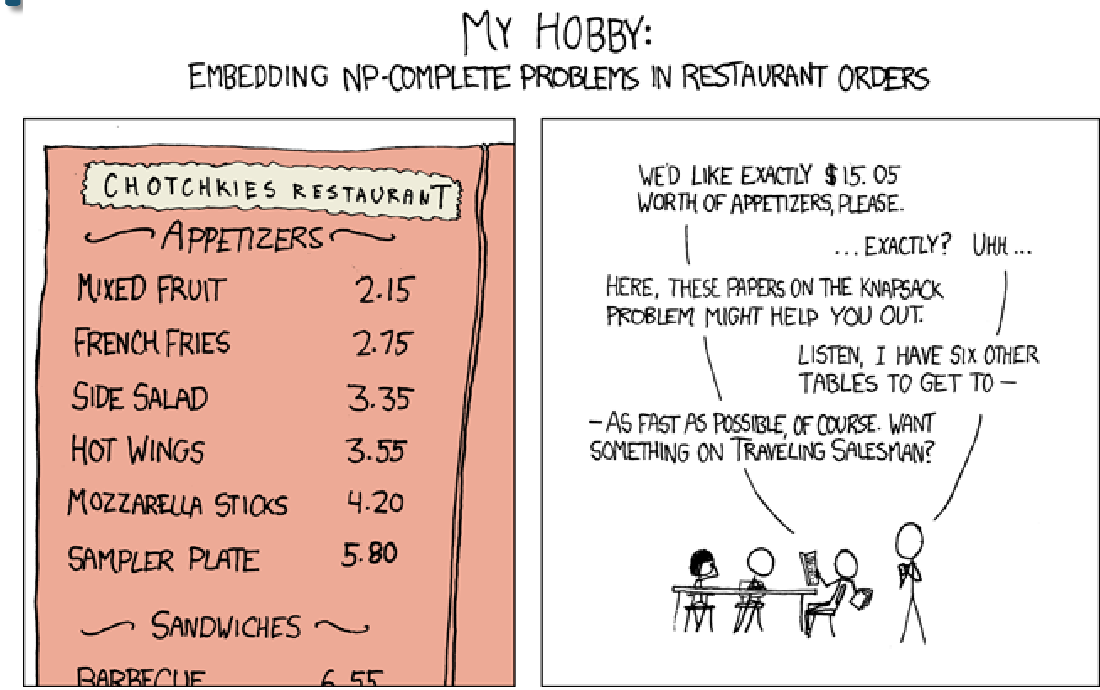
Logic and Computability

Lecture 9

Combinational Equivalence Checking

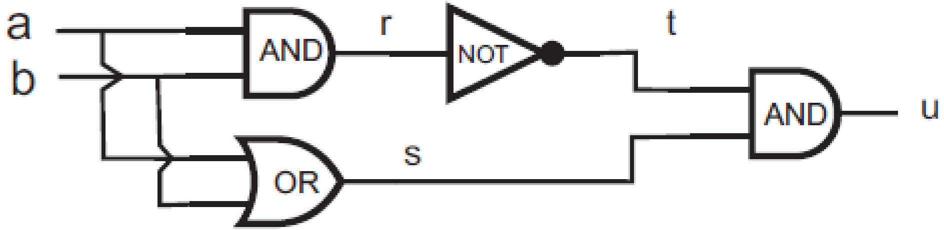


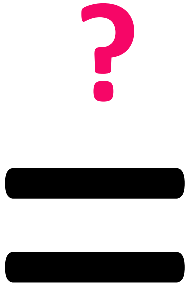
Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at

