Logic, Language, and Information

Reinhard Muskens

July 8, 2016

Contents

Chapter 1

Predicate Logic

Predicate logic is the logician's common workhorse. It is a logical theory that has proved enormously successful in many fields of application, including the formalization of ordinary language and the philosophical analysis that are the concerns of this book. In later chapters we will see that given our purpose it is expedient to enrich this logic with all kinds of modal operators, but it will also become apparent there that these additions can in fact be viewed as *abbreviations* that will not take us out of the classical realm. And while at the end of the book we will argue that it may be a good idea to replace the *first-order* syntax of predicate logic with a *higher-order* one, as this will bring us closer to natural language,[1] the logic that will ensue will in many respects be a generalization of the original, retaining many of its properties.[2]

This means that predicate logic will be a natural point for us to start. But our treatment will not be geared to the completely uninitiated, as there are already many excellent introductions to the topic that start from scratch.[3] It will be assumed that the reader has worked through one of these before he picks up this book. The discussion here will mainly serve the purpose of setting the stage, fixing notation, and introducing some concepts that will be needed in the following chapters. We will define the language of

[1] A first-order syntax only allows for binding of individual variables and for predicating things of individuals (as in *Socrates is wise*), while a higher-order syntax makes it possible to say of a property P that it has some ("higher-order") property $\mathcal{P}$ (as in *Wisdom is desirable*) and to bind variables ranging over properties of various kinds.

[2] This is especially true when the so-called *generalized* interpretation of higher-order logic is adopted. Some of this will be explained in the last chapter.

[3] Here we mention Hodges (2001), Gamut (1991), and Barwise and Etchemendy (2002).

predicate logic, describe the structures which serve as its interpretation, give a reasoning system called *the method of tableaus*, and, lastly, describe a typed variant of the logic that is useful for the applications we are after.

1.1 The Language of Predicate Logic

The formulas of predicate logic are built up from atomic formulas with the help of truth-functional connectives and the universal and existential quantifiers. So in order to say what a formula is we must first say what an atomic formula is and then explain how complex formulas are built up from less complex ones.

Atomic formulas can be used to express that some relation holds between objects. For example, if you want to express that Harvey kissed Sue at noon, you can do that by using the atomic formula $Khst_1$, in which K is a (three place) relation symbol (for *kiss*), and h, s, and t_1 are terms that are meant to refer to Harvey, Sue and noon respectively. A time of day is considered to be an 'object' here and we use that term in a very wide sense. Terms can refer to whatever one thinks exists (and perhaps, as we shall see, even to things that do not in fact exist).

The terms h, s and t_1 occurring in the atomic formula $Khst_1$ are *constants*, intended to refer to some fixed object each, but *variables*, which are also terms, are allowed in the same positions. So, for example, if x and t are variables, $Kxst$ says that some unspecified person x kissed Sue at some unspecified time t. Variables become important in conjunction with the notion of *quantification*, to which we will turn shortly.

Here is a definition of terms and atomic formulas.

DEFINITION 1 A *term* is a constant or a variable.[4] An *atomic formula* is either an n-place relation symbol ($n \geq 0$) followed by n terms or a string of the form $\gamma_1 = \gamma_2$, where γ_1 and γ_2 are terms.

Atomic formulas $R\gamma_1\gamma_2$ that consist of a binary (2-place) relation symbol R and two terms γ_1 and γ_2 may alternatively be written as $\gamma_1 R\gamma_2$. So, for example, if we want to express that time t_1 precedes time t_2, we can do that by writing $t_1 \prec t_2$ instead of the less natural looking $\prec t_1 t_2$.

[4] A more complete treatment would also allow the formation of terms with the help of *function symbols*, so that, for example, $x + 5$ or $5 \times (3 + 8)$ would be considered terms.

Note that in the definition above we have allowed 0-place relation symbols. We usually call them *propositional constants*. Since a 0-place relation symbol followed by 0 terms is an atomic formula according to the definition, these propositional constants will be atomic formulas.

Having defined what atomic formulas are, we can consider more complex formulas that involve the truth-functional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$ and the quantifiers $\forall$ and $\exists$. We assume these operators to be familiar but have described them in Table 1.1 anyway. The following definition allows one to build up complex formulas, using atomic formulas as the smallest building blocks and the logical operators from Table 1.1 as the mortar combining them.

DEFINITION 2 A *formula* is any string that can be built up with the help of the following clauses.

a. Any atomic formula is a formula.

b. If φ is a formula then $\neg\varphi$ is also a formula.

c. If φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$ are also formulas.

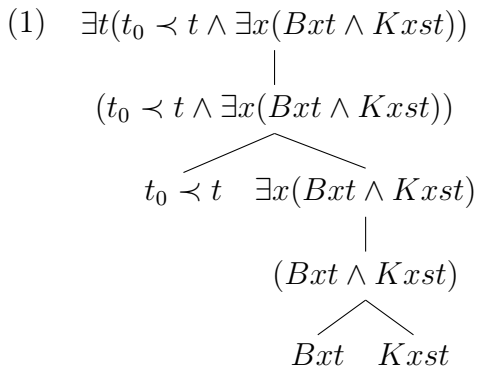
d. If φ is a formula and v is a variable, then $\forall v\varphi$ and $\exists v\varphi$ are also formulas.

For example, $\exists t(t_0 \prec t \wedge \exists x(Bxt \wedge Kxst))$ can now be shown to be a formula, because the rules in the definition can be used to construct it from the atomic formulas $t_0 \prec t$, Bxt, and $Kxst$. Such constructions proceed by successively

formula	(informal) description of meaning
$\neg\varphi$	it is not the case that φ
$\varphi \wedge \psi$	φ and ψ
$\varphi \vee \psi$	φ or ψ (inclusive 'or')
$\varphi \rightarrow \psi$	$\neg\varphi \vee \psi$
$\varphi \leftrightarrow \psi$	$(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
$\forall v\varphi$	for all v it holds that φ
$\exists v\varphi$	for at least one v it holds that φ

Table 1.1: Truth-functional connectives and quantifiers

combining elements into larger and larger wholes. In this case one may begin with Bxt and $Kxst$, conclude from clause a. that these are formulas (because they are atomic formulas) and then use clause c. to get the formula $(Bxt \wedge Kxst)$. This is followed by an application of clause d., from which $\exists x(Bxt \wedge Kxst)$ is obtained, etc. The whole process can be modeled by a construction tree such as the following.



The entire formula $\exists t(t_0 \prec t \wedge \exists x(Bxt \wedge Kxst))$ could be used to express that some boy will kiss Sue, if t_0 is identified with *now*. But this is only one example among many: an infinity of formulas can be constructed with the help of the few clauses in definition 2.

The parentheses introduced in definition 2 serve the useful purpose of disambiguation. For example, $((\varphi \wedge \psi) \vee \chi)$ and $(\varphi \wedge (\psi \vee \chi))$, two formulas that will not in general be equivalent, are kept apart by them. But parentheses are not always necessary and we will omit them when they are not, as this will make things more readable. In particular, we will always omit *outer* parentheses and will write $(\varphi \wedge \psi) \vee \chi$, for example, instead of the official $((\varphi \wedge \psi) \vee \chi)$. Also, in view of the fact that $(\varphi \wedge \psi) \wedge \chi$ and $\varphi \wedge (\psi \wedge \chi)$ are equivalent (or at least will turn out to be according to the definition of equivalence below), we will usually just write $\varphi \wedge \psi \wedge \chi$. Similarly, we will write $\varphi \vee \psi \vee \chi$ for $(\varphi \vee \psi) \vee \chi$, but also for $\varphi \vee (\psi \vee \chi)$. The reader may disambiguate either way whenever disambiguation is necessary.

A last, but not unimportant, point concerns the freedom and bondage of variables. In $\exists t(t_0 \prec t \wedge \exists x(Bxt \wedge Kxst))$ the initial $\exists t$ *binds* all subsequent occurrences of the variable t and $\exists x$ *binds* the two occurrences of x following it. The idea is that whenever clause d. of definition 2 is used to combine a binder $\exists v$ or $\forall v$ with a formula φ to form $\exists v\varphi$ or $\forall v\varphi$, this binder binds all occurrences of v in φ, unless they have already been bound earlier in

the construction process. An occurrence of a variable in a formula that is not bound in this way in is called *free*. For those who like a more precise definition, here is one.

DEFINITION 3 The following clauses define when an occurrence v of a variable in a formula χ is *free* in that formula χ.

a. Any occurrence of a variable in an atomic formula is free in that atomic formula.

b. If v occurs free in φ, it also occurs free in $\neg\varphi$.

c. If v occurs free in φ or in ψ, it also occurs free in $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, or $\varphi \leftrightarrow \psi$.

d. If v occurs free in φ and $v \neq w$, v also occurs free in $\forall w\varphi$ or $\exists w\varphi$.

In $\exists v\varphi$ the initial $\exists v$ *binds* each free occurrence of v in φ and, similarly, in $\forall v\varphi$ the initial $\forall v$ *binds* each free occurrence of v in φ. An occurrence of v in χ is said to be *bound* in χ if it is bound by a binder within χ in this way.

Note that it is necessary to talk about *occurrences* of variables here. All occurrences of y in $\forall y(Ayx \rightarrow By)$ are bound, but in the somewhat similar formula $\forall yAyx \rightarrow By$, the occurrence of y in Ayx is bound while its occurrence in By is free. It does not make sense to ask whether y is free or bound *per se*.

One more definition: If φ is a formula and γ is a term, then $[\gamma/v]\varphi$ will be our notation for the result of replacing every free occurrence of v in φ by γ. This notion of substitution of terms for free variables will play an important role in many pages to follow.

A formula that has no free variables occurring in it is called a *sentence*. Actually, for many purposes it are sentences we are really interested in. Formulas served as a stepping-stone to obtain them.

1.2 Semantics

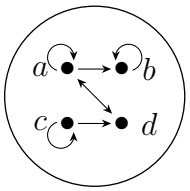
We now turn to the interpretation of predicate logic and to the definition of valid argument. Informally, an argument is *valid* if its conclusion is true in all situations in which its premises are true. This is a rather imprecise

characterisation as long as it is not clear what we mean by a 'situation' and what is meant by a sentence being true in such a situation. We will, therefore, define these notions. The vague notion of a situation will be replaced by the precise notion of a *model* and the notion of the *truth of* a sentence *in* a model can then also be made explicit.[5]

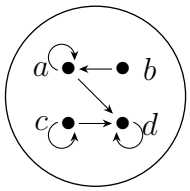
Let us introduce the idea of a model with an example. Suppose that four persons are under discussion, Ann, Ben, Carl, and Daphne, and that we are interested in who loves whom. A vocabulary with constants a, b, c, and d, naming the four, and a binary relation symbol L, for the relation of love, would make it possible to discuss the relevant facts in predicate logic. We can then express things like Lba (Ben loves Ann), $\forall x \exists y Lyx$ (Everyone is loved by someone), and $\forall x \forall y((Lxy \wedge Lyx) \rightarrow x = y)$ (No two different persons love each other).

But of course, whether such statements are true or not depends on the facts. What are the facts? The following pictures illustrate two possibilities among many.

(2) A.



B.



(2A) captures the situation in which Ann, Ben and Carl each love themselves, Ann loves Ben, Carl loves Daphne, Ann and Daphne love each other, and noone loves anyone else. (2B) models another possibility. We see that Lba is true in (2B) but not in (2A). This also holds for $\forall x \forall y((Lxy \wedge Lyx) \rightarrow x = y)$, while $\forall x \exists y Lyx$ is true in (2A) but not in (2B).

Pictures like those in (2) are extremely convenient in many cases, but not in all. For one thing, while in this book we shall mainly encounter *finite* situations, in other contexts *infinite* ones will be the norm, and while it is often possible to use a finite picture to suggest the shape of an infinite one, many times it is not. Another difficulty is the representation of n-place relations where n is larger than two. Representing binary relations with the

[5]The concept of a model derives from work in the 1950s by Alfred Tarski. This work in its turn was based on Tarski's ground-breaking ideas about the concept of truth (Tarski 1935). A translation of this paper, which has played a key role in modern logic, can be found in Tarski (1983). Tarski's explanation in Tarski (1944), which is light on formalization, is highly readable.

help of arrows as in (2A) is simple, but when the number of argument places increases, things become involved.

It is for this reason that models are defined with the help of sets. A model for a sentence will be a pair of two objects. The first of these is a so-called *domain of discourse*, or *domain* for short. This is the set of objects that the sentence talks about and that its quantifiers range over. The domain of discourse of (2A) is the set of four distinct objects drawn there; we may identify it with the set $\{1, 2, 3, 4\}$. The second ingredient that goes into a model for a sentence is an *interpretation* of all the non-logical vocabulary occurring in that sentence. This will be modeled by an *interpretation function* which sends each individual constant occurring in the sentence to an element of the domain and moreover sends each n-place relation symbol to an *n-place relation* on the domain.[6] (2B), for example, will be identified with a model $\langle D, I\rangle$, where $D = \{1, 2, 3, 4\}$ and I is the function given by

a. $I(a) = 1$, $I(b) = 2$, $I(c) = 3$, $I(d) = 4$;

b. $I(L) = \{\langle 1, 1\rangle, \langle 3, 3\rangle, \langle 4, 4\rangle, \langle 2, 1\rangle, \langle 1, 4\rangle, \langle 3, 4\rangle\}$.

The general definition of a model follows this pattern closely.

DEFINITION 4 The *vocabulary* of a formula or a set of formulas will be the set of all individual constants and relation symbols occurring in that formula or those formulas. Let $\mathcal{V}$ be a vocabulary. A *model* for $\mathcal{V}$ is a pair $\langle D, I\rangle$, where D is a non-empty set and I is a function with domain $\mathcal{V}$ such that

a. $I(c) \in D$ for each individual constant $c \in \mathcal{V}$;

b. $I(R) \subseteq D^n$ for each n-place relation symbol $R \in \mathcal{V}$. In other words, to each n-place relation symbol R occurring in the vocabulary, I assigns an n-place relation on D.[7]

This definition gives a very precise notion of a model. But that will not mean that we will give up using pictures. Pictures have great advantages and as long as they can be translated to the more official representation there is no need to give them up.

[6] An n-place relation on a domain D is a set of n-tuples each of whose elements is an element of D. For example, $\{\langle 3, 3, 2\rangle, \langle 1, 3, 1\rangle, \langle 4, 3, 2\rangle\}$ is a 3-place (or 'ternary') relation on $\{1, 2, 3, 4\}$.

[7] D^n is the common notation for the set of all n-tuples with elements from D. $I(R) \subseteq D^n$ thus means that $I(R)$ is *some* set of n-tuples of elements of D.

There is a limiting case that should be discussed here before we proceed. Remember that we have allowed relation symbols to be 0-place. According to our definition the interpretation function I of any model will assign a 0-place relation, i.e. a set of 0-tuples, to such 0-place relation symbols (propositional constants). Now, clearly, there can be only one 0-tuple, namely $\langle\rangle$, the tuple of length 0,[8] and so there are exactly two sets of 0-tuples: the empty set $\varnothing$ and the set $\{\langle\rangle\}$. The first of these may be identified with F, the truth-value *false*, and the second with T, the truth-value *true*. Thus an interpretation function will always assign T or F to any propositional constant.

Let us turn to the general notion of truth. In the preceding pages we have stated informally that some sentences hold and others do not hold in a given model. Since $\forall x \forall y((Lxy \wedge \neg x = y) \rightarrow Lyy)$, for example, means that everyone who is loved by someone else loves himself, it is clear that this statement is true in (2B) but not in (2A). For many purposes such an informal understanding of the truth conditions of predicate logical sentences will do perfectly. But a better understanding of the workings of the logic requires that the notion of truth is pinned down in a more precise way. This we will do in the next definition. In it, we will make a simplifying assumption. Let us say that in a model $M = \langle D, I\rangle$ an element $d \in D$ is an *anonymous object* if there is no individual constant c in the vocabulary of M such that $I(c) = d$, i.e. if d has no name in M. The models we have seen thus far did not contain anonymous objects, and for the moment we will define the notion of truth only with respect to models that do not.

The definition below will be followed by an explanation.

DEFINITION 5 The following clauses characterise the notion φ *is true in* $M = \langle D, I\rangle$, where M is a model that does not contain anonymous objects and φ is a sentence in the vocabulary of M.

a. $Rc_1 \ldots c_n$ is true in $\langle D, I\rangle \iff \langle I(c_1), \ldots, I(c_n)\rangle \in I(R)$;

b. $c_1 = c_2$ is true in $\langle D, I\rangle \iff I(c_1) = I(c_2)$;

c. $\neg\varphi$ is true in $M \iff \varphi$ is not true in M;

d. $\varphi \wedge \psi$ is true in $M \iff$ both φ and ψ are true in M;

e. $\varphi \vee \psi$ is true in $M \iff$ at least one of φ and ψ is true in M;

[8] Mathematically, it may be identified with $\varnothing$.

φ	ψ	$\varphi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

φ	ψ	$\varphi \vee \psi$
T	T	T
T	F	T
F	T	T
F	F	F

φ	$\neg\varphi$
T	F
F	T

φ	ψ	$\varphi \rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

φ	ψ	$\varphi \leftrightarrow \psi$
T	T	T
T	F	F
F	T	F
F	F	T

Table 1.2: Truth tables for the truth-functional connectives.

f. $\varphi \rightarrow \psi$ is true in $M \iff$ either φ is not true in M, or ψ is true in M, or both;

g. $\varphi \leftrightarrow \psi$ is true in $M \iff \varphi$ and ψ are both true in M, or neither φ nor ψ is true in M;

h. $\forall v\varphi$ is true in $M \iff [c/v]\varphi$ is true in M for each individual constant c in the vocabulary of M;

i. $\exists v\varphi$ is true in $M \iff [c/v]\varphi$ is true in M for some individual constant c in the vocabulary of M.

We will also express the fact that a sentence φ is true in a model M by using the stylistic variants *M satisfies φ*, *M verifies φ*, *M makes φ true*, and *M is a model of φ*. If φ is not true in M, φ is said to be *false* in M. This can also be expressed as *M falsifies φ* or *M makes φ false*.

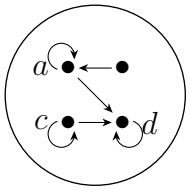
Let us look at the clauses of definition 5. Clause a. tells us that an atomic sentence $Rc_1 \dots c_n$ is true in a model if and only if the n-tuple consisting of the interpretations of the constants in it is an element of the interpretation of R. So, for example, *Ldb* is false in in (2B), or rather in the formalization of (2B) given above, because $\langle I(d), I(b)\rangle = \langle 4, 2\rangle \notin I(L)$.

Clause b. stipulates that $c_1 = c_2$ is true if and only if the interpretations of c_1 and c_2 are indeed equal, a definition that seems reasonable, and clauses c.–g. give truth conditions to the truth-functional connectives that are in accordance with the usual truth tables, given here in Table 1.2. Clauses

h. and i. tell us that $\forall v\varphi$ is true just if $[c/v]\varphi$ is true for every individual constant c and that $\exists v\varphi$ is true if $[c/v]\varphi$ is true for some c. So, for example, $\forall x\exists yLyx$ will be true in (2B) according to clause h., just if $\exists yLya$, $\exists yLyb$, $\exists yLyc$, and $\exists yLyd$ are all true in (2B) and these statements in their turn can be evaluated with the help of clause i. For example, $\exists yLyb$ will be true if and only if one of Lab, Lbb, Lcb, and Ldb is. In fact, clause a. gives that these are all false, so $\exists yLyb$ is false in (2B), and so $\forall x\exists yLyx$ is also false.

Note how the last procedure depended on our requirement that all objects are named by constants. Consider a model that only differs from (2B) in that one element has no name, as in the following picture.

