CSCI 740 - Programming Language Theory

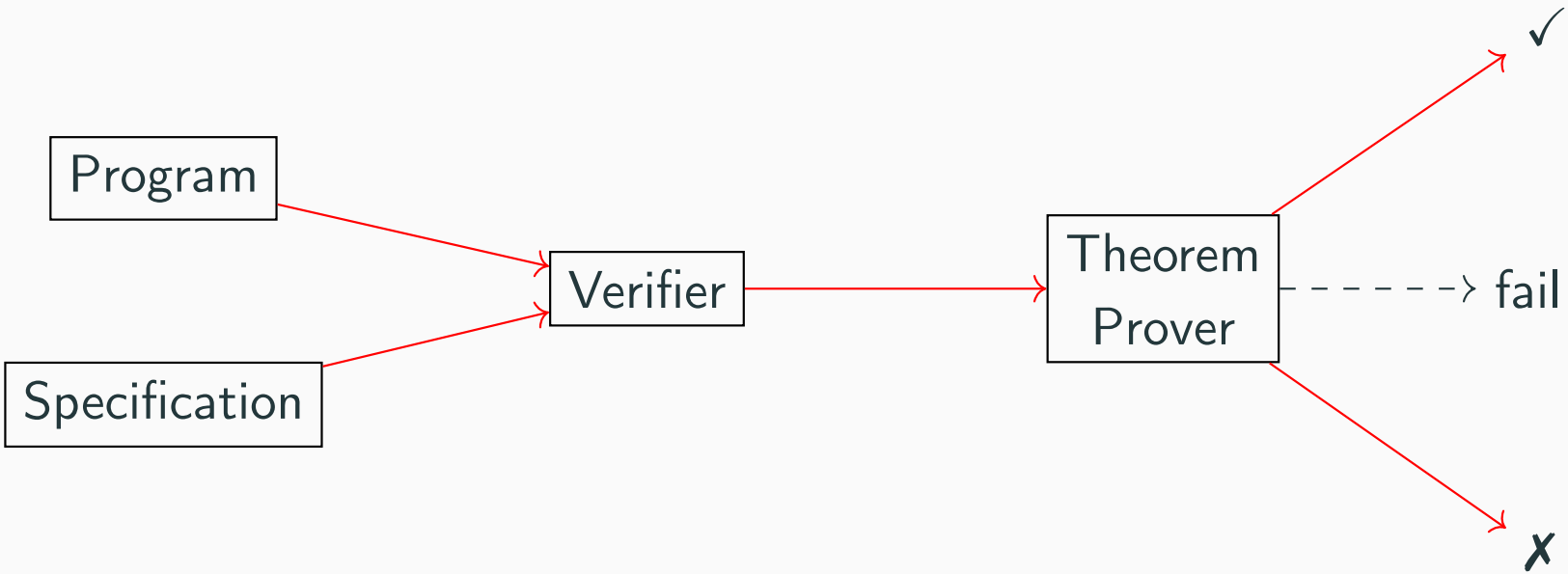
Lecture 22

Verification Condition Generation (II)

Instructor: Hossein Hojjat

October 25, 2017

Recap: Automated Verification



Weakest Precondition Rules: Summary

c	$\mathsf{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$\mathsf{assume}(b)$	$b \to Q$
$\mathsf{assert}(b)$	$\mathsf{wp}(b \wedge Q)$
$\mathsf{havoc}(x)$	$\forall y. Q[x \mapsto y]$
$c_1; c_2$	$\mathsf{wp}(c_1, \mathsf{wp}(c_2, Q))$
`if` b `then` c_1 `else` c_2	$b \to \mathsf{wp}(c_1, Q) \wedge \neg b \to \mathsf{wp}(c_2, Q)$
`while` b `do` c	$I \wedge \forall \vec{y}. \Big((I \wedge b \to \mathsf{wp}(c, I)) \wedge (I \wedge \neg b \to Q) \Big) [\vec{x} \mapsto \vec{y}]$ ($\vec{x}$ are variables modified in c and I is the loop invariant)

