Deciding the Verification Problem for Abstract State Machines

Antje Nowack

Mathematische Grundlagen der Informatik,
RWTH Aachen, D-52056 Aachen, Germany,
nowack@informatik.rwth-aachen.de

Abstract. We study the decidability of general verification problems for ASMs. This corresponds to the question whether all computations of an ASM satisfy a property (usually expressed in some temporal logic). Because of undecidability in the general case, one has to identify classes of ASMs and classes of properties in order to receive positive decidability results. One idea for this could be to put simple restrictions on the ASMs as forbidding certain advanced constructs. But it turns out that this does not help much. Additionally, the restrictions in those cases are very strong. Consequently, one has to look for others. Such restrictions may result from techniques which are intended to be used in the proofs of decidability. One proof technique is the reduction to a decidable reasoning problem in logic. In [8] and [10], this has been applied to receive some positive decidability results.
In this paper, we identify further classes of ASMs and classes of properties for which the general verification problem is decidable. Each proof is a reduction to the satisfiability problem of a fragment of linear temporal first-order logic which has respectively been proved to be decidable in [6] or [7].

1 Introduction

Abstract State Machines. Abstract State Machines (ASMs) provide a formal method for transparent design and specification of complex dynamic systems. They combine advantages of informal methods (understandability, executability) with the advantages of formal methods (precision and applicability of mathematical methods and results). The executability of ASM-specifications yields the possibility to find already a part of the errors made in a specification via testing.

However, the ASM-method is a formal method. There is a well defined semantics and this allows the application of mathematical methods to analyze ASMs. In particular, these are often results from mathematical logic as ASMs are based on first-order structures.

The verification problem. The aim of verifying ASMs is to prove that the behaviour of an ASM specification coincides with the desired one. In contrast to testing, one covers *all* possible cases and not only a part.

It would be the ideal case to have an algorithm deciding, for a given ASM and a given property (typically formulated in some temporal logic), whether the ASM satisfies the property.

There are mainly two possible meanings for the question whether "the behaviour of an ASM specification coincides with the desired one":

1. Do all computations of an ASM satisfy a property? (general verification problem)
2. Does the computation of an ASM with a given initial structure satisfy a property? (restricted verification problem)

Model checking is an algorithmic method for checking whether a system satisfies a specification. The system is modeled by a structure (e.g. a transition system) and the specification is a formula in an appropriate logic. Then one checks whether the structure is a model of the formula.

Meanwhile, model checking has become an established industrial practice. Model checking algorithms for finite state systems proceed by a systematic examination of the entire state space. In most cases, it is not possible to exploit this approach for the verification of ASMs as one deals with infinite state spaces. Consequently, one has to look for other ways to carry out model checking.

Before treating the question whether an ASM satisfies a property, one has to consider the question, how far the (restricted or general) verification problem for ASMs is decidable.

Outline. A first idea for receiving decidability results would be to impose simple restrictions on ASMs and logics as forbidding certain advanced constructs. But this leads to positive decidability results only in a very few and rather weak cases. In order to receive further positive results, one has to consider other kinds of restrictions which often do not seem to be very intuitive. The reason for this may be that one does not start with a class of ASMs and a class of properties but with the intention to find some restrictions fitting to an idea for a proof. One possibility for a proof is a reduction to a decidable reasoning problem in logic. If, in this case, we start from the reasoning problem then we may receive appropriate restrictions on ASMs and properties. In [8] and [10], some positive decidability results concerning the general verification problem have been presented. The proofs of those results are reductions to the (finite) satisfiability problem of a logic which is decidable.

In this paper we will also apply this technique to identify further classes of ASMs and fragments of temporal first-order logic for which the general verification problem is decidable. The fragments used in the reductions are fragments of linear temporal first-order logic. The decidability of the satisfiability problems of these fragments has been proved in [6] and [7].

2 Simple Restrictions and Survey of Known Decidability Results

As this paper is written for the ASM community, we assume that the reader is familiar with ASMs. For a detailed introduction to ASMs see e.g. [5].

If no restriction is given then the guard of an if-clause, forall-rule or choice-rule may be any first-order formula (possibly containing quantifiers).

2.1 Simple Restrictions on ASMs

One of the ideas for identifying verifiable classes of ASMs and formalisms could be to forbid certain advanced construction elements or restricting their use in a rather simple way. Advanced construction elements are those which allow rules of the form **forall** $\bar{x} : \varphi(\bar{x})$ **do** Π **endforall**, the choice of an element from the domain of a state or the use of import.

It turns out that the restricted verification problem for full branching time first-order logic restricted to vocabularies containing only constant and relation symbols and the class $\mathcal{K}_0$ of ASMs is decidable. The class $\mathcal{K}_0$ contains all ASMs over vocabularies containing only constant and relation symbols without choice of an element from the domain of a state and without rules of the form **forall** $\bar{x} : \varphi(\bar{x})$ **do** Π **endforall** (but the use of import and the choice of one rule from two are allowed).

Decidability for the general verification problem is received by e.g. restricting the guards of if-clauses to the guarded fragment of first-order logic which is defined in Subsection 3.4.

