Propositional Resolution

Second Part: Algorithms

Stéphane Devismes    Pascal Lafourcade    Michel Lévy
Course given by Ioana Ciuciu (ciuciu@imag.fr)

Université Joseph Fourier, Grenoble I

1st February 2013
Last course

- Resolution
Reminder

(1) $A \vdash B$

$B$ is deduced from $A$.

There is a proof by resolution of $B$ starting from $A$

(2) $A \models B$

$B$ is consequence of $A$.

Every model of $A$ is also a model of $B$

Coherence

(1) $\Rightarrow$ (2)

Completeness

(2) $\Rightarrow$ (1)

Resolution

Coherent and complete
Overview

Introduction

Complete strategy

The Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Conclusion
Homework: solution

- (H1): \( p \Rightarrow \neg j \equiv \neg p \lor \neg j \)
- (H2): \( \neg p \Rightarrow j \equiv p \lor j \)
- (H3): \( j \Rightarrow m \equiv \neg j \lor m \)
- (\neg C): \( \neg m \land \neg p \)

Clauses: \( \{\neg p \lor \neg j, p \lor j, \neg j \lor m, \neg m, \neg p\} \)

\[
\begin{array}{cccc}
p \lor j & \neg j \lor m & \neg m \\
p \lor m & p & \neg p & \neg p
\end{array}
\]
Presentation of the two algorithms

How to « systematically » decide whether $\Gamma$ is contradictory?

- **Complete strategy**
  Construction of ALL the deductible clauses (resolvents) from $\Gamma$.

- **The Davis-Putnam-Logemann-Loveland Algorithm**
  « Intelligent » traversal of the possible assignments of $\Gamma$

**Remark**

*Exponential* solutions in time in the worst case.
Consider a finite set of resolvents

**Problem**: the construction of all the clauses deductible from $\Gamma$ can never terminate.

Since two clauses have an infinite number of resolvents.

Example The resolvents of the clauses $p + \overline{q}$ and $r + q$ are:

- $p + r$
- $r + p$
- $p + r + p$
- $p + r + p$
- $p + \ldots + p + r + \ldots r$
Solution

Consider that two clauses having the same set of literals are equal.

If the length of $s(\Gamma) = n$, then we have at most $2^n$ clauses deduced from $\Gamma$. 
Reducibility of a set of clauses

In order to accelerate the algorithm, we reduce the set of clauses.

How to proceed with the reducibility?

Removing the valid clauses and the clauses containing another clause of the set.

A set of clauses is reduced if it is not reducible anymore.
Reduced set of clauses

Definition 2.1.26

A set of clauses is reduced if it does not contain any valid clause and none of the clauses is included in another clause.

Example 2.1.27

The reducibility of the set of clauses \( \{p + q + \overline{p}, p + r, p + r + \overline{s}, r + q\} \) gives the reduced set:

\[ \{p + r, r + q\} \].
Justification

Property 2.1.28

A set of clauses $E$ is equivalent to the reduced set of clauses obtained starting from $E$.

Proof.

- Removing a valid clause therefore equaling 1: $x.1 = x$
- Removing a clause including another clause: $x.(x + y) = x$
Result of the algorithm: minimum deduction clauses

**Definition 2.1.29**

Let \( \Gamma \) a set of clauses. A minimum clause for the deduction of \( \Gamma \) is a non-valid clause deduced from \( \Gamma \) and *strictly* not containing any clause deduced from \( \Gamma \).

**Example 2.1.30**

Let us consider the set of clauses \( \Gamma = \{a + \overline{b}, b + c + d\} \) the clause \( a + c + d \) is a minimum deduction clause.

On the contrary, if we add the clause \( \overline{a} + c \) to \( \Gamma \) then \( a + c + d \) is not a minimum clause since we can deduct \( c + d \) which is included in the clause \( a + c + d \).
Property 2.1.31

Let $\Theta$ the set of minimum deduction clauses for the set of clauses $\Gamma$. The set $\Gamma$ is unsatisfiable if and only if $\bot \in \Theta$.

Proof.

- Suppose $\bot \in \Theta$, then $\Gamma \vdash \bot$, hence by resolution coherence, $\Gamma$ is unsatisfiable.

