haRVey

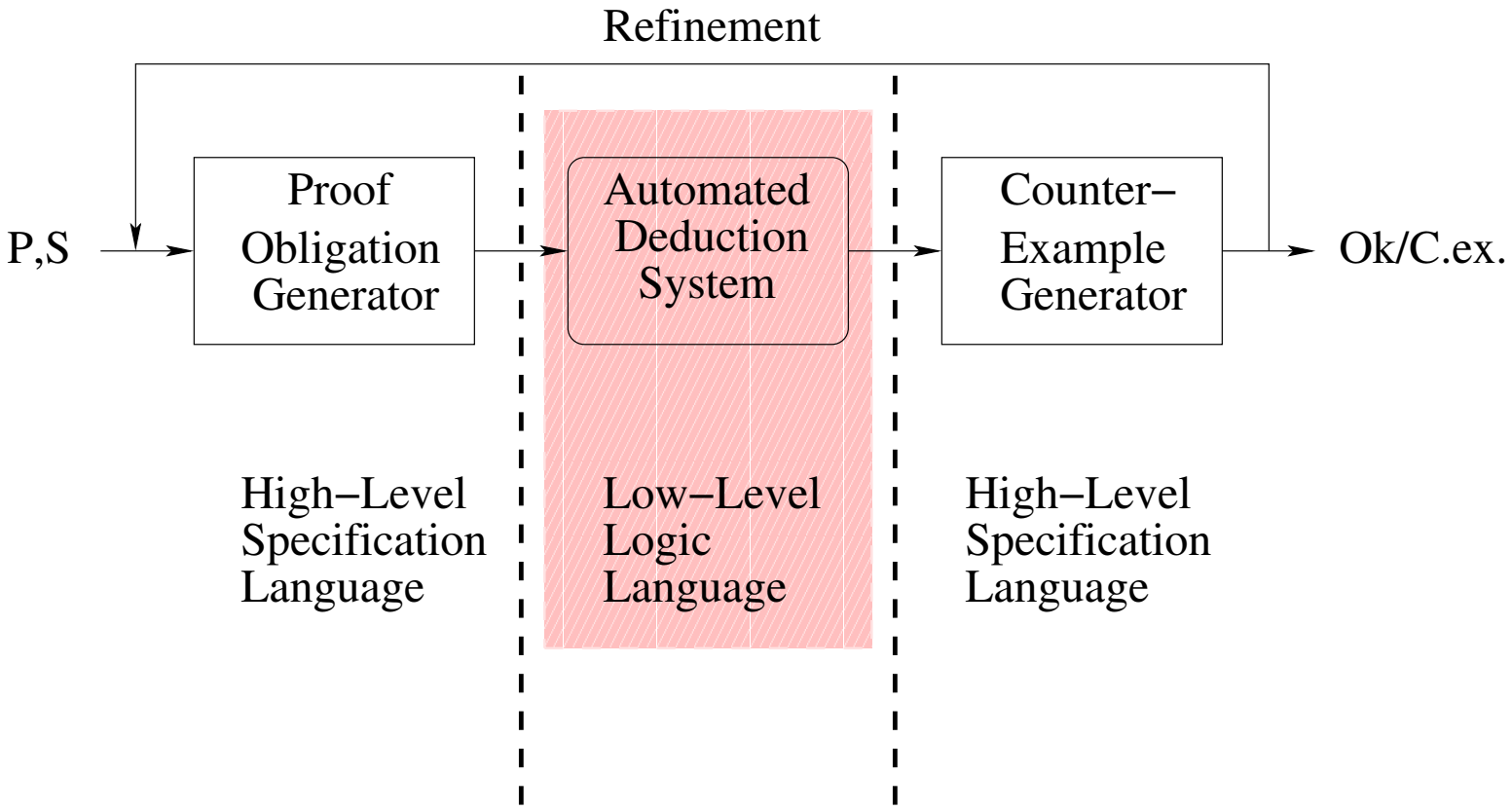
Progress Report

(jww. David, Augusto, Farad, Christophe, Duc)

`http://www.loria.fr/~ranise/haRVey/`

Silvio Ranise
Projet CASSIS (Nancy)

High-level Architecture of Verification Tools



The Problem

- Why Automated Deduction for Verification is difficult*?*
- 3 Dimensional space
  1. ▷ reasoning in the domain of computation
  2. ▷ handling complex control flow
  3. ▷ handling sophisticated syntactic constructs to express properties

