Verification Conditions and Theorem Proving

Joe Hurd

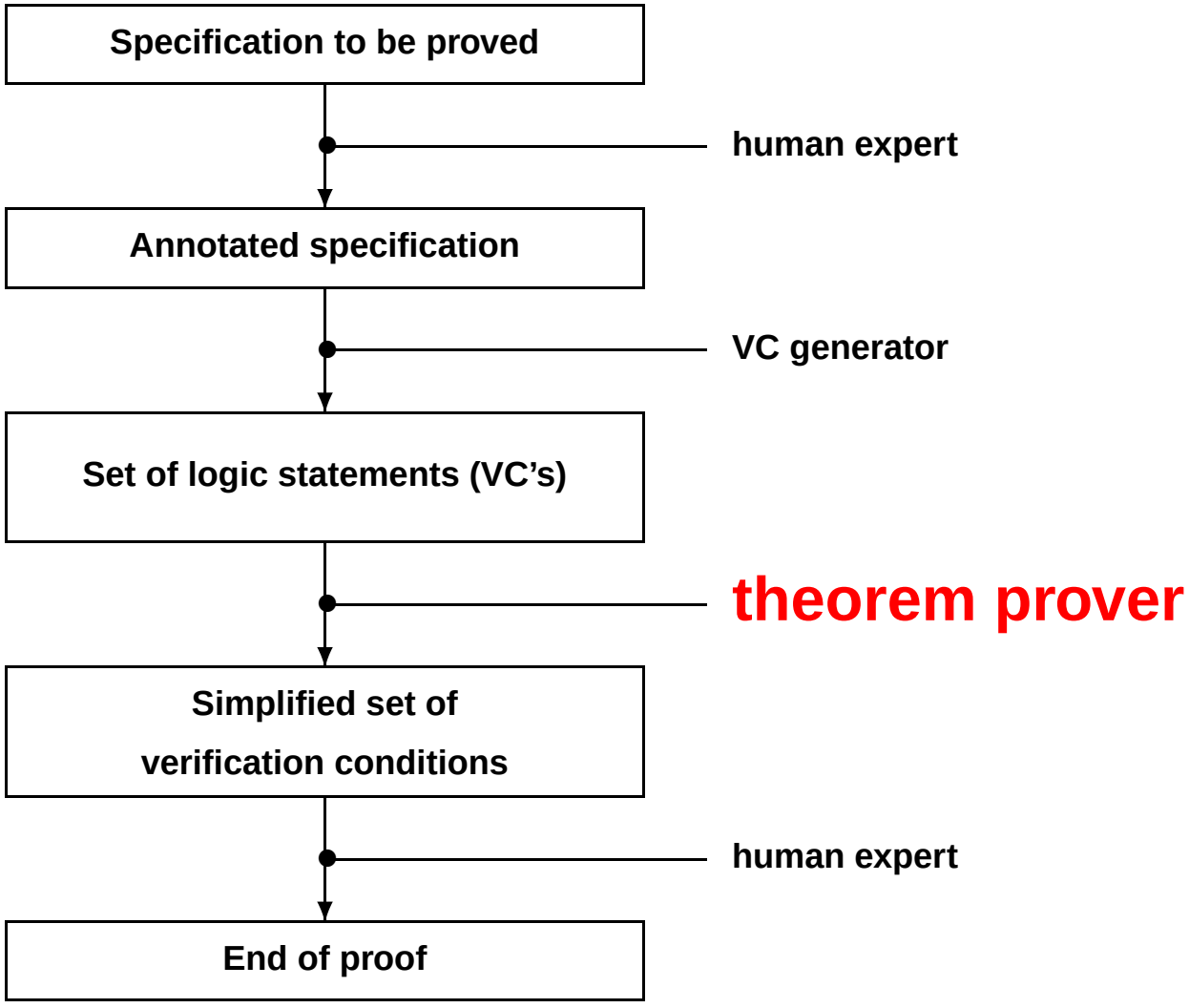
joe.hurd@comlab.ox.ac.uk.

