The Tractability of Model-Checking for LTL: The Good, the Bad, and the Ugly Fragments*

Michael Bauland[1], Martin Mundhenk[2], Thomas Schneider[3], Henning Schnoor[4], Ilka Schnoor[4], and Heribert Vollmer[5]

[1] Knipp GmbH, Martin-Schmeißer-Weg 9, 44227 Dortmund, Germany
Michael.Bauland@knipp.de
[2] Informatik, Friedrich-Schiller-Universität, 07737 Jena, Germany
mundhenk@cs.uni-jena.de
[3] Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK
schneider@cs.man.ac.uk
[4] Comp. Science, Rochester Institute of Technology, 102 Lomb Memorial Drive, NY 14623-5608 Rochester, USA
{hs,is}@cs.rit.edu
[5] Theoret. Informatik, Universität Hannover, Appelstr. 4, 30167 Hannover, Germany
vollmer@thi.uni-hannover.de

Abstract. In a seminal paper from 1985, Sistla and Clarke showed that the model-checking problem 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 systematically studies the model-checking problem for LTL formulae over restricted sets of propositional and temporal operators. For almost all combinations of temporal and propositional operators, we determine whether the model-checking problem is tractable (in P) or intractable (NP-hard). We then focus on the tractable cases, showing that they all are NL-complete or even logspace solvable. This leads to a surprising gap in complexity between tractable and intractable cases. It is worth noting that our analysis covers an infinite set of problems, since there are infinitely many sets of propositional operators.

1 Introduction

Linear Temporal Logic (LTL) has been proposed by Pnueli [Pnu77] as a formalism to specify properties of parallel programs and concurrent systems, as well as to reason about their behaviour. Since then, it has been widely used for these purposes. Recent developments require reasoning tasks—such as deciding satisfiability, validity, or model checking—to be performed automatically. Therefore, decidability and computational complexity of the corresponding decision problems are of great interest.

The earliest and fundamental source of complexity results for the satisfiability problem (SAT) and the model-checking problem (MC) of LTL is certainly Sistla and Clarke's paper [SC85]. They have established PSPACE-completeness of SAT and MC for LTL with the temporal operators F (eventually), G (invariantly), X (next-time), U (until), and S (since). They have also shown that these problems are NP-complete for certain restrictions of the set of temporal operators. This work was continued by Markey [Mar04]. The results of Sistla, Clarke, and Markey imply that SAT and MC for LTL and a multitude of its fragments are intractable. In fact, they do not exhibit any tractable fragment.

The fragments they consider are obtained by restricting the set of temporal operators and the use of negations. What they do not consider are arbitrary fragments of temporal *and* Boolean operators. For propositional logic, a complete analysis has been achieved by Lewis [Lew79]. He divides all infinitely many sets of Boolean operators into those with tractable (polynomial-time solvable) and intractable (NP-complete) SAT problems. A similar systematic classification has been obtained by Bauland et al. in [BSS+07] for LTL. They divide fragments of LTL—determined by arbitrary combinations of temporal and Boolean operators—into those with polynomial-time solvable, NP-complete, and PSPACE-complete SAT problems.

This paper continues the work on the MC problem for LTL. Similarly as in [BSS+07], the considered fragments are arbitrary combinations of temporal and Boolean operators. We will separate the MC problem for almost all LTL fragments into tractable (i.e., polynomial-time solvable) and intractable (i.e., NP-hard) cases. This extends the work of Sistla and Clarke, and Markey [SC85, Mar04], but in contrast to their results, we will exhibit many tractable fragments and exactly determine their computational

* Supported in part by DFG VO 630/6-1 and the Postdoc Programme of the German Academic Exchange Service (DAAD).

complexity. Surprisingly, we will see that tractable cases for model checking are even very easy—that is, NL-complete or even L-solvable. There is only one set of Boolean operators, consisting of the binary *xor*-operator, that we will have to leave open. This constellation has already proved difficult to handle in [BSS+07, BHSS06], the latter being a paper where SAT for basic modal logics has been classified in a similar way.

While the borderline between tractable and intractable fragments in [Lew79] and [BSS+07] is quite easily recognisable (SAT for fragments containing the Boolean function $f(x, y) = x \wedge \overline{y}$ is intractable, almost all others are tractable), our results for MC will exhibit a rather diffuse borderline. This will become visible in the following overview and is addressed in the Conclusion. Our most surprising intractability result is the NP-hardness of the fragment that only allows the temporal operator U and no propositional operator at all. Our most surprising tractability result is the NL-completeness of MC for the fragment that only allows the temporal operators F, G, and the binary *or*-operator. Taking into account that MC for the fragment with only F plus *and* is already NP-hard (which is a consequence from [SC85]), we would have expected the same lower bound for the "dual" fragment with only G plus *or*, but in fact we show that even the fragment with F and G and *or* is tractable. In the presence of the X-operator, the expected duality occurs: The fragment with F, X plus *and* and the one with G, X plus *or* are both NP-hard.

Table 1 gives an overview of our results. The top row refers to the sets of Boolean operators given in Definition 2.3. These seven sets of Boolean operators are all relevant cases, which is due to Post's fundamental paper [Pos41] and Lemma 2.2. Entries in bold-face type denote completeness for the given complexity class under logspace reductions. (All reductions in this paper are logspace reductions $\leq_m^{\log}$.) The entry L stands for logspace solvability. All other entries denote hardness results. Superscripts refer to the source of the corresponding result as explained in the legend.

prop. operators / temp. operators	I	N	E	V	M	L	BF
X	**NL**[11]	**NL**[11]	**NL**[13]	**NL**[12]	NP[2]	**NL**[15]	NP[S]
G	**NL**[11]	**NL**[11]	**NL**[13]	**NL**[14]	NP[2]		NP[S]
F	**NL**[11]	**NL**[11]	NP[5]	**NL**[12]	NP[2]		NP[S]
FG	**NL**[11]	**NL**[11]	NP[c]	**NL**[14]	NP[c]		NP[S]
FX	**NL**[11]	**NL**[11]	NP[c]	**NL**[12]	NP[c]		PS[T]
GX	**NL**[11]	**NL**[11]	**NL**[13]	NP[6]	PS[3]		PS[T]
FGX	**NL**[11]	**NL**[11]	NP[c]	NP[c]	PS[1]		PS[T]
S	L[16]	L[16]	L[16]	L[16]	L[16]	L[16]	L[16]
SX	NP[10]	NP[10]	NP[10]	NP[10]	NP[10]	NP[10]	NP[10]
SG	NP[8]	NP[8]	NP[8]	NP[8]	PS[4]	NP[8]	PS[4]
SF	**NL**[17]	NP[9]	NP[9]	**NL**[17]	PS[4]	NP[9]	PS[4]
SFG	NP[c]	NP[c]	NP[c]	NP[c]	PS[c]	NP[c]	PS[S]
SFX	NP[c]	NP[c]	NP[c]	NP[c]	PS[c]	NP[c]	PS[T]
SGX	NP[c]	NP[c]	NP[c]	NP[c]	PS[c]	NP[c]	PS[T]
SFGX	NP[c]	NP[c]	NP[c]	NP[c]	PS[c]	NP[c]	PS[T]
all other combinations (i.e., with U)	NP[7]	NP[c]	NP[c]	NP[c]	PS[3]	NP[c]	PS[T]

Legend.

(PS stands for PSPACE.)

1	Theorem 3.2 (1)
2	Theorem 3.2 (2)
3	Theorem 3.2 (3)
4	Theorem 3.2 (4)
5	Corollary 3.3
6	Theorem 3.4
7	Theorem 3.5
8	Theorem 3.6
9	Theorem 3.7
10	Theorem 3.8
11	Theorem 4.2
12	Theorem 4.3 (1)
13	Theorem 4.3 (2)
14	Theorem 4.4
15	Theorem 4.5
16	Theorem 4.6
17	Theorem 4.7
S	Theorem 2.1 (1)
T	Theorem 2.1 (2)
c	conclusion from surrounding results

Table 1. An overview of complexity results for the model-checking problem

This paper is organised as follows. Section 2 contains all necessary definitions and notation. In Section 3, we show NP-hardness of all intractable cases, followed by Section 4 with the NL-completeness of almost all remaining cases. We conclude in Section 5.

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 T be a set of temporal operators. A *temporal B-formula over T* is a formula φ that is built from variables, propositional connectors from B, and temporal operators from T. More formally, a temporal B-formula over T 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 T, f is an n-ary propositional operator from B and g is an m-ary temporal operator from T. 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 by $\mathrm{VAR}(\varphi)$. If $T = \{\mathsf{X}, \mathsf{F}, \mathsf{G}, \mathsf{U}, \mathsf{S}\}$ we call φ a *temporal B-formula*, and if $T = \emptyset$ we call φ a *propositional B-formula* or simply a *B-formula*. The set of all temporal B-formulae over T is denoted with $\mathrm{L}(T, B)$.

A *Kripke structure* is a triple $K = (W, R, \eta)$, where W is a finite set of states, $R \subseteq W \times W$ is a total binary relation (meaning that, for each $a \in W$ there is some $b \in W$ such that aRb)[6], and $\eta : W \to 2^{\mathrm{VAR}}$ for a set VAR of variables.

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 *path* p in K is an infinite sequence denoted as $(p_0, p_1, \ldots)$, where, for all $i \geq 0$, $p_i \in W$ and $p_i R p_{i+1}$.

For a temporal $\{\wedge, \neg\}$-formula over $\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}$ with variables from VAR, a Kripke structure $K = (W, R, \eta)$, and a path p in K, we define what it means that p^K *satisfies* φ *in* p_i ($p^K, i \vDash \varphi$): let φ_1 and φ_2 be temporal $\{\wedge, \neg\}$-formulae over $\{\mathsf{F}, \mathsf{G}, \mathsf{X}, \mathsf{U}, \mathsf{S}\}$ and let $x \in \mathrm{VAR}$ be a variable.

$$
\begin{array}{lll}
p^K, i \vDash 1 \quad \text{and} \quad p^K, i \nvDash 0 & & \\
p^K, i \vDash x & \text{iff} & x \in \eta(p_i), \\
p^K, i \vDash \varphi_1 \wedge \varphi_2 & \text{iff} & p^K, i \vDash \varphi_1 \text{ and } p^K, i \vDash \varphi_2, \\
p^K, i \vDash \neg\varphi_1 & \text{iff} & p^K, i \nvDash \varphi_1, \\
p^K, i \vDash \mathsf{F}\varphi_1 & \text{iff} & \text{there is a } j \geq i \text{ such that } p^K, j \vDash \varphi_1, \\
p^K, i \vDash \mathsf{G}\varphi_1 & \text{iff} & \text{for all } j \geq i,\ p^K, j \vDash \varphi_1, \\
p^K, i \vDash \mathsf{X}\varphi_1 & \text{iff} & p^K, i+1 \vDash \varphi_1, \\
p^K, i \vDash \varphi_1 \mathsf{U} \varphi_2 & \text{iff} & \text{there is an } \ell \geq i \text{ such that } p^K, \ell \vDash \varphi_2, \text{ and for every } i \leq j < \ell,\ p^K, j \vDash \varphi_1, \\
p^K, i \vDash \varphi_1 \mathsf{S} \varphi_2 & \text{iff} & \text{there is an } \ell \leq i \text{ such that } p^K, \ell \vDash \varphi_2, \text{ and for every } \ell < j \leq i,\ p^K, j \vDash \varphi_1.
\end{array}
$$

Since every Boolean operator can be composed from $\wedge$ and $\neg$, the above definition generalises to temporal B-formulae for arbitrary sets B of Boolean operators.

This paper examines the model-checking problem $\mathrm{MC}(T, B)$ for a finite set B of Boolean functions and a set T of temporal operators.

Problem:	$\mathrm{MC}(T, B)$
Input:	$\langle \varphi, K, a \rangle$, where $\varphi \in \mathrm{L}(T, B)$ is a formula, $K = (W, R, \eta)$ is a Kripke structure, and $a \in W$ is a state
Question:	Is there a path p in K such that $p_0 = a$ and $p^K, 0 \vDash \varphi$?

Sistla and Clarke [SC85] have established the computational complexity of the model-checking problem for temporal $\{\wedge, \vee, \neg\}$-formulae over some sets of temporal operators.

Theorem 2.1 ([SC85]).

(1) $\mathrm{MC}(\{\mathsf{F}\}, \{\wedge, \vee, \neg\})$ *is* NP*-complete.*
(2) $\mathrm{MC}(\{\mathsf{F}, \mathsf{X}\}, \{\wedge, \vee, \neg\})$, $\mathrm{MC}(\{\mathsf{U}\}, \{\wedge, \vee, \neg\})$, *and* $\mathrm{MC}(\{\mathsf{U}, \mathsf{S}, \mathsf{X}\}, \{\wedge, \vee, \neg\})$ *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, where $1 \leq k \leq n$, 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.

