

A Semantic Characterization of Elementary Wide-Step ASMs

Andreas Glausch

Humboldt-Universität zu Berlin
Institut für Informatik
glausch@informatik.hu-berlin.de

Abstract. Abstract State Machines (ASMs) describe their behavior by a simple, yet expressive program syntax. While such syntax is useful for the application of ASMs as a modeling and specification language, it often complicates theoretical considerations. Gurevich solved this problem for the class of *sequential small-step ASMs* by proposing an elegant semantic axiomatization for this class. Later, Blass and Gurevich generalized this result to *sequential wide-step ASMs*, which additionally may use a **forall**-operator ranging over elements of finite multisets. In this paper we present a semantic characterization of wide-step ASMs that employ the classical **forall**-operator ranging over a (possibly infinite) subuniverse of the state. We call this variant *elementary wide-step ASMs*. We apply the characterization to identify characteristic properties of elementary wide-step ASMs.

1 Introduction

In a nutshell, an Abstract State Machine (ASM) consists of two components: A set of states, represented semantically by *first-order structures*, and a set of steps, represented syntactically by an *ASM rule* [11]. This combination of a highly general notion of states and an expressive and intuitive program syntax makes ASMs a successful system design and analysis methodology [8, 6].

While the elaborate syntax of ASM rules is inevitable for the applicability of ASMs as a practical modeling language, it often complicates theoretical considerations:

- Usually, the same semantic fact can be represented syntactically in different ways. For instance, the ASM rules “**if a=b then a:=b**” and “**a:=a**” describe the same (exceptionally useless) algorithm. Hence, syntax tends to bear much redundancy, causing redundancy in proofs and argumentations.
- Often there exist several variants of the syntax. For instance, parallel execution of two ASM rules P and Q can either be written as “**par P Q endpar**”, or as “**P par Q**”. There is no “canonical syntax of ASMs”.
- The syntax of a particular class of ASMs is usually defined inductively. As a consequence, proving interesting features for this class (e.g. closure properties) requires an induction on the complete syntax. This may cause large and less intuitive proofs.

- Comparing the expressive power of two variants of ASMs requires transforming the syntax of one variant into the syntax of the other, and then proving the correctness of this transformation. It is not possible to compare both variants directly on a semantic level. Similar problems arise when ASMs are related to other machine models.

These problems are not exclusive to ASMs at all. The same difficulties appear in classical logic, for instance when comparing the expressive power of different logics.

Some of the above problems (in particular the third problem) can be alleviated by restricting the syntax of ASMs to a bare minimum syntax, avoiding any syntactic sugar such as nesting of rules. However, characterizing ASMs independently of *any* syntax, i.e. a *semantic characterization* of ASMs, allows a more convenient solution to all of the above problems. The first semantic characterization of a class of ASMs has been presented by Gurevich in [13]: He defines the class of *sequential small-step algorithms* by three surprisingly simple and general axioms, and proves this class to be equivalent to sequential small-step ASMs. These axioms do not involve any syntax except for ground terms, and allow reasoning about sequential small-step ASMs on a semantic level. For instance, in [15] Rosenzweig and Runje took advantage of the axioms in an elegant proof of the linear-speedup theorem for sequential small-step ASMs.

In subsequent work, Blass, Gurevich, and others identified similar characterizations for other variants of ASMs, including *bounded-nondeterministic*, *wide-step*, and *interactive* versions [14, 1–3]. Boker and Dershowitz characterized in [5] *effective* ASMs capturing classical computable behaviour. We showed in [9] and [10] that Gurevich’s characterization from [13] can be generalized to *distributed* and *unbounded-nondeterministic* ASMs, with only moderate adjustments of the axioms.

To date, only few applications of these characterizations have been proposed in literature. In [16] Rosenzweig and Runje present an abstract model of computational cryptography. In their model, an algorithm violating the axioms for interactive algorithms as given in [2] would break the security of an encryption scheme. In [4] Boker and Dershowitz extend the axioms for sequential small-step algorithms to obtain an abstract formalization of the Church-Turing-Thesis. We believe that semantic characterization of ASMs is a powerful tool for reasoning about ASMs and algorithms in general, whose benefit has not been fully identified yet.

2 Contribution of this Paper

In this paper we present a semantic characterization of ASMs that extend sequential small-step ASMs by the classical **forall**-operator. We call this variant *elementary wide-step ASMs*. The **forall**-operator has been introduced in [12] and is integral to the ASM design and analysis methodology [8]. The attribute *wide-step* has been suggested in [2] in order to emphasize the fact that **forall** allows for unboundedly many updates in a single step. As the **forall**-operator

occurs frequently in various ASM models, we consider it as an interesting target for theoretical investigation.

In [1] Blass and Gurevich proposed a semantic characterization of wide-step ASMs using a `forall`-operator that deviates from the classical version in [12]. Essentially, `forall` in [1] ranges over the elements of a finite multiset, whereas `forall` in [12] may range over the elements of a (possibly infinite) subuniverse. In the present paper we aim at characterizing elementary wide-step ASMs, which employ the classical version of the `forall`-operator. The major differences between elementary wide-step ASMs considered here and wide-step ASMs considered in [1] are:

- In [1] states are required to comprise a *multiset background*. That is, each state provides operations for the construction of (hereditarily) finite multisets over the universe. We lift this requirement here.
- The `forall`-operator in [1] quantifies over elements of finite multisets. We allow `forall` to quantify also over infinitely many elements of the universe.
- We do not allow *comprehension terms* as in [1]. A comprehension term defines a multiset by the property that its elements satisfy, i.e. such terms allow intensional definition of multisets.

According to the first two points, we lift two restrictions of wide-step ASMs in [1]. On the other hand, as we disallow comprehension, some expressive power of wide-step ASMs from [1] is lost. For example, elementary wide-step ASMs are not able to evaluate bounded first-order formulas as presented in [1]. We intend to address this issue in subsequent work.

The rest of this paper is organized as follows. We introduce elementary wide-step ASMs in the next section, and axiomatize the notion of elementary wide-step algorithms in Sect. 4. In Sect. 5 we present the theorem stating the equivalence of both notions, and apply the theorem to verify non-trivial properties of elementary wide-step ASMs. We prove the equivalence theorem in the last section.

