CS 292C Computer-Aided Reasoning for Software

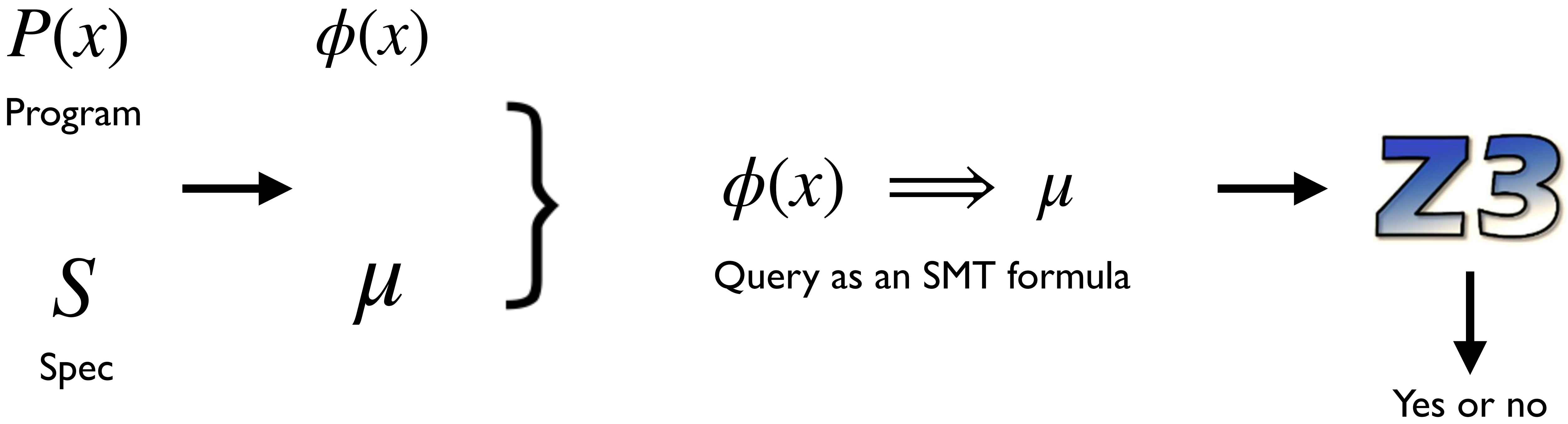
Lecture 4: SAT Solving Basics

Yu Feng
Fall 2019

Summary of previous lecture

- 1st paper review is due today
- The spectrum of program synthesis
- Solver-aided programming II (synthesis)
- Program synthesis via conflict-driven learning

Workhorse of formal methods



$P(x)$

Program

S

Spec

$\phi(x)$

μ

$\phi(x) \implies \mu$

Query as an SMT formula

Z3

Yes or no

Outline of this lecture

- Review of propositional logic
- Normal forms
- A basic SAT solver

Syntax of propositional logic

$$(\neg p \land \top) \lor (q \rightarrow \bot)$$

Atom Truth symbols: ⊤ (“true”), ⊥ (“false”)
propositional variables: *p,q,r,...*

Literal an atom α or its negation ¬α

Formula an atom or the application of a **logical connective** to formulas F_1, F_2:

$\neg F_1$	“not”	**(negation)**
$F_1 \land F_2$	“and”	**(conjunction)**
$F_1 \lor F_2$	“or”	**(disjunction)**
$F_1 \rightarrow F_2$	“implies”	**(implication)**
$F_1 \leftrightarrow F_2$	“if and only if”	**(iff)**

Semantics of propositional logic

An **interpretation** *I* for a propositional formula *F* maps every variable in *F* to a truth value:

$$I : \{ p \mapsto \text{true}, q \mapsto \text{false}, \ldots\}$$

I is a **satisfying interpretation** of *F*, written as $I \vDash F$, if *F* evaluates to true under *I*.

I is a **falsifying interpretation** of *F*, written as $I \nvDash F$, if *F* evaluates to false under *I*.