[6] In the strict sense, Kripke structures can have arbitrary binary relations. However, when referring to Kripke structures, we always assume their relations to be total.

The definitions of all clones as well as the full inclusion graph can be found, for example, in [BCRV03]. The following lemma implies that only clones with both constants $0, 1$ are relevant for the model-checking problem; hence we will only define those clones. Note, however, that our results will carry over to all clones.

Lemma 2.2. *Let B be a finite set of propositional functions and T be a set of temporal operators. Then* $\mathrm{MC}(T, B \cup \{0,1\}) \equiv_m^{\log} \mathrm{MC}(T, B)$.

Proof. $\mathrm{MC}(T, B) \leq_m^{\log} \mathrm{MC}(T, B \cup \{0,1\})$ is trivial. For $\mathrm{MC}(T, B \cup \{0,1\}) \leq_m^{\log} \mathrm{MC}(T, B)$ let $\langle \varphi, K, a \rangle$ be an instance of $\mathrm{MC}(T, B \cup \{0,1\})$ for a Kripke structure $K = (W, R, \eta)$ and let $\bot$ and $\top$ be two fresh variables. We define a new Kripke structure $K' = (W, R, \eta')$ where $\eta'(\alpha) = \eta(\alpha) \cup \{\top\}$ and we define φ' to be a copy of φ where every appearance of 0 is replaced by $\bot$ and every appearance of 1 by $\top$. It holds that $\langle \varphi', K', a \rangle$ is an instance of $\mathrm{MC}(T, B)$ and that $\langle \varphi, K, a \rangle \in \mathrm{MC}(T, B \cup \{0,1\})$ if and only if $\langle \varphi', K', a \rangle \in \mathrm{MC}(T, B)$. ❏

Because of Lemma 2.2 it is sufficient to look only at the clones with constants. Their definition and bases are given in the following definition, their inclusion structure in Figure 1.

Definition 2.3. *Let $\oplus$ denote the binary exclusive or. Let f be an n-ary Boolean function.*

(1) BF *is the set of all Boolean functions.*
(2) M *is the set of all monotone functions, that is, the set of all functions f where $a_1 \leq b_1, \ldots, a_n \leq b_n$ implies $f(a_1, \ldots, a_n) \leq f(b_1, \ldots, b_n)$.*
(3) L *is the set of all linear functions, that is, the set of all functions f that satisfy $f(x_1, \ldots, x_n) = c_0 \oplus (c_1 \wedge x_1) \oplus \cdots \oplus (c_n \wedge x_n)$, for constants c_i.*
(4) V *is the set of all functions f where $f(x_1, \ldots, x_n) = c_0 \vee (c_1 \wedge x_1) \vee \cdots \vee (c_n \wedge x_n)$, for constants c_i.*
(5) E *is the set of all functions f where $f(x_1, \ldots, x_n) = c_0 \wedge (c_1 \vee x_1) \wedge \cdots \wedge (c_n \vee x_n)$, for constants c_i.*
(6) N *is the set of all functions that depend on at most one variable.*
(7) I *is the set of all projections and constants.*

These five clones have the following bases.

clone	BF	M	L	V	E	N	I
base	$\{\wedge, \neg\}$	$\{\vee, \wedge, 0, 1\}$	$\{\oplus, 1\}$	$\{\vee, 1, 0\}$	$\{\wedge, 1, 0\}$	$\{\neg, 1, 0\}$	$\{0, 1\}$

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: it is NP-complete if $x \wedge \overline{y} \in [B]$, and solvable in P otherwise [Lew79]. Furthermore, Post's lattice has been applied to the equivalence problem [Rei01], to counting [RW05] and finding minimal [RV03] solutions, and to 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].

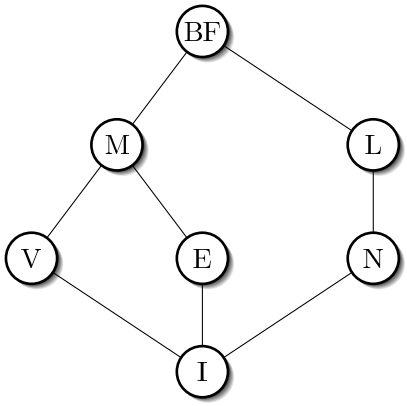

Fig. 1. Clones with constants

An important tool in restricting the length of the resulting formula in many of our reductions is the following lemma.

Lemma 2.4. *Let C be a finite set of Boolean functions such that $B \subseteq \{\wedge, \vee, \neg\}$ and $B \subseteq [C]$. Then* $\mathrm{MC}(T, B) \leq_m^{\log} \mathrm{MC}(T, C)$ *for every set T of temporal operators.*

Proof. Let $D = C \cup \{0,1\}$. From Lemmas 1.4.4 and 1.4.5 in [Sch07] we directly conclude: Let f be one of the functions *or*, *and*, and *not* such that $f \in [D]$. Let k be the arity of f. Then there is a D-formula $\varphi(x_1, \ldots, x_k)$ representing f, such that every variable occurs only once in φ. Hence $\mathrm{MC}(T, B) \leq_m^{\log} \mathrm{MC}(T, C \cup \{0,1\})$. From Lemma 2.2 follows $\mathrm{MC}(T, C \cup \{0,1\}) \leq_m^{\log} \mathrm{MC}(T, C)$. ❏

It is essential for this Lemma that $B \subseteq \{\wedge, \vee, \neg\}$. For, e.g., $B = \{\oplus\}$, it is open whether $\mathrm{MC}(T, B) \leq_m^{\log} \mathrm{MC}(T, \mathrm{BF})$. This is a reason why we cannot immediately transform upper bounds proven by Sistla and Clarke [SC85]—for example, $\mathrm{MC}(\{\mathsf{F}, \mathsf{X}\}, \{\wedge, \vee, \neg\}) \in \mathsf{PSPACE}$—to upper bounds for all finite sets of Boolean operators—i.e., it is open whether for all finite sets B of Boolean operators, $\mathrm{MC}(\{\mathsf{F}, \mathsf{X}\}, B) \in \mathsf{PSPACE}$.

3 The bad fragments: intractability results

Sistla and Clarke [SC85] and Markey [Mar04] have considered the complexity of model-checking for temporal $\{\wedge, \vee, \neg\}$-formulae restricted to atomic negation and propositional negation, respectively. We define a temporal B-formula with *propositional negation* to be a temporal B-formula where additional negations are allowed, but only in such a way that no temporal operator appears in the scope of a negation sign. In the case that negation is an element of B, a temporal B-formula with propositional negation is simply a temporal B-formula. In [SC85], *atomic negation* is considered, which restricts the use of negation even further—negation is only allowed directly for variables. We will now show that propositional negation does not make any difference for the complexity of the model checking problem. Since this obviously implies that atomic negation inherits the same complexity behaviour, we will only speak about propositional negation in the following. The proof of the following lemma is nearly identical to that of Lemma 2.2, and can be found in the Appendix.

Lemma 3.1. *Let T be a set of temporal operators, and B a finite set of Boolean functions. We use $\mathrm{MC}^+(T, B)$ to denote the model-checking problem $\mathrm{MC}(T, B)$ extended to B-formulae with propositional negation. Then $\mathrm{MC}^+(T, B) \equiv_m^{\log} \mathrm{MC}(T, B)$.*

Using Lemma 2.4, we can generalise hardness results from [SC85, Mar04] to obtain the following intractability results for model-checking.

Theorem 3.2. *Let B be a finite set of Boolean functions such that $\mathrm{M} \subseteq [B]$. Then*

(1) $\mathrm{MC}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}\}, B)$ *is* PSPACE*-hard.*
(2) $\mathrm{MC}(\{\mathsf{F}\}, B)$*,* $\mathrm{MC}(\{\mathsf{G}\}, B)$*, and* $\mathrm{MC}(\{\mathsf{X}\}, B)$ *are* NP*-hard.*
(3) $\mathrm{MC}(\{\mathsf{U}\}, B)$ *and* $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, B)$ *are* PSPACE*-hard.*
(4) $\mathrm{MC}(\{\mathsf{S}, \mathsf{G}\}, B)$ *and* $\mathrm{MC}(\{\mathsf{S}, \mathsf{F}\}, B)$ *are* PSPACE*-hard.*

In Theorem 3.5 in [SC85] it is shown that $\mathrm{MC}(\{\mathsf{F}\}, \{\wedge, \vee, \neg\})$ is NP-hard. In fact, Sistla and Clarke give a reduction from 3SAT to $\mathrm{MC}(\{\mathsf{F}\}, \{\wedge\})$. The result for arbitrary bases B generating a clone above E follows from Lemma 2.4.

Corollary 3.3. *Let B be a finite set of Boolean functions such that $\mathrm{E} \subseteq [B]$. Then $\mathrm{MC}(\{\mathsf{F}\}, B)$ is NP-hard.*

The model-checking problem for temporal $\{\wedge, \vee\}$-formulae over $\{\mathsf{G}, \mathsf{X}\}$ is PSPACE-complete (Theorem 3.2(3) due to [Mar04]). The Boolean operators $\{\wedge, \vee\}$ are a basis of M, the class of monotone Boolean formulae. What happens for fragments of M? In Theorem 4.3 we will show that $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \mathrm{E})$ is NL-complete, i.e., the model-checking problem for temporal $\{\wedge\}$-formulae over $\{\mathsf{G}, \mathsf{X}\}$ is very simple. We will now show that switching from $\wedge$ to $\vee$ makes the problem intractable. As notation, we use $\mathrm{LIT}(\varphi)$ to denote the literals obtained from variables that appear in φ.

Theorem 3.4. *Let B be a finite set of Boolean functions such that $\mathrm{V} \subseteq [B]$. Then $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, B)$ is NP-hard.*

Proof. By Lemma 2.4, it suffices to give a reduction from 3SAT to $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \{\vee\})$. A formula ψ in 3CNF is mapped to an instance $\langle \psi', K(\psi), q_1 \rangle$ of $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \{\vee\})$ as follows: Let $\psi = C_1 \wedge \ldots \wedge C_m$ consist of m clauses, and $n = |\mathrm{VAR}(\psi)|$ variables. The Kripke structure $K(\psi)$ has states $Q = \{q_1, \ldots, q_m\}$ containing one state for every clause, a sequence of states $P = \{l^j \mid l \in \mathrm{LIT}(\psi), 0 \leq j \leq m-1\}$ for every literal, and a final sink state s. That is, the set of states is $Q \cup P \cup \{s\}$. The variables of $K(\psi)$ are $b_1, \ldots, b_m, c$. Variable b_a is assigned *true* in a state l_i^0 iff literal l_i is contained in clause C_a. In all other states, every b_i is *false*. Variable c is assigned *true* in all states in $P \cup \{s\}$.

The relation between the states is $E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$ as follows. It starts with the path $q_1, \ldots, q_m$: $E_1 = \{(q_i, q_{i+1}) \mid i = 1, 2, \ldots, m-1\}$. q_m has an edge to x_1^0 and an edge to $\overline{x}_1^0$: $E_2 = \{(q_m, x_1^0), (q_m, \overline{x}_1^0)\}$. Each l_i^0 is the starting point of a path $l_i^0, l_i^1, \ldots, l_i^{m-1}$: $E_3 = \{(l_i^j, l_i^{j+1}) \mid l_i \in \mathrm{LIT}(\psi), j = 0, 1, \ldots, m-2\}$.

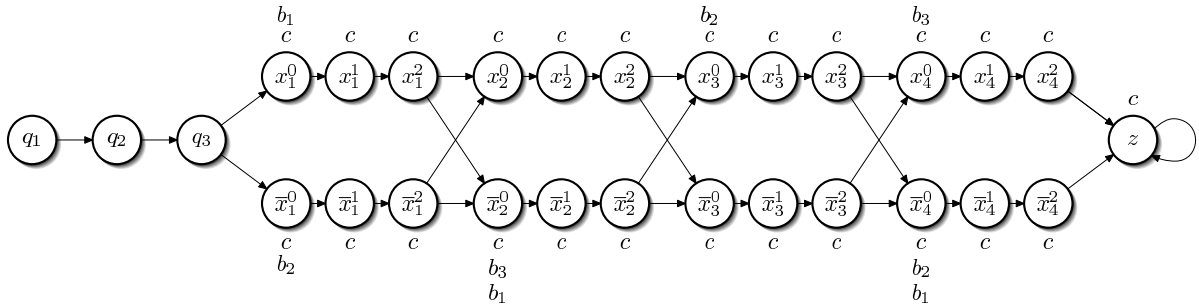

Fig. 2. The Kripke structure $K(\psi_0)$ for $\psi_0 = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_4)$.