Exercise

Is this program correct?

```
i = 5;
while (i > 0)
  invariant{i ≥ 0}
{
  i = i − 1;
}
{ i == 0 }
```

c	$\mathsf{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$c_1; c_2$	$\mathsf{wp}(c_1, \mathsf{wp}(c_2, Q))$
$\texttt{while}\ b\ \texttt{do}\ c$	$I \wedge \forall \vec{y}.$ $\Big((I \wedge b \to \mathsf{wp}(c, I)) \wedge$ $(I \wedge \neg b \to Q)\Big)[\vec{x} \mapsto \vec{y}]$

$$\mathsf{wp}\big(i := 5; \texttt{while}\ (i > 0)\ i := i - 1, i = 0\big) =$$

Exercise

Is this program correct?

```
i = 5;
while (i > 0)
  invariant{i ≥ 0}
{
  i = i − 1;
}
{ i == 0 }
```

c	$\mathsf{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$c_1; c_2$	$\mathsf{wp}(c_1, \mathsf{wp}(c_2, Q))$
$\texttt{while}\ b\ \texttt{do}\ c$	$I \wedge \forall \vec{y}.$ $\Big((I \wedge b \to \mathsf{wp}(c, I)) \wedge$ $(I \wedge \neg b \to Q)\Big)[\vec{x} \mapsto \vec{y}]$

$$\mathsf{wp}\big(i := 5; \texttt{while}\ (i > 0)\ i := i - 1, i = 0\big) =$$
$$\mathsf{wp}\big(i := 5; \mathsf{wp}(\texttt{while}\ (i > 0)\ i := i - 1, i = 0)\big) =$$
$$\mathsf{wp}\big(i := 5; i \geq 0 \wedge \forall i. i \geq 0 \to ((i > 0 \to i - 1 \geq 0) \wedge (\neg(i > 0) \to i = 0))\big) =$$
$$5 \geq 0 \wedge \forall i. i \geq 0 \to ((i > 0 \to i - 1 \geq 0) \wedge (\neg(i > 0) \to i = 0)) \quad \checkmark$$

Exercise

Is this program correct?

```
y = 5;
if (x > 0) {
  assert x + y > 5;
} else {
  assume x == 0;
  y = y + x;
  assert x + y == 5;
}
```

c	$\mathsf{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$\mathsf{assume}(b)$	$b \to Q$
$\mathsf{assert}(b)$	$\mathsf{wp}(b \wedge Q)$
$c_1; c_2$	$\mathsf{wp}(c_1, \mathsf{wp}(c_2, Q))$
$\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2$	$b \to \mathsf{wp}(c_1, Q) \wedge$ $\neg b \to \mathsf{wp}(c_2, Q)$

$\mathsf{wp}(y := 5; \texttt{if}\ \cdots\ \texttt{then}\ \cdots\ \texttt{else}\ \cdots, \mathsf{true}) =$

Exercise

Is this program correct?

```
y = 5;
if (x > 0) {
  assert x + y > 5;
} else {
  assume x == 0;
  y = y + x;
  assert x + y == 5;
}
```

c	$\mathsf{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$\mathsf{assume}(b)$	$b \to Q$
$\mathsf{assert}(b)$	$\mathsf{wp}(b \wedge Q)$
$c_1; c_2$	$\mathsf{wp}(c_1, \mathsf{wp}(c_2, Q))$
$\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2$	$b \to \mathsf{wp}(c_1, Q) \wedge$ $\neg b \to \mathsf{wp}(c_2, Q)$