A satisfying interpretation is also a **model**

Semantics of propositional logic

Base cases:

- $I \models \top$
- $I \not\models \bot$
- $I \models p$ iff $I[p]$ = true
- $I \not\models p$ iff $I[p]$ = false

Inductive cases:

- $I \models \neg F$ iff $I \not\models F$
- $I \models F_1 \wedge F_2$ iff $I \models F_1$ and $I \models F_2$
- $I \models F_1 \vee F_2$ iff $I \models F_1$ or $I \models F_2$
- $I \models F_1 \rightarrow F_2$ iff$I \not\models F_1$ or $I \models F_2$
- $I \models F_1 \leftrightarrow F_2$ iff $I \models F_1$ and $I \models F_2$,or
  $I \not\models F_1$ and $I \not\models F_2$

Semantics of propositional logic

F: $(p \wedge q) \rightarrow (p \vee \neg q)$

I: $\{p \mapsto \text{true}, q \mapsto \text{false}\}$

$I \vDash F$

Satisfiability v.s. validity

F is **satisfiable** iff $I \models F$ for some I.

F is **valid** iff $I \models F$ for all I.

Duality of satisfiability and validity:

F is valid iff $\neg F$ is unsatisfiable.

One algorithm for checking both satisfiability and validity.

Proof by induction

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F1 \land F2}{I \models F1, I \models F2} \qquad \frac{I \not\models F1 \land F2}{I \not\models F1 \mid I \not\models F2}$$

$$\frac{I \models F1 \lor F2}{I \models F1 \mid I \models F2} \qquad \frac{I \not\models F1 \lor F2}{I \not\models F1, I \not\models F2}$$

$$\frac{I \models F1 \to F2}{I \not\models F1 \mid I \models F2} \qquad \frac{I \not\models F1 \to F2}{I \models F1, I \not\models F2}$$

$$\frac{I \models F1 \leftrightarrow F2}{I \models F1 \land F2 \mid I \not\models F1 \lor F2} \qquad \frac{I \not\models F1 \leftrightarrow F2}{I \models F1 \land \neg F2 \mid I \models \neg F1 \land F2}$$

Prove $p \land \neg q$ *is valid*

1. $I \not\models p \land \neg q$ (assumed)
    1. $I \not\models p$ (1, ∧)
    2. $I \not\models \neg q$ (1,∧)
        1. $I \models q$ (1b,¬)

I = {p ↦ false,q ↦ true} is a falsifying interpretation.

Proof by induction

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F}$$

$$\frac{I \models F1 \wedge F2}{I \models F1, I \models F2} \qquad \frac{I \not\models F1 \wedge F2}{I \not\models F1 \mid I \not\models F2}$$

$$\frac{I \models F1 \vee F2}{I \models F1 \mid I \models F2} \qquad \frac{I \not\models F1 \vee F2}{I \not\models F1, I \not\models F2}$$