Each endpoint of these paths has both the literals with the next index resp. the final sink state as neighbours: $E_4 = \{(l_i^{m-1}, l_{i+1}^0) \mid i = 1, 2, \ldots, n-1, l_i \in \mathrm{LIT}(\psi)\} \cup \{(x_n^{m-1}, s), (\overline{x}_n^{m-1}, s)\}$. The final sink state s has an edge to s itself, $E_5 = \{(s, s)\}$.

Figure 2 shows an example for a formula ψ_0 and the Kripke structure $K(\psi_0)$. Notice that every path in such a Kripke structure $K(\psi)$ corresponds to an assignment to the variables in ψ. A path corresponds to a satisfying assignment iff for every b_i the path contains a state that b_i is assigned to. We are now going to construct a formula ψ' to express this property. If we were allowed to use the $\wedge$ in ψ', this would be easy. But, the formula ψ' consists only of operators G, X, $\vee$, and of variables $b_1, \ldots, b_m, c$. In order to define ψ', we use formulae φ_i and ψ'_i defined as follows. For $i = 1, 2, \ldots, m$ define

$$\varphi_i = \bigvee_{k=1,2,\ldots,n} \mathsf{X}^{k \cdot m - (i-1)} b_i \; .$$

Intuitively, φ_i says that b_i is satisfied in a state in distance d, where $d \equiv m-(i-1) \pmod m$. The state q_j is the only state in Q where φ_j can hold. Every path p in $K(\psi)$ has the form $p = (q_1, q_2, \ldots, q_m, l_1^0, \ldots, l_n^{m-1}, s, s, \ldots)$. Every state except for s appears at most once in p. For the sake of simplicity, we therefore can use $p^{K(\psi)}, q_i \vDash \alpha$ instead of $p^{K(\psi)}, i-1 \vDash \alpha$ (for $i = 1, 2, \ldots, m$), and $p^{K(\psi)}, l_i^j \vDash \alpha$ instead of $p^{K(\psi)}, m + (i-1) \cdot m + j \vDash \alpha$. We use for a path $p = (q_1, q_2, \ldots)$ in $K(\psi)$ and $1 \leq i \leq m$ the notation $p^{K(\psi)}, q_i \vDash \alpha$ instead of $p^{K(\psi)}, i-1 \vDash \alpha$.

Claim 1. For every path p in $K(\psi)$ and $1 \leq i, j \leq m$ holds: If $p^{K(\psi)}, q_i \vDash \varphi_j$, then $i = j$.

The formulae ψ'_i are defined inductively for $i = m+1, \ldots, 2, 1$ as follows (as before, we can use $\vee$ in our construction):

$$\psi'_{m+1} = c \qquad \text{and} \qquad \psi'_i = \mathsf{G}\left(\varphi_i \vee \mathsf{G}\psi'_{i+1}\right) \text{ (for } m \geq i \geq 1) \; .$$

Finally, $\psi' = \psi'_1$.

It is clear that the reduction function $\psi \mapsto \langle \psi', K(\psi), q_1 \rangle$ can be computed in logarithmic space. It remains to prove the correctness of the reduction. Using Claim 1, we make the following observation.

Claim 2. For every path $p = (q_1, q_2, \ldots)$ in $K(\psi)$ and $i = 1, 2, \ldots, m$ holds:
$$p^{K(\psi)}, q_i \vDash \psi'_i \quad \text{if and only if} \quad \text{for } j = i, i+1, \ldots, m \text{ holds } p^{K(\psi)}, q_j \vDash \varphi_j.$$

For a path p in $K(\psi)$, let $\mathcal{A}_p$ be the corresponding assignment for ψ. It is clear that $p^{K(\psi)}, q_i \vDash \varphi_i$ if and only if $\mathcal{A}_p$ satisfies clause C_i of ψ. Using Claim 2, it follows that $p^{K(\psi)}, q_1 \vDash \psi'$ if and only if $\mathcal{A}_p$ satisfies all clauses of ψ, i.e., $\mathcal{A}_p$ satisfies ψ. Using the one-to-one correspondence between paths in $K(\psi)$ and assignments to the variables of ψ we get $\psi \in$ 3SAT if and only if $\langle \psi', K(\psi), q_1 \rangle \in \mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \{\vee\})$. ❑

From [SC85] it follows that $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \mathrm{V})$ is in PSPACE. It remains open whether $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \mathrm{V})$ or $\mathrm{MC}(\{\mathsf{G}, \mathsf{X}\}, \mathrm{M})$ have an upper bound below PSPACE.

Next, we consider formulae with the until-operator or the since-operator. We first show that using the until-operator makes model-checking intractable.

Theorem 3.5. *Let B be a finite set of Boolean functions. Then* $\mathrm{MC}(\{\mathsf{U}\}, B)$ *is* NP*-hard.*

Proof. We give a reduction from 3SAT to $\mathrm{MC}(\{\mathsf{U}\}, \emptyset)$. This means, that we do not need any Boolean operators in the temporal formula over $\{\mathsf{U}\}$ to which a 3SAT instance is mapped. Let $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a 3CNF formula consisting of m clauses and n variables. The structure $K(\psi)$ has states $\{q_1, \ldots, q_m\} \cup$

$\mathrm{LIT}(\psi) \cup \{s\}$, with initial state q_1. The assignment for state q_i is $\{a_1, \ldots, a_i\}$ (for $i = 1, 2, \ldots, m$), and for state l_i it is $\{a_1, \ldots, a_m\} \cup \{b_j \mid \text{Literal } l_i \in \mathrm{LIT}(\psi) \text{ appears in clause } C_j\}$. In state s, no variable is assigned true. The relation between the states is as follows. Each q_i $(i = 1, 2, \ldots, m-1)$ has an edge to q_{i+1}, q_m has edges to x_1 and to $\overline{x}_1$, each l_i $(i = 1, 2, \ldots, n-1)$ has edges to x_{i+1} and to $\overline{x}_{i+1}$, and x_n and $\overline{x}_n$ have an edge to s. s has an edge to s only. Figure 3 gives an example. The following facts are easy to

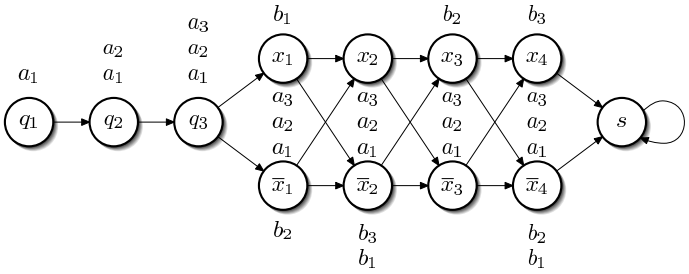

Fig. 3. Structure $K(\psi)$ for $\psi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_4)$

verify for any path p in $K(\psi)$. For the sake of simplicity, we use for a path $p = (q_1, q_2, \ldots)$ in $K(\psi)$ and $1 \leq i \leq m$ the notation $p^{K(\psi)}, q_i \vDash \alpha$ instead of $p^{K(\psi)}, i-1 \vDash \alpha$.

Fact 1 For $1 \leq j < i \leq m$ holds: $p^{K(\psi)}, q_j \nvDash a_i \mathsf{U} b_i$.
Fact 2 For $1 \leq i \leq m$ holds: $\exists t : p^{K(\psi)}, t \vDash a_i \mathsf{U} b_i$ iff $p^{K(\psi)}, q_i \vDash a_i \mathsf{U} b_i$.

The formulae $\varphi_0, \varphi_1, \ldots$ are defined inductively as follows.

$$\varphi_0 \;=\; 1 \qquad \text{and} \qquad \varphi_{i+1} \;=\; (\varphi_i \mathsf{U} (a_{i+1} \mathsf{U} b_{i+1})).$$

The reduction from 3SAT to $\mathrm{MC}(\{\mathsf{U}\}, \emptyset)$ is performed by the mapping $\psi \mapsto (\varphi_m, K(\psi), q_1)$, where ψ is a 3CNF-formula with m clauses. This reduction can evidently be performed in logarithmic space. To prove its correctness, we use the following claim.

Claim 3. Let $K(\psi)$ be constructed from a formula ψ with m clauses, and let p be a path in $K(\psi)$. For $j = 1, 2, \ldots, m$ it holds that $p^{K(\psi)}, q_1 \vDash \varphi_j$ if and only if $p^{K(\psi)}, q_1 \vDash \varphi_{j-1}$ and $p^{K(\psi)}, q_j \vDash a_j \mathsf{U} b_j$.

We have a one-to-one correspondence between paths in $K(\psi)$ and assignments to variables of ψ. For a path p we will denote the corresponding assignment by $\mathcal{A}_p$. Using Claim 3, it is easy to see that the following properties are equivalent.

1. $\mathcal{A}_p$ is a satisfying assignment for ψ.
2. Path p in $K(\psi)$ contains for every $i = 1, 2, \ldots, m$ a state with assignment b_i.
3. $p^{K(\psi)}, q_i \vDash a_i \mathsf{U} b_i$ for $i = 1, 2, \ldots, m$.
4. $p^{K(\psi)}, q_1 \vDash \varphi_m$.

This concludes the proof that $\psi \in 3\mathrm{SAT}$ if and only if $\langle \varphi_m, K(\psi), q_1 \rangle \in \mathrm{MC}(\{\mathsf{U}\}, \emptyset)$. ❑

Although the until-operator and the since-operator appear to be similar, model-checking for formulae that use the since-operator as only operator is as simple as for formulae without temporal operators — see Theorem 4.6. The reason is that the since-operator has no use at the beginning of a path of states, where no past exists. It needs other temporal operators that are able to enforce to visit a state on a path that has a past.

Theorem 3.6. *Let B be a finite set of Boolean functions. Then* $\mathrm{MC}(\{\mathsf{G}, \mathsf{S}\}, B)$ *is* NP*-hard.*

Proof. We give a reduction from 3SAT to $\mathrm{MC}(\{\mathsf{G}, \mathsf{S}\}, \emptyset)$ that is similar to that in the proof of Theorem 3.5 for $\mathrm{MC}(\{\mathsf{U}\}, \emptyset)$. Let ψ be an instance of 3SAT, and let $K(\psi)$ be the structure as in the proof of Theorem 3.5. From $K(\psi) = (W, R, \eta)$ we obtain the structure $H(\varphi) = (W', R', \eta')$ as follows. First, we add a new state t, i.e., $W' = W \cup \{t\}$. Second, replace R by its inverse $R^{-1} = \{(v, u) \mid (u, v) \in R\}$ from which the loop at state s is removed. The state s has in-degree 0 and will be seen as initial state of $H(\varphi)$. The new state t will be used as sink state. Therefore, we add the arcs (q_1, t) and (t, t). This results in $R' = (R^{-1} - \{(s, s)\}) \cup \{(q_1, t), (t, t)\}$. Finally, we add a new variable e that is *true* only in state t, and a variable d that is *true* in states $\mathrm{LIT}(\psi) \cup \{s\}$. For all other variables, η' is the same as η. (Figure 4 shows an example.)

The formulae $\varphi_1^m, \varphi_2^m, \ldots, \varphi_{m+1}^m$ are defined inductively as follows.

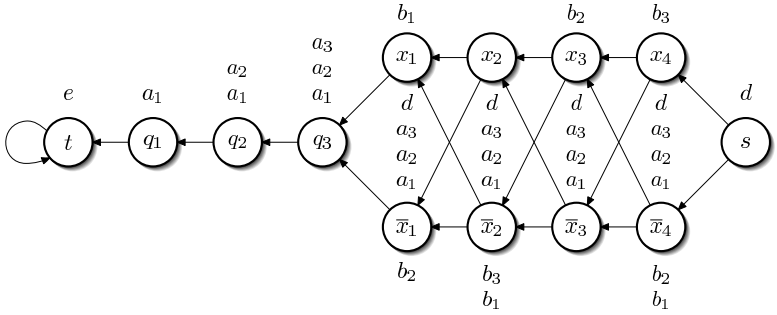

Fig. 4. Structure $H(\psi)$ for $\psi = (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee x_4)$

$$\varphi^m_{m+1} \;=\; d \qquad \text{and} \qquad \varphi^m_i \;=\; \big((a_i \mathsf{S} b_i)\;\mathsf{S}\,\varphi^m_{i+1}\big) \text{ for } i = 1, 2, \ldots, m.$$

The reduction from 3SAT to MC($\{\mathsf{G},\mathsf{S}\},\emptyset$) is performed by the mapping $\psi \mapsto \langle \mathsf{G}(e\mathsf{S}\varphi^m_1), H(\psi), s\rangle$, where ψ is a 3CNF-formula with m clauses. This reduction can evidently be performed in logarithmic space. To prove its correctness, we use the following claim. Every path $p = (s, l_n, \ldots, l_1, q_m, \ldots, q_1, t, t, \ldots)$ in $H(\psi)$ that begins in state s corresponds to an assignment $\mathcal{A}_p = \{l_1, \ldots, l_n\}$ to the variables in ψ, that sets all literals to *true* that appear on p. For the sake of simplicity, we use the notation $p^{H(\psi)}, q_i \vDash \alpha$ instead of $p^{H(\psi)}, n + m - i + 1 \vDash \alpha$.

