Journal of Universal Computer Science, vol. 7, no. 11 (2001), 980-1005
submitted: 7/6/01, accepted: 1/10/01, appeared: 28/11/01

A Logic for Abstract State Machines

Robert F. Stärk
(Computer Science Department, ETH Zürich, Switzerland
staerk@inf.ethz.ch)

Stanislas Nanchen
(Computer Science Department, ETH Zürich, Switzerland
nanchen@inf.ethz.ch)

Abstract: We introduce a logic for non distributed, deterministic Abstract State Machines with parallel function updates. Unlike other logics for ASMs which are based on dynamic logic, our logic is based on an atomic predicate for function updates and on a definedness predicate for the termination of the evaluation of transition rules. We do not assume that the transition rules of ASMs are in normal form, for example, that they concern distinct cases. Instead we allow structuring concepts of ASM rules including sequential composition and possibly recursive submachine calls. We show that several axioms that have been proposed for reasoning about ASMs are derivable in our system. We provide also an extension of the logic with explicit step information which allows to eliminate modal operators in certain cases. The main technical result is that the logic is complete for hierarchical (non-recursive) ASMs. We show that, for hierarchical ASMs, the logic is a definitional extension of first-order predicate logic.

Key Words: Abstract State Machines, dynamic logic, modal logic, logical foundations of specification languages.

Category: D.2.4, F.3.1, F.4.1

1 Introduction

Gurevich's Abstract State Machines (ASMs) are widely used for the specification of software, hardware, algorithms, and the semantics of programming languages [Gurevich 1993, Börger and Huggins 1998]. Most logics that have been proposed for ASMs are based on variants of dynamic logic. There are, however, fundamental differences between the imperative programs of dynamic logic and ASMs. In dynamic logic, states are represented with variables. In ASMs, states are represented with dynamic functions. The fundamental program constructs in dynamic logic are non-deterministic iteration (star operator) and sequential composition. The basic transition rules of ASMs consist of guarded parallel function updates. Since parallel function updates may conflict, a logic for ASMs must have a clear notion of consistency of transition rules. Therefore, rather than to encode ASMs into imperative programs of dynamic logic or extend the axioms and rules of dynamic logic to ASMs, we propose new axioms and rules which are directly based on an update predicate for transition rules.

Since ASMs are special instances of transition systems, our logic contains a modal operator, too [see van Benthem and Bergstra 1995]. We do, however, not stick to modal logic, since in some situations it can be more convenient and even more economic to use functions with an additional argument that specifies the nth state of the run of an ASM. This has been shown to be useful for proving the equivalence of ASMs, e.g. for compiler correctness proofs from Prolog to the WAM [Schmitt 1994] or from Java to the JVM [Stärk et al. 2001].

What comes closest to our system is known as dynamic logic with array assignments [Harel 1983, Harel et al. 2000]. The substitution principle which is used in its axiomatization is derivable in our system [see Lemma 18]. The dynamic logic with array assignments, however, is not concerned with parallel execution of assignments and therefore does not need a notion of consistency.

[Groenboom and Renardel de Lavalette 1995] introduce the *Formal Language for Evolving Algebras* (FLEA), a system for formal reasoning about abstract state machines. Their system is in the tradition of dynamic logic and contains for every rule R a modal operator $[R]$. The intended meaning of $[R]\varphi$ is that φ holds always after the execution or R. The logic of their formal language contains besides *true* and *false* a third truth-value which stands for *undefined*. Although they consider parallel composition of transition rules, they have no formal notion of consistency in their system. We adopt their modal operator such that the basic axioms of their system are derivable in our logic [see Lemma 13]. We use, however, the two-valued logic of the classical predicate logic.

[Schönegge 1995] extends the dynamic logic of the KIV system (Karlsruhe Interactive Verifier) to turn it into a tool for reasoning about abstract state machines. The transition rules of ASMs are considered as imperative programs of dynamic logic. While-programs of dynamic logic are used as an interpreter for abstract state machines; loop-programs are used to apply an ASM a finite number of times. Schönegge's rules for simultaneous function updates and parallel composition of transition rules are derivable in our system [see Lemma 16]. His sequent calculus is mainly designed as a practical extension of the KIV system and not as a foundation for reasoning about transition rules and ASMs.

[Schellhorn and Ahrendt 1997] simulate abstract state machines in the KIV system by formalizing dynamic functions as association lists, serializing parallel updates and transforming transition rules into flat imperative program of the underlying dynamic logic. [Schellhorn 1999] is able to fully mechanize a correctness proof of the Prolog to WAM compilation in his dissertation. He argues that the inconsistency of an ASM (clash in simultaneous updates) can only be detected when the ASM is in normal form and that the transformation of the ASM into normal form by a pre-processor is more efficient than a formalization of consistency in terms of logical axioms as we do it in our system. We do think that a suitable theorem prover can automatically process and simplify our consistency conditions [see Lemma 11 and Table 5] without problems.

[Poetzsch-Heffter 1994] introduces a basic logic for a class of ASMs consisting of simultaneous updates of 0-ary functions (dynamic constants) and if-then-else rules only. His basic axiom states that the truth of the *weakest backwards transformer* of a formula implies the truth of the formula in the next state. He then derives partial correctness logics for a class of simple imperative programming languages by specifying their semantics with ASMs of his restricted class. His basic axiom is derivable in our system [see Lemma 19].

[Gargantini and Riccobene 2000] show how the PVS theorem prover can provide tool support for ASMs. They show how ASMs can be encoded in PVS (and hence in the underlying formal system which is Church's simple theory of types). Functions are encoded as PVS functions and an interpreter for ASMs is implemented in PVS. The parallel rule application is serialized. The abstraction level of the abstract states is preserved by assuming properties of certain static functions rather than by implementing (explicitly defining) them in PVS. They do not provide a technique for proving consistency of ASMs as we do in our logic. We think that for verification of large ASMs a theorem prover like PVS should directly provide support for transition rules such that the overhead introduced by the encoding is avoided [see Shankar 2000].

The main technical result of this article is that the logic is complete for so-called hierarchical ASMs (which do not contain recursive rule definitions). The reader may wonder why a logic for a computationally universal mechanism like hierarchical ASMs can be complete at all? The answer is that the logic is not able to talk about ASM runs. Instead, the logic is talking about single steps of an ASM, only. The logic is complete for statements about single steps of an ASM like invariants of rules, consistency conditions for rules, or step-by-step equivalence of rules. Moreover, the completeness theorem is for the uninterpreted logic where the static functions do not have a fixed standard interpretation. If we allow recursive rule definitions, then the logic cannot be complete, since iteration and while-loops can be recursively defined and therefore it is possible to talk about the outcome of ASMs runs.

The plan of this paper is as follows. In [Section 2] we give a short overview on ASMs with sequential composition and (possibly recursive) rule definitions. After some considerations on formalizing the consistency of transition rules in [Section 3], we introduce in [Section 4] the basic axioms and rules of our logic. In [Section 5] we argue why we restrict the logic to deterministic ASMs only. In [Section 6] we prove that the logic is complete for hierarchical (non-recursive) ASMs. Finally, in [Section 7], we consider an extension of the basic system where the dynamic functions are augmented by an additional argument which indicates the value of the dynamic function in the nth state. A preliminary version of the paper appeared in the proceedings of CSL '01 [Stärk and Nanchen 2001].

2 ASM Rules and Update Sets

The notion of an *abstract state* is the classical notion of a mathematical structure $\mathfrak{A}$ for a vocabulary Σ consisting of a a non-empty set $|\mathfrak{A}|$ and of functions $f^{\mathfrak{A}}$ from $|\mathfrak{A}|^n$ to $|\mathfrak{A}|$ for each n-ary function name f of Σ. The terms s, t and the first-order formulas φ, ψ of the vocabulary Σ are interpreted as usual in the structure $\mathfrak{A}$ with respect to a variable assignment ζ. The value of a term t in the structure $\mathfrak{A}$ under ζ is denoted by $[\![t]\!]^{\mathfrak{A}}_{\zeta}$; the truth value of a formula φ in $\mathfrak{A}$ under ζ is denoted by $[\![\varphi]\!]^{\mathfrak{A}}_{\zeta}$ [see Table 1]. The variable assignment which is obtained from ζ by assigning the element a to the variable x is denoted by $\zeta\frac{a}{x}$. By $\mathrm{FV}(t)$ and $\mathrm{FV}(\varphi)$ we denote the set of free variables of t and φ.

Abstract State Machines (ASMs) are systems of finitely many *transition rules* which update some of the functions of the vocabulary Σ in a given state at some arguments. The functions of the vocabulary Σ are divided into *static* functions which cannot be updated by an ASM and *dynamic* ones which typically do change as a consequence of updates by the ASM. The transition rules R, S of an ASM are syntactic expressions generated as follows (the function arguments can be read as vectors):

1. *Skip Rule:* **skip**
   Meaning: Do nothing.
2. *Update Rule:* $f(t) := s$
   Syntactic condition: f is a dynamic function name of Σ
   Meaning: In the next state, the value of f at the argument t is updated to s.
3. *Block Rule:* $R\ S$
   Meaning: R and S are executed in parallel.
4. *Conditional Rule:* **if** φ **then** R **else** S
   Meaning: If φ is true, then execute R, otherwise execute S.
5. *Let Rule:* **let** $x = t$ **in** R
   Meaning: Assign the value of t to x and execute R.
6. *Forall Rule:* **forall** x **with** φ **do** R
   Meaning: Execute R in parallel for each x satisfying φ.
7. *Sequence Rule:* $R\,;S$
   Meaning: R and S are executed sequentially, first R and then S.
8. *Try Rule:* **try** R **else** S
   Meaning: If R is consistent, then execute R, otherwise execute S.
9. *Call Rule:* $\rho(t)$
   Meaning: Call ρ with parameter t.

