# The Two-Variable Guarded Fragment with Transitive Guards is 2EXPTIME-Hard*

Emanuel Kieroński

Institute of Computer Science
University of Wrocław
ul. Przesmyckiego 20, 51-151 Wrocław, Poland
email: kiero@ii.uni.wroc.pl

**Abstract.** We prove that the satisfiability problem for the two-variable guarded fragment with transitive guards $\mathsf{GF}^2 + TG$ is 2EXPTIME-hard. This result closes the open questions left in [4], [17]. In fact, we show 2EXPTIME-hardness of $min\mathsf{GF}^2 + TG$, a fragment of $\mathsf{GF}^2 + TG$ without equality and with just one transitive relation $\prec$, which is the only non-unary symbol allowed. Our lower bound for $min\mathsf{GF}^2 + TG$ matches the upper bound for the whole guarded fragment with transitive guards and the unrestricted number of variables $\mathsf{GF} + TG$ given by Szwast and Tendera [17], so in fact $\mathsf{GF}^2 + TG$ is 2EXPTIME-complete. It is surprising that the complexity of $min\mathsf{GF}^2 + TG$ is higher then the complexity of the variant with *one-way* transitive guards $\mathsf{GF}^2 + \overrightarrow{TG}$ [9]. The latter logic appears naturally as a counterpart of temporal logics without past operators.

## 1 Introduction

### 1.1 Modal Logic versus First-Order Logic

The first order logic[1] is a very natural and convenient language for expressing properties of many systems that can be encountered in various areas of computer science. Unfortunately, this convenience and expressive power are expensive and cause that decision problems for the first order logic are difficult to solve algorithmically. In particular it has been known since works of Church and Turing in 1930's, that the satisfiability problem for the first order logic is undecidable.

On the other hand *(propositional) modal and temporal logics* are widely used in computer science. Their satisfiability problems are decidable and they posses a lot of other good algorithmic and model-theoretic properties. Therefore, they have a lot of applications in database theory, artificial intelligence, verification of hardware and software, etc.

Propositional modal logic can be translated into the first order logic. The image of this translation, the so-called *modal fragment*, is a very restricted fragment

* Supported by KBN grant 8 T11C 043 19

[1] By the first order logic we mean in this paper the first order logic without constants and function symbols.

of the first-order logic. Researchers in computer science would like to extend the modal fragment to obtain such fragments of the first order logic which are still decidable and retain good properties of modal logic but are more powerful. Finding such an extension could also provide an explanation of good properties of modal logics.

Since the modal fragment is a two-variable logic, the two-variable first-order logic $\mathsf{FO}^2$ was considered a good candidate for such an extension. The decidability of the satisfiability problem for $\mathsf{FO}^2$ was proved by Mortimer [14]. Later it turned out that it is NEXPTIME-complete. The lower bound was given by Lewis [12] and the upper bound by Grädel, Kolaitis and Vardi [7], who established the exponential model property. Unfortunately, though decidable, $\mathsf{FO}^2$ lacks some good properties of modal logic. For example, if we extend $\mathsf{FO}^2$ by fixed point operators, we obtain a logic which is undecidable [6] in contrast to the $\mu$-calculus [10] – the propositional modal logic augmented with fixed point operators. A lot of algorithmic problems is also caused by the fact that $\mathsf{FO}^2$ does not posses a tree model property.

Another extension of the modal fragment, the so called *guarded fragment*, was proposed by Andréka, van Benthem and Németi [1].

### 1.2 Guarded Fragment of First Order Logic

In the guarded fragment $\mathsf{GF}$ we do not restrict the number of variables or the arity of relation symbols. Some restrictions are imposed on the usage of quantifiers but we do not demand a special prefix of quantifiers, which is usual in many known decidable fragments of the first-order logic. In $\mathsf{GF}$, every quantifier has to be relativized by an atomic formula containing all variables that are free in the scope of this quantifier.

**Definition 1. (Andréka, van Benthem and Németi [1])** *The* guarded fragment $\mathsf{GF}$ *of the first order logic is defined inductively:*

1. *Every atomic formula belongs to* $\mathsf{GF}$.
2. $\mathsf{GF}$ *is closed under* $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
3. *If* $\boldsymbol{x}, \boldsymbol{y}$ *are tuples of variables,* $\alpha(\boldsymbol{x}, \boldsymbol{y})$ *is an atomic formula,* $\psi(\boldsymbol{x}, \boldsymbol{y})$ *is in* $\mathsf{GF}$ *and* $free(\psi) \subseteq free(\alpha) = \{\boldsymbol{x}, \boldsymbol{y}\}$, *where* $free(\phi)$ *is the set of free variables of* $\phi$, *then formulas*
$$\exists \boldsymbol{y}(\alpha(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y})),$$
$$\forall \boldsymbol{y}(\alpha(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \psi(\boldsymbol{x}, \boldsymbol{y}))$$
*belong to* $\mathsf{GF}$.

Atoms $\alpha(\boldsymbol{x}, \boldsymbol{y})$, that relativize quantifiers in the above definition, are called *guards*. The order of variables in a guard $\alpha(\mathbf{x}, \mathbf{y})$ can be arbitrary.