Claim 4. Let $H(\psi)$ be constructed from a formula $\psi = C_1 \wedge \ldots \wedge C_m$ with m clauses, and let $p = (s, l_n, \ldots, l_1, q_m, \ldots, q_1, t, t, \ldots)$ be a path in $H(\psi)$. For $j = 1, 2, \ldots, m$ it holds that

$$p^{H(\psi)}, q_j \vDash \varphi^m_j \quad \text{if and only if} \quad \text{the assignment } \mathcal{A}_p \text{ satisfies clauses } C_j, \ldots, C_m.$$

Finally, let ψ be a 3CNF formula, and consider a path $p = (s, l_n, \ldots, l_1, q_m, \ldots, q_1, t, t, \ldots)$ in $H(\psi)$. On the first $n + 1$ states of p, the variable d holds. Therefore, φ^m_1 and henceforth $e\mathsf{S}\varphi^m_1$ is satisfied in all these states. On the m following states $q_m, \ldots, q_1$, neither d nor e holds. Notice that $p^{H(\psi)}, q_i \vDash \varphi^m_i$ iff $p^{H(\psi)}, q_i \vDash \varphi^m_{i-1}$ (for $i = 2, 3, \ldots, m$). By Claim 4, φ^m_1 and henceforth $e\mathsf{S}\varphi^m_1$ is satisfied in all these states iff $\mathcal{A}_p$ satisfies ψ. On the remaining states, only the variable e holds. Hence, $e\mathsf{S}\varphi^m_1$ is satisfied in all the latter states iff $\mathcal{A}_p$ satisfies ψ. Concluding, it follows that $p^{H(\psi)}, 0 \vDash \mathsf{G}(e\mathsf{S}\varphi^m_1)$ iff $\mathcal{A}_p$ satisfies ψ. Since for every assignment to ψ the structure $H(\psi)$ contains a corresponding path, the correctness of the reduction is proven. ❑

The future-operator F alone is not powerful enough to make the since-operator S NP-hard: We will show in Theorem 4.7 that MC($\{\mathsf{F},\mathsf{S}\}, B$) for $[B] \subseteq \mathrm{V}$ is NL-complete. But with the help of $\neg$ or $\wedge$, the model-checking problem for F and S becomes intractable.

Theorem 3.7. *Let B be a finite set of Boolean functions such that* $\mathrm{N} \subseteq [B]$. *Then* MC($\{\mathsf{F},\mathsf{S}\}, B$) *is* NP*-hard.*

Proof. By Lemma 2.4 it suffices to give a reduction from 3SAT to MC($\{\mathsf{F},\mathsf{S}\}, \{\neg\}$). For a 3CNF formula ψ, let $\langle \mathsf{G}(e\mathsf{S}\varphi^m_1), H(\psi), s\rangle$ be the instance of MC($\{\mathsf{G},\mathsf{S}\},\emptyset$) as described in the proof of Theorem 3.6. Using $\mathsf{G}\alpha \equiv \neg\mathsf{F}\neg\alpha$, it follows that $\mathsf{G}(e\mathsf{S}\varphi^m_1) \equiv \neg\mathsf{F}(\neg(e\mathsf{S}\varphi^m_1))$, where the latter is a N-formula over $\{\mathsf{F},\mathsf{S}\}$. The correctness of the reduction the same line as the proof of Theorem 3.6. ❑

Theorem 3.8. *Let B be a finite set of Boolean functions. Then* MC($\{\mathsf{X},\mathsf{S}\}, B$) *is* NP*-hard.*

Proof. To prove NP-hardness, we give a reduction from 3SAT to MC($\{\mathsf{X},\mathsf{S}\},\emptyset$). For a 3CNF formula ψ, let $H(\psi)$ be the structure as described in the proof of Theorem 3.6. The reduction function maps ψ to $\langle \mathsf{X}^{n+m+1}\varphi_1, H(\psi), s\rangle$. The X^{n+m+1} "moves" to state q_1 on any path in $H(\psi)$. The correctness proof follows the same line as the proof of Theorem 3.6. ❑

An upper bound better than PSPACE for the intractable cases with the until-operator or the since-operator remains open. We will now show that one canonical way to prove an NP upper bound fails, in showing that these problems do not have the "short path property", which claims that a path in the structure that fulfills the formula has length polynomial in the length of the structure and the formula. A counterexample can be found in Appendix C. Hence, it will most likely be nontrivial to obtain a better upper bound.

4 The good fragments: tractability results

This subsection is concerned with fragments of LTL that have a tractable model-checking problem. We will provide a complete analysis for these fragments by proving that model checking for all of them is NL-complete or even solvable in logarithmic space. This exhibits a surprisingly large gap in complexity between easy and hard fragments.

The following lemma establishes NL-hardness for all tractable fragments.

Lemma 4.1. *Let B be a finite set of Boolean functions. Then* $\mathrm{MC}(\{\mathsf{F}\}, B)$, $\mathrm{MC}(\{\mathsf{G}\}, B)$, *and* $\mathrm{MC}(\{\mathsf{X}\}, B)$ *are* NL*-hard.*

Proof. First consider $\mathrm{MC}(\{\mathsf{F}\}, B)$. We reduce the accessibility problem for digraphs, GAP, to $\mathrm{MC}(\{\mathsf{F}\}, \emptyset)$. The reduction is via the following logspace computable function. Given an instance $\langle G, a, b\rangle$ of GAP, where $G = (V, E)$ is a digraph and $a, b \in V$, map it to the instance $\langle \mathsf{F}y, K(G), a\rangle$ of $\mathrm{MC}(\{\mathsf{F}\}, \emptyset)$ with $K(G) = (V, E^+, \eta)$, where E^+ denotes the reflexive closure of E, and η is given by $\eta(b) = \{y\}$ and $\eta(v) = \emptyset$, for all $v \in V - \{b\}$. It is immediately clear that there is a path from a to b in G if and only if there is a path p in $K(G)$ starting from a such that $p^{K(G)}, 0 \vDash \mathsf{F}y$.

For $\mathrm{MC}(\{\mathsf{X}\}, B)$, we use an analogous reduction from GAP to $\mathrm{MC}(\{\mathsf{X}\}, \emptyset)$. Given an instance $\langle G, a, b\rangle$ of GAP, where $G = (V, E)$, transform it into the instance $\langle \mathsf{X}^{|V|}y, K(G), a\rangle$ of $\mathrm{MC}(\{\mathsf{X}\}, \emptyset)$ with the Kripke structure $K(G)$ from above. Now it is clear that there is a path from a to b in G if and only if there is a path of length $|V|$ from a to b in the reflexive structure $K(G)$, if and only if there is a path p in $K(G)$ starting from a such that $p^{K(G)}, 0 \vDash \mathsf{X}^{|V|}y$.

Now consider $\mathrm{MC}(\{\mathsf{G}\}, B)$. We reduce the following problem to $\mathrm{MC}(\{\mathsf{G}\}, \emptyset)$. Given a directed graph $G = (V, E)$ and a vertex $a \in V$, is there an infinite path in G starting at a? It is folklore that this is an NL-hard problem (see Lemma D.1 in the Appendix). Given an instance $\langle G, a\rangle$ of this problem, transform it into the instance $\langle \mathsf{G}y, K'(G), a\rangle$ of $\mathrm{MC}(\{\mathsf{G}\}, \emptyset)$, where $K'(G) = (V', E', \eta)$. Here $V' = V \dot\cup \{\tilde v \mid v \in V,\ v \text{ has no successor in } V\}$, $E' = E \cup \{(v, \tilde v),\ (\tilde v, \tilde v) \mid \tilde v \in V'\}$, $\eta(v) = y$ for all $v \in V$, and $\eta(\tilde v) = \emptyset$, for all $\tilde v \in V'$. It is immediately clear that there is an infinite path in G starting at a if and only if there is a path p in $K'(G)$ starting from a such that $p^{K'(G)}, 0 \vDash \mathsf{G}y$. ❑

It now remains to establish upper complexity bounds. Let C be one of the clones N, E, V, and L, and let B be a finite set of Boolean functions such that $[B] \subseteq C$. Whenever we want to establish NL-membership for some problem $\mathrm{MC}(\cdot, B)$, it will suffice to assume that formulae are given over one of the bases $\{\neg, 0, 1\}$, $\{\wedge, 0, 1\}$, $\{\vee, 0, 1\}$, or $\{\oplus, 0, 1\}$, respectively. This follows since these clones only contain constants, projections, and multi-ary versions of *not*, *and*, *or*, and $\oplus$, respectively.

Theorem 4.2. *Let B be a finite set of Boolean functions such that* $[B] \subseteq \mathrm{N}$. *Then* $\mathrm{MC}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}\}, B)$ *is* NL*-complete.*

Proof. The lower bound follows from Lemma 4.1. For the upper bound, first note that for an LTL formula ψ the following equivalences hold: $\mathsf{FF}\psi \equiv \mathsf{F}\psi$, $\mathsf{GG}\psi \equiv \mathsf{G}\psi$, $\mathsf{FGF}\psi \equiv \mathsf{GF}\psi$, $\mathsf{GFG}\psi \equiv \mathsf{FG}\psi$, $\mathsf{G}\psi \equiv \neg\mathsf{F}\neg\psi$, and $\mathsf{F}\psi \equiv \neg\mathsf{G}\neg\psi$. Furthermore, it is possible to interchange X and adjacent G-, F-, or $\neg$-operators without affecting satisfiability. Under these considerations, each formula $\varphi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}, \mathsf{X}\}, B)$ can be transformed without changing satisfiability into a normal form $\boldsymbol{\varphi' = \mathsf{X}^m P \sim y}$, where P is a prefix ranging over the values "empty string", F, G, FG, and GF; m is the number of occurrences of X in φ; $\sim$ is either the empty string or $\neg$; and y is a variable or a constant.

This normal form has two important properties. First, it can be represented in logarithmic space using two binary counters a and b. The counter a stores m, and b takes on values $0, \ldots, 9$ to represent each possible combination of P and $\sim$. Note that a takes on values less than $|\varphi|$, and b has a constant range. Hence both counters require at most logarithmic space. It is not necessary to store any information about y, because it can be taken from the representation of φ.

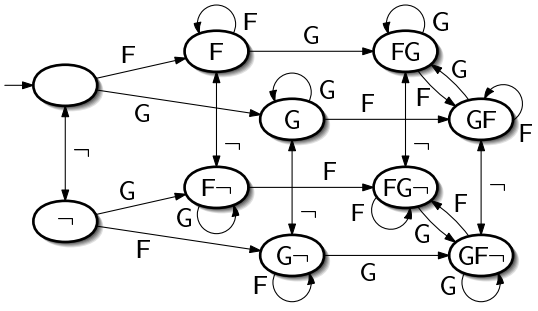

Fig. 5. An automaton that computes $P\sim$

Second, φ' can be *computed* from φ in logarithmic space. The value of a is obtained by counting the occurrences of X in φ, and b is obtained by linearly parsing φ with the automaton that is given in Figure 5, and which ignores all occurrences of X.

The state of this automaton at the end of the passage through φ determines the values of P and $\sim$ in φ. Now let φ be an $\mathrm{L}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$-formula, $K=(W,R,\eta)$ a Kripke structure and $a\in W$. If y is constant, the problem is trivial, therefore it remains to consider the case where y is a variable. According to the possible values of P and $\sim$ in φ, there are ten cases to consider. We only present the argumentation for those five in which $\sim$ is empty. (For the dual cases, kindly replace each occurrence of "$\in\eta(b)$" by "$\notin\eta(b)$".) In the following list, we assume that $m=0$. As per explanation below, this is not a significant restriction.

P is empty. Then $\langle\varphi,K,a\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$ if and only if there is a state b in K accessible from a via R such that $y\in\eta(b)$.

