The Complexity of Generalized Satisfiability for Linear Temporal Logic

Michael Bauland[1], Thomas Schneider[2], Henning Schnoor[1], Ilka Schnoor[1], and Heribert Vollmer[1]

[1] Theoret. Informatik, Universität Hannover, Appelstr. 4, 30167 Hannover, Germany
{bauland,henning.schnoor,ilka.schnoor,vollmer}@thi.uni-hannover.de
[2] Informatik, Friedrich-Schiller-Universität, 07737 Jena, Germany
schneider@cs.uni-jena.de

Abstract. In a seminal paper from 1985, Sistla and Clarke showed that satisfiability for Linear Temporal Logic (LTL) is either NP-complete or PSPACE-complete, depending on the set of temporal operators used. If, in contrast, the set of propositional operators is restricted, the complexity may decrease. This paper undertakes a systematic study of satisfiability for LTL formulae over restricted sets of propositional and temporal operators. Since every propositional operator corresponds to a Boolean function, there exist infinitely many propositional operators. In order to systematically cover all possible sets of them, we use Post's lattice. With its help, we determine the computational complexity of LTL satisfiability for all combinations of temporal operators and all but two classes of propositional functions. Each of these infinitely many problems is shown to be either PSPACE-complete, NP-complete, or in P.

Keywords: computational complexity, linear temporal logic

1 Introduction

Linear Temporal Logic (LTL) was introduced by Pnueli in [Pnu77] as a formalism for reasoning about the properties and the behavior of parallel programs and concurrent systems, and has widely been used for these purposes. Because of the need to perform reasoning tasks — such as deciding satisfiability, validity, or truth in a structure generated by binary relations — in an automated manner, their decidability and computational complexity is an important issue.

It is known that in the case of full LTL with the operators F (eventually), G (invariantly), X (next-time), U (until), and S (since), satisfiability and determination of truth are PSPACE-complete [SC85]. Restricting the set of temporal operators leads to NP-completeness in some cases [SC85]. These results imply that reasoning with LTL is difficult in terms of computational complexity.

This raises the question under which restrictions the complexity of these problems decreases. Since the semantics of LTL is rather fixed, such restrictions can only be of syntactic nature. However, there are several possible constraints that can be posed on the syntax. One possibility is to restrict the set of temporal operators, which has been done almost exhaustively in [SC85].

Another constraint is to allow only a certain "degree of propositionality" in the language, i. e., to restrict the set of allowed propositional operators. Every propositional operator represents a Boolean function — e. g., the operator $\wedge$ (*and*) corresponds to the binary function whose value is 1 if and only if both arguments have value 1. There are infinitely many Boolean functions and hence an infinite number of propositional operators.

If these propositional restrictions are considered in a systematic way, this will lead to a complete classification of the complexity of the reasoning problems for LTL. Not only will this reveal all cases in which, say, satisfiability is tractable. It will also provide a better insight into the sources of hardness by explicitly stating the combinations of temporal and propositional operators that lead to NP- or PSPACE-hard fragments. In addition, the "sources of hardness" will be identified whenever a proof technique is not transferable from an easy to a hard fragment.

The effect of propositional restrictions on the complexity of the satisfiability problem was first considered by Lewis for the case of classical propositional logic in [Lew79]. He established a dichotomy — depending on the set of propositional operators, satisfiability is either NP-complete or decidable in polynomial time. In the case of modal propositional logic, a trichotomy has been achieved in [BHSS06]: modal satisfiability is PSPACE-complete, coNP-complete, or in P. That complete classification in terms of restriction on the propositional operators follows the structure of Post's lattice of closed sets of Boolean functions [Pos41].

This paper analyzes the same restrictions for LTL and combines them with restrictions on the temporal operators. Using Post's lattice, we examine the satisfiability problem for every combination of temporal *and* propositional operators. We determine the computational complexity of these problems, except for one case — the one in which only propositional operators based on the binary *xor* function (and, perhaps, constants) are allowed. We show that all remaining cases are either PSPACE-complete, NP-complete, or in P.

It is not the aim of this paper to focus on particular propositional restrictions that are motivated by certain applications. We prefer to give a classification as complete as possible which allows to choose a fragment that is appropriate, in terms of expressivity and tractability, for any given application.

Among our results, we exhibit cases with non-trivial tractability as well as the smallest possible sets of propositional and temporal operators that already lead to NP-completeness or PSPACE-completeness, respectively. Examples for the first group are cases in which only the unary *not* function, or only monotone functions are allowed, but there is no restriction on the temporal operators. As for the second group, if only the binary function f with $f(x, y) = (x \wedge \overline{y})$ is permitted, then satisfiability is NP-complete already in the case of propositional logic [Lew79]. Our results show that the presence of the same function f separates the tractable languages from the NP-complete and PSPACE-complete ones, depending on the set of temporal operators used. According to this, minimal sets of temporal operators leading to PSPACE-completeness together with f are, for example, $\{\mathsf{U}\}$ and $\{\mathsf{F}, \mathsf{X}\}$.

The technically most involved proof is that of PSPACE-hardness for the language with only the temporal operator S and the boolean operator f (Theorem 3.3). The difficulty lies in simulating the quantifier tree of a Quantified Boolean Formula (QBF) in a linear structure.

Our results are summarized in Table 1. The first column contains the propositional restrictions in terms of closed sets of Boolean functions (clones) whose terminology is introduced in the following section. The second column shows the classification of classical propositional logic as known from [Lew79] and [Coo71]. The last line in column 3 and 4 is largely due to [SC85]. All other entries are the main results of this paper. The only open case appears in the third line and is discussed in the Conclusion. Note that the case distinction also covers all clones which are not mentioned in the present paper.

temporal operators / function class (propositional operators)	$\emptyset$	$\{\mathsf{F}\}$, $\{\mathsf{G}\}$, $\{\mathsf{F}, \mathsf{G}\}$, $\{\mathsf{X}\}$	any other combination
below R_1 or below D	trivial	trivial	trivial
below M or below N	in P	in P	in P
L_0, L	in P	?	?
above S_1	NP-c.	NP-c.	PSPACE-c.
BF (all Boolean functions)	NP-c.	NP-c.	PSPACE-c.

Table 1. Complexity results for satisfiability. The entries "trivial" denote cases in which a given formula is always satisfiable. The abbreviation "c." stands for "complete." Question marks stand for open questions.

2 Preliminaries

A *Boolean function* or *Boolean operator* is a function $f : \{0,1\}^n \to \{0,1\}$. We can identify an n-ary propositional connector c with the n-ary Boolean operator f defined by: $f(a_1, \ldots, a_n) = 1$ if and only if the formula $c(x_1, \ldots, x_n)$ becomes true when assigning a_i to x_i for all $1 \leq i \leq n$. Additionally to propositional connectors we use the unary temporal operators X (next-time), F (eventually), G (invariantly) and the binary temporal operators U (until), and S (since).

Let B be a finite set of Boolean functions and M be a set of temporal operators. A *temporal B-formula over M* is a formula φ that is built from variables, propositional connectors from B, and temporal operators from M. More formally, a temporal B-formula over M is either a propositional variable or of the form $f(\varphi_1, \ldots, \varphi_n)$ or $g(\varphi_1, \ldots, \varphi_m)$, where φ_i are temporal B-formulae over

M, f is an n-ary propositional operator from B and g is an m-ary temporal operator from M. In [SC85], complexity results for formulae using the temporal operators F, G, X (unary), and U, S (binary) were presented. We extend these results to temporal B-formulae over subsets of those temporal operators. The set of variables appearing in φ is denoted with V_φ. If $M = \{\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{S}\}$ we call φ a *temporal B-formula*, and if $M = \emptyset$ we call φ a *propositional B-formula* or simply a *B-formula*. The set of all temporal B-formulae over M is denoted with $\mathrm{L}(M, B)$.

A model in linear temporal logic is a linear structure of states, which intuitively can be seen as different points of time, with propositional assignments. Formally a *structure* $S = (s, V, \xi)$ consists of an infinite sequence $s = (s_i)_{i\in\mathbb{N}}$ of distinct states, a set of variables V, and a function $\xi : \{s_i \mid i \in \mathbb{N}\} \to 2^V$ which induces a propositional assignment of V for each state. For a temporal $\{\wedge, \neg\}$-formula over $\{\mathsf{X}, \mathsf{U}, \mathsf{S}\}$ with variables from V we define what it means that *S satisfies φ in s_i* ($S, s_i \vDash \varphi$): let φ_1 and φ_2 be temporal $\{\wedge, \neg\}$-formulae over $\{\mathsf{X}, \mathsf{U}, \mathsf{S}\}$ and $x \in V$ a variable.

$$\begin{array}{ll}
S, s_i \vDash x & \text{if and only if } x \in \xi(s_i),\\
S, s_i \vDash \varphi_1 \wedge \varphi_2 & \text{if and only if } S, s_i \vDash \varphi_1 \text{ and } S, s_i \vDash \varphi_2,\\
S, s_i \vDash \neg\varphi_1 & \text{if and only if } S, s_i \nvDash \varphi_1,\\
S, s_i \vDash \mathsf{X}\varphi_1 & \text{if and only if } S, s_{i+1} \vDash \varphi_1,\\
S, s_i \vDash \varphi_1\mathsf{U}\varphi_2 & \text{if and only if there is a } k \geq i \text{ such that } S, s_k \vDash \varphi_2,\\
 & \qquad\qquad\qquad \text{and for every } i \leq j < k,\ \ S, s_j \vDash \varphi_1,\\
S, s_i \vDash \varphi_1\mathsf{S}\varphi_2 & \text{if and only if there is a } k \leq i \text{ such that } S, s_k \vDash \varphi_2,\\
 & \qquad\qquad\qquad \text{and for every } k < j \leq i,\ \ S, s_j \vDash \varphi_1.
\end{array}$$

The remaining temporal operators are interpreted as abbreviations: $\mathsf{F}\varphi = \mathit{true}\,\mathsf{U}\varphi$ and $\mathsf{G}\varphi = \neg\mathsf{F}\neg\varphi$. Therefore and since every Boolean operator can be composed from $\wedge$ and $\neg$, the above definition generalizes to temporal B-formulae for arbitrary sets B of Boolean operators.

