CS228 : Logic for Computer Science

Module 1: Propositional Logic

Lecture 7: Normal Forms

Instructor: S. Akshay

IIT Bombay, India

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)
- Problems of interest - Satisfiability, Validity, Equivalence, Counting, etc.

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)
- Problems of interest - Satisfiability, Validity, Equivalence, Counting, etc.
- Ways to solve these problems:
  1. Use the Truth Table
  2. Use the rules of Natural Deduction: a formal proof system

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)
- Problems of interest - Satisfiability, Validity, Equivalence, Counting, etc.
- Ways to solve these problems:
  1. Use the Truth Table
     - Advantages: Can solve by just table lookup.
     - Disadvantages: TOO large!
  2. Use the rules of Natural Deduction: a formal proof system

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)
- Problems of interest - Satisfiability, Validity, Equivalence, Counting, etc.
- Ways to solve these problems:
  1. Use the Truth Table
     - Advantages: Can solve by just table lookup.
     - Disadvantages: TOO large!
  2. Use the rules of Natural Deduction: a formal proof system
     - Advantages: Purely symbolic, often shorter than Truth Table, Sound and Complete.
     - Disadvantages: Which rule to use when requires ingenuity, worst case proof will be huge.

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)
- Problems of interest - Satisfiability, Validity, Equivalence, Counting, etc.
- Ways to solve these problems:
  1. Use the Truth Table
     - Advantages: Can solve by just table lookup.
     - Disadvantages: TOO large!
  2. Use the rules of Natural Deduction: a formal proof system
     - Advantages: Purely symbolic, often shorter than Truth Table, Sound and Complete.
     - Disadvantages: Which rule to use when requires ingenuity, worst case proof will be huge.

Can we say when these problems are easy?

So far: Propositional Logic

Module 1: Propositional Logic - Encoding knowledge into simple formulas

Logic: a formal language & calculus for reasoning

- Propositional Logic Syntax (write a grammar)
- Propositional Logic Semantics (via truth tables)
- Problems of interest - Satisfiability, Validity, Equivalence, Counting, etc.
- Ways to solve these problems:
  1. Use the Truth Table
     - Advantages: Can solve by just table lookup.
     - Disadvantages: TOO large!
  2. Use the rules of Natural Deduction: a formal proof system
     - Advantages: Purely symbolic, often shorter than Truth Table, Sound and Complete.
     - Disadvantages: Which rule to use when requires ingenuity, worst case proof will be huge.

Can we say when these problems are easy?

- Exploit their structure!

Topic 7.1

Normal Forms

Normal forms

Questions

1. Can some representation make the formula easier to analyze?
   - Example: is $(x \vee \neg x) \vee (\neg x \wedge y) \vee (x \wedge y \wedge w) \vee (\neg x \wedge \neg z \wedge \neg w)$ valid?
2. Are there "Normal forms" which allow for easy and uniform algorithmic analysis?

Normal forms

Questions

1. Can some representation make the formula easier to analyze?
   - Example: is $(x \vee \neg x) \vee (\neg x \wedge y) \vee (x \wedge y \wedge w) \vee (\neg x \wedge \neg z \wedge \neg w)$ valid?
     - Would you write the truth table with 16 rows?
     - Easier: would you try to derive it from no premises?
     - Even easier: just observe that it's a disjunction of many subformulas, and the first one is valid, so the entire thing is valid!
2. Are there "Normal forms" which allow for easy and uniform algorithmic analysis?

Normal forms

Questions

1. Can some representation make the formula easier to analyze?
   - Example: is $(x \vee \neg x) \vee (\neg x \wedge y) \vee (x \wedge y \wedge w) \vee (\neg x \wedge \neg z \wedge \neg w)$ valid?
     - Would you write the truth table with 16 rows?
     - Easier: would you try to derive it from no premises?
     - Even easier: just observe that it's a disjunction of many subformulas, and the first one is valid, so the entire thing is valid!
2. Are there "Normal forms" which allow for easy and uniform algorithmic analysis?
   - Building algorithms and tools for handling arbitrary formulas is difficult
   - Normalization also results in standardization and interoperability of tools.

First step : remove $\oplus$, $\rightarrow$, and $\leftrightarrow$.

Use equivalences to remove $\oplus$, $\rightarrow$, and $\leftrightarrow$ from a formula.

First step : remove $\oplus$, $\rightarrow$, and $\leftrightarrow$.

Use equivalences to remove $\oplus$, $\rightarrow$, and $\leftrightarrow$ from a formula. Recall that $p \equiv q$ denotes semantic equivalence.

First step : remove $\oplus$, $\rightarrow$, and $\leftrightarrow$.

