# Introduction to Modal Logics
**(Draft)**

September 16, 2011

# Contents

# 1 From Propositional to Modal Logic

## 1.1 Propositional logic

Let $P$ be a set of *propositional variables*. The language $\mathcal{L}_{\mathrm{PL}}(P)$ has the following list of symbols as *alphabet*: variables from $P$, the logical symbols $\bot$, $\top$, $\neg$, $\rightarrow$, $\wedge$, $\vee$, $\leftrightarrow$, and brackets. The set of $\mathcal{L}_{\mathrm{PL}}(P)$*-formulae*, then, is defined as the smallest set of words over this alphabet that contains $\bot$, $\top$, each $p \in P$, and is closed under the formation rule: If $\varphi$ and $\psi$ are formulae, then so are $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, and $(\varphi \rightarrow \psi)$. We may express this in an abbreviated manner as:

$$\varphi ::= \bot \mid \top \mid p \mid \neg\varphi \mid (\varphi \wedge \varphi') \mid (\varphi \vee \varphi') \mid (\varphi \rightarrow \varphi') \mid (\varphi \leftrightarrow \varphi')$$

— where $p$ ranges over $P$. Propositional variables are also referred to as *atomic formulae*. $\mathcal{L}_{\mathrm{PL}}(P)$ will also denote the set of $\mathcal{L}_{\mathrm{PL}}(P)$-formulae, and for abbreviation we will use the notations $\mathcal{L}_{\mathrm{PL}}$, $\mathcal{L}(P)$, or simply $\mathcal{L}$ if this is understood from the context.

The semantics of propositional logic is defined in terms of truth assignments. A *truth assignment* is simply a function $V : P \rightarrow \{0,1\}$, i.e., $V$ assigns to each atomic formula a truth value 0 ("false") or 1 ("true"). Given a truth assignment $V$, the *satisfaction* relation is then defined as follows:

$$\begin{aligned}
V &\not\models \bot \\
V &\models \top \\
V \models p &\iff V(p) = 1 \\
V \models \neg\varphi &\iff V \not\models \varphi \\
V \models \varphi \wedge \psi &\iff V \models \varphi \text{ and } V \models \psi \\
V \models \varphi \vee \psi &\iff V \models \varphi \text{ or } V \models \psi \\
V \models \varphi \rightarrow \psi &\iff V \not\models \varphi \text{ or } V \models \psi \\
V \models \varphi \leftrightarrow \psi &\iff V \models \varphi \text{ iff } V \models \psi
\end{aligned}$$

**Definition 1.1.** An $\mathcal{L}_{\mathrm{PL}}$-formula $\varphi$ is *satisfiable* if there exists a truth assignment $V$ such that $V \models \varphi$. It is *valid* if for each truth assignment $V$, it holds $V \models \varphi$ and *contingent* if it is neither valid nor unsatisfiable. Two formulae $\varphi$ and $\psi$ are *(logically) equivalent* (or: PL*-equivalent*) if they are satisfied in the same sets of truth assignments. Given a set of $\mathcal{L}_{\mathrm{PL}}$-formulae, $\Sigma$, a formulae $\varphi$, and a truth assignment $V$, we write: $V \models \Sigma$ if $V$ satisfies each $\varphi \in \Sigma$, and $\Sigma \models \varphi$ if for each truth assignment $V$ with $V \models \Sigma$, it holds that $V \models \varphi$. $\varphi$ is then said to be a PL-*consequence* of $\Sigma$.

We remark that most of these concepts also make sense if we restrict them to classes of truth assignments: Let $\mathscr{C}$ be a class of truth assignments (on the same set of propositional variables $P$). Then $\varphi$ is called $\mathscr{C}$*-satisfiable* if $\varphi$ is satisfied in some $V$ chosen from $\mathscr{C}$, $\mathscr{C}$*-valid* if $\varphi$ is satisfied in each $V$ in $\mathscr{C}$, etc.

From these definitions, it is clear that for all formulae $\varphi$ and $\psi$, $(\varphi \wedge \psi)$ is equivalent to $\neg(\varphi \rightarrow \neg\psi)$ and $(\varphi \vee \psi)$ equivalent to $(\neg\varphi \rightarrow \psi)$, etc. That is, one can drop, for example, $\wedge$, $\vee$, $\leftrightarrow$,

$\top$, and $\bot$ from the alphabet and instead use them as abbreviations in the meta-language:

$$\begin{aligned}
(\varphi \wedge \psi) &:= \neg(\varphi \to \neg\psi) \\
(\varphi \vee \psi) &:= (\neg\varphi \to \psi) \\
(\varphi \leftrightarrow \psi) &:= ((\varphi \to \psi) \wedge (\psi \to \varphi)) \\
\top &:= (p_0 \to p_0) \\
\bot &:= \neg\top
\end{aligned}$$

where $p_0$ is a fixed chosen propositional variable in $P$.

A *complete system of truth-functional connectives* is any set of truth-functional connectives in which each truth-functional connective of arbitrary arity is definable. Examples include the systems $\{\neg, \to\}$, $\{\neg, \wedge\}$, $\{\neg, \vee\}$, and $\{\bot, \to\}$. In what follows we will often switch between different such complete systems as appropriate in the respective context. Most of the times, however, we will stick to the system $\{\neg, \to\}$.

For truth assignments $V$ and $V'$, we say that $V$ and $V'$ *coincide* on $Q \subseteq P$ (and write $V =_Q V'$) if for each $q \in Q$, $V(q) = V'(q)$.

**Lemma 1.1** (Coincidence Lemma)**.** *Let $V$ and $V'$ be truth assignments that coincide on $Q \subseteq P$. Then for each formula $\varphi$ in which only propositional variables from $Q$ occur, it holds:*

$$V \models \varphi \iff V' \models \varphi. \qquad \lhd$$

**Definition 1.2.** A *substitution* is a map $\sigma : P \to \mathcal{L}_{\mathrm{PL}}$. Each substitution can be extended to a function $\cdot^\sigma : \mathcal{L}_{\mathrm{PL}} \to \mathcal{L}_{\mathrm{PL}}$ via the following recursive definition:

$$\begin{aligned}
p^\sigma &:= \sigma(p) \\
(\neg\varphi)^\sigma &:= \neg\varphi^\sigma \\
(\varphi \to \psi)^\sigma &:= \varphi^\sigma \to \psi^\sigma \\
&\cdots
\end{aligned}$$

If a substitution $\sigma$ simply “replaces” each occurrence of variable $p$ by a formula $\varphi$, we simply write $\psi[p/\varphi]$ instead of $\psi^\sigma$. Analogously, we will use the notation $\psi[p_1/\varphi_1, \ldots, p_k/\varphi_k]$ to indicate that the variables $p_1, \ldots, p_n$ are *simultaneously replaced* by $\varphi_1, \ldots, \varphi_n$, respectively.

**Lemma 1.2** (Substitution Lemma)**.** *Let $\sigma$ be a substitution and $V$ be a truth assignment. Define $V^\sigma$ by $V^\sigma(p) = 1$ if and only if $V \models \sigma(p)$ for each $p \in P$. Then for each formula $\varphi$,*

$$V^\sigma \models \varphi \iff V \models \varphi^\sigma. \qquad \lhd$$

**Lemma 1.3** (Interpolation Theorem)**.** *For any $\mathcal{L}_{\mathrm{PL}}$-formulae $\varphi$ and $\psi$ with $\varphi \models \psi$, there exists an $\mathcal{L}_{\mathrm{PL}}$-formula $\chi$ such that*

*(a) in $\chi$ only such propositional variables occur that occur in both $\varphi$ and $\psi$, and*

*(b) it holds $\varphi \models \chi$ and $\chi \models \psi$.* $\lhd$

In the situation of the lemma the formula $\chi$ is called an *interpolant* of $\varphi$ and $\psi$.

**Definition 1.3.** The *degree* of an $\mathcal{L}_{\mathrm{PL}}$-formula $\varphi$, $\deg\varphi$, is the number of logical connectives occurring in $\varphi$. The *length* of $\varphi$ is the number of alphabet symbols occurring $\varphi$. Note that both notions depend on the chosen system of truth-functional connectives.

**Remark 1.1.** It is an NP-complete problem (referred to as SAT) to decide for a given $\mathcal{L}_{\mathrm{PL}}$-formula $\varphi$ whether it is satisfiable. The problem is also NP-complete if satisfiability is to be checked for propositional formulae in *conjunctive normal form* (CNF). A formula is in CNF if it is a conjunction of disjunctions of literals (a *literal* is a negated or unnegated atomic formula), i.e., the formula has the form

$$\bigwedge_{i=1}^{n} \bigvee_{j_i}^{m_i} l_{ij_i}, \tag{CNF}$$

where $l_{ij_i}$ is of the form $p$ or $\neg p$ for some propositional variable $p$. Accordingly, the problem to decide whether a given formula is valid is co-NP-complete.

## 1.2 A simple modal logic

In what follows we will introduce a very simple "modal logic". Again let $P$ be a set of atomic propositions (or *propositional variables*). The language $\mathcal{L}_\Box(P)$ has the following list of symbols as alphabet: propositions of $P$, the logical symbols $\neg$, $\rightarrow$, and $\Box$, and brackets. The set of $\mathcal{L}_\Box(P)$*-formulae*, then, is defined as follows:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \rightarrow \varphi') \mid \Box\varphi$$

— where $p$ ranges over $P$. $\mathcal{L}_\Box(P)$ will also denote the set of $\mathcal{L}_\Box(P)$-formulae, and for abbreviation we will use the notations $\mathcal{L}_\Box$, $\mathcal{L}_\Box(P)$, or simply $\mathcal{L}$ if this is understood.

On the basis of the connectives of $\mathcal{L}_\Box(P)$, we can define:

$$\Diamond\varphi := \neg\Box\neg\varphi$$
$$(\varphi \mathrel{\Box\!\!\rightarrow} \psi) := \Box(\varphi \rightarrow \psi)$$

**Definition 1.4.** A *valuation model* is a non-trivial family $V = \{V_s\}_{s\in S}$ of truth assignments of $\mathcal{L}_\Box$ ("non-trivial" means that $S$ is a non-empty set). The elements of $S$ are referred to as *states* (or *possible worlds*). $V$ can be thought of as a function (called *P-valuation* on $S$) $V : S \rightarrow 2^P$, i.e., it assigns to each state $s$ in $S$ a (possibly empty) set of propositional variables (namely those that are true in $V_s$).

Given a valuation model $V$, we define a satisfaction relation as follows:

$$\begin{aligned}
V \models_s p &\iff p \in V(s)\\
V \models_s \neg\varphi &\iff V \not\models_s \varphi\\
V \models_s \varphi \rightarrow \psi &\iff V \not\models_s \varphi \text{ or } V \models_s \psi\\
V \models_s \Box\varphi &\iff V \models_{s'} \varphi, \text{ for each } s' \in S
\end{aligned}$$

By this definition and the definition of $\Diamond$, the truth condition for $\Diamond\varphi$ is then given by:

$$\begin{aligned}
V \models_s \Diamond\varphi &\iff V \models_s \neg\Box\neg\varphi\\
&\iff V \not\models_s \Box\neg\varphi\\
&\iff V \not\models_{s'} \neg\varphi, \text{ for some } s' \in S\\
&\iff V \models_{s'} \varphi, \text{ for some } s' \in S
\end{aligned}$$

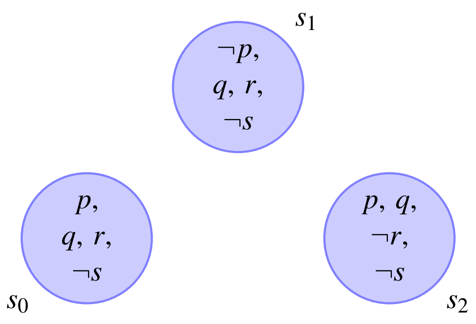


Figure 1.1: An example of a valuation model

**Definition 1.5.** An $\mathcal{L}_\Box$-formula $\varphi$ is called *satisfiable in a valuation model* $V$ if there exists an $s \in S$ such that $V \models_s \varphi$. It is called *valid in* $V$ ($V \models \varphi$) if $V \models_s \varphi$ for each $s \in S$. $\varphi$ is *valuation-satisfiable* if it is satisfiable in some valuation model, and it is *valuation-valid* if it is valid in each valuation model. Given a set of $\mathcal{L}_\Box$-formulae $\Sigma$ and an $\mathcal{L}_\Box$-formula $\varphi$, we say that $\Sigma$ *(locally) entails* $\varphi$ if for each valuation model $V = (V_s)_{s\in S}$ and each state $s \in S$ with $V \models_s \Sigma$, it holds that $V \models_s \varphi$. $\Sigma$ *globally entails* $\varphi$ if for each valuation model $V$ with $V \models \Sigma$, it holds $V \models \varphi$.

An example of a valuation model is graphically depicted in Figure 1.1. The model itself can be formally defined by $S := \{s_o, s_1, s_2\}$, $V(s_0) := \{p,q,r\}$, $V(s_1) := \{q,r\}$, and $V(s_2) := \{p,q\}$. In this model it holds, for example $V \models \Box q$, $V \models \neg s$, $V \models \Diamond\neg r$, $V \models \Box(\neg p \to r)$. The formula $s$ is unsatisfiable in this model, and both $r$ and $\neg r$ are satisfiable in the model.

The logic characterized by valuation models (i.e., the set of valid formulae wrt. the valuation semantics) is rather special. It not only allows for reducing *iterated* modalities, but also allows for reducing *nested* occurrences of modalities. A measure for nestedness is the so-called *modal depth* of a formula, $\operatorname{depth}\varphi$, which counts the depth of nested occurrences of the modal operators:

$$\begin{aligned}
\operatorname{depth}(p) &= 0\\
\operatorname{depth}(\neg\varphi) &= \operatorname{depth}(\varphi)\\
\operatorname{depth}(\varphi \to \psi) &= \max(\operatorname{depth}(\varphi), \operatorname{depth}(\psi))\\
\operatorname{depth}(\Box\varphi) &= \operatorname{depth}(\varphi) + 1
\end{aligned}$$

In what follows we want to prove that each $\mathcal{L}_\Box$-formula $\varphi$ is equivalent (with respect to valuation models) to an $\mathcal{L}_\Box$-formula $\psi$ with $\operatorname{depth}\psi \leq 1$. In order to prove this claim, we will need “reduction principles” that allow for reducing the modal depth of formulae. For the sake of simplicity we will work in a setting in which $\wedge$, $\vee$, and $\Diamond$ are present in the object language. Note that the modal depth of a formula does not depend on the chosen set of connectives.

**Lemma 1.4.** *The following $\mathcal{L}_\Box$-formulae are valuation-valid:*

(R0) $\neg\Box\varphi \leftrightarrow \Diamond\neg\varphi$

(R0*) $\neg\Diamond\varphi \leftrightarrow \Box\neg\varphi$

(R1) $\Box\Box\varphi \leftrightarrow \Box\varphi$

(R1*) $\Diamond\Diamond\varphi \leftrightarrow \Diamond\varphi$

(R2) $\Diamond\Box\varphi \leftrightarrow \Box\varphi$

(R2*) $\Box\Diamond\varphi \leftrightarrow \Diamond\varphi$

(R3) $\Box(\varphi \wedge \psi) \leftrightarrow (\Box\varphi \wedge \Box\psi)$

(R3*) $\Diamond(\varphi \vee \psi) \leftrightarrow (\Diamond\varphi \vee \Diamond\psi)$

(R4) $\Box(\varphi \vee \Box\psi) \leftrightarrow (\Box\varphi \vee \Box\psi)$

(R4*) $\Diamond(\varphi \wedge \Diamond\psi) \leftrightarrow (\Diamond\varphi \wedge \Diamond\psi)$

(R5) $\Box(\varphi \vee \Diamond\psi) \leftrightarrow (\Box\varphi \vee \Diamond\psi)$

(R5*) $\Diamond(\varphi \wedge \Box\psi) \leftrightarrow (\Diamond\varphi \wedge \Box\psi)$ ◁

**Theorem 1.5.** *Each $\mathcal{L}_\Box$-formula $\varphi$ is equivalent (with respect to valuation models) to an $\mathcal{L}_\Box$-formula $\psi$ with* $\operatorname{depth}\psi \leq 1$.

*Proof.* For the proof let $\varphi$ be a formula with $\operatorname{depth}\varphi \geq 2$. We present a procedure that syntactically transforms $\varphi$ into an equivalent formula with modal depth at most 1.

1. Rewrite $\varphi$ in such a way that only $\neg$, $\Box$, $\Diamond$, $\wedge$, and $\vee$ occur in it, i.e., we replace in $\varphi$ each subformula of the form $(\psi \to \psi')$ by $(\neg\psi \vee \psi')$ and each subformula of the form $(\psi \leftrightarrow \psi')$ by $((\neg\psi \vee \psi') \wedge (\psi \vee \neg\psi'))$.
2. Rewrite $\varphi$ such that the negation symbol only occurs in front of propositional variables. For this, until $\varphi$ can not be further modified,

   *a)* apply the *de Morgan laws*, i.e., replace subformulae of $\varphi$ of the form $\neg(\psi \wedge \psi')$ by $(\neg\psi \vee \neg\psi')$ and subformulae of the form $\neg(\psi \vee \psi')$ by $(\neg\psi \wedge \neg\psi')$;

   *b)* apply rules (R0) and (R0*): replace subformulae of the form $\neg\Box\psi$ by $\Diamond\neg\psi$ and subformulae of the form $\neg\Diamond\psi$ by $\Box\neg\psi$;

   *c)* absorb iterated negations ($\neg\neg\varphi \leftrightarrow \varphi$);

   *d)* absorb iterated modalities where possible (by applying rules (R1), (R1*), (R2), and (R2*)).
3. Distribute modalities in front of disjunctions and conjunctions, where possible. For this, as long as the depth of $\varphi$ is greater than 1,

   *a)* apply rules (R3) and (R3*) on suitable subformulae of $\varphi$;

   *b)* apply rules (R4), (R4*), (R5), and (R5*) on suitable subformulae of $\varphi$ (this reduces the modal depth);

   *c)* absorb iterated modalities where possible (by applying rules (R1), (R1*), (R2), and (R2*));

   *d)* apply the PL commutativity, associativity, or distributivity laws if necessary.

In fact, if after step (2.) $\varphi$ has modal depth greater than 2, then $\varphi$ has a subformula $\psi$ of the form $\Box\psi'$ or $\Diamond\psi'$, where $\operatorname{depth}\psi' = \operatorname{depth}\varphi - 1$ and $\psi'$ is a conjunction or a disjunction of formulae.

*Case 1:* Assume $\psi = \Box\psi'$.

*Case 1.1:* If $\psi'$ is a conjunction of formulae $\psi'_1, \psi'_2$, then by (R3) $\Box\psi'$ is replaced by $(\Box\psi'_1 \wedge \Box\psi'_2)$.

*Case 1.2:* If $\psi'$ is a disjunction of formulae $\psi'_1$ and $\psi'_2$ and one of these has the form $\Box\tau$ or $\Diamond\tau$, then we can apply (R4) or (R5), thus reducing the depth of $\psi'$.

*Case 1.3:* If $\psi'$ is a disjunction $\psi'_1 \vee \psi'_2$, but neither of the formulae $\psi'_1, \psi'_2$ has the form $\Box\tau$ or $\Diamond\tau$, then one of it must be a disjunction or a conjunction of formulae. In the first case apply the associative law and in the second case apply the distributive law to the formula $\psi'$. Note that after applying the distributivity law we obtain a formula that can be handled as in Case 1.1.

*Case 2:* $\psi = \Diamond\psi'$. The procedure is analogous to that of case 1.

Since all the transformations applied in this procedure result in equivalent formulae, the resulting formula is equivalent to the input formula. $\lhd$

**Example 1.1.** Consider the formula $\neg\Diamond(p \wedge (q \vee \Diamond r))$. By applying the procedure presented in the proof, we obtain:

(1) $\neg\Diamond(p \wedge (q \vee \Diamond r))$
(2) $\Box\neg(p \wedge (q \vee \Diamond r))$
(3) $\Box(\neg p \vee \neg(q \vee \Diamond r))$
(4) $\Box(\neg p \vee (\neg q \wedge \neg\Diamond r))$
(5) $\Box(\neg p \vee (\neg q \wedge \Box\neg r))$
(6) $\Box((\neg p \vee \neg q) \wedge (\neg p \vee \Box\neg r))$
(7) $\Box(\neg p \vee \neg q) \wedge \Box(\neg p \vee \Box\neg r)$
(8) $\Box(\neg p \vee \neg q) \wedge (\Box\neg p \vee \Box\neg r)$

**Definition 1.6.** Let $\varphi$ be an $\mathcal{L}_\Box$-formula.

(a) $\varphi$ is said to have *modal conjunctive normal form (MCNF)* if $\varphi$ has the form

$$\bigwedge_{i=1}^{n} \bigvee_{j_i=1}^{m_i} \alpha_{ij_i}, \tag{CNF}$$

where each "atom" $\alpha_{ij_i}$ is an $\mathcal{L}_{\mathrm{PL}}$-formula or a formula of the forms $\Box\psi'$ or $\Diamond\psi'$ with $\operatorname{depth}(\psi') = 0$.

(b) $\varphi$ is said to have *ordered MCNF* if $\varphi$ has MCNF and each conjunct $\bigvee_{j_i} \alpha_{ij_i}$ has the form

$$\beta \vee \Box\gamma_1 \vee \cdots \vee \Box\gamma_n \vee \Diamond\delta. \tag{$*$}$$

(Note that we also allow degenerated cases of $(*)$, where one or more of the disjuncts in the form are not present).

Obviously, each formula in MCNF has modal depth $\leq 1$. Vice versa, by extending/modifying the procedure presented in the proof of Theorem 1.5, one obtains:

**Theorem 1.6.** *Each $\mathcal{L}_\Box$-formula $\varphi$ is valuation-equivalent to an $\mathcal{L}_\Box$-formula $\psi$ with (ordered) MCNF.*

*Proof.* By the procedure presented in the proof of Theorem 1.5, we obtain a formula with modal depth $\leq 1$ that is a disjunction of conjunctions (or a conjunction of disjunctions) of formulae that are of modal depth 0 or have the form $\Box\gamma$ or $\Diamond\gamma$, where $\gamma$ is a formula with modal depth 0. In the first case we apply the distributive and (if necessary) the associative laws to obtain a formula in MCNF. Obviously, if a formula $\varphi$ has MCNF, it can easily be transformed into an equivalent formula with ordered MCNF by applying the theorem $(\Diamond\delta_1 \vee \Diamond\delta_2) \leftrightarrow \Diamond(\delta_1 \vee \delta_2)$. $\lhd$

**Example 1.2.** As an example consider the formula $\Box(\Diamond p \to p) \to \Box(p \to \Box p)$. We first reduce this formula to a formula with modal depth 1:

(1) $\neg\Box(\neg\Diamond p \vee p) \vee \Box(\neg p \vee \Box p)$
(2) $\Diamond(\Diamond p \wedge \neg p) \vee \Box(\neg p \vee \Box p)$
(3) $(\Diamond p \wedge \Diamond\neg p) \vee (\Box\neg p \vee \Box p)$

From this one obtains MCNF by the distributivity laws:

(4) $(\Diamond p \vee \Box\neg p \vee \Box p) \wedge (\Diamond\neg p \vee \Box\neg p \vee \Box p)$

By simply reordering (4) we get a formula in ordered MCNF

(5) $(\Box\neg p \vee \Box p \vee \Diamond p) \wedge (\Box\neg p \vee \Box p \vee \Diamond\neg p)$

Using this result, we can now introduce a test for deciding whether a given formula $\varphi$ is valuation-valid. Thereto, we can assume that $\varphi$ already has ordered MCNF. Then, by definition, $\varphi$ is a conjunction of formulae that have the form

$$\beta \vee \Box\gamma_1 \vee \cdots \vee \Box\gamma_n \vee \Diamond\delta, \tag{$*$}$$

where all formulae $\beta, \gamma_1, \ldots, \gamma_n, \delta$ have modal depth 0. Consider then the disjunctions

$$\beta \vee \delta,\ \gamma_1 \vee \delta,\ \ldots,\ \gamma_n \vee \delta, \tag{$**$}$$

which are obviously formulae of PL. A formula of form $(*)$ is said to *pass the disjunction test* if at least one of the disjunctions in $(**)$ is valid in PL and a conjunction of such formulae is said to *pass the disjunction test* if each of its conjuncts passes it.

In the example presented above, formula (5) has two conjuncts and we have to test both of them. In the disjunction test for the first conjunct, $\Box\neg p \vee \Box p \vee \Diamond p$, we have to test whether one of the disjunctions

(6) $\neg p \vee p,\ p \vee p$

is PL-valid, which of course is the case. For the second conjunct, $\Box\neg p \vee \Box p \vee \Diamond\neg p$, the disjunction test gives

(7) $\neg p \vee \neg p,\ p \vee \neg p$

and here the second formula is obviously PL-valid. Thus, both conjuncts pass the disjunction test, and hence so does (5).

**Lemma 1.7.** *Let $\varphi$ be an $\mathcal{L}_\Box$-formula of the form $(*)$. Then $\varphi$ passes the disjunction test if and only if $\varphi$ is valid in each valuation model.*

*Proof.* Let $\varphi$ be a formula of the form $(*)$ that passes the disjunction test. Then at least one of the disjunctions in $(**)$ is PL-valid. We have to show that $\varphi$ is valid in each valuation model. For proof by contradiction, assume that this is not the case. Then there exists a valuation model $\{V_s\}_s$ and a state $s_0$ such that $V \not\models_{s_0} \varphi$. This means $V \not\models_{s_0} \beta$, $V \not\models_{s_0} \Diamond\delta$, and for each $1 \leq i \leq n$, $V \not\models_{s_0} \Box\gamma_i$. Thus $V \not\models_s \delta$ for each $s \in S$. $V \not\models_{s_0} \beta \vee \delta$. Moreover, for each $i$ there is a state $s_i$ such that $V \not\models \gamma_i \vee \delta$ — in contradiction to the assumption that $\beta \vee \delta$ or one of the $\beta \vee \gamma_i$ is PL-valid.

For the other direction, let $\varphi$ be a formula of the form $(*)$ that does not pass the disjunction test. This means that none of the disjuncts $\beta \vee \delta$, $\gamma_1 \vee \delta$, $\ldots$, $\gamma_n \vee \delta$ is PL-valid. We have to show that $\varphi$ is not valid in the sense of the valuation semantics (that is, it needs to be shown that there exists a valuation model and a state in that model that falsifies $\varphi$). Since each of these disjunctions is not valid in PL, there exist truth assignments $V_0, \ldots, V_n$ such that $V_0(\beta \vee \delta) = 0$ and $V_i(\gamma_i \vee \delta) = 0$, for each $1 \leq i \leq n$. Consider the valuation model $V = \{V_i\}_{i \in \{0,\ldots,n\}}$. Then for each $0 \leq i \leq n$ and each formula $\psi$ with depth $\psi = 0$,

$$V \models_i \psi \iff V_i \models \psi,$$

i.e., $V \not\models_0 \beta$, $V \not\models_i \gamma_i$ for each $1 \leq i \leq n$, and $V \not\models_i \delta$ for each $0 \leq i \leq n$. Thus, $V \not\models_0 \varphi$. $\lhd$

Since a conjunction of formulae is valuation-valid if and only if each of the conjuncts is so, we obtain:

**Theorem 1.8.** *An $\mathcal{L}_\Box$-formula is valuation-valid if and only if it is valuation-equivalent to a formula in ordered MCNF that passes the disjunction test.* $\lhd$

In fact, what we have done is the following: we have "reduced" the problem of testing the validity of $\mathcal{L}_\Box$-formulae to the problem of testing a suitable set of $\mathcal{L}_{\mathrm{PL}}$-formulae for validity. Since transforming a formula into ordered MCNF can be done by an algorithm (note that by applying the distributive laws, the length of the formula may increase exponentially) and since testing PL-validity is decidable, testing $\mathcal{L}_\Box$-formulae for validity is a decidable decision problem.

**Theorem 1.9.** *The problem of testing a $\mathcal{L}_\Box$-formula for validity (in terms of valuation semantics) is decidable.* $\lhd$

## Bibliographic remarks

There are a series of text books that introduce into classic propositional and classical first-order logics (Ebbinghaus et al., 2007; Schöning, 2000; Kutschera and Breitkopf, 2007). There are also quite good introductions to modal logics. To list some of them (some are somewhat advanced): Hughes and Cresswell (1996) – a classical, comprehensive book on modal logics that supersedes two previous books (Hughes and Cresswell, 1985, 1990) –, Chellas (1980), Chagrov and Zakharyaschev (1997), Kutschera (1976), Kracht (1999), Gabbay et al. (2003) (a very good introduction to multi-dimensional modal logics) and Blackburn et al. (2002), which is in most parts the basis of these lecture notes.

In this chapter, section 1.2 (specifically the parts on modal conjunctive normal forms and the decision procedure based on the disjunction test) draws much from Hughes and Cresswell (1996), chapter 5.

# 2 Modal Language, Frames, and Models

In this section we describe basic modal-logical concepts in a rather general setting. We introduce multi-modal languages with modal operators of arbitrary arity and introduce the relational semantics for these languages. Moreover we discuss different concrete examples of such modal logics, namely temporal logic, epistemic logic, and dynamic logic. However, we start by defining relational structures and discussing different types of such structures that will be topic in later sections.

## 2.1 Relational structures

A relation on a set $S$ is any subset $R \subseteq S^n$ for a fixed natural number $n$ (the *arity* of $R$). Since relations are sets, we can apply set-theoretical operations on them, for example, if $R$ is an $n$-ary relation, then its complement $S^n \setminus R$ is an $n$-ary relation as well. And if $R_1$ and $R_2$ are such relation, then so are $R_1 \cap R_2$ and $R_1 \cup R_2$. If $\sigma$ is a permutation of $\{1,\ldots,n\}$, then $R^\sigma := \{(s_1,\ldots,s_n) \in S^n : (s_{\sigma^{-1}(1)},\ldots,s_{\sigma^{-1}(n)}) \in R\}$ is an $n$-ary relation. Instead of $s = (s_1,\ldots,s_n) \in R$, we often simply write $R\,s$ or $R\,s_1 \ldots s_n$. If $R$ is a binary relation, we will use infix notation $s_1\,R\,s_2$ instead of writing $R\,s_1 s_2$.

Often we will be interested in binary relations (also called *dyadic* relations) and their properties.

**Definition 2.1.** A binary relation $R$ on $S$ is called

(a) *reflexive* if $s\,R\,s$ for each $s \in S$;

(b) *irreflexive* if $s\,R\,s$ for no $s \in S$;

(c) *serial* if for each $s \in S$ there is an $s' \in S$ such that $s\,R\,s'$;

(d) *symmetric* if $s_2\,R\,s_1$ whenever $s_1\,R\,s_2$;

(e) *asymmetric* if there are no $s_1, s_2 \in S$ such that $s_1\,R\,s_2$ and $s_2\,R\,s_1$;

(f) *antisymmetric* if for all $s_1, s_2 \in S$ with $s_1\,R\,s_2$ and $s_2\,R\,s_1$, $s_1 = s_2$;

(g) *transitive* if for all $s_1, s_2, s_3 \in S$ with $s_1\,R\,s_2$ and $s_2\,R\,s_3$, $s_1\,R\,s_3$;

(h) *Euclidean* if for all $s_1, s_2, s_3 \in S$ with $s_1\,R\,s_2$ and $s_1\,R\,s_3$, $s_2\,R\,s_3$;

(i) *universal* for all $s_1, s_2 \in S$, $s_1\,R\,s_2$.

(j) *connected* if for all $s_1, s_2 \in S$, $s_1\,R\,s_2$ or $s_2\,R\,s_1$;

(k) *weakly connected* if for all $s_1, s_2, s_3 \in S$ with $s_1\,R\,s_2$ and $s_1\,R\,s_3$, $s_2\,R\,s_3$ or $s_3\,R\,s_2$;

(l) *well-founded* if there is no infinite chain $\ldots s_2\,R\,s_1\,R\,s_0$ in $S$ (see Definition 2.2);

(m) *functional* if for all $s_1, s_2, s_3 \in S$ with $s_1\,R\,s_2$ and $s_1\,R\,s_3$, $s_2 = s_3$.

(n) *convergent* if for all $s_1, s_2, s_3 \in S$ with $s_1\,R\,s_2$ and $s_1\,R\,s_3$, there exists an $s_4 \in S$ with $s_2\,R\,s_4$ and $s_3\,R\,s_4$.

**Lemma 2.1.**

(a) *If $R$ is reflexive and Euclidean, then it is symmetric and transitive.*

(b) *If $R$ is symmetric and transitive, then it is Euclidean.*

(c) *If $R$ is serial, symmetric, transitive, then it is reflexive.*

(d) *A transitive relation $R$ on a finite set $S$ is well-founded if and only if it is irreflexive.*

**Definition 2.2.** The *converse* of a binary relation $R$ on $S$ is the relation $R^{-1} := \{ (s_1,s_2) \in S^2 : (s_2,s_1) \in R\}$. The *composition* of binary relations $R_1$ and $R_2$ on $S$ is defined as:

$$R_1 \circ R_2 := \{ (s_1,s_2) \in S^2 : \text{for some } s' \in S,\, s_1 \, R_1 \, s' \text{ and } s' \, R_2 \, s_2\}.$$

For $R$ a binary relation, we set

$$R^0 := \Delta_S := \{ (s,s') \in S^2 : s = s'\}$$
$$R^m := R^{m-1} \circ R$$

(note that $\circ$ is associative). An $R$-chain in $S$ is a (finite or infinite) sequence $s_0,\dots,s_n,\dots$ such that for each $i \geq 0$, $s_i \, R \, s_{i+1}$. Moreover, $R[s] := \{ s' \in S : s \, R \, s'\}$ denotes the set of *$R$-successors* of $s$. Given a relation $R$ on $S$, $R^r$ denotes the smallest superset of $R$ that is reflexive (the *reflexive closure* of $R$), $R^+$ the smallest superset of $R$ that is transitive (*transitive closure*), and $R^*$ the smallest superset of $R$ that is reflexive and transitive (*reflexive transitive closure*). That is,

$$R^r := \bigcap\{ R' \subseteq S^2 : R' \text{ is reflexive and } R \subseteq R'\}$$
$$R^+ := \bigcap\{ R' \subseteq S^2 : R' \text{ is transitive and } R \subseteq R'\}$$
$$R^* := \bigcap\{ R' \subseteq S^2 : R' \text{ is reflexive and transitive, and } R \subseteq R'\}.$$

**Definition 2.3** (Relational structure)**.** A *relational structure* is an ordered pair $\mathscr{R} = \langle S, \{R_i\}_{i\in I}\rangle$, where $S$ is a non-empty set and $\{R_i\}_{i\in I}$ is a family of relations on $S$.

**Definition 2.4** (Homomorphism)**.** Let $\mathscr{R} = \langle S, \{R_i\}_{i\in I}\rangle$ and $\mathscr{R}' = \langle S', R'_{i\in I}\rangle$ be relational structures such that for each $i \in I$, $R_i$ and $R'_i$ are relations of the same arity. A *homomorphism* from $\mathscr{R}$ to $\mathscr{R}'$ (written as $f : \mathscr{R} \to \mathscr{R}'$) is a function $f : S \to S'$ such that for each $i \in I$,

$$R_i \, s_1 \dots s_{r_i} \Rightarrow R'_i \, f(s_1) \dots f(s_{r_i}).$$

An *embedding* of $\mathscr{R}$ into $\mathscr{R}'$ is an injective function $f : S \to S'$ such that the stronger condition

$$R_i \, s_1 \dots s_{r_i} \iff R'_i \, f(s_1) \dots f(s_{r_i})$$

holds. An *isomorphism* is an bijective embedding. An *endomorphism* is a homomorphism $f : \mathscr{R} \to \mathscr{R}$ and an *automorphism* is an isomorphism $f : \mathscr{R} \to \mathscr{R}$.

In what follows we provide an overview over different types of relational structures that will occur frequently in modal logics.

**Example 2.1** (Orders)**.** Natural examples of ordering relations are the natural orders defined on the natural, the rational, and the real numbers. More generally, a *partial order* is an ordered pair $\langle X, \leq\rangle$ where the ordering relation $\leq$ is reflexive, transitive, and antisymmetric. A *linear order* is a partial order where the ordering relation $\leq$ is connected (that is, all $x,y \in X$ are *comparable*; in order theory the connectedness condition is also referred to as *totality* condition, so total partial order and linear order means the same). A *strict partial order* is an ordered pair $\langle X, <\rangle$ in which the ordering relation $<$ is irreflexive and transitive. Each strict partial order defines a partial order if one sets $x \leq y := x < y \vee x = y$. Conversely, each partial order defines a strict partial order by $x < y := x \leq y \wedge x \neq y$. In general, $x > y$ and $x \geq y$ denote

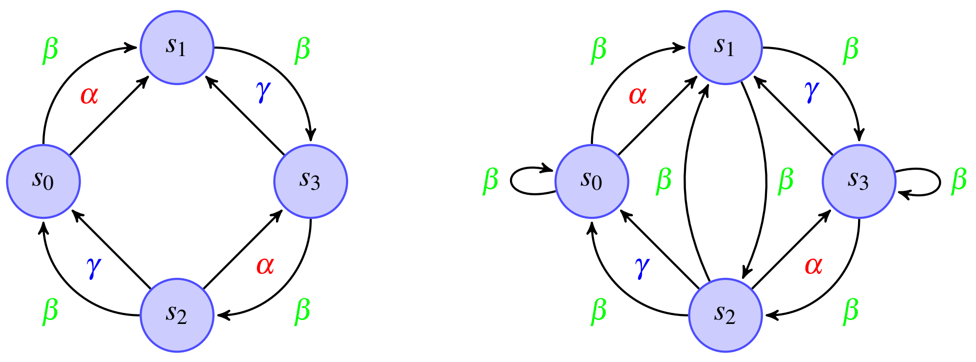


Figure 2.1: Transition systems

the converse relations of $x < y$ and $x \leq y$, respectively. If for all $x, y \in X$, either $x < y$, $x = y$, or $y < x$ holds (*trichotomy condition*), a strict partial order is said to be a *strict linear order*. There are even weaker ordering concepts: In a *preoder* the ordering is just assumed to be reflexive and transitive. Thus partial orders are those preorders in which the ordering relation is antisymmetric. A *weak order* is a total preorder and a *strict weak order* is a strict partial order in which the incomparability relation is transitive.

**Example 2.2** (Equivalence relation)**.** An *equivalence relation* on $S$ is a binary relation $\sim$ on $S$ that is reflexive, symmetric, and transitive. For each $s \in S$, $[s]_\sim := \sim[s]$ is called the *equivalence class* of $s$ (and $s$ is called a *representative* of that class). The set of all equivalence classes wrt. $\sim$ is denoted by $S/\sim$ and forms a partition of $S$, i.e., $S = \bigcup_{X \in S/\sim} X$ and for all $X, Y \in S/\sim$, $X \cap Y = \emptyset$ or $X = Y$.

**Example 2.3** (Tree)**.** In graph theory, a *tree* is an undirected simple graph $G = \langle V, E \rangle$ that is connected and acyclic. A *rooted tree* is a tree with one node $r$ designated as the *root* of the tree. In a rooted tree all (undirected) edges can be conceived of as *arcs* (i.e., directed edges); the root defines a direction on the edges. Hence, each rooted tree can be cast as a relational structure $\langle V, A \rangle$ consisting of a set of *nodes*, $V$, and a set of arcs, $A$, on $V$ ($(v, v') \in A$ is read as "$v'$ is a *child* of $v$") such that:

1. $V$ contains a unique element $v_0$ (the *root* of the tree) such that for each $v \in V \setminus \{v_0\}$, there exists a directed path from $v_0$ to $v$ (or equivalently $v_0 \, A^+ \, v$).
2. Each node except $v_0$ has a unique *parent*, i.e., for each $v \in S \setminus \{v_0\}$, there exists exactly one $v' \in V$ with $v' \, A \, v$.
3. $A^+$ is irreflexive.

**Example 2.4** (Labeled transition system)**.** In automata theory a *labeled transition system* is an ordered triple $\langle S, L, T \rangle$, where $S$ is a non-empty set of *states*, $L$ is a set of *labels*, and $T \subseteq S \times L \times S$ is a ternary relation (its elements are referred to as *labeled transitions*). The transition relation is often written as an arrow: for example, $s \xrightarrow{\alpha} s'$ instead of $(s, \alpha, s') \in T$. A transition system is called *deterministic* if for all $s \xrightarrow{\alpha} s'$ and $s \xrightarrow{\alpha} s''$, it holds $s' = s''$. Otherwise, the transition system is called *non-deterministic* (the examples in Figure 2.1 depict a deterministic and a non-deterministic transition system, respectively). It is clear that labeled transition systems can be cast as relational structures $\langle S, \{T_\alpha\}_{\alpha \in L} \rangle$ where $T_\alpha := \{ (s, s') \in S^2 : (s, \alpha, s') \in T \}$.

## 2.2 Modal languages

**Definition 2.5.** Let $P$ be a set of *propositional variables* and let $\tau = \{\Diamond_i\}_{i\in I}$ be a family of "diamond" symbols (its elements are referred to as *modal operators*). The *basic modal language* (with modal operators from $\tau$), $\mathcal{L}_\tau(P)$, has the following formulae:

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid (\varphi_1 \vee \varphi_2) \mid \Diamond_i\varphi.$$

More generally, a *(modal) similarity type* is a family $\tau = \{(\triangle_i, r_i)\}_{i\in I}$ where each "triangle" $\triangle_i$ is equipped with an arity $r_i \in \mathbb{N}$. The set of $\mathcal{L}_\tau(P)$-formulae is then defined by:

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid (\varphi_1 \vee \varphi_2) \mid \triangle_i(\varphi_1, \dots, \varphi_{r_i})$$

The *dual* of $\triangle_i$ is defined as:

$$\triangledown_i(\varphi_1, \dots, \varphi_{r_i}) := \neg \triangle_i (\neg\varphi_1, \dots, \neg\varphi_{r_i})$$

and often referred to as *nabla*. In the basic modal logic, "nablas" correspond to boxes. In many contexts it is usual to write $\langle i\rangle$ instead of $\Diamond_i$ and accordingly $[i]$ instead of $\Box_i$. 0-ary triangles are called *modal constants*. To indicate nullary or unary model operators, we will use the symbol $\bigcirc$ and $\Diamond$, respectively.

**Example 2.5** (Doxastic/epistemic logic)**.** In doxastic logic $\Diamond\varphi$ can be read as "it is consistent with the agent's belief state that $\varphi$." Accordingly, the reading of $\Box\varphi$ (often written as $\mathsf{B}\varphi$) is: "the agent believes that $\varphi$ is true." Similarly, in epistemic logic $\Diamond\varphi$ and $\Box\varphi$ (or $\mathsf{K}\varphi$) can be read as "it is consistent with what the agent knows that $\varphi$" and "the agent knows that $\varphi$ is true", respectively. There are also logics that combine knowledge and belief. The formula $\mathsf{K}\varphi \leftrightarrow \mathsf{B}\varphi \wedge \varphi$, for example, expresses that knowledge is just true belief.
There are also multi-agent doxastic/epistemic logics. Given a set of agents $A$, one considers a modal language with one modality operator for each agent in $A$ or even languages with a modality operator for each group (set of agents) in $A$. Possible readings of $\mathsf{K}_\gamma\varphi$ include "each agent $a \in \gamma$ knows $\varphi$" or "it is common knowledge in the group $\gamma$ that $\varphi$ is true."
*Autoepistemic logic* is a special epistemic logic to represent, and reason about, "knowledge about knowledge". Contrary to the usual epistemic logic, autoepistemic logic is based on a semantics that generalizes the stable model semantics used in the context of logic programming with negation as failure. That is, the semantics of autoepistemic logic builds on a concept of extension similar to that considered in default logic to characterize the different belief states of an agent that are justified by a default theory (see lecture on Knowledge Representation and Reasoning).

