Specification, Proof, and Model Checking of the Mondex Electronic Purse

Chris George and Anne Haxthausen

United Nations University

International Institute for Software Technology

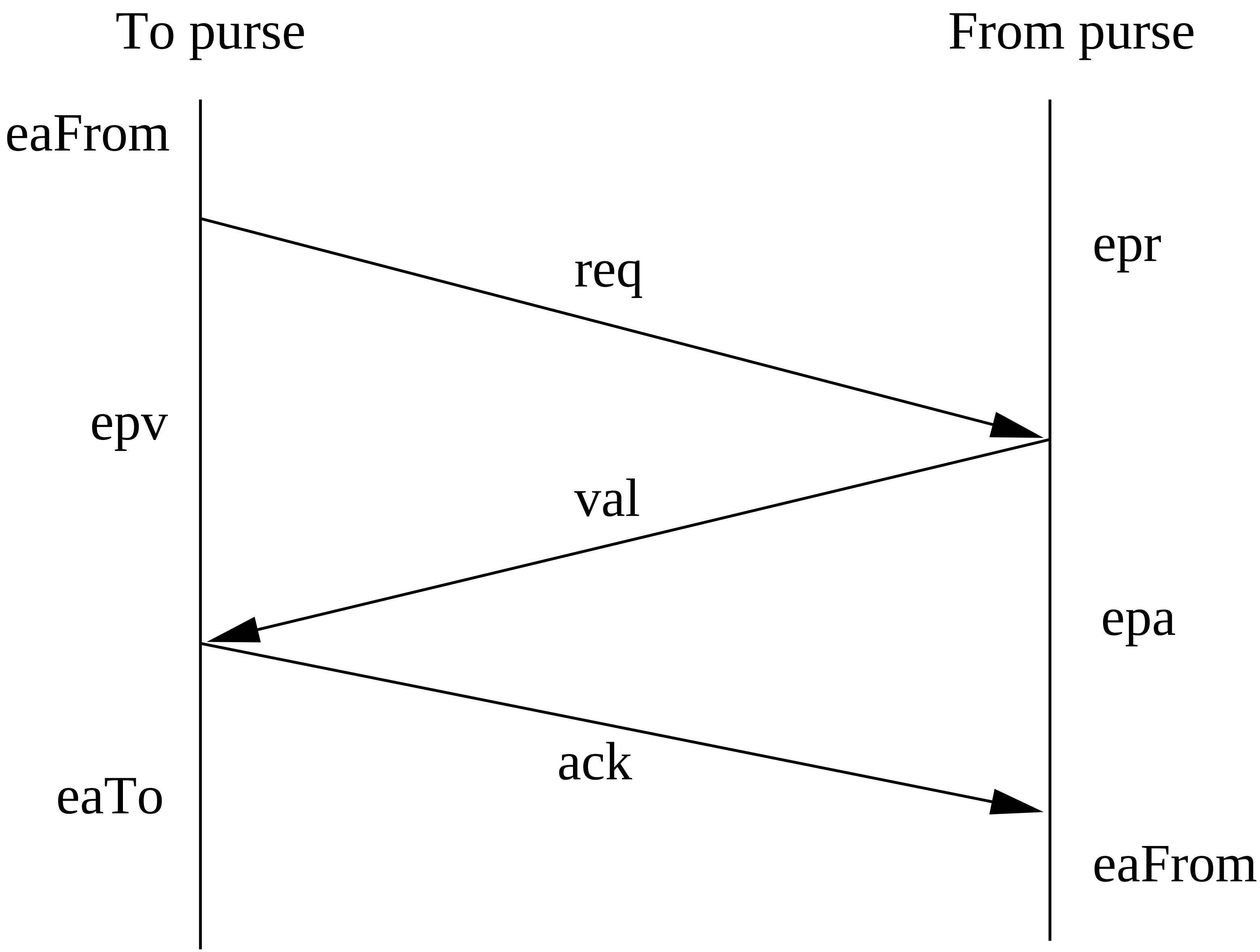
Macao SAR, China

and

Informatics and Mathematical Modelling

Technical University of Denmark

Lyngby, Denmark


To purse
From purse
eaFrom
epr
req
epv
val
epa
ack
eaTo
eaFrom

The Problem

1. Specify the protocol in detail
2. Prove that each operation satisfies two conditions:
   (a) *NoValueCreation*:
   inPurses + *inTransit* never increases
   (b) *AllValueAccounted*:
   inPurses + *inTransit* + *lost* is constant

We will call such an operation *correct*.