Use equivalences to remove $\oplus$, $\rightarrow$, and $\leftrightarrow$ from a formula. Recall that $p \equiv q$ denotes semantic equivalence.

Is there a difference between $p \equiv q$ and $\vdash p \leftrightarrow q$?

First step : remove $\oplus$, $\rightarrow$, and $\leftrightarrow$.

Use equivalences to remove $\oplus$, $\rightarrow$, and $\leftrightarrow$ from a formula. Recall that $p \equiv q$ denotes semantic equivalence.

Is there a difference between $p \equiv q$ and $\vdash p \leftrightarrow q$?

- $(p \rightarrow q) \equiv (\neg p \vee q)$
- $(p \oplus q) \equiv (p \vee q) \wedge (\neg p \vee \neg q)$
- $(p \leftrightarrow q) \equiv \neg(p \oplus q)$

First step : remove $\oplus$, $\rightarrow$, and $\leftrightarrow$.

Use equivalences to remove $\oplus$, $\rightarrow$, and $\leftrightarrow$ from a formula. Recall that $p \equiv q$ denotes semantic equivalence.

Is there a difference between $p \equiv q$ and $\vdash p \leftrightarrow q$?

- $(p \rightarrow q) \equiv (\neg p \vee q)$
- $(p \oplus q) \equiv (p \vee q) \wedge (\neg p \vee \neg q)$
- $(p \leftrightarrow q) \equiv \neg(p \oplus q)$

For the ease of presentation, we will assume you can remove them at will.

First step : remove $\oplus$, $\rightarrow$, and $\leftrightarrow$.

Use equivalences to remove $\oplus$, $\rightarrow$, and $\leftrightarrow$ from a formula. Recall that $p \equiv q$ denotes semantic equivalence.

Is there a difference between $p \equiv q$ and $\vdash p \leftrightarrow q$?

- $(p \rightarrow q) \equiv (\neg p \vee q)$
- $(p \oplus q) \equiv (p \vee q) \wedge (\neg p \vee \neg q)$
- $(p \leftrightarrow q) \equiv \neg(p \oplus q)$

For the ease of presentation, we will assume you can remove them at will.

So our propositional formulas are only in terms of $\neg, \vee, \wedge$.

Topic 7.2

Negation normal form

Negation normal form

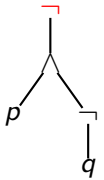
Definition

A formula is in Negation Normal Form (NNF) if $\neg$ is only in front of the propositional variables and only other connectives are $\wedge, \vee$.

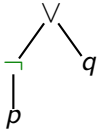
Negation normal form

Definition

A formula is in Negation Normal Form (NNF) if $\neg$ is only in front of the propositional variables and only other connectives are $\wedge, \vee$.



$\neg(p \wedge \neg q)$ ✗ Not in NNF



$\neg p \vee q$ ✓ In NNF

Negation normal form (contd.)

Theorem

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

Negation normal form (contd.)

Theorem

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

Proof.

Convert to formula with only $\neg, \wedge, \vee$ and then apply DeMorgan's laws! □

Negation normal form (contd.)

Theorem

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

Proof.

Convert to formula with only $\neg, \wedge, \vee$ and then apply DeMorgan's laws! □

Exercise 7.1

Convert the following formulas into NNF

- $\neg(p \to q)$
- $\neg(\neg((s \to \neg(p \leftrightarrow q))) \oplus (\neg q \vee r))$
- $\neg((p \to q) \to (p \to q))$
- $\neg\neg\neg p$

Negation normal form (contd.)

Theorem

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

Proof.

Convert to formula with only $\neg, \wedge, \vee$ and then apply DeMorgan's laws! □

Exercise 7.1

Convert the following formulas into NNF

- $\neg(p \to q)$
- $\neg((p \to q) \to (p \to q))$
- $\neg(\neg((s \to \neg(p \leftrightarrow q))) \oplus (\neg q \vee r))$
- $\neg\neg\neg p$

Exercise 7.2 (Is the NNF "canonical"?)

Do there exist two distinct formulae that are both in NNF but are equivalent?

Topic 7.3

Conjunctive normal form

Some terminology

- Propositional variables are also referred as atoms

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Examples

- p is an atom but $\neg p$ is not.

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Examples

- p is an atom but $\neg p$ is not.
- $\neg p$ and p both are literals.

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Examples

- p is an atom but $\neg p$ is not.
- $\neg p$ and p both are literals.
- $p \vee \neg p \vee p \vee q$ is a clause.

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Examples

- p is an atom but $\neg p$ is not.
- $\neg p$ and p both are literals.
- $p \vee \neg p \vee p \vee q$ is a clause.