Let us consider some examples. It is easy to express in $\mathsf{GF}$ that a binary relation $R$ is symmetric:

$$\forall x, y\ (R(x, y) \rightarrow R(y, x)).$$

Unfortunately, a formula stating that a binary relation $R$ is transitive:

$$\forall x, y, z\ ((R(x,y) \wedge R(y,z)) \rightarrow R(x,z))$$

is not in $\mathsf{GF}$, since the formula $R(x,y) \wedge R(y,z)$, relativizing the quantifier, is not atomic.

It has been shown that $\mathsf{GF}$ retains a lot of good properties of modal and temporal logic. In particular, Grädel proved that it has the finite model property, and that its every satisfiable formula has a tree-like model [5]. The satisfiability problem is decidable and has double exponential complexity. The reason for such a high complexity is the unrestricted number of variables. In practical applications only several variables are usually used. The bounded version $\mathsf{GF}^k$ of $\mathsf{GF}$, allowing only $k$ variables, is EXPTIME-complete. Grädel and Walukiewicz investigated *guarded fixed point logic* $\mu\mathsf{GF}$ [8]. They proved that adding least and greatest fixed point operators to $\mathsf{GF}$ gives a decidable logic, and moreover, does not increase the complexity, i.e. the satisfiability problem for $\mu\mathsf{GF}$ is in 2EXPTIME and for $\mu\mathsf{GF}^k$ – in EXPTIME. It is worth mentioned, that $\mu\mathsf{GF}$ is a very powerful logic. For example, even its two-variable version without equality allows to encode $\mu$-calculus with backward modalities.

### 1.3 Guarded Fragment with Transitive Relations

In some modal logics transitivity axioms are built-in. Such logics would be conveniently embedded in the guarded fragment if we could specify that some binary relations are transitive. As we have seen, a straightforward idea of expressing transitivity of a binary relation leads to a formula which is not properly guarded. In fact, it can be shown that transitivity cannot be expressed in $\mathsf{GF}$ in any other way.

Grädel [5] considered an extension of $\mathsf{GF}$: we require that some binary relations are transitive. He proved that $\mathsf{GF}^3$ with transitivity statements is undecidable. This result was then improved by Ganzinger, Meyer and Veanes [4] who proved that even $\mathsf{GF}^2$ without equality and with transitivity is undecidable. On the other hand they observed that the two-variable, *monadic* guarded fragment with transitivity $\mathsf{MGF}^2 + TG$ is decidable. A guarded formula is monadic if all of its non-unary predicates can appear only in guards. It seems that $\mathsf{MGF}^2 + TG$ is very close to modal logic with transitivity axioms but as pointed out by Ganzinger, Meyer and Veanes it is stronger since it does not have the finite model property.

### 1.4 Guarded Fragment with Transitive Guards

In the acronym $\mathsf{MGF}^2 + TG$ letters $TG$ denote *transitive guards*. Indeed, if a formula is monadic then in particular all binary and hence all transitive symbols can appear only in guards.

Ganzinger, Meyer and Veanes did not give good complexity estimates for $\mathsf{MGF}^2 + TG$ since their proof was obtained by a reduction to Rabin's theory of $k$

successors [16]. They also left another, natural open question – the decidability of $\mathsf{GF}{+}TG$, the whole guarded fragment with transitive relations, where transitive relations are admitted only in guards but where non-transitive relations and equality can occur elsewhere. The last question was answered by Szwast and Tendera [17] who proved that $\mathsf{GF}{+}TG$ is decidable, and that its satisfiability problem is in 2EXPTIME.

Surprisingly, the two-variable guarded fragment with transitive guards is not in EXPTIME. By an elegant reduction from $\mathsf{FO}^2$ Szwast and Tendera showed that it is NEXPTIME-hard. In [9] we slightly improved this bound. We introduced the two-variable guarded fragment with *one-way* transitive guards $\mathsf{GF}^2 + \overrightarrow{TG}$, which is a proper subset of $\mathsf{GF}^2 + TG$, and showed that its satisfiability problem is EXPSPACE-complete. In this paper we close the remaining gap and prove that the satisfiability problem for $\mathsf{GF}^2 + TG$ is 2EXPTIME-hard.

Our 2EXPTIME lower bound is obtained for a very restricted version of $\mathsf{GF}^2 + TG$ denoted by $min\mathsf{GF}^2 + TG$. This logic does not allow equality and contains only one transitive relation $\prec$, which is the only non-unary symbol. The lack of equality reduces the expressive power of the logic since it is impossible to define cliques. With equality we can enforce that every model of a formula contains transitive cliques of size exponential with respect to the size of the formula.

We believe that this lower bound is surprising for at least two reasons. First, it matches the upper bound for the whole $\mathsf{GF}{+}TG$. It is not usual that the complexity of the two-variable version of a logic equals the complexity of the same logic with the unbounded number of variables. The second reason concerns the correspondence between the guarded fragment with transitive guards and branching temporal logics and will be explained in the next subsection.