- Suppose $\Gamma$ unsatisfiable, by resolution completeness, $\Gamma \vdash \bot$. Consequently $\bot$ is minimum clause for the deduction of $\Gamma$, therefore $\bot \in \Theta$. 

□
Interpretation

When the following algorithm terminates:

\[ \bot \in \Theta_k : \Gamma \text{ is unsatisfiable} \]

\[ \bot \notin \Theta_k : \Gamma \text{ is satisfiable, but what does } \Theta_k \text{ represent?} \]
$\Theta_k = \text{minimum clauses for the consequence}$

**Definition 2.1.32**

Let $\Gamma$ a set of clauses. A **minimum clause for the consequence** of $\Gamma$ is a non valid consequence clause of $\Gamma$ *strictly* not containing any consequence clause of $\Gamma$.

**Theorem 2.1.35**

Let $\Gamma$ a set of clauses. A clause is minimum for the deduction of $\Gamma$ if and only if it is minimum for the consequence of $\Gamma$.

Proofs given in the course support.
Example

Example 2.1.33

Consider the set of clauses $\Gamma = \{a + d, \overline{a} + b, \overline{b} + c\}$. The clause $d + c$ is minimum for the consequence of $\Gamma$.

Consequence: $d + c$ is a consequence of $\Gamma$ since in all model of $\Gamma$, either $d$ is true or $c$ is true.

Minimality: There exist models of $\Gamma$ which are not models of $d$ (respectively $c$): $a = 1$, $d = 0$, $c = 1$ and $b = 1$ (respectively $a = 0$, $d = 1$, $c = 0$ and $b = 0$).
Principle of the algorithm: Construct all the clauses deduced from $\Gamma$

Following the height of the proof trees.

Algorithm

For any integer $i$

While it is possible to construct new clauses

Construct the reduced set of all the clauses having a proof tree of height at most $i$.

In practice:

Maintain two sequences of the sets of clauses, $\Delta_{i(i\geq 0)}$ and $\Theta_{i(i\geq 0)}$.
Two sequences of sets of clauses

\[ \Delta_{i(i \geq 0)} \]

Clauses deduced from \( \Gamma \) by a proof of height \( i \), after clauses removal:

- valid clauses
- clauses including another clause of the proof of height at most \( i \).

\( \Delta_0 \) is obtained by reducing \( \Gamma \)
Two sequences of sets of clauses

\[ \Theta_i(i \geq 0) \]

Clauses deduced from \( \Gamma \) by a proof of height inferior to \( i \) after clauses removal:
- valid clauses
- clauses including another clause of the proof of height at most \( i \).

\( \Theta_0 \) is the empty set.
Details of the method

If $\Delta_k = \emptyset$, stop the construction:

- $k - 1$ is then the maximum height of a proof
- $\Theta_k$ is the reduced set of the clauses deduced from $\Gamma$
Construction of the sequences $\Delta_{i(i\geq 0)}$ and $\Theta_{i(i\geq 0)}$

$\Delta_{i+1}$

- Construct all the resolvents of $\Delta_i$ and $\Delta_i \cup \Theta_i$
- Reduce this set
- Remove the new resolvents including a clause of $\Delta_i \cup \Theta_i$

$\Theta_{i+1}$

Remove from $\Delta_i \cup \Theta_i$ the clauses which include one of the clauses of $\Delta_{i+1}$. 
### Example 2.2.1

Let $\Gamma = \{a + b + \bar{a}, a + b, a + b + c, a + \bar{b}, \bar{a} + b, \bar{a} + \bar{b}\}$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Delta_i$</th>
<th>$\Theta_i$</th>
<th>$\Delta_i \cup \Theta_i$</th>
<th>Resolvents of $\Delta_i$ and $\Delta_i \cup \Theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$a + b, a + \bar{b}$, $\bar{a} + b, \bar{a} + \bar{b}$</td>
<td>$\emptyset$</td>
<td>$a + b, a + \bar{b}$, $\bar{a} + b, \bar{a} + \bar{b}$</td>
<td>$b, b + \bar{b}, a, a + \bar{a}, \bar{a}, \bar{b}$</td>
</tr>
<tr>
<td>1</td>
<td>$a, b, b, \bar{a}$</td>
<td>$\emptyset$</td>
<td>$a, b, b, \bar{a}$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>0</td>
<td>$a + b, a + \bar{b}$, $\bar{a} + b, \bar{a} + \bar{b}$</td>
<td>$\emptyset$</td>
<td>$a + b, a + \bar{b}$, $\bar{a} + b, \bar{a} + \bar{b}$</td>
<td>$b, b + \bar{b}, a, a + \bar{a}, \bar{a}, \bar{b}$</td>
</tr>
<tr>
<td>1</td>
<td>$a, b, b, \bar{a}$</td>
<td>$\emptyset$</td>
<td>$a, b, b, \bar{a}$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>2</td>
<td>$\bot$</td>
<td>$\emptyset$</td>
<td>$\bot$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
Example 2.2.1 (contd.)