Since $\vee$ is associative, commutative, and idempotent, a clause is sometimes referred to as a set of literals.
e.g., $p \vee \neg p \vee p \vee q$ can be written as $\{p, \neg p, q\}$.

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Examples

- p is an atom but $\neg p$ is not.
- $\neg p$ and p both are literals.
- $p \vee \neg p \vee p \vee q$ is a clause.

idempotent:
$p \vee p \equiv p$
(repeats don't matter)

Since $\vee$ is associative, commutative, and idempotent, a clause is sometimes referred to as a set of literals.
e.g., $p \vee \neg p \vee p \vee q$ can be written as $\{p, \neg p, q\}$.

Conjunctive normal form (CNF)

Definition

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

Conjunctive normal form (CNF)

Definition

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

- $\neg p$ and p both are in CNF.

Conjunctive normal form (CNF)

Definition

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

- $\neg p$ and p both are in CNF.
- $F = (p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ is in CNF.

Conjunctive normal form (CNF)

Definition

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

- $\neg p$ and p both are in CNF.
- $F = (p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ is in CNF.

Since $\wedge$ is associative, commutative and idempotent, a CNF formula is sometimes written as a set of clauses. E.g., F above can be written as:

- $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$

Conjunctive normal form (CNF)

Definition

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

- $\neg p$ and p both are in CNF.
- $F = (p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ is in CNF.

Since $\wedge$ is associative, commutative and idempotent, a CNF formula is sometimes written as a set of clauses. E.g., F above can be written as:

- $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$ or
- $\{\{p, \neg q\}, \{r, \neg q\}, \{\neg r\}\}$

Conjunctive normal form (CNF)

Definition

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

- $\neg p$ and p both are in CNF.
- $F = (p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ is in CNF.

Since $\wedge$ is associative, commutative and idempotent, a CNF formula is sometimes written as a set of clauses. E.g., F above can be written as:

- $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$ or
- $\{\{p, \neg q\}, \{r, \neg q\}, \{\neg r\}\}$

Exercise 7.3 Which of the following are in CNF?

- $(p \vee q) \wedge (\neg p \wedge q)$
- $(p \wedge q) \vee (\neg p \vee q)$
- $p \rightarrow q$
- $(p \vee q) \vee (\neg p \vee q)$

Exercise 7.4 Write a formal grammar for CNF.

Conjunctive normal form (CNF)

Definition

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

- $\neg p$ and p both are in CNF.
- $F = (p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ is in CNF.

Since $\wedge$ is associative, commutative and idempotent, a CNF formula is sometimes written as a set of clauses. E.g., F above can be written as:

- $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$ or
- $\{\{p, \neg q\}, \{r, \neg q\}, \{\neg r\}\}$

Exercise 7.3 Which of the following are in CNF?

- $(p \vee q) \wedge (\neg p \wedge q)$
- $(p \wedge q) \vee (\neg p \vee q)$
- $p \rightarrow q$
- $(p \vee q) \vee (\neg p \vee q)$

Exercise 7.4 Write a formal grammar for CNF.

$CNF ::= Clause \mid Clause \wedge CNF$

$Clause ::= p \mid \neg p \mid Clause \vee Clause$

CNF conversion

Theorem

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

CNF conversion

Theorem

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

Example

Conversion to CNF
$(p \rightarrow (\neg q \wedge r)) \wedge (p \rightarrow \neg q)$

CNF conversion

Theorem

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

Example

Conversion to CNF

$(p \rightarrow (\neg q \wedge r)) \wedge (p \rightarrow \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q)$

CNF conversion

Theorem

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

Example

Conversion to CNF

$(p \rightarrow (\neg q \wedge r)) \wedge (p \rightarrow \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

CNF conversion

Theorem

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

Example

Conversion to CNF

$(p \rightarrow (\neg q \wedge r)) \wedge (p \rightarrow \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

Proof.

This gives an easy algorithm!

CNF conversion

Theorem

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

Example

Conversion to CNF

$(p \to (\neg q \wedge r)) \wedge (p \to \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

Proof.

This gives an easy algorithm!

- Step 1: Convert F to NNF.

CNF conversion

Theorem

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

Example

Conversion to CNF

$(p \to (\neg q \wedge r)) \wedge (p \to \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

Proof.

This gives an easy algorithm!

- Step 1: Convert F to NNF.
- Step 2: Flatten the $\wedge$ and $\vee$ gates, e.g., $(x \wedge y) \wedge (z \wedge y) \wedge (w \wedge z) \equiv (x \wedge y \wedge z \wedge w)$

CNF conversion

Theorem

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

Example

Conversion to CNF

$(p \to (\neg q \wedge r)) \wedge (p \to \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

Proof.

This gives an easy algorithm!

