Logic and Sets 3

disjunctive and conjunctive normal forms

adequate systems of connectives

satisfiability

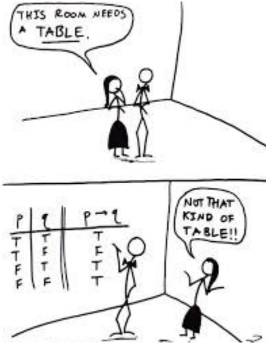
Propositional logic: The story so far

- propositional logic formulas
  - truth tables for $\neg$, $\wedge$, $\vee$, $\oplus$, $\rightarrow$, $\leftrightarrow$
- semantic entailment $\models$
  - $\phi_1, \ldots, \phi_n \models \psi$
- metalogic statements
  - $\phi \wedge \psi$ is a tautology $\iff$ ϕ and ψ are tautologies
- logic puzzles
  - on an island with liars and truth speakers

Today

- disjunctive normal form (DNF)
  - turning a truth table into DNF
- adequate systems of connectives
  - $\{\neg, \vee\}$
  - Sheffer stroke $|$
- conjunctive normal form (CNF)
  - turning a truth table into CNF
  - rewriting a formula into CNF
  - tautology test for CNF
- satisfiability
  - solving Sudoku puzzles
  - DPLL procedure for CNF

Disjunctive normal form



Expressive power of propositional logic

Theorem

Propositional logic is functionally complete:

Each truth table can be represented by a propositional formula.

Darling, does your Swiss army knife have a garlic press?

Extracting formulas from truth tables

As an example, we construct a formula that corresponds to the following truth table.

This formula is true exactly on the rows marked by $\Leftarrow$.

p	q	r	$?$		
T	T	T	F		
T	T	F	F		
T	F	T	F		
T	F	F	T	$\Leftarrow$	$p \wedge \neg q \wedge \neg r$
F	T	T	T	$\Leftarrow$	$\neg p \wedge q \wedge r$
F	T	F	F		
F	F	T	F		
F	F	F	T	$\Leftarrow$	$\neg p \wedge \neg q \wedge \neg r$

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

Extracting formulas from truth tables

Truth table of $\underbrace{(p \wedge \neg q \wedge \neg r)}_{\psi_1} \vee \underbrace{(\neg p \wedge q \wedge r)}_{\psi_2} \vee \underbrace{(\neg p \wedge \neg q \wedge \neg r)}_{\psi_3}$

p	q	r	?	$p \wedge \neg q \wedge \neg r$	$\neg p \wedge q \wedge r$	$\neg p \wedge \neg q \wedge \neg r$	$\psi_1 \vee \psi_2 \vee \psi_3$
T	T	T	F	F	F	F	F
T	T	F	F	F	F	F	F
T	F	T	F	F	F	F	F
T	F	F	T	T	F	F	T
F	T	T	T	F	T	F	T
F	T	F	F	F	F	F	F
F	F	T	F	F	F	F	F
F	F	F	T	F	F	T	T

Formulas corresponding to a truth table constructed in this way are of a special syntactic shape: disjunctive normal form (DNF).

Disjunctive normal form

Definition

- A literal is a propositional variable or the negation of a propositional variable: $p, q, \neg q, \neg r, s, \ldots$
- A *disjunctive normal form* is a *disjunction* $\psi_1 \vee \cdots \vee \psi_n$ where the formulas ψ_i are *conjunctions* of literals.

- $\underbrace{(p \wedge \neg q \wedge \neg r)}_{\psi_1} \vee \underbrace{(\neg p \wedge q \wedge r)}_{\psi_2} \vee \underbrace{(\neg p \wedge \neg q \wedge \neg r)}_{\psi_3}$
- $\underbrace{(\neg q \wedge r)}_{\psi_1} \vee \underbrace{(p \wedge q \wedge \neg s)}_{\psi_2} \vee \underbrace{\neg r}_{\psi_3}$
  - ‘conjunction’ ψ_3 consists of just one single literal $\neg r$
- $\underbrace{p \wedge q \wedge \neg s}_{\psi_1}$
  - one single conjunction ψ_1 of literals

Disjunctive normal form

Question

Why don't we need to write brackets around disjunctions, and only a single pair of brackets around conjunctions ?

Answer: Disjunction and conjunction are both *associative*.

