



AUTOMATED TRANSLATION OF VDM-SL TO JML-ANNOTATED JAVA

TECHNICAL REPORT

PETER W. V. TRAN-JØRGENSEN

Aarhus University - Department of Engineering



AARHUS
UNIVERSITY

DEPARTMENT OF ENGINEERING

Table of Contents

Table of Contents	i
Chapter 1 Introduction	1
1.1 The tool implementation	1
1.2 About this report	2
Chapter 2 The Translation	3
A The Automated Teller Machine (ATM) VDM-SL Example	4
B The code-generated ATM example	10
Bibliography	40

Introduction

A model specified using the Vienna Development Method (VDM) can be validated against its contract-based elements (e.g. pre and postconditions and invariants) in order to ensure that the system behaves as intended. One way to validate the model against its contracts is by animation using Overture's VDM interpreter.

When sufficient insight into the system under development has been obtained during the formal analysis, development proceeds to the implementation phase, where the system is realised. One way to realise a VDM model is by implementing it in a programming language, for example, using code generation. However, since no guarantees are made about the correctness of the generated code, other measures must be taken to increase the confidence in the correctness of the generated code.

To support this approach, Overture enables fully automated translation of VDM-SL's contract-based elements (pre- and postconditions, and invariants) and type constraints to Java Modeling Language (JML) annotations. The JML generator is developed as an extension of Overture's VDM-to-Java code generator and produces JML annotated Java programs. In this way JML tools can be used to validate the generated Java code against the *intended* system behaviour, described using JML. This work-flow is illustrated in fig. 1.1.

1.1 The tool implementation

The translation is defined as a set of rules that are implemented as an extension of Overture's VDM-to-Java code generator [2] to make the approach fully automated. The tool implementation of the JML translation is available in Overture 2.3.8 (as of July 2016), which can be downloaded via the Overture tool's website [5]. For instructions on how to use the JML translator we refer the reader to Overture's user guide [3], specifically the chapter on the VDM-to-Java code generator.

The generated Java programs can be checked for correctness with JML tools that support Java 7 or later. In particular, the generated Java programs, including the translation, have been tested using the OpenJML [1] runtime assertion checker. This is the only JML tool that we are aware of that currently supports all the JML constructs generated by the JML translator. The most recent version of OpenJML is available via [4].

1.2 About this report

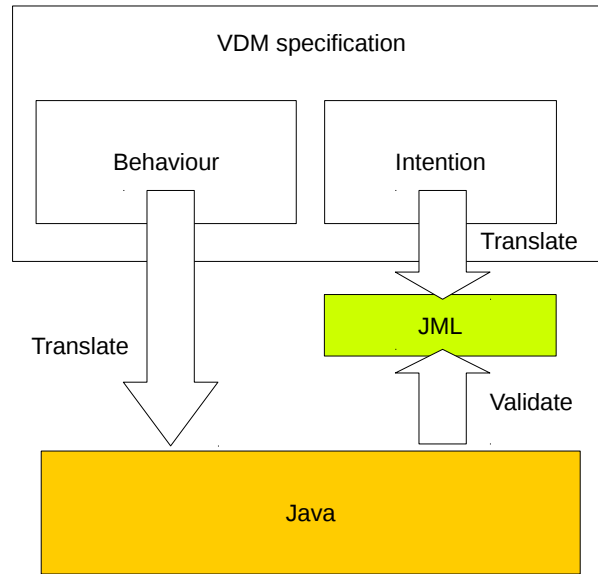


Figure 1.1: Overview of the VDM-to-JML translation.

The purpose of this report is to assist VDM users with getting started using the JML translator. This report is complementary to [6], which presents the different translation rules using an example of an ATM that is modelled in VDM-SL and implemented using the JML translator. The implementation of the ATM model is validated using the OpenJML runtime assertion checker. Due to space limitations only a reduced number of examples of limited detail are provided in [6]. To improve this, this report presents a more complete definition of the translation that uses several examples to demonstrate the translation.

The report is structured as follows: Chapter 2 presents the translation and demonstrate how the JML translator supports the implementation of VDM-SL models in Java. Appendix A contains the complete version of the ATM example used to demonstrate the translation in [6], and the corresponding Java/JML implementation is available in appendix B.



The Translation

Add some of the 85 examples used to validate the translation



The ATM VDM-SL Example

This appendix includes the complete version of the VDM-SL ATM example used to demonstrate the JML translation presented in [6] as well as this technical report.

```
module ATM

imports from IO all
exports all

definitions

state St of
  validCards : set of Card
  currentCard : [Card]
  pinOk : bool
  accounts : map AccountId to Account
  init St == St = mk_St({}, nil, false, {|->})
  inv mk_St(v, c, p, a) ==
    (p or c <> nil => c in set v)
    and
    forall id1, id2 in set dom a &
      id1 <> id2 =>
        a(id1).cards inter a(id2).cards = {}
end

types

Card ::
  id : nat
  pin : Pin;

Pin = nat
inv p == 0 <= p and p <= 9999;

AccountId = nat
inv id == id > 0;

Account ::
  cards : set of Card
  balance : real
  inv a == a.balance >= -1000;
```

Appendix A. The ATM VDM-SL Example

```
Amount = nat1
inv a == a < 2000;

functions

TotalBalance : set of Account -> real
TotalBalance (acs) ==
  if acs = {} then
    0
  else
    let a in set acs
    in
      a.balance + TotalBalance(acs \ {a})
measure TotalBalanceMes;

TotalBalanceMes: set of Account +> nat
TotalBalanceMes(acs) == card acs;

operations

GetStatus : () ==> bool * seq of char
GetStatus () ==
if currentCard <> nil then
  if pinOk then
    return mk_(false, "transaction in progress.")
  else
    return mk_(false, "debit card inserted. Awaiting pin code.")
else
  return mk_(true, "no debit card is currently inserted into the machine.");

OpenAccount : set of Card * AccountId ==> ()
OpenAccount (cards, id) ==
  accounts := accounts munion {id |-> mk_Account(cards, 0.0)}
pre id not in set dom accounts
post id in set dom accounts and
  accounts(id).balance = 0;

AddCard : Card ==> ()
AddCard (c) ==
  validCards := validCards union {c}
pre c not in set validCards
post c in set validCards;

RemoveCard : Card ==> ()
RemoveCard (c) ==
  validCards := validCards \ {c}
pre c in set validCards
post c not in set validCards;

InsertCard : Card ==> <Accept>|<Busy>|<Reject>
InsertCard (c) ==
if c in set validCards then
  (
    currentCard := c;
    return <Accept>;
  )
elseif currentCard <> nil then
  return <Busy>
else
  return <Reject>
```

Appendix A. The ATM VDM-SL Example

```

pre currentCard = nil
post
  if RESULT = <Accept> then
    currentCard = c
  else if RESULT = <Busy> then
    currentCard = currentCard~
  else currentCard = nil;

Display : seq of char ==> ()
Display (msg) ==
  IO'println(msg);

NotifyUser : <Accept>|<Busy>|<Reject> ==> ()
NotifyUser (outcome) ==
if outcome = <Accept> then
  Display("Card accepted")
elseif outcome = <Busy> then
  Display("Another card has already been inserted")
else if outcome = <Reject> then
  Display("Unknown card")
else
  error;

EnterPin : Pin ==> ()
EnterPin (pin) ==
  pinOk := (currentCard.pin = pin)
pre currentCard <> nil;

ReturnCard : () ==> ()
ReturnCard () ==
  atomic
  (
    currentCard := nil;
    pinOk := false;
  )
pre currentCard <> nil
post currentCard = nil and not pinOk;

Withdraw : AccountId * Amount ==> real
Withdraw (id, amount) ==
let newBalance = accounts(id).balance - amount
in
  (
    accounts(id).balance := newBalance;
    return newBalance;
  )
pre currentCard in set validCards and pinOk and
  currentCard in set accounts(id).cards and
  id in set dom accounts
post
let accountPre = accounts~(id),
  accountPost = accounts(id)
in
  accountPre.balance = accountPost.balance + amount and
  accountPost.balance = RESULT;

Deposit : AccountId * Amount ==> real
Deposit (id, amount) ==
let newBalance = accounts(id).balance + amount
in