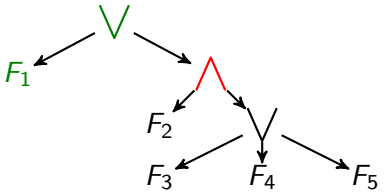
- Step 1: Convert F to NNF.
- Step 2: Flatten the $\wedge$ and $\vee$ gates, e.g., $(x \wedge y) \wedge (z \wedge y) \wedge (w \wedge z) \equiv (x \wedge y \wedge z \wedge w)$
- Step 3: Use distributivity (see next slide!)

...

CNF conversion (contd.)

Proof(contd.)

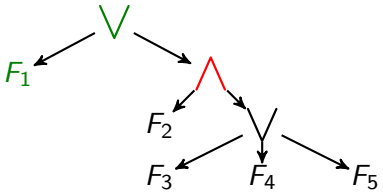
Now the formulas have the following form with literals at leaves, e.g., $F_1 \vee (F_2 \wedge (F_3 \vee F_4 \vee F_5))$.

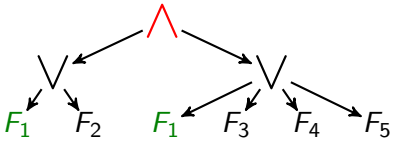


CNF conversion (contd.)

Proof(contd.)

Now the formulas have the following form with literals at leaves, e.g., $F_1 \vee (F_2 \wedge (F_3 \vee F_4 \vee F_5))$.

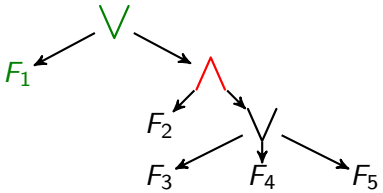


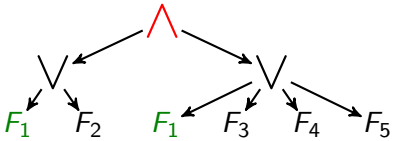


CNF conversion (contd.)

Proof(contd.)

Now the formulas have the following form with literals at leaves, e.g., $F_1 \vee (F_2 \wedge (F_3 \vee F_4 \vee F_5))$.



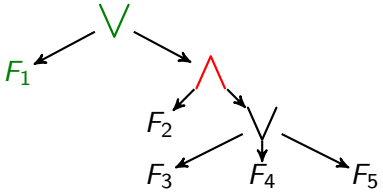


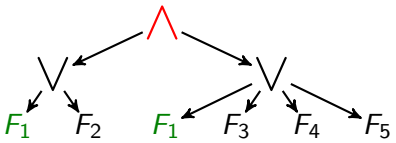
Since $\vee$ distributes over $\wedge$, we can always push $\vee$ inside $\wedge$.

CNF conversion (contd.)

Proof(contd.)

Now the formulas have the following form with literals at leaves, e.g., $F_1 \vee (F_2 \wedge (F_3 \vee F_4 \vee F_5))$.



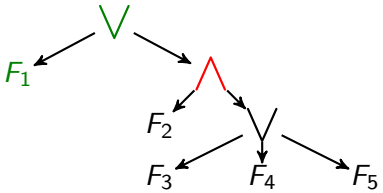


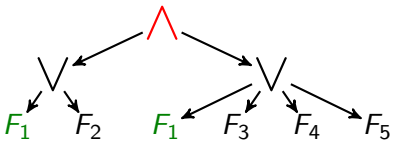
Since $\vee$ distributes over $\wedge$, we can always push $\vee$ inside $\wedge$.
Eventually, we will obtain a formula that is CNF.

CNF conversion (contd.)

Proof(contd.)

Now the formulas have the following form with literals at leaves, e.g., $F_1 \vee (F_2 \wedge (F_3 \vee F_4 \vee F_5))$.





Since $\vee$ distributes over $\wedge$, we can always push $\vee$ inside $\wedge$.
Eventually, we will obtain a formula that is CNF. (Note how F_1 gets duplicated.) □

Exercise 7.5

Convert $(a \vee b) \wedge (c \vee (d \wedge e))$ to CNF using distributivity.

Solution: $(a \vee b) \wedge (c \vee (d \wedge e)) \equiv (a \vee b) \wedge (c \vee d) \wedge (c \vee e)$

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

- Truth table: exponential in the number of variables
- Natural deduction: no algorithm – requires ingenuity

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

- Truth table: exponential in the number of variables
- Natural deduction: no algorithm – requires ingenuity

What if F is in CNF?

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

- Truth table: exponential in the number of variables
- Natural deduction: no algorithm – requires ingenuity

What if F is in CNF?

Exercise 7.6 Give an algorithm to check for validity of formula in CNF. What is its complexity?