$P=\mathsf{F}$. In this case we have to check whether there is a state $b\in W$ that can be reached from a via R, and $y\in\eta(b)$.

$P=\mathsf{G}$. We define $W'=\{b\in W\mid y\in\eta(b)\}$ and $R'=R\cap W'\times W'$. It holds that $\langle\varphi,K,a\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$ if and only if there is some $b\in W'$ such that b is accessible from a via R' and b belongs to a cycle in R'.

$P=\mathsf{FG}$. This case can be reduced to the previous one: $\langle\varphi,K,a\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$ if and only if there is some $b\in W'$ that can be reached from a via R, and $\langle\mathsf{G}y,K,b\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$.

$P=\mathsf{GF}$. We have to check whether there exists some $b\in W$ that can be reached from a via R such that $y\in\eta(b)$ and b belongs to a cycle.

Since the questions whether there is a path from any vertex to another and whether any vertex belongs to a cycle in a directed graph can be answered in NL, all previously given procedures are NL-algorithms. The restriction $m=0$ is removed by the observation that $\langle\mathsf{X}^mP{\sim}y,K,a\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$ if and only if there exists some state b in K that is accessible from a in m R-steps such that $\langle P{\sim}y,K,b\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$. This reduces the case $m>0$ to $m=0$.

Hence we have found an NL-algorithm deciding $\mathrm{MC}(\{\mathsf{F},\mathsf{G},\mathsf{X}\},B)$: Given $\langle\varphi,K,a\rangle$, compute φ', guess a state b accessible from a in m R-steps, apply the procedure of one of the above five cases to $\langle\varphi',K,a\rangle$, and accept if the last step was successful. ❑

Theorem 4.3. *(1) Let B be a finite set of Boolean functions such that $[B]\subseteq\mathrm{V}$. Then $\mathrm{MC}(\{\mathsf{F},\mathsf{X}\},B)$ is NL-complete.*
(2) Let B be a finite set of Boolean functions such that $[B]\subseteq\mathrm{E}$. Then $\mathrm{MC}(\{\mathsf{G},\mathsf{X}\},B)$ is NL-complete.

Proof. The lower bounds follow from Lemma 4.1.

First consider the case $[B]\subseteq\mathrm{V}$. It holds that $\mathsf{F}(\psi_1\vee\cdots\vee\psi_n)\equiv\mathsf{F}\psi_1\vee\cdots\vee\mathsf{F}\psi_n$ as well as $\mathsf{XF}\varphi\equiv\mathsf{FX}\varphi$ and $\mathsf{X}(\varphi\vee\psi)\equiv\mathsf{X}\varphi\vee\mathsf{X}\psi$. Therefore, every formula $\varphi\in\mathrm{L}(\{\mathsf{F},\mathsf{X}\},B)$ can be rewritten as

$$\varphi'=\mathsf{FX}^{i_1}y_1\vee\cdots\vee\mathsf{FX}^{i_n}y_n\vee\mathsf{X}^{i_{n+1}}y_{n+1}\vee\cdots\vee\mathsf{X}^{i_m}y_m,$$

where $y_1,\ldots,y_m$ are variables or constants (note that this representation of φ can be constructed in L). Now let $\langle\varphi,K,a\rangle$ be an instance of $\mathrm{MC}(\{\mathsf{F},\mathsf{X}\},B)$, where $K=(W,R,\eta)$, and let φ be of the above form. Thus, $\langle\varphi,K,a\rangle\in\mathrm{MC}(\{\mathsf{F},\mathsf{X}\},B)$ if and only if for some $j\in\{n+1,\ldots,m\}$, there is a state $b\in W$ such that $y_j\in\eta(b)$ and b is accessible from a in exactly i_j R-steps or if, for some $j\in\{1,\ldots,n\}$, there is a state $b\in W$ such that $y_j\in\eta(b)$ and b is accessible from a in at least i_j R-steps. This can be tested in NL.

As for the case $[B]\subseteq\mathrm{E}$, we take advantage of the duality of F and G, and $\wedge$ and $\vee$, respectively. Analogous considerations as above lead to the logspace computable normal form

$$\varphi'=\mathsf{GX}^{i_1}y_1\wedge\cdots\wedge\mathsf{GX}^{i_n}y_n\wedge\mathsf{X}^{i_{n+1}}y_{n+1}\wedge\cdots\wedge\mathsf{X}^{i_m}y_m.$$

Let $I=\max\{i_1,\ldots,i_m\}$. For each $j=1,\ldots,m$, we define $W^j=\{b\in W\mid y_j\in\eta(b)\}$ and $R^j=R\cap W^j\times W^j$. Furthermore, let W' be the union of W^j for $j=1,\ldots,n$ (!), and let $R'=R\cap W'\times W'$. Now $\langle\varphi,K,a\rangle\in\mathrm{MC}(\{\mathsf{G},\mathsf{X}\},B)$ if and only if there is some state $b\in W'$ satisfying the following conditions.

- There is an R-path p of length at least I from a to b, where the first $I+1$ states on p are $c_0=a$, c_1, …, c_I.
- The state b' lies on a cycle in W'.
- For each $j=1,\ldots,n$, each state of p from c_{i_j} to c_I is from W^j.
- For each $j=n+1,\ldots,m$, the state c_{i_j} is from W^j.

These conditions can be tested in NL as follows. Successively guess $c_1, \ldots, c_I$ and verify their membership in the appropriate sets W^j. Then guess b, verify whether $b \in W'$, whether b lies on some R'-cycle, and whether there is an R'-path from c_I to b. ❑

In the proof of Theorem 4.3, we have exploited the duality of F and G, and $\vee$ and $\wedge$, respectively. Furthermore, the proof relied on the fact that F and $\vee$ (and G and $\wedge$) are interchangeable. This is not the case for F and $\wedge$, or G and $\vee$, respectively. Hence it is not surprising that MC($\{\mathsf{F}\}, \{\wedge\}$) is NP-hard (Corollary 3.3). However, the NL-membership of MC($\{\mathsf{F}, \mathsf{G}\}, \{\vee\}$) *is* surprising. Before we formulate this result, we try to provide an intuition for the tractability of this problem. The main reason is that an inductive view on L($\{\mathsf{F}, \mathsf{G}\}, \{\vee\}$)-formulae allows us to subsequently guess parts of a satisfying path without keeping the previously guessed parts in memory. This is possible because each L($\{\mathsf{F}, \mathsf{G}\}, \{\vee\}$)-formula φ can be rewritten as

$$\varphi = y_1 \vee \cdots \vee y_n \vee \mathsf{F}z_1 \vee \cdots \vee \mathsf{F}z_m \vee \mathsf{G}\psi_1 \vee \cdots \vee \mathsf{G}\psi_\ell \vee \mathsf{FG}\psi_{\ell+1} \vee \cdots \vee \mathsf{FG}\psi_k, \tag{1}$$

where the y_i, z_i are variables (or constants), and each ψ_i is an L($\{\mathsf{F}, \mathsf{G}\}, \{\vee\}$)-formula of the same form with a strictly smaller nesting depth of G-operators. Now, φ is *true* at the begin of some path p iff one of its disjuncts is *true* there. In case none of the y_i or $\mathsf{F}z_i$ is *true*, we must guess one of the $\mathsf{G}\psi_i$ (or $\mathsf{FG}\psi_j$) and check whether ψ_i (or ψ_j) is *true* on the entire path p (or on p minus some finite number of initial states). Now ψ_i is again of the above form. So we must either find an infinite path on which $y_1 \vee \cdots \vee y_n \vee \mathsf{F}z_1 \vee \cdots \vee \mathsf{F}z_m$ is *true* everywhere (a cycle containing at least $|N|$ states satisfying some y_i or z_i suffices, where N is the set of states of the Kripke structure), or we must find a *finite* path satisfying the same conditions and followed by an infinite path satisfying one of the $\mathsf{G}\psi_i$ (or $\mathsf{FG}\psi_j$) at its initial point. Hence we can recursively solve a problem of the same kind with reduced problem size.

Note that it is neither necessary to explicitly compute the normal form for φ or one of the ψ_i, nor need previously visited states be stored in memory.

Theorem 4.4. *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathrm{V}$. Then* MC($\{\mathsf{G}\}, B$) *and* MC($\{\mathsf{F}, \mathsf{G}\}, B$) *are* NL*-complete.*

Proof. The lower bound follows from Lemma 4.1. It remains to show NL-membership of MC($\{\mathsf{F}, \mathsf{G}\}, B$). For this purpose, we devise the recursive algorithm $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$ as given in Figure 6. Note that we have deliberately left out constants. This is no restriction, since we have observed in Lemma 2.2 that each constant can be regarded as a variable that is set to *true* or *false* throughout the whole Kripke structure.

The parameter *mode* indicates the current "mode" of the computation. The idea is as follows. In order to determine whether φ is satisfiable at the *initial* point of some structure starting at a in K, the algorithm has to be in mode `now`. This, hence, is the default setting for the first call of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$. As soon as the algorithm chooses to process a G-subformula $\mathsf{G}\alpha$ of φ, it has to determine whether α is satisfiable at *every* point in some structure starting at the currently visited state in K. It therefore changes into `always` mode and calls itself recursively with the first parameter set to α, see Line 17.

Hence, given an instance $\langle \varphi, K, a \rangle$ of MC($\{\mathsf{F}, \mathsf{G}\}, B$), we have to invoke $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{now})$ in order to determine whether there is a satisfying path for φ in K starting at a. It is easy to see that this call always terminates: First, whenever the algorithm calls itself recursively, the first argument of the new call is a strict subformula of the original first argument. Therefore there can be at most $|\varphi|$ recursive calls. Second, within each call, each passage through the *while* loop (Lines 2–32) either decreases ψ or increases c. Hence, there can be at most $|\varphi| \cdot (|W| + 1)$ passages through the *while* loop until the algorithm accepts or rejects.

$\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$ is an NL algorithm: The values of all parameters and programme variables are either subformulae of the original formula φ, states of the given Kripke structure K, counters of range $0, \ldots, |W| + 1$, or booleans. They can all be represented using $\lceil \log |\varphi| \rceil$, $\lceil \log(|W| + 1) \rceil$, or constantly many bits. Furthermore, since the algorithm uses no *return* command, the recursive calls may re-use the space provided for all parameters and programme variables, and no return addresses need be stored.

It remains to show the correctness of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$, which we will do in two steps. in `always` mode, which will be shown by induction on the nesting depth of the G-operator in φ. We denote this value by $\mu_{\mathsf{G}}(\varphi)$. Claim 6 will then ensure the correct behaviour in `now` mode. Its proof is similar to, but technically different from, the proof of Claim 5.

Claim 5. For each $\varphi \in \mathrm{L}(\{\mathsf{F}, \mathsf{G}\}, \mathrm{V})$, each $K = (W, R, \eta)$, and each $a \in W$:

$$\langle \mathsf{G}\varphi, K, a \rangle \in \mathrm{MC}(\{\mathsf{F}, \mathsf{G}\}, B) \quad \Leftrightarrow \quad \text{there exists an accepting run of } \mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{always})$$

Algorithm $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathsf{V}}$

Input $\varphi \in \mathrm{L}(\{\mathsf{F},\mathsf{G}\}, B)$
Kripke structure $K = (W, R, \eta)$
$a \in W$
additional parameter $mode \in \{\mathtt{now}, \mathtt{always}\}$

Output **accept** or **reject**

```
 1: c ← 0;   ψ ← φ;   b ← a;   Ffound ← false
 2: while c ≤ |W| do
 3:   if ψ = α0 ∨ α1  (for some α0, α1)  then
 4:     guess i ∈ {0, 1}
 5:     ψ ← αi
 6:   else if ψ = Fα  (for some α)  then
 7:     Ffound ← true
 8:     ψ ← α
 9:   else                                   /* ψ is some Gα or a variable */
10:     if Ffound then                        /* process encountered F */
11:       guess n with 0 ≤ n ≤ |W|
12:       for i = 1, 2, ..., n do           /* if n = 0, ignore this loop */
13:         b ← guess some R-successor of b
14:       end for
15:     end if
16:     if ψ = Gα  (for some α)  then
17:       call MC_{F,G},V(α, K, b, always)
18:     else                                  /* ψ is a variable */
19:       if ψ ∉ η(b) then
20:         reject
21:       end if
22:       if mode = always then
23:         c ← c + 1
24:         b ← guess some R-successor of b
25:         Ffound ← false
26:         ψ ← φ
27:       else
28:         accept
29:       end if
30:     end if
31:   end if
32: end while
33: accept
```

Fig. 6. The algorithm $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$

Proof of Claim 5. *For the base case of the induction,* let $\mu_{\mathsf{G}}(\varphi) = 0$. Because of the equivalences $\mathsf{F}(\psi_1 \vee \psi_2) \equiv \mathsf{F}\psi_1 \vee \mathsf{F}\psi_2$ and $\mathsf{FF}\psi \equiv \mathsf{F}\psi$, we may assume w.l.o.g. that any occurrence of the F-operator is in front of some variable in φ. If we think of φ as a tree, this means that F-operators can only occur in direct predecessors of leaves. Note that the algorithm computes this normal form implicitly: Whenever it guesses a path from the root (φ) to some leaf (a variable) in the tree and encounters an F-operator in Line 6, the flag *Ffound* is set. Only after processing all $\vee$-operators on the remaining part of the path, the F-operator is processed in Lines 10–15. Now let $\mathrm{VAR}_1(\varphi)$ be all variables that occur in the scope of an F-operator in φ, and let $\mathrm{VAR}_0(\varphi)$ be all other variables in φ.

