CS228 : Logic for Computer Science

Module 2: The Problem of Satisfiability

Lecture 9: The DPLL Algorithm

Instructor: S. Akshay

IIT Bombay, India

Logistics

Quiz 1 (reminder)

- Aug 28th 5.10pm
- Venue: LH 101, 102,
  LT 105 (for PwD students who wish to avail compensatory time.)
- Syllabus: All of Module 1 (Propositional Logic), up to and including Tseitin's Encoding.

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Module 2: The Problem of Satisfiability

- Propositional Resolution - soundness, termination and completeness

Topics Covered till date

Module 1: Propositional Logic

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Module 2: The Problem of Satisfiability

- Propositional Resolution - soundness, termination and completeness
- Today: wrapping up completeness, the DPLL algorithm

A Quick Recap: Propositional Resolution

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- Resolution rule: $p \vee C$ and $\neg p \vee D$ are clauses, p is pivot, clause $C \vee D$ resolvent.

A Quick Recap: Propositional Resolution

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- Resolution rule: $p \vee C$ and $\neg p \vee D$ are clauses, p is pivot, clause $C \vee D$ resolvent.

Resolution as a Saturation Algorithm

Given propositional logic formula F:

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till possible, the following:
   - Apply Resolution Rule to two clauses in F' to obtain new clause $C \notin F'$. Update $F' := F' \cup \{C\}$.
3. At fixed point (i.e., when set F' does not change), answer YES if $\bot \in F'$, else answer NO.

A Quick Recap: Propositional Resolution

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- Resolution rule: $p \vee C$ and $\neg p \vee D$ are clauses, p is pivot, clause $C \vee D$ resolvent.

Resolution as a Saturation Algorithm

Given propositional logic formula F:

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till possible, the following:
   - Apply Resolution Rule to two clauses in F' to obtain new clause $C \notin F'$. Update $F' := F' \cup \{C\}$.
3. At fixed point (i.e., when set F' does not change), answer YES if $\bot \in F'$, else answer NO.

Properties

- The algo is sound, complete, always terminates and non-deterministic.
- It can be lifted to proof by refutation/contradiction of $F \vdash G$.

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $Res^1(F) = F \cup \{q,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p, \neg p,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p, \neg p, \bot\}$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

$Res^n(F)$: set of clauses derivable via resolution proof of depth at most n from clauses in F.

- $Res^0(F) := F$, $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

$Res^*(F)$: the fixed point

Since there are only finitely many variables in F, $Res^n(F)$ must saturate at some point: there exists m such that $Res^{m+1}(F) = Res^m(F)$. We define $Res^*(F) \triangleq Res^m(F)$.

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p, \neg p, \bot\}$

$Res^3(F) = Res^2(F)$ – saturated! $\Rightarrow Res^*(F) = Res^3(F)$.

Completeness: Main Theorem

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Completeness: Main Theorem

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.

Completeness: Main Theorem

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F.

Completeness: Main Theorem

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F.

base case:

p is the only variable in F.

Completeness: Main Theorem

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F.

base case:

p is the only variable in F.
Now, if F is unsatisfiable, then either $\perp \in F$ already (trivial: $\perp \in Res^0(F)$), or we must have $\{p, \neg p\} \subseteq F$.

Completeness: Main Theorem

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F.

base case:

p is the only variable in F.
Now, if F is unsatisfiable, then either $\perp \in F$ already (trivial: $\perp \in Res^0(F)$), or we must have $\{p, \neg p\} \subseteq F$.
In the latter case, applying the resolution rule with $C = D = \emptyset$, we have the following resolution derivation of $\perp$.

$$\frac{p \qquad \neg p}{\perp}$$

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.
- $F_{np} \triangleq$ set of clauses from F that have $\neg p$.

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.
- $F_{np} \triangleq$ set of clauses from F that have $\neg p$.
- $F_o \triangleq$ set of clauses from F that have neither p nor $\neg p$.

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.
- $F_{np} \triangleq$ set of clauses from F that have $\neg p$.
- $F_o \triangleq$ set of clauses from F that have neither p nor $\neg p$.

$F = F_p \wedge F_{np} \wedge F_o$

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.
- $F_{np} \triangleq$ set of clauses from F that have $\neg p$.
- $F_o \triangleq$ set of clauses from F that have neither p nor $\neg p$.

$F = F_p \wedge F_{np} \wedge F_o$

Furthermore, let

- $F'_p \triangleq \{C - \{p\} | C \in F_p\}$, i.e., from each clause in F_p drop the p!

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.
- $F_{np} \triangleq$ set of clauses from F that have $\neg p$.
- $F_o \triangleq$ set of clauses from F that have neither p nor $\neg p$.

$F = F_p \wedge F_{np} \wedge F_o$

Furthermore, let

- $F'_p \triangleq \{C - \{p\} | C \in F_p\}$, i.e., from each clause in F_p drop the p!
- $F'_{np} \triangleq \{C - \{\neg p\} | C \in F_{np}\}$

...

Completeness - 2 (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all sets of clauses containing at most n propositions.

Consider an unsatisfiable set F of clauses containing propositions $p_1, \ldots p_n, p$.

We define

- $F_p \triangleq$ set of clauses from F that have p.
- $F_{np} \triangleq$ set of clauses from F that have $\neg p$.
- $F_o \triangleq$ set of clauses from F that have neither p nor $\neg p$.