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Delta_i$</th>
<th>$\Theta_i$</th>
<th>$\Delta_i \cup \Theta_i$</th>
<th>Resolvents of $\Delta_i$ and $\Delta_i \cup \Theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$a + b, a + \tilde{b}, \tilde{a} + b, \tilde{a} + \tilde{b}$</td>
<td>0</td>
<td>$a + b, a + \tilde{b}, \tilde{a} + b, \tilde{a} + \tilde{b}$</td>
<td>$b, b + \tilde{b}, a, a + \tilde{a}, \tilde{a}, \tilde{b}$</td>
</tr>
<tr>
<td>1</td>
<td>$a, b, \tilde{b}, \tilde{a}$</td>
<td>0</td>
<td>$a, b, \tilde{b}, \tilde{a}$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>2</td>
<td>$\bot$</td>
<td>0</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>3</td>
<td>$0$</td>
<td>$\bot$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 2.2.2

\{ a, c, \overline{a + b}, \overline{c + e} \}

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Delta_i$</th>
<th>$\Theta_i$</th>
<th>$\Delta_i \cup \Theta_i$</th>
<th>Resolvents of $\Delta_i$ and $\Delta_i \cup \Theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$a, c, \overline{a + b}, \overline{c + e}$</td>
<td>$\emptyset$</td>
<td>$a, c, \overline{a + b}, \overline{c + e}$</td>
<td>$e, \overline{b}$</td>
</tr>
<tr>
<td>1</td>
<td>$e, \overline{b}$</td>
<td>$a, c$</td>
<td>$a, \overline{b}, c, e$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>2</td>
<td>$\emptyset$</td>
<td>$a, \overline{b}, c, e$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
End of the algorithm: idea

There are at most $2^n$ clauses deduced from $\Gamma$.

$\Delta_{i(i \geq 0)}$ only contains clauses deduced from $\Gamma$

$\Delta_{i(i \geq 0)}$ are mutually disjoint (To demonstrate)

Hence there are at most $2^n + 1$ sets, therefore $k \leq 2^n + 1$
\( \Delta_i(i \geq 0) \) are mutually disjoint

Property 2.2.3

Let \( i \leq k \). All clause of \( \bigcup_{j \leq i} \Delta_j \) contains a clause of \( \Delta_i \cup \Theta_i \).

Proof.

By induction.

- For \( i = 0 \) the property is trivial since \( \Theta_0 = \emptyset \).
- Suppose the property true for \( i \), let us show that it is also true for \( i + 1 \). Let \( C \in \bigcup_{j \leq i+1} \Delta_j \). Let us show that \( C \) contains a clause of \( \Delta_{i+1} \cup \Theta_{i+1} \). We examine all the possible cases for \( C \).
  1. \( C \in \Delta_{i+1} \). Hence \( C \) contains a clause of \( \Delta_{i+1} \cup \Theta_{i+1} \).
  2. \( C \in \bigcup_{j \leq i} \Delta_j \). By induction hypothesis, \( C \) contains a clause \( D \in \Delta_i \cup \Theta_i \). We distinguish two situations for \( D \).
     2.1 \( D \in \Theta_{i+1} \). Hence \( C \) contains a clause of \( \Delta_{i+1} \cup \Theta_{i+1} \).
     2.2 \( D \notin \Theta_{i+1} \). By construction of \( \Theta_{i+1} \), since \( D \in \Delta_i \cup \Theta_i \) and \( D \notin \Theta_{i+1} \), it means that \( D \) contains a clause of \( \Delta_{i+1} \). Or \( C \) contains \( D \), hence \( C \) also contains a clause of \( \Delta_{i+1} \cup \Theta_{i+1} \).
Propositional Resolution

Complete strategy

$$\Delta_i(i \geq 0)$$ are mutually disjoint

Property 2.2.4

For all $$i \leq k$$, the sets $$\Delta_i$$ are mutually disjoint.