A *rule definition* for a rule name ρ is an expression $\rho(x) = R$, where R is a transition rule in which there are no free occurrences of variables except of x. Rules are called *by name*. This means that in a call $\rho(t)$ the variable x is replaced in the body R of the rule by the parameter t. The parameter t is not evaluated in the state where the rule is called but only later when it is used in the body (maybe in different states due to sequential compositions). Call-by-value evaluation of rule calls can be simulated as follows:

$$\rho(y) = \textbf{let } x = y \textbf{ in } R$$

Then upon calling $\rho(t)$ the parameter t is evaluated in the same state.

The syntax of rules is related to the *named parameterized ASM rules* in [Börger and Schmid 2000] which include also recursive definitions of transition rules. Recursive rule definitions in combination with sequential compositions of transition rules are maybe too powerful and not in the spirit of the basic ASM concept [Gurevich 1993]. Nevertheless we include them and solve the technical problems that arise in the logic with an explicit definedness predicate.

Definition 1 (ASM). An *abstract state machine* consists of a vocabulary Σ, an initial state $\mathfrak{A}$ for Σ, a rule definition for each rule name, and a distinguished rule name of arity zero called the *main rule name* of the machine.

The semantics of transition rules is given by sets of updates. Since due to the parallelism (in the Block and the Forall rules), a transition rule may prescribe to update the same function at the same arguments several times, such updates are required to be consistent. The concept of consistent update sets is made more precise by the following definitions.

Definition 2 (Update). An *update* for $\mathfrak{A}$ is a triple $\langle f, a, b\rangle$, where f is a dynamic function name, and a and b are elements of $|\mathfrak{A}|$.

The meaning of the update is that the interpretation of the function f in $\mathfrak{A}$ has to be changed at the argument a to the value b. The pair of the first two components of an update is called a *location*. An update specifies how the function table of a dynamic function has to be updated at the corresponding location. An *update set* is a set of updates.

Definition 3 (Consistent update set). An update set U is called *consistent*, if it satisfies the following property: If $\langle f, a, b\rangle \in U$ and $\langle f, a, c\rangle \in U$, then $b = c$.

This means that a consistent update set contains for each function and each argument at most one value. If an update set U is consistent, it can be fired in a given state. The result is a new state in which the interpretations of dynamic function names are changed according to U and nothing else changes.

Definition 4 (Firing of updates). The result of firing a consistent update set U in a state $\mathfrak{A}$ is a new state $U(\mathfrak{A})$ with the same universe as $\mathfrak{A}$ satisfying the following two conditions for the interpretations of function names f of Σ:

$$
\begin{aligned}
[\![s = t]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if } [\![s]\!]^{\mathfrak{A}}_{\zeta} = [\![t]\!]^{\mathfrak{A}}_{\zeta};\\ \mathit{false}, & \text{otherwise.}\end{cases}\\
[\![\neg\varphi]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{false};\\ \mathit{false}, & \text{otherwise.}\end{cases}\\
[\![\varphi \land \psi]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true} \text{ and } [\![\psi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true};\\ \mathit{false}, & \text{otherwise.}\end{cases}\\
[\![\varphi \lor \psi]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true} \text{ or } [\![\psi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true};\\ \mathit{false}, & \text{otherwise.}\end{cases}\\
[\![\varphi \to \psi]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{false} \text{ or } [\![\psi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true};\\ \mathit{false}, & \text{otherwise.}\end{cases}\\
[\![\forall x\, \varphi]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta\frac{a}{x}} = \mathit{true} \text{ for all } a \in |\mathfrak{A}|;\\ \mathit{false}, & \text{otherwise.}\end{cases}\\
[\![\exists x\, \varphi]\!]^{\mathfrak{A}}_{\zeta} &:= \begin{cases} \mathit{true}, & \text{if there exists an } a \in |\mathfrak{A}| \text{ with } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta\frac{a}{x}} = \mathit{true};\\ \mathit{false}, & \text{otherwise.}\end{cases}
\end{aligned}
$$

Table 1: The semantics of formulas.

1. If $\langle f, a, b\rangle \in U$, then $f^{U(\mathfrak{A})}(a) = b$.
2. If there is no b with $\langle f, a, b\rangle \in U$, then $f^{U(\mathfrak{A})}(a) = f^{\mathfrak{A}}(a)$.

Since U is consistent, the state $U(\mathfrak{A})$ is determined in a unique way. Notice that only those locations can have a new value in state $U(\mathfrak{A})$ with respect to state $\mathfrak{A}$ for which there is an update in U.

The composition '$U\,;V$' of two update sets U and V is defined such that the following equation is true for any state $\mathfrak{A}$:

$$(U\,;V)(\mathfrak{A}) = V(U(\mathfrak{A}))$$

The equation says that applying the update set '$U\,;V$' to state $\mathfrak{A}$ should be the same as first applying U and then V. Hence, '$U\,;V$' is the set of updates obtained from U by adding the updates of V and overwriting updates in U which are redefined in V. If U and V are consistent, then $U\,;V$ is consistent, too.

Definition 5 (Composition of update sets). The composition of two update sets U and V is defined by $U\,;V := \{\langle f, a, b\rangle \in U \mid \neg\exists c\, \langle f, a, c\rangle \in V\} \cup V$.

In a given state, a transition rule of an ASM produces for each variable assignment an update set. Since the rule can contain recursive calls to other rules, it is also possible that the rule does not terminate and has no semantics at all. Therefore, the semantics of ASM rules is given by an inductive definition

$\dfrac{}{\llbracket \mathbf{skip} \rrbracket_\zeta^{\mathfrak{A}} \triangleright \emptyset}$		(skip)
$\dfrac{}{\llbracket f(s) := t \rrbracket_\zeta^{\mathfrak{A}} \triangleright \{\langle f, a, b\rangle\}}$	if $a = \llbracket s \rrbracket_\zeta^{\mathfrak{A}}$ and $b = \llbracket t \rrbracket_\zeta^{\mathfrak{A}}$	(upd)
$\dfrac{\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U \qquad \llbracket S \rrbracket_\zeta^{\mathfrak{A}} \triangleright V}{\llbracket R\ S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U \cup V}$		(par)
$\dfrac{\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}{\llbracket \mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}$	if $\llbracket \varphi \rrbracket_\zeta^{\mathfrak{A}} = \mathit{true}$	(if_1)
$\dfrac{\llbracket S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}{\llbracket \mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}$	if $\llbracket \varphi \rrbracket_\zeta^{\mathfrak{A}} = \mathit{false}$	(if_2)
$\dfrac{\llbracket R \rrbracket_{\zeta\frac{a}{x}}^{\mathfrak{A}} \triangleright U}{\llbracket \mathbf{let}\ x = t\ \mathbf{in}\ R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}$	if $a = \llbracket t \rrbracket_\zeta^{\mathfrak{A}}$	(let)
$\dfrac{\llbracket R \rrbracket_{\zeta\frac{a}{x}}^{\mathfrak{A}} \triangleright U_a \qquad \text{for each } a \in I}{\llbracket \mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ R \rrbracket_\zeta^{\mathfrak{A}} \triangleright \bigcup_{i \in I} U_i}$	if $I = \{a \in \lvert\mathfrak{A}\rvert : \llbracket \varphi \rrbracket_{\zeta\frac{a}{x}}^{\mathfrak{A}} = \mathit{true}\}$	(forall)
$\dfrac{\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U \qquad \llbracket S \rrbracket_\zeta^{U(\mathfrak{A})} \triangleright V}{\llbracket R\,;S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U\,;V}$	if U is consistent	(seq_1)
$\dfrac{\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}{\llbracket R\,;S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}$	if U is inconsistent	(seq_2)
$\dfrac{\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}{\llbracket \mathbf{try}\ R\ \mathbf{else}\ S \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}$	if U is consistent	(try_1)
$\dfrac{\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U \qquad \llbracket S \rrbracket_\zeta^{\mathfrak{A}} \triangleright V}{\llbracket \mathbf{try}\ R\ \mathbf{else}\ S \rrbracket_\zeta^{\mathfrak{A}} \triangleright V}$	if U is inconsistent	(try_2)
$\dfrac{\llbracket R\frac{t}{x} \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}{\llbracket \rho(t) \rrbracket_\zeta^{\mathfrak{A}} \triangleright U}$	if $\rho(x) = R$ is a rule definition	(def)

Table 2: Inductive definition of the semantics of ASM rules.

of a predicate $\mathcal{S}(R, \mathfrak{A}, \zeta, U)$ with the meaning 'rule R yields in state $\mathfrak{A}$ under the variable assignment ζ the update set U.' Instead of $\mathcal{S}(R, \mathfrak{A}, \zeta, U)$, however, we write $\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U$ and present the closure conditions of the inductive definition in [Table 2] as rules. The relation $\mathcal{S}(R, \mathfrak{A}, \zeta, U)$ is then the least relation satisfying the properties in [Table 2].

Note that for each state $\mathfrak{A}$ and each variable assignment ζ there exists at most one update set U such that $\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U$. Hence, the transition rules are deterministic. We say that a rule R is defined in state $\mathfrak{A}$ under the variable assignment ζ iff there exists an update set U such that $\llbracket R \rrbracket_\zeta^{\mathfrak{A}} \triangleright U$. This means that a rule R is defined in a state iff there is no infinite chain of recursive rule calls in the evaluation of the rule. If there are no loops in the call graph of an ASM, then all rules are defined in any state. If there are loops in the call graph of an ASM, then the evaluation of a rule may not terminate, i.e. there may be no update set U such that $[\![R]\!]_\zeta^{\mathfrak{A}} \triangleright U$.

Example 1. To illustrate the problem of definedness we consider a directed graph $G = \langle V, E\rangle$ and the following recursive rule definition:

$$\begin{array}{l} \texttt{explore}(x) = \\ \quad \texttt{reachable}(x) := \mathit{true} \\ \quad \textbf{forall } y \textbf{ with } \langle x, y\rangle \in E \textbf{ do } \texttt{explore}(y) \end{array}$$

For a vertex $x \in V$, the rule call $\texttt{explore}(x)$ is defined iff there exists no infinite path in the graph G starting at x. Moreover, if the rule call $\texttt{explore}(x)$ is defined, then its update set consists of all triples $\langle \texttt{reachable}, y, \mathit{true}\rangle$ such that there is a path from x to y in the graph G.

