KU KONKUK UNIVERSITY

Yices – A SMT solver

Introduction

Sanghyun Yoon

Dependable Software Laboratory

SAT solver

- (Boolean) SAT(satisfiability) solver
  - A CASE tool which finds set of values satisfy logical formula.
  - Usually embedded other CASE tool.
- Basic SAT solver
  - A&B = true -> SAT solver -> unsat
  - A&B&(A=true)&(B=true) = true-> SAT solver -> sat
- General SAT solver
  - A&B = true -> SAT solver -> A=true, B= true
    - Only one set of values

Satisfiability Modulo Theories(SMT)

- SMT is a problem of determining satisfiability of formulas modulo background theories.

- Examples of background theories:
  - Linear arithmetic: $x + 1 \leq y$
  - Arrays: $a[i := v_i][j] = v_2$
  - Uninterpreted function: $f(f(fx))) = x$
  - Datatypes: $car(cons(v_1, v_3)) = v_2$
  - Bitvectors: $concat(bv_1, bv_2) = bv_3$

User interface

- No GUIs.
- Input file: *.ys
- User friendly mode (shell)
  - ./yices –i
- Usage (cygwin)
  - ./yices -[options] *.ys

Types

- Basic types
  - int
  - bool
  - nat (natural numbers)
  - real
  - bit-vector (user defined size)
  - User defined types

- Types
  - Function, record(stack), tuple, constant

- Form
  - define [name]::[type]
    - e. g.) basic: (define i::int)
    - e. g.) functions: (define f::(-> int int))
  - tuple: (tuple [type_1] ... [type_n])
    - e. g.) (define t1::(tuple int bool int))

Expressions

- Boolean operations
  - Form: (and [expr_1] ... [expr_n]) (or [expr_1] ... [expr_n]) (not [expr]) (=> [expr_1] [expr_2])
  - not expr_i: !expr_i
  - or, and expr_i expr_j: (or, and expr_i expr_j)

- Equality and disequality
  - Form: (= [expr_1] [expr_2]) (/= [expr_1] [expr_2])

- If-then-else
- Let (similar with #define in C)
- Quantifier
- Arithmetic(linear): +, -, *, /, <, >, <=, >=

Assert

- Assert: gets only tirivial inconsistencies
  - Form: assert [expr]

- Extended Assert: gets unsatisfiable cores;

```
(define x::bool)
(define y::bool)
(assert (not x))
(assert (not y))
(assert+ (or x y))
  => 1
(check)
  => unsat
(retract 1)
(check)
  => sat
```

id of assert+

```
(set-evidence! true)
(set-verbosity! 2)
(define x::bool)
(define y::bool)
(define z::bool)
(define w::bool)

(assert+ (/= x y))
  => id: 1
(assert+ (/= y z))
  => id: 2
(assert+ (or w x y))
  => id: 3
(assert+ (/= z x))
  => id: 4
(assert+ (or w z))
  => id: 5
(check)
  => unsat
  => unsat core ids: 1 2 4
```

Assert (cont'd)

```
(define x::bool)
(define y::bool)
(assert (not x))
(assert (not y))
(assert+ (or x y))
  => 1
(check)
  => unsat
(retract 1)
(check)
  => sat
```

```
(define x::bool)
(define y::bool)
(assert (not x))
(assert (not y))
(assert+ (or x y));;id:1
(check)
(retract 1)
(check)
```

```
shyoon@shyoon-PC ~/yices-1.0.29-i686-p
$ ./yices assert.ys
unsat
sat
```

Extracting models

./yices –e exmodel.ys

```
(define x::nat)
(define y::nat)
(assert (< x 3))
(assert (< y 3))
(assert (= x y))
(check)
```

```
shyoon@shyoon-PC ~/yices-
$ ./yices -e exmodel.ys
sat
(= x 0)
(= y 0)
```

- Finding another value set

```
(define x::nat)
(define y::nat)
(assert (< x 3))
(assert (< y 3))
(assert (= x y))
;;second
(assert (and (/= x 0) (/= y 0)))
(check)
```

```
shyoon@shyoon-PC ~/yices-
$ ./yices -e exmodel.ys
sat
(= x 1)
(= y 1)
```

Other value set

Conclusion

- Yices is and efficient and flexible SMT solver.
- Yices is freely available for end-users.
  - http://yices.csl.sri.com/
- Supported platform
  - Linux
  - Windows: cygwin & MinGW
  - Mac OSX
- Supported as a standalone tool and a library.
  - But, only GMT library(Linux).