Lecture Notes, Summer Term 2012

Advanced Logics

Dr. Arne Meier

Version of March 19, 2020

Institut für Theoretische Informatik
Leibniz Universität Hanover

Contents

1. Post's Lattice

. . . after all it is not ideas but the execution of ideas that constitute a mark of greatness.

(Emil Post in a Letter to Kurt Gödel, 30.10.1938)

In this chapter we will repeat some of the fundamental definitions which are the basis for the modal logics we will investigate later in Chapter 2. The complexity of a decision problem settled in (an extension of) propositional logic may depend on several aspects. One possibility is that the difficulty to solve such a problem sticks to the available Boolean functions. Thus, the aim to study this hardness in full detail implies working with infinite many different restrictions on the problem where each such problem is defined via the allowed Boolean connectives. For taming this unrestricted size of sets Emil Post defined 1941 a lattice of all Boolean functions. Of course this lattice still has infinite size but the structure allows to circumvent this evil part. In this lecture we will drift through the landscape of modal logics and visit several prominent and to practical applications of the computer scientist very relevant extensions of this logic. There we will understand how intractability of a recent problem will interact with not only Boolean concepts within this logic but also with concepts and operators which are significant for this logic.

In the first section we will start with the universal algebra around Post's lattice and use this concept to define the upcoming logics.

1.1. Properties of Boolean Functions

A *Boolean function* f is a mapping from $\{0,1\}^n$ to $\{0,1\}$, for $n \in \mathbb{N}$. Denote with B_n the set of all n-ary Boolean functions.

- f is c-*reproducing* if $f(c,\dots,c) = c$.
- f is *monotone* if $a_1 \leqslant b_1, \dots, a_n \leqslant b_n$ implies $f(a_1,\dots,a_n) \leqslant f(b_1,\dots,b_n)$.
- f is c-*separating* if there exists an $i \in \{1,\dots,n\}$ such that $f(a_1,\dots,a_n) = c$ implies $a_i = c$.
- f is c-*separating of degree* n if all $A \subseteq f^{-1}(c)$ with $|A| = n$ are c-separating; we say $A \subseteq \{0,1\}^n$ is c-separating if there exists an $1 \leqslant i \leqslant n$ such that for all $(b_1,\dots,b_n) \in A$ it holds that $b_i = c$.
- f is *self-dual* if $f \equiv \mathbf{dual}(f)$, where $\mathbf{dual}(f)(x_1,\dots,x_n) = \neg f(\neg x_1,\dots,\neg x_n)$.
- f is *linear* (or *affine*) if $f(x_1,\dots,x_n) \equiv c_0 \oplus c_1x_1 \oplus \dots \oplus c_nx_n$ for some $c_i \in \{\bot,\top\}$ and $x \wedge y$ is written as xy.

For the special case $n = 0$, there are two 0-ary functions, namely, 0 and 1 (other used symbols for these functions are $\bot$ and $\top$). The 1-ary functions are *identity* $\mathbf{id}(x)$ and *negation* $\neg x$ or $\bar{x}$. Some prominent 2-ary functions are *and* $x \wedge y$, *or* $x \vee y$, *implies* $x \rightarrow y$, *exclusive-or* $x \oplus y$, *nand* $x|y$, *equivalent* $x \leftrightarrow y$.

For *projection functions* $\mathbf{proj}_{i,n}$ with $1 \leqslant i \leqslant n \in \mathbb{N}$ it holds that $\mathbf{proj}_{i,n}(x_1, \ldots, x_n) =_{\mathsf{def}} x_i$, informally, the function is a projection from n elements to one of its elements $1 \leqslant i \leqslant n$.

For $n \geqslant m \in \mathbb{N}$,

$$\mathsf{T}^n_m =_{\mathsf{def}} \bigvee_{\substack{S \subseteq \{1,\ldots,n\}, \\ |S|=m}} \bigwedge_{i \in S} x_i$$

defines a threshold function requiring m bits out of n set to $\top$.

Definition (Compositions). *Let* B *be a set of Boolean functions. The* closure under arbitrary composition of B*, written* $\langle B \rangle$*, is defined as follows:* $f \in \langle B \rangle$ *iff* $f \in B$ *or there are* $g \in \langle B \rangle$ *and* $\mathcal{X}_1, \ldots, \mathcal{X}_n$ *which are variables or functions from* $\langle B \rangle$*, such that* $f \equiv g(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ *holds.*

Definition (Clone). *A set of Boolean functions* B *is a* clone *if it contains all projection functions and is closed under arbitrary composition. The* smallest clone *is denoted with* $[B]$*.*

If $B = \{f_1, \ldots, f_n\}$ is a set of Boolean function then we will always write $[f_1, \ldots, f_n]$ and omit the curly braces inside of $[\cdot]$.

Example. *Let* $f \in B_2$ *such that* $f(x,y) =_{\mathsf{def}} x \wedge \bar{y}$ *holds. Is the function* $x \wedge y$ *in* $[f]$*? Thus it suffices to find a composition of* f*'s expressing* $x \wedge y$*, which can be done via* $f(x, f(x,y))$*.*

Definition (Base). *Let* B *be a set of Boolean functions. Every set* $B_0 \subseteq B$ *with* $[B_0] = B$ *is called a* base *of* B*.*

Definition. *The following clones are called* basic clones*:*

- BF *is the class of all Boolean functions,*
- *for* $a \in \{0,1\}$*,* R_a *contains all* a*-reproducing functions,*
- M *contains all monotone functions,*
- D *contains all self-dual functions,*
- L *contains all linear functions,*
- *for* $a \in \{0,1\}$*,* S_a *contains all* a*-separating functions,*
  and for $k \in \mathbb{N}$*,* S^k_a *contains all* a*-separating functions of level* k*,*
- E *contains all conjunction functions (plus both constants),*
- V *contains all disjunction functions (plus both constants),*
- I_2 *(resp.,* I*) contains all projections (and all constants),*
- N_2 *(resp.,* N*) contains all projections and all negations of projections (and all constants).*

Definition. *Let* A *and* B *be clones and let* $\mathsf{A} \sqcap \mathsf{B}$ *(resp.* $\mathsf{A} \sqcup \mathsf{B}$*) be the largest (smallest) clone that is contained in (resp. contains) both* A *and* B*.*

Theorem 1.1.
If A *and* B *are clones, then* $\mathsf{A} \cap \mathsf{B} = \mathsf{A} \sqcap \mathsf{B}$ *holds.*

Proof. $\mathsf{A} \cap \mathsf{B} \subseteq [\mathsf{A} \cap \mathsf{B}]$ is true by definition. As $\mathsf{A} \cap \mathsf{B} \subseteq \mathsf{A}$ holds therefore we have $[\mathsf{A} \cap \mathsf{B}] \subseteq [\mathsf{A}] = \mathsf{A}$. Conclusively it holds that $[\mathsf{A} \cap \mathsf{B}] = \mathsf{A} \cap \mathsf{B}$, hence $\mathsf{A} \cap \mathsf{B}$ is again a clone. Since $\mathsf{A} \sqcap \mathsf{B}$ is the largest clone that is contained in both A and B, it holds that $\mathsf{A} \cap \mathsf{B} \subseteq \mathsf{A} \sqcap \mathsf{B}$. From the definition we obtain $\mathsf{A} \sqcap \mathsf{B} \subseteq \mathsf{A} \cap \mathsf{B}$ which lets us conclude that $\mathsf{A} \cap \mathsf{B} = \mathsf{A} \sqcap \mathsf{B}$. Similarly it holds $[\mathsf{A} \cup \mathsf{B}] = \mathsf{A} \sqcup \mathsf{B}$. It can be easily verified that $\sqcap$ and $\sqcup$ are both associative and commutative. Furthermore it holds that $\mathsf{A} \sqcap (\mathsf{A} \sqcup \mathsf{B}) = \mathsf{A}$ and $\mathsf{A} \sqcup (\mathsf{A} \sqcap \mathsf{B}) = \mathsf{A}$, hence the Boolean clones form a lattice. $\square$

Figure 1.1 shows all Boolean clones arranged in a lattice and all clones with their bases are shown in Table 1.1.

Corollary 1.2.
Any non-basic clone in Post's lattice can be obtained by intersection of two basic clones.

Definition (Functional (in)completeness). *Let* B *be a finite set of Boolean functions and* f *be a Boolean function.*

- *The function* f *is* representable by B *iff* $\mathsf{f} \equiv \varphi$ *for some formula* φ *constructed of functions in* B*.*
- B *is* functional complete *iff* $\wedge, \vee,$ *and* $\neg$ *are representable by* B*.*
- B *is* functional maximal incomplete *if and only if for any Boolean function* f *the following holds: if* f *is not representable in* B *then* $\mathsf{B} \cup \{\mathsf{f}\}$ *is functional complete.*

Remark. Post's classes *are those classes* $\mathcal{B}$ *in the lattice such that* $\mathcal{B} \subsetneq \mathsf{BF}$ *and for every* $\mathcal{B}'$ *with* $\mathcal{B} \subsetneq \mathcal{B}' \subseteq \mathsf{BF}$ *we obtain* $[\mathcal{B}'] = \mathsf{BF}$*. Hence the clones* $\mathcal{B}$ *are the maximal incomplete sets.*

Example. *Consider the Boolean function* nand $x|y$*. Obviously this function is neither* 0*-nor* 1*-reproducing wherefore it is neither contained in* R_1 *nor in* R_0*. Further the function is not monotone, as* $(0,0) \leqslant (1,1)$ *but* $0|0 \nleqslant 1|1$*. Furthermore the function is not self-dual since* $0|1 \neq \neg(1|0)$*. Finally, the function is not linear: suppose the opposite which yields* $x|y \equiv c_0 \oplus c_1 x \oplus c_2 y$*. From* $0|0 = 1$ *we have* $c_0 = 1$*, by* $1|0 = 1$ *we obtain* $c_1 = 0$*, but* $0|1 = 1$ *yields* $c_2 = 0$*, too. This is a contradiction to* nand *not being constant. Thus* $x|y$ *is not contained in one of the maximal clones which lets us conclude that* $[x|y] = \mathsf{BF}$*. Note that each Boolean function can be represented by* $x|y$*.*

Example. *Suppose* f *is an* n*-ary monotonic Boolean function. Is* f *also in* L*? Observe that* $\mathsf{L} \nsubseteq \mathsf{M}$*, and* $\mathsf{M} \cap \mathsf{L} = \mathsf{I}$*. Since* $\mathsf{I} = [\mathsf{I}_2 \cup \{\top, \bot\}]$*,* f *is in* L *if and only if it is a projection or constant function.*

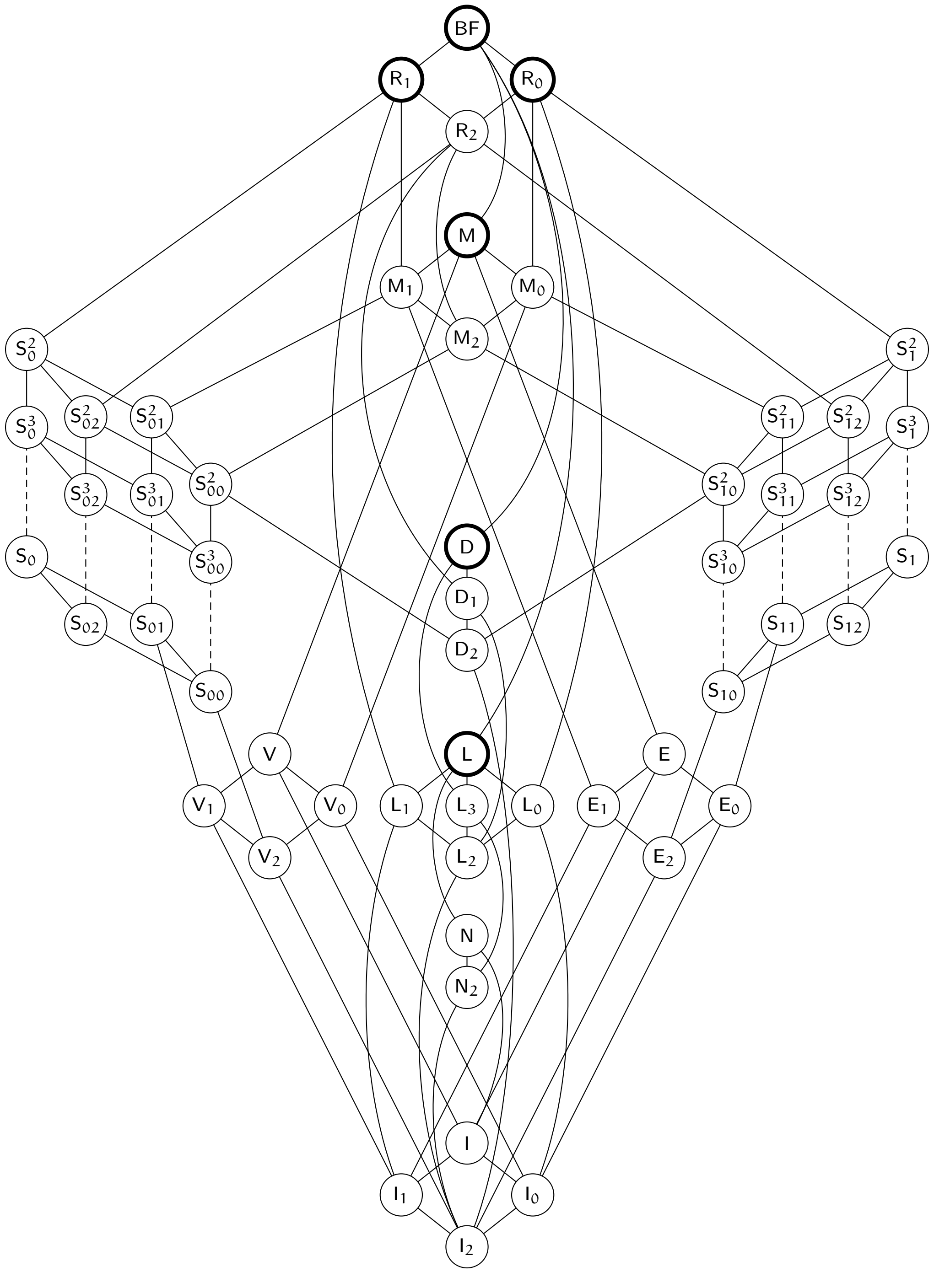


Figure 1.1.: Post's lattice. Post's classes and BF are marked with a thick border.

Class	Definition	Base	
BF	All Boolean functions	$\{x \wedge y, \neg x\}$	
R_0	$\{f \mid f$ is $\bot$-reproducing$\}$	$\{x \wedge y, x \oplus y\}$	
R_1	$\{f \mid f$ is $\top$-reproducing$\}$	$\{x \vee y, x \leftrightarrow y\}$	
R_2	$\mathsf{R}_0 \cap \mathsf{R}_1$	$\{x \vee y, x \wedge (y \leftrightarrow z)\}$	
M	$\{f \mid f$ is monotone$\}$	$\{x \vee y, x \wedge y, \bot, \top\}$	
M_0	$\mathsf{M} \cap \mathsf{R}_0$	$\{x \vee y, x \wedge y, \bot\}$	
M_1	$\mathsf{M} \cap \mathsf{R}_1$	$\{x \vee y, x \wedge y, \top\}$	
M_2	$\mathsf{M} \cap \mathsf{R}_2$	$\{x \vee y, x \wedge y\}$	
S_0	$\{f \mid f$ is $\bot$-separating$\}$	$\{x \rightarrow y\}$	
S_1	$\{f \mid f$ is $\top$-separating$\}$	$\{x \nrightarrow y\}$	
S_0^n	$\{f \mid f$ is $\bot$-separating of degree $n\}$	$\left\{x \rightarrow y, \mathrm{T}_2^{n+1}\right\}$	
S_1^n	$\{f \mid f$ is $\top$-separating of degree $n\}$	$\left\{x \nrightarrow y, \mathrm{T}_n^{n+1}\right\}$	
S_{00}	$\mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z)\}$	
S_{00}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\left\{x \vee (y \wedge z), \mathrm{T}_2^3\right\}$	if $n = 2$,
		$\left\{\mathrm{T}_2^{n+1}\right\}$	if $n \geqslant 3$
S_{01}	$\mathsf{S}_0 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \top\}$	
S_{01}^n	$\mathsf{S}_0^n \cap \mathsf{M}$	$\left\{\mathrm{T}_2^{n+1}, \top\right\}$	
S_{02}	$\mathsf{S}_0 \cap \mathsf{R}_2$	$\{x \vee (y \nrightarrow z)\}$	
S_{02}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2$	$\left\{x \vee (y \nrightarrow z), \mathrm{T}_2^{n+1}\right\}$	
S_{10}	$\mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z)\}$	
S_{10}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\left\{x \wedge (y \vee z), \mathrm{T}_2^3\right\}$	if $n = 2$,
		$\left\{\mathrm{T}_n^{n+1}\right\}$	if $n \geqslant 3$
S_{11}	$\mathsf{S}_1 \cap \mathsf{M}$	$\{x \wedge (y \vee z), \bot\}$	
S_{11}^n	$\mathsf{S}_1^n \cap \mathsf{M}$	$\left\{\mathrm{T}_n^{n+1}, \bot\right\}$	
S_{12}	$\mathsf{S}_1 \cap \mathsf{R}_2$	$\{x \wedge (y \rightarrow z)\}$	
S_{12}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2$	$\left\{x \wedge (y \rightarrow z), \mathrm{T}_n^{n+1}\right\}$	
D	$\{f \mid f$ is self-dual$\}$	$\{\mathbf{maj}\{x, \overline{y}, \overline{z}\}\}$	
D_1	$\mathsf{D} \cap \mathsf{R}_2$	$\{\mathbf{maj}\{x, y, \overline{z}\}\}$	
D_2	$\mathsf{D} \cap \mathsf{M}$	$\{\mathbf{maj}\{x, y, z\}\}$	
L	$\{f \mid f$ is linear$\}$	$\{x \oplus y, \top\}$	
L_0	$\mathsf{L} \cap \mathsf{R}_0$	$\{x \oplus y\}$	
L_1	$\mathsf{L} \cap \mathsf{R}_1$	$\{x \leftrightarrow y\}$	
L_2	$\mathsf{L} \cap \mathsf{R}_2$	$\{x \oplus y \oplus z\}$	
L_3	$\mathsf{L} \cap \mathsf{D}$	$\{x \oplus y \oplus z \oplus \top\}$	
V	$\{f \mid f$ is a disjunction or constant$\}$	$\{x \vee y, \bot, \top\}$	
V_0	$\mathsf{M}_0 \cap \mathsf{V}$	$\{x \vee y, \bot\}$	
V_1	$\mathsf{M}_1 \cap \mathsf{V}$	$\{x \vee y, \top\}$	
V_2	$\mathsf{M}_2 \cap \mathsf{V}$	$\{x \vee y\}$	
E	$\{f \mid f$ is a conjunction or constant$\}$	$\{x \wedge y, \bot, \top\}$	
E_0	$\mathsf{M}_0 \cap \mathsf{E}$	$\{x \wedge y, \bot\}$	
E_1	$\mathsf{M}_1 \cap \mathsf{E}$	$\{x \wedge y, \top\}$	
E_2	$\mathsf{M}_2 \cap \mathsf{E}$	$\{x \wedge y\}$	
N	$\{f \mid f$ depends on at most one variable$\}$	$\{\neg x, \bot, \top\}$	
N_2	$\mathsf{L}_3 \cap \mathsf{N}$	$\{\neg x\}$	
I	$\{f \mid f$ is a projection or a constant$\}$	$\{\mathbf{id}, \bot, \top\}$	
I_0	$\mathsf{R}_0 \cap \mathsf{I}$	$\{\mathbf{id}, \bot\}$	
I_1	$\mathsf{R}_1 \cap \mathsf{I}$	$\{\mathbf{id}, \top\}$	
I_2	$\mathsf{R}_2 \cap \mathsf{I}$	$\{\mathbf{id}\}$	

Table 1.1.: List of all Boolean clones with their bases

1.2. Succinctness

From a complexity theoretic aspect the main benefit of Post's lattice is the possibility to carry over lower bounds (hardness results) and upper bounds (membership results) with respect to some fragment parameterized by a clone. In the world of circuits this result is very strong, i.e., without any restrictions.

Theorem 1.3.
Let $\Gamma(B)$ be a decision problem defined over circuits and only uses Boolean gates from B. Then $\Gamma(B) \leqslant_{\mathbf{cd}} \Gamma(B')$ for all $B \subseteq [B']$ holds.

In order to prove this theorem the simple idea is simply to substitute the gates in B with the ones in B'.

Unfortunately in the propositional world of formulas this theorem does not even hold for $\leqslant^{\mathbf{P}}_{\mathbf{m}}$-reductions which is explained in the following example.

Example. *Let $B = \{\oplus\}$ and $B' = \{\wedge, \neg\}$. Then it holds that $B \subseteq [B']$. Now consider the formula family $(\varphi_n)_{n\in\mathbb{N}}$ such that $\varphi_n =_{\text{def}} x_1 \oplus \cdots \oplus x_n$. Now it holds that $x \oplus y \equiv \neg(\neg(x \wedge \bar{y}) \wedge \neg(\bar{x} \wedge y))$. The size of φ_n over B' using this equivalence leads to an formula which size is exponential in n. Thus such canonical reduction does not need to fulfill the polynomial time computable property of the reduction.*

However, by a specific precondition the result can be pulled to hold in the propositional world. In this context the term of short representations has to be considered. The main idea is to consider the representation of formulas which do not exponentially enlarge formulas when substituting them with the new function happens.

Definition (Short representations). *If B is a set of Boolean functions, then an n-ary Boolean function $f \in B_n$ has a* short representation *in B if and only if there is a B-formula φ (which uses only connectives from B) with $f \equiv \varphi$ and each variable of f occurs at most once in the body of φ. Also we say B* efficiently implements *f. Similarly if B' is a set of Boolean functions, we say that B' efficiently implements B iff for all functions $f \in B$ the set B' efficiently implements f.*

Hence, we immediately can deduce that the investigated representation of the exclusive-or function in the previous example is not a short representation because both arguments of the function occur twice in the substitution.

Corollary 1.4.
Let B be a finite set of Boolean functions and $\Gamma(B)$ be a decision problem over Boolean formulas using functions in B. Let B' be a set of Boolean functions such that B' efficiently implements B. Then $\Gamma(B) \leqslant^{\mathbf{P}}_{\mathbf{m}} \Gamma(B')$ for $B \subseteq [B']$.

The following result shows that in several important clones the short representation of the function in their respective base is given. This lemma will be used later on in several proofs.