The notion of ASM run is the classical notion of computation of transition systems. A computation step in a given state consists in executing *simultaneously* all updates of the main transition rule of the ASM, if these updates are consistent. The run stops if the main transition rule is not defined or yields an inconsistent update set. If the update set is empty, then the ASM produces an infinite run (stuttering, never changing anymore the state). We do not allow that so-called *monitored* functions change during a computation [see Section 5].

Definition 6 (Run of an ASM). Let M be an ASM with vocabulary Σ, initial state $\mathfrak{A}$ and main rule name ρ. Let ζ be a variable assignment. A *run* of M is a finite or infinite sequence $\mathfrak{B}_0, \mathfrak{B}_1, \dots$ of states for Σ such that the following conditions are satisfied:

1. $\mathfrak{B}_0 = \mathfrak{A}$.
2. If $[\![\rho]\!]_\zeta^{\mathfrak{B}_n}$ is not defined or inconsistent, then $\mathfrak{B}_n$ is the last state.
3. Otherwise, $\mathfrak{B}_{n+1} = U(\mathfrak{B}_n)$, where $[\![\rho]\!]_\zeta^{\mathfrak{B}_n} \triangleright U$.

Runs are deterministic and independent of the variable assignment ζ, since we forbid global variables in rule definitions.

Remark. In the presence of sequential composition the following two rules are not equivalent:

$$\textbf{let } x = t \textbf{ in } R \quad \not\equiv \quad R\tfrac{t}{x}.$$

For a counter example, consider the following transition rule:

$$\textbf{let } x = f(0) \textbf{ in } (f(0) := 1\,; f(1) := x)$$

If we substitute the term $f(0)$ for x, then we obtain:

$$f(0) := 1 \,;\, f(1) := f(0)$$

In general, the two rules are not the same, because $f(0)$ is evaluated in different states. The following substitution property, however, is true for *static* terms t: If t is static and $[\![t]\!]^{\mathfrak{A}}_{\zeta} = a$, then

$$[\![R]\!]^{\mathfrak{A}}_{\zeta\frac{a}{x}} \vartriangleright U \iff [\![R\tfrac{t}{x}]\!]^{\mathfrak{A}}_{\zeta} \vartriangleright U.$$

If the term t contains dynamic functions, then the equivalence is not necessarily true, because t could be evaluated in different states on the right-hand side (due to sequential compositions).

3 Formalizing the Consistency of ASMs

Following [Groenboom and Renardel de Lavalette 1995] we extend the language of first-order predicate logic by a modal operator $[R]$ for each rule R. The intended meaning of a formula $[R]\varphi$ is that the formula φ is true after firing R. More precisely, the formula $[R]\varphi$ is true iff one of the following conditions is satisfied:

1. R is not defined or the update set of R is inconsistent, or
2. R is defined, the update set of R is consistent and φ is true in the next state after firing the update set of R.

Equivalently we can say that the formula $[R]\varphi$ is true in state $\mathfrak{A}$ under the variable assignment ζ iff for each set U such that $[\![R]\!]^{\mathfrak{A}}_{\zeta} \vartriangleright U$ and U is consistent, the formula φ is true in the state $U(\mathfrak{A})$ under ζ [see Table 3].

In order to express the definedness and the consistency of transition rules we extend the set of formulas by atomic formulas $\mathrm{def}(R)$ and $\mathrm{upd}(R, f, x, y)$. The semantics of these formulas is defined in [Table 3]. The formula $\mathrm{def}(R)$ asserts that the rule R is defined. The formula $\mathrm{upd}(R, f, x, y)$ expresses that rule R is defined and yields an update set which contains an update for f at x to y.

The basic properties of 'def' and 'upd' are listed in [Table 4] and [Table 5]. Note, that the equivalences in [Table 4] and [Table 5] are not the *definitions* of 'def' and 'upd'. The equivalences are just properties which are true under the interpretation of 'def' and 'upd' given in [Table 3]. The equivalences cannot be considered as definitions, since D9 and U9 depend on the rule definitions of the given ASM and the call graph of the rule definitions can contain cycles.

The formula $\mathrm{Con}(R)$ used in D8, U7 and U8 is defined as follows:

$$\mathrm{Con}(R) := \mathrm{def}(R) \wedge \bigwedge_{f \text{ dyn.}} \forall x, y, z\, (\mathrm{upd}(R, f, x, y) \wedge \mathrm{upd}(R, f, x, z) \rightarrow y = z)$$

$$[\![[R]\varphi]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if } [\![\varphi]\!]^{U(\mathfrak{A})}_{\zeta} = \mathit{true} \text{ for each consistent } U \text{ with } [\![R]\!]^{\mathfrak{A}}_{\zeta} \rhd U; \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$[\![\mathrm{def}(R)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if there exists an update set } U \text{ with } [\![R]\!]^{\mathfrak{A}}_{\zeta} \rhd U; \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$[\![\mathrm{upd}(R, f, s, t)]\!]^{\mathfrak{A}}_{\zeta} := \begin{cases} \mathit{true}, & \text{if ex. } U \text{ with } [\![R]\!]^{\mathfrak{A}}_{\zeta} \rhd U \text{ and } \langle f, [\![s]\!]^{\mathfrak{A}}_{\zeta}, [\![t]\!]^{\mathfrak{A}}_{\zeta} \rangle \in U; \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

Table 3: The semantics of modal formulas and basic predicates.

D1.	$\mathrm{def}(\mathbf{skip})$	
D2.	$\mathrm{def}(f(s) := t)$	
D3.	$\mathrm{def}(R\ S) \leftrightarrow \mathrm{def}(R) \land \mathrm{def}(S)$	
D4.	$\mathrm{def}(\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S) \leftrightarrow (\varphi \land \mathrm{def}(R)) \lor (\neg\varphi \land \mathrm{def}(S))$	
D5.	$\mathrm{def}(\mathbf{let}\ x = t\ \mathbf{in}\ R) \leftrightarrow \exists x\,(x = t \land \mathrm{def}(R))$	if $x \notin \mathrm{FV}(t)$
D6.	$\mathrm{def}(\mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ R) \leftrightarrow \forall x\,(\varphi \to \mathrm{def}(R))$	
D7.	$\mathrm{def}(R\,;S) \leftrightarrow \mathrm{def}(R) \land [R]\mathrm{def}(S)$	
D8.	$\mathrm{def}(\mathbf{try}\ R\ \mathbf{else}\ S) \leftrightarrow \mathrm{def}(R) \land (\mathrm{Con}(R) \lor \mathrm{def}(S))$	
D9.	$\mathrm{def}(\rho(t)) \leftrightarrow \mathrm{def}(R\frac{t}{x})$	if $\rho(x) = R$ is a rule definition of M

Table 4: Axioms for definedness.

U1.	$\neg\mathrm{upd}(\mathbf{skip}, f, x, y)$	
U2.	$\mathrm{upd}(f(s) := t, f, x, y) \leftrightarrow s = x \land t = y, \qquad \neg\mathrm{upd}(f(s) := t, g, x, y)$	if $f \neq g$
U3.	$\mathrm{upd}(R\ S, f, x, y) \leftrightarrow \mathrm{def}(R\ S) \land (\mathrm{upd}(R, f, x, y) \lor \mathrm{upd}(S, f, x, y))$	
U4.	$\mathrm{upd}(\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S, f, x, y) \leftrightarrow (\varphi \land \mathrm{upd}(R, f, x, y)) \lor (\neg\varphi \land \mathrm{upd}(S, f, x, y))$	
U5.	$\mathrm{upd}(\mathbf{let}\ z = t\ \mathbf{in}\ R, f, x, y) \leftrightarrow \exists z\,(z = t \land \mathrm{upd}(R, f, x, y))$	if $z \notin \mathrm{FV}(t)$
U6.	$\mathrm{upd}(\mathbf{forall}\ z\ \mathbf{with}\ \varphi\ \mathbf{do}\ R, f, x, y) \leftrightarrow$ $\mathrm{def}(\mathbf{forall}\ z\ \mathbf{with}\ \varphi\ \mathbf{do}\ R) \land \exists z\,(\varphi \land \mathrm{upd}(R, f, x, y))$	
U7.	$\mathrm{upd}(R\,; S, f, x, y) \leftrightarrow$ $(\mathrm{upd}(R, f, x, y) \land [R](\mathrm{def}(S) \land \mathrm{inv}(S, f, x))) \lor$ $(\mathrm{Con}(R) \land [R]\mathrm{upd}(S, f, x, y))$	
U8.	$\mathrm{upd}(\mathbf{try}\ R\ \mathbf{else}\ S, f, x, y) \leftrightarrow$ $(\mathrm{Con}(R) \land \mathrm{upd}(R, f, x, y)) \lor$ $(\mathrm{def}(R) \land \neg\mathrm{Con}(R) \land \mathrm{upd}(S, f, x, y))$	
U9.	$\mathrm{upd}(\rho(t), f, x, y) \leftrightarrow \mathrm{upd}(R\frac{t}{z}, f, x, y)$	if $\rho(z) = R$ is a rule definition of M

Table 5: Axioms for updates.

It is true in a state iff the rule R is defined and yields a consistent update set:

$$[\![\mathrm{Con}(R)]\!]_\zeta^{\mathfrak{A}} = \mathit{true} \iff \text{there exists a consistent } U \text{ with } [\![R]\!]_\zeta^{\mathfrak{A}} \rhd U.$$

The formula $\mathrm{inv}(R, f, x)$ in U7 expresses that the rule R does not update the function f at the argument x. It is a simple abbreviation defined as follows:

$$\mathrm{inv}(R, f, x) := \forall y\, \neg\mathrm{upd}(R, f, x, y)$$

Note, that it would be wrong to define the predicate $\mathrm{upd}(R, f, x, y)$ by saying that $f(x)$ is different from y in the present state but equal to y in the next state after firing rule R:

$$\mathrm{upd}(R, f, x, y) := f(x) \neq y \land [R]f(x) = y \qquad \text{(wrong definition)}$$

Using this definition, the predicate $\mathrm{upd}(f(0) := 1, f, 0, 1)$ would be false in a state where $f(0)$ is equal to 1, although the rule $f(0) := 1$ does update the function f at the argument 0 to 1.