(3)



In official notation this is the model with vocabulary $\{a, c, d, L\}$, with domain $\{1, 2, 3, 4\}$, and an interpretation function I given by

$$I(L) = \{\langle 1,1\rangle, \langle 3,3\rangle, \langle 4,4\rangle, \langle 2,1\rangle, \langle 1,4\rangle, \langle 3,4\rangle\} \ ,$$

$I(a) = 1$, $I(c) = 3$, and $I(d) = 4$. It is clear that the sentence $\forall xLxx$ should come out false in this model, since the anonymous element 2 is not in the relation L to itself (i.e. $\langle 2,2\rangle \notin I(L)$). Clause h. of definition 5 would assign truth to the sentence, however, since Laa, Lcc, and Ldd all hold.

So what should we do with models containing anonymous elements? Forgo them altogether? That would not be a very natural option. We can say that every sparrow in the garden is hungry while not each of them may bear a name. It is a better idea to say that a sentence φ is true in a model M if M can be expanded to a model M' in which all objects have a name and φ is true in M' according to definition 5. In this view we do allow models to have anonymous elements, but in order to check whether a sentence is true in such a model or not, we temporarily name them.[9] This move will have as a consequence that $\forall xLxx$ comes out true in (3). Its element 2 can

[9]This may require extending the vocabulary considerably if the model is large. If, for example, the model contains the real numbers (a kind of model we would surely not want to disallow on logical grounds) we would need to extend the vocabulary with as many names as there are reals. There are ways to avoid such large extensions. For example, consider replacing clauses h. and i. in definition 5 with the following.

temporarily be named with the constant b and then definition 5 can be used to check that the sentence is true in the resulting model (2B).

With the help of the notion of truth in a model two other central logical notions can be defined, the notion of *entailment* and that of *satisfiability*.[10] Entailment captures the idea of valid argument with which we began this section: a set of premises entails a conclusion if and only if the conclusion is true in every model in which the premises are true. A set of sentences is satisfiable, on the other hand, if there is a model making them all true. Here is the definition.

DEFINITION 6 Let Γ be a set of sentences and let φ be a sentence, all in some vocabulary $\mathcal{V}$. We say that Γ *entails* φ, and write $\Gamma \models \varphi$, if and only if φ is true in each model for $\mathcal{V}$ in which all $\gamma \in \Gamma$ are true.

A set of sentences Γ in the vocabulary $\mathcal{V}$ will be called *satisfiable* if and only if there is a model for $\mathcal{V}$ in which all $\gamma \in \Gamma$ are true.

We will write $\psi_1, \ldots, \psi_n \models \varphi$ for $\{\psi_1, \ldots, \psi_n\} \models \varphi$ and $\models \varphi$ for $\varnothing \models \varphi$, where $\varnothing$ is the empty set. If $\varphi \models \psi$ and $\psi \models \varphi$, then φ and ψ will be called *equivalent*.

Satisfiability and entailment are closely related: $\Gamma \models \varphi$ holds if and only if $\Gamma \cup \{\neg\varphi\}$ is not satisfiable, as the reader may care to verify. Thus $\Gamma \not\models \varphi$ if and only if $\Gamma \cup \{\neg\varphi\}$ is satisfiable, or, in other words, if and only if there is a model in which all $\gamma \in \Gamma$ are true, but φ is false. Such a model will be called a *counterexample* to $\Gamma \models \varphi$. The set $\Gamma \cup \{\neg\varphi\}$ will be called the *counterexample set* of $\Gamma \models \varphi$.

---

h′. $\forall v\varphi$ is true in $\langle D, I\rangle \iff [c/v]\varphi$ is true in $\langle D, I \cup \{\langle c, d\rangle\}\rangle$ for each $d \in D$, where c is some new individual constant;

i′. $\exists v\varphi$ is true in $\langle D, I\rangle \iff [c/v]\varphi$ is true in $\langle D, I \cup \{\langle c, d\rangle\}\rangle$ for some $d \in D$, where c is some new individual constant.

To see what is happening here, note that $I \cup \{\langle c, d\rangle\}$ is the function I' that is just like I except that it is also defined for the new constant c and that $I'(c) = d$. So the recipe for evaluating $\forall v\varphi$ now is to replace the free occurrences of v in φ by some new constant c and to then evaluate the result with respect to *all* possible values of c. If each of these evaluations leads to truth, the original $\forall v\varphi$ is true, if not, not. The clause for $\exists v\varphi$ is similar. The new set of clauses would also work for models with anonymous elements and the vocabulary is never extended with more than a finite number of new constants.

[10]Some authors (e.g. Hodges (2001)) speak of *consistency* where we use *satisfiability*. We like to reserve this term for a closely related syntactic notion. See the section on tableaus below.

1.3 Tableaus

There are various systems of proof for predicate logic. In some systems deduction proceeds from a given set of axioms with the help of a limited set of inference rules. These are called *Hilbert systems*, named after the great mathematician David Hilbert (1862–1943). Their definition is very simple and they have great value for the abstract game of proving things about the logical provability relation. That game is interesting, but it will not be ours in these notes and, unfortunately, Hilbert systems are not very practical when it comes to *finding* proofs, something that will be of importance to us. There are at least two methods that are much more useful in this respect. The first of these, *natural deduction*, allows one to introduce *assumptions* into a proof, to reason on the basis of these assumptions, and to retract them later, much in the way it is done in standard mathematical reasoning. The second method, the method of *tableaus* (also called *truth trees*), has an interesting dual character. On the one hand tableaus are purely formal objects that can be defined without any reference to the semantic notions that concerned us in the previous section. This makes them count as *proof* systems because a proof should just be symbol manipulation, without reference to the meaning of the symbols involved. On the other hand there is also a semantic interpretation of tableaus, as they can alternatively be viewed as a *systematic search for counterexamples*. The idea here is that, in order to prove that a certain entailment holds, one systematically searches for a model in which its premises are true but its conclusion is false. If, in a sense to be explained below, the search fails, it can be concluded that such models do not exist and that the entailment therefore is valid.

Natural deduction systems and tableau systems have complementary virtues and anyone who is more than superficially interested in logic should really become familiar with both at some point. But here we will opt for tableaus. We will first explain tableaus for the propositional fragment of predicate logic, then for predicate logic without identity, and lastly for full predicate logic with identity.

1.3.1 Rules for the Truth-functional Connectives

We will take the semantic perspective on tableaus first. Suppose we have a (finite) set of sentences containing no quantifiers and that we want to find out whether this set is satisfiable, i.e. whether there is a model that makes

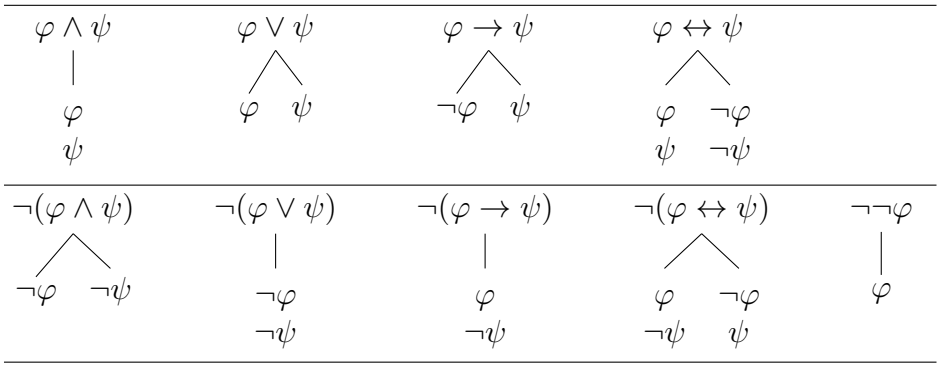


Table 1.3: Tableau expansion rules for truth-functional connectives.

all sentences in the set true. One thing that could be done is to write out a truth table for the sentences in question, but this might be a lot of work if they contain many different atomic formulas. Tableaus often are a more practicable method. In order to illustrate it, let us assume that the set of sentences under consideration is $\{(Pa \wedge \neg Qa) \rightarrow Sb,\ \neg Qa \vee Rab,\ Pa\}$.

We will start our tableau by writing down the elements of this set as in (4).

(4) $(Pa \wedge \neg Qa) \rightarrow Sb$
$\neg Qa \vee Rab$
Pa

This is a tableau, but it is an incomplete one. In order to complete it, the *tableau expansion rules* of Table 1.3 may be used. These rules will allow us to successively break down complex sentences into less complex ones, until we are left with what are called *literals*, atomic formulas and their negations.

Consider $\neg Qa \vee Rab$, one of the sentences in (4). We are after a model in which all sentences in (4) are true. But a model in which $\neg Qa \vee Rab$ is true would be one in which either $\neg Qa$ is true or Rab is. And vice versa: if either $\neg Qa$ or Rab is true (or both), the sentence $\neg Qa \vee Rab$ will also be true. We can therefore split our search in two sub-searches: one for a model in which the sentences $(Pa \wedge \neg Qa) \rightarrow Sb$, $\neg Qa$, and Pa are simultaneously true and one for a model in which $(Pa \wedge \neg Qa) \rightarrow Sb$, Rab, and Pa are. We could (but will not) represent these two new searches as follows.

(5) A.
$$(Pa \wedge \neg Qa) \rightarrow Sb$$
$$\checkmark \neg Qa \vee Rab$$
$$Pa$$
$$\neg Qa$$

B.
$$(Pa \wedge \neg Qa) \rightarrow Sb$$
$$\checkmark \neg Qa \vee Rab$$
$$Pa$$
$$Rab$$

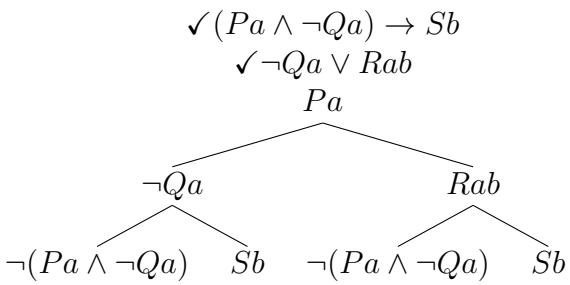
Note that $\neg Qa \vee Rab$ has been marked with a tick in each of the two representations of searches (5A) and (5B). This means that this sentence is no longer under consideration.

(6)
$$(Pa \wedge \neg Qa) \rightarrow Sb$$
$$\checkmark \neg Qa \vee Rab$$
$$Pa$$
$$\neg Qa \quad Rab$$

The representation in (5) is transparent, but rather inefficient, because it requires writing down formulas over and over again. For this reason the slightly different tree representation of (6) is chosen. (5A) and (5B) are still recognizable as *branches* of the tree (a *branch* of a tree is a path from the top of the tree to any of its lowest nodes, without detours), but most material is shared now.

In fact, by going from (4) to (6) we have applied one of the nine rules in Table 1.3, the one where $\varphi \vee \psi$ is split into φ and ψ. In order to make it easy to refer to this rule, we will call it $[\vee]$ and in general we may refer to any of the rules in Table 1.3 by just mentioning the connectives occurring in its top formula, in the order in which they appear. So we will have rules $[\wedge]$, $[\rightarrow]$, $[\neg\vee]$, $[\neg\neg]$, etc. Note that all these rules, read from top to bottom, remove connectives from the formulas involved. The formulas on the lower level always have at least one connective less than the formula on top. The result is a steady decrease in the number of connectives in sub-searches. For example, in the move from (4) to (6), we replaced a problem involving six connectives with two problems involving just four (here the ticked sentence is discounted of course).

(7)



Let us continue with the construction of our tableau. The only sentence that can be dealt with at this point is $(Pa \wedge \neg Qa) \rightarrow Sb$ and the only rule that is applicable to it is $[\rightarrow]$. Since the tableau now consists of two branches and $(Pa \wedge \neg Qa) \rightarrow Sb$ is part of each, we must apply the rule to it in each of these two. The result is (7).

We can continue in this vein until all non-literals have been dealt with. This leads to (8).

(8)

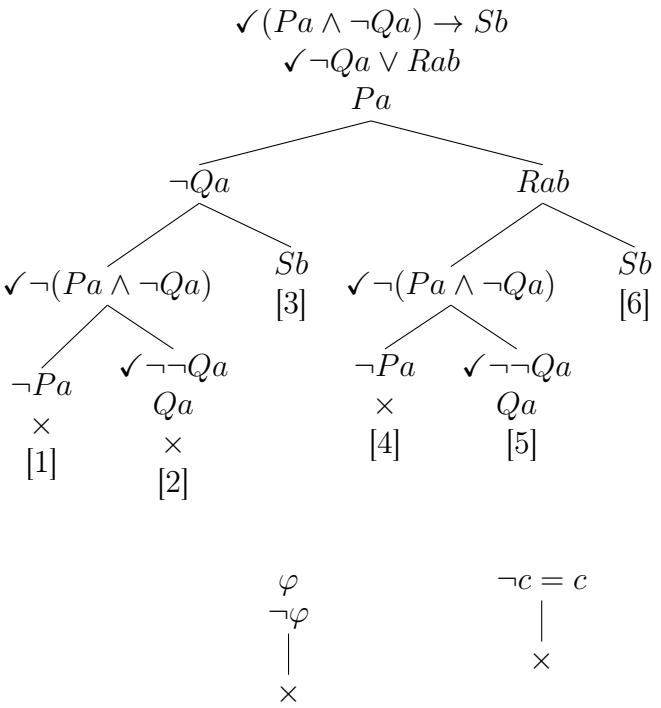


Table 1.4: Tableau closure rules

This tableau has six branches, but three of them contain a contradiction. For example, branch [1] contains Pa and $\neg Pa$ and thus pronounces Pa to be both true and false. Branches [2] and [4] contain similar contradictions. Such *closed* branches are marked with a $\times$. It is clear that the set of sentences on them is not satisfiable. Branches [3], [5] and [6], on the other hand, do not contain any contradiction and are called *open*. Table 1.4 gives the two rules that allow closure of a branch. From an open branch that is *finished*,[11] we can easily read off a model satisfying all formulas on the branch. Here are the models that can be read off from (8).

a. From branch [3]: $\langle D_1, I_1\rangle$ with $D_1 = \{1,2\}$, $I_1(a) = 1$, $I_1(b) = 2$, $I_1(P) = \{1\}$, $I_1(Q) = \varnothing$, $I_1(S) = \{2\}$, and $I_1(R) = \varnothing$.

b. From branch [5]: $\langle D_2, I_2\rangle$ with $D_2 = \{1,2\}$, $I_2(a) = 1$, $I_2(b) = 2$, $I_2(P) = \{1\}$, $I_2(Q) = \{1\}$, $I_2(S) = \varnothing$, and $I_1(R) = \{\langle 1,2\rangle\}$.

c. From branch [6]: $\langle D_3, I_3\rangle$ with $D_3 = \{1,2\}$, $I_3(a) = 1$, $I_3(b) = 2$, $I_3(P) = \{1\}$, $I_3(Q) = \varnothing$, $I_3(S) = \{2\}$, and $I_3(R) = \{\langle 1,2\rangle\}$.

The general recipe for reading off a model $\langle D, I\rangle$ from an open branch is the following. First choose interpretations $I(c)$ for all individual constants c occurring somewhere on the branch, only identifying interpretations $I(c_1)$ and $I(c_2)$ when the branch contains the statement $c_1 = c_2$. The set of interpretations thus obtained will be the domain D, unless there were no individual constants at all, in which case D will be $\{1\}$. For each n-ary relation symbol R occurring on the branch, lastly, let $I(R)$ be the set of all n-tuples $\langle I(c_1), \ldots, I(c_n)\rangle$ such that the atomic formula $Rc_1 \cdots c_n$ occurs on the branch.

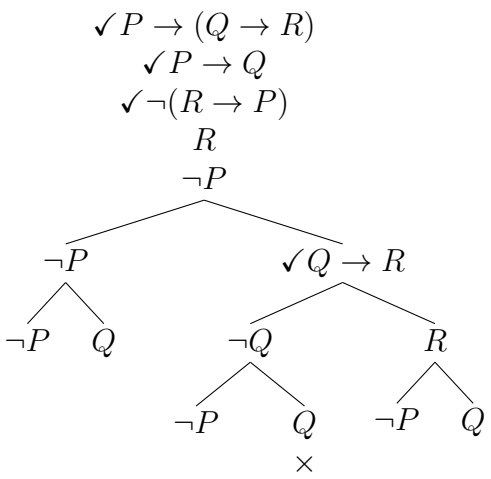
Any model read off from an open branch in this way will not only satisfy the *atomic* sentences on the branch, but in fact all of them. In particular, the sentences on top with which the tableau was begun will be satisfied by such a model.

EXERCISE 1 Check that all three models just described make each sentence in $\{(Pa \wedge \neg Qa) \rightarrow Sb,\ \neg Qa \vee Rab,\ Pa\}$ true.

[11] For the moment this will mean that all non-literals are decorated with a tick. The notion will be revised when we discuss tableaus involving quantifiers.

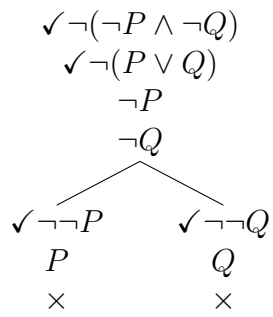
The tableau method is also useful for finding counterexamples to arguments. Remember that an argument $\varphi_1, \ldots, \varphi_n \models \psi$ holds if and only if its counterexample set $\{\varphi_1, \ldots, \varphi_n, \neg\psi\}$ is not satisfiable. A model for the counterexample set will refute the argument. Let us apply this idea to $P \to (Q \to R),\ P \to Q \models R \to P$.[12] Is this sequent correct? (9) shows that a tableau for the counterexample set $\{P \to (Q \to R),\ P \to Q,\ \neg(R \to P)\}$ has several open branches. From the first open branch a model can be read off with $I(P) = I(Q) = \mathrm{F}$ and $I(R) = \mathrm{T}$,[13] which serves as a counterexample to the argument. Hence $P \to (Q \to R), P \to Q \not\models R \to P$.

(9)



What if all branches close, as in the following example, where the correctness of $\neg(\neg P \wedge \neg Q) \models P \vee Q$ is tested?

(10)



[12] Here P, Q and R are propositional constants. Remember that propositional constants are atomic formulas according to the latter's definition.

[13] Note that the domain of the model that we get according to our rule for reading off models from finished open branches will be $\{1\}$, but that it plays no real role in interpreting the sentences involved here.

A tableau all of whose branches are closed is called *closed* itself and the answer to the previous question is that a closed tableau for an argument *proves* that that argument is valid. Let us write $\varphi_1, \ldots, \varphi_n \vdash \psi$ if there is a closed tableau for $\{\varphi_1, \ldots, \varphi_n, \neg\psi\}$. A central theorem of predicate logic (the *completeness* theorem) states that $\vdash$ and $\models$ in fact correspond, in the sense that $\varphi_1, \ldots, \varphi_n \vdash \psi$ if and only if $\varphi_1, \ldots, \varphi_n \models \psi$, for all φ_1, $\ldots$, φ_n, and ψ. In the example above, since (10) shows that $\neg(\neg P \wedge \neg Q) \vdash P \vee Q$, it can in fact be concluded that $\neg(\neg P \wedge \neg Q) \models P \vee Q$. The argument is valid.

We opened this section by saying that we would take a semantic perspective on tableaus first and we introduced the method as a systematic search for models satisfying certain finite sets of sentences. But note that if $\varphi_1, \ldots, \varphi_n \vdash \psi$, that fact can be established without any reference to models at all, since the notion of a closed tableau rests purely on rules such as the ones in Tables 1.3 and 1.4. Finding a closed tableau, or checking whether some given tree of formulas is a closed tableau is pure symbol manipulation. This offers a second, *syntactic*, perspective on the method. The importance of the completeness theorem lies in the fact that it shows that the semantic perspective and the syntactic perspective in fact lead to notions of validity that in fact coincide.