3 Elementary Wide-Step ASMs

In this section we recall and clarify the basic notions and notations used throughout this paper, and introduce the syntax and semantics of elementary wide-step ASMs. For the sake of brevity, we sometimes skip the predicate “elementary” where it causes no confusion.

As usual, a *signature* Σ is a nonempty, finite set of function symbols \mathbf{f} , each of which equipped with an arity $n \in \mathbb{N}$. A Σ -*structure* S comprises a non-empty set U_S (the *universe of* S), and associates each n -ary function symbol \mathbf{f} from Σ with an n -ary function $\mathbf{f}_S : U_S^n \rightarrow U_S$. For a set V of variables, *terms* over Σ and V are constructed inductively in the usual way. For a term t , a Σ -structure S , and a *variable assignment* $\alpha : V \rightarrow U_S$, we denote the *value* of t at S and α by $\llbracket t \rrbracket_S^\alpha$. In case t is *ground* (i.e. t contains no variables), we may skip α and write $\llbracket t \rrbracket_S$. A *Boolean formula* ϕ consists of term equations of the form $t_1 = t_2$, connected by the usual propositional connectives such as \neg , \wedge , and \vee . For S and α as above,

we write $S, \alpha \models \phi$ to denote that ϕ is satisfied at S and α . An *isomorphism* between two Σ -structures R and S is a bijective mapping $i : U_R \rightarrow U_S$ that preserves the functions of R , i.e. $i(\mathbf{f}_R(u_1, \dots, u_n)) = \mathbf{f}_S(i(u_1), \dots, i(u_n))$ for all n -ary function symbols and all $u_1, \dots, u_n \in U_R$.

We employ of the usual notion of updates as given in [11]. For a Σ -structure S , an n -ary function symbol \mathbf{f} , and $u_0, \dots, u_n \in U_S$, the triple $(\mathbf{f}, [u_1, \dots, u_n], u_0)$ is an *update of S* . A set Δ of updates of S is an *update set of S* . Two updates $(\mathbf{f}, [u_1, \dots, u_n], u_0)$ and $(\mathbf{f}, [u_1, \dots, u_n], u'_0)$ with $u_0 \neq u'_0$ are *conflicting*. An update set is *consistent* if it contains no conflicting updates. For a consistent update set Δ of S , the *application $S + \Delta$* denotes the Σ -structure obtained from S by applying all updates from Δ . In case Δ is inconsistent, $S + \Delta$ denotes S .

In the following we present the syntax and semantics of elementary wide-step ASMs. As stated previously, this variant is a straight extension of sequential small-step ASMs by the `forall`-operator. For more details on the `forall`-operator, we refer to [12] and Chapter 2.4 in [8].

The syntax of *elementary wide-step ASM rules* is defined inductively in the usual way: For n -ary function symbols \mathbf{f} , for terms t_0, \dots, t_n , Boolean formulas ϕ , variables x , and elementary wide-step ASM rules `RULE`, `RULE1`, \dots , `RULEn`, the following symbol sequences are also elementary wide-step ASM rules:

$$\begin{aligned} \mathbf{f}(t_1, \dots, t_n) := t_0 & \quad (\text{assignment rule}), \\ \text{if } \phi \text{ then RULE} & \quad (\text{conditional rule}), \\ \text{par RULE}_1 \dots \text{RULE}_n \text{ endpar} & \quad (\text{block rule}), \\ \text{forall } x \text{ with } \phi \text{ do RULE} & \quad (\text{forall rule}). \end{aligned}$$

We consider the usual semantics of ASM rules: For each Σ -structure and each variable assignment α , an elementary wide-step ASM rule `RULE` yields an update set $\llbracket \text{RULE} \rrbracket_S^\alpha$, the *interpretation of `RULE` at S and α* . Formally, the semantics of `RULE` is inductively defined as follows, where $\alpha[x \mapsto u]$ denotes the variable assignment obtained from α by overriding the value of x by u :

$$\begin{aligned} \llbracket \mathbf{f}(t_1, \dots, t_n) := t_0 \rrbracket_S^\alpha & =_{\text{def}} \{ (\mathbf{f}, \llbracket t_1 \rrbracket_S^\alpha, \dots, \llbracket t_n \rrbracket_S^\alpha, \llbracket t_0 \rrbracket_S^\alpha) \} \\ \llbracket \text{if } \phi \text{ then RULE} \rrbracket_S^\alpha & =_{\text{def}} \begin{cases} \llbracket \text{RULE} \rrbracket_S^\alpha & , \text{ in case } S, \alpha \models \phi \\ \emptyset & , \text{ otherwise} \end{cases} \\ \llbracket \text{par RULE}_1 \dots \text{RULE}_n \text{ endpar} \rrbracket_S^\alpha & =_{\text{def}} \bigcup_{1 \leq i \leq n} \llbracket \text{RULE}_i \rrbracket_S^\alpha \\ \llbracket \text{forall } x \text{ with } \phi \text{ do RULE} \rrbracket_S^\alpha & =_{\text{def}} \bigcup_{u \in U_S \text{ with } S, \alpha[x \mapsto u] \models \phi} \llbracket \text{RULE} \rrbracket_S^{\alpha[x \mapsto u]}. \end{aligned}$$

We use “`forall x do RULE`” as an abbreviation for “`forall x with $x = x$ do RULE`”. As usual, the rule “`forall x with ϕ do RULE`” *binds* the occurrences of x in ϕ and `RULE`. A rule is *closed* if all variable occurrences are bound by a `forall`-rule. A closed rule `RULE` can be interpreted at a Σ -structure S independently of any variable assignment α . In that case we may skip α and write $\llbracket \text{RULE} \rrbracket_S$ to denote the update set of `RULE` at S .

An *elementary wide-step ASM* \mathcal{M} then comprises

- a signature $\Sigma_{\mathcal{M}}$,
- a nonempty set of $\Sigma_{\mathcal{M}}$ -structures $\mathcal{S}_{\mathcal{M}}$, closed under isomorphism (the *states*),
- a nonempty set $\mathcal{J}_{\mathcal{M}} \subseteq \mathcal{S}_{\mathcal{M}}$, closed under isomorphism (the *initial states*),
- a closed wide-step ASM rule $\text{RULE}_{\mathcal{M}}$.