**Example 2.6** (Temporal logic)**.** Basic temporal logic has two modal operators, one to look into the future and one to look into the past, that is, the modal language of basic temporal logic has the "diamonds" $\mathsf{F}$ ("it will be the case") and $\mathsf{P}$ (read as: "it has been the case"). The box operators corresponding to $\mathsf{F}$ and $\mathsf{P}$ are $\mathsf{G}$ ("it will always be the case") and $\mathsf{H}$ (read as: "it has always been the case that"), respectively. In philosophical logic these operators are usually used in the context of strict linear orders (*flows of time*).
In computer science (in particular in *model checking*) future-directed languages are more prominent. For example, *Linear temporal logic* (LTL) has been introduced by Amir Pnueli in order to formally verify properties of computer programs. LTL has two unary modal operators $\mathsf{X}$ (read: "next") and $\mathsf{F}$ (read: "in the future") and a binary modal operator $\mathsf{U}$ (read: "until"). In

the context of verification even richer languages are considered. For example, in *Computation tree logic* (CTL) one considers tree-like flows of time in which a state may have many possible futures. Accordingly, CTL has a series of modal connectives: $\mathsf{AX}$, $\mathsf{EX}$, $\mathsf{AF}$, $\mathsf{EF}$, $\mathsf{AG}$, $\mathsf{EG}$, $\mathsf{AU}$, $\mathsf{EU}$. Here, $\mathsf{A}$ means "in all future paths" and $\mathsf{E}$ means "in some future path", while $\mathsf{X}$, $\mathsf{F}$, $\mathsf{U}$ preserve their meaning (with respect to the considered future path, which builds given the fixed past of the current state a linear flow of time). For example, the formula $\mathsf{AF}\,\mathsf{EG}\,\varphi$ means: "it is inevitable that there is some future state such that $\varphi$ will always be true in a possible future of that state." And $\mathsf{EG}\,\mathsf{AF}\,\varphi$ means: "the current state has a future in which it is inevitable that $\varphi$ will be true at some future point in time."

**Example 2.7** (Provability logic)**.** Consider an arbitrary formal system $S$, i.e., a set of formulae and a consequence relation on this set. For typical formal systems the consequence relation $\vdash$ will be a reflexive and transitive relation. This gives rise to a modal logic with one modality $\Diamond$, where its dual $\Box$ is read as: "In $S$ it is provable that".
There are two cases where such an interpretation is interesting. First, modal logic allows for an interpretation of formal systems such as intuitionistic propositional logic (which strengthen the notion of truth to the notion of provability).
The other, even more interesting use case is if the considered formal systems are rather expressive. In Peano arithmetics (PA), for example, one can express formulae that express their own unprovability within the system. By Gödel's incompleteness theorem such formulae are not provable within PA (and any formal system containing PA), though they are true. But what about formulae that express their own provability? By Löb's theorem, in PA the formula expressing "When I'm provable, I'm true" can be proved in PA only if the self-referential formula can be proved in PA. In modal-logical terms Löb's theorem can be expressed by the formula

$$\Box(\Box\varphi \to \varphi) \to \Box\varphi.$$

**Example 2.8** (Propositional dynamic logic (PDL))**.** The language of PDL, $\mathcal{L}_{PDL}$, is slightly more difficult than the languages considered so far. Let $A$ be a (finite) set of terms (called *atomic programs*). Then the set of $\mathcal{L}_{PDL}$-formulae is defined by:

$$\begin{aligned}\varphi &::= p \mid \bot \mid \neg\varphi \mid (\varphi_1 \vee \varphi_2) \mid \langle\pi\rangle\varphi \\ \pi &::= a \mid (\pi_1 \cup \pi_2) \mid (\pi_1\,;\pi_2) \mid \pi^* \mid \varphi?\end{aligned}$$

That is, formulae and programs are defined by a mutual recursion on both formulae and programs. To explain this in more detail: $\langle\pi\rangle\varphi$ means that some execution of program $\pi$ starting in the present state will terminate in a state in which $\varphi$ is true. For the programs: $\pi_1 \cup \pi_2$ is the program that executes $\pi_1$ or executes $\pi_2$ (*(non-deterministic) choice*). $\pi_1;\pi_2$ is the program that executes first $\pi_1$ and then $\pi_2$ (*composition*). $\pi^*$ is the program that executes $\pi$ for a finite number of times or not at all (*iteration*). Finally, $\varphi?$ is the program that tests whether $\varphi$ holds. If so, it continues, otherwise it fails.
If we drop the "test"-constructor, the resulting logic is called *regular PDL* (and its formulae can be constructed by a simple recursion and thus within the setting given by Definition 2.5). Some PDL languages also consider programs that allow for expressing parallel execution of programs: $\langle\pi_1 \cap \pi_2\rangle\varphi$ is true in a state $s$ when there are parallel executions of $\pi_1$ and $\pi_2$ and a state $s'$ in which $\varphi$ is true such that both executions terminate in $s'$. We will come back to this point later.
A nice example of a regular PDL-formula is the Segerberg axiom:

$$[\pi^*](\varphi \to [\pi]\varphi) \to (\varphi \to [\pi^*]\varphi).$$

**Example 2.9** (Description logics)**.** Description logics (DL) comes with an own syntax that is closely related to that of modal languages. A typical set of DL-formulae can be split into two parts, namely formulae that ascribe predicates to objects and formulae that describe terminological knowledge, i.e., conceptual dependencies between predicates. Let $\langle N_C, N_R, N_I \rangle$ be a tuple of pairwise disjoint symbol sets. The elements of $N_C$, $N_R$, $N_I$ are referred to as *concept names*, *role names*, and *individual names*, respectively. Consider then the following rules to form *concepts*, *roles*, *assertions*, *definitions*, *role specifications*, and *formulae* (in this order):

$$\begin{aligned}
C &::= A \mid \bot \mid \top \mid \neg A \mid \neg C \mid (C_1 \sqcup C_2) \mid (C_1 \sqcap C_2) \mid \\
&\qquad \forall r \mid \exists r \mid \forall r.C \mid \exists r.C \mid \forall^{\geq n} r.C \mid \exists r^{\geq n}.C \mid \{a_1, \ldots, a_n\} \\
r &::= R \mid r_1 \circ r_2 \mid r^{-1} \mid r^* \\
\alpha &::= a : C \mid (a_1, a_2) : r \mid \neg a : C \mid \neg(a_1, a_2) : r \\
\delta &::= A \doteq C \mid C_1 \sqsubseteq C_2 \\
\rho &::= \mathrm{sub}(R_1, R_2) \mid \mathrm{disj}(R_1, R_2) \mid \mathrm{trans}(R) \mid \mathrm{refl}(R) \mid \mathrm{irrefl}(R) \mid r_1 \sqsubseteq r_2 \\
\varphi &::= \alpha \mid \delta \mid \rho
\end{aligned}$$

where $A$ ranges over concept names, $R, R_1, R_2$ over roles names, and $a, a_1, a_2, \ldots, a_n$ over individual names. By altering the set of allowed constructors one can define a family of rather expressive description logics. There are by far less expressive languages. For example: $\mathcal{AL}$ allows only negations of atomic concepts, concept intersection, universal restrictions ($\forall r.C$), and limited existential quantification ($\exists r$). $\mathcal{ALC}$ also allows negations of arbitrary concepts. It is clear that the concept part of $\mathcal{ALC}$ can be cast as a modal language in the sense of the term used here. We will later see that the linkage between modal logic and description goes even deeper, i.e., some (not all!) DLs can be conceived of as modal logics.
One interesting feature of DLs is that they provide a precise semantics for the *Web Ontology Languages (OWL)*, a family of languages designed to facilitate the specification of ontologies developed by the World Wide Web Consortium (W3C). OWL comes in many flavors, for example, OWL DL and OWL Lite. The new standard, OWL2, is based on a particular description logic, the logic $\mathcal{SROIQ}(D)$. We will describe the DL naming scheme as well as specific reasoning tasks in description logics in a later section.

**Example 2.10** (Arrow logic)**.** Arrow logic is a modal logic that enables to talk about entities that can be graphically depicted as arrows. The similarity type of arrow logic $\tau_{\rightarrow}$ has a nullary modality $1'$ (*identity* or *skip*), a unary modality $\otimes$ (*reverse*), and a binary modality $\circ$ (*composition*).

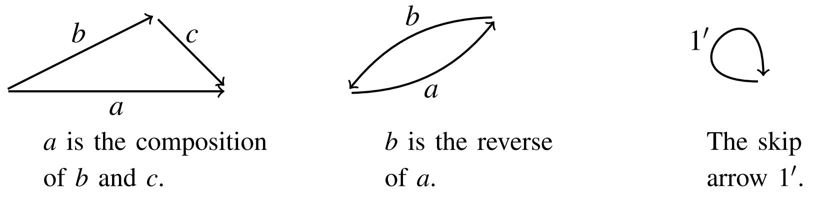

$b$
$c$
$a$
$a$ is the composition of $b$ and $c$.
$b$
$a$
$b$ is the reverse of $a$.
$1'$
The skip arrow $1'$.

Figure 2.2: Basic relations between arrows

A natural relational structure for the interpretation of the modal connectives considered in arrow logic are *two-dimensional arrow structures*: given a set $X$, consider a subset $A \subseteq X \times X$.

Then on $A$ the ternary composition relation, the binary reverse relation, and the unary identity relation are defined by:

$$\begin{aligned} Cabc &:\Leftrightarrow a_0 = b_0,\ a_1 = c_1, \text{ and } c_0 = b_1 \\ Rab &:\Leftrightarrow a_0 = b_1 \text{ and } a_1 = b_0 \\ Ia &:\Leftrightarrow a_0 = a_1 \end{aligned}$$

where $a = (a_0,a_1), b = (b_0,b_1), c = (c_0,c_1) \in A$. That is, $Cabc$ holds if $a$ is the composition of arrows $b$ and $c$, $Rab$ holds if $a$ is the reverse arrow of arrow $b$, and $Ia$ holds if the arrow $a$ forms a loop. If $A$ is the set of all element pairs from $X$ and the relations $I,R,C$ are defined on $A$ by the equations above, the relational structure $\langle A,C,R,I\rangle$ is called the *square* over $X$. More generally, relational structures of the form $\langle A,C,R,I\rangle$ are called *arrow frames*. Notice that arrows in arrow frames need not be arrows in a two-dimensional arrow structure.
Examples of such arrow frames arise in various contexts. Consider, for example, a set of functions, groups (in the sense of mathematical group theory), categories (in the sense of category theory), etc.

## 2.3 Relational models and satisfaction

**Definition 2.6.** A *frame* for a modal similarity type $\tau = \{\triangle_i\}_{i\in I}$ (short: $\tau$*-frame*) is a relational structure $\mathcal{F} = \langle S, \{R_i\}_{i\in I}\rangle$ consisting of a non-empty set $S$ of *states* (or: *possible worlds*) and a family of relations $R_i$ on $S$ such that for each $i \in I$, the arity of $R_i$ is $r_i + 1$ whence $r_i$ is the arity of $\triangle_i$. $|\mathcal{F}|$ denotes the set of states in $\mathcal{F}$.

For the similarity type of basic modal languages a $\tau$-frame is simply a set $S$ that is equipped with a family of binary relations: each diamond $\Diamond_i$ is associated with exactly one binary relation in the family. The relations $R_i$ are called *accessibility relations*. We say that state $s$ *sees* state $s'$ (or: $s'$ *is accessible from* $s$) via $R_i$ if $s\,R_i\,s'$. $\tau$-frames are also often called *Kripke frames*. An example of a Kripke frame is depicted in Figure 2.3.

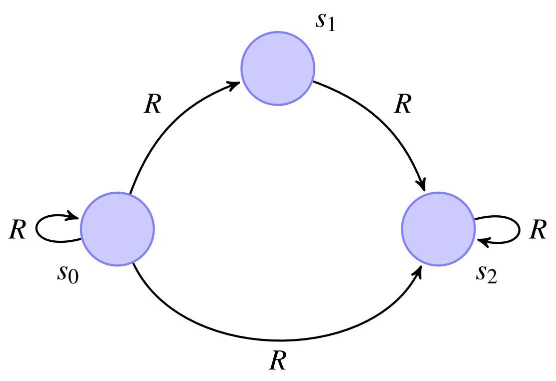


Figure 2.3: An example of a Kripke frame

**Definition 2.7.** For a modal language $\mathcal{L} = \mathcal{L}_\tau(P)$, an $\mathcal{L}$*-model* is a pair $\mathcal{M} = \langle \mathcal{F}, V\rangle$ consisting of a $\tau$-frame $\mathcal{F}$ and a valuation function $V : P \to 2^{|\mathcal{F}|}$.

If $P$ is understood, we will refer to $\mathcal{L}_\tau(P)$-models as $\tau$*-models*. In the situation of basic modal languages $\tau$-models are also called *Kripke models*. Kripke models can be depicted as directed multi-graphs that have both a vertex and an edge labeling (see Figure 2.4): vertices are labeled by a truth assignment and edges by accessibility relations.

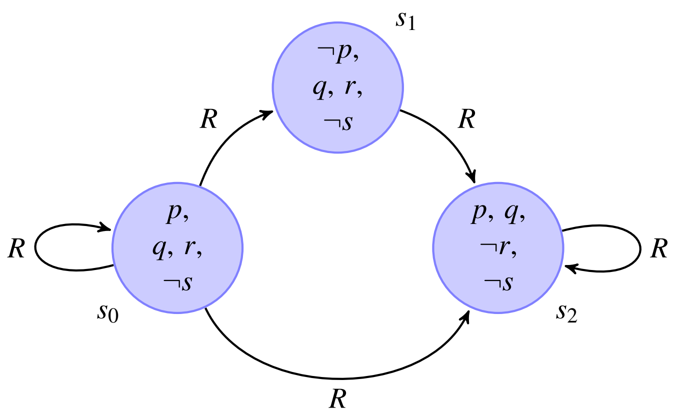


Figure 2.4: An example of a Kripke model

Given an $\mathcal{L}_\tau(P)$-model, $\mathcal{M} = \langle S, \{R_i\}_{i\in I}, V\rangle$, we define the *satisfaction relation*, $\mathcal{M} \models_s \varphi$, recursively as follows:

$$\begin{aligned}
\mathcal{M} &\not\models_s \bot \\
\mathcal{M} \models_s p &\iff s \in V(p) \\
\mathcal{M} \models_s \neg\varphi &\iff \mathcal{M} \not\models_s \varphi \\
\mathcal{M} \models_s \varphi \vee \psi &\iff \mathcal{M} \models_s \varphi \text{ or } \mathcal{M} \models_s \psi \\
\mathcal{M} \models_s \bigcirc_i &\iff s \in R_i \\
\mathcal{M} \models_s \lozenge_i\varphi &\iff \text{there exists } s' \in S \text{ with } sR_is' \text{ such that } \mathcal{M} \models_{s'} \varphi \\
\mathcal{M} \models_s \triangle_i(\varphi_1, \dots, \varphi_n) &\iff \text{there exist } s_1, \dots, s_n \in S \text{ with } R_iss_1\dots s_n \text{ such} \\
&\qquad \text{that for each } 1 \le j \le n,\ \mathcal{M} \models_{s_j} \varphi_j
\end{aligned}$$

An $\mathcal{L}$-formula $\varphi$ is said to be *satisfied in an $\mathcal{L}$-model $\mathcal{M}$ at state $s \in S$* if $\mathcal{M} \models_s \varphi$. $[\varphi]^{\mathcal{M}}$ denotes the set of all states at which $\varphi$ is satisfied in $\mathcal{M}$.

Notice that the condition for unary modal operators is just a special case of the general condition for modal operators with higher arity.

**Definition 2.8.** Let $\varphi$ be an $\mathcal{L}$-formula.

(a) $\varphi$ is *satisfied in an $\mathcal{L}$-model $\mathcal{M}$* if $[\varphi]^{\mathcal{M}} \neq \emptyset$.

(b) $\varphi$ is *valid in an $\mathcal{L}$-model $\mathcal{M}$*, $\mathcal{M} \models \varphi$, if $[\varphi]^{\mathcal{M}} = S$.

(c) $\varphi$ is *satisfiable* if it is satisfied in some $\mathcal{L}$-model, i.e., there exists an $\mathcal{L}$-model $\mathcal{M}$ and a state $s \in |\mathcal{M}|$ such that $\mathcal{M} \models_s \varphi$.

(d) $\varphi$ is *valid* if it is valid in all $\mathcal{L}$-models, i.e., for each $\mathcal{L}$-model $\mathcal{M}$ and each state $s \in |\mathcal{M}|$, $\mathcal{M} \models_s \varphi$.

In what follows we will omit indices of modal connectives and accessibility relations when possible.

**Lemma 2.2** (Coincidence Lemma)**.** *Let $\mathcal{M}$ and $\mathcal{M}'$ be $\mathcal{L}_\tau$-models on the same frame $\mathcal{F}$ that coincide on $Q \subseteq P$, i.e., $[p]^{\mathcal{M}} = [p]^{\mathcal{M}'}$ for each $p \in Q$. Then for each formula $\varphi$ in which only propositional variables from $Q$ occur, it holds $[\varphi]^{\mathcal{M}} = [\varphi]^{\mathcal{M}'}$, that is, for each state $s \in |\mathcal{F}|$,*

$$\mathcal{M} \models_s \varphi \iff \mathcal{M}' \models_s \varphi. \qquad \lhd$$

**Lemma 2.3** (Substitution Lemma)**.** *Let $\sigma : P \to \mathcal{L}_\tau(P)$ be a substitution and $\mathcal{M} = \langle \mathcal{F}, V \rangle$ be an $\mathcal{L}$-model. Let $\mathcal{M}^\sigma$ denote the model $\langle \mathcal{F}, V^\sigma \rangle$ where $V^\sigma$ is defined by $V^\sigma(p) := \{s \in |\mathcal{F}| : \mathcal{M} \models_s \sigma(p)\}$. Then for all formulae $\varphi$ and all states $s \in |\mathcal{F}|$,*

$$\mathcal{M}^\sigma \models_s \varphi \iff \mathcal{M} \models_s \varphi^\sigma. \qquad \lhd$$

**Lemma 2.4.** *The following formulae are valid in all $\mathcal{L}$-models:*

(a) $\Diamond(\varphi \vee \psi) \leftrightarrow (\Diamond\varphi \vee \Diamond\psi)$

(b) $\Box(\varphi \wedge \psi) \leftrightarrow (\Box\varphi \wedge \Box\psi)$

(c) $\Diamond(\varphi \wedge \psi) \to (\Diamond\varphi \wedge \Diamond\psi)$

(d) $(\Box\varphi \vee \Box\psi) \to \Box(\varphi \vee \psi)$

(e) $\Box\top$

(f) $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$

(g) $\Box(\varphi \to \psi) \to (\Diamond\varphi \to \Diamond\psi)$

(h) $(\Diamond p \wedge \Box q) \to \Diamond(p \wedge q)$

*Furthermore, it holds:*

(i) *If $\varphi$ is valid in $\mathcal{M}$, then so is $\Box\varphi$.*

(j) *If $\varphi$ is valid and $\sigma : P \to \mathcal{L}_\tau(P)$ is a substitution, then $\varphi^\sigma$ is valid.* $\lhd$

Note that if $\varphi$ is valid in a model, the formula $\Diamond\varphi$ need not be valid in the model as well. A simple (though not minimal) counterexample is presented in Figure 2.5. This example also shows that $\Box p \to \Diamond p$ is not valid in each Kripke model.

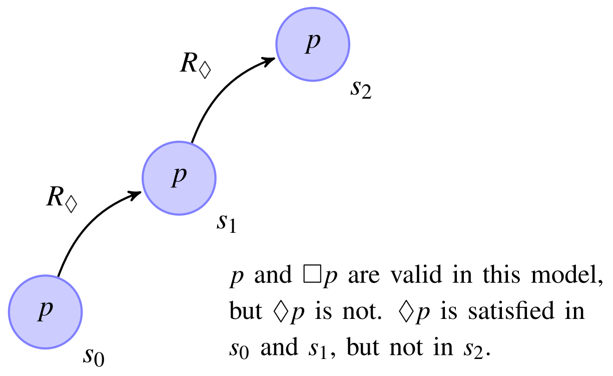

$R_\Diamond$
$p$
$s_2$
$R_\Diamond$
$p$
$s_1$
$p$
$s_0$
$p$ and $\Box p$ are valid in this model, but $\Diamond p$ is not. $\Diamond p$ is satisfied in $s_0$ and $s_1$, but not in $s_2$.

Figure 2.5: A Kripke model used as a counterexample

**Lemma 2.5.** *Let $\mathcal{M}$ be an $\mathcal{L}_\tau$-model with binary relations $R_i$ and $R_j$ such that $R_i \subseteq R_j$. Then the following formulae are valid in $\mathcal{M}$:*

*1.* $\Box_j\varphi \to \Box_i\varphi$

*2.* $\Diamond_i\varphi \to \Diamond_j\varphi$

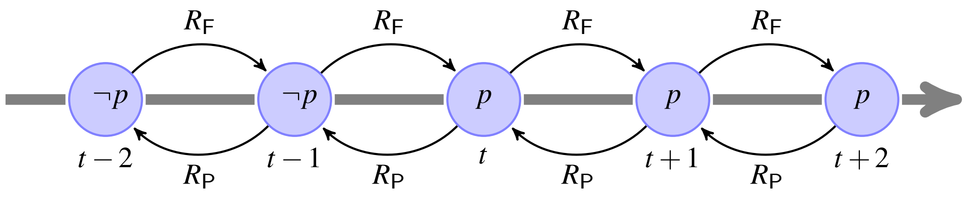
*In particular, if $\Box_j\varphi$ is valid in $\mathcal{M}$, then so is $\Box_i\varphi$. And if $\Box_j\varphi$ is satisfiable in $\mathcal{M}$, then so is $\Box_i\varphi$.* $\lhd$

In what follows we will consider in more detail the satisfaction condition for some of the special modal logics presented in the previous section.

**Example 2.11** (Basic temporal logic)**.** The basic temporal logic with the two diamond operators $\mathsf{F}$ and $\mathsf{P}$ is usually interpreted over frames $\langle T, R_{\mathsf{F}}, R_{\mathsf{P}}\rangle$, where $R_{\mathsf{P}}$ is the converse of relation $R_{\mathsf{F}}$. Such frames are called *bidirectional frames*. Frames for the basic temporal logic are typically just given by some strict order $\langle T, <\rangle$ and accordingly temporal models by triples $\mathcal{T} = \langle T, <, V\rangle$, where $\langle T, <\rangle$ is a temporal frame and $V$ is a valuation function. Thus the truth conditions for $\mathsf{F}$ and $\mathsf{P}$ can be written as follows:

$$\mathcal{T} \models_t \mathsf{F}\varphi \iff \mathcal{T} \models_{t'} \varphi, \text{ for some } t < t', \text{ and}$$
$$\mathcal{T} \models_t \mathsf{P}\varphi \iff \mathcal{T} \models_{t'} \varphi, \text{ for some } t' < t.$$

In temporal models the formulae $\mathsf{FH}\varphi \to \varphi$ and $\mathsf{PG}\varphi \to \varphi$ as well as their "duals" $\varphi \to \mathsf{GP}\varphi$ and $\varphi \to \mathsf{HF}\varphi$ are valid. This follows immediately from the following lemma.



The relation $R_{\mathsf{P}}$ is the converse of relation $R_{\mathsf{F}}$. The transitive closures of both relations are not depicted. In $t$ $\mathsf{H}\neg p$ is true, but $\mathsf{FH}\neg p$ is false. For any formula $\varphi$ true at $t$, $\mathsf{GP}\varphi$ and $\mathsf{HF}\varphi$ is true as well. The gray arrow represents the temporal ordering (the direction of the flow of time).

Figure 2.6: A temporal model

**Lemma 2.6.** *Let $\mathcal{M}$ be an $\mathcal{L}_\tau$-model with binary relations $R_i$ and $R_j$ such that $R_i \subseteq R_j^{-1}$. Then the following formulae are valid in $\mathcal{M}$:*

*1.* $\Diamond_i \Box_j \varphi \to \varphi$

*2.* $\varphi \to \Box_i \Diamond_j \varphi$

*Moreover, if $\Diamond_j \varphi \to \psi$ is valid in $\mathcal{M}$, then so is $\varphi \to \Box_i \psi$.* $\lhd$

**Example 2.12** (LTL)**.** We first consider LTL without the until operator. To define the semantics of such LTL formulae, let $s = s_1 s_2 \ldots$ be an infinite sequence of truth assignments $s_i : P \to \{0, 1\}$. For $i > 1$, let $s[i]$ denote $s_i$ and $s^i$ denote the suffix of $s$ that starts in $s_i$, i.e., $s^i := s_i s_{i+1} \ldots$ (which, of course, is also an infinite sequence of truth assignments). The semantics of LTL is usually defined as follows:

$$\begin{aligned} s \models p &\iff s[1](p) = 1 \\ s \models \neg\varphi &\iff s \not\models \varphi \\ s \models \varphi \vee \psi &\iff s \models \varphi \text{ or } s \models \psi \\ s \models \mathsf{F}\varphi &\iff s^i \models \varphi, \text{ for some } i \geq 1 \\ s \models \mathsf{X}\varphi &\iff s^2 \models \varphi \end{aligned}$$

It is clear that this semantics can also be cast in terms of Kripke models as introduced in this section. To see this, define such a model $\mathcal{M}^s$ as follows:

$$
\begin{aligned}
S &:= \mathbb{N} \\
R_{\mathsf{F}} &:= \mathrm{Succ}^* \\
R_{\mathsf{X}} &:= \mathrm{Succ} \\
V(s) &:= \{\, i \in \mathbb{N} : s[i](p) = 1 \}
\end{aligned}
$$

where $\mathrm{Succ} := \{\,(i, i+1) : i \in \mathbb{N}\}$ denotes the successor relation on $\mathbb{N}$. Then for each LTL formula $\varphi$ and each $i \in \mathbb{N}$, it holds:

$$\mathcal{M}^s \models_i \varphi \iff s^i \models \varphi.$$

In LTL with "until" the truth condition for $\mathsf{U}$ is given by

$$
\begin{aligned}
s \models \mathsf{U}(\varphi, \psi) \iff & \text{ there exist } i \geq 1 \text{ such that } s^i \models \psi \text{ and } s^j \models \varphi \\
& \text{ for each } 1 \leq j < i
\end{aligned}
$$

which corresponds to the following truth condition for temporal models:

$$
\begin{aligned}
\mathcal{M} \models_t \mathsf{U}(\varphi, \psi) \iff & \text{ there is a } t' \geq t \text{ with } \mathcal{M} \models_{t'} \psi \text{ and } \mathcal{M} \models_{t''} \varphi \\
& \text{ for each } t \leq t'' < t'.
\end{aligned}
$$

**Example 2.13** (Regular PDL)**.** PDL- and in particular regular PDL-formulae are also interpreted on Kripke models, but (as in the case of temporal logic) the class of considered PDL-models is restricted to Kripke-models satisfying the following equations:

$$
\begin{aligned}
R_{\pi_1 \cup \pi_2} &= R_{\pi_1} \cup R_{\pi_2} \\
R_{\pi_1 ; \pi_2} &= R_{\pi_1} \circ R_{\pi_2} \\
R_{\pi *} &= R_{\pi}^*
\end{aligned}
$$

that is, the accessibility relations need to satisfy dependency conditions to express the intended meaning of the constructors of programs.

**Example 2.14** ($\mathcal{ALC}$)**.** As mentioned in Example 2.9, in the description logic $\mathcal{ALC}$ complex concepts are generated from concept names and role names by applying concept negation ($\neg C$), concept intersection ($C_1 \sqcap C_2$), concept disjunction ($C_1 \sqcup C_2$), universal restrictions ($\forall r.C$), and existential restrictions ($\exists r.C$). As simple $\mathcal{ALC}$-formulae we confine here to simple assertions of the form $a : C$ or $(a_1, a_2) : R$. To interpret this formal language one considers relational structures of the form

$$\mathcal{M} = \langle U, r_1^{\mathcal{M}}, \ldots, r_n^{\mathcal{M}}, C_1^{\mathcal{M}}, \ldots, C_m^{\mathcal{M}}, a_1^{\mathcal{M}}, \ldots, a_k^{\mathcal{M}} \rangle$$

where $U$ is a non-empty set (*universe*). For each role name $r_i$, $r_i^{\mathcal{M}}$ is a binary relation on $U$, for each concept name $C_j$, $C_j^{\mathcal{M}}$ is a subset of $U$, and for each individual name $a_l$, $a_l^{\mathcal{M}}$ is an element of $U$. Inductively, we define the interpretation of concept terms as follows:

$$\begin{aligned}
\top^{\mathcal{M}} &:= U\\
\bot^{\mathcal{M}} &:= \emptyset\\
(\neg C)^{\mathcal{M}} &:= U \setminus C^{\mathcal{M}}\\
(C \sqcap C')^{\mathcal{M}} &:= C^{\mathcal{M}} \cap C'^{\mathcal{M}}\\
(C \sqcup C')^{\mathcal{M}} &:= C^{\mathcal{M}} \cup C'^{\mathcal{M}}\\
(\exists r.C)^{\mathcal{M}} &:= \{x \in U \,:\, \text{there is a } y \in C^{\mathcal{M}} \text{ with } x\, r^{\mathcal{M}}\, y\}\\
(\forall r.C)^{\mathcal{M}} &:= \{x \in U \,:\, \text{for each } y \in U \text{ with } x\, r^{\mathcal{M}}\, y,\, y \in C^{\mathcal{M}}\}
\end{aligned}$$

Then the satisfaction concept for simple $\mathcal{ALC}$-formulae is defined by:

$$\begin{aligned}
\mathcal{M} \models a : C &\iff a^{\mathcal{M}} \in C^{\mathcal{M}}\\
\mathcal{M} \models (a_1, a_2) : R &\iff a^{\mathcal{M}}\, R^{\mathcal{M}}\, b^{\mathcal{M}}
\end{aligned}$$

An $\mathcal{ALC}$-formula $\varphi$ is said to be *satisfiable* if there exists an $\mathcal{ALC}$-model $\mathcal{M}$ with $\mathcal{M} \models \varphi$.

**Example 2.15** (Arrow logic)**.** Remember that the modal type of arrow logic, $\tau_{\rightarrow}$ consists of a binary operator $\circ$, a unary one $\otimes$, and a nullary one $1'$. Then an *arrow model* is given by a tuple $\mathcal{M} = \langle A, C, R, I, V\rangle$, where $\langle A, C, R, I\rangle$ is an arrow frame (see Example 2.10). The satisfaction relation then is exactly the satisfaction relation defined for $\tau_{\rightarrow}$-models, i.e.,

$$\begin{aligned}
\mathcal{M} \models_a 1' &\iff I\, a\\
\mathcal{M} \models_a \otimes\varphi &\iff \mathcal{M} \models_b \varphi, \text{ for some } b \in A \text{ with } a\, R\, b\\
\mathcal{M} \models_a \varphi \circ \psi &\iff \mathcal{M} \models_b \varphi \text{ and } \mathcal{M} \models_c \psi, \text{ for some } b, c \in A \text{ with } C\, abc
\end{aligned}$$

## 2.4 Constructing models

In this section we will introduce some basic methods on how models can be constructed from other methods.

**Definition 2.9.** Let $\mathcal{M} = \langle S, \{R_i\}_{i\in I}, V\rangle$ and $\mathcal{M}' = \langle S', \{R'_i\}_{i\in I}, V'\rangle$ be $\mathcal{L}$-models. A function $f : S \to S'$ is called a *bounded morphism* if the following conditions are satisfied:

(a) $f$ is a homomorphism of the frames (conceived of as relational structures).

(b) For all $s \in S, s'_1, \ldots, s'_n \in S'$, if $R'_i f(s) s'_1 \ldots s'_n$, then there exists $s_1, \ldots, s_n \in S$ with $f(s_i) = s'_i$ such that $R_i s s_1 \ldots s_n$.

(c) For each $s \in S$ and each $p \in P$, $s \in V(p)$ if and only if $f(s) \in V'(p)$.

Condition (a) is called the *forth condition* and (b) the *back condition*. In the case of basic modal languages these two conditions can be restated by:

$$\begin{aligned}
s_1\, R_i\, s_2 &\Rightarrow f(s_1)\, R'_i\, f(s_2)\\
f(s_1)\, R'_i\, s'_2 &\Rightarrow s_1\, R_i\, s \text{ and } f(s) = s'_2 \text{ for some } s \in S.
\end{aligned}$$

Similarly, the concept of *bounded morphism* between frames, $f : \mathcal{F} \to \mathcal{F}'$, is introduced (i.e. $f : S \to S'$ is a function satisfying the forth and back conditions of Definition 2.9).

**Proposition 2.7** (Invariance under bounded morphisms)**.** *Let $f : \mathcal{M} \to \mathcal{M}'$ be a bounded morphism of $\mathcal{L}$-models. Then for each $s \in S$ and each $\mathcal{L}$-formula $\varphi$,*

$$\mathcal{M} \models_s \varphi \iff \mathcal{M}' \models_{f(s)} \varphi.$$

*Proof.* The proof proceeds by induction on the length of formulae. Indeed, for propositional variables the claim follows immediately from Definition 2.9(c). And if $\varphi$ is the formula $\bot$, nothing is to be shown.

Assume now that $\varphi$ is a formula of length $n$ and that the claim holds for all formulae of length $< n$. If $\varphi$ has the form $\neg\varphi'$, it holds

$$\begin{aligned}\mathcal{M} \models_s \neg\varphi' &\iff \mathcal{M} \not\models_s \varphi' \\ &\iff \mathcal{M}' \not\models_{f(s)} \varphi' \\ &\iff \mathcal{M}' \models_{f(s)} \neg\varphi'.\end{aligned}$$

In case $\varphi$ has the form $\varphi_1 \vee \varphi_2$, we obtain

$$\begin{aligned}\mathcal{M} \models_s \varphi_1 \vee \varphi_2 &\iff \mathcal{M} \models_s \varphi_1 \text{ or } \mathcal{M} \models_s \varphi_2 \\ &\iff \mathcal{M}' \models_{f(s)} \varphi_1 \text{ or } \mathcal{M}' \models_{f(s)} \varphi_2 \\ &\iff \mathcal{M}' \models_{f(s)} \varphi_1 \vee \varphi_2.\end{aligned}$$

If $\varphi$ has the form $\bigcirc_i$,

$$\mathcal{M} \models_s \bigcirc_i \iff s \in R_i \iff f(s) \in R'_i \iff \mathcal{M}' \models_{f(s)} \bigcirc_i.$$

Finally if $\varphi$ has the form $\triangle_i(\varphi_1, \dots, \varphi_n)$, we obtain:

$$\begin{aligned}\mathcal{M} \models_s \triangle_i(\varphi_1, \dots, \varphi_n) &\iff \text{there are } s_1, \dots, s_n \in S \text{ with } R_i s s_1 \dots s_n \text{ and} \\ &\qquad \mathcal{M} \models_{s_i} \varphi_i \text{ for each } 1 \le i \le n \\ &\iff \text{there are } s_1, \dots, s_n \in S \text{ with } R'_i f(s) f(s_1) \dots f(s_n) \text{ and} \\ &\qquad \mathcal{M}' \models_{f(s_i)} \varphi_i \text{ for each } 1 \le i \le n \\ &\iff \text{there are } s'_1, \dots, s'_n \in S' \text{ with } R'_i f(s) s'_1 \dots s'_n \text{ and} \\ &\qquad \mathcal{M}' \models_{s'_i} \varphi_i \text{ for each } 1 \le i \le n \\ &\iff \mathcal{M}' \models_{f(s)} \triangle_i(\varphi_1, \dots, \varphi_n).\end{aligned}$$

Here in one direction the forth condition and in the other direction the back condition of bounded morphisms is used. In all cases the induction hypothesis has been applied to formulae of shorter length. ◁

**Corollary 2.8.** *Let $f : \mathcal{F} \to \mathcal{F}'$ be a bounded morphism of $\tau$-frames that is surjective. Then for each formula $\varphi$, if $\varphi$ is valid in each model defined on frame $\mathcal{F}$, then $\varphi$ is valid in each model defined on frame $\mathcal{F}'$.*

*Proof.* We prove the contraposition of the claim: Assume that there is a model $\mathcal{M}' = \langle \mathcal{F}', V' \rangle$ and a state $s'$ of $\mathcal{F}'$ such that $\mathcal{M}' \not\models_{s'} \varphi$. Define a model $\mathcal{M}$ on $\mathcal{F}$ by $V(p) := \{s \in S : \mathcal{M}' \models_{f(s)} p\}$. Then $f$ is a bounded morphism of the $\mathcal{L}$-models. Since $f$ is surjective, there exists an $s \in S$ with $f(s) = s'$. By Proposition 2.7, it follows that $\mathcal{M} \not\models_s \varphi$. This shows that $\varphi$ is not valid in each model over $\mathcal{F}$. ◁

**Definition 2.10** (Submodel)**.** Let $\mathcal{M} = \langle S, \{R_i\}_{i\in I}, V\rangle$ be an $\mathcal{L}$-model and $S'$ be a non-empty subset of $S$. The $\mathcal{L}$-model $\mathcal{M}' = \langle S', \{R'_i\}_{i\in I}, V'\rangle$ defined by

$$R'_i := R_i \cap S'^{r_i} \text{ and } V'(p) := V(p) \cap S'$$

(where for each $i \in I$, $r_i$ is the arity of $R_i$ and $p \in P$) is called *the submodel of $\mathcal{M}$ induced by $S'$*. Given $S'$, let $<S'>$ denote the smallest superset of $S'$ that is *closed* with respect to all relations $R_i$ in $\mathcal{M}$, i.e., for all $s, s_1, \ldots, s_n \in S$ and $i \in I$, it holds:

$$s \in <S'>, \ \ R_i s s_1 \ldots s_n \Rightarrow s_1, \ldots, s_n \in <S'>.$$

Then the *submodel of $\mathcal{M}$ generated by $S'$* is the submodel of $\mathcal{M}$ that is induced by $<S'>$. A *generated submodel* of $\mathcal{M}$ is any submodel that is generated by some subset of $S$. A *point generated submodel* is a submodel that is generated by a singleton set.

**Proposition 2.9.** *Let $\mathcal{M} = \langle S, \{R_i\}_{i\in I}, V\rangle$ be an $\mathcal{L}$-model, $S'$ be a non-empty subset of $S$, and $\mathcal{M}'$ be the submodel of $\mathcal{M}$ generated by $S'$. Then the inclusion function $\iota : <S'> \hookrightarrow S$ is a bounded morphism. In particular, for each $s \in <S'>$ and each $\mathcal{L}$-formula $\varphi$,*

$$\mathcal{M}' \models_s \varphi \iff \mathcal{M} \models_s \varphi.$$

*Proof.* It can easily be shown that $\iota : <S'> \hookrightarrow S$ is a bounded morphism of $\mathcal{L}$-models. The second claim then follows immediately from Proposition 2.7. $\lhd$

**Definition 2.11** (Disjoint union)**.** Let $\{\mathcal{M}_j\}_{j\in J}$ be a family of pairwise disjoint $\mathcal{L}$-models $\mathcal{M}_j = \langle S^j, \{R^j_i\}_{i\in I}, V^j\rangle$, i.e., $S^j \cap S^{j'} = \emptyset$ for all $j, j' \in J$ with $j \neq j'$. Then define an $\mathcal{L}$-model $\biguplus_{j\in J} \mathcal{M}_j = \langle S, \{R_i\}_{i\in I}, V\rangle$ by

$$S := \bigcup_{j\in J} S^j, \ \ R_i := \bigcup_{j\in J} R^j_i, \ \text{ and } \ V(p) := \bigcup_{j\in J} V^j(p)$$

(for $i \in I$ and $p \in P$). This model is called the *disjoint union* of the models $\mathcal{M}_j$.

**Proposition 2.10.** *Let $\{\mathcal{M}_j\}_{j\in J}$ be a family of pairwise disjoint $\mathcal{L}$-models. Then for each $j \in J$, $\mathcal{M}_j$ is a generated submodel of $\biguplus_{j\in J} \mathcal{M}_j$ and the inclusion function $\iota : \mathcal{M}_j \hookrightarrow \biguplus_{j\in J} \mathcal{M}_j$ is a bounded morphism. That is, for each $s \in S^j$ and each $\mathcal{L}$-formula $\varphi$,*

$$\mathcal{M}_j \models_s \varphi \iff \biguplus_{j\in J} \mathcal{M}_j \models_s \varphi.$$

*Proof.* In fact, each $\mathcal{M}_j$ is a submodel (generated by $S^j$) of $\biguplus_{j\in J} \mathcal{M}_j$. Thus the claim follows from Proposition 2.9. $\lhd$

## 2.5 Translating modal logic into first-order logic

Modal logic is an extension of propositional logic, syntactically as well as semantically. But how does modal logic compare to first order logic? From the modal-logical semantics defined previously it is obvious that modal logic can be embedded into first order logic (FOL). More precisely, there is a standard translation of modal logic (with the standard relational semantics) into first-order logic (FOL). In this section we will spell out this translation in a precise way and we will draw some immediate conclusions from theoretical results concerning FOL.

In what follows, let $\mathcal{L} := \mathcal{L}_\tau(P)$ be a fixed language of modal logic. There is a first order language that is associated with $\mathcal{L}_\tau(P)$ in a natural way, namely the language $\mathcal{L}_{\text{FOL}} := \mathcal{L}_{\text{FOL}}(\tau, P)$ with the following alphabet:

- denumerably many variables $v_0, v_1, v_2, \ldots$;
- for each $p \in P$, a unary relation symbol $F_p$;
- for each $\triangle_i \in \tau$, an $r_i + 1$-ary relation symbol $A_i$;
- logical symbols $=$, $\neg$, $\rightarrow$, $\wedge$, $\vee$, $\exists$, and $\forall$.

That is, the *signature* of $\mathcal{L}_{\text{FOL}}$ depends on the set of propositional variables and the similarity type of $\mathcal{L}_\tau(P)$.