A temporal B-formula φ over M is *satisfiable* if there exists a structure S such that $S, s_i \vDash \varphi$ for some state s_i from S. That allows us to define the problems we want to look at in this paper: Let B be a finite set of Boolean functions and M a set of temporal operators. Then $\mathrm{SAT}(M, B)$ is the problem to decide whether a given temporal B-formula over M is satisfiable. In the literature, another notion of satisfiability is sometimes considered, where we ask if a formula can be satisfied at the first state in a structure. It is easy to see that, in terms of computational complexity, this does not make a difference for our problems as long as we do not have the temporal operator S in our language. For this paper, we only study the satisfiability problem as defined above.

Sistla and Clarke analyzed the satisfiability problem for temporal $\{\wedge, \vee, \neg\}$-formulae over some sets of temporal operators.

Theorem 2.1 ([SC85]).

(1) $\mathrm{SAT}(\{\mathsf{F}\}, \{\wedge, \vee, \neg\})$ *is* NP*-complete.*

(2) $\mathrm{SAT}(\{\mathsf{F},\mathsf{X}\},\{\wedge,\vee,\neg\})$, $\mathrm{SAT}(\{\mathsf{U}\},\{\wedge,\vee,\neg\})$, *and* $\mathrm{SAT}(\{\mathsf{U},\mathsf{S},\mathsf{X}\},\{\wedge,\vee,\neg\})$ *are* PSPACE*-complete.*

Since there are infinitely many finite sets of Boolean functions, we introduce some algebraic tools to classify the complexity of the infinitely many arising satisfiability problems. We denote with id_k^n the n-ary projection to the k-th variable, i.e., $\mathrm{id}_k^n(x_1,\ldots,x_n) = x_k$, and with c_a^n the n-ary constant function defined by $c_a^n(x_1,\ldots,x_n) = a$. For $c_1^1(x)$ and $c_0^1(x)$ we simply write 1 and 0. A set C of Boolean functions is called a *clone* if it is closed under superposition, which means C contains all projections and C is closed under arbitrary composition [Pip97]. For a set B of Boolean functions we denote with $[B]$ the smallest clone containing B and call B a *base* for $[B]$. In [Pos41] Post classified the lattice of all clones and found a finite base for each clone.

With $\oplus$ we denote the binary exclusive or. Let f be an n-ary Boolean function. We define some properties for f:

- f is 1*-reproducing* if $f(1,\ldots,1) = 1$.
- f is *monotone* if $a_1 \leq b_1,\ldots,a_n \leq b_n$ implies $f(a_1,\ldots,a_n) \leq f(b_1,\ldots,b_n)$.
- f is 1*-separating* if there exists an $i \in \{1,\ldots,n\}$ such that $f(a_1,\ldots,a_n) = 1$ implies $a_i = 1$.
- f is *self-dual* if $f \equiv \mathrm{dual}(f)$, where $\mathrm{dual}(f)(x_1,\ldots,x_n) = \neg f(\neg x_1,\ldots,\neg x_n)$.
- f is *linear* if $f \equiv x_1 \oplus \cdots \oplus x_n \oplus c$ for a constant $c \in \{0,1\}$ and variables $x_1,\ldots,x_n$.

In Table 2 we define those clones that are essential for this paper plus four basic ones, and give Post's bases [Pos41] for them. The inclusions between them are given in Figure 1. The definitions of all clones as well as the full inclusion graph can be found, for example, in [BCRV03].

Name	Definition	Base
BF	All Boolean functions	$\{\vee,\wedge,\neg\}$
R_1	$\{f \in \mathrm{BF} \mid f$ is 1-reproducing $\}$	$\{\vee,\leftrightarrow\}$
M	$\{f \in \mathrm{BF} \mid f$ is monotone $\}$	$\{\vee,\wedge,0,1\}$
S_1	$\{f \in \mathrm{BF} \mid f$ is 1-separating $\}$	$\{x \wedge \overline{y}\}$
D	$\{f \mid f$ is self-dual$\}$	$\{x\overline{y} \vee x\overline{z} \vee (\overline{y} \wedge \overline{z})\}$
L	$\{f \mid f$ is linear$\}$	$\{\oplus,1\}$
L_0	$[\{\oplus\}]$	$\{\oplus\}$
V	$\{f \mid$ There is a formula of the form $c_0 \vee c_1x_1 \vee \cdots \vee c_nx_n$ such that c_i are constants for $1 \leq i \leq n$ that describes $f\}$	$\{\vee,1,0\}$
E	$\{f \mid$ There is a formula of the form $c_0 \wedge (c_1 \vee x_1) \wedge \cdots \wedge (c_n \vee x_n)$ such that c_i are constants for $1 \leq i \leq n$ that describes $f\}$	$\{\wedge,1,0\}$
N	$\{f \mid f$ depends on at most one variable$\}$	$\{\neg,1,0\}$
I	$\{f \mid f$ is a projection or constant$\}$	$\{0,1\}$
I_2	$\{f \mid f$ is a projection$\}$	$\emptyset$

Table 2. List of some closed classes of Boolean functions with bases

There is a strong connection between propositional formulae and Post's lattice. If we interpret propositional formulae as Boolean functions, it is obvious that $[B]$ includes exactly those functions that can be represented by B-formulae. This connection has been used various times to classify the complexity of problems related to propositional formulae: For example, Lewis presented a dichotomy for the satisfiability problem for propositional B-formulae: $\mathrm{SAT}(\emptyset, B)$ is NP-complete if $\mathrm{S}_1 \subseteq [B]$, and solvable in P otherwise [Lew79].

Post's lattice was applied for the equivalence problem [Rei01], counting [RW05] and finding minimal [RV03] solutions, and learnability [Dal00] for Boolean formulae. The technique has been used in non-classical logic as well: Bauland et al. achieved a trichotomy in the context of modal logic, which says that the satisfiability problem for modal formulae is, depending on the allowed propositional connectives, PSPACE-complete, coNP-complete, or solvable in P [BHSS06]. For the inference problem for propositional circumscription, Nordh presented another trichotomy theorem [Nor05].

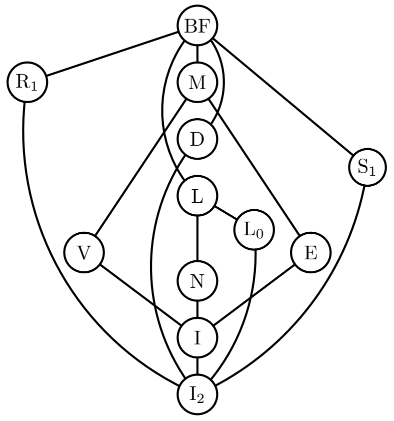


Fig. 1. Graph of some closed classes of Boolean functions

An important tool in restricting the length of the resulting formula in many of our reductions is the following lemma. It shows that for certain sets B, there are always short formulae representing the functions *and*, *or*, or *not*, respectively. Point (2) and (3) follow directly from the proofs in [Lew79], point (1) is Lemma 3.3 from [Sch05].

Lemma 2.2.

(1) Let B be a finite set of Boolean functions such that $\mathrm{V} \subseteq [B] \subseteq \mathrm{M}$ ($\mathrm{E} \subseteq [B] \subseteq \mathrm{M}$, resp.). Then there exists a B-formula $f(x, y)$ such that f represents $x \vee y$ ($x \wedge y$, resp.) and each of the variables x and y occurs exactly once in $f(x, y)$.

(2) Let B be a finite set of Boolean functions such that $[B] = \mathrm{BF}$. Then there are B-formulae $f(x, y)$ and $g(x, y)$ such that f represents $x \vee y$, g represents $x \wedge y$, and both variables occur in each of these formulae exactly once.

(3) Let B be a finite set of Boolean functions such that $\mathrm{N} \subseteq [B]$. Then there is a B-formula $f(x)$ such that f represents $\neg x$ and the variable x occurs in f only once.

3 Results

3.1 Hard cases

The following lemma gives our general upper bounds for various combinations of temporal operators. The proof of part (1) and (2) is a variation of the proof for Theorem 3.4 in [BHSS06], where, using a similar reduction, an analogous result for circuits was proved.

Lemma 3.1. *Let B be a finite set of Boolean functions. Then the following holds:*

(1) If $M \subseteq \{\mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{S}, \mathsf{X}\}$, then $\mathrm{SAT}(M, B)$ is in PSPACE*,*
(2) if $M \subseteq \{\mathsf{F}, \mathsf{G}\}$, then $\mathrm{SAT}(M, B)$ is in NP*, and*
(3) if $M \subseteq \{\mathsf{X}\}$, then $\mathrm{SAT}(M, B)$ is also in NP*.*

Proof. For (1), we will show that $\mathrm{SAT}(M, B) \leq_m^{\log} \mathrm{SAT}(\{\mathsf{U}, \mathsf{S}, \mathsf{X}\}, \{\wedge, \vee, \neg\})$, and for (2), we will show that $\mathrm{SAT}(M, B) \leq_m^{\log} \mathrm{SAT}(\{\mathsf{F}\}, \{\wedge, \vee, \neg\})$. The complexity result for these cases then follows from Theorem 2.1.

