MODELING AND VERIFICATION OF HYBRID SYSTEMS BASED ON EQUATIONS

Kazuhiro Ogata
NEC Software Hokuriku, Ltd. / JAIST
ogatak@acm.org

Daigo Yamagishi, Takahiro Seino, Kokichi Futatsugi
Graduate School of Information Science, JAIST
{d-yamagi, t-seino, kokichi}@jaist.ac.jp

Abstract We describe hybrid observational transition systems, or HOTSs. HOTSs are written in terms of equations and verified by means of equational reasoning. More concretely, CafeOBJ, an algebraic specification language, is used to specify HOTSs and verify that HOTSs have properties by writing proofs, or proof scores. One case study is used to demonstrate how to model hybrid systems as HOTSs, specify HOTSs in CafeOBJ and verify that HOTSs have properties with the CafeOBJ system.

Keywords: CafeOBJ, HOTS, hybrid systems, modeling, verification

1. INTRODUCTION

Embedded systems are inherently hybrid. Hybrid systems are sensitive to physical continuity such as real-time and real-temperature. Real-time systems are one sub-class of hybrid systems. Hybrid systems can be complex and subject to subtle errors. Therefore, several formal methods of modeling and verification of hybrid systems have been recently proposedKesten et al., 2000; Lamport, 1993; Lynch et al., 2003. Our method proposed in this paper is one of such formal methods. Our method is based on equations and equational reasoning. Equations are the most basic logical formulas and equational reasoning is the most fundamental way of reasoningGries and Schneider, 1993, which can moderate the difficulties of proofs that might otherwise become too hard to understand.

Consequently, we expect that our method is easier to learn and use than existing formal methods for hybrid systems.

We have been successfully applying equations and equational reasoning to modeling and verification of distributed systems such as security protocolsOgata and Futatsugi, 2003a; Ogata and Futatsugi, 2003b. The method used is called the OTS/CafeOBJ methodOgata and Futatsugi, 2003c, in which systems are modeled as observational transition systems, or OTSs, which are written in CafeOBJDiaconescu and Futatsugi, 1998, an algebraic specification language. It is verified that systems have properties by writing proofs, or proof scores in CafeOBJ and executing the proof scores with the CafeOBJ system.

The OTS/CafeOBJ method is evolved in order to deal with hybrid systems. OTSs are a definition of transition systems for writing transition systems in terms of equations. An OTS consists of a set of observers, which correspond to variables in the usual transition system definition, a set of initial states and a set of transitions. Observers are functions that return observable values in an OTS. By introducing observers that return real numbers, called physical observers, OTSs can deal with hybrid systems. Such OTSs are called hybrid observational transition systems, or HOTSs. In this paper, we describe HOTSs and demonstrate that equations can be used to specify hybrid systems and equational reasoning can be used to verify that hybrid systems have properties using a case study in which a temperature stabilizer is modeled and verified.

The rest of the paper is organized as follows. Section 2 mentions CafeOBJ. HOTSs are described in Section 3. A temperature stabilizer is modeled and verified in Section 4. We finally mention related work in Section 5.

2. CAFEOBJ IN A NUTSHELL

CafeOBJDiaconescu and Futatsugi, 1998 can be used to specify abstract machines as well as abstract data types. A visible sort denotes an abstract data type, while a hidden sort denotes the state space of an abstract machine. There are two kinds of operators to hidden sorts: action and observation operators. An action operator can change states of an abstract machine. Only observation operators can be used to observe the inside of an abstract machine. An action operator is basically specified with equations by describing how the value returned by each observation operator changes. Declarations of observation and action operators start with `bop` or `bops`, and those of other operators start with `op` or `ops`. Declarations of equations start with `eq`, and those of

conditional ones start with `ceq`. The CafeOBJ system rewrites a given term by regarding equations as left-to-right rewrite rules.

3. HYBRID OBSERVATIONAL TRANSITION SYSTEMS

HOTSs are OTSs that are evolved by introducing physical observers in order to deal with physical continuity. We assume that there exists a universal state space called Υ. We also suppose that each data type used has been defined beforehand, including the equivalence between two values v_1, v_2 of the data type denoted by $v_1 = v_2$. Let R and R^+ be a set of real numbers and a set of non-negative real numbers, respectively. An HOTS $\mathcal{S}$ is defined as $\langle \mathcal{O}, \mathcal{I}, \mathcal{T} \cup \{\mathrm{tick}_r \mid r \in R^+\}\rangle$ where

- $\mathcal{O}$: A set of observers. The set $\mathcal{O} = \mathcal{D} \cup \mathcal{P}$ is classified into the set $\mathcal{D}$ of discrete observers and the set $\mathcal{P}$ of physical observers. Each $o \in \mathcal{O}$ is a function $o : \Upsilon \to D$, where D is a data type. For each $p \in \mathcal{P}$, $D \subseteq R \cup \{\infty\}$.