For a state $S \in \mathcal{S}_{\mathcal{M}}$, we call $\llbracket \text{RULE}_{\mathcal{M}} \rrbracket_S$ the *update set of \mathcal{M} at S* , denoted by $\Delta_{\mathcal{M}}(S)$. We assume that, for all states S of \mathcal{M} , the application $S + \Delta_{\mathcal{M}}(S)$ also is a state of \mathcal{M} . A *run* of \mathcal{M} is an infinite sequence of states $S_0 S_1 S_2 \dots$ with $S_0 \in \mathcal{J}_{\mathcal{M}}$ and $S_{i+1} = S_i + \Delta_{\mathcal{M}}(S_i)$ for $i = 0, 1, 2, \dots$.

4 Elementary Wide-Step Algorithms

In this section define the class of *elementary wide-step algorithms* by four general and semantic axioms, and compare them to Gurevich’s axioms for sequential small-step algorithms from [13]. As it turns out, much of the original axioms can be preserved when considering wide-step algorithms. In the forthcoming Section 5 we show that wide-step algorithms indeed capture wide-step ASMs as introduced in the previous section.

First of all, we briefly recall Gurevich’s axioms for sequential small-step algorithms in order to compare them to our axioms for wide-step algorithms. Slightly rearranged, the axioms for sequential small-step algorithms can be formulated as follows:

Axiom S1 A sequential small-step algorithm comprises a set \mathcal{S} of Σ -structures (the *states*), and a subset $\mathcal{J} \subseteq \mathcal{S}$ (the *initial states*).

Axiom S2 The algorithm determines for each state $S \in \mathcal{S}$ a next-state $S' \in \mathcal{S}$ such that S and S' have the same universe.

Axiom S3 The state sets \mathcal{S} and \mathcal{J} are closed under isomorphism, and each isomorphism $i : R \rightarrow S$ between states S and R also is an isomorphism $i : R' \rightarrow S'$ between their next-states.

The decisive fourth axiom for sequential algorithm is *bounded exploration*. Intuitively, the axiom states that only a finite set T of ground terms is explored at all states S to determine the next-state S' . Formally, the axiom employs the notions of *coincidence on T* and *difference* between states [13]: Two states R and S *coincide on T* iff $\llbracket t \rrbracket_S = \llbracket t \rrbracket_R$ for all $t \in T$. The *difference* $\Delta(S, S')$ between two states S and S' over the same universe is the least set of updates such that $S' = S + \Delta(S, S')$. Gurevich’s fourth axiom then reads:

Axiom S4 There is a finite set T of ground terms such that for all states S and R with $S =_T R$ holds $\Delta(S, S') = \Delta(R, R')$.

Hence, in each state only the values of the terms in T determine the updates applied to that state.

Gurevich calls any entity satisfying the above Axioms S1 – S4 a sequential small-step algorithm. In the following we show how these axioms can be adjusted

in order to capture elementary wide-step algorithms. Thereby, the first three axioms can be taken over nearly unchanged, whereas the fourth axiom requires more adjustment.

4.1 The Axioms for Elementary Wide-Step Algorithms

The first axiom for wide-step algorithms is fully identical to axiom S1.

Axiom W1 (states) *An elementary wide-step algorithm A determines*

- a nonempty set \mathcal{S}_A , the states of A ,
- a nonempty set $\mathcal{I}_A \subseteq \mathcal{S}_A$, the initial states of A ,
- a signature Σ_A such that each state of A is a Σ_A -structure.

Slightly deviating from Axiom S2, we demand that a wide-step algorithm provides for each state an update set.

Axiom W2 (updates) *An elementary wide-step algorithm A provides for each state S an update set $\Delta_A(S)$ of S such that $S + \Delta_A(S)$ is a state of A .*

Note the difference to axiom S2: For a sequential algorithm, the update set applied to a state S is given only *implicitly* by the difference $\Delta(S, S')$. By contrast, Axiom W2 requires a wide-step algorithm to *explicitly* specify the update set $\Delta_A(S)$ that is applied to S . Thus, a wide-step algorithm provides more information at S than only its next-state S' : $\Delta_A(S)$ may contain trivial updates (i.e. updates that do not change the state), and may contain conflicting updates. Such updates cannot be observed when considering only the difference $\Delta(S, S')$. However, this modification is not new and also appears in [2] and [15].

There are several reasons why we decided for the above modification. *First*, as stated in [2], trivial updates are relevant when composing algorithms in parallel. *Second*, several different conventions have been proposed in order to deal with conflicting updates (e.g., leave state unchanged, do not perform any step, reach an error state, choose nondeterministically one of the conflicting updates). As Axiom W2 considers update sets instead of next-states, we do not have to stick to one of these conventions. There is a third, more technical reason, which we discuss in the following section.

The third axiom is merely a canonical adjustments of Axiom S3, with next-states replaced by update sets. More precisely, Axiom W3 demands that the states are closed under isomorphism, and that isomorphic states yield isomorphic update sets. To this end, we extend each isomorphisms $i : R \rightarrow S$ between two states canonically to updates $\delta = (\mathbf{f}, [u_1, \dots, u_n], u_0)$ of R by

$$i(\delta) \stackrel{\text{def}}{=} (\mathbf{f}, [i(u_1), \dots, i(u_n)], i(u_0)).$$

The third axiom then can be formulated as follows:

Axiom W3 (isomorphism) *For an elementary wide-step algorithm A , the sets \mathcal{S}_A and \mathcal{I}_A are closed under isomorphism. For isomorphic states $S, R \in \mathcal{S}_A$ with an isomorphism $i : R \rightarrow S$ holds $\delta \in \Delta_A(R)$ iff $i(\delta) \in \Delta_A(S)$.*

The fourth axiom significantly differs from Axiom S4. Intuitively, we require a wide-step algorithm to explore for each update only a bounded subuniverse of the current state. This intuition already points out a decisive difference to the classical Axiom S4: Instead of exploration of syntax (a set of terms), Axiom W4 considers exploration of semantics (a subuniverse). In this sense, Axiom W4 is “more semantic” than Axiom S4.

Formally, we introduce the notion of *coincidence on a set W* , in analogy to Gurevich’s notion of coincidence on a set T of ground terms. We say that two states S, R coincide on a set W ($S =_W R$ for short) in case $W \subseteq U_S \cap U_R$ and for all n -ary function symbols and all $u_0, \dots, u_n \in W$ holds

$$\mathbf{f}_S(u_1, \dots, u_n) = u_0 \quad \Leftrightarrow \quad \mathbf{f}_R(u_1, \dots, u_n) = u_0.$$

By help of this notion, we now present the final axiom. In a state S , only a bounded subuniverse W of S is explored in order to determine whether an update δ is included in the update set $\Delta_{\mathcal{A}}(R)$.