Lemma 1.5 (Short representations, [Lewis, 1979, Schnoor, 2010, Thomas, 2010]). *Let* B *be a finite set of Boolean functions.*

(1.) If $[\mathsf{B}] = \mathsf{BF}$*, then* B *efficiently implements* $\{\vee, \wedge, \neg\}$*.*

(2.) If $\mathsf{N} \subseteq [\mathsf{B}]$*, then* B *efficiently implements* $\neg$ *via some formula* f*. If* $[\mathsf{B}] \subseteq \mathsf{L}$*, then* f *can be chosen in such a way that the variable* x *occurs in* f *as the last symbol.*

(3.) If $[\mathsf{B}] = \mathsf{L}$*, then* B *efficiently implements* $\oplus$*.*

(4.) If $\mathsf{L}_2 \subseteq [\mathsf{B}] \subseteq \mathsf{L}$*, then* B *efficiently implements* $x \oplus y \oplus z$*.*

(5.) If $[\mathsf{B}] \in \{\mathsf{V}, \mathsf{M}\}$ *(*$[\mathsf{B}] \in \{\mathsf{E}, \mathsf{M}\}$*, resp.), then* B *efficiently implements* $\vee$ *(resp.* $\wedge$*).*

Proof. (1.)+(2.) First we show that $\neg$ is efficiently implemented. Since $[\mathsf{B}] = \mathsf{BF}$ is functional complete there exist B-formulas $\top_\mathsf{B}, \bot_\mathsf{B}, \phi_\mathsf{B}^\neg$ which are equivalent to $\top, \bot$, and $\neg x$. W.l.o.g. assume that $\top_\mathsf{B}$ and $\bot_\mathsf{B}$ do not contain occurrences of x.

If $\phi_\mathsf{B}^\neg$ contains only one occurrence of x then $\phi_\mathsf{B}^\neg$ is the required formula representing $\neg$. Otherwise $\phi_\mathsf{B}^\neg$ has $n \geqslant 2$ occurrences of x. Any easy induction now proves that substituting either the first or the second occurrence of x in $\phi_\mathsf{B}^\neg$ leads to the desired formula representing $\neg$. The cases $\wedge$ and $\vee$ are left as an exercise.

(3.) Since $\mathsf{L} \subseteq [\mathsf{B}]$ there is a B-formula $f(x, y)$ such that f represents $x \oplus y$. Since $[\mathsf{B}] \supseteq \mathsf{L}$, we know $f_\#(x_1, \ldots, x_n)$ represents a function of the form $c \oplus x_{i_1} \oplus \cdots \oplus x_{i_k}$ for some $c \in \{0, 1\}$ and $i_1, \ldots, i_k \in \{1, \ldots, n\}$. Obviously we can replace all but two of the variables by 0 and end up with a formula for $\oplus$ having exactly one occurrence of each variable.

(4.) Let $\mathsf{L}_2 \subseteq [\mathsf{B}]$ and let $g(x, y, z) \in [\mathsf{B}]$. As g is affine the value of g does not change if we replace two occurrences of any variable with a fresh variable t. Assume x occurs an even number of n-times in g. Thus there is a function g' such that $g'(y, z, t) \equiv y \oplus z \not\equiv g(x, y, z)$ leading to a contradiction when $[\mathsf{B}] = \mathsf{L}_2 \not\ni y \oplus z$. Similarly one can argue for y and z. Hence x, y, and z occur an odd number of times. If we now replace all but one occurrence of each x, y, and z with t, we get a function $g'(x, y, z, t) \equiv x \oplus y \oplus z$ and all variables occur exactly once.

(5.) We will prove the statement for $\vee$. The case for $\wedge$ follows from the duality principle. Again, assume that $n \geqslant m \geqslant 2$ denotes n occurrences of x and m of y in $\varphi(x, y) \equiv x \vee y$ and $\varphi(x, y)$ is minimal w.r.t. the occurrences of its variables. Order the variables and define $\varphi_\#(x_1, \ldots, x_n, y_1, \ldots, y_m)$. Since $\top \in [\mathsf{B}]$ we can construct a B-formula which is equivalent to

$$\varphi'(x, y) =_{\mathsf{def}} \varphi_\#(x_1/1, x_2/x, \ldots, x_n/x, y_1/y, \ldots, y_m/y).$$

As φ was chosen minimal the formula $\varphi'(x, y)$ does not represent $x \vee y$. Since $[\mathsf{B}] \subseteq \mathsf{M}$ the function represented by $\varphi_\#$ is monotone whence $\varphi'(x/\alpha, y/\beta) \geqslant \alpha \vee \beta$ holds for $\alpha, \beta \in \{0, 1\}$. Thus it holds that

$$\varphi'(x/0, y/1) = \varphi'(x/1, y/0) = \varphi'(x/1, y/1) = 1.$$

Now assume that $\varphi'(x/0, y/0) = 0$ holds. This implies that φ' represents $\vee$ which is a contradiction to φ's minimality. Hence $\varphi'(x/0, y/0) = 1$ must hold which leads to $\varphi'(x/\alpha, y/\beta) = 1$ for all α, β. In particular we get

$$\varphi'(x/0, y/0) = \varphi_{\#}(x_1/1, x_2/0, \dots, x_n/0, y_1/0, \dots, y_m/0) = 1,$$

and since $\varphi_{\#}$ is of course monotone, we have

$$\varphi_{\#}(x_1/1, x_2/\alpha_2, \dots, x_n/\alpha_n, y_1/\beta_1, \dots, y_m/\beta_m) = 1 \qquad (\star)$$

for all $\alpha_2, \dots, \beta_m \in \{0, 1\}$. Since $\bot \in [B]$ we can construct the formula

$$\varphi''(x, y) =_{\mathsf{def}} \varphi(x_1/x, x_2/0, x_3/x, \dots, x_n/x, y_1/y, \dots, y_m/y).$$

Now observe that the following holds:

$$\begin{aligned}
&\varphi''(x/1, y/0) \overset{(\star)}{=} \varphi_{\#}(x_1/1, x_2/0, x_3/1, \dots, x_n/1, y_1/0, \dots, y_m/0) = 1 \\
&\varphi''(x/1, y/1) \geqslant \varphi''(x/1, y/0) = 1 && (\varphi'' \text{ is monotone}) \\
&\varphi''(x/0, y/1) = \varphi(x/0, y/1) = 1 && (\text{choice of } \varphi) \\
&\varphi''(x/0, y/0) = \varphi(x/0, y/0) = 0 && (\text{choice of } \varphi).
\end{aligned}$$

Hence φ'' represents $\vee$, and φ'' has one variable less than φ which contradicts the minimality of φ. Thus φ contains only two variable occurrences. ☐

1.3. Propositional Logic and Satisifability

Atomic formulas are the constant functions $\top, \bot$, and every *atomic proposition* (or *variable*) $p \in \mathsf{PROP}$ where PROP is the set of all atomic propositions. If B be a finite set of Boolean functions, then the set of all *propositional formulas restricted to* B, in symbols $\mathrm{PL}(B)$, is defined inductively via $\varphi ::= p \mid f(\varphi, \dots, \varphi)$, for $p \in \mathsf{PROP}$ and for all $f \in B$. For the case $B = \{\wedge, \neg\}$ it holds that $\mathrm{PL}(B)$ consists of the *full set of propositional formulas*, denoted by only PL, as $[B] = \mathsf{BF}$. Further slightly abusing the notation we also write $\mathrm{PL}(\mathcal{B})$ to denote the set of all propositional formulas φ defined over functions from $\mathcal{B}$ if $\mathcal{B}$ is a clone. For a given formula $\varphi \in \mathrm{PL}$ let denote $\mathbf{Vars}(\varphi)$ denote the set of the variables in φ.

An *assignment* θ is a function $\theta\colon \{x_1, \dots, x_n\} \to \{\top, \bot\}$ for $n \in \mathbb{N}$. Now define the extended assignment function $\hat{\theta}\colon \mathrm{PL} \to \{\top, \bot\}$ as follows in the usual inductive way:

$$\hat{\theta}(p) =_{\mathsf{def}} \theta(p), \quad \hat{\theta}(\top) =_{\mathsf{def}} \top, \quad \hat{\theta}(\bot) =_{\mathsf{def}} \bot$$
$$\hat{\theta}(f(\varphi_1, \dots, \varphi_n)) =_{\mathsf{def}} f(\hat{\theta}(\varphi_1), \dots, \hat{\theta}(\varphi_n)),$$

where $p \in \mathsf{PROP}$, $f \in B_n$, and $\varphi_i \in \mathrm{PL}$ for $1 \leqslant i \leqslant n \in \mathbb{N}$. Let B be a set of Boolean functions. We say a formula $\varphi \in \mathrm{PL}(B)$ is satisfiable if and only if there exists an assignment θ such that $\hat{\theta}(\varphi) = \top$; the corresponding problem will be denoted by $\mathrm{SAT}(B)$, similarly if B is a clone.

Theorem 1.6 (Lewis' Dichotomy Theorem, 1979).
Let B *be a set of Boolean functions. If* $\nrightarrow \in [\mathsf{B}]$*, then* $\mathrm{SAT}(\mathsf{B})$ *is* NP*-complete. For any other case* $\mathrm{SAT}(\mathsf{B}) \in \mathsf{P}$ *is true.*

Proof. Due to Cook's famous theorem from 1971 the full fragment $\mathrm{SAT}(\mathsf{BF})$ is NP-complete in combination with Lemma 1.5. At first observe that $[\nrightarrow] = \mathsf{S}_1 \supset \mathsf{E}_2 = [x \wedge y]$ and also $[\mathsf{S}_1 \cup \{\top\}] = \mathsf{BF}$ (cf. Figure 1.1), thus $\mathsf{B} \cup \{\top\}$ efficiently implements $\{\wedge, \vee, \neg\}$ for $[\mathsf{B}] \supseteq \mathsf{S}_1$ by Lemma 1.5. By Corollary 1.4 we can conclude that $\mathrm{SAT}(\mathsf{B} \cup \{\top\})$ is NP-hard. Finally we need to prove that $\mathrm{SAT}(\mathsf{B} \cup \{\top\}) \leqslant^{\mathbf{P}}_{\mathbf{m}} \mathrm{SAT}(\mathsf{B})$. Substitute every occurrence of the constant function $\top$ in φ with a fresh variable t and let denote this change by $\varphi|_{\top/t}$. Then it holds that $\varphi|_{[\top/t]} \wedge t$ is satisfiable iff φ is satisfiable.

Now we turn towards the polynomial time cases which must correspond to the remaining clones, that is, with $\mathsf{R}_1, \mathsf{M}, \mathsf{D}$, and L we have four cases to differentiate. For R_1 every formula is 1-reproducing, thus satisfiable. Every monotone formula φ is satisfiable iff the assignment which maps every variable to true satisfies φ. Every D-formula is self-dual, thus either the all 1's assignment satisfies φ or the all 0's assignment. Finally consider the case L. Since $\{\oplus, \top\}$ is a basis for L we consider only formulas over these functions. Algorithm 1.1 on page 11 solves this case in polynomial runtime. □

Algorithm 1.1: Procedure deciding $\mathrm{SAT}(\mathsf{B})$ for $[\mathsf{B}] \subseteq \mathsf{L}$ in polynomial time.

```
  Input : $\varphi \in \mathrm{PL}(\mathsf{B})$ with $[\mathsf{B}] \subseteq \mathsf{L}$
1 rename the variables in $\varphi$ to $x_1, \dots, x_n$;
2 rename the variables $x_i$ to $x_{i,j}$ if it is the jth occurrence of $x_i$ for $1 \leqslant i \leqslant n$ and some
  $1 \leqslant j \in \mathbb{N}$;
3 $\varphi$ is then equal to

      $\varphi' \equiv t_1 \oplus t_2 \oplus \cdots \oplus t_\ell \oplus x_{1,1} \oplus \dots \oplus x_{1,m_1} \oplus \cdots \oplus x_{n,1} \oplus \cdots \oplus x_{n,m_n}$,

  where $\ell, m, m_v \in \mathbb{N}$ for $1 \leqslant m_v \leqslant m$, and $\ell$ is the number of $\top$'s in $\varphi$;
4 compute $S := \{t_i \mid 1 \leqslant i \leqslant \ell\} \cup \{x_{i,j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m_i\}$;
5 if $\ell \equiv_2 1$ then delete $t_2, \dots, t_\ell$ from S else delete $t_1, t_2, \dots, t_\ell$ from S;
6 for $i = 1, \dots, n$ do
7     if $m_i \equiv_2 1$ then delete $x_{i,2}, \dots, x_{i,m_i}$ from S else delete $x_{i,1}, x_{i,2}, \dots, x_{i,m_i}$ from S;
8 if $S \neq \emptyset$ then return true else return false;
```

1.4. Expressivity of Propositional Logic

With respect to the end of this lecture where we turn towards Descriptive Complexity, we want to mention what one can express with propositional logic at all. Consider the graph coloring problem

Problem (Colorability)
Input: A graph $G = (V, E)$, where V is the set of vertices and E is the set of edges; a natural number k of colors.

Question: Can all nodes $v \in V$ be colored with one of the k colors such that for every edge $(u, v) \in E$ the color of u is different from the color of v?

Of course it must be possible to encode such a problem into propositional logic as we know from the lecture *Komplexität von Algorithmen* (Complexity of Algorithms) that graph coloring is NP-complete.

Sketch of Construction. We use $|V| \cdot k$ propositions p_{ij} where $1 \leqslant i \leqslant |V|$ and $1 \leqslant j \leqslant k$ denoting that "Node i has color j". Further we have to say that

- Each node has (at least) one color: $\bigwedge_{i=1}^{|V|} \bigvee_{j=1}^{k} p_{ij}$
- Each node has no more than one color: $\bigwedge_{i=1}^{|V|} \bigwedge_{j=1}^{k} \left(p_{ij} \to \bigwedge_{\ell \neq j} \neg p_{i\ell} \right)$
- Nodes connected in G have different colors: $\bigwedge_{(u,v) \in E} \bigwedge_{j=1}^{k} (p_{f(u)j} \to \neg p_{f(v)j})$, where $f(u)$ is equal to the number of vertex $u \in V$.

Thus the formula $\varphi_{G,k}$ which is a conjunction of the three items above is satisfiable iff G is colorable with k colors.

Observation. *If we are able to state the reduction from Colorability to* SAT(B) *sketched above with a set of Boolean functions* B *such that* $S_1 \not\subseteq [B]$ *holds, then* $\mathsf{P} = \mathsf{NP}$ *by Theorem 1.6. More formally, given an instance* (G, k) *of the Colorability problem, if there exists a* $\leqslant_{\mathbf{m}}^{\mathbf{P}}$*-reduction computed by* f *s.t.* $f(G, k) = \phi$ *and* $\phi \equiv \varphi_{G,k}$ *s.t.* $\phi \in$ SAT(B) *for* $S_1 \not\subseteq [B]$*, then* $\mathsf{P} = \mathsf{NP}$*.*

2. Modal Logic

It really is a nice theory. The only defect I think it has is probably common to all philosophical theories. It's wrong.

(Saul Kripke, Naming and Necessity (1980, p. 64))

Modal logics origin can be dated back to until Aristoteles, or more recently, to Leibniz who worked with the expressibility of "possibly" and "necessarily" in logic. Due to a lack of these expressions in propositional logic an extension is motivated. This extension yields the $\Diamond$ and $\Box$ operator. Given a set B of Boolean functions the modal logic $\mathrm{ML}(B)$ is defined via the following grammar

$$\varphi ::= \psi \mid \Box\varphi \mid \Diamond\varphi \mid f(\varphi, \ldots, \varphi),$$

where $\psi \in \mathrm{PL}(B)$ and $f \in B$; it holds that $\Diamond\varphi =_{\mathsf{def}} \neg\Box\neg\varphi$. For the semantical point of view we first have to deal with an appropriate assignment model which extends the usual assignments of variables in order to cope with the introduction of the two new operators.

2.1. Frames

Definition. *A* frame $\mathcal{F}$ *is a tuple* $\mathcal{F} = (W, \mathbf{R})$*, where* W *is a set of worlds and* $\mathbf{R} = \{R_1, \ldots, R_n\}$ *is a finite set of transition relations such that* $R_i \subseteq W \times W$ *holds for* $1 \leqslant i \leqslant n$*. If* $|\mathbf{R}| = 1$ *holds then we also write* (W, R)*.*

Usually unless otherwise mentioned we will stick to a single modality in this lecture. Now we are able adjust the definition of assignments to the modal world.

Definition. *Let* PROP *be the set of atomic propositions. A* model *is a tuple* $\mathcal{M} = (\mathcal{F}, V)$ *such that* $F = (W, \mathbf{R})$ *is a frame and* $V\colon \mathsf{PROP} \to \mathfrak{P}(W)$ *is a valuation function.*

Informally speaking, the function V labels propositions to states, or says in which points of the frame a proposition holds. The frame F can be seen as a transition system. The corresponding semantics of modal logic formulas are defined as follows.

Definition. *Let* $\mathcal{M} = (\mathcal{F}, V)$ *be a model over the frame* $\mathcal{F} = (W, \mathbf{R})$*. Then the satisfaction relation* $\models$ *is defined inductively as follows:*

$$\begin{array}{lll}
\mathcal{M}, w \models \top & & \textit{always holds}, \\
\mathcal{M}, w \models \bot & & \textit{never holds}, \\
\mathcal{M}, w \models p & \textit{iff} & p \in \mathsf{PROP} \land w \in V(p), \\
\mathcal{M}, w \models \Box_R\varphi & \textit{iff} & \forall v \in R(w) : \mathcal{M}, v \models \varphi, \\
\mathcal{M}, w \models f(\varphi_1, \ldots, \varphi_n) & \textit{iff} & f(\llbracket \mathcal{M}, w \models \varphi_1 \rrbracket, \ldots, \llbracket \mathcal{M}, w \models \varphi_n \rrbracket) = \top,
\end{array}$$

Name	Conditions	First-order description
$\mathbf{K}$	–	–
$\mathbf{D}$	serial (total)	$\forall v \in W \exists w \in W : vRw$
$\mathbf{T}$	reflexive	$\forall w \in W : wRw$
$\mathbf{K}4$	transitive	$\forall u, v, w \in W : (uRv \land vRw) \to uRw$
$\mathbf{S}4$	reflexive, transitive	$\mathbf{T}$ and $\mathbf{K}4$
$\mathbf{S}5$	reflexive, transitive, Euclidean	$\mathbf{S}4$ and $\forall u, v, w \in W : (uRw \land vRw) \to uRv$

Table 2.1.: Frame classes. The last two columns refer to a frame (W, R)

where $\varphi \in \mathrm{ML}$*,* $f \in B_n$*,* $R \in \mathbf{R}$ *and* $[\![\mathcal{M}, w \models \varphi_i]\!] = \top$ *iff* $\mathcal{M}, w \models \varphi_i$ *holds for* $1 \leqslant i \leqslant n$*.*

Notation: Sometimes in literature $\Box_R$ is written $[R]$, or resp., $\Diamond_R$ is written as $\langle R \rangle$.

Given a set of Boolean formulas B, we say a modal logic formula $\varphi \in \mathrm{ML}(B)$ is *satisfiable* w.r.t. the frame class $\mathcal{C}$ if and only if there exists a model $\mathcal{M} = (\mathcal{F}, V)$ with $\mathcal{F} = (W, \mathbf{R})$ such that $\mathcal{F} \in \mathcal{C}$ holds and there exists some $w \in W$ with $\mathcal{M}, w \models \varphi$ (sometimes we also write $\mathcal{M} \models \varphi$ without mentioning the world explicitly). We denote this problem with $\mathcal{C}$-ML-SAT$^i(B)$ where $i \in \mathbb{N}$ denotes the number of modalities. Usually, whenever $\mathcal{C} = \mathbf{K}$, we just write ML-SAT$^i(B)$, and if also $[B] = \mathsf{BF}$ holds we write ML-SAT^i. Whenever we want to speak only about the case $i = 1$ we just omit the upper index and just write ML-SAT which will be our focus for this lecture.

Often one wants to consider only a specific set of frames, e.g., only total frames modeling a system which never terminates. Table 2.1 shows some important properties of frames and their first order description.

2.2. Enforcing the Size of a Model

In this section we want to investigate what sizes of models can be enforced by a given modal formula.

Theorem 2.1.
There exists a family of modal formulas $(\varphi_n)_{n \in \mathbb{N}} \in \mathrm{ML\text{-}SAT}$ *with* $|\varphi_n| \in O(n^2)$ *such that for any model* $\mathcal{M}$ *with* $\mathcal{M} \models \varphi_n$ *it holds that* $|\mathcal{M}| \geqslant 2^n$*.*

Proof. In the following we will express a binary tree with a modal formula. There the bits will correspond to the propositions $p_1, \dots, p_n$ whose all possible combinations will occur in the leafs of the tree. The formula φ_n will use two shortcuts:

$$\begin{aligned} \mathit{branch}(p_i) &=_{\mathsf{def}} \Diamond p_i \land \Diamond \neg p_i \\ \mathit{store}(p_i) &=_{\mathsf{def}} (p_i \to \Box p_i) \land (\neg p_i \to \Box \neg p_i). \end{aligned}$$

Now we can define the formula φ_n as follows:

$$\varphi_n =_{\mathsf{def}} \mathit{branch}(p_1) \land \bigwedge_{i=1}^{n-1} \Box^i \left(\mathit{branch}(p_{i+1}) \land \bigwedge_{j=1}^{i} \mathit{store}(p_j) \right),$$

where $\Box^i \varphi =_{\mathsf{def}} \overbrace{\Box \cdots \Box}^{\text{i many}} \varphi$. For the correctness, any model $\mathcal{M}$ such that $\mathcal{M} \models \varphi_n$ holds contains a complete binary tree of depth n which consists of 2^n leafs. For each variables subset of $S \subseteq \{p_1, \dots, p_n\}$ there is a leaf world $\ell \in \mathcal{M}$ such that $\ell \in V(p)$ for every $p \in S$.$\Box$

An $\mathbf{R}$-*path* π in a given frame $\mathcal{F} = (W, \mathbf{R})$ for $R \in \mathbf{R}$ is a sequence of states $\pi = w_1, \dots, w_n$ such that for every $1 \leqslant i \leqslant n \in \mathbb{N}$ there is an $R \in \mathbf{R}$ s.t. $w_i R w_{i+1}$. Further we say π is infinite if $|\pi| = \infty$ (note: this can be the case for finite $\mathcal{F}$). Furthermore we say that π is *exponentially deep* w.r.t. $k \in \mathbb{N}$ iff π is an $\mathbf{R}$-path, $2^k < n$, and $w_i \neq w_j$ for all $1 \leqslant i \neq j \leqslant 2^k$ hold.

Let E and A denote *existential/universal global modalities* where their semantics is defined w.r.t. some model $\mathcal{M} = (\mathcal{F}, V)$ as follows

$$\begin{aligned} \mathcal{M} \models \mathrm{E}\varphi \quad &\text{iff} \quad \exists w \in \mathcal{F} : \mathcal{M}, w \models \varphi, \\ \mathcal{M} \models \mathrm{A}\varphi \quad &\text{iff} \quad \forall w \in \mathcal{F} : \mathcal{M}, w \models \varphi. \end{aligned}$$

We write ML-SAT$_{\mathrm{E,A}}$ to denote that global modalities are allowed. Using this operator allows us to construct formulas which require exponentially deep models.

