

Java and Web Extensions of the Yices Little Engine of Proof

Sokharith Sok

Seidenberg School of Computer Science and Information
Systems

Pace University, New York, USA

December 14, 2006

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

API

Web Based Extension

Conclusion

Stakeholders

- ▶ Master thesis supervisor: Dr. Christelle Scharff
- ▶ SRI researchers: Drs. Natarajan Shankar and Leonardo de Moura
- ▶ The work of this master thesis was supported by an NSF ITR grant #0326540 entitled *Little Engines of Proof*
- ▶ <http://atlantis.seidenberg.pace.edu/wiki/lep>

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

API

Web Based Extension

Conclusion

Automated Deduction and Theorem Proving

What is Theorem Proving?

- ▶ Active area of research since the 1930s
- ▶ Theorem proving = Automated deduction = Automated reasoning
- ▶ It is concerned with the mechanization of the deductive process in its fullest meaning [Loveland 86]
- ▶ The most recent achievements are software

Theorem Prover

What is a Theorem Prover?

- ▶ Theorem prover = Inference rules + Proof Strategies [Bonacina 99]
- ▶ Inference rules: How to deduce new data?
- ▶ Strategies: How to apply inference rules?

Theorem Prover Properties

The 2 most important properties of theorem provers are:

- ▶ **Soundness** Do not prove that $\text{True} = \text{False}$. Proving only true formulas (Required)
- ▶ **Completeness** If a formula is true, it can be proved (Optional)

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

API

Web Based Extension

Conclusion

Little Engines of Proof

A Little Engine of proof is a domain/theory-specific decision procedure [Hao 63, Shankar 02]

The most popular Little Engines are:

- ▶ Barcelogic, Technical University of Catalonia, Spain
- ▶ CVC lite, New York University, USA
- ▶ HarVey, INRIA, France
- ▶ ICS, Stanford Research Institute (SRI), USA
- ▶ Math-SAT, The Trentino Cultural Institute, Italy
- ▶ **Yices, Stanford Research Institute (SRI), USA**

Theories

- ▶ Uninterpreted Symbol $f(x)$
- ▶ Equality $f(x) = y$
- ▶ Integer $x < 3 \&\& x \geq 2$
- ▶ Real $17/2x + 2y = 18$
- ▶ Array (with and without extensionality) $a[i := v1][j] = v2$
- ▶ Recursive datatype (e.g. lists) $car(cons(v1, v3)) = v2$
- ▶ Bitvectors $concat(bv1, bv2) = bv3$

Each theory has its own axioms and associated algorithms.

Satisfiability Modulo Theory (SMT)

- ▶ SMT Library
 - ▶ Benchmark for SMT to facilitate assessment and comparison of existing SMT solvers
 - ▶ Standard format for SMT solvers
- ▶ Competition
 - ▶ SMT-COMP
- ▶ Results
 - ▶ **2005 Winner:** Barcelogic
 - ▶ **2006 Winner:** Yices
 - ▶ The complete result can be found at <http://goedel.cs.uiowa.edu/smtlib/>

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

API

Web Based Extension

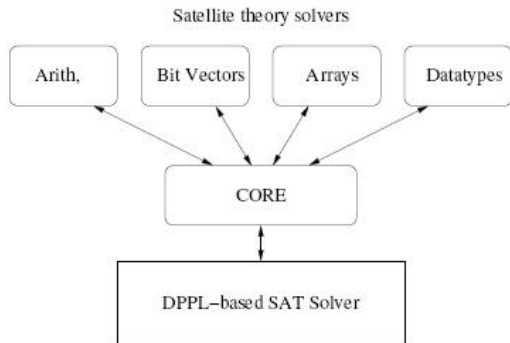
Conclusion

Yices

What is Yices? [<http://yices.csl.sri.com>]

- ▶ Yices is an SMT Solver developed at SRI International
- ▶ It is the backend of SAL, PVS and CALO
- ▶ It support all theories in SMT-LIB and Quantification
- ▶ Yices is not ICS
- ▶ It is available for different platforms:
 - ▶ Linux
 - ▶ Mac OSX
 - ▶ Window (Cygwin, MinGW)
- ▶ Yices can use as interactive or batch mode

Architecture



Input Language

- ▶ define : to declare
- ▶ assert : to add a formula to a logical context
- ▶ check : to test satisfiability
- ▶ set-verbosity
- ▶ set-evidence
- ▶ reset
- ▶ status
- ▶ push
- ▶ pop
- ▶ retract

Output

- ▶ sat
- ▶ unsat
- ▶ model or counter-example

Example

This Yices example checks if a bitvector represents an even number.

