Logic and theory of computation

3rd lecture

01-10-2018

Conjunctive normal form (CNF)

A *literal* is either an atom or a negation of an atom.

Conjunctive normal form (CNF)

A formula is in *conjunctive normal form (CNF)* iff it is a conjunction of disjunctions of literals.

Example:

$(\neg p \lor q \lor r) \land (\neg q \lor r) \land (\neg r)$	CNF
$(\neg p \lor q \lor r) \land ((p \land \neg q) \lor r) \land (\neg r)$	not in CNF
$(\neg p \lor q \lor r) \land \neg(\neg q \lor r) \land (\neg r)$	not in CNF

Theorem

For every formula in propositional logic there is a logically equivalent formula in CNF.

Conjunctive normal form (CNF)

Transformation steps

1st step: Eliminate implications $A \to B \equiv \neg A \vee B$

2nd step: Using De Morgan's laws and double negation law the formula can be tranformed into a formula with the property that negations can occur only right before the atoms
$\neg(A \wedge B) \equiv \neg A \vee \neg B, \quad \neg(A \vee B) \equiv \neg A \wedge \neg B$ and $\neg\neg A \equiv A$,

3rd step: Now, the formula is literals connected by $\wedge$ and $\vee$. Use distributive laws to eliminate conjunctions within disjunctions or vica versa.
$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ and
$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$

Example:

$$\begin{aligned}(\neg p \to \neg q) \to (p \to q) &\equiv \neg(\neg\neg p \vee \neg q) \vee (\neg p \vee q)\\ &\equiv (\neg\neg\neg p \wedge \neg\neg q) \vee (\neg p \vee q)\\ &\equiv (\neg p \wedge q) \vee (\neg p \vee q)\\ &\equiv (\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q).\end{aligned}$$

Clausal form

Clausal form is another representation of a CNF.

- A *clause* is a set of literals. Example: $\{\neg q, \neg p, q\}$.
- A clause is considered to be an implicit disjunction of its literals. Example: $\{\neg q, \neg p, q\}$ is $\neg q \vee \neg p \vee q$.
- A *unit clause* is a clause consisting of exactly one literal. Example: $\{\neg q\}$.
- The empty set of literals is the *empty clause*, denoted by $\square$.
- A formula in *clausal form* is a set of clauses. Example: $\{\{p, r\}, \{\neg q, \neg p, q\}\}$.
- A formula is considered to be an implicit conjunction of its clauses. Example: $(p \vee r) \wedge (\neg q \vee \neg p \vee q)$ for the previous one.
- The formula that is the empty set of clauses is denoted by $\emptyset$.

Removing trivial clauses

Corollary

Every formula in propositional logic can be transformed into an logically equivalent formula in clausal form.

multiple occurrences of literals and clauses $\Rightarrow$ single occurance equivalent because of idempotence ($A \vee A \equiv A,\ A \wedge A \equiv A$)

Example:

Formula: $(p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

Clausal form: $\{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$

A clause if *trivial* if it contains a pair of clashing literals.

Let S be a set of clauses and let $C \in S$ be a trivial clause. Then $S - \{C\}$ is logically equivalent to S.

True, because of $A \vee \top \equiv \top$ and $A \wedge \top \equiv A$. So we can delete trivial clauses.

Empty clause and the empty set of clauses

Proposition

$\square$ (empty clause) is unsatisfiable. $\emptyset$ (the empty set of clauses) is valid.

Proof: A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true.

Let I be an arbitrary interpretation. Since there are no literals in $\square$, there are no literals whose value is true under I.

But I was an arbitrary interpretation, so is unsatisfiable.

A set of clauses is valid iff every clause in the set is true in every interpretation.

But there are no clauses in $\emptyset$ that need to be true, so $\emptyset$ is valid.

A short notation for clausal form

Notation

The set delimiters { and } are removed from each clause and a negated literal is denoted by a bar over the atomic proposition.

Let CNF(A) and cf(A) denote a CNF and a clausal form for a formula A, respectively.

Example:

$A = (p \vee r) \wedge (q \rightarrow \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

$\mathrm{CNF}(A) = (p \vee r) \wedge (\neg q \vee \neg p \vee q) \wedge (p \vee \neg p \vee q \vee p \vee \neg p) \wedge (r \vee p)$

$\mathrm{cf}(A) = \{\{p, r\}, \{\neg q, \neg p, q\}, \{p, \neg p, q\}\}$ which becomes

$\mathrm{cf}(A) = \{pr, \bar{q}\bar{p}q, p\bar{p}q\}$ using this shorter notation.

