SAT@Mandi 2019

Lecture 1: Introduction and background

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2019-03-26

Topic 1.1

What is automated reasoning?

Automated reasoning (logic)

We will use reasoning and logic synonymously.

- Have you ever said to someone "be reasonable"?
  - whatever your intuition was that is reasoning
- Why we care?

  Logic is the **calculus** of computer science

Example: applying logic

Logic is about inferring conclusions from given premises

Example 1.1

1. *Humans are mortal*
2. *Socrates is a human*

Socrates is mortal ✓

1. *Apostles are twelve*
2. *Peter is an apostle*

Peter is twelve ✗

Invalid reasoning?

Automated reasoning aims to

enable machines to

identify the valid reasoning!!

Automated reasoning is a backbone technology!!

Applications in verification, synthesis, solving NP-hard problems, and so on.

Automated reasoning for verification tools are like engines for the cars.

Topic 1.2

Spectra of logic

Propositional logic (PL)

Propositional logic

- deals with propositions,
- only infers from the structure over the propositions, and
- does not look inside the propositions.

Example 1.2

Is the following argument valid?
If the seed catalog is correct then if seeds are planted in April then the flowers bloom in July. The flowers do not bloom in July. Therefore, if seeds are planted in April then the seed catalog is not correct.

Let us symbolize our problem
If c then if s then f. not f. Therefore, if s then not c.

- c = *the seed catalogue is correct*
- s = *seeds are planted in April*
- f = *the flowers bloom in July*

PL reasons over propositional symbols and logical connectives

First-order logic (FOL)

First-order logic

- looks inside the propositions,
- much more expressive,
- includes parameterized propositions and quantifiers over individuals, and
- can express lots of interesting math.

Example 1.3

Is the following argument valid?
Humans are mortal. Socrates is a human. Therefore, Socrates is mortal.

In the symbolic form,
For all x if $H(x)$ then $M(x)$. $H(s)$. Therefore, $M(s)$.

- $H(x) = x$ *is a human*
- $M(x) = x$ *is mortal*
- $s =$ *Socrates*

FOL is not the most general logic.
Many arguments can not be expressed in FOL

Logical theories

In a theory, we study validity of FOL arguments under some specialized assumptions (called axioms).

Example 1.4

The number theory uses symbols $0, 1, .., <, +, \cdot$ *with specialized meanings*

The following sentence has no sense until we assign the meanings to $>$ *and* $\cdot$

$$\forall x \, \exists p \, (p > x \wedge (\forall v_1 \forall v2 \, (v_1 > 1 \wedge v_2 > 1 \Rightarrow p \neq v_1 \cdot v_2)))$$

Under the meanings *it says that there are arbitrarily large prime numbers.*

In the earlier example, we had no interpretation of predicate 'x is human'. Here we precisely know what is predicate '$x < y$'.

Commentary: The specialized meaning are defined using axioms. For example, the sentence $\forall x.\ 0 + x = x$ describes one of the properties of 0 and $+$.

Higher-order logic (HOL)

Higher-order logic

- includes quantifiers over "anything",
- consists of hierarchy first order, second order, third order and so on,
- most expressive logic.

Example 1.5

$$\forall P \forall x.\ (P(x) \vee \neg P(x))$$

The quantifier is over proposition P. Therefore, the formula belongs to the second-order logic.

Commentary: The first quantifier is not allowed in the first-order logic. the first-order logic quantifies over individuals, the second-order logic quantifies over sets, the third-order logic quantifies over set of sets, and so on.

Topic 1.3

Satisfiability problem

Example: Satisfiability problem

Let x, y be rational variables.

Choose a value of x and y such that the following formula holds true.

$$x + y = 3$$

We say

$$\{x \mapsto 1, y \mapsto 2\} \models x + y = 3.$$

model formula

Commentary: We are not calling x and y rational numbers. They are not numbers. They are symbols that can hold numbers.

Reasoning == Satisfiability problem

All reasoning problems can be reduced to satisfiability problems.

Often abbreviated to **SAT problem**

Exercise 1.1

How to convert checking a valid argument into a satisfiability problem?

Example: SAT problem(contd.)

Let x, y be rational variables.

