Bug Catching: Automated Program Verification
15414/15614 Spring 2024
Lecture 23: Review

Ruben Martins

April 16, 2024

Resolution

Resolution rule

$$\frac{p \vee C \quad \neg p \vee D}{C \vee D} \bowtie_p$$

Exercise: Resolution

Show the following CNF formula is unsatisfiable by using resolution to derive the empty clause:

$C_1 \bowtie_{x_1} C_4 = \neg x_2 \vee x_3 = C_7$

$C_7 \bowtie_{x_2} C_6 =$

$x_3 \vee x_3 \equiv x_3 \quad = C_8$

$\neg x_1 \vee \neg x_2$	C_1
$x_1 \vee x_2$	C_2
$\neg x_1 \vee \neg x_3$	C_3
$x_1 \vee x_3$	C_4
$\neg x_2 \vee \neg x_3$	C_5
$x_2 \vee x_3$	C_6

$C_1 \bowtie_{x_1} C_2 = \neg x_2 \vee x_2$

Saturation

Saturation

- When we are not able to reach a contradiction with resolution, then by necessity the sequence of clauses must reach *saturation*, that is, any further application of resolution will only lead to clauses already in the sequence.
- If we reach saturation without deducing a contradiction and we conclude the initial theory is satisfiable.

Exercise: Saturation

Show that the following CNF formula is satisfiable by saturation

$$
\begin{array}{ll}
\neg p \vee \neg q \vee r & C_0 \\
p & C_1 \\
\neg r & C_2
\end{array}
$$

Reconstructing a Satisfying Assignment

Robinson's algorithm

Choose an ordering of the variables $p_0, p_1, \ldots, p_{n-1}$

$$
\begin{array}{lcll}
M_0 & = & \{\,\} & \\
M_{i+1} & = & M_i \cup \{p_i\} & \text{provided there is no } C \in S \text{ s.t. } C \subseteq \overline{M_i \cup \{p_i\}} \\
M_{i+1} & = & M_i \cup \{\neg p_i\} & \text{provided there is a } C \in S \text{ s.t. } C \subseteq \overline{M_i \cup \{p_i\}} \\
 & & & \\
M & = & M_n &
\end{array}
$$

Then $M \models S$, as proved by Robinson.

Exercise: Robinson's Algorithm

Reconstruct a satisfying assignment using Robinson's algorithm

$\neg p \vee \neg q \vee r$	C_0
p	C_1
$\neg r$	C_2
$\neg q \vee r$	$C_3 = C_1 \bowtie_p C_0$
$\neg q$	$C_4 = C_3 \bowtie_r C_2$
$\neg p \vee \neg q$	$C_5 = C_0 \bowtie_r C_2$

p, q, r

$M_0 = \{\}$

$M_1 = \{p\}, \{\neg p\}$

$M_2 = \{p, \neg q\}$

$\{\neg p, \neg q\}$ C_4

$M_3 = \{p, \neg q, \neg r\}$

$\{\neg p, q, \neg r\}$ C_2

Tseitin Encoding

Tseitin Encoding

- Introduce fresh variables to encode subformulas
- Encode the meaning of these fresh variables with clauses
- Guarantees equisatisfiability with a linear increase in the size of the formula

Exercise: Tseitin Encoding

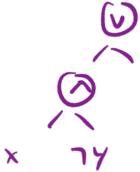
Use the Tseitin Encoding to transform the following propositional formula to CNF:

$\phi = (\underbrace{x \wedge \neg y}_{a}) \vee (z \vee (\underbrace{x \wedge \neg w}_{b}))$

$(a \vee z \vee b) \wedge$

$a \leftrightarrow (x \wedge \neg y)$

$b \leftrightarrow (x \wedge \neg w)$



$\rightsquigarrow$ $a \rightarrow (x \wedge \neg y)$

$\neg a \vee (x \wedge \neg y) \equiv (\neg a \vee x) \wedge (\neg a \vee \neg x)$