```


Appendix A. The ATM VDM-SL Example

```
(
  accounts(id).balance := newBalance;
  return newBalance;
)
pre pre-Withdraw(id, amount, St)
post
let accountPre = accounts~(id),
    accountPost = accounts(id)
in
  accountPre.balance + amount = accountPost.balance and
  accountPost.balance = RESULT;

PrintAccount: AccountId ==> ()
PrintAccount(id) ==
let balance = accounts(id).balance
in
  IO`printf("Balance is for account %s is %s\n", [id,balance]);

GetCurrentCardId : () ==> [nat]
GetCurrentCardId () ==
  if currentCard <> nil then
    return currentCard.id
  else
    return nil;

--
-- Test operations
--

TestCurrentCardId : () ==> [nat]
TestCurrentCardId () ==
let id = GetCurrentCardId()
in
  return id;

TestStatus : () ==> real
TestStatus () ==
let accId = 1,
    c1 = mk_Card(1,1234)
in
  (
    AddCard(c1);
    OpenAccount({mk_Card(1,1234)}, accId);

    let status = GetStatus(),
        awaitingCard = status.#1,
        msg = status.#2
    in
      (
        IO`println("Message: " ^ msg);
        if awaitingCard and <Accept> = InsertCard(c1) then
          (
            NotifyUser(<Accept>);
            EnterPin(1234);
            Deposit(accId,100);
          );
      );
  );

  return 0;
```

```

);

TestWithdraw : () ==> real
TestWithdraw () ==
let accId = 1,
      cardId = 1,
      pin = 1234,
      c1 = mk_Card(cardId, pin)
in
(
  AddCard(c1);
  OpenAccount({mk_Card(1, 1234)}, accId);

  if InsertCard(c1) = <Accept> then
  (
    EnterPin(pin);
    let expense = 600,
        profit = 100
    in
      let amount : nat1 = expense - profit
      in
        Withdraw(accId, amount);
  );

  error;
);

TestTotalBalance : () ==> real
TestTotalBalance () ==
let card1 = mk_Card(1, 1234),
      card2 = mk_Card(2, 5678),
      ac1 = mk_Account({card1}, 1000),
      ac2 = mk_Account({card2}, 500)
in
  TotalBalance({ac1, ac2});

TestScenario : () ==> ()
TestScenario() ==
let accId1 : AccountId = 1,
      pin1 = 1234,
      card1 = mk_Card(1, pin1),
      pin2 = 2345,
      card2 = mk_Card(2, pin2)
in
(
  AddCard(card1);
  AddCard(card2);
  OpenAccount({card1, card2}, accId1);
  let - = InsertCard(card2) in skip;
  PrintAccount(accId1);
  EnterPin(2345);
  let - = Deposit(accId1, 200) in skip;

  PrintAccount(accId1);

  ReturnCard();
  RemoveCard(card1);
  RemoveCard(card2);
);

```

Appendix A. The ATM VDM-SL Example

end ATM



The code-generated ATM example

```
package atm;

import java.util.*;
import org.overture.codegen.runtime.*;
import org.overture.codegen.vdm2jml.runtime.*;

@SuppressWarnings("all")
//@ nullable_by_default

final public class ATM implements java.io.Serializable {
    /*@ spec_public @*/

    private static atm.ATMtypes.St St =
        new atm.ATMtypes.St(SetUtil.set(), null, false, MapUtil.map());
    /*@ public ghost static boolean invChecksOn = true; @*/

    private ATM() {}

    public static Tuple GetStatus() {

        if (!(Utils.equals(Utils.copy(St.get_currentCard()), null))) {
            if (St.get_pinOk()) {
                Tuple ret_1 = Tuple.mk_(false, "transaction in progress.");
                /*@ assert (V2J.isTup(ret_1,2) && Utils.is_bool(V2J.field(ret_1,0)) &&
                    Utils.is_(V2J.field(ret_1,1),String.class));

                return Utils.copy(ret_1);

            } else {
                Tuple ret_2 = Tuple.mk_(false, "debit card inserted. Awaiting pin code
                    .");
                /*@ assert (V2J.isTup(ret_2,2) && Utils.is_bool(V2J.field(ret_2,0)) &&
                    Utils.is_(V2J.field(ret_2,1),String.class));

                return Utils.copy(ret_2);
            }

        } else {
            Tuple ret_3 = Tuple.mk_(true, "no debit card is currently inserted into
                the machine.");
```

Appendix B. The code-generated ATM example

```
//@ assert (V2J.isTup(ret_3,2) && Utils.is_bool(V2J.field(ret_3,0)) &&
    Utils.is_(V2J.field(ret_3,1),String.class));

    return Utils.copy(ret_3);
}
}
//@ requires pre_OpenAccount(cards,id,St);
//@ ensures post_OpenAccount(cards,id,\old(St.copy()),St);

public static void OpenAccount(final VDMSet cards, final Number id) {

    //@ assert (V2J.isSet(cards) && (\forall int i; 0 <= i && i < V2J.size(
        cards); Utils.is_(V2J.get(cards,i),atm.ATMtypes.Card.class));

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert St != null;

    St.set_accounts(
        MapUtil.munion(
            Utils.copy(St.get_accounts()),
            MapUtil.map(new Maplet(id, new atm.ATMtypes.Account(cards, 0.0)))
        );
    }
    //@ requires pre_AddCard(c,St);
    //@ ensures post_AddCard(c,\old(St.copy()),St);

    public static void AddCard(final atm.ATMtypes.Card c) {

        //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

        //@ assert St != null;

        St.set_validCards(SetUtil.union(Utils.copy(St.get_validCards()), SetUtil.
            set(Utils.copy(c))));
    }
    //@ requires pre_RemoveCard(c,St);
    //@ ensures post_RemoveCard(c,\old(St.copy()),St);

    public static void RemoveCard(final atm.ATMtypes.Card c) {

        //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

        //@ assert St != null;

        St.set_validCards(SetUtil.diff(Utils.copy(St.get_validCards()), SetUtil.
            set(Utils.copy(c))));
    }
    //@ requires pre_InsertCard(c,St);
    //@ ensures post_InsertCard(c,\result,\old(St.copy()),St);

    public static Object InsertCard(final atm.ATMtypes.Card c) {

        //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

        if (SetUtil.inSet(c, Utils.copy(St.get_validCards()))) {
            //@ assert St != null;

            St.set_currentCard(Utils.copy(c));
        }
    }
}
```

Appendix B. The code-generated ATM example

```
Object ret_4 = atm.quotes.AcceptQuote.getInstance();
/*@ assert (Utils.is_(ret_4,atm.quotes.AcceptQuote.class) || Utils.is_(
    ret_4,atm.quotes.BusyQuote.class) || Utils.is_(ret_4,atm.quotes.
    RejectQuote.class));

return ret_4;

} else if (!(Utils.equals(Utils.copy(St.get_currentCard()), null))) {
Object ret_5 = atm.quotes.BusyQuote.getInstance();
/*@ assert (Utils.is_(ret_5,atm.quotes.AcceptQuote.class) || Utils.is_(
    ret_5,atm.quotes.BusyQuote.class) || Utils.is_(ret_5,atm.quotes.
    RejectQuote.class));

return ret_5;

} else {
Object ret_6 = atm.quotes.RejectQuote.getInstance();
/*@ assert (Utils.is_(ret_6,atm.quotes.AcceptQuote.class) || Utils.is_(
    ret_6,atm.quotes.BusyQuote.class) || Utils.is_(ret_6,atm.quotes.
    RejectQuote.class));

return ret_6;
}
}

public static void Display(final String msg) {

    /*@ assert Utils.is_(msg,String.class);

    IO.println(msg);
}

public static void NotifyUser(final Object outcome) {

    /*@ assert (Utils.is_(outcome,atm.quotes.AcceptQuote.class) || Utils.is_(
        outcome,atm.quotes.BusyQuote.class) || Utils.is_(outcome,atm.quotes.
        RejectQuote.class));

    if (Utils.equals(outcome, atm.quotes.AcceptQuote.getInstance())) {
        Display("Card accepted");
    } else if (Utils.equals(outcome, atm.quotes.BusyQuote.getInstance())) {
        Display("Another card has already been inserted");
    } else {
        if (Utils.equals(outcome, atm.quotes.RejectQuote.getInstance())) {
            Display("Unknown card");
        } else {
            throw new RuntimeException("ERROR statement reached");
        }
    }
}

/*@ requires pre_EnterPin(pin,St);

public static void EnterPin(final Number pin) {

    /*@ assert (Utils.is_nat(pin) && inv_ATM_Pin(pin));

    /*@ assert St != null;

    St.set_pinOk(Utils.equals(St.get_currentCard().get_pin(), pin));
}
```