$F = F_p \wedge F_{np} \wedge F_o$

Furthermore, let

- $F'_p \triangleq \{C - \{p\} | C \in F_p\}$, i.e., from each clause in F_p drop the p!
- $F'_{np} \triangleq \{C - \{\neg p\} | C \in F_{np}\}$

Exercise 9.1 ...

Using natural deduction or truth tables, show that $F'_p \models F_p$ and $F'_{np} \models F_{np}$

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1.

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.
  - Observe that F'_p and F_o have no p, hence $\pi' \models F'_p$ and $\pi' \models F_o$.

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.
  - Observe that F'_p and F_o have no p, hence $\pi' \models F'_p$ and $\pi' \models F_o$.
  - Now, since $F'_p \models F_p$, we get $\pi' \models F_p$.
  - Finally, all clauses of F_{np} have $\neg p$ appearing in them, hence $\pi' \models F_{np}$.

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.
  - Observe that F'_p and F_o have no p, hence $\pi' \models F'_p$ and $\pi' \models F_o$.
  - Now, since $F'_p \models F_p$, we get $\pi' \models F_p$.
  - Finally, all clauses of F_{np} have $\neg p$ appearing in them, hence $\pi' \models F_{np}$.
  - Therefore we conclude that $\pi' \models F_p \wedge F_{np} \wedge F_o$, i.e., $\pi' \models F$.

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.
  - Observe that F'_p and F_o have no p, hence $\pi' \models F'_p$ and $\pi' \models F_o$.
  - Now, since $F'_p \models F_p$, we get $\pi' \models F_p$.
  - Finally, all clauses of F_{np} have $\neg p$ appearing in them, hence $\pi' \models F_{np}$.
  - Therefore we conclude that $\pi' \models F_p \wedge F_{np} \wedge F_o$, i.e., $\pi' \models F$.
- Case 2: $\pi \models (F'_{np} \wedge F_o)$. Call this G_2

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.
  - Observe that F'_p and F_o have no p, hence $\pi' \models F'_p$ and $\pi' \models F_o$.
  - Now, since $F'_p \models F_p$, we get $\pi' \models F_p$.
  - Finally, all clauses of F_{np} have $\neg p$ appearing in them, hence $\pi' \models F_{np}$.
  - Therefore we conclude that $\pi' \models F_p \wedge F_{np} \wedge F_o$, i.e., $\pi' \models F$.
- Case 2: $\pi \models (F'_{np} \wedge F_o)$. Call this G_2 **Exercise 9.2: Argue symmetrically that π' with $\pi'(p) = 1$ will satisfy F in this case.**

Completeness - 3 (contd.)

Proof(contd.)

Now consider the formula $G = \underbrace{(F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)}$

Note! p does not appear in this formula

Claim: If G is satisfiable then F is satisfiable.

- Consider a satisfying assignment π of G, i.e., $\pi \models G$. π is defined on $p_1, \ldots p_n$
- Case 1: $\pi \models (F'_p \wedge F_o)$. Let's call this G_1. Consider assignment π', $\pi'(p) = 0$, $\pi'(p') = \pi(p')$ for all $p' \in \{p_1, \ldots p_n\}$.
  - Observe that F'_p and F_o have no p, hence $\pi' \models F'_p$ and $\pi' \models F_o$.
  - Now, since $F'_p \models F_p$, we get $\pi' \models F_p$.
  - Finally, all clauses of F_{np} have $\neg p$ appearing in them, hence $\pi' \models F_{np}$.
  - Therefore we conclude that $\pi' \models F_p \wedge F_{np} \wedge F_o$, i.e., $\pi' \models F$.
- Case 2: $\pi \models (F'_{np} \wedge F_o)$. Call this G_2 **Exercise 9.2: Argue symmetrically that π' with $\pi'(p) = 1$ will satisfy F in this case.**
- Therefore, in both cases we obtain that $F = F_p \wedge F_{np} \wedge F_o$ is satisfiable. ...

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved. So, if we put back p in the clauses of ρ_1, this derivation of $\perp$ will now give a derivation of p!

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved. So, if we put back p in the clauses of ρ_1, this derivation of $\perp$ will now give a derivation of p!
    - Similarly, if we put back $\neg p$ in the clauses of ρ_2, the derivation of $\perp$ will give a derivation of $\neg p$!

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved. So, if we put back p in the clauses of ρ_1, this derivation of $\perp$ will now give a derivation of p!
    - Similarly, if we put back $\neg p$ in the clauses of ρ_2, the derivation of $\perp$ will give a derivation of $\neg p$!
    - Thus, we obtain $p \in Res^*(F_p \wedge F_o)$ and $\neg p \in Res^*(F_{np} \wedge F_o)$

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved. So, if we put back p in the clauses of ρ_1, this derivation of $\perp$ will now give a derivation of p!
    - Similarly, if we put back $\neg p$ in the clauses of ρ_2, the derivation of $\perp$ will give a derivation of $\neg p$!
    - Thus, we obtain $p \in Res^*(F_p \wedge F_o)$ and $\neg p \in Res^*(F_{np} \wedge F_o)$, which (again by weakening) implies $p, \neg p \in Res^*(F_p \wedge F_{np} \wedge F_o)$, and hence $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved. So, if we put back p in the clauses of ρ_1, this derivation of $\perp$ will now give a derivation of p!
    - Similarly, if we put back $\neg p$ in the clauses of ρ_2, the derivation of $\perp$ will give a derivation of $\neg p$!
    - Thus, we obtain $p \in Res^*(F_p \wedge F_o)$ and $\neg p \in Res^*(F_{np} \wedge F_o)$, which (again by weakening) implies $p, \neg p \in Res^*(F_p \wedge F_{np} \wedge F_o)$, and hence $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$.