$(x \wedge \neg y) \rightarrow a \equiv (\neg x \vee y \vee a)$

Unary and Binary Representations

Unary and binary representations

1. Unary representation: a Boolean variable for each possible value
2. Binary representation: binary representation of an integer

Exercise: Representing Integer Variables in SAT

Suppose we want to encode the domain of an integer variable $X = \{1, 2, 3\}$
Encode the domain of this integer variables using:

- Unary representation
- Binary representation

x_1

x_2

x_3

x_i is true iff X has value i

$(x_1 \vee x_2 \vee x_3)$ at-least-one

$\left.\begin{array}{l}(\neg x_1 \vee \neg x_2) \\ (\neg x_1 \vee \neg x_3) \\ (\neg x_2 \vee \neg x_3)\end{array}\right\}$ at-most-one

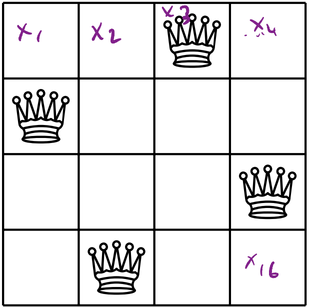
SAT Encodings

Variables and Constraints

1. Define the meaning of variables
2. Encode the constraints of the problem into CNF

$CNF(x_1 + x_2 + \dots + x_{16} = 4)$

$x_1 \vee x_2 \vee x_3 \vee x_4$
$\neg x_1 \vee \neg x_2$
$\vdots$



- There should be 4 queens on the board. $x_1 + x_2 + x_3 + x_4 = 1$
- Two queens should never be on the same line.
- Two queens should never be on the same column.
- Two queens should never be on the same diagonal.

Status of a Clause under Partial Interpretation

↳ partial assignment

Status of a clause

Given a partial interpretation I, a clause is:

- Satisfied, if one or more of its literals is satisfied
- Conflicting, if all of its literals are assigned but not satisfied
- Unit, if it is not satisfied and all but one of its literals are assigned
- Unresolved, otherwise

Exercise: Status of Clauses

p_2 is assigned false

Given the partial interpretation $I = \{p_1, \neg p_2, p_4\}$ what is the status of the following clauses:

- $(p_1 \vee p_3 \vee \neg p_4)$ satisfied
- $(\cancel{\neg p_1} \vee \cancel{p_2})$ conflict
- $(\cancel{p_2} \vee \neg \cancel{p_4} \vee p_3)$ unit
- $(\neg \cancel{p_1} \vee p_3 \vee p_5)$ unresolved

Unit Propagation

Unit Propagation

- Identify unit clauses:
  - Unit clauses: clauses that have exactly one unassigned literal
- Satisfy the unassigned literal by assigning true if it is positive (l_i) and false if it is negative ($\neg l_i$)
- Repeat until fix point

DPLL algorithm

```
let rec dpll (f: formula) : bool =
  let fp = bcp f in
  match fp with
  | Some True -> true
  | Some False -> false
  | None ->
    begin
    let p = choose_var f in
    let ft = (subst_var f p true) in
    let ff = (subst_var f p false) in
    dpll ft || dpll ff
    end
```

Exercise: DPLL algorithm

Considering the following propositional formula in CNF:

$$
\begin{array}{ll}
\neg x_1 \vee \cancel{x_2} \vee \neg x_3 & C_1 \\
\neg x_1 \vee \neg x_2 \vee \neg x_3 & C_2 \\
\cancel{x_1} \vee \neg x_2 \vee \neg x_3 & C_3 \\
\cancel{x_1} \vee \cancel{x_2} \vee \neg x_3 & C_4 \\
\neg x_1 \vee \cancel{x_2} \vee x_3 & C_5 \\
\cancel{x_1} \vee \cancel{x_2} \vee x_3 & C_6 \\
\neg x_1 \vee \neg x_2 \vee x_3 & C_7 \\
\cancel{x_1} \vee \neg x_2 \vee x_3 & C_8
\end{array}
$$

Decide x_2

Propagate $\neg x_3$ @ C_3

Conflict C_8

$C_3 \bowtie_{x_3} C_8 = x_1 \vee \neg x_2 = C_9$

Backtrack x_2

Propagate $\neg x_2$ @ C_9

Start with the following decision and apply the DPLL algorithm with unit propagation:

1. (1) Decide $\neg x_1$
2. ...