Specification & Verification I

Part II of the Computer Science Tripos

University of Cambridge

Verification Architecture



Example: Specification

ESC/Java is a verification system that allows users to annotate Java programs with specifications, such as this program to sort an array of two integers:

```
void sort2(int[] a) {
//@ requires a != null && a.length == 2
//@ ensures a[0] <= a[1]
  if (a[0] > a[1]) {
    int t = a[0];
    a[0] = a[1];
    a[1] = t;
  }
}
```

Do we need a theorem prover to tackle the resulting VC?

Example: Resulting VC

```
(FORALL (t1) (FORALL (t2) (IMPLIES (AND (NEQ a null) (EQ (arrayLength
a) 2)) (AND (NEQ a null) (AND (AND (<= 0 0) (< 0 (arrayLength a)))
(AND (NEQ a null) (AND (AND (<= 0 1) (< 1 (arrayLength a))) (AND
(IMPLIES (> (select (select elem a) 0) (select (select elem a) 1))
(FORALL (t3) (AND (NEQ a null) (AND (AND (<= 0 0) (< 0 (arrayLength
a))) (FORALL (t) (IMPLIES (EQ t (select (select elem a) 0)) (FORALL
(t1) (IMPLIES (EQ t1 a) (AND (NEQ a null) (AND (AND (<= 0 1) (< 1
(arrayLength a))) (AND (NEQ t1 null) (AND (AND (<= 0 0) (< 0
(arrayLength t1))) (FORALL (t2) (IMPLIES (EQ t2 a) (AND (NEQ t2 null)
(AND (AND (<= 0 1) (< 1 (arrayLength t2))) (AND (<= (select (select
(store (store elem t1 (store (select elem t1) 0 (select (select elem
a) 1))) t2 (store (select (store elem t1 (store (select elem t1) 0
(select (select elem a) 1))) t2) 1 t)) a) 0) (select (select (store
(store elem t1 (store (select elem t1) 0 (select (select elem a) 1)))
t2 (store (select (store elem t1 (store (select elem t1 0 (select
(select elem a) 1))) t2 1 t)) a) 1)) (EQ true true))))))))))))))))
(IMPLIES (NOT (> (select (select elem a) 0) (select (select elem a)
1))) (AND (<= (select (select elem a) 0) (select (select elem a) 1))
(EQ true true)))))))))))
```

Yes, we need a theorem prover.

Verification Conditions In The Wild

- Most VCs are “big but dumb”.
- Arithmetic VCs are common.
  - These arise from conditions on numeric variables.
  - Example: array bounds checking.
- Equality VCs are also common.
  - These arise from conditions on data structures:

$$\forall m, a, v.\ \mathsf{load}(\mathsf{store}(m, a, v), a) = v$$

  - Example: checking references are non-null.
- Many VCs are trivially true, such as

$$n + 1 < n \Rightarrow \text{[huge formula]}$$

Theorem Prover: Requirements

1. Must interpret formulas of integer arithmetic.
2. Must handle equality of uninterpreted function symbols.
   - Uninterpreted functions are described by axioms.
   - Example: $\forall\, h, t.\ \mathsf{hd}(\mathsf{cons}(h, t)) = h$
3. Must be able to combine theorem provers for different theories.
   - One VC can refer to multiple theories.
   - Example: arithmetic on array indexes.
4. Must prove a wide range of VCs fully automatically with no user assistance.
   - In some applications (such as a compiler), it must always work fully automatically.

Theorem Prover: Technique 1

- Presburger arithmetic (a.k.a. linear arithmetic).
- First order formulas with signature $(\mathbb{Z}, +, =)$.
- Can define *syntactic sugar* such as constants $0, 1, 2$, functions $+, -$, relations $\leq, <, >, \geq$, even, odd and multiplication by a **constant**.
- Example:

$$\forall\, n > 7.\ \exists\, i, j \geq 0.\ 3i + 5j = n$$

- Shown to be decidable by Presburger in 1930.

Theorem Prover: Technique 1

- Decision procedure is by quantifier elimination.
- Base case: formulas with no quantifiers are just arithmetic expressions with numbers as arguments, so can simply be evaluated.
- Eliminating a quantifier: the simple case

$$\begin{aligned} \exists x.\ \phi(x) &\equiv \exists x.\ (\textstyle\bigwedge_i s_i \leq x) \wedge (\bigwedge_j x \leq t_j) \\ &\equiv \textstyle\bigwedge_{i,j} s_i \leq t_j \end{aligned}$$

- In the general case, the inequalities will all refer to cx for some integer constant c, and we will have to introduce divisibility constraints.