https://xkcd.com/287/

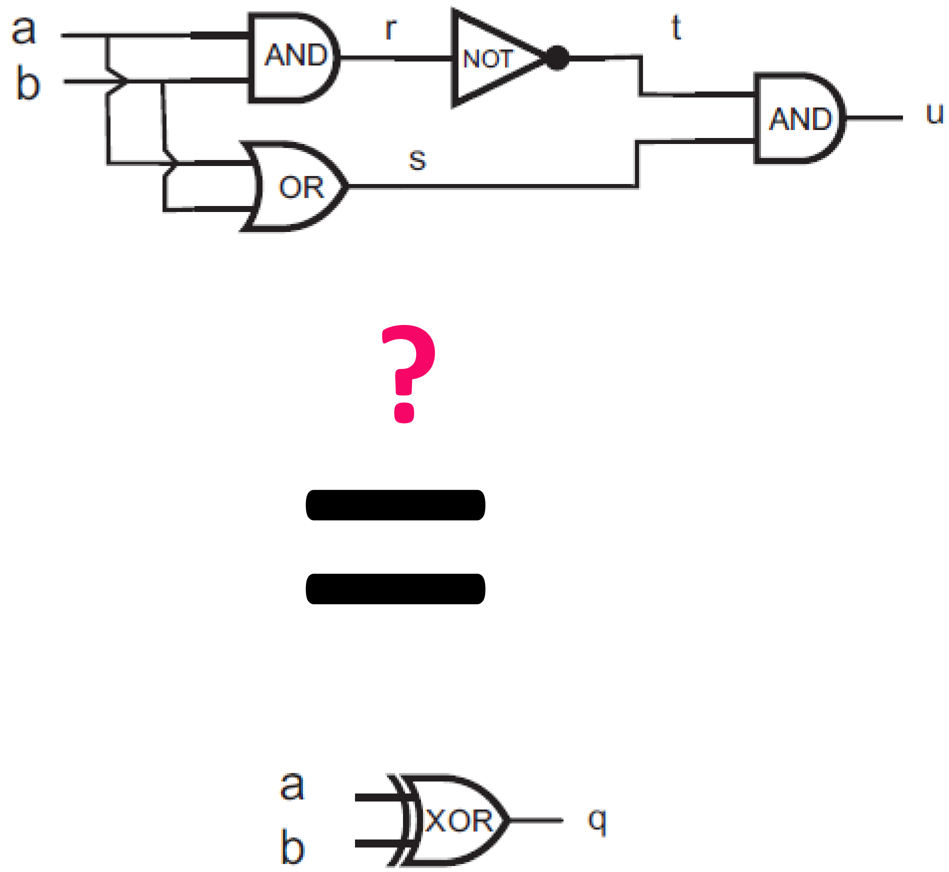
Motivation – Equivalence Checking





Motivation – Equivalence Checking

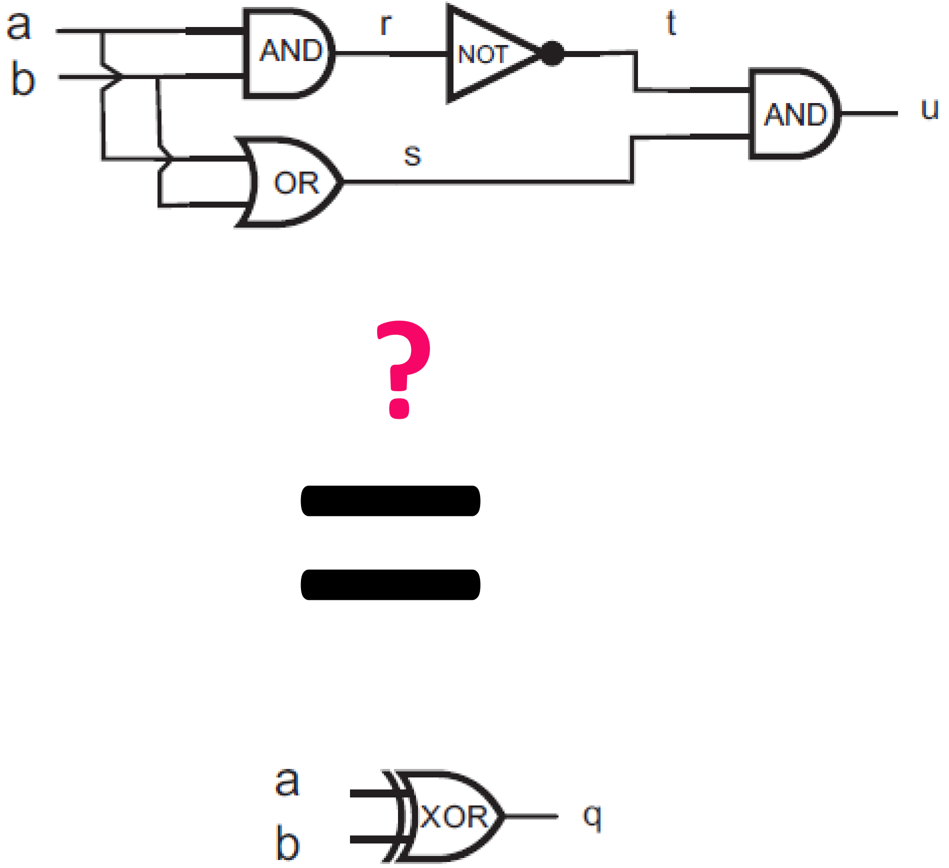
- Circuit Optimization and Synthesis Tools
  - Big Market
  - Tools can make mistakes!
  - Need to check for equivalence



Motivation – Equivalence Checking

- Circuit Optimization and Synthesis Tools
  - Big Market
  - Tools can make mistakes!
  - Need to check for equivalence

- Gives us a context to discuss basic topics
  - Normal Forms (CNF, DNF)
  - Relations between
    - Satisfiability
    - Validity
    - Semantic Entailment
    - Equivalence
  - Tseitin Encoding



Outline

- Algorithm - Decide equivalence of combinational circuits
  - Based on reduction to Satisfiability
- Translation of a Circuit into a Formula
- Relations between Satisfiability, Validity, Equivalence and Semantic Entailment
- Normal Forms
- Tseitin Encoding

Algorithm - Circuit Equivalence via Truth Tables

- **Using Truth Tables**: Check for $\phi \vDash \psi$ and $\psi \vDash \phi$?
  i.e., ϕ and ψ are true for the same models
  - Exponentially large
  - → Not practicable!

Algorithm - Circuit Equivalence via Truth Tables

- Using Truth Tables: Check for $\phi \vDash \psi$ and $\psi \vDash \phi$?

  i. e., ϕ and ψ are true for the same models
  - Exponentially large
  - → Not practicable!