Choose a value of x and y such that the following formula holds true.

$x + y = 3 \wedge y > 10 \wedge x > 0$ theory formulas Easy

$x + y = 3 \wedge y > 10 \wedge (x > 0 \vee x < -4)$ quantifier-free formulas Hard

$\forall y.\ x + y = 3 \wedge y > 10 \wedge (x > 0 \vee x < -4)$ quantified formulas Very Hard

Commentary: The above are increasingly hard classes of satisfiability problems. The names used for hardness are informal to minimize jargon.

Quantifier-free formulas

Quantifier-free formulas consists of

- theory atoms and
- Boolean structure

Example 1.6

$$x + y = 3 \wedge y > 10 \wedge (x > 0 \vee x < -4)$$

Theory atoms

Boolean Structure

Commentary: We typically assume $\wedge$(conjunction) occurs in any formula. We say a formula has Boolean structure if it has $\vee$(disjunction).

Propositional formulas

Propositional formulas are a special case, where the theory atoms are Boolean variables.

Example 1.7

Let p_1, p_2, p_3 be Boolean variables

$$p_1 \wedge \neg p_2 \wedge (p_3 \vee p_2)$$

A satisfying model:

$$\{p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 1\} \models p_1 \wedge \neg p_2 \wedge (p_3 \vee p_2)$$

A bit of jargon

- Solvers for quantifier-free propositional formulas are called

SAT solvers.

- Solvers for quantifier-free formulas with the other theories are called

SMT solvers.

SMT = satisfiability modulo theory

Theory solvers

SMT solvers are divided into two components.

- SAT solver: it solves the Boolean structure
- Theory solver: it solves the theory constraints

Example 1.8

Let x,y be rational variables.

$$x + y = 3 \wedge y > 10 \wedge x > 0$$

Since the formula has no $\vee$ (disjunction), a solver of linear rational arithmetic can find satisfiable model using simplex algorithm.

Commentary: Simplex can not directly handle the formulas that have disjunctions.

Quantified formulas

Quantified formulas also include quantifiers.

Example 1.9

The following formulas says: give x such that for each y the body holds true.

$$\forall y.\ \underbrace{(x + y = 3 \Rightarrow y > 10 \wedge (x > 0 \vee x < -4))}_{Body}$$

A satisfying model:

$$\{x \to 1\} \models \forall y.\ (x + y = 3 \Rightarrow y > 10 \wedge (x > 0 \vee x < -4))$$

Exercise 1.2

Can we eliminate y in the above formula to obtain constraints just over x?

Commentary: Since y is quantified, the model does not assign value to y.

Exercise

Exercise 1.3

Give satisfying assignments of the following formulas

- $\neg p_1 \wedge (p_1 \vee \neg p_2)$
- $x < 3 \wedge y < 1 \wedge (x + y > 5 \vee x - y < 3)$
- $\forall x\, (x > y \Rightarrow \exists z\, (2z = x))$

Topic 1.4

Course contents and logistics

Content of the course

We will study the following topics

- Background: propositional and first-order logic (FOL) basics
- SAT solvers: satisfiability solvers for propositional logic
- SMT solvers: satisfiability modulo theory solvers

(topic uncovered in the short course)

- Decision procedures: algorithms for solving theory constraints
- Solvers for quantifiers
  - SMT+quantifier
  - Saturation solvers: FOL solvers
- Interactive theorem prover for higher order logics

Evaluation and website

Evaluation

- Two programming assignments

For the further information

`https://www.cse.iitb.ac.in/~akg/courses/2019-sat-mandi/`

All the assignments and slides will be posted on the website.

Please read the conditions to attend the course. They are on the website.

Topic 1.5

Propositional logic

Propositional variables

We assume that there is a set **Vars** of countably many propositional variables.

- Since **Vars** is countable, we assume that variables are indexed.

$$\textbf{Vars} = \{p_1, p_2, \dots\}$$

- The variables are just names/symbols without inherent meaning
- We may also use $p, q, r, .., x, y, z$ to denote the propositional variables

Commentary: All results presented in this course are extendable to uncountable **Vars**. For the uncountable setting, we need transfinite induction. We will ignore those extensions.