$\mathcal{L}_{\text{FOL}}$-formulae are defined in the usual manner (the set of $\mathcal{L}_{\text{FOL}}$-formulae is denoted by $\mathcal{L}_{\text{FOL}}$ as well). In particular, this means that for each $\mathcal{L}_{\text{FOL}}$-formula $\varphi$ and each variable $x$, $\forall x \varphi$ and $\exists x \varphi$ are $\mathcal{L}_{\text{FOL}}$-formulae. One should recall usual notions of FOL such as *free (occurrence of a) variable*, *(variable) assignment*, *first order structure*, etc: for example, a first-order $\mathcal{L}_{\text{FOL}}$-structure is a tuple

$$\mathcal{S} = \left\langle U, \{A_i^{\mathcal{S}}\}_{i \in I}, \{F_p^{\mathcal{S}}\}_{p \in P} \right\rangle$$

where each $A_i^{\mathcal{S}}$ is an $r_i + 1$-ary relation on $U$ and each $F_p^{\mathcal{S}}$ is a subset of $U$. A variable assignment in $\mathcal{S}$ is simply a function $a : V \to U$ and the satisfaction relation is defined by:

$$\begin{aligned}
\mathcal{S}, a \models F_p(x) &\iff a(x) \in F_p^{\mathcal{S}} \\
\mathcal{S}, a \models A_i(x_1, \ldots, x_{r_i+1}) &\iff (a(x_1), \ldots, a(x_{r_i+1})) \in A_i^{\mathcal{S}} \\
\mathcal{S}, a \models x = y &\iff a(x) = a(y) \\
\mathcal{S}, a \models \neg\varphi &\iff \mathcal{S}, a \not\models \varphi \\
&\ldots \\
\mathcal{S}, a \models \exists x \varphi &\iff \mathcal{S}, a[x \mapsto u] \models \varphi \text{ for some } u \in U \\
\mathcal{S}, a \models \forall x \varphi &\iff \mathcal{S}, a[x \mapsto u] \models \varphi \text{ for each } u \in U
\end{aligned}$$

Here $a[x \mapsto u]$ is the variable assignment that coincides with $a$ in the assignment of all variables except $x$ and assigns $u$ to $x$. We write $\mathcal{S} \models \varphi$ if $\mathcal{S}, a \models \varphi$ for each variable assignment $a$. For $\mathcal{L}_{\text{FOL}}$-*sentences* (i.e., formulae in which no variable occurs free) the satisfaction relation does not depend on the variable assignment, i.e., for each sentence $\varphi$, it holds $\mathcal{S} \models \varphi$ or $\mathcal{S} \models \neg\varphi$. A set of $\mathcal{L}_{\text{FOL}}$-sentences is *satisfiable* if there exists an $\mathcal{L}_{\text{FOL}}$-structure $\mathcal{S}$ with $\mathcal{S} \models \varphi$ for each $\varphi \in \Sigma$. We write $\Phi(x)$ to denote a set of formulae in which at most variable $x$ occurs free. Such a $\Phi(x)$ is called *satisfiable* if there exists a structure $\mathcal{S}$ and an element $u \in U$ such that $\mathcal{S}, (x \mapsto u) \models \varphi$ for each $\varphi \in \Phi(x)$ (here $(x \mapsto u)$ refers to an arbitrary variable assignment $a$ with $a(x) = u$). Alternatively, we may expand $\mathcal{L}_{\text{FOL}}$ by an individual constant $c$; then the set $\Phi(x)[x/c]$ resulting from replacing in each formula $\varphi \in \Phi(x)$ each *free* occurrence of $x$ by $c$ is a set of $\mathcal{L}_{\text{FOL}} \cup \{c\}$-sentences. It is easy to see that $\Phi(x)[x/c]$ is satisfiable (as a set of $\mathcal{L}_{\text{FOL}} \cup \{c\}$-sentences) if and only if $\Phi(x)$ is satisfiable (as a set of $\mathcal{L}_{\text{FOL}}$-formulae).

We now define the *standard translation* of $\mathcal{L}_\tau(P)$ into $\mathcal{L}_{\text{FOL}}(\tau, P)$, which is a mapping

$$\text{ST}_x : \mathcal{L}_\tau(P) \to \mathcal{L}_{\text{FOL}}(\tau, P)$$

where $x$ is some variable of $\mathcal{L}_{\mathrm{FOL}}$. Then define

$$\begin{aligned}
\mathrm{ST}_x(p) &= F_p(x)\\
\mathrm{ST}_x(\bot) &= x \neq x\\
\mathrm{ST}_x(\neg\varphi) &= \neg\,\mathrm{ST}_x(\varphi)\\
\mathrm{ST}_x((\varphi\vee\psi)) &= (\mathrm{ST}_x(\varphi)\vee\mathrm{ST}_x(\psi))\\
\mathrm{ST}_x(\bigcirc_i) &= A_i(x)\\
\mathrm{ST}_x(\Diamond_i\varphi) &= \exists y(A_i(x,y)\wedge\mathrm{ST}_y(\varphi))\\
\mathrm{ST}_x(\triangle_i(\varphi_1,\dots,\varphi_{r_i})) &= \exists y_1\dots y_{r_i}(A_i(x,y_1,\dots,y_{r_i})\wedge\bigwedge_{1\le j\le r_i}\mathrm{ST}_{y_i}(\varphi_i))
\end{aligned}$$

where $y$ and $y_1,\dots,y_{r_i}$ are the variables with smallest index (in the sequence $v_o,v_1,\dots$) that do not occur in $\mathrm{ST}_x(\varphi)$ and all the $\mathrm{ST}_x(\varphi_i)$, respectively. Notice that in $\mathrm{ST}_x(\varphi)$ exactly one variable occurs free (not within the scope of a quantifier), namely the variable $x$.

**Example 2.16.** As an example consider the formula $\Diamond\Diamond\varphi\rightarrow\Diamond\varphi$. The standard translation of this formula is

$$\mathrm{ST}_x(\neg\Diamond\Diamond p\vee\Diamond p) = \neg\exists y\,(A(x,y)\wedge\exists z\,(A(y,z)\wedge F_p(z)))\vee\exists y\,(A(x,y)\wedge F_p(y))),$$

where $y,z$ are suitable variables of $\mathcal{L}_{\mathrm{FOL}}$. This formula is FO-equivalent to:

$$\exists y\,(A(x,y)\wedge\exists z\,(A(y,z)\wedge F_p(z)))\rightarrow\exists y\,(A(x,y)\wedge F_p(y))),$$

and this formula in turn is equivalent to:

$$\exists y\,z(A(x,y)\wedge A(y,z)\wedge F_p(z))\rightarrow\exists y\,(A(x,y)\wedge F_p(y)).$$

By contraposition, we obtain:

$$\forall y\,(A(x,y)\rightarrow\neg F_p(y))\rightarrow\forall y\,z(A(x,y)\wedge A(y,z)\rightarrow\neg F_p(z)).$$

In section 3 we will see that the formula $\Diamond\Diamond\varphi\rightarrow\Diamond\varphi$ is closely related to the transitivity of the accessibility relation. This connection becomes more obvious if we consider a fixed interpretation of the predicates $F_p(v)$: let us assume that for each variable $v$,

$$F_p(v)\iff\neg A(x,v)$$

(note that $x$ is a fixed variable). Thus, the formula on the right-hand side of the equation is equivalent to

$$\forall y\,z(A(x,y)\wedge A(y,z)\rightarrow A(x,z)).$$

This observation is the starting point of the so-called *correspondence theory* of modal logic. Correspondence theory deals with the question of which formulae of modal logic correspond to first order properties of Kripke frames. We will discuss this in more detail in the next section.

How are models of $\mathcal{L}_\tau$ and models of $\mathcal{L}_{\mathrm{FOL}}$ related to each other? Let $\mathcal{M}=\langle S,\{R_i\}_i,V\rangle$ be an $\mathcal{L}$-model (i.e., a Kripke model in the case of basic modal languages). Then the ordered pair $\mathcal{S}_{\mathcal{M}}=\langle S,\{A_i^{\mathcal{S}_{\mathcal{M}}}\}_i,\{F_p^{\mathcal{S}_{\mathcal{M}}}\}_p\rangle$ with

$$F_p^{\mathcal{S}_{\mathcal{M}}}:=V(p)\quad\text{and}\quad A_i^{\mathcal{S}_{\mathcal{M}}}:=R_i$$

is an $\mathcal{L}_{\text{FOL}}$-structure. Conversely, if $\mathcal{S} = \langle U, \{A_i^{\mathcal{S}}\}_i, \{F_p^{\mathcal{S}}\}_p\}\rangle$ is an $\mathcal{L}_{\text{FOL}}(\tau, P)$-structure, then the triple $\mathcal{M}^{\mathcal{S}} = \langle U, \{R_i\}_i, V\rangle$ with

$$R_i := A_i^{\mathcal{S}} \quad \text{and} \quad V(p) := F_p^{\mathcal{S}}$$

defines an $\mathcal{L}_\tau(P)$-model.

**Theorem 2.11** (Standard translation into FOL)**.** *The mapping $\mathcal{M} \mapsto \mathcal{S}_{\mathcal{M}}$ defines a bijection between the class of $\mathcal{L}_\tau(P)$-models and the class of first order $\mathcal{L}_{\text{FOL}}(\tau, P)$-structures. Moreover, for each $\mathcal{L}_\tau(P)$-formula $\varphi$, it holds:*

(a) *$\mathcal{M} \models_s \varphi$ if and only if $\mathcal{S}_{\mathcal{M}}, (x \mapsto s) \models \mathrm{ST}_x(\varphi)$;*

(b) *$\mathcal{M} \models \varphi$ if and only if $\mathcal{S}_{\mathcal{M}} \models \forall x\, \mathrm{ST}_x(\varphi)$.*

*Proof.* Obviously, (b) follows from (a). And (a) can easily be proven by induction on $\varphi$. $\lhd$

## 2.6 Consequence relation and compactness

In propositional logic, the *(semantic) consequence relation*, $\Sigma \models \varphi$, is defined as follows: $\varphi$ *follows* from $\Sigma$ if each truth assignment that makes all formulae in $\Sigma$ true, makes $\varphi$ true as well. In modal logic there are at least two alternative definitions of consequence, since there are different concepts of a formula being true in a model. In the *global reading*, $\Sigma \models \varphi$ means that for each model in which each formula of $\Sigma$ is valid, $\varphi$ is valid as well. In the *local reading*, $\Sigma \models \varphi$ means that for each model and each state $s$ in that model in which each formula of $\Sigma$ is satisfied, $\varphi$ is satisfied as well. In what follows we use the local definition.

**Definition 2.12** (Semantic consequence)**.** The formula $\varphi$ is a *consequence* of a set of formulae, $\Sigma$, $\Sigma \models \varphi$, if in each state $s$ of each model $\mathcal{M}$, $\mathcal{M} \models_s \Sigma$ implies $\mathcal{M} \models_s \varphi$.

**Lemma 2.12.** (a) *$\emptyset \models \varphi$ if and only if $\varphi$ is valid.*

(b) *$\Sigma \not\models \varphi$ if and only if $\Sigma \cup \{\neg\varphi\}$ is satisfiable.* $\lhd$

FOL is characterized by two properties. First, FOL is *compact*, i.e., a set of first order sentences (formulae without free variables) is satisfiable if and only if each of its finite subsets is satisfiable. Secondly, FOL has the Löwenheim-Skolem property, i.e., each set of first order sentences (in a language with countable signature) that is satisfiable in some model is satisfiable in a model that is defined on a countable domain (here and in what follows, "countable" means "finite" or "countably infinite"). In fact, as was proven by Lindstroem, FOL is the strongest logic that is compact and satisfies Löwenheim-Skolem (see Ebbinghaus et al., 2007, chapter XIII).

In the sequel we will show that analogous results carry over to modal logic. Thereby we will apply the standard translation introduced above.

**Theorem 2.13** (Compactness for models)**.** *Let $\Sigma$ be a set of $\mathcal{L}_\tau(P)$-formulae. If each finite subset of $\Sigma$ is satisfiable, then so is $\Sigma$.*

*Proof.* Assume that each finite subset of $\Sigma$ is satisfiable in some $\mathcal{L}$-model. We expand $\mathcal{L}_{\text{FOL}}$ by a new individual constant $c$ and consider the set of $\mathcal{L}_{\text{FOL}} \cup \{c\}$-formulae

$$\mathrm{ST}_{x/c}(\Sigma) := \{\, (\mathrm{ST}_x(\varphi))[x/c] \,:\, \varphi \in \Sigma\}.$$

First we show that each finite subset of $\mathrm{ST}_{x/c}(\Sigma)$ is satisfiable (in some first-order structure). For proof by contradiction assume that there is a finite subset $\{\varphi_1,\ldots,\varphi_n\}$ of $\Sigma$ such that

$$\{(\mathrm{ST}_x(\varphi_1))[x/c],\ldots,(\mathrm{ST}_x(\varphi_n))[x/c]\}$$

is not satisfiable. This means that

$$\bigwedge_{i=1}^{n}(\mathrm{ST}_x(\varphi_i))[x/c] = (\bigwedge_{i=1}^{n}\mathrm{ST}_x(\varphi_i))[x/c] = (\mathrm{ST}_x(\bigwedge_{i=1}^{n}\varphi_i))[x/c]$$

is not satisfiable either. But each finite subset of $\Sigma$ is satisfiable, in particular $\{\varphi_1,\ldots,\varphi_n\}$. Hence there is a (modal-logical) $\mathcal{L}$-model $\mathcal{M}$ with some state $s^*$ such that $\mathcal{M}\models_{s^*}\bigwedge_{i=1}^{n}\varphi_i$. By Theorem 2.11 we obtain that $\mathcal{S}_{\mathcal{M}},(x\mapsto s^*)\models\mathrm{ST}_x(\bigwedge_{i=1}^{n}\varphi_i)$. $\mathcal{S}_{\mathcal{M}}$ can be expanded to an $\mathcal{L}_{\mathrm{FOL}}\cup\{c\}$-model $\mathcal{S}'$ if we set $c^{\mathcal{S}'}:=s^*$. Obviously $\mathcal{S}'\models(\mathrm{ST}_x(\bigwedge_{i=1}^{n}\varphi_i))[x/c]$ — in contradiction to our observation that this set is not satisfiable.
Thus, each finite subset of $\mathrm{ST}_{x/c}(\Sigma)$ is satisfiable. By compactness of FOL, $\mathrm{ST}_{x/c}(\Sigma)$ must be satisfiable as well. That is, there exists an $\mathcal{L}_{\mathrm{FOL}}\cup\{c\}$-structure $\mathcal{S}'$ with $\mathcal{S}'\models(\mathrm{ST}_x(\varphi))[x/c]$ for each $\varphi\in\Sigma$. If we "forget" in $\mathcal{S}'$ the interpretation of $c$, we obtain an $\mathcal{L}_{\mathrm{FOL}}$-structure $\mathcal{S}$ with

$$\mathcal{S},(x\mapsto c^{\mathcal{S}'})\models\mathrm{ST}_x(\varphi)$$

for each $\varphi\in\Sigma$. By Theorem 2.11, it follows

$$\mathcal{M}^{\mathcal{S}}\models_{c^{\mathcal{S}'}}\varphi$$

for each $\varphi\in\Sigma$. Thus $\Sigma$ is satisfiable in $\mathcal{M}^{\mathcal{S}}$. $\lhd$

The compactness theorem stated above can just serve as some initial observation. Usually, one is interested in compactness results that relate the model of the large set $\Sigma$ somehow to the models of its finite subsets. We come back to this point later.

**Theorem 2.14** (Löwenheim-Skolem)**.** *Let $\mathcal{L}$ be a basic modal language with countably many propositional variables and $\Sigma$ be a set of $\mathcal{L}$-formulae. If $\Sigma$ is satisfiable in some $\mathcal{L}$-model, then $\Sigma$ is satisfiable in a countable $\mathcal{L}$-model $\mathcal{M}$ (i.e., $\mathcal{M}$ is defined on a finite or countably infinite set of states).*

*Proof.* Assume that $\Sigma$ is satisfiable. That is, there exists an $\mathcal{L}$-model $\mathcal{M}$ with a state $s$ such that $\mathcal{M}\models_s\Sigma$. By Theorem 2.11, we obtain that

$$\mathcal{S}_{\mathcal{M}},(x\mapsto s)\models\mathrm{ST}_x(\Sigma),$$

i.e., the set of sentences $\mathrm{ST}_{x/c}(\Sigma)$ is satisfiable in an $\mathcal{L}_{\mathrm{FOL}}\cup\{c\}$-structure (again $c$ is a new individual constant). Since FOL has the Löwenheim-Skolem property, $\mathrm{ST}_{x/c}(\Sigma)$ is satisfiable in a countable $\mathcal{L}_{\mathrm{FOL}}\cup\{c\}$-structure $\mathcal{S}'$, i.e., if we drop $c$ from the signature, $\mathcal{S}'$ can be reduced to an $\mathcal{L}_{\mathrm{FOL}}$-structure $\mathcal{S}$ with

$$\mathcal{S},(x\mapsto c^{\mathcal{S}'})\models\mathrm{ST}_x(\Sigma),$$

By Theorem 2.11, $\mathcal{S}$ defines an $\mathcal{L}_\tau$-model $\mathcal{M}^{\mathcal{S}}$ such that

$$\mathcal{M}^{\mathcal{S}}\models_{c^{\mathcal{S}'}}\Sigma,$$

which is obviously countable. Thus, $\Sigma$ is satisfiable in a countable $\mathcal{L}$-model. $\lhd$

## 2.7 Expressiveness

What is the expressive power of modal logic? What can be expressed by modal formulae? The key idea to investigate such questions is that expressiveness of a formal language can be measured by its power of making distinctions between different "situations". Adapted to our context, we can put it more precisely as follows: Consider Kripke models $\mathcal{M}$ and $\mathcal{M}'$ and states $s$ and $s'$ of $\mathcal{M}$ and $\mathcal{M}'$, respectively. Is there any modal formula by which $s$ and $s'$ can be distinguished?

**Definition 2.13.** The *type* of $s$ in $\mathcal{M}$ is defined as the set

$$T_{\mathcal{M}}(s) := \{\varphi : \mathcal{M} \models_s \varphi\}.$$

States $s$ and $s'$ are said to be *modally equivalent*, $s \equiv_{\mathcal{M},\mathcal{M}'} s'$, if $T_{\mathcal{M}}(s) = T_{\mathcal{M}'}(s')$.

As has been proven in section 2.4, a sufficient criterion for modal equivalence is the existence of a bounded morphism, i.e., if $f$ is a bounded morphism from $\mathcal{M}$ to $\mathcal{M}'$, then for each state $s$ of $\mathcal{M}$,

$$s \equiv_{\mathcal{M},\mathcal{M}'} f(s).$$

However, the converse of this observation does not hold in general.
A somewhat weaker notion than that of a bounded morphism is that of *bisimulation*. We will see that modal equivalence follows from bisimilarity and that there is a fairly wide class of models where the converse of this implication holds, too. For the sake of simplicity we confine to basic modal languages with one modality.

**Definition 2.14.** Let $\mathcal{M} = \langle S, R, V\rangle$ and $\mathcal{M}' = \langle S', R', V'\rangle$ be Kripke models for $\mathcal{L}_{\Diamond}(P)$. A non-void relation $Z \subseteq S \times S'$ is said to be a *bisimulation* between $\mathcal{M}$ and $\mathcal{M}'$ if the following conditions are satisfied:

(a) For all $s, s'$ with $s\,Z\,s'$ and each $p \in P$, $s \in V(p)$ if and only if $s' \in V'(p)$;

(b) If $s\,Z\,s'$ and $s\,R\,t$, then there exists a $t' \in S'$ with $t\,Z\,t'$ and $s'\,R'\,t'$ (*forth condition*);

(c) If $s\,Z\,s'$ and $s'R't'$, then there exists a $t \in S$ with $t\,Z\,t'$ and $s\,R\,t$ (*back condition*).

States $s$ and $s'$ are called *bisimilar*, $s \leftrightsquigarrow_{\mathcal{M},\mathcal{M}'} s'$, if there exists a bisimulation $Z$ between $\mathcal{M}$ and $\mathcal{M}'$ with $s\,Z\,s'$.

Obviously, the notion of bisimulation provides a generalization of the notion of p-morphism: p-morphisms are exactly those bisimulations that are left-total and functional. Before illustrating this new concept by examples, we mention a fundamental statement.

**Proposition 2.15.** *Let $\mathcal{M} = \langle S, R, V\rangle$ and $\mathcal{M}' = \langle S', R', V'\rangle$ be bisimilar Kripke models. Let $s \in S$ and $s' \in S'$ be states such that $s \leftrightsquigarrow_{\mathcal{M},\mathcal{M}'} s'$. Then $s$ and $s'$ are modally equivalent, i.e., for each formula $\varphi$,*

$$\mathcal{M} \models_s \varphi \iff \mathcal{M}' \models_{s'} \varphi.$$

*Proof.* Straight forward, see the proof of Proposition 2.7. $\triangleleft$

In the situation depicted in Figure 2.7, the states $s$ and $s'$ are bisimilar if we assume that in all nodes all propositional variables are true. The bisimulation presented there is a bounded morphism, obviously. But in general, not each bisimulation defines such a morphism. In the situation of Figure 2.8, for example, we can construct step by step a bisimulation from the

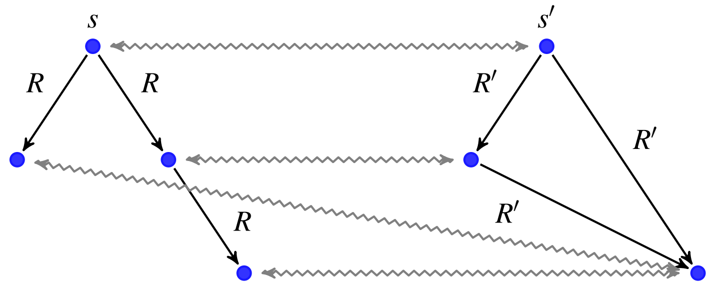


Figure 2.7: Example of a bisimulation

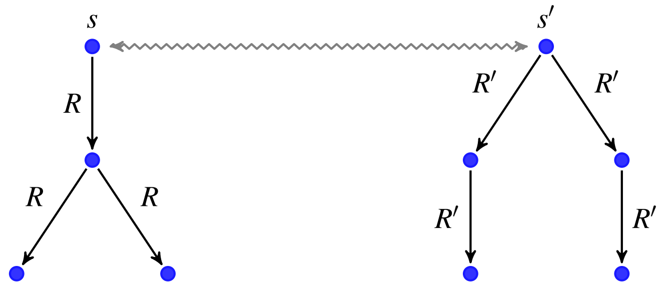


Figure 2.8: Is there a bisimulation with $s\,Z\,s'$?

initial link between states $s$ and $s'$. However, it can be shown that there does not exist any bounded morphism $f$ from the model on the left-hand side to that on the right-hand side with $f(s) = s'$.

A more interesting, but also more technical example is given by a new kind of model construction, namely by the method of "unraveling" a Kripke model. More precisely, each Kripke model can be shown to be modally equivalent to a rooted tree-like Kripke model. To see this, let $\mathcal{M} = \langle S, R, V \rangle$ be an arbitrary Kripke model. Let $s_0 \in S$ be a fixed state. Define a tree-like model (with $s_0$ as its root), $\mathcal{M}_{s_0}^{\text{tree}} = \langle S', R', V' \rangle$, by

$$\begin{aligned} S' &:= \left\{ \bar{s} \,:\, \bar{s} \text{ is a finite } R\text{-chain of length } n_{\bar{s}} > 0 \text{ with } \bar{s}[1] = s_0 \right\} \\ R' &:= \left\{ (\bar{s}, \bar{t}) \in S'^2 \,:\, \bar{s} = (s_0, \ldots, s_i, \ldots, s_n), \bar{t} = (s_0, \ldots, s_i, \ldots, s_n, s_{n+1}) \right\} \\ V'(p) &:= \left\{ \bar{s} \in S' \,:\, \bar{s} = (s_0, \ldots, s_n) \in S' \text{ and } s_n \in V(p) \right\} \end{aligned}$$

Then the relation

$$Z := \left\{ \left(s, \bar{t}\right) \in S \times S' \,:\, \bar{t} = (s_0, \ldots, s_n) \ \text{ and } s_n = s \right\}$$

defines a bisimulation between $\mathcal{M}$ and $\mathcal{M}_{s_0}^{\text{tree}}$. More exactly, the mapping

$$f : \mathcal{M}_{s_0}^{\text{tree}} \to \mathcal{M}, \quad (s_0, \ldots, s_n) \mapsto s_n$$

defines a bounded morphism from $\mathcal{M}_{s_0}^{\text{tree}}$ onto $\mathcal{M}$. Figure 2.9 depicts a simple example.

In general, the converse of Proposition 2.15 does not hold (see exercises). However, if we restrict consideration on image-finite models, then it is possible to establish that bisimilarity follows from modal equivalence.

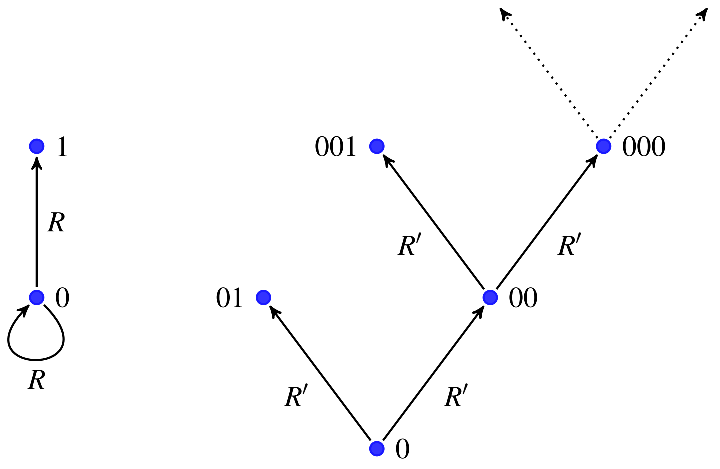


Figure 2.9: Unraveling a simple Kripke model

**Definition 2.15.** Let $\mathcal{M} = \langle S, R, V\rangle$ be a Kripke model. $\mathcal{M}$ is said to be *image-finite* if for each $s \in S$,

$$sR := \{s' \in S : sRs'\}$$

is finite.

Note that each finite model is image-finite.

**Theorem 2.16** (Hennessy-Milner). *Let $\mathcal{M}$ and $\mathcal{M}'$ be image-finite Kripke models. Then for each state $s$ of $\mathcal{M}$ and each state $s'$ of $\mathcal{M}'$,*

$$s \equiv_{\mathcal{M},\mathcal{M}'} s' \iff s \leftrightsquigarrow_{\mathcal{M},\mathcal{M}'} s'.$$

*Proof.* The direction from right to left follows from Proposition 2.15. For the other direction we show that the relation

$$Z := \left\{ (s,s') \in S \times S' \,:\, s \equiv_{\mathcal{M},\mathcal{M}'} s' \right\}$$

defines a bisimulation between $\mathcal{M}$ and $\mathcal{M}'$. Condition (a) of Definition 2.14 is clear. For condition (b) let us suppose that $s \equiv_{\mathcal{M},\mathcal{M}'} s'$ and that $sRt$. Since $\mathcal{M}'$ is image-finite, we may assume that

$$s'R' = \{s'_0, \dots, s'_n\}$$

with $s'_0, \dots, s'_n \in S'$. Note that $s'R'$ is not empty: otherwise, $\mathcal{M} \models_{s'} \Box\bot$, hence by $s \equiv_{\mathcal{M},\mathcal{M}'} s'$, $\mathcal{M} \models_s \Box\bot$ and thus, by $sRt$, $\mathcal{M} \models_t \bot$. Assume now for proof by contradiction that there does not exist any $t' \in S'$ with $s'R't'$ and $t \equiv_{\mathcal{M},\mathcal{M}'} t'$, i.e., for each $s'_i \in s'R'$, there is a formula $\varphi_i$ such that

$$\mathcal{M} \models_t \varphi_i \quad \text{and} \quad \mathcal{M}' \not\models_{s'_i} \varphi_i.$$

Then it follows that

$$\mathcal{M} \models_s \Diamond(\varphi_0 \wedge \cdots \wedge \varphi_n),$$

and hence, by $s \equiv_{\mathcal{M},\mathcal{M}'} s'$,

$$\mathcal{M}' \models_{s'} \Diamond(\varphi_0 \wedge \cdots \wedge \varphi_n).$$

Thus, there exists an $s'_i \in s' R'$ with

$$\mathcal{M}' \models_{s'_i} \varphi_0 \wedge \cdots \wedge \varphi_n$$

— in contradiction to $\mathcal{M}' \not\models_{s'_i} \varphi_i$. $\lhd$

Thus, we finally have to give an answer to the question of how the modal fragment can be characterized. Or, to put the question in other words: Which first order formulae are equivalent to the standard translation of some modal formula?

**Definition 2.16.** Let $\varphi(x)$ be a $\mathcal{L}_{\text{FOL}}$-formula in which exactly one variable $x$ occurs free. $\varphi(x)$ is said to be *invariant under bisimulation* if for each pair of Kripke models $\mathcal{M}$ and $\mathcal{M}'$, each pair of possible state $s$ and $s'$ of $\mathcal{M}$ and $\mathcal{M}'$, respectively, and each bisimulation $Z$ between $\mathcal{M}$ and $\mathcal{M}'$ with $s\,Z\,s'$, it holds

$$\mathcal{M}_{\text{FOL}}, x \mapsto s \models \varphi(x) \iff \mathcal{M}'_{\text{FOL}}, x \mapsto s' \models \varphi(x).$$

Invariance under bisimulation, then, is a criterion to characterize the modal fragment.

**Theorem 2.17** (van Benthem)**.** *Let $\varphi(x)$ be a $\mathcal{L}_{\text{FOL}}$-formula in which exactly one variable $x$ occurs free. Then the following statements are equivalent:*

(a) *$\varphi(x)$ is equivalent to the standard translation $\text{ST}_x(\psi)$ of some $\mathcal{L}_\tau(P)$-formula $\psi$.*

(b) *$\varphi(x)$ is invariant under bisimulation.* $\lhd$

In this context it is worthwhile to discuss a game-theoretical characterization of bisimulation. Let $\mathcal{M} = \langle S, R, V \rangle$ and $\mathcal{M} = \langle S', R', V' \rangle$ be Kripke models, and let $s_0$ and $s'_0$ be states of $\mathcal{M}$ and $\mathcal{M}'$, respectively. The *bisimulation game* is then defined as follows:

- There are two players $A$ ('attacker', 'spoiler') and $D$ ('defender', 'duplicator'). $A$ seeks to show that there does not exist any bisimulation $Z$ between $\mathcal{M}$ and $\mathcal{M}'$ with $s_0\,Z\,s'_0$, while $D$ tries to prevent this.
- The game is played in rounds (where the number of rounds, $n$, is fixed before the game is started). There are two variants: the finite game and the infinite one (i.e., the game is played over infinitely many rounds).
- If $s_0$ and $s'_0$ differ in the interpretation of at least one propositional variable, then $A$ wins the game.
- In the first round $A$ chooses in one of the models $\mathcal{M}$ or $\mathcal{M}'$ a successor of the current state, i.e., an $R$-successor $s_1$ of $s_0$ or an $R'$-successor $s'_1$ of $s'_0$. If $A$ is not able to find such a successor, then $A$ loses the game. If $A$ finds a successor, then $D$ has to choose a successor of the current state in the *other* model. For example, $D$ has to choose an $R$-successor $s_1$ of $s_0$ provided that $A$ has chosen an $R'$-successor $s'_1$ of $s'_0$. If $D$ is not able to find a successor, she will lose the game. If both $A$ and $D$ were able to find successors and if these successors differ in the interpretation of at least one propositional variable, then $A$ wins the game. Otherwise we continue the game as in round 1, but with the pair of current states $(s_0, s'_0)$ replaced by the new pair of successor states $(s_1, s'_1)$.
- In the finite game, i.e., in the game over a given number $n$ of rounds, if $A$ has not won the game after $n$ rounds, then $A$ loses the game. In the infinite game, if $A$ has not won the game after a finite number of rounds, $A$ loses the game.

- Thus, while the game runs, (possibly infinite) sequences of states are constructed in both models. In each round, the defender is to imitate the behavior of the attacker in the respective other model.
- Let $G^n_{\mathcal{M},\mathcal{M}'}(s,s')$ denote the finite game over $n$ rounds that starts at states $s$ and $s'$. The infinite game starting at $s$ and $s'$ is denoted by $G^\infty_{\mathcal{M},\mathcal{M}'}(s,s')$.

It is obvious that the bisimulation game (the finite one as well as the infinite one) is determined, i.e., exactly one of both players has a strategy for winning the game. A *strategy* is a function that tells each player for each finite sequence $(s_0,s'_0),\dots,(s_n,s'_n)$ of moves in the game how she/he has to continue the game. For the attacker, the strategy says which model and successor is to be selected; for the defender, the strategy instructs how to respond to the attacker's move.

We illustrate these basic concepts of bisimulation games by discussing some examples.

**Example 2.17.** Consider the models given in Figure 2.10. If in the first round of $G^1_{\mathcal{M},\mathcal{M}'}(s_0,s'_0)$, $A$ chooses $s_1$ in the left model, then $D$ is able to imitate this by choosing $s'_1$, i.e., $D$ will win that game. But, if $A$ chooses $s'_2$ in the right model, then $D$ has to choose $s_1$, and thus $D$ loses this game. Thus, $A$ has a strategy for winning the game, while $D$ has not.

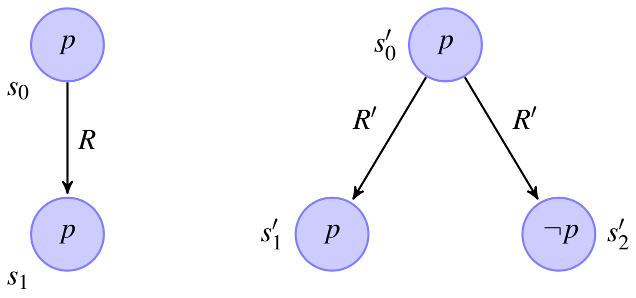


Figure 2.10: A bisimulation game: $A$ can easily win the game

**Example 2.18.** Let us now consider the situation presented in Figure 2.11. Here $A$ has various strategies for winning $G^2_{\mathcal{M},\mathcal{M}'}(s_0,s'_0)$. For example, if $A$ chooses in the model on the right side $s'_3$, then $D$ has to choose $s_1$. Then $A$ chooses $s_2$ (note that $A$ changes the model). Thus, $D$ has to choose $s'_4$, but then she loses the game. Another strategy: In the first round, $A$ chooses $s_1$ in the model on the left side. Then $D$ has to choose between $s'_1$ and $s'_3$. But in each of both cases $A$ can win the game by choosing $s_3$ or $s_2$, respectively.

Without proof we now present the following theorems:

**Theorem 2.18** (see, e.g., Goranko and Otto (2007))**.** *Let $\mathcal{L}_\Diamond(P)$ be a basic modal logic with finitely many propositional variables. Let $\mathcal{M} = \langle S,R,V\rangle$ and $\mathcal{M}' = \langle S',R',V'\rangle$ be Kripke models and let $s$ and $s'$ be states of $\mathcal{M}$ and $\mathcal{M}'$, respectively. Then the following statements are equivalent:*

(a) *$D$ has a strategy for winning $G^n_{\mathcal{M},\mathcal{M}'}(s,s')$.*

(b) *For all formulae $\varphi$ of $\mathcal{L}_\tau(P)$ with* depth $\varphi \le n$,

$$\mathcal{M} \models_s \varphi \iff \mathcal{M}' \models_{s'} \varphi \qquad \lhd$$

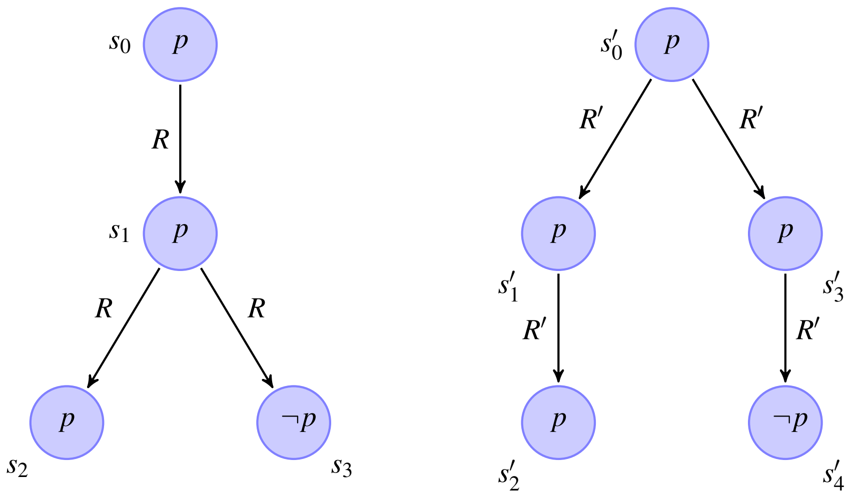


Figure 2.11: Another example of a bisimulation game: $A$ can win again

**Theorem 2.19** (see, e.g., Goranko and Otto, 2007)**.** *Let $\mathcal{M} = \langle S, R, V \rangle$ and $\mathcal{M} = \langle S', R', V' \rangle$ be Kripke-models, let $s$ be a state of $\mathcal{M}$, and let $s'$ be state $\mathcal{M}'$. Then the following statements are equivalent:*

(a) *$D$ has a strategy for winning $G^{\infty}_{\mathcal{M},\mathcal{M}'}(s, s')$.*

(b) *There exists a bisimulation $Z$ between $\mathcal{M}$ and $\mathcal{M}'$ with $s\,Z\,s'$.* $\lhd$

## Bibliographic remarks

A general reference for the material presented in this chapter is Blackburn et al. (2002), chapter 1 and chapter 2.

Epistemic modal logics were presumably first discussed by von Wright (1951) and further developed by Hintikka (1962). For an introduction to the philosophical foundations of epistemic logics see, for example, Lenzen (1981) and for an introduction to epistemic logics and its computational aspects we refer to Fagin et al. (1995).

Temporal logic (also known as *tense logic*) has been first studied by Prior (1957, 1967). An important field of application of temporal logic in computer science, is the domain of model checking, i.e. the formal verification of system or software properties. The first logics studied in this context are LTL (*Linear-time Temporal Logic*) and CTL (*Computation Tree Logic*) (see, e.g., Pnueli, 1977; Clarke et al., 1986; Emerson, 1990). A good starting point for further reading is the Stanford Encyclopedia of Philosophy on temporal logic (Galton, 2008).

Propositional dynamic logic (PDL) is one of the logics discussed in the context of *dynamic logics*, that is, logics for reasoning about computer programs: Pratt (1976) may count as a starting point in this research direction. PDL itself was introduced by Fischer and Ladner (1977).

# 3 Normal Modal Logics, Frame Classes, and Definability

In this section we introduce the concept of normal modal logics. We will discuss how classes of frames can be defined by axioms, how modal logics can be axiomatized, and a standard procedure for proving soundness and completeness of such axiomatizations. Moreover, we will learn more on special uni-modal logics.

## 3.1 Normal modal logics

In what follows we will confine to basic modal languages with a countable set of propositional variables. Often we even confine to modal languages with a single modality. Almost all concepts can be generalized to arbitrary modal languages in a straightforward way.

We start by introducing a rather general concept of modal logics.

**Definition 3.1.** A *modal logic* is a set of $\mathcal{L}_\tau(P)$-formulae, $\Lambda$, that contains all propositional tautologies and is closed under *modus ponens* and (uniform) substitution, that is,

(a) If $\varphi \in \Lambda$ and $\varphi \to \psi \in \Lambda$, then $\psi \in \Lambda$.

(b) If $\varphi \in \Lambda$ and $\sigma : P \to \mathcal{L}_\tau(P)$ is a substitution, then $\varphi^\sigma \in \Lambda$.

A modal logic $\Lambda$ is said to be *normal* if furthermore the following conditions are satisfied:

(c) $\Lambda$ contains all valid $\mathcal{L}_\tau(P)$-formulae.

(d) $\Lambda$ is closed under *necessitation* for each $\tau$-modality $\Diamond$, i.e., if $\varphi \in \Lambda$, then $\Box\varphi \in \Lambda$.

Recall the small list of valid formulae presented in Lemma 2.4. All these formulae are contained in any *normal* modal logic, but not in every modal logic.

**Remark 3.1.** In the case of non-basic modal languages condition (d) in Definition 3.1 needs to be adapted. More precisely, (d) has to be replaced by

(d) If $\varphi \in \Lambda$, then $\triangledown(\bot, \dots, \varphi, \dots, \bot) \in \Lambda$.

Notice that no special requirements are imposed for 0-ary modalities.

Instead of conditions like “all propositional tautologies” or “all valid formulae”, one may resort to appropriate axiomatizations. For example, in order to guarantee that $\Lambda$ contains all propositional tautologies (see Definition 3.1), it suffices to require that all formulae of an axiomatization of propositional logic are included in $\Lambda$. In a similar way, for normal logics it is sufficient that $\Lambda$ contains all formulae of a suitable axiomatization of the set of formulae that are valid in all models. As we will later see, for basic modal languages the following axioms suffice (see also Lemma 2.4):

**K** $\quad \Box(p \to q) \to (\Box p \to \Box q)$

**Dual** $\quad \Diamond p \leftrightarrow \neg\Box\neg p$

Notice that in a setting in which the boxes $\Box$ are chosen as primitives, axiom (Dual) is not needed.

More generally, axiom **K** can be replaced by:

$\mathbf{K}^i_\triangledown$ $\quad \triangledown(\dots, p \to q, \dots) \to (\triangledown(\dots, p, \dots) \to \triangledown(\dots, q, \dots))$

It is easy to see that formulae of this form are valid in all models.

Some important examples of modal logics and normal modal logics are presented in the following list.

**Example 3.1.** (a) The set of all $\mathcal{L}_\tau$-formulae is a normal modal logic, *the inconsistent (normal) modal logic*.

(b) If $\{\Lambda_i\}_{i\in I}$ is a family of normal modal logics, then the intersection of all these logics $\bigcap_i \Lambda_i$ is a normal modal logic. In particular, for each pair of normal modal logics, $\Lambda_1$ and $\Lambda_2$, $\Lambda_1 \cap \Lambda_2$ is a normal modal logic.

(c) Given a single frame $\mathcal{F}$ or a class of frames, $\mathcal{C}$, the sets

$$\Lambda(\mathcal{F}) := \{\varphi \in \mathcal{L} : \mathcal{F} \models \varphi\}$$
$$\Lambda(\mathcal{C}) := \{\varphi \in \mathcal{L} : \mathcal{F} \models \varphi \text{ for each } \mathcal{F} \text{ in } \mathcal{C}\}$$

are normal modal logics, the *normal modal logics of* $\mathcal{F}$ and $\mathcal{C}$, respectively. Recall that $\varphi$ is valid in a frame $\mathcal{F}$, $\mathcal{F} \models \varphi$, if it is valid in each model defined on that frame.

(d) Given a set of $\mathcal{L}$-formulae $\Gamma$, there exists a smallest normal modal logic that contains $\Gamma$ as a subset. This logic is called the *normal modal logic axiomatized by* $\Gamma$ and is denoted by $\Lambda(\Gamma)$.

**Remark 3.2.** Are there "modal logics" that are not normal? Of course, the smallest modal logic is not a normal modal logic (since it does not contain axiom **K** or its variants). In fact, the smallest modal logic is just propositional logic on top of a language with some "strange" propositional variables (namely formulae of the form $\Diamond\varphi$ or $\triangle\varphi$). But there are also more interesting non-normal modal logics, *quasi-normal modal logics*, *regular modal logics*, etc. Quasi-normal modal logics are modal logics that contain all theorems of **K**, but in which the application of the necessitation rule is somehow restricted (for example the Lewis systems **S2** and **S3** are quasi-normal). The main semantic difference is that in such systems validity of a formula is evaluated as truth in some selection of *normal states*. In regular modal logics the necessitation rule is replaced by the weaker rule: If $\varphi \to \psi \in \Lambda$, then $\Box\varphi \to \Box\psi \in \Lambda$.

**Remark 3.3.** We have introduced normal modal logics that are defined in terms of frames or frame classes and normal modal logics defined in terms of "axioms". Both concepts should be kept apart carefully. In what follows we will study the connection between these concepts in more detail: for example, which normal modal logics (that is, a set of formulae) can be axiomatized by a set of formulae ($\Lambda = \Lambda(\Gamma)$ for some set of formulae $\Gamma$?) and which normal modal logics are the normal logics defined by some (frame or) class of frames ($\Lambda = \Lambda(\mathcal{C})$ for some frame class $\mathcal{C}$?).