4 Basic Axioms and Rules of the Logic

The formulas of the logic for abstract state machines are generated by the following grammar:

$$\begin{aligned} \varphi, \psi \ ::= \ & s = t \mid \neg\varphi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \varphi \to \psi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \\ & \mathrm{def}(R) \mid \mathrm{upd}(R, f, s, t) \mid [R]\varphi \end{aligned}$$

A formula is called *pure* (or *first-order*), if it contains neither the predicate 'def' nor 'upd' nor the modal operator $[R]$. A formula is called *static*, if it does not contain dynamic function names. The formulas used in If-Then-Else and Forall rules must be pure formulas.

The semantics of formulas is given by the definitions in [Table 1] and [Table 3]. The equivalence $\varphi \leftrightarrow \psi$ is defined by $(\varphi \to \psi) \land (\psi \to \varphi)$. A formula φ is called *valid*, if $[\![\varphi]\!]_\zeta^{\mathfrak{A}} = \mathit{true}$ for all states $\mathfrak{A}$ and variable assignments ζ. A formula φ is a *logical consequence* of a set of sentences Ψ (written $\Psi \models_M \varphi$), if $[\![\varphi]\!]_\zeta^{\mathfrak{A}} = \mathit{true}$ for all states $\mathfrak{A}$ and variable assignments ζ such that $[\![\psi]\!]_\zeta^{\mathfrak{A}} = \mathit{true}$ for every $\psi \in \Psi$. Note that the interpretation of rule names in transition rules depends on a given abstract state machine M which is made explicit as a subscript in $\Psi \models_M \varphi$.

The substitution of a term t for a variable x in a formula φ is denoted by $\varphi\frac{t}{x}$ and is defined as usual. Variables bound by a quantifier, a **let** or a **forall** have to be renamed when necessary. The substitution is also performed inside of transition rules that occur in formulas. The following substitution property holds.

Lemma 7 (Substitution). *If t is static and $a = [\![t]\!]^{\mathfrak{A}}_{\zeta}$, then $[\![\varphi]\!]^{\mathfrak{A}}_{\zeta\frac{a}{x}} = [\![\varphi\frac{t}{x}]\!]^{\mathfrak{A}}_{\zeta}$.*

We define two transition rules R and S to be *equivalent*, if they are defined and consistent in the same states and produce the same next state when they are fired. An observer from outside cannot distinguish two equivalent transition rules by just looking at the runs generated by them.

Definition 8 (Equivalence). The formula $R \simeq S$ is defined as follows:

$$
\begin{aligned}
R \simeq S \;:\Longleftrightarrow\; & (\mathrm{Con}(R) \vee \mathrm{Con}(S)) \rightarrow (\mathrm{Con}(R) \wedge \mathrm{Con}(S) \wedge \\
& \bigwedge_{f\ \mathrm{dyn.}} \forall x, y\, (\mathrm{upd}(R, f, x, y) \rightarrow (\mathrm{upd}(S, f, x, y) \vee f(x) = y)) \wedge \\
& \bigwedge_{f\ \mathrm{dyn.}} \forall x, y\, (\mathrm{upd}(S, f, x, y) \rightarrow (\mathrm{upd}(R, f, x, y) \vee f(x) = y)))
\end{aligned}
$$

The formula $R \simeq S$ has the intended meaning:

Lemma 9. *The formula $R \simeq S$ is true in $\mathfrak{A}$ under ζ iff the following two conditions are true:*

1. $[\![\mathrm{Con}(R)]\!]^{\mathfrak{A}}_{\zeta} = true$ *iff* $[\![\mathrm{Con}(S)]\!]^{\mathfrak{A}}_{\zeta} = true$.
2. *If* $[\![R]\!]^{\mathfrak{A}}_{\zeta} \rhd U$, $[\![S]\!]^{\mathfrak{A}}_{\zeta} \rhd V$ *and U and V are consistent, then* $U(\mathfrak{A}) = V(\mathfrak{A})$.

We already know that the axioms D1–D9 and U1–U9 are valid for a given abstract state machine M. Together with the following principles they will be the basic axioms and rules of our logic $\mathcal{L}(M)$. We start with the standard axioms and rules of the classical predicate calculus with equality. The quantifier axioms 1 and 2 as well as the substitution scheme, however, have to be restricted to static terms which do not contain dynamic function names. The reason for the restriction is that, if we substitute a term t for a variable x, then t can be evaluated in different states due to sequential compositions of transition rules.

Axiom 3 is the so-called axiom K of modal logic. Together with the necessitation rule 4 it allows to derive all modal principles that are valid in arbitrary Kripke frames. Axiom 5 uses the fact that a rule R which is not defined or yields an inconsistent update set cannot be fired in a state. Since there is no successor state in this case, the necessity operator $[R]$ is trivial. Axiom 6 can be applied because the transition rules are deterministic. In modal logic the axiom for deterministic accessibility relations is written as $\Diamond\varphi \rightarrow \Box\varphi$ or $\neg\Box\neg\varphi \rightarrow \Box\varphi$.

The Barcan axiom 7 is true, since the universe does not change during the run of an ASM. Hence the quantifiers range over the same set in every state of a computation. Axioms 8 and 9 assert that the meaning of pure, static first-order formulas (which do not contain dynamic function names) is the same in all states of a computation.

Axiom 12 asserts that if a rule updates a function, then the rule is defined. Axiom 13 says that, if a rule updates a function f at the argument x to the value y, then in the next state the value of f at x is equal to y. If the rule does not update f at the argument x, then the value of f in the next state is the same as in the present state [Axiom 14].

The extensionality axiom 15 asserts that the modal operators of two equivalent transition rules are the same. Axioms 16 and 17 are well-known from dynamic logic. They express that the empty rule has no effect on a state and that the sequential composition of transition rules corresponds to their sequential execution.

I. Classical logic with equality: We use the axioms and rules of the classical predicate calculus with equality. The quantifier axioms, however, are restricted.

II. Restricted quantifier axioms:

1. $\forall x\, \varphi \to \varphi\frac{t}{x}$ — if t is static or φ is pure
2. $\varphi\frac{t}{x} \to \exists x\, \varphi$ — if t is static or φ is pure

III. Modal axioms and rules:

3. $[R](\varphi \to \psi) \wedge [R]\varphi \to [R]\psi$
4. $\dfrac{\varphi}{[R]\varphi}$
5. $\neg\mathrm{Con}(R) \to [R]\varphi$
6. $\neg[R]\varphi \to [R]\neg\varphi$

IV. The Barcan axiom:

7. $\forall x[R]\varphi \to [R]\forall x\varphi$ — if $x \notin \mathrm{FV}(R)$.

V. Axioms for pure static formulas:

8. $\varphi \to [R]\varphi$ — if φ is pure and static
9. $\mathrm{Con}(R) \wedge [R]\varphi \to \varphi$ — if φ is pure and static

VI. Axioms for def **and** upd**:**

10. D1–D9 in Table 4
11. U1–U9 in Table 5

VII. Update axioms for transition rules:

12. $\mathrm{upd}(R, f, x, y) \to \mathrm{def}(R)$
13. $\mathrm{upd}(R, f, x, y) \to [R]f(x) = y$
14. $\mathrm{inv}(R, f, x) \wedge f(x) = y \to [R]f(x) = y$

VIII. Extensionality axiom for transition rules:

15. $R \simeq S \to ([R]\varphi \leftrightarrow [S]\varphi)$

IX. Axioms from dynamic logic:

16. $[\mathbf{skip}]\varphi \leftrightarrow \varphi$
17. $[R\,;S]\varphi \leftrightarrow [R][S]\varphi$

The notion of derivability is defined as usual. We write $\Psi \vdash_M \varphi$, if the formula φ is derivable from the set of sentences Ψ using the axioms and rules I–IX. Note that axioms D9 and U9 depend on the rule definitions of the given abstract state machine M. Therefore, M has to be added as a parameter in $\Psi \vdash_M \varphi$. Since the principles I–IX are valid, the logic is sound.

Theorem 10 (Soundness). *If $\Psi \vdash_M \varphi$, then $\Psi \models_M \varphi$.*

Remark. The formula $\forall x\, \varphi \to \varphi\frac{t}{x}$ is not valid for non-static terms t. Consider the following tautology:

$$\forall x\, (x = 0 \to [f(0) := 1]x = 0).$$

If we substitute the term $f(0)$ for x, then we obtain the formula

$$f(0) = 0 \to [f(0) := 1]f(0) = 0.$$

This formula is not valid. Hence, the quantifier axioms must be restricted.

Several axioms use the formula $\mathrm{Con}(R)$ which asserts the consistency of the transition rule R. For example, $\mathrm{Con}(R)$ is used in axioms 5 and 9 as well as in the extensionality axiom 15. Since the notion of consistency is fundamental, we mention several equivalences which express the consistency of a compound transition rule in terms of consistency of its components.

