A Logic for Non-Deterministic Parallel Abstract State Machines*

Flavio Ferrarotti[1], Klaus-Dieter Schewe[1], Loredana Tec[1], and Qing Wang[2]

[1] Software Competence Center Hagenberg, A-4232 Hagenberg, Austria
{flavio.ferrarotti,klaus-dieter.schewe,loredana.tec}@scch.at
[2] Research School of Computer Science, The Australian National University
qing.wang@anu.edu.au

Abstract. We develop a logic which enables reasoning about single steps of non-deterministic parallel Abstract State Machines (ASMs). Our logic builds upon the unifying logic introduced by Nanchen and Stärk for reasoning about hierarchical (parallel) ASMs. Our main contribution to this regard is the handling of non-determinism (both bounded and unbounded) within the logical formalism. Moreover, we do this without sacrificing the completeness of the logic for statements about single steps of non-deterministic parallel ASMs, such as invariants of rules, consistency conditions for rules, or step-by-step equivalence of rules.

1 Introduction

Gurevich's Abstract State Machines (ASMs) provide not only a formal theory of algorithms, but also are the basis for a general software engineering method based in the specification of higher-level ground models and step-by-step refinement. Chapter 9 in the book [5] gives a summary of many application projects that have developed complex systems solutions on the grounds of ASMs. A major advantage of the ASM method and a key for its success resides in the fact that it provides, not only a simple and precise framework to communicate and document design ideas, but also an accurate and checkable overall understanding of complex systems. In this context, formal verification of dynamic properties for given ASMs is a fundamentally important task, in particular in the case of modelling safety critical systems, where there is a need to ensure the integrity and reliability of the system. Clearly, a logical calculi appropriate for the formalisation and reasoning about dynamic properties of ASMs is an essential and valuable tool for this endeavour.

Numerous logics have been developed to deal with specific features of ASM verification such as correctness and deadlock-freeness (see Section 9.4.3 in the book [5]) for detailed references), but a complete logic for ASMs was only developed in [13] by Nanchen and Stärk. The logic formalizes properties of a single

* Work supported by the **Austrian Science Fund (FWF: [P26452-N15])**. Project: *Behavioural Theory and Logics for Distributed Adaptive Systems*. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-30024-5_18

step of an ASM, which permits to define Hilbert-style proof theory and to show its completeness. In this work the treatment of non-determinism was deliberately left out. Same as parallelism, which is on the other hand captured by the logic for ASMs of Nanchen and Stärk, non-determinism is also a prevalent concept in the design and implementation of software systems, and consequently a constitutive part of the ASM method for systems development [5]. Indeed, nondeterminism arises in the specification of many well known algorithms and software applications. Examples range from graph algorithms, such as minimum spanning tree and shortest path, to search techniques whose objective is to arrive at some admissible goal state (as in the n-queens and combinatorial-assignment problems [7]), and learning strategies such as converging on some classifier that labels all data instances correctly [14]. Non-deterministic behavior is also common in cutting edge fields of software systems. Distributed systems frequently need to address non-deterministic behaviour such as changing role (if possible) as strategic response to observed problems concerning load, input, throughput, etc. Also, many cyber-physical systems and hybrid systems such as railway transportation control systems [2] and systems used in high-confidence medical healthcare devices exhibit highly non-deterministic behaviour.

Notice that although we could say that there is a kind of latent parallelism in non-determinism, they represent completely different behaviours and thus both are needed to faithfully model the behaviour of complex systems, more so in the case of the ASM method where the ability to model systems at every level of abstraction is one of its main defining features. For instance, while a nondeterministic action can evaluate to multiple behaviors, only if at least one of these behaviors does not conflict with concurrent tasks, then there is an admissible execution of the action in parallel with these tasks.

The ASM method allows for two different, but complementary, approaches to non-determinism. The first approach assumes that choices are made by the environment via monitored functions that can be viewed as external oracles. In this case, non-deterministic ASMs are just interactive ASMs. The second approach assumes the ASMs themselves rather than the environment, to have the power of making non-deterministic choices. In this case the one-step transition function of the ASMs is no longer a function but a binary relation. This is also the approach followed by non-deterministic Turing machines. However, in the case of non-deterministic Turing machines the choice is always bounded by the transition relation. For ASMs the non-determinism can also be unbounded, i.e., we can choose among an infinite number of possibilities. Clearly, unbounded nondeterminism should also be allowed if we want our ASMs to be able to faithfully model algorithms at any level of abstraction.

In this work we develop a logic which enables reasoning about single steps of non-deterministic parallel ASMs, i.e., ASMs which include the well known **choose** and **forall** rules [5]. This builds upon the complete logic introduced in the work of Nanchen and Stärk [13] for reasoning about single steps of hierarchical ASMs. Hierarchical ASMs capture the class of synchronous and deterministic parallel algorithms in the precise sense of the ASM thesis of Blass and Gure-

vich [3,4] (see also [6]). Our main contribution to this regard is the handling of non-determinism (both bounded and unbounded) within the logical formalism. More importantly, this is done without sacrificing the completeness of the logic. As highlighted by Nanchen and Stärk [13], non-deterministic transitions manifest themselves as a difficult task in the logical formalisation for ASMs.

The paper is organized as follows. The next section introduces the required background from ASMs. Section 3 formalises the model of non-deterministic parallel ASM used through this work. In Section 4 we introduce the syntax and semantics of the proposed logic for non-deterministic parallel ASMs. Section 5 presents a detailed discussion regarding consistency and update sets, and the formalisation of a proof system. In Section 6 we use the proof system to derive some interesting properties of our logic, including known properties of the ASM logic in [13]. In Section 7 we present our main result, namely that the proposed logic is complete for statements about single steps of non-deterministic parallel ASMs, such as invariants of rules, consistency conditions for rules, or step-by-step equivalence of rules. We conclude our work in Section 8.

2 Preliminaries

The concept of Abstract State Machines (ASMs) is well known [5]. In its simplest form an ASM is a finite set of so-called *transition rules* of the form **if** *Condition* **then** *Updates* **endif** which transforms abstract states. The condition or guard under which a rule is applied is an arbitrary first-order logic sentence. *Updates* is a finite set of assignments of the form $f(t_1, \ldots, t_n) := t_0$ which are executed in parallel. The execution of $f(t_1, \ldots, t_n) := t_0$ in a given state proceeds as follows: first all parameters $t_0, t_1, \ldots t_n$ are evaluated to their values, say $a_0, a_1, \ldots, a_n$, then the value of $f(a_1, \ldots, a_n)$ is updated to a_0, which represents the value of $f(a_1, \ldots, a_n)$ in the next state. Such pairs of a function name f, which is fixed by the signature, and optional argument $(a_1, \ldots, a_n)$ of dynamic parameters values a_i, are called *locations*. They represent the abstract ASM concept of memory units which abstracts from particular memory addressing. Location value pairs (ℓ, a), where ℓ is a location and a a value, are called *updates* and represent the basic units of state change.

The notion of ASM *state* is the classical notion of *first-order structure* in mathematical logic. For the evaluation of first-order terms and formulae in an ASM state, the standard interpretation of function symbols by the corresponding functions in that state is used. As usually in this setting and w.l.o.g., we treat predicates as characteristic functions and constants as 0-ary functions.

The notion of the ASM *run* is an instance of the classical notion of the computation of transition systems. An ASM computation step in a given state consists in executing *simultaneously* all updates of all transition rules whose guard is true in the state, if these updates are consistent, in which case the result of their execution yields a next state. In the case of inconsistency, the computation does not yield a next state. A set of updates is *consistent* if it contains no pairs (ℓ, a), (ℓ, b) of updates to a same location ℓ with $a \neq b$.

Simultaneous execution, as obtained in one step through the execution of a set of updates, provides a useful instrument for high-level design to locally describe a global state change. This synchronous parallelism is further enhanced by the transition rule **forall** x **with** φ **do** r **enddo** which expresses the simultaneous execution of a rule r for each x satisfying a given condition φ.

Similarly, non-determinism as a convenient way of abstracting from details of scheduling of rule executions can be expressed by the rule **choose** x **with** φ **do** r **enddo**, which means that r should be executed with an arbitrary x chosen among those satisfying the property φ.

The following example borrowed from [5] clearly illustrates the power of the **choose** and **forall** rules.

Example 1. The following ASM generates all and only the pairs $vw \in A^*$ of different words v, w of same length (i.e., $v \neq w$ and $|v| = |w|$).

```
choose n, i with i < n do
   choose a, b with a ∈ A ∧ b ∈ A ∧ a ≠ b do
      v(i) := a
      w(i) := b
      forall j with j < n ∧ j ≠ i do
         choose a, b with a ∈ A ∧ b ∈ A do
            v(j) := a
            w(j) := b
         enddo
      enddo
   enddo
enddo
```

When all possible choices are realized, the set of reachable states of this ASM is the set of all "vw" states with $v \neq w$ and $|v| = |w|$.

3 Non-Deterministic Parallel ASMs

It is key for the completeness of our logic to make sure that the ASMs do not produce infinite update sets. For that we formally define ASM states as simple metafinite structures [8] instead of classical first-order structures, and restrict the variables in the **forall** rules to range over the finite part of such metafinite states. Nevertheless, the class of algorithms that are captured by these ASM machines coincides with the class of parallel algorithms that satisfy the postulates of the parallel ASM thesis of Blass and Gurevich [3,4] (see [6] for details).