The construction for (1) and (2) is nearly identical: Let φ be a formula with arbitrary temporal operators and Boolean functions from B. We recursively transform the formula to a new formula using only the Boolean operators $\wedge$, $\vee$, and $\neg$, and the temporal operators U, S, and X for the first case and the temporal operator F for the second cases. For this we construct several formulae, which will be connected via conjunction. Let k be the number of subformulae of φ. Accordingly let $\varphi_1, \dots, \varphi_k$ be those subformulae with $\varphi = \varphi_1$. Let $x_1, \dots, x_k$ be new variables, i.e., distinct from the input variables of φ. For all i from 1 to k we make the following case distinction:

- If $\varphi_i = y$ for a variable y, then let $f_i(\varphi) = x_i \leftrightarrow y$.
- If $\varphi_i = \mathsf{X}\varphi_j$, then let $f_i(\varphi) = x_i \leftrightarrow \mathsf{X}x_j$.
- If $\varphi_i = \mathsf{F}\varphi_j$, then let $f_i(\varphi) = x_i \leftrightarrow \mathsf{F}x_j$.
- If $\varphi_i = \mathsf{G}\varphi_j$, then let $f_i(\varphi) = x_i \leftrightarrow \mathsf{G}x_j$.
- If $\varphi_i = \varphi_j \mathsf{U} \varphi_\ell$, then let $f_i(\varphi) = x_i \leftrightarrow x_j \mathsf{U} x_\ell$.
- If $\varphi_i = \varphi_j \mathsf{S} \varphi_\ell$, then let $f_i(\varphi) = x_i \leftrightarrow x_j \mathsf{S} x_\ell$.
- If $\varphi_i = g(\varphi_{i_1}, \dots, \varphi_{i_n})$ for some $g \in B$, then let $f_i(\varphi) = x_i \leftrightarrow h(x_{i_1}, \dots, x_{i_n})$, where h is a formula using only $\wedge$, $\vee$, and $\neg$, representing the function g.

Such a formula h always exists with constant length, because the set B is fixed and does not depend on the input. Now let $f(\varphi) = x_1 \wedge \bigwedge_{i=1}^{k} (\mathsf{G} f_i(\varphi) \wedge \neg(\mathit{true}\, \mathsf{S} \neg f_i(\varphi)))$ for case (1) and $f(\varphi) = x_1 \wedge \bigwedge_{i=1}^{k} \mathsf{G} f_i(\varphi)$ for case (2). The part $\mathsf{G} f_i(\varphi)$ makes sure that $f_i(\varphi)$ holds in every future state of the structure and $\neg(\mathit{true}\, \mathsf{S} \neg f_i(\varphi)))$ does the same for the past states of the structure. Additionally we consider $x \leftrightarrow y$ as a shorthand for $(x \wedge y) \vee (\neg x \wedge \neg y)$. For case (1) we consider $\mathsf{F}x$ as a shorthand for $\mathit{true}\, \mathsf{U} x$ and $\mathsf{G}x$ as a shorthand for $\neg(\mathit{true}\, \mathsf{U} \neg x)$, and for case (2) we consider $\mathsf{G}x$ as a shorthand for $\neg \mathsf{F} \neg x$. Thus we have that $f(\varphi)$ is from $\mathrm{L}(\{\mathsf{U}, \mathsf{S}, \mathsf{X}\}, \{\wedge, \vee, \neg\})$ in case (1) and from $\mathrm{L}(\{\mathsf{F}\}, \{\wedge, \vee, \neg\})$ in case (2). Furthermore f is computable in logarithmic space, because the length of f_i is polynomial and neither $\leftrightarrow$ nor the formulae h occur nested. In order to show that f is the reduction we are looking for, we still need to prove that φ is satisfiable if and only if $f(\varphi)$ is satisfiable. Assume an arbitrary structure S, such that $S, s_i \vDash f(\varphi)$ for some s_i. We first prove by induction on the structure of the formula that x_i holds if and only if φ_i holds in every state s of S (for (1)) respectively in every state which lies in the future of s_i (for (2)). Therefore for (1) let s be an arbitrary state and for (2) let s be an arbitrary state in the future of s_i. Thus by construction of $f(\varphi)$ the formulae $f_p(\varphi)$ hold at s for all $1 \leq p \leq k$. Then the following holds:

- If $\varphi_p = y$ for a variable y, then $f_p(\varphi) = x_p \leftrightarrow y$ and trivially $S, s \vDash x_p$ iff $S, s \vDash y$.
- If $\varphi_p = \mathsf{X}\varphi_j$, then $f_p(\varphi) = x_p \leftrightarrow \mathsf{X}x_j$. Thus $S, s \vDash x_p$ iff for the successor state s' of s, we have $S, s' \vDash x_j$. By induction this is equivalent to $S, s' \vDash \varphi_j$ and therefore $S, s \vDash \varphi_p$ iff $S, s \vDash x_p$.
- The cases for the temporal operator F or G work analogously.
- If $\varphi_p = \varphi_j \mathsf{U} \varphi_\ell$, then $f_p(\varphi) = x_p \leftrightarrow x_j \mathsf{U} x_\ell$. Thus $S, s \vDash x_p$ iff there exists a state s' in the future of s, such that $S, s' \vDash x_\ell$ and in all states s_m in between (including s) $S, s_m \vDash x_j$. By induction this is equivalent to $S, s' \vDash \varphi_\ell$ and for all states in between $S, s_m \vDash \varphi_j$ and therefore $S, s \vDash \varphi_p$ iff $S, s \vDash x_p$.
- If $\varphi_p = \varphi_j \mathsf{S} \varphi_\ell$, then $f_p(\varphi) = x_p \leftrightarrow x_j \mathsf{S} x_\ell$. Thus $S, s \vDash x_p$ iff there exists a state s' in the past of s, such that $S, s' \vDash x_\ell$ and in all states s_m in between (including s) $S, s_m \vDash x_j$. By induction this is equivalent to $S, s' \vDash \varphi_\ell$ and for all states in between $S, s_m \vDash \varphi_j$ and therefore $S, s \vDash \varphi_p$ iff $S, s \vDash x_p$.
- If $\varphi_p = g(\varphi_{i_1}, \dots, \varphi_{i_n})$, then $f_p(\varphi) = x_p \leftrightarrow h(x_{i_1}, \dots, x_{i_n})$, where h is a formula using only $\wedge$, $\vee$, and $\neg$, representing the function g. Thus $S, s \vDash x_p$ iff $S, s \vDash h(x_{i_1}, \dots, x_{i_n})$. Let I be the subset of $I^n = \{i_1, \dots, i_n\}$, such that $S, s \vDash x_m$ for all $m \in I$ and $S, s \vDash \neg x_m$ for all $m \in I^n \setminus I$. By induction $S, s \vDash \varphi_m$ for all $m \in I$ and $S, s \vDash \neg\varphi_m$ for all $m \in I^n \setminus I$ and therefore $S, s \vDash h(\varphi_{i_1}, \dots, \varphi_{i_n})$. Since h represents the function g, we have that $S, s \vDash \varphi_p$ iff $S, s \vDash x_p$.

Now, assume that $f(\varphi)$ is satisfiable. Then there exists a structure $S, s_i \vDash f(\varphi)$ and thus $S, s_i \vDash x_1$. Since in every state x_j holds if and only if φ_j holds, we have that $S, s_i \vDash \varphi = \varphi_1$. For the other direction, assume that φ is satisfiable. Then there exists a structure $S, s_i \vDash \varphi = \varphi_1$. Now we can extend S by adding new variables $x_1, \dots, x_k$ in such a way, that x_j holds in a state s from S if and only if φ_j holds in that state. Call this new structure S'. Then by construction of $f(\varphi)$, we have $S', s_i \vDash f(\varphi)$, since in every state x_j holds if and only if φ_j holds. This concludes the proof of the first two cases.

We now show (3). For a formula φ in which X is the only temporal operator, let $\text{depth}_{\mathsf{X}}(\varphi)$ denote the maximal nesting degree of the X-operator in φ, which we call the X-depth of φ. It is obvious that this number is linear in the length of φ. Therefore, to show that the problem can be solved in NP, it suffices to prove the following:

(a) Such a formula φ is satisfiable if and only if there is a structure S with the sequence $(s_i)_{i\in\mathbb{N}}$ such that for every $i > \text{depth}_{\mathsf{X}}(\varphi)$, every variable in s_i is false, and $S, s_0 \models \varphi$.

(b) Given the assignments to the variables in the first $\text{depth}_{\mathsf{X}}(\varphi)$ states in the structure above, it can be verified in polynomial time if $S, s_0 \models \varphi$.

These claims immediately imply the complexity result. For the first point, it obviously suffices to show one direction. Therefore, let S be an arbitrary structure with sequence $(s_i)_{i\in\mathbb{N}}$ such that $S, s_0 \models \varphi$, and let S' be the structure with sequence $(s'_i)_{i\in\mathbb{N}}$ obtained from S as follows: For $i \leq \mathrm{depth}_{\mathsf{X}}(\varphi)$, the assignment of the variables in the state s'_i is the same as in s_i. For $i > \mathrm{depth}_{\mathsf{X}}(\varphi)$, every variable is false in s'_i. To prove claim (a) above, it suffices to prove that $S', s'_0 \models \varphi$.

To show this, we prove that for every subformula ψ of φ and every $i \leq \mathrm{depth}_{\mathsf{X}}(\varphi)$, if $\mathrm{depth}_{\mathsf{X}}(\psi) \leq \mathrm{depth}_{\mathsf{X}}(\varphi) - i$, then $S, s_i \models \psi$ if and only if $S', s'_i \models \psi$. For $i = 0$ and $\psi = \varphi$, this implies the desired result $S', s'_0 \models \varphi$.

We show the claim by induction on the formula ψ. If ψ is a variable, then, by construction, $S', s'_i \models \psi$ if and only if $S, s_i \models \psi$, since the truth assignments of s'_i and s_i are identical. Now let ψ be of the form $f(\psi_1, \ldots, \psi_n)$ for an n-ary function $f \in B$. In this case, it immediately follows that $\mathrm{depth}_{\mathsf{X}}(\psi) = \max \{\mathrm{depth}_{\mathsf{X}}(\psi_1), \ldots, \mathrm{depth}_{\mathsf{X}}(\psi_n)\}$. Because of the prerequisites, $\mathrm{depth}_{\mathsf{X}}(\psi) \leq \mathrm{depth}_{\mathsf{X}}(\varphi) - i$, and hence we know that for each $j \in \{1, \ldots, n\}$, it holds that $\mathrm{depth}_{\mathsf{X}}(\psi_j) \leq \mathrm{depth}_{\mathsf{X}}(\varphi) - i$. Therefore, we can apply the induction hypothesis to all of the ψ_j, and we know that $S, s_i \models \psi_j$ if and only if $S', s'_i \models \psi_j$. This immediately implies that $S, s_i \models \psi$ if and only if $S', s'_i \models \psi$, since f is a Boolean function.