Logical connectives

The following 10 symbols that are called logical connectives.

formal name	symbol	read as	
true	$\top$	top	0-ary symbols
false	$\bot$	bot	
negation	$\neg$	not	unary symbols
conjunction	$\wedge$	and	binary symbols
disjunction	$\vee$	or	
implication	$\Rightarrow$	implies	
equivalence	$\Leftrightarrow$	iff	
exclusive or	$\oplus$	xor	
open parenthesis	(		
close parenthesis	)		

We assume that the logical connectives are not in **Vars**.

Propositional formulas

A propositional formula is a finite string containing symbols in **Vars** and logical connectives.

Definition 1.1

The set of propositional formulas is the smallest set $\mathbf{P}$ *such that*

- $\top, \bot \in \mathbf{P}$
- *if* $p \in \mathbf{Vars}$ *then* $p \in \mathbf{P}$
- *if* $F \in \mathbf{P}$ *then* $\neg F \in \mathbf{P}$
- *if* $\circ$ *is a binary symbol and* $F, G \in \mathbf{P}$ *then* $(F \circ G) \in \mathbf{P}$

Definition 1.2 (Alternate presentation of the above definition)

$F \in \mathbf{P}$ *if*

$$F \triangleq p \mid \top \mid \bot \mid \neg F \mid (F \vee F) \mid (F \wedge F) \mid (F \Rightarrow F) \mid (F \Leftrightarrow F) \mid (F \oplus F)$$

where $p \in \mathbf{Vars}$.

Commentary: For someone unfamiliar with context free grammar, please ignore the alternative definition.

Some notation

Definition 1.3

$\top, \bot$, *and* $p \in$ **Vars** *are atomic formulas.*

Definition 1.4

For each $F \in$ **P**, *let* **Vars**(F) *be the set of variables appearing in* F.

Exercise 1.4

"appear in" is not defined yet. Give a formal definition of the phrase.

Examples of propositional formulas

Exercise 1.5

Is the following belongs to **P***?*

- $\top \Rightarrow \bot$ ✗
- $(\top \Rightarrow \bot)$ ✓
- $(p_1 \Rightarrow \neg p_2)$ ✓
- (p_1)✗
- $\neg\neg\neg\neg\neg\neg\neg p_1$ ✓

Not all strings over **Vars** and logical connectives are in **P**.

How can we argue that a string does or does not belong to **P**?

We need a method to recognize a string belongs to **P** or not.

Parse tree

By def. 1.1, $F \in \mathbf{P}$ iff F is obtained by unfolding of the generation rules

Definition 1.5

A parse tree of a formula $F \in \mathbf{P}$ is a tree such that

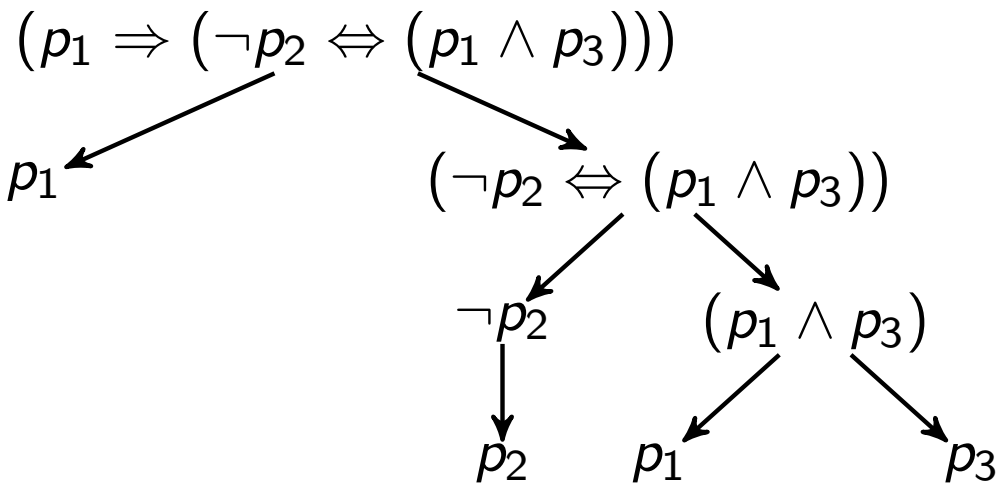
- *the root is F,*
- *leaves are atomic formulas, and*
- *each internal node is formed by applying some formation rule on its children.*

Theorem 1.1

reverse direction is immediate. In forward direction, we can prove a stronger theorem, i.e., existance of unique parsing tree

$F \in \mathbf{P}$ iff there is a parse tree of F.

