

IN3070/4070 – Logic – Autumn 2022

Lecture 10: DPLL

Martin Giese

25th October 2022



DEPARTMENT OF
INFORMATICS



UNIVERSITY OF
OSLO

Today's Plan

- Motivation
- Simplification Rules
- Atomic Cut
- The DPLL Algorithm
- Other Tricks

Motivation

Outline

- Motivation
- Simplification Rules
- Atomic Cut
- The DPLL Algorithm
- Other Tricks

Motivation

Smullyan's categories: $\alpha/\beta/\gamma/\delta$

- Many similar cases in proofs and implementations
- Categorise rules, rule applications, and formulae

α -rules

Propositional, one branch, e.g.

$$\frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta} \wedge\text{-left}$$

β -rules

Propositional, splitting, e.g.

$$\frac{\Gamma \Rightarrow A, \Delta \quad \Gamma, B \Rightarrow \Delta}{\Gamma, A \rightarrow B \Rightarrow \Delta}$$

γ -rules

Apply for all terms t , e.g.

$$\frac{\Gamma, A[x \setminus t], \forall x A \Rightarrow \Delta}{\Gamma, \forall x A \Rightarrow \Delta} \forall\text{-left}$$

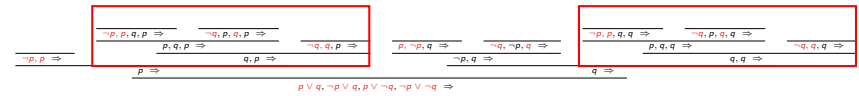
δ -rules

Introduce new constant c , e.g.

$$\frac{\Gamma, A[x \setminus c], \Rightarrow \Delta}{\Gamma, \exists x A \Rightarrow \Delta} \exists\text{-left}$$

A Naïve SAT Solver at Work

Problems



- Costly repetitions of identical proof trees
- 9 Branches
- Can often be avoided by using β rules in the “right” order
- But finding the best order is harder (!) than finding a proof
- Better: avoid using β (i.e. splitting) rules

Outline

- Motivation
- Simplification Rules
- Atomic Cut
- The DPLL Algorithm
- Other Tricks

Simplification Rules: Motivation

- Given two formulas p and $q \wedge (r \rightarrow s \wedge p)$
- And an interpretation $\mathcal{I}$ with $\mathcal{I} \models p$
- $v_{\mathcal{I}}(q \wedge (r \rightarrow s \wedge p)) = v_{\mathcal{I}}(q \wedge (r \rightarrow s \wedge \text{true})) = v_{\mathcal{I}}(q \wedge (r \rightarrow s))$
- An interpretation $\mathcal{I}$ falsifies a sequent

$$p, q \wedge (r \rightarrow s \wedge p), \Gamma \vdash \Delta$$

if and only if $\mathcal{I}$ falsifies the sequent

$$p, q \wedge (r \rightarrow s), \Gamma \vdash \Delta$$

Simplification

Definition 2.1 (Simplification).

Given two formulas A and B , where B does not have $\neg$ as top-symbol, the simplification of A with B , written $A[B]$, is the result of

- ▶ Replacing all occurrences of B in A by $true$, and
- ▶ Simplifying subformulae as long as possible using the rewritings

$$\begin{array}{ll} A \vee true \mapsto true & A \vee false \mapsto A \\ A \wedge true \mapsto A & A \wedge false \mapsto false \\ A \rightarrow true \mapsto true & A \rightarrow false \mapsto \neg A \\ true \rightarrow A \mapsto A & false \rightarrow A \mapsto true \\ \neg true \mapsto false & \neg false \mapsto true \end{array}$$

The simplification of A with $\neg B$, written $A[\neg B]$, is the result of

- ▶ Replacing all occurrences of B in A by $false$, and
- ▶ Applying the same rewritings.

Simplification Examples

- ▶ To compute $(q \wedge (r \rightarrow s \wedge p))[p]$
 - ▶ Do the replacement: $q \wedge (r \rightarrow s \wedge true)$
 - ▶ Then simplify $q \wedge (r \rightarrow s \wedge true) \mapsto q \wedge (r \rightarrow s)$
- ▶ So $(q \wedge (r \rightarrow s \wedge p))[p] = q \wedge (r \rightarrow s)$
- ▶ To compute $(q \wedge (r \rightarrow s \wedge p))[\neg p]$
 - ▶ Do the replacement: $q \wedge (r \rightarrow s \wedge false)$
 - ▶ Then simplify $q \wedge (r \rightarrow s \wedge false) \mapsto q \wedge (r \rightarrow false) \mapsto q \wedge \neg r$
- ▶ So $(q \wedge (r \rightarrow s \wedge p))[\neg p] = q \wedge \neg r$

Main property of Simplification

Lemma 2.1.

Given a formula A that contains a subformula B , and let $B \equiv B'$. Then A is logically equivalent to the result of replacing B by B' in A .

Proof.