Lemma 11. *The following consistency properties are derivable:*

18. $\mathrm{Con}(\mathbf{skip})$
19. $\mathrm{Con}(f(s) := t)$
20. $\mathrm{Con}(R\ S) \leftrightarrow \mathrm{Con}(R) \wedge \mathrm{Con}(S) \wedge \mathrm{joinable}(R, S)$
21. $\mathrm{Con}(\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S) \leftrightarrow (\varphi \wedge \mathrm{Con}(R)) \vee (\neg\varphi \wedge \mathrm{Con}(S))$
22. $\mathrm{Con}(\mathbf{let}\ x = t\ \mathbf{in}\ R) \leftrightarrow \exists x\, (x = t \wedge \mathrm{Con}(R))$ *if* $x \notin \mathrm{FV}(t)$
23. $\mathrm{Con}(\mathbf{forall}\, x\, \mathbf{with}\, \varphi\, \mathbf{do}\, R) \leftrightarrow \forall x\, (\varphi \to \mathrm{Con}(R) \wedge \forall y(\varphi\frac{y}{x} \to \mathrm{joinable}(R, R\frac{y}{x}))$
24. $\mathrm{Con}(R\, ; S) \leftrightarrow \mathrm{Con}(R) \wedge [R]\mathrm{Con}(S)$
25. $\mathrm{Con}(\mathbf{try}\ R\ \mathbf{else}\ S) \leftrightarrow \mathrm{Con}(R) \vee (\mathrm{def}(R) \wedge \mathrm{Con}(S))$
26. $\mathrm{Con}(\rho(t)) \leftrightarrow \mathrm{Con}(R\frac{t}{x})$ *if $\rho(x) = R$ is a rule definition of M*

The predicate $\mathrm{joinable}(R, S)$ which is used in 20 to reduce the consistency of a parallel composition $R\ S$ into consistency properties of R and S is defined as follows (where x, y, z are not free in R):

$$\mathrm{joinable}(R, S) := \bigwedge_{f \text{ dyn.}} \forall x, y, z\, (\mathrm{upd}(R, f, x, y) \wedge \mathrm{upd}(S, f, x, z) \to y = z),$$

It expresses that the update sets of R and S do not conflict. This means, whenever R and S both update a function f at the same argument x, then the new values of f at x are the same.

Lemma 12. *The following principles are derivable:*

27. $\mathrm{Con}(R) \wedge [R]f(x) = y \rightarrow \mathrm{upd}(R, f, x, y) \vee (\mathrm{inv}(R, f, x) \wedge f(x) = y)$
28. $\mathrm{Con}(R) \wedge [R]\varphi \rightarrow \neg[R]\neg\varphi$
29. $[R]\exists x\, \varphi \leftrightarrow \exists x\, [R]\varphi,$ *if* $x \notin \mathrm{FV}(R)$.

[Groenboom and Renardel de Lavalette 1995] introduce different axioms for transition rules. Their axioms FM1, FM2, AX1, AX2 are derivable in our system using the update axioms 13 and 14.

Lemma 13. *The following principles of [Groenboom and Renardel de Lavalette 1995] are derivable:*

30. $s = x \rightarrow (y = t \leftrightarrow [f(s) := t]f(x) = y)$
31. $s \neq x \rightarrow (y = f(x) \leftrightarrow [f(s) := t]f(x) = y)$
32. $[R]f(x) = y \wedge [S]f(x) = y \rightarrow [R\ S]f(x) = y$
33. $f(x) \neq y \wedge ([R]f(x) = y \vee [S]f(x) = y) \rightarrow [R\ S]f(x) = y.$

The following inverse implication of 32 and 33 is not mentioned in [Groenboom and Renardel de Lavalette 1995] (maybe because of the lack of a consistency notion), but is derivable in our system:

$$\begin{aligned} &\mathrm{Con}(R\ S) \wedge [R\ S]f(x) = y \rightarrow \\ &\quad ([R]f(x) = y \wedge [S]f(x) = y) \vee (f(x) \neq y \wedge ([R]f(x) = y \vee [S]f(x) = y)) \end{aligned}$$

Several principles known from dynamic logic are derivable using the extensionality axiom 15.

Lemma 14. *The following principles are derivable:*

34. $[\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S]\psi \leftrightarrow (\varphi \wedge [R]\psi) \vee (\neg\varphi \wedge [S]\psi)$
35. $[\mathbf{let}\ x = t\ \mathbf{in}\ R]\varphi \leftrightarrow \exists x\, (x = t \wedge [R]\varphi),$ *if* $x \notin \mathrm{FV}(t) \cup \mathrm{FV}(\varphi)$.
36. $[\mathbf{try}\ R\ \mathbf{else}\ S]\varphi \leftrightarrow [R]\varphi \wedge (\mathrm{def}(R) \wedge \neg\mathrm{Con}(R) \rightarrow [S]\varphi)$
37. $[\rho(t)]\varphi \leftrightarrow [R\frac{t}{x}]\varphi,$ *if* $\rho(x) = R$ *is a rule definition of* M.

A more intensional equivalence of transition rules can be defined as follows:

$$R \mathrel{\dot{\simeq}} S \ :\Longleftrightarrow\ (\mathrm{def}(R) \vee \mathrm{def}(S)) \rightarrow (\mathrm{def}(R) \wedge \mathrm{def}(S) \wedge \bigwedge_{f \text{ dyn.}} \forall x, y\, (\mathrm{upd}(R, f, x, y) \leftrightarrow \mathrm{upd}(S, f, x, y)))$$

The formula $R \mathrel{\dot{\simeq}} S$ asserts that the transition rules R and S are defined in the same states and yield the same update sets. The so obtained notion of equivalence is stronger than the one used in the extensionality axiom 15.

Lemma 15. $R \mathrel{\dot{\simeq}} S \rightarrow R \simeq S$ *is derivable.*

[Schönegge 1995] uses in his sequent calculus for the extended dynamic logic new rules that express the commutativity, the associativity and similar properties of the parallel combination of transition rules. In our system, these properties are derivable.

Lemma 16. *The following principles of [Schönegge 1995] are derivable:*

38. $(R\ \mathbf{skip}) \doteq R$
39. $(R\ S) \doteq (S\ R)$
40. $((R\ S)\ T) \doteq (R\ (S\ T))$
41. $(R\ R) \doteq R$
42. $(\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S)\ T \doteq \mathbf{if}\ \varphi\ \mathbf{then}\ (R\ T)\ \mathbf{else}\ (S\ T)$
43. $T\ (\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S) \doteq \mathbf{if}\ \varphi\ \mathbf{then}\ (T\ R)\ \mathbf{else}\ (T\ S)$

If we can derive $R \doteq S$, then we immediately obtain the principle $[R]\varphi \leftrightarrow [S]\varphi$ using [Lemma 15] and the extensionality axiom 15. It is not clear to us, whether for example the commutativity of the parallel composition, $[R\ S]\varphi \leftrightarrow [S\ R]\varphi$, could be derived in the formal system of [Groenboom and Renardel de Lavalette 1995].

Lemma 17. *The following properties of the sequential composition are derivable:*

44. $(R\,;\mathbf{skip}) \doteq R$
45. $(\mathbf{skip}\,; R) \doteq R$
46. $((R\,;S)\,;T) \doteq (R\,;(S\,;T))$
47. $(\mathbf{if}\ \varphi\ \mathbf{then}\ R\ \mathbf{else}\ S)\,;T \doteq \mathbf{if}\ \varphi\ \mathbf{then}\ (R\,;T)\ \mathbf{else}\ (S\,;T)$

The dynamic logic with array assignments (see Harel [1983]) uses a substitution principle which is derivable in our system. Let φ be a quantifier-free, pure (first-order) formula. Then by $\varphi\frac{t}{f(s)}$ we denote the formula which is obtained in the following way. First, φ is transformed into an equivalent formula

$$\varphi \leftrightarrow \exists\mathbf{x}\,\exists\mathbf{y}\left(\bigwedge_{i=1}^{n} f(x_i) = y_i \wedge \psi\right),$$

where $\mathbf{x} = x_1, \ldots, x_n$, $\mathbf{y} = y_1, \ldots, y_n$ and ψ does not contain f. Then we define:

$$\varphi\frac{t}{f(s)} := \exists\mathbf{x}\,\exists\mathbf{y}\left(\bigwedge_{i=1}^{n}((x_i = s \wedge y_i = t) \vee (x_i \neq s \wedge f(x_i) = y_i)) \wedge \psi\right)$$

The substitution of t for $f(s)$ can be generalized to arbitrary first-order formulas by first bringing them into prenex form and then applying the transformation to the quantifier-free kernel.

Lemma 18. *For any first-order formula* φ*, the following substitution principle is derivable:* $\varphi\frac{t}{f(s)} \leftrightarrow [f(s) := t]\varphi$.

The If-Then rule can be defined in terms of If-Then-Else in the standard way:

$$\textbf{if } \varphi \textbf{ then } R := \textbf{if } \varphi \textbf{ then } R \textbf{ else skip}$$

An ASM is called *simple*, if it is defined by a single rule R, which has the following form:

$$\begin{array}{c} \textbf{if } \varphi_1 \textbf{ then } f(s_1) := t_1 \\ \textbf{if } \varphi_2 \textbf{ then } f(s_2) := t_2 \\ \vdots \\ \textbf{if } \varphi_n \textbf{ then } f(s_n) := t_n \end{array}$$

Simple ASMs have the obvious properties formulated in the following lemma. Property 51 is a variant of the basic axiom of [Poetzsch-Heffter 1994]. It can easily be extended to disjoint If-Then rules with simultaneous function updates.