EXERCISE 2 Use tableaus to find out whether the following sets of formulas are consistent. In case they are, give a model in which all formulas are true.

a. $\{P \to Q,\ \neg P,\ Q\}$

b. $\{P \leftrightarrow Q,\ \neg P,\ Q\}$

c. $\{P \wedge Q,\ \neg P \vee \neg Q\}$

d. $\{P \wedge Q,\ \neg P \wedge \neg Q\}$

EXERCISE 3 Use tableaus to check whether the following entailments hold. Give a counterexample whenever an argument is not valid.

a. $\vdash (P \wedge Q) \vee (\neg P \vee \neg Q)$

b. $(P \wedge Q) \vee (P \wedge R) \vdash P \wedge (Q \vee R)$

c. $P \vee Q \vee R,\ Q \to (P \to \neg R),\ P \leftrightarrow R \vdash P \leftrightarrow \neg R$

d. $P \vee Q \vee R,\ Q \to (P \to R),\ P \leftrightarrow R \vdash P \leftrightarrow \neg R$

e. $\vdash P \to (P \to Q)$

f. $\vdash P \to (Q \to P)$

g. $P \to (Q \to R) \vdash (P \to Q) \to (P \to R)$

h. $\vdash ((P \to Q) \to P) \to P$

EXERCISE 4 Check that all tableau rules in Table 1.3 have the property that if the formula on top is true in a model then the formulas on at least one of the branches below it must be true in that model.

EXERCISE 5 Check that in any of the tableau rules in Table 1.3, on each branch, the sentence on top holds if all sentences on the lower level hold. For example, $\neg(\varphi \wedge \psi)$ is true if $\neg\varphi$ is true, but also if $\neg\psi$ is. This is why we can discard sentences after we have applied a rule to them. The sentences on each of the resulting branches still contain the original information.

EXERCISE 6 Explain why the process of building a propositional tableau always halts, i.e. why we always come to a point where the tableau is finished.

1.3.2 Rules for the Quantifiers

It is time to consider tableau rules for the quantifiers. Table 1.5 gives four of them: one for $\forall$, one for $\exists$, and two rules combining $\neg$ with each of these. We will refer to these rules with $[\forall]$, $[\exists]$, $[\neg\forall]$, and $[\neg\exists]$, in the obvious way. The $[\neg\forall]$ and $[\neg\exists]$ rules are easy to explain, as it should obviously be the case that a sentence $\neg\forall\varphi$ is true in a model M if and only if $\exists x\neg\varphi$ is true in M and similar for $\neg\exists\varphi$ and $\forall x\neg\varphi$.

EXERCISE 7 Use definition 5 to show that it is indeed the case that $\neg\forall\varphi$ is true in M if and only if $\exists x\neg\varphi$ is true in M and that $\neg\exists\varphi$ and $\forall x\neg\varphi$ are similarly equivalent.

The two other rules, $[\forall]$ and $[\exists]$, *look* very similar: In each case the result of applying a rule boils down to substituting a constant for a variable. But in fact there are important differences between these two rules. These have to do with *constraints* on the constant that is to be substituted for the variable and *how often* the rule needs to be applied. Let us explain.

Suppose $\forall x\varphi$ occurs on some open branch of a tableau and suppose the constant c also occurs on that branch. It is clear (from definition 5) that a model M satisfying $\forall x\varphi$ should also satisfy $[c/x]\varphi$, and so an extension of the tableau with that formula is warranted. This explains rule [$\forall$]. But while other rules had the property that the formula that was fed to the rule could be ticked and forgotten, this is no longer the case with [$\forall$]. Given the meaning of the universal quantifier it stands to reason that an open branch will not be finished unless the [$\forall$] rule is applied to *each* combination of a constant and a formula of the form $\forall x\varphi$ on that branch. We also require that the [$\forall$] rule is applied at least once to every formula of the form $\forall x\varphi$ on a branch. If there is no constant already on the branch (an 'old' constant) to do that with, a first constant may be introduced for this explicit purpose. To repeat: the [$\forall$] rule can be applied to old constants or to a first constant and in fact it must be applied to *all* old constants and at least once.

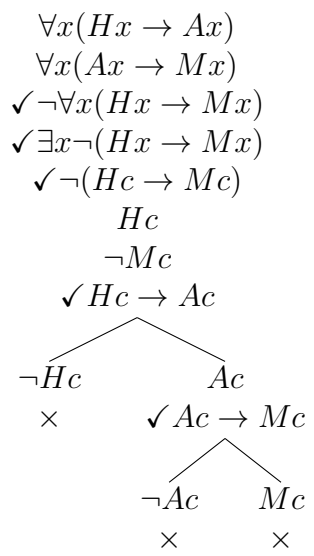
Now let us look at [$\exists$]. If $\exists x\varphi$ is true, then φ must hold of *some* object a, and, introducing a constant c to stand for that a, we find that $[c/x]\varphi$ holds. But c must be *new*. It cannot be assumed that anything referred to by an old constant has the property φ. Overloading a symbol that was already in use may lead to unwarranted conclusions as exercise 9 will make clear. On the other hand, once $[c/x]\varphi$ is added to the branch, the sentence $\exists x\varphi$ can be ticked and forgotten. This explains [$\exists$].

The four quantifier rules just given, plus the rules in Tables 1.3 and 1.4, constitute the set of tableau rules for predicate logic without the identity symbol. Let us apply them to test whether the syllogism $\forall x(Hx \rightarrow Ax),\ \forall x(Ax \rightarrow Mx) \vdash \forall x(Hx \rightarrow Mx)$ (All humans are animals; all animals are mortal; therefore, all humans are mortal) holds.

$\forall x\varphi$	$\exists x\varphi$	$\neg\forall x\varphi$	$\neg\exists x\varphi$
$\vert$	$\vert$	$\vert$	$\vert$
$[c/x]\varphi$	$[c/x]\varphi$	$\exists x\neg\varphi$	$\forall x\neg\varphi$
(c old or first)	(c new)		

Table 1.5: Tableau expansion rules for quantifiers

(11)



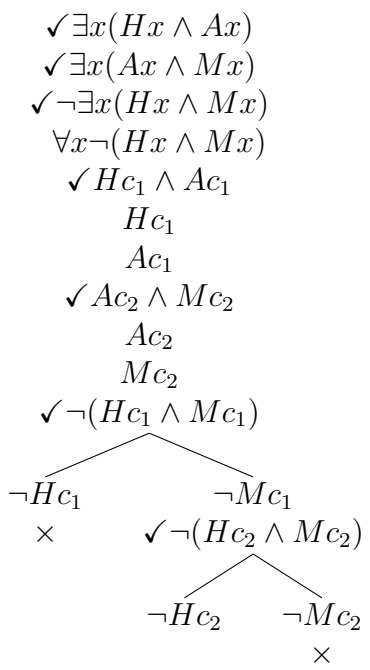
The tableau closes, and we conclude that $\forall x(Hx \to Ax),\ \forall x(Ax \to Mx) \vdash \forall x(Hx \to Mx)$ indeed holds, as (11) is a formal proof of the correctness of the argument. Note that the two sentences starting the tableau are *not* marked with a ✓: universal statements never are.

EXERCISE 8 Make sure you understand how the rules were used in constructing the tableau in (11).

EXERCISE 9 The constant c that occurs in the $[\exists]$ rule is required to be new. In order to see that this must be so, drop the requirement temporarily, and then "prove" that $\exists xAx$ entails $\forall xAx$.

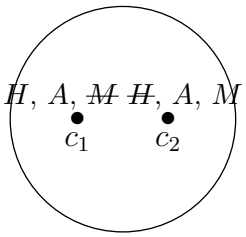
Let us now test a syllogistic argument that is *not* valid: Some human is an animal. Some animal is mortal. Therefore, some human is mortal. In other words, let us test whether the sequent $\exists x(Hx \wedge Ax),\ \exists x(Ax \wedge Mx) \vdash \exists x(Hx \wedge Mx)$ is correct.

(12)



The tableau in (12) has been expanded up to a point where it isn impossible to go further and where the [∀] rule has been applied *twice* to the sentence $\forall x\neg(Hx\wedge Mx)$, once for each of the constants c_1 and c_2. One branch remains open, and in fact we have found a countermodel. The atomic statements on the open branch are $\{Hc_1, Ac_1, Ac_2, Mc_2\}$. This corresponds to the model in (13), in which the arguments' premises are true but its conclusion is false.

(13)



With the [∀] rule around, which may be applied over and over again to the same formula, some care must be taken in defining exactly when an open branch is finished. This is done in the next definition.

Definition 7 An open branch in a tableau is called *finished* if all of the following conditions are fulfilled.

a. Each sentence on the branch that is not a literal and is not of the form $\forall x\varphi$ has received a tick.

b. To each sentence on the branch of the form $\forall x\varphi$ the [$\forall$] rule has been applied at least once (to a freshly introduced 'first' constant, if necessary).

c. For each combination of a sentence of the form $\forall x\varphi$ on the branch and a constant c occurring on the branch (i.e. an 'old' c), one of the following two conditions is the case.

- The [$\forall$] rule has been applied to $\forall x\varphi$ and c, or
- there is a constant c' such that $c = c'$ is on the branch and the [$\forall$] rule has been applied to $\forall x\varphi$ and c'.

The second condition under c. in this definition need not concern us much at the moment. But it will become important shortly, when identity is considered.

Using tableaux for testing the validity of syllogisms is somewhat heavy artillery, as there are simpler ways to do this. But the method becomes really useful when relational information is present and when quantifiers are iterated. The following is a tableau proving that $\exists x\forall yRxy \vdash \forall y\exists xRxy$.

(14)
$$\begin{array}{c} \checkmark\exists x\forall yRxy \\ \checkmark\neg\forall y\exists xRxy \\ \forall yRc_1y \\ \checkmark\exists y\neg\exists xRxy \\ \checkmark\neg\exists xRxc_2 \\ \forall x\neg Rxc_2 \\ Rc_1c_2 \\ \neg Rc_1c_2 \\ \times \end{array}$$

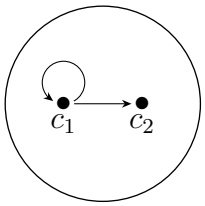
EXERCISE 10 Give an informal argument to the effect that $\forall y\exists xRxy$ follows from $\exists x\forall yRxy$ and compare it with the formal proof in (14).

The argument $\exists x\forall yRxy \vdash \forall y\exists xRyx$ differs only in detail from the one tested in (14). But it is not valid, as (15) shows.

(15)
$$\begin{array}{c} \checkmark \exists x \forall y Rxy \\ \checkmark \neg \forall y \exists x Ryx \\ \forall y Rc_1 y \\ Rc_1 c_1 \\ \checkmark \exists y \neg \exists x Ryx \\ \checkmark \neg \exists x Rc_2 x \\ \forall x \neg Rc_2 x \\ \neg Rc_2 c_1 \\ \neg Rc_2 c_2 \\ Rc_1 c_2 \end{array}$$

The tableau consists of one finished open branch and therefore leads to a counterexample, depicted in (16).

(16)



EXERCISE 11 Check that $\exists x \forall y Rxy$ is true in (16), but that $\forall y \exists x Ryx$ is false in this model.

The next tableau is an attempt at proving $\forall x \exists y Rxy \vdash \exists x Rxx$. It illustrates that the rules for tableau construction may lead to a *loop*. Virtually the same computation is repeated over and over again. At each cycle in the loop a new constant is created, which then, together with the premise $\forall x \exists y Rxy$ 'feeds' the [$\forall$] rule. The single branch does not close, but does not finish after any finite number of steps either.

(17)
$$\begin{array}{c} \forall x\exists y Rxy \\ \checkmark\neg\exists x Rxx \\ \forall x\neg Rxx \\ \neg Rc_1c_1 \\ \checkmark\exists y Rc_1y \\ Rc_1c_2 \\ \neg Rc_2c_2 \\ \checkmark\exists y Rc_2y \\ Rc_2c_3 \\ \neg Rc_3c_3 \\ \vdots \end{array}$$

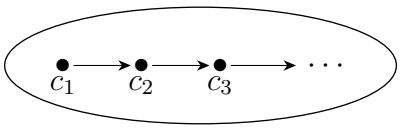
Since the tableau in (17) does not give us a *finite* finished open branch, we consider the tableau that results after an *infinite* number of steps. Then we get a branch that *is* finished in the sense that: 1) the [$\forall$] rule has been applied to $\forall x\exists y Rxy$ an infinite number of times, so that whenever a constant c_i is on the branch, $\exists y Rc_iy$ is too; 2) the [$\forall$] rule has been applied to $\forall x\neg Rxx$ and each constant c_i on the branch in a similar way; and 3) all non-literals other than $\forall x\exists y Rxy$ and $\forall x\neg Rxx$ are ticked.

The set of atomic sentences we get from this infinite branch is

$$\{Rc_1c_2,\ Rc_2c_3,\ Rc_3c_4,\ \ldots\}$$

and the model that we obtain is depicted in (18).

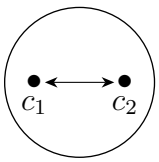
(18)



It can easily be checked that $\forall x\exists y Rxy$ is true in this model, but that $\exists x Rxx$ is false.

In fact there is a much smaller model refuting $\forall x\exists y Rxy \vdash \exists x Rxx$, as (19) shows.

(19)



It may be thought a shortcoming of the tableau rules discussed thus far that a search for a model of $\{\forall x\exists yRxy,\ \neg\exists xRxx\}$ only yields the infinite model in (18) and does not return the simple model in (19), or indeed any other finite model. It will be seen in the next section that this shortcoming can be repaired in the sense that addition of a rule allowing the identification of constants gives a tableau system in which a finite model may be obtained if there is one. But some sets of sentences only have infinite models. For example, if $\forall x\forall y\forall z((Rxy \wedge Ryz) \rightarrow Rxz)$ (transitivity of R) is added to the set of sentences we have been discussing, finite models are excluded, as the following exercise shows.

EXERCISE 12 Give an infinite model for

$$\{\forall x\exists yRxy, \neg\exists xRxx, \forall x\forall y\forall z((Rxy \wedge Ryz) \rightarrow Rxz)\}$$

and explain informally why this set of sentences can have no finite model.

Tableaus for propositional logic always either lead to closure within a finite amount of time, or to a branch that is open and finished and from which a model can be read off.[14] This gives what is called a *decision method* for propositional logic. To know whether a propositional argument is valid it suffices to mechanistically work out a tableau to get an answer in finite time. But for predicate logic the situation is dramatically different. Tableau branches may be infinite and during a search it may be totally unclear whether the tableau will eventually close, or will lead to a finished open branch, finite or infinite. We know from the completeness theorem that *if* an argument is valid it will lead to a closed tableau. But as long as it is unknown whether a given argument is valid or not, it is unknown how a tableau for it will end. There is no algorithm or computer program that, for any predicate logical argument, can give a correct answer to the question whether that argument is valid or not. Tableaus may *help* us finding answers to such questions, but they are not guaranteed to come up with answers in finite time.

1.3.3 Rules for Identity

The tableau rules given thus far are sufficient to formalize the entailment relation of predicate logic without identity. For predicate logic *with* identity

[14] Just a reminder: A finite amount of time may not be a reasonable amount of time. The number of seconds since the Big Bang is finite.

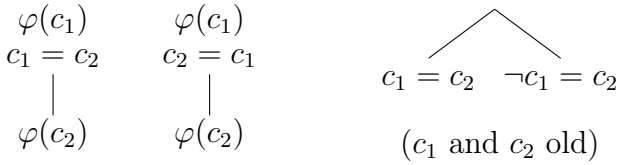


Table 1.6: Tableau expansion rules for identity.

additional rules are needed, such as the ones in Table 1.6, which we will adopt.

The first two of these rules, together sometimes called *Leibniz' Law*, say that if a sentence φ holds, and $c_1 = c_2$ (or $c_2 = c_1$) does too, then replacing one or more occurrences of c_1 in φ by c_2 will lead to a true result.[15] The following closed tableau for $\exists x(x = a \wedge Pxa) \vdash Paa$ illustrates the use of the first rule. It is applied in the last step.

(20)
$$\begin{array}{c} \exists x(x = a \wedge Pxa) \\ \neg Paa \\ b = a \wedge Pba \\ b = a \\ Pba \\ Paa \\ \times \end{array}$$

Exercise 13 Show that the following hold.

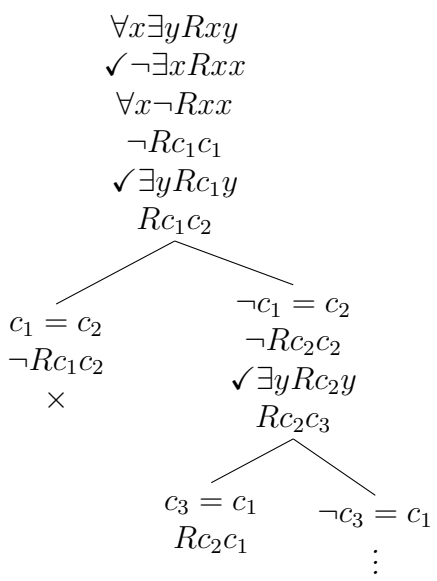
$\vdash \forall x\ x = x$	(= is reflexive)
$\vdash \forall x \forall y(x = y \to y = x)$	(= is symmetric)
$\vdash \forall x \forall y \forall z((x = y \wedge y = z) \to x = z)$	(= is transitive)

Together with the rules in Tables 1.3, 1.4, and 1.5, Leibniz' Law already gives a complete characterization of predicate logic with identity. It can be shown that any argument that is valid in the semantic sense is now provable (i.e. we have $\varphi_1, \ldots, \varphi_n \models \psi$ implies $\varphi_1, \ldots, \varphi_n \vdash \psi$, the 'completeness' part of the completeness theorem). However, we have thrown in an extra rule that, while it is superfluous in the sense that it will not allow us to derive more statements of the form $\varphi_1, \ldots, \varphi_n \vdash \psi$, it is still very useful when it

[15] The notation $\varphi(c)$ is an alternative for $[c/x]\varphi$, where x is any variable that is free in φ. For example, in (20) the rule can be applied to $b = a$ and Pba to get Paa because $Pba = [b/y]Pya$ and $Paa = [a/y]Pya$.

comes to finding counterexamples. This is the third rule in Table 1.6, which states that if two constants c_1 and c_2 already occur on a given branch (i.e. are 'old' to the branch), it is possible to split that branch and consider the cases that $c_1 = c_2$ and that $\neg c_1 = c_2$ separately. In order to illustrate the usefulness of this rule, let us return to our previous attempt to show that $\forall x \exists y Rxy \vdash \exists x Rxx$. This only led to an infinite counterexample in (17), but armed with the new rule we now find a finite finished open branch.

(21)



Here, while an attempted identification of c_1 and c_2 quickly leads to closure, identification of c_3 and c_1 leads to a finite open branch that is finished (compare the conditions in definition 7). The finite model already considered in (19) results. Note that if the expansion of the third branch is continued, not only lots of other finite counterexamples will be found, but one open branch will always remain unfinished. This need not worry us, as we have found our counterexample and have shown that the argument is not valid.

The addition of the third rule in Table 1.6 (let us call it the Identification rule) makes that our tableau system now has the property that if a sequent has a finite counterexample at all, there will be a tableau leading to that finite counterexample. (21) is an illustration of this phenomenon. A proof can be given with the help of the observations in Boolos (1984).

1.3.4 Axioms, Derived Rules, and Sorts

Predicate logic is often used for reasoning in theories given by *axioms*. Suppose, for example, we want to talk about time (as we will want to do in the next chapter). Then we could decide to model the relation of strict precedence between points of time by introducing a relation symbol $\prec$ and by considering axioms like the following. (The axioms in (22) are those of the so-called *dense linear orders without endpoints*.)