Question

Is the formula p in DNF ?

Answer: Yes !

The 'disjunction' consists of a single 'conjunction', which in turn consists of a single literal.

Extracting DNF-formulas from known truth tables

Extraction of DNF's from truth tables yields semantic equivalences.

A DNF for $p \to q$

p	q	$p \to q$		
T	T	T	$\Leftarrow$	$p \wedge q$
T	F	F		
F	T	T	$\Leftarrow$	$\neg p \wedge q$
F	F	T	$\Leftarrow$	$\neg p \wedge \neg q$

This produces the DNF $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$.

So we find the semantic equivalence

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

DNF-extraction from truth tables

Question

Which DNF is extracted from the truth table of $p \wedge q$?

Answer: The formula itself, $p \wedge q$.

Question

Which DNF is extracted from the truth table of $p \vee q$?

Answer: $(p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge q)$

So we find the semantic equivalence

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

DNF-extraction from truth tables of contradictions

Let us try to extract a DNF from the truth table of $p \wedge \neg p$.

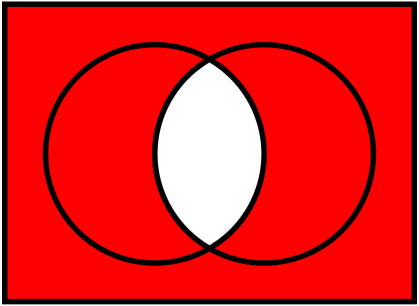
p	$p \wedge \neg p$
T	F
F	F

In this case DNF-extraction as described does not succeed, since there are no T's.

Here one can stipulate as the result of the extraction process a contradiction $q \wedge \neg q$, which is a DNF.

Or we can nominate $\perp$, a logical constant denoting *false* which will be introduced later on, to be an "empty" DNF.

Adequate systems of connectives



Adequate systems of connectives

Definition

A system $\mathcal{C}$ of connectives is *adequate* if *every* truth table can be expressed as a formula with connectives in $\mathcal{C}$.

- The DNF corresponding to a given truth table contains only the connectives $\neg$, $\wedge$, $\vee$.

  Hence $\{\neg, \wedge, \vee\}$ is an *adequate* system of connectives.

- Actually the system $\{\neg, \wedge\}$ is already adequate.
  - Because we have $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$.
  - So all occurrences of $\vee$ can be eliminated from a formula:

    Replace subformulas $\phi \vee \psi$ by $\neg(\neg\phi \wedge \neg\psi)$.

- Likewise, the system $\{\neg, \vee\}$ is adequate.
  - Because we have $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$.

Examples

Express $(p \land \neg q) \lor (\neg p \land q)$ in the system $\{\neg, \land\}$

Transform this formula using $\phi \lor \psi \equiv \neg(\neg\phi \land \neg\psi)$.

$$\underbrace{(p \land \neg q)}_{\phi} \lor \underbrace{(\neg p \land q)}_{\psi} \quad \rightsquigarrow \quad \neg(\neg\underbrace{(p \land \neg q)}_{\phi} \land \neg\underbrace{(\neg p \land q)}_{\psi})$$

Express $(p \land \neg q) \lor (\neg p \land q)$ in the system $\{\neg, \lor\}$

Transform this formula using $\phi \land \psi \equiv \neg(\neg\phi \lor \neg\psi)$.

$$\begin{aligned} \underline{(p \land \neg q)} \lor (\neg p \land q) \quad &\rightsquigarrow \quad \neg(\neg p \lor \neg\neg q) \lor \underline{(\neg p \land q)} \\ &\rightsquigarrow \quad \neg(\neg p \lor \neg\neg q) \lor \neg(\neg\neg p \lor \neg q) \end{aligned}$$

Only negation or conjunction

Question

Is negation $\neg$ by itself an adequate system ?

Answer: No, it can only express negation and identity (by a double negation) on a single propositional variable.

Question

Is conjunction $\wedge$ by itself an adequate system ?

Answer: No, since substituting $\top$ for all propositional variables will always yield $\top$. So e.g. $\neg p$ cannot be expressed.

Sheffer stroke

The *Sheffer stroke* $|$ forms an adequate system by itself.

The meaning of $\phi \mid \psi$ is: not both ϕ and ψ.

$$\phi \mid \psi \equiv \neg(\phi \wedge \psi)$$

It is also called the NAND-operation.