Theorem 2.2.
There exists a family of modal formulas $(\psi_n)_{n \in \mathbb{N}} \in$ ML-SAT$_{\mathrm{A}}$ with $|\psi_n| \in O(n^2)$ such that for any model $\mathcal{M}$ with $\mathcal{M} \models \psi_n$ it holds that $|\mathcal{M}| \geqslant 2^n$ and contains exponentially deep paths w.r.t. n.

Proof. Similarly as before we will use the idea of a binary counter which, this time, counts up on a path in the model and takes all values from 0 to $2^n - 1$ in consecutive states. Note that the defined formulas have length only in $O(n^2)$. At first we will define two formulas

$$\begin{aligned} \psi_{\mathit{init}} &=_{\mathsf{def}} \neg p_1 \wedge \cdots \wedge \neg p_n, \\ \psi_{\mathit{increase}} &=_{\mathsf{def}} \bigwedge_{i=1}^{n} \left(\Big(\neg p_i \wedge \bigwedge_{j<i} p_j \Big) \to \Big(\Box (p_i \wedge \bigwedge_{j<i} \neg p_j) \wedge \bigwedge_{j>i} \mathit{store}(p_j) \Big) \right), \end{aligned}$$

where empty conjunctions are set to true. Now set

$$\psi_n =_{\mathsf{def}} \psi_{\mathit{init}} \wedge \mathrm{A}(\Diamond \top \wedge \psi_{\mathit{increase}}).$$

In order to prove correctness we have to show that for every model $\mathcal{M} = ((W, R), V)$ it holds that there are $w_0, \dots, w_{2^n - 1} \in W$ and $w_i R w_{i+1}$ for $i < 2^n - 1$ holds. For w_0 it holds that for any proposition p_i we have $w_0 \notin V(p_i)$ by ψ_{init}. Then every world must be total by $\Diamond \top$ in the A subformula. ψ_{increase} requires the behavior of a counter. Let $\vec{x} \in \{0, 1\}^n$ be a short description of which propositions hold in world x. Then consider a world $w \in W$ such that the ith bit of $\vec{w}$ is false and all bits $j < i$ are set to true. Then in the successor world v of w bit i is true and all bits $j < i$ are set to false; and the remaining bits $k > i$ are maintained. Hence $\vec{w} + 1 = \vec{v}$ encodes the next larger bit vector. This observation holds until there is no such bit i which is equal to 0 wherefore all bits hold. Hence there are 2^n many such situations and all are connected wherefore the theorem applies. $\Box$

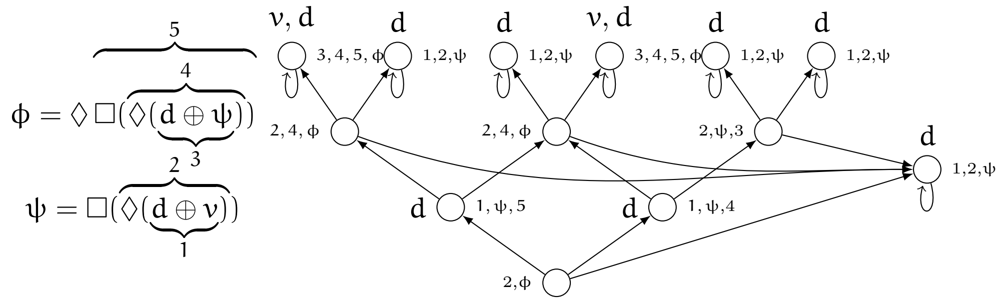


Figure 2.1.: An example of a run of Algorithm 2.1. Note that the formulas in the example use the function $\oplus$ which here is not expressed with $\wedge, \neg$ due to convenience. Propositions in the model are d, ν.

2.3. Model Checking and Ladner's Algorithm

Model checking deals with the question whether a given model satisfies a given formula. From a computer scientific aspect this is the question if a given computer program (the model) satisfies its specification (the formula). As we will see later this problem, the verification of the correctness of a given model w.r.t. a formula, in general, is not necessarily easier than the satisfiability problem itself. At first before we turn towards Ladner's tableaux algorithm we will visit the model checking algorithm for modal formulas. Formally the model checking problem for modal logic is defined as:

Problem (ML-MC)
Input: A model $\mathcal{M} = ((W, \mathbf{R}), V)$, $w_0 \in W$, a formula $\varphi \in$ ML.
Question: Does $\mathcal{M}, w_0 \models \varphi$ hold?

For modal logic the problem can be efficiently solved in polynomial time.

Theorem 2.3.
ML-MC $\in$ P.

Proof. The idea is that start to label subformulas of the given formula φ to worlds where they hold. We do this beginning at the smallest such subformulas until we reach the largest one which is φ itself. Then we only have to check if φ is labelled in the given world w_0.

Algorithm 2.1 runs in time $O(|\mathcal{M}|^2 \cdot |\varphi|)$. The algorithm always halts because the outer while loop eventually is not satisfied. The correctness follows from an induction on the formula size and the corresponding labels. Figure 2.1 shows how Algorithm 2.1 works for an example. $\square$

Now we turn towards the modal satisfiability problem. The following theorem is proven through a tableaux algorithm which was stated explicitly by Ladner in 1977. Ladner's algorithm and the correctness proof inherently grounds on Kripke's considerations and proofs [Kripke, 1963].

Algorithm 2.1: Procedure deciding ML-MC in polynomial time.

```
Input : Model $\mathcal{M} = ((W, R), V), w_0 \in W$, $\varphi \in \mathrm{ML}$
1  initially set the labeling function $\ell\colon W \to \mathbf{SF}(\varphi)$ to $\ell(w) = \emptyset$ for all $w \in W$;
2  forall the $\psi \in \mathbf{SF}(\varphi)$ ordered w.r.t. $|\psi|$ do
3      case $\psi = x \in \mathsf{PROP}$
4          forall the $w \in W$ s.t. $x \in V(w)$ do $\ell(w) \leftarrow \ell(w) \cup \{\psi\}$ ;
5      case $\psi = \alpha \vee \beta$
6          forall the $w \in W$ s.t. $\alpha \in \ell(w)$ or $\beta \in \ell(w)$ do $\ell(w) \leftarrow \ell(w) \cup \{\psi\}$ ;
7      case $\psi = \alpha \wedge \beta$
8          forall the $w \in W$ s.t. $\alpha \in \ell(w)$ and $\beta \in \ell(w)$ do $\ell(w) \leftarrow \ell(w) \cup \{\psi\}$ ;
9      case $\psi = \neg\alpha$
10         forall the $w \in W$ s.t. $\alpha \notin \ell(w)$ do $\ell(w) \leftarrow \ell(w) \cup \{\psi\}$ ;
11     case $\psi = \Diamond\alpha$
12         forall the $w \in W$ s.t. $\alpha \in \ell(w)$ do
13             forall the $w' \in W$ with $w'Rw$ do $\ell(w') \leftarrow \ell(w') \cup \{\psi\}$ ;
14     case $\psi = \Box\alpha$
15         forall the $w \in W$ do
16             if $\forall w' \in W$ with $wRw'$: $\alpha \in \ell(w')$ then $\ell(w) \leftarrow \ell(w) \cup \{\psi\}$ ;
17 return true iff $\varphi \in \ell(w_0)$
```

Theorem 2.4 ([Kripke, 1963, Ladner, 1977]).
ML-SAT $\in$ PSPACE.

Proof. The return value of the procedure `World`$(\mathcal{T}, \mathcal{F}, \mathcal{T}', \mathcal{F}')$ in Algorithm 2.2 is true iff there exists a model $\mathcal{M} = ((W, R), V)$ and a world $w \in W$ such that

$$\mathcal{M}, w \models \bigwedge_{\phi \in \mathcal{T}} \phi \wedge \bigwedge_{\phi \in \mathcal{F}} \neg\phi \wedge \bigwedge_{\phi \in \mathcal{T}'} \Box\phi \wedge \bigwedge_{\phi \in \mathcal{F}'} \Diamond\neg\phi.$$

Informally, the procedure returns true iff there is a world in which all $\mathcal{T}$-formulas are true, all $\mathcal{F}$-formulas are false, in each world accessible from w all $\mathcal{T}'$-formulas are true, and for each $\mathcal{F}'$-formulas ϕ there is a successor world from w falsifying ϕ.

Claim. *A given modal formula* $\varphi \in \mathrm{ML}$ *is in* ML-SAT *iff* `World`$(\{\phi\}, \emptyset, \emptyset, \emptyset) \equiv \top$.

Proof of Claim. Follows by an induction on the formula length. $\dashv$

Claim. `World` *requires polynomial space.*

Proof of Claim. The algorithm is implemented on a Turing machine by simulating a stack. Each subformula on the stack is represented by one of four different pointer types (one for each set). Let n denote the number of subformulas of φ. Further the storage at each level of the recursion is $O(n \cdot \log n)$. In the following we will show that the procedure `World` has a recursion depth of $O(n)$. For a finite set S of formulas define $|S| = \sum_{\phi \in S} |\phi|$.

We prove by induction on $n = |\mathcal{T}| + |\mathcal{F}| + |\mathcal{T}'| + |\mathcal{F}'|$ that $\texttt{World}(\mathcal{T}, \mathcal{F}, \mathcal{T}', \mathcal{F}')$ has recursion depth $\leqslant 2n + 1$. Denote the first recursive call of $\texttt{World}(\mathcal{T}, \mathcal{F}, \mathcal{T}', \mathcal{F}')$ with $(\mathcal{T}_1, \mathcal{F}_1, \mathcal{T}'_1, \mathcal{F}'_1)$ and let the claim hold for all $i < n$. If $\mathcal{T} \cup \mathcal{F} \not\subseteq \mathsf{PROP}$ then $|\mathcal{T}_1| + |\mathcal{F}_1| + |\mathcal{T}'_1| + |\mathcal{F}'_1| < n$ as in every case of (3-9) one subformula is removed. If $\mathcal{T} \cup \mathcal{F} \subseteq \mathsf{PROP}$ then line (11) or (12) is executed (possibly without removing a subformula), but can lead only to a recursive call where $\mathcal{F}'$ is empty. Thus every two steps of recursion reduces $|\mathcal{T}| + |\mathcal{F}| + |\mathcal{T}'| + |\mathcal{F}'|$ by at least 1. Therefore the procedure has a recursion depth $\leqslant 2|\varphi| + 1$. $\dashv$

Algorithm 2.2: Procedure `World`, conjunction over the empty set is defined to be $\top$

```
World($\mathcal{T}, \mathcal{F}, \mathcal{T}', \mathcal{F}'$):
1  if $\mathcal{T} \cup \mathcal{F} \not\subseteq \mathsf{PROP}$ then
2      choose $\psi \in (\mathcal{T} \cup \mathcal{F}) \setminus \mathsf{PROP}$;
3      if $\psi = \neg\chi$ and $\psi \in \mathcal{T}$ then return World($\mathcal{T}\setminus\{\psi\}, \mathcal{F} \cup \{\chi\}, \mathcal{T}', \mathcal{F}'$);
4      if $\psi = \neg\chi$ and $\psi \in \mathcal{F}$ then return World($\mathcal{T} \cup \{\chi\}, \mathcal{F}\setminus\{\psi\}, \mathcal{T}', \mathcal{F}'$);
5      if $\psi = \chi_1 \wedge \chi_2$ and $\psi \in \mathcal{T}$ then return World($(\mathcal{T} \cup \{\chi_1, \chi_2\})\setminus\{\psi\}, \mathcal{F}, \mathcal{T}', \mathcal{F}'$);
6      if $\psi = \chi_1 \wedge \chi_2$ and $\psi \in \mathcal{F}$ then
7          return World($\mathcal{T}, (\mathcal{F} \cup \{\chi_1\})\setminus\{\psi\}, \mathcal{T}', \mathcal{F}'$) $\vee$ World($\mathcal{T}, (\mathcal{F} \cup \{\chi_2\})\setminus\{\psi\}, \mathcal{T}', \mathcal{F}'$);
8      if $\psi = \Box\chi$ and $\psi \in \mathcal{T}$ then return World($\mathcal{T}\setminus\{\psi\}, \mathcal{F}, \mathcal{T}' \cup \{\chi\}, \mathcal{F}'$);
9      if $\psi = \Box\chi$ and $\psi \in \mathcal{F}$ then return World($\mathcal{T}, \mathcal{F}\setminus\{\psi\}, \mathcal{T}', \mathcal{F}' \cup \{\chi\}$);
10 if $\mathcal{T} \cup \mathcal{F} \subseteq \mathsf{PROP}$ then
11     if $\mathcal{T} \cap \mathcal{F} \neq \emptyset$ then return $\bot$;
12     if $\mathcal{T} \cap \mathcal{F} = \emptyset$ then return $\bigwedge_{\psi \in \mathcal{F}'}$World($\mathcal{T}', \{\psi\}, \emptyset, \emptyset$);
```

The following proof requires the notion of quantified Boolean formulas which are defined inductively as follows. It holds that the constants $\top, \bot \in$ QBF. If x is a variable, then $x \in$ QBF. If $\phi, \psi \in$ QBF then $\phi \wedge \psi, \phi \vee \psi, \exists x\phi, \forall x\phi \in$ QBF, where $\exists x\phi = \phi(x/\top) \vee \phi(x/\bot)$ and $\forall x\phi = \phi(x/\top) \wedge \phi(x/\bot)$. Given a formula $\varphi \in$ QBF we say φ is *closed* if all variables in φ are quantified. Then we define QBF-VAL as the set of all closed formulas $\varphi \in$ QBF such that $\varphi \equiv \top$ and φ is of the form $\varphi = Q_1 p_1 \cdots Q_n p_n \psi$ and ψ is a quantifier-free propositional 3CNF formula.

Theorem 2.5.
ML-SAT *is* PSPACE*-hard.*

Proof. We will state a reduction to show that QBF-VAL $\leqslant^{\mathrm{P}}_{\mathrm{m}}$ ML-SAT holds. In the following we use the macros defined in the proof of Theorem 2.1. Let $\phi = Q_1 p_1 \ldots Q_n p_n \psi$ be a QBF over variables $p_1, \ldots, p_n$. Further assume that ψ is quantifier-free and in 3CNF with m clauses. At first to express a binary tree we will use the formula φ_n from Theorem 2.1. Here each leaf corresponds to a possible assignment in $\{0, 1\}^n$ encoded via the propositions p_i. Secondly we need to bind the fulfilled clauses to one of its matching literals:

$$\phi_{\mathrm{clause}} := \bigwedge_{i=1}^{n} \Box^i \left(\bigwedge_{j=1}^{n} \left((p_j \to \bigwedge_{C \in \mathcal{C}(j)} C) \wedge (\neg p_j \to \bigwedge_{C \in \mathcal{C}'(j)} C) \right) \right),$$

where $\mathcal{C}(j) =_{\mathsf{def}} \{C \mid p_j \text{ is in clause } C\}$ and $\mathcal{C}'(j) =_{\mathsf{def}} \{C \mid \neg p_j \text{ is in clause } C\}$.

Now we need to make sure that the clause propositions are labelled iff a matching literal is labelled:

$$\phi_{\text{match}} := \bigwedge_{i=1}^{n} \Box^i \left(\bigwedge_{j=1}^{m} C_j \to (\ell_{j1} \vee \ell_{j2} \vee \ell_{j3}) \right),$$

where ℓ_{ji} corresponds to the ith literal in the jth clause.

Lastly we can state the complete reduction $f\colon$ QBF $\to$ ML defined as

$$Q_1 p_1 \dots Q_n p_n \psi \in \text{QBF-VAL} \quad \text{iff} \quad \varphi_n \wedge \phi_{\text{clause}} \wedge \phi_{\text{match}} \wedge \Delta_1 \dots \Delta_n \bigwedge_{i=1}^{m} C_i \in \text{ML-SAT},$$

where $\Delta_i = \Box$ if $Q_i = \forall$ and $\Delta_i = \Diamond$ if $Q_i = \exists$ holds for $1 \leqslant i \leqslant n$.

Let $\phi := Q_1 p_1 \cdots Q_n p_n \psi$ be the given QBF and $f(\phi)$ the constructed modal formula from above.

$\phi \in$ QBF-VAL $\Rightarrow f(\phi) \in$ ML-SAT. Now let Θ be one of the sets of assignments according to $Q_1 \cdots Q_n$ which all satisfy ψ. We describe how to construct a satisfying model M for $f(\phi)$. M will include the binary assignment tree of depth n such that in each leaf there exists a corresponding labeling of propositions p_i depending on the assignment. For each leaf if p holds then label all clause propositions C to this leaf where p is in clause C. In particular, this means that for every leaf which corresponds to a $\theta \in \Theta$ the propositions $C_1, \dots, C_m$ are added to the leaf. For each non-leaf world label the corresponding clauses to the world. Now this model satisfies

- φ_n as it is a correct encoded assignment tree,
- ϕ_{clause} because whenever a proposition p holds in a world w we added the respecting clause proposition to the world,
- ϕ_{match} because only if a proposition p holds in a world w then the corresponding clause propositions were added,
- $\Delta_1 \cdots \Delta_n \bigwedge_{i=1}^{m} C_i$ because the leafs corresponding to Θ are those which are defined by the Δ_i.

$\phi \notin$ QBF-VAL $\Rightarrow f(\phi) \notin$ ML-SAT. Further assume for contradiction that M (wrongly) satisfies $f(\phi)$. Hence M must include a complete binary assignment tree and the root of the tree must satisfy $f(\phi)$. Let L be the set of leafs in this tree corresponding to the Δ_i. Now for each $\ell \in L$ it must hold $M, \ell \models \bigwedge_{i=1}^{m} C_i$. Hence $\ell \in V(C_1) \cap \cdots \cap V(C_m)$ for every $\ell \in L$. Thus for every $\ell \in L$ and every clause at least one of its literals must be satisfied in ℓ. Consequently we can deduce that for each assignment corresponding to a leaf it satisfies ψ. But therefore $Q_1 p_1 \cdots Q_n p_n \psi$ is equal to $\top$ which is a contradiction.$\Box$

Corollary 2.6.
ML-SAT *is* PSPACE*-complete.*

Hence under the assumption that $\mathsf{P} \neq \mathsf{PSPACE}$ is true, the question for satisfiability of modal formulas is much more difficult than model checking them.

2.4. Generalized Satisfiability of Modal Logic

In this section we aim to prove a result for satisfiability in modal logic which is similar to Theorem 1.6. Therefore we will use Post's lattice as tool in order to determine the influence of Boolean functions on the complexity of the problem ML-SAT. The following theorem states the result and the proofs for all cases are split up into several subresults.

Theorem 2.7 ([Hemaspaandra et al., 2010]).
Let B *be a finite set of Boolean functions. Then the following hold:*

(1.) If $[\mathsf{B}] \subseteq \mathsf{R}_1, \mathsf{D}, \mathsf{V}$, *or* L, *then* $\text{ML-SAT}(\mathsf{B}) \in \mathsf{P}$.

(2.) If $\mathsf{E}_0 \subseteq [\mathsf{B}] \subseteq \mathsf{E}$, *then* $\text{ML-SAT}(\mathsf{B})$ *is* coNP*-complete.*

(3.) If $\mathsf{S}_{11} \subseteq [\mathsf{B}]$ *then* $\text{ML-SAT}(\mathsf{B})$ *is* PSPACE*-complete.*

(4.) Otherwise, $\mathsf{S}_1 \subseteq [\mathsf{B}]$ *and* $\text{ML-SAT}(\mathsf{B})$ *is* PSPACE*-complete.*

All completeness results are with respect to $\leqslant_{\mathbf{m}}^{\mathbf{P}}$ *reductions.*

Proof. The theorem is proven by several upcoming results. The results are visualized in Figure 2.4.

(1.) The first two cases are proven in Lemma 2.15. Case V is shown in Lemma 2.16. The affine cases are covered in Lemma 2.17.

(2.) Proven by Lemma 2.14 and Corollary 2.13.

(3.) Proven by Corollary 2.11 in combination with Lemma 2.9.

(4.) The lower bound is proven by Lemma 2.8. The upper bound for BF follows from a straightforward modification of the hard coded Boolean cases in Algorithm 2.2. □

Intractable cases

Lemma 2.8.
If B *is a finite set of Boolean functions s.t.* $\mathsf{S}_1 \subseteq [\mathsf{B}]$ *then* $\text{ML-SAT}(\{\land, \neg\}) \leqslant_{\mathbf{m}}^{\mathbf{P}} \text{ML-SAT}(\mathsf{B})$.

Proof. Let $\phi \in \text{ML}(\land, \neg)$. We show:

$$\text{ML-SAT}(\land, \neg) \leqslant_{\mathbf{m}}^{\mathbf{P}} \text{ML-SAT}(\mathsf{B} \cup \{\top\}) \leqslant_{\mathbf{m}}^{\mathbf{P}} \text{ML-SAT}(\mathsf{B}).$$

From Figure 1.1 it follows that $[\mathsf{B} \cup \{\top\}] = \mathsf{BF}$. By Lemma 1.5 we have short representations for $\land$ and $\neg$. Every occurrence of $\top$ in ϕ is replaced with a fresh variable t. Now add the conjunct $\land \bigwedge_{i=0}^{\text{md}(\phi)} \Box^i t$, where $\text{md}(\phi)$ denotes the modal depth of ϕ (nesting depth of modalities $\Box$ and $\Diamond$). Insert paranthesis in such a way we get a tree of $\land$'s of logarithmic depth and express the $\land$'s with their (possibly large) representation, which exists since $[\mathsf{B}] \supseteq \mathsf{S}_1 \supset \mathsf{E}_2 = [\land]$ with the result only increasing polynomially in size (due to the logarithmic nesting depth). □

Lemma 2.9.
If B *is a finite set of Boolean functions s.t.* $\mathsf{S}_{11} \subseteq [\mathsf{B}]$ *then* $\text{ML-SAT}(\{\wedge, \vee, \bot\}) \leqslant_{\mathbf{m}}^{\mathbf{P}} \text{ML-SAT}(\mathsf{B})$.

Proof. Observe that $[\mathsf{S}_{11} \cup \{\top\}] = \mathsf{M}$. Analogously to Lemma 2.8 as $\mathsf{M} \subseteq [\mathsf{B} \cup \{\top\}] \subseteq \mathsf{BF}$. $\square$

The following theorem deals with a restricted version of modal logic which was studied by Edith Hemaspaandra in 2005.

