Automatically Discharging VDM Proof Obligations

IPA Lentedagen: Checking models
Sander Vermolen
May 7, 2008
This work

Thesis at
Radboud University Nijmegen

- Jozef Hooman – Radboud University Nijmegen
- Peter Gorm Larsen – Engineering College of Århus
- Frits Vaandrager – Radboud University Nijmegen
- John Fitzgerald – Newcastle University
Outline

• Domain & Goals
• Approach
• Translation
• Proof
• Results & Concluding remarks
Arbitrary code

boolean isPalindrome(String in) {
    if (in.length() == 0)
        return true;
    else {
        int lastIndex = in.length() - 1;
        return in.charAt(0) == in.charAt(lastIndex)
            &&
            isPalindrome(in.substring(1, lastIndex));
    }
}

madam

racecar

testset
boolean isPalindrome(String in) {
    if (in.length() == 0)
        return true;
    else {
        int lastIndex = in.length() - 1;
        return in.charAt(0) == in.charAt(lastIndex)
                && isPalindrome(in.substring(1, lastIndex));
    }
}
Model Inconsistencies - VDM++

Color = <red> | <yellow> | <green>;
turnTo (..., <purple>);

turnTo(
    mk_TrafficLight(<red>),
    <red>
)

turnTo(x, <red>)
Model inconsistencies - Characteristics

- Trigger run-time errors
- Detectable using only the model
Model inconsistencies - Classes
Model inconsistencies - Prevention

- Testing
  - Completeness
- Proof
  - Limited number of types of inconsistencies
  - Using proof obligations
Proof Obligations

Model

Integrity checker

Proof Obligation 1

Proof Obligation 2

Proof Obligation 3

Proof Obligation 4

True or False?
Theorem provers

- Theorems
- Proof search
  - Interactive
  - Automatic
- Tactics
Theorem provers - HOL

- Higher Order Logic (version 4)
  - Small axiom base
  - Adding theorems by proof only
  - Large number of libraries
Goal

Automating (as far as possible) the discharging of proof obligations generated by the integrity examiner

1. VDM++ to HOL translation
2. Automated proof of the Proof Obligations
Outline

- Domain & Goals
- **Approach**
- Translation
- Proof
- Results & Concluding remarks
Architecture

1. Preparation
2. Translation
3. Proof attempts

VDM+ + Model 1 2 HOL4 Model 3 Consistency Proof
Architecture - Translation
Architecture - Proof

Diagram:
- Abstract HOL Model & Proof obligations
  - VDM specific tactics
  - Built-in HOL tactics
  - HOL tactics selection
    - HOL code generation
    - Concrete HOL model
      - HOL engine
        - Proof of obligations
Outline

- Domain & Goals
- Approach
- Translation
- Proof
- Results & Concluding remarks
Translation - Types

\[ \langle T_1 \ast T_2 \ast \cdots \ast T_n \rangle = \langle T_1 \rangle \ast \langle T_2 \rangle \ast \cdots \ast \langle T_n \rangle \]

\[ \langle \text{map } T_d \text{ to } T_r \rangle = (\langle T_d \rangle \vdash \rightarrow \langle T_r \rangle) \]

\[ \langle T_1 \ast T_2 \ast \cdots \ast T_{n-1} \rightarrow T_{n+1} \rangle = \langle T_1 \rangle \ast \langle T_2 \rangle \ast \cdots \ast \langle T_n \rangle \rightarrow \langle T_{n+1} \rangle \]
Translation - Expressions

\[ \langle \text{Identifier} \rangle = \text{Identifier} \]

\[ \langle \text{if } P \text{ then } E_1 \text{ else } E_2 \rangle = \text{if } \langle P \rangle \text{ then } \langle E_1 \rangle \text{ else } \langle E_2 \rangle \]

\[ \langle (E_f E_{P_1}, \ldots, E_{P_n}) \rangle = \langle E_f \rangle (\langle E_{P_1} \rangle) \ldots (\langle E_{P_n} \rangle) \]
Translation - Complications

- Type management
  - Type definition using existing type vs no type definition
  - Translating invariants
- Partiality
- Patterns
  
```ml
let mk_(a, b, 3, c) = mk_(1, 2, 3, 4)
in ... 
```
- Dependencies
Proof – Domain checking