$$\mathsf{wp}(y := 5; \texttt{if } \cdots \texttt{ then } \cdots \texttt{ else } \cdots, \mathsf{true}) =$$
$$\mathsf{wp}(y := 5; \mathsf{wp}(\texttt{if } \cdots \texttt{ then } \cdots \texttt{ else } \cdots, \mathsf{true})) =$$
$$\mathsf{wp}(y := 5; (x > 0) \to \mathsf{wp}(\texttt{assert}(x + y > 5), \mathsf{true}) \wedge$$
$$(x \leq 0) \to \mathsf{wp}(\texttt{assume}(x = 0); y := y + x; \texttt{assert}(x + y = 5), \mathsf{true})) =$$
$$\mathsf{wp}(y := 5; (x \leq 0 \vee x + y > 5) \ \wedge \ (x > 0 \vee (x = 0 \to x + y + x = 5))) =$$
$$(x \leq 0 \vee x + 5 > 5) \ \wedge \ (x > 0 \vee (x = 0 \to x + 5 + x = 5))$$

Exercise

Is this program correct?

```
y = 5;
if (x > 0) {
  assert x + y > 5;
} else {
  assume x == 0;
  y = y + x;
  assert x + y == 5;
}
```

c	$\mathsf{wp}(c, Q)$
$x := e$	$Q[x \mapsto e]$
$\mathsf{assume}(b)$	$b \to Q$
$\mathsf{assert}(b)$	$\mathsf{wp}(b \wedge Q)$
$c_1; c_2$	$\mathsf{wp}(c_1, \mathsf{wp}(c_2, Q))$
$\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2$	$b \to \mathsf{wp}(c_1, Q) \wedge$ $\neg b \to \mathsf{wp}(c_2, Q)$

$\mathsf{wp}(y := 5; \texttt{if} \cdots \texttt{then} \cdots \texttt{else} \cdots, \mathsf{true}) =$

$\mathsf{wp}(y := 5; \mathsf{wp}(\texttt{if} \cdots \texttt{then} \cdots \texttt{else} \cdots, \mathsf{true})) =$

$\mathsf{wp}(y := 5; (x \leq 0) \to \mathsf{wp}(\texttt{assume}(x = 0); y := y + x; \texttt{assert}(x + y = 5), \mathsf{true})) =$

$\mathsf{wp}(y := 5; (x \leq 0 \vee x + y > 5) \ \wedge \ (x > 0 \vee (x = 0 \to x + y + x = 5))) =$

$(x \leq 0 \vee x + 5 > 5) \ \wedge \ (x > 0 \vee (x = 0 \to x + 5 + x = 5))$

Now what? How do we decide if this formula is **valid**?

Validity

- **Validity:** formula is valid if it is true for all values of its variables
  - Boolean formula: if all results in its truth table are true
- **Satisfiability:** existence of a combination of values to make the expression true
  - Boolean formula: if it has at least one true result in its truth table
- Reduction of validity to satisfiability:
  Formula ϕ is valid if and only if $\neg\phi$ is unsatisfiable
- General-purpose theorem provers are typically not able to solve such satisfiability problems automatically
- We know how to solve satisfiability for Boolean formula:
  SAT Solvers!
- What about satisfiability with respect to some background theory (e.g. Integer arithmetic)?

Satisfiability Modulo Theories (SMT)

- Is formula ϕ satisfiable modulo theory T?
- SMT solvers have specialized algorithms for T

Satisfiability Modulo Theories (SMT)