Proof.

We perform an induction on the sets $$\Delta_j$$ with $$0 \leq j \leq i$$ and $$i \leq k$$.

The base case (basis) : If $$i = 0$$, there is only one set, hence the property is verified.

Inductive step : Let $$i < k$$. Suppose that all the sets $$\Delta_j$$ where $$j \leq i$$ are mutually disjoint. Let us show that $$\Delta_{i+1}$$ is disjoint with respect to these sets. Let $$C \in \Delta_{i+1}$$. Suppose, on the contrary, that $$C \in \bigcup_{j \leq i} \Delta_j$$. According to the preceding property, $$C$$ includes a clause of $$\Delta_i \cup \Theta_i$$. Hence by construction of $$\Delta_{i+1}$$, $$C \not\in \Delta_{i+1}$$, contradiction. Consequently, $$C \not\in \bigcup_{j \leq i} \Delta_j$$.

Hence, END of the algorithm
Result of the algorithm

- $\Gamma$ and $\Theta_k$ are equivalent
- $\Theta_k = \text{set of minimum deduction clauses.}$
Γ and Θₖ are equivalent

Property 2.2.5

For all \( i < k \), the sets \( \Delta_i \cup \Theta_i \) and \( \Delta_{i+1} \cup \Theta_{i+1} \) are equivalent.

Proof.

1. All clause of \( \Delta_{i+1} \cup \Theta_{i+1} \) is consequence of \( \Delta_i \cup \Theta_i \). All clause of \( \Delta_{i+1} \cup \Theta_{i+1} \) is element of \( \Delta_i \cup \Theta_i \) or resolvent of two elements of this set, therefore it is consequence of this set.

2. All clause of \( \Delta_i \cup \Theta_i \) is consequence of \( \Delta_{i+1} \cup \Theta_{i+1} \). Let \( C \in \Delta_i \cup \Theta_i \). We distinguish two possible cases:
   2.1 \( C \in \Theta_{i+1} \), thus \( C \) is consequence of \( \Delta_{i+1} \cup \Theta_{i+1} \).
   2.2 \( C \notin \Theta_{i+1} \), thus \( C \) contains a clause of \( \Delta_{i+1} \) hence is consequence of \( \Delta_{i+1} \cup \Theta_{i+1} \).
Γ and Θₖ are equivalent

Property 2.2.6

The sets Γ and Θₖ are equivalent.

Proof.

- Δ₀ is the set obtained by reduction of Γ, according to property 2.1.28, these two sets are equivalent.
- Since Θ₀ is empty, Γ is equivalent to Δ₀ ∪ Θ₀.
- According to property 2.2.5 and by induction, Δ₀ ∪ Θ₀ is equivalent to the set of clauses Δₖ ∪ Θₖ.
- Since the algorithm terminates when Δₖ is the empty set, the sets Γ and Θₖ are equivalent.
**Θ_k** = set of minimum deduction clauses

Property 2.2.13

**Θ_k** is the set of minimum deduction clauses of **Γ**.

**Proof.**

Cf. Course support (Poly)

Example from 1.6.2: \( \text{maj}(x, y, z) = (x + y + z)(x + y + \overline{z})(x + \overline{y} + z)(\overline{x} + y + z) \).

<table>
<thead>
<tr>
<th>( \Delta_0 )</th>
<th>( \Theta_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( x + y + z )</td>
</tr>
<tr>
<td>2</td>
<td>( x + y + \overline{z} )</td>
</tr>
<tr>
<td>3</td>
<td>( x + \overline{y} + z )</td>
</tr>
<tr>
<td>4</td>
<td>( \overline{x} + y + z )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \Delta_1 )</th>
<th>( \Theta_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( x + y ) resolvent of 1, 2</td>
</tr>
<tr>
<td>6</td>
<td>( x + z ) resolvent of 1, 3</td>
</tr>
<tr>
<td>7</td>
<td>( y + z ) resolvent of 1, 4</td>
</tr>
</tbody>
</table>

\( \Delta_2 \) is empty and \( \Theta_2 = \Delta_1 \).