Axiom W4 (locally bounded-exploration) *For each elementary wide-step algorithm \mathcal{A} there exists a constant $k_{\mathcal{A}} \in \mathbb{N}$ such that for all $S \in \mathcal{S}_{\mathcal{A}}$ and for all $\delta \in \Delta_{\mathcal{A}}(S)$ the following holds: There is a subuniverse $W \subseteq U_S$ with $|W| \leq k_{\mathcal{A}}$ such that for all $R \in \mathcal{S}_{\mathcal{A}}$ with $S =_W R$ holds $\delta \in \Delta_{\mathcal{A}}(R)$.*

Adopting the naming from [13], we say that W is a *witness* for the update δ at state S . We do not impose any further requirements to wide-step algorithms: We call *any* entity satisfying the axioms W1 – W4 a wide-step algorithm.

Finally, we would like to point out a particular convenience of the above axioms: the Axioms W1 – W4 are completely free of any inductive or syntactic constructions such as terms. As we show in the following section, this simplifies the examination of elementary wide-step algorithms to a great extent.

5 The Equivalence Theorem and its Applications

As announced previously, in this section we present a theorem stating that elementary wide-step ASMs as introduced in Sect. 3 and elementary wide-step algorithms as introduced in Sect. 4 are equivalent in a strong sense. We call two wide-step algorithms *equivalent* if they have the same states, the same initial states, and provide the same update sets at all states. The theorem then reads:

Theorem 1. *Each elementary wide-step ASM is an elementary wide-step algorithm, and for each elementary wide-step algorithm there exists an equivalent elementary wide-step ASM.*

Before presenting the proof in Section 6, we exemplify the usability of Theorem 1 for reasoning about elementary wide-step ASMs. In particular, we show that wide-step ASMs

- are monotonous, i.e. for a substructure S of a state R , the update set of S is a subset of the update set of R ,

- are not linear-speedup, i.e. cannot accelerate computation by arbitrary linear factors,
- cannot avoid conflicts in general, i.e. not every wide-step ASM \mathcal{M} can be transformed into an ASM \mathcal{M}' that has the same runs as \mathcal{M} and provides only consistent update sets at all states.

We briefly review and discuss elementary wide-step ASMs with respect to these properties at the end of this section.

5.1 Elementary Wide-Step ASMs are Monotonous

We apply Theorem 1 to show that update sets computed by a wide-step ASM \mathcal{M} are *monotonous* in the following sense: If a state S is a substructure of a state R , then the update set $\Delta_{\mathcal{M}}(S)$ is a subset of $\Delta_{\mathcal{M}}(R)$.

As usual, a Σ -structure S is a *substructure* of a Σ -structure R ($S \subseteq R$ for short) iff $S =_{U_S} R$ (or, more classically, iff for all function symbols \mathbf{f} , \mathbf{f}_S is the restriction of \mathbf{f}_R to U_S). Hence, we show that for all states S, R of a wide-step ASM \mathcal{M} , $S \subseteq R$ implies $\Delta_{\mathcal{M}}(S) \subseteq \Delta_{\mathcal{M}}(R)$.

By help of Theorem 1, the proof is simple: Let $\delta \in \Delta_{\mathcal{M}}(S)$. According to Axiom W4, there is a witness $W \subseteq U_S$ for δ at S . As $S \subseteq R$, it follows by definition $S =_{U_S} R$, which in turn implies $S =_W R$. Hence, as W is a witness for δ , $\delta \in \Delta_{\mathcal{M}}(R)$.

5.2 Elementary Wide-Step ASMs are Not Linear-Speedup

As usual, a machine model is *linear-speedup* iff for each machine \mathcal{M} there exists a machine \mathcal{M}^2 that performs two steps of \mathcal{M} in a single step. Intuitively, \mathcal{M}^2 computes “twice as fast as \mathcal{M} ”. Linear-speedup confirms that constant linear factors are irrelevant when doing complexity analysis in the machine model. This property has been shown for various machine models such as Turing machines, cellular automata, and in particular for sequential and interactive ASMs [13, 2].

As it turns out, elementary wide-step ASMs are not linear-speedup. As a counterexample, we consider the problem of determining whether a particular function of a state is undefined at some argument. We show that this problem can be solved by a wide-step ASM in two steps, and apply Theorem 1 to show that this problem cannot be solved in a single step.

At first, we state the problem more clearly. Consider the signature Σ comprising the usual symbols **true**, **false**, and **undef**, together with a nullary symbol **r** and a unary symbol **f**. Let \mathcal{J} be the set of all Σ -structures over infinite universes that satisfy **r=undef** and interpret **true**, **false**, and **undef** in the usual way. We look for an ASM that determines for each state $S \in \mathcal{J}$ whether the function \mathbf{f}_S is *partial*. As usual in the ASM literature, we consider a function \mathbf{f}_S of a state S as partial if $\mathbf{f}_S(\bar{u}) = \mathbf{undef}_S$ for some argument \bar{u} . Hence, we look for an ASM that, for each initial state $S \in \mathcal{J}$, reaches a fixpoint state R with **r=true** if S satisfies $\exists x : \mathbf{f}(x) = \mathbf{undef}$, and **r=false** otherwise.

A solution is the following ASM rule CHECKPARTIAL, which solves the problem in two steps:

```

if r=undef then r:=false
if r=false then forall x do if (f(x)=undef) then r:=true.

```

By Theorem 1, we can show that no wide-step ASM rule solves the problem in a single step: In order to validate in a single step that \mathbf{r} gets the value **false** (which is the case if S satisfies $\forall x : \mathbf{f}(x) \neq \mathbf{undef}$), the whole universe of S needs to be explored. (Note that this is not the case for $\exists x : \mathbf{f}(x) = \mathbf{undef}$, which can be validated by a single element x of the universe.) Hence, for the update $\delta = (\mathbf{a}, [], \mathbf{false}_S)$ there cannot exist a finite witness W at S . As this fact violates Axiom W4, no wide-step ASM solves the problem in a single step.