Lemma 19. *Let R be the rule of a simple ASM. Then,*

48. $\mathrm{Con}(R) \leftrightarrow \bigwedge_{i<j} (\varphi_i \wedge \varphi_j \wedge s_i = s_j \rightarrow t_i = t_j)$

49. $\mathrm{upd}(R, f, x, y) \leftrightarrow \bigvee_{i=1}^{n} (\varphi_i \wedge x = s_i \wedge y = t_i)$

50. $\mathrm{inv}(R, f, x) \leftrightarrow \bigwedge_{i=1}^{n} (\varphi_i \rightarrow x \neq s_i)$

51. $\bigvee_{i=1}^{n} \varphi_i \wedge \bigwedge_{i<j} \neg(\varphi_i \wedge \varphi_j) \wedge \bigwedge_{i=1}^{n} (\varphi_i \rightarrow \psi \frac{t_i}{f(s_i)}) \rightarrow [R]\psi,$ *if ψ is first-order.*

Iteration can be reduced to recursion. We can define the While rule recursively, as follows:

$$\textbf{while } \varphi \textbf{ do } R = \textbf{if } \varphi \textbf{ then } (R\,; \textbf{while } \varphi \textbf{ do } R)$$

The expression **while** φ **do** R has to be read as a rule call $\rho(\mathbf{x})$, where $\mathbf{x}$ are the free variables of φ and R. So the above equation stands for the following rule definition:

$$\rho(\mathbf{x}) = \textbf{if } \varphi \textbf{ then } (R\,; \rho(\mathbf{x}))$$

Lemma 20. *The following properties of the While rule are derivable:*

52. $\mathrm{Con}(\textbf{while } \varphi \textbf{ do } R) \leftrightarrow (\varphi \rightarrow \mathrm{Con}(R) \wedge [R]\mathrm{Con}(\textbf{while } \varphi \textbf{ do } R))$

53. $[\textbf{while } \varphi \textbf{ do } R]\psi \leftrightarrow (\varphi \wedge [R][\textbf{while } \varphi \textbf{ do } R]\psi) \vee (\neg\varphi \wedge \psi)$

Several properties of ASMs can be expressed in the basic logic. Let M be the distinguished rule name of the ASM and φ_{init} a formula characterizing initial states of M. Then we can express:

- ψ is an invariant of M by $(\varphi_{\mathrm{init}} \rightarrow \psi) \wedge (\psi \rightarrow [M]\psi)$.
- ψ ensures the consistency of M by $(\varphi_{\mathrm{init}} \rightarrow \psi) \wedge (\psi \rightarrow \mathrm{Con}(M) \wedge [M]\psi)$.

As another example, the statement in [Stärk et al. 2001] for the correctness of the compiler from Java to the JVM can be formulated as follows:

$$(\varphi_{\text{init}} \to \varphi_{\text{eqv}}) \land (\varphi_{\text{eqv}} \to [J](\varphi_{\text{eqv}} \lor [V]\varphi_{\text{eqv}} \lor [V][V]\varphi_{\text{eqv}} \lor [V][V][V]\varphi_{\text{eqv}}))$$

Here, J is an ASM that specifies the semantics of a Java source level program according to the *Java Language Specification*; V is an ASM that specifies the *Java Virtual Machine*. The two ASMs have disjoint dynamic function names and use the same static functions. The formula φ_{eqv} expresses that two dynamic states of the two ASMs are equivalent for a given Java program and its compiled bytecode program. The above formula says, that if two states are equivalent, then for each step of J the ASM V has to make zero, one, two or three steps to reach an equivalent state again. The proof in [Stärk et al. 2001] which comprises 83 cases could be carried out in the basic system with appropriate structural induction principles for lists and abstract syntax trees (which are encoded using static functions).

5 Why Deterministic Transition Rules?

There are different possibilities to add non-determinism to ASMs. We first show that the obvious Choose rule leads to a much more complex system and therefore we do not recommend it. The Choose rule has the following syntax:

$$\textbf{choose } x \textbf{ with } \varphi \textbf{ do } R$$

The intended meaning is that we choose an element x satisfying the property φ and then fire the rule R with the chosen x. This semantics can be expressed by the following two rules:

$$\frac{[\![R]\!]^{\mathfrak{A}}_{\zeta\frac{a}{x}} \rhd U}{[\![\textbf{choose } x \textbf{ with } \varphi \textbf{ do } R]\!]^{\mathfrak{A}}_{\zeta} \rhd U} \text{ if } a \in |\mathfrak{A}| \text{ and } [\![\varphi]\!]^{\mathfrak{A}}_{\zeta\frac{a}{x}} = \mathit{true}$$

$$\frac{}{[\![\textbf{choose } x \textbf{ with } \varphi \textbf{ do } R]\!]^{\mathfrak{A}}_{\zeta} \rhd \emptyset} \text{ if } [\![\neg\exists x\, \varphi]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true}$$

The second rule is for the case when there is no element x satisfying the property φ. In this case, the semantics of the Choose rule is the empty set of updates.

The problem is now that the relation $\rhd$ is no longer deterministic. For a given transition rule R there can be several different update sets U such that $[\![R]\!]^{\mathfrak{A}}_{\zeta} \rhd U$ is derivable. In the definition of a run of an ASM [Definition 6] on has to choose an update set in each step to obtain the next state of the run. As a consequence, there can be several different runs for a given initial state of a machine.

Unfortunately, the formalization of consistency cannot be applied directly to non-deterministic ASMs. The formula $\mathrm{Con}(R)$ (as defined in [Section 3]) expresses that the *union of all possible* update sets of R in a given state is consistent. This is clearly not what is meant by consistency. Therefore one has to add $\mathrm{Con}(R)$ as atomic formula to the logic.

The first question is then, what is the semantics of $\mathrm{Con}(R)$ for possibly non-deterministic rules? One possibility is to define

$$[\![\mathrm{Con}(R)]\!]_\zeta^{\mathfrak{A}} = \mathit{true} \;:\Longleftrightarrow\; \text{for each } U\text{, if } [\![R]\!]_\zeta^{\mathfrak{A}} \rhd U\text{, then } U \text{ is consistent.}$$

Hence, R is consistent, if each update set for R is consistent. (If there is no U with $[\![R]\!]_\zeta^{\mathfrak{A}} \rhd U$, then R is considered as consistent, too.) But then, the following implication is no longer true:

$$\mathrm{Con}(R\ S) \rightarrow \mathrm{Con}(R)$$

For example, if S is not defined in a state, then the parallel composition $R\ S$ is also not defined and therefore, by definition, consistent. The rule R, however, could be inconsistent. The property 20 in [Lemma 11] could then be written as follows:

$$\mathrm{Con}(R\ S) \leftrightarrow (\mathrm{def}(R\ S) \rightarrow \mathrm{Con}(R) \wedge \mathrm{Con}(S) \wedge \mathrm{joinable}(R, S))$$

Another possibility is to define the consistency of rules as *weak consistency*:

$$[\![\mathrm{Con}(R)]\!]_\zeta^{\mathfrak{A}} = \mathit{true} \;:\Longleftrightarrow\; \text{there exists a consistent } U \text{ such that } [\![R]\!]_\zeta^{\mathfrak{A}} \rhd U.$$

For this weak notion of consistency, however, we could not use the relation $\mathrm{joinable}(R, S)$ in property 20 in [Lemma 11]. It is not clear what we should use instead:

$$\mathrm{Con}(R\ S) \leftrightarrow \mathrm{Con}(R) \wedge \mathrm{Con}(S) \wedge ?$$

A third possibility is to combine definedness and strong consistency as follows:

$$[\![\mathrm{Con}(R)]\!]_\zeta^{\mathfrak{A}} = \mathit{true} \;:\Longleftrightarrow\; \begin{array}{l}\text{(a) there exists a } U \text{ such that } [\![R]\!]_\zeta^{\mathfrak{A}} \rhd U\text{, and}\\ \text{(b) for each } U\text{, if } [\![R]\!]_\zeta^{\mathfrak{A}} \rhd U\text{, then } U \text{ is consistent.}\end{array}$$

But what about property 24 in [Lemma 11]? Is the following implication still true under the new interpretation of consistency?

$$\mathrm{Con}(R\,;S) \rightarrow [R]\mathrm{Con}(S)$$

Assume that $[\![R]\!]_\zeta^{\mathfrak{A}} \rhd U$ and U is consistent. How do we know that the rule S is defined in state $U(\mathfrak{A})$? How do we know that there exists a V with $[\![S]\!]_\zeta^{U(\mathfrak{A})} \rhd V$?

One problem remains in all three cases, namely how to characterize the update predicate for sequential compositions (Axiom U7 in [Table 5]). What should be on the right-hand side of the following equivalence?

$$\mathrm{upd}(R\,;S, f, x, y) \leftrightarrow ?$$

Consider the case, where $[\![R]\!]_\zeta^{\mathfrak{A}} \rhd U$ and U is consistent, $\langle f, a, b\rangle \in U$ and the rule S does not update the function f at the argument a in the state $U(\mathfrak{A})$ after firing the rule R with the update set U in $\mathfrak{A}$. This case cannot be expressed by the formula

$$\mathrm{upd}(R, f, a, b) \wedge [R](\mathrm{def}(S) \wedge \mathrm{inv}(S, f, a)),$$

since the rule R could have several different consistent update sets and the rule S could update the function f at the argument a after firing one of them, although it does not after firing the given update set U of R.

Because of all the problems, it seems impossible to find natural and simple axioms for the Choose rule together with sequential composition and recursive rule definitions. Therefore we follow the well-known approach that non-determinism is modeled from outside such that, in the view of the transition rules, everything is deterministic. For example, we take a *static* function *choose* from $\mathbb{N}$ into the set $\{0, 1\}$ and use it in the following way:

$$\begin{array}{l} \mathit{state} := \mathit{state} + 1 \\ \textbf{if } \mathit{choose}(\mathit{state}) = 1 \textbf{ then } R \textbf{ else } S \end{array}$$

Hence, the possible choices are made in advance and stored in the static function *choose*. Once the choices have been made, the run of the ASM is completely deterministic and we can apply the axioms and rules of our logic.

6 Completeness for hierarchical ASMs

An ASM is called *hierarchical*, if the call graph of the rule definitions does not contain cycles, in other words, if the ASM does not contain recursive rule definitions. An ASM is hierarchical iff it is possible to assign levels to the rule names such that in a rule definition $\rho(x) = R$ the levels of rule names in R are less than the level of ρ.

In an earlier version of the article the completeness of the logic for hierarchical ASMs was obtained via an extension of the Henkin model construction. Later, Gerard R. Renardel de Lavalette observed that, in the case of hierarchical ASMs, the logic for ASMs is a *definitional extension* of first-order logic (FOL). This means that there exists a translation of formulas φ of the logic for a hierarchical ASM M into first-order formulas φ^* with the following properties:

1. The equivalence $\varphi \leftrightarrow \varphi^*$ is provable in $\mathcal{L}(M)$.

2. If φ is provable in $\mathcal{L}(M)$, then φ^* is provable in FOL.

Hence, for hierarchical ASMs, it is possible to eliminate the modal operator $[R]$ as well as the atomic formulas $\mathrm{def}(R)$ and $\mathrm{upd}(R, f, s, t)$. Due to parallel updates, however, the translation of modal formulas into FOL is more complicated than the comparable embedding of the logic of modification and creation [see Renardel de Lavalette 2001, Theorem 7].