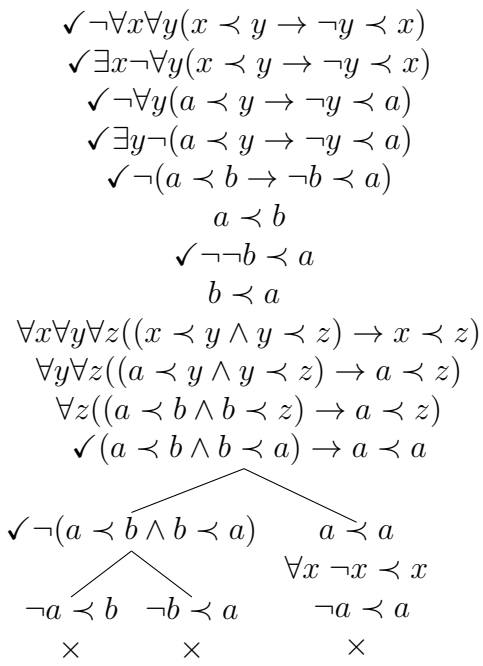
(22)

A1	$\forall x\, \neg x \prec x$	(irreflexivity)
A2	$\forall x \forall y \forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z)$	(transitivity)
A3	$\forall x \forall y(x \prec y \vee y \prec x \vee x = y)$	(connectedness)
A4	$\forall x \forall y(x \prec y \rightarrow \exists z(x \prec z \wedge z \prec y))$	(density)
A5	$\forall x \exists y\, x \prec y$	(no end)
A6	$\forall x \exists y\, y \prec x$	(no beginning)

This set of axioms already embodies various choices about the ontology of time that although not unreasonable are open to discussion. Further axioms could also be considered, or any of the axioms given here could be called into question.

If tableaus are used to obtain consequences of such a set of axioms this could be done by letting each tableau begin with all the axioms from the set, provided there are a finite number of them. But such a procedure is clumsy at best, as even with a finite number of axioms not all of them may be needed to establish what one wants to establish, and it is more practicable to work with a rule that says that any axiom of the theory under consideration can always be introduced at any point. Here is a tableau that shows that precedence is asymmetric, given these axioms, or, in other words, that A1,. . . ,A6 $\vdash \forall x \forall y(x \prec y \rightarrow \neg y \prec x)$.

(23)



Note that only the transitivity and irreflexivity axioms have been used. This is still a rather verbose way of proving asymmetry in the theory of dense linear orders, but at least the four axioms that were not used do not appear in the tableau. In theories with an *infinite* number of axioms introducing axioms by means of a rule becomes a necessity, of course.

There is another way to speed up tableau proofs that is very useful in practice. Looking at (23) one is struck by the amount of work that was necessary to obtain $a \prec a$ from $a \prec b$ and $b \prec a$. A *rule* to the effect that, given axiom A2, it is always possible to derive $a \prec c$ from $a \prec b$ and $b \prec c$, for any a, b, and c would have shortened the proof considerably. That such a rule is derivable is seen from the following tableau.

(24)
$$
\begin{array}{c}
a \prec b \\
b \prec c \\
\forall x \forall y \forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z) \\
\forall y \forall z((a \prec y \wedge y \prec z) \rightarrow a \prec z) \\
\forall z((a \prec b \wedge b \prec z) \rightarrow a \prec z) \\
\checkmark (a \prec b \wedge b \prec c) \rightarrow a \prec c \\
\hline
\checkmark \neg(a \prec b \wedge b \prec c) \quad a \prec c \\
\hline
\neg a \prec b \quad \neg b \prec c \\
\times \qquad \times
\end{array}
$$

This tableau shows that, in the presence of transitivity, the following rule may be used in tableau proofs.

(25)
$$
\begin{array}{l}
a \prec b \\
b \prec c \\
\quad | \\
a \prec c
\end{array}
$$

An application of (25) could always be replaced by (24), so the latter may be considered an abbreviation of the former. Table 1.7 contains more rules that are derivable given certain axioms, but its purpose is mainly illustrative. There are many more axioms that could be considered in certain contexts and more often than not these axioms correspond to rules that can improve the simplicity of tableaus.

EXERCISE 14 Show that all rules in Table 1.7 are derivable, in the same way as it was done for transitivity above. Explain why certain constants must be old and others must be new in these rules.

EXERCISE 15 Using each of the rules in Table 1.7, derive its corresponding axiom. For example, show that a tableau starting with $\neg\forall x\forall y\forall z((x \prec y \wedge y \prec z) \rightarrow x \prec z)$ closes if the transitivity rule may be used.

Rules such as the ones in Table 1.7 only hold in the presence of certain axioms, but derived rules that can be used in any context can also be formulated. For example, the reader may have noticed that applications of $[\neg\forall]$ almost invariably are followed by an application of $[\exists]$ to the resulting sentence and

Irreflexivity:	Transitivity:	Connectedness:
$\neg a \prec a$	$a \prec b$ $b \prec c$ $a \prec c$	$a \prec b \quad b \prec a \quad a = b$
(a old)	(a, b, c old)	(a, b old)
Density:	**No End:**	**No Beginning:**
$a \prec b$ $a \prec c$ $c \prec b$	$a \prec b$	$b \prec a$
(a, b old; c new)	(a old; b new)	(a old; b new)

Table 1.7: Derived rules in the presence of certain axioms.

that a use of $[\neg\exists]$ often precedes one or more uses of $[\forall]$. Tableaus can therefore be sped up by the adoption of the rules in Table 1.8. Note that an application of the second rule should *not* lead to a tick on $\neg\exists x\varphi$. This is now considered a "universal" sentence and the rule can be used over and over again, with the same sentence as input, but different constants.

EXERCISE 16 Redo (23) with all new derived rules in place.

The introduction of axiom rules and derived rules discussed above is one way of making life more pleasant when predicate logic is put to use. Another way is the introduction of all kinds of abbreviations. We will discuss one important kind of abbreviation here that will be used in following chapters over and over again. In many contexts of use it is natural to distinguish

$\neg\forall x\varphi$	$\neg\exists x\varphi$
$[c/x]\neg\varphi$	$[c/x]\neg\varphi$
(c new)	(c old or first)

Table 1.8: Derived tableau expansion rules for negated quantifiers.

between various *sorts* (or *types*) of objects that the variables of the theory range over. For example, a theory may not only talk about time but also about the cabbages and kings kind of objects that exist at certain moments. This could be modeled by introducing predicate constants T (is a time) and E (is an entity) into the language. Statements about all times will then have the form $\forall v(Tv \to \ldots)$ while existential statements about times have the form $\exists v(Tv \wedge \ldots)$. There will also be universal and existential quantifications over entities in such a theory, which will then take the forms $\forall v(Ev \to \ldots)$ and $\exists v(Ev \wedge \ldots)$ respectively. The transitivity axiom for the precedence relation would now, slightly normalized, look as follows.

(26) $\forall v_1 \forall v_2 \forall v_3((Tv_1 \wedge Tv_2 \wedge Tv_3 \wedge v_1 \prec v_2 \wedge v_2 \prec v_3) \to v_1 \prec v_3)$

This proliferation of T's and other restricting predicates in all quantified formulas is rather cumbersome to work with and logicians in such cases therefore move to a *many-sorted* version of predicate logic. In such a logic variables and constants all have *sorts*, which we may identify with predicates like T and E and similar. Now if t is a variable of sort T a quantification $\forall t\varphi$ will be considered to be an abbreviation of $\forall t(Tt \to \varphi)$ and $\exists t\varphi$ will be an abbreviation of $\exists t(Tt \wedge \varphi)$. Similar conventions will hold for other sorts. Transitivity of precedence now reobtains its familiar shape.

(27) $\forall t \forall t' \forall t''((t \prec t' \wedge t' \prec t'') \to t \prec t'')$

It is a usual practice to indicate the sort of a variable with the help of a *typographical convention*. We will, for example, always use t (possibly with primes) as a variable over times. For the possible worlds of the next chapter we will use w, and for entities, we will stick to the use of x, y, and z, which will therefore now get a more restricted interpretation. Constants will also be sorted and allocating a constant a given sort implies that a tacit axiom to that effect is adopted. Suppose, for example, that a constant n (for "now') of sort T is introduced. Then allocating the constant n to T means the adoption an axiom Tn, saying that n is a time.

An important bonus of sorting variables and constants in this way is that applications of the $[\forall]$ rule can be restricted. Whenever a sentence $\forall v\varphi$ appears on a branch of a tableau and v is of a certain sort S, the $[\forall]$ rule need only be applied to combinations of $\forall v\varphi$ and (old) constants of sort S. The $[\exists]$ rule, on the other hand, will only introduce constants of a given type. If $\exists v\varphi$ occurs on a tableau branch, an application of $[\exists]$ to it will create a constant c of the sort of v.

Chapter 2

Time and Modality

2.1 The Logic of Time

Philosophers have always been fascinated with time and change. An early example was Heraclites the Ephesian (ca. 535–475 B.C.), who emphasized that everything is in flux and who famously said that you cannot step into the same river twice. Parmenides of Elea (born around 510 B.C.), on the other hand, thought all change illusory and Zeno, his pupil or lover or both, gave a series of paradoxes that were meant to refute the idea that change is possible at all. We will not review Zeno's paradoxes here, or their modern analysis with the help of the calculus, but will start our considerations with the Scottish idealist philosopher John McTaggart (1866–1925), who in attempting to prove that time is unreal (McTaggart 1908) introduced two competing perspectives on time that have been of great value since. The first of these perspectives, which results in what McTaggart called the *A series* of time, is related to phrases such as *a year ago*, *two days ago*, *now*, *tomorrow*, *ten years from now*, etc. It views time as constantly flowing from the past through the present into the future and gives the present special status in that all times are viewed from its perspective. The A series notion of time therefore squares well (but need not coincide) with *presentism*, the position that only the present is real, while past and future are not.

While the A series concept of time looks at time from within, from the perspective of a continuously moving present, the *B series* notion takes an atemporal perspective and views time from an external angle. It is related to such phrases as *a year earlier than*, *two days before*, *simultaneously with*,

a day later than, etc. In fact we have taken the B series view in the previous chapter, where we modeled the 'earlier than' relation with $\prec$ and discussed possible properties of this relation, such as transitivity, irreflexivity, density, etc. A statement $\mathbf{t_1} \prec \mathbf{t_2}$ is *eternal* in the sense that if it is true now that time $\mathbf{t_1}$ precedes time $\mathbf{t_2}$ this has not only always been the case, but will also always remain so. Contrast this with sentences from ordinary language such as *it rains* whose truth value in past and future do not depend on their truth or falsity now.

With respect to such sentences there are two possible points of view. On one approach they express propositions that are still *incomplete* in that they are lacking an explicit time reference. On this view *it is raining on April 3rd 2008 at 4:41 pm* expresses something that is closer to a proposition than the sentence *it is raining* does, although it is still incomplete since a precise location is lacking. The second point of view is the one that was subscibed to by the Greek philosopher Diodorus Cronus, who lived in the second half of the 4th century BC and who allowed for the possibility that propositions can be true at one time and false at another. Diodorus' view squares well with an A series perspective on time, whereas the view that temporally contingent statements must be anchored to some specific time in order to express a proposition squares well with the B series.

In modern times the Diodorean view that truth-values of propositions may shift over time was taken up by Arthur Prior (1914-1969), a logician who stood at the cradle of tense logic and the semantics of modal logic. Prior observed that a proposition whose truth value shifts over time can be identified with a *property of time points*. For example, the property associated with *it rains* can be predicated of some time point t if it actually rains at t. Such propositions can be combined with the usual logical connectives. If, for example, φ and ψ denote properties of time points, it is natural to let $\varphi \wedge \psi$ denote their intersection; $\varphi \wedge \psi$ will then be true at some time if both φ and ψ are. But the move from truth-values to sets of times as the principal semantic values of sentencess also leaves room for non-Boolean temporal operators, of which Prior (1967) considers the following.

(28) P — it has at some time been the case that
F — it will at some time be the case that
H — it has always been the case that
G — it will always be the case that

The idea here is that if φ expresses some property of times (or 'dates' as

Prior called them), $\mathsf{P}\varphi$ will express the property that holds at a time t if there is a t' earlier than t such that φ holds at t'. The other operators have similar meanings.

With the help of P, F, H, and G, reasonable approximations of the English tenses become possible. If, for example, we agree to formalize *John is kissing Mary* as Kjm, the sentence *John will kiss Mary* can be rendered as $\mathsf{F}Kjm$. The following table gives more translations.

(29)

Kjm	John is kissing Mary
$\mathsf{F}Kjm$	John will kiss Mary
$\mathsf{P}Kjm$	John has kissed Mary
$\mathsf{PP}Kjm$	John had kissed Mary
$\mathsf{FP}Kjm$	John will have kissed Mary
$\mathsf{PF}Kjm$	John would kiss Mary
$\mathsf{PFP}Kjm$	John would have kissed Mary

This approximation of the English tenses is quite rough. The distinction between simple past and present perfect cannot be expressed, for example, so there is room for improvement.

EXERCISE 17 Are the operators P, F, H, and G *truth-functional*? For example, is there a truth function that gives the truth value of $\mathsf{P}\varphi$ in terms of the truth value of φ? Explain your answer.

The temporal statements in (29) did not involve the usual connectives, but with their help sentences such as the following can be formed (the glosses are Prior's own—see also Galton (2008)).

(30) a. $\mathsf{G}P \to \mathsf{F}P$
What will always be, will be

b. $\mathsf{G}(P \to Q) \to (\mathsf{G}P \to \mathsf{G}Q)$
If P will always imply Q, then if P will always be the case, so will Q

c. $\mathsf{F}P \to \mathsf{FF}P$
If it will be the case that P, it will be—in between—that it will be

d. $\neg\mathsf{F}P \to \mathsf{F}\neg\mathsf{F}P$
If it will never be that P then it will be that it will never be that P

Let us give a precise definition of the language these sentences are framed in.

DEFINITION 8 An *atomic sentence of propositional tense logic* is an n-ary relation symbol followed by n individual constants (as in Kjm), while a *sentence of propositional tense logic* is any string that can be built up with the help of the following clauses.

a. Any atomic sentence of propositional tense logic is a sentence of propositional tense logic.

b. If φ is a sentence of propositional tense logic then $\neg\varphi$ is too.

c. If φ and ψ are sentences of propositional tense logic, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$ are too.

d. If φ is a sentence of propositional tense logic, then $\mathsf{P}\varphi$, $\mathsf{F}\varphi$, $\mathsf{H}\varphi$ and $\mathsf{G}\varphi$ are too.

EXERCISE 18 Translate the following into Prior's propositional tense logic.

a. I have loved you and I will always love you.

b. There will come a day when you are happy and have learned to like spinach.

c. John has kissed Mary and Bill has too.

d. John had cleaned all the dishes when Susan rang.

e. If John will kiss Mary then he will from that moment on always admire her.

f. Either there will not be a sea battle or it has always been the case that there will be a sea battle.

A semantics for propositional tense logic can be given by first defining certain temporal structures consisting of time points ordered by a precedence relation and to then define what it means for a sentence φ of the logic to be true at some time t in such a structure. This means that it is possible to consider temporal logic as a logic in its own right with a model theory of its own, more or less along the lines of, but in competition with, the model theory we have given for predicate logic in the previous chapter. Here we will take

another path, leading to the same goal. Following Prior (1967) we will give a translation that, given any combination of a sentence φ of propositional tense logic with a term (variable or constant) t, will return a predicate logical sentence $t : \varphi$ that can be interpreted as 'φ holds at t'. Tense logical sentences will thus get a meaning by translating them to predicate logic (always in combination with a term t), and a notion of tense logical entailment will be obtained from the corresponding predicate logical notion, as will be explained shortly.

The sentences of propositional tense logic are built up according to definition 8 and we will follow the pattern of this definition when translating them.[1] Let us start with the simplest sentences, the atomic ones. These have the form $Rc_1 \dots c_n$, where R is an n-place relation symbol and $c_1, \dots, c_n$ are individual constants (the case that $n = 0$ is allowed and, e.g., *there is a sea battle* could be formalised simply as S). We will assume that each n-place relation symbol R in the vocabulary of the source language (the language of propositional tense logic) is an $n + 1$-place relation symbol R of the target language of predicate logic.[2] Here is the clause in our definition that tells what $t : Rc_1 \dots c_n$ means. (The symbol $\triangleq$ is used for 'equals by definition'. It is not part of the language we are talking about.).

(31) $t : Rc_1 \dots c_n \triangleq Rc_1 \dots c_n t$

So, for example, $t : Kjm \triangleq Kjmt$ and $t : S \triangleq St$. The idea is that $Kjmt$ formalizes *John kisses Mary at time t* and that St says *a sea battle is going on at time t*. Note that, while S expresses a Diodorean proposition, true at some times but not at others, $t : S$ (or St) expresses an eternal proposition, true or false at all times, depending on whether a sea-battle is raging at t or not.

The abbreviation in (31) tells what $t : \varphi$ means when φ is atomic and corresponds to clause a. of definition 8. A next step that needs to be taken (corresponding with clauses b. and c.) is to give translations for the logical connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. Here they are.

[1] The translation will therefore be an example of what mathematicians call an *inductive* definition.

[2] I.e. given a vocabulary $\mathcal{V}$ of the source language, we define the vocabulary of the target language to be just $\mathcal{V}$, but with all n-ary relation symbols now $n+1$-ary relation symbols. An alternative way to set things up would be to have a function $(.)^+$ sending each n-ary relation symbol R of the source language to a $n + 1$-ary relation symbol R^+ of the target language.

(32) a. $t : (\neg\varphi) \triangleq \neg(t : \varphi)$

b. $t : (\varphi \wedge \psi) \triangleq (t : \varphi) \wedge (t : \psi)$

c. $t : (\varphi \vee \psi) \triangleq (t : \varphi) \vee (t : \psi)$

d. $t : (\varphi \rightarrow \psi) \triangleq (t : \varphi) \rightarrow (t : \psi)$

e. $t : (\varphi \leftrightarrow \psi) \triangleq (t : \varphi) \leftrightarrow (t : \psi)$

In all these cases the t just distributes over the relevant connective. Intuitively, this is correct. For example, it is true at t that John kissed and Bill admired Mary just in case when John kissed Mary at t and Bill admired her at t.

The last important step is to translate the new operators. If, at some moment t, I utter the words *John has kissed Mary*, I have spoken truth if there is a moment t_1 preceding t at which *John is kissing Mary* is true. This means that the translation clause for the P operator can run as in (33a). The clauses for the other temporal operators are similar and are given in (33b), (33c), and (33d).

(33) a. $t : \mathsf{P}\varphi \triangleq \exists t'(t' \prec t \wedge t' : \varphi)$

b. $t : \mathsf{F}\varphi \triangleq \exists t'(t \prec t' \wedge t' : \varphi)$

c. $t : \mathsf{H}\varphi \triangleq \forall t'(t' \prec t \rightarrow t' : \varphi)$

d. $t : \mathsf{G}\varphi \triangleq \forall t'(t \prec t' \rightarrow t' : \varphi)$

The clauses in (33) (corresponding with clause d. of definition 8) form the heart of the translation. They explain the P operator by means of an existential quantification over moments preceding the time of evaluation, F as existential quantification over future moments, and H and G as universal quantification over past moments and future moments respectively.

With the help of these abbreviations it now becomes possible to compute the predicate logical translation of any sentence of propositional tense logic (given a term t). Here is an example showing how $t : \mathsf{F}(P \rightarrow \mathsf{G}Q)$ can be rewritten. Each step is annotated with the clause that has been used.

(34)
$$\begin{aligned} t : \mathsf{F}(P \to \mathsf{G}Q) & \\ \exists t_1(t \prec t_1 \wedge t_1 : (P \to \mathsf{G}Q)) & \quad \text{using (33b)} \\ \exists t_1(t \prec t_1 \wedge (t_1 : P \to t_1 : \mathsf{G}Q)) & \quad \text{using (32b)} \\ \exists t_1(t \prec t_1 \wedge (t_1 : P \to \forall t_2(t_1 \prec t_2 \to t_2 : Q))) & \quad \text{using (33d)} \\ \exists t_1(t \prec t_1 \wedge (Pt_1 \to \forall t_2(t_1 \prec t_2 \to t_2 : Q))) & \quad \text{using (31)} \\ \exists t_1(t \prec t_1 \wedge (Pt_1 \to \forall t_2(t_1 \prec t_2 \to Qt_2))) & \quad \text{using (31)} \end{aligned}$$

In the first step rule (33b) was used to rewrite the F operator, which then leaves the task of translating $t_1 : (P \to \mathsf{G}Q)$. The rule for $\to$, (32b), allows rewriting this as $t_1 : P \to t_1 : \mathsf{G}Q$, after which two rewriting tasks remain. One of these concerns $t_1 : \mathsf{G}Q$, which translates as $\forall t_2(t_1 \prec t_2 \to t_2 : Q)$ with (33d). Note that a variable different from t_1 needed to be chosen for the universal quantification here. Two more steps bring everything in predicate logical form.

