# IS893: Advanced Software Security

## 8. Logic and Constraint Languages

Kihong Heo

KAIST

# Constraint Solving in Program Analysis

- Various uses of logical constraints in program analysis
  - Verification condition: software verification
  - Path reachability: fuzzing, symbolic execution
  - Exploitable input: automated exploit generation
  - Etc

# Example

```
int arr[size];
for (int i = 0; i < size - 1; i++) {
  arr[i] = 0;
}
assert(i < size);
arr[i] = 1;
```

```
if (a < b) {
  flag = 0;
} else {
  flag = 1;
}

if (a < b) {
  // Is this branch reachable if flag = 1?
}
```

```
array = random_array();
quicksort(array);

for(i = 0; i < size - 1; i++)
  assert(arr[i] <= array[i + 1];
```

```
// Does any exploitable input exist?
str = input();
int arr[10];
strcpy(arr, str);
```

# Common Constraint Languages

- Commonly used languages in the community
  - SAT
  - SMT
  - Horn clause
  - Constrained Horn clause (CHC)
- Different expressiveness and efficiency

All languages supported by Z3

# Propositional Logic: Syntax

- Atom: truth symbols ("true" and "false") and propositional variables

- Literal: atom $\alpha$ or its negation $\neg\alpha$

- Formula: literal or application of a logical connective to formulae $F, F_1, F_2$

$$
\begin{aligned}
&\neg F \\
&F_1 \land F_2 \\
&F_1 \lor F_2 \\
&F_1 \implies F_2 \\
&F_1 \iff F_2
\end{aligned}
$$

# Propositional Logic: Semantics

- Interpretation: assignment to every propositional variable exactly one truth value
$$I : \{p \mapsto \mathsf{true}, q \mapsto \mathsf{false}, \cdots\}$$
- Formula $F$ + Interpretation $I$ = Truth value
- We write $I \models F$ if $F$ evaluates to true under $I$
- We write $I \not\models F$ if $F$ evaluates to false under $I$
- Example: $\{p \mapsto \mathsf{true}, q \mapsto \mathsf{false}\} \models (p \land q) \to (p \lor \neg q)$

# The SAT Problem

- Boolean satisfiability problem
- "Given a **propositional** formula, decide whether it is satisfiable"
  - If satisfiable, there exists an satisfying assignment to the variables
  - NP-complete
- Example:

$$(\neg p \lor q) \land (\neg q \lor r) \land (p \lor \neg r \lor q)$$

Satisfiable when $p$ = false, $q$ = true, $r$ = true

$$p \land \neg p$$

Unsatisfiable

# First-order Logic

- Logical symbols
  - Quantifiers (∀ and ∃), logical connectives, variables, equality symbol
- Non-logical symbols
  - Predicates (relation): $greaterThan$, $isFemale$, etc
  - Functions: $fatherOf$, $plus$, etc
  - Constants: “Kihong”, 1, etc (special case of a function with arity 0)
- Example: $\forall x \in \mathbb{Z} \,.\, p(x) \wedge f(x) > 0 \implies q(x)$

# Terms

- Variables: any variable is a term
- Functions: any expression $f(t_1, \ldots, t_n)$ is a term if $f$ is a function symbol and $t_i$ is a term
- Example:
  - $x,\ 1, f(f(x), f(f(f(x))))$

# First-order Formula

- Predicates: $P(t_1, \ldots, t_n)$ is a formula if $P$ is a predicate symbol and $t_i$ is a term
- Equality: $t_1 = t_2$ is a formula if $t_1$ and $t_2$ are terms
- Negation: $\neg\varphi$ is a formula if $\varphi$ is a formula
- Connectives: $\varphi \oplus \psi$ is a formula if $\varphi$ and $\psi$ are formulas, and $\oplus$ is a connectives
- Quantifiers: $\forall x . \varphi$ and $\exists x . \varphi$ are formulas if $\varphi$ is a formula and $x$ is a variable
- Example:
  - $P(f(0),1,2)$ is a formula but, $P(P(1,2),2)$ is NOT a formula

# The SMT Problem

- Satisfiability Modulo Theories
- “Given a **first-order** formula, decide whether it is satisfiable”
- Higher level reasoning than the Boolean level by theories
- Complexity: depending on the underlying theories

$$x + 2 = y \implies f(read(write(a, x, 3), y - 2)) = f(y - x + 1)$$

Arithmetic

Array

Uninterpreted Functions

# Theories

- Signature + Axiom
  - Signature: a set of non-logical symbols (predicates, constants, functions)
  - Axiom: a set of true statements
  - E.g., linear arithmetic theory

**Signature:** $(0, 1, +, -, \leq)$

**Axioms:**

$$\forall a, b.\ a + b = b + a$$

$$\forall a, b, c.\ (a + b) + c = a + (b + c)$$

# Common Theories

- Equality with uninterpreted function (EUF): $x = y \implies f(x) = f(y)$
- Arrays: two axioms with two interpreted function $read$ and $write$
  $read(write(A, i, d), i) = d$ and $read(write(A, i, d), j) = read(A, j)$ for $i \neq j$
- Bit-vectors (integers or floating-point)
- Linear arithmetic
- Inductive datatypes
- Etc

# Horn Clause

- Clause: a disjunction of literals (e.g., $p \lor \neg q \lor \neg r$)
- Horn clause: a clause with at most one positive
    - E.g., $\neg p \lor \neg q \lor r$ which is equivalent to $p \land q \implies r$
- Horn clause logic: basis of logic programming languages such as Prolog and Datalog

# Horn Clause Satisfiability

- Propositional Horn clause (HORNSAT): linear time
- First-order Horn clause (e.g., Prolog): undecidable
- First-order Horn clause w/o function symbols (e.g., Datalog): EXPTIME

# Constrained Horn Clause (CHC)

- A first-order logic formula,

$$\underbrace{\varphi}_{\textbf{Constraint}} \land \underbrace{p_1(X_1) \land \cdots \land p_n(X_n) \implies h(X)}_{\textbf{Datalog rule}}$$

- $\varphi$: a constraint in a background theory (e.g., linear)

# Conclusion

- Constraint solving: check satisfiability of logical constraints
  - Constraints on program properties: PL, SE, Security, etc
  - Constraints on general facts: symbolic AI, knowledge discovery
- Various constraint languages with different expressive power