For basic modal languages $\mathcal{L}$ with exactly one modality, the smallest normal modal logic is called **K**, for basic modal languages with $m$ diamonds, the smallest normal modal logic is called $\mathbf{K}(m)$. In the general, polyadic case the smallest normal modal logic is called $\mathbf{K}_\tau$, where $\tau$ is the modal similarity type of the considered modal language. All these logics are the normal modal logics that are axiomatized by the empty set.
Further normal modal logics can be generated on top of **K** if we consider the modal logics axiomatized by additional formulae. Prominent examples are the "axioms":

**T** $\quad \Box p \to p \quad$ (alternatively: $p \to \Diamond p$)

| | |
|---|---|
| **4** | $\Box p \to \Box\Box p$ (alternatively: $\Diamond\Diamond p \to \Diamond p$) |
| **E** | $\Diamond p \to \Box\Diamond p$ (alternatively: $\Diamond\Box p \to \Box p$) |
| **B** | $p \to \Box\Diamond p$ (alternatively: $\Diamond\Box p \to p$) |
| **D** | $\Box p \to \Diamond p$ |
| **.3** | $\Diamond p \wedge \Diamond q \to \Diamond(p \wedge \Diamond q) \vee \Diamond(p \wedge q) \vee \Diamond(\Diamond p \wedge q)$ |
| **L** | $\Box(\Box p \to p) \to \Box p$ |

Then, for example, the smallest normal modal logic that contains axiom **T** is called **KT**, the smallest normal modal logic axiomatized by $\{\mathbf{T}, \mathbf{B}\}$ is called **KTB**, etc. Prominent examples are the logics **K4**, **KB**, **KBE**, **KD4**, **KDB**, **KL**. Even more prominent logics have a special name such as **S4** (= **KT4**) or **S5** (= **KTE** = **KT4B**). The latter example shows that, in general, the set of axioms used to generate the normal modal logic is not unique. Finally, some modal logics such as **S4** are the basis for new modal logics, for example, **S4.1**, **S4.3**, etc. Some of them will be studied later in more detail.

**Remark 3.4.** In general, if $\Gamma \subseteq \Gamma'$, then $\Lambda(\Gamma) \subseteq \Lambda(\Gamma')$. Thus, for some of the above mentioned modal logics it is quite easy to see whether they are contained in another normal modal logic. More generally, it holds that if $\Gamma \subseteq \Lambda(\Gamma')$, then $\Lambda(\Gamma) \subseteq \Lambda(\Gamma')$. Hence it suffices to show that all the axioms in $\Gamma$ are contained in $\Lambda(\Gamma')$ (i.e., they are "derivable" from $\Gamma'$ using an axiomatization of **K** and the "axioms" in $\Gamma'$).

**Remark 3.5.** Notice that adding axioms to system **K** may even result in the inconsistent logic. An example is the axiom set $\{\mathbf{T}, \mathbf{L}\}$, i.e., the modal logic **KTL** coincides with the inconsistent modal logic.

## 3.2 Kripke frames and definability

What makes the relational semantics interesting is the close connection between valid "axioms" and formal properties of the accessibility relation in Kripke frames. In this section we will throw a first glance on this connection. For the sake of simplicity, we will consider unimodal languages. Then it is convenient to talk about reflexive, transitive, etc., Kripke frames, when the accessibility relation has the respective property.

**Proposition 3.1.** *A Kripke frame $\mathcal{F}$ is reflexive if and only if axiom **T** is valid in $\mathcal{F}$.*

*Proof.* Assume first that $\mathcal{F} = \langle S, R\rangle$ is a reflexive frame, and let $\mathcal{M}$ be an arbitrary Kripke model defined on $\mathcal{F}$. For each $s \in S$, if $\mathcal{M} \models_s p$, then by reflexivity, there exists a state $s' \in S$ with $s\,R\,s'$ and $\mathcal{M} \models_{s'} p$, namely state $s$. Hence $\mathcal{M} \models_s \Diamond p$. This shows that **T** is satisfied in each state in each model defined on $\mathcal{F}$.
For the other direction, assume that **T** is valid in $\mathcal{F}$. We have to show that $\mathcal{F}$ is reflexive, i.e., $s\,R\,s$ for each $s \in S$. For an arbitrarily fixed $s \in S$ define a model $\mathcal{M}_s$ on $\mathcal{F}$ with $V(p) = \{s\}$. Since **T** is valid in $\mathcal{F}$, it follows that $\mathcal{M}_s \models_s p \to \Diamond p$. Because of $\mathcal{M}_s \models_s p$, it follows $\mathcal{M}_s \models_s \Diamond p$. Hence there exists an $s' \in S$ with $s\,R\,s'$ and $\mathcal{M}_s \models_{s'} p$. Then it follows $s = s'$ and hence $s\,R\,s$. $\lhd$

**Proposition 3.2.** *A Kripke frame $\mathcal{F}$ is transitive if and only if axiom **4** is valid in $\mathcal{F}$.*

*Proof.* Assume first that $\mathcal{F} = \langle S, R\rangle$ is a transitive frame, and let $\mathcal{M}$ be a Kripke model defined on $\mathcal{F}$. For each $s \in S$, if $\mathcal{M} \models_s \Diamond\Diamond p$, then there exist states $s', s''$ such that $s\,R\,s'$, $s'\,R\,s''$, and

$\mathcal{M} \models_{s''} p$. By transitivity of $R$, it follows $s\,R\,s''$ and hence $\mathcal{M} \models_s \Diamond p$. This shows that **4** is satisfied in each state in each model defined on $\mathcal{F}$.
For the other direction, assume that **4** is valid in $\mathcal{F}$. We have to show that $\mathcal{F}$ is transitive, i.e., for all $s, s', s'' \in S$, $s\,R\,s'$ and $s'\,R\,s''$ implies $s\,R\,s''$. For arbitrary such $s, s', s''$ define a model $\mathcal{M}$ on $\mathcal{F}$ with $V(p) = \{s''\}$. Then it holds $\mathcal{M} \models_{s''} p$, hence $\mathcal{M} \models_{s'} \Diamond p$, and consequently, $\mathcal{M} \models_s \Diamond\Diamond p$. Since **4** is valid in $\mathcal{F}$ and thus in $\mathcal{M}$, it follows $\mathcal{M} \models_s \Diamond p$. Thus there exists a state $s'''$ with $s\,R\,s'''$ and $\mathcal{M} \models_{s'''} p$. Since $p$ is true in $s''$ only, it follows $s'' = s'''$ and hence $s\,R\,s''$ — as to be shown. $\lhd$

As indicated in Example 2.16, a different proof of the back direction could be as follows: consider $s, s', s'' \in S$, with $s\,R\,s'$ and $s'\,R\,s''$. Define a model $\mathcal{M}$ on $\mathcal{F}$ by $V(p) := S \setminus R[s]$. Then it follows $\mathcal{M} \not\models_s \Diamond p$ (otherwise there would be an $s''' \in S$ with $s\,R\,s'''$ and $\mathcal{M} \models_{s'''} p$, and hence $s''' \in R[s]$ and $s''' \in V(p)$ — in contradiction to the definition of $V(p)$). Since **4** is valid in $\mathcal{M}$, we obtain $\mathcal{M} \not\models_s \Diamond\Diamond p$. Because of $s\,R\,s'$, we can conclude $\mathcal{M} \not\models_{s'} \Diamond p$, and thus $\mathcal{M} \not\models_{s''} p$. By definition of $V(p)$, we obtain $s'' \in R[s]$, i.e., $s\,R\,s''$ — as to be shown.

**Proposition 3.3.** *A Kripke frame $\mathcal{F}$ is Euclidean if and only if axiom **E** is valid in $\mathcal{F}$.*

*Proof.* Exercise. $\lhd$

**Proposition 3.4.** *A Kripke frame $\mathcal{F}$ is symmetric if and only if axiom **B** is valid in $\mathcal{F}$.*

*Proof.* Exercise. $\lhd$

**Proposition 3.5.** *A Kripke frame $\mathcal{F}$ is serial if and only if axiom **D** is valid in $\mathcal{F}$.*

*Proof.* Exercise. $\lhd$

All these propositions claim that certain classes of frames are *definable* by formulae.

**Definition 3.2.** Let $\varphi$ be an $\mathcal{L}_\tau$-formula, and let $\mathcal{C}$ and $\mathcal{D}$ be a classes of $\tau$-frames.

(a) $\varphi$ is said to *define* (or: *characterize*) $\mathcal{D}$ *within* $\mathcal{C}$ if for each $\tau$-frame in $\mathcal{C}$, $\mathcal{F}$ is in $\mathcal{D}$ if and only if $\mathcal{F} \models \varphi$.

(b) $\varphi$ is said to *define* (or: *characterize*) $\mathcal{C}$ if $\varphi$ defines $\mathcal{C}$ within the class of all $\tau$-frames.

(c) In a similar way, we say that a set of $\mathcal{L}_\tau$-formulae $\Delta$ defines (or characterizes) a class of frames (possibly within some other class of frames).

So what we have proven above is that **T** defines the class of reflexive frames, **4** the class of transitive frames, **B** the class of symmetric frames, **E** the class of Euclidean frames, etc.

But what about the following axioms? Do similar "correspondence" results hold there as well?

| | |
|---|---|
| $\mathbf{T}_c$ | $p \to \Box p$ |
| **Triv** | $p \leftrightarrow \Box p$ |
| $\mathbf{E}_1$ | $\Box\Diamond\Box p \to p$ |
| $\mathbf{E}_2$ | $\Diamond\Box p \to \Box\Diamond\Box\Box p$ |
| $\mathbf{D}_c$ | $\Diamond p \to \Box p$ |

| | |
|---|---|
| **D1** | $\Box(\Box p \to q) \lor \Box(\Box q \to p)$ |
| **F** | $(\Box\Diamond p \land \Box\Diamond q) \to \Diamond(p \land q)$ |
| **M** | $\Box\Diamond p \to \Diamond\Box p$ |
| **G1** | $\Diamond\Box p \to \Box\Diamond p$ |
| **BM** | $p \to \Box\Diamond\Diamond p$ |
| **TM** | $\Diamond\Box p \to \Diamond p$ |
| **Lem$_0$** | $\Box(p \land \Box p \to q) \lor \Box(q \land \Box q \to p)$ |
| **H** | $\Box(\Box p \leftrightarrow p) \to \Box p$ |
| **Alt$_n$** | $\Box p_1 \lor \Box(p_1 \to p_2) \lor \cdots \lor \Box(p_1 \land \cdots \land p_n \to p_{n+1})$ |
| **N1** | $\Box(\Box(p \to \Box p) \to p) \to (\Diamond\Box p \to p)$ |
| **H1** | $p \to \Box(\Diamond p \to p)$ |
| **G$_0$** | $\Diamond(p \land \Box q) \to \Box(p \lor \Diamond q)$ |
| **J1** | $\Box(\Box(p \to \Box p) \to p) \to p$ |
| | ... |

Candidate frame properties are the properties mentioned in Definition 2.1 and the following:

(a) The frame consists of a single reflexive state.

(b) The frame consists of single dead end.

(c) The frame consists of a single state.

(d) In the frame each state sees at most one state.

(e) In the frame each state sees at most $n$ worlds.

(f) In the frame each state sees a dead end.

(g) The frame is convergent.

(h) The frame is universal.

(i) The frame is finite, reflexive, transitive, and symmetrical.

(j) The frame is finite, irreflexive, and transitive.

(k) ...

Similar to the first-order language considered for the standard translation of a modal language, we can define the *first-order frame language* of a modal similarity type $\tau$, $\mathcal{L}_{\mathrm{FOL}}(\tau)$: just drop the unary relation symbols introduced for the translation of propositional letters of $\mathcal{L}_\tau(P)$. This language is also called the *first-order correspondence language* for $\tau$. Given a modal formula $\varphi$ and a formula $\alpha$ of the frame language, $\varphi$ and $\alpha$ are called *frame correspondents* if the same class of frames is defined by $\varphi$ and $\alpha$, that is $\mathcal{F} \models \varphi \iff \mathcal{F} \models \alpha$ for each $\tau$-frame $\mathcal{F}$ ("$\mathcal{F} \models \varphi$" means that the modal formula is *valid* in $\mathcal{F}$, while "$\mathcal{F} \models \alpha$" means that $\alpha$ is true in $\mathcal{F}$ when $\mathcal{F}$ is conceived of as a first-order structure for the language $\mathcal{L}_{\mathrm{FOL}}(\tau)$). The results presented above then can be restated as follows: **T** is the frame correspondent of $\forall x xRx$, **B** the frame correspondent of $\forall xy(xRy \to yRx)$, etc.

**Remark 3.6.** Note that there exist modal formulae that do not have a frame correspondent in the first-order frame language, but only in some higher-order frame language (for example, the PDL axiom $[\pi^*](p \to [\pi]p) \to (p \to [\pi^*]p)$). Another example is the McKinsey formula **M** $\Box\Diamond p \to \Diamond\Box p$ that does not have a correspondent in the first-order frame language (which can be proven by a particular version of the Löwenheim-Skolem theorem).

We finally add an example that provides a relative definability result:

**Proposition 3.6.** *In the class of reflexive and transitive frames,* **M** *defines the "finality condition":*

$$\forall s \exists s'(s\,R\,s' \wedge \forall s''(s'\,R\,s'' \to s' = s'')).$$

*Proof (sketch).* Let $\mathcal{F}$ be any frame that satisfies the finality condition. Let $s$ be some state in an arbitrary model $\mathcal{M}$ on $\mathcal{F}$ such that $\mathcal{M} \models_s \Box\Diamond p$. Since $s$ sees a final state $s'$ (that can see at most itself), it holds $\mathcal{M} \models_{s'} \Diamond p$, i.e., there is a state $s''$ with $s'\,R\,s''$ and $\mathcal{M} \models_{s''} p$. Now $s'$ is final, hence $s' = s''$, hence $\mathcal{M} \models_{s'} p$, and moreover $\mathcal{M} \models_{s'} \Box p$, which shows that $\mathcal{M} \models_s \Diamond\Box p$.
For the other direction let $\mathcal{F}$ be a reflexive and transitive frame in which **M** is valid. For proof by contradiction, assume that the frame does not satisfy the finality condition. Hence there is a state $s_0$ in $\mathcal{F}$ that cannot see a final state. $s_0$ itself cannot be a final state, since otherwise, by reflexivity, $s_0$ could see a final state (namely itself). But then there must be a state $s_1 \neq s_0$ with $s_0\,R\,s_1$ and this $s_1$ cannot be a final state (as otherwise $s_0$ could see a final state), either. But then again $s_1$ can see a different non-final state $s_2$ (possibly $s_1$), and so on.
Notice that not only $s_0$ cannot see a final state, but by transitivity, also all its $R$-successors cannot. Define now a model on that frame in such a way that each state that is seen by $s_0$ sees a state in which $p$ is true and sees another state in which $p$ is not true. Then $\mathcal{M} \models_{s_0} \Box\Diamond p$ and $\mathcal{M} \not\models_{s_0} \Diamond\Box p$. $\lhd$

As it is shown in Blackburn et al. (2002), the reflexivity condition is not needed. That is, **M** also defines the finality condition within the class of transitive frames.

## 3.3 Proving theorems

So far, we have studied modal logics primarily from a semantic point of view. In what follows we provide some proof-theoretic notions. For the sake of simplicity, we consider a fixed modal language $\mathcal{L}_\tau(P)$ with one diamond. Not exactly: for a smooth presentation we will switch to a different set of logical primitives: we take $\{\neg, \to\}$ as propositional basis and assume that $\Diamond$ is defined by $\Box$ (it is quite usual to consider only negation, implication, and universal quantification in Hilbert-style deduction systems).
In such a setting the following axioms together with the *modus ponens* rule and the rule of uniform substitution provide a *sound* and *complete* axiomatization of propositional logic.

**PL1** $\quad p \to (q \to p)$

**PL2** $\quad (p \to (q \to r)) \to ((p \to q) \to (p \to r))$

**PL3** $\quad (\neg p \to \neg q) \to (q \to p)$

This means: a formula is a propositional tautology if and only if there exists a proof for it in this axiomatization. A *proof* is a finite sequence of formulae $\varphi_1, \ldots, \varphi_n$ in which each formula is an axiom or the result of applying one of the rules on previous formulae in the sequence. For propositional logic, modus ponens and the substitution rule are often written in the form:

**MP** $\quad \vdash \varphi,\ \vdash \varphi \to \psi \implies \vdash \psi$

**US** $\vdash \varphi \implies \vdash \varphi^{\sigma}$, where $\sigma$ is a substitution.

In a sequence of formulae $\varphi_1, \dots, \varphi_n$, $\varphi_n$ is a result of applying **MP** on the formulae $\varphi_i$ and $\varphi_j$ $(i, j < n)$ if $\varphi_j$ is the formula $(\varphi_i \to \varphi_n)$. And, $\varphi_n$ is a result of applying **US** on formula $\varphi_i$ $(i < n)$ if there exists a substitution $\sigma$ such that $\varphi_n = \varphi_i^{\sigma}$.

**Definition 3.3.** Let $\Gamma$ be a set of $\mathcal{L}_\tau$-formulae. As $K(\Gamma)$*-axioms* we refer to all $\mathcal{L}_\tau$-formulae that are equal to formula **K** or a formula in $\Gamma$. $K(\Gamma)$*-rules* are **MP**, **US**, and the necessitation rule:

**R-□** $\vdash \varphi \implies \vdash \Box\varphi$

A $K(\Gamma)$*-proof* is a finite sequence of $\mathcal{L}_\tau$-formulae $\varphi_1, \dots, \varphi_n$ in which each formula is a propositional tautology, a $K(\Gamma)$-axiom, or the result of applying one of the $K(\Gamma)$-rules on previous formulae in the sequence. A formula $\varphi$ is called a $K(\Gamma)$*-theorem*, written $\vdash_{K(\Gamma)} \varphi$, if there exists a $K(\Gamma)$-proof $\varphi_1, \dots, \varphi_n$ with $\varphi_n = \varphi$.

**Remark 3.7.** Notice that proofs are finite sequences and hence only finitely many axioms occur in each proof. If $\Gamma$ is the empty set, we will refer to $K(\emptyset)$-proofs as **K**-proofs. In the context of the normal modal logics introduced in section 3.1 we will use concepts like **KD**-axiom, **KT**-proof, **S4**-theorem, etc. to indicate that the respective "characteristic axiom" of that modal logic may occur as an axiom in a proof.

**Remark 3.8.** In the chosen set of modal connectives $\Diamond$ is introduced by definition. That is, $\Diamond\varphi$ is just an abbreviation of $\neg\Box\neg\varphi$. Hence **Dual** $(\Diamond p \leftrightarrow \neg\Box\neg p)$ is a trivial **K**-theorem.

Following we list some **K**-theorems and "derived rules".

**Lemma 3.7.**

(a) $\vdash_{\mathbf{K}} \varphi \to \psi \implies \vdash_{\mathbf{K}} \Box\varphi \to \Box\psi$

(b) $\vdash_{\mathbf{K}} \varphi \leftrightarrow \psi \implies \vdash_{\mathbf{K}} \Box\varphi \leftrightarrow \Box\psi$

(c) $\vdash_{\mathbf{K}} \varphi \leftrightarrow \psi \implies \vdash_{\mathbf{K}} \Diamond\varphi \leftrightarrow \Diamond\psi$

(d) $\vdash_{\mathbf{K}} \top \leftrightarrow \Box\top$

(e) $\vdash_{\mathbf{K}} (\Box\varphi \wedge \Box\psi) \leftrightarrow \Box(\varphi \wedge \psi)$

(f) $\vdash_{\mathbf{K}} (\Box\varphi \vee \Box\psi) \to \Box(\varphi \vee \psi)$

(g) $\vdash_{\mathbf{K}} (\Box\varphi \wedge \Diamond\psi) \to \Diamond(\varphi \wedge \psi)$

*Proof.* (a) is a derived rule that we obtain easily: if there is a proof of $\varphi \to \psi$, then we can extend this proof as follows:

| | | |
|---|---|---|
| $n.$ | $\varphi \to \psi$ | |
| $n+1.$ | $\Box(\varphi \to \psi)$ | **R-□**: $n$ |
| $n+2.$ | $\Box(p \to q) \to (\Box p \to \Box q)$ | **K** |
| $n+3.$ | $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$ | **US**: $n+2$ |
| $n+4.$ | $\Box\varphi \to \Box\psi$ | **MP**: $n+1, n+3$ |

For (b) and (c) the argument is quite similar. For (d) consider the following proof:

| | | |
|---|---|---|
| 1. | $\top$ | PL |
| 2. | $\Box\top \to (\top \to \Box\top)$ | PL |
| 3. | $\Box\top$ | **R-□**: 1 |
| 4. | $\top \to \Box\top$ | **MP**: 3, 2 |
| 5. | $\top \to (\Box\top \to \top)$ | PL |
| 6. | $\Box\top \to \top$ | **MP**: 1, 5 |
| 7. | $(\Box\top \to \top) \to ((\top \to \Box\top) \to (\top \leftrightarrow \Box\top))$ | PL |
| 8. | $(\top \to \Box\top) \to (\top \leftrightarrow \Box\top)$ | **MP**: 6, 7 |
| 9. | $\top \leftrightarrow \Box\top$ | **MP**: 4, 8 |

For (e) consider the following proof:

| | | |
|---|---|---|
| 1. | $\varphi \wedge \psi \to \varphi$ | PL |
| 2. | $\Box(\varphi \wedge \psi) \to \Box\varphi$ | (a): 1 |
| 3. | $\varphi \wedge \psi \to \psi$ | PL |
| 4. | $\Box(\varphi \wedge \psi) \to \Box\psi$ | (a): 3 |
| 5. | $\Box(\varphi \wedge \psi) \to (\Box\varphi \wedge \Box\psi)$ | PL: 2, 4 |
| 6. | $\varphi \to (\psi \to (\varphi \wedge \psi))$ | PL |
| 7. | $\Box\varphi \to \Box(\psi \to (\varphi \wedge \psi))$ | (a): 6 |
| 8. | $\Box(p \to q) \to (\Box p \to \Box q)$ | **K** |
| 9. | $\Box(\psi \to (\varphi \wedge \psi)) \to (\Box\psi \to \Box(\varphi \wedge \psi))$ | **US**: 8 |
| 10. | $\Box\varphi \to (\Box\psi \to \Box(\varphi \wedge \psi))$ | PL: 7, 9 |
| 11. | $(\Box\varphi \wedge \Box\psi) \to \Box(\varphi \wedge \psi)$ | PL: 10 |
| 12. | $(\Box\varphi \wedge \Box\psi) \leftrightarrow \Box(\varphi \wedge \psi)$ | PL: 11, 5 |

The other proofs are left as an exercise. ◁

It is clear that the "derived rules" (a), (b), and (c) in the lemma also hold for all extensions of **K**. Moreover, by induction, one can easily show that also the following rule can be derived:

$$\vdash_{\mathbf{K}} \varphi_1 \wedge \cdots \wedge \varphi_n \to \psi \implies \vdash_{\mathbf{K}} \Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \to \Box\psi.$$

**Remark 3.9.** We did not introduce a deducibility relation $\Sigma \vdash_{K(\Gamma)} \psi$. The usual meaning of $\Sigma \vdash_{K(\Gamma)} \psi$ is that $\psi$ is provable when we assume $\Sigma$. But it is not uniquely clear what "assume" means in that context. Does it mean that the formulae in $\Sigma$ are axioms (*globally* valid formulae) or does it just mean that the formulae in $\Sigma$ are some specific, *local* conditions for $\psi$?
We will adopt the local view: whenever we write $\Sigma \vdash_{K(\Gamma)} \psi$, we will understand it as follows: $\Sigma \vdash_{K(\Gamma)} \psi$ if and only if there are formulae $\varphi_1, \ldots, \varphi_n \in \Sigma$ such that $\vdash_{K(\Gamma)} \varphi_1 \wedge \cdots \wedge \varphi_n \to \psi$. Of course, such a relation is just syntactic sugar and the so-called deduction theorem

$$\varphi_1, \ldots, \varphi_n \vdash_{K(\Gamma)} \psi \implies \varphi_1, \ldots, \varphi_{n-1} \vdash_{K(\Gamma)} \varphi_n \to \psi$$

becomes a propositional triviality.

**Remark 3.10.** A non-trivial meaning of $\Sigma \vdash_{K(\Gamma)} \psi$ could be defined if one allows the formulae in $\Sigma$ to be used as axioms in proofs, that is, $\Sigma \vdash_{K(\Gamma)} \psi$ is defined as $\Sigma \vdash_{K(\Gamma \cup \Sigma)} \psi$. Again such a concept is syntactic sugar, but proving the deduction theorem would become a challenge: given the deduction theorem, we could for example conclude that $\vdash_{\mathbf{K}} p \to \Box p$, which is of course not justified by the Kripke semantics.

**Remark 3.11.** But one can do better than suggested in the last remark. Assume that we want to prove $\psi$ from "assumptions" $\varphi_1, \ldots, \varphi_n$ by using these assumptions as axioms in a proof. We proceed so, but during the proof we maintain for each $\varphi_i$ a dependency list (initially containing $\varphi_i$ only). Whenever we extend a sequence of formulae $\tau_1,\dots,\tau_{m-1}$ by applying one of the rules **MP**, **US**, and **R-□** on formulae depending on $\varphi_i$, we add the new formula $\tau_m$ to the dependency list of $\varphi_i$. Moreover, when we extend $\tau_1,\dots,\tau_{m-1}$ by applying one of the rules **US** or **R-□** on a formula depending on $\varphi_i$, then $\varphi_i$ is added to an (initially empty) blacklist $L$. Then one can prove:

$$\varphi_1,\dots,\varphi_n \vdash^L_{K(\Gamma)} \psi,\ \ \varphi_n \notin L \implies \varphi_1,\dots,\varphi_{n-1} \vdash^L_{K(\Gamma)} \varphi_n \to \psi.$$

That is: if there is a proof of $\psi$ from $\varphi_1,\dots,\varphi_n$ with blacklist $L$ such that $\varphi_n \notin L$, then one can prove $\varphi_n \to \psi$ from $\varphi_1,\dots,\varphi_{n-1}$ while preserving the blacklist.

**Remark 3.12.** Another way to go is to introduce the deduction relation in a way as natural deduction systems do. We introduce $\Sigma \vdash \varphi$ ("assuming $\Sigma$, $\varphi$ is provable") by a set of rules. In our setting the propositional part of the rules could take the following form:

$$\text{(Ass)}\ \frac{\varphi \in \Sigma}{\Sigma \vdash \varphi} \qquad \text{(Add)}\ \frac{\Sigma \vdash \varphi,\ \Sigma \subseteq \Sigma'}{\Sigma' \vdash \varphi} \qquad \text{(MPP)}\ \frac{\Sigma \vdash \varphi \to \psi,\ \Sigma \vdash \varphi}{\Sigma \vdash \psi}$$

$$\text{(CP)}\ \frac{\Sigma \cup \{\varphi\} \vdash \psi}{\Sigma \vdash \varphi \to \psi} \qquad \text{(USS)}\ \frac{\Sigma \vdash \varphi,\ \ \sigma \text{ a substitution}}{\Sigma^\sigma \vdash \varphi^\sigma} \qquad \text{(MTT)}\ \frac{\Sigma \vdash \varphi \to \psi,\ \Sigma \vdash \neg\psi}{\Sigma \vdash \neg\varphi}$$

$$\text{(DN)}\ \frac{\Sigma \vdash \neg\neg\varphi}{\Sigma \vdash \varphi} \qquad \text{(RAA)}\ \frac{\Sigma \cup \{\varphi\} \vdash \bot}{\Sigma \vdash \neg\varphi}$$

In the modal-logical setting we add to these rules (*assumption*, *addition of assumptions*, *modus ponendo ponens*, *conditional proof*, *uniform and simultaneous substitution*, *modus tollendo tollens*, *double negation*, *reductio ad absurdum*) the following rules:

$$\text{(S}\Box\text{)}\ \frac{\Sigma \vdash \varphi}{\Box(\Sigma) \vdash \Box\varphi} \qquad \text{(Ax)}\ \frac{\varphi \text{ is an axiom}}{\Sigma \vdash \varphi}$$

Here $\Box(\Sigma)$ is defined as $\{\Box\varphi : \varphi \in \Sigma\}$. Then, for example, axiom **K** can be derived as follows:

$$\text{(CP)}\ \cfrac{\text{(CP)}\ \cfrac{\text{(S}\Box\text{)}\ \cfrac{\text{(MPP)}\ \cfrac{\text{(Ass)}\ \cfrac{p \in \{p, p\to q\}}{\{p,p\to q\} \vdash p} \qquad \text{(Ass)}\ \cfrac{p\to q \in \{p,p\to q\}}{\{p,p\to q\} \vdash p\to q}}{\{p,p\to q\} \vdash q}}{\{\Box p, \Box(p\to q)\} \vdash \Box q}}{\{\Box(p\to q)\} \vdash \Box p \to \Box q}}{\emptyset \vdash \Box(p\to q) \to (\Box p \to \Box q)}$$

Also an analogue of the necessitation rule **R-□** can be easily proven: From $\emptyset \vdash \varphi$ we obtain $\Box(\emptyset) \vdash \Box\varphi$. Since $\Box(\emptyset)$ is empty, we obtain $\emptyset \vdash \Box\varphi$. The "disadvantage" of natural deduction systems is that, in general, additional axioms often cannot be easily expressed as rules that fit nicely into the rule scheme used for the propositional logic connectives.

## 3.4 Soundness and completeness

In the last section we introduced Hilbert-style "axiomatizations" and a concept of proof that allows us to enumerate all formulae that are provable in the axiomatization. In section 3.1 we stated that the set of formulae valid in all Kripke frames can be axiomatized using axiom **K** and rule **R-**$\Box$. This means that the formulae that are valid in all Kripke frames are exactly those that have a **K**-proof. In this section we will show this claim.
For the purposes of this section it is convenient to say that a modal logic $\Lambda$ is *normal* (cp. Definition 3.1) if it contains axioms **K** and **Dual**, and is closed under the necessitation rule. It is clear that for any set of $\mathcal{L}_\tau(P)$-formulae, $\Gamma$, the set of $K(\Gamma)$-theorems is such a modal logic.

**Definition 3.4** (Soundness)**.** A modal logic $\Lambda \subseteq \mathcal{L}_\tau$ is called *sound* with respect to a class of $\tau$-frames, $\mathcal{C}$, if $\Lambda \subseteq \Lambda(\mathcal{C})$.

Hence a modal logic is sound with respect to $\mathcal{C}$ if each formula in $\Lambda$ is valid in each frame in $\mathcal{C}$. To check that an axiomatization $K(\Gamma)$ (i.e., the set of its theorems) is sound one needs to show that all axioms in $\Gamma$ are valid in $\mathcal{C}$ and that validity in $\mathcal{C}$ is preserved by the rules of $K(\Gamma)$. A simple consequence of this definition is the following:

**Lemma 3.8.** *$K(\emptyset)$ is sound for the class of all Kripke frames.*

*Proof.* In fact, each propositional tautology is valid in each Kripke frame. Moreover, axiom **K** is valid in each Kripke frame (see Lemma 2.4).
Furthermore, all rules preserve validity. For **MP**, if both $\varphi$ and $\varphi \to \psi$ are valid in each Kripke frame, then so is $\psi$.
**US**: If $\varphi$ is valid in each Kripke frame, then for each substitution $\sigma$, $\varphi^\sigma$ is valid in each Kripke frame. For proof by contraposition, assume that $\varphi^\sigma$ is not valid in each frame, then there is a model $\mathcal{M}$ based on some frame $\mathcal{F}$ and a state $s$ such that $\mathcal{M} \not\models_s \varphi^\sigma$. Then, by Lemma 2.3, $\mathcal{M}^\sigma \not\models_s \varphi$. $\mathcal{M}^\sigma$ is defined on the same frame as $\mathcal{M}$. Hence $\varphi$ is not valid in $\mathcal{F}$, and thus not valid in all frames. By contraposition, the claim follows.
**R-**$\Box$: If $\varphi$ is valid in each Kripke frame, then $\Box\varphi$ is valid in each Kripke frame. For if $\Box\varphi$ is not valid in some frame $\mathcal{F}$, then $\mathcal{M} \not\models_s \Box\varphi$ for some state $s$ in some model $\mathcal{M}$ defined on $\mathcal{F}$. But then there exists a state $s'$ (accessible from $s$) such that $\mathcal{M} \not\models_{s'} \varphi$. Hence $\varphi$ is not valid in $\mathcal{F}$ and hence not valid in all frames. Again, by contraposition, the claim follows. $\lhd$

**Remark 3.13.** It is worth mentioning that the proof of the lemma shows that validity is preserved with respect to single frames. Thus, these rules also preserve validity for *any* class of frames.

**Definition 3.5** (Weak completeness)**.** A modal logic $\Lambda \subseteq \mathcal{L}_\tau$ is said to be *(weakly) complete* with respect to a class of $\tau$-frames, $\mathcal{C}$, if and only if $\Lambda(\mathcal{C}) \subseteq \Lambda$.

Hence a modal logic is weakly complete for $\mathcal{C}$ if all the formulae that are valid in each frame in $\mathcal{C}$ are actually contained in $\Lambda$. In order to check that an axiomatization $K(\Gamma)$ is weakly complete we need to show that all formulae that are valid in all frames of $\mathcal{C}$ are $K(\Gamma)$-theorems. That is, we have to show that each such formula has a $K(\Gamma)$-proof.

There is an even stronger concept of completeness. To explain that we will need some further concepts.

**Definition 3.6.** A set of $\mathcal{L}_\tau$-formulae, $\Sigma$, is called $\Lambda$-*inconsistent* if there exist $\varphi_1, \ldots, \varphi_n \in \Sigma$ with $\varphi_1 \wedge \cdots \wedge \varphi \to \bot \in \Lambda$. Otherwise, $\Sigma$ is said to be $\Lambda$-*consistent*.

That is, $\Sigma$ is $K(\Gamma)$-consistent if there exist no formulae $\varphi_1, \dots, \varphi_n \in \Sigma$ such that

$$\vdash_{K(\Gamma)} \varphi \wedge \cdots \wedge \varphi_n \to \bot.$$

Using the notation introduced in Remark 3.9, $\Sigma$ is inconsistent if $\Sigma \vdash_{K(\Gamma)} \bot$.

Weak completeness states that each valid formula is provable. Hence when a singleton set of formulae $\{\varphi\}$ is consistent, the formula $\neg\varphi$ is not provable, hence not valid, and thus $\varphi$ is satisfiable. Strong completeness generalizes this connection between consistency and satisfiability to *sets* of formulae.

**Definition 3.7** (Strong completeness)**.** A modal logic $\Lambda$ is *strongly complete* with respect to a class of $\tau$-frames, $\mathcal{C}$, if each $\Lambda$-consistent set of $\mathcal{L}_\tau$-formulae is satisfiable in $\mathcal{C}$.

Another characterization of strong completeness is in terms of deducibility (see Remark 3.9 and Remark 3.12).

**Definition 3.8.** Let $\mathcal{C}$ be a class of frames. We say that a formula $\varphi$ is a *$\mathcal{C}$-consequence* of $\Sigma$, $\Sigma \models_{\mathcal{C}} \varphi$, if for each state $s$ in each model $\mathcal{M}$ defined on any $\mathcal{F}$ in $\mathcal{C}$,

$$\mathcal{M} \models_s \Sigma \implies \mathcal{M} \models_s \varphi.$$

Clearly, $\Sigma \models_{\mathcal{C}} \varphi$ if and only if $\Sigma \cup \{\neg\varphi\}$ is not satisfiable in $\mathcal{C}$. On the other hand, $\Sigma \vdash_{K(\Gamma)} \varphi$ if and only if $\Sigma \cup \{\neg\varphi\}$ is $K(\Gamma)$-inconsistent.

**Lemma 3.9.** *A normal modal logic $\Lambda(\Gamma) \subseteq \mathcal{L}_\tau$ is* strongly complete *with respect to a class of $\tau$-frames, $\mathcal{C}$, if and only if*

$$\Sigma \models_{\mathcal{C}} \varphi \implies \Sigma \vdash_{K(\Gamma)} \varphi. \qquad \lhd$$

In the rest of this section we show the completeness of some normal modal logics. The most fundamental notion is that of a maximal consistent set of formulae.

**Definition 3.9** (Maximal consistent sets)**.** A set of $\mathcal{L}_\tau(P)$-formulae, $\Sigma$, is called *maximal $\Lambda$-consistent* if it is $\Lambda$-consistent and there is no $\Lambda$-consistent set of $\mathcal{L}_\tau(P)$-formulae $\Sigma'$ with $\Sigma \subsetneq \Sigma'$.

Thus, a consistent set of formulae is maximal consistent if each proper extension of it is inconsistent. The most important properties of maximal consistent sets are summarized in the following proposition.

**Proposition 3.10.** *Let $\Sigma$ be a maximal $\Lambda$-consistent set. Then the following claims hold:*

(a) *$\Lambda$ is contained in $\Sigma$.*

(b) *$\Sigma$ is closed under consequences, i.e., if $\varphi \in \Sigma$ and $\varphi \to \psi \in \Sigma$, then $\psi \in \Sigma$.*

(c) *$\Sigma$ is (theory-) complete, i.e., for each $\mathcal{L}_\tau(P)$-formula, either $\varphi \in \Sigma$ or $\neg\varphi \in \Sigma$.*

(d) *For each $\mathcal{L}_\tau(P)$-formula $\varphi$, $\neg\varphi \in \Sigma$ if and only if $\varphi \notin \Sigma$.*

(e) *For all $\mathcal{L}_\tau(P)$-formulae $\varphi$ and $\psi$, $\varphi \vee \psi \in \Sigma$ if and only if $\varphi \in \Sigma$ or $\psi \in \Sigma$.*

*Proof.* (a) For proof by contradiction, assume that $\Lambda$ is not contained in $\Sigma$. Then there is a formula $\psi \in \Lambda$ such that $\psi \notin \Sigma$. Hence $\Sigma \cup \{\psi\}$ is a proper superset of $\Sigma$. As $\Sigma$ is maximal $\Lambda$-consistent, $\Sigma \cup \{\psi\}$ must be $\Lambda$-inconsistent. That is, there exist formulae $\varphi_1, \dots, \varphi_n \in \Sigma$ such that $\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \psi \to \bot \in \Lambda$. But then by using propositional logic transformations,

$\psi \to (\varphi_1 \wedge \cdots \wedge \varphi_n \to \bot) \in \Lambda$. Since $\Lambda$ is closed under **MP**, it follows from $\psi \in \Lambda$ that $\varphi_1 \wedge \cdots \wedge \varphi_n \to \bot \in \Lambda$. Thus $\Sigma$ is $\Lambda$-inconsistent — in contradiction to the choice of $\Sigma$.

(b) Assume that $\varphi$ and $\varphi \to \psi$ are in $\Sigma$, but $\psi$ is not. As $\Sigma$ is maximal $\Lambda$-consistent, $\Sigma \cup \{\psi\}$ must be $\Lambda$-inconsistent. Then there exist $\chi_1, \ldots, \chi_n \in \Sigma$ such that $\chi_1 \wedge \cdots \wedge \chi_n \wedge \psi \to \bot \in \Lambda$. Since $\varphi \wedge (\varphi \to \psi) \to \psi \in \Lambda$, we obtain that $\chi_1 \wedge \cdots \wedge \chi_n \wedge \varphi \wedge (\varphi \to \psi) \to \bot \in \Lambda$. Since all the $\chi_i$ as well as $\varphi$ and $\varphi \to \psi$ are in $\Sigma$, it follows that $\Sigma$ is $\Lambda$-inconsistent — a contradiction.

The claims (c), (d), and (e) are left as an exercise. $\lhd$

**Lemma 3.11** (Lindenbaum lemma)**.** *Let $\Lambda$ be a modal logic. Then each $\Lambda$-consistent set of $\mathcal{L}_\tau(P)$-formulae, $\Sigma$, is contained in a maximal $\Lambda$-consistent set of $\mathcal{L}_\tau(P)$-formulae.*

*Proof.* For the proof we assume that $\mathcal{L}_\tau(P)$ contains only a countable number of propositional variables (in the multi-modal case we would also assume that the language contains a countable number of modal connectives only). Since the set of $\mathcal{L}_\tau(P)$-formulae is countable, there exists an enumeration of all formulae in $\mathcal{L}_\tau$, say $\varphi_1, \varphi_2, \ldots$ We use this enumeration to define an ascending sequence of $\Lambda$-consistent sets of formulae, $\Sigma =: \Sigma_0 \subseteq \Sigma_1 \subseteq \cdots$, as follows: given that $\Sigma_{n-1}$ has already been defined, we set:

$$\Sigma_n := \begin{cases} \Sigma_{n-1} \cup \{\varphi_n\} & \text{if } \Sigma_{n-1} \cup \{\varphi_n\} \text{ is } \Lambda\text{-consistent} \\ \Sigma_{n-1} & \text{otherwise} \end{cases}$$

Clearly, each $\Sigma_n$ is $\Lambda$-consistent. It remains to be shown that

$$\Sigma' := \bigcup_n \Sigma_n$$

is maximal $\Lambda$-consistent (obviously, $\Sigma$ is contained in $\Sigma'$).

To prove that $\Sigma'$ is $\Lambda$-consistent, assume the contrary. Then there exist formulae $\psi_1, \ldots, \psi_n$ in $\Sigma'$ such $\psi_1 \wedge \cdots \wedge \psi_n \to \bot \in \Lambda$. Each $\psi_j$ occurs as a formula $\varphi_{i_j}$ in the enumeration (we may assume that $i_1 < \cdots < i_n$). Thus it follows $\varphi_{i_1} \wedge \cdots \wedge \varphi_{i_n} \to \bot \in \Lambda$. Since the chain of formula sets is ascending, all $\varphi_{i_j}$ are in $\Sigma_{i_n}$. But this shows that $\Sigma_{i_n}$ is $\Lambda$-inconsistent — in contradiction to the observation that each $\Sigma_j$ is $\Lambda$-consistent.

Finally, $\Sigma'$ is maximal $\Lambda$-consistent. Assume that $\varphi_n$ is any formula in the enumeration such that $\Sigma' \cup \{\varphi_n\}$ is $\Lambda$-consistent. Then, of course, the subset $\Sigma_{n-1} \cup \{\varphi_n\}$ must be $\Lambda$-consistent as well. By definition of $\Sigma_n$, $\varphi_n$ is contained in $\Sigma_n$, and hence also contained in $\Sigma'$. This shows that there is no proper extension of $\Sigma'$ that is $\Lambda$-consistent. $\lhd$

So far we just reported on facts that hold for any modal logic. Next we show that whenever a maximal consistent set contains a formula $\Diamond\varphi$, there exists a "witness", a maximal consistent set of formulae that contains $\varphi$. This follows from the Lindenbaum Lemma and the following lemma.