- Using **Natural Deduction**: Check for $\phi \vdash \psi$ and $\psi \vdash \phi$?

  $\text{i. e., From } \phi \text{ we can prove } \psi$ and vice versa
  - Hard to automate (efficiently)
  - → Not practicable!

Algorithm - Circuit Equivalence via Truth Tables

- Using Truth Tables: Check for $\phi \vDash \psi$ and $\psi \vDash \phi$?

  i. e., ϕ and ψ are true for the same models

  - Exponentially large
  - → Not practicable!

- Using **Natural Deduction**: Check for $\phi \vdash \psi$ and $\psi \vdash \phi$?

  i. e., From ϕ we can prove ψ and vice versa

  - Hard to automate (efficiently)
  - → Not practicable!

- Better way: Reduction to SAT

Algorithm - Circuit Equivalence based on SAT

1. Encode C_1 and C_2 into two formulas φ_1 and φ_2

Algorithm - Circuit Equivalence based on SAT

1. Encode C_1 and C_2 into two formulas φ_1 and φ_2

2. Compute the Conjunctive Normal Form (CNF) of $\varphi_1 \oplus \varphi_2$
   - Use Tseitin Encoding

Algorithm - Circuit Equivalence based on SAT

1. Encode C_1 and C_2 into two formulas φ_1 and φ_2
2. Compute the Conjunctive Normal Form (CNF) of $\varphi_1 \oplus \varphi_2$
   - Use Tseitin Encoding
3. Give CNF($\varphi_1 \oplus \varphi_2$) to a **SAT solver**

Algorithm - Circuit Equivalence based on SAT

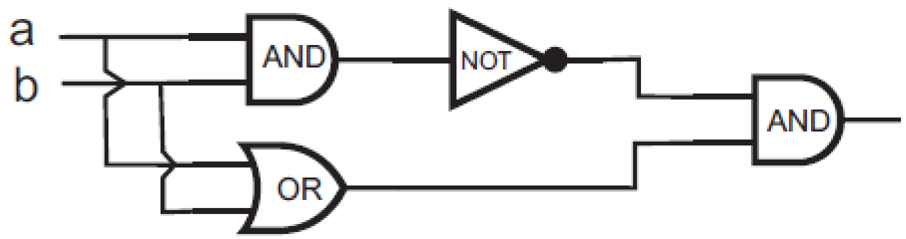
1. Encode C_1 and C_2 into two formulas φ_1 and φ_2

2. Compute the Conjunctive Normal Form (CNF) of $\varphi_1 \oplus \varphi_2$
   - Use Tseitin Encoding

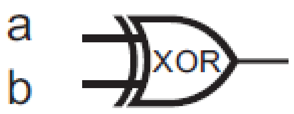
3. Give CNF($\varphi_1 \oplus \varphi_2$) to a **SAT solver**

4. C_1 and C_2 are **equivalent** if and only if CNF($\varphi_1 \oplus \varphi_2$) is **UNSAT**

Algorithm - Circuit Equivalence based on SAT



$\varphi_1 = \neg(a \wedge b) \wedge (a \vee b)$



$\varphi_2 = (a \wedge \neg b) \vee (\neg a \wedge b)$

Circuits are **equivalent** $\Leftrightarrow$ $\underbrace{\text{CNF}(\varphi_1 \oplus \varphi_2)}_{\text{Convert to CNF using Tseitin Encoding}}$ is **unsatisfiable**.

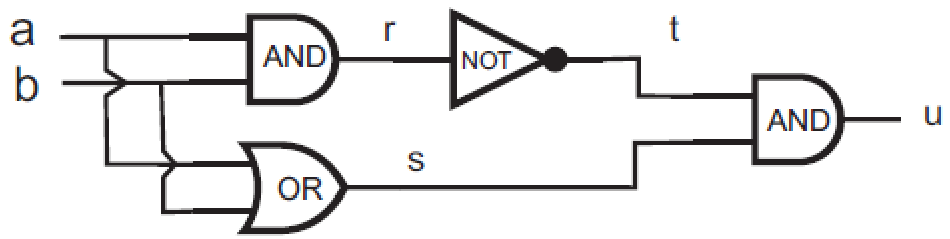
Outline

- Algorithm - Decide equivalence of combinational circuits
  - Based on reduction to Satisfiability
- Translation of a Circuit into a Formula
- Relations between Satisfiability, Validity, Equivalence and Semantic Entailment
- Normal Forms
- Tseitin Encoding

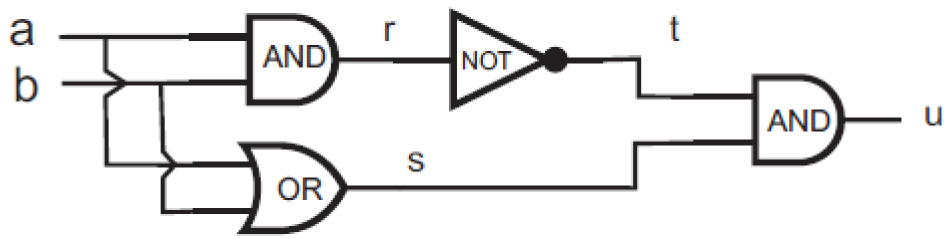
Circuits are **equivalent** $\Leftrightarrow$ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable**.