### 1.5 Guarded Fragment with One-Way Transitive Guards

What do we mean by the guarded fragment with one-way transitive guards $\mathsf{GF}{+}\overrightarrow{TG}$? Consider a subformula of the form $\exists y\ \alpha(x,y) \wedge \psi(x,y)$, where a binary, transitive symbol $\prec$ is used in the guard $\alpha$. There are two possibilities $\alpha(x,y) = x \prec y$ or $\alpha(x,y) = y \prec x$. In $\mathsf{GF}{+}\overrightarrow{TG}$ we allow only the first one[2]. This is similar to the situation in most temporal logics where we can quantify points in the future but not in the past. In this context, our results become quite interesting: we expose the difference in the complexity of reasoning about the future only and both the future and the past, in the two-variable guarded fragment with transitive guards.

If we interpret the only binary symbol $\prec$ in $min\mathsf{GF}^2 + TG$ as a relation representing time then we can consider $min\mathsf{GF}^2 + TG$ a counterpart of a simple branching temporal logic with both future and past operators. This logic is 2EXPTIME-hard. On the other hand when we disallow past then, even if we allow equality and additional, transitive and non-transitive, binary relations

[2] Our choice of the first possibility is not essential and similar results can be obtained when the second possibility is chosen.

("modalities"), we get a logic which is EXPSPACE-complete, so has lower complexity (assuming hypothesis EXPSPACE $\neq$2EXPTIME).

This is rather surprising since for temporal and process logics the complexities of the satisfiability problem for versions with past operators and versions without them usually coincide. For example Kupferman and Pnueli [11] investigated two variants of CTL augmented with past operators. Both of them turned out to be EXPTIME-complete (similarly to CTL). Adding inverse modalities to $\mu$-calculus does not increase its complexity [18]. Also propositional dynamic logic PDL and its version with converse CPDL have the same complexity [13, 15].

## 2 Preliminaries

In Definition 1 we introduced the guarded fragment GF of the first-order logic. Then we informally introduced some of its variants and extensions. Let us give the formal definition of the variants we are interested in this paper.

**Definition 2.** Guarded fragment with transitive guards GF+$TG$ *is an extension of* GF *by transitive relations. A* GF+$TG$ *formula* $\Phi$ *consists of a* GF *formula* $\Phi'$ *and a list* $\mathcal{T}$ *of binary relation symbols that are required to be transitive. A relation symbol* $T$ *can belong to* $\mathcal{T}$ *if it appears in* $\Phi$ *only in guards. We say that* $\Phi$ *is satisfiable if* $\Phi'$ *is satisfiable in a model in which all interpretations of symbols from* $\mathcal{T}$ *are transitive.*

We distinguish a restricted, *minimal* version of the considered logic, for which we prove our lower bound:

**Definition 3.** *By* $min\mathsf{GF}^2 + TG$ *we denote the fragment of the two-variable* GF$+TG$ *without equality, which contains only one non-unary symbol* $\prec$*. Symbol* $\prec$ *is binary and can be used only in guards.*

An *alternating Turing machine* is a generalization of a nondeterministic Turing machine. States, and hence configurations of such a machine, are partitioned into four groups: existential, universal, accepting and rejecting. The notion of an accepting (rejecting) configuration is extended to the case of configurations in which states are existential or universal. This can be done inductively: a universal configuration is accepting if all its successor configurations, i.e. configurations obtained by performing one machine step according to the transition function, are accepting and an existential configuration is accepting if at least one of its successor configurations is accepting. A machine $M$ accepts its input $w$ if the initial configuration of $M$ on $w$ is accepting. By ASPACE($f$) we denote the set of problems that can be solved by alternating Turing machines working in space bounded by a function $f$. In this paper we will use the following theorem:

**Theorem 1. (Chandra, Kozen, Stockmeyer [3])** *For* $t(n) \geq \log n$*:*

$$ASPACE[t(n)] = DTIME[2^{t(n)}].$$

*Let AEXPSPACE*$=\bigcup_{k=1}^{\infty} ASPACE(2^{n^k})$. *Then in particular:*

$$AEXPSPACE = 2EXPTIME.$$

For details about alternating Turing machines you can see for example [2].

## 3 Proof

In this section we prove the main result of the paper.

**Theorem 2.** *The satisfiability problem for* $min\mathsf{GF}^2 + TG$ *is 2EXPTIME-hard.*

*Proof.* By Theorem 1, it suffices to prove that every problem in AEXPSPACE can be reduced in polynomial time to the satisfiability problem for $min\mathsf{GF}^2 + TG$.

Let $M$ be an alternating Turing machine working in space bounded by $2^{n^k}$. Let $w$ be an input for $M$. We construct a $min\mathsf{GF}^2 + TG$ sentence $\Phi$ which is satisfiable if and only if $M$ accepts $w$. Without any loss of generality we can assume that in every configuration, $M$ has exactly two possible transitions and that on its every computation path it enters an accepting or rejecting state at exactly $2^{2^{n^k}}$-th step. To simplify our proof we assume that after entering an accepting or rejecting state, $M$ does not stop. More precisely, we assume that accepting and rejecting states are universal. In each of such states, $M$ has two identical transitions: it does not write any symbol on the tape and it does not move its head. In other words, after accepting or rejecting $M$ stays infinitely in the same configuration.