  For each $p \in \mathcal{P}$, there are $l\,(\geq 1)$ aspects, each of which is denoted by a predicate $a_p^i : \Upsilon \to \{\mathrm{true}, \mathrm{false}\}$ where $i \in \{1, \ldots, l\}$. We suppose that the predicates are exhaustive and exclusive. For each aspect denoted by a_p^i, we have a value $\iota_{a_p^i}$ and function $f_{a_p^i} : R^+ \to R$ such that $f_{a_p^i}(0) = 0$. $\iota_{a_p^i}$ is the initial value returned by p as the aspect denoted by a_p^i starts and $f_{a_p^i}$ denotes how values returned by p change in the aspect as time goes by. There is a special physical observer $now : \Upsilon \to R^+$. There is one aspect called *cosmos* for *now*, ι_{cosmos} is 0 and $f_{cosmos}(t) = t$. *now* serves as the master clock that returns the time amount that has passed after starting the execution of $\mathcal{S}$.

  Given two states $v_1, v_2 \in \Upsilon$, the equivalence between two states, denoted by $v_1 =_{\mathcal{S}} v_2$, with respect to $\mathcal{S}$ is defined as $v_1 =_{\mathcal{S}} v_2 \stackrel{\mathrm{def}}{=} \forall o \in \mathcal{O}.o(v_1) = o(v_2)$.

- $\mathcal{I}$: A set of initial states such that $\mathcal{I} \subseteq \Upsilon$. Master clock *now* initially returns 0.

- $\mathcal{T} \cup \{tick_r \mid r \in R^+\}$: A set of conditional transitions. Each $\tau \in \mathcal{T} \cup \{tick_r \mid r \in R^+\}$ is a function $\tau : \Upsilon/=_{\mathcal{S}} \to \Upsilon/=_{\mathcal{S}}$ on equivalence classes of Υ with respect to $=_{\mathcal{S}}$. Let $\tau(v)$ be the representative element of $\tau([v])$ for each $v \in \Upsilon \cup \{tick_r \mid r \in R^+\}$ and it is called *the successor state* of v with respect to τ.

  The condition of a transition $\tau \in \mathcal{T} \cup \{tick_r \mid r \in R^+\}$ is called the effective condition. The effective condition is supposed to satisfy

the following requirement: given a state $\upsilon \in \Upsilon$, if the effective condition of τ is false in υ, then $\upsilon =_{\mathcal{S}} \tau(\upsilon)$.

For each $\tau \in \mathcal{T}$, the effective condition of τ consists of the physical part c^{p}_{τ} and the non-physical part c_{τ}. There is also the time advancing condition c^{t}_{τ} corresponding to c^{p}_{τ}. Any $\tau \in \mathcal{T}$ does not change values returned by *now*.

Each $tick_r$ is a time advancing transition, where $r \in R^+$. Given a state $\upsilon \in \Upsilon$, $now(tick_r(\upsilon))$ is $now(\upsilon)+r$ if for each $\tau \in \mathcal{T}$, c^{t}_{τ} keeps holding while r time units are passing, and is $now(\upsilon)$ otherwise. $tick_r$ does not change values returned by any discrete observers.

Observers and transitions may be indexed such as $o_{i_1,\ldots,i_m}$ and $\tau_{j_1,\ldots,j_n}$, where $m, n \geq 0$ and we assume that there exist data types D_k such that $k \in D_k$ $(k = i_1, \ldots, i_m, j_1, \ldots, j_n)$.

An execution of $\mathcal{S}$ is an infinite sequence $\upsilon_0, \upsilon_1, \ldots$ of states satisfying

- *Initiation*: $\upsilon_0 \in \mathcal{I}$,
- *Consecution*: For each $i \in \{0, 1, \ldots\}$, $\upsilon_{i+1} =_{\mathcal{S}} \tau(\upsilon_i)$ for some $\tau \in \mathcal{T}$.
- *Time Divergence*: As i increases, $now(\upsilon_i)$ increases without bound.