Convert to CNF using **Tseitin** Encoding

Translation of a Circuit into a Formula



Translation of a Circuit into a Formula

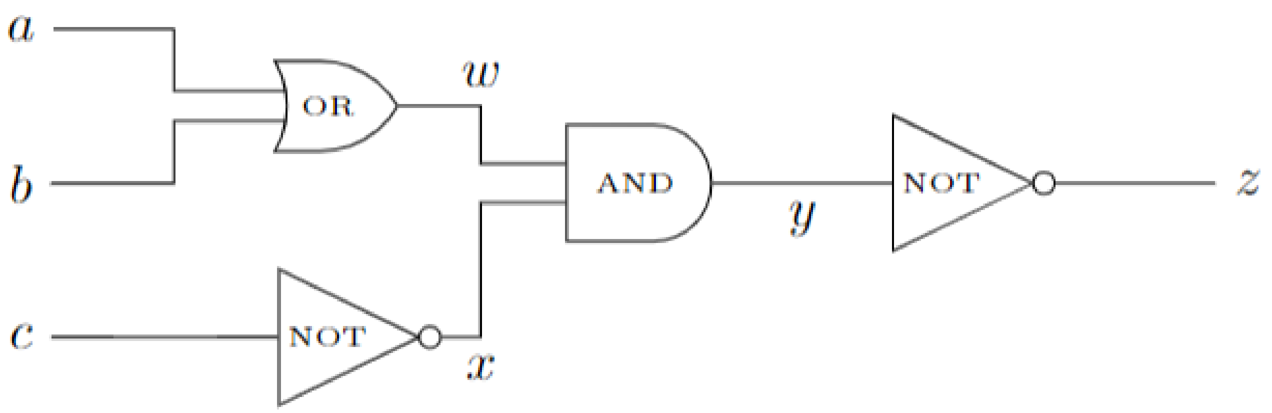


$$\begin{aligned}\varphi_1 &= t \wedge s \\ &= \neg r \wedge (a \vee b) \\ &= \neg(a \wedge b) \wedge (a \vee b)\end{aligned}$$

$$\begin{aligned}\varphi_2 &= a \oplus b \\ &= (a \wedge \neg b) \vee (\neg a \wedge b)\end{aligned}$$

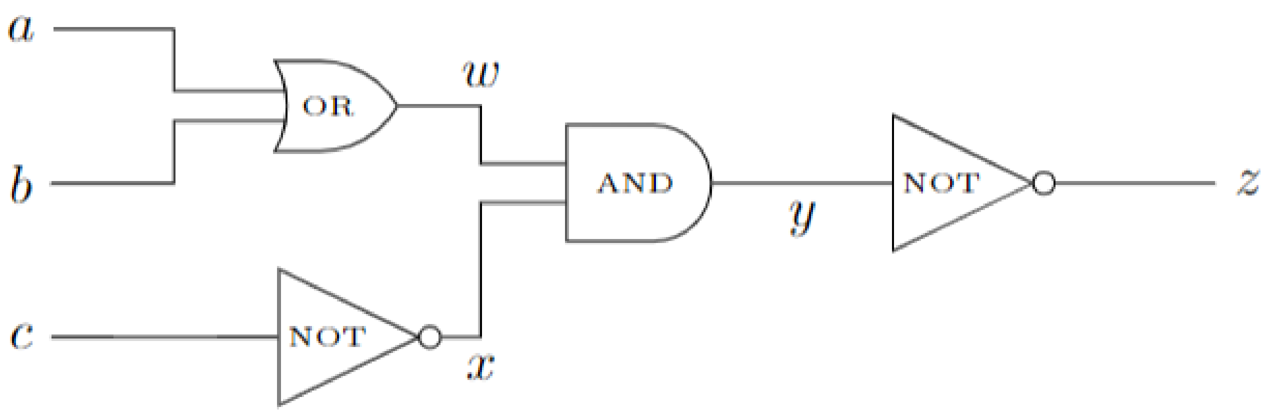
Translation of a Circuit into a Formula

[Lecture] Compute the propositional formula of the following circuit.



Translation of a Circuit into a Formula

[Lecture] Compute the propositional formula of the following circuit.



$$
\begin{aligned}
z &= \neg y \\
&= \neg(w \wedge x) \\
&= \neg((a \vee b) \wedge x) \\
&= \neg((a \vee b) \wedge \neg c)
\end{aligned}
$$

Outline

- Algorithm - Decide equivalence of combinational circuits
  - Based on reduction to Satisfiability ✓
- Translation of a Circuit into a Formula ✓
- Relations between Satisfiability, Validity, Equivalence and Semantic Entailment
- Normal Forms
- Tseitin Encoding

Circuits are **equivalent** $\Leftrightarrow$ $\underbrace{\text{CNF}(\varphi_1 \oplus \varphi_2)}_{\text{Convert to CNF using Tseitin Encoding}}$ is **unsatisfiable**.

