Theory Reasoning with KeY and Z3 for Deductive Program Verification

Vladimir Klebanov | 4 November 2011
Deductive Verification of
- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic
The Project

Deductive Verification of
- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

KeY Tool
- Symbolic execution

Vladimir Klebanov – Theory Reasoning with KeY and Z3
4 November 2011
The Project

Deductive Verification of
- Java programs
- specified with the
  Java Modeling Language
- in Dynamic Logic

KeY Tool
- Symbolic execution
- Supports 100% Java Card
The Project

Deductive Verification of
- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

KeY Tool
- Symbolic execution
- Supports 100% Java Card
- High degree of automation/usability
  >10,000 loc / expert year
The Project

Deductive Verification of
- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

KeY Tool
- Symbolic execution
- Supports 100% Java Card
- High degree of automation/usability
  >10,000 loc / expert year
The Project

Deductive Verification of
- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

KeY Tool
- Symbolic execution
- Supports 100% Java Card
- High degree of automation/usability
  > 10,000 loc / expert year

Vladimir Klebanov – Theory Reasoning with KeY and Z3
4 November 2011
KeY-based Technology

- KeYmaera
- KeY for Creol, C, ASM, ODL, multi-threaded Java
- KeY symbolic debugger
- KeY test case generator
- KeY Hoare (great for teaching)
Relations Between KeY and Z3

in the following

- What we do with Z3
- What we do not (yet) do with Z3
- Where Z3 cannot help us (or can it?)
Dynamic Logic

... in a nutshell

- every FOL formula is a DL formula
Dynamic Logic

... in a nutshell

- every FOL formula is a DL formula
- $\langle p \rangle \phi$ means: $p$ terminates and then $\phi$ holds
Dynamic Logic

... in a nutshell

- every FOL formula is a DL formula
- $\langle p \rangle \phi$ means: $p$ terminates and then $\phi$ holds
- $[p] \phi$ means: if $p$ terminates, then $\phi$ holds
Dynamic Logic

... in a nutshell

- every FOL formula is a DL formula
- $\langle p \rangle \phi$ means: $p$ terminates and then $\phi$ holds
- $[p] \phi$ means: if $p$ terminates, then $\phi$ holds
- $\psi \rightarrow [p] \phi$ is same as $\{ \psi \} p \{ \phi \}$
Dynamic Logic

... in a nutshell

- every FOL formula is a DL formula
- $\langle p \rangle \phi$ means: $p$ terminates and then $\phi$ holds
- $[p] \phi$ means: if $p$ terminates, then $\phi$ holds
- $\psi \rightarrow [p] \phi$ is same as $\{\psi\} p \{\phi\}$
- DL is closed under nesting, negation, etc.
Verification of Sequential Java Programs
KeY Currently Supports
due to Philipp Rümmer

- normalisation of polynomials
- linear equations: Gaussian elimination and the Euclidean algorithm
- linear inequalities: Fourier-Motzkin variable elimination
- nonlinear (polynomial) equations: Gröbner bases
- nonlinear inequalities: heuristical cross-multiplication and systematic case analysis
- quantifiers: E-matching

...but some goals can only be discharged by calling external SMT solvers.
Problem 2: Inverting an Injection

Invert an injective array $A$ of $N$ elements in the subrange from 0 to $N - 1$, i.e., the output array $B$ must be such that $B[A[i]] = i$ for $0 \leq i < N$. You can assume that $A$ is surjective.

Show that the resulting array is also injective. [...] Demonstrate that $A$ and $B$ are inverses.
Difficult Subproblem

The goal is to prove that for any $N > 0$, the injectivity of $B$

$$\forall x, y. \ 0 \leq x < y < N \rightarrow B[x] \neq B[y]$$  \hspace{1cm} (1)

follows from the inverse relation between the arrays $A$ and $B$ (which per loop invariant holds after the loop)

$$\forall x. \ (0 \leq x < N \rightarrow B[A[x]] = x)$$  \hspace{1cm} (2)

and the surjectivity of $A$ (which is a lemma that the problem description allowed to assume)

$$\forall x. \ ((0 \leq x < N) \rightarrow \exists x'. \ (0 \leq x' < N) \land x = A[x']) \ .$$  \hspace{1cm} (3)
The goal is to prove that for any $N > 0$, the injectivity of $B$ follows from the inverse relation between the arrays $A$ and $B$ (which per loop invariant holds after the loop):

$$\forall x. (0 \leq x < N \rightarrow B[A[x]] = x)$$

(2)

and the surjectivity of $A$ (which is a lemma that the problem description allowed to assume):

$$\forall x. ((0 \leq x < N) \rightarrow \exists x'. (0 \leq x' < N) \land x = A[x']) .$$

(3)

Difficult Subproblem

Difficulties in this problem:

- only interpreted arithmetical symbols in the quantifier guard
- required instantiations are Skolem constants (not present in the source code)
The goal is to prove that for any \( N > 0 \), the injectivity of \( B \)

\[
\forall x, \ y. \ 0 \leq x < y < N \rightarrow B[x] \neq B[y] \tag{1}
\]

follows from the inverse relation between the arrays \( A \) and \( B \) (which per loop invariant holds after the loop)

\[
\forall x. \ (0 \leq x < N \rightarrow B[A[x]] = x) \tag{2}
\]

and the surjectivity of \( A \) (which is a lemma that the problem description allowed to assume)

\[
\forall x. \ \exists x'. \ (0 \leq x < N \land x = A[x']) . \tag{3}
\]

Difficult Subproblem

Difficulties in this problem

- only interpreted arithmetical symbols in the quantifier guard
- required instantiations are Skolem constants (not present in the source code)

Range of solutions

- Manual instantiation
- Dummy function trigger
- Complex reformulations

Vladimir Klebanov – Theory Reasoning with KeY and Z3

4 November 2011
Difficult Subproblem

The goal is to prove that for any $N > 0$, the injectivity of $B$ follows from the inverse relation between the arrays $A$ and $B$ (which per loop invariant holds after the loop):

$$\forall x. (0 \leq x < N \rightarrow B[A[x]] = x)$$  \hfill (2)

and the surjectivity of $A$ (which is a lemma that the problem description allowed to assume):

$$\forall x. (0 \leq x < N \rightarrow \exists x'. (0 \leq x' < N) \land x = A[x'])$$  \hfill (3)

Difficulties in this problem

- only interpreted arithmetical symbols in the quantifier guard
- required instantiations are Skolem constants (not present in the source code)

Needed

- Better control of solver

Range of solutions

- Manual instantiation
- Dummy function trigger
- Complex reformulations
Wishlist
for different parties

- Z3 for Mac (thanks!)
- Java bindings / SMT platform
- (we should) Run Z3 in the background
- (we should) Collect SMT benchmarks
Wishlist

for different parties

- (we should) Try theory of arrays
- Simplification!
Simplification

We need more than just proving. We need goal simplification.

Current procedures often leave the goal hard-to-read for humans.

This is an open research issue.
Other SMT Application Areas in KeY

- Functional verification of sequential Java programs
- Functional verification of multithreaded Java programs
- Test case generation
- Quantitative information flow analysis
Finally

An Advertisement
COST Action IC0701 presents **VerifyThus**—
a Linux distribution with 10 program verification tools.

**Available as:**
- bootable USB stick
- bootable DVD
- virtual machine image

**Included verification tools:**
Boogie, Dafny, ESC/Java2, Jahob, JavaF-AN, jStar, KeY, KIV, Krakatoa, Verifast

http://verifythus.cost-ic0701.org
Questions?