5.3 Elementary Wide-Step ASMs Cannot Avoid Conflicts

We call an ASM rule *RULE non-conflicting* if it provides consistent update sets at all states. Otherwise we call *RULE conflicting*. It is well-known that each conflicting sequential ASM rule can be transformed into an equivalent non-conflicting sequential ASM rule [13]. Here we show that this convenient property, unfortunately, does not hold for elementary wide-step ASM rules.

A simple counterexample is the following ASM rule *CONF* over the signature Σ containing only the constant symbol \mathbf{a} :

```

forall x do if  $\neg(x=\mathbf{a})$  then  $\mathbf{a}:=x$ .

```

For a state S of *CONF*, $\llbracket \text{CONF} \rrbracket_S$ is inconsistent iff $|U_S| \geq 3$. In order to resolve these conflicts, we look for an ASM rule *NONCONF* such that

- (i) for all states S , $\llbracket \text{NONCONF} \rrbracket_S$ is consistent,
- (ii) for all states S with consistent $\llbracket \text{CONF} \rrbracket_S$ holds $\llbracket \text{NONCONF} \rrbracket_S = \llbracket \text{CONF} \rrbracket_S$.

We show now that these requirements contradict each other.

Assume a wide-step ASM rule *NONCONF* that satisfies (ii). At the states S, S' with $U_S = \{0, 1\}$, $U_{S'} = \{0, 2\}$ and $\mathbf{a}_S = \mathbf{a}'_S = 0$, *CONF* yields the consistent update sets $\{(a, [], 1)\}$ and $\{(a, [], 2)\}$, respectively. According to (ii), *NONCONF* yields the same update sets at S and S' . Now consider the state R with $U_R = \{0, 1, 2\}$ and $\mathbf{a}_R = 0$. As $S \subseteq R$ and $S' \subseteq R$, the monotonicity of *NONCONF* implies $\{(a, [], 1), (a, [], 2)\} \subseteq \llbracket \text{NONCONF} \rrbracket_R$, which in turn contradicts (i).

The unavoidability of conflicting updates is the third reasons why we consider update sets instead of next-states in Axiom W2 (see discussion after the axiom). Conflicting updates are inherent to elementary wide-step ASMs, and such updates are only observable by considering update sets instead of next-states.

5.4 Discussion

We have shown that elementary wide-step ASMs are monotonous, are not linear-speedup, and cannot prevent conflicting updates in general. Here we want to briefly discuss those properties and their relevance for elementary wide-step ASMs.

As exemplified by the proof in the previous subsection, monotonicity can be quite helpful for reasoning about wide-step ASMs in general. In addition, monotonicity can also be used for the examination of a particular wide-step ASM model, e.g. for proving that certain properties are preserved under extension of states.

While monotonicity is a desirable property, the lack of linear-speedup is problematic. It confirms that elementary wide-step ASMs are not closed under sequential composition. Similarly, the unavoidability of conflicting updates is unfortunate: While the problem of conflicting updates exists since the first appearance of ASMs, it was always known that this problem can be ignored (at least in theory) by constructing equivalent conflict-free ASMs. But this does not hold for elementary wide-step ASMs. Consequently, conflicting updates need special attention here.

These problems show that a straight extension of sequential ASMs by the `forall`-operator yields a somewhat limited computational model. However, we do not claim that elementary wide-step ASMs are particularly well-designed. Instead, we consider this variant as a starting point for the investigation of more evolved variants of wide-step ASMs. In particular, we hope to lift the above limitations by considering more expressive variants of wide-step ASMs, which, for instance, may employ first-order formulas instead of Boolean formulas. This remains a task for future work.

6 Proof of the Equivalence Theorem

In this section we prove Theorem 1 as presented in the previous section. The proof divides into two parts: We first show that every wide-step algorithm is equivalent to a wide-step ASM. We then prove that each wide-step ASM constitutes a wide-step algorithm.

6.1 Each Wide-Step Algorithm is Equivalent to a Wide-Step ASM

We assume an arbitrary wide-step algorithm \mathcal{A} . In this section we construct a wide-step ASM \mathcal{M} that is equivalent to \mathcal{A} .

We start by a lemma stating that each witness W of an update δ contains all elements of δ . To this end, we define the *elements of an update* $\delta = (\mathfrak{f}, [u_1, \dots, u_n], u_0)$ as $E(\delta) =_{\text{def}} \{u_0, \dots, u_n\}$.

Lemma 1. *Let S be a state of \mathcal{A} , let $\delta \in \Delta_{\mathcal{A}}(S)$, and let W be a witness for δ at S . Then $E(\delta) \subseteq W$.*

Proof. Proof by contradiction: Assume an element $u \in E(\delta)$ with $u \notin W$. Construct R from S by replacing the element u by a new element not contained in U_S . By construction, R and S are isomorphic and $R =_W S$. Hence, by Axiom W3, R is a state of \mathcal{A} . As W is a witness for δ at S , $R =_W S$ implies $\delta \in \Delta_{\mathcal{A}}(R)$. But this violates Axiom W2, as $u \in E(\delta)$ and $u \notin U_R$, i.e. δ is not an update for R . \square

The following lemma contains the main idea of the proof: For each update δ of the algorithm \mathcal{A} , we can construct an ASM rule **RULE** such that

- **RULE** yields the update δ ,
- all other updates of **RULE** are also updates of \mathcal{A} .

This idea will be stated more precisely in the lemma. For the rest of this section, let $k_{\mathcal{A}}$ be the constant as given in Axiom W4, and let V be a set of variables with $|V| = k_{\mathcal{A}}$.

Lemma 2. *Let S be a Σ -structure and let $\delta \in \Delta_{\mathcal{A}}(S)$. Then there exists an ASM rule **RULE** over Σ and V such that*

- (i) $\delta \in \llbracket \mathbf{RULE} \rrbracket_S^\alpha$ for one variable assignment α ,
- (ii) $\llbracket \mathbf{RULE} \rrbracket_R^\beta \subseteq \Delta_{\mathcal{A}}(R)$ for all states R and all variable assignments β .

Proof. Let $\delta = (\mathbf{f}, [u_1, \dots, u_n], u_0)$ and let W be a witness for δ at S (see Axiom W4). As $|W| \leq k$, choose a subset $V_W \subseteq V$ of variables such that $|V_W| = |W|$, and let $\alpha : V_W \rightarrow W$ be a bijective variable assignment. For each element $u \in W$, let $v^u \in V_W$ denote the variable with $\alpha(v^u) = u$. Hence, each element from W is represented by one and only one variable from V_W .

