The ASMKeY Theorem Prover

Stanislas Nanchen, Hubert Schmid, Peter H. Schmitt and Robert Stärk

ETH Zurich
Department of Computer Science

Abstract. In [8], we presented a logic for Abstract State Machines (ASMs). In this technical report, we present an interactive theorem prover call ASMKeY where we implemented this logic. We also present a small case study to show the effectivness of our approach.

1 Introduction

Abstract State Machines [4] (ASMs) are widely used as specification method for software, algorithms, protocols and as operational semantic for many programming languages. [2] is a thorough presentation of ASMs and their applications.

In [8], we presented a logic for ASMs for expressing and proving properties of them. We will call this logic the *basic logic for ASMs*.

In this report, we present ASMKeY, a interactive theorem proved based on the KeY system [1]. The KeY system is a proving environment integrated in the CASE-tool Together from Borland; the KeY project is aimed into providing a complete system for specification and verification of object oriented programs in particular Java-Card programs. KeY is a project of the University of Karlsruhe in Germany and Chalmers University of Technology in Sweden.

The report is organized as follow. After the introduction, we give a reminder of the basic logic for ASMs and then introduce for it a sequent calculus version.

In section 3, we first discuss the ASMKeY prover and its relation with KeY; in particular, we expose the *taclet* language which allows to express the sequent calculus for use with the interactive prover.

Then we discuss automatisation and simplification technics for the proofs. We move then on a case study, the correctness of the MergeSort algorithm, before concluding on the present and future works.

1.1 Related Works

In his thesis [6], Schellhorn proofs the correctness of the refinements from the Prolog language to the WAM virtual machine. To achieve that, the ASMs are first translated into the programs of Dynamic Logic (DL) : the dynamic functions are represented by association lists, the parallel rules are sequentialized and the if-then-else statements are flattened. The properties on ASMs are then expressed as DL formulas and proved with the KIV theorem prover.

In [3], Gargantini and Ricobene show how to use the PVS theorem prover for verifying ASMs. Functions are represented by PVS functions and a interpreter for ASMs is programed in PVS; in particular the parallel (block) rule is sequentialized. It is possible to retain the abstraction level of the ASMs by giving properties of PVS functions instead of defining them.

In her thesis [10], Winter applies to ASMs the other widely used verification technique, namely Model Checking. The ASMs are encoded in the languages of several Model Checking tools. In particular, the transformation algorithm unfold the dynamic functions with n arguments into dynamic variables and flatten the if-then-else structures. Properties are given as temporal formulas.

2 A Logical Calculus for ASMs

Before the presentation of the sequent caluculus, we briefly introduce the ASMs and quickly recall the logic of [8].

2.1 ASM rules and update sets

ASMs are transition systems whose states are *abstract states*. The notion of abstract state is the classical notion of first-order structure $\mathcal{A}$ over a vocabulary Σ. A structure $\mathcal{A}$ over a vocabulary Σ consists of a underlying set $|\mathcal{A}|$, called the superuniverse of the structure, and the interpretation of the function symbols of Σ over $|\mathcal{A}|$. The interpretation of terms s, t and of first-order formulas φ, ψ is done the standart way with respect to a variable assignement ζ.

The functions of the vocabulary Σ are classified in two categories, the static and the dynamic functions. As their name indicates, the static ones do not change during the computation where the dynamic ones are updated during the computation. The updates of dynamic functions at each step of the computation are expressed with transition rules.

1. *Skip Rule:* **skip**
   Meaning: Do nothing.
2. *Update Rule:* $f(s) := t$
   Syntactic condition: f is a dynamic function name of Σ
   Meaning: In the next state, the value of f at s is updated to t.
3. *Block Rule:* P **par** Q
   Meaning: P and Q are executed in parallel.
4. *If Rule:* **if** φ **then** P **else** Q
   Meaning: If φ is true, then execute P, otherwise execute Q.
5. *Let Rule:* **let** $x = t$ **in** P
   Meaning: Assign the value of t to x and execute P.
6. *Forall Rule:* **forall** x **with** φ **do** P
   Meaning: Execute P in parallel for each x satisfying φ.
7. *Sequence Rule:* P **seq** Q
   Meaning: P and Q are executed sequentially, first P and then Q.

8. *Try Rule:* **try** P **else** Q
   Meaning: If P is consistent, execute P, else execute Q.
9. *Call Rule:* $\rho(t)$
   Meaning: Call transition rule $\rho(x) = P$ with parameters t.

Definition 1 (ASM). An *abstract state machine* M consists of a vocabulary Σ, an initial state $\mathcal{A}$ 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 of ASMs are given by sets of updates. Because the presence of the Block and Forall rules, a ASM may try to fire updates for a dynamic function f to the same argument a but with different values, which is clearly inconstitent.

To capture the notion of consistency, we need to define what updates and set of updates are.

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

The meaning of an update is that the interpretation of the dynamic function f has, for the next state, to be changed at point a with new value b. A update set is simply a set of such updates.

Intuitively, an update set is consistent if its updates do not clash; more precisely, 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 $b = c$.

The semantic of rules are given by update sets. We write $\mathsf{yields}(P, \mathcal{A}, \zeta, U)$ to mean 'rule P yields in state $\mathcal{A}$ under the variable assignment ζ the update set U'. The table 1 gives, in a inductive manner, the properties of the yields relation. The yields relation is the least relation satisfying these properties. Also notice that, in the presence of recursive rule definitions, a rule M may induce an infinite loop and therefore have no update set i.e., be undefined.

2.2 A logic for ASMs

To formalize the consistency and other properties of ASMs, we extend the language of first-order logic with a modal operator $[P]\varphi$ and two new predicates $\mathsf{def}(P)$ and $\mathsf{upd}(P, f, x, y)$.

The meaning of the modal operator $[P]\varphi$ is that the formula φ is true after firing P. The predicates $\mathsf{def}(P)$ and $\mathsf{upd}(P, f, x, y)$ asserts that the rule P is defined, respectively than the rule P produces an update for the dynamic function f at point x with new value y.

The formulas of the logic for ASMs are generated by the following grammar:

$$\begin{aligned}\varphi, \psi \ ::= \ & s = t \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \forall x\, \varphi \mid \exists x\, \varphi \mid \\ & \mathsf{def}(P) \mid \mathsf{upd}(P, f, s, t) \mid [P]\varphi\end{aligned}$$

A formula is called *pure* (or *first-order*), if it contains neither the predicate 'def' nor 'upd' nor the modal operator $[P]$. A formula is called *static*, if it does not

$\dfrac{}{\mathsf{yields}(\mathbf{skip}, \mathcal{A}, \zeta, \emptyset)}$	
$\dfrac{}{\mathsf{yields}(f(s) := t, \mathcal{A}, \zeta, \{(l, v)\})}$	where $l = (f, (\llbracket s \rrbracket_\zeta^\mathcal{A}))$ and $v = \llbracket t \rrbracket_\zeta^\mathcal{A}$
$\dfrac{\mathsf{yields}(P, \mathcal{A}, \zeta, U) \quad \mathsf{yields}(Q, \mathcal{A}, \zeta, V)}{\mathsf{yields}(P \ \mathbf{par}\ Q, \mathcal{A}, \zeta, U \cup V)}$	
$\dfrac{\mathsf{yields}(P, \mathcal{A}, \zeta, U)}{\mathsf{yields}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q, \mathcal{A}, \zeta, U)}$	if $\llbracket \varphi \rrbracket_\zeta^\mathcal{A} = \mathit{true}$
$\dfrac{\mathsf{yields}(Q, \mathcal{A}, \zeta, V)}{\mathsf{yields}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q, \mathcal{A}, \zeta, V)}$	if $\llbracket \varphi \rrbracket_\zeta^\mathcal{A} = \mathit{false}$
$\dfrac{\mathsf{yields}(P, \mathcal{A}, \zeta[x \mapsto a], U)}{\mathsf{yields}(\mathbf{let}\ x = t\ \mathbf{in}\ P, \mathcal{A}, \zeta, U)}$	where $a = \llbracket t \rrbracket_\zeta^\mathcal{A}$
$\dfrac{\mathsf{yields}(P, \mathcal{A}, \zeta[x \mapsto a], U_a) \quad \text{for each } a \in I}{\mathsf{yields}(\mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ P, \mathcal{A}, \zeta, \bigcup_{a \in I} U_a)}$	where $I = \mathsf{range}(x, \varphi, \mathcal{A}, \zeta)$
$\dfrac{\mathsf{yields}(P, \mathcal{A}, \zeta, U) \quad \mathsf{yields}(Q, \mathcal{A} + U, \zeta, V)}{\mathsf{yields}(P\ \mathbf{seq}\ Q, \mathcal{A}, \zeta, U \oplus V)}$	if U is consistent
$\dfrac{\mathsf{yields}(P, \mathcal{A}, \zeta, U)}{\mathsf{yields}(P\ \mathbf{seq}\ Q, \mathcal{A}, \zeta, U)}$	if U is inconsistent
$\dfrac{\mathsf{yields}(P\frac{t}{x}, \mathcal{A}, \zeta, U)}{\mathsf{yields}(r(t), \mathcal{A}, \zeta, U)}$	where $r(x) = P$ is a rule declaration of M

Table 1. Semantic of rules: yields predicate

contain dynamic function names. The formulas used in If-Then-Else and Forall rules must be pure formulas.