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

- Truth table: exponential in the number of variables
- Natural deduction: no algorithm – requires ingenuity

What if F is in CNF?

Exercise 7.6 Give an algorithm to check for validity of formula in CNF. What is its complexity?

What about satisfiability?

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

- Truth table: exponential in the number of variables
- Natural deduction: no algorithm – requires ingenuity

What if F is in CNF?

Exercise 7.6 Give an algorithm to check for validity of formula in CNF. What is its complexity?

What about satisfiability?

Fact: Checking satisfiability of a CNF formula is hard – in fact it is NP-complete!

CNF Conversion Applications

Checking validity: how hard is it in general?

Given an arbitrary formula F, how do we check if it's valid?

- Truth table: exponential in the number of variables
- Natural deduction: no algorithm – requires ingenuity

What if F is in CNF?

Exercise 7.6 Give an algorithm to check for validity of formula in CNF. What is its complexity?

What about satisfiability?

Fact: Checking satisfiability of a CNF formula is hard – in fact it is NP-complete!

(NP-complete, informally: among the hardest problems whose solutions can be *verified* in polynomial time – if you could solve CNF-SAT quickly, you could solve *every* such problem quickly.)

Advantages of CNF

- Like NNF, it has very few logical connectives

Advantages of CNF

- Like NNF, it has very few logical connectives
- Simple structure

Advantages of CNF

- Like NNF, it has very few logical connectives
- Simple structure
- Validity is easy

Advantages of CNF

- Like NNF, it has very few logical connectives
- Simple structure
- Validity is easy
- Many problems naturally encode into CNF.

  We will see more examples on this soon!

Advantages of CNF

- Like NNF, it has very few logical connectives
- Simple structure
- Validity is easy
- Many problems naturally encode into CNF.

  We will see more examples on this soon!
- As a result, many solvers take input in CNF

Advantages of CNF

- Like NNF, it has very few logical connectives
- Simple structure
- Validity is easy
- Many problems naturally encode into CNF.

  We will see more examples on this soon!

- As a result, many solvers take input in CNF

So given formula F, an important question is to convert it to CNF.

Topic 7.4

Tseitin's encoding

Conversion to CNF

We already saw a naive algorithm

- Step 1: Convert F to NNF.
- Step 2: Flatten the $\wedge$ and $\vee$ gates, e.g., $(x \wedge y) \wedge (z \wedge y) \wedge (w \wedge z) \equiv (x \wedge y \wedge z \wedge w)$
- Step 3: Use distributivity

Conversion to CNF

We already saw a naive algorithm

- Step 1: Convert F to NNF.
- Step 2: Flatten the $\wedge$ and $\vee$ gates, e.g., $(x \wedge y) \wedge (z \wedge y) \wedge (w \wedge z) \equiv (x \wedge y \wedge z \wedge w)$
- Step 3: Use distributivity

But the transformation using distributivity explodes the formula.

Conversion to CNF

We already saw a naive algorithm

- Step 1: Convert F to NNF.
- Step 2: Flatten the $\wedge$ and $\vee$ gates, e.g., $(x \wedge y) \wedge (z \wedge y) \wedge (w \wedge z) \equiv (x \wedge y \wedge z \wedge w)$
- Step 3: Use distributivity

But the transformation using distributivity explodes the formula.

What is the complexity blow-up in converting an arbitrary formula to CNF?

Conversion to CNF

We already saw a naive algorithm

- Step 1: Convert F to NNF.
- Step 2: Flatten the $\wedge$ and $\vee$ gates, e.g., $(x \wedge y) \wedge (z \wedge y) \wedge (w \wedge z) \equiv (x \wedge y \wedge z \wedge w)$
- Step 3: Use distributivity

But the transformation using distributivity explodes the formula.

What is the complexity blow-up in converting an arbitrary formula to CNF?

Exercise 7.7 Give an example of a propositional formula with $2n$ variables whose conversion to CNF produces a formula with 2^n clauses.

Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?

Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?
- $CNF(F_2) = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2)$

Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?
- $CNF(F_2) = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2)$
- Now consider $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$. What is $CNF(F_n)$?

Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?
- $CNF(F_2) = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2)$
- Now consider $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$. What is $CNF(F_n)$?
- How many clauses does F_n have?

Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?
- $CNF(F_2) = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2)$
- Now consider $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$. What is $CNF(F_n)$?
- How many clauses does F_n have? 2^n!

Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?
- $CNF(F_2) = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2)$
- Now consider $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$. What is $CNF(F_n)$?
- How many clauses does F_n have? 2^n!

Is there a way to avoid the explosion?

