Automated Deduction Compendium SS2023

Ida Hönigmann

May 8, 2023

1 Introduction, SAT Solving

1.0.1 Proposition, Formulas

Def (Proposition)**.** *Proposition is a statement that can be either true or false.*

Def (Propositional formula, Atom, Connective)**.** *Atoms are boolean variables (e.g. p,q).*

1. *Atoms are formulas.*
2. $\top$, $\bot$ *are formulas.*
3. *If A is a formula, then $\neg A$ is a formula.*
4. *If $A_1, ..., A_n$ are formulas, then $(A_1 \wedge ... \wedge A_n)$ and $(A_1 \vee ... \vee A_n)$ are formulas.*
5. *If A and B are formulas, then $A \rightarrow B$ and $A \leftrightarrow B$ are formulas.*

The symbols $\top, \bot, \wedge, \vee, \neg, \rightarrow, \leftrightarrow$ are called logical connectives.

1.0.2 Precedence

Connective	Name	Precedence
$\top$	verum	
$\bot$	falsum	
$\neg$	negation	5
$\wedge$	conjunction	4
$\vee$	disjunction	3
$\rightarrow$	implication	2
$\leftrightarrow$	equivalence	1

1.0.3 Boolean Values, Interpretation

Def (Boolean values, Interpretation)**.** *There are two boolean vales: true (1) and false (0).*
An interpretation for a set P of boolean variables is a mapping $I : P \rightarrow \{0, 1\}$.

1.0.4 Interpreting formulas

1. $I(\top) = 1$ and $I(\bot) = 0$
2. $I(A_1 \wedge ... \wedge A_n) = 1$ iff $I(A_i) = 1$ for all i
3. $I(A_1 \vee ... \vee A_n) = 1$ iff $I(A_i) = 1$ for some i
4. $I(\neg A) = 1$ iff $I(A) = 0$
5. $I(A_1 \rightarrow A_2) = 1$ iff $I(A_1) = 0$ or $I(A_2) = 1$
6. $I(A_1 \leftrightarrow A_2) = 1$ iff $I(A_1) = I(A_2)$

1.0.5 Safisfiable, Valid, Model

Def (Satisfiable, Model, Valid). *If $I(A) = 1$ then I satisfies A and I is a model of A, denoted by $I \models A$.*
A is satisfiable if some interpretation is a model of A.
A is valid if every interpretation is a model of A.
A and B are equivalent, denoted by $A \equiv B$, if they have the same models.

1.0.6 Connection valid, satisfiable

1. A is valid iff $\neg A$ is unsatisfiable.
2. A is satisfiable iff $\neg A$ is not valid.

1.0.7 Equivalent replacement

Def (Equivalent replacement). *$A[B]$ is a formula A with a fixed occurrence of subformula B. $A[B']$ is the formula A where every occurrence of B is replaced by B'.*

Lemma 1 (Equivalent Replacement). *Let I be an interpretation and $I \models A_1 \leftrightarrow A_2$. Then $I \models B[A_1] \leftrightarrow B[A_2]$.*
Let $A_1 \equiv A_2$. Then $B[A_1] \equiv B[A_2]$.

1.0.8 Evaluating a formula

Algorithm 1.
```
procedure evaluate(G,I)
input: formula G, interpretation I
output: the boolean value I(G)
begin
  forall atoms p occurring in G
    if I models p
      then replace all occurrences of p in G by 1;
      else replace all occurrences of p in G by 0;
  rewrite G into a normal form using the rewrite rules
  if G = 1 then return 1 else return 0
end
```

2 Splitting, Polarities

2.0.1 Soundness of Splitting

$A_p^\perp$ and $A_p^\top$ are obtained by replacing in A all occurrences of p by $\perp$ and $\top$ respectively.

Lemma 2. *Let p be an atom, A be a formula, and I be an interpretation.*

1. *If $I \not\models p$, then A is equivalent to $A_p^\perp$ in I.*
2. *If $I \models p$, then A is equivalent to $A_p^\top$ in I.*

Lemma 3. *Let A be a formula and p an atom.*
Then A is satisfiable iff at least one of the formulas $A_p^\top$ and $A_p^\perp$ is satisfiable.

2.0.2 Splitting

Algorithm 2.
```
procedure split(G)
parameters: function select
input: formula G
output: ''satisfiable'' or ''unsatisfiable''
begin
  G := simplify(G) # rewrite rules
  if G = 1 then return ''satisfiable''
  if G = 0 then return ''unsatisfiable''
```

```
  (p,b) := select(G)
  case b of
  1 =>
    if split(replace(G,p,1)) = ''satisfiable''
      then return ''satisfiable''
      else return split(replace(G,p,0))
  0 =>
    if split(replace(G,p,0)) = ''satisfiable''
      then return ''satisfiable''
      else return split(replace(G,p,1))
end
```

2.0.3 Polarities

1. $A|_\epsilon = A$ and $pol(A, \epsilon) = 1$
2. If $A|\pi = B_1 \wedge ... \wedge B_n$ or $A|\pi = B_1 \vee ... \vee B_n$ then $A|_{\pi.i} = B_i$ and $pol(A, \pi.i) = pol(A, \pi)$.
3. If $A|_p i = \neg B$ then $A|_{\pi.1} = B$ and $pol(A, \pi.1) = -pol(A, \pi)$.
4. If $A|_\pi = B_1 \rightarrow B_2$ then $A|_{\pi.1} = B_1$, $A|_{\pi.2} = B_2$ and $pol(A, \pi.1) = -pol(A, \pi)$, $pol(A, \pi.2) = pol(A, \pi)$.
5. If $A|_\pi = B_1 \leftrightarrow B_2$ then $A|_{\pi.1} = B_1$, $A|_{\pi.2} = B_2$ and $pol(A, \pi.1) = 0 = pol(A, \pi.2)$.