Every configuration will be represented by a set of $2^{n^k}$ elements, each of them corresponding to a single cell of the tape. To encode the position of an element in a configuration, i.e. the consecutive number of the tape cell it represents, we use unary relation symbols $P_1, \ldots, P_{n^k}$. Formally, $P_i(a)$ is true if the $i$-th bit of the position of the element $a$ is set to 1. We use the abbreviation $\overline{P}(a)$ to describe this position $(0 \le \overline{P}(a) < 2^{n^k})$.

Observe that it is not difficult to express the following properties with formulas of polynomial length:

$$\overline{P}(x) = l \text{ for fixed } l,\ 0 \le l < 2^{n^k},$$
$$\overline{P}(x) \ge l \text{ for fixed } l,\ 0 \le l < 2^{n^k},$$
$$\overline{P}(x) = \overline{P}(y),$$
$$\overline{P}(x) = \overline{P}(y) + 1.$$

For example the last property can be expressed as follows:

$$\bigvee_{0 \le i < n^k} \big(P_i(x) \wedge \neg P_i(y) \wedge \bigwedge_{j>i} (\neg P_j(x) \wedge P_j(y)) \wedge \bigwedge_{j<i} (P_j(x) \leftrightarrow P_j(y))\big).$$

We connect each pair of elements $a, b$, such that $\overline{P}(a) < \overline{P}(b)$, belonging to the same configuration, with the transitive symbol $\prec$, i.e. we want $a \prec b$ to be true in our model. Figure 1 gives a representation of a configuration.


**Fig. 1.** A representation of an example configuration

We describe a configuration in a standard way: for each symbol $a_i$ in the alphabet of $M$ (including **blank**) we use a unary relation symbol $A_i$, for each state $q_i$ – a unary symbol $Q_i$. We also have a unary symbol $H$ describing the head position. For element $x$ representing a tape cell scanned by the head, $H(x)$ and $Q_i(x)$, for some $i$, will be true.

We begin our construction by enforcing that every model of $\Phi$ contains a substructure that can be viewed as an infinite binary tree. The set of $2^{n^k}$ elements describing a single configuration of $M$ will be treated as a "node" of this tree. In fact, we are not able to enforce that a model contains a "real" tree. For example, it is possible that some of elements of a model represent more than one tape cell. What we do is ensure that every node can identify its two successor nodes.

We organize the structure in such a way that elements belonging to an even configuration, i.e. a configuration whose depth in a computation tree is even, will be smaller, with respect to relation $\prec$, than elements belonging to its successor configurations, and elements belonging to an odd configuration will be greater than elements belonging to its successor configurations. We do not impose any relations between elements that do not belong the same configuration or to two consecutive configurations. Additionally, we introduce unary symbols $D_0, D_1, D_2, D_3$ and enforce that $D_i$ is true exactly for elements belonging to configurations whose number is of the form $4k + i$. One more unary symbol $L$ will indicate that the element belongs to the left son of some node.

The structure of the tree is shown in Fig. 2. Horizontal arrows represent configurations (that are internally ordered by $\prec$ as described above). Orientation of arrows represents the relation $\prec$.

First, we express that for every element of a model at most one of unary relations $D_i$ is true:

$$\Phi_1 \equiv \bigwedge_i (\forall x\ (D_i(x) \rightarrow \bigwedge_{j \neq i} \neg D_j(x))).$$

Formulas $\Phi_2$ and $\Phi_3$ say that there exists a node representing the initial configuration of $M$ on $w$. For all elements of this node, the special unary symbol $I$ is true. We assume that this configuration is a left configuration.

$$\Phi_2 \equiv \exists x\ (I(x) \wedge D_0(x) \wedge L(x) \wedge \overline{P}(x) = 2^{n^k} - 1),$$

$$\begin{aligned}\Phi_3 \equiv \forall x\ (I(x) \rightarrow (&\overline{P}(x) \neq 0 \\ &\rightarrow\ \exists y\ (y \prec x \wedge I(y) \wedge D_0(y) \wedge L(y) \wedge \overline{P}(x) = \overline{P}(y) + 1))).\end{aligned}$$