Theorem Prover: Technique 2

- Equality of uninterpreted functions.
- Quantifier free first order formulas with signature $(D, \mathcal{F}, =)$.
  - Variables are implicitly universally quantified.
  - Example:

$$f(f(f(f(f(x))))) = x$$
$$\wedge\ f(f(f(x))) = x$$
$$\Rightarrow\ \ f(x) = x$$

- Shown to be decidable by Ackermann in 1954.

Theorem Prover: Technique 2

Procedure to decide the validity of the formula

$$\forall\, x_1, \ldots, x_n.\ \phi(x_1, \ldots, x_n)$$

1. Equivalent to deciding the unsatisfiability of

$$\exists\, x_1, \ldots, x_n.\ \neg\phi(x_1, \ldots, x_n)$$

2. Skolemize: $\neg\phi(c_1, \ldots, c_n)$.
3. Convert to DNF: $Q_1 \vee \cdots \vee Q_k$.
   - If any Q_i is satisfiable then so is the disjunction.
4. Each Q_i is a conjunction of literals, like so:

$$s_1 = t_1 \wedge \cdots \wedge s_n = t_n \wedge u_1 \neq v_1 \wedge \cdots \wedge u_m \neq v_m$$

Theorem Prover: Technique 2

$$
\begin{aligned}
Q_i &\equiv s_1 = t_1 \wedge \cdots \wedge s_n = t_n \wedge u_1 \neq v_1 \wedge \cdots \wedge u_m \neq v_m \\
&\equiv \neg(s_1 = t_1 \wedge \cdots \wedge s_n = t_n \Rightarrow u_1 = v_1) \\
&\quad \wedge \ \cdots \\
&\quad \wedge \neg(s_1 = t_1 \wedge \cdots \wedge s_n = t_n \Rightarrow u_m = v_m)
\end{aligned}
$$

We decide this type of formula using congruence closure:

1. Create a class for each subterm of the formula.
2. Merge classes corresponding to s_i and t_i.
3. For every function symbol f, merge the classes containing $f(s)$ and $f(t)$ if s and t are in the same class.
4. When all the merging is finished, check whether u_j and v_j are in the same class.

Theorem Prover: Technique 3

- Given theorem provers for theories T_1 and T_2, combine them to get a theorem prover for theory $T_1 \cup T_2$.
- Nelson-Oppen combination of theorem provers:
  - run each theorem prover separately;
  - if one finds a contradiction, then finished;
  - otherwise get them to return any equalities found;
  - and restart the process with the extra facts.
- Example combining arithmetic and equality:

$$f(f(x) - f(y)) \neq f(z) \ \wedge \ y \leq x \ \wedge \ x \leq y + z \ \wedge z \ \leq 0$$

- Warning: Can't combine theories that interpret the same function symbol!

Theorem Prover: Technique 3

- Problem: non-convex theories.
- A theory is non-convex if there is a set of literals that imply a disjunction of equalities without implying a single equality.
- Example with arithmetic and equality:

$$1 \leq x \leq 2 \ \wedge \ a = 1 \ \wedge \ b = 2 \ \wedge \ f(x) \neq f(a) \ \wedge \ f(x) \neq f(b)$$

  - Entails $x = 1 \vee x = 2$ but no single equality.
  - No contradiction found, but the formula is unsatisfiable.
- Solution: split proof state on disjunction of equalities.
  - A very expensive operation :-(

Theorem Proving In The Wild

- The Simplify theorem prover is used to prove VCs from ESC/Java.
- It uses all the techniques covered in this lecture.
- From a paper on Simplify by Detlefs, Nelson and Saxe:

  > "With the released version of Simplify, ESC/Java is able to check the 44794 thousand lines of Java source in its own front end (comprising 2331 routines and 29431 proof obligations) in 91 minutes. This is much faster than the code could be checked by a human design review, so we feel we have succeeded."