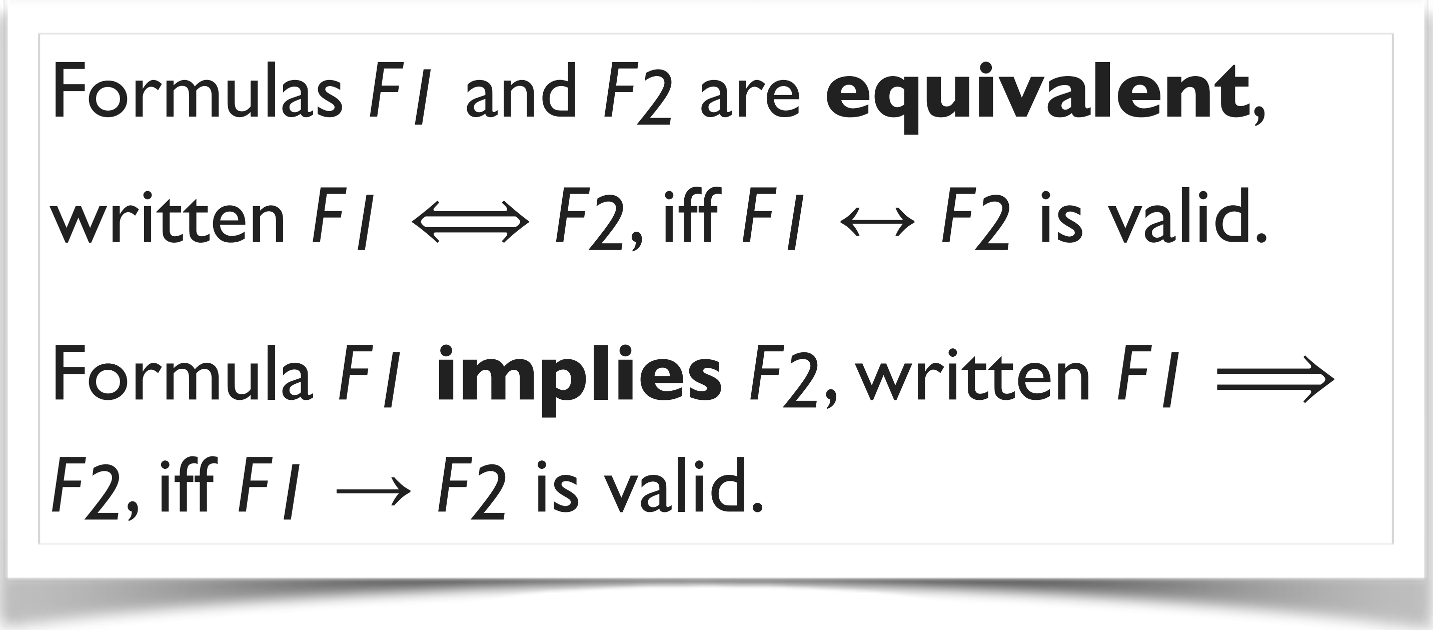
$$\frac{I \models F1 \rightarrow F2}{I \not\models F1 \mid I \models F2} \qquad \frac{I \not\models F1 \rightarrow F2}{I \models F1, I \not\models F2}$$

$$\frac{I \models F1 \leftrightarrow F2}{I \models F1 \wedge F2 \mid I \not\models F1 \vee F2} \qquad \frac{I \not\models F1 \leftrightarrow F2}{I \models F1 \wedge \neg F2 \mid I \models \neg F1 \wedge F2}$$

Prove $(p \wedge (p \rightarrow q)) \rightarrow q$ *is valid*

1. $I \not\models (p \wedge (p \rightarrow q)) \rightarrow q$
2. $I \not\models q$ (1,→)
3. $I \models (p \wedge (p \rightarrow q))$ (1,→)
4. $I \models p$ (3,∧)
5. $I \models p \rightarrow q$ (3,∧)
   a. $I \not\models p$ (5,→)
   b. $I \models q$ (5,→)

Semantic judgements



Formulas $F1$ and $F2$ are **equivalent**, written $F1 \iff F2$, iff $F1 \leftrightarrow F2$ is valid.

Formula $F1$ **implies** $F2$, written $F1 \implies F2$, iff $F1 \rightarrow F2$ is valid.

$F1 \iff F2$ and $F1 \implies F2$ are not propositional formulas (not part of syntax).They are properties of formulas.

SAT solving with normal forms

A **normal form** for a logic is a syntactic restriction such that every formula in the logic has an equivalent formula in the normal form.

Three important normal forms:

- Negation Normal Form (NNF)
- Disjunctive Normal Form (DNF)
- Conjunctive Normal Form (CNF)

Negation normal form

Atom := Variable | ⊤ | ⊥

Literal := Atom | ¬Atom

Formula := Literal | Formula op Formula

op := ∧ | ∨

- The only allowed connectives are ∧, ∨, and ¬.
- ¬ can appear only in literals

Conversion to NNF performed using DeMorgan's Laws:

$\neg(F \wedge G) \iff \neg F \vee \neg G \qquad \neg(F \vee G) \iff \neg F \wedge \neg G$

Disjunctive normal form

Atom := Variable | ⊤ | ⊥
Literal := Atom | ¬Atom
Formula := Clause ∨ Formula
Clause := Literal | Literal ∧ Clause

- Disjunction of conjunction of literals
- Trivial to decide if a DNF formula is SAT, why?
- Why not modern SAT solvers use DNF?