Logical consequence in propositional logic

Equivalent forms of logical consequence

Let $U = \{A_1, \ldots A_n\}$ be a finite set of formulas and let B be a formula. Then the following are equivalent.

- $\{A_1, \ldots A_n\} \models B$
- $\{A_1 \wedge \cdots \wedge A_n\} \models B$
- $\models A_1 \wedge \cdots \wedge A_n \rightarrow B$
- $\models \neg A_1 \vee \cdots \vee \neg A_n \vee B$
- $\models A_1 \rightarrow (A_2 \rightarrow (\cdots (A_n \rightarrow B) \cdots))$
- $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ unsatisfiable
- $\{A_1, \ldots, A_n, \neg B\}$ unsatisfiable
- $\{\mathrm{CNF}(A_1), \ldots, \mathrm{CNF}(A_n), \mathrm{CNF}(\neg B)\}$ unsatisfiable
- $\mathrm{cf}(A_1) \cup \cdots \cup \mathrm{cf}(A_n) \cup \mathrm{cf}(\neg B)$ unsatisfiable

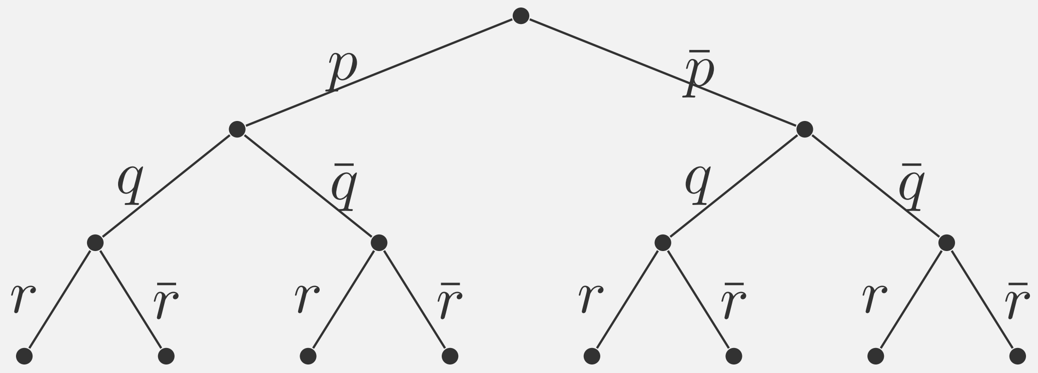
Semantic tree

A semantic tree is a data structure for recording assignments of T and F to the atoms of a formula in the process of searching for a model.

Semantic tree

Let S be a set of clauses and let $\mathcal{P}_S = \{p_1, \ldots, p_n\}$ be the set of atoms appearing in S. $\mathcal{T}$, the semantic tree for S, is a complete binary tree of depth n such that for $1 \leq i \leq n$, every left-branching edge from a node at depth $i-1$ is labeled p_i and every right-branching edge is labeled by $\overline{p_i}$.

Example: a semantic tree for $S = \{p, \bar{p}q, \bar{r}, \bar{p}\bar{q}r\}$. $\mathcal{P}_S = \{p, q, r\}$,



Semantic tree

Closed/open semantic tree

Each branch b from the root to a leaf in a semantic tree $\mathcal{T}$ is labeled by a sequence of literals $\{\ell_1, \ldots, \ell_n\}$, where $\ell_i = p_i$ or $\ell_i = \overline{p_i}$.

Each branch b defines an interpretation I_b as follows

$$\begin{aligned} I_b(p_i) &= T, && \text{if } \ell_i = p_i \\ I_b(p_i) &= F && \text{if } \ell_i = \overline{p_i} \end{aligned}$$

Closed/open branch

A branch b is *closed* if $v_b(S) = F$, otherwise b is *open*.