It is worth noting that at each stage of the process the complexity of the rewriting tasks went down. In general the procedure consists in rewriting simpler and simpler sentences, until the level of atoms is reached, after which one is done.

This translation of the propositional tense logic language into the language of predicate logic provides the former with truth conditions. For example, we now know that $\mathsf{F}(P \to \mathsf{G}Q)$ holds at point t in a model M if and only if $\exists t_1(t \prec t_1 \wedge (Pt_1 \to \forall t_2(t_1 \prec t_2 \to Qt_2)))$ is true in M. The translation also gives a notion of validity for propositional tense logic, since if $\varphi_1, \ldots, \varphi_n$ and ψ are sentences of this logic it is reasonable to stipulate that $\varphi_1, \ldots, \varphi_n$ entails ψ just if, for all models and all time points $\mathbf{t}$ in those models, ψ is true at time $\mathbf{t}$ if $\varphi_1, \ldots, \varphi_n$ are true at time $\mathbf{t}$. This in fact boils down to the statement $\mathbf{t} : \varphi_1, \ldots, \mathbf{t} : \varphi_n \models \mathbf{t} : \psi$ (where $\mathbf{t}$ should not occur in any of $\varphi_1, \ldots, \varphi_n$ and ψ). A special case are sentences φ which hold in all models and at at all time points, so that $\models \mathbf{t} : \varphi$. Here are some sentences that are valid in this way.

(35) a. $\mathsf{P}\varphi \leftrightarrow \neg\mathsf{H}\neg\varphi$

b. $\mathsf{F}\varphi \leftrightarrow \neg\mathsf{G}\neg\varphi$

c. $\mathsf{H}(\varphi \to \psi) \to (\mathsf{H}\varphi \to \mathsf{H}\psi)$

d. $\mathsf{G}(\varphi \to \psi) \to (\mathsf{G}\varphi \to \mathsf{G}\psi)$

e. $\varphi \to \mathsf{HF}\varphi$

f. $\varphi \to \mathsf{GP}\varphi$

A way to check the validity of these statements is to use the translation rules. $t : (\mathsf{P}\varphi \leftrightarrow \neg\mathsf{H}\neg\varphi)$, for example, will reduce to $\exists t_1(t_1 \prec t \wedge t : \varphi) \leftrightarrow \neg\forall t_1(t_1 \prec t \to \neg(t : \varphi))$, which can easily be shown to be valid with the help of a tableau.

EXERCISE 19 Check the validity of $\mathsf{H}(\varphi \to \psi) \to (\mathsf{H}\varphi \to \mathsf{H}\psi)$. First give an annotated computation of $t : (\mathsf{H}(\varphi \to \psi) \to (\mathsf{H}\varphi \to \mathsf{H}\psi))$ until the process can go no further. Then use a tableau to show validity of the result.

EXERCISE 20 Also check the validity of $\varphi \to \mathsf{HF}\varphi$ in this way.

Let us take stock. We have defined the language of propositional tense logic and have given a translation procedure that translates every combination $t : \varphi$, with t a term denoting moments of time and φ a sentence of the new language, into predicate logic. This immediately gives a semantics to the logic and we can check validity of tense logical formulas using methods that were developed for predicate logic. But it will not have escaped the reader that the procedure is somewhat unwieldy. The validity of a simple formula such as $Q \to \mathsf{GP}Q$ needs to be checked via a cumbrous translation into a much longer formula of predicate logic, which can then subsequently be tested with a tableau. It would be much nicer if there were a test procedure working on tense logical formulas directly. Can such a procedure be devised?

The answer is yes, provided we are willing to work with combinations $\mathbf{t} : \varphi$ again. In order to test whether ψ is entailed by $\varphi_1, \ldots, \varphi_n$, we can directly work out a tableau for $\mathbf{t} : \varphi_1, \ldots, \mathbf{t} : \varphi_n \models \mathbf{t} : \psi$, on the basis of the rules in Tables 2.1, 2.2 and 2.3. All sentences occurring in tableau rules in these tables are either of the form $t : \varphi$ or express a direct relation between time points. Table 2.1 gives rules for the propositional connectives strongly resembling similar rules for predicate logic; Table 2.2 lays down rules for the new temporal operators P, F, H, and G; and Table 2.3 gives rules for equality and for closing branches.

Let us consider the propositional rules of Table 2.1. In fact all of them follow from the tableau rules for predicate logic plus our translation of combinations $t : \varphi$. By way of example, let us focus on the rule for $\mathbf{t} : \neg(\varphi \wedge \psi)$. The abbreviation rules in (32) give

$$\mathbf{t} : \neg(\varphi \wedge \psi) \triangleq \neg(\mathbf{t} : (\varphi \wedge \psi)) \triangleq \neg((\mathbf{t} : \varphi) \wedge (\mathbf{t} : \psi)) \ .$$

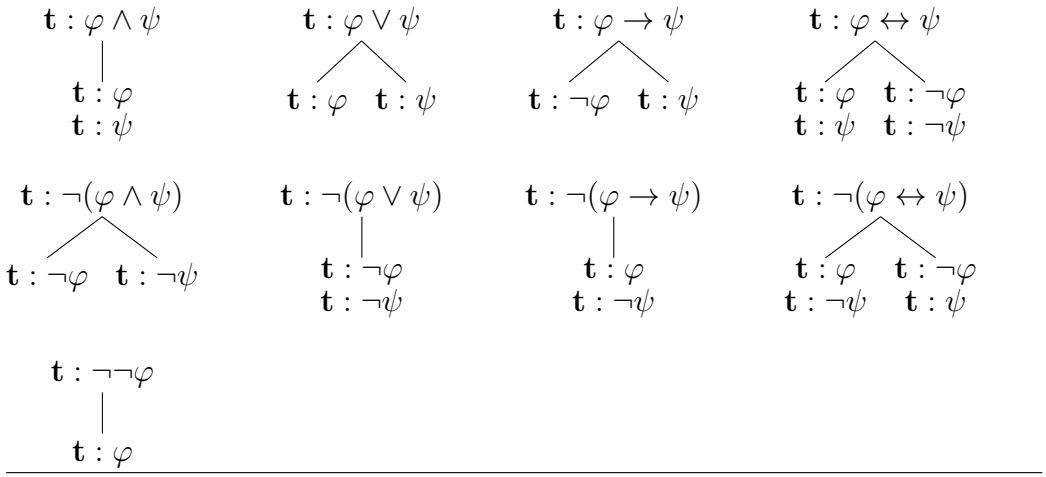


Table 2.1: Tableau expansion for temporal logic: propositional rules.

This means that if a sentence of the form $\mathbf{t} : \neg(\varphi \wedge \psi)$ occurs in a tableau, it may be replaced by $\neg((\mathbf{t} : \varphi) \wedge (\mathbf{t} : \psi))$, since the first sentence is just an abbreviation of the second. But to this latter sentence the $[\neg\wedge]$ rule is applicable and we can split the tableau, writing $\neg(\mathbf{t} : \varphi)$ on one side and $\neg(\mathbf{t} : \psi)$ on the other. These are just alternatives for $\mathbf{t} : \neg\varphi$ and $\mathbf{t} : \neg\psi$ respectively, and may therefore be replaced by them. (36) sums up the procedure.

(36)
$$\begin{array}{c} \mathbf{t} : \neg(\varphi \wedge \psi) \\ \neg(\mathbf{t} : (\varphi \wedge \psi)) \\ \neg((\mathbf{t} : \varphi) \wedge (\mathbf{t} : \psi))) \\ \hline \neg(\mathbf{t} : \varphi) \quad \neg(\mathbf{t} : \psi) \\ \mathbf{t} : \neg\varphi \quad\quad \mathbf{t} : \neg\psi \end{array}$$

The justification of the other propositional rules is not much different and is left as an exercise.

Exercise 21 Give justifications of two more propositional rules in this way.

While the justification of each of the propositional rules in Table 2.1 hinges upon a similar rule for predicate logic, the correctness of the rules for P, F,

$\mathbf{t} : \mathsf{P}\varphi$	$\mathbf{t} : \mathsf{H}\varphi$	$\mathbf{t} : \neg\mathsf{P}\varphi$	$\mathbf{t} : \neg\mathsf{H}\varphi$
$\mid$	$\mathbf{t}_1 \prec \mathbf{t}$	$\mathbf{t}_1 \prec \mathbf{t}$	$\mid$
$\mathbf{t}_n \prec \mathbf{t}$	$\mid$	$\mid$	$\mathbf{t}_n \prec \mathbf{t}$
$\mathbf{t}_n : \varphi$	$\mathbf{t}_1 : \varphi$	$\mathbf{t}_1 : \neg\varphi$	$\mathbf{t}_n : \neg\varphi$
$\mathbf{t} : \mathsf{F}\varphi$	$\mathbf{t} : \mathsf{G}\varphi$	$\mathbf{t} : \neg\mathsf{F}\varphi$	$\mathbf{t} : \neg\mathsf{G}\varphi$
$\mid$	$\mathbf{t} \prec \mathbf{t}_1$	$\mathbf{t} \prec \mathbf{t}_1$	$\mid$
$\mathbf{t} \prec \mathbf{t}_n$	$\mid$	$\mid$	$\mathbf{t} \prec \mathbf{t}_n$
$\mathbf{t}_n : \varphi$	$\mathbf{t}_1 : \varphi$	$\mathbf{t}_1 : \neg\varphi$	$\mathbf{t}_n : \neg\varphi$

Table 2.2: Tableau expansion for temporal logic: Rules for temporal operators. In all cases $\mathbf{t}_n$ must be new.

H, and G in Table 2.2 crucially requires $[\forall]$ and $[\exists]$. The following tableau shows how the rule for $\mathbf{t} : \neg\mathsf{P}\varphi$ can be obtained.

(37)
$$\begin{array}{c} \mathbf{t} : \neg\mathsf{P}\varphi \\ \mathbf{t}_1 \prec \mathbf{t} \\ \neg(\mathbf{t} : \mathsf{P}\varphi) \\ \neg\exists t'(t' \prec \mathbf{t} \wedge t' : \varphi) \\ \forall t'\neg(t' \prec \mathbf{t} \wedge t' : \varphi) \\ \neg(\mathbf{t}_1 \prec \mathbf{t} \wedge \mathbf{t}_1 : \varphi) \\ \diagup \quad \diagdown \\ \neg\mathbf{t}_1 \prec \mathbf{t} \quad \neg(\mathbf{t}_1 : \varphi) \\ \times \qquad \mathbf{t}_1 : \neg\varphi \end{array}$$

Note that in this justification use was made of the $[\forall]$ rule and that $\neg(\mathbf{t}_1 \prec \mathbf{t} \wedge \mathbf{t}_1 : \varphi)$ can be obtained for *any* $\mathbf{t}_1$ already on the branch. When a concrete tableau is made it is therefore not correct to tick a sentence $\mathbf{t} : \neg\mathsf{P}\varphi$ when the rule has been applied to it. This holds for all rules that work with 'old' constants $\mathbf{t}_1$. Application of a rule that creates new constants $\mathbf{t}_n$ always comes with a tick.

EXERCISE 22 Give justifications similar to the one in (37) for one more rule for old constants and a rule creating a new constant.

Let us illustrate the tableau method with the help of some examples. In (38) the validity of $\mathsf{H}(P \to Q) \to (\mathsf{H}P \to \mathsf{H}Q)$ is checked by starting a tableau

$\mathbf{t}_1 : \varphi$ $\mathbf{t}_1 = \mathbf{t}_2$ $\mid$ $\mathbf{t}_2 : \varphi$	$\mathbf{t}_1 : \varphi$ $\mathbf{t}_2 = \mathbf{t}_1$ $\mid$ $\mathbf{t}_2 : \varphi$	$\mathbf{t}_1 = \mathbf{t}_2 \quad \neg\mathbf{t}_1 = \mathbf{t}_2$	$\neg\mathbf{t} = \mathbf{t}$ $\mid$ $\times$	$\mathbf{t} : \varphi$ $\mathbf{t} : \neg\varphi$ $\mid$ $\times$	γ $\neg\gamma$ $\mid$ $\times$

Table 2.3: Tableau expansion rules for temporal logic: Equality rules and closure rules. All constants must be old.

with $\mathbf{t} : \neg(\mathsf{H}(P \to Q) \to (\mathsf{H}P \to \mathsf{H}Q))$ (i.e. by assuming that the sentence does *not* hold at some point of time). Systematic search for a model by means of the tableau rules then fails.

(38)
$$\begin{array}{c}
\checkmark\mathbf{t} : \neg(\mathsf{H}(P \to Q) \to (\mathsf{H}P \to \mathsf{H}Q)) \\
\mathbf{t} : \mathsf{H}(P \to Q) \\
\checkmark\mathbf{t} : \neg(\mathsf{H}P \to \mathsf{H}Q) \\
\mathbf{t} : \mathsf{H}P \\
\checkmark\mathbf{t} : \neg\mathsf{H}Q \\
\mathbf{t}_1 \prec \mathbf{t} \\
\mathbf{t}_1 : \neg Q \\
\mathbf{t}_1 : P \\
\checkmark\mathbf{t}_1 : P \to Q \\
\mathbf{t}_1 : \neg P \quad \mathbf{t}_1 : Q \\
\times \qquad \times
\end{array}$$

Note that while many lines in the tableau receive a ✓, those that start with $\mathbf{t} : \mathsf{H}$ do not: the rule for H is essentially universal and has to be applied over and over again if necessary, as have the rules for G, $\neg\mathsf{F}$ and $\neg\mathsf{P}$.

A next example checks whether $Q \models \mathsf{HF}Q$. Note that this principle establishes a connection between the forward-looking F and the backward-looking H. A 'dual' principle is $Q \models \mathsf{GP}Q$; it can be proved in a similar way.

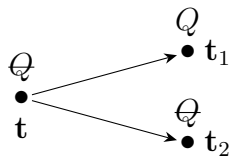
(39)
$$\begin{array}{c} \mathbf{t} : Q \\ \checkmark\mathbf{t} : \neg\mathsf{HF}Q \\ \mathbf{t}_1 \prec \mathbf{t} \\ \mathbf{t}_1 : \neg\mathsf{F}Q \\ \mathbf{t} : \neg Q \\ \times \end{array}$$

Counterexamples can also easily be obtained by the method, just as in the case of predicate logic. In the next example a model is sought that refutes $\models \mathsf{F}Q \rightarrow \mathsf{G}Q$—'If Q will once be, it will always be'. The search ends with an open branch.

(40)
$$\begin{array}{c} \checkmark\mathbf{t} : \neg(\mathsf{F}Q \rightarrow \mathsf{G}Q) \\ \checkmark\mathbf{t} : \mathsf{F}Q \\ \checkmark\mathbf{t} : \neg\mathsf{G}Q \\ \mathbf{t} \prec \mathbf{t}_1 \\ \mathbf{t}_1 : Q \\ \mathbf{t} \prec \mathbf{t}_2 \\ \mathbf{t}_2 : \neg Q \end{array}$$

This branch can then be converted into a counterexample, as before. It is given in (41) and additionally to the time of evaluation $\mathbf{t}$ involves two points later than it, one where Q is true (so that $\mathsf{F}Q$ will be true at $\mathbf{t}$), and one where Q is false (so that $\mathsf{G}Q$ fails to be true at $\mathbf{t}$).

(41)



The model in (41) does not give any ordering between the time points $\mathbf{t}_1$ and $\mathbf{t}_2$. In general the logic that we have obtained thus far leaves open many possibilities that some may want to exclude. This is good inasfar as logic alone should not decide what the structure of time should be. In fact, there have been widely diverging opinions about the basic ontology of time. It is not the task of logic to choose between such different perspectives. What logic can do is to hand tools to the metaphysician who wants to investigate the consequences of his basic assumptions. Logic should rule out as few reasonable assumptions as possible.

EXERCISE 23 Check the following with the help of tableaus using only rules from Tables 2.1, 2.2 and 2.3. Give counterexamples where appropriate.

a. $\mathsf{G}P \models \mathsf{G}(P \vee Q)$

b. $\models \mathsf{H}(P \wedge Q) \rightarrow (\mathsf{H}P \wedge \mathsf{H}Q)$

c. $\models \mathsf{H}(P \vee Q) \rightarrow (\mathsf{H}P \vee \mathsf{H}Q)$

d. $\models \mathsf{F}(P \vee Q) \rightarrow (\mathsf{F}P \vee \mathsf{F}Q)$

e. $\mathsf{G}(P \vee Q), \mathsf{G}(P \rightarrow R), \mathsf{G}(Q \rightarrow R) \models \mathsf{G}R$

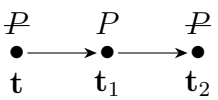
f. $\models \mathsf{G}P \rightarrow P$

g. $\models \mathsf{P}Q \rightarrow \mathsf{H}(\mathsf{F}Q \vee Q \vee \mathsf{P}Q)$

h. $\models \mathsf{P}Q \rightarrow \mathsf{GP}Q$

That the logic developed thus far does not exclude possibilities that a reasonable theorist of time may want to exclude is also illustrated in the next example, where a model refuting $\mathsf{G}P \rightarrow \mathsf{GG}P$ is obtained. The model in question is given in (43).

(42)
$$\begin{array}{c} \checkmark\mathbf{t} : \neg(\mathsf{G}P \rightarrow \mathsf{GG}P) \\ \mathbf{t} : \mathsf{G}P \\ \checkmark\mathbf{t} : \neg\mathsf{GG}P \\ \mathbf{t} \prec \mathbf{t}_1 \\ \checkmark\mathbf{t}_1 : \neg\mathsf{G}P \\ \mathbf{t}_1 : P \\ \mathbf{t}_1 \prec \mathbf{t}_2 \\ \mathbf{t}_2 : \neg P \end{array}$$

(43)


In this model $\mathbf{t}$ precedes $\mathbf{t}_1$, $\mathbf{t}_1$ precedes $\mathbf{t}_2$, but $\mathbf{t}$ does not precede $\mathbf{t}_2$! It is for this reason that P holds at all times that are later than $\mathbf{t}$ (only $\mathbf{t}_1$ fits that bill) so that $\mathsf{G}P$ holds at $\mathbf{t}$. On the other hand, $\mathsf{GG}P$ does not hold at $\mathbf{t}$: Because P fails at $\mathbf{t}_2$ and $\mathbf{t}_2$ is later than $\mathbf{t}_1$, $\mathsf{G}P$ fails at $\mathbf{t}_1$ and, since $\mathbf{t}_1$ is later than $\mathbf{t}$, $\mathsf{GG}P$ therefore fails to hold at $\mathbf{t}$.

Logic should not preclude the possibility of certain temporal ontologies. In Antiquity, for example, the notion of circular time, with eternal recurrence of events, was the norm. There are two ways to make sense of this notion. Either each moment precedes itself, in which case the $\prec$ relation cannot be taken to be irreflexive, or transitivity of $\prec$ fails to hold. In that case $t_1 \prec t_2$ comes to mean something like 't_2 is after t_1, but not more than half way round the time cycle'. The possibility is investigated in Reynolds (1994).

Another perspective on time which deviates from the usual pattern of viewing the 'earlier than' relation as linearly ordered is the view that, while the past is linear, time lines in the future may fork. This is the idea of *branching* time. It combines ideas about time with ideas about necessity and possibility.

Those who want time to be neither branching nor circular should perhaps adopt the first three axioms of (44), a set of constraints already discussed in the previous chapter. Together they turn the precedence relation into a *(strict) linear order*. The fourth axiom, *density*, which says that there is a point of time strictly between any two given ones, also seems reasonable for ordinary time, though not for computer time, which is measured in discrete steps. The fifth axiom says that time has no end, an idea at odds with the views expounded in Douglas Adams' *Hitchhiker's Guide to the Galaxy*; and the sixth axiom claims that time has no beginning, which, more seriously, may not be in accordance with certain interpretations of the Big Bang theory.