Theorem 2.10 (Poor Man's Satisfiability; Thm. 6.7 in [Hemaspaandra, 2005]). *Satisfiability for modal formulas without literals and only the constant false is* PSPACE*-complete.*

Proof. In 1995 Halpern has proven that the restriction of modal satisfiability with one propositional variable is PSPACE-complete [Halpern, 1995] and we will reduce further on to *zero* propositions.

Assume given ϕ consists of one proposition p, and operators in $\{\neg, \wedge, \Box\}$. *Proof idea:* Supposing ϕ is satisfiable implies that in each satisfying model $\mathcal{M} = ((W, R), V)$ every path has length $\leqslant \text{md}(\phi)$. Now we extend the model in such a way that this assignment is encoded into the frame. Define $\mathcal{M}' = ((W', R'), V')$ as follows:

$$\begin{aligned}
W' &= W \cup \{w_1, w_2, \ldots, w_{\text{md}(\phi)+1}\},\\
R' &= R \cup \{(w_i, w_{i+1}) \mid 1 \leqslant i \leqslant \text{md}(\phi)\} \cup \{(w, w_1) \mid w \in W \wedge \mathcal{M}, w \models p\},\\
V' &= \emptyset, \text{ as we do not have any propositions this function is irrelevant.}
\end{aligned}$$

Hence in $\mathcal{M}'$ all information of $\mathcal{M}$ is still contained plus there exists a maximal path of length $\text{md}(\phi) + 1$ from a world $w \in W$ in $\mathcal{M}'$ iff $\mathcal{M}, w \models p$. We will simulate p by the formula $\Diamond^{\text{md}(\phi)+1}\Box\bot$ (note that R does not need to be total).

Supposing $\mathcal{M}, w \models \Box\psi$ holds for an arbitrary $\Box\psi \in \mathbf{SF}(\phi)$ yields the requirement that we do not force ψ to be true on the new world w_1. Hence we will enforce that ψ holds in all successor world that do *not* have a maximal path of length $\text{md}(\phi)$. Let $f(\phi) = f_{\text{md}(\phi)}(\phi)$, and $f_k(\phi)$ be defined inductively as

$$\begin{aligned}
f_k(p) &= \Diamond^{k+1}\Box\bot,\\
f_k(\neg\psi) &= \neg f_k(\psi),\\
f_k(\psi \wedge \chi) &= f_k(\psi) \wedge f_k(\chi),\\
f_k(\Box\psi) &= \Box(\Diamond^k\Box\bot \vee f_k(\psi)).
\end{aligned}$$

Now it holds that for all $w \in W$, all formulas ψ consisting of the only proposition p s.t. $\text{md}(\psi) \leqslant \text{md}(\phi)$, $\mathcal{M}, w \models \psi$ if and only if $\mathcal{M}', w \models f_{\text{md}(\phi)}(\psi)$. If ϕ is satisfiable then $f(\phi)$ is satisfiable.

Now the opposite case. Suppose that $f(\phi)$ is satisfiable. Let $\mathcal{M}' = ((W', R'), V')$ be an acyclic model and $w_0 \in W'$ s.t. $M', w_0 \models f(\phi)$ holds. Define $\mathcal{M} = ((W, R), V)$ as follows:

$$\begin{aligned}
W &= (W' \backslash \{w \in W' \mid \mathcal{M}', w \models \Diamond^{\text{md}(\phi)}\Box\bot\}) \cup \{w_0\},\\
R &= R' \cap (W \times W),\\
V(p) &= \{w \in W \mid \mathcal{M}', w \models \Diamond^{\text{md}(\phi)+1}\Box\bot\}.
\end{aligned}$$

Again it holds that for all $w \in W$, all formulas ψ with the only proposition p s.t. $\mathrm{md}(\psi) \leqslant \mathrm{md}(\phi)$, $\mathcal{M}, w \models \psi$ iff $\mathcal{M}', w \models f_{\mathrm{md}(\phi)}(\psi)$. As f is clearly computable in polynomial time this proves the theorem. $\square$

Corollary 2.11.
ML-SAT($\{\wedge, \vee, \bot\}$) *is* PSPACE-*hard with respect to* $\leqslant^{\mathrm{p}}_{\mathrm{m}}$.

Proof. Theorem 2.10 talks about formulas with the operators $\wedge, \vee, \neg, \Box, \Diamond, \bot$. Now translate the produced formula $f(\phi)$ into NNF, then substitute constant $\top$ by fresh variable and proceed as in Lemma 2.8. $\square$

Theorem 2.12 (Poor Man's Conjunctive Satisfiability, [Donini et al., 1992]).
Unsatisfiability for modal formulas with only conjunction and constant $\bot$ *is* NP-*hard.*

Proof. We will state a reduction from the NP-complete problem

Problem (All-Pos-One-In-Three-3SAT)
Input: A formula $\phi = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{3} \ell_{i,j}$ and $\ell_{i,j} \in \{x_1, \dots, x_n\}$.
Question: Does there exist a *traversal*, i.e., an assignment $\theta \colon \{x_1, \dots, x_n\} \to \{0, 1\}$ s.t. for each $1 \leqslant i \leqslant m$ there is only one $1 \leqslant j \leqslant 3$ s.t. $\hat{\theta}(\ell_{i,j}) = 1$?

The idea is to associate with every ϕ a modal formula $f(\phi)$ such that ϕ has a traversal if and only if $f(\phi)$ is not satisfiable. Then the reduction $f(\phi)$ is defined as follows

$$f(\phi) = (\mathcal{X}_1)^2\top \wedge \cdots \wedge (\mathcal{X}_n)^2\top \wedge \Box^m\bot$$
$$\mathcal{X}_i = \Delta_i^1 \cdots \Delta_i^m, \quad \Delta_i^j = \begin{cases} \Diamond & \text{, if } x_i \in C_j \\ \Box & \text{, otherwise} \end{cases} \text{ for } 1 \leqslant i \leqslant n$$

The correctness proof for the reduction is omitted (for details see [Donini et al., 1992, pp. 315]). $\square$

Here we take a look at an example for the reduction instead.

Example. *(1.) Let* $\phi = (x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_4 \vee x_5)$. *The modal formula* $f(\phi)$ *looks as follows*

$$\Diamond\Box\Box\Diamond\Box\Box\top \wedge \Box\Diamond\Diamond\Box\Diamond\Diamond\top \wedge \Diamond\Diamond\Box\Diamond\Diamond\Box\top \wedge \Box\Diamond\Diamond\Box\Diamond\Diamond\top \wedge \Diamond\Box\Diamond\Diamond\Box\Diamond\top \wedge \Box^6\bot.$$

Obviously ϕ *has a traversal* $\theta(x_1) = \theta(x_2) = 1$ *and* $\theta(x_3) = \theta(x_4) = \theta(x_5) = 0$. *The only possible way to satisfy such a formula is to encode a "dead end" into the structure in order to satisfy eventually the last conjunct. The first two conjuncts together make this impossible to build such kind of structures, wherefore* $f(\phi)$ *is not satisfiable.*

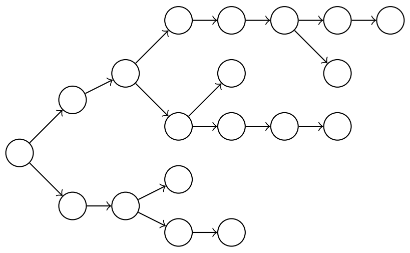


Figure 2.2.: Satisfying model for example (2) for Theorem 2.12.

(2.) Let $\phi = (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_4 \vee x_5) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4)$*. Observe that there exist no traversal for* ϕ*. The resulting formula* $f(\phi)$ *is*

$$\begin{aligned}&\Diamond\Box\Diamond\Box\Diamond\Box\Diamond\Box\top \wedge \Diamond\Diamond\Box\Diamond\Diamond\Diamond\Box\Diamond\top\wedge\\ &\Box\Box\Diamond\Diamond\Box\Box\Diamond\Diamond\top \wedge \Diamond\Diamond\Box\Diamond\Diamond\Diamond\Box\Diamond\top\wedge\\ &\Box\Diamond\Diamond\Box\Box\Diamond\Diamond\Box\top \wedge \Box^8\bot,\end{aligned}$$

and it is satisfied in the model depicted in Figure 2.2.

Corollary 2.13.
Let B *be a finite set of Boolean function s.t.* $E_0 \subseteq [B]$*. Then* ML-SAT(B) *is* coNP*-hard w.r.t.* $\leqslant^{\mathbf{P}}_{\mathbf{m}}$.

Proof. We reduce from the complement of ML-SAT$(\{\wedge, \bot\})$ which is by Theorem 2.12 NP-hard via

$$\begin{aligned}\varphi \notin \text{ML-SAT}(\{\wedge,\bot\}) \text{ iff } &\varphi' \notin \text{ML-SAT}(\overbrace{\{\wedge,\bot\}\cup\{\top\}}^{=E})\\ \text{iff } &\varphi'|_{[\top/t]} \wedge \bigwedge_{i=0}^{md(\varphi)} \Box^i t \notin \text{ML-SAT}(B).\end{aligned}$$

Here φ' uses short representations of $\wedge$ by Lemma 1.5 as we have a basis over E. The large conjunction of the formula $\Box^i t$ is treated in the same way as in the proof of Lemma 2.8. □

Lemma 2.14.
Let B *be a finite set of Boolean function s.t.* $[B] \subseteq E$*. Then* ML-SAT(B) *is in* coNP.

Proof. We show the upper bound just for $\{\wedge, \top, \bot\}$. A generalization to E is possible [Hemaspaandra et al., 2010, Lemma 3.12] but uses an approach involving modal circuits.

A given formula ϕ of this type can be assumed to be of the form

$$\phi = \psi \wedge \bigwedge_{i\in I} \Box\phi_i^{\Box} \wedge \bigwedge_{j\in J} \Diamond\phi_j^{\Diamond},$$

Algorithm 2.3: Algorithm deciding ML-SAT(E).

```
check($\phi \in \mathrm{ML}(\mathsf{E})$):
restructure $\phi$ to be of the form $\psi \wedge \bigwedge_{i\in I} \Box\phi_i^\Box \wedge \bigwedge_{j\in J} \Diamond\phi_j^\Diamond$
check if the all true assignment satisfies $\psi$
if $\theta \models \psi$ then
    for $j \in J$ do
        if not check$\left(\bigwedge_{i\in I} \phi_i^\Box \wedge \phi_j^\Diamond\right)$ then return false
return true
```

where I, J are finite sets of indices, $\phi_i^\Box, \phi_j^\Diamond \in \mathrm{ML}(B)$ for $i \in I, j \in J$ and $\psi \in \mathrm{PL}(B)$. Then $\phi \in \text{ML-SAT}(B)$ iff $\psi \in \mathrm{SAT}(B)$ and for all $j \in J$ it holds that $\bigwedge_{i\in I} \phi_i^\Box \wedge \phi_j^\Diamond \in \text{ML-SAT}(B)$. This immediately leads to a recursive coNP-algorithm:

Step one can be done in linear time. Step two is due to monotonicity and in step 3 we do a universal nondeterministic branching. □

Tractable cases

Lemma 2.15.
Let B *be a finite set of Boolean functions s.t.* $[\mathsf{B}] \subseteq \mathsf{R}_1$ *or* $[\mathsf{B}] \subseteq \mathsf{D}$ *hold. Then every formula in* $\mathrm{ML}(B)$ *is satisfiable.*

Proof. Every function in R_1 is $\top$-reproducing and therefore it holds that for every $\phi \in \mathrm{ML}(B)$ is satisfied via $\mathcal{M} = ((\{w\}, \{(w,w)\}), \{(p, \{w\}) \mid p \in \mathsf{PROP}\})$.

For every function in $f \in \mathsf{D}$ it holds that f is self-dual implying that either the all $\top$- or the all $\bot$-assignment satisfies f. Hence a model similar to $\mathcal{M}$ can be constructed. □

Lemma 2.16.
Let B *be a finite set of Boolean functions s.t.* $[\mathsf{B}] \subseteq \mathsf{V}$ *holds. Then* $\text{ML-SAT}(B)$ *is in* P.

Proof. A *modal circuit* is a usual circuit extended by gates for $\Box$ and $\Diamond$. Translate given $\varphi \in \mathrm{ML}(B)$ into a modal circuit C. By Theorem 1.3 we can translate C into a circuit C' over $\{\vee, \top, \bot\}$ without exponential blowups. This circuit C' can be seen as a directed graph. A gate which is a variable, a $\Box$, or the constant $\top$ is reachable from the output gate of C' if and only if φ is satisfiable. This property can be decided in polynomial time by plain graph search. □

Lemma 2.17.
Let B *be a finite set of Boolean functions s.t.* $[\mathsf{B}] \subseteq \mathsf{L}$ *holds. Then* $\text{ML-SAT}(B)$ *is in* P.

Proof. We present a polynomial time algorithm which uses the syntactic structure of these formulas by means of modal circuit representation to decide their satisfiability. Observe that $\phi \oplus \phi \equiv \bot$ holds for any formula ϕ. Utilizing this property the algorithm determines the equivalence of two modal circuits, since $\phi(B)$ is satisfiable iff ϕ is not equivalent to $\bot$. Theorem 1.3 again allows us to focus only on $\{\Diamond, \bot, \top, \oplus\}$ as $\Box\phi \equiv$

$\Diamond(\phi \oplus \top) \oplus \top$ holds. Further for a set Φ of circuits we write $\bigoplus \Phi$ to denote $\bigoplus_{\phi \in \Phi} \phi$, and define $\bigoplus \emptyset = \bot$. The algorithm accepts its input ϕ_1, ϕ_2 iff $\phi_1 \equiv \phi_2$ iff $\phi_1 \oplus \phi_2 \equiv \bot$. The parameter tuple for the first algorithmic call is then $(\phi, \bot)$.

(1.) Write $\phi_1 \oplus \phi_2$ as $\bigoplus \Phi$ where ϕ is a set of sub-circuits of $\phi_1 \oplus \phi_2$ s.t. for every $\phi \in \Phi$ it holds that $\phi \neq \alpha \oplus \beta$.

(2.) For every $\Diamond\psi \in \Phi$ s.t. $\psi \equiv \bot$, remove $\Diamond\psi$ from Φ.

(3.) For every distinct pair $(\Diamond\psi_1, \Diamond\psi_2) \in \Phi$ s.t. $\psi_1 \equiv \psi_2$ remove $\Diamond\psi_1, \Diamond\psi_2$ from Φ.

(4.) Accept iff Φ is propositional and $\bigoplus \Phi$ is *not* satisfiable.

Cases (2.)-(4.) can clearly be done in polynomial time. The correctness proof for case (1.) is omitted and can be found in [Hemaspaandra et al., 2010, Thm. 3.19]. □

Example. *Consider the formula from Figure 2.1. After substituting $\Box\phi$ with $\Diamond(\phi \oplus \top) \oplus \top$ we get*

$$\Diamond\Bigg(\Diamond\Bigg(\Diamond\bigg(\Big(d \oplus \Diamond(\Diamond(d \oplus v) \oplus \top)\Big) \oplus \top\bigg) \oplus \top\Bigg) \oplus \top\Bigg).$$

Now we start with the procedure call

$$\textit{\textbf{equivCheck}}\Bigg(\Diamond\Bigg(\Diamond\Bigg(\Diamond\bigg(\Big(d \oplus \Diamond(\Diamond(d \oplus v) \oplus \top)\Big) \oplus \top\bigg) \oplus \top\Bigg) \oplus \top\Bigg), \bot\Bigg).$$

This leads to the computation depicted in Figure 2.3. The next example shows a very simple unsatisfiable formula. ***equivCheck***$(\Diamond(\bot), \bot)$ *leads in line 2 to the call* ***equivCheck***$(\bot, \bot)$ *which accepts, then $\Diamond\bot$ is removed. The set Φ contains only $\bot$ which is not satisfiable; so accept is returned.*

rec. call/line	result
0/1	$\Phi = \left\{\Diamond\left(\Diamond\left(\Diamond\left(\left(\mathsf{d} \oplus \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top)\right) \oplus \top\right) \oplus \top\right) \oplus \top\right), \bot\right\}$
0/2	$\texttt{equivCheck}\left(\Diamond\left(\Diamond\left(\left(\mathsf{d} \oplus \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top)\right) \oplus \top\right) \oplus \top\right) \oplus \top, \bot\right)$
1/1	$\Phi = \left\{\Diamond\left(\Diamond\left(\left(\mathsf{d} \oplus \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top)\right) \oplus \top\right) \oplus \top\right), \top, \bot\right\}$
1/2	$\texttt{equivCheck}\left(\Diamond\left(\left(\mathsf{d} \oplus \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top)\right) \oplus \top\right) \oplus \top, \bot\right)$
2/1	$\Phi = \left\{\Diamond\left(\left(\mathsf{d} \oplus \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top)\right) \oplus \top\right), \top, \bot\right\}$
2/2	$\texttt{equivCheck}\left(\left(\mathsf{d} \oplus \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top)\right) \oplus \top, \bot\right)$
3/1	$\Phi = \{\mathsf{d}, \Diamond(\Diamond(\mathsf{d} \oplus \nu) \oplus \top), \top, \bot\}$
3/2	$\texttt{equivCheck}\,(\Diamond(\mathsf{d} \oplus \nu) \oplus \top, \bot)$
4/1	$\Phi = \{\Diamond(\mathsf{d} \oplus \nu), \top, \bot\}$
4/2	$\texttt{equivCheck}\,(\mathsf{d} \oplus \nu, \bot)$
5/1	$\Phi = \{\mathsf{d}, \nu, \bot\}$
5/4	$\bigoplus\Phi$ is propositional and satisfiable $\rightarrow$ return reject to 4/2
4/2	becomes return value reject $\rightarrow$ skip then-part; remove nothing
4/4	$\bigoplus\Phi$ is not propositional $\rightarrow$ return reject to 3/2
3/2	becomes return value reject $\rightarrow$ skip then-part; remove nothing
3/4	$\bigoplus\Phi$ is not propositional $\rightarrow$ return reject to 2/2
2/2	becomes return value reject $\rightarrow$ skip then-part; remove nothing
2/4	$\bigoplus\Phi$ is not propositional $\rightarrow$ return reject to 1/2
1/2	becomes return value reject $\rightarrow$ skip then-part; remove nothing
1/4	$\bigoplus\Phi$ is not propositional $\rightarrow$ return reject to 0/2
0/2	becomes return value reject $\rightarrow$ skip then-part; remove nothing
0/4	$\bigoplus\Phi$ is not propositional $\rightarrow$ return reject and halt.

Figure 2.3.: The row "rec. call/line" denotes in which recursion depth we are and what line of the respective recursive call is executed.

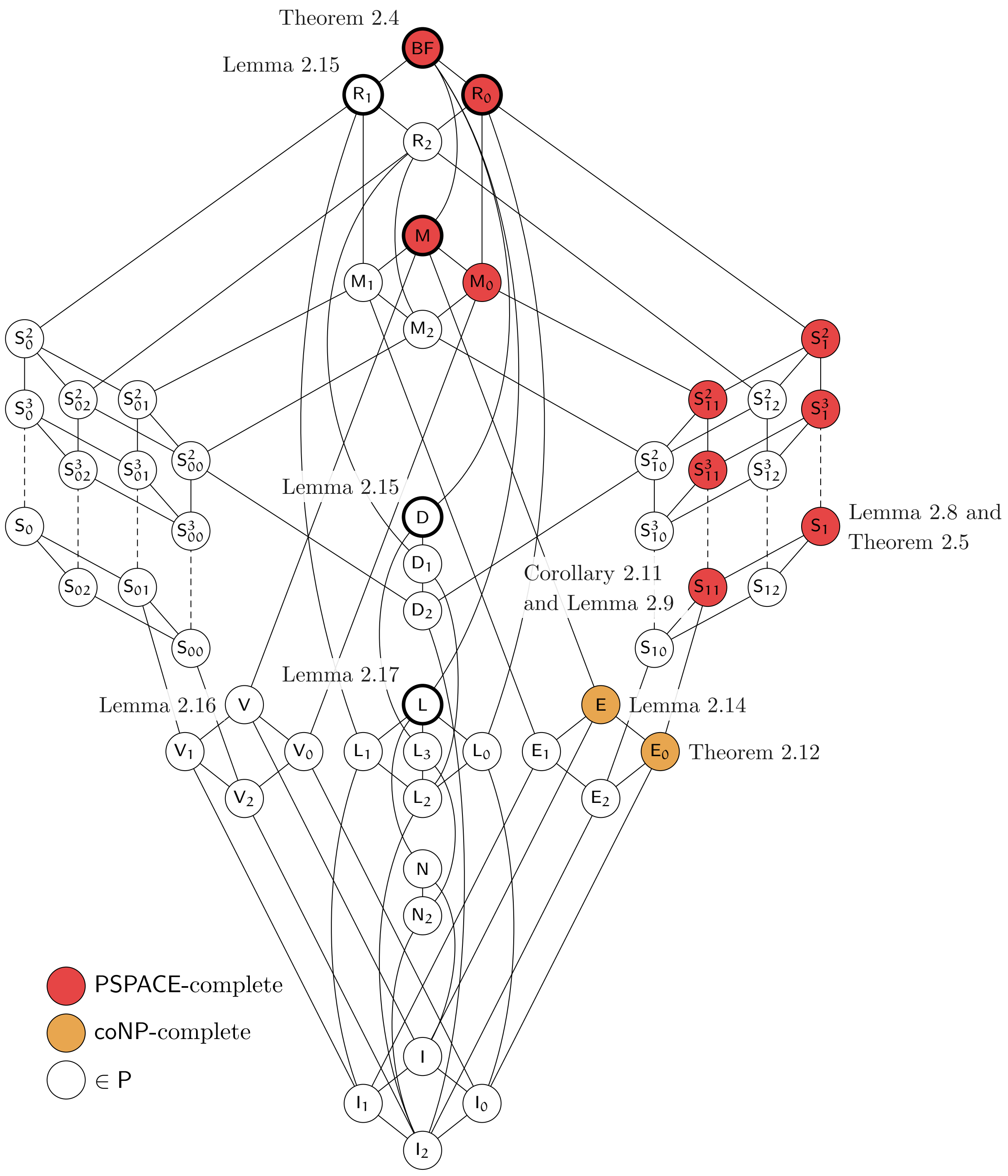


Figure 2.4.: Complexity classification of ML-SAT(B) w.r.t. Post's lattice denoting the corresponding result.

3. Variants of Modal Logic

Philosophy, including logic, is not primarily about language, but about the real world.

(Artur Norman Prior)

After the systematic study on the influence of Boolean operators to modal satisfiability we now turn towards several different *variants* of modal logic and their most prominent decision problems. Usually this is, of course, the satisfiability problem, but for some variants, e.g., default logic, it will be different. The section about nonmonotonic logics though investigates a different kind of deducibility which varies from common sense reasoning how we know it.

Again Post's lattice will be the recurrent theme through this chapter, but this time we will take the focus on interesting fragments that are of special interest due to either their complexity degree or the occurring uncommon concepts.