**Lemma 3.12.** *Let $\Lambda$ be a normal modal logic and $\Sigma$ be a $\Lambda$-consistent set of $\mathcal{L}_\tau(P)$-formulae with $\Diamond\psi \in \Sigma$. Then the set of $\mathcal{L}_\tau(P)$-formulae*

$$\{\varphi : \Box\varphi \in \Sigma\} \cup \{\psi\}$$

*is $\Lambda$-consistent.*

*Proof.* Assume that $\{\varphi : \Box\varphi \in \Sigma\} \cup \{\psi\}$ is $\Lambda$-inconsistent. Then there exist $\Box\varphi_1, \dots, \Box\varphi_n \in \Sigma$ such that $\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \psi \to \bot \in \Lambda$. By PL-transformations, we obtain that $\varphi_1 \wedge \cdots \wedge \varphi_n \to \neg\psi \in \Lambda$. Since $\Lambda$ is a normal modal logic, it follows: $\Box(\varphi_1 \wedge \cdots \wedge \varphi_n) \to \Box\neg\psi \in \Lambda$. By applying Lemma 3.7, we obtain that $\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \to \Box\neg\psi \in \Lambda$. By **DUAL**, we can conclude that $\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \to \neg\Diamond\psi \in \Lambda$ and hence that $\Box\varphi_1 \wedge \cdots \wedge \Box\varphi_n \wedge \Diamond\psi \to \bot \in \Lambda$. This shows that $\Sigma$ is $\Lambda$-inconsistent. $\lhd$

**Lemma 3.13.** *Let $\Lambda$ be a normal modal logic and let $\Sigma$ and $\Sigma'$ be maximal $\Lambda$-consistent sets of formulae. Then the following statements are equivalent*

(a) *For each formula $\varphi$, if $\Box\varphi \in \Sigma$, then $\varphi \in \Sigma'$.*

(b) *For each formula $\varphi$, if $\varphi \in \Sigma'$, then $\Diamond\varphi \in \Sigma$.*

*Proof.* (a) $\Rightarrow$ (b): Assume $\varphi \in \Sigma'$. Then $\neg\varphi \notin \Sigma'$ (since $\Sigma$ is $\Lambda$-consistent) and hence by (a) $\Box\neg\varphi \notin \Sigma$. As $\Sigma$ is maximal $\Lambda$-consistent, it follows that $\neg\Box\neg\varphi \in \Sigma$ (by Proposition 3.10). Axiom **Dual** ensures that $\Diamond\varphi \in \Sigma$.
(b) $\Rightarrow$ (a): follows analogously. $\lhd$

**Proposition 3.14.** *Let $\Lambda$ be a normal modal logic and let $S^\Lambda$ be the set of all maximal $\Lambda$-consistent sets of $\mathcal{L}_\tau$-formulae. Then*

$$\Sigma R^\Lambda_\Diamond \Sigma' :\Longleftrightarrow \{\varphi : \Box\varphi \in \Sigma\} \subseteq \Sigma'$$

*defines a binary relation on $S^\Lambda$ (for each unary modality $\Diamond$ of $\tau$) and hence*

$$\mathcal{F}^\Lambda = \langle S^\Lambda, \{R^\Lambda_\Diamond\}_{\Diamond\in\tau}\rangle$$

*is a Kripke frame,* the canonical frame of $\Lambda$. *Moreover, the following implications hold:*

(a) *If **T** is in $\Lambda$, then $R^\Lambda_\Diamond$ is reflexive.*

(b) *If **4** is in $\Lambda$, then $R^\Lambda_\Diamond$ is transitive.*

(c) *If **B** is in $\Lambda$, then $R^\Lambda_\Diamond$ is symmetric.*

(d) *If **D** is in $\Lambda$, then $R^\Lambda_\Diamond$ is serial.*

(e) *If **E** is in $\Lambda$, then $R^\Lambda_\Diamond$ is Euclidean.*

*Proof.* (a) We need to prove that for each MCS (= maximal $\Lambda$-consistent set of formulae) $\Sigma$, $\Sigma R^\Lambda_\Diamond \Sigma$. By Lemma 3.13, we have to show that $\Diamond\varphi \in \Sigma$ whenever $\varphi \in \Sigma$. Since **T** is in $\Lambda$, $\Sigma$ contains all formulae of the form $\varphi \to \Diamond\varphi$ (by **US** and Proposition 3.10 (a)). Thus, by Proposition 3.10 (b), $\Diamond\varphi \in \Sigma$ whenever $\varphi \in \Sigma$.
(c) Let $\Sigma$ and $\Sigma'$ be MCS with $\Sigma R^\Lambda_\Diamond \Sigma'$. We have to show that $\Sigma' R^\Lambda_\Diamond \Sigma$. So assume $\Box\varphi \in \Sigma'$. Then by $\Sigma R^\Lambda_\Diamond \Sigma'$ and Lemma 3.13, $\Diamond\Box\varphi \in \Sigma$. Since **B** is in $\Lambda$, $\Sigma$ contains all formulae of the form $\Diamond\Box\psi \to \psi$ (by **US** and Proposition 3.10 (a)). Thus, by Proposition 3.10 (b), $\varphi \in \Sigma$ — as to be shown.
The proofs of (b), (d), and (e) are left as an exercise. $\lhd$

**Theorem 3.15.** *Let $\Lambda$ be a normal modal logic and let $S^\Lambda$ be the set of all maximal $\Lambda$-consistent sets of $\mathcal{L}_\tau(P)$-formulae. Define $V^\Lambda : P \to 2^{S^\Lambda}$ by $V^\Lambda(p) := \{\Sigma \in S^\Lambda : p \in \Sigma\}$. Then,*

$$\mathcal{M}^\Lambda = \langle \mathcal{F}^\Lambda, V^\Lambda \rangle$$

*defines a Kripke model on the canonical frame of* $\Lambda$, the canonical model of $\Lambda$. *Moreover, for each* $\mathcal{L}_\tau(P)$*-formula* $\varphi$,

$$\mathcal{M}^\Lambda \models_\Sigma \varphi \iff \varphi \in \Sigma.$$

*Proof.* We only need to show the last equivalence and proceed by induction on the formula length.

For propositional variables $p$ the claim holds trivially, because:

$$\mathcal{M}^\Lambda \models_\Sigma p \iff \Sigma \in V^\Lambda(p) \iff p \in \Sigma.$$

If $\varphi$ has the form $\neg\psi$, the claim can be concluded from Proposition 3.10 and the induction hypothesis as follows:

$$\mathcal{M}^\Lambda \models_\Sigma \neg\psi \iff \mathcal{M}^\Lambda \not\models_\Sigma \psi \iff \psi \notin \Sigma \iff \neg\psi \in \Sigma.$$

If $\varphi$ has the form $\psi_1 \vee \psi_2$, the claim follows analogously:

$$\begin{aligned} \mathcal{M}^\Lambda \models_\Sigma \psi_1 \vee \psi_2 &\iff \mathcal{M}^\Lambda \models_\Sigma \psi_1 \text{ or } \mathcal{M}^\Lambda \models_\Sigma \psi_2 \\ &\iff \psi_1 \in \Sigma \text{ or } \psi_2 \in \Sigma \\ &\iff \psi_1 \vee \psi_2 \in \Sigma. \end{aligned}$$

If $\varphi$ has the form $\Diamond\psi$, the claim follows from the following equivalences:

$$\begin{aligned} \mathcal{M}^\Lambda \models_\Sigma \Diamond\psi &\iff \text{there is a } \Sigma' \in S^\Lambda \text{ with } \Sigma R^\Lambda_\Diamond \Sigma' \text{ and } \mathcal{M}^\Lambda \models_{\Sigma'} \psi \\ &\iff \text{there is a MCS } \Sigma' \text{ with } \{\varphi : \Box\varphi \in \Sigma\} \subseteq \Sigma' \text{ and } \psi \in \Sigma' \\ &\iff \Diamond\psi \in \Sigma. \end{aligned}$$

The last equivalence can be proven as follows: "$\Rightarrow$" has been shown in Lemma 3.13. "$\Leftarrow$" follows from the "witness"-lemma (Lemma 3.12) and the Lindenbaum lemma (Lemma 3.11). $\lhd$

An immediate consequence of the theorem is that $K(\emptyset)$ is a complete axiomatization of the set of formulae that are valid in each Kripke frame.

**Corollary 3.16.** $K(\emptyset)$ *is strongly complete for the class of all Kripke frames.*

*Proof.* Let $\Lambda$ be the set of $K(\emptyset)$-theorems and let $\Sigma$ be a $\Lambda$-consistent set of formulae. By the Lindenbaum Lemma 3.11, this set is contained in a "state" $\Sigma'$ of the canonical model $\mathcal{M}^\Lambda$ of $\Lambda$. By Theorem 3.15, $\mathcal{M}^\Lambda \models_{\Sigma'} \varphi$ for each formula $\varphi$ in $\Sigma'$, and hence for each $\varphi \in \Sigma$. Thus, $\Sigma$ is satisfiable. $\lhd$

From now on we may alway assume that a normal modal logic contains all formulae valid in all Kripke frames: **K** and even $\mathbf{K}_n$ is sound and complete for the class of all Kripke frames.

## 3.5 Canonical frames, definability, and compactness

Before we start to investigate selected modal logics in more detail, we want to draw some conclusions from the results of the last section. In particular, the canonical frame of a normal modal logic will be of special interest: as we have already seen, this frame reflects some of the properties on Kripke frames defined in section 3.2 (see Proposition 3.14). The key notion in this section is that of a *frame for a modal logic*.

**Definition 3.10.** Let $\Lambda \subseteq \mathcal{L}_\tau$ be a normal modal logic. A $\tau$-frame $\mathcal{F}$ is called a *frame for* $\Lambda$ if $\mathcal{F} \models \Lambda$. $\Lambda$ is called *canonical* if the canonical frame of $\Lambda$ is a frame for $\Lambda$, i.e., $\mathcal{F}^\Lambda \models \Lambda$.

Notice that for normal modal logics $\Lambda$ in general a *weaker* claim holds:

$$\mathcal{M}^\Lambda \models \Lambda. \tag{$*$}$$

In fact, $\Lambda$ is contained in each maximal $\Lambda$-consistent set $\Sigma \in S^\Lambda$. Thus, by Theorem 3.15, $\mathcal{M}^\Lambda \models_\Sigma \Lambda$, for each $\Sigma \in S^\Lambda$. But from $(*)$ it does not follow that each normal modal logic is canonical. Nevertheless, as proven in Proposition 3.14, many modal logics are well-behaved: **K**, **S4**, **S5**, etc. are canonical. For example, for **S4** the canonical frame $\mathcal{F}^{\mathbf{S4}}$ is reflexive and transitive. Hence by Proposition 3.1 and Proposition 3.2, it holds $\mathcal{F}^{\mathbf{S4}} \models p \to \Diamond p$ and $\mathcal{F}^{\mathbf{S4}} \models \Diamond\Diamond p \to \Diamond p$. Consequently, $\mathcal{F}^{\mathbf{S4}} \models \mathbf{S4}$, which shows that **S4** is canonical. The following proposition explains the situation more generally:

**Proposition 3.17.** *Let $\Gamma$ be a set of $\mathcal{L}_\tau$-formulae that defines a class $\mathcal{C}$ of $\tau$-frames. Then the frames for $\Lambda(\Gamma)$ are exactly the frames in $\mathcal{C}$. In particular, $\Lambda(\Gamma)$ is canonical if and only if the canonical frame $\mathcal{F}^{\Lambda(\Gamma)}$ is in $\mathcal{C}$.*

*Proof.* Because $\Gamma$ defines $\mathcal{C}$, the frames in $\mathcal{C}$ are exactly the frames $\mathcal{F}$ with $\mathcal{F} \models \Gamma$. Hence, for the first claim, we just need to show that the frames for $\Lambda(\Gamma)$ are exactly the frames $\mathcal{F}$ with $\mathcal{F} \models \Gamma$.

Trivially, if $\mathcal{F} \models \Lambda(\Gamma)$, then $\mathcal{F} \models \Gamma$, since the normal modal logic $\Lambda(\Gamma)$ contains $\Gamma$ as a subset. Conversely, if $\mathcal{F} \models \Gamma$, then $\Gamma$ is a subset of $\Lambda(\mathcal{F})$. Since $\Lambda(\Gamma)$ is the smallest normal modal logic with subset $\Gamma$, it follows that $\Lambda(\Gamma) \subseteq \Lambda(\mathcal{F})$, that is, each formula in $\Lambda(\Gamma)$ is valid in $\mathcal{F}$, i.e., $\mathcal{F}$ is a frame for $\Lambda(\Gamma)$.

For the second claim: by definition, $\Lambda(\Gamma)$ is canonical if and only if the canonical frame $\mathcal{F}^{\Lambda(\Gamma)}$ is a frame for $\Lambda(\Gamma)$. As just shown, this is the case if and only if $\mathcal{F}^{\Lambda(\Gamma)}$ is in $\mathcal{C}$. $\lhd$

The canonical frame of a normal modal logic $\Lambda$ has its own (normal) modal logic $\Lambda(\mathcal{F}^\Lambda)$. The following lemma explains the connection between these modal logics.

**Lemma 3.18.** *Let $\Lambda$ be a normal modal logic. Then the following statements hold:*

(a) $\Lambda(\mathcal{F}^\Lambda) \subseteq \Lambda$.

(b) *$\Lambda$ is canonical if and only if $\Lambda \subseteq \Lambda(\mathcal{F}^\Lambda)$.*

*Proof.* (a) If $\varphi \notin \Lambda$, then the set $\{\neg\varphi\}$ is $\Lambda$-consistent. By the Lindenbaum Lemma 3.11, this set is contained in a maximal $\Lambda$-consistent set $\Sigma'$. It follows (by Theorem 3.15) $\mathcal{M}^\Lambda \not\models_{\Sigma'} \varphi$, hence $\mathcal{M}^\Lambda \not\models \varphi$, thus $\mathcal{F}^\Lambda \not\models \varphi$. This shows that $\varphi \notin \Lambda(\mathcal{F}^\Lambda)$.

(b) Obviously, $\mathcal{F}^\Lambda \models \Lambda$ if and only if $\mathcal{F}^\Lambda \models \varphi$ for each $\varphi \in \Lambda$ if and only if $\varphi \in \Lambda(\mathcal{F}^\Lambda)$ for each $\varphi \in \Lambda$ if and only if $\Lambda$ is a subset of $\Lambda(\mathcal{F}^\Lambda)$. $\lhd$

This means that a normal modal logic is canonical if and only if it is the logic of its canonical frame. An immediate consequence is the following:

**Proposition 3.19.** *Let $\Lambda$ be a normal modal logic that is sound and complete for a class $\mathcal{C}$ of $\tau$-frames. If $\Lambda$ is canonical, then $\Lambda(\mathcal{F}^\Lambda)$ is sound and complete for $\mathcal{C}$.*

*Proof.* This follows immediately from Lemma 3.18. Notice that for the soundness claim the assumption that $\Lambda$ be canonical is not required. $\lhd$

**Proposition 3.20.** *If a normal modal logic $\Lambda$ is canonical, then it is sound and strongly complete for the class of its frames.*

*Proof.* Clearly, $\Lambda$ is sound with respect to any class of frames for $\Lambda$.

For the completeness claim, let $\Sigma$ be $\Lambda$-consistent. By the Lindenbaum Lemma 3.11, we may assume that $\Sigma$ is maximal $\Lambda$-consistent. Then it holds $\mathcal{M}^{\Lambda} \models_{\Sigma} \Sigma$. Thus, $\Sigma$ is satisfiable in a model based on the canonical frame $\mathcal{F}^{\Lambda}$. Since $\Lambda$ is canonical, it follows that $\Sigma$ is satisfiable in a frame for $\Lambda$. $\lhd$

**Corollary 3.21.** *Let $\mathcal{C}$ be a class of $\tau$-frames defined by a set of $\mathcal{L}_{\tau}$-formulae $\Gamma$. If $\Lambda(\Gamma)$ is canonical, then $\Lambda(\Gamma)$ is sound and strongly complete for $\mathcal{C}$.*

*Proof.* By Proposition 3.20, $\Lambda(\Gamma)$ is sound and strongly complete for the class of its frames. Since $\Gamma$ defines $\mathcal{C}$, the frames for $\Lambda(\Gamma)$ are exactly the frames in $\mathcal{C}$ (see Proposition 3.17). This shows that $\Lambda(\Gamma)$ is sound and strongly complete for the frames in $\mathcal{C}$. $\lhd$

**Remark 3.14.** In the literature there are different notions of a normal logic being *characterized* by a class of frames. In (Blackburn et al., 2002) the term is mostly used synonymously to definability: a normal modal logic is characterized by a class of frames if the logic *defines* that class of frames. In (Hughes and Cresswell, 1996) a normal modal logic is said to be characterized by a class of frames if the logic is sound and complete for that class of frames. Often both properties come hand in hand, but one should keep in mind that even if a normal modal logic $\Lambda$ is sound and complete for a class of frames, $\mathcal{C}$, nothing follows about whether $\Lambda$ defines $\mathcal{C}$. To mention just a simple example: **K** is sound and complete with respect to the class of all irreflexive frames, but **K** does not define that class of frames. However, Corollary 3.21 shows that for a large class of modal logics the other direction holds.

In the rest of this section we will investigate the connection between compactness and strong completeness.

**Definition 3.11.** A normal modal logic $\Lambda$ is called *compact* if each $\Lambda$-consistent set of $\mathcal{L}_{\tau}$-formulae is satisfiable in a frame for $\Lambda$.

Thus, a normal modal logic is compact if and only if it is strongly complete for the class of its frames. Next we present a "finite subset" characterization of compactness (cp. section 2.6).

**Proposition 3.22.** *Let $\Lambda$ be a normal modal logic that is sound and weakly complete for the class of its frames. Equivalent are:*

(a) *$\Lambda$ is compact.*

(b) *For each set $\Sigma$ of $\mathcal{L}_{\tau}$-formulae, $\Sigma$ is satisfiable in a frame for $\Lambda$ whenever each of its finite subsets is satisfiable in a frame for $\Lambda$.*

*Proof.* (a) $\Rightarrow$ (b): Assume that $\Sigma$ is not satisfiable in a frame for $\Lambda$. Since $\Lambda$ is compact, it follows that $\Sigma$ is $\Lambda$-inconsistent. Hence there exists a finite subset of $\Sigma$, say $\Sigma'$, such that $\bigwedge\Sigma' \to \bot \in \Lambda$. Hence, $\neg\bigwedge\Sigma' \in \Lambda$. Moreover, $\Lambda$ is sound for the class of all frames for $\Lambda$, in what follows denoted by $\mathcal{C}_{\Lambda}$. That is, $\Lambda \subseteq \Lambda(\mathcal{C}_{\Lambda})$. Hence, $\neg\bigwedge\Sigma' \in \Lambda(\mathcal{C}_{\Lambda})$, that is, $\mathcal{F} \models \neg\bigwedge\Sigma'$ for each frame $\mathcal{F}$ for $\Lambda$. Thus $\bigwedge\Sigma'$ and hence $\Sigma'$ are not satisfiable in any frame for $\Lambda$. By contraposition, this shows (b).

(b) $\Rightarrow$ (a): Again we assume that $\Sigma$ is not satisfiable in a frame for $\Lambda$. We have to show that $\Sigma$ is $\Lambda$-inconsistent. By condition (b), there exists a finite subset $\Sigma'$ of $\Sigma$ that is not satisfiable in any frame for $\Lambda$. Thus, $\bigwedge\Sigma'$ is not satisfiable in any frame for $\Lambda$. Consequently, we obtain that $\mathcal{F} \models \neg\bigwedge\Sigma'$ for each frame $\mathcal{F}$ for $\Lambda$, and hence $\neg\bigwedge\Sigma' \in \Lambda(\mathcal{C}_\Lambda)$. Since $\Lambda$ is complete for $\mathcal{C}_\Lambda$, that is, $\Lambda(\mathcal{C}_\Lambda) \subseteq \Lambda$, it follows that $\neg\bigwedge\Sigma' \in \Lambda$. Thus, $\bigwedge\Sigma' \to \bot \in \Lambda$, which shows that $\Sigma$ is $\Lambda$-inconsistent. $\lhd$

As said above, a normal modal logic is compact if it is strongly complete for the class of its frames. This entails that if a normal modal logic $\Lambda = \Lambda(\Gamma)$ is strongly complete for a class of frames $\mathcal{C}$ that is defined by the set of $\mathcal{L}_\tau$-formulae $\Gamma$, then $\Lambda$ is compact.

**Proposition 3.23.** *Let $\mathcal{C}$ be a class of frames defined by a set of FOL-sentences in the frame language of $\tau$. If $\Lambda$ is sound and complete for $\mathcal{C}$, then $\Lambda$ is compact.*

*Proof.* Let $\Sigma$ be a $\Lambda$-consistent set of formulae. Furthermore, let $\Delta$ be a set of FO-sentences such that the frames in $\mathcal{C}$ are exactly those frames $\mathcal{F}$ with $\mathcal{F} \models \Delta$ ($\mathcal{F}$ is here conceived of as a first-order structure). We show that each finite subset of $\Delta \cup \mathrm{ST}_x(\Sigma)$ is satisfiable (as a first-order set of formulae with one free variable $x$). Then, by compactness of first-order logic, there exists a first-order structure $\mathcal{S}$ (for the language $\mathcal{L}_{\mathrm{FOL}}(\tau, P)$) and an $s \in \mathcal{S}$ such that $\mathcal{S}, (x \mapsto s) \models \Delta \cup \mathrm{ST}_x(\Sigma)$. Consider now the reduct of $\mathcal{S}$ on the frame language of $\tau$, $\mathcal{F}$: we obtain this structure by simply dropping from $\mathcal{S}$ the interpretation of unary relation symbols that are associated in the standard translation to propositional variables in $P$. Because of $\mathcal{S} \models \Delta$, it holds that $\mathcal{F} \models \Delta$ and hence $\mathcal{F}$ (conceived of as a $\tau$-frame) is contained in $\mathcal{C}$. Hence $\mathcal{F} \models \Lambda(\mathcal{C})$ and thus (as $\Lambda$ is sound for $\mathcal{C}$) $\mathcal{F} \models \Lambda$, i.e., $\mathcal{F}$ is a frame for $\Lambda$. Furthermore, as $\mathcal{S}, (x \mapsto s) \models \mathrm{ST}_x(\Sigma)$, we obtain $\mathcal{M}_\mathcal{S} \models_s \Sigma$. But this shows that $\Sigma$ is satisfiable in a frame for $\Lambda$ (since $\mathcal{M}_\mathcal{S}$ is a model on the frame $\mathcal{F}$).
It remains to be shown that $\Delta' \cup \mathrm{ST}_x(\Sigma')$ is satisfiable for all finite subsets $\Delta'$ and $\Sigma'$ of $\Delta$ and $\Sigma$, respectively. For proof by contradiction, we assume that this is not the case. But then there is no frame $\mathcal{F}$ in $\mathcal{C}$ in which $\Sigma'$ is satisfiable, that is, in each frame in $\mathcal{C}$ it holds $\mathcal{F} \models \neg\bigwedge\Sigma'$. Since $\Lambda$ is complete for $\mathcal{C}$, it follows that $\bigwedge\Sigma' \to \bot \in \Lambda$ — in contradiction to the assumption that $\Sigma$ is $\Lambda$-consistent. This completes the proof. $\lhd$

## 3.6 The modal logic S4

A popular modal logic is the modal logic **S4**. In section 3.1 we defined **S4** as the modal logic **KT4**. That is, **S4** is the smallest normal modal logic (with a single modal connective) that contains the axioms:

**T** $\quad \Box p \to p \quad$ (alternatively: $p \to \Diamond p$)

**4** $\quad \Box p \to \Box\Box p \quad$ (alternatively: $\Diamond\Diamond p \to \Diamond p$)

Moreover we have already shown:

**Corollary 3.24.** ***S4** is characterized by the class of reflexive and transitive Kripke frames, i.e., the frames for **S4** are exactly the reflexive and transitive Kripke frames. In particular, **S4** is canonical.*

*Proof.* See Propositions 3.1, 3.2, and 3.17. $\lhd$

**Corollary 3.25.** ***S4** is sound and strongly complete for the class of reflexive and transitive Kripke frames.*

*Proof.* The claim follows immediately from the previous corollary and Proposition 3.20.

A more direct proof proceeds as follows: Soundness follows from Propositions 3.1 and 3.2: axioms **K**, **T**, and **4** are valid in all reflexive and transitive Kripke frames. As we noted in the remark after Lemma 3.8, all rules of **S4** (or $K(\mathbf{T},\mathbf{4})$) preserve validity with respect to reflexive and transitive frames.

Weak completeness follows from the fact that the canonical frame of **S4** is a frame for **S4**: assume $\varphi$ is not contained in **S4**, i.e., it is not provable in $K(\mathbf{T},\mathbf{4})$. Then $\{\neg\varphi\}$ is **S4**-consistent, hence contained in a maximal **S4**-consistent set of formulae $\Sigma$. In the canonical model of **S4**, then, $\mathcal{M}^{\mathbf{S4}} \not\models_{\Sigma} \varphi$. Since this model is based on a reflexive and transitive frame (see Proposition 3.14), it follows that $\varphi$ is not valid for the class of that frames. For strong completeness the proof is similar. $\lhd$

**Corollary 3.26.** *S4 is compact.*

*Proof.* This follows from the fact that **S4** is strongly complete for the class of its frames.
Of course, one can also argue with Proposition 3.23: reflexivity and transitivity can be expressed as sentences in the first-order frame language. $\lhd$

**Remark 3.15.** **S4** also has an interesting characterization in terms of the filter semantics discussed in the exercises. Assume that instead of relational frames we consider "frames" consisting of a set of states, $S$, and a function that assigns to each state $s$ a set filter $N(s)$ of subsets of $S$ (the elements of $N(s)$ are called *neighborhoods* of $s$). Then axiom **T** corresponds to the condition that each neighborhood of each state $s$ contains state $s$ as an element. And axiom **4** corresponds to the condition that each neighborhood of a state $s$ contains an *open neighborhood*: a neighborhood is said to be *open* if it is a neighborhood for all its elements, and *closed* if its set-theoretical complement in $S$ is open. Thus, the filter system becomes a "neighborhood system", a *topological space* in terms of point-set topology. The interior of a set of states $X$, $X^{int}$, is the largest open set that is contained in $X$. The closure of a set of states $X$, $X^{cls}$, is the smallest closed set in which $X$ is contained. Define models on top of such frames in the usual way and use the following clauses for defining the satisfaction relation:

$$\mathcal{M} \models_s \Box\varphi \iff s \in \{s' \in S : \mathcal{M} \models_{s'} \varphi\}^{int}$$
$$\mathcal{M} \models_s \Diamond\varphi \iff s \in \{s' \in S : \mathcal{M} \models_{s'} \varphi\}^{cls}$$

Then **T** corresponds to the condition that $X^{int} \subseteq X$ for each set of states $X$ (or equivalently, that $X \subseteq X^{cls}$, for $X \subseteq S$). And **4** corresponds to the condition that $X^{int} \subseteq (X^{int})^{int}$ (or equivalently, $(X^{cls})^{cls} \subseteq X^{cls}$). To sum up, **S4** is the normal modal logic of arbitrary topological spaces.

In the next chapter we will learn more on the complexity of checking whether an **S4**-formula is satisfiable.

An important feature of **S4** is that this logics allows for *reducing modalities*. In this context a modality is any sequence of the logical symbols $\{\neg,\Diamond,\Box\}$ (all of them considered primitives). The empty sequence is denoted by $-$. Thus, for example, $-$, $\neg$, $\Box$, $\Diamond$, and $\neg\Diamond\Diamond\neg\Box$ are modalities. An *affirmative modality* is a modality in which the negation symbol does not occur. Modalities $X$ and $X'$ are called *equivalent* with respect to a normal modal logic $\Lambda$ if $\Lambda$ contains the formula $X\,p \leftrightarrow X'\,p$.

**S4** has up to equivalence only 14 different modalities. To prove this, we first show the following lemma:

**Lemma 3.27.** *The following formulae are* **S4***-theorems:*

(a) $\Box\Box p \leftrightarrow \Box p$

(b) $\Diamond\Diamond p \leftrightarrow \Diamond p$

(c) $\Box\Diamond\Box\Diamond p \leftrightarrow \Box\Diamond p$

(d) $\Diamond\Box\Diamond\Box p \leftrightarrow \Diamond\Box p$

*Proof.* Obviously, (a) and its mirror formulae (b) directly follow from axioms **T** and **4**.

For (c) we may argue as follows (see Lemma 3.7): Since $\Box\Diamond p \rightarrow \Diamond p$ is in **S4**, so is $\Diamond\Box\Diamond p \rightarrow \Diamond\Diamond p$. Hence, by (b), $\Diamond\Box\Diamond p \rightarrow \Diamond p$ is in **S4** and consequently $\Box\Diamond\Box\Diamond p \rightarrow \Box\Diamond p$ is in in **S4**. For the other direction, by axiom **T**, $\Box\Diamond p \rightarrow \Diamond\Box\Diamond p$ is in **S4**. Hence, $\Box\Box\Diamond p \rightarrow \Box\Diamond\Box\Diamond p$ and then, by axiom **4**, $\Box\Diamond p \rightarrow \Box\Diamond\Box\Diamond p$ are in **S4**. Thus, $\Box\Diamond p \leftrightarrow \Box\Diamond\Box\Diamond p$ is in **S4**, since each modal logic is closed with respect to conjunctions of formulae contained in it.

(d) follows immediately from (c). ◁

**Proposition 3.28.** *In S4 each modality is equivalent to one of the modalities*

$$-,\ \Box,\ \Diamond, \Box\Diamond,\ \Diamond\Box,\ \Box\Diamond\Box, \Diamond\Box\Diamond$$

*or a negation of them.*

*Proof.* The proof proceeds by induction of the length of modalities. Let us assume that $X$ is a modality for which the claim has already been proven, that is, in **S4** $X$ is equivalent to one of the modalities in the set

$$\mathscr{B} = \{-, \neg, \Box, \neg\Box, \Diamond, \neg\Diamond, \Box\Diamond, \neg\Box\Diamond, \Diamond\Box, \neg\Diamond\Box, \Box\Diamond\Box, \neg\Box\Diamond\Box, \Diamond\Box\Diamond, \neg\Diamond\Box\Diamond\}.$$

We have to show that $\neg X$, $\Box X$, and $\Diamond X$ are equivalent to one of these modalities as well. But obviously for each modality $X \in \mathscr{B}$, each of $\neg X$, $\Diamond X$, and $\Box X$ is equivalent to some modality in $\mathscr{B}$ (see Lemma 3.27). That is, whenever $X$ is equivalent to some modality in $\mathscr{B}$, the same holds true for $\neg X$, $\Diamond X$, and $\Box X$. ◁

The diagram of all affirmative **S4** modalities is depicted in Figure 3.1. In the graph an arc between modalities $X$ and $X'$ means that $X\,p \rightarrow X'\,p$ is an **S4**-theorem (transitive links are omitted). Notice that the set of 14 "basic" modalities cannot be further reduced in **S4** (why?).

## 3.7 The modal logic S5

Let us now turn to the modal logic **S5**, which was defined in section 3.1 as the modal logic **KTE**. Thus, **S5** is the smallest normal modal logic (with a single modal connective) that contains the axioms:

| | | |
|---|---|---|
| **T** | $\Box p \rightarrow p$ | (alternatively: $p \rightarrow \Diamond p$) |
| **E** | $\Diamond p \rightarrow \Box\Diamond p$ | (alternatively: $\Diamond\Box p \rightarrow \Box p$) |

It is an easy exercise to verify that **S5** is identical to the modal logic **KT4B**, that is, **S5** is an extension of **S4**.

Let us sum up basic statements about **S5** that have been proven in sections section 2.1 and section 3.2.

$$\begin{tikzcd}
 & & \Box \arrow[dl] \arrow[ddrr] & & \\
 & \Box\Diamond\Box \arrow[dl] \arrow[dr] & & & \\
\Box\Diamond \arrow[dr] & & \Diamond\Box \arrow[dl] & & - \arrow[ddll] \\
 & \Diamond\Box\Diamond \arrow[dr] & & & \\
 & & \Diamond & &
\end{tikzcd}$$

Figure 3.1: Affirmative **S4** modalities

**Corollary 3.29.** *$\mathbf{S5}$ is characterized by the class of reflexive and Euclidean Kripke frames, i.e., the frames for $\mathbf{S5}$ are exactly the reflexive and Euclidean Kripke frames.* $\lhd$

Since the reflexive and Euclidean frames are exactly those frames that are reflexive, symmetric, and transitive (see Lemma 2.1), we obtain the following alternative characterization:

**Corollary 3.30.** *$\mathbf{S5}$ is characterized by the class of reflexive, symmetric, and transitive Kripke frames, i.e., the frames for $\mathbf{S5}$ are exactly that Kripke frames in which the accessibility relation is an equivalence relation.*

**Corollary 3.31.** *$\mathbf{S5}$ is sound and strongly complete for the class of Kripke frames in which the accessibility relation is an equivalence relation. Moreover, $\mathbf{S5}$ is compact.*

*Proof.* Analogous to the proofs of Corollaries 3.25 and 3.26. Here we use Propositions 3.1 and 3.3 (or alternatively, Propositions 3.1, 3.2, and 3.4). $\lhd$

Notice that the accessibility relation in each universal frame (i.e., the accessibility relation is universal) is an equivalence relation. Conversely, in a Kripke frame or Kripke model based on an equivalence relation each equivalence class forms a subframe or submodel (see Definition 2.10) in which the accessibility relation is universal.

**Corollary 3.32.** *$\mathbf{S5}$ is sound and strongly complete for the class of universal Kripke frames.*

*Proof.* Soundness is clear, as each universal frame trivially is a reflexive, symmetric, and transitive frame. For strong completeness, we may argue as follows: Let $\Sigma$ be a set of formulae that is **S5**-consistent. By the Lindenbaum Lemma 3.11, we may assume that $\Sigma$ is already maximal $\Lambda$-consistent. By Proposition 3.14, the accessibility relation $R_{\Diamond}^{\mathbf{S5}}$ of the canonical frame $\mathcal{F}^{\Lambda}$ is an equivalence relation on $S^{\Lambda}$. Thus we consider the submodel $\mathcal{M}_{\Sigma}^{\mathbf{S5}}$ of the canonical model $\mathcal{M}^{\mathbf{S5}}$ that is generated by $\Sigma$. The states in this submodel are exactly the states in the equivalence class of $\Sigma$ with respect to $R_{\Diamond}^{\mathbf{S5}}$ and hence this model is universal. Clearly, $\mathcal{M}_{\Sigma}^{\mathbf{S5}} \models_{\Sigma} \varphi$ for each $\varphi \in \Sigma$. This shows that $\Sigma$ is satisfiable in a universal frame. $\lhd$

Since **S4** is contained in **S5**, all **S4**-theorems are **S5**-theorems. Moreover, in **S5** we have a series of axioms that allow for reducing modalities.

**Lemma 3.33.** *The following formulae are* **S5***-theorems:*

(a) $\Diamond\Box\varphi \leftrightarrow \Box\varphi$

(b) $\Box\Diamond\varphi \leftrightarrow \Diamond\varphi$

(c) $\Box(\varphi\vee\Box\psi) \leftrightarrow (\Box\varphi\vee\Box\psi)$

(d) $\Diamond(\varphi\wedge\Diamond\psi) \leftrightarrow (\Diamond\varphi\wedge\Diamond\psi)$

(e) $\Box(\varphi\vee\Diamond\psi) \leftrightarrow (\Box\varphi\vee\Diamond\psi)$

(f) $\Diamond(\varphi\wedge\Box\psi) \leftrightarrow (\Diamond\varphi\wedge\Box\psi)$ $\lhd$

**Proposition 3.34.** *In S5 each modality is equivalent to one of the modalities*

$$-,\ \Box,\ \Diamond$$

*or the negation of one of these.*

*Proof.* It suffices to show that the set of S4 modalities collapses to the given set of modalities. But this follows immediately from Lemma 3.33. $\lhd$

It is clear that in **S5** each finite sequence of $\Box$ and $\Diamond$'s, $O_1\dots O_n$, is equivalent to $O_n$. And the **S4**-diagram of modalities (as shown in Figure 3.1) reduces in **S5** to the graph presented in Figure 3.2)

$$\begin{tikzcd}
\Box \arrow[d] \\
- \arrow[d] \\
\Diamond
\end{tikzcd}$$

Figure 3.2: Affirmative **S5** modalities

**Remark 3.16.** It is worth mentioning that the logic of universal frames is exactly the logic with respect to the valuation semantics introduced in section 1.2. Consequently, the formulae valid in **S5** are exactly those formulae that are valid in each valuation model. Moreover, the procedure presented at the end of that section can be used to decide **S5**-validity of formulae.

## 3.8 The modal logic KL

Let us now turn to the modal logic **KL**, which is an interesting logic, since it will allow us to have a more in-depth view on the concepts discussed in section 3.5. **KL** is the smallest normal modal logic (with a single modal connective) that contains the axiom:

**L** $\quad\Box(\Box p\to p)\to\Box p$

Interestingly, we have that **KL** is an extension of the normal modal logic **K4**.

**Lemma 3.35.** *Axiom* ***4*** *is a theorem of* ***KL****.*

*Proof.*

| | | |
|---|---|---|
| 1. | $(\Box p \wedge \Box\Box p) \to \Box p$ | PL |
| 2. | $p \to ((\Box p \wedge \Box\Box p) \to \Box p)$ | PL: 1 |
| 3. | $p \to ((\Box p \wedge \Box\Box p) \to p)$ | **PL1** |
| 4. | $p \to ((\Box p \wedge \Box\Box p) \to (p \wedge \Box p))$ | PL: 3, 2 |
| 5. | $p \to (\Box(p \wedge \Box p) \to (p \wedge \Box p))$ | Lemma 3.7 (e): 4 |
| 6. | $\Box p \to \Box(\Box(p \wedge \Box p) \to (p \wedge \Box p))$ | Lemma 3.7 (a): 5 |
| 7. | $\Box(\Box p \to p) \to \Box p$ | **L** |
| 8. | $\Box(\Box(p \wedge \Box p) \to (p \wedge \Box p)) \to \Box(p \wedge \Box p)$ | **US**: 7 |
| 9. | $\Box p \to \Box(p \wedge \Box p)$ | PL: 6, 8 |
| 10. | $\Box p \to \Box\Box p$ | PL: 9 |

$\lhd$

Since **4** is contained in **KL**, we know that each frame for **KL** is a transitive frame. But each such frame is also irreflexive (recall that there is no formula that defines irreflexivity of frames).

**Lemma 3.36.** *Each frame for **KL** is irreflexive and transitive.*

*Proof.* Let $\mathcal{F}$ be a frame for **KL**. We have to show that no state sees itself. For proof by contradiction, assume that $s_0 R s_0$ for some state $s_0$. Define a model $\mathcal{M} = \langle \mathcal{F}, V \rangle$ on $\mathcal{F}$ by $V(p) := \{s \in S : s \neq s_0\}$. Thus we obtain that $\mathcal{M} \not\models_{s_0} \Box p$ and hence $\mathcal{M} \models_{s_0} \Box p \to p$. For each $s \in S$ different from $s_0$, $\mathcal{M} \models_s \Box p \to p$ holds trivially. This shows that for each $s \in S$ (in particular for each $s$ with $s_0 R s$), $\mathcal{M} \models_s \Box p \to p$. Thus, $\mathcal{M} \models_{s_0} \Box(\Box p \to p)$. Since $\mathcal{F}$ is a frame for **L**, we obtain $\mathcal{M} \models_{s_0} \Box(\Box p \to p) \to \Box p$, and hence $\mathcal{M} \models_{s_0} \Box p$. From $s_0 R s$ it follows that $\mathcal{M} \models_{s_0} p$—in contradiction to our definition of $V$. This shows that $\mathcal{F}$ is irreflexive.

Secondly, let us suppose that $\mathcal{F}$ is not transitive. Then there exist $s_0$, $s_1$, and $s_2$ such that $s_0 R s_1$, $s_1 R s_2$, but not $s_0 R s_2$. Define a model $\mathcal{M}$ on $\mathcal{F}$ by $V(p) := \{s \in S : s \neq s_1 \text{ and } s \neq s_2\}$. Consider an arbitrary $s \in S$ with $s_0 R s$. By assumption, $s$ is distinct from $s_2$. If $s = s_1$, then from $\mathcal{M} \not\models_{s_2} p$ it follows that $\mathcal{M} \not\models_s \Box p$ and thus $\mathcal{M} \models_s \Box p \to p$. If $s \neq s_1$, then $\mathcal{M} \models_s p$ and hence $\mathcal{M} \models_s \Box p \to p$. Thus, we have shown that for each $s$ with $s_0 R s$, $\mathcal{M} \models_s \Box p \to p$. This shows that $\mathcal{M} \models_{s_0} \Box(\Box p \to p)$. By axiom **L**, we can conclude that $\mathcal{M} \models_{s_0} \Box p$, and hence $\mathcal{M} \models_{s_1} p$—in contradiction to the definition of $V$. This shows that $\mathcal{F}$ is transitive. $\lhd$

A further interesting property of **KL** is that its frames do not contain infinite $R$-chains: an *$R$-chain* is a (finite or infinite) sequence $s_0, s_1, s_2, \ldots$ of worlds such that for each $i \geq 0$, $s_i R s_{i+1}$ (cp. Definition 2.1).

**Lemma 3.37.** *In each frame $\mathcal{F}$ for **KL**, the converse of the accessibility relation is well-founded.*

*Proof.* Let $\mathcal{F} = \langle S, R \rangle$ be a Kripke frame such that $S$ contains an infinite $R$-chain $s_0, s_1, s_2, \ldots$. Define a model $\mathcal{M}$ on $\mathcal{F}$ by $V(p) := \{s \in S : s \neq s_i \text{ for each } i \geq 0\}$. By definition of the model and $s_0 R s_1$, we obtain that $\mathcal{M} \not\models_{s_0} \Box p$. Let now $s \in S$ with $s_0 R s$. If $s$ is one of the states $s_i$ in the chain, then $s_i$ has an $R$-successor in which $p$ is false. That means $\mathcal{M} \not\models_{s_i} \Box p$ and hence $\mathcal{M} \models_s \Box p \to p$. If $s$ is none of the states $s_i$, then $\mathcal{M} \models_s \Box p \to p$ holds trivially. This shows that for each $s_0 R s$, $\mathcal{M} \models_s \Box p \to p$, and thus $\mathcal{M} \models_{s_0} \Box(\Box p \to p)$. In summary, in the given model it holds that $\mathcal{M} \models_{s_0} \Box(\Box p \to p)$, but $\mathcal{M} \not\models_{s_0} \Box p$. Hence, in $\mathcal{F}$ **L** is not valid, i.e., $\mathcal{F}$ is not a frame for **KL**. $\lhd$

**Proposition 3.38.** *The frames for **KL** are the transitve and converse well-founded Kripke frames.*

*Proof.* We show that **L** defines the class of transitive and converse well-founded Kripke frames. Notice that if a frame is converse well-founded, it is also irreflexive.

By Lemma 3.36 and Lemma 3.37, we already know that each frame $\mathcal{F}$ with $\mathcal{F} \models \mathbf{L}$ is transitive and the converse of the accessibility relation is well-founded.

For the other direction, let $\mathcal{F} = \langle S, R \rangle$ be a transitive frame that does not contain an infinite $R$-chain. Let $\mathcal{M}$ be a model defined on $\mathcal{F}$ and $s_0 \in S$ be a state. Assume that $\mathcal{M} \models_{s_0} \Box(\Box p \to p)$, but that $\mathcal{M} \not\models_{s_0} \Box p$. Hence there exists an $s_1 \in S$ such that $s_0 R s_1$ and $\mathcal{M} \not\models_{s_1} p$. Of course, $\mathcal{M} \models_{s_1} \Box p \to p$, and hence $\mathcal{M} \not\models_{s_1} \Box p$. But then there exists an $s_2 \in S$ such that $s_1 R s_2$ and $\mathcal{M} \not\models_{s_2} p$. As $R$ is transitive, we obtain that $s_0 R s_2$ and hence that $\mathcal{M} \models_{s_2} \Box p \to p$. Again we obtain $\mathcal{M} \not\models_{s_2} \Box p$, which entails that there is an $s_3$ with $s_2 R s_3$, and so on. This shows that the $\mathcal{F}$ contains an infinite $R$-chain — in contradiction to the assumption. $\lhd$

**Proposition 3.39.** *KL is sound for the class of finite strict partial orders. Thus, KL is also sound for the class of finite transitive trees.*

*Proof.* The first claim follows from the fact that each finite strict partial order is a transitive frame in which the accessibility relation is converse well-founded (note that by irreflexivity and transitivity a finite frame cannot contain an infinite $<$-chain).
The second claim will be proven later. $\lhd$

In the next section we will present methods that will help to establish also weak completeness results. That is, **KL** is sound and weakly complete for the class of finite strict partial orders (or alternatively, for the class of finite transitve trees). In what follows, however, we will see that **KL** is not strongly complete for any class of frames. In particular, **KL** is not compact.