The semantic of the formulas given a state $\mathcal{A}$ and a variable assignement ζ is the usual one for the first-order formulas; the semantic for the new modal operator and predicates is given in table 2.

The basic properties for def and upd are listed in the table table 3 and table 4. Along with those properties, the following principles will be the basic axioms and rules of our logic $\mathcal{L}(M)$, for an abstract state machine M, the logic being parametrized by M.

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. $[P](\varphi \to \psi) \wedge [P]\varphi \to [P]\psi$
4. $\dfrac{\varphi}{[P]\varphi}$
5. $\neg\mathsf{Con}(P) \to [P]\varphi$
6. $\neg[P]\varphi \to [P]\neg\varphi$

IV. The Barcan axiom:

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

V. Axioms for pure static formulas:

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

VI. Axioms for def and upd:

10. D1–D9 in Table 3
11. U1–U9 in Table 4

VII. Update axioms for transition rules:

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

VIII. Extensionality axiom for transition rules:

15. $P \simeq Q \to ([P]\varphi \iff [Q]\varphi)$

$$\llbracket\,[P]\varphi\,\rrbracket_\zeta^\mathcal{A} := \begin{cases} \mathit{true}, & \text{if } \llbracket\varphi\rrbracket_\zeta^{\mathcal{A}+U} = \mathit{true} \text{ for each consistent } U \\ & \text{with } \mathsf{yields}(P,\mathcal{A},\zeta,U); \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$\llbracket\mathsf{def}(P)\rrbracket_\zeta^\mathcal{A} := \begin{cases} \mathit{true}, & \text{if there exists an update set } U \text{ with } \mathsf{yields}(P,\mathcal{A},\zeta,U); \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

$$\llbracket\mathsf{upd}(P,f,s,t)\rrbracket_\zeta^\mathcal{A} := \begin{cases} \mathit{true}, & \text{if ex. } U \text{ with } \mathsf{yields}(P,\mathcal{A},\zeta,U) \text{ and} \\ & \langle f, \llbracket s\rrbracket_\zeta^\mathcal{A}, \llbracket t\rrbracket_\zeta^\mathcal{A}\rangle \in U; \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

Table 2. The semantics of modal formulas and basic predicates.

D1.	$\mathsf{def}(\mathbf{skip})$	
D2.	$\mathsf{def}(f(s) := t)$	
D3.	$\mathsf{def}(P\ \mathbf{par}\ Q) \leftrightarrow \mathsf{def}(P) \wedge \mathsf{def}(Q)$	
D4.	$\mathsf{def}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q) \iff (\varphi \wedge \mathsf{def}(P)) \vee (\neg\varphi \wedge \mathsf{def}(Q))$	
D5.	$\mathsf{def}(\mathbf{let}\ x = t\ \mathbf{in}\ P) \iff \exists x\,(x = t \wedge \mathsf{def}(P))$	if $x \notin \mathrm{FV}(t)$
D6.	$\mathsf{def}(\mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ P) \iff \forall x\,(\varphi \rightarrow \mathsf{def}(P))$	
D7.	$\mathsf{def}(P\ \mathbf{seq}\ Q) \iff \mathsf{def}(P) \wedge [P]\mathsf{def}(Q)$	
D8.	$\mathsf{def}(\mathbf{try}\ P\ \mathbf{else}\ Q) \iff \mathsf{def}(P) \wedge (\mathsf{Con}(P) \vee \mathsf{def}(Q))$	
D9.	$\mathsf{def}(\rho(t)) \iff \mathsf{def}(P\frac{t}{x})$	if $\rho(x) = P$ is a rule definition of M

Table 3. Axioms for definedness.

U1. $\neg\mathsf{upd}(\mathbf{skip}, f, x, y)$

U2. $\mathsf{upd}(f(s) := t, f, x, y) \iff s = x \wedge t = y, \quad \neg\mathsf{upd}(f(s) := t, g, x, y) \quad$ if $f \neq g$

U3. $\mathsf{upd}(P \;\mathbf{par}\; Q, f, x, y) \iff \mathsf{def}(P \;\mathbf{par}\; Q) \wedge (\mathsf{upd}(P, f, x, y) \vee \mathsf{upd}(Q, f, x, y))$

U4. $\mathsf{upd}(\mathbf{if}\; \varphi \;\mathbf{then}\; P \;\mathbf{else}\; Q, f, x, y) \iff (\varphi \wedge \mathsf{upd}(P, f, x, y)) \vee (\neg\varphi \wedge \mathsf{upd}(Q, f, x, y))$

U5. $\mathsf{upd}(\mathbf{let}\; z = t \;\mathbf{in}\; P) \iff \exists z\, (z = t \wedge \mathsf{upd}(P, f, x, y)) \quad$ if $z \notin \mathrm{FV}(t)$

U6. $\mathsf{upd}(\mathbf{forall}\; z \;\mathbf{with}\; \varphi \;\mathbf{do}\; P, f, x, y) \iff$
$\mathsf{def}(\mathbf{forall}\; z \;\mathbf{with}\; \varphi \;\mathbf{do}\; P) \wedge \exists z\, (\varphi \wedge \mathsf{upd}(P, f, x, y))$

U7. $\mathsf{upd}(P \;\mathbf{seq}\; Q, f, x, y) \iff$
$(\mathsf{upd}(P, f, x, y) \wedge [P](\mathsf{def}(Q) \wedge \mathsf{inv}(Q, f, x))) \vee$
$(\mathsf{Con}(P) \wedge [P]\mathsf{upd}(Q, f, x, y))$

U8. $\mathsf{upd}(\mathbf{try}\; P \;\mathbf{else}\; Q, f, x, y) \iff$
$(\mathsf{Con}(P) \wedge \mathsf{upd}(P, f, x, y)) \vee$
$(\mathsf{def}(P) \wedge \neg\mathsf{Con}(P) \wedge \mathsf{upd}(Q, f, x, y))$

U9. $\mathsf{upd}(\rho(t), f, x, y) \iff \mathsf{upd}(P\frac{t}{z}, f, x, y) \quad$ if $\rho(z) = P$ is a rule definition of M

Table 4. Axioms for updates.

IX. Axioms from dynamic logic:

16. $[\mathbf{skip}]\varphi \iff \varphi$
17. $[P \;\mathbf{seq}\; Q]\varphi \iff [P][Q]\varphi$

It is possibly to show in the system the following properties for the consistency predicate $\mathsf{Con}(P)$:

Lemma 1. *The following consistency properties are derivable:*

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

2.3 A sequent calculus for ASMs

To get a sequent calculus for ASMs, we translate the basic logic, which has a Hilbert-style, to a sequent-style.

The First-Order part of the logic is translated into the classical sequent calculus without forgetting the restriction of the quantifier rules.

The translation of the axioms for the main predicates def and upd is using the following schema.

Let $\forall\bar{x}(\Phi(\bar{x}) \leftrightarrow \Psi(\bar{x}))$ be an axiom of the Hilbert-style caluculus, we then introduce the two following rules (with $\bar{t} staticterms$:

$$\frac{\Gamma, \Psi(\bar{t}) \Rightarrow \Delta}{\Gamma, \Phi(\bar{t}) \Rightarrow \Delta}\ \text{left} \qquad \frac{\Gamma \Rightarrow \Psi(\bar{t}), \Delta}{\Gamma \Rightarrow \Phi(\bar{t}), \Delta}\ \text{right}$$

The translation of the Step Rule is a little more tricky, as it is required that the right side is not empty; for Δ_2 not empty and Ψ pure and static formulas;

$$\frac{\Gamma_2, \Psi \Rightarrow \Delta_2}{\Gamma_1, [P]\Gamma_2, \Psi \Rightarrow \Delta_1, [P]\Delta_2}$$

We can see that the pure and static assumptions Ψ are preserved during the step.

The rest of the translation is rather straightforward. The whole sequent calculus is rather large and the reader can find it in the appendix A. We already know the Step Rule; we give two more rules as example.