For the "$\Rightarrow$" direction, suppose $\langle \mathsf{G}\varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B)$. Then there exists a path p in K such that $p_0 = a$, and for all $i \geq 0$, $p^K, i \vDash \varphi$. This means that, for each i, either there exists some $x_i \in \mathrm{VAR}_0(\varphi)$ such that $p^K, i \vDash x_i$, or there is some $x_i \in \mathrm{VAR}_1(\varphi)$ such that $p^K, i \vDash \mathsf{F}x_i$. Now it can be seen that there is a non-rejecting sequence of runs through the *while* loop in Lines 2–32 after which c has value $|W| + 1$, which then leads to the *accept* in Line 33:

Consider the begin of an arbitrary single run through the *while* loop in Line 2. Let p_i be the current value of b. If $x_i \in \mathrm{VAR}_0(\varphi)$, then the algorithm can "guess its way through the tree of φ" in Lines 3–5 and finally reaches Line 19 with $\psi = x_i$. It does not reject in Line 20, increases c in Line 23, guesses

p_{i+1} in Line 24, and resets *Ffound* and ψ appropriately in Lines 25, 26. Otherwise, if $x_i \in \mathrm{VAR}_1(\varphi)$, then there is some $n \geq 0$ such that p_{i+n} satisfies x_i. It is safe to assume that $n \leq |W|$ because otherwise the path from p_i to p_{i+n} would describe a cycle within K which could be replaced by a shorter, more direct, path without affecting satisfiability of the relevant subformulae in the states $p_0, \ldots, p_i$. Now the algorithm can proceed as in the previous case, but, in addition, it has to guess the correct value of n and the sequence $p_{i+1}, \ldots, p_{i+n}$ in Lines 10–15.

For the "$\Leftarrow$" direction, suppose there exists an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{always})$. Since the algorithm is in `always` mode, and φ is G-free, the acceptance can only take place in Line 33, without a recursive call in Line 17. Hence the counter c reaches value $|W|+1$ in the *while* loop in Lines 2–32.

Let $p = p_0, p_1, \ldots, p_m$ be the sequence of states guessed in this run in Lines 13 and 24, where $p_0 = a$. Furthermore, let $i_0, \ldots, i_{|W|+1}$ be an index sequence that determines a subsequence of p such that

- $0 = i_0 < i_1 < \cdots < i_{|W|+1} = m$, and
- for each $j > 0$, p_{i_j} is the value assigned to b in Line 24 after having set c to value j in Line 23.

Now it is clear that for all $j = 0, \ldots, |W|$, there is a variable x_j such that $x_j \in \eta(p_{i_{j+1}-1})$. If $x_j \in \mathrm{VAR}_0(\varphi)$, then $p_{i_{j+1}} = p_{i_j} + 1$, and each structure p' extending p beyond p_m satisfies x_j (and hence φ) at p_{i_j}. Otherwise $x_j \in \mathrm{VAR}_1(\varphi)$, and the accepting run of the algorithm has guessed the states $p_{i_j}, \ldots, p_{i_{j+1}-1}$ in Line 13. In this case, each structure p' extending p beyond p_m satisfies $\mathsf{F}x_j$ (and hence φ) at $p_{i_j}, \ldots, p_{i_{j+1}-1}$. From these two cases, it follows that each such p' satisfies φ in all states $p_0, \ldots, p_m$.

We now restrict attention to the states $p_{i_1-1}, \ldots, p_{i_{|W|+1}-1}$. Among these $|W|+1$ states, some of the $|W|$ states of K has to occur twice. Assume p_{i_j-1} and p_{i_k-1} represent the same state from K, where $j < k$. Then we can create an (infinite) structure p'' from p that consists of states $p_0, \ldots, p_{i_k-1}$, followed by an infinite repetition of the sequence $p_{i_j}, \ldots, p_{i_k-1}$. It is now obvious that p'' satisfies φ in every state, hence $p'', 0 \vDash \varphi$, that is, $\langle \mathsf{G}\varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B)$.

For the induction step, let $\mu_{\mathsf{G}}(\varphi) > 0$. For the same reasons as above, we can assume that any F-operator only occurs in front of variables or in front of some G-operator in φ. This "normal form" is taken care of by setting *Ffound* to `true` when F is found (Line 7) and processing this occurrence of F only when a variable or some G-operator is found (Lines 10–15).

For the "$\Rightarrow$" direction, suppose $\langle \mathsf{G}\varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B)$. Then there exists a path p in K such that $p_0 = a$, and for all $i \geq 0$, $p^K \vDash \varphi$. We describe an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{always})$. Consider a single passage through the *while* loop with the following configuration. The programme counter has value 2, c has value at most $|W|$, b has value p_i, and ψ has value φ. Since $p^K \vDash \varphi$, there are four possible cases. The argumentation for the first two of them is the same as in the base case.

Case 1. $p^K, i \vDash x$, for some $x \in \mathrm{VAR}_0(\varphi)$.
Case 2. $p^K, i \vDash \mathsf{F}x$, for some $x \in \mathrm{VAR}_1(\varphi)$.
Case 3. $p^K, i \vDash \mathsf{G}\alpha$, for some maximal G-subformula $\mathsf{G}\alpha$ of φ that is *not* in the scope of some F-operator. This means that α is *true* everywhere on the path $p_i, p_{i+1}, p_{i+2}, \ldots$. Hence, due to the induction hypothesis, $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\alpha, K, b_i, \mathtt{always})$ has an accepting run. By appropriate guesses in Line 4, the current call of the algorithm can reach that accepting recursive call in Line 17.
Case 4. $p^K, i \vDash \mathsf{G}\alpha$, for some maximal G-subformula $\mathsf{G}\alpha$ of φ that *is* in the scope of some F-operator. By combining the arguments of Cases 3 and 2, we can find an accepting run for this case.

If only Cases 1 or 2 occur more than $|W|$ times in a sequence, then c will finally take on value $|W|+1$, and this call will accept in Line 31. Otherwise, whenever one of Cases 3 and 4 occurs, than the acceptance of the new call—and hence of the current call—is due to the induction hypothesis.

For the "$\Leftarrow$" direction, suppose there exists an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{always})$. Since the algorithm is in `always` mode, the acceptance can only take place in Line 33 or in the recursive call in Line 17. If the run accepts in Line 33, the same arguments as in the base case apply. If the acceptance is via the recursive call, then let $p = p_0, \ldots, p_m$ be the sequence of states guessed such that $p_0 = a$, and p_m is the value of b when the recursive call with $\mathsf{G}\alpha$ takes place. Due to the induction hypothesis, $\langle \mathsf{G}\alpha, K, b_m\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B)$ and, hence, there is an infinite structure p' extending p beyond p_m such that $(p')^K, m \vDash \mathsf{G}\varphi$. Furthermore, we can use the same argumentation as in the base case to show that, for each $i \leq m$, $(p')^K, i \vDash \varphi$. Therefore, $(p')^K, 0 \vDash \mathsf{G}\varphi$, which proves $\langle \mathsf{G}\varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B)$. ∎

Claim 6. For each $\varphi \in \mathrm{L}(\{\mathsf{F},\mathsf{G}\}, B)$, each $K = (W, R, \eta)$, and each $a \in W$:

$$\langle \varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B) \quad \Leftrightarrow \quad \text{there exists an accepting run of } \mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{now})$$

Claim 6 is proven in the Appendix. ❑

Unfortunately, the above argumentation fails for MC($\{\mathsf{G},\mathsf{X}\},V$) because of the following considerations. The NL-algorithm in the previous proof relies on the fact that a satisfying path for $\mathsf{G}\psi$, where ψ is of the form (1), can be divided into a "short" initial part satisfying the disjunction of the atoms, and the remaining end path satisfying one of the $\mathsf{G}\psi_i$ at its initial state. When guessing the initial part, it suffices to separately guess each state and consult η.

If X were in our language, the disjuncts would be of the form $\mathsf{X}^{k_i}y_i$ and $\mathsf{X}^{\ell_i}\mathsf{G}\psi_i$. Not only would this make the guessing of the initial part more intricate. It would also require memory for processing each of the previously satisfied disjuncts $\mathsf{X}^{k_i}y_i$. An adequate modification of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$ would require more than logarithmic space. We have shown NP-hardness for MC($\{\mathsf{G},\mathsf{X}\},V$) in Theorem 3.4.

Theorem 4.5. *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathrm{L}$. Then* MC($\{\mathsf{X}\},B$) *is* NL*-complete.*

Proof. The lower bound follows from Lemma 4.1.

For the upper bound, let $\varphi \in \mathrm{L}(\{\mathsf{X}\},B)$ be a formula, $K=(W,R,\eta)$ a Kripke structure, and $a \in W$ a state. Let m denote the maximal nesting depth of X-operators in φ. Since for any k-ary Boolean operator f from B, the formula $\mathsf{X}f(\psi_1,\ldots,\psi_k)$ is equivalent to $f(\mathsf{X}\psi_1,\ldots\mathsf{X}\psi_k)$, φ is equivalent to a formula $\varphi' \in \mathrm{L}(\{\mathsf{X}\},B)$ of the form $\boldsymbol{\varphi' = \mathsf{X}^{i_1}p_1 \oplus \cdots \oplus \mathsf{X}^{i_\ell}p_\ell}$, where $0 \le i_j \le m$ for each $j=1,\ldots,\ell$. It is not necessary to compute φ' all at once, because it will be sufficient to calculate i_j each time the variable p_j is encountered in the algorithm $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$ given in Figure 4.

It is easy to see that $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$ returns 1 if and only if φ is satisfiable. From the used variables, it is clear that $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$ runs in nondeterministic logspace. ❑

Algorithm $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$

Input $\varphi' = \mathsf{X}^{i_1}p_1 \oplus \cdots \oplus \mathsf{X}^{i_\ell}p_\ell$
Kripke structure $K=(W,R,\eta)$
$a \in W$

Output **accept** or **reject**

1: *parity* $\leftarrow 0$; $b \leftarrow a$; $k \leftarrow 0$
2: **while** $k \le m$ **do**
3: **for** $j=1,\ldots,\ell$ **do**
4: **if** $i_j = k$ and $p_j \in \eta(b)$ **then**
5: *parity* $\leftarrow 1 -$ *parity*
6: **end if**
7: **end for**
8: $k \leftarrow k+1$
9: $b \leftarrow$ guess some R-successor of b
10: **end while**
11: **return** *parity*

Fig. 7. The algorithm $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$

In the fragment with S as the only temporal operator, S is without effect, since we can never leave the initial state. Hence, any formula $\alpha\mathsf{S}\beta$ is satisfied at the initial state of any structure K if and only if β is. This leads to a straightforward logspace reduction from MC($\{\mathsf{S}\},\mathrm{BF}$) to MC($\emptyset,\mathrm{BF}$): Given a formula $\varphi \in \mathrm{L}(\{\mathsf{S}\},\mathrm{BF})$, successively replace every subformula $\alpha\mathsf{S}\beta$ by β until all occurrences of S are eliminated. The resulting formula φ' is initially satisfied in any structure K iff φ is. Now MC($\emptyset,\mathrm{BF}$) is the Formula Value Problem, which has been shown to be solvable in logarithmic space in [Lyn77]. Thus we obtain the following result.

Theorem 4.6. *Let B be a finite set of Boolean functions. Then* MC($\{\mathsf{S}\},B$) *is in* L.

In our classification of complexity, which is based on logspace reductions $\le_m^{\log}$, a further analysis of S-fragments is not possible. However, a more detailed picture emerges if stricter reductions are considered, see [Sch07, Chapter 2].