The rest of the proof divides into three parts. We first give the construction of the desired ASM rule **RULE**. We then show that **RULE** indeed satisfies the properties (i) and (ii).

Construction of **RULE.** For a m -ary function symbol \mathbf{g} from $\Sigma_{\mathcal{A}}$ and for variables $v_0, \dots, v_m \in V_W$, we call the formula $\mathbf{g}(v_1, \dots, v_m) = v_0$ an *atomic formula*. For an atomic formula ψ , we call ψ and $\neg\psi$ *literal formulas*. Note that, as V_W and Σ are finite, there are only finitely many literal formulas. We then define $\phi =_{\text{def}} \phi_1 \wedge \phi_2$ with

$$\begin{aligned} \phi_1 &=_{\text{def}} \bigwedge_{v_1 \neq v_2 \in V_W} \neg(v_1 = v_2), \\ \phi_2 &=_{\text{def}} \bigwedge \{ \psi \mid \psi \text{ is a literal formula with } S, \alpha \models \psi \} \end{aligned}$$

Note that both ϕ_1 and ϕ_2 are satisfied at S and α .

By Lemma 1, for the above update $\delta = (\mathbf{f}, [u_1, \dots, u_n], u_0)$ holds $E(\delta) = \{u_0, \dots, u_n\} \subseteq W$. We can therefore derive from δ the ASM rule **ASSIGN** as

$$\mathbf{f}(v^{u_1}, \dots, v^{u_n}) := v^{u_0}.$$

Note that, as $\alpha(v^u) = u$, $\llbracket \mathbf{ASSIGN} \rrbracket_S^\alpha = \{ \delta \}$. Finally, construct the ASM rule **RULE** as

$$\mathbf{if} \ \phi_1 \wedge \phi_2 \ \mathbf{then} \ \mathbf{ASSIGN}.$$

As $|W|$ is bounded by the constant $k_{\mathcal{A}}$ which is uniform for the algorithm \mathcal{A} , the length of **RULE** is bounded by a constant $c_{\mathcal{A}}$ uniform for \mathcal{A} .

Proof of (i). As stated above, ϕ_1 and ϕ_2 are satisfied at S and α , and $\llbracket \mathbf{ASSIGN} \rrbracket_S^\alpha = \{ \delta \}$. Therefore, by the semantics of ASM rules,

$$\llbracket \mathbf{RULE} \rrbracket_S^\alpha = \llbracket \mathbf{ASSIGN} \rrbracket_S^\alpha = \{ \delta \}.$$

Hence, $\delta \in \llbracket \text{RULE} \rrbracket_S^\alpha$.

Proof of (ii). Now, let R be an arbitrary state of \mathcal{A} and let $\beta : V_W \rightarrow U_R$ be an arbitrary variable assignment. We have to show that $\llbracket \text{RULE} \rrbracket_R^\beta \subseteq \Delta_{\mathcal{A}}(R)$. This is trivially true if $\phi_1 \wedge \phi_2$ is not satisfied at R and β , as $\llbracket \text{RULE} \rrbracket_R^\beta = \emptyset$ in that case.

Hence, we now consider the case that $\phi_1 \wedge \phi_2$ is satisfied at R and β . As ϕ_1 holds, β is bijective. Construct from R a state Q by bijectively replacing for each $v \in V_W$ the element $\beta(v)$ by $\alpha(v)$, and each other element from U_R by a new element. Let $i : R \rightarrow Q$ be the corresponding isomorphism. As R and Q are isomorphic, Axiom W3 asserts that Q is a state of \mathcal{A} . Note that, by construction of the isomorphism i , it holds $i \circ \beta = \alpha$. Further, as ϕ_2 is satisfied at R and β , the isomorphism $i : R \rightarrow Q$ asserts that ϕ_2 is satisfied at Q and $i \circ \beta = \alpha$.

We now show that $S =_W Q$. By construction of Q holds $W \subseteq U_Q$. For each m -ary function symbol \mathbf{g} and each $p_0, \dots, p_m \in W$ we have to show that $\mathbf{g}_S(p_1, \dots, p_m) = p_0$ iff $\mathbf{g}_Q(p_1, \dots, p_m) = p_0$. *Firstly*, we consider the case that $\mathbf{g}_S(p_1, \dots, p_m) = p_0$. Then ϕ_2 contains the literal $\mathbf{g}(v^{p_1}, \dots, v^{p_m}) = v^{p_0}$. Hence, this literal is satisfied at Q and α . As $\alpha(v^{p_i}) = p_i$, this means that $\mathbf{g}_Q(p_1, \dots, p_m) = p_0$. *Secondly*, we consider the case that $\mathbf{g}_S(p_1, \dots, p_m) \neq p_0$. Then $\mathbf{g}_Q(p_1, \dots, p_m) \neq p_0$ can be shown analogously to the first case by help of the literal $\neg(\mathbf{g}(v^{p_1}, \dots, v^{p_m}) = v^{p_0})$.

Summing up, we have states S , Q , and R , an isomorphism $i : R \rightarrow Q$ with $i \circ \beta = \alpha$ (thus, $\beta = i^{-1} \circ \alpha$), and $S =_W Q$. Then

$$\begin{aligned} \llbracket \text{RULE} \rrbracket_R^\beta &= \llbracket \text{ASSIGN} \rrbracket_R^\beta \\ &= \{ (\mathbf{f}, [\beta(v^{u_1}), \dots, \beta(v^{u_n})], \beta(v^{u_0})) \} \\ &= \{ (\mathbf{f}, [i^{-1} \circ \alpha(v^{u_1}), \dots, i^{-1} \circ \alpha(v^{u_n})], i^{-1} \circ \alpha(v^{u_0})) \} \\ &= \{ (\mathbf{f}, [i^{-1}(u_1), \dots, i^{-1}(u_n)], i^{-1}(u_0)) \} \\ &= \{ i^{-1}(\delta) \}. \end{aligned}$$

Hence, in order to obtain $\llbracket \text{RULE} \rrbracket_R^\beta \subseteq \Delta_{\mathcal{A}}(R)$, we have to show that $i^{-1}(\delta) \in \Delta_{\mathcal{A}}(R)$. But this immediately follows from the Axioms W3 and W4: As W is a witness for δ at S , $S =_W Q$ implies $\delta \in \Delta_{\mathcal{A}}(Q)$. As $i^{-1} : Q \rightarrow R$ is an isomorphism, W3 implies $i^{-1}(\delta) \in \Delta_{\mathcal{A}}(R)$. \square

Finally, by the help of Lemma 2, we prove Theorem 1:

Proof (of Theorem 1). For each state $S \in \mathcal{S}_{\mathcal{A}}$, and each $\delta \in \Delta_{\mathcal{A}}(S)$, let $\text{RULE}_{S,\delta}$ be the ASM rule as constructed in Lemma 2. Let \mathcal{R} be the set of all of these ASM rules. By construction (see proof of Lemma 2), the size of all rules in \mathcal{R} is bounded by a constant $c_{\mathcal{A}}$. As the rules in \mathcal{R} are built over finitely many different symbols (i.e. over a finite alphabet), \mathcal{R} is finite.