Duality: Validity and Satisfiability

- ϕ is valid $\Leftrightarrow$ $\neg\phi$ is not satisfiable
  ϕ is satisfiable $\Leftrightarrow$ $\neg\phi$ is not valid

Duality: Validity and Satisfiability

- **ϕ is valid $\Leftrightarrow$ $\neg\phi$ is not satisfiable**
  ϕ is satisfiable $\Leftrightarrow$ $\neg\phi$ is not valid

- Example:
  - $\phi = (x \vee \neg x)$ is valid. Truth Table: All rows **T**.

Duality: Validity and Satisfiability

- **ϕ is valid ⇔ $\neg\phi$ is not satisfiable**
  ϕ is satisfiable ⇔ $\neg\phi$ is not valid

- Example:
  - $\phi = (x \vee \neg x)$ is valid. Truth Table: All rows **T**.
  - $\neg\phi = \neg(x \vee \neg x) \equiv \neg x \wedge x$ is not satisfiable. Truth Table: All rows **F**.

Duality: Validity and Satisfiability

- **ϕ is valid $\Leftrightarrow$ $\neg\phi$ is not satisfiable**
  ϕ is satisfiable $\Leftrightarrow$ $\neg\phi$ is not valid

- Example:
  - $\phi = (x \vee \neg x)$ is valid. Truth Table: All rows **T**.
  - $\neg\phi = \neg(x \vee \neg x) \equiv \neg x \wedge x$ is not satisfiable. Truth Table: All rows **F**.

- Only one decision procedure needed

Reductions

Solve / using	ϕ satisfiable?	ϕ valid?	$\phi \vdash \psi$?	$\phi \equiv \psi$?
Satisfiability	✓	$\neg\phi$ not satisfiable?	$\phi \wedge \neg\psi$ not satisfiable?	$\phi \oplus \psi$ not satisfiable?
Validity	$\neg\phi$ not valid?	✓	$\phi \rightarrow \psi$ valid?	$\phi \leftrightarrow \psi$ valid?
Entailment	$\top \nvdash \neg\phi$?	$\top \vdash \phi$?	✓	$\phi \vdash \psi$ and $\psi \vdash \phi$?
Equivalence	$\phi \not\equiv \bot$?	$\phi \equiv \top$?	$\phi \rightarrow \psi \equiv \top$?	✓

Outline

- Algorithm - Decide equivalence of combinational circuits
  - Based on reduction to Satisfiability ✓
- Translation of a Circuit into a Formula ✓
- Relations between Satisfiability, Validity, Equivalence and Semantic Entailment ✓
- Normal Forms
- Tseitin Encoding

Circuits are **equivalent** $\Leftrightarrow$ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable**.

Convert to CNF using **Tseitin** Encoding

Terminology

- **Literal:** propositional variable or its negation
  - Example: $p, \ \neg q$
- **Clause:** disjunction of literals
  - Example: $(p \lor \neg q \lor r)$
- **Cube:** conjunction of literals
  - Example: $(\neg x \land y \land \neg z)$

Normal Forms

- **Disjunctive Normal Form (DNF)**
  - Disjunction of cubes:

$$(a_1 \land a_2 \land \cdots \land a_n) \lor (b_1 \land \cdots \land b_m) \lor \cdots$$

where each a_i, b_j is a literal

- **Conjunctive Normal Form (CNF)**
  - Conjunction of clauses:

$$(a_1 \lor a_2 \lor \cdots \lor a_n) \land (b_1 \lor \cdots \lor b_m) \land \cdots$$

where each a_i, b_j is a literal

Ways to Obtain a CNF

- Via Truth Table
  - Exponential size
- Via Replacement Rules, DeMorgan, Distributivity
  - Exponential size
- **Tseitin Encoding**
  - Use auxiliary variables
  - Linear blow-up
  - Produces **equisatisfiable formula** with **linear blowup**

Definition of Equisatisfiability

ϕ and ψ are *equisatisfiable*

$\Updownarrow$

either both satisfiable, or both unsatisfiable

- For equivalence checking, we only need the info SAT or UNSAT

Tseitin Encoding

- Step 1
  - Assign new variables to all nodes in the parse tree / to each sub-formula
- Step 2
  - Add new clauses for each new variable
  - Apply Tseitin Rewrite Rules:

$$
\begin{aligned}
\chi \leftrightarrow (\varphi \vee \psi) \quad &\Leftrightarrow \quad (\neg\varphi \vee \chi) \wedge (\neg\psi \vee \chi) \wedge (\neg\chi \vee \varphi \vee \psi) \\
\chi \leftrightarrow (\varphi \wedge \psi) \quad &\Leftrightarrow \quad (\neg\chi \vee \varphi) \wedge (\neg\chi \vee \psi) \wedge (\neg\varphi \vee \neg\psi \vee \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \vee \neg\varphi) \wedge (\varphi \vee \chi)
\end{aligned}
$$

Tseitin Encoding

[Lecture] Apply Tseitin's encoding to the following formula: $\varphi = ((p \vee q) \wedge r) \vee \neg p$. For each variable you introduce, clearly indicate which subformula of φ it represents.