Example 1.10



Truth values

We denote the set of truth values as $\mathcal{B} \triangleq \{0, 1\}$.

0 and 1 are only distinct objects without any intuitive meaning.

We may view 0 as false and 1 as true but this is only our emotional response to the symbols.

Model

Definition 1.6

A model is an element of $\mathbf{Vars} \to \mathcal{B}$.

Example 1.11

$\{p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 0, \dots\}$ *is a model*

Since **Vars** is countable, the set of models is non-empty, and infinitely many.

A model m may or may not satisfy a formula F.
The satisfaction relation is usually denoted by $m \models F$ in infix notation.

Propositional Logic Semantics

Why is "smallest relation" not mentioned?

Definition 1.7

The satisfaction relation $\models$ between models and formulas is the relation that satisfies the following conditions.

$$
\begin{aligned}
m &\models \top \\
m &\models p && \text{if } m(p) = 1 \\
m &\models \neg F && \text{if } m \not\models F \\
m &\models F_1 \vee F_2 && \text{if } m \models F_1 \text{ or } m \models F_2 \\
m &\models F_1 \wedge F_2 && \text{if } m \models F_1 \text{ and } m \models F_2 \\
m &\models F_1 \oplus F_2 && \text{if } m \models F_1 \text{ or } m \models F_2, \text{ but not both} \\
m &\models F_1 \Rightarrow F_2 && \text{if if } m \models F_1 \text{ then } m \models F_2 \\
m &\models F_1 \Leftrightarrow F_2 && \text{if } m \models F_1 \text{ iff } m \models F_2
\end{aligned}
$$

Theorem 1.2

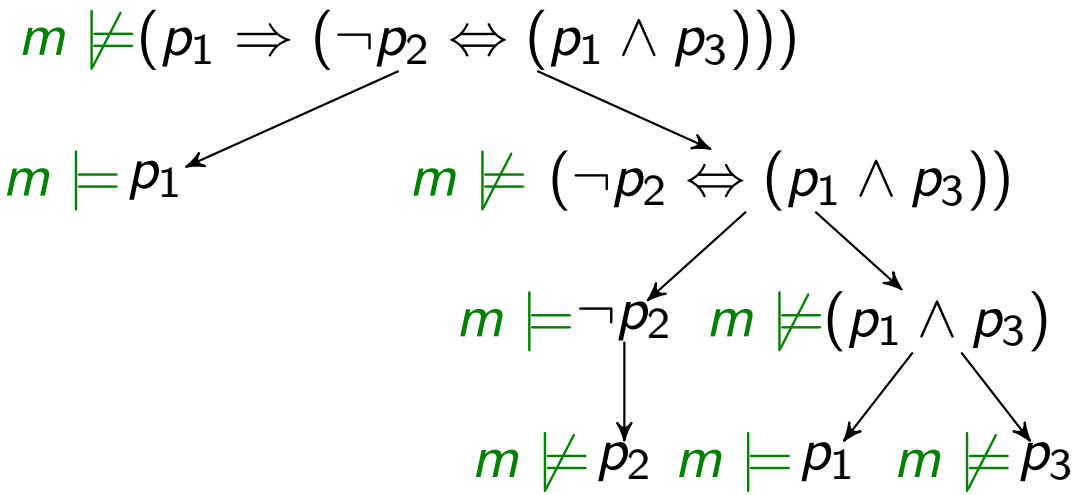
There is exactly one relation that satisfies the above conditions.

Proof. Since each $F \in \mathbf{P}$ has a unique parse tree, we have a terminating procedure to check if $m \models F$ or not.

Example: satisfaction relation

Example 1.12

Consider model $m = \{p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 0, \dots\}$
And, formula $(p_1 \Rightarrow (\neg p_2 \Leftrightarrow (p_1 \wedge p_3)))$



Exercise 1.6

write the satisfiability checking procedure formally.