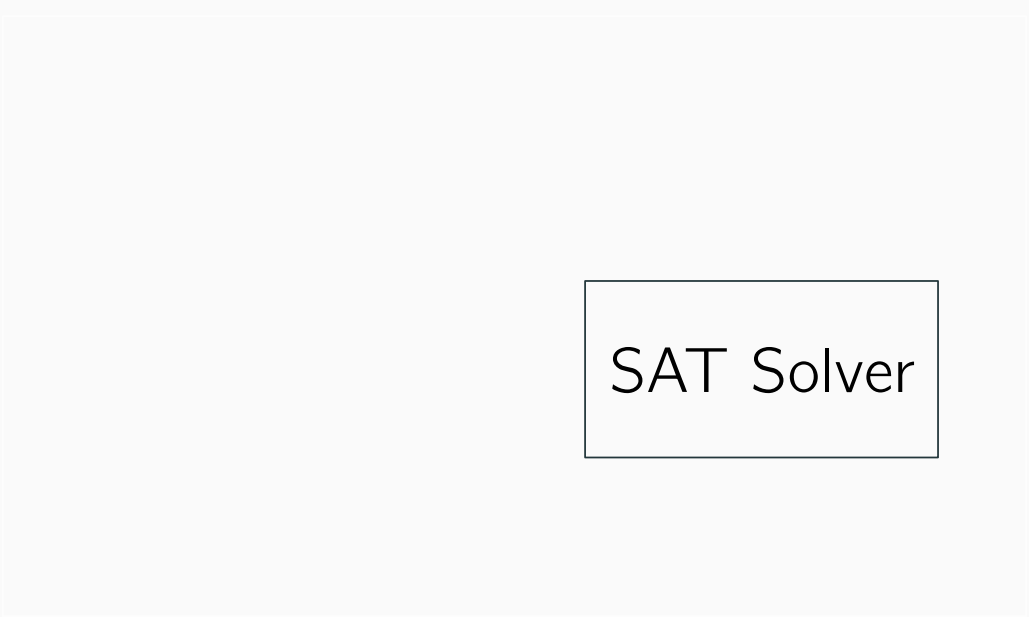
$$x + 2 = y \rightarrow f\,(\,\mathit{select}\,(\,\mathit{store}\,(a, x, 3), y - 2)) = f\,(\,y - x + 1)$$

Array Theory

Arithmetic

Uninterpreted Functions

From SAT to SMT

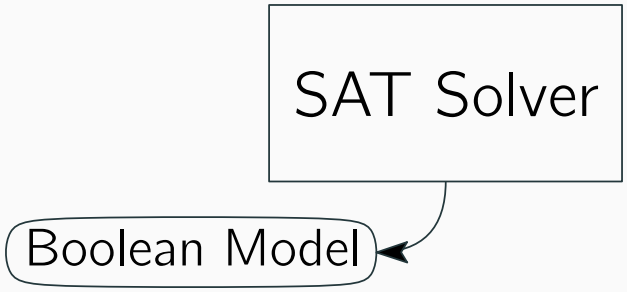


- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$

From SAT to SMT

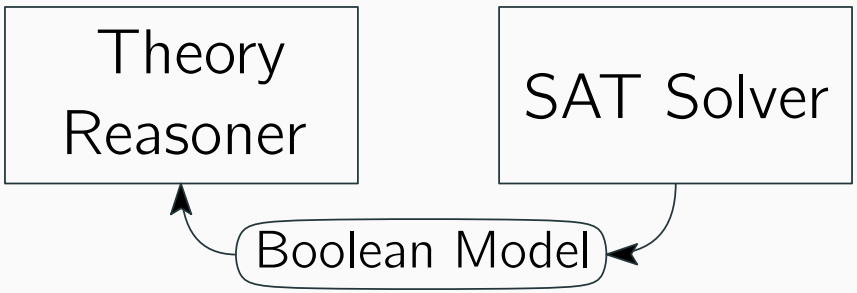
- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$
- To SAT solver:
  $p_{a \leq b} \wedge p_{b \leq a+x} \wedge p_{x=0} \wedge \big(\neg p_{f(a)=f(b)} \vee (p_{q(a)} \wedge \neg p_{q(b+x)})\big)$