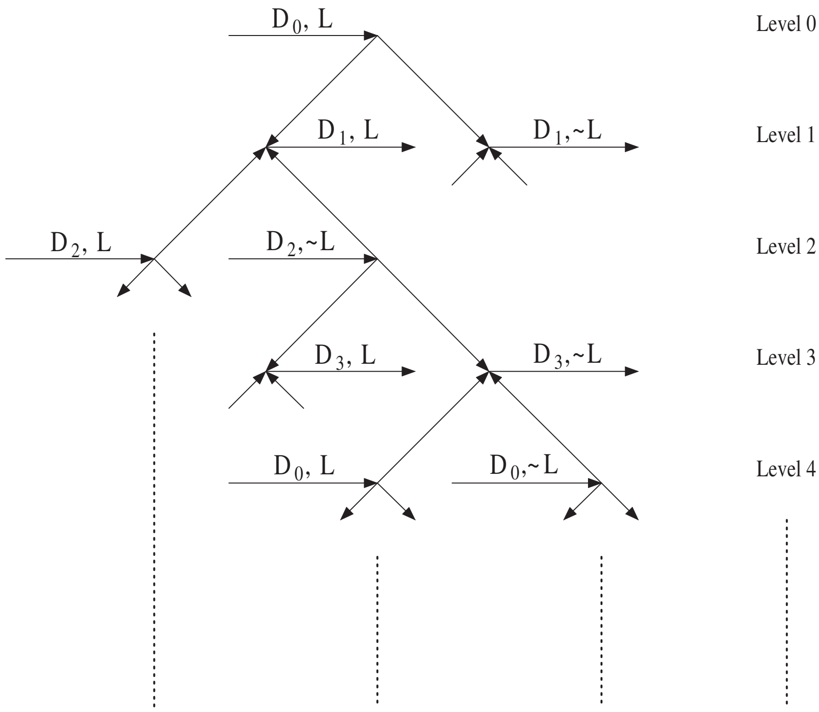


**Fig. 2.** The structure of the tree

Formulas $\Phi_4 - \Phi_7$ express that for every element, except the first one, belonging to a description of an even configuration there exists a predecessor in this configuration, and for every element, except the last one, belonging to a description on an odd configuration there exists a successor in this configuration:

$$\begin{aligned}
\Phi_4 \equiv \forall x \;\; & (D_0(x) \to (\overline{P}(x) \neq 0 \\
& \to \quad \exists y \; (y \prec x \wedge D_0(x) \wedge (L(x) \leftrightarrow L(y)) \wedge \overline{P}(x) = \overline{P}(y) + 1))), \\
\Phi_5 \equiv \forall x \;\; & (D_1(x) \to (\overline{P}(x) \neq 2^{n^k} - 1 \\
& \to \quad \exists y \; (x \prec y \wedge D_1(x) \wedge (L(x) \leftrightarrow L(y)) \wedge \overline{P}(y) = \overline{P}(x) + 1))), \\
\Phi_6 \equiv \forall x \;\; & (D_2(x) \to (\overline{P}(x) \neq 0 \\
& \to \quad \exists y \; (y \prec x \wedge D_2(x) \wedge (L(x) \leftrightarrow L(y)) \wedge \overline{P}(x) = \overline{P}(y) + 1))), \\
\Phi_7 \equiv \forall x \;\; & (D_3(x) \to (\overline{P}(x) \neq 2^{n^k} - 1 \\
& \to \quad \exists y \; (x \prec y \wedge D_3(x) \wedge (L(x) \leftrightarrow L(y)) \wedge \overline{P}(y) = \overline{P}(x) + 1))).
\end{aligned}$$

For every node of the tree there exist left and right successor nodes. For a node representing an even configuration, we connect successors directly to the element $a$ which is the last element in this configuration and enforce that $a$ is smaller than elements in successors. For a node representing an odd configuration, successors are connected to the first element $a$ in the configuration and the element $a$ is made greater than its successors. The existence of appropriate successors will be implied by formulas $\Phi_8 - \Phi_{11}$.

$$\begin{aligned}
\Phi_8 \equiv \forall x \; \big(D_0(x) \to (\overline{P}(x) = 2^{n^k} - 1 \to (&\exists y \; (x \prec y \wedge D_1(y) \wedge L(y) \wedge \overline{P}(y) = 0) \\
&\wedge \exists y \; (x \prec y \wedge D_1(y) \wedge \neg L(y) \wedge \overline{P}(y) = 0)))\big), \\
\Phi_9 \equiv \forall x \; \big(D_1(x) \to (\overline{P}(x) = 0 \to (&\exists y \; (y \prec x \wedge D_2(y) \wedge L(y) \wedge \overline{P}(y) = 2^{n^k} - 1) \\
&\wedge \exists y \; (y \prec x \wedge D_2(y) \wedge \neg L(y) \wedge \overline{P}(y) = 2^{n^k} - 1)))\big), \\
\Phi_{10} \equiv \forall x \; \big(D_2(x) \to (\overline{P}(x) = 2^{n^k} - 1 \to (&\exists y \; (x \prec y \wedge D_3(y) \wedge L(y) \wedge \overline{P}(y) = 0) \\
&\wedge \exists y \; (x \prec y \wedge D_3(y) \wedge \neg L(y) \wedge \overline{P}(y) = 0)))\big), \\
\Phi_{11} \equiv \forall x \; \big(D_3(x) \to (\overline{P}(x) = 0 \to (&\exists y \; (y \prec x \wedge D_0(y) \wedge L(y) \wedge \overline{P}(y) = 2^{n^k} - 1) \\
&\wedge \exists y \; (y \prec x \wedge D_0(y) \wedge \neg L(y) \wedge \overline{P}(y) = 2^{n^k} - 1)))\big),
\end{aligned}$$

We introduce two abbreviations which will help to present some of the remaining formulas in a more succinct way. The formula $SameLetter(x, y)$ says that elements $x$ and $y$ are marked with the same unary symbol from the set $\{D_0, D_1, D_2, D_3\}$. $NEXT(x, y)$ is true for two consecutive elements of a description of a configuration.