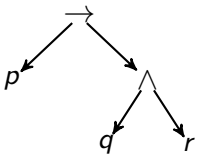
Conversion to CNF (contd.)

- Consider $F_2 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$. What is $CNF(F_2)$?
- $CNF(F_2) = (x_1 \vee x_2) \wedge (x_1 \vee y_2) \wedge (y_1 \vee x_2) \wedge (y_1 \vee y_2)$
- Now consider $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$. What is $CNF(F_n)$?
- How many clauses does F_n have? 2^n!

Is there a way to avoid the explosion? Yes!

Tseitin's encoding

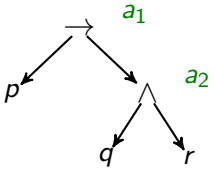
Tseitin's encoding



Consider $F = (p \rightarrow (q \wedge r))$

1. Write the parse tree of F.

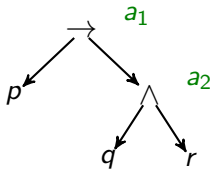
Tseitin's encoding



Consider $F = (p \rightarrow (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\rightarrow, \wedge$

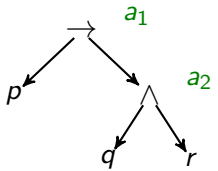
Tseitin's encoding



Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$

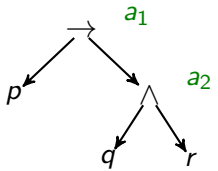
Tseitin's encoding



Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$

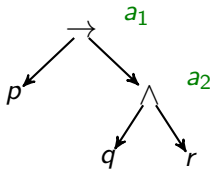
Tseitin's encoding



Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$
5. Each constraint can be written as CNF with just 3-4 clauses:
   - $a_1 \leftrightarrow (p \to a_2) \equiv (a_1 \vee p) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg p \vee a_2)$
   - $a_2 \leftrightarrow (q \wedge r) \equiv (\neg a_2 \vee q) \wedge (\neg a_2 \vee r) \wedge (a_2 \vee \neg q \vee \neg r)$

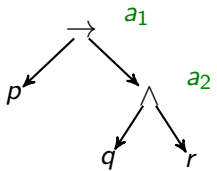
Tseitin's encoding



Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$
5. Each constraint can be written as CNF with just 3-4 clauses:
   - $a_1 \leftrightarrow (p \to a_2) \equiv (a_1 \vee p) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg p \vee a_2)$
   - $a_2 \leftrightarrow (q \wedge r) \equiv (\neg a_2 \vee q) \wedge (\neg a_2 \vee r) \wedge (a_2 \vee \neg q \vee \neg r)$
6. Note: G is in CNF!

Tseitin's encoding

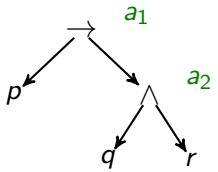


Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$
5. Each constraint can be written as CNF with just 3-4 clauses:
   - $a_1 \leftrightarrow (p \to a_2) \equiv (a_1 \vee p) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg p \vee a_2)$
   - $a_2 \leftrightarrow (q \wedge r) \equiv (\neg a_2 \vee q) \wedge (\neg a_2 \vee r) \wedge (a_2 \vee \neg q \vee \neg r)$
6. Note: G is in CNF!

What is the relation between F and G? Are they equivalent?

Tseitin's encoding



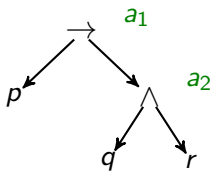
Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$
5. Each constraint can be written as CNF with just 3-4 clauses:
   - $a_1 \leftrightarrow (p \to a_2) \equiv (a_1 \vee p) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg p \vee a_2)$
   - $a_2 \leftrightarrow (q \wedge r) \equiv (\neg a_2 \vee q) \wedge (\neg a_2 \vee r) \wedge (a_2 \vee \neg q \vee \neg r)$
6. Note: G is in CNF!

What is the relation between F and G? Are they equivalent?

- No! Not even same set of variables.

Tseitin's encoding



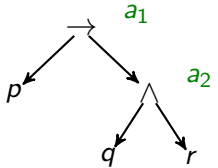
Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$
5. Each constraint can be written as CNF with just 3-4 clauses:
   - $a_1 \leftrightarrow (p \to a_2) \equiv (a_1 \vee p) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg p \vee a_2)$
   - $a_2 \leftrightarrow (q \wedge r) \equiv (\neg a_2 \vee q) \wedge (\neg a_2 \vee r) \wedge (a_2 \vee \neg q \vee \neg r)$
6. Note: G is in CNF!

What is the relation between F and G? Are they equivalent?

- No! Not even same set of variables. But they are equisatisfiable: F is satisfiable iff G is.