To obtain DNF, convert to NNF and distribute ∧ over ∨:

$$(F\wedge(G\vee H))\Longleftrightarrow (F\wedge G)\vee(F\wedge H)$$

$$((G\vee H)\wedge F)\Longleftrightarrow (G\wedge F)\vee(H\wedge F)$$

Conjunctive normal form

Atom := Variable | $\top$ | $\bot$

Literal := Atom | $\neg$Atom

Formula := Clause $\vee$ Formula

Clause := Literal | Literal $\wedge$ Clause

- Conjunction of disjunction of literals
- Hard to decide if a CNF formula is SAT
- Default language in modern SAT solvers

To obtain CNF, convert to NNF and distribute $\vee$ over $\wedge$:

$$(F \vee (G \wedge H)) \Longleftrightarrow (F \vee G) \wedge (F \vee H)$$

$$((G \wedge H) \vee F) \Longleftrightarrow (G \vee F) \wedge (H \vee F)$$

Equisatisfiability and Tseitin's transformation

Formulas F and G are **equisatisfiable** if they are both satisfiable or they are both unsatisfiable.

Tseitin's transformation converts a propositional formula F into an equisatisfiable CNF formula that is **linear** in the size of F.

$x \rightarrow (y \wedge z)$

$a1$

$a1 \leftrightarrow (x \rightarrow a2)$

$a2 \leftrightarrow (y \wedge z)$

Key idea: introduce auxiliary variables to represent the output of subformulas, and constrain those variables using CNF clauses.

Equisatisfiability and Tseitin’s transformation

Formulas F and G are **equisatisfiable** if they are both satisfiable or they are both unsatisfiable.

Tseitin’s transformation converts a propositional formula F into an equisatisfiable CNF formula that is **linear** in the size of F.

Key idea: introduce auxiliary variables to represent the output of subformulas, and constrain those variables using CNF clauses.

$x \rightarrow (y \wedge z)$

$a1$

$a1 \rightarrow (x \rightarrow a2)$

$(x \rightarrow a2) \rightarrow a1$

$a2 \leftrightarrow (y \wedge z)$

Equisatisfiability and Tseitin’s transformation

Formulas F and G are **equisatisfiable** if they are both satisfiable or they are both unsatisfiable.

Tseitin’s transformation converts a propositional formula F into an equisatisfiable CNF formula that is **linear** in the size of F.

Key idea: introduce auxiliary variables to represent the output of subformulas, and constrain those variables using CNF clauses.

$x \rightarrow (y \land z)$

$a1$

$\neg a1 \lor \neg x \lor a2$

$x \lor a1$

$\neg a2 \lor a1$

$\neg a2 \lor y$

$\neg a2 \lor z$

$\neg y \lor \neg z \lor a2$

Key feature of CNF: unit resolution

Resolution rule

$$\frac{a_1 \vee \ldots \vee a_n \vee \beta \qquad b_1 \vee \ldots \vee b_m \vee \neg\beta}{a_1 \vee \ldots \vee a_n \vee b_1 \vee \ldots \vee b_m}$$

Proving that a CNF formula is valid can be done using just this one proof rule!
Apply the rule until a contradiction (empty clause) is derived, or no more applications are possible.

Unit resolution rule

$$\frac{\beta \qquad b_1 \vee \ldots \vee b_m \vee \neg\beta}{b_1 \vee \ldots \vee b_m}$$

Unit resolution specializes the resolution rule to the case where one of the clauses is unit (a single literal).
SAT solvers use unit resolution in combination with backtracking search to implement a sound and complete procedure for deciding CNF formulas.

A basic SAT solver (DPLL)

```
// Returns true if the CNF formula F is
// satisfiable; otherwise returns false.
DPLL(F)
  G ← BCP(F)
  if $G = \top$ then return true
    if $G = \bot$ then return false
  p ← choose(vars(G))
  return DPLL($G\{p \mapsto \top\}$) ||
              DPLL($G\{p \mapsto \bot\}$)
```

Boolean constraint propagation applies unit resolution until fixed point.

If BCP cannot reduce F to a constant, we choose an unassigned variable and recurse assuming that the variable is either true or false.

If the formula is satisfiable under either assumption, then we know that it has a satisfying assignment (expressed in the assumptions). Otherwise, the formula is unsatisfiable.

Davis-Putnam-Logemann-Loveland (1962)

TODOs by next lecture

- The 2nd reading assignment is out
- Start working your homework assignment
- Form your team for the final project!
- Discuss your final project during office hour