Congruence Closure

Congruence closure

1. Let S_P be the set of all terms, and their subterms (recursively), in P.
2. Initialize $\cong$ by placing each element of S_P in its own congruence class.
3. For every positive literal $s = t$ in P, merge the congruence classes of s and t.
4. While $\cong$ changes, repeat the following:
    4.1 Propagate the congruence axiom, to account for any merged congruence classes from the previous step. For any $s \cong t$, if $f(\ldots, s, \ldots)$ and $f(\ldots, t, \ldots)$ are currently in different congruence classes, then merge them.
5. Check the negative equality literals in P against the computed $\cong$.
    - For any $s \neq t$ appearing in P, if $s \cong t$, then return that P is unsat.
    - Otherwise, $s \ncong t$ for all $s \neq t$ appearing in P, so return that P is sat.

Exercise: Congruence closure

Use the congruence algorithm to determine the satisfiability of the following formula:
$f(g(x)) = g(f(x)) \wedge f(g(f(y))) = x \wedge f(y) = x \wedge g(f(x)) \neq x$

Nelson-Oppen

Nelson-Oppen

The Nelson-Oppen procedure for a formula φ that combines different theories consists of:

1. **Purification**: Purify φ into $F_1, \ldots, F_n$.
2. Apply the decision procedure for T_i to F_i. If there exists i such that F_i is unsatisfiable in T_i, then φ is unsatisfiable.
3. **Equality propagation**: If there exists i, j such that F_i T_i-implies an equality between variables of φ that is not T_j-implied by F_j, add this equality to F_j and go to step 2.
4. If all equalities have been propagated then the formula is satisfiable.

Exercise: Nelson-Oppen algorithm

Purification

$T_{\mathbb{R}}$	T_{EUF}
$u_4 = x + u_1 \wedge$	$u_1 = g(y) \wedge$
$u_5 \leq u_2 + u_3$	$u_2 = g(a) \wedge$
	$u_3 = f(b) \wedge$

$u_5 = f(u_4)$

Solve the following formula using the Nelson-Oppen algorithm:

$$\varphi = \underbrace{f(\overbrace{x + \underbrace{g(y)}_{u_1}}^{u_4}))}_{u_5} \leq \underbrace{g(a)}_{u_2} + \underbrace{f(b)}_{u_3}$$

Equality propagation

$T_{\mathbb{R}}$	T_{EUF}

DPLL(T)

DPLL(T)

The key idea behind this framework is to decompose the SMT problem into parts we can deal with efficiently:

- Use SAT solver to cope with the **Boolean structure** of the formula;
- Use dedicated conjunctive **theory solver** to decide satisfiability in the background theory.

Exercise: DPLL(T)

Theory	SAT
$g(a) = c \wedge$	$(P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4) \wedge$
$f(g(a)) \neq f(c) \wedge$	$\neg(P_1 \wedge \neg P_2 \wedge P_3 \wedge \neg P_4)$
$g(a) = d \wedge$	
$c \neq d$	

Use the DPLL(T) algorithm to determine if the following formula is satisfiable:

$$\varphi : \underbrace{g(a) = c}_{P_1} \wedge (\underbrace{f(g(a)) \neq f(c)}_{P_2} \vee \underbrace{g(a) = d}_{P_3}) \wedge \underbrace{c \neq d}_{P_4}$$

$I = \{ P_1, \neg P_2, P_3, \neg P_4 \}$

Bounded Model Checking

Bounded Model Checking

BMC computes an *underapproximation* of a program by assuming that all loops in the program are unrolled to some fixed, pre-determined finite depth k.