**Theorem 3.40.** *KL is not compact.*

*Proof.* We show that there exists a **KL**-consistent set of $\mathcal{L}_\tau$-formulae, $\Phi$, that is not satisfiable in any frame for **KL**. For this, consider the set of formulae

$$\Phi := \{\Diamond p_1\} \cup \{\Box(p_i \to \Diamond p_{i+1}) : i \geq 1\}.$$

First, we show that $\Phi$ is not satisfiable in any frame for **KL**. For proof by contradiction, assume that $\mathcal{M}$ is a model defined on such a frame and $s_0$ is a state such that

$$\mathcal{M} \models_{s_0} \Phi,$$

then $\mathcal{M} \models_{s_0} \Diamond p_1$ and $\mathcal{M} \models_{s_0} \Box(p_1 \to \Diamond p_2)$. Hence there exists a state $s_1$ with $s_0 R s_1$ such that $\mathcal{M} \models_{s_1} p_1$ and hence $\mathcal{M} \models_{s_1} \Diamond p_2$. Then there exists a state $s_2 \in S$ such that $s_1 R s_2$ (hence by transitivity, $s_0 R s_2$) and $\mathcal{M} \models_{s_2} \Diamond p_3$, etc. Thus we obtain an infinite chain that is contained in $S$ — in contradiction to Lemma 3.37.

We now have to show that $\Phi$ is **KL**-consistent. Assume that this is not the case. Then there exists a finite subset $\Phi'$ of $\Phi$ such that $\bigwedge \Phi' \to \bot \in \mathbf{KL}$. Obviously, there exists an $n \geq 1$ such that $\Phi'$ is a subset of

$$\Phi_n := \{\Diamond p_1\} \cup \{\Box(p_i \to \Diamond p_{i+1}) : i < n\}$$

and hence it holds that $\bigwedge \Phi_n \to \bot \in \mathbf{KL}$. Since **KL** is sound for the class of all finite partially ordered sets, $\bigwedge \Phi_n \to \bot$ is valid in each such frame, i.e., $\bigwedge \Phi_n$ is not satisfiable in any finite partially ordered set. However, the following model $\mathcal{M}_n$ gives a counterexample:

$$S := \{0, \ldots, n+1\}, \quad R := <, \quad V(p_i) := \{i\}.$$

since $\mathcal{M}_n \models_0 \Diamond p_1$ and for each $s \in S$ with $s > 0$ and $\mathcal{M}_n \models_s p_i$, it follows $s = i$ and hence $\mathcal{M} \models_s \Diamond p_{i+1}$. Thus, $\mathcal{M} \models_0 \Box(p_i \to \Diamond p_{i+1})$, and consequently, $\mathcal{M}_n \models_0 \Phi_n$.

In summary we have shown that $\Phi$ is a **KL**-consistent set of formulae that is not satisfiable in any frame for **KL**. Thus, **KL** is not compact. $\lhd$

**Remark 3.17.** By a similar argument one can show that **KL** is not sound and strongly complete for any class of frames. This also shows that **KL** is not canonical (see Proposition 3.20).

## Bibliographic remarks

A general reference for the material presented in this chapter is Blackburn et al. (2002), chapter 1 and chapter 2. A quite comprehensive list of modal logics and their corresponding frame properties can be found in the appendix of Hughes and Cresswell (1996). Non-normal modal logics are discussed, for example, in Chellas (1980).

# 4 Decidability and Complexity

As has been pointed out several times, an important reason for the popularity of modal logics is the fact that many modal logics are decidable. In the light of section 2.5 and 2.7, many modal logics deal with decidable fragments of first order logic. In this section, we shall present some techniques for proving decidability results in modal logic.

To recall important concepts: A formula $\varphi$ is said to be *satisfiable* in a frame $\mathcal{F}$ if there exists a Kripke model $\mathcal{M}$ defined on $\mathcal{F}$ and a state $s$ of $\mathcal{F}$ such that $\mathcal{M} \models_s \varphi$. $\varphi$ is *satisfiable in a class of frames, $\mathcal{C}$,* if it is satisfiable in a frame in $\mathcal{C}$. $\varphi$ is said to be *valid in $\mathcal{C}$* if $\mathcal{F} \models \varphi$. Decidability results, then, provide answers to the following problems:

- SAT$(\mathcal{C})$: Is a given formula $\varphi$ satisfiable in $\mathcal{C}$?
- VAL$(\mathcal{C})$: Is a given formula $\varphi$ valid in $\mathcal{C}$?

Since a formula $\varphi$ is valid in $\mathcal{C}$ if and only if $\neg\varphi$ is not satisfiable in $\mathcal{C}$, we can reduce (as is done in propositional logic) each of these problems to the other one. For if we have found a procedure or an algorithm that decides SAT$(\mathcal{C})$ for any formula $\varphi$ the same procedure can be used for deciding VAL$(\mathcal{C})$ for $\neg\varphi$, and vice versa.

In section 1.2 we did already present a procedure for deciding validity in **S5**, i.e., we proved that VAL(**S5**) is decidable. But the method used in the proof of that result was tailored to the very special situation in **S5**, namely, that in **S5** each formula is modally equivalent to a formula that has modal depth $\leq 1$. In what is to follow we will discuss decidability results in a more general setting, namely, classes of frames that do not allow for reducing modalities as does **S5**.

## 4.1 Finite Model Property

Let us assume that we want to decide whether some formula $\varphi$ is satisfiable in a given class of Kripke frames, $\mathcal{C}$. If the formula is unsatisfiable, we need to check all the various models that are definable on some of the maybe very huge frames contained in that class. For some classes of frames it is possible to restrict the class of models to "small", countable models. But in case such a model is defined on an infinite set of states, it is, in general, not decidable whether a given formula is satisfiable in that model. Hence the following property of classes of frames is essential for proving decidability results.

**Definition 4.1.** A class of Kripke frames, $\mathcal{C}$, is said to have the *finite model property (FMP)* if each formula that is satisfiable in $\mathcal{C}$ is satisfiable in a finite frame contained in $\mathcal{C}$. A normal modal logic $\Lambda$ is said to have the *finite model property* if the class of its frames has the finite model property.

The finite model property does not guarantee decidability. But we will now prove that if a class of frames, $\mathcal{C}$, has the finite model property *and* furthermore it is finitely axiomatizable, then VAL$(\mathcal{C})$ is decidable.

**Definition 4.2.** A class of frames, $\mathcal{C}$, is said to be *finitely axiomatizable* if there exists a finite set of $\mathcal{L}_\tau(P)$-formulae, $\Sigma$, such that $\Lambda(\mathcal{C}) = \Lambda(\Sigma)$.

For example, the class of frames for **K**-, **KD**, ..., **S4**-, and **S5** are finitely axiomatizable.

**Lemma 4.1.** *Let $\mathcal{M} = \langle S, R, V\rangle$ be a finite Kripke model. Then for each $s \in S$ and each $\mathcal{L}_\tau(P)$-formula $\varphi$, it is decidable in time $\mathcal{O}\left(\|\varphi\| \cdot \|\mathcal{M}\|\right)$ whether $\mathcal{M} \models_s \varphi$.*

Here and in what is to follow $\|\mathcal{M}\|$ denotes $|S| + |R|$.

*Proof.* Let $\varphi_1, \ldots, \varphi_n$ be an enumeration of all subformulae of $\varphi$, in the order of degree, i.e., $\varphi = \varphi_n$. For each $1 \leq i \leq n$, label each state $s \in S$ by '$\varphi_i$' or '$\neg\varphi_i$' depending on whether $\mathcal{M} \models_s \varphi_i$ or $\mathcal{M} \models_s \neg\varphi_i$. Finally return '$\varphi$ is satisfiable' if $s$ is labeled with "$\varphi$". Otherwise, return '$\varphi$ is not satisfiable'.

Obviously, this algorithm terminates after $n = |\mathrm{Sub}(\varphi)|$ rounds. In each round $|S|$ states are to be checked. If in the $i$-th round $\varphi_i$ has the form $\neg\psi$ or $(\psi \vee \psi')$, each state has already been labeled in an earlier round by '$\psi$' or by '$\neg\psi$' (and by '$\psi'$' or by '$\neg\psi'$'). So it can immediately be seen, whether $\mathcal{M} \models_s \varphi_i$ holds or not. If in the $i$-th round $\varphi_i$ has the form $\Diamond\psi$, the algorithm has to check whether there is a pair of states $(s', s'')$ where $s'$ can see $s''$ via $R$ and $s''$ is labeled by '$\psi$', or not. Thus, each round needs at most $\mathcal{O}(\max(|S|, |R|)) = \mathcal{O}(\|\mathcal{M}\|)$, and the algorithm terminates in time $\mathcal{O}(\|\varphi\| \cdot \|\mathcal{M}\|)$. $\lhd$

**Theorem 4.2.** *Let $\mathcal{C}$ be a class of Kripke frames that has the finite model property and is finitely axiomatizable. Then VAL($\mathcal{C}$) and SAT($\mathcal{C}$) are decidable.*

*Proof.* We show that both (a) $\Lambda(\mathcal{C})$ and (b) $\mathcal{L}_\tau \setminus \Lambda(\mathcal{C})$ are recursively enumerable. From this it follows that '$\varphi \in \Lambda(\mathcal{C})$' and '$\varphi \notin \Lambda(\mathcal{C})$' are semi-decidable, and hence that '$\varphi \in \Lambda(\mathcal{C})$' is decidable.

(a) By assumption, $\Lambda(\mathcal{C}) = \Lambda(\Sigma)$ for some finite set of $\mathcal{L}_\tau$-formulae, $\Sigma$. But $\Lambda(\Sigma)$ is recursively enumerable, for $\Lambda(\Sigma)$ consists of exactly those formulae that are provable from $\Sigma$. A proof from $\Sigma$ is a finite sequence of formulae $\varphi_1, \ldots, \varphi_n$ where each $\varphi_i$ is (K) or a formula from $\Sigma$ or the result of applying one of the rules (R-MP), (R-US), or (R-$\Box$) on preceding formulae of that sequence. Obviously, the set of all proofs from $\Sigma$ is recursively enumerable.

(b) We have to define a recursive enumeration of all formulae $\varphi$ with $\mathcal{C} \not\models \varphi$. Since $\mathcal{C}$ has the finite model property we may assume that $\mathcal{C}$ is a class of finite frames (otherwise, consider the class of all finite frames contained in $\mathcal{C}$). Note that

1. $\mathcal{C}$ may contain infinitely many, non-isomorphic frames (for that reason, the procedure presented now, is not a decision procedure).
2. Each finite frame $\mathcal{F} = \langle S, R\rangle$ is isomorphic to a Kripke frame on the set $\{1, \ldots, n\}$ where $n = |S|$.
3. For each set $\{1, \ldots, n\}$ there exist $2^{n^2}$ distinct Kripke frames defined on it.
4. For each finite frame $\mathcal{F} = \langle S, R\rangle$ and each finite subset $P'$ of $P$ there exist at most $2^{|S|^m}$ distinct Kripke models, where $m = |P'|$.
5. By Lemma 4.1, for each finite model $\mathcal{M}$ and each formula $\varphi$, it is decidable whether $\mathcal{M} \models \varphi$ for a given formula $\varphi$.

Let now $p_1, p_2, \ldots$ be an enumeration of all propositional variables of $P$. Let $L_0 := \emptyset$ and let $\Phi_0 := \emptyset$. Let $P_\varphi$ denote the set of all propositional variables that occur in $\varphi$. Assume now that $n > 0$ and that $L_k$ and $\Phi_k$ are defined for all $k < n$. Construct all models on the set $\{1, \ldots, n\}$. Check for each such model $\mathcal{M}$ whether $\mathcal{M} \models \Sigma$ (note that $\Sigma$ is finite). If so, then add $\mathcal{M}$ to the set $L_n$ and check for each $\mathcal{L}_\tau$-formula $\varphi$ with $\deg\varphi \leq n$ and $P_\varphi \subseteq \{p_1, \ldots, p_n\}$ whether $\mathcal{M} \not\models \varphi$. If so, add $\varphi$ to the set $\Phi_n$. If $\mathcal{M} \models \varphi$, check whether $\mathcal{M}' \not\models \varphi$ for some model $\mathcal{M}'$ in $L_k$ $(k < n)$. If this is the case, then add $\varphi$ to the set $\Phi_n$ as well.

Thus, we obtain a recursively defined set $\Phi$ of all formulae of $\mathcal{L}_\tau$ that are not valid in $\mathcal{C}$. For if $\varphi$ is not valid, there exists a finite Kripke model $\mathcal{M}$ defined on some frame in $\mathcal{C}$ with $\mathcal{M} \not\models \varphi$.

This model is isomorphic to some model $\mathcal{M}'$ defined on the set $\{1,\dots,n\}$. Hence $\mathcal{M}' \models \Sigma$. If $P_\varphi \subseteq \{p_1,\dots,p_m\}$, $\varphi$ appears in $\Phi$ after at most $\max(m,n)$ steps. $\lhd$

**Definition 4.3.** A class of Kripke frames, $\mathcal{C}$, is said to have the *exponentially bounded model property (EXPMP)* if there is a polynom $p(x)$ in $\mathbb{Z}[x]$ such that each formula $\varphi$ satisfiable in $\mathcal{C}$ is satisfiable in a frame $\mathcal{F} = \langle S,R\rangle$ with $|S| \leq 2^{p(\|\varphi\|)}$. $\mathcal{C}$ is said to have the *polynomially bounded model property (PMP)* if each formula $\varphi$ satisfiable in $\mathcal{C}$ is satisfiable in a frame $\mathcal{F} = \langle S,R\rangle$ with $|S| \leq p(\|\varphi\|)$ where $p(x)$ is a polynom in $\mathbb{Z}[x]$.

Obviously, FMP follows from EXPMP and the latter one follows from PMP.

**Corollary 4.3.** *Let $\mathcal{C}$ be a class of Kripke frames that has the exponentially bounded model property. If '$\mathcal{F} \in \mathcal{C}^{fin}$' is decidable, then SAT($\mathcal{C}$) and VAL($\mathcal{C}$) are decidable.*

Here $\mathcal{C}^{\text{fin}}$ denotes the class of all finite frames contained in $\mathcal{C}$.

*Proof.* Let $\varphi$ be any formula. Enumerate first all Kripke models that are defined on a frame with at most $2^{p(|\varphi|)}$ states. For each of these models, check whether $\mathcal{M}$ is defined on a frame in $\mathcal{C}^{\text{fin}}$. If this is the case, check whether $\mathcal{M} \models_s \varphi$ for some state $s$ of $\mathcal{M}$. If so, return '$\varphi$ is satisfiable in $\mathcal{C}$' and halt. If not, continue with the next model. Finally, if no model has been found that satisfies $\varphi$ in some state, return '$\varphi$ is not satisfiable in $\mathcal{C}$'. $\lhd$

Obviously, '$\mathcal{F} \in \mathcal{C}^{\text{fin}}$' is decidable for many of the frame classes we introduced in the preceding sections. For example, it can easily be checked by an algorithm whether a given finite frame is reflexive, transitive, etc. (A more general result: '$\mathcal{F} \in \mathcal{C}^{\text{fin}}$' is decidable whenever $\mathcal{C}^{\text{fin}}$ is defined by some first-order formula.) Thus, for each of these classes of Kripke frames, we have reduced the problem of proving decidability to the question whether the frame class under consideration has some bounded model property.

**Remark 4.1.** In order to prove the finite model property for **K**, we could argue as follows: Let us assume that $\varphi$ is satisfiable in the class of all Kripke frames. Since **K** is sound with respect to that class of frames, $\varphi$ must be **K**-consistent. Now the canonical way of proving that $\varphi$ is satisfiable (which we already know, of course) is to look at the canonical model of **K**. However, this model is defined on an infinite frame. But we can also define a finite 'canonical model' that behaves similar (in many respects) to the real canonical model. Thereto, consider the set of all subformulae of $\varphi$, Sub($\varphi$). Define a model $\mathcal{M}_\varphi := \langle S_\varphi, R_\varphi, V_\varphi\rangle$ as follows: Let $S_\varphi$ be the set of all subsets $\Gamma$ of $\text{Sub}(\varphi) \cup \{\neg\psi : \psi \in \text{Sub}(\varphi)\}$ that are $\varphi$-maximal consistent in the following sense: (a) $\Gamma$ is consistent with respect to **K**, and (b) for each subformula $\psi$ of $\varphi$, $\psi \in \Gamma$ or $\neg\psi \in \Gamma$. Then, let $R_\varphi$ and $V_\varphi$ be defined as in the canonical model. It can easily be seen that the *finite* model constructed in this way satisfies $\varphi$.

## 4.2 Filtration

To prove that a given class of Kripke frames has the finite model property (or better: some bounded model property), we have to provide a method to transform an infinite model of that class into a finite model that is also contained in that class. The remark discussed at the end of the last subsection presents the red line along which such a method can be developed.

Let $\mathcal{M}$ be a Kripke model and $\Gamma$ be a set of formulae. We say that states $s$ and $s'$ are $\Gamma$-*equivalent*, $s \sim_\Gamma s'$, if for each formula $\gamma \in \Gamma$,

$$\mathcal{M} \models_s \gamma \iff \mathcal{M} \models_{s'} \gamma.$$

Using the notations from section 2.7, $s$ and $s'$ are $\Gamma$-equivalent if and only if

$$T_{\mathcal{M}}(s) \cap \Gamma = T_{\mathcal{M}}(s') \cap \Gamma.$$

Let $s_\Gamma$ denote the equivalence class of $s$ with respect to $\Gamma$-equivalence.

**Lemma 4.4.** *Let $\mathcal{M} = \langle S, R, V\rangle$ be a Kripke model and let $\Gamma$ be a finite set of $\mathcal{L}_\tau$-formulae. Then $|S/{\sim_\Gamma}| \leq 2^{|\Gamma|}$.*

*Proof.* For each pair of states $s$ and $s'$, $s$ and $s'$ are $\Gamma$-equivalent if and only if $T_{\mathcal{M}}(s) \cap \Gamma = T_{\mathcal{M}}(s') \cap \Gamma$. Hence the mapping $\iota : S/{\sim_\Gamma} \to 2^\Gamma$, $s_\Gamma \mapsto T_{\mathcal{M}}(s) \cap \Gamma$ is well-defined and injective. Thus, $|S/{\sim_\Gamma}| \leq 2^{|\Gamma|}$. $\triangleleft$

**Definition 4.4.** A set of formulae, $\Sigma$, is said to be *closed under subformulae* if for each $\varphi \in \Sigma$, $\mathrm{Sub}(\varphi) \subseteq \Sigma$.

**Definition 4.5.** Let $\mathcal{M}$ be a Kripke model and let $\Gamma$ be a set of formulae that is closed under subformulae. A model $\mathcal{M}' = \langle S', R', V'\rangle$ is said to be a *filtration* of $\mathcal{M}$ *through* $\Gamma$ if each of the following conditions is satisfied:

(a) $S' = S/{\sim_\Gamma}$;

(b) For all $s, s' \in S$, $s\,R\,s'$ implies $s_\Gamma\,R'\,s'_\Gamma$;

(c) For all $s, s' \in S$, if $s_\Gamma\,R'\,s'_\Gamma$, $\Diamond\varphi \in \Gamma$, and $\mathcal{M} \models_{s'} \varphi$, then $\mathcal{M} \models_s \Diamond\varphi$;

(d) For each $p \in P$, $s_\Gamma \in V'(p)$ if and only if $s \in V(p)$.

By Lemma 4.4, each filtration through a finite set of formulae, $\Gamma$, is a model that has at most $2^{|\Gamma|}$ distinct states.

In general, there can be many filtrations through a set of formulae. Two borderline cases are given as follows: let $\mathcal{M}'$ be a filtration of $\mathcal{M}$ through a set of formulae, $\Gamma$. $\mathcal{M}'$ is said to be the *finest* $\Gamma$*-filtration* if for all states $s$ and $t$,

$$s_\Gamma\,R'\,t_\Gamma \iff \text{there exist } s' \in s_\Gamma, t' \in t_\Gamma \text{ with } s'\,R\,t'. \tag{1}$$

$\mathcal{M}'$ is the *coarsest* $\Gamma$*-filtration* if for all states $s$ and $t$,

$$s_\Gamma\,R'\,t_\Gamma \iff \text{for all } \Diamond\varphi \in \Gamma, \text{ if } \mathcal{M} \models_t \varphi \text{ then } \mathcal{M} \models_s \Diamond\varphi. \tag{2}$$

It is worth noting that the defining clause in equation (2) follows from that in equation (1). To see this, let $\Diamond\varphi$ be any formula in $\Gamma$ with $\mathcal{M} \models_t \varphi$. Choose $s' \in s_\Gamma$ and $t' \in t_\Gamma$ such that $s'\,R\,t'$. Then $\mathcal{M} \models_{t'} \varphi$, consequently $\mathcal{M} \models_{s'} \Diamond\varphi$, and thus $\mathcal{M} \models_s \Diamond\varphi$.

**Theorem 4.5.** *Let $\Gamma$ be a finite set of $\mathcal{L}_\tau$-formulae that is closed under subformulae. Let $\mathcal{M}$ be a Kripke model and let $\mathcal{M}'$ be a filtration of $\mathcal{M}$ through $\Gamma$. Then*

$$T_{\mathcal{M}}(s) \cap \Gamma = T_{\mathcal{M}'}(s') \cap \Gamma,$$

*i.e., for each $\gamma \in \Gamma$ and each state $s$ of $\mathcal{M}$,*

$$\mathcal{M} \models_s \gamma \iff \mathcal{M}' \models_{s_\Gamma} \gamma.$$

*Proof.* Straight forward by induction on the degree of $\gamma$. Note first that each subformula of $\gamma$ is contained in $\Gamma$, too. $\triangleleft$

**Theorem 4.6.** *The normal modal logic* ***K*** *has the exponentially bounded model property. In particular, each formula satisfiable in some Kripke model is satisfiable in a Kripke model with at most* $2^{\|\varphi\|}$ *states.*

*Proof.* By assumption, there exist a model $\mathcal{M}$ and a state $s$ such that $\mathcal{M} \models_s \varphi$. Set $\Gamma := \mathrm{Sub}(\varphi)$. Then $\Gamma$ is a finite set that is closed under subformulae. Furthermore, $|\Gamma| \leq \|\varphi\|$. Let $\mathcal{M}'$ be the coarsest or the finest filtration of $\mathcal{M}$ through $\Gamma$. Then $\mathcal{M}'$ has at most $2^{|\Gamma|} \leq 2^{\|\varphi\|}$ states and $\mathcal{M}' \models_{s_\Gamma} \varphi$. ◁

As an immediate conclusion of Corollary 4.3, we thus obtain the following corollary.

**Corollary 4.7.** *SAT(**K**) and VAL(**K**) are decidable.* ◁

At this point it is worthwhile to reconsider the proof of Theorem 4.6. There the finest filtration *and* the coarsest filtration were sufficiently small models, since both filtrations are defined on a frame for **K**. In general this is not the case. For example, in general the coarsest filtration of a model does not preserve symmetry. To see this, let us consider the set $\Gamma := \{p, \Diamond p\}$, which is closed under subformulae, and a Kripke model $\mathcal{M} = \langle S, R, V\rangle$ with $S = \{s_0, s_1\}$, $R = \{(s_1, s_1)\}$, and $V(p) = \{s_0\}$. Then the coarsest filtration of $\mathcal{M}$ through $\Gamma$ is given by $\mathcal{M}' = \langle S', R', V'\rangle$ where $S' = \{s'_0, s'_1\}$ ($s'_i$ is the $\Gamma$-equivalence class of $s_i$), and $R' = \{(s'_0, s'_1), (s'_1, s'_1)\}$, which is not symmetric (see Figure 4.1). However, the finest filtration preserves symmetry (as can easily be checked by its definition).

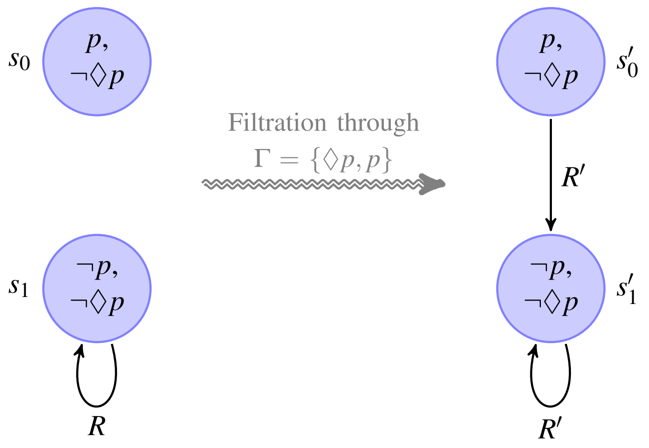


Figure 4.1: Coarsest filtration of a symmetric model

In general, it holds:

**Lemma 4.8.** *Let $\mathcal{M}$ be a Kripke model and let $\mathcal{M}'$ be any filtration of $\mathcal{M}$ through some set of $\mathcal{L}_\tau$-formulae.*

(a) *If $\mathcal{M}$ is serial, then so is $\mathcal{M}'$.*

(b) *If $\mathcal{M}$ is reflexive, then so is $\mathcal{M}'$.* ◁

**Corollary 4.9.** *SAT(**D**) and SAT(**T**) as well as VAL(**D**) and VAL(**T**) are decidable.* ◁

## 4.3 Complexity

In what follows we shall prove that SAT(**S5**) and VAL(**S5**) are decidable. At first sight this result is not very interesting, since we did already prove this fact at the end of section 1.2. However, by proving that SAT(**S5**) is decidable we will prove a much stronger result, namely that reasoning in **S5** falls into the same complexity class as reasoning in propositional logic. This means that although we gain more expressive power by using the language of modal logic, we do not have to pay an essentially more expensive bill for runtime costs.

Some notions from complexity theory: a decision problem '$x \in X$?' is said to *be in NP* if '$x \in X$?' can be decided by a non-deterministic algorithm in polynomial time (in the size of $x$); in other words, if a guessed solution to the problem can be checked in polynomial time. We say that a decision problem '$x \in X$?' is *NP-complete* if '$x \in X$?' is in NP and if it is *NP-hard*, i.e., if each problem '$y \in Y$?' in NP can be polynomially reduced to '$x \in X$?'. A polynomial-time (many-one) reduction from '$y \in Y$?' to '$x \in X$?' is a deterministic algorithm (a recursive function) $f$ that takes as input a word $y$ in the language of '$y \in Y$?' and returns in polynomial time a word $f(y)$ in the language of '$x \in X$?' such that

$$y \in Y \iff f(y) \in X.$$

Note that the existence of a polynomial reduction of one problem to another gives a transitive relation. Hence, in order to prove that '$x \in X$?' is NP-complete it is sufficient to show that

(a) '$x \in X$?' is in NP, and

(b) there exists an NP-complete problem '$y \in Y$?' and a polynomial reduction of '$y \in Y$?' to '$x \in X$?'.

Analogously, further complexity classes may be introduced as follows:

- *EXPTIME:* '$x \in X$?' can be decided by a deterministic algorithm in exponential time in the size of $x$, i.e., in time $\leq 2^{|x|^k}$ for some $k > 0$.
- *2EXPTIME:* '$x \in X$?' can be decided by a deterministic algorithm in double exponential time in the size of $x$, i.e., in time $\leq 2^{2^{|x|^k}}$ for some $k > 0$.
- *PSPACE:* '$x \in X$?' can be decided by a deterministic algorithm that uses polynomially large space in the size of $x$.
- *NPSPACE:* '$x \in X$?' can be decided by a non-deterministic algorithm that uses polynomially large space in the size of $x$.
- etc.

There are some relations between these complexity classes that should be listed in this context:

$$\text{NPSPACE} = \text{PSPACE} \qquad \text{NEXPSPACE} = \text{EXPSPACE}$$

$$\begin{aligned} \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \\ \subseteq \text{EXPSPACE} \subseteq \text{2EXPTIME} \subseteq \text{N2EXPTIME} \end{aligned}$$

It is further known that

$$\begin{array}{rlrl} \text{P} & \neq \text{EXPTIME} & \text{NP} & \neq \text{NEXPTIME} \\ \text{PSPACE} & \neq \text{EXPSPACE} & & \\ \text{EXPTIME} & \neq \text{2EXPTIME} & \text{NEXPTIME} & \neq \text{N2EXPTIME} \end{array}$$

We will start discussing complexity results by presenting modal logics which are not essentially more complex than propositional logic. As we have seen, a formula $\varphi$ is valid in some **S5**-Kripke model $\mathcal{M}$ if and only if for each state $s$ of $\mathcal{M}$, $\varphi$ is valid in the submodel of $\mathcal{M}$ generated by $s$. But each of these submodels is defined on a *universal* frame, i.e., a frame $\mathcal{F} = \langle S, R\rangle$ with $R = S \times S$. Since the class of all universal frames is a (proper) subclass of that of all **S5**-frames, SAT(**S5**) can be reduced to SAT($\mathcal{C}^{\text{univ}}$), where $\mathcal{C}^{\text{univ}}$ denotes the class of all universal frames.

**Lemma 4.10.** *An $\mathcal{L}_\tau$-formula $\varphi$ is satisfiable in $\mathcal{C}^{univ}$ if and only if it is satisfiable in a universal frame $\mathcal{F} = \langle S, R\rangle$ with $|S| \leq \|\varphi\|$.*

*Proof.* The 'if'-direction holds obviously. For the other direction, let $\mathcal{M} = \langle S, R, V\rangle$ be a universal Kripke model and let $s_0$ be a state with $\mathcal{M} \models_{s_0} \varphi$. Define

$$N_\varphi := \{\Diamond\psi \in \text{Sub}(\varphi) : \mathcal{M} \models_{s_0} \Diamond\psi\}.$$

Note that $|N_\varphi| < |\text{Sub}(\varphi)| \leq \|\varphi\|$ (since no propositional variable occurs in $N_\varphi$). Then for each $\Diamond\psi \in N_\varphi$, choose a state $s_\psi \in S$ such that $\mathcal{M} \models_{s_\psi} \psi$. Define a universal Kripke model $\mathcal{M}' = \langle S', R', V'\rangle$ by

$$\begin{aligned} S' &:= \{s_0\} \cup \{s_\psi \in N_\varphi\} \\ R' &:= S' \times S' \\ V'(p) &:= V(p) \qquad (p \in P) \end{aligned}$$

By structural induction it is easy to verify that for each $s \in S'$ and each $\psi \in \text{Sub}(\varphi)$, $\mathcal{M} \models_s \varphi$ if and only if $\mathcal{M}' \models_s \psi$. $\lhd$

**Theorem 4.11.** *SAT(**S5**) is NP-complete and hence VAL(**S5**) is coNP-complete.*

*Proof.* Obviously, SAT(**S5**) is NP-hard since SAT(PL) is NP-complete and polynomially reducible to SAT(**S5**) (trivial because of 'PL $\subseteq$ **S5**').
It remains to be shown that SAT(**S5**) is in NP. Thereto, let $\varphi$ be any formula. To check its satisfiability wrt. **S5** it is sufficient to consider universal models $\mathcal{M}$ with $|\mathcal{M}| \leq \|\varphi\|$ (cf. Lemma 4.10). Then the following non-deterministic algorithm will do the job:

1. "Guess" a model $\mathcal{M}$ with $|S| \leq \|\varphi\|$;
2. Check whether $\mathcal{M}$ is universal;
3. If so, check whether $\mathcal{M} \models_s \varphi$.

Note that step 2 can be done in quadratic time (in time $\leq |S|^2 \leq \|\varphi\|^2$). For step 3, we have to visit at most $|S| \leq \|\varphi\|$ states of the model, and in each of these states we have to check whether $\mathcal{M} \models_s \varphi$, which can be done in time $\mathcal{O}(\|\varphi\| \cdot \|\mathcal{M}\|)$. Since $\|\mathcal{M}\| = |S| + |S|^2 \leq \|\varphi\| + \|\varphi\|^2 \in \mathcal{O}(\|\varphi^2\|)$, the algorithm runs in time $\mathcal{O}\left(\|\varphi\|^4\right)$. $\lhd$

In what is to follow let **S4.3** be the smallest normal modal logic that is obtained from **S4** by adding the axiom

(.3) $\Box(\Box p \to q) \vee \Box(\Box q \to p)$,

i.e., **S4**.3 $=$ **S4**$[(.3)]$. It can easily be proven that **S4**.3 is characterized by the class of all reflexive, transitive, and weakly connected frames. A frame $\langle S,R\rangle$ is said to be *weakly connected* if for all $s,s',s''\in S$ with $s\,R\,s'$ and $s\,R\,s''$, $s'\,R\,s''$ or $s''\,R\,s'$.
The following two theorems, then, present further complexity results of modal logic.

**Theorem 4.12.** *Let $\Lambda$ be a consistent and normal modal logic that contains **S4**.3. Then SAT$(\mathcal{C}(\Lambda))$ is NP-complete.*

**Theorem 4.13.** *For each modal logic $\Lambda\in\{$**S5**,**KD45**,**K4.3**,**KL.3**$\}$, the validity problem '$\varphi\in\Lambda$?' is coNP-complete.*

$$\begin{tikzcd}
 & & {} \\
 & \textbf{S5} \arrow[ur, dashed] & \\
\textbf{S4.3} \arrow[ur] & \textbf{KD45} \arrow[u] & \textbf{KL.3} \arrow[uu, dashed] \\
 & \textbf{K4.3} \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 4.2: Logics above **K4.3**

To summarize the results presented so far, some modal logics allow to consider only very small models for finding a satisfying model. The general situation, however, is not such good. To see this, we will show that there exists a formula $\varphi$ that is satisfiable in some Kripke model $\mathcal{M}$ if and only if this model contains a binary tree.

**Theorem 4.14.** *The modal logic **K** does not have the polynomially bounded model property. More precisely, for each $n>0$ there exists a satisfiable formula $\varphi_n$ such that*

(a) $\|\varphi_n\|\in\mathcal{O}\left(n^2\right)$;

(b) *If $\mathcal{M}$ is a Kripke model and $s_0$ a state of $\mathcal{M}$ with $\mathcal{M}\models_{s_0}\varphi_n$, then the submodel generated by $s_0$ contains a full binary tree of depth $n$.*

*Proof.* Suppose that we have found such a formula $\varphi_n$. Since a full binary tree of depth $n$ contains $2^n$ branches, each model satisfying $\varphi_n$ must contain at least $2^n$ states. This shows that **K** does not have the PMP.

Let now $r_0,r_1,r_2$ be an enumeration of all propositional variables of $\mathcal{L}_\tau$. Set $p_i:=r_{2i}$ and $q_i:=r_{2i+1}$ for $i\in\mathbb{N}$. Then define

$$\beta_i := q_i\rightarrow\Diamond(q_{i+1}\wedge p_{i+1})\wedge\Diamond(q_{i+1}\wedge\neg p_{i+1})$$
$$\tau_i := (p_i\rightarrow\Box p_i)\wedge(\neg p_i\rightarrow\Box\neg p_i)$$

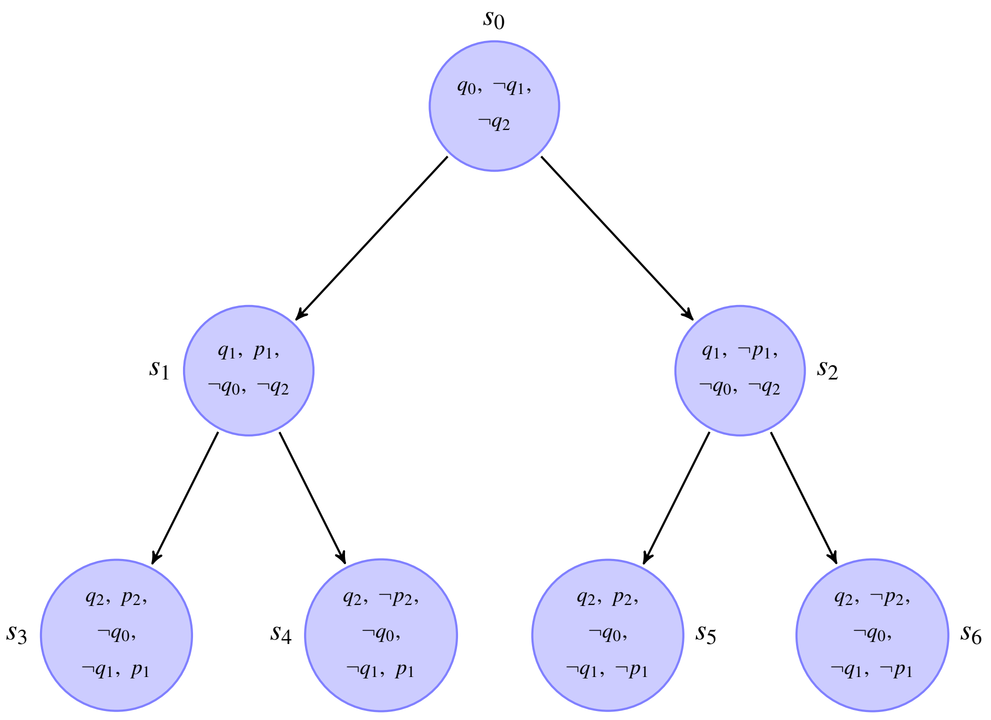
The formula $\beta_i$ expresses binary branching at level $i$, while $\tau_i$ expresses that the truth values of the $p_i$ are inherited to the next level. In the definition of $\varphi_n$ we use the following abbreviations:

$\Box^k\psi := \Box\cdots\Box\psi$, and $\Box^{(k)}\psi := \bigwedge_{i=0}^{k}\Box^i\psi$. Then we set

$$\begin{array}{lll}
\varphi_n := q_0 & & \text{(I)}\\
\wedge & \bigwedge_{i=0}^{n}\Box^{(n)}(q_i \to \bigwedge_{j=0,j\neq i}^{n}\neg q_i) & \text{(II)}\\
\wedge & \beta_0 \wedge \Box\beta_1 \wedge \Box^2\beta_2 \wedge \Box^3\beta_3 \wedge \cdots \wedge \Box^{n-1}\beta_{n-1} & \text{(III)}\\
\wedge & \Box\tau_1 \wedge \Box^2\tau_1 \wedge \Box^3\tau_1 \wedge \cdots \wedge \Box^{n-1}\tau_1 & \text{(IV}_1)\\
\wedge & \Box^2\tau_2 \wedge \Box^3\tau_2 \wedge \cdots \wedge \Box^{n-1}\tau_2 & \text{(IV}_2)\\
\wedge & \Box^3\tau_3 \wedge \cdots \wedge \Box^{n-1}\tau_3 & \text{(IV}_3)\\
\wedge & \vdots & \\
\wedge & \Box^{n-1}\tau_{n-1} & \text{(IV}_{n-1})
\end{array}$$

It is worthwhile to remark that $\|\varphi_n\|$ increases polynomially in $n$. But a model that satisfies $\varphi_{n+1}$ must have twice the size of the smallest model that satisfies $\varphi_n$. Furthermore, it is clear that if some model $\mathcal{M}$ satisfies $\varphi_n$ in some state $s_0$, then the submodel generated by $s$ must contain a binary tree with root $s_0$. The details of the proof are left to the reader. $\lhd$

The model depicted in Figure 4.3 presents a minimal model of $\varphi_2$. Note that there $\Box\tau_1$ guarantees that $s_3 \neq s_5$ and $s_4 \neq s_6$.



$s_0$
$q_0, \neg q_1, \neg q_2$
$s_1$
$q_1, p_1, \neg q_0, \neg q_2$
$s_2$
$q_1, \neg p_1, \neg q_0, \neg q_2$
$s_3$
$q_2, p_2, \neg q_0, \neg q_1, p_1$
$s_4$
$q_2, \neg p_2, \neg q_0, \neg q_1, p_1$
$s_5$
$q_2, p_2, \neg q_0, \neg q_1, \neg p_1$
$s_6$
$q_2, \neg p_2, \neg q_0, \neg q_1, \neg p_1$

Figure 4.3: A minimal model satisfying $\varphi_2$.

In the sequel we will show that SAT(**K**) is PSPACE-hard. Thereto, we will first present a reasoning problem which is known to be PSPACE-complete. A *quantified Boolean formula* (for short: *QBF-formula*) is a formula of the form

$$Q_1p_1\dots Q_np_n\varphi,$$

where $n \geq 0$, each $Q_i \in \{\forall, \exists\}$, and $\varphi$ is a PL-formula with $\operatorname{Var}\varphi \subseteq \{p_1, \dots, p_n\}$. Note that $\bot$ and $\top$ are considered as undefined symbols. Hence, formulae of PL in which no quantifier and no propositional variable occur count as QBF-formulae, too. Semantically, this language is interpreted as follows:

$$\begin{aligned}
&\models \varphi && \iff \varphi \text{ is PL-valid} \\
&\models \forall p_1 Q_2 p_2 \dots Q_n p_n \varphi && \iff \models Q_2 p_2 \dots Q_n p_n\, \varphi(p_1/\top) \text{ and} \\
& && \qquad \models Q_2 p_2 \dots Q_n p_n\, \varphi(p_1/\bot) \\
&\models \exists p_1 Q_2 p_2 \dots Q_n p_n \varphi && \iff \models Q_2 p_2 \dots Q_n p_n\, \varphi(p_1/\top) \text{ or} \\
& && \qquad \models Q_2 p_2 \dots Q_n p_n\, \varphi(p_1/\bot)
\end{aligned}$$

For each valid QBF-formula, we can then construct a tree that witnesses its validity. As an example consider the formula $\forall p_1 \exists p_2 \forall p_3 (p_1 \to p_2 \vee p_3)$. By reading the tree in Figure 4.4 from its leaves, it is clear that the formula in the root of the tree is QBF-valid.