(44)

A1	$\forall t\, \neg t \prec t$	(irreflexivity)
A2	$\forall t_1 \forall t_2 \forall t_3((t_1 \prec t_2 \wedge t_2 \prec t_3) \rightarrow t_1 \prec t_3)$	(transitivity)
A3	$\forall t_1 \forall t_2 (t_1 \prec t_2 \vee t_2 \prec t_1 \vee t_1 = t_2)$	(connectedness)
A4	$\forall t_1 \forall t_2 (t_1 \prec t_2 \rightarrow \exists t_3 (t_1 \prec t_3 \wedge t_3 \prec t_2))$	(density)
A5	$\forall t_1 \exists t_2\ t_1 \prec t_2$	(no end)
A6	$\forall t_1 \exists t_2\ t_2 \prec t_1$	(no beginning)

In the previous chapter we have seen how principles such as the ones in (44) can sometimes be turned into derived rules and in Table 2.4 the derived rules corresponding to the axioms in (44) are repeated. Such rules can be used once it is decided that certain axioms are to be adopted. For example, in the presence of transitivity the validity of $\mathsf{P}Q \rightarrow \mathsf{GP}Q$ ('If Q has been true in the past, it will always be the case that it has been true') can be shown: The sentence is true at all time points in all models where $\prec$ is transitive. Note the crucial step concluding $\mathbf{t}_1 \prec \mathbf{t}_2$ from $\mathbf{t}_1 \prec \mathbf{t}$ and $\mathbf{t} \prec \mathbf{t}_2$.

Irreflexivity:	Transitivity:	Connectedness:
$\neg \mathbf{t} \prec \mathbf{t}$	$\mathbf{t}_1 \prec \mathbf{t}_2$ $\mathbf{t}_2 \prec \mathbf{t}_3$ $\mathbf{t}_1 \prec \mathbf{t}_3$	$\mathbf{t}_1 \prec \mathbf{t}_2 \quad \mathbf{t}_2 \prec \mathbf{t}_1 \quad \mathbf{t}_1 = \mathbf{t}_2$
($\mathbf{t}$ old)	($\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$ old)	($\mathbf{t}_1$, $\mathbf{t}_2$ old)
Density:	**No End:**	**No Beginning:**
$\mathbf{t}_1 \prec \mathbf{t}_2$ $\mathbf{t}_1 \prec \mathbf{t}_3$ $\mathbf{t}_3 \prec \mathbf{t}_2$	$\mathbf{t}_1 \prec \mathbf{t}_2$	$\mathbf{t}_2 \prec \mathbf{t}_1$
($\mathbf{t}_1$, $\mathbf{t}_2$ old; $\mathbf{t}_3$ new)	($\mathbf{t}_1$ old; $\mathbf{t}_2$ new)	($\mathbf{t}_1$ old; $\mathbf{t}_2$ new)

Table 2.4: Derived rules in the presence of certain axioms.

(45)
$$\begin{array}{c} \checkmark \mathbf{t} : \neg(\mathsf{P}Q \rightarrow \mathsf{GP}Q) \\ \checkmark \mathbf{t} : \mathsf{P}Q \\ \checkmark \mathbf{t} : \neg\mathsf{GP}Q \\ \mathbf{t}_1 \prec \mathbf{t} \\ \mathbf{t}_1 : Q \\ \mathbf{t} \prec \mathbf{t}_2 \\ \mathbf{t}_2 : \neg\mathsf{P}Q \\ \mathbf{t}_1 \prec \mathbf{t}_2 \\ \mathbf{t}_1 : \neg Q \\ \times \end{array}$$

Let us write $\varphi_1, \ldots, \varphi_n \models_{\mathrm{AX}} \psi$ if AX is a set of axioms and it holds that $\mathrm{AX}, \mathbf{t} : \varphi_1, \ldots, \mathbf{t} : \varphi_n \models \mathbf{t} : \psi$ (i.e. the axioms from AX plus $\mathbf{t} : \varphi_1, \ldots, \mathbf{t} : \varphi_n$ entail $\mathbf{t} : \psi$). The following exercise lets you derive certain validities in the presence of some of the axioms in (44).

EXERCISE 24 a. Show that $\models_{\mathrm{AX}} \mathsf{PP}Q \rightarrow \mathsf{P}Q$ if Transitivity is an element of AX.

b. Show that $\models_{\mathrm{AX}} \mathsf{P}Q \rightarrow \mathsf{H}(\mathsf{F}Q \vee Q \vee \mathsf{P}Q)$ if Connectedness is an element of AX.

c. Show that $\models_{\mathrm{AX}} \mathsf{G}(\mathsf{G}Q \to P) \vee \mathsf{G}((P \wedge \mathsf{G}P) \to Q)$ if Connectedness is an element of AX.

d. Show that $\models_{\mathrm{AX}} \mathsf{P}Q \to \mathsf{PP}Q$ if Density is an element of AX.

e. Show that $\models_{\mathrm{AX}} \mathsf{G}Q \to \mathsf{F}Q$ if No End is an element of AX.

f. Show that $\models_{\mathrm{AX}} \mathsf{H}Q \to \mathsf{P}Q$ if No Beginning is an element of AX.

EXERCISE 25 Suppose that No End is the only axiom in AX. Use the tableau method to find a counterexample to $\models_{\mathrm{AX}} \mathsf{G}Q \to \mathsf{GG}Q$. What happens? Can you let the tableau generate a *finite* counterexample?

2.2 Modal Logics

Modal logic deals with operators expressing properties of propositions, such as 'it is necessary that', 'it ought to be the case that', 'it is permissable that', and 'Mary believes that'. This branch of logic is as old as the discipline itself and starts with Aristotle, who devotes two chapters of *De Interpretatione* to necessity and possibility.

The temporal logic considered in the previous section studied operators such as 'it will be the case that' and 'it has always been true that'. These also express properties of propositions ('it will be the case that pigs have wings', for example, expresses a property of the proposition expressed by 'pigs have wings', namely that it will be true at some point in the future) and temporal logic therefore can also be considered to be a modal logic. In fact, the formal aspects of the logic we are about to define will resemble those of temporal logic closely. But while we have analysed the temporal operators in terms of quantification over moments of time, we will analyse the new modal operators in terms of quantification over *possible worlds*. And while in temporal logic the 'earlier than' relation played an important role, this role will now be taken over by various binary relations between possible worlds which will be called *accessibility relations*.

A possible world is commonly understood as a 'way things might have been'. There are possible worlds in which pigs fly, in which alligators are vegetarians, and in which objects travel faster than the speed of light. The *actual* world, 'the way things are', is also considered to be a possible world. In fact, any complete and consistent description of facts may be thought to describe a possible world. The idea of possible worlds dates back at least as far as Leibniz, who used them in his *Theodicy* in an attempt to solve the problem of evil (the inconsistency of the existence of evil with the Christian doctrines of the omniscience, omnipotence and moral perfection of God). Leibniz' solution was that God bestows actuality on the best of all possible worlds, a theory later ridiculed by Voltaire in his novel *Candide*.

Some possible worlds are more possible than others. A world in which particles travel faster than light may be logically possible, but physically it is not. And while a world in which you kill your grandmother may be logically and physically possible, it may not be deemed possible from a moral point of view. On the other hand, while it may be true in all physically possible worlds that a comet will destroy the Earth tomorrow, there may be epistemically possible worlds where the calamity does not occur. All these different kinds of

possibility will be modeled with the accessibility relations mentioned above. For example, in order to model the notion of physical possibility of worlds we introduce a binary relation Φ, where $\Phi ww'$ formalizes 'w' obeys the physical laws that hold in w'. If this is indeed the case then w' can be said to be physically possible in w, or Φ-*accessible from* w.

Once a binary relation R between worlds is selected it can be used to define the kind of operators we are after. For each such relation R we will define a *box* $[R]$ and a *diamond* $\langle R\rangle$ that can be written in front of a sentence. Below we will define a translation into predicate logic to the effect that a statement $[R]\varphi$ (φ is R-necessary) is true in a given world w if φ is true in all worlds w' such that Rww', while $\langle R\rangle\varphi$ (φ is R-possible) will be defined to be true in w if φ is true in some w' such that Rww'. In such a set-up $[\Phi]\varphi$ says that φ is physically necessary (in a given world) and $\langle\Phi\rangle\varphi$ that φ is physically possible.

Before we flesh out this intended semantics further, let us introduce some more accessibility relations and the operators that come with them.

Deontic accessibility. Deontic logic is the logic of obligation and permission. In order to model it, we introduce a deontic accessibility relation O with the informal meaning that $\mathsf{O}ww'$ stands for 'everything that is morally desirable in w holds in w''. The operators $[\mathsf{O}]$ and $\langle\mathsf{O}\rangle$ can be glossed as follows.

$$\begin{array}{rcl} [\mathsf{O}]\varphi & — & \text{it is obligatory that } \varphi \\ \langle\mathsf{O}\rangle\varphi & — & \text{it is permitted that } \varphi \end{array}$$

Metaphysical accessibility. A metaphysical (or: alethic) accessibility relation N will underly operators $[\mathsf{N}]$ and $\langle\mathsf{N}\rangle$, which have the following informal meanings.

$$\begin{array}{rcl} [\mathsf{N}]\varphi & — & \text{it is metaphysically necessary that } \varphi \\ \langle\mathsf{N}\rangle\varphi & — & \text{it is metaphysically possible that } \varphi \end{array}$$

Exactly how metaphysical accessibility is informally interpreted will have to depend on one's metaphysics, of course. Logic can only study the formal properties of such notions.

Universal accessibility. The universal (or: global) accessibility relation A is just the relation that holds between *any* two possible worlds. This can be a

useful notion to have around for technical reasons. Some authors treat the metaphysical accessibility relation N as universal, but that is a choice that need not be made. The operators that come with A can be glossed as follows.

$[\mathsf{A}]\varphi$ — it is globally necessary that φ
$\langle\mathsf{A}\rangle\varphi$ — it is globally possible that φ

Given the semantics below, $[\mathsf{A}]\varphi$ will just mean that φ holds in *all* possible worlds, while $\langle\mathsf{A}\rangle\varphi$ says that φ is true in *some* world. This is the semantics that modal operators obtain in treatments that make no use of accessibility.

Doxastic and epistemic accessibility. Doxastic logic (the logic of beliefs) and epistemic logic (the logic of knowledge) can also be treated as modal logics, as was shown in Hintikka (1962). To this end two *ternary* relations are introduced, the doxastic accessibility relation B, and the epistemic relation K, each of them relations between an entity (the person who believes or knows something) and two possible worlds. The following are intuitive glosses.

$\mathsf{B}aww'$ — world w' is compatible with agent a's beliefs in w
$\mathsf{K}aww'$ — world w' is compatible with agent a's knowledge in w

While B and K are ternary relations officially, we consider $\mathsf{B}a$ and $\mathsf{K}a$ to be binary relations between possible worlds, for any a. In order to emphasize this we will often write B_a for $\mathsf{B}a$ and K_a for $\mathsf{K}a$. The chapter on type logics will give formal underpinnings to our decision to view B_a and K_a as binary relations.

These accessibility relations lead to the following operators.

$[\mathsf{B}_a]\varphi$ — a believes that φ (better: φ follows from a's beliefs)
$\langle\mathsf{B}_a\rangle\varphi$ — φ is compatible with a's beliefs
$[\mathsf{K}_a]\varphi$ — a knows that φ (better: φ follows from a's knowledge)
$\langle\mathsf{K}_a\rangle\varphi$ — φ is compatible with a's knowledge

2.2.1 Basic Modal Logic

Now that we have introduced the modal operators we want to investigate, let us define the language of propositional modal logic. In the following $\mathcal{R} = \{\Phi,\ \mathsf{O},\ \mathsf{N},\ \mathsf{A}\} \cup \{\mathsf{B}_c \mid c \in \mathcal{C}\} \cup \{\mathsf{K}_c \mid c \in \mathcal{C}\}$, where $\mathcal{C} \subseteq \mathcal{V}$ is the

set of individual constants in our vocabulary. Readers interested in further modalities may want to add extra elements to the set $\mathcal{R}$ under consideration.

DEFINITION 9 An *atomic sentence of propositional modal logic* is an n-ary relation symbol followed by n individual constants (as in Kjm), while a *sentence of propositional modal logic* is any string that can be built up with the help of the following clauses.

a. Any atomic sentence of propositional modal logic is a sentence of propositional modal logic.

b. If φ is a sentence of propositional modal logic then $\neg\varphi$ is also a sentence of propositional modal logic.

c. If φ and ψ are sentences of propositional modal logic, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$ are sentences of propositional modal logic.

d. If φ is a sentence of propositional modal logic and $R \in \mathcal{R}$, then $[R]\varphi$ and $\langle R\rangle\varphi$ are sentences of propositional modal logic.

The language as it is defined here allows us to combine the new operators freely, as in $[\mathsf{O}]([\mathsf{K}_m]Sj \rightarrow [\mathsf{B}_m]Sb)$, which might be used to render *if Mary knows that John sleeps she ought to believe that Bill does.*

EXERCISE 26 Translate the following into our modal language.

a. You ought to know that I love you

b. Mary believes that John ought to kiss her

c. Mary knows that John must kiss her if he believes that she knows that he ought to admire her

In giving a semantics for this language we will again use Prior's (1967) approach, but will now consider pairs $w : \varphi$ (where w stands for a possible world) instead of the pairs $t : \varphi$ used earlier. Pairs $w : \varphi$ will be considered as shorthand for predicate logical formulas. The following clauses give the translation.

(46) a. $w : Pc_1 \ldots c_n \triangleq Pc_1 \ldots c_n w$

b. $w : (\neg\varphi) \triangleq \neg(w : \varphi)$

c. $w : (\varphi \wedge \psi) \triangleq (w : \varphi) \wedge (w : \psi)$

d. $w : (\varphi \vee \psi) \triangleq (w : \varphi) \vee (w : \psi)$

e. $w : (\varphi \rightarrow \psi) \triangleq (w : \varphi) \rightarrow (w : \psi)$

f. $w : (\varphi \leftrightarrow \psi) \triangleq (w : \varphi) \leftrightarrow (w : \psi)$

g. $w : [R]\varphi \triangleq \forall w'(Rww' \rightarrow w' : \varphi)$, if $R \in \mathcal{R}$

h. $w : \langle R\rangle\varphi \triangleq \exists w'(Rww' \wedge w' : \varphi)$, if $R \in \mathcal{R}$

Note that these clauses follow the ones in (31), (32) and (33) very closely. The clause for $w : [R]\varphi$, in particular, should be compared to that for $t : \mathsf{G}\varphi$ in (33), while the treatment of $w : \langle R\rangle\varphi$ should be compared to that of $t : \mathsf{F}\varphi$.

EXERCISE 27 Explain how the clauses for $t : \mathsf{P}\varphi$ and $t : \mathsf{H}\varphi$ in (33) can be adapted to follow those for $w : \langle R\rangle\varphi$ and $w : [R]\varphi$ more closely.

EXERCISE 28 Do a step-by-step translation of $w : [\mathsf{O}]([\mathsf{K}_m]Sj \rightarrow [\mathsf{B}_m]Sb)$.

Tableau rules follow in this translation's footsteps. They are listed in Tables 2.5, 2.6 and 2.7. In (47) it is shown how one of the propositional rules can be obtained with the help of the translation in (46) and the tableau rules for predicate logic. The reasoning here is completely analogous to that in (36) (see also the discussion of (36) in the previous section). More in general the rules in Table 2.5 are very similar to those in Table 2.1; the only difference being that the ts are replaced by ws.

(47)
$$\begin{array}{c} \mathbf{w} : \neg(\varphi \wedge \psi) \\ \neg(\mathbf{w} : (\varphi \wedge \psi)) \\ \neg((\mathbf{w} : \varphi) \wedge (\mathbf{w} : \psi)) \\ \swarrow \quad \searrow \\ \neg(\mathbf{w} : \varphi) \quad \neg(\mathbf{w} : \psi) \\ \mathbf{w} : \neg\varphi \quad\quad \mathbf{w} : \neg\psi \end{array}$$

The most interesting rules of the calculus are those for the box and the diamond, but again will not introduce really new ideas. In particular, the rules for $[R]$ in Table 2.6 should be compared to those for G in Table 2.2

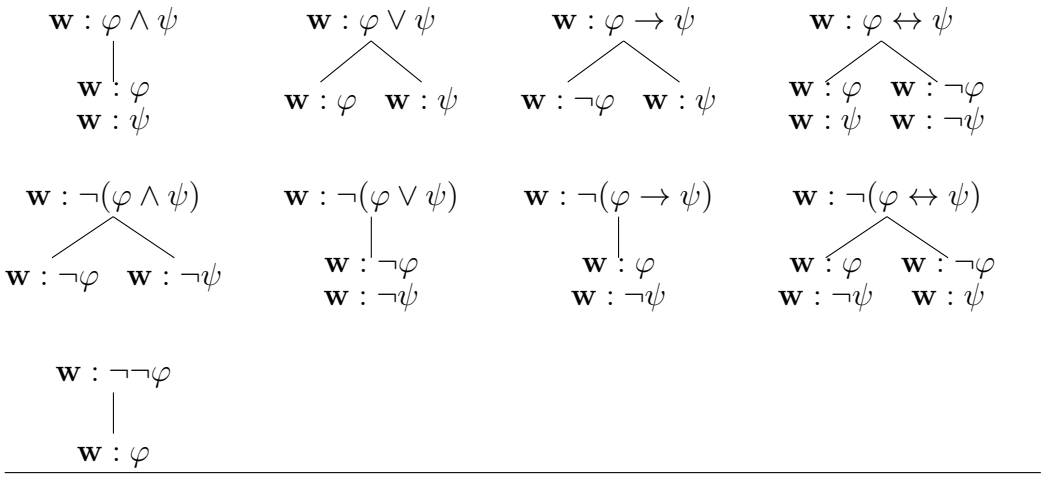


Table 2.5: Tableau expansion for modal logic: propositional rules.

while the two rules for $\langle R\rangle$ should be compared to those for F. These four rules are really templates and in each of them the Rs can be replaced by any of the accessibility relations in $\mathcal{R}$, so that (48), for example, is one of the many possible instantiations.

(48)
$$\begin{array}{c}\mathbf{w} : \langle \mathsf{B}_a\rangle\varphi\\ \hline \mathsf{B}_a\mathbf{w}\mathbf{w}_n\\ \mathbf{w}_n : \varphi\end{array}$$

Here is a justification of the rule for $\mathbf{w} : \neg\langle R\rangle\varphi$. It should be compared to (37).

(49)
$$\begin{array}{c}\mathbf{w} : \neg\langle R\rangle\varphi\\ R\mathbf{w}\mathbf{w}_1\\ \neg(\mathbf{w} : \langle R\rangle\varphi)\\ \neg\exists w'(R\mathbf{w}w' \wedge w' : \varphi)\\ \forall w'\neg(R\mathbf{w}w' \wedge w' : \varphi)\\ \neg(R\mathbf{w}\mathbf{w}_1 \wedge \mathbf{w}_1 : \varphi)\\ \hline \begin{array}{cc}\neg R\mathbf{w}\mathbf{w}_1 & \neg(\mathbf{w}_1 : \varphi)\\ \times & \mathbf{w}_1 : \neg\varphi\end{array}\end{array}$$

$\mathbf{w} : [R]\varphi$	$\mathbf{w} : \neg[R]\varphi$	$\mathbf{w} : \langle R\rangle\varphi$	$\mathbf{w} : \neg\langle R\rangle\varphi$
$R\mathbf{w}\mathbf{w}_1$	$\vert$	$\vert$	$R\mathbf{w}\mathbf{w}_1$
$\vert$	$R\mathbf{w}\mathbf{w}_n$	$R\mathbf{w}\mathbf{w}_n$	$\vert$
$\mathbf{w}_1 : \varphi$	$\mathbf{w}_n : \neg\varphi$	$\mathbf{w}_n : \varphi$	$\mathbf{w}_1 : \neg\varphi$

Table 2.6: Rules for modal operators. In all cases $\mathbf{w}_n$ must be new. R can be replaced by any of the accessibility relations we have discussed.