Exercise: From programs to SAT

$number_1^0$, $number_1^1$
$number_2^0$, $number_2^1$
sum_1^0 , sum_1^1
sum_2^0 , sum_2^1 , sum_2^2

Consider the domain of the numbers to be $\{0, 1, 2\}$ and:

- Precondition: number1 can be 0 or 1
- Precondition: number2 can be 0 or 1
- Postcondition: sum will be the sum of number1 and number2

```
int number1;
int number2;
int sum = number1;
 for(int i = 0; i < number2; i++) {
    sum += 1; // Increment sum by 1, number2 times
}
assert (sum == number1 + number2);
```

$(number_1^0 \lor number_1^1)$ $(\neg number_1^0 \lor \neg number_1^1)$

$number_2^0 \land sum_1^0 \rightarrow sum_2^0$

$number_2^0 \land sum_1^1 \rightarrow sum_2^1$

Write a CNF formula that corresponds to verifying the postcondition.

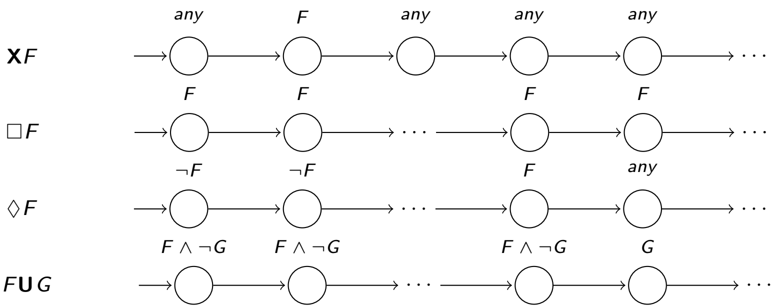
LTL Semantics

LTL Semantics

The truth of LTL formulas in a trace σ is defined inductively as follows:

1. $\sigma \models F$ iff $\sigma_0 \models F$ for a state formula F provided that $\sigma_0 \neq \Lambda$
2. $\sigma \models \neg P$ iff $\sigma \not\models P$, i.e. it is not the case that $\sigma \models P$
3. $\sigma \models P \wedge Q$ iff $\sigma \models P$ and $\sigma \models Q$
4. $\sigma \models \mathbf{X}P$ iff $\sigma^1 \models P$
5. $\sigma \models \Box P$ iff $\sigma^i \models P$ for all $i \geq 0$
6. $\sigma \models \Diamond P$ iff $\sigma^i \models P$ for some $i \geq 0$
7. $\sigma \models P \mathbf{U} Q$ iff there is an $i \geq 0$ such that $\sigma^i \models Q$ and $\sigma^j \models P$ for all $0 \leq j < i$

Examples of traces satisfying LTL formulas



Kripke Structure

Definition (Kripke structure)

- A *Kripke frame* $(W, \curvearrowright)$ consists of a set W with a transition relation $\curvearrowright \subseteq W \times W$
  - $s \curvearrowright t$ indicates that there is a direct transition from s to t in the Kripke frame $(W, \curvearrowright)$
  - The elements $s \in W$ are called states.
- A*Kripke structure* $K = (W, \curvearrowright, v, I)$ is a Kripke frame $(W, \curvearrowright)$ with a mapping $v : W \to 2^V$
  - 2^V is the powerset of V assigning truth-values to all the propositional atoms in all states.
  - A Kripke structure has a set of initial states $I \subseteq W$.

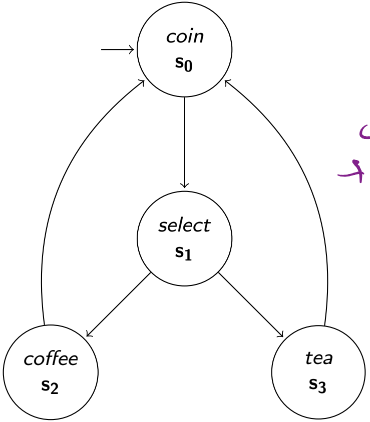
Kripke Structure

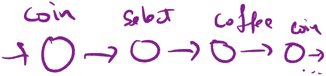
Computation Structure

A Kripke structure $K = (W, \curvearrowright, v, I)$ is called a *computation structure* if:

- W is a finite set of states
- every element $s \in W$ has at least one direct successor $t \in W$ with $s \curvearrowright t$.