A *metafinite structure* S consists of: a finite first-order structure S_1 –the *primary part* of S; a possibly infinite first-order structure S_2 –the *secondary part* of S; and a finite set of functions which map elements of S_1 to elements of S_2 –the *bridge functions*. A signature Υ of metafinite structures comprises a sub-signature Υ_1 for the primary part, a sub-signature Υ_2 for the secondary part and a finite set $\mathcal{F}_b$ of bridge function names. The *base set* of a state S is a nonempty set of values $B = B_1 \cup B_2$, where B_1 is the finite domain of S_1, and

B_2 is the possibly infinite domain of S_2. Function symbols f in Υ_1 and Υ_2 are interpreted as functions f^S over B_1 and B_2, respectively. The interpretation of a n-ary function symbol $f \in \mathcal{F}_b$ defines a function f^S from B_1^n to B_2. As usual, we distinguish between *updatable* dynamic functions and static functions.

Let $\Upsilon = \Upsilon_1 \cup \Upsilon_2 \cup \mathcal{F}_b$ be a signature of metafinite states. Fix a countable set $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ of first-order variables. Variables in $\mathcal{X}_1$, denoted with standard lowercase letters $x, y, z, \ldots$, range over the primary part of a meta-finite state (i.e., the finite set B_1), whereas variables in $\mathcal{X}_2$, denoted with typewriter-style lowercase letters $\mathtt{x}, \mathtt{y}, \mathtt{z}, \ldots$, range over B_2. The set of first-order terms $\mathcal{T}_{\Upsilon,\mathcal{X}}$ of vocabulary Υ is defined in a similar way than in meta-finite model theory [8]. That is, $\mathcal{T}_{\Upsilon,\mathcal{X}}$ is constituted by the set $\mathcal{T}_p$ of *point terms* and the set $\mathcal{T}_a$ of *algorithmic terms*. The set of point terms $\mathcal{T}_p$ is the closure of the set $\mathcal{X}_1$ of variables under the application of function symbols in Υ_1. The set of algorithmic terms $\mathcal{T}_a$ is defined inductively: Every variable in $\mathcal{X}_2$ is an algorithmic term in $\mathcal{T}_a$; If $t_1, \ldots, t_n$ are point terms in $\mathcal{T}_p$ and f is an n-ary bridge function symbol in $\mathcal{F}_b$, then $f(t_1, \ldots, t_n)$ is an algorithmic term in $\mathcal{T}_a$; if $t_1, \ldots, t_n$ are algorithmic terms in $\mathcal{T}_a$ and f is an n-ary function symbol in Υ_2, then $f(t_1, \ldots, t_n)$ is an algorithmic term in $\mathcal{T}_a$; nothing else is an algorithmic term in $\mathcal{T}_b$.

Let S be a meta finite state of signature Υ. A *valuation* or *variable assignment* ζ is a function that assigns to every variable in $\mathcal{X}_1$ a value in the base set B_1 of the primary part of S and to every variable in $\mathcal{X}_2$ a value in the base set B_2 of the secondary part of S. The value $val_{S,\zeta}(t)$ of a term $t \in \mathcal{T}_{\Upsilon,\mathcal{X}}$ in the state S under the valuation ζ is defined as usual in first-order logic. The *first-order logic of metafinite structures* (states) is defined as the first-order logic with equality which is built up from equations between terms in $\mathcal{T}_{\Upsilon,\mathcal{X}}$ by using the standard connectives and first-order quantifiers. Its semantics is defined in the standard way. The truth value of a first-order formula of meta finite structures φ in S under the valuation ζ is denoted as $[\![\varphi]\!]_{S,\zeta}$.

In our definition of ASM rule, we use the fact that function arguments can be read as tuples. Thus, if f is an n-ary function and $t_1, \ldots, t_n$ are arguments for f, we write $f(t)$ where t is a term which evaluates to the tuple $(t_1, \ldots, t_n)$, instead of $f(t_1, \ldots, t_n)$. This is not strictly necessary, but it greatly simplifies the presentation of the technical details in this paper. Let t and s denote terms in $\mathcal{T}_p$, let $\mathtt{t}$ and $\mathtt{s}$ denote terms in $\mathcal{T}_a$ and let φ denote a first-order formula of metafinite structures of vocabulary Υ. The set of *ASM rules* over Υ is inductively defined as follows:

- *update rule* 1: $f(t) := s$ (where $f \in \Upsilon_1$);
- *update rule* 2: $f(\mathtt{t}) := \mathtt{s}$ (where $f \in \Upsilon_2$);
- *update rule* 3: $f(t) := \mathtt{s}$ (where $f \in \mathcal{F}_b$);
- *conditional rule*: **if** φ **then** r **endif**
- *forall rule*: **forall** x **with** φ **do** r **enddo**
- *bounded choice rule*: **choose** x **with** φ **do** r **enddo**
- *unbounded choice rule*: **choose** $\mathtt{x}$ **with** φ **do** r **enddo**
- *parallel rule*: **par** r_1 r_2 **endpar** (execute the rules r_1 and r_2 in parallel);
- *sequence rule*: **seq** r_1 r_2 **endseq** (first execute rule r_1 and then rule r_2).

If r is an ASM rule of signature Υ and S is a state of Υ, then we associate to them a set $\Delta(r, S, \zeta)$ of update sets which depends on the variable assignment ζ. Let $\zeta[x \mapsto a]$ denote the variable assignment which coincides with ζ except that it assigns the value a to x. We formally define in Figure 1 the sets of update sets yielded by the ASM rules. Items 1–3 in Figure 1 correspond to the update rules 1–3, respectively. Each update rules yields a set which contains a single update set, which in turns contains a single update to a function of S. Depending on whether the function name f belongs to Υ_1, Υ_2 or $\mathcal{F}_b$, the produced update corresponds to a function in the primary or secondary part of S or to a bridge function, respectively. The choice rules introduce non-determinism. The bounded choice rule yields a finite set of update sets, since x range over the (finite) primary part of S (see item 6 in Figure 1). The unbounded choice rule yields a possibly infinite set of update sets (see item 7 in Figure 1). In this latter case, $\mathtt{x}$ range over the (possible infinite) secondary part of S and it might happen that there are infinite valuations for $\mathtt{x}$ that satisfy the condition φ, each resulting in a different update set. All other rules only rearrange updates into different update sets. Update sets are explained in more detail in Section 5.2.

Remark 1. For every state S, ASM rule r and variable assignment ζ, we have that every $\Delta \in \Delta(r, S, \zeta)$ is a finite set of updates. This is a straightforward consequence of the fact that the variable x in the definition of the **forall** rule ranges over the (finite) primary part of S, and it is also the case in the ASM thesis for parallel algorithms of Blass and Gurevich [3,4] where it is implicitly assumed that the **forall** rule in the parallel ASMs range over finite hereditary multisets. See our work in [6] for a detailed explanation. Regarding the set $\Delta(r, S, \zeta)$ of update sets, we note that it might be infinite since the unbounded choice rule can potentially produce infinitely many update sets. In fact, this is the case if we consider the first unbounded choice rule in Example 1.

Formally, a *non-deterministic parallel* ASM M over a signature Υ of metafinite states consists of: (a) a set $\mathcal{S}$ of metafinite states over Υ, (b) non-empty subsets $\mathcal{S}_I \subseteq \mathcal{S}$ of *initial states* and $\mathcal{S}_F \subseteq \mathcal{S}$ of *final states*, and (c) a *closed* ASM rule r over Υ, i.e., a rule r in which all free variables in the first-order formulae of the rule are bounded by **forall** or **choose** constructs.

Every non-deterministic parallel ASM M defines a corresponding *successor relation* δ over $\mathcal{S}$ which is determined by the main rule r of M. A pair of states (S_1, S_2) belongs to δ iff there is a consistent update set $\Delta \in \Delta(r, S)$ (the valuation ζ is omitted from $\Delta(r, S, \zeta)$ since r is closed) such that S_2 is the unique state resulting from updating S_1 with Δ. A *run* of an ASM M is a finite sequence $S_0, \ldots, S_n$ of states with $S_0 \in \mathcal{S}_I$, $S_n \in \mathcal{S}_F$, $S_i \notin \mathcal{S}_F$ for $0 < i < n$, and $(S_i, S_{i+1}) \in \delta$ for all $i = 0, \ldots, n-1$.

The following example, adapted from [10], illustrates a parallel ASMs with bounded non-determinism.