A state υ is reachable with respect to $\mathcal{S}$ if and only if there exists an execution of $\mathcal{S}$ in which υ appears. Let $\mathcal{R}_{\mathcal{S}}$ be the set of all reachable states with respect to $\mathcal{S}$.

All properties considered in this paper are invariants, which are defined as follows:

$$\mathsf{invariant}\ p \stackrel{\mathrm{def}}{=} \forall \upsilon \in \mathcal{R}_{\mathcal{S}}.p(\upsilon)\,.$$

Let $\boldsymbol{x}$ be all free variables in $\mathsf{invariant}\, p$. We suppose that $\mathsf{invariant}\, p$ is interpreted as $\forall \boldsymbol{x}.(\mathsf{invariant}\, p)$ in this paper. When proof scores are written to prove $\forall \boldsymbol{x}.(\mathsf{invariant}\, p)$, $\boldsymbol{x}$ are replaced with constants that denote arbitrary values corresponding to $\boldsymbol{x}$ and the universally quantifier is eliminated. If a variable is existentially quantified in $\mathsf{invariant}\, p$, the variable is replaced with a Skolem constant or function and the existential quantifier is eliminated.

HOTSs are written in CafeOBJ as OTSs. Observers and transitions are denoted by CafeOBJ observation and action operators, respectively. We prove a predicate invariant to an HOTS by reduction, case analysis and/or induction as we prove a predicate invariant to an OTS. In any case, proofs, or proof scores are written in CafeOBJ. A method of writing proof scores of invariants for OTSs is described in Ogata and Futatsugi, 2003c, which can be used for HOTSs.

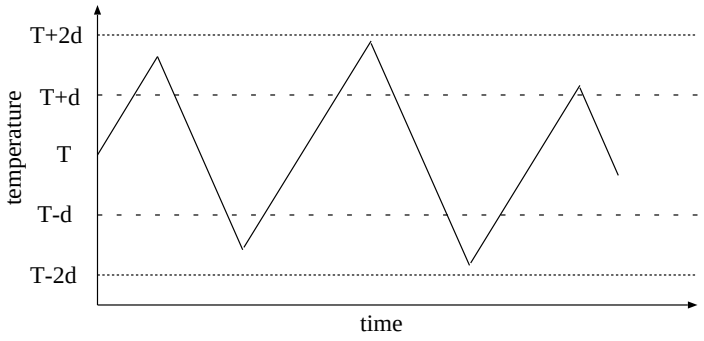

Figure 1. Behavior of the temperature stabilizer.

4. TEMPERATURE STABILIZER

Let us consider a temperature stabilizer that keeps the temperature between $T-2\times d$ and $T+2\times d$, where $d>0$. The temperature stabilizer consists of two processors called Cooler and Heater. Cooler lowers the temperature $b\,(>0)$ per unit time if it is active, while Heater raises the temperature $a\,(>0)$ per unit time if it is active. Cooler or Heater is active, and only one of them is active. Cooler can become active if the temperature is greater than or equal to $T+d$ and must become active by the time when the temperature is greater then $T+2\times d$. Heater can become active if the temperature is less than or equal to $T-d$ and must become active by the time when the temperature is less than $T-2\times d$. Initially the temperature is T and Heater is active. The behavior of the temperature stabilizer is depicted in Figure 1.

Modeling

The temperature stabilizer is modeled as an HOTS.

Observers.

Discrete observers.

- $\mathit{phase} : \Upsilon \to \{\text{true}, \text{false}\}$. It means that the temperature is heated if it returns true and the temperature is cooled otherwise. It initially returns true.
- $\mathit{point} : \Upsilon \to R^{+}$. It returns the latest time when the temperature has become heated (or cooled) if the temperature is heated (or cooled). It initially returns 0.

- $value : \Upsilon \rightarrow R$. It returns the temperature at the time returned by *point*. It initially returns T.

Physical observers.

- $temp : \Upsilon \rightarrow R$. It returns the temperature, initially T. There are two aspects heating and cooling that are denoted by *phase* and $\neg phase$, respectively. ι_{heating} is the value returned by *temp* as the aspect changes to heating from cooling and $f_{\text{heating}}(t) = a \times t$. ι_{cooling} is the value returned by *temp* as the aspect changes to cooling from heating and $f_{\text{cooling}}(t) = -b \times t$.
- $now : \Upsilon \rightarrow R^+$.