Definition 3 (Rules).

step *For Δ_2 not empty and Ψ pure and static formulas;*

$$\frac{\Gamma_2, \Psi \Rightarrow \Delta_2}{\Gamma_1, [P]\Gamma_2, \Psi \Rightarrow \Delta_1, [P]\Delta_2}$$

all-left *(Axiom 10.D4) For t static or φ pure*

$$\frac{\Gamma, \forall x\, \varphi, \varphi\frac{t}{x} \Rightarrow \Delta}{\Gamma, \forall x\, \varphi \Rightarrow \Delta}$$

upd-par-right *(Axiom 11.U3)*

$$\frac{\Gamma \Rightarrow \mathsf{def}(P\ \mathbf{par}\ Q) \wedge (\mathsf{upd}(P, f, t, s) \vee \mathsf{upd}(Q, f, t, s)), \Delta}{\Gamma \Rightarrow \mathsf{upd}(P\ \mathbf{par}\ Q, f, t, s), \Delta}$$

2.4 Equivalence

Theorem 1 (Equivalence). *The Hilbert-style calculus $\mathcal{L}(M)$ is equivalent to the sequent calculus $\mathcal{S}(M)$ in the sense that $\mathcal{L}(M) \vdash \varphi$ if and only if $\mathcal{S}(M) \vdash \varphi$.*

Proof. The proof is given by a transformation algorithm from a proof tree of one calculus in a proof tree of the other and vice-versa. The proof is rather long and technical, so it is not given here. The result is well-known for First-Order Logic and this proof uses the same method. The method used can be found in [9]. □

3 KeY and ASMKeY

In this section, we present the ASMKeY interactive theorem prover (the tool). Before presenting ASMKeY, we have first a look at the KeY theorem prover and introduce the *taclet* language used to write the sequent calculus rules of the logic.

3.1 The KeY theorem prover

The KeY theorem prover is only a part of the KeY System. However the rest of the KeY system (e.g. OCL to dynamic logic translator) has little relevance for us. The KeY theorem prover is invoked when a user want to prove a properties of a method or a class invariant. It present itself as a window with two distinguised part : a tree viewer for presenting the proof tree and a view for presenting a sequent. As ASMKeY conserves almost the same interface, the screenshot on fig. 1 gives a fair impression of what the KeY theorem prover looks like.

The proving is done by clicking on a term or formula in the presented sequent. Then a pop menu appears presenting the available rule applications that are possible for the selected term or formula. If the user chooses such a rule, then the rule is applied to the sequent and one or more new goals are added to the proof tree.

3.2 Taclets

KeY is implemented in Java. But most of the rules of its logic are implemented using the *taclet* language. Taclets (from tactic and applet, litterally *lightweight tactics*) describe the way the rule is applied to the current goal. We do not want to describe here the taclet language in its totality, but concentrate on its main features. For a thorough introduction of the tactics language, the reader is referred to [5].

To illustrate the taclet language, we give the taclet equivalent of the rules in definition 3 in the table 5. To further illustrate the taclet language, we also reproduce the rule for the conjunction on the right part of the sequent.

The statment `find` precise the pattern of the taclet which must match with the principal formula on the current goal in order to be applicable; the `==>` indicates wether the principal formula is on the left part or on the right part of the current goal. The statment `varcond` precises additional conditions, for instance in the `all_left_static`, the term `#t` we want to instanciate with must be static.

The `add` and `replacewith` statments describe how new goal(s) must be created. With `add`, the next goal simply receive a new formula; with `replacewith` the principal formula is replaced. When many `add` and `replacewith` statments are present, the goals splits in new subgoals, one per statment.

It is not possible to express the Step rule in the taclet language. The Step rule is directly implemented in ASMKeY (in Java) and is presented to the user as a built-in rule.

```
all_left_static {
    find (all #x. #phi ==>)
    varcond (static #t)
    add ({#x #t} #phi ==>)
};

upd_par_right {
    find (==> upd(#t <- #s in #R, #S))
    varcond (dynamic #t, staticargs #t, static #s)
    replacewith (==> def(#R, #S) &
                     (upd(#t <- #s in #R) | upd(#t <- #s in #S)));
};

and_right {
    find (==> #phi & #psi)
    replacewith (==> #phi);
    replacewith (==> #psi)
}
```

Table 5. Some taclets

3.3 The ASMKeY theorem prover

The ASMKeY theorem prover is based on the KeY theorem prover and shares almost all of its base code.

As in KeY, the proof process is done by clicking on formula in a sequent and choosing a appropriate rule to simplify or close a goal. The figure 1 is a screenshot of the ASMKeY theorem prover in action.

4 Simplification and Automatisation of Proof

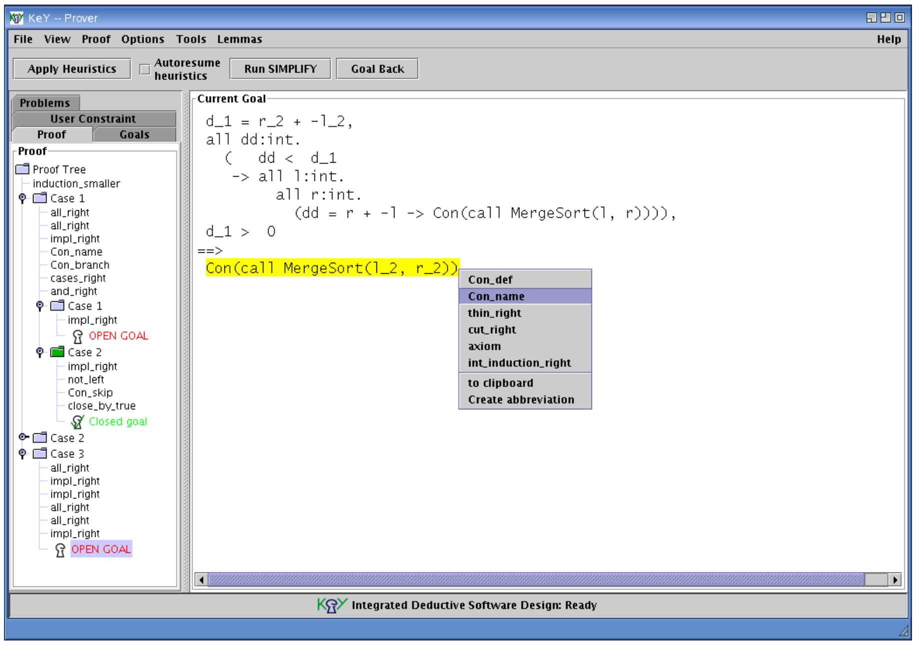
The calculus presented in the previous section is low-level and as such ill-adapted for proving properties on a large scale.

In this section, we present features we have developped or are developping to improve the usability of the logic:

1. Taclets for Con and other derived predicates (implemented);
2. Simplified Taclets (in progress);
3. Heuristics (in progress).

4.1 Taclets for Con and other derived predicates

In the basic logic, derived predicates (especially the consistency Con) play an important role in the axioms already (see [8]). Furthermore, these derived predicates plays an important role in correctness properties.



```
Proof
Proof Tree
  induction_smaller
  Case 1
    all_right
    all_right
    impl_right
    Con_name
    Con_branch
    cases_right
    and_right
    Case 1
      impl_right
      OPEN GOAL
    Case 2
      impl_right
      not_left
      Con_skip
      close_by_true
      Closed goal
  Case 2
  Case 3
    all_right
    impl_right
    impl_right
    all_right
    all_right
    impl_right
    OPEN GOAL
```

```
 d_1 = r_2 + -l_2,
 all dd:int.
   (   dd <  d_1
    -> all l:int.
         all r:int.
           (dd = r + -l -> Con(call MergeSort(l, r)))),
 d_1 >  0
==>
 Con(call MergeSort(l_2, r_2))
```

Fig. 1. A screenshot of ASMKeY

The consistency predicate Con is the more important of them, as proving the consistency of an ASM M ensure the terminaison (of one step) of the program. Also, as one generally includes consistency in the notion of correctness, the consistency of a program must virtually always be proved. The invariant predicate inv is very important when proving update properties in the presence of the **seq** rule construct.

Therefore, we introduced taclets for directly using properties of Con, inv and other properties found in [8]. This allow the user to directly proof those properties without using the definition of the predicates. You can found the sequent rules in the appendix B.

These taclets are not primitive in the sense that they do not come from the axioms of the logic but from some of its properties. Unfortunately, at the time of this writting, it is impossible in ASMKeY to prove a taclet from more primitive taclets. However, we were able to prove them in ASMKeY under their equivalence form for anonymous ASM rules.

4.2 Simplified Taclets

As previously stated, the consistency of an ASM is very often considered to be part of its correctness. Together with the definedness def, the consistency of an ASM will be often proved very early in the correctness proof of this ASM.

Therefore, it might be usefull to take in account that the consistency of an ASM has already been proved when proving subsequent propreties.

As example, take $M = P$ **par** Q, suppose we want to prove $\mathsf{upd}(M, f, x, y)$. The rule for proving $\mathsf{upd}(M, f, x, y)$ is :

$$\frac{\Gamma \Rightarrow \mathsf{def}(P\ \mathbf{par}\ Q) \wedge (\mathsf{upd}(P, f, t, s) \vee \mathsf{upd}(Q, f, t, s)), \Delta}{\Gamma \Rightarrow \mathsf{upd}(M, f, t, s), \Delta}\ \text{upd-par-right}$$

But, if we know that consistency of M holds, we could get rid of the $\mathsf{def}(M)$ as consistency implies definedness. Furthermore, we could forward the consistency information to subgoals for later use. The simplified rule for upd in the case would look like :

$$\frac{\Gamma, \mathsf{Con}(P) \wedge \mathsf{Con}(Q) \wedge \mathsf{joinable}(P, Q) \Rightarrow \mathsf{upd}(P, f, t, s) \vee \mathsf{upd}(Q, f, t, s), \Delta}{\Gamma, \mathsf{Con}(M) \Rightarrow \mathsf{upd}(M, f, t, s), \Delta}$$

Notice that we do not have to prove $\mathsf{def}(M)$ anymore.