Henry M. Sheffer

(1882-1964)

Truth table of $\phi \mid \psi$

ϕ	ψ	$\phi \mid \psi$
T	T	F
T	F	T
F	T	T
F	F	T

Expressing $\neg$ and $\vee$ with the Sheffer stroke

Question

What does $\phi \mid \phi$ mean ?

Answer: $\neg\phi$

Question

How can $\phi \vee \psi$ be expressed using only $\mid$ and $\neg$?

Hint: $\phi \vee \psi \quad \equiv \quad \neg(\neg\phi \wedge \neg\psi)$

Answer: $\neg\phi \mid \neg\psi$

Question

How can $\phi \vee \psi$ be expressed using only $\mid$?

Answer: $(\phi \mid \phi) \mid (\psi \mid \psi)$

Sheffer stroke on its own is adequate

$\{\,|\,\}$ is an adequate system.

We show this by expressing the connectives of the known adequate system $\{\neg, \vee\}$ using only $|$.

$\neg\phi \equiv \phi \mid \phi \qquad \phi \vee \psi \equiv \neg\phi \mid \neg\psi \equiv (\phi \mid \phi) \mid (\psi \mid \psi)$

ϕ	$\phi \mid \phi$	$\neg\phi$
T	F	F
F	T	T

ϕ	ψ	$\phi \mid \phi$	$\psi \mid \psi$	$(\phi \mid \phi) \mid (\psi \mid \psi)$	$\phi \vee \psi$
T	T	F	F	T	T
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	F	F

Expressing $\wedge$ with the Sheffer stroke

Question

Express $\phi \wedge \psi$ using only $|$.

Hint: Use $\phi \wedge \psi \equiv \neg\neg(\phi \wedge \psi)$.

Answer:

$$\phi \wedge \psi \equiv \neg\neg(\phi \wedge \psi) \equiv \neg(\phi \mid \psi) \equiv (\phi \mid \psi) \mid (\phi \mid \psi)$$

Recapitulating: Adequate systems of connectives

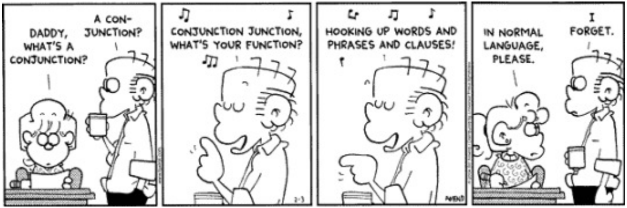
Definition

A system $\mathcal{C}$ of connectives is *adequate* if every truth table can be expressed as a formula with connectives in $\mathcal{C}$.

- $\{\neg, \wedge, \vee\}$ is an adequate system of connectives
- $\{\neg, \wedge\}$ is adequate
- $\{\neg, \vee\}$ is adequate
- $\{\,|\,\}$ is adequate

So the *NAND-gate* is sufficient to build all hardware circuits.

Conjunctive normal form



Conjunctive normal form

A *conjunctive normal form (CNF)* is analogous to a DNF, but with the roles of $\wedge$ and $\vee$ reversed.

Definition

- A *clause* is a disjunction of literals.
- A *conjunctive normal form* is a *conjunction* $\psi_1 \wedge \cdots \wedge \psi_n$ where the formulas ψ_i are *clauses*.

Examples

- $(p \vee r) \;\wedge\; (p \vee \neg q) \qquad = \psi_1 \wedge \psi_2$
- $(\neg q \vee r) \;\wedge\; (p \vee q \vee \neg s) \;\wedge\; \neg r \qquad = \psi_1 \wedge \psi_2 \wedge \psi_3$
- $p \vee q \vee \neg s \qquad = \psi_1$

Conjunctive normal form

Parse tree of a CNF

- only connectives $\wedge, \vee, \neg$
- $\neg$ occurs only directly above a propositional variable
- $\wedge$ never occurs below $\vee$

Challenge

Given ϕ, find a CNF ψ such that $\phi \equiv \psi$.

Method I: via a truth table.

Method II: stepwise transformation with algorithm $\mathtt{CNF}$.