Finally, let ψ be of the form $\mathsf{X}\xi$ for some formula ξ. Hence, $\mathrm{depth}_{\mathsf{X}}(\psi) = \mathrm{depth}_{\mathsf{X}}(\xi) + 1$. Since $\mathrm{depth}_{\mathsf{X}}(\psi) \leq \mathrm{depth}_{\mathsf{X}}(\varphi) - i$, this implies that $\mathrm{depth}_{\mathsf{X}}(\xi) \leq \mathrm{depth}_{\mathsf{X}}(\varphi) - (i+1)$. Hence, we can apply the induction hypothesis, and conclude that $S, s_{i+1} \models \xi$ if and only if $S', s'_{i+1} \models \xi$. This immediately implies that $S, s_i \models \psi$ if and only if $S', s'_i \models \psi$, and hence concludes the induction and the proof of claim (a).

For claim (b), assume that φ and the truth assignments for the first $\mathrm{depth}_{\mathsf{X}}(\varphi)$ states in the structure S are given, where all variables are assumed to be false in all further states. We can now, for each subformula ψ of φ, mark those states s_i (for $i \leq \mathrm{depth}_{\mathsf{X}}(\varphi)$) in which ψ holds. Starting with $j = 0$, consider the subformulae of X-depth j. The question if a formula of X-depth j holds at a given state can easily be decided when this is known for all formulae of lower X-depth. For $j = 0$, this can be decided easily, since the subformulae of X-depth 0 are exactly the propositional subformulae, and for these, each state can be considered separately. Additionally, observe that in the structure S, all states beyond the first $\mathrm{depth}_{\mathsf{X}}(\varphi)$ states satisfy exactly the same set of subformulae of φ, hence only $\mathrm{depth}_{\mathsf{X}}(\varphi) + 1$ many states need to be considered. □

The following two theorems show that the case in which our Boolean operators are able to express the function $x \wedge \overline{y}$, leads to PSPACE-complete problems in the same cases as for the full set of Boolean operators. This function already played an important role in the classification result from [Lew79], where it also marked the "jump" in complexity from polynomial time to NP-complete.

Theorem 3.2. *Let B be a finite set of Boolean functions such that $\mathrm{S}_1 \subseteq [B]$. Then $\mathrm{SAT}(\{\mathsf{G},\mathsf{X}\}, B)$ and $\mathrm{SAT}(\{\mathsf{F},\mathsf{X}\}, B)$ are PSPACE-complete.*

Proof. Since we can express F using G and negation, Theorem 2.1 implies that $\mathrm{SAT}(\{\mathsf{G},\mathsf{X}\}, \{\wedge,\vee,\neg\})$ and $\mathrm{SAT}(\{\mathsf{F},\mathsf{X}\}, \{\wedge,\vee,\neg\})$ are PSPACE-hard. Now, let φ be a formula in which only temporal operators G and X, or F and X, and the Boolean connectives $\wedge, \vee$, and $\neg$ appear. Let $B' = B \cup \{1\}$. The complete structure of Post's lattice [BCRV03] shows that $[B'] = \mathrm{BF}$. Now we can rewrite φ as a B'-formula with the same temporal operators appearing. Due to Lemma 2.2, we can express the crucial operators $\wedge, \vee, \neg$ with short B'-formulae, i.e., formulae in which every relevant variable occurs only once. Therefore, this transformation can be performed in polynomial time. Now, in the B'-representation of φ, we exchange every occurrence of 1 with a new variable t, and call the result φ', which is a B-formula. It is obvious that φ is satisfiable if and only if the B-formula $\varphi' \wedge t \wedge \mathsf{G}t$ is. Since $B \supseteq \mathrm{S}_1$, we can express the occurring conjunctions using operators from B (since these are a constant number of conjunctions, we do not need to worry about needing long B-formulae to express conjunction). This finishes the proof for $\mathrm{SAT}(\{\mathsf{G}, \mathsf{X}\}, B)$. For the problem $\mathrm{SAT}(\{\mathsf{F}, \mathsf{X}\}, B)$, observe that the function $g(x, y) = x \wedge \overline{y}$ generates the clone S_1, and therefore there is some B-formula equivalent to g. Now observe that the formula $t \wedge \overline{\mathsf{F}(t \wedge \overline{\mathsf{X}t})} = g(t, \mathsf{F}(g(t, \mathsf{X}t)))$ is equivalent to $\mathsf{G}t$. Since this formula is independent of the input formula φ, this can be computed in polynomial time, and therefore this formula can be used to express $\varphi' \wedge t \wedge \mathsf{G}t$ in the same way as in the first case. Additionally, observe that if the operator F appears in the original formula φ, then a subformula $\mathsf{F}\psi$ can be expressed as $(1\mathsf{U}\psi)$. Hence we conclude from Theorem (2) that $\mathrm{SAT}(\{\mathsf{U}, \mathsf{X}\}, \mathrm{BF})$ is PSPACE-complete. □

The construction in the proof of Theorem 3.2 does not seem to be applicable to the languages with U and/or S, as it requires a way to express $\mathsf{G}t$ using these operators. Hence, proving the desired completeness result requires significantly more work.

Theorem 3.3.

(1) Let B be a finite set of Boolean functions with $[B] = \mathrm{BF}$. Then $\mathrm{SAT}(\{\mathsf{S}\}, B)$ is PSPACE-complete.

(2) Let B be a finite set of Boolean functions with $\mathrm{S}_1 \subseteq [B]$. Then $\mathrm{SAT}(\{\mathsf{S}\}, B)$ and $\mathrm{SAT}(\{\mathsf{U}\}, B)$ are PSPACE-complete.

Proof. Since the membership for PSPACE is shown in Lemma 3.1 we only need to show hardness.

(1) We first prove an auxiliary proposition.

Claim. Let $\varphi_1, \dots, \varphi_n$ be satisfiable propositional formulae such that $\varphi_i \rightarrow \neg\varphi_j$ is true for all $i, j \in \{1, \dots, n\}$ with $i \neq j$. Then the formula

$$\varphi = \varphi_1 \wedge (\varphi_1 \mathsf{S}(\varphi_2 \mathsf{S}(\dots \mathsf{S}(\varphi_{n-1} \mathsf{S} \varphi_n) \dots))) \wedge ((\dots((\varphi_1 \mathsf{S} \varphi_2)\mathsf{S}\varphi_3)\mathsf{S} \dots)\mathsf{S}\varphi_n)$$

is satisfiable and every structure S that satisfies φ in a state s_m fulfills the following property: there exist natural numbers $0 = a_0 < a_1 < \dots < a_n \leq m+1$ such that $m - a_i < j \leq m - a_{i-1}$ implies $S, s_j \vDash \varphi_i$ for every $i \in \{1 \dots, n\}$.

Proof. Clearly φ is satisfiable: since all formulae φ_i are satisfiable we can find a structure S such that $S, s_0 \vDash \varphi_n, S, s_1 \vDash \varphi_{n-1}, \dots, S, s_{n-1} \vDash \varphi_1$. One can verify that S satisfies φ in s_{n-1}.

Let S be a structure that satisfies φ in a state s_m. Since $\varphi_i \rightarrow \neg\varphi_j$ is true for all $i, j \in \{1, \dots, n\}$ with $i \neq j$, in every state only one of the formulae φ_i can be satisfied by S. Therefore and since $S, s_m \vDash \varphi_1 \mathsf{S}(\varphi_2 \mathsf{S}(\dots \mathsf{S}(\varphi_{n-1} \mathsf{S} \varphi_n)\dots))$ holds, there are natural numbers $0 = a_0 \leq a_1 \leq \cdots \leq a_{n-1} < a_n \leq m+1$ such that $m - a_i < l \leq m - a_{i-1}$ implies $S, s_l \vDash \varphi_i$ for every $i \in \{1 \dots, n\}$. Since $S, s_m \vDash \varphi_1$, it holds that $a_1 > 0$. Because $S, s_m \vDash (\dots((\varphi_1 \mathsf{S} \varphi_2)\mathsf{S}\varphi_3)\mathsf{S} \dots)\mathsf{S}\varphi_n$ we conclude that $a_1 < \cdots < a_{n-1}$, which proves the claim. □

To show hardness for PSPACE, we reduce QBF, which is PSPACE-complete due to [Sto77], to SAT$(\{\mathsf{S}\}, B)$. Let $\psi = Q_1 x_1 \dots Q_n x_n \varphi$ for some propositional $\{\wedge, \vee, \neg\}$-formula φ with variables $x_1, \dots, x_n$ and for quantifiers $Q_1, \dots, Q_n \in \{\forall, \exists\}$.
Let $I_\forall = \{p_1, \dots, p_k\} = \{i \in \{1, \dots, n\} \mid Q_i = \forall\}$ and $I_\exists = \{q_1, \dots, q_l\} = \{i \in \{1, \dots, n\} \mid Q_i = \exists\}$ such that $p_1 < \cdots < p_k$ and $q_1 < \cdots < q_l$.
We construct a temporal formula $\psi' \in \mathrm{L}(\{\mathsf{S}\}, B)$ such that ψ is valid if and only if ψ' is satisfiable. Let $t_0, \dots, t_n, u_0, \dots, u_n$ be new variables. We construct subformulae of ψ' which we will combine afterwards.