Appendix B. The code-generated ATM example

```
//@ requires pre_ReturnCard(St);
//@ ensures post_ReturnCard(\old(St.copy()), St);

public static void ReturnCard() {

    atm.ATMtypes.Card atomicTmp_1 = null;
    //@ assert ((atomicTmp_1 == null) || Utils.is_(atomicTmp_1, atm.ATMtypes.
        Card.class));

    Boolean atomicTmp_2 = false;
    //@ assert Utils.is_bool(atomicTmp_2);

    {
        /* Start of atomic statement */
        //@ set invChecksOn = false;

        //@ assert St != null;

        St.set_currentCard(Utils.copy(atomicTmp_1));

        //@ assert St != null;

        St.set_pinOk(atomicTmp_2);

        //@ set invChecksOn = true;

        //@ assert St.valid();

    } /* End of atomic statement */
}
//@ requires pre-Withdraw(id, amount, St);
//@ ensures post-Withdraw(id, amount, \result, \old(St.copy()), St);

public static Number Withdraw(final Number id, final Number amount) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert (Utils.is_nat1(amount) && inv_ATM_Amount(amount));

    final Number newBalance =
        ((atm.ATMtypes.Account) Utils.get(St.accounts, id)).get_balance().
        doubleValue()
        - amount.longValue();
    //@ assert Utils.is_real(newBalance);

    {
        VDMMap stateDes_1 = St.get_accounts();

        atm.ATMtypes.Account stateDes_2 = ((atm.ATMtypes.Account) Utils.get(
            stateDes_1, id));

        //@ assert stateDes_2 != null;

        stateDes_2.set_balance(newBalance);
        //@ assert (V2J.isMap(stateDes_1) && (\forall int i; 0 <= i && i < V2J.
            size(stateDes_1); (Utils.is_nat(V2J.getDom(stateDes_1, i)) &&
            inv_ATM_AccountId(V2J.getDom(stateDes_1, i)) && Utils.is_(V2J.getRng(
            stateDes_1, i), atm.ATMtypes.Account.class)));

        //@ assert Utils.is_(St, atm.ATMtypes.St.class);
    }
}
```

Appendix B. The code-generated ATM example

```
//@ assert St.valid();

Number ret_7 = newBalance;
//@ assert Utils.is_real(ret_7);

return ret_7;
}
}
//@ requires pre_Deposit(id,amount,St);
//@ ensures post_Deposit(id,amount,\result,\old(St.copy()),St);

public static Number Deposit(final Number id, final Number amount) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert (Utils.is_nat1(amount) && inv_ATM_Amount(amount));

    final Number newBalance =
        ((atm.ATMtypes.Account) Utils.get(St.accounts, id)).get_balance().
        doubleValue()
        + amount.longValue();
    //@ assert Utils.is_real(newBalance);

    {
        VDMMap stateDes_3 = St.get_accounts();

        atm.ATMtypes.Account stateDes_4 = ((atm.ATMtypes.Account) Utils.get(
            stateDes_3, id));

        //@ assert stateDes_4 != null;

        stateDes_4.set_balance(newBalance);
        //@ assert (V2J.isMap(stateDes_3) && (\forall int i; 0 <= i && i < V2J.
            size(stateDes_3); (Utils.is_nat(V2J.getDom(stateDes_3,i)) &&
            inv_ATM_AccountId(V2J.getDom(stateDes_3,i)) && Utils.is_(V2J.getRng(
            stateDes_3,i),atm.ATMtypes.Account.class)));

        //@ assert Utils.is_(St,atm.ATMtypes.St.class);

        //@ assert St.valid();

        Number ret_8 = newBalance;
        //@ assert Utils.is_real(ret_8);

        return ret_8;
    }
}

public static void PrintAccount(final Number id) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    final Number balance = ((atm.ATMtypes.Account) Utils.get(St.accounts, id))
        .get_balance();
    //@ assert Utils.is_real(balance);

    IO.printf("Balance is for account %s is %s\n", SeqUtil.seq(id, balance));
}
```


Appendix B. The code-generated ATM example

```
public static Number GetCurrentCardId() {

    if (!(Utils.equals(Utils.copy(St.get_currentCard()), null))) {
        Number ret_9 = St.get_currentCard().get_id();
        //@ assert ((ret_9 == null) || Utils.is_nat(ret_9));

        return ret_9;
    } else {
        Number ret_10 = null;
        //@ assert ((ret_10 == null) || Utils.is_nat(ret_10));

        return ret_10;
    }
}

public static Number TestCurrentCardId() {

    final Number id = GetCurrentCardId();
    //@ assert ((id == null) || Utils.is_nat(id));

    Number ret_11 = id;
    //@ assert ((ret_11 == null) || Utils.is_nat(ret_11));

    return ret_11;
}

public static Number TestStatus() {

    final Number accId = 1L;
    //@ assert Utils.is_nat1(accId);

    final atm.ATMtypes.Card c1 = new atm.ATMtypes.Card(1L, 1234L);
    //@ assert Utils.is_(c1, atm.ATMtypes.Card.class);

    {
        AddCard(Utils.copy(c1));
        OpenAccount(SetUtil.set(new atm.ATMtypes.Card(1L, 1234L)), accId);
        {
            final Tuple status = GetStatus();
            //@ assert (V2J.isTup(status, 2) && Utils.is_bool(V2J.field(status, 0))
                && Utils.is_(V2J.field(status, 1), String.class));

            final Boolean awaitingCard = ((Boolean) status.get(0));
            //@ assert Utils.is_bool(awaitingCard);

            final String msg = SeqUtil.toStr(status.get(1));
            //@ assert Utils.is_(msg, String.class);

            {
                IO.println("Message: " + msg);
                Boolean andResult_8 = false;
                //@ assert Utils.is_bool(andResult_8);

                if (awaitingCard) {
                    if (Utils.equals(atm.quotes.AcceptQuote.getInstance(), InsertCard(
                        Utils.copy(c1)))) {
                        andResult_8 = true;
                        //@ assert Utils.is_bool(andResult_8);
                    }
                }
            }
        }
    }
}
```

Appendix B. The code-generated ATM example

```
    }
  }

  if (andResult_8) {
    NotifyUser(atm.quotes.AcceptQuote.getInstance());
    EnterPin(1234L);
    return Deposit(accId, 100L);
  }
}

Number ret_12 = 0L;
//@ assert Utils.is_real(ret_12);

return ret_12;
}

}

public static Number TestWithdraw() {

  final Number accId = 1L;
  //@ assert Utils.is_nat1(accId);

  final Number cardId = 1L;
  //@ assert Utils.is_nat1(cardId);

  final Number pin = 1234L;
  //@ assert Utils.is_nat1(pin);

  final atm.ATMtypes.Card c1 = new atm.ATMtypes.Card(cardId, pin);
  //@ assert Utils.is_(c1, atm.ATMtypes.Card.class);

  {
    AddCard(Utils.copy(c1));
    OpenAccount(SetUtil.set(new atm.ATMtypes.Card(1L, 1234L)), accId);
    if (Utils.equals(InsertCard(Utils.copy(c1)), atm.quotes.AcceptQuote.
      getInstance())) {
      EnterPin(pin);
      {
        final Number expense = 600L;
        //@ assert Utils.is_nat1(expense);

        final Number profit = 100L;
        //@ assert Utils.is_nat1(profit);

        {
          final Number amount = expense.longValue() - profit.longValue();
          //@ assert Utils.is_nat1(amount);

          return Withdraw(accId, amount);
        }
      }
    }
  }

  throw new RuntimeException("ERROR statement reached");
}

}

public static Number TestTotalBalance() {
```

