Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 231 (2009) 277–292
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

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

Michael Bauland[a] Martin Mundhenk[b] Thomas Schneider[c]
Henning Schnoor[d,1] Ilka Schnoor[d] Heribert Vollmer[e,2]

[a] *Knipp GmbH, Dortmund, Germany*
Michael.BaulandATknipp.de

[b] *Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany*
mundhenkATcs.uni-jena.de

[c] *School of Computer Science, University of Manchester, UK*
schneiderATcs.man.ac.uk

[d] *Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA*
{hs,is}ATcs.rit.edu

[e] *Theoretische Informatik, Leibniz-Universität Hannover, Germany*
vollmerATthi.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.

Keywords: computational complexity, linear temporal logic, model checking

1 Introduction

Linear Temporal Logic (LTL) has been proposed by Pnueli [11] 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

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

doi:10.1016/j.entcs.2009.02.041

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 [17]. 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 [8]. 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 [6]. 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 [3] 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 [3], 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 [17,8], but in contrast to their results, we will exhibit many tractable fragments and exactly determine their computational 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 [3,1], 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 [6] and [3] 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 [17]), 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,

prop. operators / temp. operators	I	N	E	V	M	L	BF
X	**NL**[10]	**NL**[10]	**NL**[12]	**NL**[11]	NP[2]	**NL**[14]	NP[S]
G	**NL**[10]	**NL**[10]	**NL**[12]	**NL**[13]	NP[2]		NP[S]
F	**NL**[10]	**NL**[10]	NP[5]	**NL**[11]	NP[2]		NP[S]
FG	**NL**[10]	**NL**[10]	NP[c]	**NL**[13]	NP[c]		NP[S]
FX	**NL**[10]	**NL**[10]	NP[c]	**NL**[11]	NP[c]		PS[T]
GX	**NL**[10]	**NL**[10]	**NL**[12]	NP[6]	PS[3]		PS[T]
FGX	**NL**[10]	**NL**[10]	NP[c]	NP[c]	PS[1]		PS[T]
S	L[15]	L[15]	L[15]	L[15]	L[15]	L[15]	L[15]
SX	NP[8]	NP[8]	NP[8]	NP[8]	NP[8]	NP[8]	NP[8]
SG	NP[8]	NP[8]	NP[8]	NP[8]	PS[4]	NP[8]	PS[4]
SF	**NL**[16]	NP[9]	NP[9]	**NL**[16]	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 3.1 (1)
2 Theorem 3.1 (2)
3 Theorem 3.1 (3)
4 Theorem 3.1 (4)
5 Corollary 3.2
6 Theorem 3.3
7 Theorem 3.4
8 Theorem 3.5
9 Theorem 3.6
10 Theorem 4.2
11 Theorem 4.3 (1)
12 Theorem 4.3 (2)
13 Theorem 4.4
14 Theorem 4.5
15 Theorem 4.6
16 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

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 [12] 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.

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. Due to the limitations of space, we have left out a number of proofs. These can be found in the Technical Report [2].

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 [17], 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)[3], 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.

[3] 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.

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

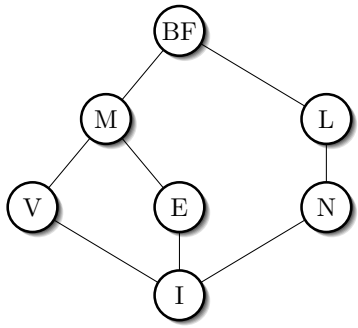

Fig. 1. Clones with constants

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 [17] 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 ([17]) *(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 [10]. For a set B of Boolean functions we denote with $[B]$ the smallest clone containing B and call B a *base* for $[B]$. In [12] Post classified the lattice of all clones and found a finite base for each clone.

The definitions of all clones as well as the full inclusion graph can be found, for example, in [4]. The following lemma, which we prove in [2], 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* $\mathrm{MC}(T, B \cup \{0, 1\}) \equiv_m^{\log} \mathrm{MC}(T, B)$.

Because of Lemma 2.2 it is sufficient to look only at the clones with constants, which are introduced in Definition 2.3. Their bases and inclusion structure are given 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.