Satisfiable, valid, unsatisfiable

We say

- m *satisfies* F if $m \models F$,
- F is *satisfiable* if there is a model m such that $m \models F$,
- F is *valid* (written $\models F$) if for each model m $m \models F$, and
- F is *unsatisfiable* (written $\not\models F$) if there is no model m such that $m \models F$.

Exercise 1.7

If F is sat then $\neg F$ is _____.
If F is valid then $\neg F$ is _____.
If F is unsat then $\neg F$ is _____.

Implication

We extend the usage of $\models$.

Definition 1.8

Let M be a (possibly infinite) set of models.
$M \models F$ if for each $m \in M$, $m \models F$.

Definition 1.9

Let Σ be a (possibly infinite) set of formulas.
$\Sigma \models F$ if for each model m that satisfies each formula in Σ, $m \models F$.

$\Sigma \models F$ is read Σ implies F.
If $\{G\} \models F$ then we may write $G \models F$.

Theorem 1.3

$F \models G$ iff $\models (F \Rightarrow G)$.

Equivalent

Definition 1.10

Let $F \equiv G$ *if* $m \models F$ *iff* $m \models G$.

Theorem 1.4

$F \equiv G$ *iff* $\models (F \Leftrightarrow G)$.

Equisatisfiable and Equivalid

Definition 1.11

Formulas F and G are equisatisfiable if

$$F \text{ is sat} \quad \text{iff} \quad G \text{ is sat.}$$

Definition 1.12

Formulas F and G are equivalid if

$$\models F \quad \text{iff} \quad \models G.$$

Commentary: The concept of equisatisfiable is used in formula transformations. We often say that after a transformation the formula remained equisatisfiable. Equivalid is the dual concept, rarely used in practice.

Topic 1.6

Decidability of SAT

Partial models

Let $m|_{\mathbf{Vars}(F)} : \mathbf{Vars}(F) \to \mathcal{B}$ and for each $p \in \mathbf{Vars}(F)$, $m|_{\mathbf{Vars}(F)}(p) = m(p)$

Theorem 1.5

If $m|_{\mathbf{Vars}(F)} = m'|_{\mathbf{Vars}(F)}$ *then* $m \models F$ *iff* $m' \models F$

Proof sketch.

The procedure to check $m \models F$ only looks at the $\mathbf{Vars}(F)$ part of m.
Therefore, any extension of $m|_{\mathbf{Vars}(F)}$ will have same result either $m \models F$ or $m \not\models F$. □

Definition 1.13

We will call elements of $\mathbf{Vars} \hookrightarrow \mathcal{B}$ *as partial models.*

Exercise 1.8

Write the above proof formally.

Propositional satisfiability problem

The following problem is called the satisfiability problem

$$\text{For a given } F \in \mathbf{P}, \text{ is } F \text{ satisfiable?}$$

Theorem 1.6

The propositional satisfiability problem is decidable.

Proof.

Let $n = |\mathbf{Vars}(F)|$.
We need to enumerate 2^n elements of $\mathbf{Vars}(F) \to \mathcal{B}$.
If any of the models satisfy the formula, then we can always extend to a the full model and F is sat
Otherwise, F is unsat. □

Exercise 1.9

Give a procedure to decide the validity of a formula.

Topic 1.7

Truth tables

Truth tables

Truth tables was the first method to decide propositional logic.

The method is usually presented in slightly different notation.

We need to assign a truth value to every formula.

Truth function

A model m is in $\mathbf{Vars} \to \mathcal{B}$.

We can extend m to $\mathbf{P} \to \mathcal{B}$ in the following way.

$$m(F) = \begin{cases} 1 & m \models F \\ 0 & otherwise. \end{cases}$$

The extended m is called truth function.

Since truth functions are natural extensions of models, we did not introduce new symbols.

Truth functions for logical connectives

Let F and G are logical formulas, and m is a model.
Due to the semantics of the propositional logic, the following holds for the truth functions.