3.1. Hybrid Logic

Hybrid logics are a very prominent extension of modal logic. They introduce not only the ability to speak about single points of time (nominals) but also to name such points (binder operator $\downarrow$) and refer back to them (@ operator). We will see how powerful these extensions are and how they interact in a way to be able to express undecidable problems within the satisfiability problem of this logic. In the following we will introduce the syntax and semantics of this modal variant which follows the notion of [Areces et al., 2000].

Let NOM be a countable set of *nominals*, SVAR be a countable set of *variables* and $\mathsf{ATOM} = \mathsf{PROP} \cup \mathsf{NOM} \cup \mathsf{SVAR}$. We will stick with the common practice to denote atomic propositions by $p, q, \ldots$, nominals by $i, j, \ldots$, and variables by $x, y, \ldots$. We define the language of *hybrid (modal) logic* HL(B) for a finite set of Boolean functions as the set of well-formed formulas of the form

$$\varphi ::= a \mid f(\varphi, \ldots, \varphi) \mid \Diamond\varphi \mid \Box\varphi \mid \downarrow x.\varphi \mid @_t\varphi$$

where $a \in \mathsf{ATOM}$, $f \in B$ is a Boolean function, $x \in \mathsf{SVAR}$ and $t \in \mathsf{NOM} \cup \mathsf{SVAR}$.

The formulas of HL are interpreted on *(hybrid) models* $\mathcal{M} = ((W, R), V)$, consisting of a set of *states* W, a *transition relation* $R\colon W \times W$, and a *labeling function* $V\colon \mathsf{PROP} \cup \mathsf{NOM} \to \mathfrak{P}(W)$ that maps PROP and NOM to subsets of W such that $|V(i)| = 1$ for all $i \in \mathsf{NOM}$. In order to evaluate $\downarrow$-formulas, an assignment $g\colon \mathsf{SVAR} \to W$ is necessary. Given an assignment g, a state variable x and a state w, an x-*variant* g^x_w *of* g is defined by $g^x_w(x) = w$ and $g^x_w(x') = g(x')$ for all $x \neq x'$. For any $a \in \mathsf{ATOM}$, let $[V, g](a) = \{g(a)\}$ if $a \in \mathsf{SVAR}$ and $[V, g](a) = V(a)$, otherwise.

Definition (Semantics of Hybrid Logic). *Let* $\mathcal{M} = ((W, R), V)$ *be a hybrid model,* $f \in B$ *for a finite set of Boolean functions* B*, and* g *be an assignment on* K*. Further let* $\varphi_i \in \mathrm{HL}(B)$ *for* $1 \leqslant i \leqslant n$*.*

$$\begin{array}{ll}
\mathcal{M}, g, w \models a & \textit{iff } w \in [V, g](a),\ a \in \mathsf{ATOM}, \\
\mathcal{M}, g, w \models f(\varphi_1, \dots, \varphi_n) & \textit{iff } f([\![\mathcal{M}, g, w, \models \varphi_1]\!], \dots, [\![\mathcal{M}, g, w \models \varphi_n]\!]) = \top \\
\mathcal{M}, g, w \models \Diamond\varphi & \textit{iff } \mathcal{M}, g, w' \models \varphi \textit{ for some } w' \in W \textit{ with } wRw', \\
\mathcal{M}, g, w \models \Box\varphi & \textit{iff } \mathcal{M}, g, w' \models \varphi \textit{ for all } w' \in W \textit{ with } wRw', \\
\mathcal{M}, g, w \models @_t\varphi & \textit{iff } \mathcal{M}, g, w' \models \varphi \textit{ for } w' \in W \textit{ such that } w' \in [V, g](t), \\
\mathcal{M}, g, w \models \downarrow x.\varphi & \textit{iff } \mathcal{M}, g^x_w, w \models \varphi.
\end{array}$$

A hybrid formula φ is said to be *satisfiable* if there exists a Kripke structure $K = ((W, R), V)$, a $w \in W$ and an assignment $g\colon \mathsf{SVAR} \to W$ such that $K, g, w \models \varphi$. We say ϕ is $\mathcal{F}$-satisfiable for some frame class $\mathcal{F}$ if there exists a satisfying Kripke structure of the frame class $\mathcal{F}$.

The *at* operator $@_t$ shifts evaluation to the state named by $t \in \mathsf{NOM} \cup \mathsf{SVAR}$. The *downarrow binder* $\downarrow x.$ binds the state variable x to the current state. The symbols $@_x$, $\downarrow x.$ are called *hybrid operators.*

In order to consider fragments of hybrid logics, we define subsets of the language HL as follows. Let B be a finite set of Boolean functions and let O be a set of hybrid and modal operators. We define $\mathrm{HL}(O, B)$ to denote the set of well-formed hybrid formulas using the operators in O and the Boolean connectives in B only.

We define the *satisfiability problems* for the fragments of HL over frame classes defined above as:

Problem ($\mathcal{F}$-HL-SAT(O, B))
Input: an $\mathrm{HL}(O, B)$-formula φ.
Question: is φ $\mathcal{F}$-satisfiable?

How to Tile an Infinite Plane.

Proving undecidability usually involves stating a reduction from such a problem, e.g., the halting problem. In 1966 Robert Berger has proven that the TILING problem is coRE-complete [Berger, 1966]. This problem was firstly introduced by Berger's doctoral advisor Hao Wang [Wang, 1961].

This Domino-like game uses quadratic tokens consisting of at most four different colors. Now these tokens need to be arranged in a way such that the colors of adjacent tokens are the same and tile an infinite plane.

Problem (TILING)
Input: a finite set of Wang tiles $T = \{t_1, \dots, t_n\}$ over the colors $C = \{c_1, \dots, c_m\}$.
Question: can T tile an infinite plane?

Theorem 3.1.
D-HL-SAT$(\{\Diamond, \downarrow, @\}, B)$ *is undecidable (*coRE*-hard) for* $S_1 \subseteq [B]$*.*

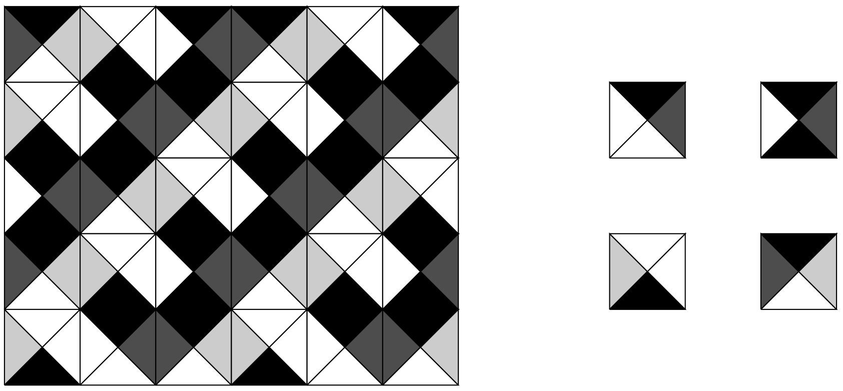


Figure 3.1.: Four Wang tiles and an extract of a tessellation with them. If one substitutes the two occurrences of the tile from the upper left in the bottommost line with the tile from the upper right, then one can tile an infinite plane.

Proof. First note that $S_1 \subseteq [B]$ implies $[B \cup \{\top\}] = BF$. As $\top \equiv \downarrow x.x$ holds, we may hence assume that the connectives $\wedge, \vee$ and $\neg$ are available.

We give a $\leqslant^{\mathbf{P}}_{\mathbf{m}}$-reduction from the undecidable TILING problem. The formula $\varphi := \varphi_s \wedge \varphi_{succ} \wedge \varphi_{distinct} \wedge \varphi_{tiling}$ we will construct, is a conjunction over the following subformulas. The subformula φ_s states that any satisfying model embeds a state s and a set of states P (which will correspond to the plane) such that s is connected to each node in P and s is unreachable from P:

$$\varphi_s := s \wedge \Diamond(p_{00}) \wedge @_s\Box\neg\Diamond s \wedge \Box\Box(\downarrow x.@_s\Diamond x).$$

The next formula φ_{succ} forces every node in P labelled with p_{ij} to have two different successor nodes labelled with p_{i+1j} and p_{ij+1}:

$$\begin{aligned}\varphi_{succ} := \Box \bigwedge_{0\leqslant i,j\leqslant 2}\Big(\downarrow x.p_{ij} \to \Big(&\Diamond\big(p_{ij+1} \wedge \downarrow y.@_x\Box(p_{ij+1} \to y)\big)\wedge\\ &\Diamond\big(p_{i+1j} \wedge \downarrow y'.@_x\Box(p_{i+1j} \to y')\big)\wedge\\ &\Box\big(p_{i+1j} \vee p_{ij+1}\big)\Big)\Big),\end{aligned}$$

where i and j are counters modulo 3, i. e., $i+1 = 0$ and $j+1 = 0$ for $i, j = 2$. The formula $\varphi_{distinct}$ ensures that every path from p_{ij} to p_{i+1j+1} is unambiguous:

$$\begin{aligned}\varphi_{distinct} := \Box\downarrow x.\Big(\bigvee_{0\leqslant i,j\leqslant 2}\Big(p_{ij} \wedge \Diamond\Big(p_{i+1j} \wedge \Diamond\big(\downarrow y.p_{i+1j+1} \wedge @_x\Diamond(p_{ij+1} \wedge \Diamond y)\big)\Big)\Big)\Big)\wedge\\ \Box\downarrow x.\Big(\bigwedge_{0\leqslant i,j\leqslant 2}\Big(p_{ij} \to \Big(\Diamond\big(p_{i+1j} \wedge \downarrow y.@_s\Box(p_{ij} \wedge \Diamond y) \to x\big)\wedge\\ \Diamond\big(p_{ij+1} \wedge \downarrow y.@_s\Box(p_{ij} \wedge \Diamond y) \to x\big)\Big)\Big)\Big).\end{aligned}$$

Finally we need to encode the tiling property into the grid. Consider sets of propositions $\mathsf{C}^{\mathsf{d}} = \{\mathsf{c}^{\mathsf{d}} \mid \mathsf{c} \in \mathsf{C}\}$, $\mathsf{d} \in \{\mathrm{t}, \mathrm{r}, \mathrm{b}, \mathrm{l}\}$ ($\{\mathrm{t}, \mathrm{r}, \mathrm{b}, \mathrm{l}\}$ abbreviate top, right, bottom, left). We indentify each tile $\mathsf{t} \in \mathsf{T}$ with its four colors and their resp. positions $(\mathsf{c}^{\mathrm{t}}_{\mathsf{t}}, \mathsf{c}^{\mathrm{r}}_{\mathsf{t}}, \mathsf{c}^{\mathrm{b}}_{\mathsf{t}}, \mathsf{c}^{\mathrm{l}}_{\mathsf{t}}) \in \mathsf{C}^{\mathrm{t}} \times \mathsf{C}^{\mathrm{r}} \times \mathsf{C}^{\mathrm{b}} \times \mathsf{C}^{\mathrm{l}}$. Hence, $\mathsf{c}^{\mathsf{d}}_{\mathsf{t}}$ is a placeholder for color $\mathsf{c} \in \mathsf{C}$ in position $\mathsf{d} \in \{\mathrm{t}, \mathrm{r}, \mathrm{b}, \mathrm{l}\}$ on tile $\mathsf{t} \in \mathsf{T}$ and $\mathsf{c}^{\mathsf{d}} \in \mathsf{C}^{\mathsf{d}}$ is a placeholder for color $\mathsf{c} \in \mathsf{C}$ in position $\mathsf{d} \in \{\mathrm{t}, \mathrm{r}, \mathrm{b}, \mathrm{l}\}$. We define $\varphi_{\mathit{tiling}}$ as

$$\begin{aligned}\varphi_{\mathit{tiling}} := {} & \Box\Bigg(\bigvee_{0 \leqslant i,j \leqslant 2}\bigg(p_{ij} \wedge \bigwedge_{c \in C}\Big(\big(c^{r} \rightarrow \Diamond(p_{ij+1} \wedge c^{l})\big) \wedge \big(c^{b} \rightarrow \Diamond(p_{i+1j} \wedge c^{t})\big)\Big)\bigg)\Bigg) \wedge \\ & \Box\Bigg(\bigwedge_{t \in T}\bigg(t \rightarrow c^{t}_{t} \wedge c^{r}_{t} \wedge c^{b}_{t} \wedge c^{l}_{t} \wedge \bigwedge_{t' \in T \setminus \{t\}} \neg t'\bigg) \wedge \bigwedge_{\substack{c \in C,\\ d \in \{t,r,b,l\}}}\bigg(c^{d} \rightarrow \bigwedge_{c' \in C \setminus \{c\}} \neg c'^{d}\bigg)\Bigg) \wedge \\ & \Box \bigvee_{t \in T} t\end{aligned}$$

If there is a valid tiling of an infinite plane then this tiling can be transferred into a satisfying model for φ which does satisfy each of the subformulas from above, and if there exists a satisfying model for φ then this states in which order the tiles are arranged implying a valid tiling.

Thus it holds that $\langle t_1, \ldots, t_n \rangle \in$ TILING if and only if there is an infinite model $\mathcal{M} = ((W, R), V)$ and a $w \in W$ and an assignment $g \colon \mathsf{SVAR} \rightarrow W$ such that $\mathcal{M}, g, w \models \varphi$.$\Box$

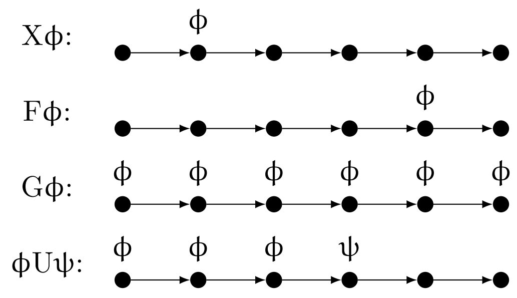
3.2. Temporal Logic

Enriching modal logic with concepts to interact more densely with computer programs leads to the field of temporal logics which have been introduced by A. N. Prior in 1957 who has been called "the founding father of temporal logic" by the Danish Centre for Philosophy and Science Studies. From 1971 to 1986 significant effort by Pnueli, Emerson, Halpern, and Clarke resulted in the definition of the linear time logic LTL and the computation tree logics CTL* and CTL. These logics have been invented to be of great benefit in the process of software engineering for verifying non-terminating programs. Describing specifications through formulae results in an evaluation of the written programs which are modeled by the Kripke structures as explained above. In the course of time, temporal logics emerged as being useful with major relevance for practical experience. In this context the *model checking problem* and the *satisfiability problem* of these logics are of great interest.

For a finite set of Boolean function, the syntax of the *full branching time logic* CTL*(B) is inductively defined as follows

$$\varphi ::= p \mid \psi \mid f(\varphi, \ldots, \varphi) \mid \mathrm{A}\varphi \mid \mathrm{E}\varphi \mid \mathrm{X}\varphi \mid \mathrm{F}\varphi \mid \mathrm{G}\varphi \mid \varphi\mathrm{U}\varphi,$$

where $p \in \mathsf{PROP}, \psi \in \mathrm{PL}(B), f \in B$. A, E are called *path operators*, and X, F, G, U are called temporal operators. For a *path formula* (*state formula*) $\phi \in \mathrm{CTL}^\star$ in holds that ϕ starts with a temporal (path) operator.



$X\phi$:
ϕ
$F\phi$:
ϕ
$G\phi$:
ϕ ϕ ϕ ϕ ϕ ϕ
$\phi U\psi$:
ϕ ϕ ϕ ψ

Figure 3.2.: Semantics of temporal operators.

The *computation tree logic* CTL is the restriction of CTL* to formulae where directly before each temporal operator there is a path operator and after each path operator there is a temporal operator. In general, temporal logic is defined as a unimodal logic wherefore corresponding models consist of only one transition function. Furthermore for any model $\mathcal{M} = ((W, R), V)$ in the temporal world it must hold that for any $w \in W$ there exists a $w' \in W$ with wRw'.

Definition (Semantics of Temporal Logic). *Let $\varphi_i, \chi_i \in$ CTL* for state formulae φ_i, path formulae χ_i, $1 \leqslant i \leqslant n$. Further let $\mathcal{M} = ((W, R), V)$ be a model and x be an infinite path $x = (x_1, x_2, \dots)$ with x_iRx_{i+1} and $x_i, s \in W$ for $i \in \mathbb{N}$.*

$\mathcal{M}, s \models \top$	*always holds,*	
$\mathcal{M}, s \models \bot$	*never holds,*	
$\mathcal{M}, s \models p$	*iff*	$p \in \mathsf{PROP}$ *and* $p \in l(s)$,
$\mathcal{M}, s \models f(\varphi_1, \dots, \varphi_n)$	*iff*	$f([\![\mathcal{M}, s \models \varphi_1]\!], \dots, [\![\mathcal{M}, s \models \varphi_n]\!])$ *is true,*
$\mathcal{M}, s \models A\chi_1$	*iff*	*for all paths* $x = (s, x_2, x_3, \dots)$ *it holds that* $M, x \models \chi_1$,
$\mathcal{M}, s \models E\chi_1$	*iff*	*there exists a path* $x = (s, x_2, x_3, \dots)$ *such that* $M, x \models \chi_1$ *holds,*
$\mathcal{M}, x \models \varphi_1$	*iff*	$M, x_1 \models \varphi_1$,
$\mathcal{M}, x \models f(\chi_1, \dots, \chi_n)$	*iff*	$f([\![\mathcal{M}, x \models \chi_1]\!], \dots, [\![\mathcal{M}, x \models \chi_n]\!])$ *is true,*
$\mathcal{M}, x \models X\chi_1$	*iff*	$\mathcal{M}, x_2 \models \chi_1$,
$\mathcal{M}, x \models F\chi_1$	*iff*	*there exists an* $i \geqslant 1$ *such that* $\mathcal{M}, x_i \models \chi_1$ *holds,*
$\mathcal{M}, x \models G\chi_1$	*iff*	*for all* $i \geqslant 1$ *it holds that* $\mathcal{M}, x_i \models \chi_1$,
$\mathcal{M}, x \models [\chi_1 U \chi_2]$	*iff*	$\mathcal{M}, x_k \models \chi_2$ *for some* $k \in \mathbb{N}$, *and* $\mathcal{M}, x_i \models \chi_1$ *for all* $1 \leqslant i < k$,

where the expression $[\![\mathcal{M}, s \models \phi]\!]$ is equal to $\top$ if $\mathcal{M}, s \models \phi$ holds and equal to $\bot$ otherwise.

In Figure 3.2 each temporal operator is shown in an example. Let B be a finite set of Boolean functions and T be a set of temporal and/or path operators. Then for a formula $\phi \in$ CTL$^\star(T, B)$ it holds that $\phi \in$ CTL$^\star(B)$ and ϕ contains only operators in T. The formula ϕ is *satisfiable* iff there exists a model $\mathcal{M} = ((W, R), V)$ such that $\mathcal{M}, w \models \phi$ for some $w \in W$. Then we say that $\phi \in$ CTL*-SAT(T, B) holds. For the logic CTL we proceed similarly, then however the set T contains only pairs of path and temporal operators.

Corollary 3.2.
Let B *be a finite set of Boolean functions. Then* $\text{CTL-SAT}(\{\text{AX}, \text{EX}\}, \mathsf{B}) \equiv_{\mathbf{m}}^{\mathbf{log}} \mathbf{KD}\text{-ML-SAT}(\mathsf{B})$ *holds.*

Logarithmic Time

Intuitively for decision problems one usually needs to read the input completely in order to determine its membership behavior. Of course, above the complexity class P this is possible, but if we move below P maintaining this property may be challenging. At first one needs to separate input and working tapes into two different types of tapes. After this one can think about more restrictions, e.g., what happens for languages where we do not need to necessarily read the complete input?

With respect to this a logarithmic time hierarchy can be established. We will define the complexity class $\mathsf{NLOGTIME}$ as the set of languages decided by a nondeterministic Turing machine M with a special *index tape* for giving 'random access' to the input word x, where everytime M enters the query state $\mathsf{q}_?$ for accessing bit i it is charged 1 time unit for this query and this query may only used once w.l.o.g. at the end of the computation—analogously $\mathsf{coNLOGTIME}$ is defined as the complement of languages in $\mathsf{NLOGTIME}$.

Another way to define the access to the input tape of a logtime machine is a *block read/write* approach: In this model, M can write two addresses $\mathsf{i} \leqslant \mathsf{j}$ on the index tape, and it then receives the string consisting of bit i to bit j of the input, at cost $\log|\mathsf{x}| + (\mathsf{j} - \mathsf{i})$.

Now as we are located very deep inside the class P (and also inside a very small circuit complexity class AC^0) we need to refine the current definition of reductions to a sufficient concept.

Definition ([Regan and Vollmer, 1997]). *Let* A *and* B *languages, and* f *be a many-one reduction from* A *to* B *s.t.* $|\mathsf{f}(\mathsf{x})|$ *is polynomial in* $|\mathsf{x}|$.

(1.) A *reduces to* B *via a* deterministic logtime reduction *if there is a DTM* M *that computes* f *in the following way: On input* x *and auxiliary input* $|\mathsf{x}|$ *and* j, M *outputs* $|\mathsf{f}(\mathsf{x})|$ *and the* j*-th bit of* $\mathsf{f}(\mathsf{x})$. *If additionally* M *only makes one query to the input, then the reduction is a* Ruzzo reduction.

(2.) A *reduces to* B *via a* DLT reduction *if there is a deterministic logtime TM* M *that works in block read/write mode and on input* x *and auxiliary input* $|\mathsf{x}|, \mathsf{i}, \mathsf{j}$ *with* $\mathsf{j} - \mathsf{i} = \mathsf{O}(\log|\mathsf{x}|)$, *outputs bits* i *to* j *of* $\mathsf{f}(\mathsf{x})$ *together with* $|\mathsf{f}(\mathsf{x})|$.

(3.) A *reduces to* B *via a* deterministic logtime projection, $\mathsf{A} \leqslant_{\mathbf{proj}}^{\mathbf{dlt}} \mathsf{B}$, *if* A *reduces to* B *via a reduction function* f *which is both a Ruzzo and a DLT reduction.*

Theorem 3.3 ([Schnoor, 2010]).
The language $\mathsf{L} = \{0,1\}^* 1 \{0,1\}^*$ *is* $\mathsf{NLOGTIME}$*-complete under* $\leqslant_{\mathbf{proj}}^{\mathbf{dlt}}$.

Proof. Of course $\mathsf{L} \in \mathsf{NLOGTIME}$ holds because we can simply guess the position of the 1. Hence let $\mathsf{L}_1 \in \mathsf{NLOGTIME}$ be some language over an alphabet Σ, and let M be a

TM which accepts L_1 in logarithmic time with the restriction that at most one bit of the input string is read. For L_1 there exists a function $g\colon \mathbb{N} \times \Sigma \to \{0,1\}$ which can be computed in linear deterministic time which has the property $g(i,c) = 1$ iff the machine M accepts if it nondeterministically chooses to read the i-th bit of the input, and this bit is the character c (since the length of i is logarithmic in $|x|$).