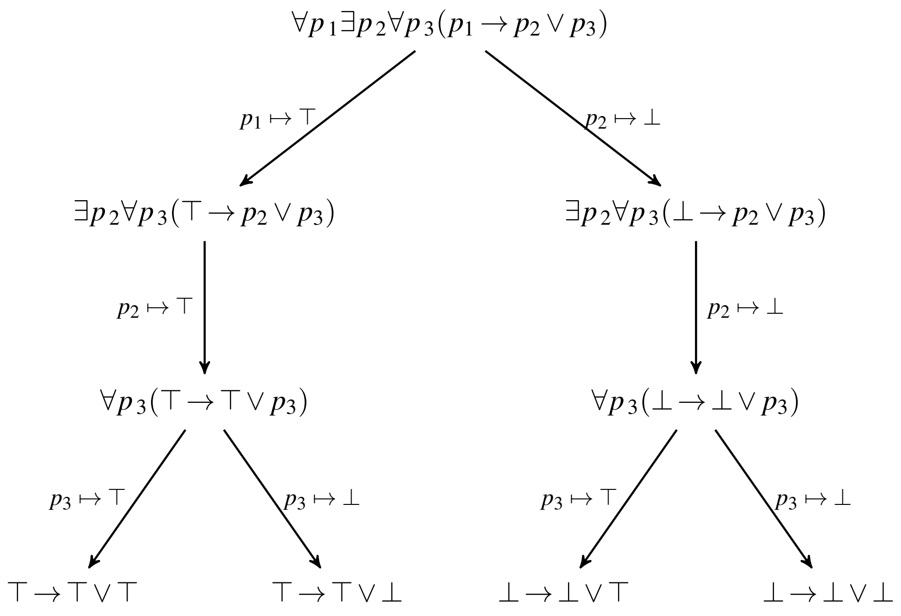

$\forall p_1 \exists p_2 \forall p_3 (p_1 \to p_2 \vee p_3)$
$p_1 \mapsto \top$
$p_2 \mapsto \bot$
$\exists p_2 \forall p_3 (\top \to p_2 \vee p_3)$
$\exists p_2 \forall p_3 (\bot \to p_2 \vee p_3)$
$p_2 \mapsto \top$
$p_2 \mapsto \bot$
$\forall p_3 (\top \to \top \vee p_3)$
$\forall p_3 (\bot \to \bot \vee p_3)$
$p_3 \mapsto \top$
$p_3 \mapsto \bot$
$p_3 \mapsto \top$
$p_3 \mapsto \bot$
$\top \to \top \vee \top$
$\top \to \top \vee \bot$
$\bot \to \bot \vee \top$
$\bot \to \bot \vee \bot$

Figure 4.4: $\forall p_1 \exists p_2 \forall p_3 (p_1 \to p_2 \vee p_3)$ is a QBF-valid formula.

Note that in that tree each level corresponds to exactly one quantifier of the root formula. A second example of a QBF-valid formula is $\forall p_1 \exists p_2 (p_1 \leftrightarrow \neg p_2)$ (see Figure 4.5).
In what is to follow we will sketch the proof of a fundamental theorem by Ladner.

**Theorem 4.15** (Ladner's Theorem)**.** *For each normal modal logic between* ***K*** *and* ***S4****, the problem of testing satisfiability is PSPACE-hard.*

For the proof we will present a translation $\Phi$ from the language of QBF-formulae into the language of modal logic such that

(a) If $\varphi$ is QBF-valid, then $\Phi(\varphi)$ is **S4**-satisfiable;

(b) If $\Phi(\varphi)$ is **K**-satisfiable, then $\varphi$ is QBF-valid.

The following lemma, then, will be sufficient for the proof of Ladner's theorem:

$$\begin{tikzcd}
 & \forall p_1 \exists p_2 (p_1 \leftrightarrow \neg p_2) \arrow[dl, "p_1 \mapsto \top"'] \arrow[dr, "p_1 \mapsto \bot"] & \\
\exists p_2 (\top \leftrightarrow \neg p_2) \quad \text{‘}p_1\text{’} \arrow[d, "p_2 \mapsto \bot"'] & & \exists p_2 (\bot \leftrightarrow \neg p_2) \quad \text{‘}\neg p_1\text{’} \arrow[d, "p_2 \mapsto \top"] \\
\top \leftrightarrow \neg \bot \quad \text{‘}\neg p_2\text{’} & & \bot \leftrightarrow \neg \top \quad \text{‘}p_2\text{’}
\end{tikzcd}$$

Figure 4.5: $\forall p_1 \exists p_2 (p_1 \leftrightarrow \neg p_2)$ is QBF-valid.

**Lemma 4.16.** *The problem of testing validity of QBF-formulae is PSPACE-complete.* $\lhd$

$\Phi(\varphi)$ is defined like the formula presented in the proof of Theorem 4.14. For a QBF-formula $\varphi = Q_1 p_1 \dots Q_n p_n \, \psi$, let $\Phi(\varphi)$ be defined as:

$$\begin{aligned}
\Phi(\varphi) := \quad & (\mathrm{I}) \wedge (\mathrm{II}) & \\
& \wedge \; \Box^{(n)} \bigwedge_{i=0}^{n} (q_i \to \Diamond q_{i+1}) & \text{(IIa)} \\
& \wedge \bigwedge_{1 \le i \le n, Q_i = \forall} \Box^i \beta_i & \text{(IIb)} \\
& \wedge \; (\mathrm{IV}) & \\
& \wedge \; \Box^{(n)} (q_n \to \psi) &
\end{aligned}$$

**Lemma 4.17.** *For each valid QBF-formula $\varphi$, $\Phi(\varphi)$ is **S4**-satisfiable.*

*Proof.* Let $\varphi = Q_1 p_1 \dots Q_n p_n \, \psi$ be a valid QBF-formula. As we saw above, we can construct a tree $\mathscr{T}$ with $n+1$ levels that witnesses its validity. In that tree label each node by ‘$p_i$’ that is obtained by replacing $p_i$ by $\top$. We now take the reflexive-transitive closure of this tree, thus obtaining an **S4**-frame $\mathcal{F}$. On this frame we define a Kripke model $\mathcal{M}$ by

$$\begin{aligned}
V(q_i, s) = 1 &\iff s \text{ is node of } \mathscr{T} \text{ at level } i \\
V(p_i, s) = 1 &\iff s \text{ is node or successor of a node of } \mathscr{T} \\
& \qquad \text{that is labelled by ‘}p_i\text{’at level } i
\end{aligned}$$

In the root of $\mathcal{M}$, then, $\Phi(\varphi)$ is true. $\lhd$

As an example of the procedure presented in this proof let us reconsider the QBF-formula $\forall p_1 \exists p_2 (p_1 \leftrightarrow \neg p_2)$ (cf. Figure 4.5). The **S4**-frame that corresponds to that tree is depicted in Figure 4.6). It can easily be checked that in the root of the tree $\Phi(\varphi)$ is true.

**Lemma 4.18.** *If $\Phi(\varphi)$ is **K**-satisfiable, then $\varphi$ is QBF-valid.*

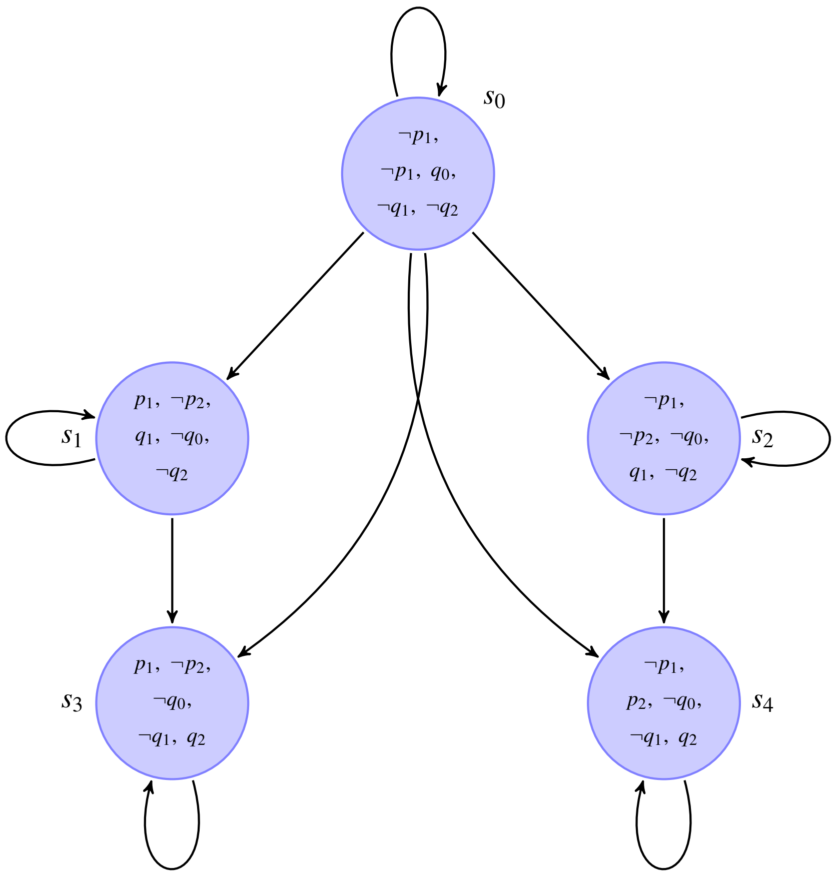

$s_0$
$\neg p_1$, $\neg p_1$, $q_0$, $\neg q_1$, $\neg q_2$
$s_1$
$p_1$, $\neg p_2$, $q_1$, $\neg q_0$, $\neg q_2$
$s_2$
$\neg p_1$, $\neg p_2$, $\neg q_0$, $q_1$, $\neg q_2$
$s_3$
$p_1$, $\neg p_2$, $\neg q_0$, $\neg q_1$, $q_2$
$s_4$
$\neg p_1$, $p_2$, $\neg q_0$, $\neg q_1$, $q_2$

Figure 4.6: **S4**-model associated with the QBF-formula $\forall p_1 \exists p_2 (p_1 \leftrightarrow \neg p_2)$.

*Proof.* Let $\mathcal{M}$ be a Kripke model and $s_0$ be a state of $\mathcal{M}$ with $\mathcal{M} \models_{s_0} \Phi(\varphi)$. Obviously, the submodel generated by $s_0$ contains a tree. If $n = \operatorname{depth}\Phi(\varphi)$, then cut this tree such that nodes at level $i < n$ have one or two sucessors, while nodes at level $n$ do not have any successors. The tree thus obtained will witness that $\varphi$ is QBF-valid. $\lhd$

Thus, the problem of testing **K**-satisfiability is PSPACE-hard. In the remainder of this section we will sketch the proof that SAT($\mathbf{K}$) indeed is in PSPACE.

**Theorem 4.19.** *SAT($\boldsymbol{K}$) is PSPACE-complete.*

To sketch the proof of this claim, consider the NPSPACE-algorithm presented in Algorithm 4.1. One needs to show that the algorithm is correct and complete and that it can be implemented in such a way that only space polynomial in the size of the input formula is required. By the Theorem of Savitch, NPSPACE = PSPACE, thus showing the claim.

In what follows, the *closure* of a formula $\chi$, $\mathrm{Cls}(\chi)$, is the smallest set of formulae that contains all subformulae of $\chi$ (note that $\varphi$ is a *subformula* of $\chi$ if $\varphi$ is $\chi$ or has been used to generate $\chi$ according to the usual syntactic rules) and is closed under single negation (i.e., if $\varphi$ is an unnegated formula in $\mathrm{Cls}(\chi)$, then $\mathrm{Cls}(\chi)$ also contains $\neg\varphi$). The closure of a formula set $\Sigma$ is the smallest superset of $\Sigma$ that is closed for subformulae and single negation. Finally, $\Sigma$ is called *closed* if it coincides with its closure.

**Definition 4.6.** Let $\Sigma$ be a closed set of $\mathcal{L}_\tau$-formulae. A *Hintikka set* over $\Sigma$ is a maximal subset $H$ of $\Sigma$ satisfying the following conditions:

(a) $\bot \notin H$.

(b) For each $\neg\varphi \in \Sigma$, $\neg\varphi \in H$ if and only if $\varphi \notin H$;

(c) For each $(\varphi \wedge \psi) \in \Sigma$, $(\varphi \wedge \psi) \in H$ if and only if $\varphi \in H$ and $\psi \in H$.

Note that Hintikka sets need not be satisfiable. Moreover, a formula $\varphi$ is satisfiable if and only if there exists a satisfiable Hintikka set $H$ over $\mathrm{Cls}(\varphi)$ that contains $\varphi$.

**Algorithm 4.1** SAT/WITNESS-Algorithm

```
procedure SAT($\varphi$)
    inputs: $\varphi$, a formula of K
    returns: true or false
    guess a set $H \subseteq \mathrm{Cls}(\varphi)$ with $\varphi \in H$
    return WITNESS($H, \varphi$)

procedure WITNESS($H, \Sigma$)
    inputs: $H, \Sigma$, finite sets of formulae
    returns: true (if $H$ is a satisfiable Hintikka set over $\mathrm{Cls}(\Sigma)$) or false (otherwise)
    if $H$ is not a Hintikka set over $\mathrm{Cls}(\Sigma)$ then
        return false
    for $\Diamond\psi \in H$ do
        set $\Sigma_\psi := \{\chi : \Box\chi \in H\} \cup \{\psi\}$
        guess a subset $H_\psi$ with $\Sigma_\psi \subseteq H_\psi \subseteq \mathrm{Cls}(\Sigma_\psi)$
        if WITNESS($H_\psi, \Sigma_\psi$) = false then
            return false
    return true
```

Without proving it here, we mention (for the proof, see Lemma 6.46 in Blackburn et al. (2002)):

**Lemma 4.20.** *A Hintikka set $H$ is satisfiable (in some Kripke model) if and only if it is satisfiable in a finite tree-like model of depth at most* $\max(\{\operatorname{depth}\varphi : \varphi \in H\})$.

The interesting aspect is that the "shallow" tree model mentioned in the lemma can be constructed from a system $\mathscr{H}$ of Hintikka sets that has the following properties: (a) $H \in \mathscr{H}$; (b) for all $H' \in \mathscr{H}$ and $\Diamond\psi \in H'$, there exists an $H'' \in \mathscr{H}$ that such that $\{\chi : \Box\chi \in H'\} \cup \{\psi\} \subseteq H''$; and (c) for each $H' \in \mathscr{H}$ there exists a "chain" $H = H_0, \ldots, H_n = H' \in \mathscr{H}$ such that for each $0 \leq i < n$, $\{\chi : \Box\chi \in H_i\} \cup \{\psi_i\} \subseteq H_{i+1}$ for some $\Diamond\psi_i \in H_i$. This fact can be used to show the correctness of Algorithm 4.1. Note that all sets of formulae occurring while executing the algorithm are subsets of $\mathrm{Cls}(\varphi)$ and hence have an encoding polynomial in the size of $\varphi$ (for example, if $\varphi$ is represented by its syntax tree, a set of subformulae can be represented by a list of pointers to nodes in that tree). Moreover, each assignment to $\Sigma_\psi$ (line 12) reduces the maximal modal depth of the formulae sets that are used as input for the next recursive call of the WITNESS procedure. Thus, the recursion depth is bounded by the maximal depth of a formula in $\mathrm{Cls}(\varphi)$ and hence by the size of $\varphi$.

Table 4.1: Complexity of the satisfiability problem for selected modal logics

| | **K**, **KD**, **KT**, **KB**, **KDB** | **K4**, **KD4**, **S4** | **K5**, **KD5**, **KB5**, **K45**, **KD45**, **S5** |
|---|---|---|---|
| general | PSPACE-cmpl | PSPACE-cmpl | NP-cmpl |
| depth $\leq k$ | NP-cmpl | PSPACE-cmpl | NP-cmpl |
| depth $= 1$ | NP-cmpl | NP-cmpl | NP-cmpl |

**Remark 4.2.** Even if the complexity of the general satisfiability problem is PSPACE-hard for many modal logics, it is often the case that we get tighter complexity bounds when we impose restrictions on the formulae that are to be checked for satisfiability (Ladner, 1977; Halpern, 1995; Nguyen, 2004). Remark 4.3 gives some overview.

**Remark 4.3.** For multi-modal languages the complexity of reasoning problems may increase considerably.

- For $n > 1$ the satisfiability problem for $\mathbf{S5}_n$ and $\mathbf{KD45}_n$ is PSPACE-complete (remember that for $n = 1$ the satisfiability problem for these logics is NP-complete).
- If the system **K** is extended by a further diamond operator which is interpreted by the universal relation, then for the resulting system the satisfiability problem shifts from PSPACE to EXPTIME.
- Later we will discuss a system that allows for modeling common knowledge of $n$ agents. However, by adding common knowledge to one of the systems $\mathbf{K4}_n$, $\mathbf{S4}_n$, $\mathbf{KD45}_n$, or $\mathbf{S5}_n$, the complexity of the satisfiability problem increases from PSPACE to EXPTIME, though the resulting system still has the finite model property.

# 5 Decision Procedures

In this section we will present decision procedures for testing satisfiability or validity of formulae in some of the more prominent modal logics. We will start with tableaux algorithms, as these have been widely used in modal logics (and related logics such as description logics) to implement reasoners.
In general, we have different *reasoning tasks* in such logics:

**Satisfiability problem.** Given a normal modal logic $\Lambda$ and an $\mathcal{L}_\tau$-formula $\varphi$, is $\varphi$ satisfiable in some (Kripke) model for $\Lambda$?

**Validity problem.** Given a normal modal logic $\Lambda$ and an $\mathcal{L}_\tau$-formula $\varphi$, is $\varphi$ valid in each (Kripke) model for $\Lambda$?

**Model checking problem.** Given a model $\mathcal{M}$, a state $s_0$ in the model and a formula $\varphi$, is $\varphi$ satisfied in $s_0$.

While for most modal logics the model checking problem can be solved in polynomial time (in the size of the input model and the input formula), the first two problems are for many modal logics computationally hard problems (we will return to this point in the next section).
Remember that the satisfiability and the validity problem are dual problems: a formula $\varphi$ is satisfiable in a class of frames if and only if its negation is not valid in that class.

## 5.1 A tableaux procedure for K(*m*)

Tableaux procedures aim at constructing a model that shows that an input formula is satisfiable. The key idea of these procedures is to construct a tableau, a graph representing such a model, by applying iteratively so-called expansion rules (also called *completion rules*). For many logics these graphs are trees; this explains why tableaux are also referred to as *truth trees*): the root of the tree is given by the input formula, and in each step the nodes of the tree are expanded, while the formulae in parent nodes are transformed to simpler formulae in the child nodes.
In what follows we confine to the basic modal logic **K**(*m*). The basic tableaux rules then are the following:

**$\alpha$-rules:**

$$\frac{s \models \neg\varphi}{s \not\models \varphi} \qquad \frac{s \not\models \neg\varphi}{s \models \varphi} \qquad \frac{s \models \varphi \wedge \psi}{\begin{array}{c} s \models \varphi \\ s \models \psi \end{array}} \qquad \frac{s \not\models \varphi \vee \psi}{\begin{array}{c} s \not\models \varphi \\ s \not\models \psi \end{array}} \qquad \frac{s \not\models \varphi \to \psi}{\begin{array}{c} s \models \varphi \\ s \not\models \psi \end{array}}$$

**$\beta$-rules:**

$$\frac{s \models \varphi \vee \psi}{s \models \varphi \mid s \models \psi} \qquad \frac{s \models \varphi \to \psi}{s \not\models \varphi \mid s \models \psi} \qquad \frac{s \not\models \varphi \wedge \psi}{s \not\models \varphi \mid s \not\models \psi}$$

**$\nu$-rules:**

$$\frac{\begin{array}{c} s \models \Box\varphi \\ s R_\Diamond s' \end{array}}{s' \models \varphi} \qquad \frac{\begin{array}{c} s \not\models \Diamond\varphi \\ s R_\Diamond s' \end{array}}{s' \not\models \varphi}$$

**$\pi$-rules:**

$$\frac{s \not\models \Box\varphi}{\begin{array}{c} s\,R_{\Diamond}\,s' \\ s' \not\models \varphi \end{array}} \quad \text{where } s' \text{ is a fresh state name} \qquad \frac{s \models \Diamond\varphi}{\begin{array}{c} s\,R_{\Diamond}\,s' \\ s' \models \varphi \end{array}} \quad \text{where } s' \text{ is a fresh state name}$$

As can be seen from the rules, we are considering “state-labelled” formulae of the form $s \models \varphi$ and $s \not\models \varphi$, where $s$ is a state name and $\varphi$ is a modal-logical formula. Starting from a state-labelled formula (the input formula that is to be tested for satisfiability labelled with an arbitrary state name), one iteratively applies these rules on the branches of the tree so-far constructed: a rule may be applied on a branch, when every antecedent of the rule occurs in the branch. A branch is *closed* if it contains a *clash*, i.e., state-labelled formulae $s \models \varphi$ and $s \not\models \varphi$. Non-closed branches are called *open*. A branch is *finished* if no rule can be applied on it. An open branch that contains state-labelled formulae $s \models p$ or $s \not\models p$ for each propositional variable $p$ occurring in the root formula defines a Kripke model that makes the statement of the root formula true. So if the root is of the form $s \models \varphi$, the open branch defines a Kripke model in which $\varphi$ is true at state $s$: this shows that $\varphi$ is satisfiable. If the root is of the form $s \not\models \varphi$, the open branch gives a counterexample showing that $\varphi$ is not valid. Hence in order to prove that a formula $\varphi$ is valid, one tries to find a counterexample by developing the tableau of $s \not\models \varphi$. When each branch is closed, the formula is indeed valid; otherwise, it is not.

**Remark 5.1.** $\alpha$- and $\beta$-rules handle the propositional-logical connectives. $\alpha$-rules are deterministic rules: applying one of them on a branch in a tableau, they just expand the current branch. Contrary to this, $\beta$-rules are non-deterministic, i.e., they generate alternative child nodes and thus alternative completions of the current branch. We will see that both $\alpha$- and $\beta$-rules need to be applied only once. $\nu$- and $\pi$-rules handle the modal-logical connectives. $\nu$-rules are also deterministic rules, but it may be necessary to apply them more than one time on a state-labelled formula $s \models \Box\varphi$ (or $s \not\models \Diamond\varphi$) occurring in the current branch. $\pi$-rules are also deterministic (given a fixed sequence of “unused” state names), but need to be applied only once.

Examples of tableaux proofs are presented in Figures 5.1 – 5.3. The first example is a simple, propositional logic example: the tableau proves that the formula $p \to (q \to (p \wedge q))$ is valid in $\mathbf{K}(m)$. The second example shows that axiom $\mathbf{K}$ is valid and the third example shows that the formula $\Diamond p \wedge \Diamond q \to \Diamond(p \wedge q)$ is not valid.

The idea of the (semantic) tableaux method presented above is to construct Kripke models. In order to make this explicit, we introduce the notion of completion trees (which represent “partial” Kripke models) and reformulate the tableaux rules in a way that shows how the rules modify these trees. For the sake of simplicity we make the following assumptions: (a) we assume a setting in which $\neg, \wedge, \vee, \Diamond_i, \Box_i$ $(1 \leq i \leq m)$ are chosen as syntactic primitives; and (b) we assume that the formulae that are input to the tableaux procedure are in *negation normal form*, i.e., the negation symbol occurs only in front of propositional variables.
Remember that the *closure* of a formula $\chi$, $\mathrm{Cls}(\chi)$, is the smallest set of formulae that contains all subformulae of $\chi$ and is closed under “single” negation (i.e., if $\varphi$ is an unnegated formula in $\mathrm{Cls}(\chi)$, then $\mathrm{Cls}(\chi)$ also contains $\neg\varphi$).

**Definition 5.1.** A *completion tree* for a formula $\chi$ is a (vertex- and edge-labeled) graph $G_\chi = \langle V, A, L \rangle$, where $V$ is a set of vertices, $A$ is a set of arcs on $V$ (i.e., $A \subseteq V^2$), and $L$ is a labeling

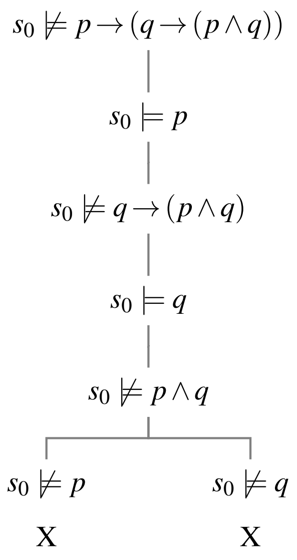

$s_0 \not\models p \to (q \to (p \wedge q))$
$s_0 \models p$
$s_0 \not\models q \to (p \wedge q)$
$s_0 \models q$
$s_0 \not\models p \wedge q$
$s_0 \not\models p$
$s_0 \not\models q$
X
X

Figure 5.1: An example of a propositional logic tableau

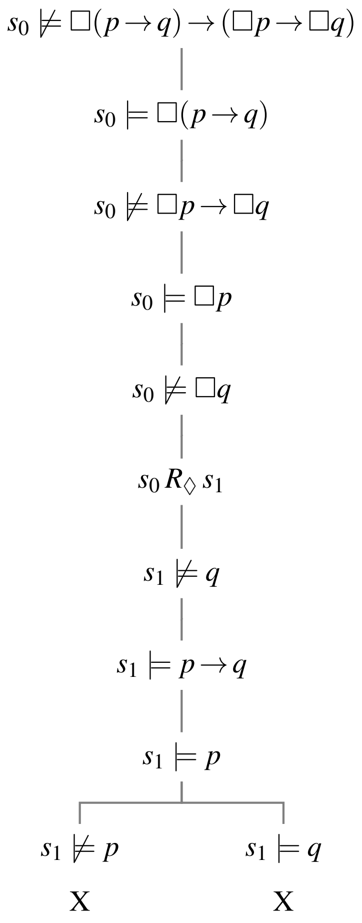


Figure 5.2: Tableau proof of axiom **K**

$$\begin{tikzcd}
 & s_0 \not\models \Diamond p \land \Diamond q \to \Diamond(p \land q) \arrow[d, dash] & & \\
 & s_0 \models \Diamond p \land \Diamond q \arrow[d, dash] & & \\
 & s_0 \not\models \Diamond(p \land q) \arrow[d, dash] & & \\
 & s_0 \models \Diamond p \arrow[d, dash] & & \\
 & s_0 \models \Diamond q \arrow[d, dash] & & \\
 & s_0 \, R_\Diamond \, s_1 \arrow[d, dash] & & \\
 & s_1 \models p \arrow[d, dash] & & \\
 & s_0 \, R_\Diamond \, s_2 \arrow[d, dash] & & \\
 & s_2 \models q \arrow[d, dash] & & \\
 & s_1 \not\models p \land q \arrow[d, dash] & & \\
 & s_2 \not\models p \land q \arrow[dl, dash] \arrow[dr, dash] & & \\
s_1 \not\models p & & s_1 \not\models q \arrow[ddl, dash] \arrow[ddr, dash] & \\
\mathrm{X} & & & \\
 & s_2 \not\models p & & s_2 \not\models q \\
 & \mathrm{O} & & \mathrm{X}
\end{tikzcd}$$

Figure 5.3: Constructing a counterexample by a tableau proof

function that assigns to each $v \in V$ a set $L(v) \subseteq \mathrm{Cls}(\chi)$ and to each arc $a = (v, v') \in A$ a natural number $L(a) \in \{1, \ldots, m\}$.

Given a completion tree $\langle V, A, L \rangle$, we can reformulate the tableaux rules for our syntactic setting as described in the following definition.

**Definition 5.2.** The *tableaux rules for **K(m)*** are the following:

**$\wedge$-rule.** If $\varphi \wedge \psi \in L(v)$ and $\{\varphi, \psi\} \not\subseteq L(v)$, then set $L(v) \leftarrow L(v) \cup \{\varphi, \psi\}$.

**$\vee$-rule.** If $\varphi \vee \psi \in L(v)$ and $\{\varphi, \psi\} \cap L(v) = \emptyset$, then set $L(v) \leftarrow L(v) \cup \{\varphi\}$ or set $L(v) \leftarrow L(v) \cup \{\psi\}$.

**$\Diamond$-rule.** If $\Diamond_i \varphi \in L(v)$ and there is no $v' \in V$ with $(v, v') \in A$, $L(v, v') = i$, and $\varphi \in L(v')$, then extend $V$ by a fresh node $v^*$ and set $A \leftarrow A \cup \{(v, v^*)\}$, $L(v^*) \leftarrow \{\varphi\}$, and $L(v, v^*) \leftarrow i$.

**$\Box$-rule.** If $\Box_i \varphi \in L(v)$ and there is some $v' \in V$ with $(v, v') \in A$, $L(v, v') = i$, and $\varphi \notin L(v')$, then set $L(v') \leftarrow L(v') \cup \{\varphi\}$.

**Definition 5.3.** A completion tree $G = \langle V, A, L \rangle$ is said to be *clash-free* if there is no $v \in V$ such that $\{p, \neg p\} \subseteq L(v)$ for some propositional variable $p$. A completion tree is called *complete* if none of the completion rules presented in Definition 5.2 can be applied. A complete and clash-free completion tree is also often referred to as *tableau*.

We say that a completion tree $\langle V, A, L \rangle$ is satisfiable if there exists a Kripke model $\mathcal{M}$ with frame $\langle S, \{R_{\Diamond_i}\}_{1 \leq i \leq m} \rangle$ such that $V \subseteq S$, $\{a \in A : L(a) = i\} \subseteq R_{\Diamond_i}$ $(1 \leq i \leq m)$, and $\mathcal{M} \models_v L(v)$ for each $v \in V$. Note that for the definition of the model it is sufficient to interpret the propositional variables that occur in the closure of the input formula.

**Lemma 5.1.**

(a) *A completion tree G is satisfiable if and only if one of the completion trees resulting from applying any **K(m)** tableaux rule on G is satisfiable.*

(b) *A completion tree that contains a clash is not satisfiable.*

(c) *If a completion tree $\langle V', A', L' \rangle$ is complete and clash-free, then*

$$S := V', \quad R_{\Diamond_i} := \{a \in A' : L'(a) = i\}, \quad V(p) := \{s \in S : p \in L'(s)\}$$

*defines a Kripke model $\mathcal{M}$ that trivially satisfies $\mathcal{M} \models_s L'(s)$ for each state s.* $\lhd$

In what follows we describe the tableaux procedure in more detail. Algorithm 5.1 provides a rather sketchy version that closely reflects the basic idea: apply the tableaux rules on the initial completion tree that is given by the input formula. If a clash is detected in the completion tree, we return *unsatisfiable*; if the completion tree is clash-free and complete, we have found a Kripke model that satisfies the completion tree and hence also satisfies the completion tree defined by the input formula of the procedure. If the completion tree is clash-free, but not complete, we can apply further rules and then check the satisfiability of the modified completion tree. Note that this algorithm is non-deterministic. For proving its correctness we may assume that (when possible) the right choice is made whenever the $\vee$-rule is applied.

**Remark 5.2.** Notice that a non-deterministic Turing machine accepts a word if *at least one* computation on the input word results in an accepting state. It should be noted that the

**Algorithm 5.1** CHECKSAT (non-deterministic version)

```
    procedure CHECKSAT($\varphi$)
        inputs: $\varphi$, a formula of $\mathbf{K}(m)$ in NNF
        returns: satisfiable or unsatisfiable
        $V \leftarrow \{v_0\}$ , $A \leftarrow \emptyset$ , $L(v_0) \leftarrow \{\varphi\}$
5:      return COMPLETETREE($\langle V,A,L \rangle$)
    procedure COMPLETETREE($G$)
        inputs: $G$, a completion tree
        returns: satisfiable or unsatisfiable
        if $G$ contains a clash then
10:         return unsatisfiable
        if $G$ is complete then
            return satisfiable
        $G \leftarrow$ apply some completion rule on some formula in $G$
        return COMPLETETREE($G$)
```

non-deterministic version of the CHECKSAT algorithm involves different versions of non-determinism. First, with regard to correctness of the algorithm, it is irrelevant which rule is applied in each step (*"don't care" non-determinism*). Contrary to that, it is essential which choice is made when the $\vee$-rule is applied (*"don't know" non-determinism*).

Algorithm 5.2 provides a deterministic version. It uses two auxiliary functions: SELECTFORMULA selects a formula from one of the label sets $L(v)$ in the completion tree that is not a literal (a literal is a formula of the form $p$ or $\neg p$ where $p$ is a propositional variable); to avoid early branching, it is in most cases reasonable to defer the selection of an $\vee$-formula or a $\Diamond$-formula as long as possible. APPLYRULE takes as input a completion tree, a vertex in that tree, and a formula (on which a rule can be applied) and returns the list of completion trees that result from applying an appropriate rule (as listed in Definition 5.2); in the case of the $\wedge$-, $\Box$-, or $\Diamond$-rule the resulting list consists of a single completion tree and in the case of the $\vee$-rule it consists of two completion trees (one for each choice).
The search for a tableau for an input formula starts with an initial completion tree with exactly one vertex $v_0$ that has a singleton label set containing the input formula. On this initial completion tree we execute the procedure COMPLETETREE, which searches for a clash-free and complete completion of the input tree in depth-first manner. On an input completion tree COMPLETETREE first checks, whether the completion tree is clash-free and whether still rules can be applied on a formula occurring in one of the label sets. When we have found a clash, we need to backtrack. If we have found a clash-free and complete completion tree, we are finished and thus the algorithm returns "satisfiable". Otherwise, we select one of the formulae on which rules can be applied, generate a list of possible completion trees, and execute in a recursive manner on each of these trees COMPLETETREE. Note that in each recursive call, COMPLETETREE operates on a completion tree in which some label set will contain new formulae of smaller degree. Thus, eventually we will end up with a completion tree in which no rule can be applied any more (on any vertex introduced earlier into the completion tree). This shows that COMPLETETREE will not generate an infinite sequence of completion trees in which each results from its predecessor by applying a rule. That is, termination of the tableaux procedure

is guaranteed. Moreover, Lemma 5.1 shows that CHECKSAT is correct.

**Algorithm 5.2** CHECKSAT (deterministic version)

```
    procedure CHECKSAT($\varphi$)
        inputs: $\varphi$, a formula of $\mathbf{K}(m)$ in NNF
        returns: satisfiable or unsatisfiable
        $V \leftarrow \{v_0\}$ , $A \leftarrow \emptyset$ , $L(v_0) \leftarrow \{\varphi\}$
5:      return COMPLETETREE($\langle V, A, L \rangle$)
    procedure COMPLETETREE($G$)
        inputs: $G$, a completion tree
        returns: satisfiable, unsatisfiable, or fail
        if $G$ contains a clash then
10:         return fail
        if $G$ is complete then
            return satisfiable
        $v, \psi \leftarrow$ SELECTFORMULA($G$)
        $C \leftarrow$ APPLYRULE($G, v, \psi$)
15:     for $G'$ in $C$ do
            $result \leftarrow$ COMPLETETREE($G'$)
            if $result = satisfiable$ then
                return satisfiable
        return unsatisfiable
20: procedure SELECTFORMULA($G$)
        inputs: $G$, a completion tree
        returns: selects a vertex $v$ and a non-literal formula $\psi$ in $L(v)$
    procedure APPLYRULE($G, v, \psi$)
        inputs: $G$, a completion tree; $v$, a vertex in $G$; $\psi$, a formula in $L(v)$ on which
25:         some completion rule $r$ can be applied
        returns: the list of completion trees resulting from applying rule $r$ on $\psi \in L(v)$
```

## 5.2 Tableaux procedures for other modal logics

How can the decision procedure for the modal logic **K**($m$) presented in the last section be extended to other modal logics such as multi-modal variants of **KT** or **K4**? In general, there are two different ways for dealing with such logics. To illustrate these, consider the modal logic **KT**, the modal logic of reflexive frames. One way to ensure that the frame associated to a complete and clash-free completion tree is reflexive is to introduce a self-arc for each vertex generated while executing CHECKSAT (that is, we have to change the definition of the initial completion tree that is passed to COMPLETETREE and modify the $\Diamond$-rule). The other possibility is to apply axiom **T** on each label set, that is, we extend the rule set listed in Definition 5.2 by a new rule

**$\Box^{\mathbf{r}}$-rule.** If $\Box_i\varphi \in L(v)$, $\Diamond_i$ is a reflexive modality, and $\varphi \notin L(v)$, then set $L(v) \leftarrow L(v) \cup \{\varphi\}$.

Moreover, the Kripke model associated to a complete and clash-free completion tree needs to be adapted: Given such a completion tree, the accessibility relation of a reflexive modality $\Diamond_i$ is not just the set of arcs $a$ with $L(a) = i$, but the reflexive closure of this set. The main advantage of the latter method is that we preserve a tree structure of completion "trees".

While extending the tableaux procedure on reflexive modalities is simple, the case of transitive modalities is more involved. The reason for this is that a naive extension will lead to a non-terminating procedure. As an example consider the formula $\Diamond p \wedge \Box(p \to \Diamond p)$ where $\Diamond$ is a transitive modality. By applying the $\Diamond$-rule on $\Diamond p \in L(v)$, we will generate a new vertex $v'$ with $L(v') = \{p\}$. When we apply the $\Box$-rule on $\Box(p \to \Diamond p) \in L(v)$, we need to extend $L(v')$ by $p \to \Diamond p$ and hence eventually by $\Diamond p$. Thus we will add a new vertex $v''$ to the completion tree (a successor of $v'$) with $L(v'') = \{p\}$. By "transitivity" (in either of the ways described above implemented), we will extend $L(v'')$ by the formula $p \to \Diamond p$ and thus we will need to introduce a further vertex, etc.
This problem can be dealt with by a technique called *blocking*. The idea is to block the expansion of a completion tree whenever the process runs into a "loop", that is, the newly generated vertex together with its labels set contains only irrelevant information that is already encoded in its parent vertex or somewhere else in the completion tree. In the case of transitivity, the blocking condition is as follows: A vertex $v$ is said to *block* $v'$ if $L(v') \subseteq L(v)$ (*subset blocking*) and $v$ is called a *witness* for $v'$ (note that is not required that there is an $(v, v') \in A$). Hence we have a modified $\Diamond$-rule and an additional rule that employs axiom **4** to add further formulae to "neighbored" label sets.

**$\Diamond$'-rule.** If (a) $\Diamond_i \varphi \in L(v)$, (b) $v$ is not blocked, and (c) there is no $v' \in V$ with $(v, v') \in A$, $L(v, v') = i$, and $\varphi \in L(v')$, then extend $V$ by a fresh node $v^*$ and set $A \leftarrow A \cup \{(v, v^*)\}$, $L(v^*) \leftarrow \{\varphi\}$, and $L(v, v^*) \leftarrow i$.

**$\Box^+$-rule.** If (a) $\Box_i \varphi \in L(v)$, (b) $\Diamond_i$ is a transitive modality, and (c) there is some $v' \in V$ with $(v, v') \in A$, $L(v, v') = i$, and $\Box_i \varphi \notin L(v')$, then set $L(v') \leftarrow L(v') \cup \{\Box_i \varphi\}$.

Do these modifications guarantee termination? That is not yet clear. If there is no fixed order in which rules are applied, it may happen that a blocked vertex is getting unblocked when further rules are applied on the parent vertex that modify the blocked vertex in such a way that the blocking condition does no longer hold (*dynamic blocking*). For this reason it makes sense to defer the execution of the generating $\Diamond$-rule on a vertex until no non-generating rule can be applied any more on any available vertex in the completion tree. Moreover, if no rule is applied that modifies the label set of a blocked vertex (*static blocking*), a blocked vertex cannot get unblocked any more. This "strategy" of rule application ensures that no infinite sequence of completion trees in which each tree results from its predecessor by some rule application can be generated.

**Lemma 5.2.** *Let $G = \langle V, A, L \rangle$ be a completion tree for a formula $\chi$ and $n_\chi = |\mathrm{Cls}(\chi)|$.*

(a) *Let $G'$ be the result of applying one of the completion rules on $G$ under the static blocking strategy. If $v'$ is blocked by $v$ in $G$, then it is also blocked by $v'$ in $G'$.*

(b) *If $V$ contains more than $2^{n_\chi}$ elements, then there exist $v, v' \in V$ with $L(v) = L(v')$.*

(c) *Let $G'$ be the result of applying a finite number of completion rules on $G$ under the static blocking strategy. Then $G'$ contains at most $2^{n_\chi}$ non-blocked vertices.*

(d) *Let $G'$ be any completion tree resulting from $G$ by finitely many rule applications under the static blocking strategy. Then the number of vertices in the completion tree $G'$ is* *bounded by* $2^{2\cdot n_\chi}$.

*Proof.* (a) Obviously, each rule application can only monotonically add formulae to label sets of unblocked vertices (because of static blocking). Hence if $L(v') \subseteq L(v)$ holds in $G$, this subset relation must be true in $G'$ as well.

(b) Since each $L(v)$ is a subset of $\mathrm{Cls}(\chi)$, there are at most $2^{n_\chi}$ pairwise distinct label sets.

(c) If $G'$ contains more than $2^{n_\chi}$ unblocked nodes, then there exist unblocked vertices $v, v'$ in $G'$ with $L(v) = L(v')$. Since all vertices are introduced in some order, we may assume that $v$ had been added to $V$ before $v'$ was added. But this means that at some step vertex $v$ blocks $v'$ (in contradiction to the assumption that $v'$ is unblocked): by (a), a blocked vertex cannot get unblocked by further rule applications under the static blocking strategy.

(d) In each finite sequence of rule applications we can only generate $2^{n_\chi}$ non-blocked vertices in the completion tree. Under the given strategy, rules are only applied on these non-blocked vertices. Each non-blocked vertex can have at most $2^{n_\chi}$ outgoing arcs in the completion tree (since label sets contain at most $2^{n_\chi}$ formulae) and blocked vertices do not have outgoing arcs. This entails that the number of possibly introduced vertices in the completion tree is bounded by $2^{n_\chi} \times 2^{n_\chi} = 2^{2\cdot n_\chi}$. In addition to the claim: Since each vertex contains at most $2^{n_\chi}$ different formulae in its label set, it can be modified at most $2^{n_\chi}$ many times (note that label sets are extended in a monotone way only). This shows that any sequence of completion trees in which each tree results from its predecessor by some rule application has a length bounded by $\mathcal{O}(2^{3\cdot n_\chi})$ since in each step a label set is modified or a new vertex is introduced. $\lhd$

## 5.3 Optimization techniques

In this section we describe some optimization techniques for tableaux procedures: these techniques aim at either improving search (for example, reduce the width of the search tree and skip irrelevant branching points when backtracking after a clash has been detected).

**Semantic branching.** The $\vee$-rule as given in Definition 5.2 performs *syntactic branching*: when it is applied on a disjunction $\varphi \vee \psi \in L(v)$, one first checks whether, say, $L(v) \cup \{\varphi\}$ leads to a clash in the modified completion tree. If so, one continues with $L(v) \cup \{\psi\}$. But this means that the information is lost that $L(v) \cup \{\varphi\}$ (and hence each superset of this label set) leads to a clash. To avoid this it is suggesting to continue with the label set $L(v) \cup \{\psi, \neg\varphi\}$ (or: $L(v) \cup \{\psi, NNF(\neg\varphi)\}$) when the decision $L(v) \cup \{\varphi\}$ turns out to be unsatisfiable. This technique is called *semantic branching*: it is based on the propositional logic equivalence $(\varphi \vee \psi) \leftrightarrow (\varphi \vee (\neg\varphi \wedge \psi))$. A potential drawback of semantic branching is that adding $\neg\varphi$ to the label set increases the amount of formulae that need to be processed further on: such formulae may be disjunctions or $\Diamond$-formulae and hence may lead to the generation of new, but unnecessary vertices in the completion tree or branching nodes during search.

**Local simplification, Boolean constraint propagation.** Local simplification methods aim at reducing the width of the search tree by simplifying formulae before non-deterministic completion rules are applied. A special simplification method is Boolean constraint propagation: the idea is to use literals occurring in a label set in order to simplify in a deterministic way disjunctions in such sets. For example, if a label set $L(v)$ contains the formulae $p \vee \varphi$ and $\neg p$, $L(v)$ can be immediately extended by the formula $\varphi$. An example combining semantic branching with Boolean constraint propagation is depicted in Figure 5.4. The benefit of such propagation techniques is that search space is never increased. A potential drawback is that in order to use propagation techniques efficiently, more complex data structures for representing formulae may be necessary. Horrocks (2003) reports that such techniques perform well on randomly generated problem instances, in particular on over-constrained instances that are likely to be unsatisfiable.