$m(F)$	$m(\neg F)$
0	1
1	0

$m(F)$	$m(G)$	$m(F \wedge G)$	$m(F \vee G)$	$m(F \oplus G)$	$m(F \Rightarrow G)$	$m(F \Leftrightarrow G$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Exercise 1.10

Verify the above table against the definition of $\models$

Truth table

For a formula F, a truth table consists of $2^{|\mathbf{Vars}(F)|}$ rows. Each row considers one of the partial models and computes the truth value of F for each model.

Example 1.13

Consider $(p_1 \Rightarrow (\neg p_2 \Leftrightarrow (p_1 \wedge p_3)))$

We will not write $m(.)$ *in the top row for brevity.*

p_1	p_2	p_3	$(p_1$	$\Rightarrow$	$(\ \neg$	p_2	$\Leftrightarrow ($	p_1	$\wedge$	$p_3\ ))$
0	0	0	0	1	1	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	1
0	1	0	0	1	0	1	1	0	0	0
0	1	1	0	1	0	1	1	0	0	1
1	0	0	1	0	1	0	0	1	0	0
1	0	1	1	1	1	0	1	1	1	1
1	1	0	1	1	0	1	1	1	0	0
1	1	1	1	0	0	1	0	1	1	1

The column under the leading connective has 1s therefore the formula is sat.

But, there are some 0s in the column therefore the formula is not valid.

Tedious truth tables

- We need to write 2^n rows even if some simple observations about the formula may prove unsatisfiablity/satisfiability.
  For example,
  - $(a \vee (c \wedge a))$ is sat (why? - no negation)
  - $(a \vee (c \wedge a)) \wedge \neg(a \vee (c \wedge a))$ is unsat (why?- contradiction at top level)
- We should be able to take such shortcuts?

We will see many methods that will allow us to take such shortcuts. But not now!

Normal forms

We usually try to aovid too many cases, if we want to develop algorithms for checking satisfiability.

Therefore, we handle the propositional formulas in normal forms.

Topic 1.8

Negation normal form

Negation normal form(NNF)

Definition 1.14

A formula is in NNF if $\neg$ appears only in front of the propositional variables.

Theorem 1.7

For every formula F, there is a formula F' in NNF such that $F \equiv F'$.

Proof.

Due to the following equivalences, we can push $\neg$ under the connectives

- $\neg\neg p \equiv p$
- $\neg(p \vee q) \equiv \neg p \wedge \neg q$
- $\neg(p \wedge q) \equiv \neg p \vee \neg q$
- $\neg(p \Rightarrow q) \equiv p \wedge \neg q$
- $\neg(p \oplus q) \equiv \neg p \oplus q \equiv p \Leftrightarrow q$
- $\neg(p \Leftrightarrow q) \equiv p \oplus q$

□

- Often we assume that the formulas are in NNF.
- However, there are negations hidden inside $\oplus$, $\Rightarrow$, and $\Leftrightarrow$. In practice, these symbols are also expected to be removed while producing NNF

Exercise 1.11

Write an efficient algorithm to convert a propositional formula to NNF?

Example :NNF

Example 1.14

Consider $\neg(q \Rightarrow ((p \vee \neg s) \oplus r))$
$\equiv q \wedge \neg((p \vee \neg s) \oplus r)$
$\equiv q \wedge \neg(p \vee \neg s) \oplus r$
$\equiv q \wedge (\neg p \wedge \neg\neg s) \oplus r$
$\equiv q \wedge (\neg p \wedge s) \oplus r$

Exercise 1.12

Convert the following formulas into NNF

- $\neg(p \Rightarrow q)$
- $\neg(\neg((s \Rightarrow \neg(p \Leftrightarrow q))) \oplus (\neg q \vee r))$

Exercise 1.13

Are there any added difficulties if the formula is given as a DAG not as a tree?

Eliminating $\oplus$, $\Rightarrow$, and $\Leftrightarrow$

- $(p \Rightarrow q) \equiv (\neg p \vee q)$
- $(p \oplus q) \equiv (p \vee q) \wedge (\neg p \vee \neg q)$
- $(p \Leftrightarrow q) \equiv (p \vee \neg q) \wedge (q \vee \neg p)$

Topic 1.9

Conjunctive normal form

Conjunctive normal form(CNF)

Definition 1.15

A formula is in CNF if it is a conjunction of clauses.