We are currently extending the taclet language used in ASMKeY in order to write such rules.

4.3 Heuristics

Heuristics are already a part of the KeY theorem prover. A heuristic is a list of taclets that may be applied automatically by the prover. By prioritazing the heuristics and by carefully composing them, it is possible to achieve a great automatisation of the proof.

As stated above, the heuristics must be carefully composed in order to get good results; otherwise, we could end up during a proof in a very complicate goal for which we have no idea from where it comes.

We are currently testing some basic heuristics; more elaborated heuristic will come only when we have done enough case studies.

5 MergeSort.

As a first case study for the ASMKeY theorem prover, we use the MergeSort algorithm as it is a simple and well-known example, yet not trivial.

We reproduce in table 6 the algorithm in its ASM version using parallel updates as much as possible.

We summerize below the four properties one has to prove in order to verify the correctness of the MergeSort algorithm. For all values of l and r,

1. The rule $\textsc{MergeSort}(l, r)$ makes updates within its range.

$$\forall i\, (\exists y\, \mathsf{upd}(\textsc{MergeSort}(l, r), f, i, y) \to (l \leq i \leq r))$$

```
MergeSort$(l, r)$ =
  if $l < r$ then
    let $m = \lfloor (l + r)/2 \rfloor$ in
      (MergeSort$(l, m)$ par MergeSort$(m + 1, r)$) seq
      Merge$(l, m, r)$

Merge$(l, m, r)$ =
  (forall $i$ with $l \leq i \leq r$ do $g(i) := f(i)$) seq
  MergeCopy$(l, m, m + 1, r, l)$

MergeCopy$(i, m, j, r, k)$ =
  if $k \leq r$ then
    if $(i \leq m$ && $j \leq r$ && $g(i) \leq g(j))$ || $(r < j)$ then
      $f(k) := g(i)$ par MergeCopy$(i + 1, m, j, r, k + 1)$
    else
      $f(k) := g(j)$ par MergeCopy$(i, m, j + 1, r, k + 1)$
```

Table 6. The MergeSort algorithm

2. The rule MergeSort(l, r) is consistent.

$$\mathsf{Con}(\textsc{MergeSort}(l, r))$$

3. After firing the rule MergeSort(l, r), the array is sorted.

$$[\textsc{MergeSort}(l, r)]\forall i\, ((l \leq i < r) \rightarrow f(i) \leq f(i + 1))$$

4. The sorted array is a permutation of the (old) unsorted array.

$$\exists \pi \in \mathit{Perm}([l \ldots r]) \\ \quad \forall i\, (i \in [l \ldots r] \rightarrow \mathsf{upd}(\textsc{MergeSort}(l, r), f, i, f(\pi(i))))$$

We were able to fully prove in ASMKeY the three first properties of the correctness. We are still working on ASMKeY to mechanically prove the fourth as it requires to manipulate *permutations* directly. This is currently not possible in ASMKeY.

6 Future work and extensions

As presented in this report, ASMKeY is still in its infancy and will need subsequent development before doing large case studies. In this section, we present three extensions that we think necessary to achieve larger case studies.

6.1 Basic lemma managment

The ability to reuse what has already been done is one of the common features of computer systems in general and interactive theorem provers are no exceptions:

the possibility to define lemmas and theorems to be reused in other proofs is therefore mandatory.

We are currently working on a basic lemma managment tool; the main difficulty in the programming of such tool is to ensure the non-circularity of the dependancy of the lemmas.

6.2 Access properties and sequentialisation of parallel programs

In [7], we have worked on an extension of the basic logic to handle access (read) properties. In particular, the logic allows the formulation and verification of security properties like ensuring that certain locations are not accesssed. In addition, it allows also the verification of sequentialisation of parallel programs.

We are currently in the process of adding the necessary predicates and taclets to ASMKeY and will soon begin with the first tests.

6.3 Run properties and explicit step logic

Is [8], we presented also an explicit step extension to the basic logic. With this extension, it is possible to express run properties of ASM, a feature that the basic logic is missing. We are currently working on the integration of the explicit step extension in ASMKeY.

7 Conclusion

We have presented in this technical report the actual state of the ASMKeY interactive prover along with the future challenges in order to make the tool more usable for mid-size case studies.

References

1. W. Ahrendt, T. Baar, B. Beckert, R. Bubel, M. Giese, R. Hähnle, W. Menzel, W. Mostowski, A. Roth, S. Schlager, and P. H. Schmitt. The KeY tool.
2. E. Börger and R. F. Stärk. *Abstract State Machines—A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.
3. 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.
4. Y. Gurevich. Evolving algebras 1993: Lipari guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1993.
5. E. Habermalz. Interactive theorem proving with schematic theory specific rules. Technical Report 19/00, 2000.
6. G. Schellhorn. *Verification of Abstract State Machines*. PhD thesis, Universität Ulm, 1999.
7. S.Nanchen and R. Stärk. A security logic for abstract state machines. Submitted to ASM2004, 11 2003.

8. R. F. Stärk and S. Nanchen. A logic for abstract state machines. *Journal of Universal Computer Science*, 7(11):981–1006, 2001.
9. A. Troelstra and H.Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, second edition, 2000.
10. K. Winter. *Model Checking Abstract State Machines*. PhD thesis, Technical University of Berlin, 2001.

A The sequent calulus

In the following tables, we are presenting all the rules and heuristics of the basic logic of the ASMKeY theorem prover. We present here the different sections of the calculus. Some section, like FOL, are well-known and are given for the sake of integrality.

A.I First-order logic rules

axiom

$$\frac{}{\Gamma, \varphi \Rightarrow \varphi, \Delta}$$

true-right

$$\frac{}{\Gamma \Rightarrow \mathsf{true}, \Delta}$$

false-left

$$\frac{}{\Gamma, \mathsf{false} \Rightarrow \Delta}$$

not-left

$$\frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma, \neg\varphi \Rightarrow \Delta}$$

not-right

$$\frac{\Gamma, \varphi \Rightarrow \Delta}{\Gamma \Rightarrow \neg\varphi, \Delta}$$

impl-left

$$\frac{\Gamma \Rightarrow \varphi, \Delta \qquad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \to \psi \Rightarrow \Delta}$$

impl-right

$$\frac{\Gamma, \varphi \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \to \psi, \Delta}$$

iff-left

$$\frac{\Gamma, \varphi \to \psi \wedge \psi \to \varphi \Rightarrow \Delta}{\Gamma, \varphi \leftrightarrow \psi \Rightarrow \Delta}$$

iff-right

$$\frac{\Gamma \Rightarrow \varphi \to \psi \wedge \psi \to \varphi, \Delta}{\Gamma \Rightarrow \varphi \leftrightarrow \psi, \Delta}$$

and-left

$$\frac{\Gamma, \varphi, \psi \Rightarrow \Delta}{\Gamma, \varphi \wedge \psi \Rightarrow \Delta}$$

and-right

$$\frac{\Gamma \Rightarrow \varphi, \Delta \qquad \Gamma \Rightarrow \psi, \Delta}{\Gamma \Rightarrow \varphi \wedge \psi, \Delta}$$

or-left

$$\frac{\Gamma, \varphi \Rightarrow \Delta \qquad \Gamma, \psi \Rightarrow \Delta}{\Gamma, \varphi \vee \psi \Rightarrow \Delta}$$

or-right

$$\frac{\Gamma \Rightarrow \varphi, \psi, \Delta}{\Gamma \Rightarrow \varphi \vee \psi, \Delta}$$

all-left
(where t is a ground term; and t is static or φ is pure)

$$\frac{\Gamma, \forall x\varphi, \varphi\frac{t}{x} \Rightarrow \Delta}{\Gamma, \forall x\varphi \Rightarrow \Delta}$$

all-right
(where c is a new constant symbol)

$$\frac{\Gamma \Rightarrow \varphi\frac{c}{x}, \Delta}{\Gamma \Rightarrow \forall x\varphi, \Delta}$$

ex-right
(where t is ground term; and t is static or φ is pure)

$$\frac{\Gamma \Rightarrow \exists x\varphi, \varphi\frac{t}{x}, \Delta}{\Gamma \Rightarrow \exists x\varphi, \Delta}$$

ex-left
(where c is a new constant symbol)

$$\frac{\Gamma, \varphi\frac{x}{c} \Rightarrow \Delta}{\Gamma, \exists x\varphi \Rightarrow \Delta}$$