Completeness - 4 (contd.)

Proof(contd.)

- As we started with F unsat, we obtain $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o)$ is unsat. (why?)
- Now we apply the induction hypothesis on G_1 and G_2 (which do not contain p).
- Thus we obtain, $\perp \in Res^*(G_1)$ and $\perp \in Res^*(G_2)$.
- Take any derivation ρ_1 of $\perp$ from G_1 and a derivation ρ_2 of $\perp$ from G_2. Two cases:
  - Case 1: $\perp$ was derived using only clauses from F_o in ρ_1 or ρ_2. In this case $\perp \in Res^*(F_o)$. This implies $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$ (by weakening: if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$).
  - Case 2: In ρ_1 a clause from F'_p is involved AND in ρ_2 a clause from F'_{np} is involved.
    - In derivation ρ_1, p does not occur, hence p cannot get resolved. So, if we put back p in the clauses of ρ_1, this derivation of $\perp$ will now give a derivation of p!
    - Similarly, if we put back $\neg p$ in the clauses of ρ_2, the derivation of $\perp$ will give a derivation of $\neg p$!
    - Thus, we obtain $p \in Res^*(F_p \wedge F_o)$ and $\neg p \in Res^*(F_{np} \wedge F_o)$, which (again by weakening) implies $p, \neg p \in Res^*(F_p \wedge F_{np} \wedge F_o)$, and hence $\perp \in Res^*(F_p \wedge F_{np} \wedge F_o)$.

  Thus in both cases, $\perp \in Res^*(F)$ which completes the induction step, and hence the proof.

□

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1:

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p =$ $F_{np} =$ $F_o =$

  $F'_p =$ $F'_{np} =$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$

  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$

  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$

- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) =$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_p' = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_{np}' = \{\neg p_2\}$ $\quad F_o = \{p_2\}$

- Step 2: $G = G_1 \vee G_2 = (F_p' \wedge F_o) \vee (F_{np}' \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$
- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\perp$ and G_2 to $\perp$

$$\dfrac{\dfrac{p_1 \qquad \neg p_1 \vee \neg p_2}{\neg p_2} \qquad p_2}{\perp} \qquad\qquad \dfrac{\neg p_2 \qquad p_2}{\perp}$$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$
- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\bot$ and G_2 to $\bot$
- Step 4: Convert to proof from F to $\bot$ by adding p and $\neg p$ (Note that we are in Case 2)

$$\dfrac{\dfrac{p_1 \qquad \neg p_1 \vee \neg p_2}{\neg p_2} \qquad p_2}{\bot} \qquad\qquad \dfrac{\neg p_2 \qquad p_2}{\bot}$$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$
- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\perp$ and G_2 to $\perp$
- Step 4: Convert to proof from F to $\perp$ by adding p and $\neg p$ (Note that we are in Case 2)

$$\dfrac{\dfrac{p_1 \vee \mathbf{p} \quad \neg p_1 \vee \neg p_2}{\neg p_2} \qquad p_2}{\perp} \qquad \dfrac{\neg p_2 \qquad p_2}{\perp}$$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$
- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\perp$ and G_2 to $\perp$
- Step 4: Convert to proof from F to $\perp$ by adding p and $\neg p$ (Note that we are in Case 2)

$$\dfrac{\dfrac{p_1 \vee \mathbf{p} \quad \neg p_1 \vee \neg p_2 \vee \mathbf{p}}{\neg p_2} \qquad p_2}{\perp} \qquad\qquad \dfrac{\neg p_2 \qquad p_2}{\perp}$$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$

- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\perp$ and G_2 to $\perp$
- Step 4: Convert to proof from F to $\perp$ by adding p and $\neg p$ (Note that we are in Case 2)

$$\dfrac{\dfrac{p_1 \vee \mathbf{p} \quad \neg p_1 \vee \neg p_2 \vee \mathbf{p}}{\neg p_2 \vee \mathbf{p}} \qquad p_2}{\perp \vee \mathbf{p}} \qquad\qquad \dfrac{\neg p_2 \qquad p_2}{\perp}$$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$
- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\bot$ and G_2 to $\bot$
- Step 4: Convert to proof from F to $\bot$ by adding p and $\neg p$ (Note that we are in Case 2)

$$\dfrac{\dfrac{p_1 \vee \mathbf{p} \quad \neg p_1 \vee \neg p_2 \vee \mathbf{p}}{\neg p_2 \vee \mathbf{p}} \qquad p_2}{\bot \vee \mathbf{p}} \qquad \dfrac{\neg p_2 \vee \neg \mathbf{p} \qquad p_2}{\bot \vee \neg \mathbf{p}}$$

An example of the proof of completeness of resolution

Let $F = \{p_1 \vee p, \quad p_2, \quad \neg p_1 \vee \neg p_2 \vee p, \quad \neg p_2 \vee \neg p\}$