Let v_1, \dots, v_n be the variables in V and let R_1, \dots, R_m be the rules in \mathcal{R} . Construct the closed ASM rule RULE as

$$\begin{aligned} &\text{forall } v_1 \text{ do...forall } v_n \text{ do} \\ &\quad \text{par } R_1 \dots R_m \text{ endpar } . \end{aligned}$$

Let S be an arbitrary state of \mathcal{A} . By the semantics of **RULE**, it immediately follows that

$$\llbracket \text{RULE} \rrbracket_S = \bigcup_{\alpha: V \rightarrow U_S} \llbracket R_1 \rrbracket_S^\alpha \cup \dots \cup \llbracket R_m \rrbracket_S^\alpha. \quad (1)$$

We now show that $\llbracket \text{RULE} \rrbracket_S = \Delta_{\mathcal{A}}(S)$ by mutual inclusion. According to Lemma 2, $\llbracket R_i \rrbracket_S^\alpha \subseteq \Delta_{\mathcal{A}}(S)$ for all $i = 1, \dots, m$ and all variable assignments α . By (1) then follows $\llbracket \text{RULE} \rrbracket_S \subseteq \Delta_{\mathcal{A}}(S)$. It remains to show $\Delta_{\mathcal{A}}(S) \subseteq \llbracket \text{RULE} \rrbracket_S$. Let $\delta \in \Delta_{\mathcal{A}}(S)$. By Lemma 2, for the ASM rule $\text{RULE}_{S,\delta}$ there is a variable assignment α with $\delta \in \llbracket \text{RULE}_{S,\delta} \rrbracket_S^\alpha$. By definition of \mathcal{R} , $\text{RULE}_{S,\delta} \in \mathcal{R}$, i.e. there is an index $i \in \{1, \dots, m\}$ with $\text{RULE}_{S,\delta} = R_i$. Therefore $\delta \in \llbracket R_i \rrbracket_S^\alpha$. By (1) follows $\delta \in \llbracket \text{RULE} \rrbracket_S$.

The wide-step ASM \mathcal{M} with $\mathcal{S}_{\mathcal{M}} = \mathcal{S}_{\mathcal{A}}$, $\mathcal{J}_{\mathcal{M}} = \mathcal{J}_{\mathcal{A}}$, $\text{RULE}_{\mathcal{M}} = \text{RULE}$ then is equivalent to \mathcal{A} . \square

6.2 Each Wide-Step ASM is a Wide-Step Algorithm

Assume an arbitrary wide-step ASM \mathcal{M} with its ASM rule **RULE**. Here we show that \mathcal{M} indeed satisfies the Axioms W1 – W4. Axioms W1 and W2 are satisfied by the definition of wide-step ASMs. Axiom W3 is proven by induction on the syntax of **RULE**.

It remains to verify Axiom W4. That is, we are obliged to specify for each state S of \mathcal{M} and for each $\delta \in \llbracket \text{RULE} \rrbracket_S$ a witness W at S . We solve this task by a technical, yet helpful extension of the semantics of ASM rules: Instead of deriving at state S a set $\llbracket \text{RULE} \rrbracket_S$ of updates δ , we derive a set $\langle\langle \text{RULE} \rangle\rangle_S$ of pairs (W, δ) , where W is a witness for δ . We call such a set a *witness-extended update set*.

For a state S , we determine the witness-extended update set by tracking the elements that are explored by **RULE** at S . In particular, for a term t , a state S and a variable assignment α , the *set of explored elements* $\langle t \rangle_S^\alpha$ is the set of values of all subterms of t (including t itself) at S and α . Similarly, for a boolean formula ϕ , the *set of explored elements* $\langle \phi \rangle_S^\alpha$ is the union of all $\langle t \rangle_S^\alpha$ where t is a term occurring in ϕ .

In order to derive witness-extended update sets from **RULE**, we introduce the following auxiliary notations. An update set Δ can be extended by a set W (which will be a witness) as follows:

$$W \oplus \Delta \stackrel{\text{def}}{=} \{ (W, \delta) \mid \delta \in \Delta \}.$$

Similarly, for a witness-extended update set Δ^+ , the extension of Δ^+ by another set W is defined as

$$W \oplus \Delta^+ \stackrel{\text{def}}{=} \{ (W \cup W', \delta) \mid (W', \delta) \in \Delta^+ \}.$$

Using the above notations, we define the *witness-extended semantics* of elementary wide-step ASM rules:

$$\begin{aligned}
\langle\langle \mathbf{f}(t_1, \dots, t_n) := t_0 \rangle\rangle_S^\alpha &=_{\text{def}} (\langle t_0 \rangle_S^\alpha \cup \dots \cup \langle t_n \rangle_S^\alpha) \oplus \llbracket \mathbf{f}(t_1, \dots, t_n) := t_0 \rrbracket_S^\alpha \\
\langle\langle \mathbf{if} \ \phi \ \mathbf{then} \ \mathbf{R} \rangle\rangle_S^\alpha &=_{\text{def}} \begin{cases} \langle \phi \rangle_S^\alpha \oplus \langle\langle \mathbf{R} \rangle\rangle_S^\alpha & , \text{ in case } (S, \alpha) \models \phi \\ \emptyset & , \text{ otherwise} \end{cases} \\
\langle\langle \mathbf{par} \ \mathbf{R}_1 \dots \mathbf{R}_n \ \mathbf{endpar} \rangle\rangle_S^\alpha &=_{\text{def}} \bigcup_{1 \leq i \leq n} \langle\langle \mathbf{R}_i \rangle\rangle_S^\alpha \\
\langle\langle \mathbf{forall} \ x \ \mathbf{with} \ \phi \ \mathbf{do} \ \mathbf{R} \rangle\rangle_S^\alpha &=_{\text{def}} \bigcup_{u \in U_S} \langle \phi \rangle_S^{\alpha[x \mapsto u]} \oplus \langle\langle \mathbf{R} \rangle\rangle_S^{\alpha[x \mapsto u]}
\end{aligned}$$