Tseitin's encoding



Consider $F = (p \to (q \wedge r))$

1. Write the parse tree of F.
2. Introduce a fresh "auxiliary" variable for each internal node, e.g., a_1, a_2 for $\to, \wedge$
3. Add constraints to define these variables: e.g., $a_1 \leftrightarrow (p \to a_2)$, $a_2 \leftrightarrow (q \wedge r)$
4. Conjunct them and add root: $G = (a_1 \leftrightarrow (p \to a_2)) \wedge (a_2 \leftrightarrow (q \wedge r)) \wedge (a_1)$
5. Each constraint can be written as CNF with just 3-4 clauses:
   - $a_1 \leftrightarrow (p \to a_2) \equiv (a_1 \vee p) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg p \vee a_2)$
   - $a_2 \leftrightarrow (q \wedge r) \equiv (\neg a_2 \vee q) \wedge (\neg a_2 \vee r) \wedge (a_2 \vee \neg q \vee \neg r)$
6. Note: G is in CNF!

What is the relation between F and G? Are they equivalent?

- No! Not even same set of variables. But they are equisatisfiable: F is satisfiable iff G is.

Theorem: The Tseitin algo converts any formula into an equisatisfiable CNF formula.

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$.

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$. (Small optimization: *root* doesn't really need its own variable, since nothing refers to it – we can assert its defining formula directly instead.)

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$. (Small optimization: *root* doesn't really need its own variable, since nothing refers to it – we can assert its defining formula directly instead.)
- So we have $3n + 1$ clauses only (instead of 2^n)!

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$. (Small optimization: *root* doesn't really need its own variable, since nothing refers to it – we can assert its defining formula directly instead.)
- So we have $3n + 1$ clauses only (instead of 2^n)!
- Number of variables gone up to $3n$ from $2n$.

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$. (Small optimization: *root* doesn't really need its own variable, since nothing refers to it – we can assert its defining formula directly instead.)
- So we have $3n + 1$ clauses only (instead of 2^n)!
- Number of variables gone up to $3n$ from $2n$.
- Exponential reduction in size at cost of new variables!

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$. (Small optimization: *root* doesn't really need its own variable, since nothing refers to it – we can assert its defining formula directly instead.)
- So we have $3n + 1$ clauses only (instead of 2^n)!
- Number of variables gone up to $3n$ from $2n$.
- Exponential reduction in size at cost of new variables!

Theorem

Tseitin Encoding always leads to an at most linear blowup!

How much do we improve using Tseitin?

Consider again $F_n = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

- With Tseitin's encoding we need
  - n auxiliary variables for each $\wedge$ gate.
  - Each gives a constraint of the form $a_i \leftrightarrow (x_i \wedge y_i)$ which can be written in CNF with just 3 constraints.
  - 1 more constraint (not a new variable!) to directly assert the root: $(a_1 \vee \ldots \vee a_n)$. (Small optimization: *root* doesn't really need its own variable, since nothing refers to it – we can assert its defining formula directly instead.)
- So we have $3n + 1$ clauses only (instead of 2^n)!
- Number of variables gone up to $3n$ from $2n$.
- Exponential reduction in size at cost of new variables!

Theorem

Tseitin Encoding always leads to an at most linear blowup!

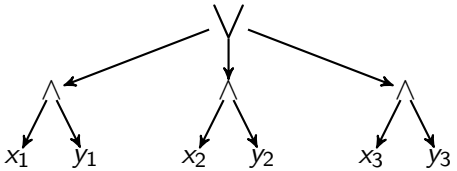
Proof: Exercise!

Tseitin encoding of F_3

$F_3 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_3 \wedge y_3)$ – 6 variables.

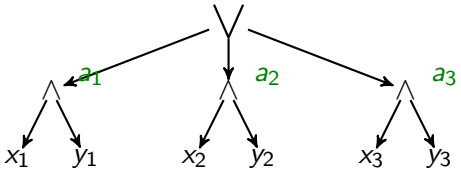
Tseitin encoding of F_3

$F_3 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_3 \wedge y_3)$ – 6 variables.



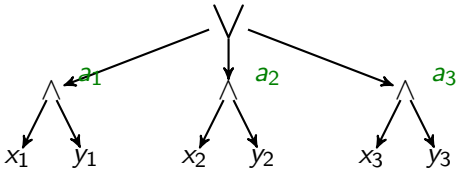
Tseitin encoding of F_3

$F_3 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_3 \wedge y_3)$ – 6 variables.



Tseitin encoding of F_3

$F_3 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_3 \wedge y_3)$ – 6 variables.