- Step 1: $F_p = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$ $\quad F_{np} = \{\neg p_2 \vee \neg p\}$ $\quad F_o = \{p_2\}$
  $F'_p = \{p_1, \neg p_1 \vee \neg p_2\}$ $\quad F'_{np} = \{\neg p_2\}$
- Step 2: $G = G_1 \vee G_2 = (F'_p \wedge F_o) \vee (F'_{np} \wedge F_o) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$
- Step 3: Proofs from G_1 to $\perp$ and G_2 to $\perp$
- Step 4: Convert to proof from F to $\perp$ by adding p and $\neg p$ (Note that we are in Case 2)

$$\dfrac{\dfrac{\dfrac{p_1 \vee \mathbf{p} \quad \neg p_1 \vee \neg p_2 \vee \mathbf{p}}{\neg p_2 \vee \mathbf{p}} \qquad p_2}{\perp \vee \mathbf{p}} \qquad \dfrac{\dfrac{\neg p_2 \vee \neg \mathbf{p} \qquad p_2}{\perp \vee \neg \mathbf{p}}}{\perp}$$

Exercises

Exercise 9.3

As in the previous slide, work out the proof of completeness for another set of unsatisfiable clauses namely $F_1 = \{\neg p_1 \vee \neg p_2 \vee \neg p_4, \neg p_2 \vee p_4, \neg p_1 \vee p_2, p_1\}$.

Exercise 9.4

Let F be an unsatisfiable CNF formula with n propositions. Show that there is a resolution proof of $\perp$ from F of size at most $2^{n+1} - 1$ (size = number of clause-occurrences/nodes in the derivation tree).

Tips and Tricks in resolution

Unit Clauses

- Clauses where there is only one literal.
  e.g, $q, \neg p$ are unit clauses in $F_1 = (p \vee r \vee \neg q) \wedge (\neg p) \wedge (\neg q \vee \neg r) \wedge q$.

Tips and Tricks in resolution

Unit Clauses

- Clauses where there is only one literal.
  e.g, $q, \neg p$ are unit clauses in $F_1 = (p \vee r \vee \neg q) \wedge (\neg p) \wedge (\neg q \vee \neg r) \wedge q$.
  - Obs1: If literal ℓ is a unit clause in F, then F is sat iff $F[\ell \mapsto 1]$ is sat.

Tips and Tricks in resolution

Unit Clauses

- Clauses where there is only one literal.
  e.g, $q, \neg p$ are unit clauses in $F_1 = (p \vee r \vee \neg q) \wedge (\neg p) \wedge (\neg q \vee \neg r) \wedge q$.
  - Obs1: If literal ℓ is a unit clause in F, then F is sat iff $F[\ell \mapsto 1]$ is sat.
  - e.g., F_1 is sat iff $F_1[p \mapsto 0, q \mapsto 1]$ is sat.

Tips and Tricks in resolution

Unit Clauses

- Clauses where there is only one literal.

Pure literals

- Literals whose negation does not occur in the formula, i.e., the corresponding variable occurs throughout only "positively" or "negatively" but not both.
  e.g., p and $\neg q$ are pure in $F_2 = (p \vee r \vee \neg q) \wedge (p \vee \neg r) \wedge (\neg q \vee \neg r) \wedge (\neg q \vee r)$ (q occurs only negatively) but r is not.

Tips and Tricks in resolution

Unit Clauses

- Clauses where there is only one literal.

Pure literals

- Literals whose negation does not occur in the formula, i.e., the corresponding variable occurs throughout only "positively" or "negatively" but not both.
  e.g., p and $\neg q$ are pure in $F_2 = (p \vee r \vee \neg q) \wedge (p \vee \neg r) \wedge (\neg q \vee \neg r) \wedge (\neg q \vee r)$ (q occurs only negatively) but r is not.
  - If ℓ is a pure literal, then F is sat iff $F[\ell \mapsto 1]$ is sat.
  - e.g., F_2 is sat iff $F_2[p \mapsto 1, q \mapsto 0]$ is sat.

Tips and Tricks in resolution

Unit Clauses

- Clauses where there is only one literal.

Pure literals

- Literals whose negation does not occur in the formula.

Exercise 9.5 Consider $F_3 = (p_1 \vee p_2) \wedge (p_3 \vee p_2 \vee \neg p_1) \wedge p_3 \wedge (\neg p_4 \vee p_2)$

1. What are the unit clauses in F_3, if any?
2. What are the pure literals in F_3, if any?

Topic 9.1

DPLL

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if **AP** $= \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false. Otherwise, F is unassigned.

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false. Otherwise, F is unassigned.
  - **Exercise 9.6** Under π_1, what is the status of the following formulas?
    - $(p_3 \vee \neg p_1) \wedge (p_1 \vee \neg p_2)$

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false. Otherwise, F is unassigned.
  - **Exercise 9.6** Under π_1, what is the status of the following formulas?
    - $(p_3 \vee \neg p_1) \wedge (p_1 \vee \neg p_2)$ is false.
    - $(p_1 \vee p_2 \vee p_3) \wedge \neg p_1$

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false. Otherwise, F is unassigned.
  - **Exercise 9.6** Under π_1, what is the status of the following formulas?
    - $(p_3 \vee \neg p_1) \wedge (p_1 \vee \neg p_2)$ is false.
    - $(p_1 \vee p_2 \vee p_3) \wedge \neg p_1$ is true.
    - $p_1 \vee p_3$

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false. Otherwise, F is unassigned.
  - **Exercise 9.6** Under π_1, what is the status of the following formulas?
    - $(p_3 \vee \neg p_1) \wedge (p_1 \vee \neg p_2)$ is false.
    - $(p_1 \vee p_2 \vee p_3) \wedge \neg p_1$ is true.
    - $p_1 \vee p_3$ is unassigned.
    - $\emptyset$ (empty formula)

Partial Assignments

Another Idea: Can we build satisfying assignments incrementally?