Example 2. We consider metafinite states with: (a) a primary part formed by a connected weighted graph $G = (V, E)$, (b) a secondary part formed by the set of natural numbers $\mathbb{N}$, and (c) a bridge function *weight* from the set of edges in E to

1. $\Delta(f(t) := s, S, \zeta) = \{\{(f, (a), b)\}\}$ for $a = val_{S,\zeta}(t) \in B_1$ and $b = val_{S,\zeta}(s) \in B_1$

2. $\Delta(f(\mathtt{t}) := \mathtt{s}, S, \zeta) = \{\{(f, (a), b)\}\}$ for $a = val_{S,\zeta}(\mathtt{t}) \in B_2$ and $b = val_{S,\zeta}(\mathtt{s}) \in B_2$

3. $\Delta(f(t) := \mathtt{s}, S, \zeta) = \{\{(f, (a), b)\}\}$ for $a = val_{S,\zeta}(t) \in B_1$ and $b = val_{S,\zeta}(\mathtt{s}) \in B_2$

4. $\Delta(\textbf{if } \varphi \textbf{ then } r \textbf{ endif}, S, \zeta) = \begin{cases} \Delta(r, S, \zeta) & \text{if } [\![\varphi]\!]_{S,\zeta} = \text{true} \\ \{\emptyset\} & \text{otherwise} \end{cases}$

5. $\Delta(\textbf{forall } x \textbf{ with } \varphi \textbf{ do } r \textbf{ enddo}, S, \zeta) =$
$\{\Delta_1 \cup \cdots \cup \Delta_n \mid \Delta_i \in \Delta(r, S, \zeta[x \mapsto a_i])\}$,
where $\{a_1, \ldots, a_n\} = \{a_i \in B_1 \mid [\![\varphi]\!]_{S,\zeta[x \mapsto a_i]} = \mathit{true}\}$

6. $\Delta(\textbf{choose } x \textbf{ with } \varphi \textbf{ do } r \textbf{ enddo}, S, \zeta) =$
$\bigcup_{a_i \in B_1} \{\Delta(r, S, \zeta[x \mapsto a_i]) \mid [\![\varphi]\!]_{S,\zeta[x \mapsto a_i]} = \text{true}\}$

7. $\Delta(\textbf{choose } \mathtt{x} \textbf{ with } \varphi \textbf{ do } r \textbf{ enddo}, S, \zeta) =$
$\bigcup_{a_i \in B_2} \{\Delta(r, S, \zeta[x \mapsto a_i]) \mid [\![\varphi]\!]_{S,\zeta[x \mapsto a_i]} = \text{true}\}$

8. $\Delta(\textbf{par } r_1\ r_2 \textbf{ endpar}, S, \zeta) =$
$\{\Delta_1 \cup \Delta_2 \mid \Delta_1 \in \Delta(r_1, S, \zeta) \text{ and } \Delta_2 \in \Delta(r_2, S, \zeta)\}$

9. $\Delta(\textbf{seq } r_1\ r_2 \textbf{ endseq}, S, \zeta) =$
$\{\Delta_1 \oslash \Delta_2 \mid \Delta_1 \in \Delta(r_1, S, \zeta) \text{ is consistent and } \Delta_2 \in \Delta(r_2, S + \Delta_1, \zeta)\} \cup$
$\{\Delta_1 \in \Delta(r_1, S, \zeta) \mid \Delta_1 \text{ is inconsistent}\}$,
where $\Delta_1 \oslash \Delta_2 = \Delta_2 \cup \{(\ell, a) \in \Delta_1 \mid \ell \neq \ell' \text{ for all } (\ell', a') \in \Delta_2\}$

Fig. 1. Sets of update sets of non-deterministic parallel ASMs

$\mathbb{N}$. Apart from the static (Boolean) function symbols V and E, the vocabulary of the primary part of the states also includes dynamic function symbols *label* and T, and static function symbols *first* and *second*, the last two for extracting the first and second element of an ordered pair, respectively. Since G is an undirected graph, we have that $(x, y) \in E$ iff $(y, x) \in E$.

The non-deterministic parallel ASM in this example, which we denote as M, formally expresses Kruskal's algorithm [12] for computing the *minimum spanning tree* in a connected, weighted graph. Recall that a spanning tree T of a graph G is a tree such that every pair of nodes in G are connected via edges in T. We say that T is minimum if the sum of the weights of all its edges is the least among all spanning trees of G. We assume that in every initial state of M, $\mathit{label}(x) = x$ for every $x \in V$ and that $T((x, y)) = \mathit{false}$ for every $(x, y) \in E$.

The condition in the first **choose** rule is simply ensuring that the chosen edge x is eligible, i.e., that the nodes $\mathit{first}(x)$ and $\mathit{second}(x)$ that make up the endpoints of the edge x have different labels, and that x has minimal weight among the set of eligible edges. The following two update rules simply add the

edge x to the tree T. The second **choose** rule reflects the fact that from the point of view of the correctness of the algorithm, it does not matter which endpoint y of the edge x we choose at this stage. Finally, the **forall** rule simply relabels (as expected) every node with the same label than the endpoint y of x (including the node y itself) with the label of the opposite endpoint of x.

```
choose x with E(x) ∧ label(first(x)) ≠ label(second(x))∧
          ∀y (E(y) ∧ label(first(y)) ≠ label(second(y)) → weight(y) ≥ weight(x)) do
  T(x) := true
  T((second(x), first(x))) := true
  choose y with y = first(x) ∨ y = second(x) do
      forall z with label(z) = label(y) do
          if label(y) = label(first(x)) then label(z) := label(second(x)) endif
          if label(y) = label(second(x)) then label(z) := label(first(x)) endif
      enddo
  enddo
enddo
```

4 A Logic for Non-Deterministic Parallel ASMs

The logic for non-deterministic parallel ASMs (denoted $\mathcal{L}$) is a dynamic first-order logic extended with membership predicates over finite sets, an update set predicate and a multi-modal operator. $\mathcal{L}$ is defined over many sorted first-order structures which have:

- a *finite individual sort* with variables $x_1, x_2, ...$ which range over a finite domain D_1,
- an *individual sort* with variables $\mathtt{x}_1, \mathtt{x}_2, \ldots$, which range over a (possibly infinite) domain D_2, and
- a *predicate sort* with variables $x_1^1, x_2^1, \ldots$, which range over the domain P_1 formed by all finite subsets (relations) on $\mathcal{F}_{dyn} \times (D_1 \cup D_2) \times (D_1 \cup D_2)$.
- a *predicate sort* with variables $x_1^2, x_2^2, \ldots$, which range over the domain P_2 formed by all finite subsets (relations) on $\mathcal{F}_{dyn} \times (D_1 \cup D_2) \times (D_1 \cup D_2) \times D_1$.

A signature Σ of the logic $\mathcal{L}$ comprises a finite set F_1 of names for functions on D_1, a finite set F_2 of names for functions on D_2, and a finite set F_b of names for functions which take arguments from D_1 and return values on D_2.

We define terms of $\mathcal{L}$ by induction. Variables $x_1, x_2, ...$ and $\mathtt{x}_1, \mathtt{x}_2, \ldots$ are terms of the first and second individual sort, respectively. Variables $x_1^1, x_2^1, \ldots$ and $x_1^2, x_2^2, \ldots$ are terms of the first and second predicate sort, respectively. If f is an n-ary function name in F_1 and $t_1, \ldots, t_n$ are terms of the first individual sort, then $f(t_1, \ldots, t_n)$ is a term of the first individual sort. If f is an n-ary function name in F_2 and $t_1, \ldots, t_n$ are terms of the second individual sort, then $f(t_1, \ldots, t_n)$ is a term of the second individual sort. If f is an n-ary function name in F_b and $t_1, \ldots, t_n$ are terms of the first individual sort, then $f(t_1, \ldots, t_n)$ is a term of the second individual sort.

The formulae of $\mathcal{L}$ are those generated by the following grammar:

$$\varphi, \psi ::= s = t \mid s_a = t_a \mid \neg\varphi \mid \varphi \wedge \psi \mid \forall x(\varphi) \mid \forall \mathbf{x}(\varphi) \mid \forall x^1(\varphi) \mid \forall x^2(\varphi) \mid$$
$$\in^1(x^1, f, t_0, s_0) \mid \in^2(x^2, f, t_0, s_0, s) \mid \mathrm{upd}(r, x^1) \mid [x^1]\varphi$$

where s and t denote terms of the first individual sort, s_a and t_a denote terms of the second individual sort, f is a dynamic function symbol, r is an ASM rule and, t_0 and s_0 denote terms of either the first or the second individual sort.

The interpretation of terms and the semantics of the first-order formulae is defined in the standard way. This includes equality which is used under a fixed interpretation and only between terms of a same individual sort.

The update set predicate $\mathrm{upd}(r, x^1)$ states that the *finite* update set represented by x^1 is generated by the rule r. Let S be a state of some signature Σ of the logic $\mathcal{L}$. Let ζ be a variable assignment over S which maps each variable of the first and second individual sort to a value in D_1 and D_2, respectively, and maps each variable of the first and second predicate sort to a value in P_1 and P_2, respectively. The truth value of $\mathrm{upd}(r, x^1)$ is defined by $[\![\mathrm{upd}(r, x^1)]\!]_{S,\zeta} = \mathit{true}$ iff $\mathit{val}_{S,\zeta}(x^1) \in \Delta(r, S, \zeta)$.