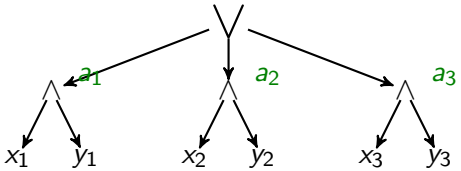


Tseitin encoding G_3

$G_3 = (\neg a_1 \vee x_1) \wedge (\neg a_1 \vee y_1) \wedge (a_1 \vee \neg x_1 \vee \neg y_1)$
$\wedge(\neg a_2 \vee x_2) \wedge (\neg a_2 \vee y_2) \wedge (a_2 \vee \neg x_2 \vee \neg y_2)$
$\wedge(\neg a_3 \vee x_3) \wedge (\neg a_3 \vee y_3) \wedge (a_3 \vee \neg x_3 \vee \neg y_3)$
$\wedge(a_1 \vee a_2 \vee a_3)$

Tseitin encoding of F_3

$F_3 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_3 \wedge y_3)$ – 6 variables.



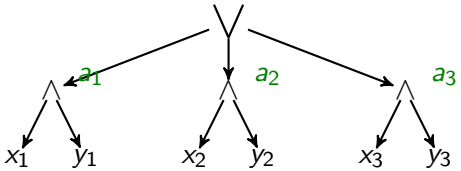
Tseitin encoding G_3

$G_3 = (\neg a_1 \vee x_1) \wedge (\neg a_1 \vee y_1) \wedge (a_1 \vee \neg x_1 \vee \neg y_1)$
$\wedge(\neg a_2 \vee x_2) \wedge (\neg a_2 \vee y_2) \wedge (a_2 \vee \neg x_2 \vee \neg y_2)$
$\wedge(\neg a_3 \vee x_3) \wedge (\neg a_3 \vee y_3) \wedge (a_3 \vee \neg x_3 \vee \neg y_3)$
$\wedge(a_1 \vee a_2 \vee a_3)$

9 variables, 10 clauses – matches $3n + 1 = 10$ for $n = 3$.

Tseitin encoding of F_3

$F_3 = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee (x_3 \wedge y_3)$ – 6 variables.



Tseitin encoding G_3

$$G_3 = (\neg a_1 \vee x_1) \wedge (\neg a_1 \vee y_1) \wedge (a_1 \vee \neg x_1 \vee \neg y_1)$$
$$\wedge (\neg a_2 \vee x_2) \wedge (\neg a_2 \vee y_2) \wedge (a_2 \vee \neg x_2 \vee \neg y_2)$$
$$\wedge (\neg a_3 \vee x_3) \wedge (\neg a_3 \vee y_3) \wedge (a_3 \vee \neg x_3 \vee \neg y_3)$$
$$\wedge (a_1 \vee a_2 \vee a_3)$$

9 variables, 10 clauses – matches $3n + 1 = 10$ for $n = 3$. (If instead we added one extra variable a_4 for the root, we'd need $n + 1$ more clauses: 4 for $n = 3$, giving $14 = 4n + 2$.)

Topic 7.5

Disjunctive normal form

Disjunctive normal form (DNF)

Definition

A formula is in DNF if it is a disjunction of conjunctions of literals.

Disjunctive normal form (DNF)

Definition

A formula is in DNF if it is a disjunction of conjunctions of literals.

Examples: $(p \wedge q) \vee (\neg p \wedge r \wedge s)$ is in DNF, but $(p \vee q) \wedge r$ is not.

Disjunctive normal form (DNF)

Definition

A formula is in DNF if it is a disjunction of conjunctions of literals.

Examples: $(p \wedge q) \vee (\neg p \wedge r \wedge s)$ is in DNF, but $(p \vee q) \wedge r$ is not.

Theorem

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

Proof.

Proof is similar to CNF. □

Disjunctive normal form (DNF)

Definition

A formula is in DNF if it is a disjunction of conjunctions of literals.

Examples: $(p \wedge q) \vee (\neg p \wedge r \wedge s)$ is in DNF, but $(p \vee q) \wedge r$ is not.

Theorem

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

Proof.

Proof is similar to CNF. □

Exercise 7.8

a. Give the formal grammar of DNF

b. Give a linear time algorithm to prove satisfiability of a DNF formula

Disjunctive normal form (DNF)

Definition

A formula is in DNF if it is a disjunction of conjunctions of literals.

Examples: $(p \wedge q) \vee (\neg p \wedge r \wedge s)$ is in DNF, but $(p \vee q) \wedge r$ is not.

Theorem

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

Proof.

Proof is similar to CNF. □

Exercise 7.8

a. Give the formal grammar of DNF

b. Give a linear time algorithm to prove satisfiability of a DNF formula

So, CNF makes validity easy, DNF makes satisfiability easy.