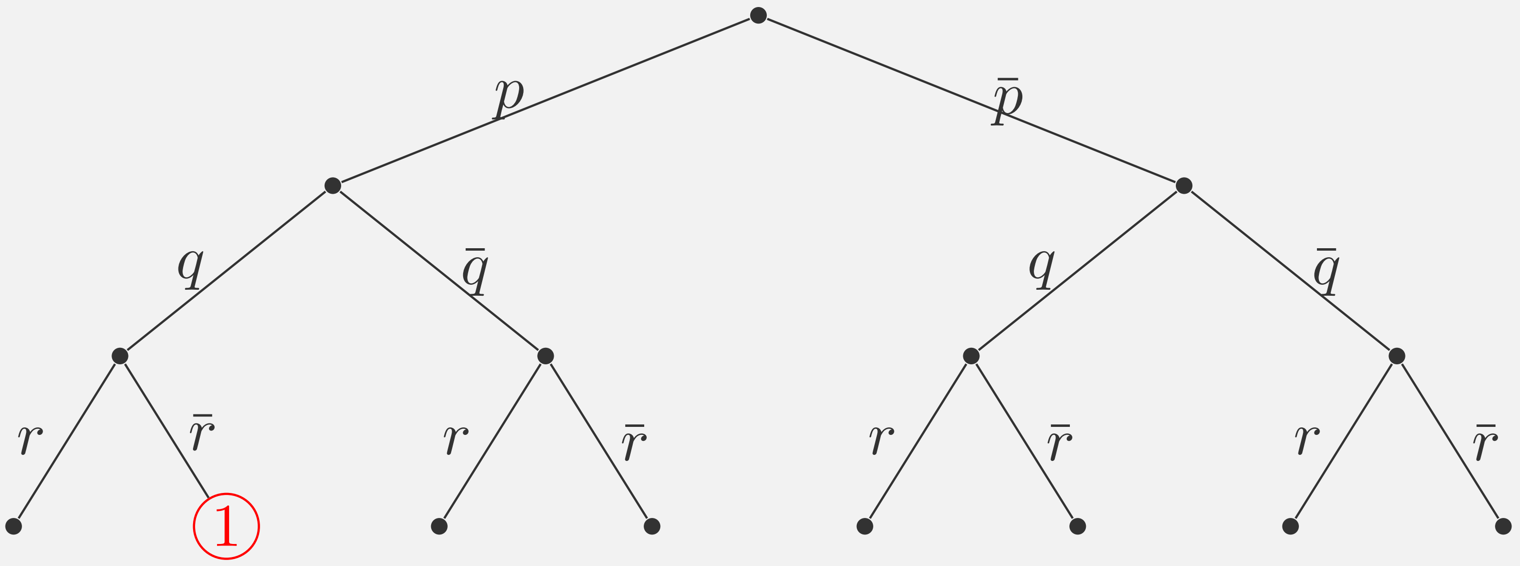
Closed/open semantic tree

$\mathcal{T}$ is closed if all branches are *closed*, otherwise $\mathcal{T}$ is *open*.

Semantic tree

Example

Example: Semantic tree for $S = \{p, \bar{p}q, \bar{r}, \bar{p}\bar{q}r\}$.



Branch b ending in the leaf labeled 1 defines the interpretation: $I_b(p) = T, I_b(q) = T, I_b(r) = F$.
For the interpretation I_b: $v(\bar{p}\bar{q}r) = F$, so $v(S) = F$ holds, too. This means, that the branch b is closed. One can check that, the same holds for the other 7 interpretations as well. Therefore the semantic tree is closed.

Partial interpretation

Every interpretation I for S corresponds to I_b for some branch b in $\mathcal{T}$, and conversely, every I_b is an interpretation for S.

Theorem

The semantic tree $\mathcal{T}$ for a set of clauses S is closed if and only if the set S is unsatisfiable.

