Mondex, an Electronic Purse: Specification & Refinement Checks with the Alloy Model-Finding method

Tahina Ramananandro
École Normale Supérieure
Paris, France

Daniel Jackson
Massachusetts Institute of Technology
CSAIL Software Design
Cambridge MA, USA
Outline

- The Mondex Project
- Alloy Principles
- Technical Issues
- Results
- Using FOL theorem provers
- Conclusions
The Mondex Project

- Grand Challenges in Computer Science
- UK Computer Research Committee
  - Dependable Systems Evolution
    - Jim Woodcock, University of York
  - Verified Software Repository
    - several formal methods for machine-aided verification
      - Mondex Case Study
The Mondex Case Study

- An electronic purse (smart card) system
  - Replace physical coins with values stored in the card (not remotely: not a credit card)
- Highly critical security issues for banks
- Specified by hand in Z (Stepney, Cooper, Woodcock 2000)
  - Granted ITSEC security level 6 out of 6 (2001)
- Aim: machine-check this specification with automated formal methods
Mondex

Total balances not increasing
Total balances and lost constant

Abstract

From balance lost

atomic transfer

To balance lost

World
messages ("ether" = comm channel)
public archive of lost transactions

Between (constrained)

Concrete

From balance private archive

1.startFrom

2.startTo

3.req

4.val

5.ack

To balance private archive
Outline

- The Mondex Project
- Alloy Principles
- Technical Issues
- Results
- Using FOL Theorem Provers
- Conclusions
Alloy Spec Language & Logic

- Typed and modular specification language
- Sets and relations
  - Signatures define “basic” sets and relations
    - Can be abstract, extended ("inheritance" as in Java)
    - Typing, overloading, modularity
    - quite like Z schema extensions
  - Specification can be constrained
- Relational first-order logic + transitive closure

abstract sig Person {}  
sig Man extends Person {wife:set Woman}  
sig Woman extends Person {husband:set Man}  

fact Constraint {  
  all m:Man |  
  some m.wife implies m.wife.husband = m  
  all w:Woman |  
  some w.husband implies w.husband.wife = w  
}
Alloy relations vs. Z sets

- sets are unary relations
- scalars are singletons
Joining relations (.)

- Let \( \alpha \) and \( \beta \) be two relations
  - \( \text{sig U \{alpha : set X\}} \)
  - \( \text{sig X \{beta : set V\}} \)
  - \( \text{sig V} \)
Joining relations (.)

- Let $\alpha$ and $\beta$ be two relations
- so we define $\alpha \cdot \beta$ the *joined relation*
  - Cf. database $\triangleright \triangleleft$
- We may write $u_2.(\alpha . \beta) = v_1 + v_3$, it is the same join operator because:
  - sets are unary relations
  - scalars are singletons
Alloy Analyzer, a Model Finder

- Specification Analysis by Model Finding
  - "Run" predicate: find example
  - Check assertion: find counterexample
- "Scope" required: bounded finite models
  - Number of objects for each signature
  - Can show theorems hold in specified scope

pred Married (p:Person) {some p.(wife+husband)}

pred Simulation () {some p:Person|Married(p)}
run Simulation for 18 Man, 1 Woman

assert Theorem {
  all p:Person|lone p.(wife+husband)
  all p,q:Person|p.husband=q iff q.wife=p }
check Theorem for 7
A naive attempt

sig NAME {}

[ NAME ]
A naive attempt

\[\text{sig NAME} \{}\]

\[\text{[ NAME ]}\]

\[\text{AbPurse}\]

\[\text{balance, lost : N}\]

\[\text{sig AbPurse} \{\text{balance, lost : Int}\}\]
A naive attempt

[ NAME ]

AbPurse

balance, lost : N

AbWorld

abAuthPurse : NAME \rightarrow AbPurse

sig NAME {}

sig AbPurse {balance, lost : Int}

pred Abstract (abAuthPurse:NAME-->Purse) {
  -- functional
  all n:NAME | lone n.abAuthPurse
}

sig AbWorld {abAuthPurse: NAME -> AbPurse}

fact AbWorldConstr {
  all a : AbWorld | Abstract (a.abAuthPurse)
}
A naive attempt