The set membership predicate $\in^1(x^1, f, t_0, s_0)$ indicates that (f, t_0, s_0) is an update in the update set represented by x^1 while the auxiliary set membership predicate $\in^2(x^2, f, t_0, s_0, s)$ is used to keep track of which parallel branch produced each update in x^2. Their truth values are formally defined as follows:
$[\![\in^1(x^1, f, t_0, s_0)]\!]_{S,\zeta} = \mathit{true}$ iff $(f, \mathit{val}_{S,\zeta}(t_0), \mathit{val}_{S,\zeta}(s_0)) \in \mathit{val}_{S,\zeta}(x^1)$
$[\![\in^2(x^2, f, t_0, s_0, s)]\!]_{S,\zeta} = \mathit{true}$ iff$(f, \mathit{val}_{S,\zeta}(t_0), \mathit{val}_{S,\zeta}(s_0), \mathit{val}_{S,\zeta}(s)) \in \mathit{val}_{S,\zeta}(x^2)$

Finally, we use $[x^1]\varphi$ to express the evaluation of φ over the successor state obtained by applying the updates in x^1 to the current state. Its truth value is defined by: $[\![[x^1]\varphi]\!]_{S,\zeta} = \mathit{true}$ iff $\Delta = \zeta(x^1)$ is inconsistent or $[\![\varphi]\!]_{S+\Delta,\zeta} = \mathit{true}$ for $\zeta(x^1) = \Delta \in \Delta(r, S, \zeta)$. That is, when $\Delta = \zeta(x^1)$ is inconsistent, successor states for the current state S do not exist and thus $S + \Delta$ is undefined. In this case, $[x^1]\varphi$ is interpreted as *true*. With the use of the modal operator $[\,]$ for an update set $\Delta = \zeta(x^1)$ (i.e., $[x^1]$), $\mathcal{L}$ is empowered to be a multi-modal logic.

We say that a formula φ of $\mathcal{L}$ is *static* if all the function symbols which appear in φ are static and say that it is *pure* if it is generated by the following grammar: $\varphi, \psi ::= s = t \mid s_a = t_a \mid \neg\varphi \mid \varphi \wedge \psi \mid \forall x(\varphi) \mid \forall \mathbf{x}(\varphi)$.

Since metafinite states are just a special kind of two sorted first-order structures in which one of the sorts is finite, we can identify every metafinite state S of $\mathcal{L}$ with a corresponding many sorted first-order structure S' of the class used in definition of $\mathcal{L}$. This can be done by taking the domains D_1 and D_2 of the individual sorts of S' to be the base sets B_1 and B_2 of S, respectively, the sets F_1, F_2 and F_b of function names of the signature Σ of S' to be the sets Υ_1, Υ_2 and $\mathcal{F}_b$ of the signature Υ of S, respectively, and the interpretation in S' of the function names in Σ to coincide with the interpretation in S of the corresponding function symbols in Υ. Following this transformation we have that for every state S, every corresponding pair of many sorted first-order structure S' and S'' are isomorphic by an isomorphism which is the identity among elements of the individual sorts. Thus, we can talk of *the* many sorted structure S corresponding

to a state S and, when it is clear from the context, we can even talk of the state S meaning the many sorted structure S.

In what follows, we use the somehow clearer and more usual syntax of second-order logic to denote the set membership predicates and the quantification over the predicate sorts. Thus we use upper case letters $X, Y, \ldots$ and $\mathcal{X}, \mathcal{Y}, \ldots$ to denote variables $x_1^1, x_2^1, \ldots$ and $x_1^2, x_2^2, \ldots$ of the first and second predicate sorts, respectively, and we write $\forall X(\varphi)$, $\forall \mathcal{X}(\varphi)$, $[X]\varphi$, $X(f, t_0, s_0)$, $\mathcal{X}(f, t_0, s_0, s)$ and $\mathrm{upd}(r, X)$ instead of $\forall x^1(\varphi)$, $\forall x^2(\varphi)$, $[x^1]\varphi$, $\in^1(x^1, f, t_0, s_0)$, $\in^1(x^1, f, t_0, s_0, s)$ and $\mathrm{upd}(r, x^1)$, respectively. Furthermore, in our formulae we use disjunction $\vee$, implication $\rightarrow$, double implication $\leftrightarrow$ and existential quantification $\exists$. All of them are defined as abbreviations in the usual way.

Example 3. $\mathcal{L}$ can express properties of the ASM in Example 2 such as:

- If r yields in the current state S an update set Δ with an update (T, x, true), then in the successor state $S + \Delta$ the vertices of x have a same label.

$$\forall X(\mathrm{upd}(r, X) \rightarrow \forall x(X(T, x, \mathit{true}) \rightarrow [X](\mathit{label}(\mathit{first}(x)) = \mathit{label}(\mathit{second}(x)))))$$

- Each update set yielded by r updates T in no more than one location.

$$\forall X(\mathrm{upd}(r, X) \rightarrow \neg(\exists xy(X(T, x, \mathit{true}) \wedge X(T, y, \mathit{true}) \wedge x \neq y)))$$

- If an edge x meets in a state S the criteria of the first **choose** rule in r, then there is an update set $\Delta \in \Delta(r, S)$ such that $T(x) = \mathit{true}$ holds in $S + \Delta$.

$$\begin{aligned}&\forall x(E(x) \wedge \mathit{label}(\mathit{first}(x)) \neq \mathit{label}(\mathit{second}(x)) \wedge \\ &\quad \forall y(E(y) \wedge \mathit{label}(\mathit{first}(y)) \neq \mathit{label}(\mathit{second}(y)) \rightarrow \mathit{weight}(y) \geq \mathit{weight}(x)) \\ &\quad \rightarrow \exists X(\mathit{upd}(r, X) \wedge [X](T(x) = \mathit{true})))\end{aligned}$$

5 A Proof System

In this section we develop a proof system for the logic $\mathcal{L}$ for non-deterministic parallel ASMs.

Definition 1. *We say that a state S is a* model *of a formula φ (denoted as $S \models \varphi$) iff $[\![\varphi]\!]_{S,\zeta} = \mathit{true}$ holds for every variable assignment ζ. If Ψ is a set of formulae, we say that S* models *Ψ (denoted as $S \models \Psi$) iff $S \models \varphi$ for each $\varphi \in \Psi$. A formula φ is said to be a* logical consequence *of a set Ψ of formulae (denoted as $\Psi \models \varphi$) if for every state S, if $S \models \Psi$, then $S \models \varphi$. A formula φ is said to be* valid *(denoted as $\models \varphi$) if $[\![\varphi]\!]_{S,\zeta} = \mathit{true}$ in every state S for every variable assignment ζ. A formula φ is said to be* derivable *from a set Ψ of formulae (denoted as $\Psi \vdash_{\mathfrak{R}} \varphi$) if there is a deduction from formulae in Ψ to φ by using a set $\mathfrak{R}$ of axioms and inference rules.*

We will define such a set $\mathfrak{R}$ of axioms and rules in Subsection 5.3. Then we simply write $\vdash$ instead of $\vdash_{\mathfrak{R}}$. We also define equivalence between two ASM rules. Two equivalent rules r_1 and r_2 are either both defined or both undefined.

Definition 2. *Let r_1 and r_2 be two ASM rules. Then r_1 and r_2 are* equivalent *(denoted as $r_1 \equiv r_2$) if for every state S it holds that $S \models \forall X(\mathrm{upd}(r_1, X) \leftrightarrow \mathrm{upd}(r_2, X))$.*

5.1 Consistency

In [13] Nanchen and Stärk use a predicate $\mathrm{Con}(r)$ as an abbreviation for the statement that the rule r is consistent. As every rule r in their work is deterministic, there is no ambiguity with the reference to the update set associated with r, i.e., each deterministic rule r generates exactly one (possibly empty) update set. Thus a deterministic rule r is consistent iff the update set generated by r is consistent. However, in our logic $\mathcal{L}$, the presence of non-determinism makes the situation less straightforward.

Let r be an ASM rule and Δ be an update set. Then the consistency of an update set Δ, denoted by the formula $\mathrm{conUSet}(X)$ (where X represents Δ), can be expressed as:

$$\mathrm{conUSet}(X) \equiv \bigwedge_{f \in \mathcal{F}_{dyn}} \forall xyz((X(f, x, y) \wedge X(f, x, z)) \rightarrow y = z) \tag{1}$$

Then $\mathrm{con}(r, X)$ is an abbreviation of the following formula which expresses that an update set Δ (represented by the variable X) generated by the rule r is consistent.

$$\mathrm{con}(r, X) \equiv \mathrm{upd}(r, X) \wedge \mathrm{conUSet}(X) \tag{2}$$

As the rule r may be non-deterministic, it is possible that r yields several update sets. Thus, we develop the consistency of ASM rules in two versions:

- A rule r is *weakly consistent* (denoted as $\mathrm{wcon}(r)$) if at least one update set generated by r is consistent. This can be expressed as follows:

$$\mathrm{wcon}(r) \equiv \exists X(\mathrm{con}(r, X)) \tag{3}$$

- A rule r is *strongly consistent* (denoted as $\mathrm{scon}(r)$) if every update set generated by r is consistent. This can be expressed as follows:

$$\mathrm{scon}(r) \equiv \forall X(\mathrm{upd}(r, X) \Rightarrow \mathrm{con}(r, X)) \tag{4}$$

In the case that a rule r is deterministic, the weak notion of consistency coincides with the strong notion of consistency, i.e., $\mathrm{wcon}(r) \leftrightarrow \mathrm{scon}(r)$.