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 [6]. Furthermore, Post's lattice has been applied to the equivalence problem [13], to counting [15] and finding minimal [14] solutions, and to learnability [5] 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 [1]. For the inference problem for propositional circumscription, Nordh presented another trichotomy theorem [9].

An important tool in restricting the length of the resulting formula in many of our reductions is the following lemma, which we prove in [2].

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

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 [17]—for example, $\mathrm{MC}(\{\mathsf{F}, \mathsf{X}\}, \{\wedge, \vee, \neg\}) \in$ 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, $\mathrm{MC}(\{\mathsf{F}, \mathsf{X}\}, B) \in$ PSPACE.

3 The bad fragments: intractability results

Sistla and Clarke [17] and Markey [8] have considered the complexity of model-checking for temporal $\{\wedge, \vee, \neg\}$-formulae restricted to atomic negation and propositional negation, respectively. Propositional negation does not affect the complexity of the model checking problem. This can be proven similarly to Lemma 2.2. Using Lemma 2.4 in addition, we can generalise the above mentioned hardness results from [17,8] for temporal monotone formulae to obtain the following intractability results for model-checking. The proofs of all results in this section are given in [2].

Theorem 3.1 *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 [17] 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 3.2 *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 $\{\mathsf{G}, \mathsf{X}\}$-$\{\wedge, \vee\}$-formulae is PSPACE-complete (Theorem 3.1(3) due to [8]). 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 can prove that switching from $\wedge$ to $\vee$ makes the problem intractable.

Theorem 3.3 *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.*

From [17] 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 3.4 *Let B be a finite set of Boolean functions. Then* $\mathrm{MC}(\{\mathsf{U}\}, B)$ *is* NP*-hard.*

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.

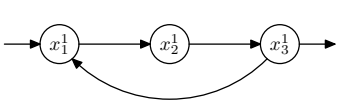

Fig. 2. The graph G_1

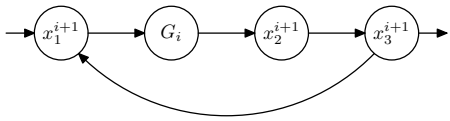

Fig. 3. The graph G_{i+1}

Theorem 3.5 *Let B be a finite set of Boolean functions. Then* $\mathrm{MC}(\{\mathsf{X},\mathsf{S}\},B)$ *and* $\mathrm{MC}(\{\mathsf{G},\mathsf{S}\},B)$ *are* NP*-hard.*

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 $\mathrm{MC}(\{\mathsf{F},\mathsf{S}\},B)$ 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 3.6 *Let B be a finite set of Boolean functions such that* $\mathrm{N} \subseteq [B]$*. Then* $\mathrm{MC}(\{\mathsf{F},\mathsf{S}\},B)$ *is* NP*-hard.*

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. Hence, it will most likely be nontrivial to obtain a better upper bound.

We will now sketch such families of structures and formulae using an inductive definition. Let $G_1, G_2, \ldots$ be the family of graphs presented in Figures 2 and 3. Notice that G_i is inserted into G_{i+1} using the obvious lead-in and lead-out arrows. The truth assignments for these graphs are as follows:

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

Now the formulae are defined as follows:

$$\varphi_1 := (a_1 \mathsf{U} b_1)\mathsf{U} c_1, \qquad \text{and} \qquad \varphi_{i+1} := ((a_{i+1}\mathsf{U}\varphi_i)\mathsf{U} b_{i+1})\mathsf{U} c_{i+1}.$$

The rough 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.

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. It is proven in [2].

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.*

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.

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 4, and which ignores all occurrences of X.