A.II Modal rules

mod-not-left (Lemma 4.28)

$$\frac{\Gamma \Rightarrow \mathsf{Con}(P), \Delta \qquad \Gamma, \neg[P]\varphi \Rightarrow \Delta}{\Gamma, [P]\neg\varphi \Rightarrow \Delta}$$

mod-not-right

$$\frac{\Gamma \Rightarrow \neg[P]\varphi, \Delta}{\Gamma \Rightarrow [P]\neg\varphi, \Delta}$$

mod-and-left

$$\frac{\Gamma, [P]\varphi_1 \wedge [P]\varphi_2 \Rightarrow \Delta}{\Gamma, [P](\varphi_1 \wedge \varphi_2) \Rightarrow \Delta}$$

mod-and-right

$$\frac{\Gamma \Rightarrow [P]\varphi_1 \wedge [P]\varphi_2, \Delta}{\Gamma \Rightarrow [P](\varphi_1 \wedge \varphi_2), \Delta}$$

mod-or-left

$$\frac{\Gamma, [P]\varphi_1 \vee [P]\varphi_2 \Rightarrow \Delta}{\Gamma, [P](\varphi_1 \vee \varphi_2) \Rightarrow \Delta}$$

mod-or-right

$$\frac{\Gamma \Rightarrow [P]\varphi_1 \vee [P]\varphi_2, \Delta}{\Gamma \Rightarrow [P](\varphi_1 \vee \varphi_2), \Delta}$$

mod-all-left (x not in P)

$$\frac{\Gamma, \forall x[P]\varphi \Rightarrow \Delta}{\Gamma, [P]\forall x\varphi \Rightarrow \Delta}$$

mod-all-right (x not in P)

$$\frac{\Gamma \Rightarrow \forall x[P]\varphi, \Delta}{\Gamma \Rightarrow [P]\forall x\varphi, \Delta}$$

mod-ex-left (x not in P)

$$\frac{\Gamma, \exists x[P]\varphi \Rightarrow \Delta}{\Gamma, [P]\exists x\varphi \Rightarrow \Delta}$$

mod-ex-right (x not in P)

$$\frac{\Gamma \Rightarrow \exists x[P]\varphi, \Delta}{\Gamma \Rightarrow [P]\exists x\varphi, \Delta}$$

step For Δ_2 not empty and Ψ pure and static formulas;

$$\frac{\Gamma_2, \Psi \Rightarrow \Delta_2}{\Gamma_1, [P]\Gamma_2, \Psi \Rightarrow \Delta_1, [P]\Delta_2}$$

mod-not-con (Axiom 5)

$$\frac{\Gamma \Rightarrow \neg\mathsf{Con}(P), \Delta}{\Gamma \Rightarrow [P]\varphi, \Delta}$$

A.III def predicate

def-skip-left (Axiom 10.D1)

$$\frac{\Gamma, \mathsf{true} \Rightarrow \Delta}{\Gamma, \mathsf{def}(\mathbf{skip}) \Rightarrow \Delta}$$

def-skip-right (Axiom 10.D1)

$$\frac{\Gamma \Rightarrow \mathsf{true}, \Delta}{\Gamma \Rightarrow \mathsf{def}(\mathbf{skip}), \Delta}$$

def-assign-left (Axiom 10.D2)

$$\frac{\Gamma, \mathsf{true} \Rightarrow \Delta}{\Gamma, \mathsf{def}(f(t) := s) \Rightarrow \Delta}$$

def-assign-right (Axiom 10.D2)

$$\frac{\Gamma \Rightarrow \mathsf{true}, \Delta}{\Gamma \Rightarrow \mathsf{def}(f(t) := s), \Delta}$$

def-par-left (Axiom 10.D3)

$$\frac{\Gamma, \mathsf{def}(P) \wedge \mathsf{def}(Q) \Rightarrow \Delta}{\Gamma, \mathsf{def}(P \ \mathbf{par} \ Q) \Rightarrow \Delta}$$

def-par-right (Axiom 10.D3)

$$\frac{\Gamma \Rightarrow \mathsf{def}(P) \wedge \mathsf{def}(Q), \Delta}{\Gamma \Rightarrow \mathsf{def}(P \ \mathbf{par} \ Q), \Delta}$$

def-branch-left (Axiom 10.D4)

$$\frac{\Gamma, \varphi, \mathsf{def}(P) \Rightarrow \Delta \quad \Gamma, \neg\varphi, \mathsf{def}(Q) \Rightarrow \Delta}{\Gamma, \mathsf{def}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q) \Rightarrow \Delta}$$

def-branch-right (Axiom 10.D4)

$$\frac{\Gamma, \varphi \Rightarrow \mathsf{def}(P), \Delta \quad \Gamma\neg\varphi \Rightarrow \mathsf{def}(Q), \Delta}{\Gamma, \Rightarrow \mathsf{def}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q), \Delta}$$

def-let-left (Axiom 10.D5)
(where c is a new constant symbol)

$$\frac{\Gamma, c = t, \mathsf{def}(P\tfrac{c}{x}) \Rightarrow \Delta}{\Gamma, \mathsf{def}(\mathbf{let}\ x = t\ \mathbf{in}\ P) \Rightarrow \Delta}$$

def-let-right (Axiom 10.D5)
(where c is a new constant symbol)

$$\frac{\mathit{\Gamma}, c = t \Rightarrow \mathsf{def}(P\frac{c}{x}), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{def}(\mathbf{let}\ x = t\ \mathbf{in}\ P), \mathit{\Delta}}$$

def-all-left (Axiom 10.D6)
(where $I = \mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ P$)

$$\frac{\mathit{\Gamma}, \forall x(\varphi \rightarrow \mathsf{def}(P)) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{def}(I) \Rightarrow \mathit{\Delta}}$$

def-all-right (Axiom 10.D6)
(where $I = \mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ P$)

$$\frac{\mathit{\Gamma} \Rightarrow \forall x(\varphi \rightarrow \mathsf{def}(P)), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{def}(I), \mathit{\Delta}}$$

def-seq-left (Axiom 10.D7)

$$\frac{\mathit{\Gamma}, \mathsf{def}(P) \wedge [P]\mathsf{def}(Q) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{def}(P\ \mathbf{seq}\ Q) \Rightarrow \mathit{\Delta}}$$

def-seq-right (Axiom 10.D7)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{def}(P) \wedge [P]\mathsf{def}(Q), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{def}(P\ \mathbf{seq}\ Q), \mathit{\Delta}}$$

def-try-left (Axiom 10.D8)

$$\frac{\mathit{\Gamma}, \mathsf{def}(P) \wedge (\mathsf{Con}(P) \vee \mathsf{def}(Q)) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{def}(\mathbf{try}\ P\ \mathbf{else}\ Q) \Rightarrow \mathit{\Delta}}$$

def-try-right (Axiom 10.D8)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{def}(P) \wedge (\mathsf{Con}(P) \vee \mathsf{def}(Q)), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{def}(\mathbf{try}\ R\ \mathbf{else}\ Q), \mathit{\Delta}}$$

def-rule-left (Axiom 10.D9)
(with rule definition $\rho(x) = P$.)

$$\frac{\mathit{\Gamma}, \mathsf{def}(P(t)) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{def}(\rho(t)) \Rightarrow \mathit{\Delta}}$$

def-rule-right (Axiom 10.D9)
(with rule definition $\rho(x) = P$.)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{def}(P(t)), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{def}(\rho(t)), \mathit{\Delta}}$$

A.IV upd predicate

t and s are static terms.

upd-skip-left (Axiom 11.U1)

$$\frac{\mathit{\Gamma}, \mathsf{false} \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{upd}(\mathbf{skip}, f, t, s) \Rightarrow \mathit{\Delta}}$$

upd-skip-right (Axiom 11.U1)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{false}, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{upd}(\mathbf{skip}, f, t, s), \mathit{\Delta}}$$

upd-assign-id-left (Axiom 11.U2)

$$\frac{\Gamma, t = t' \land s = s' \Rightarrow \Delta}{\Gamma, \mathsf{upd}(f(t) := s, f, t', s') \Rightarrow \Delta}$$

upd-assign-diff-left (Axiom 11.U2) (for $f \neq g$)

$$\frac{\Gamma, \mathsf{false} \Rightarrow \Delta}{\Gamma, \mathsf{upd}(f(t) := s, g, t', s') \Rightarrow \Delta}$$

upd-assign-id-right (Axiom 11.U2)

$$\frac{\Gamma \Rightarrow t = t' \land s = s', \Delta}{\Gamma \Rightarrow \mathsf{upd}(f(t) := s, f, t', s'), \Delta}$$

upd-assign-diff-right (Axiom 11.U2) (for $f \neq g$)

$$\frac{\Gamma \Rightarrow \mathsf{false}, \Delta}{\Gamma \Rightarrow \mathsf{upd}(f(t) := s, g, t', s'), \Delta}$$

upd-par-left (Axiom 11.U3)

$$\frac{\Gamma, \mathsf{def}(P \ \mathbf{par} \ Q) \land (\mathsf{upd}(P, f, t, s) \lor \mathsf{upd}(Q, f, t, s)) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(P \ \mathbf{par} \ Q, f, t, s) \Rightarrow \Delta}$$

upd-par-right (Axiom 11.U3)

$$\frac{\Gamma \Rightarrow \mathsf{def}(P \ \mathbf{par} \ Q) \land (\mathsf{upd}(P, f, t, s) \lor \mathsf{upd}(Q, f, t, s)), \Delta}{\Gamma \Rightarrow \mathsf{upd}(P \ \mathbf{par} \ Q, f, t, s), \Delta}$$