Theorem 4.7. *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathrm{V}$. Then* MC($\{\mathsf{S},\mathsf{F}\},B$) *is* NL*-complete.*

Proof. The lower bound follows from Lemma 4.1. For the upper bound, we will show that MC($\{\mathsf{S},\mathsf{F}\},B$) can be reduced to MC($\{\mathsf{F}\},B$) by disposing of the S-operator as follows. Consider an arbitrary Kripke structure K and a path p therein. Then the following equivalences hold.

$$p^K,0 \vDash \alpha\mathsf{S}\beta \quad \text{iff} \quad p^K,0 \vDash \beta \qquad (2)$$

$$p^K,0 \vDash \mathsf{F}(\alpha\mathsf{S}\beta) \quad \text{iff} \quad p^K,0 \vDash \mathsf{F}\beta \qquad (3)$$

$$p^K,0 \vDash \mathsf{F}(\alpha \vee \beta) \quad \text{iff} \quad p^K,0 \vDash \mathsf{F}\alpha \vee \mathsf{F}\beta \qquad (4)$$

$$p^K,0 \vDash \mathsf{FF}\alpha \quad \text{iff} \quad p^K,0 \vDash \mathsf{F}\alpha \qquad (5)$$

Statements (4) and (5) are standard properties and follow directly from the definition of satisfaction for F and $\vee$. Statement (2) is simply due to the fact that there is no state in the past of p_0. As for (3), we consider both directions separately. Assume that $p^K,0 \vDash \mathsf{F}(\alpha\mathsf{S}\beta)$. Then there is some $i \ge 0$ such that $p^K,i \vDash \alpha\mathsf{S}\beta$. This implies that there is some j with $0 \le j \le i$ and $p^K,j \vDash \beta$. Hence, $p^K,0 \vDash \mathsf{F}\beta$. For the other direction, let $p^K,0 \vDash \mathsf{F}\beta$. Then there is some $i \ge 0$ such that $p^K,i \vDash \beta$. This implies $p^K,i \vDash \alpha\mathsf{S}\beta$. Hence, $p^K,0 \vDash \mathsf{F}(\alpha\mathsf{S}\beta)$.

Now consider an arbitrary formula $\varphi \in \mathrm{L}(\{\mathsf{S},\mathsf{F}\},B)$. Let φ' be the formula obtained from φ by successively replacing the outermost S-subformula $\alpha\mathsf{S}\beta$ by β until all occurrences of S are eliminated. This procedure can be performed in logarithmic space, and the result φ' is in $\mathrm{L}(\{\mathsf{F}\},B)$. Due to (2)–(5), for any path p in any Kripke structure K, it holds that $p^K,0 \vDash \varphi$ if and only if $p^K,0 \vDash \varphi'$. Hence, the mapping $\varphi \mapsto \varphi'$ is a logspace reduction from $\mathrm{MC}(\{\mathsf{S},\mathsf{F}\},B)$ to $\mathrm{MC}(\{\mathsf{F}\},B)$. ❑

5 Conclusion, and open problems: the ugly fragments

We have almost completely separated the model-checking problem for Linear Temporal Logic with respect to arbitrary combinations of temporal and propositional operators into tractable and intractable cases. We have shown that all tractable MC problems are at most NL-complete or even easier to solve. This exhibits a surprisingly large gap in complexity between tractable and intractable cases. The only fragments that we have not been able to cover by our classification are those where only the binary *xor*-operator is allowed. However, it is not for the first time that this constellation has been difficult to handle, see [BHSS06, BSS+07]. Therefore, these fragments can justifiably be called ugly.

The borderline between tractable and intractable fragments is somewhat diffuse among all sets of temporal operators without U. On the one hand, this borderline is not determined by a single set of propositional operators (which is the case for the satisfiability problem, see [BSS+07]). On the other hand, the columns E und V do not, as one might expect, behave dually. For instance, while $\mathrm{MC}(\{\mathsf{G}\},\mathrm{V})$ is tractable, $\mathrm{MC}(\{\mathsf{F}\},\mathrm{E})$ is not — although F and G are dual, and so are V and E.

Further work should find a way to handle the open *xor* cases from this paper as well as from [BHSS06, BSS+07]. In addition, the precise complexity of all hard fragments not in bold-face type in Table 1 could be determined. Furthermore, we find it a promising perspective to use our approach for obtaining a fine-grained analysis of the model-checking problem for more expressive logics, such as CTL, CTL*, and hybrid temporal logics.

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 B. Durand and W. Thomas, editors, *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 500–511. Springer, 2006.

[BSS+07] M. Bauland, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The complexity of generalized satisfiability for linear temporal logic. In H. Seidl, editor, *FoSSaCS*, volume 4423 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2007.

[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.

[Lyn77] Nancy A. Lynch. Log space recognition and translation of parenthesis languages. *Journal of the ACM*, 24(4):583–590, 1977.

[Mar04] Nicolas Markey. Past is for free: on the complexity of verifying linear temporal properties with past. *Acta Inf.*, 40(6-7):431–458, 2004.

[Nor05] G. Nordh. A trichotomy in the complexity of propositional circumscription. In *LPAR*, 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 *MFI 99*. World Science Publishing, 2005.

[Sav73] W. J. Savitch. Maze recognizing automata and nondeterministic tape complexity. *Journal of Computer and Systems Sciences*, 7:389–403, 1973.

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

[Sch07] H. Schnoor. *Algebraic Techniques for Satisfiability Problems*. PhD thesis, University of Hannover, 2007.

A Proof of Lemma 3.1

Proof. The reduction $\mathrm{MC}(T,B) \leq_m^{\log} \mathrm{MC}^+(T,B)$ is trivial. We show $\mathrm{MC}^+(T,B) \leq_m^{\log} \mathrm{MC}(T,B)$. Assume that negation is not an element of B, otherwise there is nothing to prove. Let $\langle \varphi, K, a\rangle$ be an instance of $\mathrm{MC}^+(T,B)$, where $K=(W,R,\eta)$. Let $x_1,\ldots,x_m$ be the variables that appear in φ, and for each formula of the kind $\neg\psi(x_1,\ldots,x_n)$ appearing in φ, let $y_{\neg\psi}$ be a new variable. Note that since only propositional negation is allowed in φ, in these cases ψ is purely propositional.

We obtain $K'=(W,R,\eta')$ from K by extending η to the variables $y_{\neg\psi}$ in such a way that $y_{\neg\psi}$ is *true* in a state if and only if $\psi(x_1,\ldots,x_n)$ is *false*. Finally, to obtain φ' from φ we replace every appearance of $\neg\psi(x_1,\ldots,x_n)$ with $y_{\neg\psi}$. Now, φ' is a temporal B-formula. By the construction it is straightforward to see that $\langle \varphi, K, a\rangle \in \mathrm{MC}^+(T,B)$ iff $\langle \varphi', K', a\rangle \in \mathrm{MC}(T,B)$. ❑

B Proofs of Claims

Claim 1. For every path p in $K(\psi)$ and $1 \leq i,j \leq m$ holds: If $p^{K(\psi)}, q_i \vDash \varphi_j$, then $i=j$.

Proof. Assume $p^{K(\psi)}, q_i \vDash \varphi_j$, where $1 \leq i,j \leq m$. By the definition of φ_j, it follows that $p^{K(\psi)}, (j-1)+k\cdot m-(i-1) \vDash b_j$ for some k with $1 \leq k \leq n$. Consider any path p in $K(\psi)$. After the initial part $(p_0,\ldots,p_{m-1}) = (q_1,\ldots,q_m)$ of p follows a sequence $(p_m,\ldots,p_{m\cdot(n+1)})$ of $n\cdot m$ states, where $p_i = l^{i \bmod m}_{\lfloor i/m \rfloor}$ (for $i=m,\ldots,m\cdot(n+1)$). Therefore, $p_{(j-1)+k\cdot m-(i-1)} = l_r^{(j+k\cdot m-i) \bmod m} = l_r^{(j-i) \bmod m}$ for some r (that does not matter here). But $p^{K(\psi)}, l_r^w \vDash b_j$ implies $w=0$, by the definition of $K(\psi)$, and therefore $(j-i) \bmod m = 0$. Since $1 \leq i,j \leq m$, it follows that $j=i$. ■

Claim 2. For every path $p=(q_1,q_2,\ldots)$ in $K(\psi)$ and $i=1,2,\ldots,m$ holds:
$$p^{K(\psi)}, q_i \vDash \psi'_i \quad \text{if and only if} \quad \text{for } j=i,i+1,\ldots,m \text{ holds } p^{K(\psi)}, q_j \vDash \varphi_j.$$

Proof. The direction from right to left is straightforward. To prove the other direction, we use induction.

As base case we consider $i=m$. Assume $p^{K(\psi)}, q_m \vDash \mathsf{G}(\varphi_m \vee \mathsf{G}c)$. By construction of $K(\psi)$ holds $p^{K(\psi)}, q_m \nvDash c$, and therefore $p^{K(\psi)}, q_m \vDash \varphi_m$ holds.

For the inductive step, assume $p^{K(\psi)}, q_i \vDash \mathsf{G}(\varphi_i \vee \mathsf{G}\psi'_{i+1})$. Claim 1 proves $p^{K(\psi)}, q_i \nvDash \varphi_j$ for $j \neq i$, and with $p^{K(\psi)}, q_i \nvDash c$ we obtain $p^{K(\psi)}, q_i \nvDash \mathsf{G}\psi'_{i+1}$. This implies $p^{K(\psi)}, q_i \vDash \varphi_i$ and $p^{K(\psi)}, q_{i+1} \vDash \psi'_{i+1}$. By the inductive hypothesis, the claim follows. ■

Claim 3. Let $K(\psi)$ be constructed from a formula ψ with m clauses, and let p be a path in $K(\psi)$. For $j=1,2,\ldots,m$ it holds that

$$p^{K(\psi)}, q_1 \vDash \varphi_j \quad \text{if and only if} \quad p^{K(\psi)}, q_1 \vDash \varphi_{j-1} \text{ and } p^{K(\psi)}, q_j \vDash a_j \mathsf{U} b_j \ .$$

Proof. We prove the claim by induction. The base case $j=1$ is straightforward: $p^{K(\psi)}, q_1 \vDash \mathit{true}\mathsf{U}(a_1\mathsf{U}b_1)$ is equivalent to $\exists t : p^{K(\psi)}, t \vDash (a_1\mathsf{U}b_1)$ which by Fact 2 is equivalent to $p^{K(\psi)}, q_1 \vDash a_1\mathsf{U}b_1$. The inductive step is split into two cases. First, assume $p^{K(\psi)}, q_1 \vDash \varphi_{j+1}$. Since $\varphi_{j+1} = \varphi_j\mathsf{U}(a_{j+1}\mathsf{U}b_{j+1})$, it follows that $\exists t : p^{K(\psi)}, t \vDash a_{j+1}\mathsf{U}b_{j+1}$. Using Fact 2, we conclude $p^{K(\psi)}, q_{j+1} \vDash a_{j+1}\mathsf{U}b_{j+1}$. By Fact 1, $p^{K(\psi)}, q_1 \nvDash a_{j+1}\mathsf{U}b_{j+1}$. By the initial assumption, this leads to $p^{K(\psi)}, q_1 \vDash \varphi_j$. Second, assume $p^{K(\psi)}, q_1 \vDash \varphi_j$ and $p^{K(\psi)}, q_{j+1} \vDash a_{j+1}\mathsf{U}b_{j+1}$. Using the induction hypothesis, we obtain $p^{K(\psi)}, q_i \vDash a_i\mathsf{U}b_i$ for $i=1,2,\ldots,j+1$. By the construction of φ_{j+1} we straightforwardly get $p^{K(\psi)}, q_1 \vDash \varphi_{j+1}$. ■

Claim 4. Let $H(\psi)$ be constructed from a formula $\psi = C_1 \wedge \ldots \wedge C_m$ with m clauses, and let $p = (s, l_n, \ldots, l_1, q_m, \ldots, q_1, t, t, \ldots)$ be a path in $H(\psi)$. For $j=1,2,\ldots,m$ it holds that

$$p^{H(\psi)}, q_j \vDash \varphi_j^m \quad \text{if and only if} \quad \text{the assignment } \mathcal{A}_p \text{ satisfies clauses } C_j,\ldots,C_m.$$

Proof. Notice that $\mathcal{A}_p$ satisfies clause C_j if and only if p contains a state w with $b_j \in \eta'(w)$. We prove the claim by induction. Since the variable d holds in all predecessors of q_m in p but not in q_m, it follows that $\varphi_m^m = (a_m\mathsf{S}b_m)\mathsf{S}d$ holds in q_m iff $a_m\mathsf{S}b_m$ holds in q_m. Since $b_m \notin \eta'(q_m)$, it follows that $a_m\mathsf{S}b_m$ holds in q_m iff b_m holds in a predecessor of q_m iff $\mathcal{A}_p$ satisfies C_m. This completes the base case. For the inductive step, notice that $p^{H(\psi)}, q_j \vDash \varphi_j^m$ iff $p^{H(\psi)}, q_j \vDash a_j\mathsf{S}b_j$ and $p^{H(\psi)}, q_{j+1} \vDash \varphi_{j+1}$. By the construction of $H(\psi)$ it follows that $p^{H(\psi)}, q_j \vDash a_j\mathsf{S}b_j$ iff $\mathcal{A}_p$ satisfies C_j, and the rest follows from the induction hypothesis. ■

Claim 6. For each $\varphi \in \mathrm{L}(\{\mathsf{F},\mathsf{G}\}, B)$, each $K = (W, R, \eta)$, and each $a \in W$:

$$\langle \varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B) \quad \Leftrightarrow \quad \text{there exists an accepting run of } \mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{now})$$

Proof. *For the "⇒" direction,* suppose $\langle \varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\}, B)$. Then there exists a path p in K such that $p_0 = a$ and $p^K, 0 \vDash \varphi$. We describe an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{now})$. Consider the first passage through the *while* loop with the following configuration. The programme counter has value 2, c has value 0 (this value does not change in *now* mode), b has value a, and ψ has value φ. Since $p^K, 0 \vDash \varphi$, there are four possible cases. The argumentation for them is very similar to that in the proof of Claim 5.