If one restricts the specification logic to the first-order variant of Hennessy-Milner logic (in addition to first-order constructs, we only allow formulae of the form $\Diamond\varphi$ expressing that there exists a successor satisfying φ) then no restriction on the ASMs is necessary in the case of the restricted verification problem. Already a restriction similar to the guarded fragment of first-order logic leads to decidability in the case of the general verification problem (where the use of forall corresponds to universal quantification and choice to existential quantification). Note that the restriction similar to the guarded fragment contains a restriction to vocabularies containing only constant and relation symbols.

Weaker restrictions on ASMs or the specifying logic that we have received in this way do only lead to undecidability results.

From these considerations, one can see that such simple restrictions do not lead to decidability results in many cases.

The first-order variant of Hennessy-Milner logic (HM-FO) is a rather weak logic. A given formula only allows propositions about a fixed number of steps of an ASM (bounded by the maximal number of nested $\Diamond$). However, there is no uniform bound for all formulae. Therefore, for some applications, HM-FO may already be strong enough.

On the other hand, the class $\mathcal{K}_0$ of ASMs does not provide too many possibilities. The reason is that it is not possible to update the content of locations with elements which are not from the reserve (addressable via import) and which are not the content of a constant. Therefore, all locations of the input state containing at least one such element are static.

For receiving further decidability results, one has to consider classes that result from other kinds of restrictions than those above. Such restrictions have already been considered in [8] and [10]. They are briefly resumed in the rest of this section.

2.2 Nullary Programs

In [8], the class of nullary (ASM-)programs is defined. A nullary program is a nondeterministic basic ASM-program where every dynamic function is nullary. "Basic" means the following in this context:

- the guards in the if-clauses and the choose-rules are not allowed to contain quantifiers
- the use of import or forall is not allowed

M. Spielmann has proven that the general verification problem for the existential and the universal fragment of the fragment of a CTL*-like logic containing only state formulae is decidable. The proof is done via a reduction to the satisfiability problem of existential transitive closure logic.

2.3 ASM-Transducers

In [10], ASM-transducers have been defined. A state of a transducer is a relational database containing the (dynamic) memory for the (static) database of the transducer. In every computation step, the transducer receives from the environment all input relations and reacts by producing output relations and by updating its memory. The memory contains all dynamic data of the transducer necessary to keep track of ongoing transactions.

An ASM-transducer program is a finite set of rules of the form

$$\textbf{if } \varphi \textbf{ then } (\neg)R(\bar{x})$$

where φ is an FO-formula and R is a memory-relation or an output-relation. $R(\bar{x})$ occurs positively if R is an output relation. If a location with an output-relation is not updated by the program then its content is false in the next state. In the case that $R(\bar{x})$ occurs negatively in the above rule, the content all locations $(R, \bar{a})$ such that $\varphi[\bar{x}/\bar{a}]$ is satisfied is set to false. If it occurs positively then the content of these locations is set to true. Free variables are considered as bound by forall. Initially, all memory relations are empty.
In [10], the following restrictions have been introduced:

1. The database is provided and fixed during verification.
2. The maximal input flow, to which an ASM transducer is exposed, is a priori bounded.

3. The arities of the relations are bounded a priori.
4. The specification logic UT is the universal closure of the logic T in which it is not allowed to quantify into temporal contexts.

The general verification problem for ASM-transducers and the logic UT is decidable in the following cases (see [10]):

- The first and the third restriction are satisfied.
- The first and the second restriction are satisfied.
- The second restriction is satisfied and only formulae from the fragment of UT are allowed where every quantifier of a formula is relativized by an atomic input fact.

3 Foundations

3.1 ASMs

Starting from here, all ASMs referred to in this paper are deterministic and satisfy the following conditions:

- the vocabulary of an ASM contains only relation and constant symbols
- import is not used
- the states of an ASM are finite (as the use of import is not allowed, it is not necessary to take the reserve into account)
- the guard of an if-clause or a forall-rule may be any first-order formula possibly containing quantifiers
- updates of the form $c := t$ where c is a constant symbol are not allowed

3.2 Linear Temporal First-Order Logic

Linear temporal first-order logic (LTFO) is obtained by adding to first-order logic FO the following rules for constructing temporal formulae:

If φ and ψ are formulae from LTFO then $\varphi \mathrm{U} \psi$ and $\varphi \mathrm{S} \psi$ are in LTFO.

The semantics for LTFO is the usual one for linear temporal first-order logic (see [6]):

LTFO is interpreted in structures of the form $\mathcal{M} = (\mathcal{F}, D, I)$ where $\mathcal{F} = (W, <)$ is a strict linear order, D is a non-empty set, the domain of $\mathcal{M}$, and I is a function associating with every moment of time $w \in W$ a first-order structure

$$I(w) = (D, P_0^{I(w)}, ..., c_0^{I(w)}, ...),$$

the state at time w, in which $P_i^{I(w)}$, for each i, is a predicate on D of the same arity as P_i and $c_i^{I(w)}$ is an element of D. We require that the interpretations of constants c_i do not change, i.e. $c_i^{I(w)} = c_i^{I(v)}$ for all $w, v \in W$.