Consequently \( \text{maj}(x, y, z) = (x + y)(x + z)(y + z) \).
Historic

The Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

- Introduced by Martin Davis and Hilary Putnam in 1960, then refined by Martin Davis, George Logemann and Donald Loveland in 1962
- Indicates if a set of clauses is satisfiable.
- Basis for (most efficient) complete SAT-solvers such as **chaff**, **zchaff** and **satz**.
Principle I

Two types of formulae transformation:

1. **preserving the truth value**: transforming a formula into an equivalent formula
   - reduction

2. **preserving the satisfiability only**: transforming a satisfiable formula into another satisfiable formula
   - removal of clauses containing isolated literals
   - unit resolution

DPLL is efficient since it uses these two transformations.
Principle II

« Branching/Backtracking » (splitting rule)

- **Branching**: After simplification, assign to **true** a heuristically chosen variable (branching literal).
- Continue the algorithm recursively.
- **Backtracking**: If we arrive to a contradiction, we return to the last choice, and we « branch » by assigning **false** to the chosen variable.
Removal of clauses having isolated literals.

Definition 2.3.1 **Isolated** literal $L$

If none of the clauses of $\Gamma$ contains $L^c$.

Lemme 2.3.2

Removing clauses with an isolated literal preserves the satisfiability.

Proof : Proof is requested in exercise 48.
Example 2.3.3

Let $\Gamma$ the set of clauses

1. $p + q + r$
2. $\overline{q} + \overline{r}$
3. $q + s$
4. $\overline{s} + t$

Simplify $\Gamma$ by removing clauses having isolated literals.

The literals $p$ and $t$ are isolated. We therefore obtain

2. $\overline{q} + \overline{r}$
3. $q + s$

The literals $\overline{r}$ and $s$ are isolated. We obtain the empty set.

According to lemma 2.3.2, $\Gamma$ has a model. But there is a counter-model, e.g. $p = 0, q = 0, r = 0$!!!
Unit resolution

**Definition 2.3.4**

A *unit clause* is a clause which only contains one literal.

**Lemma 2.3.5**

Let $L$ the set of literals of the unit clauses of $\Gamma$. Let $\Theta$ the set of clauses obtained starting from $\Gamma$, as follows

- if $L$ contains two complementary literals, then $\Theta = \{\bot\}$.
- else $\Theta$ is obtained as follows
  - removing the clauses containing an element of $L$
  - in the remaining clauses, remove the complementary literals of the elements of $L$

$\Gamma$ has a model if and only if $\Theta$ has a model.

Proof : The proof is requested in exercise 49.
Example 2.3.6 Unit resolution

Simplify the following sets of clauses by unit resolution:

- Let $\Gamma$ the set of clauses: $p + q, \overline{p}, \overline{q}$
  - $\perp$ by unit resolution, hence $\Gamma$ has no model.

- Let $\Gamma$ the set of clauses: $a + b + \overline{d}, \overline{a} + c + \overline{d}, \overline{b}, d, \overline{c}$.
  1. $a, \overline{a}$.
  2. Empty clause.
  hence $\Gamma$ has no model.

- Let $\Gamma'$ the set of clauses: $p, q, p + r, \overline{p} + r, q + \overline{r}, \overline{q} + s$.
  - By unit resolution, we obtain: $r, s$.
  - This set of clauses has a model, hence $\Gamma'$ has a model.
Removal of valid clauses

Lemma 2.3.7

Let $\Theta$ the set of clauses obtained by removing the valid clauses of $\Gamma$. $\Gamma$ has a model iff $\Theta$ has a model.

Proof.

- Suppose that $\Gamma$ has a model $v$, since $\Theta$ is a subset of clauses of $\Gamma$, $v$ is also model of $\Theta$. Hence $\Theta$ has a model.

