Mutating the Purse

Susan Stepney

RefineNet workshop, York

September 2004

Mondex : Security properties (SP)

1. "No value created"

$$\Sigma\, £ \geq \Sigma\, £'$$

2. "All value accounted"

3. "This transfer permitted"
   (classes, etc)

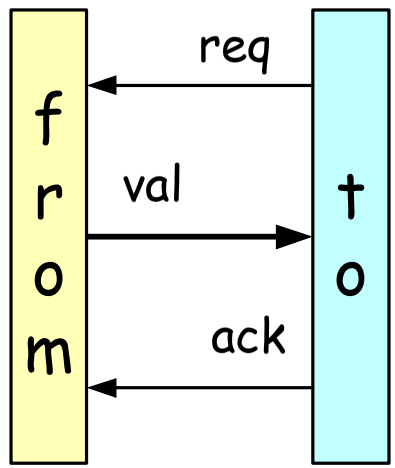
- SP comprises functional properties, which are preserved by refinement

http://www-users.cs.york.ac.uk/~susan/bib/ss/e6.htm

Z models

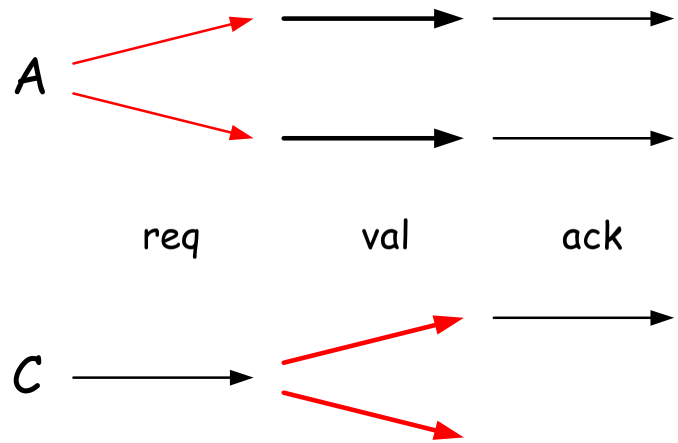
- Abstract model
  - promoted world of 'purses'
  - atomic value transfers
  - 'lost' component to account for failed transfers

- Concrete model
  - promoted world of 'purses'
  - n-step value transfer protocol
  - logging protocol to account for failures
  - ether of protocol messages



challenges

- backward refinement rules
  - resolution of non-determinism
- input/output refinement



- retrenchment opportunities
  - finite sequence number: $seqNo : \mathbb{N}$
  - finite exception log: $log : \mathbb{P}\ Details$
  - non-injective hash: $hash : \mathbb{P}\ Details \rightarrowtail Hash$
  - balance enquiry special state

welcome to the real world …

Formalist

"But I don't *need* retrenchment to solve this problem. I'll just *change the specification*, then I can use refinement!"

Customer

"*Oh no you don't*. We've already *implemented* it, and it would cost too much to do it again!"

"*Oh no you don't*. We have to do it that ISO-standard way."

"*Oh no you don't*. There's no *time*: we ship next week!"

"*Oh no you don't*. We need to use the *same* spec for *all* platforms - your change will result in one spec per platform!"

"*Oh no you don't*. The developers/testers/reviewers/certifiers/… won't understand the new spec."

"***Oh no you don't***."

and now for something completely different ...

the Purse in CSP

paper

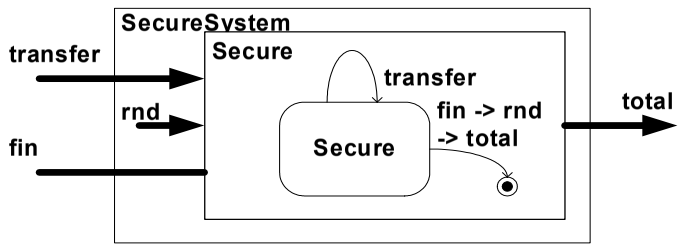
Thitima Srivatanakul, John A. Clark, Susan Stepney, Fiona Polack. Challenging formal specifications by mutation: a CSP security example. *APSEC-2003: 10th Asia-Pacific Software Engineering Conference, Chiang Mai, Thailand, December, 2003*. IEEE, 2003.

http://www-users.cs.york.ac.uk/~susan/bib/ss/occam/apsec03.htm

security property : a Secure system

$$
\begin{array}{l}
Secure = \\
\quad fin \rightarrow rnd?v \rightarrow total!(TotalBal - v).v \rightarrow Skip \\
\quad \Box\ transfer?from.to.val \rightarrow Secure \\
\\
SecureSystem = Secure \setminus \{|rnd|\}
\end{array}
$$