2.0.4 Monotonic replacement

Denote with $A[B]_\pi$ formula A with the subformula at the position π replaced by B.

Lemma 4 (Monotonic Replacement)**.** *Let A, B, B' be formulas, I be an interpretation, and $I \models B \rightarrow B'$. If $pol(A, \pi) = 1$, then $I \models A[B]_\pi \rightarrow A[B']_\pi$. Likewise, if $pol(A, \pi) = -1$ then $I \models A[B']_\pi \rightarrow A[B]_\pi$.*

2.0.5 Pure Atom

Def. *Atom p is pure in a formula A, if either all occurrences of p in A are positive or all occurrences of p in A are negative.*

Lemma 5 (Pure Atom)**.** *Let p have only positive occurrences in A and $I \models A$. Define $I' = I + (p \mapsto 1)$. Then $I' \models A$. Likewise, let p have only negative occurrences in A and $I \models A$. Define $I' = I + (p \mapsto 0)$. Then $I' \models A$.*

Lemma 6 (Pure Atom)**.** *Let an atom p have only positive (respectively, only negative) occurrences in A. Then A is satisfiable iff $A_p^\top$ (respectively, $A_p^\bot$) is satisfiable.*

2.0.6 Splitting with pure atom optimization

Algorithm 3. `procedure split(G)`

```
parameters: function select
input: formula G
output: ''satisfiable'' or ''unsatisfiable''
begin
  G := simplify_with_pure_atoms(G)
  if G = 1 then return ''satisfiable''
  if G = 0 then return ''unsatisfiable''
  (p,b) := select(G)
  case b of
  1 =>
    if split(replace(G,p,1)) = ''satisfiable''
      then return ''satisfiable''
      else return split(replace(G,p,0))
  0 =>
```

```
    if split(replace(G,p,0)) = ''satisfiable''
      then return ''satisfiable''
      else return split(replace(G,p,1))
end
```

3 CNF, DPLL, MiniSat

3.0.1 Clause

Def (Literal, Clause, Empty clause, Unit clause, Horn clause)**.** *A literal is either an atom p or its negation $\neg p$.*
A clause is a disjunction of literals $L_1 \vee ... \vee L_n$.
The empty clause $\square$ is false in every interpretation.
If $n = 1$ then the clause is called unit clause.
A horn clause is a clause with at most one positive literal.

3.0.2 CNF

Def (CNF)**.** *A formula A is in conjenctive normal form if it is $\top$, $\bot$ or a conjunction of disjunctions of literals $\bigwedge_i \bigvee_j L_{i,j}$.*

3.0.3 Naming

If A is a non-trivial subformula A. Introduce a new name n for it. Add formula $n \leftrightarrow A$ and replace subformula by its name in the original formula.

Lemma 7 (Naming)**.** *Let S be a set of formulas and A a formula. Let n be a boolean variable not occurring in S, nor in A.*
Then S is satisfiable iff the set of formulas $S \cup \{n \leftrightarrow A\}$ is satisfiable.

3.0.4 Optimized CNF Transformation

Introduce a new name n every subformula B and replace it with the name. If the subformula occurs only positively then add $n \rightarrow B$. If it occurs only negatively then add $B \rightarrow n$ and if it does not occur only positively or negatively than add $n \leftrightarrow B$.

Lemma 8. *A set of formulas is satisfiable iff the optimized CNF transformation of these formulas is satisfiable.*

3.0.5 Unit propagation

Let S be a set of clauses. If S contains a unit clause L then remove from S every clause of the form $L \vee C$ and replace in S every clause of the form $\bar{L} \vee C$ by the clause C.

3.0.6 DPLL = splitting + unit propagation

Algorithm 4. `procedure DPLL(S)`

```
input: set of clauses S
output: satisfiable or unsatisfiable
parameters: function select_literal
begin
  S := propagate(S) # unit propagation
  if S is empty then return satisfiable
  if S contains 0 then return unsatisfiable
  L := select_literals(S) # splitting
  if DPLL(S union {L}) = satisfiable
    then return satisfiable
    else return DPLL(S union {not L})
end
```

Tautologies (e.g. $p \vee \neg p \vee C$) can be removed.

3.0.7 Pure literals

Def (Pure literal)**.** *A literal L in S is called pure if S contains no clauses of the form $\bar{L} \vee C$.*

If L is a pure literal in S then all clauses containing this literal can be removed.

4 Random SAT, Horn clauses

5 First-Order Logic, Theories

6 SMT, Theory of Equality, DPLL(T)

7 Theory of Arrays, Theory Combination, Nelson-Oppen, Z3

8 First-Order Theorem Proving, TPTP, Inference Systems

9 Selection functions, Saturation, Fairness and Redundancy

10 Redundancy, First-Order Reasoning with Equality

11 Ground Superposition, Term Orderings

12 Unification and Lifting

13 Non-Ground Superposition