Easily shown by structural induction over A . □

Theorem 2.1.

Given formulas A and B and an interpretation $\mathcal{I}$ with $\mathcal{I} \models B$. Then $\mathcal{I} \models A$ if and only if $\mathcal{I} \models A[B]$.

Proof.

For the first step (replacing B by $true$ or $false$), the proof is by structural induction on A . For the simplification steps, each formula is logically equivalent to the next, due to the preceding lemma. □

Simplification Rules

We add the following four “simplification rules” to LK:

$$\begin{array}{c} \frac{B, A[B], \Gamma \Rightarrow \Delta}{B, A, \Gamma \Rightarrow \Delta} \qquad \frac{A[\neg B], \Gamma \Rightarrow B, \Delta}{A, \Gamma \Rightarrow B, \Delta} \\[2ex] \frac{B, \Gamma \Rightarrow A[B], \Delta}{B, \Gamma \Rightarrow A, \Delta} \qquad \frac{\Gamma \Rightarrow B, A[\neg B], \Delta}{\Gamma \Rightarrow B, A, \Delta} \end{array}$$

Example: (one-sided) LK with Simplification Rules

$$\begin{array}{c}
 \frac{p, q, \text{true}, (\neg p \vee \neg q)[p] \Rightarrow}{p, q, (p \vee \neg q)[p], \neg p \vee \neg q \Rightarrow} \\
 \frac{p, (\neg p \vee q)[p], p \vee \neg q, \neg p \vee \neg q \Rightarrow}{p, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow} \\
 \frac{p, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}
 \end{array}$$

- Strategy: Apply simplification as much as possible, before β rules
- In this case: from 9 branches down to 2.

Simplification for Clauses

- Simplify a clause C with a literal L
- Case 1: C contains L , $C = A_1 \vee \dots \vee A_k \vee L$
 - Then $C[L] = \text{true}$
 - In refutation (left of sequent, resolution), true is useless and can be removed
 - Removing C because $L \in C$ is called **unit subsumption**
- Case 2: C contains $\bar{L}$, $C = A_1 \vee \dots \vee A_k \vee \bar{L}$
 - Then $C[L] = A_1 \vee \dots \vee A_k$
 - $C[L]$ is the resolvent of C and L !
 - Replacing C by $A_1 \vee \dots \vee A_k$ is called **unit resolution**
 - Note that C is subsumed by $A_1 \vee \dots \vee A_k$
- Unit subsumption and unit resolution together: **unit propagation**
- Given a literal L , every clause can be either removed completely, or shortened by removing $\bar{L}$, unit propagation can be used to remove L from every other clause containing L or $\bar{L}$.

Outline

- Motivation
- Simplification Rules
- Atomic Cut
- The DPLL Algorithm
- Other Tricks

Atomic Cut: Motivation

- For a sequent with n different β -formulas,
- each of them has to be expanded on every branch...
- ... which gives 2^n branches...
- even though there might be only $k < n$ propositional variables,
- and therefore only 2^k different interpretations!
- E.g. in the motivating example:
 - 9 branches for 4 interpretations for 2 prop. variables.
- Idea: max. 1 split per propositional variable

The Cut Rule

- ▶ The cut rule for LK:

$$\frac{\Gamma \Rightarrow A, \Delta \quad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

- ▶ The rule is sound (exercise) but not needed for completeness.
- ▶ It is a bit like proving a lemma A and then using it.
- ▶ Using cut can make proofs non-elementarily shorter (in first order logic)
- ▶ I.e. size $O(k)$ with cut but $O(\underbrace{2^{2^{\dots 2}}}_k)$ without.
- ▶ Not useful for automated proof search, because A has to be guessed.
- ▶ The essence of human theorem proving: introducing the right lemmas!

Atomic Cut

- ▶ The atomic cut rule is just the cut rule

$$\frac{\Gamma \Rightarrow A, \Delta \quad A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

- ▶ Where A is restricted to be an **atomic formula**.
- ▶ No nonelementary speedup :-)
- ▶ But we don't need more atomic cuts than we have prop. variables :-)
- ▶ We can replace β rules in LK by atomic cut. . .
- ▶ . . . if we add the simplification rules to deal with β formulas.
- ▶ Atomic cut for single sided calculi:

$$\frac{A, \Gamma \Rightarrow \quad \neg A, \Gamma \Rightarrow}{\Gamma \Rightarrow} \quad \frac{\Rightarrow A, \Delta \quad \Rightarrow \neg A, \Delta}{\Rightarrow \Delta}$$

Atomic Cut + Unit Propagation

$$\frac{\frac{\frac{q, p, \neg p \Rightarrow}{q, p, \neg p \vee \neg q \Rightarrow}{q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}{q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}{q, p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow} \quad \frac{\frac{\neg q, p, \neg p, p \vee \neg q, \neg p \vee \neg q \Rightarrow}{\neg q, p, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}{\neg q, p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}}{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q \Rightarrow}$$