The RAISE Approach

3 levels of specification:

1. Abstract: a problem in accounting. No purses; no messages; just three “bottom line” values and four abstract *correct* operations that transfer money between them.

2. Intermediate: abstract states and (complete set of) abstract operations. No details of the mechanisms that preserve the (asserted) invariant. Prove that each intermediate operation implements an abstract (and hence *correct*) operation.

3. Concrete: full details of the protocol. Prove that each operation implements its intermediate version.

Abstract Specification

4 abstract operations

TransferLeft

$$\begin{aligned}&inPurses' = inPurses - valu(m) \wedge \\ &(lost' = lost \wedge inTransit' = inTransit + valu(m) \vee \\ &\ lost' = lost + valu(m) \wedge inTransit' = inTransit)\end{aligned}$$

TransferRight

$$\begin{aligned}&inPurses' = inPurses + valu(m) \wedge \\ &lost' = lost \wedge \\ &inTransit' = inTransit - valu(m)\end{aligned}$$

Abort

$$\exists\, \mathsf{v} : \mathbf{Nat} \bullet$$
$$\mathsf{inPurses}' = \mathsf{inPurses} \wedge$$
$$\mathsf{lost}' = \mathsf{lost} + \mathsf{v} \wedge$$
$$\mathsf{inTransit}' = \mathsf{inTransit} - \mathsf{v}$$

No_op

$$\mathsf{inPurses}' = \mathsf{inPurses} \wedge$$
$$\mathsf{lost}' = \mathsf{lost} \wedge$$
$$\mathsf{inTransit}' = \mathsf{inTransit}$$

No_op is really a special case of Abort.

It is easy to prove these 4 operations are *correct*.

Key relations

$\text{totalCirculating} = \text{inPurses} + \text{inTransit}$

$\text{totalAccounted} = \text{inPurses} + \text{lost} + \text{inTransit}$

$\text{inPurses} = \Sigma_{\mathrm{balance}}(\text{purses})$

$\text{inTransit} = \Sigma_{\mathrm{valu}}(\text{toInEpv} \cap (\text{fromLogs} \cup \text{fromInEpa})$

$\text{lost} = \Sigma_{\mathrm{valu}}(\text{toLogs} \cap (\text{fromLogs} \cup \text{fromInEpa}))$

toInEpv/*fromInEpa* are the payment details of purses with status *epv*/*epa* respectively.

A payment details goes into *toLogs*/*fromLogs* if a purse aborts or restarts from status *epv*/*epa* respectively.

Intermediate Specification

- Two modules: PURSE1 and WORLD1
- Both abstract:
  - Abstract state types *Purse* and *World*
  - Function signatures
  - State invariants as axioms
  - Observer-generator axioms
- Collection of operations is complete
- States are incomplete (no purse logs, sequence numbers, archive)
- *fromLogs* and *toLogs* are abstract observers

Intermediate Specification: World: invariant

axiom

[isWorldAxiom]

$\forall$ w : World, p : P.Purse •

p $\in$ **rng** purses(w) $\Rightarrow$

(P.status(p) $=$ T.epr $\Rightarrow$

P.pdAuth(p) $\notin$ fromInEpa(w) $\wedge$

P.pdAuth(p) $\notin$ fromLogs(w) $\wedge$

(T.req(P.pdAuth(p)) $\in$ ether(w) $\Rightarrow$

P.pdAuth(p) $\in$ toInEpv(w) $\wedge$ P.pdAuth(p) $\notin$ toLogs(w) $\vee$

P.pdAuth(p) $\in$ toLogs(w) $\wedge$ P.pdAuth(p) $\notin$ toInEpv(w))) $\wedge$

... $\wedge$

visible(w) $\subseteq$ ether(w)

Concrete Specification

- Two modules: PURSE2 and WORLD2
- All types concrete and complete
- Almost all functions concrete; some intentional underspecification (loss of messages; increase in sequence numbers; purse payment details in *eaTo*, *eaFrom*) handled with axioms
- *fromLogs* and *toLogs* now a construction from purse logs and archive