Exercise: LTL Formulas and Kripke Structures





Which formulas are satisfied by this Kripke structure?

- $\Box coffee$ False
- $\Box\Diamond(coffee \lor tea)$ True
- $\Box(coin \land \mathbf{X} select \to \Diamond(coffee \lor tea))$ True

Exercise: LTL Formulas

Show that the following LTL formulas are valid using the semantics of the LTL operators or provide a counterexample if they are incorrect.

1. $\Diamond(P \vee Q) \leftrightarrow \Diamond P \vee \Diamond Q$
2. $\Box(P \vee Q) \leftrightarrow \Box P \vee \Box Q$

CTL Semantics

CTL Semantics

In a fixed computation structure $K = (W, \curvearrowright, v)$, the truth of CTL formulas in state s is defined inductively as follows:

- $s \models \mathbf{AX}P$ iff all successors t with $s \curvearrowright t$ satisfy $t \models P$
- $s \models \mathbf{EX}P$ iff at least one successor t with $s \curvearrowright t$ satisfies $t \models P$
- $s \models \mathbf{A}\Box P$ iff all paths $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfy $s_i \models P$ for all $i \geq 0$
- $s \models \mathbf{E}\Box P$ iff some path $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfies $s_i \models P$ for all $i \geq 0$

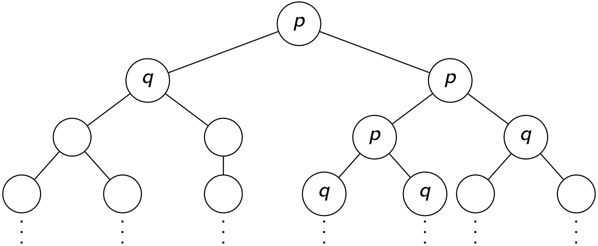
CTL Semantics

CTL Semantics

In a fixed computation structure $K = (W, \curvearrowright, v)$, the truth of CTL formulas in state s is defined inductively as follows:

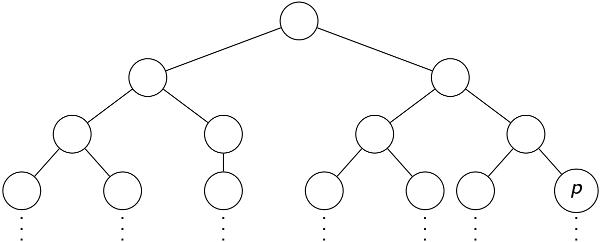
- $s \models \mathbf{A}\Diamond P$ iff all paths $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfy $s_i \models P$ for some $i \geq 0$
- $s \models \mathbf{E}\Diamond P$ iff some path $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfies $s_i \models P$ for some $i \geq 0$
- $s \models \mathbf{A} P \mathbf{U} Q$ iff all paths $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ have some $i \geq 0$ such that $s_i \models Q$ and $s_j \models P$ for all $0 \leq j < i$
- $s \models \mathbf{E} P \mathbf{U} Q$ iff some path $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ has some $i \geq 0$ such that $s_i \models Q$ and $s_j \models P$ for all $0 \leq j < i$

Example: Visualization of CTL formulas



Visualization of a CTL formula: $\mathbf{A}[P\,\mathbf{U}\,Q]$

Example: Visualization of CTL formulas



Visualization of a CTL formula: $\mathbf{E}\Diamond P$

Exercise: CTL vs. LTL

Show that the following formulas are not equivalent by giving a Kripke structure that satisfies one formula but not the other:

- LTL formula $\Diamond\Box P$ True
- CTL formula $\mathbf{AFAG}P$ False