REACH

Colin Runciman, U. of York

(joint work with Matt Naylor; thanks also to Mark Tullsen)

The problem Reach solves

GIVEN a program with some top-level functions marked as **sources** and some expressions marked as **targets**

FIND for each source the **simplest applications** that entail evaluation of one or more targets

Examples: property refutation

```
property1 x t =
    ordered t ⇒ ordered (delete x t)

property2 c s =
    ok result ⇒ result ≡ exec (opti c) s
    where result = exec c s

True ⇒ False = False
_ ⇒ False = True
```

SOURCES

TARGET

Core object language for Reach

$p ::= \overline{d}$	program
$d ::= f\ \overline{v} = e$	definition
$e ::= v$ *	variable
$\mid f\ \overline{e}$	application
$\mid c\ \overline{e}$ *	construction
$\mid$ case e of $\overline{a}$	
$\mid$ let $\overline{b}$ in e	
$\mid \triangleleft e \triangleright$ *	target
$a ::= c\ \overline{v} \mapsto e$	alternative
$b ::= v = e$	binding

* NORMAL FORM

Other notation

$fresh(f) = (f\,\bar{v} = e)$	from p, with fresh vars.
$\sigma(v) = e$	σ partial map vars. to expns.
$\sigma[v \mapsto e]$	extension
$\langle \sigma, e \rangle$	configuration
$\diamond$	configuration variable
$\langle \sigma, e \rangle \to \langle \sigma', e' \rangle$	relation, eval. to normal form
$\langle \emptyset, f\,\bar{v} \rangle \twoheadrightarrow$ YES σ	$f\,\sigma(\bar{v})$ reaches target(s)
$\twoheadrightarrow$ NO	no target reached from f

Evaluation rules (I)

$$\langle \sigma, c\,\bar{e} \rangle \to \langle \sigma, c\,\bar{e} \rangle \quad \text{(cons)}$$

$$\langle \sigma, \lozenge e \blacklozenge \rangle \to \langle \sigma, \lozenge e \blacklozenge \rangle \quad \text{(targ1)}$$

NON-DETERMINISM

$$\frac{\langle \sigma, e \rangle \to \Diamond}{\langle \sigma, \lozenge e \blacklozenge \rangle \to \Diamond} \quad \text{(targ2)}$$

$$\frac{\langle \sigma, \sigma(v) \rangle \to \langle \sigma', e \rangle}{\langle \sigma, v \rangle \to \langle \sigma'[v \mapsto e], e \rangle} \quad \text{(var1)}$$

if $v \in dom(\sigma)$

SHARING ✓ ∞ STRUCTURE ✓
BLACK-HOLE DETECTION

$$\langle \sigma, v \rangle \to \langle \sigma, v \rangle \quad \text{(var2)}$$

if $v \notin dom(\sigma)$

UNINSTANTIATED VARS. NORMAL

Evaluation rules (II)

$$\frac{\langle \sigma[\bar{v} \mapsto \bar{e}], e\rangle \to \Diamond}{\langle \sigma, f\,\bar{e}\rangle \to \Diamond} \text{(App)}$$

BOUNDED RECURSION

if $(f\,\bar{v} = e) = \text{fresh}(f)$

$$\frac{\langle \sigma[\bar{b}], e\rangle \to \Diamond}{\langle \sigma, \text{let } \bar{b} \text{ in } e\rangle \to \Diamond} \text{(Let)}$$

RECURSIVE LET ✓

NEXT

$$\frac{\langle \sigma, e\rangle \to \langle \sigma', e'\rangle, \langle \sigma', e', \bar{a}\rangle \to \Diamond}{\langle \sigma, \text{case } e \text{ of } \bar{a}\rangle \to \Diamond} \text{(Case)}$$

CASE STRICT IN SUBJECT

Evaluation rules (III)

$$\langle \sigma, \lhd e \rhd, \bar{a} \rangle \to \langle \sigma, \lhd e \rhd \rangle \quad \text{(Subject)}$$

$$\frac{\langle \sigma[\bar{v} \mapsto \bar{e}], e \rangle \to \Diamond}{\langle \sigma, c\,\bar{e}, \bar{a}[c\,\bar{v} \mapsto e] \rangle \to \Diamond} \quad \text{(Match)}$$

BOUNDED CONSTRUCTION DEPTH

$$\frac{\langle \sigma[v \mapsto c\,\bar{v}], e \rangle \to \Diamond}{\langle \sigma, v, \bar{a}[c\,\bar{v} \mapsto e] \rangle \to \Diamond} \quad \text{(Narrow)}$$

NON-DETERMINISM

Evaluation rules (IV)

$$\frac{\Diamond \to \langle \sigma, \llparenthesis e \rrparenthesis \rangle}{\Diamond \to\!\!\!\!\to \text{YES } \sigma} \quad (\text{Targ}*)$$

TOP-LEVEL DEMAND
FULL L. TO R. EVAL'N

$$\frac{\Diamond \to \langle \sigma_0, C\,\bar{e} \rangle,\ \forall i\,(1 \leqslant i < k)\ \langle \sigma_{i-1}, e_i \rangle \to\!\!\!\!\to \rho_i\,\sigma_i,\ \langle \sigma_{k-1}, e_k \rangle \to\!\!\!\!\to \text{YES } \sigma_k}{\Diamond \to\!\!\!\!\to \text{YES } \sigma_k} \quad (\text{Cons}*)$$

$$\frac{\Diamond \to \langle \sigma, v \rangle}{\Diamond \to\!\!\!\!\to \text{No } \sigma} \quad (\text{Var}*)$$

Bounded computation in Reach

* CONSTRUCTION DEPTH is bounded by an extra condition on the Narrow rule — the only place input variables are instantiated.
* DEPTH LIMITS are attached to variables.
* RECURSION DEPTH is bounded by an extra condition on the App rule — the only place recursive functions are applied.
* CALL-CHAIN QUOTAS, multisets of function symbols, are attached to function applications.

Implementation of Reach in and for Haskell

* **PRELIMINARY TRANSFORMATION** to the Reach core
  — eg. whole-program; dictionaries for type classes.

* Programs may include **HIGHER ORDER** functions,
  but only first-order functions can be Reach sources.

* Requires **ALGEBRAIC DATA TYPES**; primitive numeric
  types are redefined accordingly — eg. integers
  as signed unary.

* **SEARCH CONTROLLED** by the two parameters —
  max. construction depth of source arguments and
  max. recursion depth in evaluation of source applications

Scalability?

* Reach is currently used to examine multi-module programs of a few hundred lines to recursion depth 3-5.

* Experimental pruning rules use function dependency info. to avoid computational branches that cannot entail a target.

Some applications of Reach

* **CRASHING PROGRAMS** – set all error applications as targets; complements other tools such as Catch.

* **BREAKING ASSERTIONS** – redefine assert as follows.

  `assert c x = case c of False → ◁x▷; True → x`

* **REFUTING UNIVERSAL PROPERTIES** – recall earlier examples using $\Longrightarrow$.

* **FALLING INTO BLACK HOLES** – an easy variation of the Var1 rule.

More applications of Reach

* **TEST-SET GENERATION** for program coverage —
  eg. synthesis of iterative but terminating loop
  programs for testing compilers wrt. interpreters.

* **BOUNDED VERIFICATION** — complement of refutation.

  FOLK THEOREM: if a program fails it almost always fails for some simple case

  COROLLARY: programs that do not fail for any simple case hardly ever fail

* **PROGRAM UNDERSTANDING** — eg. find the simplest example of deletion in a balanced tree that involves the most complex re-balancing rotations.