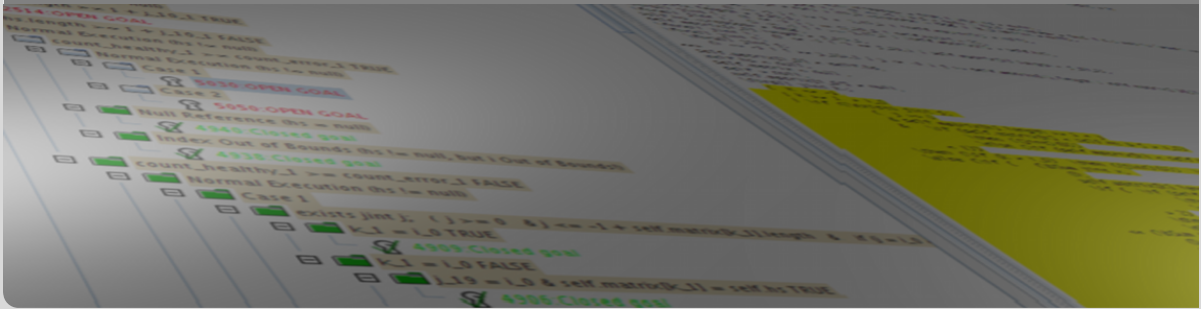
Karlsruhe Institute of Technology

Theory Reasoning with KeY and Z3

for Deductive Program Verification

Vladimir Klebanov | 4 November 2011

KIT – Institut für Theoretische Informatik



The Project

www.key-project.org

Deductive Verification of

- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

The Project

www.key-project.org

Deductive Verification of

- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

KeY Tool

- Symbolic execution

The Project

www.key-project.org

Deductive Verification of

- Java programs
- specified with the Java Modeling Language
- in Dynamic Logic

KeY Tool

- Symbolic execution
- Supports 100% Java Card

The Project

www.key-project.org

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

www.key-project.org

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

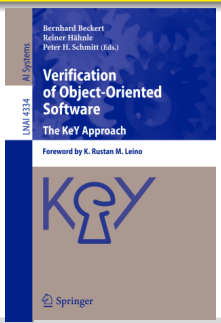
www.key-project.org

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



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 ϕ holds

Dynamic Logic

. . . in a nutshell

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

Dynamic Logic

. . . in a nutshell

- every FOL formula is a DL formula
- $\langle p \rangle \phi$ means: p terminates and then ϕ holds
- $[p]\phi$ means: if p terminates, then ϕ 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 ϕ holds
- $[p]\phi$ means: if p terminates, then ϕ 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 Euclidian algorithm
- linear inequalities: Fourier-Motzkin variable elimination
- nonlinear (polynomial) equations: Gröbner bases
- nonlinear inequalities: heuristical cross-multiplication and systematic case analysis
- quantifiers: E-matching

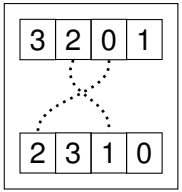
. . . but some goals can only be discharged by calling external SMT solvers.

Verified Software Competition (VSComp) 2010

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 $0 \leqslant i < N$. 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 $\texttt{B}$

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

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

$$\forall x.\ \big(0 \leqslant x < N \rightarrow \texttt{B[A[}x\texttt{]]} = x\big) \tag{2}$$

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

$$\forall x.\ \big((0 \leqslant x < N) \rightarrow \exists x'.\ (0 \leqslant x' < N) \wedge x = \texttt{A[}x'\texttt{]}\big)\ . \tag{3}$$

Difficult Subproblem

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

$$\forall x, y.\ \big(0 \leqslant x < N \wedge 0 \leqslant y < N \wedge x \neq y \rightarrow \texttt{B[}x\texttt{]} \neq \texttt{B[}y\texttt{]}\big) \tag{1}$$

follow … and `B` (which per loop invari…

$$\forall x.\ \big(0 \leqslant x < N \rightarrow \texttt{B[A[}x\texttt{]]} = x\big) \tag{2}$$

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

$$\forall x.\ \big((0 \leqslant x < N) \rightarrow \exists x'.\ (0 \leqslant x' < N) \wedge x = \texttt{A[}x'\texttt{]}\big)\ . \tag{3}$$

Difficulties in this problem

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

Difficult Subproblem

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

$\neq$ `B[`y`]` (1)

follow … and `B` (which per loop invari…

$$\forall x.\ (0 \leqslant x < N \rightarrow \texttt{B[A[}x\texttt{]]} = x) \quad (2)$$

and t… oblem description allowed to assu…

$\wedge\ x = \texttt{A[}x'\texttt{]}\big)$. (3)

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

Difficult Subproblem

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

follow … invari … (1)

$$\forall x.\ (0 \leqslant x < N \rightarrow \texttt{B[A[}x\texttt{]]} = \ldots \quad (2)$$

and t … assu … wed to

$$\ldots \wedge x \quad (3)$$

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

Needed

Better control of solver

And Now

Demo

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

Karlsruhe Institute of Technology

An Advertisement

VerifyThus

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, JavaFAN, jStar, KeY, KIV, Krakatoa, Verifast

http://verifythus.cost-ic0701.org

Thanks

Karlsruhe Institute of Technology

Questions?