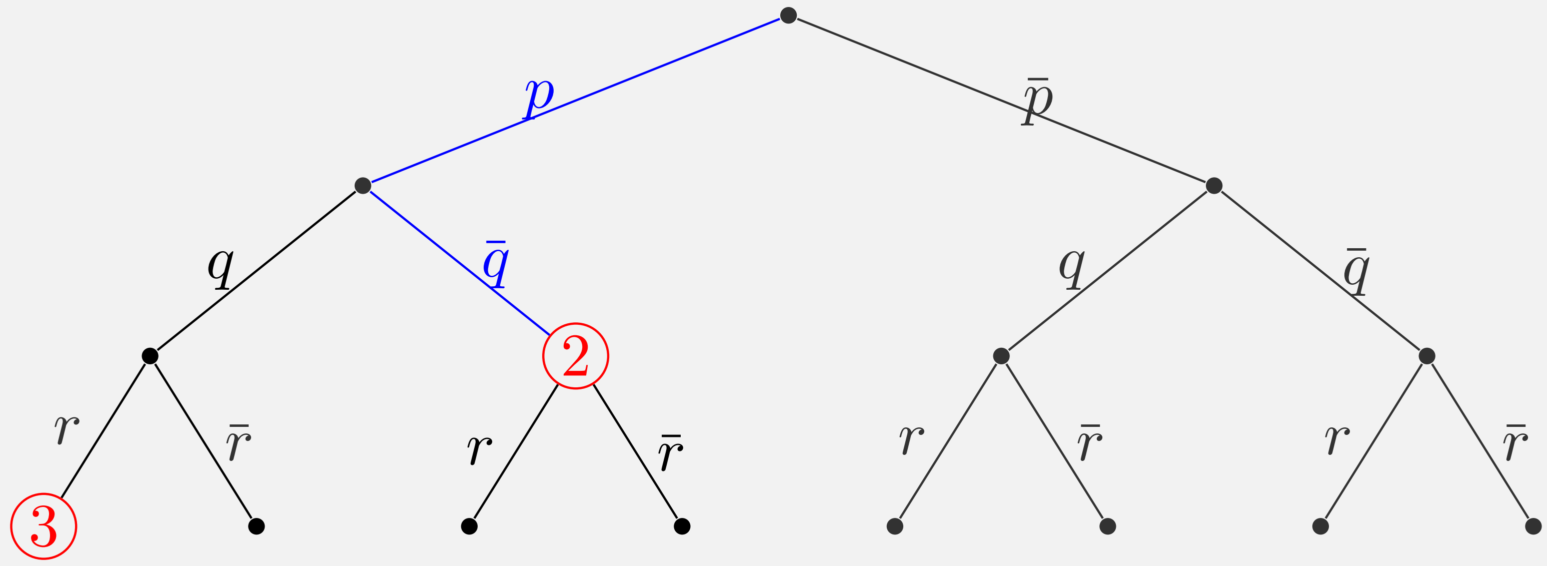
Traversing a branch top-down, a partial branch from the root to a node represents a *partial interpretation*. Partial interpretations might be sufficient to evaluate some clauses to F concluding that in any extension of that partial interpretation to a full interpretation the entire set of clauses is false.

In a closed semantic tree, there must be such a node on every branch. However, if a clause contains the literal labeling the edge to a leaf, a (full) interpretation may be necessary to falsify the clause.

Failure node

Example

Example: Semantic tree for $S = \{p, \bar{p}q, \bar{r}, \bar{p}\bar{q}r\}$.



The partial branch $b_{p\bar{q}}$ from the root to node ② defines a partial interpretation $I(p) = T, I(q) = F$, which falsifies the clause $\bar{p}q$, thus the entire set S for the full interpretation $I(p) = T, I(q) = F, I(r) = T$ and $I(p) = T, I(q) = F, I(r) = F$.

On the other hand falsifying ③ needs a full interpretation.

Associating clauses with nodes

Failure node

Let $\mathcal{T}$ be a closed semantic tree for a set of clauses S and let b be a branch in $\mathcal{T}$. The node in b closest to the root which falsifies S is a *failure node*.

Clause associated with a node

A clause falsified by a failure node is a clause associated with the node.

More than one clause might be associated with a failure node.

Associating clauses with nodes

For C to be falsified at a failure node n, all the literals in C must be assigned F in the partial interpretation corresponding to n.

Lemma

A clause C associated with a failure node n is a subset of the complements of the literals appearing on the partial branch b from the root to n.

Resolution rule

Definition and example

If ℓ is a literal, let ℓ^c denote its complementary pair.

Resolution rule

Let C_1, C_2 be clauses such that $\ell \in C_1, \ell^c \in C_2$. The clauses C_1, C_2 are said to be *clashing clauses* and to *clash on the complementary pair of literals* ℓ, ℓ^c . C, the resolvent of C_1 and C_2 , is the clause:

$$\mathrm{Res}(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\}).$$

C_1 and C_2 are the parent clauses of C.

Example: $C_1 = ab\bar{c}$ and $C_2 = bc\bar{e}$

They clash on the pair of complementary literals $c, \bar{c}$.

$\mathrm{Res}(C_1, C_2) = (ab\bar{c} - \{\bar{c}\}) \cup (bc\bar{e} - \{c\}) = ab \cup b\bar{e} = ab\bar{e}$.

Recall that a clause is a set so duplicate literals are removed when taking the union.