An assignment in D is a function a from the set of variables to D. The truth-relation $(\mathcal{M}, w) \models^a \varphi$ in the model $\mathcal{M}$ under the assignment a is defined as follows:

- $(\mathcal{M}, w) \models^a P_i(y_1, ..., y_l)$ iff $P_i^{I(w)}(a(y_1), ..., a(y_l))$ is true in $I(w)$
- $(\mathcal{M}, w) \models^a \varphi \mathrm{S} \psi$ iff there is $v < w$ such that $(\mathcal{M}, v) \models^a \psi$ and $(\mathcal{M}, u) \models^a \varphi$ for every u in the interval $(v, w) = \{u \in W : v < u < w\}$
- $(\mathcal{M}, w) \models^a \varphi \mathrm{U} \psi$ iff there is $v > w$ such that $(\mathcal{M}, v) \models^a \psi$ and $(\mathcal{M}, u) \models^a \varphi$ for every $u \in (w, v)$

The semantics of propositional connectives and quantifiers is as usual.

In this paper, we consider discrete time. Therefore, a run $\rho = (A_i)_{i \in \mathbb{N}}$ of a deterministic ASM is a structure $\mathcal{M}_\rho = ((\mathbb{N}, <), D_\rho, I_\rho)$ where D_ρ is the domain of A_0 and $I_\rho(i) = A_i$ for all $i \in \mathbb{N}$. For real-time systems with continuous time, consult e.g. [2].

Intuitively, for a run ρ of an ASM on a structure A_0, $\mathcal{M}_\rho \models \varphi \mathrm{U} \psi$ means that there is a state B in the run with $B \models \psi$ and until this state is reached, φ holds in every state. It is important that a state is eventually reached in which ψ holds.

Abbreviations. We make use of the common abbreviations of temporal logic:

$$\begin{array}{ll} \mathrm{F}\varphi = \top \mathrm{U} \varphi & \text{(some time in the future)} \\ \mathrm{G}\varphi = \varphi \wedge \neg \mathrm{F} \neg \varphi & \text{(always in the future)} \\ \mathrm{X}\varphi = \bot \mathrm{U} \varphi & \text{(at the next moment)} \end{array}$$

3.3 Computation Graph and Verification Problems

Definition 1 The computation graph $C_\Pi(\mathcal{I})$ of an ASM-program Π and an input structure $\mathcal{I}$ is a temporal first-order model $(\mathcal{F}, D, I)$ where $\mathcal{F} = (W, R, s)$ is a transition system with domain W, transition relation R and pointed element s. $\mathcal{F}$ satisfies Ruv iff $I(v)$ is a successor of $I(u)$ with respect to Π.
D and I are defined analogously to the models for linear temporal logic.

In the case of a deterministic ASM, the transition system collapses to the natural numbers with the usual successor relation and 0 as pointed element. The computation graph corresponds to a run.

Now, we define the restricted verification problem (RVerify) and the general verification problem (GVerify) formally, each for a class of ASMs and a logic.

Definition 2 Let $\mathcal{A}$ be a class of ASMs and $\mathcal{L}$ be a logic or a fragment of a logic. Then

- $\mathrm{RVerify}(\mathcal{A}, \mathcal{L}) = \{(\Pi, \mathcal{I}, \varphi) | \Pi \in \mathcal{A}, \varphi \in \mathcal{L}, C_\Pi(\mathcal{I}) \models \varphi\}$
  denotes the restricted verification problem for $\mathcal{A}$ and $\mathcal{L}$
- $\mathrm{GVerify}(\mathcal{A}, \mathcal{L}) = \{(\Pi, \varphi) | \Pi \in \mathcal{A}, \varphi \in \mathcal{L}, C_\Pi(\mathcal{I}) \models \varphi$ for all input structures $\mathcal{I}$ of $\Pi\}$
  denotes the general verification problem for $\mathcal{A}$ and $\mathcal{L}$

3.4 Guarded Fragment

The decidability of the general verification problem is strongly connected to the decidability of the (finite) satisfiability problem for certain logics.
The first reason is that many proofs are reductions to some (finite) satisfiability problem.

A further reason becomes clear if you consider the rather simple program $\Pi =$ **if** φ **then** res := true **endif** . If φ does not contain res then every computation graph Π satisfies the formula (X res) iff $\neg\varphi$ is not finitely satisfiable.

In the following, we define the guarded fragment of first-order logic for which the (finite) satisfiability problem is decidable. It has been introduced in [1].

Definition 3 Let Υ be a vocabulary containing only constant and relation symbols. The guarded fragment GF of first-order logic is defined inductively as follows:

1. Every relational atomic formula (possibly an equation) belongs to GF.
2. GF is closed under propositional connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
3. If $\psi(\bar{x}, \bar{y})$ is a formula in GF then $\exists\bar{y}(\alpha(\bar{x}, \bar{y}) \wedge \psi(\bar{x}, \bar{y}))$ and $\forall\bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{y}))$ belong to GF provided that $\mathrm{free}(\psi) \subseteq \mathrm{free}(\alpha) = \bar{x} \cup \bar{y}$ and $\alpha(\bar{x}, \bar{y})$ is a relational atomic formula.

Here, $\mathrm{free}(\psi)$ means the set of all free variables of ψ.