- A partial assignment is partial map from **AP** to $\{0, 1\}$.
  e.g, if $\mathbf{AP} = \{p_1, p_2, p_3\}$, $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$ is a partial assignment.
- Under partial assignment π,
  - literal ℓ is true if $\pi(\ell) = 1$ and ℓ is false if $\pi(\ell) = 0$. Otherwise, ℓ is unassigned.
    - e.g., under π_1, the states of literals $p_1, p_2, p_3, \neg p_1$ are 0, 1, unassigned, 1, respectively.
  - clause C is true if there is $\ell \in C$ such that ℓ is true and C is false if for each $\ell \in C$, ℓ is false.Otherwise, C is unassigned.
    - e.g., under π_1, $p_1 \vee p_2 \vee p_3$ is true while $p_1 \vee \neg p_2$ is false and $p_1 \vee p_3$ is unassigned.
  - CNF formula F is true if for each $C \in F$, C is true and F is false if there is $C \in F$ such that C is false. Otherwise, F is unassigned.
  - **Exercise 9.6** Under π_1, what is the status of the following formulas?
    - $(p_3 \vee \neg p_1) \wedge (p_1 \vee \neg p_2)$ is false.
    - $(p_1 \vee p_2 \vee p_3) \wedge \neg p_1$ is true.
    - $p_1 \vee p_3$ is unassigned.
    - $\emptyset$ (empty formula) is true.

Lifting definition of unit clauses to partial assignments

Definition (Unit clause)

C is a unit clause under π if a literal $\ell \in C$ is unassigned and the rest are false. Also, ℓ is called unit literal.

Lifting definition of unit clauses to partial assignments

Definition (Unit clause)

C is a unit clause under π if a literal $\ell \in C$ is unassigned and the rest are false. Also, ℓ is called unit literal.

Exercise 9.7

Consider again, partial assignment $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$. Which of the following are unit clauses under π_1? Identify the unit literals.

- $p_1 \vee \neg p_3 \vee \neg p_2$

Lifting definition of unit clauses to partial assignments

Definition (Unit clause)

C is a unit clause under π if a literal $\ell \in C$ is unassigned and the rest are false. Also, ℓ is called unit literal.

Exercise 9.7

Consider again, partial assignment $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$. Which of the following are unit clauses under π_1? Identify the unit literals.

- $p_1 \vee \neg p_3 \vee \neg p_2$ ✓
- $p_1 \vee \neg p_3 \vee p_2$

Lifting definition of unit clauses to partial assignments

Definition (Unit clause)

C is a unit clause under π if a literal $\ell \in C$ is unassigned and the rest are false. Also, ℓ is called unit literal.

Exercise 9.7

Consider again, partial assignment $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$. Which of the following are unit clauses under π_1? Identify the unit literals.

- $p_1 \vee \neg p_3 \vee \neg p_2$ ✓
- $p_1 \vee \neg p_3 \vee p_2$ ✗
- $p_1 \vee \neg p_3 \vee p_4$

Lifting definition of unit clauses to partial assignments

Definition (Unit clause)

C is a unit clause under π if a literal $\ell \in C$ is unassigned and the rest are false. Also, ℓ is called unit literal.

Exercise 9.7

Consider again, partial assignment $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$. Which of the following are unit clauses under π_1? Identify the unit literals.

- $p_1 \vee \neg p_3 \vee \neg p_2$ ✓
- $p_1 \vee \neg p_3 \vee p_2$ ✗
- $p_1 \vee \neg p_3 \vee p_4$ ✗
- $p_1 \vee \neg p_2$

Lifting definition of unit clauses to partial assignments

Definition (Unit clause)

C is a unit clause under π if a literal $\ell \in C$ is unassigned and the rest are false. Also, ℓ is called unit literal.

Exercise 9.7

Consider again, partial assignment $\pi_1 = \{p_1 \mapsto 0, p_2 \mapsto 1\}$. Which of the following are unit clauses under π_1? Identify the unit literals.

- $p_1 \vee \neg p_3 \vee \neg p_2$ ✓
- $p_1 \vee \neg p_3 \vee p_2$ ✗
- $p_1 \vee \neg p_3 \vee p_4$ ✗
- $p_1 \vee \neg p_2$ ✗

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input
- maintains a partial assignment, initially $\emptyset$

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input
- maintains a partial assignment, initially $\emptyset$
- assigns unassigned variables 0 or 1 randomly one after another

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input
- maintains a partial assignment, initially $\emptyset$
- assigns unassigned variables 0 or 1 randomly one after another
- sometimes forced to choose assignments due to unit/pure literals(why?)

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input
- maintains a partial assignment, initially $\emptyset$
- assigns unassigned variables 0 or 1 randomly one after another
- sometimes forced to choose assignments due to unit/pure literals(why?)

Martin Davis

Hilary Putnam

George Logemann

Donald Loveland

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input
- maintains a partial assignment, initially $\emptyset$
- assigns unassigned variables 0 or 1 randomly one after another
- sometimes forced to choose assignments due to unit/pure literals(why?)