$$\begin{aligned}
SameLetter(x, y) &\equiv (D_0(x) \leftrightarrow D_0(y)) \wedge (D_1(x) \leftrightarrow D_1(y)) \\
&\quad \wedge (D_2(x) \leftrightarrow D_2(y)) \wedge (D_3(x) \leftrightarrow D_3(y)), \\
Next(x, y) &\equiv x \prec y \wedge SameLetter(x, y) \wedge \overline{P}(y) = \overline{P}(x) + 1.
\end{aligned}$$

Now we say that a model of our formula satisfies several basic properties of a computation tree. $\Phi_{12}$ states that there is exactly one alphabet symbol in every tape cell.

$$\Phi_{12} \equiv \bigwedge_j \big(\forall x\ (D_j(x) \to \bigvee_i A_i(x))\big) \quad \wedge \quad \bigwedge_j \big(\forall x\ (A_j(x) \to \bigwedge_{i \neq j} \neg A_i(x))\big).$$

Formulas $\Phi_{13} - \Phi_{14}$ say that in each configuration at most one element is scanned by the head.

$$\begin{aligned}
\Phi_{13} &\equiv \forall x\ \big(H(x) \to \forall y\ (x \prec y \to (\mathit{SameLetter}(x,y) \to \neg H(y)))\big),\\
\Phi_{14} &\equiv \forall x\ \big(H(x) \to \forall y\ (y \prec x \to (\mathit{SameLetter}(x,y) \to \neg H(y)))\big).
\end{aligned}$$

$\Phi_{15}$ says that exactly those elements which represent tape cells observed by the head store information about state.

$$\Phi_{15} \equiv \bigwedge_i \big(\forall x\ (Q_i(x) \to H(x))\big) \quad \wedge \quad \forall x\ \big(H(x) \to \bigvee_i Q_i(x)\big).$$

Formulas $\Phi_{16} - \Phi_{18}$ ensure that the root of the tree describes the initial configuration of $M$, in the initial state $q_0$, on the input $w = w_0 \dots w_{n-1}$.

$$\begin{aligned}
\Phi_{16} &\equiv \forall x\ \big(I(x) \to (\overline{P}(x) = 0 \to (H(x) \wedge Q_0(x)))\big),\\
\Phi_{17} &\equiv \bigwedge_{i<n} \forall x\ (I(x) \to (\overline{P}(x) = i \to W_i(x))),\\
\Phi_{18} &\equiv \forall x\ \big(I(x) \to (\overline{P}(x) \geq n \to BLANK(x))\big).
\end{aligned}$$

Formulas $\Phi_{19} - \Phi_{20}$ express that if a tape cell of a configuration is not scanned by the head then in the same cell of both successor configurations the alphabet symbol does not change.

$$\begin{aligned}
\Phi_{19} \equiv \forall x, y\ \big(x \prec y \to \big(((D_0(x) \wedge D_1(y) \vee D_2(x) \wedge D_3(y))\\
\wedge \neg H(x) \wedge \overline{P}(x) = \overline{P}(y)\big) \to \bigwedge_i (A_i(x) \leftrightarrow A_i(y))\big),\\
\Phi_{20} \equiv \forall x, y\ \big(y \prec x \to \big(((D_1(x) \wedge D_2(y) \vee D_3(x) \wedge D_0(y))\\
\wedge \neg H(x) \wedge \overline{P}(x) = \overline{P}(y)\big) \to \bigwedge_i (A_i(x) \leftrightarrow A_i(y))\big).
\end{aligned}$$

Consider now a node $t$ of a tree and a configuration $c$ that is described by this node. There are two cases: the state of the machine in this configuration is existential or it is universal.

In the first case we enforce that the configuration represented by the left son of $t$ is created by applying one of the two possible transitions on $c$. Assume that for an existential state $q$ and a letter $a$ there are two possible transitions: $(q, a) \to (q', a', \rightarrow)$ and $(q, a) \to (q'', a'', \leftarrow)$.

We put:

$$\begin{aligned}
\Phi^{exists}_{(a,q)} \equiv\ & \forall x, y\ \big(x \prec y \rightarrow \big(((D_0(x) \wedge D_1(y) \vee D_2(x) \wedge D_3(y)) \\
&\qquad\qquad\qquad\qquad \wedge Q(x) \wedge A(x) \wedge L(y) \wedge \overline{P}(x) = \overline{P}(y)) \\
&\rightarrow \big((A'(y) \wedge \forall x\ (y \prec x \rightarrow (Next(y,x) \rightarrow H(x) \wedge Q'(x)))) \\
&\quad \vee (A''(y) \wedge \forall x\ (x \prec y \rightarrow (Next(x,y) \rightarrow H(x) \wedge Q''(x)))))\big)\big), \\
\Phi^{exists'}_{(a,q)} \equiv\ & \forall x, y\ \big(y \prec x \rightarrow \big(((D_1(x) \wedge D_2(y) \vee D_3(x) \wedge D_0(y)) \\
&\qquad\qquad\qquad\qquad \wedge Q(x) \wedge A(x) \wedge L(y) \wedge \overline{P}(x) = \overline{P}(y)) \\
&\rightarrow \big((A'(y) \wedge \forall x\ (y \prec x \rightarrow (Next(y,x) \rightarrow H(x) \wedge Q'(x)))) \\
&\quad \vee (A''(y) \wedge \forall x\ (x \prec y \rightarrow (Next(x,y) \rightarrow H(x) \wedge Q''(x)))))\big)\big).
\end{aligned}$$

Other possible situations, when both transitions move the head forward, both transitions move the head backward, one of transitions does not move the head, etc., can be handled similarly.