Appendix B. The code-generated ATM example

```
final atm.ATMtypes.Card card1 = new atm.ATMtypes.Card(1L, 1234L);
//@ assert Utils.is_(card1,atm.ATMtypes.Card.class);

final atm.ATMtypes.Card card2 = new atm.ATMtypes.Card(2L, 5678L);
//@ assert Utils.is_(card2,atm.ATMtypes.Card.class);

final atm.ATMtypes.Account ac1 =
    new atm.ATMtypes.Account(SetUtil.set(Utils.copy(card1)), 1000L);
//@ assert Utils.is_(ac1,atm.ATMtypes.Account.class);

final atm.ATMtypes.Account ac2 = new atm.ATMtypes.Account(SetUtil.set(
    Utils.copy(card2)), 500L);
//@ assert Utils.is_(ac2,atm.ATMtypes.Account.class);

return TotalBalance(SetUtil.set(Utils.copy(ac1), Utils.copy(ac2)));
}

public static void TestScenario() {

    final Number accId1 = 1L;
    //@ assert (Utils.is_nat(accId1) && inv_ATM_AccountId(accId1));

    final Number pin1 = 1234L;
    //@ assert Utils.is_nat1(pin1);

    final atm.ATMtypes.Card card1 = new atm.ATMtypes.Card(1L, pin1);
    //@ assert Utils.is_(card1,atm.ATMtypes.Card.class);

    final Number pin2 = 2345L;
    //@ assert Utils.is_nat1(pin2);

    final atm.ATMtypes.Card card2 = new atm.ATMtypes.Card(2L, pin2);
    //@ assert Utils.is_(card2,atm.ATMtypes.Card.class);

    {
        AddCard(Utils.copy(card1));
        AddCard(Utils.copy(card2));
        OpenAccount(SetUtil.set(Utils.copy(card1), Utils.copy(card2)), accId1);
        {
            final Object ignorePattern_1 = InsertCard(Utils.copy(card2));
            //@ assert (Utils.is_(ignorePattern_1,atm.quotes.AcceptQuote.class) ||
                Utils.is_(ignorePattern_1,atm.quotes.BusyQuote.class) || Utils.is_
                (ignorePattern_1,atm.quotes.RejectQuote.class));

            /* skip */
        }

        PrintAccount(accId1);
        EnterPin(2345L);
        {
            final Number ignorePattern_2 = Deposit(accId1, 200L);
            //@ assert Utils.is_real(ignorePattern_2);

            /* skip */
        }

        PrintAccount(accId1);
        ReturnCard();
        RemoveCard(Utils.copy(card1));
        RemoveCard(Utils.copy(card2));
    }
}
```

Appendix B. The code-generated ATM example

```

    }
  }
  /*@ pure @*/

  public static Number TotalBalance(final VDMSet acs) {

    /*@ assert (V2J.isSet(acs) && (\forall int i; 0 <= i && i < V2J.size(acs);
      Utils.is_(V2J.get(acs,i), atm.ATMtypes.Account.class));

    if (Utils.empty(acs)) {
      Number ret_13 = 0L;
      /*@ assert Utils.is_real(ret_13);

      return ret_13;

    } else {
      Number letBeStExp_1 = null;
      atm.ATMtypes.Account a = null;

      Boolean success_1 = false;
      /*@ assert Utils.is_bool(success_1);

      VDMSet set_1 = Utils.copy(acs);
      /*@ assert (V2J.isSet(set_1) && (\forall int i; 0 <= i && i < V2J.size(
        set_1); Utils.is_(V2J.get(set_1,i), atm.ATMtypes.Account.class));

      for (Iterator iterator_1 = set_1.iterator(); iterator_1.hasNext() && !(
        success_1); ) {
        a = ((atm.ATMtypes.Account) iterator_1.next());
        success_1 = true;
        /*@ assert Utils.is_bool(success_1);

      }
      if (!(success_1)) {
        throw new RuntimeException("Let Be St found no applicable bindings");
      }

      letBeStExp_1 =
        a.get_balance().doubleValue()
          + TotalBalance(SetUtil.diff(Utils.copy(acs), SetUtil.set(Utils.
            copy(a))))
            .doubleValue();
      /*@ assert Utils.is_real(letBeStExp_1);

      Number ret_14 = letBeStExp_1;
      /*@ assert Utils.is_real(ret_14);

      return ret_14;
    }
  }
  /*@ pure @*/

  public static Number TotalBalanceMes(final VDMSet acs) {

    /*@ assert (V2J.isSet(acs) && (\forall int i; 0 <= i && i < V2J.size(acs);
      Utils.is_(V2J.get(acs,i), atm.ATMtypes.Account.class));

    Number ret_15 = acs.size();
    /*@ assert Utils.is_nat(ret_15);

```

Appendix B. The code-generated ATM example

```
    return ret_15;
}
/*@ pure @*/

public static Boolean pre_OpenAccount(
    final VDMSet cards, final Number id, final atm.ATMtypes.St St) {

    //@ assert (V2J.isSet(cards) && (\forallall int i; 0 <= i && i < V2J.size(
        cards); Utils.is_(V2J.get(cards,i),atm.ATMtypes.Card.class)));

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert Utils.is_(St,atm.ATMtypes.St.class);

    Boolean ret_16 = !(SetUtil.inSet(id, MapUtil.dom(Utils.copy(St.
        get_accounts()))));
    //@ assert Utils.is_bool(ret_16);

    return ret_16;
}
/*@ pure @*/

public static Boolean post_OpenAccount(
    final VDMSet cards, final Number id, final atm.ATMtypes.St _St, final
    atm.ATMtypes.St St) {

    //@ assert (V2J.isSet(cards) && (\forallall int i; 0 <= i && i < V2J.size(
        cards); Utils.is_(V2J.get(cards,i),atm.ATMtypes.Card.class)));

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert Utils.is_(_St,atm.ATMtypes.St.class);

    //@ assert Utils.is_(St,atm.ATMtypes.St.class);

    Boolean andResult_1 = false;
    //@ assert Utils.is_bool(andResult_1);

    if (SetUtil.inSet(id, MapUtil.dom(Utils.copy(St.get_accounts())))) {
        if (Utils.equals(((atm.ATMtypes.Account) Utils.get(St.accounts, id)).
            get_balance(), 0L)) {
            andResult_1 = true;
            //@ assert Utils.is_bool(andResult_1);
        }
    }

    Boolean ret_17 = andResult_1;
    //@ assert Utils.is_bool(ret_17);

    return ret_17;
}
/*@ pure @*/

public static Boolean pre_AddCard(final atm.ATMtypes.Card c, final atm.
    ATMtypes.St St) {

    //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

    //@ assert Utils.is_(St,atm.ATMtypes.St.class);
```

Appendix B. The code-generated ATM example

```
Boolean ret_18 = !(SetUtil.inSet(c, Utils.copy(St.get_validCards())));
/*@ assert Utils.is_bool(ret_18);

return ret_18;
}
/*@ pure @*/

public static Boolean post_AddCard(
    final atm.ATMtypes.Card c, final atm.ATMtypes.St _St, final atm.ATMtypes
        .St St) {

    //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

    //@ assert Utils.is_(_St,atm.ATMtypes.St.class);

    //@ assert Utils.is_(St,atm.ATMtypes.St.class);

    Boolean ret_19 = SetUtil.inSet(c, Utils.copy(St.get_validCards()));
    //@ assert Utils.is_bool(ret_19);

    return ret_19;
}
/*@ pure @*/

public static Boolean pre_RemoveCard(final atm.ATMtypes.Card c, final atm.
    ATMtypes.St St) {

    //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

    //@ assert Utils.is_(St,atm.ATMtypes.St.class);

    Boolean ret_20 = SetUtil.inSet(c, Utils.copy(St.get_validCards()));
    //@ assert Utils.is_bool(ret_20);

    return ret_20;
}
/*@ pure @*/

public static Boolean post_RemoveCard(
    final atm.ATMtypes.Card c, final atm.ATMtypes.St _St, final atm.ATMtypes
        .St St) {

    //@ assert Utils.is_(c,atm.ATMtypes.Card.class);

    //@ assert Utils.is_(_St,atm.ATMtypes.St.class);

    //@ assert Utils.is_(St,atm.ATMtypes.St.class);

    Boolean ret_21 = !(SetUtil.inSet(c, Utils.copy(St.get_validCards())));
    //@ assert Utils.is_bool(ret_21);

    return ret_21;
}
/*@ pure @*/

public static Boolean pre_InsertCard(final atm.ATMtypes.Card c, final atm.
    ATMtypes.St St) {

    //@ assert Utils.is_(c,atm.ATMtypes.Card.class);
```