$$\alpha = u_0 \wedge \overline{t_0} \wedge (u_0 \wedge \overline{t_0})\mathsf{S}((\overline{u_0} \wedge \overline{t_0})\mathsf{S}(\overline{u_0} \wedge t_0))) \wedge (((u_0 \wedge \overline{t_0})\mathsf{S}(\overline{u_0} \wedge \overline{t_0}))\mathsf{S}(\overline{u_0} \wedge t_0))$$

$$\begin{aligned}
\beta^1[i] = {} & \\
& (u_{i-1} \wedge \overline{t_{i-1}} \wedge u_i \wedge \overline{t_i} \wedge \overline{x_i})\mathsf{S} \\
& \quad ((\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge \overline{t_i} \wedge \overline{x_i})\mathsf{S} \\
& \quad\quad ((\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge t_i \wedge \overline{x_i})\mathsf{S} \\
& \quad\quad\quad ((\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge u_i \wedge \overline{t_i} \wedge x_i)\mathsf{S} \\
& \quad\quad\quad\quad ((\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge \overline{t_i} \wedge x_i)\mathsf{S} \\
& \quad\quad\quad\quad\quad (\overline{u_{i-1}} \wedge t_{i-1} \wedge \overline{u_i} \wedge t_i \wedge x_i)))))
\end{aligned}$$

$$\begin{aligned}
\beta^2[i] = {} & \\
& (((((u_{i-1} \wedge \overline{t_{i-1}} \wedge u_i \wedge \overline{t_i} \wedge \overline{x_i}) \\
& \quad \mathsf{S}(\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge \overline{t_i} \wedge \overline{x_i})) \\
& \quad\quad \mathsf{S}(\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge t_i \wedge \overline{x_i})) \\
& \quad\quad\quad \mathsf{S}(\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge u_i \wedge \overline{t_i} \wedge x_i)) \\
& \quad\quad\quad\quad \mathsf{S}(\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge \overline{t_i} \wedge x_i)) \\
& \quad\quad\quad\quad\quad \mathsf{S}(\overline{u_{i-1}} \wedge t_{i-1} \wedge \overline{u_i} \wedge t_i \wedge x_i)
\end{aligned}$$

$$\begin{aligned}
\gamma^1[i] = {} & (u_{i-1} \wedge \overline{t_{i-1}} \wedge u_i \wedge \overline{t_i} \wedge \overline{x_i})\mathsf{S} \\
& \quad ((\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge \overline{t_i} \wedge \overline{x_i})\mathsf{S} \\
& \quad\quad ((\overline{u_{i-1}} \wedge t_{i-1} \wedge \overline{u_i} \wedge t_i \wedge \overline{x_i})))
\end{aligned}$$

$$\begin{aligned}
\gamma^2[i] = {} & (u_{i-1} \wedge \overline{t_{i-1}} \wedge u_i \wedge \overline{t_i} \wedge x_i)\mathsf{S} \\
& \quad ((\overline{u_{i-1}} \wedge \overline{t_{i-1}} \wedge \overline{u_i} \wedge \overline{t_i} \wedge x_i)\mathsf{S} \\
& \quad\quad ((\overline{u_{i-1}} \wedge t_{i-1} \wedge \overline{u_i} \wedge t_i \wedge x_i)))
\end{aligned}$$

Since $[B] = \mathrm{BF}$ and due to Lemma 2.2, there exist short B-representations for $\wedge, \vee$ and $\neg$. Let φ' be a copy of φ that uses these representations instead of $\wedge, \vee$ and $\neg$. Due to the short representations, φ' can be computed in polynomial time. We now define the formula ψ', which constitutes the reduction.

$$\psi' = \alpha \ \wedge \bigwedge_{i \in I_\forall}((\beta^1[i] \wedge \beta^2[i])\mathsf{S}\, t_0) \ \wedge \bigwedge_{i = \in I_\exists}((\gamma^1[i] \vee \gamma^2[i])\mathsf{S}\, t_0) \ \wedge (\varphi' \mathsf{S}\, t_0)$$

Since the operators $\wedge, \vee$, and $\neg$ are nested only in constant depth we can use their B-representations without increasing the size of ψ' significantly.
Assume that S is a structure that satisfies ψ' in a state s_m. We prove by induction over n that there are natural numbers $0 = a_0 < \cdots < a_{3(2^k)} \leq m+1$ and for every $q \in I_\exists$ a function $\sigma_q : \{0,1\}^{q-1} \rightarrow \{0,1\}$ such that S satisfies the following property: if $m - a_i < j \leq m - a_{i-1}$, then

1. $S, s_j \models x_{p_h}$ iff $\lceil \frac{i}{3(2^{k-h})} \rceil$ is even
2. $S, s_j \models x_{q_h}$ iff $\sigma_{q_h}(a_1 \ldots, a_{q_h-1}) = 1$ where -$a_d = 1$ if $x_d \in \xi(s_j)$ and $a_d = 0$ otherwise
3. $S, s_j \models t_0$ iff $i = 3(2^k)$
4. $S, s_j \models t_{p_h}$ iff $i = c \cdot 3(2^{k-h})$ for some $c \in \mathbb{N}$
5. $S, s_j \models t_{q_h}$ iff $S, s_j \models t_{p_h-1}$
6. $S, s_j \models u_0$ iff $i = 1$
7. $S, s_j \models u_{p_h}$ iff $i = c \cdot 3(2^{k-h}) + 1$ for some $c \in \mathbb{N}$
8. $S, s_j \models u_{q_h}$ iff $S, s_j \models u_{p_h-1}$

Note that due to point 1 for every possible assignment π to $\{x_{p_1}, \ldots, x_{p_k}\}$ there is a $j \in \{m - a_{3(2^k)} + 1, \ldots, m\}$ such that $S, s_j \models x_{p_i}$ if and only if $\pi(x_{p_i}) = 1$. This is the main feature of the construction. The other variables t_i and u_i are necessary to ensure this condition.
For $n = 0$ it holds that $\psi' = \alpha \wedge (\varphi' \mathsf{S}\, t_0)$. Since α satisfies the prerequisites of the auxiliary proposition, there exist natural numbers $0 = a_0 < a_1 < a_2 < a_3 \leq m + 1$ such that

- $m - a_1 < j \leq m - a_0$ implies $S, s_j \models u_0 \wedge \overline{t_0}$
- $m - a_2 < j \leq m - a_1$ implies $S, s_j \models \overline{u_0} \wedge \overline{t_0}$
- $m - a_3 < j \leq m - a_2$ implies $S, s_j \models \overline{u_0} \wedge t_0$

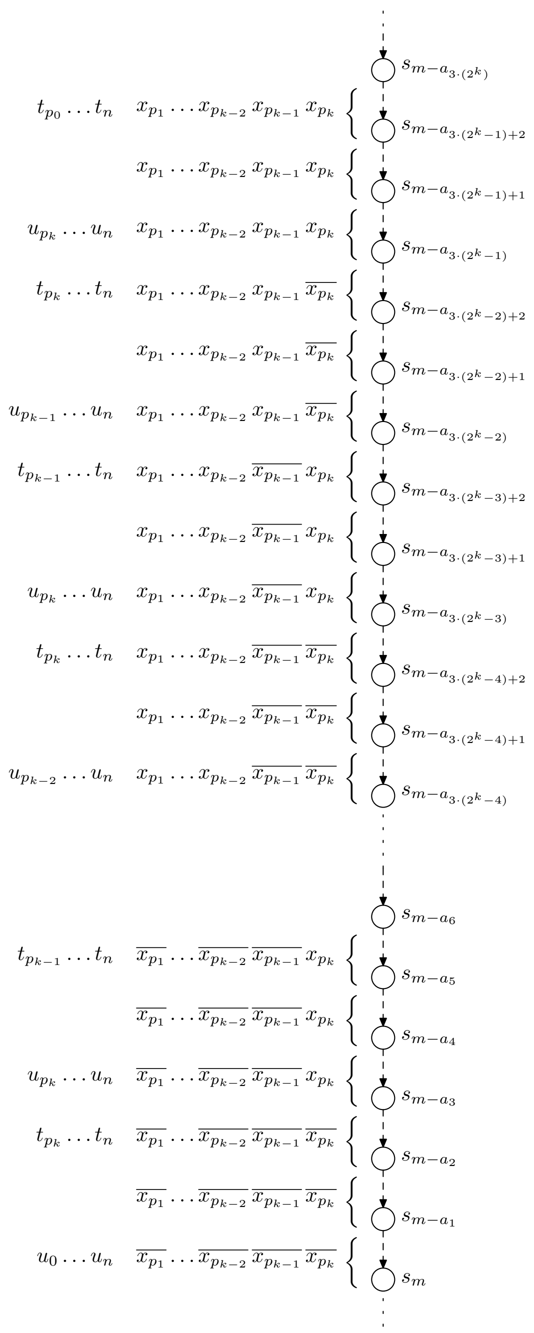
The only occurring variables are u_0 and t_0 and it is easy to see that the above property holds for both.
For the induction step assume that $n > 1$ and the claim holds for $n - 1$. There are two cases to consider:

Case 1: $Q_n = \forall$. That means

$$\begin{aligned}\psi' = \alpha \wedge \bigwedge_{i \in I_\forall \setminus \{n\}} ((\beta^1[i] \wedge \beta^2[i]) \mathsf{S}\, t_0) \wedge \bigwedge_{i \in I_\exists} ((\gamma^1[i] \vee \gamma^2[i]) \mathsf{S}\, t_0) \wedge (\varphi' \mathsf{S}\, t_0) \\ \wedge ((\beta^1[n] \wedge \beta^2[n]) \mathsf{S}\, t_0)\end{aligned}$$

It follows that there are natural numbers $0 = a_0 < \cdots < a_{3(2^{k-1})} \leq m + 1$ and for every $q \in I_\exists$ a function $\sigma_q : \{0,1\}^{q-1} \to \{0,1\}$ such that S fulfills the properties of the claim (note that the subformula $(\psi' \mathsf{S}\, t_0)$ is not necessary for our argument). Since $S, s_m \models (\beta^1[n] \wedge \beta^2[n]) \mathsf{S}\, t_0$ and for $m - a_{3(2^{k-1})} < j \leq m$ it holds that $S, s_j \models t_0$ if and only if $j \leq m - a_{3(2^{k-1})-1}$, we have $S, s_j \models \beta^1[n] \wedge \beta^2[n]$ for every $m - a_{3(2^{k-1})-1} < j \leq m$. Let $i = c \cdot 3$ for some $c \in \mathbb{N}$, then it holds that $m - a_{i+1} < j \leq m - a_i$ implies $S, s_j \models u_{n-1}$ which means that for these states s_j it holds that $S, s_j \models u_{n-1} \wedge \overline{t_{n-1}} \wedge u_n \wedge \overline{t_n} \wedge x_n$. Due to our proposition there are natural numbers $0 = b_0^i < b_1^i < \cdots < b_6^i \leq a_i + 1$ such that