Transitions.

- $cool : \Upsilon \rightarrow \Upsilon$. Given a state $\upsilon \in \Upsilon$, $c_{cool}(\upsilon)$ is $phase(\upsilon)$ and $c^{\text{p}}_{cool}(\upsilon)$ is $temp(\upsilon) \geq T + d$. $c^{\text{t}}_{cool}(\upsilon, r)$ is $value(\upsilon) + f_{\text{heating}}(now(\upsilon) + r - point(\upsilon)) \leq T + 2 \times d$ if $c_{cool}(\upsilon)$ holds, and is true otherwise. If both $c_{cool}(\upsilon)$ and $c^{\text{p}}_{cool}(\upsilon)$ hold, then $phase(cool(\upsilon))$ is false, $point(cool(\upsilon))$ is $now(\upsilon)$ and $value(cool(\upsilon))$ is $temp(\upsilon)$. In any case, $temp(cool(\upsilon))$ is $temp(\upsilon)$.
- $heat : \Upsilon \rightarrow \Upsilon$. Given a state $\upsilon \in \Upsilon$, $c_{heat}(\upsilon)$ is $\neg phase(\upsilon)$ and $c^{\text{p}}_{heat}(\upsilon)$ is $temp(\upsilon) \leq T - d$. $c^{\text{t}}_{heat}(\upsilon, r)$ is $T - 2 \times d \leq value(\upsilon) + f_{\text{cooling}}(now(\upsilon) + r - point(\upsilon))$ if $c_{heat}(\upsilon)$ holds, and is true otherwise. If both $c_{heat}(\upsilon)$ and $c^{\text{p}}_{heat}(\upsilon)$ hold, then $phase(heat(\upsilon))$ is true, $point(heat(\upsilon))$ is $now(\upsilon)$ and $value(heat(\upsilon))$ is $temp(\upsilon)$. In any case, $temp(heat(\upsilon))$ is $temp(\upsilon)$.
- $tick_r : \Upsilon \rightarrow \Upsilon\ (r \in R^+)$. Given a state $\upsilon \in \Upsilon$, if both $c^{\text{t}}_{cool}(\upsilon, r_1)$ and $c^{\text{t}}_{heat}(\upsilon, r_1)$ holds for any r_1 such that $0 \leq r_1 \leq r$, $now(tick_r(\upsilon))$ is $now(\upsilon) + r$, and $temp(tick_r(\upsilon))$ is $value(\upsilon) + f_{\text{heating}}(now(\upsilon) + r - point(\upsilon))$ if $phase(\upsilon)$ is true, and is $value(\upsilon) + f_{\text{cooling}}(now(\upsilon) + r - point(\upsilon))$ otherwise.

The HOTS is written in CafeOBJ. The signature is as follows:

```
*[Sys]*
-- any initial state
op init : -> Sys
-- observation operators
op phase : Sys -> Bool        op point : Sys -> Real+
op value : Sys -> Real        op temp  : Sys -> Real
op now   : Sys -> Real+
--  action operators
op cool : Sys -> Sys          op heat : Sys -> Sys
```

```
op tick : Sys Real+ -> Sys
```

`Sys` is the hidden sort denoting the state space. A comment starts with `--` and terminates at the end of the line. Constant `init` denotes any initial state. `Bool`, `Real+` and `Real` are the visible sorts denoting the truth values, R^+ and R, respectively. You can imagine what the observation and action operators denote.

In the following, let `S` and `R` be CafeOBJ variables whose sorts are `Sys` and `Real+`, respectively. The equations defining `cool` are as follows:

```
op c-cool : Sys -> Bool
eq c-cool(S) = phase(S) and ((T + d) <= temp(S)) .
--
ceq phase(cool(S)) = false if c-cool(S) .
ceq point(cool(S)) = now(S) if c-cool(S) .
ceq value(cool(S)) = temp(S) if c-cool(S) .
eq  temp(cool(S))  = temp(S) .
eq  now(cool(S))   = now(S) .
ceq cool(S)        = S if not c-cool(S) .
```

Operator `c-cool` denotes $c_{cool} \wedge c^{\mathrm{p}}_{cool}$. Term `cool(S)` denotes the successor state of state `S` after applying transition *cool* denoted by action operator `cool`. The six equations from the bottom describe how to change the values returned by observers when *cool* is applied. For example, the sixth equation from the bottom says that the value returned by observer *phase* changes to false when *cool* is applied in a state where $c_{cool} \wedge c^{\mathrm{p}}_{cool}$ holds, and the last equation says that nothing changes when *cool* is applied in a state where $c_{cool} \wedge c^{\mathrm{p}}_{cool}$ does not hold.