5.2 Update Sets

We present the axioms for the predicate $upd(r, X)$ in Figure 2. To simplify the presentation, we give the formulae only for the case in which all the function symbols in $\mathcal{F}_{dyn}$ correspond to functions on the primary part (finite individual sort) of the state. To deal with dynamic function symbols corresponding to

U1. $\mathrm{upd}(f(t) := s, X) \leftrightarrow X(f,t,s) \wedge \forall xy(X(f,x,y) \rightarrow x = t \wedge y = s) \wedge \bigwedge_{f \neq f' \in \mathcal{F}_{dyn},} \forall xy(\neg X(f',x,y))$

U2. $\mathrm{upd}(\mathbf{if}\,\varphi\,\mathbf{then}\,r\,\mathbf{endif}, X) \leftrightarrow (\varphi \wedge \mathrm{upd}(r,X)) \vee (\neg\varphi \wedge \bigwedge_{f \in \mathcal{F}_{dyn}} \forall xy(\neg X(f,x,y))$

U3. $\mathrm{upd}(\mathbf{forall}\,x\,\mathbf{with}\,\varphi\,\mathbf{do}\,r\,\mathbf{enddo}, X) \leftrightarrow$

$$\exists \mathcal{X}\big(\forall x\big((\varphi \rightarrow \exists Y(\mathrm{upd}(r,Y) \wedge \bigwedge_{f \in \mathcal{F}_{dyn}} \forall y_1 y_2(Y(f,y_1,y_2) \leftrightarrow \mathcal{X}(f,y_1,y_2,x)))) \wedge (\neg\varphi \rightarrow \bigwedge_{f \in \mathcal{F}_{dyn}} \forall y_1 y_2(\neg\mathcal{X}(f,y_1,y_2,x)))\big) \wedge \bigwedge_{f \in \mathcal{F}_{dyn}} \forall x_1 x_2(X(f,x_1,x_2) \leftrightarrow \exists x_3(\mathcal{X}(f,x_1,x_2,x_3)))\big)$$

U4. $\mathrm{upd}(\mathbf{par}\,r_1\ r_2\,\mathbf{endpar}, X) \leftrightarrow \exists Y_1 Y_2(\mathrm{upd}(r_1,Y_1) \wedge \mathrm{upd}(r_2,Y_2) \wedge \bigwedge_{f \in \mathcal{F}_{dyn}} \forall xy(X(f,x,y) \leftrightarrow (Y_1(f,x,y) \vee Y_2(f,x,y)))$

U5. $\mathrm{upd}(\mathbf{choose}\,x\,\mathbf{with}\,\varphi\,\mathbf{do}\,r\,\mathbf{enddo}, X) \leftrightarrow \exists x(\varphi \wedge \mathrm{upd}(r,X))$

U6. $\mathrm{upd}(\mathbf{choose}\,\mathbf{x}\,\mathbf{with}\,\varphi\,\mathbf{do}\,r\,\mathbf{enddo}, X) \leftrightarrow \exists \mathbf{x}(\varphi \wedge \mathrm{upd}(r,X))$

U7. $\mathrm{upd}(\mathbf{seq}\,r_1\ r_2\,\mathbf{endseq}, X) \leftrightarrow \big(\mathrm{upd}(r_1,X) \wedge \neg\mathrm{con}(X)\big) \vee \big(\exists Y_1 Y_2(\mathrm{upd}(r_1,Y_1) \wedge \mathrm{con}(Y_1) \wedge [Y_1]\mathrm{upd}(r_2,Y_2) \wedge \bigwedge_{f \in \mathcal{F}_{dyn}} \forall xy(X(f,x,y) \leftrightarrow ((Y_1(f,x,y) \wedge \forall z(\neg Y_2(f,x,z))) \vee Y_2(f,x,y)))))\big)$

Fig. 2. Axioms for predicate upd(r,X)

function of the secondary part and to bridge functions, we only need to slightly change the formulae by replacing some of the first-order variables in $\mathcal{X}_1$ by first-order variables in $\mathcal{X}_2$. For instance, if f is a bridge function symbol, we should write $\forall x\mathbf{y}(X(f,x,\mathbf{y}) \rightarrow x = t \wedge \mathbf{y} = \mathbf{s})$ instead of $\forall xy(X(f,x,y) \rightarrow x = t \wedge y = s)$.

In the following we explain Axioms **U1**-**U7** in turn. We assume a state S of some signature Υ and base set $B = B_1 \cup B_2$, where B_1 is the base set of the *finite* primary part of S. We also assume a variable assignment ζ.

As in our case an ASM rule may be non-deterministic, a straightforward extension from the formalisation of the **forall** and **par** rules used in the logic for ASMs in [13] would not work for Axioms **U3** and **U4**. The axioms correspond to the definition of update sets in Figure 1.

– Axiom **U1** says that X is an update yielded by the assignment rule $f(t) := s$ iff it contains exactly one update which is (f,t,s).

– Axiom **U2** asserts that, if the formula φ evaluates to *true*, then X is an update set yielded by the conditional rule **if** φ **then** r **endif** iff X is an update set yielded by the rule r. Otherwise, the conditional rule yields only an empty update set.
– Axiom **U3** states that X is an update set yielded by the rule **forall** x **with** φ **do** r **enddo** iff X coincides with $\Delta_{a_1} \cup \cdots \cup \Delta_{a_n}$, where $\{a_1, \ldots, a_n\} = \{a_i \in B_1 \mid val_{S,\zeta[x \mapsto a_i]}(\varphi) = true\}$ and Δ_{a_i} (for $1 \leq i \leq n$) is an update set yielded by the rule r under the variable assignment $\zeta[x \mapsto a_i]$. Note that the update sets $\Delta_{a_1}, \ldots, \Delta_{a_n}$ are encoded into $\mathcal{X}$.
– Axiom **U4** states that X is an update set yielded by the parallel rule **par** r_1 r_2 **endpar** iff it corresponds to the union of an update set yielded by r_1 and an update set yielded by r_2.
– Axioms **U5** asserts that X is an update set yielded by the rule **choose** x **with** φ **do** r **enddo** iff it is an update set yielded by the rule r under a variable assignment $\zeta[x \mapsto a]$ which satisfies φ.
– Axiom **U6** is similar to Axiom **U5**, but for the case of the **choose x with** φ **do** r **enddo** rule.
– Axiom **U7** asserts that X is an update set yielded by a sequence rule **seq** r_1 r_2 **endseq** iff it corresponds to either an inconsistent update set yielded by rule r_1, or to an update set formed by the updates in an update set Y_2 yielded by rule r_2 in a successor state $S + Y_1$, where Y_1 encodes a consistent set of updates produced by rule r_1, plus the updates in Y_1 that correspond to locations other than the locations updated by Y_2.

The following lemma is an easy consequence of the axioms in Figure 2.

Lemma 1. *Every formula in the logic $\mathcal{L}$ can be replaced by an equivalent formula not containing any subformulae of the form* $\mathrm{upd}(r, X)$.

Remark 2. The inclusion of the parameter X in the predicate $\mathrm{upd}(r, X)$ is important because a rule r in a non-deterministic parallel ASM rule may be associated with multiple update sets, and thus we need a way to specify which update set yielded by rule r is meant.

5.3 Axioms and Inference Rules

Now we can present a set of axioms and inference rules which constitute a proof system for the logic $\mathcal{L}$. To avoid unnecessary repetitions of almost identical axioms and rules, we describe them only considering variables of the first individual sort, but the exact same axioms and inference rules are implicitly assumed for the case of variables of the second individual sort as well as for variables of the predicate sorts. In the definition of the set of axioms and rules, we sometimes use $\varphi[t/x]$ to denote the substitution of a term t for a variable x in a formula φ. That is, $\varphi[t/x]$ is the result of replacing all free instances of x by t in φ provided that no free variable of t becomes bound after substitution.

Formally, the set $\mathfrak{R}$ of axioms and inference rules is formed by:

- The axioms **U1**-**U7** in Fig. 2 which assert the properties of $\mathrm{upd}(r, X)$.
- Axiom **M1** and Rules **M2**-**M3** from the axiom system K of modal logic, which is the weakest normal modal logic system [11]. Axiom **M1** is called *Distribution Axiom* of K, Rule **M2** is called *Necessitation Rule* of K and Rule **M3** is the inference rule called *Modus Ponens* in the classical logic. By using these axiom and rules together, we are able to derive all modal properties that are valid in Kripke frames.

  M1 $[X](\varphi \to \psi) \to ([X]\varphi \to [X]\psi)$

  M2 $\varphi \vdash [X]\varphi$ **M3** $\varphi, \varphi \to \psi \vdash \psi$
- Axiom **M4** asserts that, if an update set Δ is not consistent, then there is no successor state obtained after applying Δ over the current state and thus $[X]\varphi$ (for X interpreted by Δ) is interpreted as true for any formula φ. As applying a consistent update set Δ over the current state is deterministic, Axiom **M5** describes the deterministic accessibility relation in terms of $[X]$.

  M4 $\neg\mathrm{conUSet}(X) \to [X]\varphi$ **M5** $\neg[X]\varphi \to [X]\neg\varphi$
- Axiom **M6** is called *Barcan Axiom*. It originates from the fact that all states in a run of a non-deterministic parallel ASM have the same base set, and thus the quantifiers in all states always range over the same set of elements.

  M6 $\forall x([X]\varphi) \to [X]\forall x(\varphi)$
- Axioms **M7** and **M8** assert that the interpretation of static or pure formulae is the same in all states of non-deterministic parallel ASMs, since they are not affected by the execution of any ASM rule r.

  M7 $\mathrm{con}(r, X) \wedge \varphi \to [X]\varphi$ for static or pure φ

  M8 $\mathrm{con}(r, X) \wedge [X]\varphi \to \varphi$ for static or pure φ
- Axiom **A1** asserts that, if a consistent update set Δ (represented by X) does not contain any update to the location (f, x), then the content of (f, x) in a successor state obtained after applying Δ is the same as its content in the current state. Axiom **A2** asserts that, if a consistent update set Δ does contain an update which changes the content of the location (f, x) to y, then the content of (f, x) in the successor state obtained after applying Δ is y.

