extensions of Logics

The logic FOL extends PL in the sense that every formula of (any language $\mathcal{L}_{\mathrm{PL}}(\mathbf{V})$ of) PL is also a formula of (some language containing the elements of $\mathbf{V}$ as propositional constants of) FOL. Furthermore, PL is a *sublogic* of FOL: all consequence relationships and, hence, universal validities in PL hold in FOL as well. The logics to be defined in the next chapters will be extensions of PL or even FOL in the same way.

Staying within the classical logic framework, we still want to mention another extension of FOL which allows for addressing the non-characterizability of the standard model of natural numbers in FOL as pointed out in the previous section. The reason for this deficiency is that FOL is too weak to formalize the fundamental *Peano Postulate* of natural induction which states that for every set $\mathbb{M}$ of natural numbers,

> if $0 \in \mathbb{M}$ and if for every $n \in \mathbb{N}$, $n + 1 \in \mathbb{M}$ can be concluded from the assumption that $n \in \mathbb{M}$, then $\mathbb{M} = \mathbb{N}$.

This is only incompletely covered by the *induction axiom*

$$(A_x(0) \wedge \forall x(A \rightarrow A_x(SUCC(x)))) \rightarrow \forall x A$$

of the theory Nat. In our assumed framework of countable languages (cf. Sect. 1.1) this fact is evident since A (which "represents" a set $\mathbb{M}$) then ranges only over denumerably many formulas whereas the number of sets of natural numbers is uncountable. But even in general, the Peano Postulate cannot be completely described in FOL.

An extension of FOL in which the Peano Postulate can be described adequately is *(classical) second-order logic* SOL. Given a signature $SIG = (\mathbf{S}, \mathbf{F}, \mathbf{P})$, a *second-order language* $\mathcal{L}_{\mathrm{SOL}}(SIG)$ (again shortly: $\mathcal{L}_{\mathrm{SOL}}$) is defined like a first-order language with the following additions: the alphabet is enriched by

- denumerably many *predicate variables* for every $\vec{s} \in \mathbf{S}^*$.

Let the set of predicate variables for $\vec{s} \in \mathbf{S}^*$ be denoted by $\mathcal{R}_{\vec{s}}$ and $\mathcal{R} = \bigcup_{\vec{s} \in \mathbf{S}^*} \mathcal{R}_{\vec{s}}$. These new symbols allow for building additional atomic formulas of the form

- $r(t_1, \ldots, t_n)$, where $r \in \mathcal{R}_{s_1 \ldots s_n}$ is a predicate variable and t_i are terms of sorts s_i for $1 \leq i \leq n$,

and the inductive definition of formulas in FOL is extended by the clause

- If A is a formula and r is a predicate variable then $\exists r A$ is a formula.

$\forall r A$ abbreviates $\neg \exists r \neg A$.

The semantics of a language $\mathcal{L}_{\mathrm{SOL}}(SIG)$ is again based on the concept of a structure S for SIG. Variable valuations are redefined to assign for all $s \in \mathbf{S}$ and $s_1 \ldots s_n \in \mathbf{S}^*$

- some $\xi(x) \in |\mathsf{S}|_s$ to every individual variable $x \in \mathcal{X}_s$ (as in FOL),
- some mapping $\xi(r) : |\mathsf{S}|_{s_1} \times \ldots \times |\mathsf{S}|_{s_n} \to \{\mathsf{ff}, \mathsf{tt}\}$ to every predicate variable $r \in \mathcal{R}_{s_1 \ldots s_n}$.

The definition of $\mathsf{S}^{(\xi)}(A)$ is extended to the new kind of formulas by

- $\mathsf{S}^{(\xi)}(r(t_1, \ldots, t_n)) = \xi(r)(\mathsf{S}^{(\xi)}(t_1), \ldots, \mathsf{S}^{(\xi)}(t_n))$,
- $\mathsf{S}^{(\xi)}(\exists r A) = \mathsf{tt} \;\Leftrightarrow\;$ there is a ξ' such that $\xi \sim_r \xi'$ and $\mathsf{S}^{(\xi')}(A) = \mathsf{tt}$

where $r \in \mathcal{R}$ and $\xi \sim_r \xi' \;\Leftrightarrow\; \xi(\bar{r}) = \xi'(\bar{r})$ for all $\bar{r} \in \mathcal{R}$ other than r. Finally the notions of validity and consequence from FOL are transferred verbatim to SOL.