Appendix B. The code-generated ATM example

```
//@ assert Utils.is_(St, atm.ATMtypes.St.class);

Boolean ret_22 = Utils.equals(Utils.copy(St.get_currentCard()), null);
//@ assert Utils.is_bool(ret_22);

return ret_22;
}
/*@ pure @*/

public static Boolean post_InsertCard(
    final atm.ATMtypes.Card c,
    final Object RESULT,
    final atm.ATMtypes.St _St,
    final atm.ATMtypes.St St) {

    //@ assert Utils.is_(c, atm.ATMtypes.Card.class);

    //@ assert (Utils.is_(RESULT, atm.quotes.AcceptQuote.class) || Utils.is_(
        RESULT, atm.quotes.BusyQuote.class) || Utils.is_(RESULT, atm.quotes.
        RejectQuote.class));

    //@ assert Utils.is_(_St, atm.ATMtypes.St.class);

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    if (Utils.equals(RESULT, atm.quotes.AcceptQuote.getInstance())) {
        Boolean ret_23 = Utils.equals(Utils.copy(St.get_currentCard()), c);
        //@ assert Utils.is_bool(ret_23);

        return ret_23;
    } else {
        if (Utils.equals(RESULT, atm.quotes.BusyQuote.getInstance())) {
            Boolean ret_24 =
                Utils.equals(Utils.copy(St.get_currentCard()), Utils.copy(_St.
                    get_currentCard()));
            //@ assert Utils.is_bool(ret_24);

            return ret_24;
        } else {
            Boolean ret_25 = Utils.equals(Utils.copy(St.get_currentCard()), null);
            //@ assert Utils.is_bool(ret_25);

            return ret_25;
        }
    }
}
/*@ pure @*/

public static Boolean pre_EnterPin(final Number pin, final atm.ATMtypes.St
    St) {

    //@ assert (Utils.is_nat(pin) && inv_ATM_Pin(pin));

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    Boolean ret_26 = !(Utils.equals(Utils.copy(St.get_currentCard()), null));
    //@ assert Utils.is_bool(ret_26);
```

Appendix B. The code-generated ATM example

```
    return ret_26;
}
/*@ pure @*/

public static Boolean pre_ReturnCard(final atm.ATMtypes.St St) {

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    Boolean ret_27 = !(Utils.equals(Utils.copy(St.get_currentCard()), null));
    //@ assert Utils.is_bool(ret_27);

    return ret_27;
}
/*@ pure @*/

public static Boolean post_ReturnCard(final atm.ATMtypes.St _St, final atm.
    ATMtypes.St St) {

    //@ assert Utils.is_(_St, atm.ATMtypes.St.class);

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    Boolean andResult_2 = false;
    //@ assert Utils.is_bool(andResult_2);

    if (Utils.equals(Utils.copy(St.get_currentCard()), null)) {
        if (!(St.get_pinOk())) {
            andResult_2 = true;
            //@ assert Utils.is_bool(andResult_2);
        }
    }

    Boolean ret_28 = andResult_2;
    //@ assert Utils.is_bool(ret_28);

    return ret_28;
}
/*@ pure @*/

public static Boolean pre_Withdraw(
    final Number id, final Number amount, final atm.ATMtypes.St St) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert (Utils.is_nat1(amount) && inv_ATM_Amount(amount));

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    Boolean andResult_3 = false;
    //@ assert Utils.is_bool(andResult_3);

    if (SetUtil.inSet(Utils.copy(St.get_currentCard()), Utils.copy(St.
        get_validCards()))) {
        Boolean andResult_4 = false;
        //@ assert Utils.is_bool(andResult_4);

        if (St.get_pinOk()) {
            Boolean andResult_5 = false;
```


Appendix B. The code-generated ATM example

```
//@ assert Utils.is_bool(andResult_5);

if (SetUtil.inSet(
    Utils.copy(St.get_currentCard()),
    Utils.copy(((atm.ATMtypes.Account) Utils.get(St.accounts, id)).
        get_cards())) {
    if (SetUtil.inSet(id, MapUtil.dom(Utils.copy(St.get_accounts())))) {
        andResult_5 = true;
        //@ assert Utils.is_bool(andResult_5);

    }
}

if (andResult_5) {
    andResult_4 = true;
    //@ assert Utils.is_bool(andResult_4);

}

if (andResult_4) {
    andResult_3 = true;
    //@ assert Utils.is_bool(andResult_3);

}

Boolean ret_29 = andResult_3;
//@ assert Utils.is_bool(ret_29);

return ret_29;
}
/*@ pure @*/

public static Boolean post-Withdraw(
    final Number id,
    final Number amount,
    final Number RESULT,
    final atm.ATMtypes.St _St,
    final atm.ATMtypes.St St) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert (Utils.is_nat1(amount) && inv_ATM_Amount(amount));

    //@ assert Utils.is_real(RESULT);

    //@ assert Utils.is_(_St, atm.ATMtypes.St.class);

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    final atm.ATMtypes.Account accountPre =
        Utils.copy(((atm.ATMtypes.Account) Utils.get(_St.accounts, id)));
    //@ assert Utils.is_(accountPre, atm.ATMtypes.Account.class);

    final atm.ATMtypes.Account accountPost =
        Utils.copy(((atm.ATMtypes.Account) Utils.get(St.accounts, id)));
    //@ assert Utils.is_(accountPost, atm.ATMtypes.Account.class);

    Boolean andResult_6 = false;
```

Appendix B. The code-generated ATM example

```
//@ assert Utils.is_bool (andResult_6);

if (Utils.equals(
    accountPre.get_balance(), accountPost.get_balance().doubleValue() +
    amount.longValue())) {
    if (Utils.equals(accountPost.get_balance(), RESULT)) {
        andResult_6 = true;
        //@ assert Utils.is_bool (andResult_6);

    }
}

Boolean ret_30 = andResult_6;
//@ assert Utils.is_bool (ret_30);

return ret_30;
}
/*@ pure @*/

public static Boolean pre_Deposit(
    final Number id, final Number amount, final atm.ATMtypes.St St) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert (Utils.is_nat1(amount) && inv_ATM_Amount (amount));

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    Boolean ret_31 = pre_Withdraw(id, amount, Utils.copy(St));
    //@ assert Utils.is_bool (ret_31);

    return ret_31;
}
/*@ pure @*/

public static Boolean post_Deposit(
    final Number id,
    final Number amount,
    final Number RESULT,
    final atm.ATMtypes.St _St,
    final atm.ATMtypes.St St) {

    //@ assert (Utils.is_nat(id) && inv_ATM_AccountId(id));

    //@ assert (Utils.is_nat1(amount) && inv_ATM_Amount (amount));

    //@ assert Utils.is_real (RESULT);

    //@ assert Utils.is_(_St, atm.ATMtypes.St.class);

    //@ assert Utils.is_(St, atm.ATMtypes.St.class);

    final atm.ATMtypes.Account accountPre =
        Utils.copy((atm.ATMtypes.Account) Utils.get(_St.accounts, id));
    //@ assert Utils.is_(accountPre, atm.ATMtypes.Account.class);

    final atm.ATMtypes.Account accountPost =
        Utils.copy((atm.ATMtypes.Account) Utils.get(St.accounts, id));
    //@ assert Utils.is_(accountPost, atm.ATMtypes.Account.class);
```