$$
\begin{aligned}
\chi \leftrightarrow (\varphi \vee \psi) \quad &\Leftrightarrow \quad (\neg\varphi \vee \chi) \wedge (\neg\psi \vee \chi) \wedge (\neg\chi \vee \varphi \vee \psi) \\
\chi \leftrightarrow (\varphi \wedge \psi) \quad &\Leftrightarrow \quad (\neg\chi \vee \varphi) \wedge (\neg\chi \vee \psi) \wedge (\neg\varphi \vee \neg\psi \vee \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \vee \neg\varphi) \wedge (\varphi \vee \chi)
\end{aligned}
$$

Tseitin Encoding

[Lecture] Apply Tseitin's encoding to the following formula: $\varphi = ((p \lor q) \land r) \lor \neg p$. For each variable you introduce, clearly indicate which subformula of φ it represents.

$$
\begin{aligned}
\chi \leftrightarrow (\varphi \lor \psi) \quad &\Leftrightarrow \quad (\neg\varphi \lor \chi) \land (\neg\psi \lor \chi) \land (\neg\chi \lor \varphi \lor \psi) \\
\chi \leftrightarrow (\varphi \land \psi) \quad &\Leftrightarrow \quad (\neg\chi \lor \varphi) \land (\neg\chi \lor \psi) \land (\neg\varphi \lor \neg\psi \lor \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \lor \neg\varphi) \land (\varphi \lor \chi)
\end{aligned}
$$

$$\underbrace{(\underbrace{(\underbrace{p \lor q}_{x_1}) \land r}_{x_2}) \lor \underbrace{\neg p}_{x_3}}_{x_\varphi}$$

$$
\begin{aligned}
CNF(\varphi) = \quad & (\neg p \lor x_1) \land (\neg q \lor x_1) \land (\neg x_1 \lor p \lor q) \\
& \land (\neg x_2 \lor x_1) \land (\neg x_2 \land r) \land (\neg x_1 \lor \neg r \lor x_2) \\
& \land (\neg x_3 \lor \neg p) \land (p \lor x_3) \\
& \land (\neg x_2 \lor x_\varphi) \land (\neg x_3 \lor x_\varphi) \land (\neg x_\varphi \lor x_2 \lor x_3) \\
& \land x_\varphi
\end{aligned}
$$

Tseitin Encoding

[Lecture] Apply Tseitin's encoding to the following formula: $\varphi = \neg(a \vee \neg b) \vee (\neg a \wedge c)$. For each variable you introduce, clearly indicate which subformula of φ it represents.

$$\begin{aligned}
\chi \leftrightarrow (\varphi \vee \psi) \quad &\Leftrightarrow \quad (\neg\varphi \vee \chi) \wedge (\neg\psi \vee \chi) \wedge (\neg\chi \vee \varphi \vee \psi) \\
\chi \leftrightarrow (\varphi \wedge \psi) \quad &\Leftrightarrow \quad (\neg\chi \vee \varphi) \wedge (\neg\chi \vee \psi) \wedge (\neg\varphi \vee \neg\psi \vee \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \vee \neg\varphi) \wedge (\varphi \vee \chi)
\end{aligned}$$

Tseitin Encoding

[Lecture] Apply Tseitin's encoding to the following formula: $\varphi = \neg(a \vee \neg b) \vee (\neg a \wedge c)$. For each variable you introduce, clearly indicate which subformula of φ it represents.

$$\chi \leftrightarrow (\varphi \vee \psi) \quad \Leftrightarrow \quad (\neg\varphi \vee \chi) \wedge (\neg\psi \vee \chi) \wedge (\neg\chi \vee \varphi \vee \psi)$$
$$\chi \leftrightarrow (\varphi \wedge \psi) \quad \Leftrightarrow \quad (\neg\chi \vee \varphi) \wedge (\neg\chi \vee \psi) \wedge (\neg\varphi \vee \neg\psi \vee \chi)$$
$$\chi \leftrightarrow \neg\varphi \quad \Leftrightarrow \quad (\neg\chi \vee \neg\varphi) \wedge (\varphi \vee \chi)$$

$$\underbrace{\underbrace{\neg \underbrace{(a \vee \underbrace{\neg b}_{x_4})}_{x_3}}_{x_1} \vee \underbrace{(\underbrace{\neg a}_{x_5} \wedge c)}_{x_2}}_{x_\varphi}$$

$$\begin{aligned} CNF(\varphi) = x_\varphi \wedge & \\ & (\neg x_1 \vee x_\varphi) \wedge (\neg x_2 \vee x_\varphi) \wedge (\neg x_\varphi \vee x_1 \vee x_2) \wedge \\ & (\neg x_1 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge \\ & (\neg x_3 \vee a) \wedge (\neg x_3 \vee x_4) \wedge (\neg a \vee \neg x_4 \vee x_3) \wedge \\ & (\neg x_5 \vee x_2) \wedge (\neg c \vee x_2) \wedge (\neg x_2 \vee x_5 \vee c) \wedge \\ & (\neg x_4 \vee \neg b) \wedge (x_4 \vee b) \wedge \\ & (\neg x_5 \vee \neg a) \wedge (x_5 \vee a) \end{aligned}$$

Derive Rewrite Rules

- $r \leftrightarrow (p \wedge q)$ … rewrite it to a CNF

$$a \rightarrow b \equiv \neg a \vee b$$

De-Morgan

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$
$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

Distributive Law

$$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$$
$$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$$