As the rule **RULE** of the above wide step ASM \mathcal{M} is closed, we may write $\langle\langle \mathbf{RULE} \rangle\rangle_S$ instead of $\langle\langle \mathbf{RULE} \rangle\rangle_S^\alpha$. By induction on the syntax of **RULE**, the following three propositions can be shown:

- (i) For all states S holds $\llbracket \mathbf{RULE} \rrbracket_S = \{ \delta \mid (W, \delta) \in \langle\langle \mathbf{RULE} \rangle\rangle_S \}$, i.e. $\langle\langle \mathbf{RULE} \rangle\rangle_S$ indeed is an extension of the updates in $\llbracket \mathbf{RULE} \rrbracket_S$,
- (ii) there is a constant $k \in \mathbb{N}$ such that $|W| \leq k$ for all states S and all $(W, \delta) \in \langle\langle \mathbf{RULE} \rangle\rangle_S$,
- (iii) For all states S, R with $(W, \delta) \in \langle\langle \mathbf{RULE} \rangle\rangle_S$ and $S =_W R$ holds $(W, \delta) \in \langle\langle \mathbf{RULE} \rangle\rangle_R$.

By use of the above propositions, we show that the wide-step ASM \mathcal{M} indeed satisfies Axiom W4. Let S be a state of \mathcal{M} and let $\delta \in \Delta_{\mathcal{M}}(S)$, i.e. $\delta \in \llbracket \mathbf{RULE} \rrbracket_S$. By (i), there is a pair $(W, \delta) \in \langle\langle \mathbf{RULE} \rangle\rangle_S$. We show now that W is a witness for δ at S . By (ii), $|W|$ is bounded by the constant k . Further, assume a state R with $R =_W S$. Then by (iii) holds $(W, \delta) \in \langle\langle \mathbf{RULE} \rangle\rangle_R$, and by (i) we conclude $\delta \in \llbracket \mathbf{RULE} \rrbracket_R$.

7 Conclusion

The comprehensive experience on ASMs gained during the last two decades shows that the syntax of ASMs allows for a natural and intuitive description of algorithms and systems [11, 8, 7]. By characterizing ASMs *independently* of such syntax, we obtain a deeper understanding on the foundations and the expressive power of ASMs. Following this idea, we proposed a syntax-independent, i.e. *semantic* characterization of elementary wide-step ASMs, which extend classical sequential small-step ASMs by the **forall**-operator. A particular nice feature of this characterization is the complete absence of syntactic or inductive constructions such as terms.

We further demonstrated that such characterizations significantly simplify theoretical examination of ASMs. In particular, we obtained three immediate consequences of the characterization presented in this paper: Elementary wide-step ASMs are monotonous, lack linear-speedup, and cannot prevent conflicting updates in general. While monotonicity is advantageous, the latter two properties

point out that elementary wide-step ASMs have their limitations (see discussion in Sect. 5.4).

We intend to study more evolved variants of wide-step ASMs, in particular variants that may employ full first-order formulas instead of Boolean formulas. We hope that these variants are not subject to the aforementioned limitations. The present paper provides a solid starting point for such investigations.

References

1. Andreas Blass and Yuri Gurevich. Abstract State Machines Capture Parallel Algorithms. *ACM Trans. Comput. Logic*, 4(4):578–651, 2003.
2. Andreas Blass and Yuri Gurevich. Ordinary Interactive Small-Step Algorithms, parts I, II, III. *ACM Trans. Comput. Logic*, 2006.
3. Andreas Blass, Yuri Gurevich, Dean Rosenzweig, and Benjamin Rossman. General Interactive Small Step Algorithms. Technical Report MSR-TR-2005-113, Microsoft Research, Aug 2006.
4. Udi Boker and Nachum Dershowitz. A Formalization of the Church-Turing Thesis. preprint.
5. Udi Boker and Nachum Dershowitz. Abstract Effective Models. *Electr. Notes Theor. Comput. Sci.*, 135(3):15–23, 2006.
6. Egon Börger. The Origins and the Development of the ASM Method for High Level System Design and Analysis. *Journal of Universal Computer Science*, 8(1):2–74, 2002.
7. Egon Börger. Abstract State Machines: A Unifying View of Models of Computation and of System Design Frameworks. *Annals of Pure and Applied Logic*, 133:149–171, 2005.
8. Egon Börger and Robert Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.
9. Andreas Glausch and Wolfgang Reisig. Distributed Abstract State Machines and Their Expressive Power. Informatik-Berichte 196, Humboldt-Universität zu Berlin, January 2006.
10. Andreas Glausch and Wolfgang Reisig. A Semantic Characterization of Unbounded-Nondeterministic Abstract State Machines. In *CALCO 2007, Proceedings*, August 2007. to appear.
11. Yuri Gurevich. Evolving Algebras 1993: Lipari Guide. In Egon Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.
12. Yuri Gurevich. May 1997 Draft of the ASM Guide. Technical Report CSE-TR-336-97, University of Michigan EECS Department, May 1997.
13. Yuri Gurevich. Sequential Abstract State Machines Capture Sequential Algorithms. *ACM Transactions on Computational Logic*, 1(1):77–111, Jul 2000.
14. Yuri Gurevich and Tatiana Yavorskaya. On Bounded Exploration and Bounded Nondeterminism. Technical Report MSR-TR-2006-07, Microsoft Research, Jan 2006.
15. Dean Rosenzweig and Davor Runje. Some Things Algorithms Cannot Do. Technical Report MSR-TR-2005-52, Microsoft Research, 2005.
16. Dean Rosenzweig, Davor Runje, and Wolfram Schulte. Model-Based Testing of Cryptographic Protocols. In *Trustworthy Global Computing, International Symposium, TGC 2005, Revised Selected Papers*, pages 33–60, 2005.