Since the introduction of GF in [1], several guarded logics have been defined. They all have in common that every first-order quantifier is relativized by a guard. I.e., they have the form $\exists\bar{y}(\alpha(\bar{x}, \bar{y}) \wedge \psi(\bar{x}, \bar{y}))$ or $\forall\bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{y}))$, $\mathrm{free}(\psi) \cup \{\bar{x}\} \subseteq \mathrm{free}(\alpha)$. In GF, the guards are atomic formulae. As the translation of propositional modal logic to first-order logic (leading to the modal fragment) uses only guarded quantification, the modal fragment is a subset of GF. The guarded fragment generalizes the modal fragment as there is no restriction to the number of variables or the arities of the relations. Based on a number of results (see [1], [3]), Andréka, van Benthem, and Németi assume that the guarded nature of quantification is the main reason for the good model-theoretic and algorithmic properties of modal logic. However, most of the interesting properties are still expressible in GF. Generalizations of GF are the loosely guarded fragment LGF of FO (see [11]) and the clique-guarded fragment CGF of FO (see [4]). GF is strictly contained in CGF and every sentence in LGF is equivalent to a sentence in CGF.

Analogously to the guarded fragment of first-order logic, we can define the guarded fragment of temporal first-order logic GF(LTFO) where all first-order quantifiers are relativized by atomic formulae. Similarly, one can generalize e.g. the clique-guarded fragment of first-order logic to the clique-guarded fragment of temporal first-order logic CGF(LTFO).

4 Guarded ASMs

4.1 Guarded ASMs and the Specification Logic

Definition 4 A guarded ASM is a deterministic ASM satisfying the following conditions:

- update rules have the form $R\bar{c}_1 x \bar{c}_2 x ... \bar{c}_n := t$ where $R \in \Upsilon$ is a relation symbol (the lefthand-side of an update-rule contains at most one free variable)
- the guards of the if-clauses have to be from the guarded fragment of first-order logic
- forall-rules have the form **forall** $x : \varphi(\bar{x}, \bar{y})$ **do** Π **endforall** where φ is a relational atomic formula with $\mathrm{free}(\varphi) \subseteq \mathrm{free}(\Pi) \cup \{\bar{x}\}$

The class of all guarded ASMs is denoted by ASM_1^g.

Note that the restrictions on ASMs introduced in Subsection 3.1 are still valid.

In [6], Hodkinson et al. present fragments $\mathcal{L}$ of linear temporal first-order logic for which satisfiability in $(\mathbb{N}, <)$ over finite, not expanding domains is decidable (abbreviation: $\mathrm{FinSat}(\mathcal{L}, (\mathbb{N}, <)))$. For this purpose, they first define the monodic fragment $\mathcal{TL}_1$ of linear temporal first-order logic.

$\mathcal{TL}_1 :=$ Set of all linear temporal first-order formulae φ such that any subformula of φ of the form $\psi_1 \mathrm{U} \psi_2$ or $\psi_1 \mathrm{S} \psi_2$ has at most one free variable

It is not allowed to quantify into temporal contexts with more than one variable.

Definition 5 The guarded fragment of $\mathcal{TL}_1$ is defined as follows:

$$\mathrm{TGF}_1 := \mathcal{TL}_1 \cap \mathrm{GF}(\mathrm{LTFO})$$

In [7], it has been proved that $\mathrm{FinSat}(\mathcal{TL}_1 \cap \mathrm{CGF}(\mathrm{LTFO}), (\mathbb{N}, <))$ is decidable. This implies the decidability of $\mathrm{FinSat}(\mathrm{TGF}_1, (\mathbb{N}, <))$.

4.2 Decidability of the General Verification Problem

In this subsection, we present the sketch of a proof for the decidability of $\mathrm{GVerify}(\mathrm{ASM}_1^g, \mathrm{TGF}_1)$. It is a reduction to $\mathrm{FinSat}(\mathrm{TGF}_1, (\mathbb{N}, <))$.

Compared to full linear temporal logic, the restrictions on TGF_1 are rather strong but anyway, many interesting properties of ASMs are still expressible. For example, for safety-properties, the restriction of $\mathcal{TL}_1$ is no restriction. The reason is that in the case of safety-properties, there is an FO-sentence φ defining when a state is safe (a state $\mathcal{A}$ is safe iff $\mathcal{A} \models \varphi$). A run is safe iff all of its states are safe. Therefore, a run is safe iff it satisfies $\mathrm{G}\varphi$.

To simplify the argumentation, three definitions are given in advance.

Definition 6 A quasi-atom is a formula of the form $\alpha(\bar{x}) \wedge \bigwedge_{i=1}^{n} y_i^1 \neq y_i^2$ where α is an atomic formula with $\mathrm{free}(\alpha) \subseteq \{\bar{x}\}$ and y_i^1, y_i^2 $(i \in \{1, ..., n\})$ are constant symbols or variables from $\bar{x}$.

Definition 7 An extended substitution is a finite set of simultaneous replacements of variables and constants by terms.