- $a_i - b_1^i < j \leq a_i - b_0^i$ implies $S, s_j \models u_{n-1} \wedge \overline{t_{n-1}} \wedge u_n \wedge \overline{t_n} \wedge \overline{x_n}$
- $a_i - b_2^i < j \leq a_i - b_1^i$ implies $S, s_j \models \overline{u_{n-1}} \wedge \overline{t_{n-1}} \wedge \overline{u_n} \wedge \overline{t_n} \wedge \overline{x_n}$
- $a_i - b_3^i < j \leq a_i - b_2^i$ implies $S, s_j \models \overline{u_{n-1}} \wedge \overline{t_{n-1}} \wedge \overline{u_n} \wedge t_n \wedge \overline{x_n}$
- $a_i - b_4^i < j \leq a_i - b_3^i$ implies $S, s_j \models \overline{u_{n-1}} \wedge \overline{t_{n-1}} \wedge u_n \wedge \overline{t_n} \wedge x_n$
- $a_i - b_5^i < j \leq a_i - b_4^i$ implies $S, s_j \models \overline{u_{n-1}} \wedge \overline{t_{n-1}} \wedge \overline{u_n} \wedge \overline{t_n} \wedge x_n$
- $a_i - b_6^i < j \leq a_i - b_5^i$ implies $S, s_j \models \overline{u_{n-1}} \wedge t_{n-1} \wedge \overline{u_n} \wedge t_n \wedge x_n$



$t_{p_0} \dots t_n$
$x_{p_1} \dots x_{p_{k-2}}\, x_{p_{k-1}}\, x_{p_k}$
$s_{m-a_{3\cdot(2^k)}}$
$s_{m-a_{3\cdot(2^k-1)+2}}$
$s_{m-a_{3\cdot(2^k-1)+1}}$
$u_{p_k} \dots u_n$
$s_{m-a_{3\cdot(2^k-1)}}$
$t_{p_k} \dots t_n$
$x_{p_1} \dots x_{p_{k-2}}\, x_{p_{k-1}}\, \overline{x_{p_k}}$
$s_{m-a_{3\cdot(2^k-2)+2}}$
$s_{m-a_{3\cdot(2^k-2)+1}}$
$u_{p_{k-1}} \dots u_n$
$s_{m-a_{3\cdot(2^k-2)}}$
$t_{p_{k-1}} \dots t_n$
$x_{p_1} \dots x_{p_{k-2}}\, \overline{x_{p_{k-1}}}\, x_{p_k}$
$s_{m-a_{3\cdot(2^k-3)+2}}$
$s_{m-a_{3\cdot(2^k-3)+1}}$
$s_{m-a_{3\cdot(2^k-3)}}$
$x_{p_1} \dots x_{p_{k-2}}\, \overline{x_{p_{k-1}}}\, \overline{x_{p_k}}$
$s_{m-a_{3\cdot(2^k-4)+2}}$
$s_{m-a_{3\cdot(2^k-4)+1}}$
$u_{p_{k-2}} \dots u_n$
$s_{m-a_{3\cdot(2^k-4)}}$
s_{m-a_6}
$\overline{x_{p_1}} \dots \overline{x_{p_{k-2}}}\, \overline{x_{p_{k-1}}}\, x_{p_k}$
s_{m-a_5}
s_{m-a_4}
s_{m-a_3}
$\overline{x_{p_1}} \dots \overline{x_{p_{k-2}}}\, \overline{x_{p_{k-1}}}\, \overline{x_{p_k}}$
s_{m-a_2}
s_{m-a_1}
$u_0 \dots u_n$
s_m

Fig. 2. Structure for the proof of Theorem 3.3

The nearest state before s_{m-a_i} that satisfies $\overline{u_{n-1}}$ is $s_{m-a_{i+1}}$ and the nearest state before s_{m-a_i} that satisfies t_{n-1} is $s_{m-a_{i+2}}$, therefore it holds that $b^i_1 = a_{i+1} - a_i$ and $b^i_5 = a_{i+2} - a_i$. By denoting $b^i_j + a_i$ with c_{2i+j} we define natural numbers $c_0, \dots, c_{3(2^k)}$ for which it can be verified that they fulfill the claim.
Case 2: $Q_n = \exists$. In this case we have

$$\psi' = \alpha \wedge \bigwedge_{i \in I_\forall} ((\beta^1[i] \wedge \beta^2[i]) \mathsf{S}\, t_0) \wedge \bigwedge_{i \in I_\exists \setminus \{n\}} ((\gamma^1[i] \vee \gamma^2[i]) \mathsf{S}\, t_0) \wedge (\varphi' \mathsf{S}\, t_0)$$
$$\wedge ((\gamma^1[n] \vee \gamma^2[n]) \mathsf{S}\, t_0).$$

Because of the induction hypothesis there are natural numbers $0 = a_0 < a_1 < \dots < a_{3(2^k)} \leq m+1$ such that the required properties are satisfied. Analogously to the first case $S, s_j \vDash \gamma^1[i] \vee \gamma^2[i]$ is true for every $m - a_{3(2^k)} < j \leq m$. Let $i = c \cdot 3$, then for $m - a_{i+1} < j \leq m - a_i$ it holds that $S, s_j \vDash u_{n-1} \wedge \overline{t_{n-1}} \wedge u_n \wedge \overline{t_n} \wedge x_n$ or $S, s_j \vDash u_{n-1} \wedge \overline{t_{n-1}} \wedge u_n \wedge \overline{t_n} \wedge \overline{x_n}$, because $S, s_j \vDash u_{n-1}$. For $m - a_{i+2} < j \leq m - a_{i+1}$ we have that $S, s_j \vDash \overline{u_{n-1}} \wedge \overline{t_{n-1}} \wedge \overline{u_n} \wedge \overline{t_n} \wedge x_n$ or $S, s_j \vDash \overline{u_{i-n}} \wedge \overline{t_{i-n}} \wedge \overline{u_n} \wedge \overline{t_n} \wedge \overline{x_n}$ and for $m - a_{i+3} < j \leq m - a_{i+2}$ it must hold $S, s_j \vDash \overline{u_{n-1}} \wedge t_{n-1} \wedge \overline{u_n} \wedge t_n \wedge x_n$ or $S, s_j \vDash \overline{u_{n-1}} \wedge t_{n-1} \wedge \overline{u_n} \wedge t_n \wedge \overline{x_n}$. If $S, s_{a_i} \vDash \gamma^1[n]$, then in all these states $\overline{x_n}$ is satisfied; if $S, s_{a_i} \vDash \gamma^2[n]$, then x_n is. Therefore with σ_n defined by $\sigma_n(d_1, \dots, d_{n-1}) = 1$ if and only if $S, s_{3(d_1 2^{n-2} + \dots + d_{n-1} 2^0)} \vDash \gamma^2[n]$, the induction is complete, because the binary numbers correspond to the assignments to the $\forall$-quantified variables.
Note that for a structure that satisfies ψ' with the above notation, $S, s_j \vDash \varphi$ holds for every $m - a_{3(2^k)} < j \leq m$, since $\varphi' \mathsf{S}\, t_0$ is a conjunct of ψ'.
Now assume that ψ' is satisfiable in a state s_m of a structure S. This is if and only if for every $q \in I_\exists$ there is a function $\sigma_q : \{0,1\}^{q-1} \to \{0,1\}$ such that S fulfills the above property. Hence each possible assignment J to the $\forall$-quantified variables $\{x_{p_1}, \dots, x_{p_k}\}$ can be extended to an assignment to $\{x_1, \dots, x_n\}$ by $J(x_{q_i}) = \sigma_{q_i}(J(x_1), \dots, J(x_{q_i - 1}))$ which is equivalent to the validity of ψ.

(2) The complete structure of Post's lattice [BCRV03] shows that $[B \cup \{1\}] = \mathrm{BF}$. We modify the above reduction and show that $\mathrm{QBF} \leq^{\mathrm{P}}_m \mathrm{SAT}(\{\mathsf{S}\}, B)$. Let ψ be a QBF-instance and let ψ' be defined as in the proof of (1) where φ' is a copy of φ using short $B \cup \{1\}$-representations for $\{\wedge, \vee, \neg\}$ which exist due to Lemma 2.2. Let χ be a copy of ψ' modified by adding the new variable t to every conjunctive clause and first replacing every occurrence of $\vee$ by its $B \cup \{1\}$-representation and then every occurrence of the constant 1 by t. Hence, χ is a formula from $\mathrm{L}(\{\mathsf{S}\}, B \cup \{\wedge\})$. Since $\mathrm{S}_1 \supset \mathrm{E}_2$ we can express $\wedge$ with functions from B, that means t does not appear in the $\wedge$-representations and therefore their behavior is independent from t. Note that $\wedge$ is nested only in constant depth, hence the size of χ is polynomial in the size of ψ. Because t appears in every conjunctive clause, t must be true in every relevant state of a satisfying structure. Thus we have simulated the constant 1 with t. Therefore, if ψ' is satisfied by a structure S then χ is satisfied by S with additionally every t assigned true in every state. Hence, ψ' can be satisfied if and only if χ can and that proves the reduction.
We can prove PSPACE-hardness for $\mathrm{SAT}(\{\mathsf{U}\}, B)$ with an analogous construction. $\square$

In the following, we use the result from Lewis [Lew79] and the previously established upper bounds to obtain NP-completeness results:

Proposition 3.4. *Let B be a finite set of Boolean functions such that $\mathrm{S}_1 \subseteq [B]$. Then $\mathrm{SAT}(\{\mathsf{F}\}, B)$, $\mathrm{SAT}(\{\mathsf{G}\}, B)$, $\mathrm{SAT}(\{\mathsf{F}, \mathsf{G}\}, B)$, and $\mathrm{SAT}(\{\mathsf{X}\}, B)$ are NP-complete.*

Proof. Trivially, it holds that $\mathrm{SAT}(\emptyset, B) \leq_m^{\log} \mathrm{SAT}(M, B)$ for each set M of temporal operators, and $\mathrm{SAT}(\emptyset, B)$ is NP-complete due to [Lew79]. The upper bound follows from Theorem (1) and Lemma 3.1.