Martin Davis

Hilary Putnam

George Logemann

Donald Loveland

A bit of history: Davis & Putnam's original 1960 algorithm eliminated variables one at a time using resolution – exactly the $F_p/F_{np}/F_o$ split from our completeness proof! It then resolved every clause in F_p against every clause in F_{np} on pivot p, replacing F with F_o plus all these resolvents.

DPLL (Davis-Putnam-Loveland-Logemann) method

- takes CNF input
- maintains a partial assignment, initially $\emptyset$
- assigns unassigned variables 0 or 1 randomly one after another
- sometimes forced to choose assignments due to unit/pure literals(why?)

Martin Davis

Hilary Putnam

George Logemann

Donald Loveland

A bit of history: Davis & Putnam's original 1960 algorithm eliminated variables one at a time using resolution – exactly the $F_p/F_{np}/F_o$ split from our completeness proof! It then resolved every clause in F_p against every clause in F_{np} on pivot p, replacing F with F_o plus all these resolvents. Davis, Logemann & Loveland's 1962 fix replaced that (expensive) elimination with the backtracking search as we shall see next.

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else
  return $DPLL(F, \pi[p \mapsto 1 - b])$

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat
if *F is true under π (i.e., all clauses of F are true)* **then return** *sat* ;
if *F is false under π (some clause of F is false, i.e., conflict)* **then return** *unsat* ;

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else
  return $DPLL(F, \pi[p \mapsto 1 - b])$

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat
if *F is true under π (i.e., all clauses of F are true)* **then return** *sat* ;
if *F is false under π (some clause of F is false, i.e., conflict)* **then return** *unsat* ;
if *$\exists$ unit/pure literal ℓ under π* **then**
  return $DPLL(F, \pi[\ell \mapsto 1])$

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else
  return $DPLL(F, \pi[p \mapsto 1 - b])$

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat
if *F is true under π (i.e., all clauses of F are true)* **then return** *sat* ;
if *F is false under π (some clause of F is false, i.e., conflict)* **then return**
if *$\exists$ unit/pure literal ℓ under π* **then**
  return $DPLL(F, \pi[\ell \mapsto 1])$

Backtracking at conflict

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else
  return $DPLL(F, \pi[p \mapsto 1 - b])$

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat
if *F is true under π (i.e., all clauses of F are true)* **then return** *sat* ;
if *F is false under π (some clause of F is false, i.e., conflict)* **then return**
if *$\exists$ unit/pure literal ℓ under π* **then**
  return $DPLL(F, \pi[\ell \mapsto 1])$

Backtracking at conflict

Decision

Choose an unassigned variable p and a random bit $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else
  return $DPLL(F, \pi[p \mapsto 1 - b])$

DPLL

Algorithm 9.1: DPLL(F)

Input: CNF F **Output:** sat/unsat
return $DPLL(F, \emptyset)$

Algorithm 9.2: DPLL(F,π)

Input: CNF F, partial assignment π **Output:** sat/unsat
if *F is true under π (i.e., all clauses of F are true)* **then return** *sat* ;
if *F is false under π (some clause of F is false, i.e., conflict)* **then return**
if *$\exists$ unit/pure literal ℓ under π* **then**
  return $DPLL(F, \pi[\ell \mapsto 1])$
Choose an unassigned variable p and $b \in \{0, 1\}$;
if $DPLL(F, \pi[p \mapsto b]) == sat$ **then**
  return *sat*
else
  return $DPLL(F, \pi[p \mapsto 1 - b])$

Backtracking at conflict

Unit propagation

Decision

Three actions of DPLL

A DPLL run consists of three types of actions

- Decision
- Unit propagation
- Backtracking

Example: decide, propagate, and backtrack in DPLL

$$
\begin{aligned}
c_1 &= (\neg p_1 \vee p_2) \\
c_2 &= (\neg p_1 \vee p_3 \vee p_5) \\
c_3 &= (\neg p_2 \vee p_4) \\
c_4 &= (\neg p_3 \vee \neg p_4) \\
c_5 &= (p_1 \vee p_5 \vee \neg p_2) \\
c_6 &= (p_2 \vee p_3) \\
c_7 &= (p_2 \vee \neg p_3 \vee p_7) \\
c_8 &= (\neg p_5) \\
c_9 &= (\neg p_7)
\end{aligned}
$$

Blue : causing unit propagation
Green/Blue : true clause

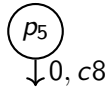
Example: decide, propagate, and backtrack in DPLL

$$
\begin{aligned}
c_1 &= (\neg p_1 \vee p_2) \\
c_2 &= (\neg p_1 \vee p_3 \vee p_5) \\
c_3 &= (\neg p_2 \vee p_4) \\
c_4 &= (\neg p_3 \vee \neg p_4) \\
c_5 &= (p_1 \vee p_5 \vee \neg p_2) \\
c_6 &= (p_2 \vee p_3) \\
c_7 &= (p_2 \vee \neg p_3 \vee p_7) \\
c_8 &= (\neg p_5) \\
c_9 &= (\neg p_7)
\end{aligned}
$$

Blue : causing unit propagation
Green/Blue : true clause

Example: decide, propagate, and backtrack in DPLL

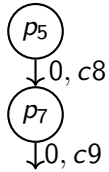
$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$