Definition 8 A quasi-atom $\beta(\bar{x}) = \alpha(\bar{x}) \wedge \bigwedge_{i=1}^{n} y_i^1 \neq y_i^2$ is unifiable with an atomic formula $\gamma(\bar{z})$ if α and γ are unifiable if there exists an extended substitution σ such that $\sigma(\alpha)$ and $\sigma(\gamma)$ are identical and $\sigma(\bigwedge_{i=1}^{n} y_i^1 \neq y_i^2)$ is satisfiable.

Lemma 1 *For every guarded ASM Π, there exists an effectively constructable sentence* $\mathrm{consistent}_\Pi \in \mathrm{GF}$ *such that for all states $\mathcal{A}$ of Π the following holds:*
$\mathcal{A} \models \mathrm{consistent}_\Pi$ *iff the update set produced by Π on $\mathcal{A}$ is consistent.*

Proof. The construction of $\mathrm{consistent}_\Pi$ is straightforward. Two rules can only cause a collision if they update the same relation. Therefore, consider every atom appearing on the left-hand side of an update-rule. The formula can be directly gained from the respective guards.
For making clear the construction, consider the following guarded ASM rule Π:

do-in-parallel
  forall $x_1, x_2 : \gamma_1(x_1, x_2)$ **do**
    if $\alpha(x_1, x_2)$ then $Qx_1 :=$ true **endif**
  endforall
  forall $y_1, y_2, y_3 : \gamma_2(y_1, y_2, y_3)$ **do**
    if $\beta(y_1, y_2, y_3)$ **then** $Qy_1 :=$ false **endif**
  endif

Then

$$\mathrm{consistent}_\Pi = \forall x_1 x_2 x_3 (\gamma_2(x_1, x_2, x_3) \\ \rightarrow (\beta(x_1, x_2, x_3) \rightarrow \neg \exists y (\gamma_1(x_1, y) \wedge \alpha(x_1, y))))$$

As the above rule is a guarded ASM rule, this formula is in GF. The construction obviously extends to arbitrary rules. □

Note that $\mathrm{GF} \subseteq \mathrm{TGF}_1$ and therefore, $\mathrm{consistent}_\Pi \in \mathrm{TGF}_1$.

Lemma 2 *For every guarded ASM Π there exist effectively constructable sentences $\psi_\Pi, \chi_\Pi \in \mathrm{TGF}_1$ such that for all states $\mathcal{A}, \mathcal{B}$ of Π and all temporal structures $\mathcal{M} = ((\mathbb{N}, <), D, I)$ with $I(1) = \mathcal{A}$, $I(2) = \mathcal{B}$ and the update set produced by Π on $\mathcal{A}$ is consistent, the following holds:*

$\mathcal{A} \vdash_\Pi \mathcal{B}$ *iff* $(\mathcal{M}, 1) \models \psi_\Pi \wedge \chi_\Pi$ *and for all quasi-atoms $\alpha(\bar{x})$ not unifiable with the lefthand-side of an update-rule in Π $\alpha(\mathcal{A}) = \alpha(\mathcal{B})$*

Proof. First, the formula ψ_Π is constructed. It formulates the changes induced by Π and is defined by induction on Π:

- If Π has the form $(s := t)$ then $\psi_\Pi = (\mathrm{X}s) \leftrightarrow t$.
- If Π has the form **if** g **then** Π' **endif** then $\psi_\Pi = g \rightarrow \psi_{\Pi'}$.

- If Π has the form **do-in-parallel** Π_0 Π_1 then $\psi_\Pi = \psi_{\Pi_0} \wedge \psi_{\Pi_1}$.
- If Π has the form **forall** $\bar{x} : \alpha(\bar{x}, \bar{y})$ **do** $\Pi'(\bar{x}, \bar{y})$ **endforall** then $\psi_\Pi = \forall \bar{x}(\alpha(\bar{x}, \bar{y}) \rightarrow \psi_{\Pi'}(\bar{x}, \bar{y}))$.

ψ_Π is a sentence because Π has no free variables. In ψ_Π, the only temporal operators are those X which are introduced via an update rule. Therefore, subformulae of ψ_Π with a temporal operator outermost are of the form Xs where s is the left-hand side of an update rule in Π. By the definition of guarded ASMs, s and Xs contain at most one variable. Because of the restriction on the guards of Π, $\psi_\Pi \in \mathrm{TGF}_1$.

χ_Π avoids changes in those atoms appearing on the lefthand-side of an update-rule in Π not indicated by the program. □

If the update set produced by a program on a structure is not consistent then nothing changes:

Lemma 3 *For every guarded ASM Π there exists an effectively constructable sentence* $\mathrm{noChanges}_\Pi \in \mathrm{TGF}_1$ *such that for all states $\mathcal{A}, \mathcal{B}$ of Π and for all temporal structures $\mathcal{M} = ((\mathbb{N}, <), D, I)$ with $I(1) = \mathcal{A}$ and $I(2) = \mathcal{B}$, the following holds:*

$(\mathcal{M}, 1) \models \mathrm{noChanges}_\Pi$ *iff for all atoms $\alpha(\bar{x})$ appearing on the lefthand-side of an update-rule in Π $\alpha(\mathcal{A}) = \alpha(\mathcal{B})$*