$\square$

3.2 Polynomial time results

The following theorem shows that for some sets B of Boolean functions, there is a satisfying model for every temporal B-formula over any set of temporal operators. These are the cases where $B \subseteq \mathrm{R}_1$, or $B \subseteq \mathrm{D}$. In the first case, every propositional formula over these operators is satisfied by the assignment giving the value 1 to all appearing variables. In the second case, every propositional B-formula describes a self-dual function. For such a formula it holds in particular that if it is not satisfied by the all-zero assignment, then it is satisfied by the all-one assignment. Hence, such formulae are always satisfiable. It is easy to see that this is also true for temporal formulae involving these propositional operators.

Theorem 3.5.

(1) *Let B be a finite subset of R_1. Then every formula φ from $\mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$ is satisfiable.*

(2) *Let B be a finite subset of D. Then every formula φ from $\mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$ is satisfiable.*

Proof.

(1) Since R_1 is the class of 1-reproducing Boolean functions, any $\psi \in \mathrm{R}_1$ is true under the assignment that makes every propositional variable in ψ true. If we apply this fact to formulae $\varphi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, then it is easy to see that any such formula φ is true in every state of a structure S_φ where the assignment of every state is V_φ.

(2) We show by induction on the operators that this holds for all formulae. Let S^1 (S^0) denote the structure where the assignment of every state is V_φ ($\emptyset$, resp.) and let s^1 (s^0, resp.) be the first state. We claim that $\varphi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$ is satisfied by S^1 iff φ is not satisfied by S^0. If φ is purely propositional the claim holds trivially. We now have to look at the following cases:
   - $\varphi = \mathsf{F}\varphi_1$: Assume the claim holds for φ_1. Since for all states s in S^1 the submodel starting at s are isomorphic, obviously φ is satisfied by S^1 iff $\mathsf{F}\varphi$ is satisfied by S^1 and the same argument also holds for S^0. Thus $S^0, s^0 \not\models \varphi$ iff $S^1, s^1 \models \varphi$.
   - $\varphi = \mathsf{G}\varphi_1$: This works analogously to F.

- $\varphi = \mathsf{X}\varphi_1$: This also works analogously to F.
- $\varphi = \varphi_1 \mathsf{U} \varphi_2$: Assume the claim holds for φ_2. Then $S^0, s^0 \not\models \varphi$ iff $S^0, s^0 \not\models \varphi_2$ iff $S^1, s^1 \models \varphi_2$ iff $S^1, s^1 \models \varphi$.
- $\varphi = \varphi_1 \mathsf{S} \varphi_2$: This works analogously to U.
- $\varphi = f(\varphi_1, \ldots, \varphi_n)$, such that f is a self-dual function from B: Assume the claim holds for φ_i, $1 \leq i \leq n$, i.e., $S^1, s^1 \models \varphi_i$ iff $S^0, s^0 \not\models \varphi_i$. Then $S^1, s^1 \not\models f(\varphi_1, \ldots, \varphi_n)$ implies $S^0, s^0 \models f(\varphi_1, \ldots, \varphi_n)$ and $S^1, s^1 \models f(\varphi_1, \ldots, \varphi_n)$ implies $S^0, s^0 \not\models f(\varphi_1, \ldots, \varphi_n)$.

□

The following two theorems prove that satisfiability for formulae with any combination of modal operators, but only very restricted Boolean operators (i.e., negation and constants in the first case and only disjunction, conjunction, and constants in the second case), is always easy to decide.

Theorem 3.6. *Let B be a finite subset of* N. *Then* $\mathrm{SAT}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$ *can be decided in polynomial time.*

Proof. We give a recursive polynomial-time algorithm deciding the following question: Given a formula φ built from propositional negation, constants, variables and arbitrary temporal operators, which of the following three cases occurs: φ is unsatisfiable, φ is a tautology, or φ is not equivalent to a constant function. We also show that in the latter case, φ is equivalent to a formula using only the above operators in which no constant appears. We will call these formulae temporal $\neg$-formulae.

We give inductive criteria for these cases. Obviously, a constant c is constant, and a variable is not, and can be written in the way defined above. The formula $\neg\varphi$ is equivalent to the constant c if and only if φ is equivalent to $\neg c$, otherwise it is equivalent to a temporal $\neg$-formula. If $\varphi = \mathsf{F}\varphi_1$, $\varphi = \mathsf{G}\varphi_1$, or $\varphi = \mathsf{X}\varphi_1$, then φ is equivalent to a constant c if and only if φ_1 is equivalent to c : Obviously $\mathsf{F}c \equiv \mathsf{G}c \equiv \mathsf{X}c \equiv c$ for a constant. On the other hand, if φ_1 is not equivalent to a constant, then due to induction, it is equivalent to a temporal $\neg$-formula. Hence, $\mathsf{F}\varphi_1$, $\mathsf{G}\varphi_1$ and $\mathsf{X}\varphi_1$ are equivalent to temporal $\neg$-formulae as well, and due to the proof of Theorem 3.5. (2), these formulae are not equivalent to constants. Hence, if φ_1 is not equivalent to a constant, then φ is not equivalent to a constant either, and can be written as a temporal $\neg$-formula.

Now, let $\varphi = \varphi_1 \mathsf{U} \varphi_2$. If φ_2 is a tautology, i.e., equivalent to the constant 1, then, by the definition of U, φ is a tautology as well. Similarly, if φ_2 is equivalent to the constant 0, then so is φ. Now assume that φ_2 is not constant. Then, by induction, φ_2 is equivalent to a temporal $\neg$-formula. If φ_1 is equivalent to the constant 0, then $\varphi_1 \mathsf{U} \varphi_2$ is equivalent to φ_2, and if φ_1 is equivalent to 1, then $\varphi_1 \mathsf{U} \varphi_2$ is equivalent to $\mathsf{F}\varphi_2$. If φ_1 is not equivalent to a constant, then, by induction, it can be written as a temporal $\neg$-formula, and obviously, this also holds for $\varphi_1 \mathsf{U} \varphi_2$. Again due to the proof of Theorem 3.5. (2), it follows that the entire formula φ is not equivalent to a constant.

For the operator S, a similar argument can be made: Consider the formula $\varphi_1 \mathsf{S} \varphi_2$. If φ_2 is a constant, then obviously the formula $\varphi_1 \mathsf{S} \varphi_2$ is equivalent to the same constant. If φ_1 is the constant 0, then $\varphi_1 \mathsf{S} \varphi_2$ is equivalent to φ_2, and if φ_1 is the constant 1, then $\varphi_1 \mathsf{S} \varphi_2$ is equivalent to "φ_2 was true at one point in the past." If φ_2 is not a constant, then this is equivalent to $\neg\varphi_2 \mathsf{S} \varphi_2$, and thus this can be written as a temporal $\neg$-formula as well. As above, this formula is not equivalent to a constant. Now if both φ_1 and φ_2 are not equivalent to a constant function, then, by induction, both can be written as temporal $\neg$-formulae, and then $\varphi_1 \mathsf{S} \varphi_2$ can be written as such a formula as well. In particular, with another application of the proof for Theorem 3.5 (2), $\varphi_1 \mathsf{S} \varphi_2$ is not equivalent to a constant.

This gives us a recursive algorithm deciding whether φ is a constant, and if it is, which constant is equivalent to φ. The polynomial-time computable function A_N is defined as follows: On input φ, $A_N(\varphi) = c \in \{0, 1\}$ if φ is equivalent to the constant c, and $A_N(\varphi)$ is the symbol `NOCONSTANT` if φ is not equivalent to a constant.

The function can be computed as follows: $A_N(c)$ is defined as c. For a variable x, $A_N(x)$ is the symbol `NOCONSTANT`. On input $\mathsf{X}\varphi$, $\mathsf{G}\varphi$, or $\mathsf{F}\varphi$, the algorithm returns $A_N(\varphi)$. On input $\varphi_1 \mathsf{U} \varphi_2$, if φ_2 is a constant c, then $A_N(\varphi_1 \mathsf{U} \varphi_2) = c$. Otherwise, if φ_1 is equivalent to 0, then return $A_N(\varphi_2)$, and if φ_1 is equivalent to 1, return $A_N(\mathsf{F}\varphi_2)$. If neither φ_1 nor φ_2 are constant, then return the symbol `NOCONSTANT`. Similarly, on input $\varphi_1 \mathsf{S} \varphi_2$, if φ_2 is a constant c, then $A_N(\varphi_1 \mathsf{S} \varphi_2) = c$. Otherwise, if φ_1 is the constant 0, then $A_N(\varphi_1 \mathsf{S} \varphi_2) = A_N(\varphi_2)$, and if φ_1 is the constant 1, and φ_2 is not a constant, then $A_N(\varphi_1 \mathsf{S} \varphi_2)$ is defined as the symbol `NOCONSTANT`. If φ_1 and φ_2 both are not a constant, then $A_N(\varphi_1 \mathsf{S} \varphi_2)$ is again defined as the symbol `NOCONSTANT`. The function A_N can obviously be computed in polynomial time, since there is at most one recursive call for each operator symbol in φ.

