

Logical equivalence

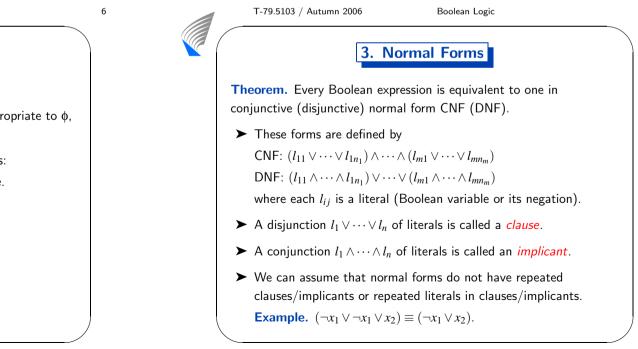
Definition. Expressions ϕ_1 and ϕ_2 are logically *equivalent* ($\phi_1 \equiv \phi_2$) iff for all truth assignments *T* appropriate to both of them,

$$T \models \phi_1$$
 iff $T \models \phi_2$.

Example.

$$\begin{split} (\phi_1 \lor \phi_2) &\equiv (\phi_2 \lor \phi_1) \\ ((\phi_1 \land \phi_2) \land \phi_3) &\equiv (\phi_1 \land (\phi_2 \land \phi_3)) \\ \neg \neg \phi &\equiv \phi \\ ((\phi_1 \land \phi_2) \lor \phi_3) &\equiv ((\phi_1 \lor \phi_3) \land (\phi_2 \lor \phi_3)) \\ \neg (\phi_1 \land \phi_2) &\equiv (\neg \phi_1 \lor \neg \phi_2) \\ (\phi_1 \lor \phi_1) &\equiv \phi_1 \end{split}$$







How to interpret Boolean expressions?

 Boolean expressions are propositions that are either true or false.
 They speak about a world where certain atomic proposition (Boolean variables) are either true or false.

This induces truth values for Boolean expressions as follows.

- ➤ A truth assignment T is mapping from a finite subset X' ⊂ X to the set of truth values {true, false}.
- ► Let $X(\phi)$ be the set of Boolean variables appearing in ϕ . **Definition.** A truth assignment $T : X' \to \{$ **true**, **false** $\}$ is appropriate to ϕ if $X(\phi) \subseteq X'$.