Derive Rewrite Rules

- $r \leftrightarrow (p \wedge q)$ … rewrite it to a CNF
- $(r \rightarrow p \wedge q) \wedge (p \wedge q \rightarrow r)$
- $(\neg r \vee (p \wedge q)) \wedge (\neg (p \wedge q) \vee r)$
- $(\neg r \vee p\ ) \wedge (\neg r \vee q) \wedge (\neg p \vee \neg\ q \vee r)$

$$a \rightarrow b \ \equiv \neg a \vee b$$

De-Morgan

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$
$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

Distributive Law

$$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$$
$$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$$

Derive Rewrite Rules

- $r \leftrightarrow (p \vee q)$ … rewrite it to a CNF

$$a \to b \equiv \neg a \vee b$$

De-Morgan

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$
$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

Distributive Law

$$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$$
$$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$$

Derive Rewrite Rules

- $r \leftrightarrow (p \vee q)$ … rewrite it to a CNF
- $((p \vee q) \rightarrow r) \wedge (r \rightarrow p \vee q)$
- $(\neg(p \vee q) \vee r) \wedge (\neg r \vee p \vee q)$
- $((\neg p \wedge \neg q) \vee r) \wedge (\neg r \vee p \vee q)$
- $(\neg p \vee r) \wedge (\neg q \vee r) \wedge (\neg r \vee p \vee q)$

$$a \rightarrow b \equiv \neg a \vee b$$

De-Morgan

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$
$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

Distributive Law

$$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$$
$$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$$

Tseitin Encoding

[Lecture] Derive a Rewrite-Rule for an implication node, i.e., what clauses are introduced by the node $x \leftrightarrow (p \rightarrow q)$?

$a \rightarrow b \equiv \neg a \vee b$

De-Morgan

$\neg(a \wedge b) \equiv \neg a \vee \neg b$
$\neg(a \vee b) \equiv \neg a \wedge \neg b$

Distributive Law

$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$
$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$

Derive Rewrite Rules

[Lecture] Derive a Rewrite-Rule for an implication node, i.e., what clauses are introduced by the node $x \leftrightarrow (p \rightarrow q)$?

Solution:

$$\begin{aligned}
x \leftrightarrow (p \rightarrow q) &\Leftrightarrow x \leftrightarrow (p \rightarrow q)\\
&\Leftrightarrow (x \rightarrow (p \rightarrow q)) \wedge ((p \rightarrow q) \rightarrow x)\\
&\Leftrightarrow (x \rightarrow (\neg p \vee q)) \wedge ((\neg p \vee q) \rightarrow x)\\
&\Leftrightarrow (\neg x \vee (\neg p \vee q)) \wedge (\neg(\neg p \vee q) \vee x)\\
&\Leftrightarrow (\neg x \vee \neg p \vee q) \wedge ((\neg\neg p \wedge \neg q) \vee x)\\
&\Leftrightarrow (\neg x \vee \neg p \vee q) \wedge ((p \wedge \neg q) \vee x)\\
&\Leftrightarrow (\neg x \vee \neg p \vee q) \wedge ((p \vee x) \wedge (\neg q \vee x))\\
&\Leftrightarrow (\neg x \vee \neg p \vee q) \wedge (p \vee x) \wedge (\neg q \vee x)
\end{aligned}$$

Tseitin Encoding

[Lecture] Explain the concept of equisatisfiability. Given a propositional logic formula φ, the Tseitin algorithm computes an equisatisfiable formula $CNF(\varphi)$ in CNF. Why is this enough for equivalence checking?

Tseitin Encoding

[Lecture] Explain the concept of equisatisfiability. Given a propositional logic formula φ, the Tseitin algorithm computes an equisatisfiable formula $CNF(\varphi)$ in CNF. Why is this enough for equivalence checking?

Solution:

Two propositional formulas φ and ψ are *equisatisfiable* if and only if either *both are satisfiable* or *both are unsatisfiable*.
When checking whether two formulas φ_1 and φ_2 are equivalent we check whether $\varphi = \varphi_1 \oplus \varphi_2$ is satisfiable. If φ is SAT we know that there is a model such that one of the input formulas evaluated to true, while the other evaluated to false. The equisatisfiable formula $CNF(\varphi)$ is satisfiable if and only if φ is satisfiable and therefore answers our question of whether the two input formulas are equivalent.

CEC Example

[Lecture] Check whether $\varphi_1 = a \land \neg b$ and $\varphi_2 = \neg(\neg a \lor b)$ are semantically equivalent using the reduction to satisfiability. Prepare everything until you have a formula CNF(φ), that you can give to a SAT solver.

$$\begin{aligned}
\chi \leftrightarrow (\varphi \lor \psi) \quad &\Leftrightarrow \quad (\neg\varphi \lor \chi) \land (\neg\psi \lor \chi) \land (\neg\chi \lor \varphi \lor \psi) \\
\chi \leftrightarrow (\varphi \land \psi) \quad &\Leftrightarrow \quad (\neg\chi \lor \varphi) \land (\neg\chi \lor \psi) \land (\neg\varphi \lor \neg\psi \lor \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \lor \neg\varphi) \land (\varphi \lor \chi)
\end{aligned}$$