Now we state a reduction f from L_1 to L as follows: For $x \in \{0,1\}^*$, the length of $f(x)$ is the same as the length of x, and the i-th position of $f(x)$ is $g(i, x_i)$ where $x = x_1 \cdots x_n$.

Now f is a many-one reduction from L_1 to L. For this, note that x is a word from L_1 if and only if there is some number i s.t. if M reads the i-th bit of x, it accepts. By definition, this is equivalent to $g(i, x_i) = 1$ for some i, and by definition of f, this is equivalent to $f(x) \in L$. As the computation of f captures the idea of a local replacement one can easily see that f is a $\leqslant^{\mathbf{dlt}}_{\mathbf{proj}}$-reduction. □

Theorem 3.4.
Let B *be a set of Boolean functions and* $\{\mathrm{AU}, \mathrm{EU}\} \cap T = \emptyset$ *be a set of operators. If* $V_0 \subseteq [B] \subseteq V$ *or* $E_0 \subseteq [B] \subseteq E$*, then* CTL-SAT$(T, B)$ *is* TC^0*-complete w.r.t.* $\leqslant_{\mathbf{cd}}$.

Proof. Exercise □

The fragments of our interest will be shown to be complete for the introduced classes from above. Before we can turn towards the result we need to define some special kind of problem type which is called *promise problem.* Here the promise is made that the given formulas are syntactically correctly encoded and only use the allowed operators. This type of problems is usually denoted with a $\mathcal{P}$ in the subscript. One usually considers these kind of problems when checking the syntax is the barrier of the complexity of the problem.

Theorem 3.5.
Let B *be a set of Boolean functions and* $\{\mathrm{AU}, \mathrm{EU}\} \cap T = \emptyset$ *be a set of operators. Then*

(1.) CTL-SAT$_{\mathcal{P}}(T, B)$ *is* $\mathsf{NLOGTIME}$*-complete w.r.t.* $\leqslant^{\mathbf{dlt}}_{\mathbf{proj}}$ *if* $V_0 \subseteq [B] \subseteq V$ *holds, and*

(2.) CTL-SAT$_{\mathcal{P}}(T, B)$ *is* $\mathsf{coNLOGTIME}$*-complete w.r.t.* $\leqslant^{\mathbf{dlt}}_{\mathbf{proj}}$ *if* $E_1 \subseteq [B] \subseteq E$ *holds.*

Proof. At first we will prove the result for (1.) and then an analogous argumentation will justify the result for (2.). For the membership in $\mathsf{NLOGTIME}$ we substitute all propositions in a given formula φ with $\top$ (monotonicity). This substitution is a straight forward $\leqslant^{\mathbf{dlt}}_{\mathbf{proj}}$-reduction. Now we only need to guess a position in φ and verify that it contains the symbol $\top$ (for (2.) we only need to verify that no $\bot$ is present).

To prove hardness we will state a $\leqslant^{\mathbf{dlt}}_{\mathbf{proj}}$-reduction from the language $\{0,1\}^*1\{0,1\}^*$ which clearly is $\mathsf{NLOGTIME}$-complete. For any base satisfiying $[B] \subseteq V$ Lemma 1.5 implies the existence of a function $f \in [B]$ such that $f(x,y) \equiv x \vee y$ and x and y occur only once in f. Let f' denote f rewritten using $\{\vee, \bot, \top\}$ and assume for a contradiction that f contains the symbol $\top$. Then we can write f' as $f'(x,y) \equiv g_1(\top \vee g_2(x,y)) \equiv g_1(\top)$ for $g_1, g_2 \in [\{\vee, \bot, \top\}]$. As f is not degenerated (i.e., a constant function) and $g_1 \in [B] \subseteq M$, we now obtain that $g_1(\top) = \top$ and hence $f(x, y) \equiv \top$. A contradiction to the assumption of f containing $\top$; thus $f \in V_0$. Let $f \equiv f^B x f^M y f^E$. Since functions in [B] are commutative, we can achieve by swapping arguments, that f^E is empty. Now let $c_1 c_2 \dots c_n$ be a string of the form $\{0, 1\}^*$ and let $c_i' = \top$ iff $c_i = 1$ and $c_i = \bot$ otherwise. Then

$$c_1 c_2 \dots c_n \mapsto f^B c_1' f^M f^B c_2 f^M \cdots f^B c_{n-1}' f^M c_n',$$

is the desired $\leqslant^{\mathbf{dlt}}_{\mathbf{proj}}$-reduction: the mapping can obviously be calculated by a logtime projection and $f^B c_1' f^M f^B c_2 f^M \cdots f^B c_{n-1}' f^M c_n'$ holds iff at least one c_i is $\top$. Similarly hardness for (2.) is established using a reduction from the coNLOGTIME-hard language 0^*. $\square$

3.3. Nonmonotonic Logics

Usual reasoning consists of building implication chains through the deductive closure of $\models$. If one believes in ϕ and $\phi \models \psi$ then one believes in ψ as well. Thus knowing more implies a monotone growth of wisdom, i.e., if I *know more* then I can *deduce more.*

However human reasoning is not monotone: getting more information may allows one deducing fewer consequences. Especially when formal specifications are to be verified against real-world situations, one has to overcome the *qualification problem* that denotes the impossibility of listing all conditions required to decide compliance with the specification. To overcome this problem, McCarthy proposed the introduction of "common-sense" into formal logic [McCarthy, 1980]. Among the formalisms developed since then, Reiter's default logic is one of the best known and most successful formalisms for modeling common-sense reasoning.

Default logic extends propositional logic by patterns for default assumptions. These are of the form "in the absence of contrary information, assume ...". Reiter argued that his logic is an adequate formalization of human reasoning under the *closed world assumption.* In fact, default logic is used in various areas of artificial intelligence and computational logic, and is known to embed other nonmonotonic formalisms such as extended logic programs [Gelfond and Lifschitz, 1991].

Definition (Default Logic). *Let* B *be a finite set of Boolean functions, and let* $\alpha, \beta, \gamma \in \mathrm{PL}(B)$. *A* B-default rule *is an expression* $d = \frac{\alpha:\beta}{\gamma}$, *where* α *is the* prerequisite, β *is the* justification, *and* γ *is the* consequent *of* d. *A* B-default theory *is a pair* (W, D) *where* $W \subseteq \mathrm{PL}(B)$ *and* D *is a finite set of* B*-default rules.*

Now consider a B-default theory as the knowledge and deduction base of an agent. This agent can use the usual $\models$ relation to obtain new information entailed by his knowledge. Now this set of formulas generally is denoted with $\mathbf{Th}(W)$. The question arises if this obtainable knowledge is consistent or not. Here we need to define a suitable term which is the extension of a default theory.

Definition (Extension). *Let* B *be a finite set of Boolean functions,* (W, D) *be a* B*-default theory and* $E \subseteq \mathrm{PL}(B)$ *be a set of* B*-formulas.*

(1.) Let $E_0 := W$ *and* $E_{i+1} := \mathbf{Th}(E_i) \cup \left\{\gamma \;\middle|\; \frac{\alpha:\beta}{\gamma} \in D, \alpha \in E_i \wedge \neg\beta \notin E\right\}$. *Then* E *is a* stable extension of (W, D) *if and only if* $E = \bigcup_{i\in\mathbb{N}} E_i$.

(2.) Let $G := \left\{\frac{\alpha:\beta}{\gamma} \in D \mid \alpha \in E \wedge \neg\beta \notin E\right\}$. *If* E *is a stable extension of* (W, D) *then* $E = \mathbf{Th}\Big(W \cup \left\{\gamma \;\middle|\; \frac{\alpha:\beta}{\gamma} \in G\right\}\Big)$. *In this case,* G *is called the set of* generating defaults *of* E.

In the vein of satisfiability we now are able define decision problems which are settled around the availability of extensions.

Problem (EXT(B))
Input: A B-default theory (W, D).
Question: Does (W, D) has a stable extension?

Problem (CRED(B))
Input: A B-default theory (W, D), and a formula $\varphi \in \mathrm{PL}(B)$.
Question: Does there exist a stable extension E of (W, D) such that $E \models \varphi$ holds?

Problem (SKEP(B))
Input: A B-default theory (W, D), and a formula $\varphi \in \mathrm{PL}(B)$.
Question: Does for every stable extension E of (W, D) hold that $E \models \varphi$ is true?

Example ([Thomas, 2010]). *The default theory* $(\emptyset, D)$ *with* $D = \{\frac{\top:x}{\neg x}\}$ *has no stable extension. On the other hand,* $(\emptyset, D)$ *with* $D = \{\frac{\top:\neg x}{\neg x}, \frac{\top:x}{x}\}$ *has exactly two stable extensions, namely* $E' = \mathbf{Th}(x)$ *and* $E'' = \mathbf{Th}(\neg x)$*, corresponding to applications of respectively the first or second default in* D.

Next, consider (W, D) *with* $W = \{x\}, D = \{\frac{x:\neg y}{z}\}$. *Then* (W, D) *has a unique stable extension which contains* z. *However, if* W *is extended by the proposition* y*, then the unique stable extension of* $(W \cup \{y\}, D)$ *does no longer contain* z*; the newly added fact* y *makes the justification of* $\frac{x:\neg y}{z}$ *inconsistent with its stable extensions.*

Example (Playing Football with Default Logic). *Let* $W = \{\mathit{football}, \mathit{rain}, \mathit{cold} \wedge \mathit{rain} \to \mathit{snow}\}$ *and* $D = \left\{\frac{\mathit{football}:\neg \mathit{snow}}{\mathit{takesPlace}}\right\}$. *Then* $\neg\mathit{snow}$ *is consistent with* W. *Hence we can infer takesPlace. But if we consider* $(W \cup \{\mathit{cold}\}, D)$ *then snow gets consistent with* W*, wherefore we cannot infer takesPlace. This shows the non-monotonicity of default logic.*

Thus thinking about stable extensions implies answering implication questions of formulas. For the next theorem we investigate this implication problem for PL(L) formulas and show how this question is connected to solving equations over the field $\mathbb{Z}_2$.

Problem (IMP(B))
Input: A finite set $\Gamma \subseteq \mathrm{PL}(B)$ and $\phi \in \mathrm{PL}(B)$.
Question: Does $\Gamma \models \phi$ hold?

Lemma 3.6 ([Thomas, 2010]).
Let B *be a finite set of Boolean functions such that* $[\mathsf{B}] \subseteq \mathsf{L}$*. Then the problem* $\mathrm{IMP}(\mathsf{B})$ *in* $\oplus\mathsf{LOGSPACE}$*.*

Proof. Let $[\mathsf{B}] \subseteq \mathsf{L}$ hold for finite B, $\Gamma \subseteq \mathrm{PL}(\mathsf{B})$ be a finite set of formulas over the variables $\{x_1, \dots, x_n\}$, and $\phi \in \mathrm{PL}(\mathsf{B})$ a formula. Now it holds that $\Gamma \models \phi$ if and only if $\Gamma \cup \{\phi \oplus t, t\}$ is inconsistent, where t is a fresh variable. Let Γ' denote $\Gamma \cup \{\phi \oplus t, t\}$ rewritten s.t. for all $\psi \in \Gamma'$,

$$\psi = c_0 \oplus c_1 x_1 \oplus \cdots \oplus c_n x_n,$$

where $c_i \in \{\top, \bot\}$ for $1 \leqslant i \leqslant n$ and $c_i x_i$ is a shorthand for $c_i \wedge x_i$. Γ' is logspace constructible since $c_0 = \top$ iff $\psi(\bot, \dots, \bot) = \top$, for $1 \leqslant i \leqslant n$, $c_i = \top$ iff

$$\psi(\bot, \dots, \bot) \not\equiv \psi(\underbrace{\bot, \dots, \bot}_{i-1}, \top, \bot, \dots, \bot),$$

and affine formulas can be evaluated in logarithmic space [Schnoor, 2010]. Γ' can now be transformed into a system of linear equations S via

$$c_0 \oplus c_1 x_1 \oplus \cdots \oplus c_n x_n \mapsto c_0 + c_1 x_1 + \cdots + c_n x_n = 1 \pmod 2.$$

Clearly, the resulting system of linear equations has a solution if and only if Γ' is consistent. The equations are furthermore defined over the field $\mathbb{Z}_2$, hence existence of a solution can be decided in $\oplus\mathsf{LOGSPACE}$ as shown in [Buntrock et al., 1992]. □

Theorem 3.7.
Let $\langle W, D \rangle$ *be a default theory.*

- *If* W *is consistent, then every stable extension of* $\langle W, D \rangle$ *is consistent.*
- *If* W *is inconsistent, then* $\langle W, D \rangle$ *has a stable extension.*

Proof. Without a proof here. □

Theorem 3.8.
Let B *be a finite set of Boolean functions such that* $\mathsf{N} \subseteq [\mathsf{B}] \subseteq \mathsf{L}$*. Then* $\mathrm{EXT}(\mathsf{B})$ *is* NP*-complete.*

Proof. Let B be a finite set of Boolean functions and (W, D) be a B-default theory. First we show the upper bound for the case $[\mathsf{B}] \subseteq \mathsf{L}$. The following algorithm decides $\mathrm{EXT}(\mathsf{B})$:

The algorithm clearly terminates and runs in nondeterministic polynomial time. The correctness follows from the definition of extensions through the stage wise construction in combination with the previous lemma which shows that line 6 can be done in polynomial time, as $\oplus\mathsf{LOGSPACE} \subseteq \mathsf{P}$.

Algorithm 3.1: Procedure deciding EXT(L) in nondeterministic polynomial time.

```
   Input: B-default theory $(W, D)$
1  guess a set $G \subseteq D$;
2  $G' := W \cup \{\gamma \mid \frac{\alpha:\beta}{\gamma} \in G\}$, $G_0 := W, i := 0$;;
3  while $i = 0$ or $G_i \neq G_{i+1}$ do
4      forall the $\frac{\alpha:\beta}{\gamma} \in D$ do
5          if $G_i \models \alpha$ and $G' \not\models \neg\beta$ then $G_{i+1} := G_i \cup \{\gamma\}$ ;
6      $i := i + 1$;
7  return true iff $G' = G_i$
```

To show NP-hardness of EXT(B) for $\mathsf{N} \subseteq [\mathsf{B}]$, we will $\leqslant_{\mathbf{cd}}$-reduce 3SAT to EXT(B). Let $\varphi = \bigwedge_{i=1}^{n}(\ell_{i1} \vee \ell_{i2} \vee \ell_{i3})$ and ℓ_{ij} be literals over propositions $\{x_1, \ldots, x_m\}$ for $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant 3$. We transform φ to the B-default theory $\langle W, D_\varphi \rangle$, where $W := \emptyset$ and

$$D_\varphi := \left\{ \frac{\top : x_i}{x_i} \;\middle|\; 1 \leqslant i \leqslant m \right\} \cup \left\{ \frac{\top : \neg x_i}{\neg x_i} \;\middle|\; 1 \leqslant i \leqslant m \right\} \cup \left\{ \frac{\sim \ell_{i\pi(1)} :\sim \ell_{i\pi(2)}}{\ell_{i\pi(3)}} \;\middle|\; 1 \leqslant i \leqslant n, \pi \text{ is a permutation of } \{1,2,3\} \right\}.$$

To prove the correctness of the reduction, first assume φ to be satisfiable. For each satisfying assignment $\sigma : \{x_1, \ldots, x_m\} \to \{0, 1\}$ for φ, we claim that

$$E := \mathbf{Th}(\{x_i \mid \sigma(x_i) = \top\} \cup \{\neg x_i \mid \sigma(x_i) = \bot\})$$

is a stable extension of $\langle W, D_\varphi \rangle$. We will verify this claim with the help of the first part of the stage wise construction of extensions. Starting with $E_0 = \emptyset$, we already get $E_1 = E$ by the default rules $\frac{\top : x_i}{x_i}$ and $\frac{\top:\neg x_i}{\neg x_i}$ in D_φ. As σ is a satisfying assignment for φ, each consequent of a default rule in D_φ is already in E. Hence $E_2 = E_1$ and therefore $E = \bigcup_{i \in \mathbb{N}} E_i$ is a stable extension of $\langle W, D_\varphi \rangle$.

Conversely, assume that E is a stable extension of $\langle W, D_\varphi \rangle$. Because of the default rules $\frac{\top : x_i}{x_i}$ and $\frac{\top:\neg x_i}{\neg x_i}$, we either get $x_i \in E$ or $\neg x_i \in E$ for all $i = 1, \ldots, m$. The rules of the type $\frac{\sim\ell_{i1} :\sim\ell_{i2}}{\ell_{i3}}$ ensure that E contains at least one literal from each clause $\ell_{i1} \vee \ell_{i2} \vee \ell_{i3}$ in φ. As E is deductively closed, E contains φ. By Theorem 3.7, the extension E is consistent, and therefore φ is satisfiable.

Hence, EXT(B) is NP-complete for every finite set B such that $\mathsf{N} \subseteq [\mathsf{B}] \subseteq \mathsf{L}$. □

Logic	Clone	Complexity
propositional; $\mathrm{SAT}(B)$	$\mathsf{S}_1 \subseteq [B]$	NP-complete
	otherwise	in P
$\mathrm{IMP}(B)$	$[B] \subseteq \mathsf{L}$	in $\oplus\mathsf{LOGSPACE}$
modal; $\mathbf{K}$-ML-SAT$_{\Box,\Diamond}(B)$	$[B] \subseteq \mathsf{R}_1, \mathsf{D}, \mathsf{V}, \mathsf{L}$	in P
	$\mathsf{E}_0 \subseteq [B] \subseteq \mathsf{E}$	coNP-complete
	$\mathsf{S}_{11} \subseteq [B] \subseteq \mathsf{M}$	PSPACE-complete
	$\mathsf{S}_1 \subseteq [B]$	PSPACE-complete
temporal; $\mathrm{CTL}(\mathrm{ALL}, B)$	BF	EXP-complete
CTL-SAT(T, B) for $T \cap \{\mathrm{AU}, \mathrm{EU}\} = \emptyset$	$\mathsf{V}_0 \subseteq [B] \subseteq \mathsf{V}$	TC^0-complete
CTL-SAT(T, B) for $T \cap \{\mathrm{AU}, \mathrm{EU}\} = \emptyset$	$\mathsf{E}_0 \subseteq [B] \subseteq \mathsf{E}$	TC^0-complete
CTL-SAT$_{\mathcal{P}}(T, B)$ for $T \cap \{\mathrm{AU}, \mathrm{EU}\} = \emptyset$	$\mathsf{V}_0 \subseteq [B] \subseteq \mathsf{V}$	$\mathsf{NLOGTIME}$-complete
CTL-SAT$_{\mathcal{P}}(T, B)$ for $T \cap \{\mathrm{AU}, \mathrm{EU}\} = \emptyset$	$\mathsf{E}_0 \subseteq [B] \subseteq \mathsf{E}$	$\mathsf{coNLOGTIME}$-complete
hybrid; $\mathbf{KD}$-HL-SAT$(\{\Diamond, \downarrow, @\}, B)$	$\mathsf{S}_1 \subseteq [B]$	coRE-hard
default; $\mathrm{EXT}(B)$	$\mathsf{N} \subseteq [B] \subseteq \mathsf{L}$	NP-complete

Table 3.1.: Overview of complexity of decision problems in investigated logics.

4. Descriptive Complexity Theory

To be is to be the value of a bound variable.

(Willard Van Orman Quine)

The field of descriptive complexity aims to characterize complexity classes $\mathcal{C}$ by a corresponding class of logic formulas $\mathcal{F}$ such that formulas from $\mathcal{F}$ must be able to express languages from $\mathcal{C}$. Herewith a connection between these two areas of research comes up and introduces a third way of classifying problems beyond time and space complexity.

In this chapter we will follow Neil Immerman and the 7th chapter of his book Descriptive Complexity [Immermann, 1998].

The main result of the upcoming section is a characterization of the complexity class NP by existential second-order logic. Second-order logic consists of first-order logic plus new relational variables over which we may quantify. For example, the formula $(\forall A^r)\phi$ means that for all choices of an r-ary relation A, ϕ holds. With SO we denote the set of all second-order expressible Boolean formulas (or queries). Observe that any second-order formula can be transformed into an equivalent formula with all second-order quantifiers in front—we know this already from the prenex normal form for first order formulas. Further denote with SO($\exists$) the restriction of SO to only existential second-order quantifiers, and respectively, SO($\forall$) the restriction of SO to only universal second-order quantifiers.

With respect to Section 1.4 we can see that it is possible to state a second-order formula Φ which expresses 3-colorability of graphs, where E^2 denotes the edge transition relation of a given graph.

$$\begin{aligned}\Phi := (\exists R^1)(\exists Y^1)(\exists B^1)(\forall x)\Big[\big(R(x) \vee Y(x) \vee B(x)\big) \wedge (\forall y)\Big(E(x,y) \rightarrow \\ \neg\big(R(x) \wedge R(y)\big) \wedge \neg\big(Y(x) \wedge Y(y)\big) \wedge \neg\big(B(x) \wedge B(y)\big)\Big)\Big]\end{aligned}$$

Any graph $G = (V, E)$ satisfies Φ iff G is 3-colorable. Usually one writes $G \models \Phi$ to denote this fact.

In the following section we will proof connections between the predicate calculus and complexity classes. Therefore we will consider only interpretations $I\colon \mathrm{STRUC}[\tau] \rightarrow \{0, 1\}$ which map structures over some vocabulary τ to $\{0, 1\}$ and hence only Boolean queries. This restriction is necessary as for a complexity class its languages are decision problems. Thus, for instance, an existential second-order formula ψ then corresponds to a language L_ψ in some complexity class $\mathcal{C}$ (we will see that, in fact, $\mathcal{C} = \mathsf{NP}$ holds). Then, the structures $\mathcal{A}$ for which $\mathcal{A} \models \psi$ holds are the corresponding members of the language ψ. Summarizing, languages in the world of the predicate calculus are formulas and satisfying structures (or models) are members (or words) of these languages.

4.1. Fagin's Theorem

Now we aim to connect second order logic (restricted to only existential relational quantifiers) with the complexity class NP. However this prove is not so easy. It will require some kind of similar technique as we know from Cook's Theorem showing the SAT is NP-complete. Though we want this proof to be as simple as possible. One first step into this direction will requires us to prove that first-order logic is a subset of the complexity class LOGSPACE.

Theorem 4.1.
The set of first-order Boolean queries is contained in the set of Boolean queries computable in deterministic logarithmic space: $\mathrm{FO} \subseteq \mathsf{LOGSPACE}$.