Input:

```
(define b::(bitvector 8))
(assert+ (= b (mk-bv 8 10)))
;; last bit is at position 0.
;; the bitvector represents an even number
(assert+ (= (bv-extract 0 0 b) 0b0))
(check)
```

Output:

```
sat
```

Demo

Yices Interactive Demo

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

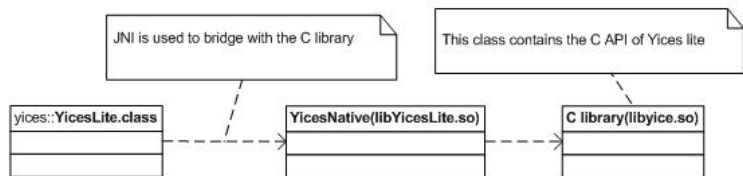
API

Web Based Extension

Conclusion

Yices Java Lite API

- ▶ Built on the top of existing Yices C Lite API
- ▶ Technologies used during development
 - ▶ Eclipse IDE
 - ▶ CDT Eclipse plug-in
 - ▶ Java Native Interface (JNI)
- ▶ Architecture



Yices Java Lite API

- ▶ `void yicesl__del__context(int ctx)`
- ▶ `void yicesl__enable__log__file(Java.lang.String filename)`
- ▶ `void yicesl__enable__type__checker(short flag)`
- ▶ `Java.lang.String Yicesl__get__last__error__message()`
- ▶ `int Yicesl__inconsistent(int ctx)`
- ▶ `int Yicesl__mk__context()`
- ▶ `int Yicesl__read(int ctx, Java.lang.String cmd)`
- ▶ `void Yicesl__set__output__file(Java.lang.String filename)`
- ▶ `void Yicesl__set__verbosity(short l)`
- ▶ `Java.lang.String Yicesl__version()`

Demo

Yices Java Lite API Demo

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

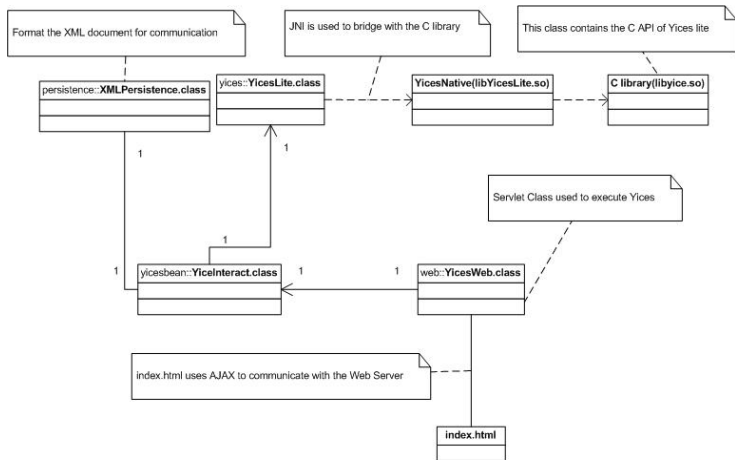
API

Web Based Extension

Conclusion

Yices Web Based Extension

Architecture



Yices Web Based Extension

- ▶ Technologies used during development
 - ▶ Eclipse IDE
 - ▶ WTP Eclipse plug-in
 - ▶ Vim
 - ▶ Tomcat (Servlet)
 - ▶ XML
 - ▶ Ajax

Demo

Yices Web Based Extension Demo

<http://atlantis.seidenberg.pace.edu:8180/yicesweb/index.html>

Difficulties

- ▶ Documentation on JNI
- ▶ Tools for developing application using JNI technology
- ▶ Datatypes in C and Java
- ▶ Testing environment
- ▶ Web Tool Project plug-in under Linux platform
- ▶ Loss of portability because no source code for Yices

Agenda

Automated Deduction & Theorem Proving

Little Engines of Proof

Yices

API

Web Based Extension

Conclusion

Conclusion and Future Work

- ▶ We have developed
 - ▶ Applicative examples
 - ▶ Yices Java Lite API
 - ▶ Yices Web Based Extension
- ▶ Future Work
 - ▶ Feedback and evaluation of SRI
 - ▶ Eclipse plug-in for Yices
 - ▶ Use of Yices in static analysis of Java code based on Hoare logic
- ▶ All this work can be found at
<http://atlantis.seidenberg.pace.edu/wiki/lep>

- ▶ Thank You!!!
- ▶ Questions?