- Domain checking
- Subtype checking
- Satisfiability of implicit definitions
- Termination

factorial(x) ==

if x = 0 then 1
else x * factorial(x – 1)

pre x >= 0

∀x:int.(pre_factorial(x) ∧ x ≠ 0) ⇒ pre_factorial(x – 1)
∀x:int.(x ≥ 0 ∧ x ≠ 0) ⇒ (x – 1) ≥ 0
Proof – Domain checking

(! val:num too:ind frm:ind wrld:AbWorld.(T ==> ((((((\x y . ~ (x = y)) frm too) \ (frm IN (FDOM wrld.abPurses) ) ) \ (too IN (FDOM wrld.abPurses) ) ) \ ((GetBalance (FAPPLY wrld.abPurses frm)) >= val ) ) ==>) (let RESULT = (let newFrm = (ReduceBalance (FAPPLY wrld.abPurses frm) val) and newTo = (IncreaseBalance (FAPPLY wrld.abPurses too) val) in (make_AbWorld wrld.authentic ((\x y . FUNION y x) wrld.abPurses ((FEMPTY |+ (too,newTo) ) |+ (frm,newFrm) ))) in (((GetTotal (FAPPLY RESULT.abPurses frm)) + (GetTotal (FAPPLY RESULT.abPurses too)) ) \ (GetBalance (FAPPLY RESULT.abPurses frm)) + (GetBalance (FAPPLY RESULT.abPurses too)) ) ) \ (GetBalance (FAPPLY RESULT.abPurses too)) ) ) ==>) (! name.(((name IN ((FDOM RESULT.abPurses) DIFF {frm;too} ) ) \ T ) ==>) (((GetBalance (FAPPLY wrld.abPurses name)) = (GetBalance (FAPPLY RESULT.abPurses name)) ) ==>) (name IN (FDOM RESULT.abPurses) ) ) ))) )))

- TAUT_TAC
- MESON_TAC
- DECIDE_TAC
- VDM_ARITH_TAC
- REDUCE_TAC
Proof – Subtype checking

- Domain checking
- Subtype checking
- Satisfiability of implicit def.
- Termination

**Types:**
- specialNat = nat
  - inv x == x <> 2

**Functions:**
- sum : specialNat * specialNat -> specialNat
  - sum (x, y) ==
    - x + y;

\[ \forall x: \text{specialNat}, y: \text{specialNat}. \text{inv\_specialNat}(x + y) \]
\[ \forall x: \text{specialNat}, y: \text{specialNat}. (x + y) \neq 2 \]
Proof – Subtype checking

Simplification support

• Stateful simplification
• Stateless simplification

Decision support

• Pre-defined theorems
• Custom theorems
Proof – Satisfiability

- Domain checking
- Subtype checking
- *Satisfiability of implicit def.*
- Termination

\[
\forall x: \text{real}. \text{pre}\_\text{sqrt}(x) \rightarrow \exists r: \text{real}. \text{post}\_\text{sqrt}(x, r)
\]

\[
\forall x: \text{real}. x \geq 0 \rightarrow \exists r: \text{real}. r \cdot r = x
\]

\[
\text{sqrt} (x : \text{real}) r : \text{real}
\]

\[
\text{pre} \; x \geq 0
\]

\[
\text{post} \; r \cdot r = x;
\]
Proof – Termination

- Domain checking
- Subtype checking
- Satisfiability of implicit definitions
- *Termination*
Results

- 15 case studies
- 4 significant case studies

<table>
<thead>
<tr>
<th>Category</th>
<th># valid obligations</th>
<th># proved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain checking</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>Subtype checking</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>Satisfiability of implicit definitions</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
Conclusions

Translation

- Correctness of translation
- Correctness of implementation
- Object orientation

Proof

- Relative high rate of success
- Time efficient
Further research

- Extension of translation
- Extension of tactic
- New concepts
  - Operational semantics
  - Validation conjectures
  - User guided proof
www.overturetool.org

www.overturetool.org/twiki/bin/view/Main/AutomaticProof