  A1 $\mathrm{conUSet}(X) \wedge \forall z(\neg X(f, x, z)) \wedge f(x) = y \to [X]f(x) = y$

  A2 $\mathrm{conUSet}(X) \wedge X(f, x, y) \to [X]f(x) = y$
- The following are axiom schemes from classical logic.

  P1 $\varphi \to (\psi \to \varphi)$

  P2 $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$

  P3 $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$
- The following four inference rules describe when the universal and existential quantifiers can be added to or deleted from a statement. Rules **UI**, **EG**, **UG** and **EI** are usually known as *Universal Instantiation*, *Existential Generalisation*, *Universal Generalisation* and *Existential Instantiation*, respectively.

  UI $\forall x(\varphi) \vdash \varphi[t/x]$ if φ is pure or t is static.

EG $\varphi[t/x] \vdash \exists x(\varphi)$ if φ is pure or t is static.

UG $\varphi[t_a/x] \vdash \forall x(\varphi)$ if $\varphi[t_a/x]$ holds for every element a in the domain of x and corresponding term t_a representing a, and further φ is pure or every t_a is static.

EI $\exists x(\varphi) \vdash \varphi[t/x]$ if t represents a valuation for x which satisfies φ, and further φ is pure or t is static.

– The following are the equality axioms from first-order logic with equality. Axiom **EQ1** asserts the reflexivity property while Axiom **EQ2** asserts the substitutions for functions.

EQ1 $t = t$ for static term t

EQ2 $t_1 = t_{n+1} \wedge ... \wedge t_n = t_{2n} \rightarrow f(t_1, ..., t_n) = f(t_{n+1}, ..., t_{2n})$ for any function f and static terms t_i $(i = 1, ..., 2n)$.

– The following axiom is taken from dynamic logic, asserting that executing a **seq** rule equals to executing rules sequentially.

DY1 $\exists X(\mathrm{upd}(\mathbf{seq}\ r_1\ r_2\ \mathbf{endseq}, X) \wedge [X]\varphi) \leftrightarrow$
$\exists X_1(\mathrm{upd}(r_1, X_1) \wedge [X_1]\exists X_2(\mathrm{upd}(r_2, X_2) \wedge [X_2]\varphi))$

– Axiom **E** is the extensionality axiom.

E $r_1 \equiv r_2 \rightarrow \exists X_1 X_2((\mathrm{upd}(r_1, X_1) \wedge [X_1]\varphi) \leftrightarrow (\mathrm{upd}(r_2, X_2) \wedge [X_2]\varphi))$

The following soundness theorem for the proof system is relatively straightforward, since the non-standard axioms and rules are just a formalisation of the definitions of the semantics of rules, update sets and update multisets.

Theorem 1. *Let φ be a formula from $\mathcal{L}$ and let Φ be a set of formulae also from $\mathcal{L}$ (all of them of the same vocabulary as φ). If $\Phi \vdash \varphi$, then $\Phi \models \varphi$.*

6 Derivation

In this section we present some properties of the logic for non-deterministic parallel ASMs which are implied by the axioms and rules from the previous section. This includes properties known for the logic for ASMs [13]. In particular, the logic for ASMs uses the modal expressions $[r]\varphi$ and $\langle r\rangle\varphi$ with the following semantics:

– $[\![[r]\varphi]\!]_{S,\zeta} = \mathit{true}$ iff $[\![\varphi]\!]_{S+\Delta,\zeta} = \mathit{true}$ for all consistent $\Delta \in \Delta(r, S, \zeta)$.
– $[\![\langle r\rangle\varphi]\!]_{S,\zeta} = \mathit{true}$ iff $[\![\varphi]\!]_{S+\Delta,\zeta} = \mathit{true}$ for at least one consistent $\Delta \in \Delta(r, S, \zeta)$.

Instead of introducing modal operators $[\,]$ and $\langle\,\rangle$ for a non-deterministic parallel ASM rule r, we use the modal expression $[X]\varphi$ for an update set yielded by a possibly non-deterministic rule. The modal expressions $[r]\varphi$ and $\langle r\rangle\varphi$ in the logic for ASMs can be treated as the shortcuts for the following formulae in our logic:

$$[r]\varphi \equiv \forall X(\mathrm{upd}(r, X) \rightarrow [X]\varphi). \tag{5}$$

$$\langle r\rangle\varphi \equiv \exists X(\mathrm{upd}(r, X) \wedge [X]\varphi). \tag{6}$$

Lemma 2. *The following axioms and rules used in the logic for ASMs are derivable in $\mathcal{L}$, where the rule r in Axioms (c) and (d) is assumed to be defined and deterministic: (a)* $([r](\varphi \to \psi) \to [r]\varphi) \to [r]\psi$*; (b)* $\varphi \to [r]\varphi$*; (c)* $\neg wcon(r) \to [r]\varphi$*; (d)* $[r]\varphi \leftrightarrow \neg[r]\neg\varphi$*.*

Proof. We prove each property in the following.

- (a): By Equation 5, we have that $[r](\varphi \to \psi) \wedge [r]\varphi \equiv \forall X(\mathrm{upd}(r, X) \to [X](\varphi \to \psi)) \wedge \forall X(\mathrm{upd}(r, X) \to [X]\varphi)$. By the axioms from classical logic, this is in turn equivalent to $\forall X(\mathrm{upd}(r, X) \to ([X](\varphi \to \psi) \wedge [X]\varphi))$. Then by Axiom **M1** and axioms from the classical logic, we get $\forall X(\mathrm{upd}(r, X) \to ([X](\varphi \to \psi) \wedge [X]\varphi)) \to \forall X(\mathrm{upd}(r, X) \to [X]\psi)$. Therefore, $([r](\varphi \to \psi) \to [r]\varphi) \to [r]\psi$ is derivable.
- (b): By Rule **M2**, we have that $\varphi \to [X_i]\varphi$. Since X is free in $\varphi \to [X]\varphi$, this holds for every possible valuation of X. Thus using Rule **UG** (applied to the variable X of the first predicate sort) and the axioms from classical logic, we can clearly derive $\varphi \to \forall X(\mathrm{upd}(r, X) \to [X]\varphi)$.
- (c): By Equation 3, we have $\neg\mathrm{wcon}(r) \leftrightarrow \neg\exists X(\mathrm{con}(r, X)$. In turn, by Equation 2, we get $\neg\mathrm{wcon}(r) \leftrightarrow \neg\exists X(\mathrm{upd}(r, X) \wedge \mathrm{conUSet}(X))$. Since a rule r in the logic for ASMs is deterministic, we get $\neg\mathrm{wcon}(r) \leftrightarrow \neg\mathrm{conUSet}(X)$. By Axiom **M4**, we get $\neg\mathrm{wcon}(r) \to [r]\varphi$.
- (d): By Equation 5, we have $\neg[r]\neg\varphi \equiv \exists X(\mathrm{upd}(r, X) \wedge \neg[X]\neg\varphi)$. By applying Axiom **M5** to $\neg[X]\neg\varphi$, we get $\neg[r]\neg\varphi \equiv \exists X(\mathrm{upd}(r, X) \wedge [X]\varphi)$. When the rule r is deterministic, the interpretation of $\forall X(\mathrm{upd}(r, X) \to [X]\varphi)$ coincides with he interpretation of $\exists X(\mathrm{upd}(r, X) \wedge [X]\varphi)$ and therefore $[r]\varphi \leftrightarrow \neg[r]\neg\varphi$.

Note that the formula $\mathrm{Con}(R)$ in **Axiom 5** in [13] (i.e., in $\neg\mathrm{Con}(R) \to [R]\varphi$) corresponds to the weak version of consistency (i.e., $\mathrm{wcon}(r)$) in the theory of $\mathcal{L}$.

Lemma 3. *The following properties are derivable in $\mathcal{L}$: (e)* $\mathrm{con}(r, X) \wedge [X]f(x) = y \to X(f, x, y) \vee (\forall z(\neg X(f, x, z)) \wedge f(x) = y)$*; (f)* $\mathrm{con}(r, X) \wedge [X]\varphi \to \neg[X]\neg\varphi$*; (g)* $[X]\exists x(\varphi) \to \exists x([X]\varphi)$*; (h)* $[X]\varphi_1 \wedge [X]\varphi_2 \to [X](\varphi_1 \wedge \varphi_2)$*.*

Proof. (e) is derivable by applying Axioms **A1** and **A2**. (f) is a straightforward result of Axiom **M5**. (g) can be derived by applying Axioms **M5** and **M6**. Regarding (h), it is derivable by using Axioms **M1**-**M3**.