Theorem 1 $\mathrm{GVerify}(\mathrm{ASM}_1^g, \mathrm{TGF}_1)$ *is decidable.*

Proof. The decidability of $\mathrm{GVerify}(\mathrm{ASM}_1^g, \mathrm{TGF}_1)$ is proved via a reduction to $\mathrm{FinSat}(\mathrm{TGF}_1, (\mathbb{N}, <))$.
Let Π be a guarded ASM program and φ be a TGF_1-formula over the same vocabulary as Π. A formula Φ_Π will be constructed such that

$$(\Pi, \varphi) \in \mathrm{GVerify}(\mathrm{ASM}_1^g, \mathrm{TGF}_1) \text{ iff } \mathrm{trans}_\Pi(\Phi_\Pi \wedge \neg\varphi) \notin \mathrm{FinSat}(\mathrm{TGF}_1, (\mathbb{N}, <))$$

where trans_Π transforms a formula in TGF_1 into another formula in TGF_1 with respect to Π as follows. $\mathrm{trans}_\Pi(\Phi_\Pi \wedge \neg\varphi) \in \mathrm{FinSat}(\mathrm{TGF}_1, (\mathbb{N}, <))$ iff there is a model $\mathcal{M} = ((\mathbb{N}, <), D, I)$ of $\Phi_\Pi \wedge \neg\varphi$ such that for all quasi-atoms $\alpha(\bar{x})$ not unifiable with the lefthand-side of an update rule in Π, $\alpha(I(n)) = \alpha(I(m))$ for all $m, n \in \mathbb{N}$.
Note that this does *not* imply that for every model $\mathcal{M} = ((\mathbb{N}, <), D, I)$ of $\mathrm{trans}_\Pi(\Phi_\Pi \wedge \neg\varphi)$ and for all quasi-atoms $\alpha(\bar{x})$ not unifiable with the lefthand-side of an update rule in Π, $\alpha(I(n)) = \alpha(I(m))$ for all $m, n \in \mathbb{N}$.
The formula $\Phi_\Pi \in \mathrm{TGF}_1$ has the form

$$\Phi_\Pi = \mathrm{G}((\mathrm{consistent}_\Pi \rightarrow (\psi_\Pi \wedge \chi_\Pi)) \wedge ((\neg\mathrm{consistent}_\Pi) \rightarrow \mathrm{noChanges}_\Pi))$$

where $\mathrm{consistent}_\Pi$ is the GF-sentence from Lemma 1, ψ_Π and χ_Π are the sentences from Lemma 2 and $\mathrm{noChanges}_\Pi$ is the one from Lemma 3.
Because $(\mathrm{consistent}_\Pi \rightarrow (\psi_\Pi \wedge \chi_\Pi)) \wedge ((\neg\mathrm{consistent}_\Pi) \rightarrow \mathrm{noChanges}_\Pi)$ is a TGF_1-formula without free variables, Φ_Π is an TGF_1-formula. □

4.3 Other Verifiable Classes of ASMs

In [6], [7] and [12], the decidability of $\mathrm{FinSat}(\mathcal{L}, (\mathbb{N}, <))$ has been proved for further fragments $\mathcal{L}$ of $\mathcal{TL}_1$. For many of these fragments $\mathcal{L}$, a class of ASMs can be defined for which the respective fragment (or to be more precise: the decidability of $\mathrm{FinSat}(\mathcal{L}, (\mathbb{N}, <)))$ implies the decidability of the general verification problem.

The results for the guarded fragment directly extend to the clique-guarded fragment and ASMs analogously defined to guarded ASMs but the condition for the guards in the if-clauses and the forall-rules are generalized according to the clique-guarded fragment. This leads to the definition of clique-guarded ASMs.

A further example is the monadic fragment of $\mathcal{TL}_1$. Monadic ASMs are deterministic ASMs over vocabularies containing only constant symbols and *unary* relation symbols. Furthermore, the use of equality is not allowed.

The proof of the decidability of the corresponding general verification problem can be done analogously to the one for guarded ASMs and it becomes even easier. The transformation of the constructed formula (as carried out in the proof for guarded ASMs) is not needed as one can avoid arbitrary changes by a formula linked conjunctively to the constructed one. Equality is only needed in a very restricted way. Namely, where the left-hand-side is a constant. Decidability of the satisfiability problem over finite, not expanding domains in this case can be proved in the same way as the one for satisfiability without equality (this could also be eliminated by disallowing the occurrence of constants in the lefthand-side of an update-rule).

4.4 Extension of Guarded ASMs by Interaction

The main difference between ASM transducers (see [10] or subsection 2.3) and guarded ASMs is that ASM transducers provide the possibility to interact with the environment. The other differences, like the static database part, the output part or the memory part of the transducers, are not really additional features.

Guarded ASMs can be extended by interaction quite easily. Instead of one input state (over the the vocabulary Υ of the program) for a run, there is a sequence of input states $\bar{I} = I_1 I_2 ...$ over the input vocabulary $\Upsilon_{in} \subseteq \Upsilon$. The notion of a run has to be modified as follows:
Let ρ be a sequence $(A_i)_{i \in \mathbb{N}}$ of states over Υ. ρ is the run of Π on $\bar{I}$ if

- $A_i|_{\Upsilon_{in}} = I_i$
- Let B_{i+1} be the successor of A_i with respect to Π. Then $A_{i+1}|_{\Upsilon - \Upsilon_{in}} = B_{i+1}|_{\Upsilon - \Upsilon_{in}}$ for all $i \in \mathbb{N}$.