Truth table method for CNF

A CNF representing this truth table should express that we are not in one of the three rows marked by $\Leftarrow$

p	q	r	ϕ		
T	T	T	T		
T	T	F	F	$\Leftarrow$	$\neg p \vee \neg q \vee r$
T	F	T	T		
T	F	F	F	$\Leftarrow$	$\neg p \vee q \vee r$
F	T	T	T		
F	T	F	T		
F	F	T	T		
F	F	F	F	$\Leftarrow$	$p \vee q \vee r$

Solution: $(\neg p \vee \neg q \vee r) \;\wedge\; (\neg p \vee q \vee r) \;\wedge\; (p \vee q \vee r)$

Transformation method to CNF using algorithm CNF

The algorithm CNF transforms formulas ϕ built from $\neg, \wedge, \vee, \rightarrow$ to a CNF in 3 steps:

1. IMPL-FREE: Eliminate all occurrences of $\rightarrow$.

$$\phi \rightarrow \psi \ \equiv \ \neg\phi \vee \psi$$

2. NNF: Bring all occurrences of $\neg$ inside, until they are all directly in front of a propositional variable.

$$\neg(\phi \wedge \psi) \ \equiv \ \neg\phi \vee \neg\psi \qquad\qquad \neg\neg\phi \ \equiv \ \phi$$

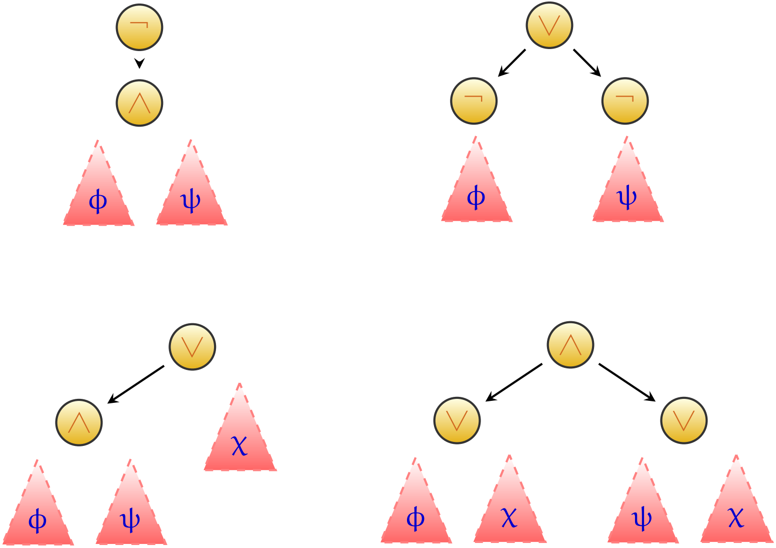
$$\neg(\phi \vee \psi) \ \equiv \ \neg\phi \wedge \neg\psi$$

3. DISTR: Use distributivity to re-arrange $\vee$ and $\wedge$.

$$(\phi \wedge \psi) \ \vee \ \chi \ \equiv \ (\phi \vee \chi) \ \wedge \ (\psi \vee \chi)$$

$$\chi \ \vee \ (\phi \wedge \psi) \ \equiv \ (\chi \vee \phi) \ \wedge \ (\chi \vee \psi)$$

Transformation `CNF` on parse trees



Example transformation CNF

Application of algorithm CNF to $p \to \neg(p \to q)$

$$
\begin{array}{lll}
p \to \neg(\underline{p \to q}) & \rightsquigarrow_{\texttt{IMPL-FREE}} & \underline{p \to \neg(\overline{\neg p \lor q})} \\
 & \rightsquigarrow_{\texttt{IMPL-FREE}} & \overline{\neg p \lor \underline{\neg(\neg p \lor q)}} \\
 & \rightsquigarrow_{\texttt{NNF}} & \neg p \lor \overline{(\underline{\neg\neg p} \land \neg q)} \\
 & \rightsquigarrow_{\texttt{NNF}} & \underline{\neg p \lor (\overline{p} \land \neg q)} \\
 & \rightsquigarrow_{\texttt{DISTR}} & \overline{(\neg p \lor p) \land (\neg p \lor \neg q)}
\end{array}
$$

Tautology test for CNFs

A simple criterion determines whether a CNF is a tautology.

We need two *metalogic* observations:

1. A conjunction is a tautology *if and only if* all its conjuncts are tautologies:

$$\models \psi_1 \wedge \ldots \wedge \psi_n \quad \Longleftrightarrow \quad \models \psi_1 \text{ and } \ldots \text{ and } \models \psi_n$$

2. A clause, i.e., a disjunction of literals, is a tautology *if and only if* it contains literals p and $\neg p$ for some p.