Proof. Let $\tau = (R_1^{a_1}, \dots, R_r^{a_r}, c_1, \dots, c_s)$ be a vocabulary, and let $\phi \in \mathrm{FO}(\tau)$ be a first-order Boolean query, $I_\phi \colon \mathrm{STRUC}[\tau] \to \{0,1\}$, such that $\phi \equiv (\exists x_1)(\forall x_2)\dots(Q_k x_k)\alpha(x_1, \dots, x_k)$, where α is quantifier-free. Now we need to construct a DTM M running in logarithmic space such that for all $\mathcal{A} \in \mathrm{STRUC}[\tau]$, $\mathcal{A} \models \phi$ iff M accepts input $\mathrm{bin}(\mathcal{A})$.

We will construct M inductively on k. If $k = 0$, then $\phi = \alpha$ is a quantifier-free sentence. Hence α is a fixed, finite Boolean combination of atomic formulae, which are either $p_1 \leqslant p_2$, or $\mathbf{BIT}(p_1, p_2)$ and $p_i \in \{c_1, \dots, c_s, 0, 1, \max\}$. Once M knows whether $\mathcal{A}$ satisfies each of these atomic subformulae, then M can determine if $\mathcal{A} \models \alpha$ holds by performing the fixed, finite Boolean combinations using state transitions.

By counting up to $\lceil \log n \rceil$ M can copy the p_i's that it needs onto its worktape. For the predicates M just needs to look at the corresponding bit position of the input string $\mathrm{bin}(\mathcal{A})$.

For example, assuming that the relation R_i is ℓ-ary, to calculate $R_i(a_1, \dots, a_\ell)$, M first needs to copy the corresponding values in the universe for the a_i's on its worktape. Denote these values with $\tilde{a}_i \in \{0, \dots, n-1\}$, where $n = \|\mathcal{A}\|$. Next M moves its head to location $n^{a_1} + n^{a_2} + \dots + n^{a_{i-1}} + 1$ which is the beginning of the array encoding R_i. Finally, it moves its read head $n^{\ell-1} \cdot \tilde{a}_1 + n^{\ell-2} \cdot \tilde{a}_2 + \dots + n \cdot \tilde{a}_{\ell-1} + \tilde{a}_\ell$ spaces to the right. The bit now being read is "1" iff $\mathcal{A} \models R_i(a_1, \dots, a_\ell)$.

It is easy to see that one can check this in logarithmic space and completes the construction of M in the base case.

Inductively, assume that all first-order queries with $k-1$ quantifiers are logspace computable. Let

$$\psi(x_1) = (\forall x_2) \cdots (Q_k x_k)\alpha(x_1, \dots, x_k).$$

Let M_0 be the logspace DTM that computes the query $\psi(c)$. Note that c is a new constant symbol substituted for the free variable x_1. To compute the query $\phi \equiv (\exists x_1)(\psi(x_1))$ we build the following logspace machine that cycles through all possible values of x_1, substitutes each of these for c, and runs M_0. IF any of these lead M_0 to accept, then we accept, else we reject. Note that the extra space needed is just $\log n$ bits to store the possible values of x_1. Simulating a universal quantifier is similar. □

Observe that this result does not imply QBF-VAL $\in$ LOGSPACE, as in QBF-VAL the size of the problem is the size of the formula and elements are just Boolean values, whereas in FO the size of the problem is the size of the structure, whereas the formula is fixed.

Corollary 4.2.
Let $\mathcal{C}$ be a complexity class which is closed under logspace many-one reduction $\leqslant_{\mathbf{m}}^{\mathbf{log}}$. Then $\mathcal{C}$ is also closed under first-order reductions $\leqslant_{\mathbf{fo}}$.

The following result is the first step of our main goal.

Lemma 4.3.
The second-order existentially definable Boolean queries are all computable in NP*. In symbols,* $\mathrm{SO}(\exists) \subseteq \mathsf{NP}$.

Proof. Given a second-order existential sentence $\Phi \equiv (\exists R_1^{r_1}) \cdots (\exists R_k^{r_k})\psi$. Let τ be the vocabulary of Φ. Our task is to build an NTM N running in polynomial time such that for all $\mathcal{A} \in \mathrm{STRUC}[\tau]$ it holds that

$$(\mathcal{A} \models \Phi) \text{ if and only if } \mathsf{N} \text{ accepts the input } \operatorname{bin}(\mathcal{A}). \tag{4.1}$$

Let $\mathcal{A}$ be an input structure to N and let $n = \|\mathcal{A}\|$ be the cardinality of $\mathcal{A}$. What N does is to nondeterministically write down a binary string of length n^{r_1} representing R_1, and so on and so forth until it has written n^{r_k} for R_k. After this polynomially number of steps, we have an expanded structure $\mathcal{A}' = (\mathcal{A}, R_1, R_2, \dots, R_k)$ yielding an FO structure. N should accept iff $\mathcal{A}' \models \psi$. By the previous theorem we can test if $\mathcal{A}' \models \psi$ in logspace, so certainly in NP. Notice that N accepts $\mathcal{A}$ iff there is some choice of relations R_1 through R_k such that $(\mathcal{A}, R_1, \dots, R_k) \models \psi$. Thus (4.1) holds. $\square$

Theorem 4.4 (Fagin's Theorem).
NP *is equal to the set of existential, second-order Boolean queries,* $\mathsf{NP} = \mathrm{SO}(\exists)$*. Furthermore, this equality remains true when the first-order part of the second-order formulas is restricted to be universal only.*

Proof. Let $\mathsf{N} = (Q, \Sigma, \Theta, \delta, q_0, \square, \{q_f\})$ be a nondeterministic Turing machine that uses time $n^k - 1$ for inputs x with $n = |x|$. Without loss of generality we may assume N that never visits cells left of the input and always has two nondeterministic selection options. Now we write a second-order sentence

$$\Phi = (\exists C_0^{2k} \cdots \exists C_g^{2k} \exists \Delta^k \exists \mathrm{pre}^1 \exists \mathrm{suc}^1)\phi_{\mathrm{suc}} \wedge \phi_{\mathrm{pre}} \wedge \phi_{\mathrm{comp}},$$

that says, "There exists an accepting computation $\overline{C}, \Delta$ of N." Later we will directly see what g is. More precisely, the first-order sentence ϕ will have the property that $(\mathcal{A}, \overline{C}, \Delta) \models \phi$ iff $\overline{C}, \Delta$ is an accepting computation of N on input $\mathcal{A}$, where $\overline{C}$ encodes the traversed configurations and Δ denotes the used transitions in each step.

We describe how to code N's computation. $\overline{C}$ consists of a matrix $\overline{C}(\overline{t}, \overline{s})$ of n^{2k} tape cells with space $\overline{s}$ and time $\overline{t}$ varying between 0 and $n^k - 1$. The computation of N can be viewed in Figure 4.1. We use k-tuples of variables $\overline{t} = t_1, \dots, t_k$ and $\overline{s} = s_1, \dots, s_k$ each ranging over the universe of $\mathcal{A}$, i.e., from 0 to $n - 1$, to code these values.

	Space							
	0	1	p	$n-1$	n		n^k-1	Δ
Time 0	(q_0, w_0)	w_1	$\cdots$	w_{n-1}	$\square$	$\cdots$	$\square$	δ_0
1	w_0	(q_1, w_1)	$\cdots$	w_{n-1}	$\square$	$\cdots$	$\square$	δ_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
t			$a_{-1}a_0a_1$					δ_t
$t+1$			b					δ_{t+1}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n^k-1	$(q_f, 1)$	$\square$	$\cdots$			$\cdots$	$\square$	

Figure 4.1.: NP computation on input $w_0 \ldots w_{n-1}$.

A configuration of N of length n^k can be described as words over $\Gamma = (Q \times \Theta) \cup \Theta$, where the string

$$a_0 a_1 \ldots a_{i-1}(q, a_i)a_{i+1} \ldots a_{n^k-1}$$

means that the tape content of N on the investigated step is $a_0 \ldots a_{n^k-1}$, the head is on position i and the recent state is q.

For each $\overline{t}, \overline{s}$ pair, $\overline{C}(\overline{t}, \overline{s})$ codes the tape symbol σ that appears in cell $\overline{s}$ at time $\overline{t}$ if N's head is not on this cell. If the head is present, then $\overline{C}(\overline{t}, \overline{s})$ codes the pair (q, σ) consisting of N's state q at time $\overline{t}$ and tape symbol σ. Let $\Gamma = \{\gamma_0, \ldots, \gamma_g\}$ be a listing of the possible contents of a computation cell. We will let C_i be a 2k-ary relation variable for $0 \leqslant i \leqslant g$. The intuitive meaning of $C_i(\overline{t}, \overline{s})$ is that computation cells $\overline{s}$ at time $\overline{t}$ contain symbol γ_i.

If we want to speak about the ordering of elements $\overline{t}, \overline{t'}$ we will use two functions pre and suc which are quantified and talk about the predecessor and the successor of an element. The formulas ϕ_{pre} and ϕ_{suc} define these functions correctly (*exercise*).

At each step, the NTM will make one of at most two possible choices. We encode these choices in k-ary relation Δ. Intuitively, $\Delta(\overline{t})$ is true if step suc$(\overline{t})$ of the computation makes choice "1"; otherwise it makes choice "0". Note that these choices can be determined from $\overline{C}$, but the formula is simplified when we explicitly quantify Δ.

It is now fairly straightforward to write the first-order sentence $\phi(\overline{C}, \Delta)$ saying that $\overline{C}, \Delta$ codes a valid accepting computation of N. The sentence ϕ_{comp} uses a formula $\phi := \phi_{\text{init}} \wedge \phi_{\text{consistent}} \wedge \phi_{\text{transition}} \wedge \phi_{\text{accept}}$, where ϕ_{init} asserts that row 0 of the computation correctly codes input bin$(\mathcal{A})$, $\phi_{\text{consistent}}$ says that it is never the case that $C_i(\overline{t}, \overline{s})$ and $C_j(\overline{t}, \overline{s})$ both hold for $i \neq j$, $\phi_{\text{transition}}$ encodes the correctness of transition windows, and ϕ_{accept} says that the last row of the computation includes the accept state q_f. We may assume that when N accepts it clears its tape and moves all the way to the left and enters this unique accept state q_f. We can write sentence ϕ_{accept} explicitly. Let γ_ℓ for $\ell \in \{0, \ldots, g\}$ be the member of Γ corresponding to the pair $(q_f, 1)$ of state q_f, looking at the symbol 1. Then $\phi_{\text{accept}} = C_\ell(\overline{\max}, \overline{0})$.

Assume that $\gamma_0, \gamma_1, \gamma_2$ are $0, 1, \Box$ respectively. The sentence ϕ_{init} must define the first row of the tableau for $\text{bin}(\mathcal{A})$. Let $\tau = (R_1^{r_1}, \ldots, R_\ell^{r_\ell}, f_1^{a_1}, \ldots, f_r^{a_r}, c_1, \ldots, c_p)$ be the vocabulary. For ease of encoding we use $a = \max\{r_1, \ldots, r_\ell, a_1, \ldots, a_p\}$. Further we may assume that $a \leqslant k$ holds (otherwise we can just increase k in advance). Now given a τ-structure $\mathcal{A}$ we need to introduce a formula which correctly associates the corresponding parts of C_0, C_1, C_2 for time step $\bar{0}$ to the binary representation of $\mathcal{A}$. We choose an n^a-block encoding form as follows

$$\begin{aligned}\text{bin}(\mathcal{A}) := \overbrace{1^n 0^{n^a - n}}^{|A|} & \text{bin}(R_1^{\mathcal{A}}) \cdots \text{bin}(R_\ell^{\mathcal{A}}) \\ & \text{bin}(f_1^{\mathcal{A}}) \cdots \text{bin}(f_r^{\mathcal{A}}) \\ & \text{bin}(c_1^{\mathcal{A}}) \cdots \text{bin}(c_p^{\mathcal{A}}) \in \{0,1\}^{(\ell+p+1)\cdot n^a}.\end{aligned}$$

If $n \geqslant \ell + p + r$ then the positions $0, \ldots, (\ell + p + 1) \cdot n^a - 1$ of $\text{bin}(\mathcal{A})$ are encoded by those k-tuples from $\bar{s} = s_1, \ldots, s_k \in \{0, \ldots, n-1\}^k$ such that $s_1 = \cdots s_{k-a-2} = 0$, s_{k-a-1} is used to reference the respective relation/constant number, and $s_{k-a}, \ldots, s_k$ are used to reference the membership of a tuple in the relation (each relation has arity a and the universe has size n thus the membership of each tuple over $\{0, \ldots, n-1\}^a$ can be encoded by a string $\{0,1\}^{n^a}$). This leads to the following FO-formula:

$$\begin{aligned}\phi_{\text{init}} := \forall s_1 \cdots \forall s_k (C_0(\bar{0}, \bar{s}) \leftrightarrow \psi_0(\bar{s}))\wedge \\ (C_1(\bar{0}, \bar{s}) \leftrightarrow \psi_1(\bar{s}))\wedge \\ (C_2(\bar{0}, \bar{s}) \leftrightarrow \neg(C_0(\bar{s}) \vee C_1(\bar{s}))\end{aligned}$$

where $\psi_0(\bar{s})/\psi_1(\bar{s})$ are two FO-formulas which define that a $0/1$ is at position $\bar{s}$ in $\text{bin}(\mathcal{A})$. These two formulas are then defined as follows:

$$\begin{aligned}\psi_1(\bar{s}) := \bigwedge_{i=1}^{k-a-2} s_i = 0 \wedge \Bigg(& \bigwedge_{i=1}^{\ell} (s_{k-a-1} = \text{suc}^i(0)) \wedge \underbrace{R_i(s_{k-a}, \ldots, s_k)}_{\star} \vee \\ & \bigwedge_{i=\ell+1}^{r+\ell} (s_{k-a-1} = \text{suc}^i(0)) \wedge \underbrace{f_i(s_{k-a}, \ldots, s_{k-1}) = s_k}_{\heartsuit} \vee \\ & \bigwedge_{i=\ell+r+1}^{\ell+r+p} (s_{k-a-1} = \text{suc}^i(0)) \wedge \underbrace{\bigwedge_{j=k-a}^{k-1} s_j = 0 \wedge s_k = c_i \vee}_{\clubsuit} \\ & \bigwedge_{i=k-a-1}^{k-1} s_i = 0 \Bigg)\end{aligned}$$

$$\begin{aligned}\psi_0(\bar{s}) :=& [\text{“}\bar{s}\text{ encodes a number between } n \text{ and } n^a - n\text{”}]\vee\\ &\bigwedge_{i=1}^{\ell}(s_{k-a-1} = \mathrm{suc}^i(0) \wedge \neg\star)\vee\\ &\bigwedge_{i=1+\ell}^{\ell+p}(s_{k-a-1} = \mathrm{suc}^i(0) \wedge \neg\heartsuit)\vee\\ &\bigwedge_{i=1+\ell+p}^{\ell+p+r}(s_{k-a-1} = \mathrm{suc}^i(0) \wedge \neg\clubsuit).\end{aligned}$$

If $n < \ell + p$ then there are only constantly many inputs, hence structures, accepted by M. In that case we can directly encode these into the FO-formula. This is expressed in the final formula later.

The following sentence $\phi_{\text{transition}}$ asserts that the contents of tape cell $(\mathrm{suc}(\bar{t}), \bar{s})$ follows from the contents of cells $(\bar{t}, \mathrm{pre}(\bar{s})), (\bar{t}, \bar{s})$, and $(\bar{t}, \mathrm{suc}(\bar{s}))$ via the move $\Delta(\bar{t})$ of N. Hence we can describe this connection via two finite 4-ary relations Δ_0, Δ_1 which includes all allowed tuples $\begin{bmatrix} a & b & c \\ & d & \end{bmatrix} \in \Delta_0$ (resp. $\in \Delta_1$). As in every step two nondeterministic choices are available Δ_0 corresponds to one of them and Δ_1 corresponds to the other.

In the following we encode the transition relation. Observe that the head-movement is irrelevant for the window construction as it is directly expressed via the corresponding symbols in Γ.

$$\begin{aligned}\phi_{\text{transition}} :=& (\forall\bar{t}.\bar{t} \neq \overline{\max})(\forall\bar{s}.\bar{0} < \bar{s} < \overline{\max})\\ &\bigwedge_{\substack{a,b,c,d\in\Gamma,\\(a,b,c,d)\in\Delta_1}}\Big(\Delta(\bar{t}) \wedge C_a(\bar{t}, \mathrm{pre}(\bar{s})) \wedge C_b(\bar{t}, \bar{s}) \wedge C_c(\bar{t}, \mathrm{suc}(\bar{s})) \wedge \Delta(\bar{t}) \to C_d(\mathrm{suc}(\bar{t}), \bar{s})\Big)\wedge\\ &\quad\bigwedge_{\substack{a,b,c,d\in\Gamma,\\(a,b,c,d)\in\Delta_0}}\Big(\neg\Delta(\bar{t}) \wedge C_a(\bar{t}, \mathrm{pre}(\bar{s})) \wedge C_b(\bar{t}, \bar{s}) \wedge C_c(\bar{t}, \mathrm{suc}(\bar{s})) \wedge \Delta(\bar{t}) \to C_d(\mathrm{suc}(\bar{t}), \bar{s})\Big).\end{aligned}$$

Further we need the same formula with Δ_0 instead of Δ_1 and $\neg\Delta(\bar{t})$ instead of $\Delta(\bar{t})$ for the other nondeterministic choice. Similarly one can construct formulae which encode the transition behavior at the borders of the computation matrix.

Finally the formula which states consistent C_is:

$$\phi_{\text{consistent}} := \forall\bar{t}\bar{s} \bigwedge_{1\leqslant i\leqslant g}\left(C_i(\bar{t}, \bar{s}) \to \bigwedge_{1\leqslant j\neq i\leqslant g}\neg C_j(\bar{t}, \bar{s})\right) \wedge \forall\bar{t}\bar{s}\bigvee_{1\leqslant i\leqslant g} C_i(\bar{t}, \bar{s})$$

The full formula then incorporates the differentiation with respect to the size of the universe. If the universe is small, i.e., $\overline{\max} < \ell + p + 1$ then $|\mathcal{A}| \leqslant (1 + \ell + p) \cdot \ell^a$ is true. Thus there are only constantly many different such structures. After knowing M we can check for all such structures which are accepted by M and directly encode those into the formula. Let denote with $\mathcal{S}$ the set of these small structures. If the size of the universe is not bound by $\ell + p + r$ then we use the formula ϕ instead.

$$
\begin{aligned}
\phi_{\text{comp}} :=& (\max \geqslant \mathrm{suc}^{\ell+p+r}(0) \rightarrow \phi) \wedge \\
& \Bigg(\bigwedge_{i=1}^{\ell+p+r-1} \max = \mathrm{suc}^{i}(0) \rightarrow \\
& \qquad \bigvee_{\substack{M \text{ accepts} \\ \mathrm{bin}(\mathcal{A}') \in \mathcal{S} \\ \|A'\| = i}} \bigwedge_{\substack{j=1 \\ k=\mathrm{arity}(R_j)}}^{\ell} \bigwedge_{(a_1,\dots,a_k) \in R_j^{\mathcal{A}'}} R_j(\mathrm{suc}^{a_1}(0), \dots, \mathrm{suc}^{a_k}(0)) \\
& \qquad \bigwedge_{\substack{j=1 \\ k=\mathrm{arity}(f_j)}}^{r} \bigwedge_{f_j^{\mathcal{A}'}(a_1,\dots,a_{k-1}) = a_k} f_j(\mathrm{suc}^{a_1}(0), \dots, \mathrm{suc}^{a_{k-1}}(0)) = \mathrm{suc}^{a_k}(0) \\
& \qquad \bigwedge_{j=1}^{p} \bigwedge_{c_j^{\mathcal{A}'} = a} c_j = \mathrm{suc}^{a}(0) \Bigg).
\end{aligned}
$$

Corollary 4.5.
$\mathsf{coNP} = \mathrm{SO}(\forall)$.

The main idea is to simulate the alternations of the polynomial time hierarchy with alternations of existentially and universally quantified relations leading to the following theorem.

Theorem 4.6 ([Stockmeyer, 1976]).
$\mathsf{PH} = \mathrm{SO}$.

4.2. Least Fixed Points

Consider you want to extend the power of first-order logic without immediately reaching second-order logic. One such way are relations that are not first-order expressible but can be defined inductively. For instance, the transitive closure is such a type of relations we consider. Let τ be a vocabulary for graphs containing the edge transition E^2 as a binary relation. Then the reflexive, transitive closure E^* of E can be defined with the help of a binary relation variable R as follows (note that the following formula is just describing *one step* of the closure):

$$\phi(R, x, y) \equiv x = y \vee \exists z (E(x, z) \wedge R(z, y)).$$

For any structure $\mathcal{A}$ we now have induced by ϕ a mapping from binary relations on the universe of $\mathcal{A}$ to binary relations on the universe of $\mathcal{A}$:

$$\phi^{\mathcal{A}}(R) = \{(a, b) \mid \mathcal{A} \models \phi(R, a, b)\}.$$

Such mappings induced by a formula ψ are called *monotone* if for all R, S it holds that

$$R \subseteq S \text{ implies } \psi^{\mathcal{A}}(R) \subseteq \psi^{\mathcal{A}}(S).$$

As R appears only positively in ϕ this mapping $\phi^{\mathcal{A}}$ is monotone. Denote with $(\phi^{\mathcal{A}})^r$ an r-times iteration of $\phi^{\mathcal{A}}$. Now it holds that

$$(\phi^{\mathcal{A}})(\emptyset) = \{(a,b) \in |\mathcal{A}|^2 \mid \text{distance}(a,b) \leqslant 0\},$$
$$(\phi^{\mathcal{A}})^2(\emptyset) = \{(a,b) \in |\mathcal{A}|^2 \mid \text{distance}(a,b) \leqslant 1\},$$

and in general,

$$(\phi^{\mathcal{A}})^r(\emptyset) = \{(a,b) \in |\mathcal{A}|^2 \mid \text{distance}(a,b) \leqslant r-1\},$$

Hence $(\phi^{\mathcal{A}})^{||\mathcal{A}||} = E^*$ which is equal to the least fixed point of $\phi^{\mathcal{A}}$.

Theorem 4.7 (Finite Version of the Knaster-Tarski Theorem).
Let R be a new relation symbol of arity k, and let $\phi(R, x_1, \ldots, x_k)$ be a monotone first-order formula. Then for any finite structure $\mathcal{A}$, the least fixed point of $\phi^{\mathcal{A}}$ exists. It is equal to $(\phi^{\mathcal{A}})^r(\emptyset)$ where r is minimal so that $(\phi^{\mathcal{A}})^r(\emptyset) = (\phi^{\mathcal{A}})^{r+1}(\emptyset)$ holds. Furthermore, we have $r \leqslant ||\mathcal{A}||^k$.