Appendix B. The code-generated ATM example

```
Boolean andResult_7 = false;
//@ assert Utils.is_bool (andResult_7);

if (Utils.equals(
    accountPre.get_balance().doubleValue() + amount.longValue(),
    accountPost.get_balance())) {
    if (Utils.equals(accountPost.get_balance(), RESULT)) {
        andResult_7 = true;
        //@ assert Utils.is_bool (andResult_7);
    }
}

Boolean ret_32 = andResult_7;
//@ assert Utils.is_bool (ret_32);

return ret_32;
}

public String toString() {

    return "ATM{" + "St := " + Utils.toString(St) + "}";
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_Pin(final Object check_p) {

    Number p = ((Number) check_p);

    Boolean andResult_9 = false;

    if (0L <= p.longValue()) {
        if (p.longValue() <= 9999L) {
            andResult_9 = true;
        }
    }

    return andResult_9;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_AccountId(final Object check_id) {

    Number id = ((Number) check_id);

    return id.longValue() > 0L;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_Amount(final Object check_a) {

    Number a = ((Number) check_a);

    return a.longValue() < 2000L;
```

Appendix B. The code-generated ATM example

```
}  
}
```

```
package atm.ATMtypes;  
  
import java.util.*;  
import org.overture.codegen.runtime.*;  
import org.overture.codegen.vdm2jml.runtime.*;  
  
@SuppressWarnings("all")  
//@ nullable_by_default  
  
final public class St implements Record, java.io.Serializable {  
    public VDMSet validCards;  
    public atm.ATMtypes.Card currentCard;  
    public Boolean pinOk;  
    public VDMMap accounts;  
    //@ public instance invariant atm.ATM.invChecksOn ==> inv_St(validCards,  
        currentCard,pinOk,accounts);  
  
    public St(  
        final VDMSet _validCards,  
        final atm.ATMtypes.Card _currentCard,  
        final Boolean _pinOk,  
        final VDMMap _accounts) {  
  
        //@ assert (V2J.isSet(_validCards) && (\forallall int i; 0 <= i && i < V2J.  
            size(_validCards); Utils.is_(V2J.get(_validCards,i),atm.ATMtypes.Card.  
            class)));  
  
        //@ assert ((_currentCard == null) || Utils.is_(_currentCard,atm.ATMtypes.  
            Card.class));  
  
        //@ assert Utils.is_bool(_pinOk);  
  
        //@ assert (V2J.isMap(_accounts) && (\forallall int i; 0 <= i && i < V2J.size  
            (_accounts); (Utils.is_nat(V2J.getDom(_accounts,i)) &&  
            inv_ATM_AccountId(V2J.getDom(_accounts,i))) && Utils.is_(V2J.getRng(  
            _accounts,i),atm.ATMtypes.Account.class)));  
  
        validCards = _validCards != null ? Utils.copy(_validCards) : null;  
        //@ assert (V2J.isSet(validCards) && (\forallall int i; 0 <= i && i < V2J.  
            size(validCards); Utils.is_(V2J.get(validCards,i),atm.ATMtypes.Card.  
            class)));  
  
        currentCard = _currentCard != null ? Utils.copy(_currentCard) : null;  
        //@ assert ((currentCard == null) || Utils.is_(currentCard,atm.ATMtypes.  
            Card.class));  
  
        pinOk = _pinOk;  
        //@ assert Utils.is_bool(pinOk);  
  
        accounts = _accounts != null ? Utils.copy(_accounts) : null;  
        //@ assert (V2J.isMap(accounts) && (\forallall int i; 0 <= i && i < V2J.size(  
            accounts); (Utils.is_nat(V2J.getDom(accounts,i)) && inv_ATM_AccountId(  
            V2J.getDom(accounts,i))) && Utils.is_(V2J.getRng(accounts,i),atm.  
            ATMtypes.Account.class)));  
    }  
}
```

Appendix B. The code-generated ATM example

```
}
/*@ pure @*/

public boolean equals(final Object obj) {

    if (!(obj instanceof atm.ATMtypes.St)) {
        return false;
    }

    atm.ATMtypes.St other = ((atm.ATMtypes.St) obj);

    return (Utils.equals(validCards, other.validCards))
        && (Utils.equals(currentCard, other.currentCard))
        && (Utils.equals(pinOk, other.pinOk))
        && (Utils.equals(accounts, other.accounts));
}
/*@ pure @*/

public int hashCode() {

    return Utils.hashCode(validCards, currentCard, pinOk, accounts);
}
/*@ pure @*/

public atm.ATMtypes.St copy() {

    return new atm.ATMtypes.St(validCards, currentCard, pinOk, accounts);
}
/*@ pure @*/

public String toString() {

    return "mk_ATM`St" + Utils.formatFields(validCards, currentCard, pinOk,
        accounts);
}
/*@ pure @*/

public VDMSet get_validCards() {

    VDMSet ret_37 = validCards;
    //@ assert atm.ATM.invChecksOn ==> ((V2J.isSet(ret_37) && (\forallall int i;
        0 <= i && i < V2J.size(ret_37); Utils.is_(V2J.get(ret_37,i),atm.
        ATMtypes.Card.class)))));

    return ret_37;
}

public void set_validCards(final VDMSet _validCards) {

    //@ assert atm.ATM.invChecksOn ==> ((V2J.isSet(_validCards) && (\forallall
        int i; 0 <= i && i < V2J.size(_validCards); Utils.is_(V2J.get(
        _validCards,i),atm.ATMtypes.Card.class)))));

    validCards = _validCards;
    //@ assert atm.ATM.invChecksOn ==> ((V2J.isSet(validCards) && (\forallall int
        i; 0 <= i && i < V2J.size(validCards); Utils.is_(V2J.get(validCards,i)
        ,atm.ATMtypes.Card.class)))));

}
/*@ pure @*/
```

Appendix B. The code-generated ATM example

```
public atm.ATMtypes.Card get_currentCard() {

    atm.ATMtypes.Card ret_38 = currentCard;
    //@ assert atm.ATM.invChecksOn ==> (((ret_38 == null) || Utils.is_(ret_38,
        atm.ATMtypes.Card.class)));

    return ret_38;
}

public void set_currentCard(final atm.ATMtypes.Card _currentCard) {

    //@ assert atm.ATM.invChecksOn ==> ((_currentCard == null) || Utils.is_(
        _currentCard, atm.ATMtypes.Card.class));

    currentCard = _currentCard;
    //@ assert atm.ATM.invChecksOn ==> ((currentCard == null) || Utils.is_(
        currentCard, atm.ATMtypes.Card.class));

}
/*@ pure @*/

public Boolean get_pinOk() {

    Boolean ret_39 = pinOk;
    //@ assert atm.ATM.invChecksOn ==> (Utils.is_bool(ret_39));

    return ret_39;
}

public void set_pinOk(final Boolean _pinOk) {

    //@ assert atm.ATM.invChecksOn ==> (Utils.is_bool(_pinOk));

    pinOk = _pinOk;
    //@ assert atm.ATM.invChecksOn ==> (Utils.is_bool(pinOk));

}
/*@ pure @*/

public VDMMap get_accounts() {

    VDMMap ret_40 = accounts;
    //@ assert atm.ATM.invChecksOn ==> ((V2J.isMap(ret_40) && (\forall int i;
        0 <= i && i < V2J.size(ret_40); (Utils.is_nat(V2J.getDom(ret_40,i)) &&
        inv_ATM_AccountId(V2J.getDom(ret_40,i)) && Utils.is_(V2J.getRng(ret_40
        ,i), atm.ATMtypes.Account.class))));

    return ret_40;
}

public void set_accounts(final VDMMap _accounts) {

    //@ assert atm.ATM.invChecksOn ==> ((V2J.isMap(_accounts) && (\forall int
        i; 0 <= i && i < V2J.size(_accounts); (Utils.is_nat(V2J.getDom(
        _accounts,i)) && inv_ATM_AccountId(V2J.getDom(_accounts,i)) && Utils.
        is_(V2J.getRng(_accounts,i), atm.ATMtypes.Account.class))));

    accounts = _accounts;
    //@ assert atm.ATM.invChecksOn ==> ((V2J.isMap(accounts) && (\forall int i
```