Examples:

$\models p \vee \neg q \vee \neg p \vee r$

but $\not\models p \vee \neg q \vee r$

Tautology test for CNFs

Criterion

A CNF $\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n$ is a tautology if and only if each *clause* ψ_i contains literals p and $\neg p$ for some p.

Example 1

$(p \vee \neg q \vee \neg p) \wedge (\neg p \vee \neg q \vee q) \wedge (\neg p \vee p)$ is a tautology.

The 1st and 3rd clause have the pair $p, \neg p$.

The 2nd clause has the pair $q, \neg q$.

Example 2

$(p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee p) \wedge (\neg q \vee q)$ is *not* a tautology.

Question: Which valuation returns $\mathtt{F}$ for this formula ?

Satisfiability

Satisfiability problem

Satisfiability problem: Given a propositional formula ϕ, try to find a valuation that applied to ϕ yields $\mathtt{T}$.

Many challenges can be represented as a satisfiability problem:

- planning in Artificial Intelligence
- automatic test pattern generation
- equivalence checking in circuit design
- theorem proving
- inversion attacks on cryptographic hash functions
- identification of haplotypes in bioinformatics
- *Sudoku puzzles*

Satisfiability problem is NP-complete

The satisfiability problem is NP-complete:

No efficient general solution for this problem has yet been found.

It is like looking for a needle in a vast haystack of valuations.

Because a formula with n variables has 2^n possible valuations.

There are $\approx 2^{270}$ atoms in the universe.

Finding an efficient solution (or proving none exists) would earn you a million dollar from the Clay Mathematics Institute, which declared it one of 7 *Millennium Problems*.

SAT solvers

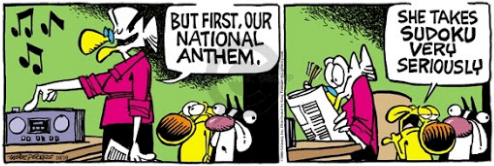
Still, algorithms exist to efficiently solve the satisfiability problem for *most* (but not all) propositional formulas.

Efficient software has been developed for solving satisfiability, called *SAT solvers*.

SAT solvers can for instance quickly solve *Sudoku puzzles*.

The number of possible (mostly wrong) solutions for a Sudoku puzzle far exceeds the number of atoms in the universe.

So when your granny solves one, show her some respect !



Sudoku from the World Championship 2008

Every row, column and 3×3 block should contain each number from 1 to 9 exactly once.

						4	6	
					3			9
	8	4			7			1
2						7	1	
5		3	8					
9			1			5	3	
	2	5			8			6
					2	9	8	5
					5			7

In 2008 the world champion needed 2.5 minutes to solve it.

Encoding Sudoku as a satisfiability problem

For $i, j, k \in \{1, \ldots, 9\}$ we use $9^3 = 729$ variables p_{ijk} such that:

p_{ijk} is true $\iff$ the field in row i, column j contains the number k.

Question

Express in a propositional formula:

- the position at row 5 and column 3 contains the number 2
- 2 is on this position *if and only if* none of the other numbers is on this position

Answer: p_{532}

$$p_{532} \leftrightarrow \neg p_{531} \wedge \neg p_{533} \wedge \neg p_{534} \wedge \ldots \wedge \neg p_{539}$$

Encoding Sudoku as a satisfiability problem

For $i, j, k \in \{1, \ldots, 9\}$ we use $9^3 = 729$ variables p_{ijk} such that:

$$p_{ijk} \text{ is true} \iff \text{the field in row } i, \text{ column } j \text{ contains the number } k.$$

We express the Sudoko rules in propositional logic:

1. In every field precisely one number k. 729 formulas like:

$$p_{127} \leftrightarrow \neg p_{121} \wedge \neg p_{122} \wedge \ldots \wedge \neg p_{126} \wedge \neg p_{128} \wedge \neg p_{129}$$

2. In every row precisely one number k. 729 formulas like:

$$p_{127} \leftrightarrow \neg p_{117} \wedge \neg p_{137} \wedge \neg p_{147} \wedge \ldots \wedge \neg p_{197}$$