We first observe that the transition rules of hierarchical ASMs are always defined. If R is a transition rule which uses rules from a hierarchical machine M, then the formula $\mathrm{def}(R)$ is derivable in $\mathcal{L}(M)$. Therefore we can identify the formula $\mathrm{def}(R)$ with the constant $\top$ (true).

Moreover, we can assume that atomic formulas are restricted to simple formulas $x = y$, $f(x) = y$, $\mathrm{upd}(R, f, x, y)$. To bring general atomic formulas into this form, one can apply the following principles of $\mathcal{L}(M)$:

$$\begin{array}{lcl} s = t & \leftrightarrow & \exists x\,(s = x \wedge t = x) \\ \mathrm{upd}(R, f, s, t) & \leftrightarrow & \exists x, y\,(s = x \wedge t = y \wedge \mathrm{upd}(R, f, x, y)) \\ f(s) = y & \leftrightarrow & \exists x\,(s = x \wedge f(x) = y) \end{array}$$

The translation of modal formulas into FOL distributes over negation, boolean connectives and quantifiers. For eliminating $\mathrm{upd}(R, f, x, y)$ we use the axioms U1–U9 in [Table 5]. For eliminating the modal operator $[R]$ in $[R]\varphi$ we first translate φ into a first-order formula and use then the following equivalences of $\mathcal{L}(M)$:

$$\begin{array}{lcl} [R]x = y & \leftrightarrow & (\mathrm{Con}(R) \to x = y) \\ [R]f(x) = y & \leftrightarrow & (\mathrm{Con}(R) \to \mathrm{upd}(R, f, x, y) \vee (\mathrm{inv}(R, f, x) \wedge f(x) = y)) \\ [R]\neg\varphi & \leftrightarrow & (\mathrm{Con}(R) \to \neg[R]\varphi) \\ [R](\varphi \wedge \psi) & \leftrightarrow & ([R]\varphi \wedge [R]\psi) \\ [R](\varphi \vee \psi) & \leftrightarrow & ([R]\varphi \vee [R]\psi) \\ [R](\varphi \to \psi) & \leftrightarrow & ([R]\varphi \to [R]\psi) \\ [R]\forall x\,\varphi & \leftrightarrow & \forall x\,[R]\varphi \\ [R]\exists x\,\varphi & \leftrightarrow & \exists x\,[R]\varphi \end{array}$$

In order to see that the translation is well-defined we define a rank for formulas and transition rules as follows:

$$\begin{array}{l} |s = t| := 0 \\ |\neg\varphi| := |\varphi| + 1 \\ |\varphi \wedge \psi| := |\varphi \vee \psi| := |\varphi \to \psi| := \max(|\varphi|, |\psi|) + 1 \\ |\forall x\,\varphi| := |\exists x\,\varphi| := |\varphi| + 1 \\ |\mathrm{upd}(R, f, x, x)| := |\psi| + 1, \quad \text{if } \mathrm{upd}(R, f, x, x) \leftrightarrow \psi \text{ is an instance of U1–U9} \\ |[R]\varphi| := |R| + |\varphi| + 1 \\ |R| := \max(|\mathrm{Con}(R)|, |\mathrm{upd}(R, f, x, y)|, |\mathrm{inv}(R, f, x)|) + 1 \end{array}$$

The rank is also used to show that the translation into FOL has the above properties 1 and 2. The completeness and compactness theorems then follow from the corresponding results for FOL.

Theorem 21 (Completeness). *Let M be a hierarchical ASM and Ψ be a set of sentences. If $\Psi \models_M \varphi$, then $\Psi \vdash_M \varphi$.*

Theorem 22 (Compactness). *If each finite subset of a set of formulas Ψ is satisfiable, then Ψ is satisfiable.*

Remark. For ASMs with recursive rule definitions the logic is incomplete. Consider an ASM over the vocabulary of arithmetic with the following recursive rule definition (there are no dynamic function names):

$$r(x, y) = \textbf{if } x = y \textbf{ then skip else } r(x, y + 1)$$

Let φ_{N} be the conjunction of the following 7 formulas:

$$\begin{array}{ll}
\forall x\,(x + 1 \neq 0) & \forall x, y\,(x + 1 = y + 1 \rightarrow x = y) \\
\forall x\,(x + 0 = x) & \forall x, y\,(x + (y + 1) = (x + y) + 1) \\
\forall x\,(x * 0 = x) & \forall x, y\,(x * (y + 1) = (x * y) + x) \\
\forall x\,\mathrm{def}(r(x, 0)) &
\end{array}$$

Note, that $[\![r(x, t)]\!]^{\mathfrak{A}}_{\zeta} \rhd U$ is derivable in [Table 2] iff U is the empty set and $\zeta(x) = [\![t + 1 + \ldots + 1]\!]^{\mathfrak{A}}_{\zeta}$. Hence, $[\![\mathrm{def}(x, 0)]\!]^{\mathfrak{A}}_{\zeta} = \mathit{true}$ iff $\zeta(x)$ is reachable from 0 by a finite number of successor steps. Therefore a structure $\mathfrak{A}$ is a model of φ_{N} iff $\mathfrak{A}$ is isomorphic to the structure of natural numbers. An arithmetical sentence ψ is true in the structure of natural numbers iff the formula $\varphi_{\mathrm{N}} \rightarrow \psi$ is valid. Since the set of true arithmetical sentences is not recursively enumerable, there cannot be a finitary, sound and complete formal system for the logic.

Remark. The extensionality axiom 15, Axiom 16 for **skip** and Axiom 17 for the sequential compositions are not used for the completeness [Theorem 21]. Since the axioms are valid, they must be derivable for hierarchical ASMs (as a consequence of the completeness theorem). Derivations of the axioms can be constructed as follows. For Axiom 17, one shows by induction on $|T_1| + \ldots + |T_n| + |\varphi|$ that

$$[R\,;S][T_1]\cdots[T_n]\varphi \leftrightarrow [R][S][T_1]\cdots[T_n]\varphi$$

is derivable. To reduce the number of cases, one can make the assumption that the atomic formulas in φ are $x = y$, $f(x) = y$, $\mathrm{upd}(R, f, x, y)$.

The extensionality axiom, $R \simeq S \rightarrow ([R]\varphi \leftrightarrow [S]\varphi)$, is then shown by induction on $|\varphi|$. In the case of a modal formula $[T]\psi$ one first shows (using the induction hypothesis), that $R \simeq S \rightarrow (R\,;T) \simeq (S\,;T)$ and obtains (again by the induction hypothesis) that $(R\,;T) \simeq (S\,;T) \rightarrow ([R\,;T]\psi \leftrightarrow [S\,;T]\psi)$. Using Axiom 17 one obtains that $R \simeq S \rightarrow ([R][T]\psi \leftrightarrow [S][T]\psi)$.

Axiom 16, $[\mathbf{skip}]\varphi \leftrightarrow \varphi$, is shown by induction on $|\varphi|$. In the case of a modal formula $[T]\psi$ one derives that $(\mathbf{skip}\,;T) \simeq T$ and obtains (using Axiom 17 and the extensionality axiom) $[\mathbf{skip}][T]\psi \leftrightarrow [\mathbf{skip}\,;T]\psi \leftrightarrow [T]\psi$.

7 An Extension with Explicit Step Information

For proving properties of ASMs it is often convenient to be able to speak about 'the function f in the nth state of the run'. For that purpose we extend the vocabulary Σ for each dynamic function name f by a new function name with one additional argument and write $f_n(t)$ instead of $f(n,t)$. We assume that an ASM is given and write $\mathfrak{A}_n$ for the nth state of the run of the ASM started in state $\mathfrak{A}$. (If the run is finite and n exceeds the length of the run, then $\mathfrak{A}_n$ is defined to be equal to the last state of the run.) The idea is that f_n is the function f in the nth state of the run.

We now work in two-sorted predicate logic. Each structure $\mathfrak{A}$ is extended by a copy of the structure of the natural numbers. The quantifiers $\forall x \in \mathbb{N}$ and $\exists x \in \mathbb{N}$ range over the set of natural numbers. Terms of the sort of natural numbers are denoted by ν. The grammar for formulas is extended by the following new atomic formulas:

$$\varphi, \psi \quad ::= \quad \ldots \mid \mathrm{def}_\nu(R) \mid \mathrm{upd}_\nu(R, f, s, t) \mid [R]_\nu \varphi$$

The semantics of the new atomic formulas is defined as follows, where $n = [\![\nu]\!]_\zeta^{\mathfrak{A}}$:

$$[\![\mathrm{def}_\nu(R)]\!]_\zeta^{\mathfrak{A}} := [\![\mathrm{def}(R)]\!]_\zeta^{\mathfrak{A}_n} \qquad [\![\mathrm{upd}_\nu(R, f, s, t)]\!]_\zeta^{\mathfrak{A}} := [\![\mathrm{upd}(R, f, s, t)]\!]_\zeta^{\mathfrak{A}_n}$$
$$[\![[R]_\nu\varphi]\!]_\zeta^{\mathfrak{A}} := [\![[R]\varphi]\!]_\zeta^{\mathfrak{A}_n}$$

For example, $\mathrm{def}_\nu(R)$ is true in the present state $\mathfrak{A}$, if the rule R is defined in the nth state of the run of the ASM starting in state $\mathfrak{A}$, where n is the value of the term ν. The new function names are interpreted in the obvious way:

$$f^{\mathfrak{A}}(n, a) := f^{\mathfrak{A}_n}(a), \quad \text{for each } n \in \mathbb{N} \text{ and } a \in |\mathfrak{A}|.$$