Proof. Consider the sequence

$$\emptyset \subseteq (\phi^{\mathcal{A}})(\emptyset) \subseteq (\phi^{\mathcal{A}})^2(\emptyset) \subseteq (\phi^{\mathcal{A}})^3(\emptyset) \subseteq \cdots .$$

This is true as $\phi^{\mathcal{A}}$ is monotone. If $(\phi^{\mathcal{A}})^{i+1}(\emptyset)$ strictly contains $(\phi^{\mathcal{A}})^i(\emptyset)$ then it must contain at least one new k-tuple from $|\mathcal{A}|$. Since there are at most $||\mathcal{A}||^k$ such k-tuples, for some $r \leqslant ||\mathcal{A}||^k$, we have $(\phi^{\mathcal{A}})^r(\emptyset) = (\phi^{\mathcal{A}})^{r+1}(\emptyset)$, that is, $(\phi^{\mathcal{A}})^r(\emptyset)$ is a fixed point of $\phi^{\mathcal{A}}$.

Now let S be any other fixed point of $\phi^{\mathcal{A}}$. We show by induction that $(\phi^{\mathcal{A}})^i(\emptyset) \subseteq S$ for all i. For the base case we have $(\phi^{\mathcal{A}})^0(\emptyset) = \emptyset \subseteq S$. Now suppose that $(\phi^{\mathcal{A}})^i(\emptyset) \subseteq S$ holds. Since $\phi^{\mathcal{A}}$ is monotone, it holds tha

$$(\phi^{\mathcal{A}})^{i+1}(\emptyset) = \phi^{\mathcal{A}}((\phi^{\mathcal{A}})^i(\emptyset)) \subseteq \phi^{\mathcal{A}}(S) = S.$$

Thus, $(\phi^{\mathcal{A}})^r(\emptyset) \subseteq S$ and $(\phi^{\mathcal{A}})^r(\emptyset)$ is the least fixed point of $\varphi^{\mathcal{A}}$ as claimed. □

If we now write $(\text{LFP}_{R^k x_1 \ldots x_k \varphi})$ denoting this least fixed point, then this new type of operator formalizes the definition of new relations by induction for positive occurrences of R in ϕ. The subscript denotes which relation and which domain variables we are referring to with the fixed point.

Hence the reachability problem in directed graphs GAP now can be expressed as via the notion of the fixed point operator: $\text{LFP}_{Rxy\phi}(s,t)$.

Definition. *Define* FO(LFP) *as the extension of first-order logic with the least fixed point operator* LFP. *If $\phi(R^k, x_1, \ldots, x_k)$ is an R^k-positive formula in* FO(LFP), *then* $(\text{LFP}_{R^k x_1 \ldots x_k \phi})$ *may be used as a new k-ary relation symbol denoting the least fixed point of ϕ.*

Lemma 4.8.
FO(LFP) *is closed under first-order reductions* $\leqslant_{\mathbf{fo}}$.

Proof. *Exercise.* □

Theorem 4.9.
Over finite, ordered structures, FO(LFP) $=$ P.

Proof. "$\subseteq$": Let $\mathcal{A}$ be an input structure, let $n = \|\mathcal{A}\|$, and let $(\mathrm{LFP}_{Rx_1\ldots x_k}\phi)$ be a fixed-point formula. By Theorem 4.7, we know that this fixed point evaluated on $\mathcal{A}$ is $(\phi^{\mathcal{A}})^{n^k}(\emptyset)$. This amounts to evaluating the first-order query ϕ at most n^k times. We have seen in Theorem 4.1 that first-order queries may be evaluated in LOGSPACE, thus easily in P.

"$\supseteq$": Since FO(LFP) includes query ALTGAP (on ordered graphs), which is complete for P via first-order reductions, and FO(LFP) is closed under first-order reduction due to Lemma 4.8, FO(LFP) includes all polynomial-time queries. □

Corollary 4.10.
P $=$ NP *if and only if over finite, ordered structures* FO(LFP) $=$ SO.

A. Foundations

Mathematical reasoning may be regarded rather schematically as the exercise of a combination of two facilities, which we may call intuition and ingenuity.

(Alan Turing)

A.1. Complexity Theory

Here some notions relevant to this lecture are repeated. For a deeper introduction into this area we suggest either the lecture notes to the lecture *Komplexitätstheorie* or the two very good books [Arora and Barak, 2009, Papadimitriou, 1994].

class name	characterization
LOGSPACE	$\mathsf{SPACE}(\log n)$
P	$\mathsf{TIME}(n^{O(1)})$
NP	$\mathsf{NTIME}(n^{O(1)})$
coNP	$\{\overline{A} \mid A \in \mathsf{NP}\}$
PSPACE	$\mathsf{SPACE}(n^{O(1)})$

Table A.1.: Important complexity classes.

Definition (Reduction Functions). *Let $\mathcal{C}$ be a complexity class, let $A \subseteq \Sigma^*, B \subseteq \Delta^*$ be languages over the alphabet Σ, resp., Δ. We say A is* polynomially many-one reducible to B *(*logspace many-one reducible to B*), $A \leqslant^{\mathbf{p}}_{\mathbf{m}} B$ ($A \leqslant^{\mathbf{log}}_{\mathbf{m}} B$, iff there exists a function $f\colon \Sigma^* \to \Delta^*$ which can be computed in P (in $\mathsf{LOGSPACE}$) such that for all $x \in \Sigma^*$ it holds that $x \in A \Leftrightarrow f(x) \in B$.*

An oracle Turing machine is a usual Turing machine which has access to an oracle B. During the computation M may write a string on its oracle tape. If M enters a state $q_?$ and w is on the oracle tape, then M moves to state q_+ if $w \in B$ and to q_- if $w \notin B$. Then M is charged one time unit and the oracle tape is emptied. If A is accepted by TM M with oracle B, then we write $A = L(M, B)$. Now define

$$\mathsf{P}^B := \{L(M, B) \mid M \text{ is a polynomial time running DTM with oracle } B\},$$
$$\mathsf{NP}^B := \{L(M, B) \mid M \text{ is a polynomial time running NTM with oracle } B\}.$$

If $\mathcal{C}$ is a class of sets, then define

$$\mathsf{P}^{\mathcal{C}} := \bigcup_{B \in \mathcal{C}} \mathsf{P}^{B} \quad \text{and} \quad \mathsf{NP}^{\mathcal{C}} := \bigcup_{B \in \mathcal{C}} \mathsf{NP}^{B}.$$

Definition (Polynomial time hierarchy). *Define*

$$\begin{aligned} \Delta_0^{\mathsf{P}} \ = \ \Sigma_0^{\mathsf{P}} \ = \ \Pi_0^{\mathsf{P}} &:= \mathsf{P} \quad \textit{and for } k \geqslant 0\textit{:} \\ \Delta_{k+1}^{\mathsf{P}} &:= \mathsf{P}^{\Sigma_k^{\mathsf{P}}}, \\ \Sigma_{k+1}^{\mathsf{P}} &:= \mathsf{NP}^{\Sigma_k^{\mathsf{P}}}, \\ \Pi_{k+1}^{\mathsf{P}} &:= \{A \mid \overline{A} \in \Sigma_{k+1}^{\mathsf{P}}\}. \\ \textit{Further let} \quad \mathsf{PH} &:= \bigcup_{k \geqslant 0} \left(\Delta_k^{\mathsf{P}} \cup \Sigma_k^{\mathsf{P}} \cup \Pi_k^{\mathsf{P}}\right). \end{aligned}$$

A.2. First-Order Logic

Here we follow, again, [Immermann, 1998].

As usual a *vocabulary* $\tau = (R_1^{a_1}, \dots, R_r^{a_r}, c_1, \dots, c_s, f_1^{r_1}, \dots, f_t^{r_t})$ is a tuple of relation symbols, constant symbols, and function symbols. R_i is a relation symbol of arity a_i, and f_j is a function symbol of arity r_j.

A *structure* with vocabulary τ is a tuple $\mathcal{A} = (|\mathcal{A}|, R_1^{\mathcal{A}}, \dots, R_r^{\mathcal{A}}, c_1^{\mathcal{A}}, \dots, c_s^{\mathcal{A}}, f_1^{\mathcal{A}}, \dots, f_t^{\mathcal{A}})$ whose universe is the nonempty set $|\mathcal{A}|$. For each relation symbol R_i, $\mathcal{A}$ has a relation $R_i^{\mathcal{A}}$ of arity a_i defined on $|\mathcal{A}|$, i.e., $R_i^{\mathcal{A}} \subseteq |\mathcal{A}|^{a_i}$. Similarly we proceed with the functional symbols: $f_i^{\mathcal{A}}$ is a total function from $|\mathcal{A}|^{r_i}$ to $|\mathcal{A}|$.

We define STRUC[τ] as the set of finite structures of vocabulary τ.

The binary predicate **BIT**(i, j) is true if the jth bit in the binary representation of i is 1.

Definition (Interpretation). ϕ *first-order formula,* $\mathcal{A} \in$ STRUC[τ] *structure. A mapping* $i\colon V \to |\mathcal{A}|$ *is an* interpretation*, where* V *is the set of free variables of* ϕ*. It holds* $i(v) = i^{\mathcal{A}}$ *for all other variables.*

Definition (Semantics). $\mathcal{A} \in$ STRUC[τ] *structure,* i *interpretation. Then*

$$\begin{aligned} (\mathcal{A}, i) \models t_1 = t_2 &\Leftrightarrow i(t_1) = i(t_2), \\ (\mathcal{A}, i) \models R_j(t_1, \dots, t_{a_j}) &\Leftrightarrow \langle i(t_1), \dots, i(t_{a_j}) \rangle \in R_j^{\mathcal{A}}, \\ (\mathcal{A}, i) \models \neg\phi &\Leftrightarrow \textit{ it is not the case } (\mathcal{A}, i) \models \phi, \\ (\mathcal{A}, i) \models \phi \wedge \psi &\Leftrightarrow (\mathcal{A}, i) \models \phi \textit{ and } (\mathcal{A}, i) \models \psi, \\ (\mathcal{A}, i) \models (\exists x)\phi &\Leftrightarrow \textit{there is an } a \in |\mathcal{A}| \textit{ such that } (\mathcal{A}, i_x^a) \models \phi, \end{aligned}$$

$$\textit{where } i_x^a(y) = \begin{cases} i(y) & , \textit{ if } y \neq x, \\ a & , \textit{ if } y = x \end{cases}$$

Definition ((Boolean) queries). *A* query *is any mapping* $I\colon \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$ *for two vocabularies* σ, τ *which is polynomially bounded, i.e,* $\|I(\mathcal{A})\| \leqslant p(\|\mathcal{A}\|)$ *for some polynomial* p*. Let* M *be a Turing machine. A* Boolean query *is a map* $I_b\colon \mathrm{STRUC}[\sigma] \to \{0,1\}$*.* I_b *may be thought of as a subset of* $\mathrm{STRUC}[\sigma]$*, hence those for which* $\mathrm{bin}(I(\mathcal{A}))$ *is true. If also* M *accepts* $\mathrm{bin}(\mathcal{A})$ *iff* $I_b(\mathcal{A}) = 1$*, then we say that* M *computes* I_b*.*

Definition (First-Order Queries). *Let* σ, τ *be vocabularies where* $\tau = (R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s)$ *and let* $k \in \mathbb{N}$*. A first-order query*

$$I\colon \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$$

is given by an $r+s+1$*-tuple of formulae* $\phi_0, \phi_1, \ldots, \phi_r, \psi_1, \ldots, \psi_s$ *from* $\mathrm{FO}(\sigma)$*. For each structure* $\mathcal{A} \in \mathrm{STRUC}[\sigma]$ *these formulae describe a structure* $I(\mathcal{A}) \in \mathrm{STRUC}[\tau]$*:*

$$I(\mathcal{A}) = (|I(\mathcal{A})|, R_1^{I(\mathcal{A})}, \ldots, R_r^{I(\mathcal{A})}, c_1^{I(\mathcal{A})}, \ldots, c_s^{I(\mathcal{A})}),$$
$$|I(\mathcal{A})| = \{(b^1, \ldots, b^k) \mid \mathcal{A} \models \phi_0(b^1, \ldots, b^k)\},$$
$$R_i^{I(\mathcal{A})} = \{((b_1^1, \ldots, b_1^k), \ldots, (b_{a_i}^1, \ldots, b_{a_i}^k)) \in |I(\mathcal{A})^{a_i}| \mid \mathcal{A} \models \phi_i(b_1^1, \ldots, b_1^k, \ldots, b_{a_i}^1, \ldots, b_{a_i}^k)\},$$
$$c_j^{I(\mathcal{A})} = \text{ the unique } (b^1, \ldots, b^k) \in |I(\mathcal{A})| \text{ such that } \mathcal{A} \models \psi_j(b^1, \ldots, b^k).$$

Further let $a = \max\{a_i \mid 1 \leqslant i \leqslant r\}$*, the free variables of* ϕ_i *be* $x_1^1, \ldots, x_1^k, \ldots, x_{a_i}^1, \ldots, x_{a_i}^k$*, and the free variables of* ϕ_0 *and all* ψ_j *are* $x_1^1, \ldots, x_1^k$*.*

If the ψ_j*'s have the property that for all* $\mathcal{A} \in \mathrm{STRUC}[\sigma]$*:*

$$|\{(b^1, \ldots, b^k) \in |\mathcal{A}|^k \mid (\mathcal{A}, b_1/x_1^1, \ldots, b_k/x_1^k) \models \phi_0 \wedge \psi_j\}| = 1$$

then we write $I = \lambda_{x_1^1 \ldots x_a^k}(\phi_0, \ldots, \psi_s)$ *and say* I *is a* k*-ary first-order query from* $\mathrm{STRUC}[\sigma]$ *to* $\mathrm{STRUC}[\tau]$*.*

Let FO *be the set of first-order Boolean queries, i.e., defined by a first-order sentence or a* k*-ary first-order query,* $k \in \mathbb{N}$*.*

Definition (First-order reductions). *Let* $A \subseteq \mathrm{STRUC}[\sigma]$*,* $B \subseteq \mathrm{STRUC}[\tau]$ *be Boolean queries. Let* $I\colon \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$ *be a first-order query such that for all* $\mathcal{A} \in \mathrm{STRUC}[\sigma]$ *it holds that* $\mathcal{A} \in A \Leftrightarrow I(\mathcal{A}) \in B$*. Then* I *is a* FO-many-one reduction *from* A *to* B*,* $A \leqslant_{\mathbf{fo}} B$*.*

Example.

$$\forall x \forall y (\neg E(x,x) \wedge (E(x,y) \to E(y,x)))$$

describes undirected, loop-free graphs.

$$\forall x \exists y z (y \neq z \wedge E(x,y) \wedge E(x,z) \wedge \forall w (E(x,w) \to (w = y \vee w = z)))$$

describes graphs whose vertices always have exactly two edges.

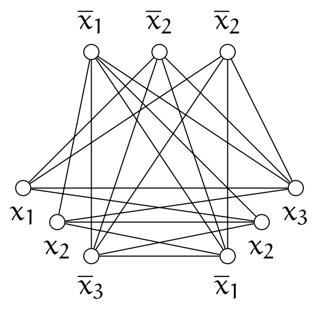
Observation: *Every FO-sentence (formulas without free variables)* φ *over vocabulary* τ *defines a Boolean first-order query* I_φ *on* $\mathrm{STRUC}[\tau]$ *where*

$$I_\varphi(\mathcal{A}) = 1 \text{ iff } \mathcal{A} \models \varphi.$$

Example. *Consider the $\leqslant_{\mathbf{m}}^{\mathbf{p}}$-reduction from 3SAT to CLIQUE as follows. Given a formula* $\varphi = (L_{1,1} \vee L_{1,2} \vee L_{1,3}) \wedge \cdots \wedge (L_{n,1} \vee L_{n,2} \vee L_{n,3})$*, one can assume that the number of clauses is equal to the number of variables, i.e., the variables are* $\{x_1, \dots, x_n\}$*. Then we map to a graph* $G = (V, E)$*, where*

$$\begin{aligned}
V &= \{L_{i,j} \mid 1 \leqslant i \leqslant, 1 \leqslant j \leqslant 3\}, \text{ and}\\
E &= \{(L_{i,j}, L_{i',j'}) \mid i \neq i', L_{i,j} \text{ and } L_{i',j'} \text{ are not contrary}\},\\
k &= n.
\end{aligned}$$

For instance $\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$ *will lead to the following graph:*



In the next step we need to formulate the first-order formulas to achieve an $\leqslant_{\mathbf{fo}}$*-reduction from 3SAT to CLIQUE. Assume the universe of* $\mathcal{A} = \varphi$ *consists of sets of clauses and variables. Hence a structure* $\mathcal{A}_\varphi = \langle A, P, N \rangle$*, where* $P(c, v)$ *is a relation stating variable* v *appears positively in clause* c *and similarly for* $N(c, v)$ *negatively seems reasonable. Thus in our case* $n = \|\mathcal{A}_{|varphi|}\| = \max\{\mathit{Vars}(\varphi), \mathit{Clauses}(\varphi)\}$*. Now turn towards the reduction function* I*. The universe of* $I(\mathcal{A})$ *is consisting of triples* $\bar{x} = \langle x^1, x^2, x^3 \rangle$*, where* x^1 *encodes the corresponding clause,* x^2 *the corresponding variable number, and* $x^3 = 1$ *means positively and* $x^3 = 2$ *means negatively.*

$$\phi_0 := x^3 \leqslant 2$$

encodes this approach. The edge relation E *is expressed via*

$$\phi'(\bar{x}, \bar{y}) := x^1 \neq y^1 \wedge (x^2 = y^2 \to x^3 = y^3)$$

where being undirected is achieved via

$$\phi(\bar{x}, \bar{y}) := \phi'(\bar{x}, \bar{y}) \vee \phi'(\bar{y}, \bar{x}).$$

Now the constant k *is missing. Here observe that we only need to agree on a good encoding. Usually the elements of the universe are ordered from* $0, \dots, n-1$ *implying a base* n *encoding leading to* $(010)_n$ *as* n *in decimal system. However we describe our* *elements shifted one to the right with* $1, \ldots, n$ *implying that symbol 1 encodes the 0 and so on. Hence,*

$$\psi(\overline{x}) := x^1 x^2 x^3 = 121.$$

Wrapping up the $\leqslant_{\mathbf{fo}}$*-reduction we desired is* $\lambda_{x^1x^2x^3y^1y^2y^3}\langle \phi_0, \phi, \psi\rangle$.

A.3. Circuit Complexity

Let AC^0 be the family of polynomial-sized circuits of constant depth using $\wedge$ and $\vee$ gates of unbounded fan-in.

Definition (Constant depth reductions). *A language* A *is* constant-depth reducible *to* B*, written* $A \leqslant_{\mathbf{cd}} B$*, if there is a logtime-uniform* AC^0*-circuit family with oracle gates for* B *that decides membership in* A.

Here, *logtime-uniform* means there is a deterministic TM that can check the structure of the circuit familiy $\mathcal{C}$ in time $O(\log n)$ where n is the size of $\mathcal{C}$.

A.4. Quantified Boolean Formulae

Definition (Quantified Boolean formulae (qbf)). $0, 1, x$ *(propositional variable) are qbf*

- F *qbf* $\Rightarrow \neg F$ *qbf*
- F_1, F_2 *qbf* $\Rightarrow (F_1 \wedge F_2)$ *and* $(F_1 \vee F_2)$ *qbf*
- F *qbf,* x *variable* $\Rightarrow \exists x\ F$ *and* $\forall x\ F$ *qbf*

Quantifier $\exists$ and $\forall$ quantify over truth values! They allow a very succinct representation of formulae

$$\exists x\ F \equiv F(0) \vee F(1), \qquad \forall x\ F \equiv F(0) \wedge F(1)$$

F is said to be *closed*, if all variables in F are quantified.

$$\text{QBF-VAL} =_{\mathsf{def}} \{F \mid F \text{ is closed qbf and } F \equiv 1\}$$
$$\text{QBF-3VAL} =_{\mathsf{def}} \{F \mid F \text{ is closed qbf in 3CNF and } F \equiv 1\}$$

Theorem A.1.
QBF-VAL *and* QBF-3VAL *are complete for* PSPACE *under* $\leqslant_{\mathbf{m}}^{\mathbf{p}}$.

Bibliography

[Areces et al., 2000] Areces, C., Blackburn, P., and Marx, M. (2000). The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8:653–679.

[Arora and Barak, 2009] Arora, S. and Barak, B. (2009). *Computational complexity: A modern approach.* Cambridge University Press.

[Berger, 1966] Berger, R. (1966). The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66.

[Buntrock et al., 1992] Buntrock, G., Damm, C., Hertrampf, U., and Meinel, C. (1992). Structure and importance of logspace-mod classes. *Mathematical Systems Theory*, 25(3):223–237.

[Donini et al., 1992] Donini, F., Lenzerini, M., Nardi, D., and Hollunder, B. (1992). The complexity of existential quantification in concept languages. *Artificial Intelligence.*

[Gelfond and Lifschitz, 1991] Gelfond, M. and Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386.

[Halpern, 1995] Halpern, J. (1995). The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2):361–372.

[Hemaspaandra, 2005] Hemaspaandra, E. (2005). The Complexity of Poor Man's Logic. *arXiv.org*, cs.LO/9911014v2.

[Hemaspaandra et al., 2010] Hemaspaandra, E., Schnoor, H., and Schnoor, I. (2010). Generalized modal satisfiability. *Journal of Computer and System Sciences*, 76(7):561–578.

[Immermann, 1998] Immermann, N. (1998). *Descriptive Complexity.* Springer.

[Kripke, 1963] Kripke, S. (1963). Semantical analysis of modal logic i normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96.

[Ladner, 1977] Ladner, R. E. (1977). The Computational Complexity of Provability in Systems of Modal Propositional Logic. pages 1–14.

[Lewis, 1979] Lewis, H. (1979). Satisfiability problems for propositional calculi. *Theory of Computing Systems.*

[McCarthy, 1980] McCarthy, J. (1980). Circumscription – a form of non-monotonic reasoning. 13:27–39.

[Papadimitriou, 1994] Papadimitriou, C. M. (1994). *Computational complexity*. Addison-Wesley.

[Regan and Vollmer, 1997] Regan, K. and Vollmer, H. (1997). Gap-languages and log-time complexity classes. *Theoretical Computer Science*, 188:101–116.

[Schnoor, 2010] Schnoor, H. (2010). The Complexity of Model Checking for Boolean Formulas. *International Journal of Foundations of Computer Science*, 21(03):289.

[Stockmeyer, 1976] Stockmeyer, L. (1976). The polynomial-time hierarchy. *Theoretical Computer Science*.

[Thomas, 2010] Thomas, M. (2010). *On the complexity of fragments of nonmonotonic logics*. PhD thesis, Cuvillier.

[Wang, 1961] Wang, H. (1961). Proving theorems by recognition–II. *The Bell System Technical Journal*, 40:1–41.