Outline

- ▶ Motivation
- ▶ Simplification Rules
- ▶ Atomic Cut
- ▶ The DPLL Algorithm
- ▶ Other Tricks

The DPLL Algorithm

- ▶ DPLL stands for Davis-Putnam-Logemann-Loveland
- ▶ Introduced in 1962 by Martin Davis, George Logemann and Donald W. Loveland
- ▶ A refinement of an earlier algorithm, invented by Martin Davis and Hilary Putnam in (1960)
- ▶ Made propositional theorem proving (“SAT solving”) practically viable
- ▶ After almost 60 years, still the basis of most efficient SAT solvers
- ▶ DPLL works on a set of propositional clauses
- ▶ DPLL Consists of
 - ▶ Atomic Cut (with a heuristic for choosing the atom)
 - ▶ Unit Propagation
 - ▶ Pure Literal Elimination (exercise!)

Outline

- ▶ Motivation
- ▶ Simplification Rules
- ▶ Atomic Cut
- ▶ The DPLL Algorithm
- ▶ Other Tricks

Lemma Generation

- ▶ Remember the exercise sheet 2?

$$\frac{\Gamma \Rightarrow A, \Delta \quad \Gamma, A \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta} \wedge\text{-lg}$$

- ▶ Closing the left branch, we “learnt the Lemma A”
- ▶ With single-sided sequents:

$$\frac{A, \Gamma \Rightarrow \quad \neg A, B, \Gamma \Rightarrow}{A \vee B, \Gamma \Rightarrow} \vee\text{-lg}$$

- ▶ We refuted A, so now we may assume $\neg A$.
- ▶ Whenever we close a branch, we learn that a certain combination of literals $L_1, \dots, L_k$ leads to a contradiction
- ▶ We can add a clause $\overline{L_1} \vee \dots \vee \overline{L_k}$ to capture this.
- ▶ “Clause Learning”

Pruning

- ▶ Pruning $\equiv$ Backjumping $\equiv$ Intelligent Backtracking $\equiv$ Non-chronological Backtracking

- ▶ Consider the following derivation

$$\frac{\frac{\text{needed B and G}}{p, q, \neg p, \neg r \Rightarrow} \quad \frac{\text{needed G}}{p, q, r, \neg r \Rightarrow} \quad \frac{p, q, \neg p \vee r, \neg r \Rightarrow}{p, q \vee s, \neg p \vee r, \neg r \Rightarrow} \quad \frac{p, s, \dots \Rightarrow}{p \vee q, q \vee s, \neg p \vee r, \neg r \Rightarrow} \quad \frac{q, \dots \Rightarrow}{p \vee q, q \vee s, \neg p \vee r, \neg r \Rightarrow} \quad \text{B}$$

- ▶ No formulae introduced by R needed to close the two left branches
- ▶ Could have closed the branch without applying R
- ▶ Pruning: after closing the left two branches, continue with

$$\frac{\frac{\text{needed B and G}}{p, \neg p, \neg r \Rightarrow} \quad \frac{\text{needed G}}{p, r, \neg r \Rightarrow} \quad \frac{p, \neg p \vee r, \neg r \Rightarrow}{p \vee q, q \vee s, \neg p \vee r, \neg r \Rightarrow} \quad \frac{q, \dots \Rightarrow}{p \vee q, q \vee s, \neg p \vee r, \neg r \Rightarrow} \quad \text{B}$$

Conflict-driven clause learning (CDCL)

- ▶ The modern take on DPLL
- ▶ See e.g. the successful MiniSat implementation <http://minisat.se/>
- ▶ A combination of
 - ▶ Atomic cut
 - ▶ Unit propagation
 - ▶ Clause learning
 - ▶ Pruning

Stålmarck's Method

- ▶ Devised by Gunnar Stålmarck, applied for patent 1989
- ▶ The Dilemma Rule:

$$\frac{\frac{\Gamma_1 \cap \Gamma_2 \Rightarrow}{\Gamma_1 \Rightarrow \quad \Gamma_2 \Rightarrow} \quad \begin{array}{c} \vdots \\ A, \Gamma \Rightarrow \end{array} \quad \begin{array}{c} \vdots \\ \neg A, \Gamma \Rightarrow \end{array}}{\Gamma \Rightarrow}$$

- ▶ After unit propagation, join branches generated by cut
- ▶ Stålmarck's discovery: often enough to consider max two branches
- ▶ Not always. Why?
- ▶ In general: nesting of Dilemma Rule.
- ▶ Still: deep nesting rarely needed.

Summary

- ▶ Efficient theorem provers *combine* formulas instead of just decomposing
 - ▶ The resolution rule is an example
 - ▶ The simplification rules are another
- ▶ For propositional logic, unit propagation is very effective
- ▶ Atomic cut and unit propagation are the main ingredients of DPLL
- ▶ DPLL has been refined to CDCL
- ▶ CDCL incorporates clause learning and pruning