A *second-order theory* $(\mathcal{L}_{\mathrm{SOL}}, \mathcal{A})$ consists of a second-order language $\mathcal{L}_{\mathrm{SOL}}$ and a set $\mathcal{A}$ of non-logical axioms (formulas of $\mathcal{L}_{\mathrm{SOL}}$). Models of such theories are defined as in FOL. Any first-order theory can be viewed as a second-order theory as well. E.g., the theory $LinOrd$ of the previous section can be made into a *second-order theory of linear orders* just by replacing $\mathcal{L}_{\mathrm{FOL}}(SIG_{lo})$ by $\mathcal{L}_{\mathrm{SOL}}(SIG_{lo})$.

A proper second-order theory for the natural numbers takes the signature SIG_{Nat} and the first six axioms of the first-order theory Nat together with the new induction axiom

$$\forall r(r(0) \wedge \forall x(r(x) \to r(SUCC(x))) \to \forall x\, r(x))$$

where $r \in \mathcal{R}_{NAT}$ is a predicate variable. The standard model N is a model of this theory and in fact it is the only one (up to "isomorphism"; this relation will be made more precise in Sect. 5.3). So this theory really characterizes the natural numbers.

But the background of Gödel's Incompleteness Theorem is some inherent incompleteness of *Peano arithmetic* and this now becomes manifest at another place: in contrast to FOL, SOL cannot be completely axiomatized, i.e., there is no sound

and complete formal system for SOL, not even in the simple sense that every valid formula should be derivable.

This statement has to be taken with some care, however. If we took all valid formulas of SOL as axioms of a formal system then this would be trivially sound and complete in that sense. But this is, of course, not what is intended by a formal system: to allow for "mechanical" generation of formulas. This intention implicitly supposes that in a formal system the set of axioms is decidable and for any finite sequence $A_1, \ldots, A_n, B$ of formulas it is decidable whether $A_1, \ldots, A_n \vdash B$ is a rule. Since SOL (like FOL) is undecidable, the above trivial approach does not meet this requirement.

To sum up, a logic LOG with a consequence relation $\models$ is called *incomplete* if there is no formal system Σ for LOG in this sense such that

$$\models A \;\Leftrightarrow\; \vdash_{\Sigma} A$$

for every formula A of (any language of) LOG. According to this definition, SOL is incomplete.

We finally note that the above considerations are not restricted to SOL. In fact, the incompleteness result of Gödel may be extended to the following general principle:

Gödel Incompleteness Principle. *Let LOG be a logic with a consequence relation* $\models$ *and* $\mathcal{L}_{\mathrm{LOG}}$ *a language of LOG such that every formula of* $\mathcal{L}_{\mathrm{FOL}}(SIG_{Nat})$ *is a formula of* $\mathcal{L}_{\mathrm{LOG}}$*. If there is a decidable set* $\mathcal{F}$ *of formulas of* $\mathcal{L}_{\mathrm{LOG}}$ *such that*

$$\mathcal{F} \models A \;\Leftrightarrow\; \models_{\mathsf{N}} A$$

holds for every closed formula A *of* $\mathcal{L}_{\mathrm{FOL}}(SIG_{Nat})$ *then* LOG *is incomplete.*

(As before, $SIG_{Nat} = (\{NAT\}, \{0, SUCC, +, *\}, \emptyset)$ and N is the standard model of natural numbers.) This shows the fundamental "trade-off" between logical completeness and characterizability of the natural numbers.

Bibliographical Notes

Logic is a discipline with a long history of more than 2,000 years. *Mathematical* logic as we understand it nowadays – investigating mathematical reasoning and itself grounded in rigorous mathematical methods – began at the end of the nineteenth century with Frege's *Begriffsschrift* [50], the *Principia Mathematica* [158] by Whitehead and Russell, and other pioneering work. For some time, logicians then had the "vision" that it should be possible to "mechanize" mathematics by completely formalizing it within logic. Gödel's work, particularly his famous incompleteness result [57], showed that there are fundamental bounds to this idea.

In the present-day literature, there is a huge number of textbooks on mathematical logic. The selection of contents and the usage of notions and terminology is not uniform in all these texts. In our presentation we mainly refer to the books [43, 60, 105, 137].