Unable to express finiteness: ignore
A naive attempt

```
sig NAME {}

sig AbPurse {balance, lost: Int}
pred Abstract (abAuthPurse: NAME->Purse) {
  -- functional
  all n:NAME | lone n.abAuthPurse
}
sig AbWorld {abAuthPurse: NAME -> AbPurse}
fact AbWorldConstr {
  all a: AbWorld | Abstract (a.abAuthPurse)
}
pred AbIgnore (a,a’:AbWorld) {
  a’.abAuthPurse = a.abAuthPurse
}```
Outline

- The Mondex Project
- Alloy Principles
- Technical Issues
- Results
- Using FOL Theorem Provers
- Conclusions
Refinements : checking method

- Follow Z spec strategy (A/B backwards, B/C forwards)
  - But separate existence and refinement

- Rbc_constr : equality predicates (explicit "construction")
  - Not necessary for RabCl (already in this form)
Integers in Alloy

- Integers in Alloy are heavy
  - Builds boolean circuits for +, <
  - Expensive operations

- So, avoid them
  - Not all properties of N used
  - Determine which
  - Pick most lightweight repr that works
Representing SEQNO

• Sequence numbers just require total order
  – No operations
  – Even no successor

• Simply use Alloy’s ordering module
Representing amounts

- Sets of coins

<table>
<thead>
<tr>
<th>$\mathbb{Z}$</th>
<th>Alloy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers</td>
<td>Sets of coins</td>
</tr>
<tr>
<td>Equality</td>
<td>Set equality</td>
</tr>
<tr>
<td>Ordering</td>
<td>Set inclusion</td>
</tr>
<tr>
<td>Sum</td>
<td>Set union</td>
</tr>
<tr>
<td>Difference</td>
<td>Set difference</td>
</tr>
</tbody>
</table>

- OK, because no comparison between purses
  - Globally: coins between whole worlds
  - Locally: between a purse balance & a payment

- Add constraints to avoid coin sharing
Existential issue

- Can’t guarantee object exists for every combination of field values
  - The empty model
  - To enforce existence with algebraic constraints would dramatically increase scope

Solution:

- Instead of $\exists$, construct explicit witness:
  all c, c’, a | some a’ | P (c, c’, a, a’)
  becomes
  all c, c’, a |
  let a’ = F(c, c’, a) | P(c, c’, a, a’)
- Requires to get rid of global constraints
  • Integrate them into theorems
The identity of objects

- Z: schemas define records
- Alloy: signatures define atomic objects
  - Objects have an *identity*
    - Notion does not exist in Z
  - Suitable for names, coins
- Two objects with same field values may be distinct
  - Naive solution: impose equality constraint

```plaintext
fact {
  no disj a1,a2:AbPurse {
    a1.balance=a2.balance
    a1.lost=a2.lost
  }
}
```
The identity of objects

- Smoother solution: represent purses and states as standalone objects rather than records
  - No names