`heat` is defined likewise.

The equations defining `tick` are as follows:

```
op c-tick : Sys Real+ -> Bool
eq c-tick(S,R)
   = (phase(S) implies
      ((value(S) + f-heat((now(S) + R) - point(S)))
       <= (T + (2 * d)))) and
     (not phase(S) implies
      ((T - (2 * d))
       <= (value(S) + f-cool((now(S) + R) - point(S))))) .
--
eq  phase(tick(S,R)) = phase(S) .
eq  point(tick(S,R)) = point(S) .
eq  value(tick(S,R)) = value(S) .
ceq temp(tick(S,R))
    = (if phase(S) then value(S) + f-heat((now(S) + R) - point(S))
                   else value(S) + f-cool((now(S) + R) - point(S)) fi)
    if c-tick(S,R) .
ceq now(tick(S,R))   = now(S) + R if c-tick(S,R) .
```

```
ceq tick(S,R)        = S if not c-tick(S,R) .
```

Operator `c-tick` denotes $c^{\mathrm{t}}_{cool} \wedge c^{\mathrm{t}}_{heat}$. Operators `f-cool` and `f-heat` denote f_{cooling} and f_{heating}, respectively. The six equations from the bottom describe how to change the values returned by observers when transition $tick_r$ denoted by action operator `tick` is applied.

Verification

We verify that the temperature is surely kept between $T - 2 \times d$ and $T + 2 \times d$. To this end, all we have to do is to prove the following predicate invariant to the HOTS:

$$T - 2 \times d \leq \mathit{temp}(\upsilon) \wedge \mathit{temp}(\upsilon) \leq T + 2 \times d\,. \tag{1}$$

To prove (1) invariant to the HOTS, we need to prove the following predicates invariant to the HOTS:

$$\mathit{point}(\upsilon) \leq \mathit{now}(\upsilon)\,, \tag{2}$$

$$T - 2 \times d \leq \mathit{value}(\upsilon)\,, \tag{3}$$

$$\mathit{value}(\upsilon) \leq T + 2 \times d\,. \tag{4}$$

The four predicates are proved invariant to the HOTS by induction on the number of transitions applied by writing proof scores in CafeOBJ.

Before writing proof scores, we first write a module, say `INV`, in which the four predicates are expressed as CafeOBJ terms as follows:

```
eq inv1(S) = (T - (2 * d)) <= temp(S) and temp(S) <= (T + (2 * d)) .
eq inv2(S) = point(S) <= now(S) .
eq inv3(S) = (T - (2 * d)) <= value(S) .
eq inv4(S) = value(S) <= (T + (2 * d)) .
```

We next write a module, say `ISTEP`, in which basic formulas to prove in each inductive case are expressed as CafeOBJ terms as follows:

```
eq istep1 = inv1(s) implies inv1(s') .
eq istep2 = inv2(s) implies inv2(s') .
eq istep3 = inv3(s) implies inv3(s') .
eq istep4 = inv4(s) implies inv4(s') .
```

Constants `s` and `s'` denote an arbitrary state and a successor state of `s`.

We then write four proof scores of (1), (2), (3) and (4). In this paper, we describe the inductive case in which $tick_r$ denoted by `tick` preserves (1). The state space, or the case is first split into two subcases: one where the condition denoted by `c-tick(s,r)` holds and the other where it does not. Since $tick_r$ does not change anything in a state in which the condition does not hold, $tick_r$ surely preserves (1).

The case in which the condition holds is divided into two: one where both `phase(s)` and `(value(s) + f-heat((now(s) + r) - point(s))) <= (T + (2 * d))` hold and the other where `phase(s)` does not hold and `(T - (2 * d)) <= (value(s) + f-cool((now(s) + r) - point (s)))` holds. The former is also split into three: (i) `point(s) <= now(s)` does not hold, (ii) `(T - (2 * d)) <= value(s)` does not hold and (iii) both `point(s) <= now(s)` and `(T - (2 * d)) <= value(s)` hold, and the latter is also split into three: (iv) `point(s) <= now(s)` does not hold, (v) `value(s) <= (T + (2 * d))` does not hold and (vi) both `point(s) <= now(s)` and `value(s) <= (T + (2 * d))` hold. Cases (i) and (iv) use (2) to strengthen the inductive hypothesis denoted by `inv1(s)`, case (ii) uses (3) to strengthen the inductive hypothesis and case (v) uses (4) to strengthen the inductive hypothesis.