From SAT to SMT



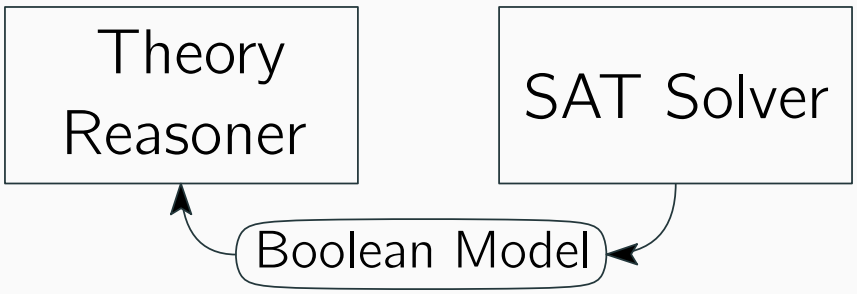
- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$
- To SAT solver:
  $p_{a \leq b} \wedge p_{b \leq a+x} \wedge p_{x=0} \wedge \big(\neg p_{f(a)=f(b)} \vee (p_{q(a)} \wedge \neg p_{q(b+x)})\big)$
- Boolean model: $p_{a \leq b}, p_{b \leq a+x}, p_{x=0}, \neg p_{f(a)=f(b)}$

From SAT to SMT



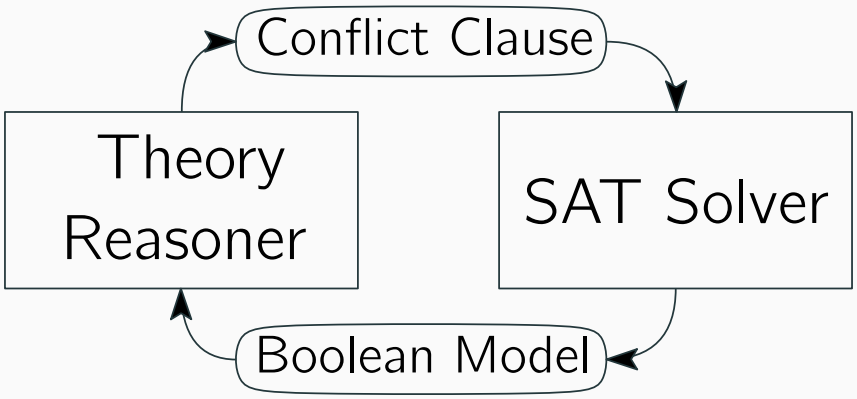
- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$
- To SAT solver:
  $p_{a \leq b} \wedge p_{b \leq a+x} \wedge p_{x=0} \wedge \big(\neg p_{f(a)=f(b)} \vee (p_{q(a)} \wedge \neg p_{q(b+x)})\big)$
- Boolean model: $p_{a \leq b}, p_{b \leq a+x}, p_{x=0}, \neg p_{f(a)=f(b)}$
- Theory reasoner: $a \leq b, b \leq a + x, x = 0, f(a) \neq f(b)$

From SAT to SMT



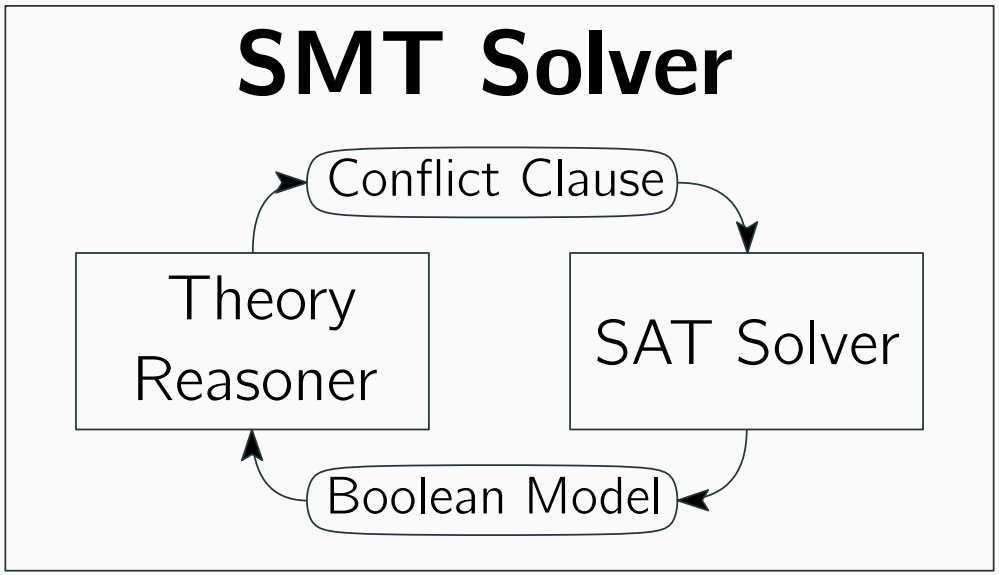
- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$
- To SAT solver:
  $p_{a \leq b} \wedge p_{b \leq a+x} \wedge p_{x=0} \wedge \big(\neg p_{f(a)=f(b)} \vee (p_{q(a)} \wedge \neg p_{q(b+x)})\big)$
- Boolean model: $p_{a \leq b}, p_{b \leq a+x}, p_{x=0}, \neg p_{f(a)=f(b)}$
- Theory reasoner: $a \leq b, b \leq a + x, x = 0, f(a) \neq f(b)$
  unsatisfiable