Appendix B. The code-generated ATM example

```
        ; 0 <= i && i < V2J.size(accounts); (Utils.is_nat(V2J.getDom(accounts,i)
    )) && inv_ATM_AccountId(V2J.getDom(accounts,i))) && Utils.is_(V2J.
    getRng(accounts,i),atm.ATMtypes.Account.class)))));

}
/*@ pure @*/

public Boolean valid() {

    return true;
}
/*@ pure @*/
/*@ helper @*/

public static Boolean inv_St(
    final VDMSet _validCards,
    final atm.ATMtypes.Card _currentCard,
    final Boolean _pinOk,
    final VDMMap _accounts) {

    Boolean success_2 = true;
    VDMSet v = null;
    atm.ATMtypes.Card c = null;
    Boolean p = null;
    VDMMap a = null;
    v = _validCards;
    c = _currentCard;
    p = _pinOk;
    a = _accounts;

    if (!(success_2)) {
        throw new RuntimeException("Record pattern match failed");
    }

    Boolean andResult_10 = false;

    Boolean orResult_1 = false;

    Boolean orResult_2 = false;

    if (p) {
        orResult_2 = true;
    } else {
        orResult_2 = !(Utils.equals(c, null));
    }

    if (!(orResult_2)) {
        orResult_1 = true;
    } else {
        orResult_1 = SetUtil.inSet(c, v);
    }

    if (orResult_1) {
        Boolean forAllExpResult_1 = true;
        VDMSet set_2 = MapUtil.dom(a);
        for (Iterator iterator_2 = set_2.iterator(); iterator_2.hasNext() &&
            forAllExpResult_1; ) {
            Number id1 = ((Number) iterator_2.next());
            for (Iterator iterator_3 = set_2.iterator(); iterator_3.hasNext() &&
                forAllExpResult_1; ) {
```

Appendix B. The code-generated ATM example

```
Number id2 = ((Number) iterator_3.next());
Boolean orResult_3 = false;

if (!(Utils.equals(id1, id2))) {
    orResult_3 = true;
} else {
    orResult_3 =
        Utils.empty(
            SetUtil.intersect(
                Utils.copy(((atm.ATMtypes.Account) Utils.get(a, id1)).
                    cards),
                Utils.copy(((atm.ATMtypes.Account) Utils.get(a, id2)).
                    cards)));
}

forAllExpResult_1 = orResult_3;
}
}
if (forAllExpResult_1) {
    andResult_10 = true;
}
}

return andResult_10;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_Pin(final Object check_p) {

    Number p = ((Number) check_p);

    Boolean andResult_9 = false;

    if (0L <= p.longValue()) {
        if (p.longValue() <= 9999L) {
            andResult_9 = true;
        }
    }

    return andResult_9;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_AccountId(final Object check_id) {

    Number id = ((Number) check_id);

    return id.longValue() > 0L;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_Amount(final Object check_a) {

    Number a = ((Number) check_a);
```


Appendix B. The code-generated ATM example

```
        return a.longValue() < 2000L;
    }
}
```

```
package atm.ATMtypes;

import java.util.*;
import org.overture.codegen.runtime.*;
import org.overture.codegen.vdm2jml.runtime.*;

@SuppressWarnings("all")
//@ nullable_by_default

final public class Account implements Record, java.io.Serializable {
    public VDMSet cards;
    public Number balance;
    //@ public instance invariant atm.ATM.invChecksOn ==> inv_Account(cards,
        balance);

    public Account(final VDMSet _cards, final Number _balance) {

        //@ assert (V2J.isSet(_cards) && (\forallall int i; 0 <= i && i < V2J.size(
            _cards); Utils.is_(V2J.get(_cards,i),atm.ATMtypes.Card.class)));

        //@ assert Utils.is_real(_balance);

        cards = _cards != null ? Utils.copy(_cards) : null;
        //@ assert (V2J.isSet(cards) && (\forallall int i; 0 <= i && i < V2J.size(
            cards); Utils.is_(V2J.get(cards,i),atm.ATMtypes.Card.class)));

        balance = _balance;
        //@ assert Utils.is_real(balance);

    }
    /*@ pure @*/

    public boolean equals(final Object obj) {

        if (!(obj instanceof atm.ATMtypes.Account)) {
            return false;
        }

        atm.ATMtypes.Account other = ((atm.ATMtypes.Account) obj);

        return (Utils.equals(cards, other.cards)) && (Utils.equals(balance, other.
            balance));
    }
    /*@ pure @*/

    public int hashCode() {

        return Utils.hashCode(cards, balance);
    }
    /*@ pure @*/

    public atm.ATMtypes.Account copy() {
```

Appendix B. The code-generated ATM example

```
    return new atm.ATMtypes.Account(cards, balance);
}
/*@ pure @*/

public String toString() {

    return "mk_ATM`Account" + Utils.formatFields(cards, balance);
}
/*@ pure @*/

public VDMSet get_cards() {

    VDMSet ret_35 = cards;
    //@ assert atm.ATM.invChecksOn ==> ((V2J.isSet(ret_35) && (\forallall int i;
        0 <= i && i < V2J.size(ret_35); Utils.is_(V2J.get(ret_35,i),atm.
        ATMtypes.Card.class)))));

    return ret_35;
}

public void set_cards(final VDMSet _cards) {

    //@ assert atm.ATM.invChecksOn ==> ((V2J.isSet(_cards) && (\forallall int i;
        0 <= i && i < V2J.size(_cards); Utils.is_(V2J.get(_cards,i),atm.
        ATMtypes.Card.class)))));

    cards = _cards;
    //@ assert atm.ATM.invChecksOn ==> ((V2J.isSet(cards) && (\forallall int i; 0
        <= i && i < V2J.size(cards); Utils.is_(V2J.get(cards,i),atm.ATMtypes.
        Card.class)))));

}
/*@ pure @*/

public Number get_balance() {

    Number ret_36 = balance;
    //@ assert atm.ATM.invChecksOn ==> (Utils.is_real(ret_36));

    return ret_36;
}

public void set_balance(final Number _balance) {

    //@ assert atm.ATM.invChecksOn ==> (Utils.is_real(_balance));

    balance = _balance;
    //@ assert atm.ATM.invChecksOn ==> (Utils.is_real(balance));

}
/*@ pure @*/

public Boolean valid() {

    return true;
}
/*@ pure @*/
/*@ helper @*/

public static Boolean inv_Account(final VDMSet _cards, final Number _balance
```

Appendix B. The code-generated ATM example

```
    ) {

        return _balance.doubleValue() >= -1000L;
    }

    /*@ pure @*/
    /*@ helper @*/

    public static Boolean inv_ATM_Pin(final Object check_p) {

        Number p = ((Number) check_p);

        Boolean andResult_9 = false;

        if (0L <= p.longValue()) {
            if (p.longValue() <= 9999L) {
                andResult_9 = true;
            }
        }

        return andResult_9;
    }

    /*@ pure @*/
    /*@ helper @*/

    public static Boolean inv_ATM_AccountId(final Object check_id) {

        Number id = ((Number) check_id);

        return id.longValue() > 0L;
    }

    /*@ pure @*/
    /*@ helper @*/

    public static Boolean inv_ATM_Amount(final Object check_a) {

        Number a = ((Number) check_a);

        return a.longValue() < 2000L;
    }
}
```

```
package atm.ATMtypes;

import java.util.*;
import org.overture.codegen.runtime.*;
import org.overture.codegen.vdm2jml.runtime.*;

@SuppressWarnings("all")
/*@ nullable_by_default

final public class Card implements Record, java.io.Serializable {
    public Number id;
    public Number pin;

    public Card(final Number _id, final Number _pin) {
```