In this paper, the proof passage of case (i) is shown, which is as follows:

```
open ISTEP
-- arbitrary objects
  op r : -> Real+ .
-- assumptions
  -- eq t-tick(s,r) = true .
  eq phase(s) = true .
  -- eq (value(s) + f-heat((now(s) + r) - point(s)))  <= (T + (2 * d))
  --    = true .
  eq (value(s) + (a * ((now(s) + r) - point(s)))) <= (T + (2 * d))
     = true .
  --
  eq point(s) <= now(s) = false .
-- successor state
  eq s' = tick(s,r) .
-- check
  red inv2(s) implies istep1 .
close
```

Constant `r` denotes an arbitrary non-negative real number. `(value(s) + (a * ((now(s) + r) - point(s)))) <= (T + (2 * d))` is the normal form of `(value(s) + f-heat((now(s) + r) - point(s))) <= (T + (2 * d))` in the sense of term rewriting. In order to make effective use of the declared equation, the left-hand side should be in normal form. The proof passage is executed by the CafeOBJ system, which returns `true`. This means that $tick_r$ preserves (1) for any $r \in R^+$ in this subcase.

The proof passages of the remaining subcases are written likewise.

5. RELATED WORK

LamportLamport, 1993 proposes a method of specifying and reasoning about hybrid systems in TLA+. TLA+ is a formal specification

language based on TLA, the Temporal Logic of Actions. Systems are specified in terms of temporal logic formulas. Kesten, et al.Kesten et al., 2000 propose phase transition systems for modeling hybrid systems and a rule for proving invariants of hybrid systems. Phase transition systems are described graphically, which seems suited for small examples. Lynch, et al.Lynch et al., 2003 propose hybrid I/O automata that are I/O automata that are evolved for modeling and analyzing hybrid systems. They verify that a hybrid system meets its specification by proving that there exists a simulation relation from a hybrid I/O automata modeling the hybrid system to a hybrid I/O automata describing the specification.

Lamport points out that any formal method that can be used to model and verify concurrent systems can be applied to distributed, real-time and/or hybrid systems. Phase transition systems and hybrid I/O automata are basically such examples, and so is our proposed method. One important difference between our method and the existing ones is that our method intensively uses equations and equational reasoning, which makes our method relatively easy to learn and use.

Maude, a sibling language of CafeOBJ, can be used to specify and analyze hybrid systemsÖlveczky and Meseguer, 2000 and is equipped with an LTL model checker. Maude can be used to complement our method. Design and implementation of a tool that translates CafeOBJ specifications into Maude ones is part of our future work.

References

Diaconescu, R. and Futatsugi, K. (1998). *CafeOBJ Report*, volume 6 of *AMAST Series in Computing*. World Scientific, Singapore.

Gries, D. and Schneider, F. B. (1993). *A Logical Approach to Discrete Math*. Texts and Monographs in Computer Science. Springer, NY.

Kesten, Y., Manna, Z., and Pnueli, A. (2000). Verification of clocked and hybrid systems. *Acta Informatica*, 36:837–912.

Lamport, L. (1993). Hybrid systems in TLA+. In *Hybrid Systems*, volume 736 of *LNCS*, pages 77–102. Springer.

Lynch, N., Segala, R., and Vaandraager, F. (2003). Hybrid I/O automata. *Information and Computation*, 185:105–157.

Ogata, K. and Futatsugi, K. (2003a). Formal analysis of the *i*KP electronic payment protocols. In *ISSS 2002*, volume 2609 of *LNCS*, pages 441–460. Springer.

Ogata, K. and Futatsugi, K. (2003b). Formal verification of the Horn-Preneel micropayment protocol. In *VMCAI 2003*, volume 2575 of *LNCS*, pages 238–252. Springer.

Ogata, K. and Futatsugi, K. (2003c). Proof scores in the OTS/CafeOBJ method. In *FMOODS 2003*, volume 2884 of *LNCS*, pages 170–184. Springer.

Ölveczky, P. C. and Meseguer, J. (2000). Real-Time Maude: A tool for simulating and analyzing real-time and hybrid systems. In *WRLA 2000*, volume 36 of *ENTCS*. Elsevier.