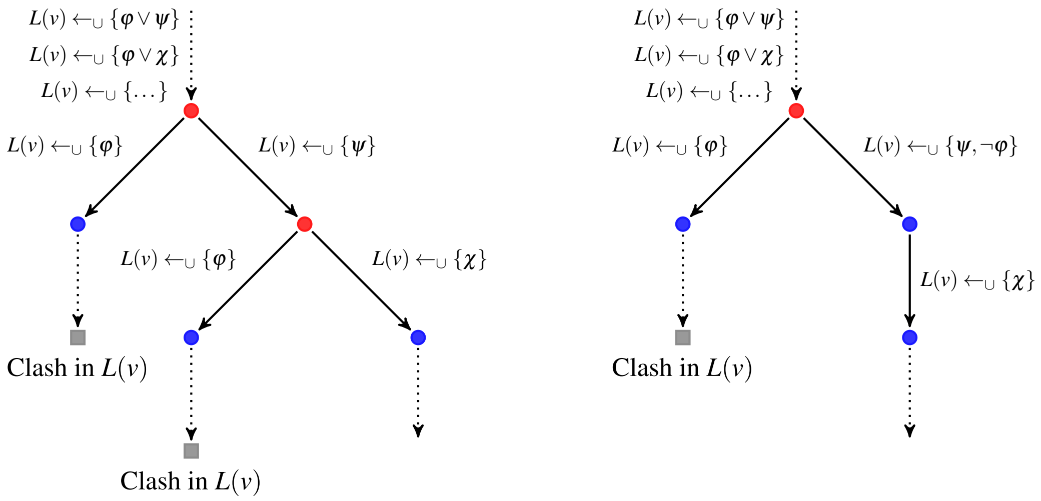

$L(v) \leftarrow_\cup \{\varphi \vee \psi\}$
$L(v) \leftarrow_\cup \{\varphi \vee \chi\}$
$L(v) \leftarrow_\cup \{\dots\}$
$L(v) \leftarrow_\cup \{\varphi\}$
$L(v) \leftarrow_\cup \{\psi\}$
$L(v) \leftarrow_\cup \{\varphi\}$
$L(v) \leftarrow_\cup \{\chi\}$
Clash in $L(v)$
Clash in $L(v)$
$L(v) \leftarrow_\cup \{\varphi \vee \psi\}$
$L(v) \leftarrow_\cup \{\varphi \vee \chi\}$
$L(v) \leftarrow_\cup \{\dots\}$
$L(v) \leftarrow_\cup \{\varphi\}$
$L(v) \leftarrow_\cup \{\psi, \neg\varphi\}$
$L(v) \leftarrow_\cup \{\chi\}$
Clash in $L(v)$

Figure 5.4: Syntactic vs semantic branching (combined with Boolean constraint propagation)

**Enhancements of backtracking search.** As an alternative to the simple backtracking scheme, more sophisticated schemes have been discussed in the literature: a typical problem for backtracking search is *thrashing*, i.e., search nodes are explored although they will not lead to a "solution".

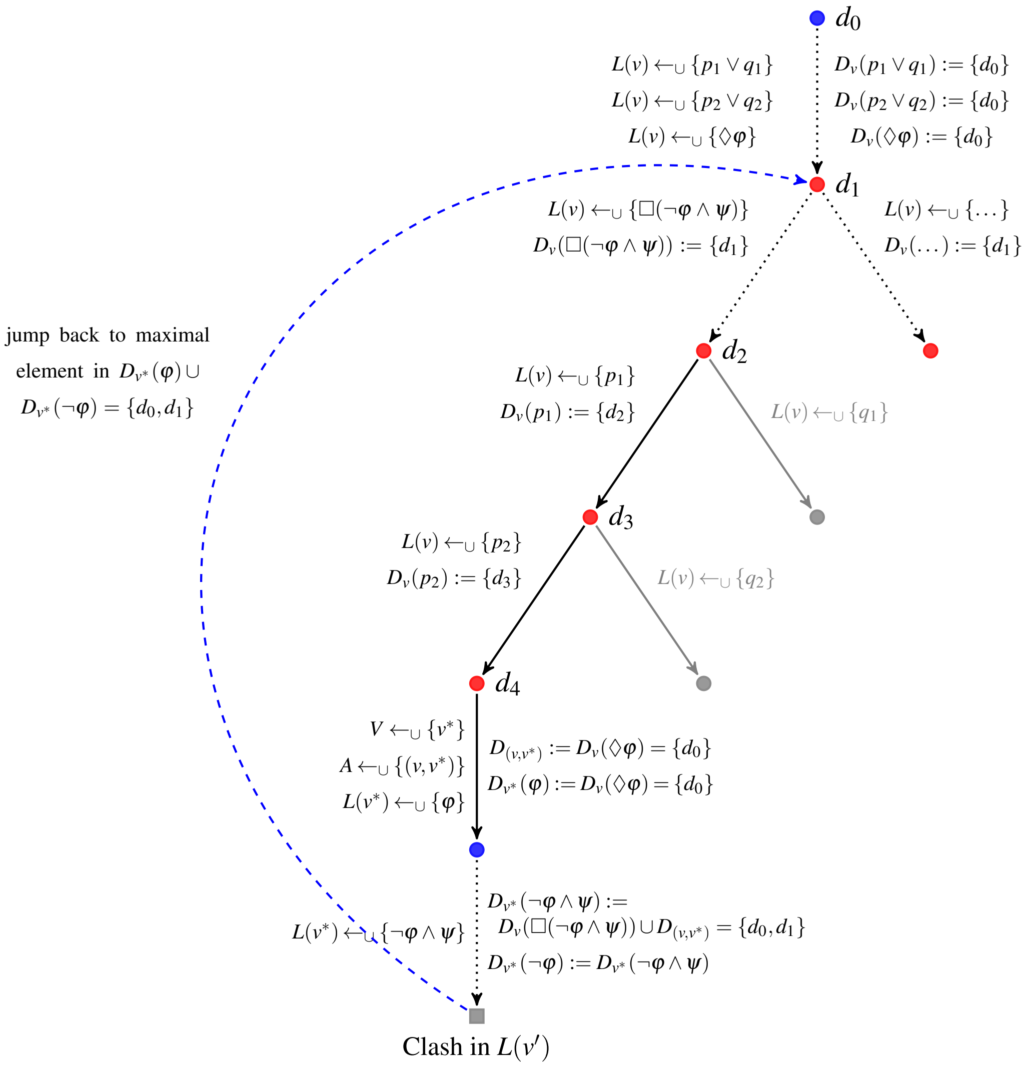
An example of such an alternative backtracking scheme is *dependency directed backjumping*, initially developed in the CSP field. The main idea is to equip each search node with a set of search nodes it depends on. When during search we discover that the current search node cannot lead to a solution, we do not just revise the last decision, but perform a *backjump* to the last search node on which the current search node depends.

In our context, in which the search nodes are given by completion trees, this basic idea can be implemented as follows: each completion tree generated during search that has more than one successor (*decision node*, *branching node*) is augmented with an identifier that is unique to the currently explored branch in the search tree. Furthermore, each formula in each label set as well as each arc records the decision nodes it depends on: a formula $\varphi \in L(v)$ is said to *depend* on a decision node $n$ if $\varphi$ has been added to $L(v)$ at $n$ or $\varphi$ depends on another formula $\psi \in L(v)$ that depends on $n$ or an arc $(v', v) \in A$ that depends on $n$. $\varphi \in L(v)$ depends on $\psi \in L(v)$ if $\varphi$ has been added to $L(v)$ by applying some deterministic rule on $\psi \in L(v)$. $\varphi \in L(v)$ depends on an arc $(v', v) \in A$ if $\varphi$ has been added to $L(v)$ by applying some deterministic rule mentioning this arc. An arc $(v', v)$ depends on a formula $\psi \in L(v')$ if $v$ has been introduced by an application of the $\Diamond$-rule on $\psi \in L(v')$.

During search we record these dependency relations. For example, if $\varphi$ has been added to $L(v)$ by applying the $\Box$-rule on $\psi \in L(v')$, the dependency set of $\varphi \in L(v)$, denoted $D_v(\varphi)$, will be the union of the dependency sets of $\psi \in L(v')$ and of the arc $(v', v)$. If we apply the $\Diamond$-rule on a formula $\Diamond\psi \in L(v)$ (thus generating a new node $v^*$), the dependency set of the arc $(v, v^*)$ and the dependency set of $\psi \in L(v^*)$ both will be set to the dependency set of $\Diamond\psi \in L(v)$.

Thus, if the search results in a completion tree with a clash $\{\varphi, \neg\varphi\} \subseteq L(v)$, we perform a backjump to the maximal member in the union of the dependency sets $D_v(\varphi)$ and $D_v(\neg\varphi)$. Figure 5.5 presents an example: depicted is a part of the search tree when the tableaux procedure is processed on a formula of the form $(p_1 \vee q_1) \wedge (p_2 \vee q_2) \wedge \Diamond\varphi \wedge (\Box(\neg\varphi \wedge \psi) \vee \ldots)$.

In general, dependency directed backjumping results in a significant simplification of the traversed search tree.



$d_0$
$L(v) \leftarrow_\cup \{p_1 \vee q_1\}$
$L(v) \leftarrow_\cup \{p_2 \vee q_2\}$
$L(v) \leftarrow_\cup \{\Diamond\varphi\}$
$D_v(p_1 \vee q_1) := \{d_0\}$
$D_v(p_2 \vee q_2) := \{d_0\}$
$D_v(\Diamond\varphi) := \{d_0\}$
$d_1$
$L(v) \leftarrow_\cup \{\Box(\neg\varphi \wedge \psi)\}$
$D_v(\Box(\neg\varphi \wedge \psi)) := \{d_1\}$
$L(v) \leftarrow_\cup \{\ldots\}$
$D_v(\ldots) := \{d_1\}$
jump back to maximal element in $D_{v^*}(\varphi) \cup D_{v^*}(\neg\varphi) = \{d_0, d_1\}$
$d_2$
$L(v) \leftarrow_\cup \{p_1\}$
$D_v(p_1) := \{d_2\}$
$L(v) \leftarrow_\cup \{q_1\}$
$d_3$
$L(v) \leftarrow_\cup \{p_2\}$
$D_v(p_2) := \{d_3\}$
$L(v) \leftarrow_\cup \{q_2\}$
$d_4$
$V \leftarrow_\cup \{v^*\}$
$A \leftarrow_\cup \{(v, v^*)\}$
$L(v^*) \leftarrow_\cup \{\varphi\}$
$D_{(v,v^*)} := D_v(\Diamond\varphi) = \{d_0\}$
$D_{v^*}(\varphi) := D_v(\Diamond\varphi) = \{d_0\}$
$L(v^*) \leftarrow_\cup \{\neg\varphi \wedge \psi\}$
$D_{v^*}(\neg\varphi \wedge \psi) :=$
$D_v(\Box(\neg\varphi \wedge \psi)) \cup D_{(v,v^*)} = \{d_0, d_1\}$
$D_{v^*}(\neg\varphi) := D_{v^*}(\neg\varphi \wedge \psi)$
Clash in $L(v')$

Figure 5.5: Dependency directed backjumping

**Search heuristics.** Search heuristics aim at guiding search when decisions need to be made. In the context of the tableaux procedure heuristics come into play when one has to decide which disjunct in a disjunctive formula occurring in a label set is to be selected first. One method (developed in the SAT field and applicable in the context of tableaux proofs as well) is the so-called *MOMS* heuristic. The slogan of the MOMS heuristic is "select a formula that has the maximal number of occurrences in disjunctions of minimal size". The MOMS heuristic aims at optimizing Boolean constraint propagation: if such a disjunct leads to a clash, Boolean constraint propagation will reduce the branching of subsequently explored search nodes.

The drawback of the MOMS heuristic is that it does not perform well in combination with dependency directed backjumping. Moreover, it may depend on the problem instances to be solved whether the MOMS heuristic provides useful information for selecting a disjunct.

Another search heuristic that is particularly useful in combination with dependency directed backjumping is the *oldest-first* heuristic. The idea is here to use the dependency sets generated during search to select a disjunct that depends on the oldest decision node in the currently expanded branch of the search tree. More precisely, one selects a disjunction that has the oldest element in its dependency set. Further heuristics can be combined with this heuristic to guide the selection of a disjunct in such a disjunction.

The benefit of using search heuristics is that they often can be combined efficiently with backtracking schemes. Moreover, search heuristics may be selected dependent on the formula that needs to be processed.

## 5.4 Modal satisfiability via propositional satisfiability

An alternative approach to checking satisfiability in **K**($m$) is to iteratively use propositional logic satisfiability checking. The idea is close to the procedure discussed in section 1.2: Consider an arbitrary formula $\varphi$ in a setting in which $\Box$'s are used as a primitive and $\Diamond$'s defined in terms of $\Box$. In what follows we refer to all subformulae of $\varphi$ that cannot be decomposed in propositional logic as *atoms* of $\varphi$. A *literal*, then, is an atom or a negated atom. A *top-level atom* is any atom that occurs in $\varphi$, but not within the scope of a $\Box$. A *truth assignment* for $\varphi$ is a function $\mu$ that assigns to each top-level atom of $\varphi$ a truth value. A truth assignment is said to *propositionally satisfy* $\varphi$, $\mu \models_p \varphi$, if $\mu$ (conceived of as a truth assignment of the top-level atoms as "variables") makes $\varphi$ true. A *partial truth assignment* is a mapping that assigns truth values to some top level atoms. A partial truth assignment $\mu$ is said to *propositionally satisfy* $\varphi$ if all (complete) truth assignments extending $\mu$ propositionally satisfy $\varphi$.

For example, assume $\varphi$ is the formula $\Diamond(p \wedge \Box q) \vee \neg\Box(q \rightarrow \Diamond q)$. Then the top-level atoms of $\varphi$ are $\Box\neg(p \wedge \Box q)$ and $\Box(q \rightarrow \Diamond q)$. The partial truth assignment $\Box\neg(p \wedge \Box q) \mapsto \mathit{false}$ propositionally satisfies $\varphi$.
Given a (partial) truth assignment $\mu$, there is an associated formula (expressing the truth assignment): This formula is a conjunction of formulae in which a top-level atom of $\varphi$, $\Box_i \psi$, occurs unnegated when $\mu$ evaluates it *true* and negated when $\mu$ evaluates it *false*. That is, $\mu$ can be written in the form

$$\bigwedge_{i_1} \Box_1 \alpha_{1,i_1} \wedge \bigwedge_{j_1} \neg\Box_1 \beta_{1,j_1} \wedge \cdots \wedge \bigwedge_{i_m} \Box_m \alpha_{m,i_m} \wedge \bigwedge_{j_m} \neg\Box_m \beta_{m,j_m} \wedge \gamma.$$

Conversely, each such formula can be read as a truth assignment.

A truth assignment $\mu$ is said to be ***K**(m)-satisfiable* if its associated formula is satisfiable (as a **K**($m$) formula). Then the following statements hold:

**Lemma 5.3.** *Let $\varphi$ be an $\mathcal{L}_\tau$-formula.*

(a) *$\varphi$ is **K**(m)-satisfiable if and only if there exists a (partial) truth assignment $\mu$ that is **K**(m)-satisfiable and satisfies $\mu$ propositionally.*

(b) *$\mu$ is **K**(m)-satisfiable if and only if in its associated formula for each $\Box_k$,*

$$\bigwedge_{i_k} \Box_k \alpha_{k,i_k} \wedge \bigwedge_{j_k} \neg\Box_k \beta_{k,j_k}$$

*(denoted by $\mu^k$) is **K**(m)-satisfiable.*

(c) *$\mu^k$ (as given in (b)) is **K**(m)-satisfiable if and only if for each conjunct $\neg\Box_k\beta_{k,j_k}$,*

$$\bigwedge_{i_k} \alpha_{k,i_k} \wedge \neg\beta_{r,j_k}$$

*is **K**(m)-satisfiable.*

*Proof.* Similar to the proof of Lemma 1.7 in section 1. ◁

This lemma shows how satisfiability testing can be "reduced" to satisfiability testing of formulae with smaller depth. Hence we can apply the procedure recursively until only the satisfiability of propositional logic formulae has to be checked. It is clear that this procedure is not a reduction of the satisfiability problem in **K**($m$) to the analogical problem in propositional logic: in order to check a single formula it may be necessary to check exponentially many partial truth assignments (see Algorithm 5.3).

## Bibliographic remarks

TO BE DONE

**Algorithm 5.3** KSAT-Algorithm (Giunchiglia and Sebastiani, 2000)

```
procedure KSAT($\varphi$)
    inputs: $\varphi$, a formula of $\mathbf{K}(m)$
    returns: true or false

    return KSAT-W($\varphi, \top$)

procedure KSAT-W($\varphi, \mu$)
    inputs: $\varphi$, a formula; $\mu$, a truth assignment
    returns: true or false

    if $\varphi = \top$ then
        return KSAT-A($\mu$)
    if $\varphi = \bot$ then
        return False
    if $\varphi$ is in CNF and contains a unit clause $l$ then
        return KSAT-W(ASSIGNLITERAL($l, \varphi$), $\mu \wedge l$)
    $l \leftarrow$ SELECTLITERAL($\varphi$)
    return KSAT-W(ASSIGNLITERAL($l, \varphi$), $\mu \wedge l$) or
            KSAT-W(ASSIGNLITERAL($\neg l, \varphi$), $\mu \wedge \neg l$)

procedure KSAT-A
    inputs: $\varphi$, a formula of the form
            $\bigwedge_i \Box_1 \alpha_{1,i} \wedge \bigwedge_j \neg\Box_1 \beta_{1,j} \wedge \cdots \wedge \bigwedge_i \Box_m \alpha_{m,i} \wedge \bigwedge_j \neg\Box_m \beta_{m,j} \wedge \gamma$
    returns: true or false

    for $1 \leq k \leq m$ do
        res $\leftarrow$ KSAT-RA($\bigwedge_i \Box \alpha_i \wedge \bigwedge_j \neg\Box \beta_j$)
        if res = False then
            return False
    return True

procedure KSAT-RA
    inputs: $\varphi$, a formula of the form $\bigwedge_i \Box_k \alpha_{k,i} \wedge \bigwedge_j \neg\Box_k \beta_{k,j}$
    returns: true or false

    for all $\neg\Box_k \beta_{k,j}$ in $\varphi$ do
        res $\leftarrow$ KSAT($\bigwedge_i \alpha_{k,i} \wedge \neg\beta_{k,j}$)
        if res = False then
            return False
    return True
```

# 6 Excursion: Description Logics

Description logics comprise a variety of formalisms in the domain of knowledge representation. For example, description logics are discussed as *the* representation and reasoning formalism for the Semantic Web. Historically, this logic(s) descends from another knowledge representation formalism, namely *semantic networks* (introduced by Quillian and others in 1967 and later).

Suppose that we want to develop a formalism for representing, and reasoning about, the biological relationships in some family. Such a formalism will have to deal with individuals like ‘Adam’ and ‘Eve’, concepts like ‘male’ and ‘female’, and notions like ‘mother’, ‘father’, and ‘child’. It is clear how this could be done in a first-order language. But, since first-order logic is undecidable, the question arises whether (and how) conceptual information can be represented in a simpler and computationally more well-behaved way.

By semantical networks, one could represent such biological relationships as a network such as that depicted in Figure 6.1. However, such a representation is problematic: The relation ‘is’ is

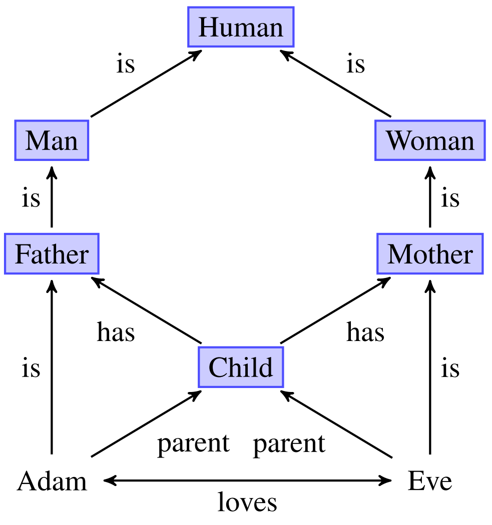


Figure 6.1: A semantical network representing biological relationships (Gabbay et al., 2003)

used in two different meanings, namely as instantiation relation (individual $a$ is an instance of concept $C$) and as subsumption relation (each individual that is an instance of concept $C$ is also an instance of concept $C'$). Moreover, the picture is ambiguous, since it is not clear, whether the lower part of the network part aims at representing that Eve is a parent of *some* child or that Eve is a parent of *every* child.

In what is to follow we will see a straight analogy between description logic and modal logic. In more detail, we will see that:

- Description logics deal with fragments of first-order and second-order logic that have good computational properties.
- Some description logics may be considered notational variants of suitable multi-modal logics. Hence, many results that we obtained in modal logic can be translated or adapted into description logics, and vice versa.

## 6.1 The description logic $\mathcal{ALC}$

As mentioned above there exists a variety of description logics that also differ in the underlying formal language. We will confine to one language, namely the language of the description logic $\mathcal{ALC}$ (*Attributive Language with Complements*). This language is introduced as follows (cp. Example 2.9):

- concept names $A_0, A_1, \dots, A_n$;
- role names $r_0, r_1, \dots, r_m$;
- individual names $a_0, a_1, \dots, a_k$;
- Boolean concept constructors $\sqcap$, $\sqcup$, $-$;
- quantifier: $\exists$, $\forall$;
- symbols: . , : , $\sqsubseteq$, and parentheses.

Then we define:

(a) Each concept name is a *concept term*.

(b) If $C$ and $C'$ are concept terms and $r$ is a role name, then $C \sqcap C'$, $\neg C$, and $\exists r.C$ and $\forall r.C$ are concept terms.

Moreover, if $a$ is an individual name, $r$ is a role name, $A$ is a concept name, and $C$ and $C'$ are concept terms, then the formulae $a : C$, $(a,b) : r$ are called *assertions* and $A \doteq C$ and $C \sqsubseteq C'$ are called *definitions* (formulae of the form $C \sqsubseteq C'$ are also called *general concept inclusions*).
An $\mathcal{L}_{\mathcal{ALC}}$*-model* is a first-order structure

$$\mathcal{M} = \left\langle U, r_0^{\mathcal{M}}, \dots, r_n^{\mathcal{M}}, A_0^{\mathcal{M}}, \dots, A_m^{\mathcal{M}}, a_0^{\mathcal{M}}, \dots, a_k^{\mathcal{M}} \right\rangle$$

where $U$ is a non-empty set (*universe*), each $r_i^{\mathcal{M}}$ is a binary relation on $U$, each $A_j^{\mathcal{M}}$ is a subset of $U$, and each $a_l^{\mathcal{M}}$ is an element of $U$. Inductively, we define the interpretation of concept terms as follows:

$$\begin{aligned}
(C \sqcap C')^{\mathcal{M}} &:= C^{\mathcal{M}} \cap C'^{\mathcal{M}} \\
(C \sqcup C')^{\mathcal{M}} &:= C^{\mathcal{M}} \cup C'^{\mathcal{M}} \\
(\neg C)^{\mathcal{M}} &:= U \setminus C^{\mathcal{M}} \\
(\exists r.C)^{\mathcal{M}} &:= \{x \in U : \text{there is a } y \in C^{\mathcal{M}} \text{ with } x r^{\mathcal{M}} y\} \\
(\forall r.C)^{\mathcal{M}} &:= \{x \in U : \text{for each } y \text{ with } x r^{\mathcal{M}} y,\ y \in C^{\mathcal{M}}\}
\end{aligned}$$

Then we continue by defining truth in $\mathcal{L}_{\mathcal{ALC}}$-models.

$$\begin{aligned}
\mathcal{M} \models a : C &\iff a^{\mathcal{M}} \in C^{\mathcal{M}} \\
\mathcal{M} \models (a,b) : r &\iff (a^{\mathcal{M}}, b^{\mathcal{M}}) \in r^{\mathcal{M}} \\
\mathcal{M} \models C \sqsubseteq C' &\iff C^{\mathcal{M}} \subseteq C'^{\mathcal{M}} \\
\mathcal{M} \models A \doteq C &\iff A^{\mathcal{M}} = C^{\mathcal{M}}
\end{aligned}$$

An $\mathcal{L}_{\mathcal{ALC}}$-formula $\varphi$ is said to be *satisfiable* if there exists an $\mathcal{L}_{\mathcal{ALC}}$-model $\mathcal{M}$ with $\mathcal{M} \models \varphi$. A concept term $C$ of $\mathcal{L}_{\mathcal{ALC}}$ is said to be *consistent* (or alternatively, *satisfiable*) if there exists an $\mathcal{L}_{\mathcal{ALC}}$-model $\mathcal{M}$ with $C^{\mathcal{M}} \neq \emptyset$.

**Definition 6.1.** An $\mathcal{ALC}$-*knowledge base* is any finite set of $\mathcal{L}_{\mathcal{ALC}}$-formulae. The *terminological box* (*TBox*) of a knowledge base $T$ is the set of definitions contained in $T$ (i.e., formulae of the form $A \doteq C$ and $C \sqsubseteq C'$). The *assertion box* (*ABox*) of $T$ is the set of assertions contained in $T$.

**Example 6.1.** The semantic network shown in Figure 6.1, can be represented by the following knowledge base:

TBox:

- $\mathsf{Woman} \sqcup \mathsf{Man} \sqsubseteq \mathsf{Human}$
- $\mathsf{Mother} \sqsubseteq \mathsf{Woman}$
- $\mathsf{Father} \sqsubseteq \mathsf{Man}$
- $\mathsf{Child} \sqsubseteq \exists\ \mathsf{has.Mother} \sqcap \exists\ \mathsf{has.Father}$

ABox:

- $Eve : \mathsf{Mother}$
- $Adam : \mathsf{Father}$
- $(Eve, Adam) : \mathsf{loves}$
- $(Adam, Eve) : \mathsf{loves}$
- $Eve : \exists\ \mathsf{parent.Child}$
- $Adam : \exists\ \mathsf{parent.Child}$

Given a knowledge base $\Sigma$, the following reasoning tasks are of interest:

1. *Concept satisfiability:* Given a concept term $C$, does there exist any model $\mathcal{M}$ of $\Sigma$ such that $C^{\mathcal{M}} \neq \emptyset$, i.e., $\Sigma \not\models C = 0$?
2. *Subsumption:* Given concept terms $C$ and $C'$, is for each model $\mathcal{M}$ with $\mathcal{M} \models \Sigma$, $C^{\mathcal{M}}$ a subset of $C'^{\mathcal{M}}$, i.e., $\Sigma \models C \sqsubseteq C'$?
3. *Instance checking:* Given an individual name $a$ and a concept term $C$, is it true that for each model $\mathcal{M}$ for $\Sigma$, $a^{\mathcal{M}} \in C^{\mathcal{M}}$, i.e., $\Sigma \models a : C$?
4. *Consistency/satisfiability:* Is there some model $\mathcal{M}$ of $\Sigma$ ?

It is not hard to see that each of the reasoning tasks (1) – (3) is polynomially reducible to the general satisfiability problem. Moreover, it holds: Subsumption testing and concept satisfiability testing are polynomially reducible to each other. Note that

$$\Sigma \models C = 0 \iff \Sigma \models C \sqsubseteq 0$$

and

$$\Sigma \models C \sqsubseteq C' \iff \Sigma \models C \sqcap \neg C' = 0.$$

Here 0 denotes *the empty / inconsistent concept*. In the context of complexity analysis, the structure of the knowledge base becomes crucial: often we restrict the general reasoning problem to cases where the TBox is empty or exhibits a particular form (see below).

## 6.2 $\mathcal{ALC}$ and modal logic

As can be seen from the syntax and the semantics of $\mathcal{L}_{\mathcal{ALC}}$, there is a strong similarity between description logic and modal logic. To spell this out in more detail, we will define a translation from the multi-modal language $\mathcal{L}_m(P)$ with $m$ diamond operators, $\Diamond_1, \dots, \Diamond_m$, into the language $\mathcal{L}_{\mathcal{ALC}}$. For each propositional variable $p \in P$, let $A_p$ be a unique concept name of $\mathcal{L}_{\mathcal{ALC}}$, and for each diamond $\Diamond_i$ of $\mathcal{L}_m(P)$, let $r_i$ be a unique role name of $\mathcal{L}_{\mathcal{ALC}}$.

Then define:

$$\begin{aligned}
\Phi(p) &:= A_p \\
\Phi(\neg\varphi) &:= \neg\Phi(\varphi) \\
\Phi(\varphi \vee \psi) &:= \Phi(\varphi) \sqcup \Phi(\psi) \\
\Phi(\Diamond_i \varphi) &:= \exists r_i.\Phi(\varphi)
\end{aligned}$$

Note that here we are translating modal formulae into *concept terms*. Vice versa, we can assign to each concept term of $\mathcal{L}_{\mathcal{ALC}}$, a formula of $\mathcal{L}_m(P)$ by:

$$\begin{aligned}
\Phi^-(A) &:= p_A \\
\Phi^-(\neg C) &:= \neg\Phi^-(C) \\
\Phi^-(C \sqcup C') &:= \Phi^-(C) \vee \Phi^-(C') \\
&\dots \\
\Phi^-(\exists r_j.C) &:= \Diamond_j \Phi^-(C) \\
\Phi^-(\forall r_j.C) &:= \Box_j \Phi^-(C)
\end{aligned}$$

At the semantical level, each Kripke model for $\mathcal{L}_m(P)$, $\mathcal{M} = \langle S, R_1, \dots, R_m, V \rangle$, with finite set of propositional variables $P = \{p_1, \dots, p_n\}$ induces an $\mathcal{L}_{\mathcal{ALC}}$-model

$$\mathcal{M}' = \langle S, R_1, \dots, R_m, V(p_1), \dots, V(p_n), s_1, s_2, \dots \rangle$$

where $s_1, s_2, \dots$ are chosen arbitrarily in $S$. $\mathcal{M}'$ will be referred to as an $\mathcal{L}_{\mathcal{ALC}}$-model *associated* with $\mathcal{M}$.

**Theorem 6.1.** *Let $\mathcal{M}$ be a Kripke-model for $\mathcal{L}_m(P)$ and $\mathcal{M}'$ be an $\mathcal{L}_{\mathcal{ALC}}$-model associated with $\mathcal{M}$. Then for each $\mathcal{L}_m(P)$-formula $\varphi$,*

$$\mathcal{M} \models_s \varphi \iff s \in (\Phi(\varphi))^{\mathcal{M}'} \qquad \lhd$$

Conversely, each $\mathcal{L}_{\mathcal{ALC}}$-model $\mathcal{M} = \langle U, r_1^{\mathcal{M}}, \dots, r_n^{\mathcal{M}}, A_1^{\mathcal{M}}, \dots, A_m^{\mathcal{M}}, a_1^{\mathcal{M}}, \dots \rangle$ induces a Kripke model for $\mathcal{L}_m(P)$, $\mathcal{M}^- = \langle U, r_1^{\mathcal{M}}, \dots, r_n^{\mathcal{M}}, V \rangle$, through $V(p_i) = A_i^{\mathcal{M}}$. Thus, by structural induction one can show the following theorem.

**Theorem 6.2.** *Let $\mathcal{M}$ be an $\mathcal{L}_{\mathcal{ALC}}$-model. Then for each $\mathcal{L}_{\mathcal{ALC}}$-concept $C$,*

$$\mathcal{M}^- \models_x \Phi^-(C) \iff x \in C^{\mathcal{M}} \qquad \lhd$$

These two theorems show that the problem of testing satisfiability for $\mathbf{K}(m)$ is equivalent to testing concept satisfiability *with respect to the empty TBoxes*. As an immediate result we obtain the following two conclusions:

**Corollary 6.3.** *With respect to knowledge bases with empty TBox, both deciding concept satisfiability and deciding the subsumption problem are PSPACE-complete.*

*Proof.* This follows from the facts that deciding satisfiability in $\mathbf{K}(m)$ is PSPACE-complete and that PSPACE = coPSPACE. $\lhd$

**Corollary 6.4** (Finite Model Property)**.** *Each satisfiable $\mathcal{L}_{\mathcal{ALC}}$-formula is satisfiable in some finite model.* $\lhd$

## 6.3 Concept definitions

An $\mathcal{L}_{\mathcal{ALC}}$-formula of the form $A \doteq C'$, where $A$ is a concept name and $C$ is a concept term, is said to be an *explicit definition* of $A$.

**Definition 6.2.** Let $\Sigma$ be an $\mathcal{L}_{\mathcal{ALC}}$-TBox.

(a) $\Sigma$ is said to be *simple* if it consists of explicit definitions only and if for each concept name $A$, $\Sigma$ contains at most one definition of $A$.

(b) A concept name $A$ is said to be *defined* in $\Sigma$ if $\Sigma$ contains a definition of $A$.

Let now $\Sigma$ be a simple TBox. We define a binary relation on the set of defined concept names in $\Sigma$, $<_\Sigma$, as follows: $A <_\Sigma A'$ if and only if $A$ occurs in the definition of $A'$.

**Definition 6.3.** A simple TBox $\Sigma$ is said to be *cyclic* if there exist concept names $A_1, A_2, \dots, A_n$ ($n \geq 1$) such that

$$A_1 <_\Sigma A_2 <_\Sigma \cdots <_\Sigma A_n <_\Sigma A_1.$$

Otherwise, $\Sigma$ is said to be *acyclic*.

**Lemma 6.5.** *Let $\Sigma$ be an acyclic simple TBox. Then the subsumption problem '$\Sigma \models C \sqsubseteq C'$?' is polynomially reducible to the subsumption problem with the empty knowledge base '$\models C \sqsubseteq C'$?'.*

*Proof.* Replace recursively each defined concepts by its definition. $\lhd$

**Theorem 6.6** (e.g, Baader et al., 2005)**.** *With respect to acyclic simple TBoxes, checking concept satisfiability is in PSPACE.* $\lhd$

The condition that the TBox is acyclic is crucial. In the general setting checking concept satisfiability is no longer in PSPACE (under the assumption PSPACE $\neq$ EXPTIME).

**Theorem 6.7** (Schild, 1991; Donini and Massacci, 2000)**.** *With respect to arbitrary TBoxes, checking concept satisfiability for $\mathcal{L}_{\mathcal{ALC}}$-knowledge bases is EXPTIME-complete.* $\lhd$

## 6.4 Unions, compositions, and reflexive and transitive closure of roles

The expressive power of $\mathcal{L}_{\mathcal{ALC}}$ is rather limited. For example, we cannot express that each child has exactly one mother, that the parent role is the union of the father and the mother role, or that children of descendants are descendants, etc. Hence we will extend $\mathcal{L}_{\mathcal{ALC}}$ in that we allow for expressing qualified number restrictions, composition of roles, inverse roles, union of roles, complex role inclusions, and reflexive transitive closure of a role. More precisely, the new language $\mathcal{L}_{s\mathcal{ROIQ}_{\text{trans}}}$ is introduced as follows:

- Each concept name is a concept term.
- Each role name is a role term.
- If $a_1, \dots, a_k$ are individual names, then $\{a_1, \dots, a_k\}$ is a concept term.
- If $r$ and $r'$ are role terms, then so are $r \sqcup r'$, $r \circ r'$, $r^{-1}$, and $r^*$.
- If $C$ and $C'$ are concept terms, $r$ is a role term, $r'$ a role name, and $n \in \mathbb{N}$, then $C \sqcap C'$, $\neg C$, $\exists r.C$, $\forall r.C$, and $\exists^{\leq n} r'.C$ are concept terms.

- If $a$ is an individual name, $r,s$ are role terms, $A$ is a concept name, and $C$ and $C'$ are concept terms, then $a : C$, $(a,b) : r$, $A \doteq C$ and $C \sqsubseteq C'$, $r \circ s \sqsubseteq r$, and $r \circ s \sqsubset r$ are formulae.

Note that number restrictions here are restricted to non-complex roles. Moreover, one can define:

$$\exists^{\geq n} r.C := \exists r.C \sqcap \neg \exists^{\leq n-1} r.C$$
$$\exists^{= n} r.C := (\exists^{\leq n} r.C) \sqcap (\exists^{\geq n} r.C)$$

The semantics of $\mathcal{L}_{s\mathcal{ROIQ}_{\text{trans}}}$ is now introduced very quickly: $\mathcal{L}_{s\mathcal{ROIQ}_{trans}}$*-models* are just $\mathcal{L}_{\mathcal{ALC}}$-models. We then extend the recursive definitions with respect to $\mathcal{L}_{\mathcal{ALC}}$ as follows

$$\begin{aligned}
(r^{-1})^{\mathcal{M}} &= (r^{\mathcal{M}})^{-1} \\
(r \sqcup r')^{\mathcal{M}} &= r^{\mathcal{M}} \cup r'^{\mathcal{M}} \\
(r \circ r')^{\mathcal{M}} &= r^{\mathcal{M}} \circ r'^{\mathcal{M}} \\
(r^*)^{\mathcal{M}} &= (r^{\mathcal{M}})^* \text{ (the refl. and transitive closure of } r^{\mathcal{M}}) \\
(\exists^{\leq n} r'.C)^{\mathcal{M}} &= \left\{ x \in U \; : \; \left| \{ y \in C^{\mathcal{M}} \; : \; x r'^{\mathcal{M}} y \} \right| \leq n \right\}
\end{aligned}$$

**Example 6.2.** We mention some examples to illustrate the expressive power of $\mathcal{L}_{s\mathcal{ROIQ}_{\text{trans}}}$. The statement "Each car has at least four wheels" can be cast as a concept inclusion:

$$\mathsf{Car} \sqsubseteq \exists^{\geq 4} \mathsf{is\text{-}equipped\text{-}with}.\mathsf{Wheels}.$$

The statement "Each friend of a friend of Eve has a bicycle" can be expressed by:

$$Eve : \forall(\mathsf{has\text{-}friend} \circ \mathsf{has\text{-}friend}).\exists \mathsf{has\text{-}vehicle}.\mathsf{Bicycle}.$$

Or one can express that one of Eve's descendants will be famous by

$$Eve : \exists(\mathsf{is\text{-}parent} \circ \mathsf{is\text{-}parent}^*).\mathsf{Famous}.$$

**Theorem 6.8.** *Deciding concept satisfiability in* $\mathcal{L}_{s\mathcal{ROIQ}_{trans}}$ *is NEXPTIME-hard.* $\lhd$

Many description logics use extensions of $\mathcal{L}_{\mathcal{ALC}}$ that are contained in $\mathcal{L}_{s\mathcal{ROIQ}_{\text{trans}}}$. Typical variants are:

- One allows for subsumption of role names, i.e., TBoxes may contain formulae of the form

  $$r \sqsubseteq r',$$

  where $r$ and $r'$ are role names.

- Instead of the transitive closure operator, one introduces transitive role names. Then, for example, one can express that parents are ancestors, but in such languages the role name "ancestor" is not definable in terms of parent.

A good resource for looking up the complexity of different description logics is the *Description Logic Complexity Navigator* (http://www.cs.man.ac.uk/˜ezolin/dl/).

## References

Andréka, H., van Benthem, J., and Németi, I. (1995). Back and forth between modal logic and classical logic. *Logic Journal of the IGPL*, 3(5):685–720.

Areces, C., de Rijke, M., and de Nivelle, H. (2001). Resolution in modal, description and hybrid logic. *J. Log. Comput.*, 11(5):717–736.

Baader, F. and Nutt, W. (2003). Basic description logics. In Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors, *Description Logic Handbook*, pages 43–95. Cambridge University Press.

Baader, F. and Sattler, U. (2001). An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40.

Blackburn, P., de Rijke, M., and Venema, Y. (2002). *Modal Logic*. Cambridge Tracks in Theoretical Computer Science. Cambridge University Press.

Buchheit, M., Donini, F. M., and Schaerf, A. (1993). Decidable reasoning in terminological knowledge representation systems. *J. Artif. Intell. Res. (JAIR)*, 1:109–138.

Burgess, J. P. (1984). Basic tense logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 89–133. Reidel, Dordrecht.

Chagrov, A. and Zakharyaschev, M. (1997). *Modal Logic*. Oxford University Press.

Chellas, B. F. (1980). *Modal Logic: An Introduction*. Cambridge University Press.

Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263.

Ebbinghaus, H.-D., Flum, J., and Thomas, W. (2007). *Einführung in die mathematische Logik (5. Aufl.)*. Spektrum Akademischer Verlag.

Emerson, E. A. (1990). Temporal and modal logic.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. (1995). *Reasoning About Knowledge*. MIT Press.

Fischer, M. J. and Ladner, R. E. (1977). Propositional modal logic of programs (extended abstract). In *STOC*, pages 286–294. ACM.

Fitting, M. (1996). *First-order logic and automated reasoning (2. ed.)*. Graduate texts in computer science. Springer.

Gabbay, D., Kurucz, A., Wolter, F., and Zakharyaschev, M. (2003). *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier.

Galton, A. (2008). Temporal logic. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Fall 2008 edition.

Giunchiglia, F. and Sebastiani, R. (2000). Building decision procedures for modal logics from propositional decision procedures: The case study of modal k(m). *Inf. Comput.*, 162(1-2):158–178.

Goranko, V. and Otto, M. (2007). Model theory of modal logic. In Blackburn, P., Benthem, J. V., and Wolter, F., editors, *Handbook of Modal Logic*, pages 249–330. Elsevier.

Halpern, J. Y. (1995). The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artif. Intell.*, 75(2):361–372.

Hintikka, J. (1962). *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca.

Hladik, J. (2002). Implementation and optimisation of a tableau algorithm for the guarded fragment. In Egly, U. and Fermüller, C. G., editors, *TABLEAUX*, volume 2381 of *Lecture Notes in Computer Science*, pages 145–159. Springer.

Horrocks, I. (2003). Implementation and optimization techniques. In Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors, *Description Logic Handbook*, pages 306–346. Cambridge University Press.

Horrocks, I., Sattler, U., and Tobies, S. (2000a). Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–263.

Horrocks, I., Sattler, U., and Tobies, S. (2000b). Practical reasoning for very expressive description logics. *CoRR*, cs.LO/0005013.

Hughes, G. E. and Cresswell, M. J. (1985). *A Companion to Modal Logic*. Routledge Kegan & Paul.

Hughes, G. E. and Cresswell, M. J. (1990). *An Introduction to Modal Logic*. Routledge.

Hughes, G. E. and Cresswell, M. J. (1996). *A New Introduction to Modal Logic*. Routledge Chapman & Hall.

Kracht, M. (1999). *Tools and Techniques in Modal Logic*. Elsevier.

Kutschera, F. v. (1976). *Einführung in die intensionale Semantik*. Walter de Gruyter.

Kutschera, F. v. and Breitkopf, A. (2007). *Einführung in die moderne Logik (8. Aufl.)*. Alber.

Ladner, R. E. (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480.

Lenzen, W. (1981). *Glauben, Wissen und Wahrscheinlichkeit: Systeme der epistemischen Logik*. Springer-Verlag.

Nguyen, L. A. (2004). On the complexity of fragments of modal logics. In Schmidt, R. A., Pratt-Hartmann, I., Reynolds, M., and Wansing, H., editors, *Advances in Modal Logic*, pages 249–268. King's College Publications.

Pnueli, A. (1977). The temporal logic of programs. In *FOCS*, pages 46–57. IEEE.

Pratt, V. R. (1976). Semantical considerations on Floyd-Hoare Logic. In *FOCS*, pages 109–121. IEEE.

Prior, A. N. (1957). *Time and Modality*. Clarendon Press, Oxford.

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

Schöning, U. (2000). *Logik für Informatiker (5. Aufl.)*. Reihe Informatik. Spektrum Akademischer Verlag.

van Benthem, J. (1984). Correspondence theory. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic: Volume II: Extensions of Classical Logic*, pages 167–247. Reidel, Dordrecht.

Verbrugge, R. L. (2010). Provability logic. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Winter 2010 edition.

von Wright, G. H. (1951). *An Essay in Modal Logic*. North-Holland, Amsterdam.