We show in [2] how to obtain an NL algorithm using the normal form φ'. □

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 $\varphi \in \mathrm{L}(\{\mathsf{F}, \mathsf{X}\}, B)$

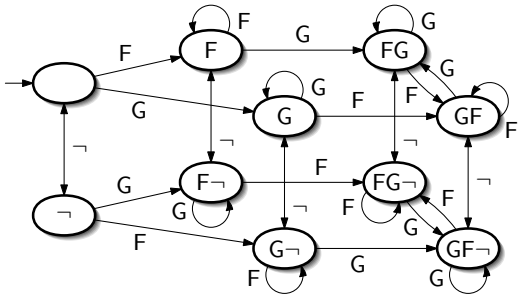

Fig. 4. An automaton that computes $P\sim$

can be rewritten as

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

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

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

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 3.2). 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

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

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 Table 2. 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, for any given 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 $|\varphi| \cdot (|W|+1)$

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

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: $c \leftarrow 0$; $\psi \leftarrow \varphi$; $b \leftarrow a$; $Ffound \leftarrow \mathtt{false}$
2: **while** $c \leq |W|$ **do**
3: **if** $\psi = \alpha_0 \vee \alpha_1$ (for some α_0, α_1) **then**
4: guess $i \in \{0, 1\}$
5: $\psi \leftarrow \alpha_i$
6: **else if** $\psi = \mathsf{F}\alpha$ (for some α) **then**
7: $Ffound \leftarrow \mathtt{true}$
8: $\psi \leftarrow \alpha$
9: **else** /* ψ *is some* $\mathsf{G}\alpha$ *or a variable* */
10: **if** *Ffound* **then** /* *process encountered* F */
11: guess n with $0 \leq n \leq |W|$
12: **for** $i = 1, 2, \ldots, n$ **do** /* *if* $n = 0$, *ignore this loop* */
13: $b \leftarrow$ guess some R-successor of b
14: **end for**
15: **end if**
16: **if** $\psi = \mathsf{G}\alpha$ (for some α) **then**
17: **call** $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\alpha, K, b, \mathtt{always})$
18: **else** /* ψ *is a variable* */
19: **if** $\psi \notin \eta(b)$ **then**
20: **reject**
21: **end if**
22: **if** $mode = \mathtt{always}$ **then**
23: $c \leftarrow c + 1$
24: $b \leftarrow$ guess some R-successor of b
25: $Ffound \leftarrow \mathtt{false}$
26: $\psi \leftarrow \varphi$
27: **else**
28: **accept**
29: **end if**
30: **end if**
31: **end if**
32: **end while**
33: **accept**

Table 2
The algorithm $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}$

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 2 will then ensure the correct behaviour in `now` mode. Both claims are proven in [2].

Claim 1. For each $\varphi \in \mathrm{L}(\{\mathsf{F},\mathsf{G}\},\mathrm{V})$, each $K=(W,R,\eta)$, and each $a \in W$:
$\langle \mathsf{G}\varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\},B)$
$\Leftrightarrow$ there is an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{always})$

Claim 2. For each $\varphi \in \mathrm{L}(\{\mathsf{F},\mathsf{G}\},B)$, each $K=(W,R,\eta)$, and each $a \in W$:
$\langle \varphi, K, a\rangle \in \mathrm{MC}(\{\mathsf{F},\mathsf{G}\},B) \Leftrightarrow$ there is an accepting run of $\mathtt{MC}_{\{\mathsf{F},\mathsf{G}\},\mathrm{V}}(\varphi, K, a, \mathtt{now})$
□

Unfortunately, the above argumentation fails for $\mathrm{MC}(\{\mathsf{G},\mathsf{X}\},V)$ 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 $\mathrm{MC}(\{\mathsf{G},\mathsf{X}\},V)$ in Theorem 3.3.

Theorem 4.5 *Let B be a finite set of Boolean functions such that $[B] \subseteq \mathrm{L}$. Then $\mathrm{MC}(\{\mathsf{X}\},B)$ 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 Table 3.