```
sig Coin

sig AbPurse {balance, lost: Coin->AbWorld}

sig AbWorld {abAuthPurse : set AbPurse}

pred AbIgnore (a,a’:AbWorld) {
  a’.abAuthPurse = a.abAuthPurse
  all p : AbPurse | p in a.abAuthPurse implies {
    p.balance.a’ = p.balance.a
    p.lost.a’ = p.lost.a
  }
}
```
Outline

- The Mondex Project
- Alloy Principles
- Technical Issues
- Results
- Using FOL Theorem Provers
- Conclusions
Bugs found in Z Specification

- Missing authenticity constraints
  - Spurious cases where purses deal with irrelevant transactions are not eliminated

- Wrong proof steps
  - Wrong assumption made by informal comments
  - 2 bugs with this form
Alloy’s Approach Summary

- Refinement checks with model finding
  - Try to find $c$, $c'$, $a$, $a'$ such that $\text{Rac}(a, c) \& \text{Rac}(a', c') \& \text{COp}(c, c')$ hold but not $\text{AOp}(a, a')$

- Original approach
  - Quite high confidence level
  - Not as high as theorem proving
  - but much cheaper!
Choosing scopes

- Must be enough for quantifications
- Started with 10
  - worked fine with Abstract theorems
  - too long for more complex theorems
    - SAT-solving time exponentially grows with the scope
    - SAT solver crashed for refinement checks
  - so grow scope incrementally
- Achieved scope of 8 for most theorems eventually
  - restricted scope for Worlds is complete
Almost everything represented

- Alloy modules close to Z specification
  - *Representation* size is comparable
  - Alloy Proof size is negligible
    - Actually no proof details in Alloy modules
- Only changes:
  - Integer representation
  - Unable to express infiniteness in Alloy
    - finiteness properties ignored
- Fits first order logic
  - No transitive closures needed
Outline

- Alloy Principles
- Mondex in Alloy: General Method
- Technical issues
- Results
- Using FOL Theorem Provers
- Conclusions
The direct attempt

- FOL atoms are Alloy atoms
  - But Alloy predicates take arbitrary relations as arguments
  - So they have to be inlined
  - Formulae become huge

- Simplifications to decrease formula size
  - Eliminate redundancy with subsumption tests
  - Split theorems through
  - Attempt to reach a normal form
    - Does not terminate

- Very few results:
  - Proved theorems relative to the abstract world (atomic transactions) alone
The “lifted” attempt

- FOL atoms are Alloy relations
- Axiomatize relational algebra
  - Bound arities according to spec in Alloy
- Problems:
  - Trouble to prove obvious-looking general theorems such as:
    - The Cartesian product of two atoms is a singleton of arity 2
  - Would have to prove intermediate lemmas
  - Loss of automation
- No significant results
Outline

- Alloy Principles
- Mondex in Alloy: General Method
- Technical issues
- Results
- Using FOL Theorem Provers
- Conclusions
General observations

- High level checking
  - Proof structure not needed: automated
  - But need to provide explicit witness for $\exists$

- SAT-Solving duration varies
  - From seconds to hours (even days!)
  - Time correlated with theorem importance?
Alloy Limitations

- **FOL and Finiteness**
  - Cannot express infiniteness
  - But in practice, world of purses finite

- **Alloy Analyzer’s analysis is bounded**
  - Results valid only on given scope
  - Is scope of 8 enough?

- **Enough for industry?**
  - Much less effort than theorem proving
  - But problems with critical security issues need a proof
Personal Experience

• Learn Z and Alloy *from scratch*

• Nice :
  – Language easy to understand
    • no $\Delta$/∃/graphical issues
  – Though quite close to Z
  – Expressive & smooth relational logic

• Nasty :
  – Signatures are not records
    • Equality & Existential theorems
  – Resource- and time-consuming SAT-Solving
    • Very long time for obvious-looking theorems (easily provable by hand, e.g. Ignore refinements)
    • Perhaps syntactic pre-analysis would help?
Lessons

- Learn another verification approach
  - Automation does not exclude proof formalism
- Even though not theorem proving
  - But allows also checking informal comments
- Discover problems more quickly
  - Alloy Analyzer allows finding several bugs
  - Counterexample gives useful information when bug found
Future work

- Argue small model theorem (Momtahan 2004)?
- Improve checking with FOL theorem provers
  - To expect better FOL theorem provers is quite hopeless: undecidable
  - Better model Alloy into FOL
  - Fit into decidable sublogic?
- Tackle finiteness
  - HOL necessary at first sight
  - Use incomplete FOL theories?
- Interface Alloy method with others
Acknowledgments

- At MIT:
  - The SDG group, in particular Daniel Jackson
  - But also the CRS group, in particular Viktor Kuncak and Charles Bouillaguet

- At ENS:
  - Patrick Cousot, who gave me the opportunity to follow the internship

- At RAL:
  - Jim Woodcock and Juan Bicarregui, for their hospitality
Any questions?

- **E-mail addresses**
  - ramanana@mit.edu Tahina Ramananandro
  - dnj@mit.edu Daniel Jackson

- **Alloy modules available at** :
  - http://www.eleves.ens.fr/~ramanana/work/mondex

- **Alloy Website** :
  - http://alloy.mit.edu