The decidability result for guarded ASMs can be directly transferred to interactive guarded ASMs. Note that in the case of interactive ASMs, the general verification problem has to be slightly modified. Not only all input states have to be taken into account but all input states and sequences of states over the input vocabulary.

Analogously, one can extend the classes from subsection 4.3 by interaction.

4.5 Comparison to Known Decidability Results

Nullary ASMs. In subsection 2.2, the results on nullary programs, originally introduced in [8], have been shortly resumed.

The result on nullary programs and the results obtained in this paper are merely comparable. Nullary programs allow the update of locations consisting only of a constant but no other location can be updated. No ASM-program except Skip is a guarded ASM-program and a nullary program. The updates in nullary ASMs are forbidden in guarded ASMs.

ASM-Transducers. On the first view, one may have the impression that the results from [10] are stronger than those presented in this paper. The reason for this impression could be the restrictions on the use of variables in interactive clique-guarded ASMs. But this impression is wrong.

The first and obvious point is that the verifiable properties are not the same. For ASM-transducers we can verify properties expressible in UT, for interactive clique-guarded ASMs we can verify properties expressible in $\mathrm{TCGF}_1 := \mathcal{TL}_1 \cap \mathrm{CGF(LTFO)}$. The two fragments do not coincide and for each of the two fragments there are properties expressible in the one but not in the other.

But this is not the essential difference. For demonstrating this, we consider an example. Using interactive clique-guarded ASMs, we want to model a very simple system governing the student data of a university.

At the end of his studies, a student has to take part in the exams of at least r combinable lectures. Only if he passes all exams then he receives a certificate. If he fails then he has the possibility to do r combinable exams again. If this was already his second try then he has to leave the university without a certificate. After every term, the results of the exams are submitted to the system and the current certificates and aborts are emitted. Internally, students are identified via their registration number. This allocation is static. Output data like aborts or certificates does not contain registration numbers.

In the following, we give an ASM-transducer and an interactive clique-guarded ASM which are equivalent and both try to model the above described system.

First, we give the ASM-transducer.

$\boxed{\textbf{relations}}$

input	: fail/2, pass/2
database	: number/2, combinable/r
memory	: try1/1, try2/1
output	: certificate/1, abort/1
log	: fail/2, pass/2, certificate/1, abort/1

$\boxed{\textbf{memory rules}}$

if exam(n) **then** try1(n)

if exam(n) $\wedge$ try1(n) **then** try2(n)

output rules

if number$(s,n) \wedge (\exists l$ fail$(n,l)) \wedge$ try1(n) **then** abort(s)

if number$(s,n) \wedge \neg$try2$(n) \wedge \neg(\exists l$ fail$(n,l))\wedge$
$(\exists l_1 ... l_r$ combinable$(l_1, ..., l_r) \wedge \bigwedge_{i\in\{1,...,r\}}$ pass$(n,l))\wedge$ **then**
certificate(s)

From the informal description, it should be clear how the transducer works. Nevertheless, we will partially resume the intended meaning of the relations.

For a registration number n and a lecture l, pass(n,l) is true if the student with registration number n has passed the exam belonging to l in the current term. fail(n,l) is true if he has failed the exam. The binary relation number/2 stores the allocation of registration numbers and students. The memory relations keep track of the number of terms in which a student has taken part in the exams. try1(n) is true if the student belonging to number n has already tried once to pass the exams (albeit whether passed or failed), try2(n) is true if he has taken part in the exams of two terms.

The above ASM-transducer can be rewritten to an equivalent interactive clique-guarded ASM with the same input vocabulary. For better readability, do-in-parallel is left away and the rules are split into memory rules and output rules.

memory rules

forall n : true **do**
  if exam(n) **then** try1(n) := true **endif**
  if exam$(n) \wedge$ try1(n) **then** try2(n) := true **endif**
endforall

output rules

forall s,n : number(s,n) **do**
  if try1$(n) \wedge \exists l$ fail(n,l) **then**
    abort(s) := true
  else abort(s) := false
  endif
  if $\neg$try2$(n) \wedge \neg(\exists l$ fail$(n,l))\wedge$
    $(\exists l_1 ... l_r$ combinable$(l_1, ..., l_r)\wedge$
      $\bigwedge_{i\in\{1,...,r\}}$ pass$(n,l))$ **then**
    certificate(s) := true
  else certificate(s) := false
  endif
endforall

forall s : true **do**
  if $\neg\exists n$ number(s,n) **then**
    certificate(s) := false
    abort(s) := false
  endif
endforall

where exam(n) abbreviates the formula $(\exists l$ pass$(n,l)) \vee (\exists l$ fail$(n,l))$.

With the results presented in this paper, we can verify any property expressible as a TCGF_1-formula without any further restrictions. The results in [10] do not yield this possibility for UT as we have to care for the restrictions introduced there. In our example, a fixed database would determine the number of students. A limited input flow would imply a limit on the number of carried out exams albeit whether passed or failed. Both restrictions are not suitable in the above case.