Case 1. $p^K, 0 \vDash x$, for some $x \in \mathrm{VAR}_0(\varphi)$.
As in the proof of Claim 5, the algorithm can guess the appropriate disjuncts in Lines 3–5, does not reject in Line 20 and accepts (it is in *now* mode!) in Line 28.

Case 2. $p^K, 0 \vDash \mathsf{F}x$, for some $x \in \mathrm{VAR}_1(\varphi)$.
As in the proof of Claim 5, there exists some n with $0 \leq n \leq |W|$ such that b_n satisfies x_i. The algorithm can proceed as in the previous case, but, in addition, it has to guess the correct value of n and the sequence $p_1, \ldots, p_n$ in Lines 10–15.

Case 3. $p^K, 0 \vDash \mathsf{G}\alpha$, for some maximal G-subformula $\mathsf{G}\alpha$ of φ that is *not* in the scope of some F-operator.
This means that α is *true* everywhere on the path p. Hence, due to the induction hypothesis, $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\alpha, K, b_i, \mathtt{now})$ has an accepting run. By appropriate guesses in Line 4, the current call of the algorithm can reach that accepting recursive call in Line 17.

Case 4. $p^K, 0 \vDash \mathsf{G}\alpha$, for some maximal G-subformula $\mathsf{G}\alpha$ of φ that *is* in the scope of some F-operator.
By combining the arguments of Cases 3 and 2, we can find an accepting run for this case.

For the "⇐" direction, suppose there exists an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{now})$. Since the algorithm is in `now` mode, the acceptance can only take place in Line 28 or in the recursive call in Line 17. If the run accepts in Line 28, then there is some variable x such that either $x \in \mathrm{VAR}_0(\varphi)$ and $x \in \eta(a)$, or $x \in \mathrm{VAR}_1(\varphi)$ and the run guesses a path $p_0, \ldots, p_m$ with $p_0 = a$ and $x \in \eta(p_m)$. In both cases, each structure p' extending the sequence of states guessed so far, satisfies φ at a. On the other hand, if the run accepts in the recursive call, we can argue as in the proof of Claim 5. ∎

C Refutation of the Small-Path Property

We will now define a family of formulae, and a family of graphs. The definition is inductive:

For $i = 1$, Let G_i be the graph presented in Figure C. For $i+1$, let G_{i+1} be the graph presented in Figure 9, where G_i is inserted into G_{i+1} using the obvious lead-in and lead-out arrows.

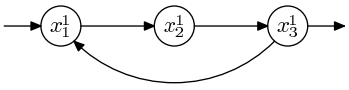

Fig. 8. The graph G_1

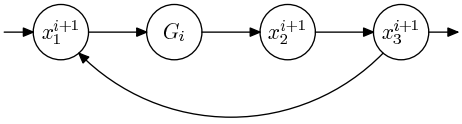

Fig. 9. The graph G_{i+1}

The truth assignments for these graphs is as follows:

$$G_1 : \begin{array}{|c|l|}\hline x_1^1 & b_1 \\ \hline x_2^1 & a_1 \\ \hline x_3^1 & a_1, c_1 \\ \hline \end{array} \qquad G_{i+1} : \begin{array}{|c|l|}\hline x_1^{i+1} & \bigwedge_{j=1}^{i+1} a_j \\ \hline x_2^{i+1} & a_{i+1} \\ \hline x_3^{i+1} & \bigwedge_{j=1}^{i+1} a_j, c_{i+1} \\ \hline x \in G_i & \text{truth assignment from } G_i, b_{i+1} \\ \hline \end{array}$$

Now the formulae are defined as follows: $\varphi_1 := (a_1 \mathsf{U} b_1)\mathsf{U} c_1$, $\varphi_{i+1} := ((a_{i+1}\mathsf{U}\varphi_i)\mathsf{U} b_{i+1})\mathsf{U} c_{i+1}$.

The idea behind the construction is as follows: To satisfy the formula φ_1 in G_1, the path has to repeat the circle once. In the inductive construction, this leads to an exponential number of repetitions. We will now define "canonical paths" in these graphs. These are meant to be the shortest paths in the graph G_i possible that satisfy the formula φ_i. These paths are as follows:

$$\begin{aligned} P_1 &= x_1^1, x_2^1, x_3^1, x_1^1, x_2^1, x_3^1, \\ P_{i+1} &= x_1^{i+1}, P_i, x_2^{i+1}, x_3^{i+1}, x_1^{i+1}, P_i, x_2^{i+1}, x_3^{i+1}. \end{aligned}$$

We claim that the path P_i is a minimal path satisfying φ_i in G_i with the property that it goes back to the out-going edge. The last condition is needed for the recursion, a minimal path satisfying φ_i in G_i can be insignificantly shorter.

We first show that $(a_i \downarrow)^* P_i \vDash \varphi_i$. Here $(a_i \downarrow)^*$ denotes a sequence of states where a_i holds, and no other variable with index of at most i holds. For $i = 1$, this is easy to verify. Therefore, let the claim hold for i. Now, it holds that

$$\varphi_{i+1} = \overbrace{(\underbrace{(a_{i+1}\mathsf{U}\varphi_i)}_{\psi'_{i+1}}\mathsf{U} b_{i+1})}^{\psi''_{i+1}}\mathsf{U} c_{i+1}.$$

For $i+1$, the path P_{i+1} and the truth assignments are as follows (here, $a_j \downarrow$ for some index j denotes the conjunction $a_1 \wedge \cdots \wedge a_j$). In the first row of the table, we list the states, in the second row we list the truth assignments to the variables. The following two rows follow from the above and the induction hypothesis. (We do not claim that the list of subformulae holding in the states presented here is complete, we just highlight those which suffice to show that φ_{i+1} holds.)

$(a_{i+1}\downarrow)^*$	x_1^{i+1}	P_i	x_2^{i+1}	x_3^{i+1}	x_1^{i+1}	P_i	x_2^{i+1}	x_3^{i+1}
$a_{i+1}\downarrow$	$a_{i+1}\downarrow$	b_{i+1}	a_{i+1}	$a_{i+1}\downarrow$ c_{i+1}	$a_{i+1}\downarrow$	b_{i+1}	a_{i+1}	$a_{i+1}\downarrow$ c_{i+1}
φ_i	φ_i			φ_i	φ_i			
ψ'_{i+1}	ψ'_{i+1}		ψ'_{i+1}	ψ'_{i+1}	ψ'_{i+1}			
ψ''_{i+1}	ψ''_{i+1}	ψ''_{i+1}	ψ''_{i+1}	ψ''_{i+1}	ψ''_{i+1}	ψ''_{i+1}		
φ_{i+1}	φ_{i+1}	φ_{i+1}	φ_{i+1}	φ_{i+1}				φ_{i+1}

We claim that P_i is the shortest path satisfying φ_i in G_i. This is clear for $i = 1$. Now suppose that there is a "shortcut" in G_{i+1}. If the shortcut does not use the back-edge (x_3^{i+1}, x_1^{i+1}), at position x_2^{i+1} (this is only visited once, therefore this position is unique), $\mathsf{F} b_{i+1}$ does not hold, and thus ψ''_{i+1} does not hold in x_2^{i+1}. This is a contradiction. Therefore, we pass G_i at least twice.

We show that both times that we pass G_i, we need to follow the full path P_i. We first look at the first passing through G_i. In the path through G_{i+1}, the formula φ_i must hold at the first state from G_i. Since a_i, b_i, and c_i all do not hold in x_2^{i+1}, and no variables with a smaller index, and path fulfilling a smaller $\varphi_j, j < i$, is interrupted here. Hence it follows that φ_i must be satisfied by the path through G_i, by induction hypothesis, this means that G_i has to be passed using the path P_i.

When we pass G_{i+1} for the second time, φ_i again must hold at the first state from G_i and by the same argument as above, this means that G_i again must be passed using the path P_i.

We disregarded the fact that in the very last occurrence of P_j for $j \leq i+1$ the path does not have to pass the whole path P_j but this makes the relevant part of the path only linearly shorter ($2 \cdot (i+1)$ states less).

The length of P_i is obviously exponential in the size of the graph plus the size of the formula.

D Known Facts from Graph Theory

Lemma D.1. *The following problem is* NL*-hard. Given a directed graph $G = (V, E)$ and a node $a \in V$, is there an infinite path in G starting at a?*

Proof. We reduce from the graph accessibility problem (GAP), which is defined as follows. Given a directed graph $G = (V, E)$ and two nodes $a, b \in V$, is there a path in G from a to b? This problem is known to be NL-complete [Sav73].

For the reduction, consider an arbitrary instance $\langle G, a, b\rangle$ of GAP, where $G = (V, E)$ and $a, b \in V$. Let $|V| = n$. We transform G into a new graph G' that consists of n "layers" each of which contains a copy of the nodes from V. Whenever there is an edge from node v to node w in G, the new graph G' will have edges from each copy of v to the copy of w on the next layer. This destroys all cycles from G. Now we add an edge from each copy of b to the first copy of a.

More formally, transform $\langle G, a, b\rangle$ into $\langle G', a^1\rangle$, where $G' = (V', E')$ with

$$V' = \{v^i \mid v \in V \text{ and } 1 \leq i \leq n\},$$
$$E' = \{(v^i, w^{i+1}) \mid (v, w) \in E \text{ and } 1 \leq i < n\} \cup \{(b^i, a^1) \mid 1 \leq i \leq n\}.$$

It is easy to see that this transformation is a logspace reduction. Let the size of a graph be determined by the size of its adjacency matrix. Hence G has size n^2, and G' is of size n^4. Apart from the representation of G', the only space required by the described transformation is spent for four counters that take values between 1 and n. With their help, each bit of the new adjacency matrix is set according to the definition of E', where only a look-up in the old adjacency matrix is required.

It remains to prove the following claim.

Claim 7. For each directed graph $G = (V, E)$ and each pair of nodes $a, b \in V$, there exists a path in G from a to b if and only if there exists an infinite path in G' starting at a^1.

Proof of Claim 7. "$\Rightarrow$". Suppose there is a path in G from a to b. Without loss of generality, we can assume that no node occurs more than once on this path, a and b included. Hence there exist nodes $c_1, \ldots, c_m \in V$ with $m \leq n$ such that $c_1 = a$, $c_m = b$, and for each $i = 1, \ldots, n-1$, $(c_i, c_{i+1}) \in E$. Due to its construction, G' has the cycle $(c_1^1,\ c_2^2,\ \ldots,\ c_m^m,\ a^1)$ that contains a^1. Hence G' has an infinite path starting at a^1.

"$\Leftarrow$". Suppose there is an infinite path p in G' starting at a^1. Since G' is finite, some node must occur infinitely often on p. This, together with the layer-wise construction of G', implies that there are infinitely many nodes of layer 1 on p. Among layer-1 nodes, only a^1 has ingoing edges. Hence a^1 must occur infinitely often on p. Now the path from some occurrence of a^1 to the next is a cycle, where the predecessor node of a^1 must be some b^m. This implies that there is a path in G' from a^1 to b^m. Due to the construction of G', this corresponds to a path in G from a to b. ∎ ❑