Consider now the case of a universal configuration. We enforce that the left son of $t$ is created by applying the first transition and the right son – the second one. For a universal state $q$, a letter $a$ and transitions $(q,a) \rightarrow (q', a', \rightarrow)$ and $(q,a) \rightarrow (q'', a'', \leftarrow)$ we put:

$$\begin{aligned}
\Phi^{univ}_{(a,q)} \equiv\ & \forall x, y\ \big(x \prec y \rightarrow \big(((D_0(x) \wedge D_1(y) \vee D_2(x) \wedge D_3(y)) \\
&\qquad\qquad\qquad\qquad\qquad \wedge Q(x) \wedge A(x) \wedge \overline{P}(x) = \overline{P}(y)) \\
&\rightarrow \big(\neg L(y) \rightarrow \big(A''(x) \wedge \forall x\ (x \prec y \rightarrow (Next(x,y) \rightarrow (Q''(x) \wedge H(x)))))\big) \\
&\wedge\ L(y) \rightarrow \big(A'(x) \wedge \forall x\ (y \prec x \rightarrow (Next(y,x) \rightarrow (Q'(x) \wedge H(x)))))\big)\big)\big), \\
\Phi^{univ'}_{(a,q)} \equiv\ & \forall x, y\ \big(y \prec x \rightarrow \big(((D_1(x) \wedge D_2(y) \vee D_3(x) \wedge D_0(y)) \\
&\qquad\qquad\qquad\qquad\qquad \wedge Q(x) \wedge A(x) \wedge \overline{P}(x) = \overline{P}(y)) \\
&\rightarrow \big(\neg L(y) \rightarrow \big(A''(x) \wedge \forall x\ (x \prec y \rightarrow (Next(x,y) \rightarrow (Q''(x) \wedge H(x)))))\big) \\
&\wedge\ L(y) \rightarrow \big(A'(x) \wedge \forall x\ (y \prec x \rightarrow (Next(y,x) \rightarrow (Q'(x) \wedge H(x)))))\big)\big)\big).
\end{aligned}$$

Note that if we had used only two relation symbols $D_0, D_1$ instead of four $D_0, D_1$, $D_2, D_3$ it would have caused problems with distinguishing between successors and predecessors of a configuration.

To finish our construction we give formula $\Phi_{21}$ stating that in none of configurations in a model, $M$ is in its only rejecting state $q_r$.

$$\Phi_{23} \equiv \forall x\ (Q_r(x) \rightarrow \mathbf{false}).$$

We define $\Phi$ as

$$\Phi \equiv \bigwedge_{1 \le i \le 21} \Phi_i \wedge \bigwedge_{(a,q')} \Phi^{exist}_{a,q'} \wedge \bigwedge_{(a,q')} \Phi^{exist'}_{a,q'} \wedge \bigwedge_{(a,q'')} \Phi^{univ}_{a,q''} \wedge \bigwedge_{(a,q'')} \Phi^{univ'}_{a,q''},$$

where $q'$ represents existential states and $q''$ represents universal states. Observe that the number of conjuncts, and the size of each of them are polynomial in $|M|$ and $|w|$.

We claim that $\Phi$ is satisfiable iff $M$ accepts $w$. Indeed, if $M$ accepts $w$ then an accepting computation tree can be transformed into a model $\mathcal{M}$ of $\Phi$ in the following way. The root of the computation tree is transformed into the root of $\mathcal{M}$. Then we proceed recursively. Let $c$ be a configuration in the computation tree and let $c'$ be its code in $\mathcal{M}$. If $c$ is universal then we transform its left subtree into the left subtree of $c'$ and its right subtree into the right subtree of $c'$. If $c$ is existential then we transform its accepting subtree into the left subtree of $c'$. Since we want to have a complete binary tree, we have to define somehow also the right subtree of $c'$. We can for example construct all nodes of this subtree in such a way that they agree with $c'$ in predicates denoting alphabet symbols and for each element $a$ from these nodes

$$\mathcal{M} \models \neg H(a) \wedge \bigwedge_i \neg Q_i(a).$$

It is easy to see that $\mathcal{M}$ is indeed a model of our formula.

For the proof of the opposite direction, we want to check that the existence of an accepting computation tree is implied by the existence of a model of $\Phi$. A minor difficulty lies in the fact that this model is not necessarily a "real" tree, i.e. some elements of the model may represent more than one tape cell. Of course tape cells represented by the same element have the same position in the a configuration but can belong to different configurations. Nevertheless, we can simply obtain a tree model from an arbitrary model of $\Phi$ by taking the appropriate number of copies of such elements. Such a model corresponds to an accepting computation tree of $M$ on $w$, with the exception that only left successors of existential nodes are correct. Now, for formal conformity, we transform the model into a computation tree by substituting right subtrees of existential nodes with subtrees which agree with transition function of $M$. This is not essential since in existential nodes we demand only one accepting successor and we know that the left one is in fact accepting.