Appendix B. The code-generated ATM example

```
//@ assert Utils.is_nat(_id);

//@ assert (Utils.is_nat(_pin) && inv_ATM_Pin(_pin));

id = _id;
//@ assert Utils.is_nat(id);

pin = _pin;
//@ assert (Utils.is_nat(pin) && inv_ATM_Pin(pin));

}
/*@ pure @*/

public boolean equals(final Object obj) {

    if (!(obj instanceof atm.ATMtypes.Card)) {
        return false;
    }

    atm.ATMtypes.Card other = ((atm.ATMtypes.Card) obj);

    return (Utils.equals(id, other.id)) && (Utils.equals(pin, other.pin));
}
/*@ pure @*/

public int hashCode() {

    return Utils.hashCode(id, pin);
}
/*@ pure @*/

public atm.ATMtypes.Card copy() {

    return new atm.ATMtypes.Card(id, pin);
}
/*@ pure @*/

public String toString() {

    return "mk_ATM`Card" + Utils.formatFields(id, pin);
}
/*@ pure @*/

public Number get_id() {

    Number ret_33 = id;
    //@ assert atm.ATM.invChecksOn ==> (Utils.is_nat(ret_33));

    return ret_33;
}

public void set_id(final Number _id) {

    //@ assert atm.ATM.invChecksOn ==> (Utils.is_nat(_id));

    id = _id;
    //@ assert atm.ATM.invChecksOn ==> (Utils.is_nat(id));
}
```

Appendix B. The code-generated ATM example

```
/*@ pure @*/

public Number get_pin() {

    Number ret_34 = pin;
    //@ assert atm.ATM.invChecksOn ==> ((Utils.is_nat(ret_34) && inv_ATM_Pin(
        ret_34)));

    return ret_34;
}

public void set_pin(final Number _pin) {

    //@ assert atm.ATM.invChecksOn ==> ((Utils.is_nat(_pin) && inv_ATM_Pin(
        _pin)));

    pin = _pin;
    //@ assert atm.ATM.invChecksOn ==> ((Utils.is_nat(pin) && inv_ATM_Pin(pin)
        ));

}
/*@ pure @*/

public Boolean valid() {

    return true;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_Pin(final Object check_p) {

    Number p = ((Number) check_p);

    Boolean andResult_9 = false;

    if (0L <= p.longValue()) {
        if (p.longValue() <= 9999L) {
            andResult_9 = true;
        }
    }

    return andResult_9;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_AccountId(final Object check_id) {

    Number id = ((Number) check_id);

    return id.longValue() > 0L;
}

/*@ pure @*/
/*@ helper @*/

public static Boolean inv_ATM_Amount(final Object check_a) {
```

Appendix B. The code-generated ATM example

```
    Number a = ((Number) check_a);

    return a.longValue() < 2000L;
}
}
```

```
package atm.quotes;

import org.overture.codegen.runtime.*;
import org.overture.codegen.vdm2jml.runtime.*;

@SuppressWarnings("all")
//@ nullable_by_default

final public class startQuote implements java.io.Serializable {
    private static int hc = 0;
    private static startQuote instance = null;

    public startQuote() {

        if (Utils.equals(hc, 0)) {
            hc = super.hashCode();
        }
    }

    public static startQuote getInstance() {

        if (Utils.equals(instance, null)) {
            instance = new startQuote();
        }

        return instance;
    }

    public int hashCode() {

        return hc;
    }

    public boolean equals(final Object obj) {

        return obj instanceof startQuote;
    }

    public String toString() {

        return "<start>";
    }
}
```

```
package atm.quotes;

import org.overture.codegen.runtime.*;
import org.overture.codegen.vdm2jml.runtime.*;
```

Appendix B. The code-generated ATM example

```
@SuppressWarnings("all")
//@ nullable_by_default

final public class RejectQuote implements java.io.Serializable {
    private static int hc = 0;
    private static RejectQuote instance = null;

    public RejectQuote() {

        if (Utils.equals(hc, 0)) {
            hc = super.hashCode();
        }
    }

    public static RejectQuote getInstance() {

        if (Utils.equals(instance, null)) {
            instance = new RejectQuote();
        }

        return instance;
    }

    public int hashCode() {

        return hc;
    }

    public boolean equals(final Object obj) {

        return obj instanceof RejectQuote;
    }

    public String toString() {

        return "<Reject>";
    }
}
```

```
package atm.quotes;

import org.overture.codegen.runtime.*;
import org.overture.codegen.vdm2jml.runtime.*;

@SuppressWarnings("all")
//@ nullable_by_default

final public class AcceptQuote implements java.io.Serializable {
    private static int hc = 0;
    private static AcceptQuote instance = null;

    public AcceptQuote() {

        if (Utils.equals(hc, 0)) {
            hc = super.hashCode();
        }
    }
}
```

Appendix B. The code-generated ATM example

```
public static AcceptQuote getInstance() {  
  
    if (Utils.equals(instance, null)) {  
        instance = new AcceptQuote();  
    }  
  
    return instance;  
}  
  
public int hashCode() {  
  
    return hc;  
}  
  
public boolean equals(final Object obj) {  
  
    return obj instanceof AcceptQuote;  
}  
  
public String toString() {  
  
    return "<Accept>";  
}  
}
```

```
package atm.quotes;  
  
import org.overture.codegen.runtime.*;  
import org.overture.codegen.vdm2jml.runtime.*;  
  
@SuppressWarnings("all")  
//@ nullable_by_default  
  
final public class appendQuote implements java.io.Serializable {  
    private static int hc = 0;  
    private static appendQuote instance = null;  
  
    public appendQuote() {  
  
        if (Utils.equals(hc, 0)) {  
            hc = super.hashCode();  
        }  
    }  
  
    public static appendQuote getInstance() {  
  
        if (Utils.equals(instance, null)) {  
            instance = new appendQuote();  
        }  
  
        return instance;  
    }  
  
    public int hashCode() {  
  
        return hc;  
    }  
}
```


Appendix B. The code-generated ATM example

```
public boolean equals(final Object obj) {  
    return obj instanceof appendQuote;  
}  
  
public String toString() {  
    return "<append>";  
}  
}
```

```
package atm.quotes;  
  
import org.overture.codegen.runtime.*;  
import org.overture.codegen.vdm2jml.runtime.*;  
  
@SuppressWarnings("all")  
//@ nullable_by_default  
  
final public class BusyQuote implements java.io.Serializable {  
    private static int hc = 0;  
    private static BusyQuote instance = null;  
  
    public BusyQuote() {  
        if (Utils.equals(hc, 0)) {  
            hc = super.hashCode();  
        }  
    }  
  
    public static BusyQuote getInstance() {  
        if (Utils.equals(instance, null)) {  
            instance = new BusyQuote();  
        }  
  
        return instance;  
    }  
  
    public int hashCode() {  
        return hc;  
    }  
  
    public boolean equals(final Object obj) {  
        return obj instanceof BusyQuote;  
    }  
  
    public String toString() {  
        return "<Busy>";  
    }  
}
```

Bibliography

- [1] Cok, D.R.: OpenJML: JML for Java 7 by Extending OpenJDK. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NASA Formal Methods, Lecture Notes in Computer Science, vol. 6617, pp. 472–479. Springer Berlin Heidelberg (2011)
- [2] Jørgensen, P.W.V., Couto, L.D., Larsen, M.: A Code Generation Platform for VDM. In: The Overture 2014 workshop (June 2014)
- [3] Larsen, P.G., Lausdahl, K., Tran-Jørgensen, P.W.V., Ribeiro, A., Wolff, S., Battle, N.: Overture VDM-10 Tool Support: User Guide. Tech. Rep. TR-2010-02, The Overture Initiative, www.overturetool.org (May 2010)
- [4] The OpenJML website. <http://http://www.openjml.org> (2016)
- [5] The Overture tool's website. <http://overturetool.org> (2016)
- [6] Tran-Jørgensen, P.W.V., Larsen, P.G., Leavens, G.T.: Automated translation of VDM to JML annotated Java (Jan 2016 Submitted to the International Journal on Software Tools for Technology Transfer (STTT))