It is easy to see that $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$ returns 1 if and only if φ is satisfiable. From the used

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 ← 0;  b ← a;  k ← 0
2:  while k ≤ m do
3:    for j = 1, ..., ℓ do
4:      if i_j = k and p_j ∈ η(b) then
5:        parity ← 1 − parity
6:      end if
7:    end for
8:    k ← k + 1
9:    b ← guess some R-successor of b
10: end while
11: return parity
```

Table 3
The algorithm $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$

variables, it is clear that $\mathtt{MC}_{\{\mathsf{X}\},\mathrm{L}}$ runs in nondeterministic logarithmic space. □

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 $\mathrm{MC}(\{\mathsf{S}\}, \mathrm{BF})$ to $\mathrm{MC}(\emptyset, \mathrm{BF})$: 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 $\mathrm{MC}(\emptyset, \mathrm{BF})$ is the Formula Value Problem, which has been shown to be solvable in logarithmic space in [7]. Thus we obtain the following result.

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

In our classification of complexity, which is based on logspace reductions $\leq_m^{\log}$, a further analysis of S-fragments is not possible. However, a more detailed picture emerges if stricter reductions are considered, see [16, Chapter 2].

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

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

$$p^K, 0 \vDash \alpha\mathsf{S}\beta \quad \text{iff} \quad p^K, 0 \vDash \beta \quad (2) \qquad p^K, 0 \vDash \mathsf{F}(\alpha \vee \beta) \quad \text{iff} \quad p^K, 0 \vDash \mathsf{F}\alpha \vee \mathsf{F}\beta \quad (4)$$

$$p^K, 0 \vDash \mathsf{F}(\alpha\mathsf{S}\beta) \quad \text{iff} \quad p^K, 0 \vDash \mathsf{F}\beta \quad (3) \qquad p^K, 0 \vDash \mathsf{FF}\alpha \quad \text{iff} \quad p^K, 0 \vDash \mathsf{F}\alpha \quad (5)$$

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 $i \geq 0$ such that $p^K, i \vDash \alpha \mathsf{S} \beta$. This implies that there is some j with $0 \leq j \leq i$ 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 $i \geq 0$ 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 [1,3]. 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 [3]). On the other hand, the columns E and 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 [1,3]. 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

[1] Bauland, M., E. Hemaspaandra, H. Schnoor and I. Schnoor, *Generalized modal satisfiability.*, in: B. Durand and W. Thomas, editors, *STACS*, Lecture Notes in Computer Science **3884** (2006), pp. 500–511.

[2] Bauland, M., M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor and H. Vollmer, *The tractability of model-checking for LTL: The good, the bad, and the ugly fragments*, Technical Report 07-04, Reports on Computer Science, Friedrich-Schiller-Universität Jena (2007).
URL http://www.minet.uni-jena.de/Math-Net/reports/reports.html

[3] Bauland, M., T. Schneider, H. Schnoor, I. Schnoor and H. Vollmer, *The complexity of generalized satisfiability for linear temporal logic*, in: H. Seidl, editor, *FoSSaCS*, Lecture Notes in Computer Science **4423** (2007), pp. 48–62.

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

[5] Dalmau, V., "Computational Complexity of Problems over Generalized Formulas," Ph.D. thesis, Department de Llenguatges i Sistemes Informàtica, Universitat Politécnica de Catalunya (2000).

[6] Lewis, H., *Satisfiability problems for propositional calculi*, Mathematical Systems Theory **13** (1979), pp. 45–53.

[7] Lynch, N. A., *Log space recognition and translation of parenthesis languages*, Journal of the ACM **24** (1977), pp. 583–590.

[8] Markey, N., *Past is for free: on the complexity of verifying linear temporal properties with past.*, Acta Inf. **40** (2004), pp. 431–458.

[9] Nordh, G., *A trichotomy in the complexity of propositional circumscription.*, in: *LPAR*, Lecture Notes in Computer Science **3452** (2005), pp. 257–269.

[10] Pippenger, N., "Theories of Computability," Cambridge University Press, Cambridge, 1997.

[11] Pnueli, A., *The temporal logic of programs*, in: *FOCS* (1977), pp. 46–57.

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

[13] Reith, S., "Generalized Satisfiability Problems," Ph.D. thesis, Fachbereich Mathematik und Informatik, Universität Würzburg (2001).

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

[15] Reith, S. and K. W. Wagner, *The complexity of problems defined by Boolean circuits*, in: *MFI 99* (2005).

[16] Schnoor, H., "Algebraic Techniques for Satisfiability Problems," Ph.D. thesis, University of Hannover (2007).

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