For formulas φ, not containing the new symbols with subscripts, we define a transformation $\varphi^{(\nu)}$ as follows:

$$\begin{array}{llll}
(s = t)^{(\nu)} & := s^{(\nu)} = t^{(\nu)} & ([R]\varphi)^{(\nu)} & := [R]_\nu\varphi \\
(\neg\varphi)^{(\nu)} & := \neg\varphi^{(\nu)} & \mathrm{def}(R)^{(\nu)} & := \mathrm{def}_\nu(R) \\
(\varphi \wedge \psi)^{(\nu)} & := \varphi^{(\nu)} \wedge \psi^{(\nu)} & \mathrm{upd}(R, f, s, t)^{(\nu)} & := \mathrm{upd}_\nu(R, f, s, t) \\
(\forall x\, \varphi)^{(\nu)} & := \forall x\, \varphi^{(\nu)} & &
\end{array}$$

Hence, $\varphi^{(\nu)}$ is obtained from φ by subscripting dynamic function names as well as the modal operators and the predicates def and upd in φ with ν. The subscript is neither applied inside modal formulas $[R]\varphi$ nor inside the atomic formulas except in equations. Inside the terms of equations it is applied everywhere:

$$x^{(\nu)} := x \qquad f(t)^{(\nu)} := \begin{cases} f_\nu(t^{(\nu)}), & \text{if } f \text{ is dynamic;} \\ f(t^{(\nu)}), & \text{otherwise.} \end{cases}$$

As expected, the value of a term $t^{(\nu)}$ is the value of t in the nth state of the run of the ASM and the formula $\varphi^{(\nu)}$ is equivalent to the formula φ in the nth state, where n is the value of the term ν.

Lemma 23. *If* $n = [\![\nu]\!]^{\mathfrak{A}}_{\zeta}$, *then* $[\![t^{(\nu)}]\!]^{\mathfrak{A}}_{\zeta} = [\![t]\!]^{\mathfrak{A}_n}_{\zeta}$ *and* $[\![\varphi^{(\nu)}]\!]^{\mathfrak{A}}_{\zeta} = [\![\varphi]\!]^{\mathfrak{A}_n}_{\zeta}$.

We write $\mathrm{Con}_\nu(R)$ for $\mathrm{Con}(R)^{(\nu)}$ and $\mathrm{inv}_\nu(R, f, x)$ for $\mathrm{inv}(R, f, x)^{(\nu)}$.

The basic logic of [Section 4] is extended by the following principles (M is the main rule name of the ASM). First we add appropriate axioms for the structure of natural numbers including the induction scheme 54. Axiom 55 asserts that $\varphi^{(0)}$ is equivalent to φ, since the functions f_0 are, by definition, the same as the functions in the initial state. Axioms 56 and 57 characterize $\varphi^{(\nu+1)}$ depending on whether the main ASM is consistent or not in state ν. If M is not consistent in state ν, then by definition the state $\nu + 1$ is the same as state ν. The rule 58 allows to transfer axioms and theorems of the basic system into statements about the νth state of a run. Hence, we do not need new axioms for $\mathrm{def}_\nu(R)$ and $\mathrm{upd}_\nu(R, f, x, y)$, since we can transfer the axioms D1–D9 and U1–U9 using rule 58.

X. Axioms for the structure of natural numbers: For example, the Peano Axioms including the induction scheme:

54. $\varphi(0) \wedge \forall x \in \mathbb{N}\, (\varphi(x) \rightarrow \varphi(x + 1)) \rightarrow \forall x \in \mathbb{N}\, \varphi(x)$

XI. Axioms and rules for formulas with step information:

55. $\varphi^{(0)} \leftrightarrow \varphi$
56. $\mathrm{Con}_\nu(M) \rightarrow (\varphi^{(\nu+1)} \leftrightarrow [M]_\nu \varphi)$
57. $\neg\mathrm{Con}_\nu(M) \rightarrow (\varphi^{(\nu+1)} \leftrightarrow \varphi^{(\nu)})$
58. $\dfrac{\varphi}{\varphi^{(\nu)}}$

Why is Rule 58 sound?—Assume that φ is derivable. We want to show that $\varphi^{(\nu)}$ is valid. Let $\mathfrak{A}$ be a state and n be the value of ν in $\mathfrak{A}$. Since φ is valid, it is true in the nth state $\mathfrak{A}_n$ of the run starting with $\mathfrak{A}$. By [Lemma 23], it follows that $\varphi^{(\nu)}$ is true in $\mathfrak{A}$.

The following rules are derivable from 55–58 using the induction scheme 54. The rule on the left-hand side says that, if φ is an invariant of M and if it is true in the starting state, then it is true in all states of the run. The rule on the right-hand side says that, if φ is an invariant which ensures the consistency of M and φ is true in the starting state, then M is consistent in all steps of the run.

$$\frac{\varphi \to [M]\varphi}{\varphi \to \forall x \in \mathbb{N}\, \varphi^{(x)}} \qquad \frac{\varphi \to \mathrm{Con}(M) \wedge [M]\varphi}{\varphi \to \forall x \in \mathbb{N}\, \mathrm{Con}_x(M)}$$

The following characterizing formulas for f_ν are derivable as well:

59. $\forall x\, (f_0(x) = f(x))$
60. $\mathrm{Con}_\nu(M) \to \forall x, y (f_{\nu+1}(x) = y \leftrightarrow \mathrm{upd}_\nu(M, f, x, y) \vee (\mathrm{inv}_\nu(M, f, x) \wedge f_\nu(x) = y))$
61. $\neg\mathrm{Con}_\nu(M) \to \forall x\, (f_{\nu+1}(x) = f_\nu(x))$

In the extended system, several properties of an ASM can be expressed in a simple way (where φ_{init} and φ_{stop} are formulas characterizing initial and halting states of the ASM):

– The ASM is consistent: $\varphi_{\mathrm{init}} \to \forall x \in \mathbb{N}\, \mathrm{Con}_x(M)$.

– The ASM terminates: $\varphi_{\mathrm{init}} \to \exists x \in \mathbb{N}\, \varphi_{\mathrm{stop}}^{(x)}$.

– The formula ψ is an invariant of the ASM: $\varphi_{\mathrm{init}} \to \forall x \in \mathbb{N}\, \psi^{(x)}$.

If the ASM does not use sequential compositions, then the statements are equivalent to first-order formulas without modal operators.

Acknowledgments

We are grateful to Egon Börger and G. R. Renardel de Lavalette for helpful comments on an earlier version of the article.

References

[Börger and Huggins 1998] E. Börger and J. Huggins. Abstract State Machines 1988–1998: Commented ASM Bibliography. *Bulletin Europ. Assoc. Theoret. Comp. Science*, 64:105–127, 1998. Updated bibliography available at `http://www.eecs.umich.edu/gasm`.

[Börger and Schmid 2000] E. Börger and J. Schmid. Composition and submachine concepts. In P. Clote and H. Schwichtenberg, editors, *Computer Science Logic (CSL 2000)*, pages 41–60. Springer-Verlag, Lecture Notes in Computer Science 1862, 2000.

[Gargantini and Riccobene 2000] A. Gargantini and E. Riccobene. Encoding abstract state machines in PVS. In Y. Gurevich, P. Kutter, M. Odersky, and L. Thiele, editors, *Abstract State Machines: Theory and Applications*, pages 303–322. Springer-Verlag, Lecture Notes in Computer Science 1912, 2000.

[Groenboom and Renardel de Lavalette 1995] R. Groenboom and G. R. Renardel de Lavalette. A formalization of evolving algebras. In *Proceedings of Accolade 95*. Dutch Research School in Logic, 1995.

[Gurevich 1993] Y. Gurevich. Evolving algebras 1993: Lipari guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1993.

[Harel 1983] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, pages 497–604. Reidel, Dordrecht, 1983.

[Harel et al. 2000] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* The MIT Press, 2000.

[Poetzsch-Heffter 1994] A. Poetzsch-Heffter. Deriving partial correctness logics from evolving algebras. In B. Pehrson and I. Simon, editors, *IFIP 13th World Computer Congress*, volume I: Technology/Foundations, pages 434–439, Elsevier, Amsterdam, 1994.

[Renardel de Lavalette 2001] G. R. Renardel de Lavalette. A logic of modification and creation. In C. Condoravdi and G. R. Renardel de Lavalette, editors, *Logical Perspectives on Language and Information.* CSLI publications, Stanford, CA, 2001.

[Schellhorn 1999] G. Schellhorn. *Verification of Abstract State Machines.* PhD thesis, Universität Ulm, 1999.

[Schellhorn and Ahrendt 1997] G. Schellhorn and W. Ahrendt. Reasoning about abstract state machines: the WAM case study. *J. of Universal Computer Science*, 3(4):377–413, 1997.

[Schmitt 1994] P. H. Schmitt. Proving WAM compiler correctness. Technical Report IRATR–1994–33, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, 1994.

[Schönegge 1995] A. Schönegge. Extending dynamic logic for reasoning about evolving algebras. Technical Report IRATR–1995–49, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, 1995.

[Shankar 2000] N. Shankar. Symbolic analysis of transition systems. In Y. Gurevich, P. Kutter, M. Odersky, and L. Thiele, editors, *Abstract State Machines: Theory and Applications*, pages 287–302. Springer-Verlag, Lecture Notes in Computer Science 1912, 2000.

[Stärk and Nanchen 2001] R. F. Stärk and S. Nanchen. A logic for abstract state machines. In L. Fribourg, editor, *Computer Science Logic (CSL 2001)*, pages 217–231. Springer-Verlag, Lecture Notes in Computer Science 2142, 2001.

[Stärk et al. 2001] R. F. Stärk, J. Schmid, and E. Börger. *Java and the Java Virtual Machine—Definition, Verification, Validation.* Springer-Verlag, 2001.

[van Benthem and Bergstra 1995] J. van Benthem and J. A. Bergstra. Logic of transition systems. *J. of Logic, Language, and Information*, 3(4):247–283, 1995.