Our solution: haRVey

- 3 specialized reasoning modules

1. reasoning in the domain of computation ⇒ Superposition

2. handling complex control flow ⇒ Boolean (SAT/BDD) solver

3. handling sophisticated syntactic constructs ⇒ Pre-processing + Superposition

- An abstract-check-refine cycle (among the 3 dimension) drives the search for a proof or the extraction of information to return a counter-example

haRVey: High-lights

- Built-in capability of reporting the information to build counter-examples or proof objects
  - ▷ key features for debugging tools and/or certifiable results
- Input syntax easy to be mechanically generated
- API exposes important functionalities (eg. for model-checking where intermediate details must be cached for efficiency)
- As a consequence...

  haRVey can be easily hidden in tools for formal analysis

The Application for GECCOO

- Verification and debugging of Java (on-going work)
  - ▷ proof obligations are generated by Krakatoa or JACK
  - ▷ resulting formulae are checked by haRVey:

when a counter-example is found, important information must be passed back to enable high-level error explanation

when the formula is discharged, a proof object should be returned for later certification by an interactive theorem prover (e.g. Coq or Isabelle)

New features to make incorporation of haRVey easier

- haRVey = `rvc` + `rv`

  ▷ `rvc` compiles proof obligations in internal syntax and now also performs some pre-processing

$\Longrightarrow$ handling of quantified formulae (previously done by `rvqe`)

$\Longrightarrow$ reduces the background theory by eliminating "unnecessary" axioms

  ▷ `rv` now features both BDD and SAT solver (Chaff)

$\Longrightarrow$ better scalability (done)

$\Longrightarrow$ combination of arithmetic, equational reasoning, and quantifier instantiation

On-going work

- With F. Metha: on combining Isabelle and haRVey for discharging the proof obligations arising in the verification of Schorr-Waite algorithm for garbage collection

  ▷ very rich background theories (derived from a translation of HOL to FOL)

  ▷ better than *Simplify* even on some basic lemmas because of a better handling of instantiating quantified variables

  ▷ how to translate back proofs generated by haRVey to be certified by Isabelle (types!)?

- With C. Ringeissen and D. Tran: on minimal conflict sets for equality, arithmetic, and their combination (crucial when combining with SAT)

Questions and Remarks (w.r.t. JACK and Krakatoa)

- `LABEL` keyword: how do you (JACK/Krakatoa) plan to use it for debugging?
- Have you got problems with handling arithmetic?
- There are still some problem in the generation of proof obligations for haRVey from Krakatoa: a patch is being prepared by Augusto

How haRVey worked

function check_ground ($\mathcal{T}$: first-order theory, ϕ_g: *ground formula*)
1 $\phi^p \longleftarrow gfol2prop(\phi_g)$
2 **while** $\phi^p \neq false$ **do**
3 **begin**
4 $\beta^p \longleftarrow choose\_ass(\phi^p)$
5 $(\rho, \xi) \longleftarrow check\_ass(\mathcal{T}, prop2gfol(\beta^p))$
6 **if** $\rho = sat$ **then return** sat
7 $\phi^p \longleftarrow \phi^p \wedge \neg gfol2prop(\xi)$
8 **end**
9 **return** $unsat$
end

How haRVey works (1)

function check_fol ($\mathcal{ET} \cup \mathcal{PA}$: *theory*; ϕ_0: *first-order formula*)
1 $\phi_0' \longleftarrow innermost\_univ(top\_ex(\phi_0))$
2 $(\phi, \Delta) \longleftarrow abs\_univ(\phi_0')$
3 **return** check_ground′$(\mathcal{ET} \cup \Delta \cup \mathcal{PA}, \phi)$
end

How haRVey works (2)

function check_comb ($EqCls$: *set of clauses*, $\mathcal{PA}$: *first-order theory*,
β: *conjunction of ground literals*)

1 $(\beta_1, \beta_2, K) \longleftarrow purify(\beta)$
2 **repeat**
3 $\alpha \longleftarrow guess(K)$
4 $(\rho_e, \xi_e) \longleftarrow check\_ass(EqCls, \beta_1 \cup \alpha)$
5 $(\rho_a, \xi_a) \longleftarrow check\_ass(\mathcal{PA}, \beta_2 \cup \alpha \cup prj(\xi_e))$
6 **if** $(\rho_e = sat) \wedge (\rho_a = sat)$ **then return** $(sat, \emptyset)$
7 **until** all possible α's have been considered
8 **return** $(unsat, \xi_e \cup \xi_a)$

end