Lemma 4. *For terms and variables of the appropriate types, the following properties in [9] are derivable in $\mathcal{L}$.*

- $x = t \to (y = s \leftrightarrow [f(t) := s]f(x) = y)$
- $x \neq t \to (y = f(x) \leftrightarrow [f(t) := s]f(x) = y)$

Following the approach of defining the predicate joinable in [13], we define the predicate joinable over two non-deterministic parallel ASMs rules. As we consider non-deterministic parallel ASMs rules, the predicate $\mathrm{joinable}(r_1, r_2)$ means that there exists a pair of update sets without conflicting updates, which are yielded

by rules r_1 and r_2, respectively. Then, based on the use of predicate joinable, the properties in Lemma 5 are all derivable.

$$\begin{aligned}\text{joinable}(r_1, r_2) \equiv & \exists X_1 X_2(\text{upd}(r_1, X_1) \wedge \text{upd}(r_2, X_2) \wedge \\ & \bigwedge_{f \in \mathcal{F}_{dyn}} \forall xyz(X_1(f, x, y) \wedge X_2(f, x, z) \rightarrow y = z))\end{aligned} \tag{7}$$

Lemma 5. *The following properties for weak consistency are derivable in $\mathcal{L}$.*

(i) $\text{wcon}(f(t) := s)$ **(j)** $\text{wcon}(f(t) := \mathtt{s})$ **(k)** $\text{wcon}(f(\mathtt{t}) := \mathtt{s})$
(j) $\text{wcon}(\textbf{\textit{if}}\,\varphi\,\textbf{\textit{then}}\,r\,\textbf{\textit{endif}}) \leftrightarrow \neg\varphi \vee (\varphi \wedge \text{wcon}(r))$
(l) $\text{wcon}(\textbf{\textit{forall}}\,x\,\textbf{\textit{with}}\,\varphi\,\textbf{\textit{do}}\,r\,\textbf{\textit{enddo}}) \leftrightarrow$
$\forall x(\varphi \rightarrow \text{wcon}(r) \wedge \forall y(\varphi[y/x] \rightarrow \mathit{joinable}(r, r[y/x])))$
(m) $\text{wcon}(\textbf{\textit{par}}\,r_1\,r_2\,\textbf{\textit{endpar}}) \leftrightarrow \text{wcon}(r_1) \wedge \text{wcon}(r_2) \wedge \mathit{joinable}(r_1, r_2)$
(n) $\text{wcon}(\textbf{\textit{choose}}\,x\,\textbf{\textit{with}}\,\varphi\,\textbf{\textit{do}}\,r\,\textbf{\textit{enddo}}) \leftrightarrow \exists x(\varphi \wedge \text{wcon}(r))$
(o) $\text{wcon}(\textbf{\textit{choose}}\,\mathtt{x}\,\textbf{\textit{with}}\,\varphi\,\textbf{\textit{do}}\,r\,\textbf{\textit{enddo}}) \leftrightarrow \exists \mathtt{x}(\varphi \wedge \text{wcon}(r))$
(p) $\text{wcon}(\textbf{\textit{seq}}\,r_1\,r_2\,\textbf{\textit{endseq}}) \leftrightarrow \exists X(\text{con}(r_1, X) \wedge [X]\text{wcon}(r_2))$

We omit the proof of the previous lemma as well as the proof of the remaining lemmas in this section, since they are lengthy but relatively easy exercises.

Lemma 6. *The following properties for the formula $[r]\varphi$ are derivable in $\mathcal{L}$.*

(q) $[\textbf{\textit{if}}, \varphi, \textbf{\textit{then}}, r, \textbf{\textit{endif}}]\psi \leftrightarrow (\varphi \wedge [r]\psi) \vee (\neg\varphi \wedge \psi)$
(r) $[\textbf{\textit{choose}}\,x\,\textbf{\textit{with}}\,\varphi\,\textbf{\textit{do}}\,r\,\textbf{\textit{enddo}}]\psi \leftrightarrow \forall x(\varphi \rightarrow [r]\psi)$
(s) $[\textbf{\textit{choose}}\,\mathtt{x}\,\textbf{\textit{with}}\,\varphi\,\textbf{\textit{do}}\,r\,\textbf{\textit{enddo}}]\psi \leftrightarrow \forall \mathtt{x}(\varphi \rightarrow [r]\psi)$

Lemma 7 states that a parallel composition is commutative and associative while a sequential composition is associative.

Lemma 7. *The following properties are derivable in $\mathcal{L}$.*

(t) $\textbf{\textit{par}}\ r_1\ r_2\ \textbf{\textit{endpar}} \equiv \textbf{\textit{par}}\ r_2\ r_1\ \textbf{\textit{endpar}}$
(u) $\textbf{\textit{par}}\ (\textbf{\textit{par}}\ r_1\ r_2\ \textbf{\textit{endpar}})\ r_3\ \textbf{\textit{endpar}} \equiv \textbf{\textit{par}}\ r_1\ (\textbf{\textit{par}}\ r_2\ r_3\ \textbf{\textit{endpar}})\ \textbf{\textit{endpar}}$
(v) $\textbf{\textit{seq}}\ (\textbf{\textit{seq}}\ r_1\ r_2\ \textbf{\textit{endseq}})\ r_3\ \textbf{\textit{endseq}} \equiv \textbf{\textit{seq}}\ r_1\ (\textbf{\textit{seq}}\ r_2\ r_3\ \textbf{\textit{endseq}})\ \textbf{\textit{endseq}}$

Lemma 8. *The extensionality axiom for transition rules in the logic for ASMs is derivable in $\mathcal{L}$: $r_1 \equiv r_2 \rightarrow ([r_1]\varphi \leftrightarrow [r_2]\varphi)$.*

7 Completeness

We can prove the completeness of $\mathcal{L}$ by using a similar strategy to that used in [13]. That is, we can show that $\mathcal{L}$ is a definitional extension of a complete logic. However, the logic for hierarchical ASMs in [13] is a definitional extension of first-order logic. In the case of the logic $\mathcal{L}$, the proof is more complicated since we have to deal with set membership predicates and corresponding predicate

sorts. The key idea is to show instead that $\mathcal{L}$ is a *definitional extension* of first-order logic extended with two membership predicates with respect to finite sets, which in turns constitutes itself a complete logic.

In the remaining of this section, we will use $\mathcal{L}^{\in}$ to denote the logic obtained by restricting the formulae of $\mathcal{L}$ to those produced by the following grammar:

$$\varphi, \psi ::= s = t \mid s_a = t_a \mid \neg\varphi \mid \varphi \wedge \psi \mid \forall x(\varphi) \mid \forall \mathtt{x}(\varphi) \mid \forall x^1(\varphi) \mid \forall x^2(\varphi) \mid \in^1(x^1, f, t_0, s_0) \mid \in^2(x^2, f, t_0, s_0, s).$$

Let us define the theory of $\mathcal{L}^{\in}$ as the theory obtained by taking the union of a sound and complete axiomatisation of first-order logic and the sound and complete axiomatisation of the properties of finite sets introduced in [1]. Clearly, such theory of $\mathcal{L}^{\in}$ is a conservative extension of the first-order theory, in the sense that if Φ is a set of pure first-order formulae and φ is a pure first-order formula (not containing subformulae of the form $\in^n(x^n, t_1, \ldots, t_n)$) and $\Phi \vdash \varphi$ holds in the theory of $\mathcal{L}^{\in}$, then there already exists a derivation using the axiomatisation for first-order logic. Indeed, due to the soundness of the axioms and rules in the theory of $\mathcal{L}^{\in}$, we obtain $\Phi \models \varphi$, which is a pure statement about models for first-order logic. Thus the known completeness for first-order logic gives $\Phi \vdash \varphi$ in an axiomatisation for first-order logic, hence the claimed conservativism of the extension. Since then the theory of $\mathcal{L}^{\in}$ proves no new theorems about first-order logic, all the new theorems belong to the theory of properties of finite sets and thus can be derived by using the axiomatisation in [1] (which also form part of the axiomatisation of $\mathcal{L}^{\in}$), we get the following key result.

Theorem 2. *Let φ be a formula and Φ be a set of formulae in the language of $\mathcal{L}^{\in}$ (all of the same vocabulary). If $\Phi \models \varphi$, then $\Phi \vdash \varphi$.*

Finally, we need to show that all the formulae in $\mathcal{L}$ which are not formulae of $\mathcal{L}^{\in}$ can be translated into formulae of $\mathcal{L}^{\in}$ based on derivable equivalences in the theory of $\mathcal{L}$. First, we reduce the general atomic formulae in $\mathcal{L}$ to atomic formulae of the form $x = y$, $\mathtt{x} = \mathtt{y}$, $f(x) = y$, $f(x) = \mathtt{y}$, $f(\mathtt{x}) = \mathtt{y}$, $\in^1(x^1, f, x, y)$, $\in^1(x^1, f, x, \mathtt{y})$, $\in^1(x^1, f, \mathtt{x}, \mathtt{y})$, $\in^2(x^2, f, x, y, z)$, $\in^2(x^2, f, x, \mathtt{y}, z)$ and $\in^2(x^2, f, \mathtt{x}, \mathtt{y}, z)$. Let t, s and s' denote point terms and let t_a and s_a denote algorithmic terms. This can be done by using the following equivalences.

