

Formal Software Verification Project Presentation

SAT Solver

Nathan Schmidt

January 6, 2025

Aarhus University

Syntax and Semantics

Syntax

$$p,q ::= x \mid \mathtt{true} \mid \mathtt{false} \mid p \land q \mid p \lor q \mid p \to q \mid \neg p$$

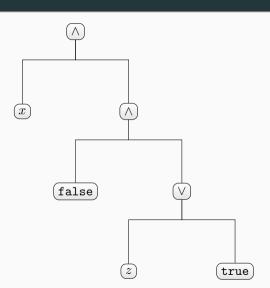
Semantics

- Valuation $v : id \rightarrow bool$
- Interpreter interp : valuation \times form \rightarrow bool

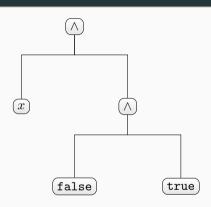
Simplifications

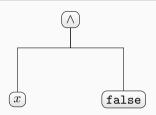
$\mathtt{true} \wedge p \equiv p$	$p \wedge \mathtt{true} \equiv p$	$\mathtt{false} \wedge p \equiv \mathtt{false}$	$p \land \mathtt{false} \equiv \mathtt{false}$
$\mathtt{true} \lor p \equiv \mathtt{true}$	$ hoee$ true \equiv true	$\mathtt{false} \lor p \equiv p$	$p \vee \mathtt{false} \equiv p$
$\mathtt{true} \to p \equiv p$	$p o exttt{true} \equiv exttt{true}$	$\mathtt{false} o p \equiv \mathtt{true}$	$ ho o exttt{false} \equiv eg ho$
	$\neg \mathtt{true} \equiv \mathtt{false}$	$\neg \mathtt{false} \equiv \mathtt{true}$	

- Depth-First-Search (DFS):
 - **X** Pre-order: no obviously decreasing arguments
 - ✓ Post-order



- Depth-First-Search (DFS):
 - **X** Pre-order: no obviously decreasing arguments
 - ✓ Post-order





- Depth-First-Search (DFS):
 - **X** Pre-order: no obviously decreasing arguments
 - ✓ Post-order

(false)

- Depth-First-Search (DFS):
 - **X** Pre-order: no obviously decreasing arguments
 - ✓ Post-order

Optimizer Minimality and Correctness

Minimality

- Theorem. optim p always in minimal form
- p in minimal form:
 - if p = true/false, or
 - true/false $\notin p$

Optimizer Minimality and Correctness

Minimality

- Theorem. optim p always in minimal form
- *p* in minimal form:
 - if p = true/false, or
 - true/false $\notin p$

Correctness

Theorem. $\forall v, p : interp \ v \ p = interp \ v \ (optim \ p)$

$$(((x
ightarrow \neg y) \lor (x \land \neg x)) \land (y \land z)) \land \mathsf{true}$$

$$(((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)) \land \text{true}$$

$$\downarrow \text{optim}$$

$$((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)$$

```
(((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)) \land \text{true}
\downarrow \text{optim}
((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)
\downarrow \text{collect\_ids}
[x; y; z]
```

```
(((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)) \land \text{true})
\downarrow \text{optim}
((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)
\downarrow \text{collect\_ids}
[x; y; z]
\downarrow \text{collect\_vals}
```

```
(((x \rightarrow \neg y) \lor (x \land \neg x)) \land (y \land z)) \land \mathsf{true}
                                                optim
                            ((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)
                                                collect_ids
                                         [x; y; z]
                                                collect vals
(x !-> true ;; y !-> true ;; z !-> true)^X; (x !-> true ;; y !-> true)^X;
(x !-> true ;; z !-> true)^X; (x !-> true)^X; (y !-> true :: z !-> true)^{\checkmark};
                (v ! \rightarrow true)^?: (z ! \rightarrow true)^?: empty_valuation?
                                                find_valuation
                         Some (y !-> true ;; z !-> true)
```

```
(((x \rightarrow \neg y) \lor (x \land \neg x)) \land (y \land z)) \land \mathsf{true}
                                                optim
                            ((x \to \neg y) \lor (x \land \neg x)) \land (y \land z)
                                                collect_ids
                                         [x; y; z]
                                                collect vals
(x !-> true ;; y !-> true ;; z !-> true)^X; (x !-> true ;; y !-> true)^X;
(x !-> true ;; z !-> true)^X; (x !-> true)^X; (y !-> true ;; z !-> true)^{\checkmark};
                (v ! \rightarrow true)^?: (z ! \rightarrow true)^?: empty_valuation?
                                                find_valuation
                        Some (y !-> true ;; z !-> true)
                                                solver
                                            true
```

Solver Soundness and Completeness

Soundness

- Lemma. $\forall p : solver \ p = true \Longrightarrow satisfiable \ p$
- Proof by induction
- Do not care about exact returned valuation

Solver Soundness and Completeness

Soundness

- Lemma. $\forall p : solver \ p = true \Longrightarrow satisfiable \ p$
- Proof by induction
- Do not care about exact returned valuation

Completeness

- Lemma. $\forall p : satisfiable p \Longrightarrow solver p = true$
- interp $v \ p = \text{true} \implies v \in \text{collect_vals} (\text{collect_ids} \ p))$

Solver Soundness and Completeness

Soundness

- Lemma. $\forall p : solver \ p = true \Longrightarrow satisfiable \ p$
- Proof by induction
- Do not care about exact returned valuation

Completeness

- Lemma. $\forall p : satisfiable p \Longrightarrow solver p = true$
- interp $v p = \text{true} \implies v \in \text{collect_vals} (\text{collect_ids } p))$
- Proof attempt:
 - $\forall v, v' : \forall x \in p, v x = v' x \Longrightarrow \text{interp } v \ p = \text{interp } v' \ p$
 - $\forall v : \exists v' : \forall x \in p, v = v' x \text{ and } v' \in \text{collect_vals (collect_ids } p)$)

Solver Decision Procedure

Theorem

$$\forall \, p : \textit{solver} \ p = \textit{true} \Longleftrightarrow \textit{satisfiable} \ p$$

Solver Decision Procedure

Theorem

 $\forall p : solver p = true \iff satisfiable p$

i.e., solver decision procedure for SAT