Table 2.7 below gives some more rules for equality and closure that are completely analogous to the rules in Table 2.3.

The rules in Tables 2.5, 2.6 and 2.7 define a proof system for a basic multimodal logic in which all boxes are treated alike, as are a all diamonds. How this basic logic can be finetuned for each set of operators $[R]$, $\langle R\rangle$ will be the topic of the next section.

EXERCISE 29 Test the following modal sentences to determine whether they are valid. Give counterexamples where possible. In each instance discuss the outcomes. Are they reasonable, given the interpretations of the operators involved?

a. $[\mathsf{O}]\langle\mathsf{O}\rangle Q \to \langle\mathsf{O}\rangle[\mathsf{O}]Q$

b. $[\mathsf{B}_j][\mathsf{B}_j]Q \to [\mathsf{B}_j]Q$

c. $[\mathsf{O}]Q \to Q$

d. $[\mathsf{O}]Q \to \langle\mathsf{O}\rangle Q$

e. $[\mathsf{K}_j]Q \to Q$

f. $[\mathsf{K}_j][\mathsf{K}_j]Q \to [\mathsf{K}_j]Q$

g. $[\mathsf{K}_j]Q \to [\mathsf{K}_j][\mathsf{K}_j]Q$

h. $\neg[\mathsf{K}_j]Q \to [\mathsf{K}_j]\neg[\mathsf{K}_j]Q$

i. $Q \to [\mathsf{N}]\langle\mathsf{N}\rangle Q$

j. $[\mathsf{O}]([\mathsf{O}]Q \to Q)$

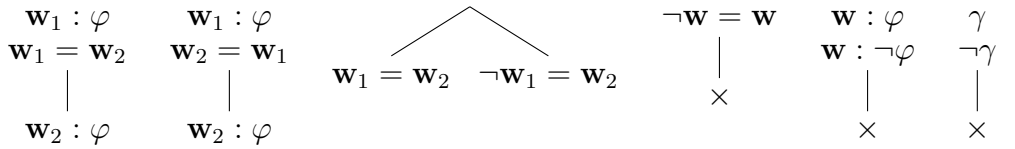


Table 2.7: Tableau expansion rules for modal logic: Equality rules and closure rules. All constants must be old.

k. $[\mathsf{O}]Q \to [\mathsf{O}](P \to Q)$

l. $[\mathsf{O}](Q \vee \neg Q)$

EXERCISE 30 Show that the following are not valid. What extra non-logical constraints on accessibility relations could be adopted to make them so?

a. $[\mathsf{O}]Q \to \langle\Phi\rangle Q$ (Ought implies Can)

b. $[\mathsf{K}_j]Q \to [\mathsf{B}_j]Q$ (Knowledge implies Belief)

2.2.2 Putting Constraints on Accessibility Relations

Until now we have treated all modalities alike, but some reflection shows that this equal treatment cannot be the whole story. In exercises 29 and 30 it became apparent that some reasonable principles remain invalid. A counterexample was found to $[\mathsf{K}_j]Q \to Q$, for example, but this surely cannot be right, since a statement must be true whenever it is known to be true. A similar principle for belief, $[\mathsf{B}_j]Q \to Q$, clearly fails, however. So, on the one hand our logic is still too weak to model reasoning with certain modal operators, while on the other hand not all operators should be treated on a par, because they exhibit different behaviour.

As in the case of tense logic, it is possible to get the desired inferential properties by imposing certain non-logical constraints on the accessibility relations under consideration—Φ, O, N, A, B_a and K_a. We will discuss them one by one.

Reflexivity of Φ:	Transitivity of Φ:	Seriality of O:
$\Phi\mathbf{ww}$	$\Phi\mathbf{w}_1\mathbf{w}_2$ $\Phi\mathbf{w}_2\mathbf{w}_3$ $\Phi\mathbf{w}_1\mathbf{w}_3$	$\mathsf{O}\mathbf{w}_1\mathbf{w}_2$
($\mathbf{w}$ old)		($\mathbf{w}_1$ old, $\mathbf{w}_2$ new)
Shift Reflexivity of O:	**Reflexivity of N:**	**Universality of A:**
$\mathsf{O}\mathbf{w}_1\mathbf{w}_2$ $\mathsf{O}\mathbf{w}_2\mathbf{w}_2$	$\mathsf{N}\mathbf{ww}$	$\mathsf{A}\mathbf{w}_1\mathbf{w}_2$
	($\mathbf{w}$ old)	($\mathbf{w}_1$, $\mathbf{w}_2$ old)

Table 2.8: Derived rules for Φ, O, N, and A in the presence of certain axioms.

Physical accessibility. We have explained the physical accessibility relation Φ by stipulating that $\Phi ww'$ formalizes 'w' obeys the physical laws that hold in w'. This relation is reflexive and transitive, so that the following must hold.

- $\forall w\, \Phi ww$
- $\forall w_1 \forall w_2 \forall w_3 ((\Phi w_1 w_2 \wedge \Phi w_2 w_3) \rightarrow \Phi w_1 w_3)$

Adopting these as axioms immediately leads to the derived tableau rules shown in Table 2.8. As a consequence the following principles become derivable.

(T_Φ) $[\Phi]\varphi \rightarrow \varphi$

(4_Φ) $[\Phi]\varphi \rightarrow [\Phi][\Phi]\varphi$

EXERCISE 31 Show that (T_Φ) is valid if Φ is assumed to be reflexive and that (4_Φ) holds if transitivity of Φ is assumed.

There is a converse to this, as it can be shown that any model M that satisfies $\mathbf{w} : [\Phi]Q \rightarrow Q$, for each interpretation of $\mathbf{w}$ and Q in M, is reflexive,[3] and that any model that satisfies $\mathbf{w} : [\Phi]Q \rightarrow [\Phi][\Phi]Q$, for any interpretation of $\mathbf{w}$ and Q, is transitive. This point will be discussed in a next chapter.

[3]More precisely: Let $M = \langle D, I\rangle$ be a model such that $\langle D, I'\rangle$ satisfies $\mathbf{w} : [\Phi]Q \rightarrow Q$ for each I' equal to I except possibly for the values of $\mathbf{w}$ and Q. Then M is reflexive.

Deontic accessibility. Here we discuss two desirable modal principles first, before introducing constraints on O from which they will be obtainable. A first reasonable principle is that everything that is obliged is also permitted:

$(\mathrm{D}_{\mathsf{O}})$ $[\mathsf{O}]\varphi \to \langle \mathsf{O} \rangle \varphi$

With $(\mathrm{D}_{\mathsf{O}})$ as the only extra axiom a system of deontic logic is obtained that is called SDL in the literature.[4] A second reasonable principle is the following.

$(\mathrm{T}^{+}_{\mathsf{O}})$ $[\mathsf{O}]([\mathsf{O}]\varphi \to \varphi)$

While it is not true that everything that ought to be the case is in fact the case ($[\mathsf{O}]\varphi \to \varphi$), this is in fact a deplorable situation and $(\mathrm{T}^{+}_{\mathsf{O}})$ says that it should be the case that if something ought to be so it is in fact true. Adding $(\mathrm{T}^{+}_{\mathsf{O}})$ to SDL leads to a system McNamara (2010) calls SDL^{+}.

It turns out that $(\mathrm{D}_{\mathsf{O}})$ leads to Seriality (the No End requirement in a temporal context), while $(\mathrm{T}^{+}_{\mathsf{O}})$ leads to a constraint called Shift Reflexivity. Here they are.

- $\forall w_1 \exists w_2 \, \mathsf{O} w_1 w_2$
- $\forall w_1 \forall w_2 (\mathsf{O} w_1 w_2 \to \mathsf{O} w_2 w_2)$

Table 2.8 again gives the corresponding tableau rules.

EXERCISE 32 Show that $(\mathrm{D}_{\mathsf{O}})$ is valid if O is assumed to be serial and that $(\mathrm{T}^{+}_{\mathsf{O}})$ can be obtained if shift reflexivity of O is assumed.

Metaphysical accessibility. If anything is metaphysically necessary it presumably is true, so we should have $[\mathsf{N}]\varphi \to \varphi$, which corresponds to the requirement that N is reflexive. We have assumed this to be the case in Table 2.8. Other principles are possible, but depend a lot on one's conception of metaphysical necessity.

EXERCISE 33 Discuss other possible principles regulating metaphysical necessity and possibility, such as $[\mathsf{N}]\varphi \to [\mathsf{N}][\mathsf{N}]\varphi$ and $\varphi \to [\mathsf{N}]\langle \mathsf{N} \rangle \varphi$.

[4] See McNamara (2010) for an overview of this system and its (many) limitations. Note that McNamara writes OB where we use $[\mathsf{O}]$ and PE for our $\langle \mathsf{O} \rangle$. His other operators ($\mathsf{IM}$, OM, and OP) are easily defined with the help of $[\mathsf{O}]$ and $\langle \mathsf{O} \rangle$.

Universal accessibility. The idea behind universal accessibility is that any two possible worlds stand in the relation. We thus adopt the axiom $\forall w_1 \forall w_2\, \mathsf{A} w_1 w_2$ and add the corresponding rule to Table 2.8.

Doxastic accessibility. It is at least arguable that if a person believes a certain statement, she also believes that she believes it and that if she does not believe something, she believes that she does not. In a rather idealised sense of 'belief' it moreover is the case that everyone's beliefs are consistent. This makes that the following principles are at least up for consideration.

(D_B) $[\mathsf{B}_a]\varphi \to \langle \mathsf{B}_a \rangle \varphi$

(4_B) $[\mathsf{B}_a]\varphi \to [\mathsf{B}_a][\mathsf{B}_a]\varphi$

(5_B) $\neg[\mathsf{B}_a]\varphi \to [\mathsf{B}_a]\neg[\mathsf{B}_a]\varphi$

These principles, which together with our basic system give a logic that is often called $\mathsf{KD45}$,[5] correspond to Seriality, Transitivity and Euclideaness of B_a respectively:

- $\forall w_1 \exists w_2\, \mathsf{B}_a w_1 w_2$
- $\forall w_1 \forall w_2 \forall w_3 ((\mathsf{B}_a w_1 w_2 \wedge \mathsf{B}_a w_2 w_3) \to \mathsf{B}_a w_1 w_3)$
- $\forall w_1 \forall w_2 \forall w_3 ((\mathsf{B}_a w_1 w_2 \wedge \mathsf{B}_a w_1 w_3) \to \mathsf{B}_a w_2 w_3)$

Table 2.9 gives the corresponding rules.

EXERCISE 34 Show that (5_B) is valid given euclideaness of B_a.

Epistemic accessibility. For a certain concept of idealised knowledge (or information) principles analogous to (4_B) and (5_B) seem reasonable, but (D_B) can be replaced by the stronger (T_K), as knowledge is *factive*—what is known is true.

(T_K) $[\mathsf{K}_a]\varphi \to \varphi$

(4_K) $[\mathsf{K}_a]\varphi \to [\mathsf{K}_a][\mathsf{K}_a]\varphi$

[5] Modal logics often get names on the basis of their defining axioms. Here K stands for (K_B), the principle $[\mathsf{B}_a](\varphi \to \psi) \to ([\mathsf{B}_a]\varphi \to [\mathsf{B}_a]\psi)$, which can be obtained from our basic system. The axiom names K, D, 4, 5, etc. are traditional.

Seriality of B_a:	Transitivity of B_a:	Euclideaness of B_a:
$\mathsf{B}_a\mathbf{w}_1\mathbf{w}_2$	$\mathsf{B}_a\mathbf{w}_1\mathbf{w}_2$ $\mathsf{B}_a\mathbf{w}_2\mathbf{w}_3$ $\mathsf{B}_a\mathbf{w}_1\mathbf{w}_3$	$\mathsf{B}_a\mathbf{w}_1\mathbf{w}_2$ $\mathsf{B}_a\mathbf{w}_1\mathbf{w}_3$ $\mathsf{B}_a\mathbf{w}_2\mathbf{w}_3$
($\mathbf{w}_1$ old, $\mathbf{w}_2$ new)		
Reflexivity of K_a:	Transitivity of K_a:	Euclideaness of K_a:
$\mathsf{K}_a\mathbf{ww}$	$\mathsf{K}_a\mathbf{w}_1\mathbf{w}_2$ $\mathsf{K}_a\mathbf{w}_2\mathbf{w}_3$ $\mathsf{K}_a\mathbf{w}_1\mathbf{w}_3$	$\mathsf{K}_a\mathbf{w}_1\mathbf{w}_2$ $\mathsf{K}_a\mathbf{w}_1\mathbf{w}_3$ $\mathsf{K}_a\mathbf{w}_2\mathbf{w}_3$
($\mathbf{w}$ old)		

Table 2.9: Derived rules for B and K in the presence of certain axioms.

(5_{K}) $\neg[\mathsf{K}_a]\varphi \to [\mathsf{K}_a]\neg[\mathsf{K}_a]\varphi$

This leads to requirements of reflexivity, transitivity and euclideaness of K_a (for rules consult Table 2.9):

- $\forall w_1 \exists w_2\, \mathsf{K}_a w_1 w_2$
- $\forall w_1 \forall w_2 \forall w_3((\mathsf{K}_a w_1 w_2 \wedge \mathsf{K}_a w_2 w_3) \to \mathsf{K}_a w_1 w_3)$
- $\forall w_1 \forall w_2 \forall w_3((\mathsf{K}_a w_1 w_2 \wedge \mathsf{K}_a w_1 w_3) \to \mathsf{K}_a w_2 w_3)$

Exercise 35 Show that any relation that is euclidean and reflexive is also symmetric. Conclude that K_a is an equivalence relation if it is relexive, transitive and euclidean.

Exercise 36 Show that $\varphi \to [\mathsf{K}_a]\langle \mathsf{K}_a\rangle\varphi$ is valid if symmetry of K_a is assumed.

Exercise 37 Show that the following principles are valid if K_a is assumed to be reflexive, transitive, and euclidean:

a. $[\mathsf{K}_a][\mathsf{K}_a]Q \leftrightarrow [\mathsf{K}_a]Q$

b. $[\mathsf{K}_a]\langle\mathsf{K}_a\rangle Q \leftrightarrow \langle\mathsf{K}_a\rangle Q$

c. $\langle\mathsf{K}_a\rangle[\mathsf{K}_a]Q \leftrightarrow [\mathsf{K}_a]Q$

The system that results from adopting reflexivity, transitivity and euclideaness of K_a, often called S5, or KT45, is widely studied in computer science. The canonical reference is Fagin, Halpern, Moses, and Vardi (1995).

Interaction postulates. There may also be interactions between various modalities. The principle that says that knowledge implies belief is an obvious example:

$$[\mathsf{K}_a]\varphi \rightarrow [\mathsf{B}_a]\varphi$$

This corresponds to the requirement that

$$\forall w_1 \forall w_2((\mathsf{B}_a w_1 w_2 \rightarrow \mathsf{K}_a w_1 w_2) \ ,$$

which in its turn boils down to the following rule:

$$\frac{\mathsf{B}_a \mathbf{w}_1 \mathbf{w}_2}{\mathsf{K}_a \mathbf{w}_1 \mathbf{w}_2}$$

EXERCISE 38 Show that adopting this rule makes $[\mathsf{K}_a]\varphi \rightarrow [\mathsf{B}_a]\varphi$ derivable.

Chapter 3

Types

Type-logical semantics[1] studies linguistic meaning with the help of the theory of types. The latter originated with Russell as an answer to the paradoxes, but has the additional virtue that it is very close to ordinary language. In fact, type theory is so much more similar to language than predicate logic is, that adopting it as a vehicle of representation can overcome the mismatches between grammatical form and predicate logical form that were observed by Frege and Russell. The grammatical forms of ordinary language sentences consequently may be taken to be much less misleading than logicians in the first half of the 20th century often thought them to be. This was realized by Richard Montague, who used the theory of types to translate fragments of ordinary language into a logical language.

Semantics is commonly divided into *lexical* semantics, which studies the meaning of words, and *compositional* semantics, which studies the way in which complex phrases obtain a meaning from their constituents. The strength of type-logical semantics lies with the latter, but type-logical theories can be combined with many competing hypotheses about lexical meaning, provided these hypotheses are expressed using the language of type theory.

3.1 Typing Words and Objects

Type-logical semantics is usually based on some variant of Alonzo Church's formulation of the simple theory of types (Church 1940) and it is impossible to explain the application without also explaining the underlying theory. In

[1]This chapter is based upon Muskens (2011).

this entry we will start with giving a more or less informal account of the formal underpinnings of the theory, moving to their linguistic application as we proceed. For an exposition of the relation between the *simple* theory of types and Russell's so-called *ramified* theory, see the entry on THEORY OF TYPES.

Typing is allotting the objects in one's ontology to separate categories. Expressions are also categorized and receive the same types as the objects they refer to. Truth-values, for example, should be distinguished from entities of the cabbages and kings variety and these in their turn from, say, possible worlds and other kinds of objects. Predicates should be distinguished from individuals; two-place predicates from one-place predicates; and predicates of predicates (*no student*, for example, is often classified as a predicate of predicates, as it combines with the predicate *smokes* to form a sentence) should be distinguished from predicates of predicates of predicates.

Type theory provides the book-keeping system that allows one to keep track of all these different categories. The idea is to first choose types for the most basic ontological categories one wants to adopt—say t for truth-values, e for entities, and s for worlds—and to then stipulate that if α and β are types, $(\alpha\beta)$ is also a type. The latter is meant to be the type of functions that take objects of type α as their arguments and return a value of type β. For example, (et) is the type of functions from entities to truth values, the type of one-place predicates in a set-up of the theory that abstracts from possible worlds. Other examples of types are (st), $(e(st))$, and $((e(st))((e(st))(st)))$, but in order to facilitate reading two conventions will be introduced. The first is that outer parentheses will always be omitted and the second is that $\alpha\beta\gamma$ will abbreviate $\alpha(\beta\gamma)$. Thus (st) is written as st, $(e(st))$ as est, and $((e(st))((e(st))(st)))$ as $(est)(est)st$. Association is to the right when parentheses are restored.

This typing scheme gives a hierarchy of functions, not relations, but the predicates and predicates of predicates just considered can be modeled by functions. In fact, there are various ways to do this, depending on the ontology one wishes to adopt. The example that will be worked out here is based on an ontology of possible worlds, but it should be kept in mind that this choice is by no means forced upon us by type-theoretical considerations. Type-theoretical systems also square well with completely different assumptions about the ontological underpinnings of natural language.

A basic idea of possible worlds semantics is that the truth of a declarative sentence should be evaluated relative to a possible world. The meaning of a

CONSTANT	TYPE
walk, talk, . . .	est
man, woman, unicorn, . . .	est
happy, bald, . . .	est
love, hate, . . .	$eest$
every, a, no, the	$(est)(est)st$
john, mary, . . .	$(est)st$
necessarily, possibly	$(st)st$
believe, know, . . .	$(st)est$
not	$(st)st$
and, or	$\alpha\alpha\alpha$

Table 3.1: Some constants and their possible types.

sentence therefore determines an object of type st, a function from worlds to truth values; each choice of a world corresponding to a truth value, namely, the truth value of the sentence at the world. Note that functions of this type are identifiable with the set of those possible worlds for which they return the value *true*, so that an object of type st can alternatively be taken to be a set of possible worlds. Such sets are often called *propositions* and this usage will be followed here.

Let us consider predicates such as **walk**, **man**, or **unicorn**. These can be taken to combine with terms for entities to form sentences (terms denoting propositions) and are therefore of type est. An example: **walk** of type est combines with **m** (for *Mary*) of type e to produce **walk m**, the sentence that Mary is walking, of type st. Functions of type est can also be viewed as relations between entities and possible worlds. If the function denoted by **walk** is applied to an entity, it returns a function that, when applied to a world, returns a truth value. So any combination of an entity and a world returns a truth value and the function denoted by **walk** can be identified with the set of pairs $\langle x, i\rangle$ (with x an entity and i a world) for which the function returns *true*. More generally, any function of a type $\alpha_1\alpha_2 \ldots \alpha_n t$ can be identified with an n-ary relation taking objects of type α_k as its k-th argument.