Blue : causing unit propagation
Green/Blue : true clause

Example: decide, propagate, and backtrack in DPLL

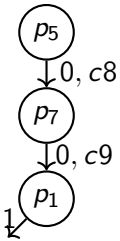
$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$



Blue : causing unit propagation
Green/Blue : true clause

Example: decide, propagate, and backtrack in DPLL

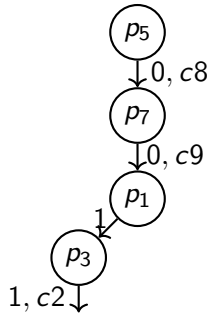
$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$



Blue : causing unit propagation
Green/Blue : true clause

Example: decide, propagate, and backtrack in DPLL

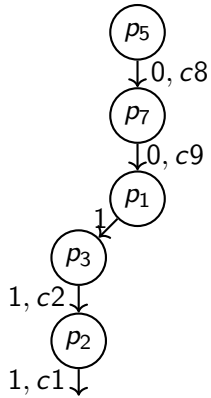
$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$



Blue : causing unit propagation
Green/Blue : true clause

Example: decide, propagate, and backtrack in DPLL

$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$

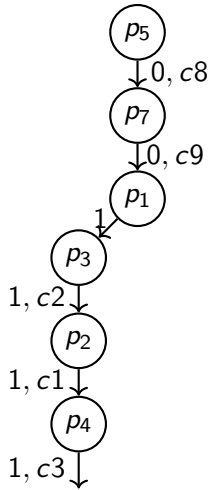


Blue : causing unit propagation
Green/Blue : true clause

Example: decide, propagate, and backtrack in DPLL

$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$

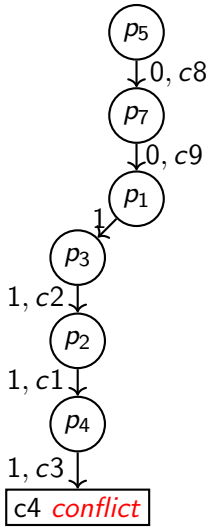
Blue : causing unit propagation
Green/Blue : true clause



Example: decide, propagate, and backtrack in DPLL

$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$

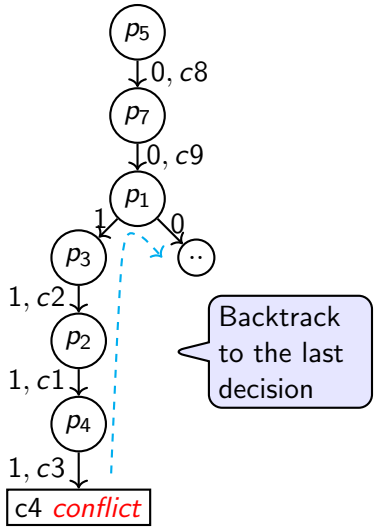
Blue : causing unit propagation
Green/Blue : true clause



Example: decide, propagate, and backtrack in DPLL

$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$

Blue : causing unit propagation
Green/Blue : true clause



Example: decide, propagate, and backtrack in DPLL

$c_1 = (\neg p_1 \vee p_2)$

$c_2 = (\neg p_1 \vee p_3 \vee p_5)$

$c_3 = (\neg p_2 \vee p_4)$

$c_4 = (\neg p_3 \vee \neg p_4)$

$c_5 = (p_1 \vee p_5 \vee \neg p_2)$

$c_6 = (p_2 \vee p_3)$

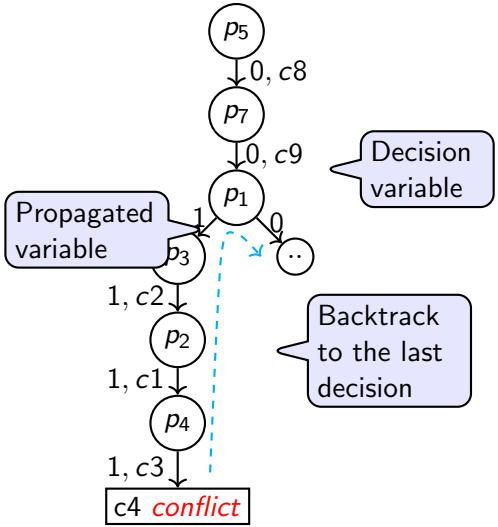
$c_7 = (p_2 \vee \neg p_3 \vee p_7)$

$c_8 = (\neg p_5)$

$c_9 = (\neg p_7)$

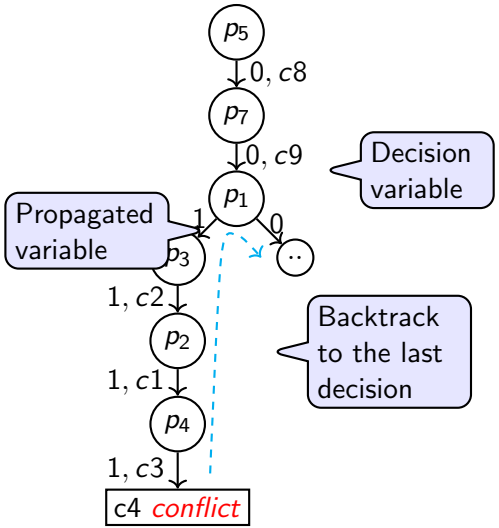
Blue : causing unit propagation

Green/Blue : true clause



Example: decide, propagate, and backtrack in DPLL

$$c_1 = (\neg p_1 \vee p_2)$$
$$c_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$c_3 = (\neg p_2 \vee p_4)$$
$$c_4 = (\neg p_3 \vee \neg p_4)$$
$$c_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$c_6 = (p_2 \vee p_3)$$
$$c_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$c_8 = (\neg p_5)$$
$$c_9 = (\neg p_7)$$