Resolution

Case of more than one pair of clashing clauses

Resolution is only performed if the pair of clauses clash on exactly one pair of complementary literals due to the following.

Observation

If two clauses clash on more than one literal, their resolvent is a trivial clause.

Remark: It is not strictly incorrect to perform resolution on such clauses, but since trivial clauses contribute nothing to the satisfiability or unsatisfiability of a set of clauses, we agree to delete them from any set of clauses and not to perform resolution on clauses with two clashing pairs of literals.

Resolution procedure

Observations

Lemma

Let I be an interpretation. If $I \vDash \{C_1, C_2\}$ then $I \vDash \mathrm{Res}(C_1, C_2)$. If $I \vDash \mathrm{Res}(C_1, C_2)$, then I can be extended to $\hat{I}$, such that $\hat{I} \vDash \{C_1, C_2\}$.

Corollary

The resolvent C is satisfiable if and only if the parent clauses C_1 and C_2 are both satisfiable.

Corollary

Let S be a set of clauses and let $C_1, C_2 \in S$ be a pair of clashing clauses. Then S is satisfiable if and only if $S \cup \{\mathrm{Res}(C_1, C_2)\}$ is satisfiable.

If a set of clauses S contains $\Box$ then S is unsatisfiable.

Resolution procedure

Algorithm

Let $\binom{S}{2}$ denote the set of 2-element subsets of S.

Algorithm Resolution procedure(S)

```
1:  while there is an unmarked pair of $\binom{S}{2}$ do
2:      choose an unmarked pair $\{C_1, C_2\}$ of $\binom{S}{2}$ and mark it
3:      if $\{C_1, C_2\}$ is a clashing pair of clauses then
4:          $C \leftarrow \mathrm{Res}(C_1, C_2)$
5:          if $C = \Box$ then
6:              return 'S is unsatisfiable'
7:          else
8:              if $C$ is not the trivial clause then
9:                  $S \leftarrow S \cup \{C\}$
10: return 'S is satisfiable'
```

Resolution procedure

Example

Consider the set of clauses

$$S = \{(1)p, (2)\bar{p}q, (3)\bar{r}, (4)\bar{p}\bar{q}r\},$$

where the clauses have been numbered. Here is a resolution derivation of $\square$ from S, where the justification for each line is the pair of the numbers of the parent clauses that have been resolved to give the resolvent clause:

(5)	$\bar{p}\bar{q}$	Res((3),(4))
(6)	$\bar{p}$	Res((5),(2))
(7)	$\square$	Res((6),(1))

Soundness and completeness of resulution

Remark: If S is finite Resolution procedure(S) terminates, since the size of S is bounded.

Proof of Completeness of resolution By our previous lemma if the input S is satisfiable then S remains satifiable during the procedure, so S can not contain $\Box$ at the end of the procedure.

Inference node

An inner node n of a semantic tree is called an inference node if both of its children are failure nodes.

Lemma 1

Let $\mathcal{T}$ be a closed semantic tree for a set of clauses S. If there are at least two failure nodes in $\mathcal{T}$, then there is at least one inference node.

Soundness and completeness of resulution

Lemma 2

Let $\mathcal{T}$ be closed semantic tree and let n be an inference node whose children n_1 and n_2 of n are (by definition) failure nodes with clauses C_1 and C_2 associated with them, respectively. Then C_1, C_2 clash and the partial interpretation defined by the branch from the root to n falsifies their resolvent.

Lemma 3

Let n be an inference node, $C_1, C_2 \in S$ clauses associated with the failure nodes that are the children of n, and C their resolvent. Then $S \cup \{C\}$ has a failure node that is either n or an ancestor of n and C is a clause associated with the failure node.

Soundness and completeness of resulution

Proof of Completeness of resolution If S is an unsatisfiable set of clauses, then there is a closed semantic tree $\mathcal{T}$ for S.
If S is unsatisfiable and does not already contain $\Box$ then there must be at least two failure nodes in $\mathcal{T}$, so there is at least one inference node in $\mathcal{T}$.
An application of the resolution rule at the inference node adds the resolvent to the set, creating a failure node and deleting two failure nodes, thus decreasing the number of failure nodes.
When the number of failure nodes has decreased to one, it must be the root which is associated with the derivation of the empty clause by the resolution rule.

Theorem

Resolution procedure is sound and complete.

Corollary

Resolution procedure is a decision procedure for satisfiability.