C 2006 TKK, Laboratory for Theoretical Computer Science

Boolean Logic

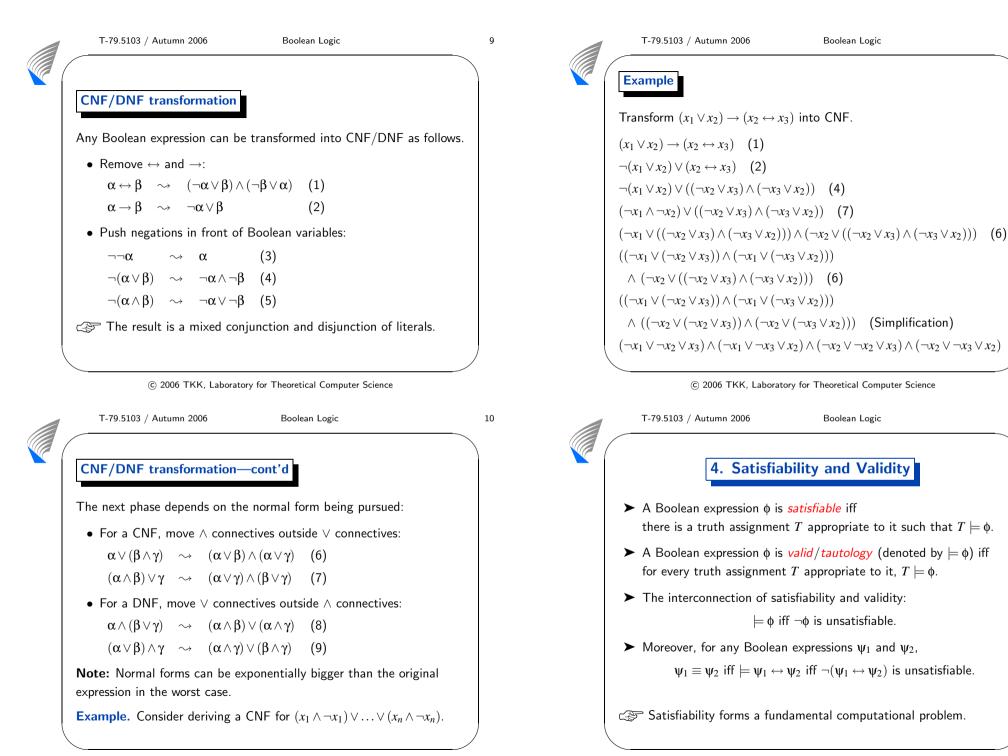
A	

Satisfaction relation

T-79.5103 / Autumn 2006

- ► Let a truth assignment $T : X' \to \{ true, false \}$ be appropriate to ϕ , i.e., $X(\phi) \subseteq X'$.
- ► $T \models \phi$ (*T* satisfies ϕ) is defined inductively as follows: If ϕ is a variable from *X'*, then $T \models \phi$ iff $T(\phi) =$ **true**. If $\phi = \neg \phi_1$, then $T \models \phi$ iff $T \nvDash \phi_1$. If $\phi = \phi_1 \land \phi_2$, then $T \models \phi$ iff $T \models \phi_1$ and $T \models \phi_2$.
 - If $\phi = \phi_1 \lor \phi_2$, then $T \models \phi$ iff $T \models \phi_1$ or $T \models \phi_2$.

Example. Let $T(x_1) =$ true, $T(x_2) =$ false. Then $T \models x_1 \lor x_2$ but $T \not\models (x_1 \lor \neg x_2) \land (\neg x_1 \land x_2)$. 8



12

^{© 2006} TKK, Laboratory for Theoretical Computer Science

14

Satisfiability Problem

- SAT problem: Given φ in CNF, is φ satisfiable?
 Example. (x₁ ∨ ¬x₂) ∧ ¬x₁ is satisfiable
 but (x₁ ∨ ¬x₂) ∧ ¬x₁ ∧ x₂ is unsatisfiable.
- > SAT can be solved in $O(n^2 2^n)$ time (e.g., truth table method).
- ▶ SAT \in **NP** but SAT \in **P** remains open!

A nondeterministic Turing machine for $\varphi \in SAT$: for all variables x in φ do choose nondeterministically: T(x) := true or T(x) := false; if $T \models \varphi$ then return "yes" else return "no"

C 2006 TKK, Laboratory for Theoretical Computer Science

Boolean Logic

1 C	

Horn clauses

T-79.5103 / Autumn 2006

- An interesting special case of SAT concerns *Horn clauses*, i.e., clauses (disjunction of literals) with *at most one positive literal*.
 Example. ¬x₁ ∨ x₂ ∨ ¬x₃ and ¬x₁ ∨ ¬x₃, x₂ are Horn clauses but ¬x₁ ∨ x₂ ∨ x₃ is not.
- A Horn clause with a positive literal is called an *implication* and can be written as (x₁ ∧ x₃) → x₂
 (or → x₂ when there are no negative literals).
- ► HORNSAT problem:

Given a conjunction of Horn clauses, is it satisfiable?

Boolean Logic



Algorithm *hornsat*(S) /* Determines whether $S \in \text{HORNSAT }^*/$ $T := \emptyset /^* T$ is the set of true atoms $^*/$ **repeat if** there is an implication $(x_1 \land x_2 \land \dots \land x_n) \rightarrow y$ in S such that $\{x_1, \dots, x_n\} \subseteq T$ but $y \notin T$ **then** $T := T \cup \{y\}$ **until** T does not change **if** for all purely negative clauses $\neg x_1 \lor \dots \lor \neg x_n$ in S, there is some literal $\neg x_i$ such that $x_i \notin T$ **then** return S is satisfiable **else** return S is not satisfiable $\blacksquare \mathsf{FORNSAT} \in \mathbf{P}.$

© 2006 TKK, Laboratory for Theoretical Computer Science

5. Boolean Functions and Expressions 5. Boolean function is a mapping {true, false}ⁿ → {true, false}. Fxample. The connectives ∨, ∧, →, and ↔ can be viewed as binary Boolean functions and ¬ is a unary function. Similarly, any Boolean expression ¢ can be interpreted as an *n*-ary Boolean function f_{\$\phi\$} where n = |X(\$\phi\$)|. A Boolean expression \$\phi\$ with variables x₁,...,x_n expresses the *n*-ary function f if for any n-tuple of truth values t = (t₁,...,t_n), f(t) = { true, if T ⊨ \$\phi\$. false, if T ⊭ \$\phi\$. where T satisfies T(x_i) = t_i for every i = 1,...,n.