upd-branch-left (Axiom 11.U4)

$$\frac{\Gamma, \varphi, \mathsf{upd}(P, f, t, s) \Rightarrow \Delta \quad \Gamma, \neg\varphi, \mathsf{upd}(Q, f, t, s) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(\mathbf{if} \ \varphi \ \mathbf{then} \ P \ \mathbf{else} \ Q, f, t, s) \Rightarrow \Delta}$$

upd-branch-right (Axiom 11.U4)

$$\frac{\Gamma, \varphi \Rightarrow \mathsf{upd}(P, f, t, s), \Delta \quad \Gamma, \neg\varphi \Rightarrow \mathsf{upd}(Q, f, t, s), \Delta}{\Gamma \Rightarrow \mathsf{upd}(\mathbf{if} \ \varphi \ \mathbf{then} \ P \ \mathbf{else} \ Q, f, t, s), \Delta}$$

upd-let-left (Axiom 11.U5)
(where c is a new constant symbol)

$$\frac{\Gamma, c = t, \mathsf{upd}(P\tfrac{c}{x}, f, t, s) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(\mathbf{let}\ x = t\ \mathbf{in}\ P, f, t, s) \Rightarrow \Delta}$$

upd-let-right (Axiom 11.U5)
(where c is a new constant symbol)

$$\frac{\Gamma, c = t \Rightarrow \mathsf{upd}(P\tfrac{c}{x}, f, t, s), \Delta}{\Gamma \Rightarrow \mathsf{upd}(\mathbf{let}\ x = t\ \mathbf{in}\ P, f, t, s), \Delta}$$

upd-all-left (Axiom 11.U6)
(where $I \equiv$**forall** x **with** φ **do** P)

$$\frac{\Gamma, \mathsf{def}(I) \wedge \exists x(\varphi \wedge \mathsf{upd}(P, f, t, s)) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(I, f, t, s) \Rightarrow \Delta}$$

upd-all-right (Axiom 11.U6)
(where $I \equiv$**forall** x **with** φ **do** P)

$$\frac{\Gamma \Rightarrow \mathsf{def}(I) \wedge \exists x(\varphi \wedge \mathsf{upd}(P, f, t, s)), \Delta}{\Gamma \Rightarrow \mathsf{upd}(I, f, t, s), \Delta}$$

upd-seq-left (Axiom 11.U7)

$$\frac{\Gamma, (\mathsf{upd}(P, f, t, s) \wedge [P]\mathsf{inv}(Q, f, t)) \vee (\mathsf{Con}(P) \wedge [P]\mathsf{upd}(Q, f, t, s)) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(P\ \mathbf{seq}\ Q, f, t, s) \Rightarrow \Delta}$$

upd-seq-right (Axiom 11.U7)

$$\frac{\Gamma \Rightarrow (\mathsf{upd}(P, f, t, s) \wedge [P]\mathsf{inv}(Q, f, t)) \vee (\mathsf{Con}(P) \wedge [P]\mathsf{upd}(Q, f, t, s)), \Delta}{\Gamma \Rightarrow \mathsf{upd}(P\ \mathbf{seq}\ Q, f, t, s), \Delta}$$

upd-try-left (Axiom 11.U8)

$$\frac{\Gamma, (\mathsf{def}(P) \wedge \neg\mathsf{Con}(P) \wedge \mathsf{upd}(Q, f, t, s)) \vee (\mathsf{Con}(P) \wedge \mathsf{upd}(P, f, t, s)) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(\mathbf{try}\ P\ \mathbf{else}\ Q, f, t, s) \Rightarrow \Delta}$$

upd-try-right (Axiom 11.U8)

$$\frac{\Gamma \Rightarrow (\mathsf{def}(P) \land \neg\mathsf{Con}(P) \land \mathsf{upd}(Q,f,t,s)) \lor (\mathsf{Con}(P) \land \mathsf{upd}(P,f,t,s)), \Delta}{\Gamma \Rightarrow \mathsf{upd}(\textbf{try } P \textbf{ else } Q,f,t,s), \Delta}$$

upd-name-left (Axiom 11.U9)
(with rule definition $\rho(x) = P$.)

$$\frac{\Gamma, \mathsf{upd}(P(u),f,t,s) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(\rho(u),f,t,s) \Rightarrow \Delta}$$

upd-name-right (Axiom 11.U9)
(with rule definition $\rho(x) = P$.)

$$\frac{\Gamma \Rightarrow \mathsf{upd}(P(u),f,t,s), \Delta}{\Gamma \Rightarrow \mathsf{upd}(\rho(u),f,t,s), \Delta}$$

A.V Rules for pure static formulas

pure-left (Axiom 9)
(φ pure and static)

$$\frac{\Gamma \Rightarrow \mathsf{Con}(P), \Delta \qquad \Gamma, \varphi \Rightarrow \Delta}{\Gamma, [P]\varphi \Rightarrow \Delta}$$

pure-right (Axiom 8)
(φ pure and static)

$$\frac{\Gamma \Rightarrow \varphi, \Delta}{\Gamma \Rightarrow [P]\varphi, \Delta}$$

A.VI Update rules for transition rules

upd-def-axiom-12 (Axiom 12)

$$\frac{\Gamma, \mathsf{def}(P) \Rightarrow \Delta}{\Gamma, \mathsf{upd}(P,f,t,s) \Rightarrow \Delta}$$

mod-eq-left

$$\frac{\Gamma \Rightarrow \mathsf{Con}(P), \Delta \quad \Gamma, \mathsf{upd}(P,f,t,s) \lor (\mathsf{inv}(P,f,t) \land f(t) = s) \Rightarrow \Delta}{\Gamma, [P]f(t) = s \Rightarrow \Delta}$$

mod-eq-right (Axiom 13)

$$\frac{\Gamma \Rightarrow \mathsf{upd}(P,f,t,s) \lor (\mathsf{inv}(P,f,t) \land f(t) = s), \Delta}{\Gamma \Rightarrow [P]f(t) = s, \Delta}$$

A.VII Extensionality

extensionality-left-1 (Axiom 15)

$$\frac{\mathit{\Gamma} \Rightarrow P \simeq Q, \mathit{\Delta} \quad \mathit{\Gamma}, [P]\varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [Q]\varphi \Rightarrow \mathit{\Delta}}$$

extensionality-right-1 (Axiom 15)

$$\frac{\mathit{\Gamma} \Rightarrow P \simeq Q, \mathit{\Delta} \qquad \mathit{\Gamma} \Rightarrow [P]\varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [Q]\varphi, \mathit{\Delta}}$$

extensionality-left-2 (Axiom 15)

$$\frac{\mathit{\Gamma} \Rightarrow P \simeq Q, \mathit{\Delta} \qquad \mathit{\Gamma}, [Q]\varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [P]\varphi \Rightarrow \mathit{\Delta}}$$

extensionality-right-2 (Axiom 15)

$$\frac{\mathit{\Gamma} \Rightarrow P \simeq Q, \mathit{\Delta} \qquad \mathit{\Gamma} \Rightarrow [Q]\varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [P]\varphi, \mathit{\Delta}}$$

A.VIII Principles from dynamic logic

skip-left (Axiom 16)

$$\frac{\mathit{\Gamma}, \varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [\mathbf{skip}]\varphi \Rightarrow \mathit{\Delta}}$$

skip-right (Axiom 16)

$$\frac{\mathit{\Gamma} \Rightarrow \varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [\mathbf{skip}]\varphi, \mathit{\Delta}}$$

seq-left (Axiom 17)

$$\frac{\mathit{\Gamma}, [P][Q]\varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [P \ \mathbf{seq}\ Q]\varphi \Rightarrow \mathit{\Delta}}$$

seq-right (Axiom 17)

$$\frac{\mathit{\Gamma} \Rightarrow [P][Q]\varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [P \ \mathbf{seq}\ Q]\varphi, \mathit{\Delta}}$$

B Additional rules

B.I Con predicate

Con-skip-left (Lemma 3.18)

$$\frac{\mathit{\Gamma}, \mathsf{true} \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{Con}(\mathbf{skip}) \Rightarrow \mathit{\Delta}}$$

Con-skip-right (Lemma 3.18)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{true}, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{Con}(\mathbf{skip}), \mathit{\Delta}}$$

Con-assign-left (Lemma 3.19)

$$\frac{\mathit{\Gamma}, \mathsf{true} \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{Con}(f(t) := s) \Rightarrow \mathit{\Delta}}$$

Con-assign-right (Lemma 3.19)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{true}, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{Con}(f(t) := s), \mathit{\Delta}}$$

Con-par-left (Lemma 3.20)

$$\frac{\Gamma, (\mathsf{Con}(P) \wedge \mathsf{Con}(Q) \wedge \mathsf{joinable}(P, Q)) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(P \ \mathbf{par}\ Q) \Rightarrow \Delta}$$