Concrete Specification: World: invariant 1

$$
\begin{array}{l}
\text{isWorld} : \text{WorldBase} \rightarrow \mathbf{Bool} \\
\text{isWorld(w)} \equiv \\
\quad (\forall\ \text{n} : \text{Name} \bullet \\
\qquad \text{n} \in \mathbf{dom}\ \text{archive(w)} \Rightarrow \text{n} \in \mathbf{dom}\ \text{purses(w))} \wedge \\
\quad (\forall\ \text{pd} : \text{PayDetails} \bullet \\
\qquad \text{req(pd)} \in \text{ether(w)} \Rightarrow \\
\qquad\quad \text{to(pd)} \in \text{purses(w)} \wedge \\
\qquad\quad \text{toSeqNo(pd)} < \text{nextSeqNo(purses(w)(to(pd))))} \wedge \\
\quad (\forall\ \text{pd} : \text{PayDetails} \bullet \\
\qquad \text{val(pd)} \in \text{ether(w)} \Rightarrow \\
\qquad\quad \text{to(pd)} \in \text{purses(w)} \wedge \text{ffrom(pd)} \in \text{purses(w)} \wedge \\
\qquad\quad \text{toSeqNo(pd)} < \text{nextSeqNo(purses(w)(to(pd)))} \wedge \\
\qquad\quad \text{fromSeqNo(pd)} < \text{nextSeqNo(purses(w)(ffrom(pd))))} \wedge
\end{array}
$$

Concrete Specification: World: invariant 2

$$
\begin{array}{l}
(\forall \ \mathrm{pd} : \mathrm{PayDetails} \bullet \\
\quad \mathrm{ack(pd)} \in \mathrm{ether(w)} \Rightarrow \\
\quad\quad \mathrm{to(pd)} \in \mathrm{purses(w)} \wedge \mathrm{ffrom(pd)} \in \mathrm{purses(w)} \wedge \\
\quad\quad \mathrm{ffrom(pd)} \in \mathrm{purses(w)} \wedge \\
\quad\quad \mathrm{toSeqNo(pd)} < \mathrm{nextSeqNo(purses(w)(to(pd)))} \wedge \\
\quad\quad \mathrm{fromSeqNo(pd)} < \mathrm{nextSeqNo(purses(w)(ffrom(pd)))} \wedge
\end{array}
$$

Concrete Specification: World: invariant 3

$$
\begin{array}{l}
(\forall \text{ pd} : \text{PayDetails} \bullet \\
\quad \text{pd} \in \text{fromLogs(w)} \Rightarrow \\
\qquad \text{req(pd)} \in \text{ether(w)} \wedge \\
\qquad \text{fromSeqNo(pd)} < \text{nextSeqNo(purses(w)(ffrom(pd)))} \wedge \\
\qquad \text{(status(purses(w)(ffrom(pd)))} \in \{\text{epr, epa}\} \Rightarrow \\
\qquad\quad \text{fromSeqNo(pd)} < \text{fromSeqNo(pdAuth(purses(w)(ffrom(pd))))))} \wedge \\
(\forall \text{ pd} : \text{PayDetails} \bullet \\
\quad \text{pd} \in \text{toLogs(w)} \Rightarrow \\
\qquad \text{req(pd)} \in \text{ether(w)} \wedge \\
\qquad \text{ack(pd)} \notin \text{ether(w)} \wedge \\
\qquad \text{(status(purses(w)(to(pd)))} \in \{\text{epv, eaTo}\} \Rightarrow \\
\qquad\quad \text{toSeqNo(pd)} < \text{toSeqNo(pdAuth(purses(w)(to(pd))))))} \wedge
\end{array}
$$

Concrete Specification: World: invariant 4

$$
\begin{array}{l}
(\forall\ \mathsf{n} : \mathsf{Name} \bullet \\
\quad \mathsf{n} \in \mathbf{dom}\ \mathsf{purses(w)} \wedge \mathsf{status(purses(w)(n))} = \mathsf{epa} \Rightarrow \\
\qquad \mathsf{req(pdAuth(purses(w)(n)))} \in \mathsf{ether(w))} \wedge \\
(\forall\ \mathsf{n} : \mathsf{Name} \bullet \\
\quad \mathsf{n} \in \mathbf{dom}\ \mathsf{purses(w)} \wedge \mathsf{status(purses(w)(n))} = \mathsf{epr} \Rightarrow \\
\qquad \mathsf{val(pdAuth(purses(w)(n)))} \notin \mathsf{ether(w)} \wedge \\
\qquad \mathsf{ack(pdAuth(purses(w)(n)))} \notin \mathsf{ether(w))} \wedge \\
(\forall\ \mathsf{n} : \mathsf{Name} \bullet \\
\quad \mathsf{n} \in \mathbf{dom}\ \mathsf{purses(w)} \wedge \mathsf{status(purses(w)(n))} = \mathsf{epv} \Rightarrow \\
\qquad \mathsf{req(pdAuth(purses(w)(n)))} \in \mathsf{ether(w)} \wedge \\
\qquad \mathsf{ack(pdAuth(purses(w)(n)))} \notin \mathsf{ether(w))} \wedge
\end{array}
$$