$$\begin{aligned}
s = t &\leftrightarrow \exists x(s = x \wedge x = t)\\
s_a = t_a &\leftrightarrow \exists \mathtt{x}(s_a = \mathtt{x} \wedge \mathtt{x} = t_a)\\
f(s) = y &\leftrightarrow \exists x(s = x \wedge f(x) = y)\\
f(s) = \mathtt{y} &\leftrightarrow \exists x(s = x \wedge f(x) = \mathtt{y})\\
f(s_a) = \mathtt{y} &\leftrightarrow \exists \mathtt{x}(s_a = \mathtt{x} \wedge f(\mathtt{x}) = \mathtt{y})\\
\in^1(x^1, f, t, s) &\leftrightarrow \exists xy(t = x \wedge s = y \wedge \in^1(x^1, f, x, y))\\
\in^1(x^1, f, t, s_a) &\leftrightarrow \exists x\mathtt{y}(t = x \wedge s_a = \mathtt{y} \wedge \in^1(x^1, f, x, \mathtt{y}))\\
\in^1(x^1, f, t_a, s_a) &\leftrightarrow \exists \mathtt{xy}(t_a = \mathtt{x} \wedge s_a = \mathtt{y} \wedge \in^1(x^1, f, \mathtt{x}, \mathtt{y}))\\
\in^2(x^2, f, t, s, s') &\leftrightarrow \exists xyz(t = x \wedge s = y \wedge s' = z \wedge \in^2(x^2, f, x, y, z))
\end{aligned}$$

$$\in^2(x^2, f, t, s_a, s') \leftrightarrow \exists x\mathtt{y}z(t = x \wedge s_a = \mathtt{y} \wedge s' = z \wedge \in^2(x^2, f, x, \mathtt{y}, z))$$
$$\in^2(x^2, f, t_a, s_a, s') \leftrightarrow \exists \mathtt{xy}z(t_a = \mathtt{x} \wedge s_a = \mathtt{y} \wedge s' = z \wedge \in^2(x^2, f, \mathtt{x}, \mathtt{y}, z))$$

The translation of modal formulae into $\mathcal{L}^{\in}$ distributes over negation, Boolean connectives and quantifiers. We eliminate atomic formulae of the form $\mathrm{upd}(r, x^1)$ using Axioms **U1**-**U7**, and the modal operator in formulae of the form $[x^1]\varphi$, where φ is already translated to $\mathcal{L}^{\in}$, using the following derivable equivalences.

$[x^1]x = y \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow x = y)$; $\quad [x^1]\mathtt{x} = \mathtt{y} \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \mathtt{x} = \mathtt{y})$;
$[x^1]f(x) = y \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^1(x^1, f, x, y) \vee (\forall z(\neg \in^1(x^1, f, x, z)) \wedge f(x) = y))$
$[x^1]f(x) = \mathtt{y} \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^1(x^1, f, x, \mathtt{y}) \vee (\forall \mathtt{z}(\neg \in^1(x^1, f, x, \mathtt{z})) \wedge f(x) = \mathtt{y}))$
$[x^1]f(\mathtt{x}) = \mathtt{y} \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^1(x^1, f, \mathtt{x}, \mathtt{y}) \vee (\forall \mathtt{z}(\neg \in^1(x^1, f, \mathtt{x}, \mathtt{z})) \wedge f(\mathtt{x}) = \mathtt{y}))$;
$[x^1]\in^1(x^1, f, x, y) \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^1(x^1, f, x, y))$;
$[x^1]\in^1(x^1, f, x, \mathtt{y}) \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^1(x^1, f, x, \mathtt{y}))$;
$[x^1]\in^1(x^1, f, \mathtt{x}, \mathtt{y}) \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^1(x^1, f, \mathtt{x}, \mathtt{y}))$;
$[x^1]\in^2(x^2, f, x, y, z) \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^2(x^2, f, x, y, z))$;
$[x^1]\in^2(x^2, f, x, \mathtt{y}, z) \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^2(x^2, f, x, \mathtt{y}, z))$;
$[x^1]\in^2(x^2, f, \mathtt{x}, \mathtt{y}, z) \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \in^2(x^2, f, \mathtt{x}, \mathtt{y}, z))$;
$[x^1]\neg\varphi \leftrightarrow (\mathrm{conUSet}(x^1) \rightarrow \neg[x^1]\varphi)$; $\quad [x^1](\varphi \wedge \psi) \leftrightarrow ([x^1]\varphi \wedge [x^1]\psi)$;
$[x^1]\forall x(\varphi) \leftrightarrow \forall x([x^1]\varphi)$; $\quad [x^1]\forall \mathtt{x}(\varphi) \leftrightarrow \forall \mathtt{x}([x^1]\varphi)$;
$[x^1]\forall y^1(\varphi) \leftrightarrow \forall y^1([x^1]\varphi)$; $\quad [x^1]\forall x^2(\varphi) \leftrightarrow \forall x^2([x^1]\varphi)$.

Our main technical result then follows from Theorem 2 and the fact that the described translation from formulae φ of $\mathcal{L}$ to formulae φ' of $\mathcal{L}^{\in}$ satisfies the properties required for $\mathcal{L}$ to be a definitional extension of $\mathcal{L}^{\in}$, i.e., (a) $\varphi \leftrightarrow \varphi'$ is derivable in $\mathcal{L}$ and (b) φ' is derivable in $\mathcal{L}^{\in}$ whenever φ is derivable $\mathcal{L}$.

Theorem 3. *Let φ be a formula and Φ a set of formulae in the language of $\mathcal{L}$ (all of the same vocabulary). If $\Phi \models \varphi$, then $\Phi \vdash \varphi$.*

8 Conclusion

Non-deterministic transitions manifest themselves as a difficult task in the logical formalisation for ASMs. Indeed, Nanchen and Stärk analysed potential problems to several approaches they tried by taking non-determinism into consideration and concluded [13]:

> Unfortunately, the formalisation of consistency cannot be applied directly to non-deterministic ASMs. The formula $\mathrm{Con}(r)$ (as defined in Sect. 8.1.2 of [5]) expresses the property that the *union of all possible* update sets of (an ASM rule) r in a given state is consistent. This is clearly not what is meant by consistency. Therefore, in a logic for ASMs with **choose** one had to add $\mathrm{Con}(r)$ as an atomic formula to the logic.

However, we observe that this conclusion is not necessarily true, as finite update sets can be made explicit in the formulae of a logic to capture non-deterministic transitions. In doing so, the formalisation of consistency defined in

[13] can still be applied to such an explicitly specified update set Δ yielded by a rule r in the form of the formula $\mathrm{con}(r, \Delta)$ as discussed in Subsection 5.1. We thus solve this problem by the addition of the modal operator $[\Delta]$ for an update set generated by a non-deterministic parallel ASM rule. The approach works well, because in the parallel ASMs the number of possible parallel branches, although unbounded, is still finite. Therefore the update sets produced by these machines are restricted to be finite as well. This is implicitly assumed in the parallel ASM thesis of Blass and Gurevich[3,4] and it is made explicit in the new parallel ASM thesis that we propose in [6].

The proof systems that we develop in this work for the proposed logic for non-deterministic parallel ASMs, extends the proof system developed in [13] in two different ways. First, an ASM rule may be associated with a set of different update sets. Applying different update sets may lead to a set of different successor states to the current state. As the logic for non-deterministic parallel ASMs includes formulae denoting explicit update sets and variables that are bounded to update sets, our proof system allows us to reason about the interpretation of a formula over all successor states or over some successor state after applying an ASM rule over the current state. Secondly, in addition to capturing the consistency of an update set yielded by an ASM rule, our proof system also develops two notions of consistency (weak and strong consistency) w.r.t. a given rule. When the rule is deterministic, these two notions coincide.

We plan as future work to embed our one-step logic into a complex dynamic logic and demonstrate how desirable properties of ASM runs can be formalised in such a logic. Of course, there is no chance of obtaining a complete proof theory for full ASM runs, but there is clearly many potential practical benefits from the perspective of the ASM method for systems development [5].

References

1. Ågotnes, T., Walicki, M.: Complete axiomatisations of properties of finite sets. Logic Journal of the IGPL 16(3), 293–313 (2008)
2. Alur, R.: Principles of Cyber-Physical Systems. MIT Press (2015)
3. Blass, A., Gurevich, Y.: Abstract state machines capture parallel algorithms. ACM Trans. on Comp. Logic 4(4), 578–651 (October 2003)
4. Blass, A., Gurevich, Y.: Abstract state machines capture parallel algorithms: Correction and extension. ACM Trans. on Comp. Logic 9(3), 1–32 (06 2008)
5. Börger, E., Stärk, R.F.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag New York, Inc. (2003)
6. Ferrarotti, F., Schewe, K., Tec, L., Wang, Q.: A new thesis concerning synchronised parallel computing - simplified parallel ASM thesis. CoRR abs/1504.06203 (2015), `http://arxiv.org/abs/1504.06203`
7. Floyd, R.W.: Nondeterministic algorithms. J. ACM 14(4), 636–644 (Oct 1967), `http://doi.acm.org/10.1145/321420.321422`
8. Grädel, E., Gurevich, Y.: Metafinite model theory. Information and Computation 140(1), 26–81 (1998)
9. Groenboom, R., Renardel de Lavalette, G.: A formalization of evolving algebras. In: Proceedings of Accolade95. Dutch Research School in Logic (1995)

10. Huggins, J.K., Wallace, C.: An abstract state machine primer. Tech. Rep. 02-04, Computer Science Department, Michigan Technological University (2002)
11. Hughes, G., Cresswell, M.: A new introduction to modal logic. Burns & Oates (1996)
12. Kruskal, J.B.: On the shortest spanning subtree of a graph and the travelling salesman problem. Proc. Amer. Math. Soc. 2, 48–50 (1956)
13. Stärk, R., Nanchen, S.: A logic for abstract state machines. Journal of Universal Computer Science 7(11) (2001)
14. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc., New York, NY, USA (1995)