- ***transfer*** : from purse, to purse, and transfer value
- ***fin*** : finalisation
  - ***rnd*** : generate a random value, considered 'lost'
  - ***total*** : output total value stored, and lost, in the system
- ***SecureSystem***
  - no matter what value ***v*** is 'lost', the system is still secure
    - $(\ TotalBal - v\ ) + v = TotalBal$



The Purse

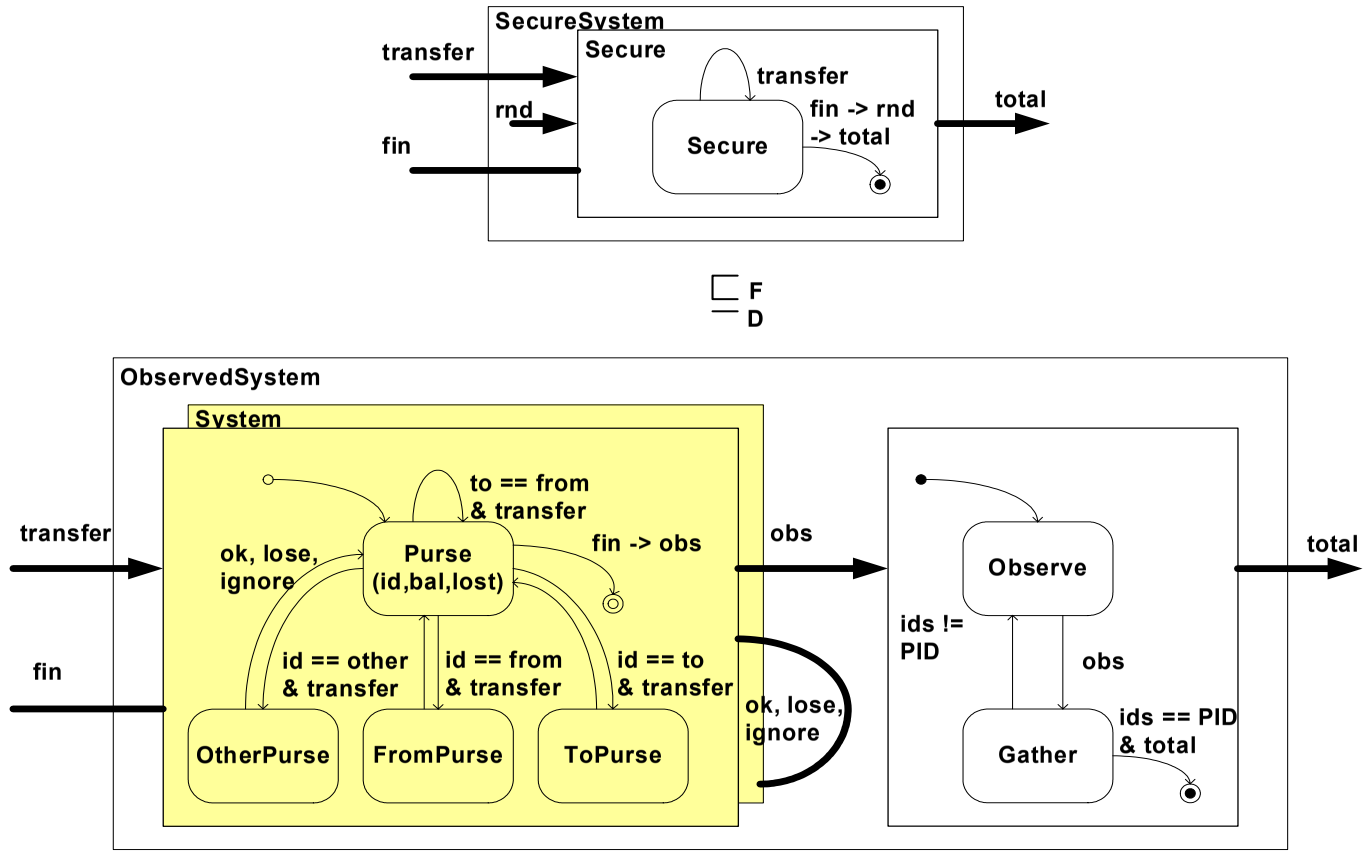
$$
\begin{array}{l}
Purse(id, bal, lost) = \\
\quad fin \rightarrow obs!id.bal.lost \rightarrow Skip \\
\quad \Box\ transfer?from.to.val \rightarrow \\
\qquad \text{if } to = from \\
\qquad\quad \text{then } Purse(id, bal, lost) \\
\qquad \text{else if } (id = from) \\
\qquad\quad \text{then } FromPurse(id, bal, lost, val) \\
\qquad \text{else if } (id = to) \\
\qquad\quad \text{then } ToPurse(id, bal, lost, val) \\
\qquad \text{else } OtherPurse(id, bal, lost)
\end{array}
$$