Concrete Specification: World: invariant 5

$$
\begin{aligned}
&(\forall\ \mathrm{pd} : \mathrm{PayDetails} \bullet \\
&\quad \mathrm{req(pd)} \in \mathrm{ether(w)} \wedge \mathrm{ack(pd)} \notin \mathrm{ether(w)} \Rightarrow \\
&\qquad (\mathrm{pd} \in \mathrm{toInEpv(w)} \vee \mathrm{pd} \in \mathrm{toLogs(w)})) \wedge \\
&(\forall\ \mathrm{pd} : \mathrm{PayDetails} \bullet \\
&\quad \mathrm{val(pd)} \in \mathrm{ether(w)} \wedge \mathrm{pd} \in \mathrm{toInEpv(w)} \Rightarrow \\
&\qquad \mathrm{pd} \in \mathrm{fromInEpa(w)} \vee \mathrm{pd} \in \mathrm{fromLogs(w)}) \wedge
\end{aligned}
$$

Concrete Specification: World: invariant 6

$$
\begin{aligned}
&(\forall\ \mathrm{pd} : \mathrm{PayDetails},\ \mathrm{n} : \mathrm{Name} \bullet \\
&\quad \mathrm{exceptionLogResult}(\mathrm{n}, \mathrm{pd}) \in \mathrm{ether}(\mathrm{w}) \Rightarrow \\
&\qquad \mathrm{n} \in \mathbf{dom}\ \mathrm{allLogs}(\mathrm{w}) \wedge \mathrm{pd} \in \mathrm{allLogs}(\mathrm{w})(\mathrm{n})) \wedge \\
&(\forall\ \mathrm{pds} : \mathrm{PayDetailsSet1},\ \mathrm{n} : \mathrm{Name} \bullet \\
&\quad \mathrm{exceptionLogClear}(\mathrm{n}, \mathrm{image}(\mathrm{pds})) \in \mathrm{ether}(\mathrm{w}) \Rightarrow \\
&\qquad \mathrm{n} \in \mathbf{dom}\ \mathrm{archive}(\mathrm{w}) \wedge \mathrm{pds} \subseteq \mathrm{archive}(\mathrm{w})(\mathrm{n})) \wedge \\
&(\forall\ \mathrm{m} : \mathrm{Message} \bullet \\
&\quad \mathrm{m} \in \mathrm{visible}(\mathrm{w}) \Rightarrow \mathrm{m} \in \mathrm{ether}(\mathrm{w}))
\end{aligned}
$$

14 conjuncts which must be proved as invariant for 11 operations!!