An example for a property expressible in TCGF_1 is that if someone fails for the second time then he will never receive a certificate in the future. It can be formalized by the following TCGF_1-formula.

$$\mathrm{G}\ \forall n(\mathrm{failed}(n) \rightarrow \mathrm{X\ G\ try1}(n)) \wedge$$
$$\mathrm{G}\ \forall sn(\mathrm{number}(s,n) \rightarrow ((\mathrm{try1}(n) \wedge \mathrm{failed}(n)) \rightarrow \neg\mathrm{F\ certificate(s)}))$$

where $\mathrm{failed}(n)$ abbreviates the formula $\exists l\ \mathrm{fail}(n,l)$.

Note, furthermore, that, in contrast to ASM-transducers the memory relations of an interactive clique-guarded ASM do not need to be initially empty but if you want to verify a property φ only for this case then you can do it by verifying $(\bigwedge_{R\in\Upsilon_{mem}} \neg\exists x_1...x_{\mathrm{arity}(R)} Rx_1...x_{\mathrm{arity}(R)}) \rightarrow \varphi$ instead of φ (where Υ_{mem} is the set of memory-relations of the transducer).

4.6 Complexity

[7] implicitly provides an algorithm deciding whether $\varphi \in \mathrm{FinSat}(\mathrm{TGF}_1, (\mathbb{N}, <))$ with deterministic time in $2^{2^{O(2^{|\varphi|\cdot\log|\varphi|})}}$.
The size of Φ_Π is linear in $|(\Pi,\varphi)|$ (the size of (Π,φ)). In the worst case, the size of $\mathrm{trans}_\Pi(\Phi_\Pi \wedge \neg\varphi)$ is in $O(2^{|(\Pi,\varphi)|})$.
The above implies that the worst-case time complexity of $\mathrm{GVerify}(\mathrm{ASM}_1^g, \mathrm{TGF}_1)$ is in $2^{2^{O(2^n\cdot 2^n)}}$.

The finite satisfiability problem for the guarded fragment of first-order logic is complete for 2EXPTIME (double exponential time). This directly implies that $\mathrm{GVerify}(\mathrm{ASM}_1^g, \mathrm{TGF}_1)$ is 2EXPTIME-hard.

The other fragments do not provide much better results concerning their worst-case complexity. E.g., the general verification problem for interactive monadic ASMs and the monadic fragment of $\mathcal{TL}_1$ is EXPSPACE-complete.

5 Conclusion

We have observed that the decidability results for both versions of the verification problem are tightly connected to decidability results in mathematical logic. This is the main reason for putting relatively strong restrictions on the ASMs and the specification logic in order to receive positive decidability results.

As the worst-case complexities of the satisfiability problem for expressive logics are usually quite high (if decidable), the worst-case complexities for both versions of the verification problem are also quite high.

To receive lower worst-case complexities, one has to accept a trade-off concerning the expressiveness. Then the question for applicability becomes more and more difficult to answer. But one should also take into account that these high complexities reflect only the worst-case. For applications, one would be more interested in the average-case complexity. This one strongly depends on the particular applications.

This is what one should have in mind when trying to identify further verifiable fragments.

Acknowledgements. I want to thank Erich Grädel for many helpful remarks on earlier drafts of this paper. I also want to thank the anonymous referees for their useful suggestions.

References

1. H. Andréka, J. van Benthem, and I. Németi. Modal Languages and Bounded Fragments of Predicate Logic. Technical Report ILLC Research Report ML-96-03, ILLC, 1996.
2. D. Beauquier and A. Slissenko. A First Order Logic for Specification of Timed Algorithms: Basic Properties and a Decidable Class. *Annals of Pure and Applied Logic*, 113(1-3):13–52, 2002.
3. E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.
4. E. Grädel. Guarded fixed point logic and the monadic theory of trees. *Theoretical Computer Science*, 288:129–152, 2002.
5. Y. Gurevich. May 1997 Draft of the ASM Guide. Technical Report CSE-TR-336-97, EECS Dept, University of Michigan, 1997.
6. I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106:85–134, 2000.
7. I.Hodkinson. Monodic packed fragment with equality is decidable. Preprint, 2001.
8. M. Spielmann. Automatic Verification of Abstract State Machines. In *Proceedings of 11th International Conference on Computer-Aided Verification (CAV '99)*, volume 1633 of *LNCS*, pages 431–442. Springer-Verlag, 1999.
9. M. Spielmann. Model Checking Abstract State Machines and Beyond. In *International Workshop on Abstract State Machines ASM 2000*, LNCS, pages 323–340. Springer-Verlag, 2000.
10. M. Spielmann. Verification of Relational Transducers for Electronic Commerce. In *19th ACM Symposium on Principles of Database Systems PODS 2000, Dallas*, pages 92–93. ACM Press, 2000.
11. J. van Benthem. Dynamic Bits and Pieces. Technical Report LP-97-01, ILLC, University of Amsterdam, 1997.
12. F. Wolter and M. Zakharyaschev. Axiomatizing the monodic fragment of first-order temporal logic, 2000. Submitted.