Con-par-right (Lemma 3.20)

$$\frac{\Gamma \Rightarrow (\mathsf{Con}(P) \wedge \mathsf{Con}(Q) \wedge \mathsf{joinable}(P, Q)) \Rightarrow \Delta}{\Gamma \Rightarrow \mathsf{Con}(P \ \mathbf{par}\ Q), \Delta}$$

Con-branch-left (Lemma 3.21)

$$\frac{\Gamma, \varphi, \mathsf{Con}(P) \Rightarrow \Delta \quad \Gamma, \neg\varphi, \mathsf{Con}(Q) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q) \Rightarrow \Delta}$$

Con-branch-right (Lemma 3.21)

$$\frac{\Gamma, \varphi \Rightarrow \mathsf{Con}(P), \Delta \quad \Gamma, \neg\varphi \Rightarrow \mathsf{Con}(Q), \Delta}{\Gamma \Rightarrow \mathsf{Con}(\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q), \Delta}$$

Con-let-left (Lemma 3.22)

$$\frac{\Gamma, c = t, \mathsf{Con}(P\frac{c}{x}) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(\mathbf{let}\ x = t\ \mathbf{in}\ P) \Rightarrow \Delta}$$

Con-let-right (Lemma 3.22)

$$\frac{\Gamma, c = t \Rightarrow \mathsf{Con}(P\frac{c}{x}), \Delta}{\Gamma \Rightarrow \mathsf{Con}(\mathbf{let}\ x = t\ \mathbf{in}\ P), \Delta}$$

Con-all-left (Lemma 3.23)

$$\frac{\Gamma, \forall x(\varphi \rightarrow \mathsf{Con}(P) \wedge \forall y(\varphi^y_x \rightarrow \mathsf{joinable}(P, P^y_x))) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(\mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ P) \Rightarrow \Delta}$$

Con-all-right (Lemma 3.23)

$$\frac{\Gamma \Rightarrow \forall x(\varphi \rightarrow \mathsf{Con}(P) \wedge \forall y(\varphi^y_x \rightarrow \mathsf{joinable}(P, P^y_x))), \Delta}{\Gamma \Rightarrow \mathsf{Con}(\mathbf{forall}\ x\ \mathbf{with}\ \varphi\ \mathbf{do}\ P), \Delta}$$

Con-seq-left (Lemma 3.24)

$$\frac{\Gamma, \mathsf{Con}(P) \wedge [P]\mathsf{Con}(Q) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(P\ \mathbf{seq}\ Q) \Rightarrow \Delta}$$

Con-seq-right (Lemma 3.24)

$$\frac{\Gamma \Rightarrow \mathsf{Con}(P) \wedge [P]\mathsf{Con}(Q), \Delta}{\Gamma \Rightarrow \mathsf{Con}(P\ \mathbf{seq}\ Q), \Delta}$$

Con-try-left (Lemma 4.25)

$$\frac{\Gamma, \mathsf{Con}(P) \vee (\mathsf{def}(P) \wedge \mathsf{Con}(Q)) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(\textbf{try } P \textbf{ else } Q) \Rightarrow \Delta}$$

Con-try-right (Lemma 4.25)

$$\frac{\Gamma \Rightarrow \mathsf{Con}(P) \vee (\mathsf{def}(P) \wedge \mathsf{Con}(Q)), \Delta}{\Gamma \Rightarrow \mathsf{Con}(\textbf{try } P \textbf{ else } Q), \Delta}$$

Con-name-left (with rule definition $\rho(t) = P$)

$$\frac{\Gamma, \mathsf{Con}(P(t)) \Rightarrow \Delta}{\Gamma, \mathsf{Con}(\rho(t)) \Rightarrow \Delta}$$

Con-name-right (with rule definition $\rho(t) = P$)

$$\frac{\Gamma \Rightarrow \mathsf{Con}(P(t)), \Delta}{\Gamma \Rightarrow \mathsf{Con}(\rho(t)), \Delta}$$

B.II inv predicate

The inv predicate is defined as follow:

$$\mathsf{inv}(P, f, x) := \mathsf{def}(P) \wedge \forall y\, \neg \mathsf{upd}(P, f, x, y)$$

t is a static term.

inv-skip-left

$$\frac{\Gamma, \mathsf{true} \Rightarrow \Delta}{\Gamma, \mathsf{inv}(\textbf{skip}, f, t) \Rightarrow \Delta}$$

inv-skip-right

$$\frac{\Gamma \Rightarrow \mathsf{true}, \Delta}{\Gamma \Rightarrow \mathsf{inv}(\textbf{skip}, f, t), \Delta}$$

inv-assign-id-left

$$\frac{\Gamma, \neg s = t \Rightarrow \Delta}{\Gamma, \mathsf{inv}(f(s) := s', f, t) \Rightarrow \Delta}$$

inv-assign-diff-left
(where $f \neq g$)

$$\frac{\Gamma, \mathsf{false} \Rightarrow \Delta}{\Gamma, \mathsf{inv}(g(s) := s', f, t) \Rightarrow \Delta}$$

inv-assign-id-right

$$\frac{\Gamma \Rightarrow \neg s = t, \Delta}{\Gamma \Rightarrow \mathsf{inv}(f(s) := s', f, t), \Delta}$$

inv-assign-diff-right
(where $f \neq g$)

$$\frac{\Gamma \Rightarrow \mathsf{false}, \Delta}{\Gamma \Rightarrow \mathsf{inv}(g(s) := s', f, t), \Delta}$$

inv-par-left

$$\frac{\Gamma, \mathsf{def}(P \textbf{ par } Q) \land \mathsf{inv}(P, f, t) \land \mathsf{inv}(Q, f, t) \Rightarrow \Delta}{\Gamma, \mathsf{inv}(P \textbf{ par } Q, f, t) \Rightarrow \Delta}$$

inv-par-right

$$\frac{\Gamma \Rightarrow \mathsf{def}(P \textbf{ par } Q) \land \mathsf{inv}(P, f, t) \land \mathsf{inv}(Q, f, t), \Delta}{\Gamma \Rightarrow \mathsf{inv}(P \textbf{ par } Q, f, t), \Delta}$$

inv-branch-left

$$\frac{\Gamma, \varphi, \mathsf{inv}(P, f, t) \Rightarrow \Delta \quad \Gamma, \neg\varphi, \mathsf{inv}(Q, f, t) \Rightarrow \Delta}{\Gamma, \mathsf{inv}(\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q, f, t) \Rightarrow \Delta}$$

inv-branch-right

$$\frac{\Gamma, \varphi \Rightarrow \mathsf{inv}(P, f, t), \Delta \quad \Gamma, \neg\varphi \Rightarrow \mathsf{inv}(Q, f, t), \Delta}{\Gamma \Rightarrow \mathsf{inv}(\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q, f, t), \Delta}$$

inv-let-left

$$\frac{\Gamma, c = t, \mathsf{inv}(P\tfrac{c}{x}, f, t) \Rightarrow \Delta}{\Gamma, \mathsf{inv}(\textbf{let } x = t \textbf{ in } P, f, t) \Rightarrow \Delta}$$

inv-let-right

$$\frac{\Gamma, c = t \Rightarrow \mathsf{inv}(P\tfrac{c}{x}, f, t), \Delta}{\Gamma \Rightarrow \mathsf{inv}(\textbf{let } c = s \textbf{ in } P, f, t), \Delta}$$

inv-all-left
(where $I \equiv$**forall** x **with** φ **do** P)

$$\frac{\Gamma, \mathsf{def}(I) \land \forall x(\varphi \rightarrow \mathsf{inv}(P, f, t)) \Rightarrow \Delta}{\Gamma, \mathsf{inv}(I, f, t) \Rightarrow \Delta}$$

inv-all-right
(where $I \equiv$**forall** x **with** φ **do** P)

$$\frac{\Gamma \Rightarrow \mathsf{def}(I) \land \forall x(\varphi \rightarrow \mathsf{inv}(P, f, t)), \Delta}{\Gamma \Rightarrow \mathsf{inv}(I, f, t), \Delta}$$

inv-seq-left

$$\frac{\Gamma, \mathsf{inv}(P, f, t) \land [P](\mathsf{inv}(Q, f, t)) \Rightarrow \Delta}{\Gamma, \mathsf{inv}(P \textbf{ seq } Q, f, t) \Rightarrow \Delta}$$

inv-seq-right

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{inv}(P, f, t) \wedge [P]\mathsf{inv}(Q, f, t), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{inv}(P \ \mathbf{seq}\ Q, f, t), \mathit{\Delta}}$$

inv-try-left

$$\frac{\mathit{\Gamma}, (\mathsf{Con}(P) \wedge \mathsf{inv}(P, f, x)) \vee (\mathsf{def}(P) \wedge \neg\mathsf{Con}(P) \wedge \mathsf{inv}(Q, f, x)) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{inv}(\mathbf{try}\ P \ \mathbf{else}\ Q, f, x) \Rightarrow, \mathit{\Delta}}$$

inv-try-right

$$\frac{\mathit{\Gamma} \Rightarrow (\mathsf{Con}(P) \wedge \mathsf{inv}(P, f, x)) \vee (\mathsf{def}(P) \wedge \neg\mathsf{Con}(P) \wedge \mathsf{inv}(Q, f, x)), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{inv}(\mathbf{try}\ P \ \mathbf{else}\ Q, f, x) \Rightarrow \mathit{\Delta}}$$

inv-name-left (with rule definition $\rho(t) = P$)

$$\frac{\mathit{\Gamma}, \mathsf{inv}(P^t_x, f, s) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, \mathsf{inv}(\rho(t), f, s) \Rightarrow \mathit{\Delta}}$$

inv-name-right (with rule definition $\rho(t) = P$)

$$\frac{\mathit{\Gamma} \Rightarrow \mathsf{inv}(P^t_x, f, s), \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow \mathsf{inv}(\rho(t), f, s), \mathit{\Delta}}$$