The Argument for Correctness

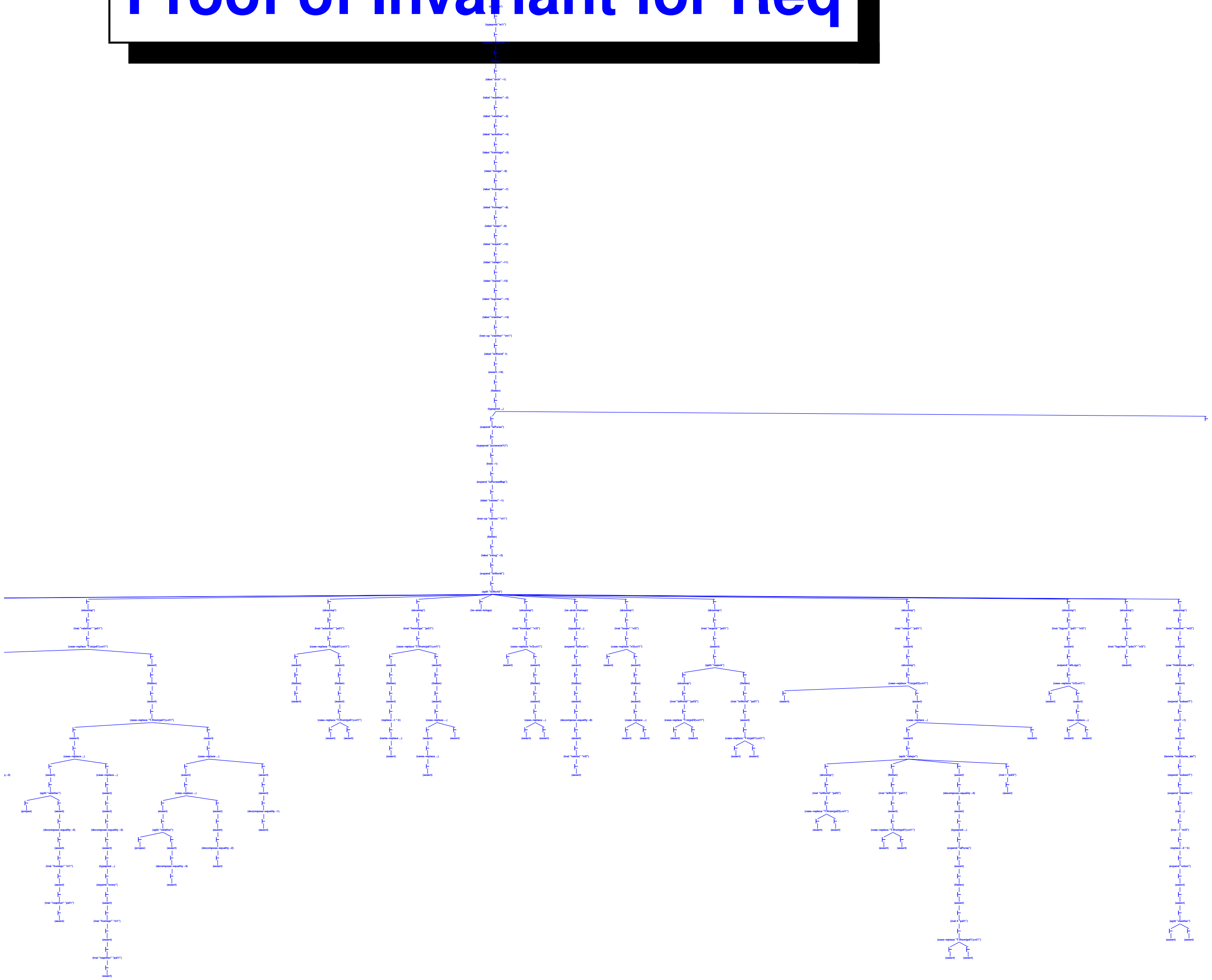
1. The abstract operations LeftTransfer, RightTransfer, Abort and No_op are *correct*; also sequence preserves correctness.
2. Each intermediate operation refines an abstract operation.
3. Each concrete operation refines the corresponding intermediate operation.

Easy!

Perhaps ...

- This is the 10th version of the specification, which is 2200 lines of RSL in 13 files.
- There are 366 proofs, perhaps half proved automatically.
- A typical invariant proof for the concrete specification is about 300 prover commands (recall there are 11 of these proofs).
- Other unpleasant proofs were that the concrete invariant implied the abstract one (150 prover commands), and that some sets defined by comprehension are finite.

Proof of Invariant for Req



Automation?

- The biggest problem is identifying the invariant.
  - Too strong: helps with proofs of refinement but can't be proved.
  - Too weak: easier to prove but refinement proofs fail.
- This problem has many large proofs with similar structure: tactics are worth developing.
  - The perfect tactic is very hard to write.
  - A tactic that does all the setting up of standard hypotheses, names them, and does the basic case analysis and tries to discharge the results can be very useful.
- One incautious `grind` generated (eventually) 1580 subgoals!

Did We Capture the Requirements Correctly?

- There may be many subtle points in 2200 lines of RSL!
- In the Z specification, for example, the description of a complete transfer is only informally stated, but seems to be unimplementable, because it requires you to know in advance a property of Abort that is underspecified (and perhaps nondeterministic): possible increase in nextSeqNo.
- Is there an "**axiom false**" somewhere?

Are our tools correct?

We rely on

- Translator from RSL to PVS
- PVS proof engine

Mondex in SAL

- Translated the concrete specification automatically from RSL to SAL.
- 8 versions to current one that runs with sal-smc (on a standard PC with 512MB of memory).
- Many changes to state structure, and reduced functionality.
- 3 versions used:
  - WORLD2 for correctness and liveness properties.
  - WORLD2INV for checking invariants.
  - CC version (special translation of WORLD2INV) for checking confidence conditions.

Why model check when you have a proof?

- Easier to do than proof.
- Could have found mistake in invariant in one version, and generally got more confidence that we were trying to prove things that are true.
- Can check confidence conditions, again before proof.
- Can show liveness properties, eg that a transfer is possible.

What was checked in SAL?

- Correctness:
  - All money is accounted.
  - The amount of money in circulation does not increase.
- World and purse invariants hold.
- Liveness in the sense that
  - An empty purse can become non-empty.
  - A non-empty purse can become empty.
  - Money can be lost.
- No confidence conditions violated.

but with only 2 purses and at most 3 transfers

Further work

- Drawing general conclusions for such systems:
  - modelling
  - checking
  - proving
- Can we improve the automation?
- Fix PVS 3.2 bug 894