By the argument above, this algorithm correctly determines if φ is equivalent to the constant 0 or the constant 1. In particular, it determines if a given formula is satisfiable. $\square$

Theorem 3.7. *Let B be a finite subset of* M. *Then* SAT$(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$ *can be decided in polynomial time.*

Proof. Remember that M is the clone of all monotone functions. Let φ be an arbitrary formula from L$(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$. The following algorithm decides whether φ is satisfiable.

```
Algorithm LTL-M-SAT
  repeat
    Replace all propositional sub-formulae that are unsatisfiable by 0
    Replace all sub-formulae $\mathsf{F}0$ by 0
    Replace all sub-formulae $\mathsf{G}0$ by 0
    Replace all sub-formulae $\mathsf{X}0$ by 0
    Replace all sub-formulae $0\mathsf{U}\psi$ by $\psi$
    Replace all sub-formulae $\psi\mathsf{U}0$ by 0
    Replace all sub-formulae $0\mathsf{S}\psi$ by $\psi$
    Replace all sub-formulae $\psi\mathsf{S}0$ by 0
    Replace all sub-formulae $\psi(\varphi_1, \dots, \varphi_k)$ by 0 if $\psi \in B$ and $\psi(\varphi'_1, \dots, \varphi'_k)$,
    where $\varphi'_i = 0$ if $\varphi_i = 0$ and $\varphi'_i = 1$ otherwise, is not true
  until there are no changes anymore
  if $\varphi = 0$ then
    return "unsatisfiable"
  else
    return "satisfiable"
  end if
```

Since checking satisfiability of propositional B-formulae is in P (a B-formula φ is satisfiable iff $\varphi(1, \dots, 1) = 1$) and there are at most as many replacements as there are operators in φ, LTL-M-Sat runs in polynomial time.

We prove that LTL-M-Sat is correct. If φ is satisfiable, then LTL-M-Sat returns "satisfiable." This is because all replacements in LTL-M-Sat do not affect satisfiability, so it follows that every formula LTL-M-Sat decides to be unsatisfiable is unsatisfiable. For the converse direction, let $\varphi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$ be such that LTL-M-Sat returns "satisfiable" and let φ' be the formula generated by LTL-M-Sat in its REPEAT loop. We show by induction on the structure of φ that $S, s_0 \vDash \varphi$, where $S = (s, V_\varphi, \xi)$ is the structure in which every variable is true in every state, i.e., $\xi(s_i) = V_\varphi$ for every $i \in \mathbb{N}$.

(1) If φ is a variable, it is satisfied in S, s_0 trivially.

(2) If $\varphi = \mathsf{F}\psi$ for a formula $\psi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, let ψ' be the formula generated in the REPEAT loop when performing LTL-M-Sat on ψ. Assume that $\psi' = 0$. Since every subformula replaced in ψ by LTL-M-Sat will be replaced in φ, too, it holds that $\mathsf{F}\psi$ will be replaced by $\mathsf{F}0$ and that will be replaced by 0. It follows that $\varphi' = 0$, but then LTL-M-Sat would return "unsatisfiable." Thus, $\psi' \neq 0$, that means LTL-M-Sat returns "satisfiable" when performed on ψ. By induction it follows that $S, s_0 \vDash \psi$ and therefore $S, s_0 \vDash \varphi$ holds as well.

(3) If $\varphi = \mathsf{G}\psi$ for a formula $\psi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, we can use exactly the same arguments as in 2.

(4) If $\varphi = \mathsf{X}\psi$ for a formula $\psi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, we can use the same arguments as in 2.

(5) If $\varphi = \psi_1 \mathsf{U} \psi_2$ for formulae $\psi_1, \psi_2 \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, we have that ψ_2 cannot be replaced by 0 (otherwise φ would be replaced by 0 and LTL-M-Sat would return "unsatisfiable"). So by induction it follows that $S, s_0 \vDash \psi_2$. Hence, it holds that $S, s_0 \vDash \varphi$ as well.

(6) If $\varphi = \psi_1 \mathsf{S} \psi_2$ for formulae $\psi_1, \psi_2 \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, we can use the same arguments as for 5.

(7) If $\varphi = \psi(\varphi_1, \dots, \varphi_k)$ for formulae $\psi \in B$ and $\varphi_i \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}, B)$, for all $i = 1, \dots, k$, let $\varphi'_1, \dots, \varphi'_k$ be the replacements of $\varphi_1, \dots, \varphi_k$. By induction it follows that $S, s_0 \vDash \varphi_i$ if and only if $\varphi'_i \neq 0$ for any $i \in \{1, \dots, k\}$. Since $\varphi' \neq 0$ and because of the last replacement rule, $S, s_0 \vDash \varphi$.

$\square$

Finally, we show that satisfiability for formulae that have X as a modal operator and the *xor* function $\oplus$ as a propositional operator is in P. This is true because functions described by these formulae have a high degree of symmetry.

Theorem 3.8. *Let B be a finite subset of* L. *Then* $\mathrm{SAT}(\{\mathsf{X}\}, B)$ *can be decided in polynomial time.*

Proof. First observe that any function from L is of the form $f(x_1, \dots, x_n) = x_{i_1} \oplus \cdots \oplus x_{i_k} \oplus c$, where the x_{i_j} are pairwise different variables from the set $\{x_1, \dots, x_n\}$, and c is either 0 or 1. Therefore, it is obvious that temporal B-formulae can be rewritten using only the connectors $\oplus$ and the constant 1 (the 0 can be omitted in the representation above). Hence, we can assume that the set B contains only the functions $\oplus$ and 1. Now observe that any formula φ from $\mathrm{L}(\{\mathsf{X}\}, \{\oplus, 1\})$ can be written as

$$\varphi = \mathsf{X}\psi_1 \oplus \cdots \oplus \mathsf{X}\psi_k \oplus \psi,$$

where ψ is a propositional formula. This representation can be computed in polynomial time, and we can determine in polynomial time whether ψ is a constant function.

If ψ is not a constant function, then φ is satisfiable: Let $S = (s, V_\varphi, \xi)$ be an arbitrary structure. If φ is not satisfied at s_0, then we can "switch over" the current truth value of ψ, thus achieving that one more (or one less) of the arguments of the outermost *xor* function becomes true. For this purpose, we change the assignment of the propositional variables at s_0 in such a way that the new assignment satisfies ψ if and only if the old assignment does not. Since this change does not affect the validity of the $\mathsf{X}\psi_i$ parts, φ holds at s_0 with the new assignment.

Now, if ψ is constant, this trick does not work. Instead, let

$$\varphi' = \psi_1 \oplus \cdots \oplus \psi_k.$$

Observe that in this case φ is satisfiable if and only if ψ is the constant 0 and φ' is satisfiable, or if ψ is the constant 1 and φ' is no tautology; and that φ is a tautology if and only if ψ is the constant 0 and φ' is a tautology, or ψ is the constant 1 and φ' is not satisfiable. Thus we have an iterative algorithm deciding $\mathrm{SAT}(\{\mathsf{X}\}, \{\oplus, 1\})$, since for a propositional B-formula, these questions can be efficiently decided. $\square$

4 Conclusion

We have almost completely classified the computational complexity of satisfiability for LTL with respect to the sets of propositional and temporal operators permitted. The only case left open is the one in which only propositional operators constructed from the binary *xor* function (and, perhaps, constants) are allowed. This case has already turned out to be difficult to handle — and hence was left open — in [BHSS06] for modal satisfiability under *restricted* frames classes. The difficulty here and in [BHSS06] is reflexivity, i. e., the property that the formula $\mathsf{F}\varphi$ is satisfied at some state if φ is satisfied at *the same* state. This does not allow for a separate treatment of the propositional part (without temporal operators) and the remainder of a given formula.

Our results bear an interesting resemblance to the classifications obtained in [Lew79] and in [BHSS06]. In all of these cases (except for one of the several classifications obtained in the latter), it turns out that sets of Boolean functions B which generate a clone above S_1 give rise to computationally hard problems, while other cases seem to be solvable in polynomial time. Therefore, in a precise sense, it is the function represented by the formula $x \wedge \overline{y}$ which turns problems in this context computationally intractable. These hardness results seem to indicate that $x \wedge \overline{y}$ and other functions which generate clones above S_1 have properties that make computational problems hard, and this notion of hardness is to a large extent independent of the actual problem considered.

It is worth knowing whether our results are transferable to what is called "determination of truth" in [SC85] — the model checking problem. In the case of LTL with no restrictions on the propositional operators, model checking has the same complexity as satisfiability [SC85]. We have done first steps towards a similar classification of this problem. The first partial results suggest that the behavior of model checking is not quite the same as that of satisfiability.

The results from this paper leave two open questions. Besides the unsolved xor case, it would be interesting to further classify the polynomial-time solvable cases. Further work could also examine related specification languages, such as CTL, CTL*, or hybrid temporal languages.

Acknowledgments

We thank Martin Mundhenk and the anonymous referees for their helpful comments and suggestions.

References

[BCRV03] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *SIGACT News*, 34(4):38–52, 2003.

[BHSS06] M. Bauland, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Generalized modal satisfiability. In *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 500–511. Springer, 2006.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Symposium on Theory of Computing*, pages 151–158. ACM Press, 1971.

[Dal00] V. Dalmau. *Computational Complexity of Problems over Generalized Formulas.* PhD thesis, Department de Llenguatges i Sistemes Informàtica, Universitat Politécnica de Catalunya, 2000.

[Lew79] H. Lewis. Satisfiability problems for propositional calculi. *Mathematical Systems Theory*, 13:45–53, 1979.

[Nor05] G. Nordh. A trichotomy in the complexity of propositional circumscription. In *Proceedings of the 11th International Conference on Logic for Programming*, volume 3452 of *Lecture Notes in Computer Science*, pages 257–269. Springer Verlag, 2005.

[Pip97] N. Pippenger. *Theories of Computability.* Cambridge University Press, Cambridge, 1997.

[Pnu77] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[Pos41] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[Rei01] S. Reith. *Generalized Satisfiability Problems.* PhD thesis, Fachbereich Mathematik und Informatik, Universität Würzburg, 2001.

[RV03] S. Reith and H. Vollmer. Optimal satisfiability for propositional calculi and constraint satisfaction problems. *Information and Computation*, 186(1):1–19, 2003.

[RW05] S. Reith and K. W. Wagner. The complexity of problems defined by Boolean circuits. In *Proceedings of the International Conference Mathematical Foundation of Informatics, (MFI99); World Science Publishing*, 2005.

[SC85] A. Sistla and E. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[Sch05] H. Schnoor. The complexity of the Boolean formula value problem. Technical report, Theoretical Computer Science, University of Hannover, 2005.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.