3. In every column precisely one number k. 729 formulas like:

$$p_{127} \leftrightarrow \neg p_{227} \wedge \neg p_{327} \wedge \ldots \wedge \neg p_{927}$$

4. In every 3×3 subgrid precisely one number k. 729 formulas like:

$$p_{127} \leftrightarrow \neg p_{117} \wedge \neg p_{137} \wedge \neg p_{217} \wedge \neg p_{227} \wedge \neg p_{237} \wedge \neg p_{317} \wedge \neg p_{327} \wedge \neg p_{337}$$

5. Number k is given to start with, at the field in row i and column j.

   Typically fewer than 30 formulas of the form p_{ijk}.

Sudoku from the World Championship 2008

7	9	2	5	8	1	**4**	**6**	3
6	5	1	2	4	**3**	8	7	**9**
3	**8**	**4**	9	6	**7**	2	5	**1**
2	4	6	3	5	9	**7**	**1**	8
5	1	**3**	**8**	7	4	6	9	2
9	7	8	**1**	2	6	**5**	**3**	4
1	**2**	**5**	7	9	**8**	3	4	**6**
4	3	7	6	1	**2**	**9**	**8**	**5**
8	6	9	4	3	**5**	1	2	**7**

SAT solver yices needs 0.015 seconds.

Davis-Putnam-Logemann-Loveland procedure (1962)

The DPLL procedure, which checks satisfiability of formulas in *CNF*, is a key ingredient of SAT solvers. *(AI course Intelligent Systems)*

Atomic propositions $\top$ ('constant true') and $\bot$ ('constant false')

$\top$ and $\bot$ are *constants* (i.e., have no arguments), where:

- under every valuation, $\top$ has truth value $\mathtt{T}$
- under every valuation, $\bot$ has truth value $\mathtt{F}$

The DPLL procedure applies the following reduction rules to eliminate connectives with $\top$ or $\bot$:

$$
\begin{array}{lll}
\psi \wedge \top \rightsquigarrow \psi & \psi \vee \top \rightsquigarrow \top & \neg\top \rightsquigarrow \bot \\
\top \wedge \psi \rightsquigarrow \psi & \top \vee \psi \rightsquigarrow \top & \neg\bot \rightsquigarrow \top \\
\psi \wedge \bot \rightsquigarrow \bot & \psi \vee \bot \rightsquigarrow \psi & \\
\bot \wedge \psi \rightsquigarrow \bot & \bot \vee \psi \rightsquigarrow \psi &
\end{array}
$$

DPLL procedure

We check the satisfiability of a CNF ϕ.

- eliminate "unit" clauses:
  - for each *unit clause* p of ϕ, replace all p's in ϕ by $\top$
  - for each *unit clause* $\neg p$ of ϕ, replace all p's in ϕ by $\bot$
- eliminate "pure" propositional variables:
  - if a p occurs only *positively* in ϕ, replace all p's in ϕ by $\top$
  - if a p occurs only *negatively* in ϕ, replace all p's in ϕ by $\bot$
- if ϕ becomes $\top$, then it is *satisfiable*
- if ϕ becomes $\bot$, then it is *unsatisfiable*
- else choose a p in ϕ:
  - replace all p's in ϕ by $\top$, and apply the DPLL procedure
  - if the outcome is "unsatisfiable", then replace all p's in ϕ by $\bot$, and *again* apply the DPLL procedure

DPLL procedure

Question

Why may the DPLL procedure take a long time to terminate ?

Note: Elimination of unit clauses and of pure propositional variables does *not* involve *backtracking*.

Question

Suppose the DPLL procedure finds a formula ϕ is satisfiable.

How can a valuation be determined that applied to ϕ yields $\mathtt{T}$?

Question

Apply the DPLL procedure to check the satisfiability of:

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

Take home

- disjunctive normal form (DNF)
  - construction of a DNF via a truth table
- adequate systems of connectives
  - $\{\neg, \vee\}$ $\{\neg, \wedge\}$
  - $\{\mid\}$ with Sheffer stroke $\mid$
- conjunctive normal form (CNF)
  - construction via a truth table
  - rewriting a formula into CNF
  - tautology test for CNF
- satisfiability
  - encoding Sudoku in propositional logic
  - DPLL procedure for CNF

Next week's lecture consists of question hours for logic and sets in preparation of the midterm exam