Since $\wedge$ is associative, commutative and absorbs multiple occurrences, a CNF formula may be referred as a set of clauses

Example 1.15

- $\neg p$ *and* p *both are in CNF.*
- $(p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ *in CNF.*
- $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$ *is the same CNF formula.*
- $\{\{p, \neg q\}, \{r, \neg q\}, \{\neg r\}\}$ *is the same CNF formula.*

Exercise 1.14

Write a formal grammar for CNF

CNF conversion

Theorem 1.8

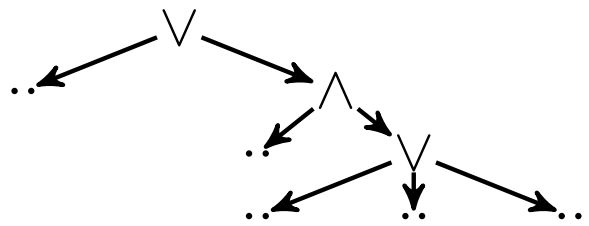
For every formula F there is another formula F' in CNF s.t. $F \equiv F'$.

Proof.

Let us suppose we have

- removed $\oplus$, $\Rightarrow$, $\Leftrightarrow$ using the standard equivalences,
- converted the formula in NNF, and
- flattened $\wedge$ and $\vee$.

Now the formulas have the following form with literals at leaves.



After the push formula size grows! Why should the procedure terminate?

Since $\vee$ distributes over $\wedge$, we can always push $\vee$ inside $\wedge$.

Are we done?

Eventually, we will obtain a formula that is CNF. □

Commentary: The above is a good example of an algorithm that has intuitively clear, but formally non-trivial termination argument.

CNF examples

Example 1.16

Consider $(p \Rightarrow (\neg q \wedge r)) \wedge (p \Rightarrow \neg q)$
$\equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q)$
$\equiv ((\neg p \vee \neg q) \wedge (\neg p \vee r)) \wedge (\neg p \vee \neg q)$
$\equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

Exercise 1.15

Convert the following formulas into CNF

1. $\neg((p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r)))$
2. $(p \Rightarrow (\neg q \Rightarrow r)) \wedge (p \Rightarrow \neg q) \Rightarrow (p \Rightarrow r)$

Conjunctive normal form(CNF) (2)

- A unit clause contains only one literal.
- A binary clause contains two literals.
- A ternary clause contains three literals.
- We naturally extend definition of the clauses to empty set of literals. We refer to $\bot$ as empty clause.

Example 1.17

- $(p \wedge q \wedge \neg r)$ *has three unit clauses*
- $(p \vee \neg q \vee \neg s) \wedge (p \vee q) \wedge \neg r$ *has a ternary, a binary and a unit clause*

Exercise 1.16

a. Show F' obtained from the procedure may be exponentially larger than F

b. Give a linear time algorithm to prove validity of a CNF formula

Tseitin's encoding (Plaisted-Greenbaum optimization included)

We can translate every formula into CNF without exponential explosion using Tseitin's encoding by introducing fresh variables.

1. Assume input formula F is NNF without $\oplus$, $\Rightarrow$, and $\Leftrightarrow$.
2. Find a $G_1 \wedge \cdots \wedge G_n$ that is just below a $\vee$ in $F(G_1 \wedge \cdots \wedge G_n)$
3. Replace $F(G_1 \wedge .. \wedge G_n)$ by $F(p) \wedge (\neg p \vee G_1) \wedge .. \wedge (\neg p \vee G_n)$, where p is a fresh variable
4. goto 2

Exercise 1.17

Convert the following formulas into CNF using Tseitin's encoding

1. $(p \Rightarrow (\neg q \wedge r)) \wedge (p \Rightarrow \neg q)$
2. $(p \Rightarrow q) \vee (q \Rightarrow \neg r) \vee (r \Rightarrow q) \Rightarrow \neg(\neg(q \Rightarrow p) \Rightarrow (q \Leftrightarrow r))$

Exercise 1.18

Modify the encoding such that it works without the assumptions at step 1

Hint: Download sat solver $wget http://fmv.jku.at/limboole/limboole1.1.tar.gz look for function tseitin in file limboole.c

End of Lecture 1