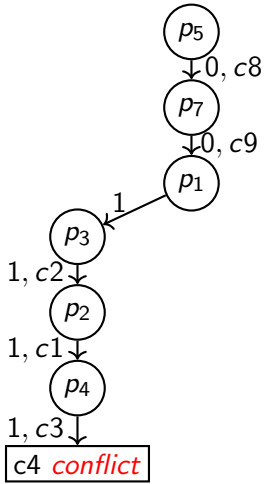
Blue : causing unit propagation

Green/Blue : true clause

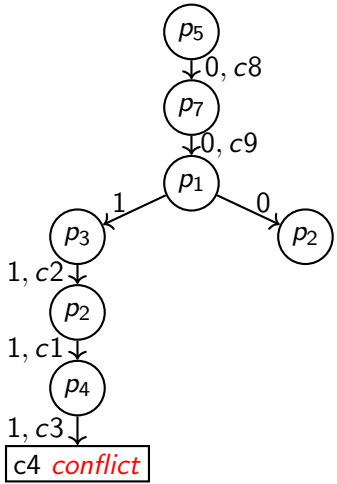
Exercise 9.8

Complete the DPLL run

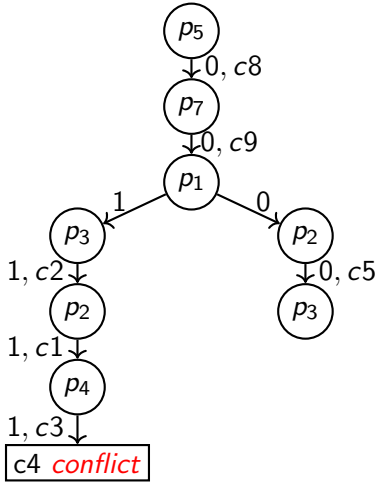
Solution: Exercise 9.8 – the completed DPLL run



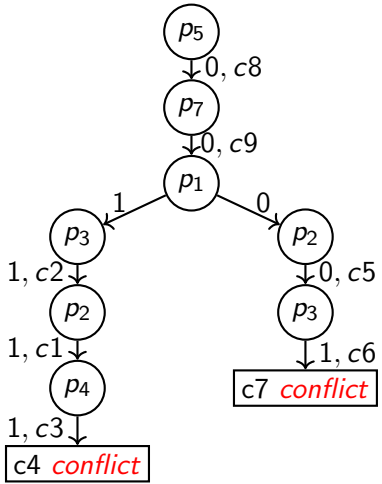
Solution: Exercise 9.8 – the completed DPLL run



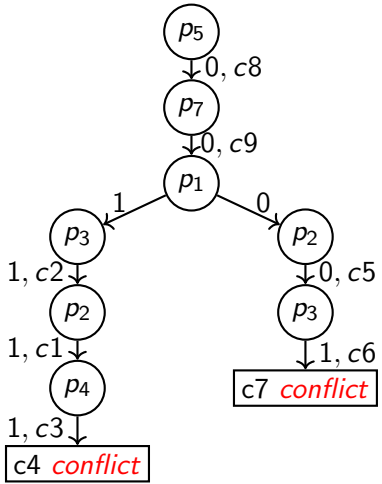
Solution: Exercise 9.8 – the completed DPLL run



Solution: Exercise 9.8 – the completed DPLL run

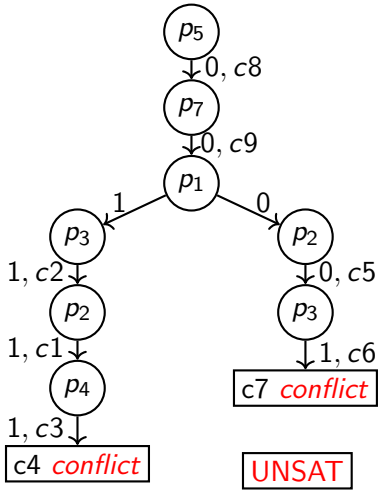


Solution: Exercise 9.8 – the completed DPLL run



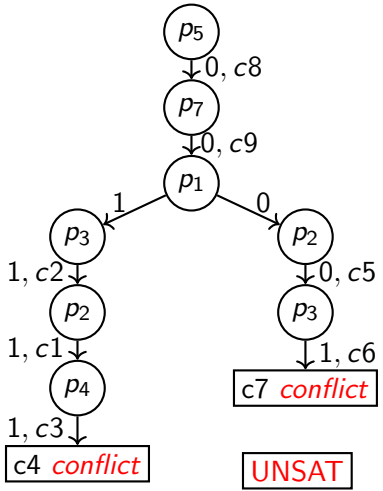
p_1 was the only decision – both values fail

Solution: Exercise 9.8 – the completed DPLL run



p_1 was the only decision – both values fail

Solution: Exercise 9.8 – the completed DPLL run



p_1 was the only decision – both values fail

Exercise 9.9 Now that you know $c_1, \ldots, c_9$ is UNSAT, derive a resolution proof of $\bot$ from these clauses.