- Suppose that $\Theta$ has a model $v$. Let $v'$ a truth assignment of $\Gamma$ so that $v'(x) = v(x)$ for all variable $x$ belonging to both $\Gamma$ and $\Theta$. Let $C$ a clause of $\Gamma$. If $C$ is also a clause of $\Theta$, then $v'$ is a model of $C$ since $v$ and $v'$ give the same value to $C$. If $C$ is not a clause of $\Theta$, then $C$ is valid, consequently all truth assignment, $v'$ in particular, is model of $C$. Hence $\Gamma$ has a model : $v'$.
The DPLL Algorithm (figure 2.1)

**bool function** $\text{Algo\_DPLL}(\ \Gamma : \text{set of clauses})$

0 Remove the valid clauses of $\Gamma$.
   If $\Gamma = \emptyset$, return (true).
   Else return ($\text{DPLL}(\Gamma)$)

**bool function** $\text{DPLL}(\ \Gamma : \text{non-valid set of clauses})$

The function returns true if and only if $\Gamma$ is satisfiable

1 If $\bot \in \Gamma$, return (false).
   If $\Gamma = \emptyset$, return (true).
2 Reduce $\Gamma$: simply remove any clause containing another clause.
3 Remove from $\Gamma$ the clauses containing isolated literals (cf. paragraph 2.3.1).
   If the set $\Gamma$ has been modified, goto 1.
4 Apply to $\Gamma$ the unit resolution (cf paragraph 2.3.2).
   If the set $\Gamma$ has been modified, goto 1.
5 Select $x$, an arbitrary variable of $\Gamma$
   return ($\text{DPLL}(\Gamma[x := 0])$ or then $\text{DPLL}(\Gamma[x := 1])$)
Example 2.3.8

Let $\Gamma$ the set of clauses: $\overline{a} + \overline{b}, a + b, \overline{a} + \overline{c}, a + c, \overline{b} + \overline{c}, b + c$.

Since all leaves contain the empty clause, the set $\Gamma$ is unsatisfiable.
Example 2.3.8

Let $\Gamma$ the set of clauses: $\overline{p} + \overline{q}, \overline{p} + s, p + q, \overline{p} + \overline{s}$.

Since one leaf contains the empty clause, the set $\Gamma$ is satisfiable. It is useless to continue the construction of the right branch.
Theorem 2.3.9 et 2.3.10

The algorithm Algo\_DPLL is correct and terminates.

Termination proof

- Step 0 is only executed once.
- Iteration in 1 : the number of clauses strictly decreases, hence termination.
- Recursivity in 5 : the number of variables strictly decreases, hence termination.

Reminder of property 2.1.21 : $\Gamma$ has a model iff $\Gamma[x := 0]$ is satisfiable or $\Gamma[x := 1]$ is satisfiable.
Correctness proof

Invariant: the current value of \( \Gamma \) has a model iff \( \Gamma \) has a model.

Verified at step 0, 1 and 5, hence correct answers. Suppose the recursive calls are correct:

- if \( \text{DPLL}(\Gamma[x := 0]) \) is true, then by induction \( \Gamma[x := 0] \) is satisfiable, hence \( \Gamma \) is satisfiable, according to property 2.1.21. which corresponds to the true value of \( \text{DPLL}(\Gamma) \).

- if \( \text{DPLL}(\Gamma[x := 0]) \) is false, then by induction \( \Gamma[x := 0] \) is unsatisfiable. In this case, \( \text{DPLL}(\Gamma) \) equals \( \text{DPLL}(\Gamma[x := 1]) \):
  - Suppose that \( \text{DPLL}(\Gamma[x := 1]) \) is true, then by induction \( \Gamma[x := 1] \) is satisfiable, hence \( \Gamma \) is satisfiable, which corresponds to the true value of \( \text{DPLL}(\Gamma) \).
  - Suppose that \( \text{DPLL}(\Gamma[x := 1]) \) is false, then by induction \( \Gamma[x := 1] \) is unsatisfiable. Hence \( \Gamma \) is unsatisfiable, which corresponds to the false value of \( \text{DPLL}(\Gamma) \).
Remarks 2.3.11 and 2.3.12

- **Forgetting simplifications**: DPLL stays correct if we forget the reduction (2), the removal of the isolated literals (3) and/or the unit reduction (4).

- **Choice of the variable (branching literal)**:
  - A good choice for the variable $x$ from step (5), is to choose the variable that appears most often.
  - A better choice is to choose the variable which will lead to the most of simplifications

Cf. Sub-section 2.3.5, for the principal branching heuristics
Planning of the Semester

TODAY

- Propositional logic
- Propositional resolution *
- Propositional natural deduction
- First order logic

MIDTERM EXAM

- Basis for the automated proof
  (« first order resolution »)
- First order natural deduction

EXAM
Conclusion : Next course

- Natural deduction
Conclusion

Thank you for your attention.

Questions?