$\square$

## 4 A Comment on the Proof

In the preliminary version of the proof some care was taken to provide a kind of the numbering of configurations. We used a unary symbol $B$. For an element $a$ such that $\overline{P}(a) = l$, $B(a)$ was true in the model if and only if $a$ belonged to a description of a configuration whose depth in a computation tree had the $l$-th bit set to 1. With such a numbering, we could encode computations of a usual alternating Turing machine which stops after accepting or rejecting. As pointed out to me by Jerzy Marcinkowski, when we enforce the machine to work infinitely we can get rid of the numbering of configurations.

## 5 Conclusion

The lower bound we gave in Theorem 2 and the upper bound given by Szwast and Tendera [17] lead to the following corollary:

**Corollary 1.** *The satisfiability problem for the two-variable guarded fragment with transitive guards* $\mathsf{GF}^2 + TG$ *is 2EXPTIME-complete.*

Since our result is obtained for $min\mathsf{GF}^2+TG$, which is a subset of $\mathsf{MGF}^2+TG$ we have also established the exact complexity bounds for $\mathsf{MGF}^2 + TG$.

In the table below we summarize the known results on the complexity of decidable extensions of the guarded fragment of first order logic. The result of this paper is denoted by *. The complexity of $\mathsf{GF} + \overrightarrow{TG}$ is implied by results for $\mathsf{GF}$ and $\mathsf{GF} + TG$.

**Table 1.** Complexity of decidable extensions of GF

| Complexity of decidable extensions of $\mathsf{GF}$ | |
|---|---|
| $\mathsf{GF}$ – 2EXPTIME [5] | $\mathsf{GF}^2$ – EXPTIME [5] |
| $\mu\mathsf{GF}$ – 2EXPTIME [8] | $\mu\mathsf{GF}^2$ – EXPTIME [8] |
| $\mathsf{GF} + TG$ – 2EXPTIME [17] | $\mathsf{GF}^2 + TG$ – 2EXPTIME [17]+* |
| $\mathsf{GF} + \overrightarrow{TG}$ – 2EXPTIME [5, 17] | $\mathsf{GF}^2 + \overrightarrow{TG}$ – EXPSPACE [9] |

## References


1. H. Andréka, J. van Benthem, I. Németi. Modal Languages and Bounded Fragments of Predicate Logic. *Journal of Philosophical Logic*, 27, pages 217-274, 1998.
2. J. Balcázar, J. Diaz, J. Gabarró. *Structural Complexity II.* Springer 1990.
3. A. Chandra, D. Kozen, L. Stockmeyer. Alternation. *Journal of the ACM*, 28 (1), pages 114-133, 1981
4. H. Ganzinger, C. Meyer, M. Veanes. The Two-Variable Guarded Fragment with Transitive Relations. *Proc. of 14-th IEEE Symp. on Logic in Computer Science (LICS)*, pages 24-34, 1999.
5. E. Grädel. On the Restraining Power of Guards. *Journal of Symbolic Logic* 64:1719-1742, 1999.
6. E. Grädel, M. Otto, E. Rosen. Undecidability Results on Two-Variable First-Order Logic. *Bulletin of Symbolic Logic*, 3(1), pages 53-69, 1997.

7. E. Grädel, P. Kolaitis, M. Vardi. On the Decision Problem for Two-Variable First Order Logic. *Bulletin of Symbolic Logic*, 3(1), pages 53-96, 1997.
8. E. Grädel, I. Walukiewicz. Guarded Fixpoint Logic. *Proc. of 14-th IEEE Symp. on Logic in Computer Science (LICS)*, pages 45-54, 1999.
9. E. Kieroński. EXPSPACE-Complete Variant of Guarded Fragment with Transitivity. *Proc. of 19-th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 608-619, 2002.
10. D. Kozen. Results on the Propositional $\mu$-Calculus. *Theoretical Computer Science* 27, pages 333-354, 1983.
11. O. Kupferman, A. Pnueli. Once and For All. *Proc. of 10-th IEEE Symp. on Logic in Computer Science (LICS)*, pages 25-35, 1995.
12. H. R. Lewis. Complexity Results for Classes of Quantificational Formulas. *Journal of Computer and System Sciences*, 21, pages 317-353, 1980.
13. G. De Giacomo, F. Masacci. Tableaux and Algorithms for Propositional Dynamic Logic with Converse. *Proc. of 13-th Int. Conf. on Automated Deduction (CADE)*, pages 613-627, 1996.
14. M. Mortimer. On Languages with Two Variables. *Zeitschr. f. Logik und Grundlagen d. Math.*, 21, pages 135-140, 1975.
15. V. R. Pratt. Models of Program Logics. *Proc. of 20-th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 115-122, 1979.
16. M. O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society*, 141, pages 1-35, 1969.
17. W. Szwast, L. Tendera. On the Decision Problem for the Guarded Fragment with Transitivity. *Proc. of 16-th IEEE Symp. on Logic in Computer Science (LICS)*, pages 147-156, 2001.
18. M. Y. Vardi. Reasoning about The Past with Two-Way Automata. *Proc. of 25-th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 628-641, 1998.