Other words of English can also be typed. Transitive verbs such as **love** can be modeled as functions of type $eest$, as they return a sentence when fed two arguments. Determiners such as **every**, **some**, **no** and **the** can be taken

to be of the type $(est)(est)st$ we met above, as they combine with two terms of type est, such as **man** and **walk** or **woman** and **talk**, to form a sentence, e.g. **every man walk**. Table 1 gives more words that have been typed according to the principles used here: Proper names combine with a predicate to form a sentence. They can therefore be viewed as predicates of predicates (type $(est)st$). (Clearly, they can also be viewed as individual constants. The two perspectives are equally valid and both will be maintained here.) Sentential modifiers such as **necessarily** return a sentence when applied to a sentence and therefore denote functions that return a proposition when applied to a proposition. Propositional attitude verbs combine with a sentence and an entity (an agent), returning a sentence. Negation goes from propositions to propositions (and from sentences to sentences on the syntactic level). The coordinating elements **and** and **or** can be associated with a whole family of types of the form $\alpha\alpha\alpha$ (where α must 'end in st'). $(st)(st)st$ is of this form and this allows for the conjunction and disjunction of propositions, but the more general typing will also make direct translations of other coordinations possible: *sing and dance* or *some or all*, for example, where in the former *and* has type $(est)(est)est$ and in the latter it has the rather complex type $((est)(est)st)((est)(est)st)(est)(est)st$, the type of functions that take two objects of type $(est)(est)st$ and return a third.

3.2 Terms

Type theory sports two term building operations: *application* and *lambda abstraction*. They are defined as follows.

(Application) If M is a term of type $\alpha\beta$ and N is a term of type α, then (MN) is a term of type β.

(Lambda Abstraction) If M is a term of type β and X is a variable of type α, then $(\lambda X.M)$ is a term of type $\alpha\beta$.

Standard notational conventions say that outer parentheses need not be written, that MN_1N_2 may abbreviate $(MN_1)N_2$, and that $\lambda X\lambda Y.M$, or even just $\lambda XY.M$, is short for $\lambda X.(\lambda Y.M)$. A similar convention holds for longer sequences of lambda binders. We will not always use these conventions here, as we want to emphasize the similarities between the structures of certain lambda terms and linguistic structures. The standard abbreviatory conventions sometimes obscure these similarities.

Let us illustrate the application and abstraction rules by forming some terms. Given the typing in Table 1, the application rule allows the formation of (**every man**) of type $(est)st$. A second use of the rule will give that ((**every man**)**walk**) is of type st. Another example, involving variables: Take the constant **love** of type $eest$, and let x and y be variables of type e. Then ((**love** x)y), or **love** xy for short, is a term of type st by the application rule (the distinction between constants and variables is irrelevant for typing). We will interpret it as expressing that y loves x, so that its structure reflects the structure found in natural language, where a transitive verb combines with its direct object before combining with its subject. The abstraction rule now allows the formation of $(\lambda x.\mathbf{love}\, xy)$ of type est, so that (**a woman**)$(\lambda x.\mathbf{love}\, xy)$ is of type st, from which it follows that $(\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy))$ is of type est again. Reasoning further along these lines, we find that (**every man**)$(\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy))$ is of type st. The latter may be taken to formalize that reading of the English sentence *every man loves a woman* in which the indefinite noun phrase *a woman* is in the scope of *every man*.

The semantic operation corresponding to the rule named application is indeed function application. The denotation of a term (MN) is the result of applying the function denoted by M to the object denoted by N. Lambda abstraction provides a dual. If the type of X is α then $(\lambda X.M)$ denotes a function with the set of all objects of type α as its domain such that, if the function is applied to an object d, the value that is returned is that of M with the variable X interpreted as d. For example, $\lambda x.\mathbf{love}\, x\mathbf{r}$, *being loved by Romeo*, denotes the function that, when applied to Mary, returns the proposition that Romeo loves Mary and, when applied to Juliet, returns the proposition that Romeo loves Juliet.

Certain principles will become valid given these interpretations of the term-building operations. Among these is *renaming of bound variables* (or α-*conversion*), familiar from predicate logic. $\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy)$ and $\lambda z.(\mathbf{a\ woman})(\lambda u.\mathbf{love}\, uz)$, for example, will denote exactly the same function (the function that, when applied to a person a, returns the proposition that a loves a woman). Another rule which becomes valid, known as the β *rule*, is formulated as follows.

(β) $(\lambda X.M)N = M[X := N]$, provided N is substitutable for X in M.

Here $M[X := N]$ should be read as 'the result of substituting N for all free occurrences of X in M.' The requirement that N be substitutable basically

says that variables free in N should remain free in $M[X := N]$. An example: Let **j** be a term of type e, denoting John perhaps. The β rule allows one to conclude that $(\lambda y.(\textbf{a woman})(\lambda x.\textbf{love}\, xy))\textbf{j}$ equals $(\textbf{a woman})(\lambda x.\textbf{love}\, x\textbf{j})$, i.e. the property of loving a woman can be predicated of John if and only if John indeed loves a woman.

Let us spell out in detail how the β rule was applied in this last example. First we deleted the initial λy from $\lambda y.(\textbf{a woman})(\lambda x.\textbf{love}\, xy)$, obtaining the term $(\textbf{a woman})(\lambda x.\textbf{love}\, xy)$, in which y now occurs free. We then replaced this free occurrence of y by **j**, with $(\textbf{a woman})(\lambda x.\textbf{love}\, x\textbf{j})$ as a result. In general, conversion of $(\lambda X.M)N$ proceeds by stripping off the initial binder λX from $\lambda X.M$ and by then substituting N for each free occurrence of X in the resulting M. More examples of the use of the β rule will follow shortly. It plays a pivotal role in type-theoretic semantics.

3.3 Logical Form and Grammatical Form

The following are some terms of type st that can be formed with the help of the type assignment in Table 1 and the term-building rules application and abstraction.

(50) a. $(\textbf{every man})(\lambda y.(\textbf{a woman})(\lambda x.\textbf{love}\, xy))$

b. $(\textbf{a woman})(\lambda x.(\textbf{every man})(\lambda y.\textbf{love}\, xy))$

c. $\textbf{not}((\textbf{the king})\textbf{bald})$

d. $(\textbf{the king})(\lambda x.(\textbf{not}(\textbf{bald}\, x)))$

e. $(\textbf{mary})(\textbf{believe}((\textbf{a unicorn})(\textbf{and walk talk})))$

f. $(\textbf{a unicorn})(\lambda x.((\textbf{mary})(\textbf{believe}((\textbf{and walk talk})x))))$

The terms in (50a) and (50b) are both meant to formalize the English sentence *every man loves a woman*, but while (50a) gives it its universal-existential reading, as discussed above, (50b) says that some woman has the property of being loved by every man, a second possible way to interpret the sentence. (50c) is a formalization of *the king is not bald* in the reading where *the king is bald* is denied to be true, while (50d) ascribes the property of not being bald to the king. The terms (50e) and (50f), lastly, both render

$$
\begin{aligned}
\textbf{every} &= \lambda P' P \lambda i. \forall x (P' x i \rightarrow P x i) \\
\textbf{a} &= \lambda P' P \lambda i. \exists x (P' x i \wedge P x i) \\
\textbf{no} &= \lambda P' P \lambda i. \forall x (P' x i \rightarrow \neg P x i) \\
\textbf{the} &= \lambda P' P \lambda i. \exists x (\forall y (P' y i \leftrightarrow x = y) \wedge P x i) \\
\textbf{john} &= \lambda P. P \mathbf{j} \\
\textbf{mary} &= \lambda P. P \mathbf{m} \\
\textbf{necessarily} &= \lambda p \lambda i. \forall j (\mathbf{R} i j \rightarrow p j) \\
\textbf{possibly} &= \lambda p \lambda i. \exists j (\mathbf{R} i j \wedge p j) \\
\textbf{believe} &= \lambda p \lambda x \lambda i. \forall j (\mathbf{B} x i j \rightarrow p j) \\
\textbf{know} &= \lambda p \lambda x \lambda i. \forall j (\mathbf{K} x i j \rightarrow p j) \\
\textbf{not} &= \lambda p \lambda i. \neg (p i) \\
\textbf{and} &= \lambda R R' \lambda \vec{x} \lambda i. (R \vec{x} i \wedge R' \vec{x} i) \\
\textbf{or} &= \lambda R R' \lambda \vec{x} \lambda i. (R \vec{x} i \vee R' \vec{x} i)
\end{aligned}
$$

Table 3.2: Meaning postulates.

the sentence *Mary believes some unicorn is walking and talking*, but while (50e) gives the *de dicto* reading of this statement, (50f) formalizes the *de re* reading, which can be paraphrased as *there is a unicorn Mary believes to be walking and talking*.

Note how close each of the terms in (50) is to the sentence it represents. (50e), for example, is essentially isomorphic to the hierarchical structure that most linguists would assign to the sentence it renders. And while other terms are not directly akin to any *surface* syntactic structure, they can still be regarded as syntactic structures in which certain noun phrases (such as *a woman* or *the king*) have moved from their original positions. Word order is ignored, though. The view that a level of structures very similar to the ones described here are part of the human grammar is widely shared among syntacticians (May 1985). This level is commonly called that of *Logical Form* or *LF*, but from our perspective it might as well have been called the level of *Grammatical Form*, as it is the proximity to natural language that is striking.

By themselves the forms in (50) are not adequate as logical formalizations of the English sentences they are rendering, as they do not give their correct

truth conditions or adequately capture entailments among them. What is needed is a systematic association of such terms with more standard logical forms and we shall proceed to define one.

We have described Church's simple type theory as a pure lambda calculus, i.e. a lambda calculus without logical constants, and this is a perspective on the theory that is often taken (for example in Hindley 1997), but Church meant his theory to be a formalization of a version of Russell's (1908) theory of types, which is a higher order *logic*. He therefore added logical constants to the theory, and we shall follow him in this by stipulating that $\neg$ is a logical constant of type tt, that $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ are of type ttt, and that logical constants $\exists$ and $\forall$ in any type $(\alpha t)t$ are present, as are symbols $=$ in any type $\alpha\alpha t$. This will give that $\wedge\varphi\psi$ is a term of type t if φ and ψ are, but such terms are more conveniently written as $\varphi \wedge \psi$ and a similar convention will hold for other connectives. We will also write $\exists X\varphi$ instead of $\exists(\lambda X.\varphi)$, $\forall X\varphi$ instead of $\forall(\lambda X.\varphi)$, $M = N$ instead of $=MN$, and conform to logical usage generally. The resulting higher order logic is provided with a model theory in Henkin (1950).

The equations in Table 2 provide the systematic correlation between grammatical forms and truly logical forms we are after. The 'logical' words of Table 1 are each equated with a term interpreting them and there are also postulates that systematically relate proper names viewed as predicates of predicates and proper names viewed as individual constants (e.g. **john** can be predicated of a predicate just if that predicate can be predicated of **j**). A typographic convention will rule the use of variables in the terms employed here: P, with or without primes, is of type est (predicates), p of type st (propositions), i and j are of type s (worlds or indices), and x, y and z are of type e (entities).

Let us see how Table 2 can be used to systematically translate terms such as those in (50) to the kind of logical forms that logicians know and love; (50a) will be used as an example. Table 2 states that $\mathbf{a} = \lambda P'P\lambda i.\exists x(P'xi \wedge Pxi)$, a term that expects two predicates in order to form a proposition with them, and we can conclude that **a woman** equals $(\lambda P'P\lambda i.\exists x(P'xi \wedge Pxi))\mathbf{woman}$, which reduces to $\lambda P\lambda i.\exists x(\mathbf{woman}\, xi \wedge Pxi)$ by the β rule. (First strip off the initial $\lambda P'$ from $\lambda P'P\lambda i.\exists x(P'xi \wedge Pxi)$; then replace the P' in the resulting $\lambda P\lambda i.\exists x(P'xi \wedge Pxi)$, which has now become free, with **woman**.) We now

reason further with the subterm (**a woman**)($\lambda x.$**love** xy) of (50a).

$$\begin{aligned}(\textbf{a woman})(\lambda x.\textbf{love}\,xy) &= (\lambda P\lambda i.\exists x(\textbf{woman}\,xi \wedge Pxi))(\lambda x.\textbf{love}\,xy)\\ &= \lambda i.\exists x(\textbf{woman}\,xi \wedge (\lambda x.\textbf{love}\,xy)xi)\\ &= \lambda i.\exists x(\textbf{woman}\,xi \wedge \textbf{love}\,xyi)\end{aligned}$$

Each of the last two reductions is obtained by applications of the β rule, the last one leading to replacement of the subterm $(\lambda x.\textbf{love}\,xy)x$ by **love** xy. In general, because terms that are related by the β rule can be identified, β reductions can be applied to subterms of any term under consideration.

Further reasoning gives the following equations.

$$\begin{aligned}(50a) &= (\lambda P\lambda i.\forall x(\textbf{man}\,xi \rightarrow Pxi))(\lambda y\lambda i.\exists x(\textbf{woman}\,xi \wedge \textbf{love}\,xyi))\\ &= \lambda i.\forall x(\textbf{man}\,xi \rightarrow (\lambda y\lambda i.\exists x(\textbf{woman}\,xi \wedge \textbf{love}\,xyi))xi)\\ &= \lambda i.\forall x(\textbf{man}\,xi \rightarrow (\lambda y\lambda i.\exists z(\textbf{woman}\,zi \wedge \textbf{love}\,zyi))xi)\\ &= \lambda i.\forall x(\textbf{man}\,xi \rightarrow \exists z(\textbf{woman}\,zi \wedge \textbf{love}\,zxi))\end{aligned}$$

The first of these equations is obtained by observing that **every man** can be replaced with $\lambda P\lambda i.\forall x(\textbf{man}\,xi \rightarrow Pxi)$ on the basis of the entry for **every** in Table 2. The second is a β reduction. The third is an α conversion and the fourth the result of two more β reductions. (Note that the α conversion step was necessary to make the first of these two last β reductions possible. Replacing $(\lambda y\lambda i.\exists x(\textbf{woman}\,xi \wedge \textbf{love}\,xyi))x$ with $(\lambda i.\exists x(\textbf{woman}\,xi \wedge \textbf{love}\,xxi))$ would have been illegal because the free occurrence of x in the first term leads to a bound occurrence in the second.)

We have found that, given the meaning postulates in Table 2, (50a) is identical to (51a), a proposition which is true in the actual world @ if and only if $\forall x(\textbf{man}\,x@ \rightarrow \exists z(\textbf{woman}\,z@ \wedge \textbf{love}\,zx@))$ is true. Very similar considerations lead to the conclusion that each of the other items in (50) equals the corresponding item in (51).

(51) a. $\lambda i.\forall x(\textbf{man}\,xi \rightarrow \exists y(\textbf{woman}\,yi \wedge \textbf{love}\,xyi))$

b. $\lambda i.\exists x(\textbf{woman}\,xi \wedge \forall z(\textbf{man}\,zi \rightarrow \textbf{love}\,zxi))$

c. $\lambda i.\neg\exists x(\forall y(\textbf{king}\,yi \leftrightarrow x = y) \wedge \textbf{bald}\,xi)$

d. $\lambda i.\exists x(\forall y(\textbf{king}\,yi \leftrightarrow x = y) \wedge \neg\textbf{bald}\,xi)$

e. $\lambda i.\forall j(\mathbf{Bm}ij \rightarrow \exists x(\mathbf{unicorn}\,xj \wedge \mathbf{walk}\,xj \wedge \mathbf{talk}\,xj))$

f. $\lambda i.\exists x(\mathbf{unicorn}\,xi \wedge \forall j(\mathbf{Bm}ij \rightarrow (\mathbf{walk}\,xj \wedge \mathbf{talk}\,xj)))$

In (51e) and (51e), read $\mathbf{Bm}ij$ as 'in world i world j is a doxastic alternative of **m**', so that (51e) comes to express that there is a unicorn who walks and talks in each of Mary's doxastic alternatives (Hintikka 1962). A technical detail: The **and** of (50e) is of type $(est)(est)est$, and so will translate as $\lambda PP'\lambda x\lambda i.(Pxi \wedge P'xi)$. The interested reader is invited to check some of the equivalences here and to experiment with some others.

The exposition thus far has been somewhat technical, but the conclusion is not technical at all. It is that the perceived gap between the grammatical form and the logical form of a sentence can be bridged. Philosophers during the first half of the 20th century adhered to the view that grammatical form was misleading and essentially different from logical form. Only the latter was correct. The *locus classicus* of this view is Russell's 'On Denoting' (Russell 1905), but the idea has found many able proponents in addition to Russell. Considerations such as the ones above, however, pioneered in the work of Richard Montague (Montague 1970a; Montague 1970b; Montague 1973) show that it is possible to have one's cake and eat it. The grammatical forms in (50) and the logical forms in (51) may look very different, but given suitable meaning postulates, are extensionally identical. This means that the linguistic and the logical perspectives on form turn out to be compatible and equally valid. It may be thought ironic that the essential ingredient in this vindication of grammatical form and rejection of Russell's 'misleading form thesis' has been a version of Russell's own Theory of Types.

Bibliography

Barwise, J. and J. Etchemendy (2002). *Language, Proof and Logic*. Stanford, CA: CSLI Publications.

Boolos, G. (1984). Trees and Finite Satisfiability: Proof of a Conjecture of Burgess. *Notre Dame Journal of Formal Logic 25*(3), 193–197.

Church, A. (1940). A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic 5*, 56–68.

Fagin, R., J. Halpern, Y. Moses, and M. Vardi (1995). *Reasoning about Knowledge*. Cambridge, MA: MIT Press.

Galton, A. (Spring 2008). Temporal logic. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy*.

Gamut, L. T. F. (1991). *Logic, Language and Meaning, Volume 1*. Chicago: University of Chicago Press.

Henkin, L. (1950). Completeness in the Theory of Types. *Journal of Symbolic Logic 15*, 81–91.

Hindley, R. (1997). *Basic Simple Type Theory*. Cambridge University Press.

Hintikka, J. (1962). *Knowledge and Belief*. Cornell University Press.

Hodges, W. (2001). *Logic*. Penguin Books.

May, R. (1985). *Logical Form: Its Structure and Derivation*. Cambridge, MA: MIT Press.

McNamara, P. (2010). Deontic logic. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2010 ed.).

McTaggart, J. (1908). The Unreality of Time. *Mind 17*, 457–474.

Montague, R. (1970a). English as a Formal Language. In B. Visenti (Ed.), *Linguaggi nella Società e nella Tecnica*, pp. 189–224. Milan: Edizioni di Comunità. Reprinted in (**?**).

Montague, R. (1970b). Universal Grammar. *Theoria 36*, 373–398. Reprinted in (**?**).

Montague, R. (1973). The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J. Moravcsik, and P. Suppes (Eds.), *Approaches to Natural Language*, pp. 221–242. Dordrecht: Reidel. Reprinted in (**?**).

Muskens, R. (2011). Type-logical Semantics. In E. Craig (Ed.), *Routledge Encyclopedia of Philosophy Online*. Routledge.

Prior, A. (1967). *Past, Present and Future*. Oxford: Clarendon Press.

Reynolds, M. (1994). Axiomatisation and Decidability of F and P in Cyclical Time. *Journal of Philosophical Logic 23*, 197–224.

Russell, B. (1905). On Denoting. *Mind 14* (56), 479–493.

Russell, B. (1908). Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics 30*, 222–262.

Tarski, A. (1935). Der Wahrheitsbegriff in den formalisierten Sprachen. *Studia Philosophica 1*, 261–405. (Polish original 1933: Pojęcie prawdy w językach nauk dedukcyjnych; German translation by L. Blaustein).

Tarski, A. (1944). The Semantic Conception of Truth and the Foundations of Semantics. *Philosophy and Phenomenological Research 4*, 341–376.

Tarski, A. (1983). *Logic, Semantics, Metamathematics*. Indianapolis: Hackett. (second edition, ed. by J. Corcoran).