 $x_1 \quad x_2$

0 0 0

0

1 0 1

1 1 0

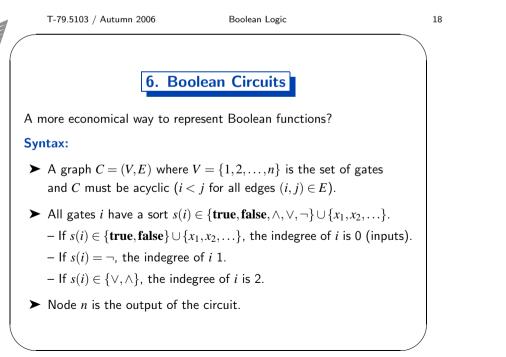
1 1

 $\phi_f = (\neg x_1 \land x_2) \lor$

 $(x_1 \wedge \neg x_2).$

- ➤ The idea: model the rows of the truth table giving true as a disjunction of conjunctions.
 Example.
- ► Let F be the set of all *n*-tuples $\mathbf{t} = (t_1, \dots, t_n)$ with $f(\mathbf{t}) = \mathbf{true}$.
- For each **t**, let D_t be a conjunction of literals x_i if $t_i =$ **true** and $\neg x_i$ if $t_i =$ **false**.
- \blacktriangleright Let $\phi_f = \bigvee_{\mathbf{t} \in F} D_{\mathbf{t}}$
- ➤ Note that \$\overline{\phi_f\$}\$ may get big in the worst case: O(n2ⁿ).
- Solution Not all Boolean functions can be expressed concisely.

C 2006 TKK, Laboratory for Theoretical Computer Science



Semantics

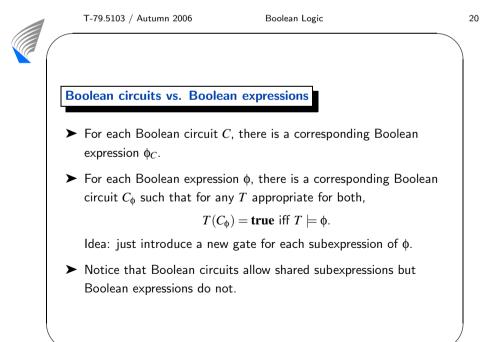
17

A truth assignment is a function $T: X(C) \rightarrow \{ true, false \}$ where X(C) is the set of variables appearing in a circuit C.

The truth value T(i) for each gate *i* is defined inductively:

- If s(i) =true, T(i) =true and if s(i) =false, T(i) =false.
- If $s(i) \in X(C)$, then T(i) = T(s(i)).
- If s(i) = ¬, then T(i) = true if T(j) = false, otherwise T(i) = false where (j,i) is the unique edge entering i.
- If s(i) = ∧, then T(i) = true if T(j) = T(j') = true else
 T(i) = false where (j,i) and (j',i) are the two edges entering i.
- If $s(i) = \lor$, then T(i) = **true** if T(j) = **true** or T(j') = **true** else T(i) = **false** where (j,i) and (j',i) are the two edges to *i*.
- T(C) = T(n), i.e. the value of the circuit C.

C 2006 TKK, Laboratory for Theoretical Computer Science



T-79.5103 / Autumn 2006

Computational problems related with Boolean circuits

► CIRCUIT SAT:

Given a circuit *C*, is there a truth assignment $T: X(C) \rightarrow \{ \text{true}, \text{false} \}$ such that T(C) = true?

- ► CIRCUIT SAT \in NP.
- ► CIRCUIT VALUE:

Given a circuit *C* with no variables, is it the case that T(C) =true?

► CIRCUIT VALUE \in **P**.

(No truth assignment is needed as $X(C) = \emptyset$).

C 2006 TKK, Laboratory for Theoretical Computer Science

Boolean Logic

T-79.5103 / Autumn 2006

22

Circuits computing Boolean functions

- A Boolean circuit with variables x_1, \ldots, x_n computes an *n*-ary Boolean function *f* if for any *n*-tuple of truth values $\mathbf{t} = (t_1, \ldots, t_n), f(\mathbf{t}) = T(C)$ where $T(x_i) = t_i$ for $i = 1, \ldots, n$.
- ➤ Any *n*-ary Boolean function *f* can be computed by a Boolean circuit involving variables x₁,...,x_n.
- ► Not every Boolean function has a concise circuit computing it.

Theorem. For any $n \ge 2$ there is an *n*-ary Boolean function f such that no Boolean circuit with $\frac{2^n}{2n}$ or fewer gates can compute it.

However, all natural families of Boolean functions seem to need only a linear number of gates to compute!

Learning Objectives

- You should deeply understand the syntax and semantics of Boolean expressions — including their use in practice.
- The relationship/difference between Boolean expressions and circuits.
- Knowing the idea of representing Boolean functions in terms of Boolean expressions and circuits.
- ➤ Four computational problems related with Boolean logic and circuits: SAT, HORNSAT, CIRCUIT SAT, and CIRCUIT VALUE.

C 2006 TKK, Laboratory for Theoretical Computer Science