B.III Principles from dynamic logic

branch-left (Lemma 6.34)

$$\frac{\mathit{\Gamma}, \varphi, [P]\psi \Rightarrow \mathit{\Delta} \quad \mathit{\Gamma}, \neg\varphi, [Q]\psi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q]\psi \Rightarrow \mathit{\Delta}}$$

branch-right (Lemma 6.34)

$$\frac{\mathit{\Gamma}, \varphi \Rightarrow [P]\psi, \mathit{\Delta} \quad \mathit{\Gamma}, \neg\varphi \Rightarrow [Q]\psi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q]\psi, \mathit{\Delta}}$$

let-left (Lemma 6.35) (c is a new constant symbol)

$$\frac{\mathit{\Gamma}, c = t, [P\frac{c}{x}]\varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [\mathbf{let}\ x = t\ \mathbf{in}\ P]\varphi \Rightarrow \mathit{\Delta}}$$

let-right (Lemma 6.35) (c is a new constant symbol)

$$\frac{\mathit{\Gamma}, c = t \Rightarrow [P\frac{c}{x}]\varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [\mathbf{let}\ x = t\ \mathbf{in}\ P]\varphi, \mathit{\Delta}}$$

try-left (Lemma 6.36)

$$\frac{\mathit{\Gamma}, ([P]\varphi \wedge ((\mathsf{def}(P) \wedge \neg\mathsf{Con}(P)) \rightarrow [Q]\varphi)) \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [\mathbf{try}\ P\ \mathbf{else}\ Q]\varphi \Rightarrow \mathit{\Delta}}$$

try-right (Lemma 6.36)

$$\frac{\Gamma \Rightarrow ([P]\varphi \wedge ((\mathsf{def}(P) \wedge \neg\mathsf{Con}(P)) \rightarrow [Q]\varphi)), \Delta}{\Gamma \Rightarrow [\textbf{try } P \textbf{ else } Q]\varphi, \Delta}$$

name-left (Lemma 6.37) (with rule definition $\rho(x) = P$.)

$$\frac{\Gamma, [P(t)]\varphi \Rightarrow \Delta}{\Gamma, [\rho(t)]\varphi \Rightarrow \Delta}$$

name-right (Lemma 6.37) (with rule definition $\rho(x) = P$.)

$$\frac{\Gamma \Rightarrow [P(t)]\varphi, \Delta}{\Gamma \Rightarrow [\rho(t)]\varphi, \Delta}$$

B.IV Equivalence rules

weaken-equiv

$$\frac{\Gamma \Rightarrow P \mathrel{\dot{\simeq}} Q, \Delta}{\Gamma \Rightarrow P \simeq Q, \Delta}$$

equi-par-skip-right (Lemma 8.38)

$$\frac{\Gamma \Rightarrow [P]\varphi, \Delta}{\Gamma \Rightarrow [P \textbf{ par skip}]\varphi, \Delta}$$

equi-par-skip-left (Lemma 8.38)

$$\frac{\Gamma, [P]\varphi \Rightarrow \Delta}{\Gamma, [P \textbf{ par skip}]\varphi \Rightarrow \Delta}$$

equi-par-com-right (Lemma 8.39)

$$\frac{\Gamma \Rightarrow [P \textbf{ par } Q]\varphi, \Delta}{\Gamma \Rightarrow [Q \textbf{ par } P]\varphi, \Delta}$$

equi-par-com-left (Lemma 8.39)

$$\frac{\Gamma, [P \textbf{ par } Q]\varphi \Rightarrow \Delta}{\Gamma, [Q \textbf{ par } P]\varphi \Rightarrow \Delta}$$

equi-par-ass-right (Lemma 8.40)

$$\frac{\Gamma \Rightarrow [(P \textbf{ par } Q) \textbf{ par } R]\varphi, \Delta}{\Gamma \Rightarrow [P \textbf{ par } (Q \textbf{ par } R)]\varphi, \Delta}$$

equi-par-ass-left (Lemma 8.40)

$$\frac{\Gamma, [(P \textbf{ par } Q) \textbf{ par } R]\varphi \Rightarrow \Delta}{\Gamma, [P \textbf{ par } (Q \textbf{ par } R)]\varphi \Rightarrow \Delta}$$

equi-par-idem-right (Lemma 8.41)

$$\frac{\Gamma \Rightarrow [P]\varphi, \Delta}{\Gamma \Rightarrow [P \textbf{ par } P]\varphi, \Delta}$$

equi-par-idem-left (Lemma 8.41)

$$\frac{\Gamma, [P]\varphi \Rightarrow \Delta}{\Gamma, [P \textbf{ par } P]\varphi \Rightarrow \Delta}$$

equi-par-if-right-1 (Lemma 8.42)

$$\frac{\Gamma \Rightarrow [\textbf{if } \varphi \textbf{ then } (P \textbf{ par } R) \textbf{ else } (Q \textbf{ par } R)]\psi, \Delta}{\Gamma \Rightarrow [(\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q) \textbf{ par } R]\psi, \Delta}$$

equi-par-if-left-2 (Lemma 8.42)

$$\frac{\Gamma, [\textbf{if } \varphi \textbf{ then } (P \textbf{ par } R) \textbf{ else } (Q \textbf{ par } R)]\psi \Rightarrow \Delta}{\Gamma, [(\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q) \textbf{ par } R]\psi \Rightarrow \Delta}$$

equi-par-if-right-2 (Lemma 8.43)

$$\frac{\Gamma \Rightarrow [\textbf{if } \varphi \textbf{ then } (R \textbf{ par } P) \textbf{ else } (R \textbf{ par } Q)]\psi, \Delta}{\Gamma \Rightarrow [R \textbf{ par } (\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q)]\psi, \Delta}$$

equi-par-if-left-2 (Lemma 8.43)

$$\frac{\Gamma, [\textbf{if } \varphi \textbf{ then } (R \textbf{ par } P) \textbf{ else } (R \textbf{ par } Q)]\psi \Rightarrow \Delta}{\Gamma, [R \textbf{ par } (\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q)]\psi \Rightarrow \Delta}$$

equi-seq-left-skip-left

$$\frac{\Gamma, [R]\varphi \Rightarrow \Delta}{\Gamma, [\textbf{skip seq } R]\varphi \Rightarrow \Delta}$$

equi-seq-left-skip-right

$$\frac{\Gamma \Rightarrow [R]\varphi, \Delta}{\Gamma \Rightarrow [\textbf{skip seq } R]\varphi, \Delta}$$

equi-seq-right-skip-left

$$\frac{\Gamma, [R]\varphi \Rightarrow \Delta}{\Gamma, [R \textbf{ seq skip}]\varphi \Rightarrow \Delta}$$

equi-seq-right-skip-right

$$\frac{\Gamma \Rightarrow [R]\varphi, \Delta}{\Gamma \Rightarrow [R \textbf{ seq skip}]\varphi, \Delta}$$

equi-seq-ass-left

$$\frac{\mathit{\Gamma}, [(P \textbf{ seq } Q) \textbf{ seq } R]\varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [P \textbf{ seq } (Q \textbf{ seq } R)]\varphi \Rightarrow \mathit{\Delta}}$$

equi-seq-ass-right

$$\frac{\mathit{\Gamma} \Rightarrow [(P \textbf{ seq } Q) \textbf{ seq } R]\varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [P \textbf{ seq } (Q \textbf{ seq } R)]\varphi, \mathit{\Delta}}$$

equi-seq-if-left

$$\frac{\mathit{\Gamma}, [\textbf{if } \varphi \textbf{ then } P \textbf{ seq } R \textbf{ else } Q \textbf{ seq } R]\varphi \Rightarrow \mathit{\Delta}}{\mathit{\Gamma}, [(\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q) \textbf{ seq } R]\varphi \Rightarrow \mathit{\Delta}}$$

equi-seq-if-right

$$\frac{\mathit{\Gamma} \Rightarrow [\textbf{if } \varphi \textbf{ then } P \textbf{ seq } R \textbf{ else } Q \textbf{ seq } R]\varphi, \mathit{\Delta}}{\mathit{\Gamma} \Rightarrow [(\textbf{if } \varphi \textbf{ then } P \textbf{ else } Q) \textbf{ seq } R]\varphi, \mathit{\Delta}}$$