- ***transfer*** : behave like from purse, to purse, other purse
- ***fin*** finalisation
  - ***obs*** : output balance, and lost, in the purse

FromPurse, ToPurse, OtherPurse

$$
\begin{array}{l}
FromPurse(id, bal, lost, val) = \\
\quad \text{if } val \leq bal \text{ then } (\\
\qquad ok \rightarrow Purse(id, bal - val, lost) \\
\qquad \sqcap\ lose \rightarrow Purse(id, bal - val, lost + val) \\
\qquad \sqcap\ ignore \rightarrow Purse(id, bal, lost) \\
\quad) \text{ else } (ignore \rightarrow Purse(id, bal, lost)) \\
\\
ToPurse(id, bal, lost, val) = \\
\quad ok \rightarrow Purse(id, bal + val, lost) \\
\quad \Box\ lose \rightarrow Purse(id, bal, lost) \\
\quad \Box\ ignore \rightarrow Purse(id, bal, lost) \\
\\
OtherPurse(id, bal, lost) = \\
\quad ok \rightarrow Purse(id, bal, lost) \\
\quad \Box\ lose \rightarrow Purse(id, bal, lost) \\
\quad \Box\ ignore \rightarrow Purse(id, bal, lost)
\end{array}
$$

diagram of Purse specification



- ***ObservedSystem*** wrapper to specify security property
- also interested in (internal) behaviours of purse ***System***

Purses are secure

- The ***ObservedSystem*** is the system of purses, and a process to gather the totals on finalisation
  - with the internal channels hidden
- The ***ObservedSystem*** (the purse system specification) is a ***SecureSystem*** (has the security property)
  - it does not create value, and accounts for all lost value

assert
$$SecureSystem \sqsubseteq_{FD} ObservedSystem$$

- assertion checked with FDR2 model checker
  - needed to add (redundant) guard to a branch

program mutation

- test suite T, program S, S passes T
  - but how good is T?

- "mutate" S to S' : small syntactic variant
- $\neg$ S' passes T
  - so, modelled fault (mutation) captured by test suite T
- S' passes T

  $\neg\, S \equiv S'$ (different behaviours) : modelled fault *not* captured by *inadequate* test suite T

  $S \equiv S'$ (identical behaviours) : an "*equivalent mutant*"; tells us nothing about T
    - much effort goes into eliminating these useless equivalent mutants
- do *lots* of mutations -- needs a tool

specification mutation

- property P : is a ***SecureSystem***
- specification S : is a ***ObservedSystem***
- $\vdash$ ***S satisfies P***
  - in this example, relationship is refinement

- now mutate S to S'
- $\vdash \neg$ ***S′ satisfies P***
  - modelled fault identifies possible single-point vulnerability
- $\vdash$ ***S′ satisfies P*** : "equivalent mutant"

  $\neg S \equiv_i S'$: validation question: which behaviour is really required?

  $S \equiv_i S'$: *why* are they the same? has an abstraction/bhvr been missed?

mutating the Purse

- CSP mutation tool http://www-users.cs.york.ac.uk/~jill/Tool.htm
  - mutation operators
  - mutants are syntactically correct, usually type correct
- FDR2 model checker

- generated 579 mutant *S'* specs of ***ObservedSystem***
  - 241 compilation errors
  - 177 not trace refinements
  - 156 trace refinements, but FD violations
  - 23 FD refinements : "equivalent mutants "
    - 20 different (internal) behaviour mutants
    - 3 same (internal) behaviour

analysis of equivalent mutants

- mutants that are also refinements
  - $\vdash$ ***S' satisfies P***

- restricted behaviours from strengthened guards

  $val \leq balance \Rightarrow val < balance$
  - sometimes ignore the transfer request -- still secure!

  $val \leq balance \Rightarrow val \leq -balance$
  - *always* ignore the transfer request -- still secure!

  $to = from \Rightarrow to \leq from$
  - ignore the transfer request to certain purses -- still secure!

how many purses?

- $id = to \Rightarrow id \neq to$
  - transfer to "other"s instead of to "to" -- should *not* be secure, but FDR2 claimed a refinement -- why?
- we were model-checking a system of three purses
  - "to", "from", "other"
    - transfer to one "other" instead of to "to" -- still secure!
- so tried *four* purses in system
  - "to", "from", "other1", "other2"
    - transfer to both "other"s instead of to "to" -- *not* secure!
  - now not a refinement
- mutation helped us find right system size to check
  - testing should *also* use many "other" purses

conclusions

- test the model-checking restrictions
  - enough purses
- non-equivalent mutants
  - indicate vulnerable parts of the design
    - "one failure away" from insecurity
      - highlighted purse id as such a vulnerability
- equivalent mutants
  - robust design
    - same small errors in implementation also secure
  - challenge design decisions : why S and not S' ?
  - specification validation approach
    - "getting the right system"