From SAT to SMT



- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$
- To SAT solver:
  $p_{a \leq b} \wedge p_{b \leq a+x} \wedge p_{x=0} \wedge \big(\neg p_{f(a)=f(b)} \vee (p_{q(a)} \wedge \neg p_{q(b+x)})\big)$
- Boolean model: $p_{a \leq b}, p_{b \leq a+x}, p_{x=0}, \neg p_{f(a)=f(b)}$
- Theory reasoner: $a \leq b, b \leq a + x, x = 0, f(a) \neq f(b)$
  unsatisfiable
- New clause: $\neg p_{a \leq b} \vee \neg p_{b \leq a+x} \vee \neg p_{x=0} \vee p_{f(a)=f(b)}$

From SAT to SMT



- Input:
  $a \leq b \wedge b \leq a + x \wedge x = 0 \wedge \big(f(a) \neq f(b) \vee (q(a) \wedge \neg q(b + x))\big)$
- To SAT solver:
  $p_{a \leq b} \wedge p_{b \leq a+x} \wedge p_{x=0} \wedge \big(\neg p_{f(a)=f(b)} \vee (p_{q(a)} \wedge \neg p_{q(b+x)})\big)$
- Boolean model: $p_{a \leq b}, p_{b \leq a+x}, p_{x=0}, \neg p_{f(a)=f(b)}$
- Theory reasoner: $a \leq b, b \leq a + x, x = 0, f(a) \neq f(b)$
  unsatisfiable
- New clause: $\neg p_{a \leq b} \vee \neg p_{b \leq a+x} \vee \neg p_{x=0} \vee p_{f(a)=f(b)}$

SMT-LIB

- SMT-LIB is a language for specifying input to SMT solvers
- Basic instructions:

`(declare-fun x () Int)`	declare an integer constant `x`
`(assert (> x 0))`	add `x > 0` to known facts
`(check-sat)`	check if there exist an assignment that makes all known facts true
`(get-model)`	print this assignment

SMT for verification

- We need to decide if wp(prog,true) is valid
  - For all values of program variables on entry
- How do we encode this as an SMT problem?
- Ask if ¬wp(prog,true) is satisfiable
- if the answer is UNSAT, the problem is correct
- if the answer is SAT, the model gives the input values that violate correctness

Example

Is this formula valid?

$$(x \leq 0 \vee x + 5 > 5) \ \wedge \ (x \leq 0 \wedge (x = 0 \rightarrow x + x + 5 = 5))$$

```
(declare-fun x () Int)
(assert (not (and  (> x 0) (> (+ x 5) 5))))
(assert (not (and (<= x 0) (or ( not (= x 0))
  (= (+ x (+ x 5)) 5)))))
(check-sat)
```