CEC Example

[Lecture] Check whether $\varphi_1 = a \land \neg b$ and $\varphi_2 = \neg(\neg a \lor b)$ are semantically equivalent using the reduction to satisfiability. Prepare everything until you have a formula CNF(φ), that you can give to a SAT solver.

$$\begin{aligned}
\chi \leftrightarrow (\varphi \lor \psi) \quad &\Leftrightarrow \quad (\neg\varphi \lor \chi) \land (\neg\psi \lor \chi) \land (\neg\chi \lor \varphi \lor \psi) \\
\chi \leftrightarrow (\varphi \land \psi) \quad &\Leftrightarrow \quad (\neg\chi \lor \varphi) \land (\neg\chi \lor \psi) \land (\neg\varphi \lor \neg\psi \lor \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \lor \neg\varphi) \land (\varphi \lor \chi)
\end{aligned}$$

$$\begin{aligned}
\varphi &= \varphi_1 \oplus \varphi_2 \\
&= [\varphi_1 \lor \varphi_2] \land \neg[\varphi_1 \land \varphi_2] = \\
&= [(a \land \neg b) \lor (\neg(\neg a \lor b))] \land \neg[(a \land \neg b) \land (\neg(\neg a \lor b))]
\end{aligned}$$

$$\underbrace{\underbrace{\left[\underbrace{(a \land \underbrace{\neg b}_{x_{10}})}_{x_4} \lor \underbrace{(\neg \underbrace{(\underbrace{\neg a}_{x_{11}} \lor b)}_{x_8})}_{x_5}\right]}_{x_1} \land \underbrace{\neg \underbrace{\left[\underbrace{(a \land \underbrace{\neg b}_{x_{12}})}_{x_6} \lor \underbrace{(\neg \underbrace{(\underbrace{\neg a}_{x_{13}} \lor b)}_{x_9})}_{x_7}\right]}_{x_3}}_{x_2}}_{x_\varphi}$$

CEC Example

[Lecture] Check whether $\varphi_1 = a \land \neg b$ and $\varphi_2 = \neg(\neg a \lor b)$ are semantically equivalent using the reduction to satisfiability. Prepare everything until you have a formula CNF(φ), that you can give to a SAT solver.

$$\begin{aligned} \chi \leftrightarrow (\varphi \lor \psi) \quad &\Leftrightarrow \quad (\neg\varphi \lor \chi) \land (\neg\psi \lor \chi) \land (\neg\chi \lor \varphi \lor \psi) \\ \chi \leftrightarrow (\varphi \land \psi) \quad &\Leftrightarrow \quad (\neg\chi \lor \varphi) \land (\neg\chi \lor \psi) \land (\neg\varphi \lor \neg\psi \lor \chi) \\ \chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow \quad (\neg\chi \lor \neg\varphi) \land (\varphi \lor \chi) \end{aligned}$$

$$\begin{aligned} \varphi &= \varphi_1 \oplus \varphi_2 \\ &= [\varphi_1 \lor \varphi_2] \land \neg[\varphi_1 \land \varphi_2] = \\ &= [(a \land \neg b) \lor (\neg(\neg a \lor b))] \land \neg[(a \land \neg b) \land (\neg(\neg a \lor b))] \end{aligned}$$

$$\underbrace{\underbrace{\Big[\underbrace{(a \land \underbrace{\neg b}_{x_7})}_{x_3} \lor \underbrace{(\neg \underbrace{(\underbrace{\neg a}_{x_8} \lor b)}_{x_6})}_{x_4}\Big]}_{x_1} \land \underbrace{\neg \underbrace{\Big[\underbrace{(a \land \underbrace{\neg b}_{x_7})}_{x_3} \lor \underbrace{(\neg \underbrace{(\underbrace{\neg a}_{x_8} \lor b)}_{x_6})}_{x_4}\Big]}_{x_1}}_{x_2}}_{x_\varphi}$$

$$\begin{aligned} CNF(\varphi) = x_\varphi \land \; & \\ & (\neg x_\varphi \lor x_1) \land (\neg x_\varphi \lor x_2) \land (\neg x_1 \lor \neg x_2 \lor x_\varphi) \land \\ & (\neg x_1 \lor \neg x_2) \land (x_1 \lor x_2) \land \\ & (\neg x_3 \lor x_1) \land (\neg x_4 \lor x_1) \land (\neg x_1 \lor x_3 \lor x_4) \land \\ & (\neg x_3 \lor a) \land (\neg x_3 \lor x_7) \land (\neg a \lor \neg x_7 \lor x_3) \land \\ & (\neg x_4 \lor \neg x_6) \land (x_4 \lor x_6) \land \\ & (\neg x_8 \lor x_6) \land (\neg b \lor x_6) \land (\neg x_6 \lor x_8 \lor b) \land \\ & (\neg x_7 \lor \neg b) \land (x_7 \lor b) \land \\ & (\neg x_8 \lor \neg a) \land (x_8 \lor a) \land \end{aligned}$$

Outline

- Algorithm - Decide equivalence of combinational circuits
  - Based on reduction to Satisfiability
- Translation of a Circuit into a Formula
- Relations between Satisfiability, Validity, Equivalence and Semantic Entailment
- Normal Forms
- Tseitin Encoding

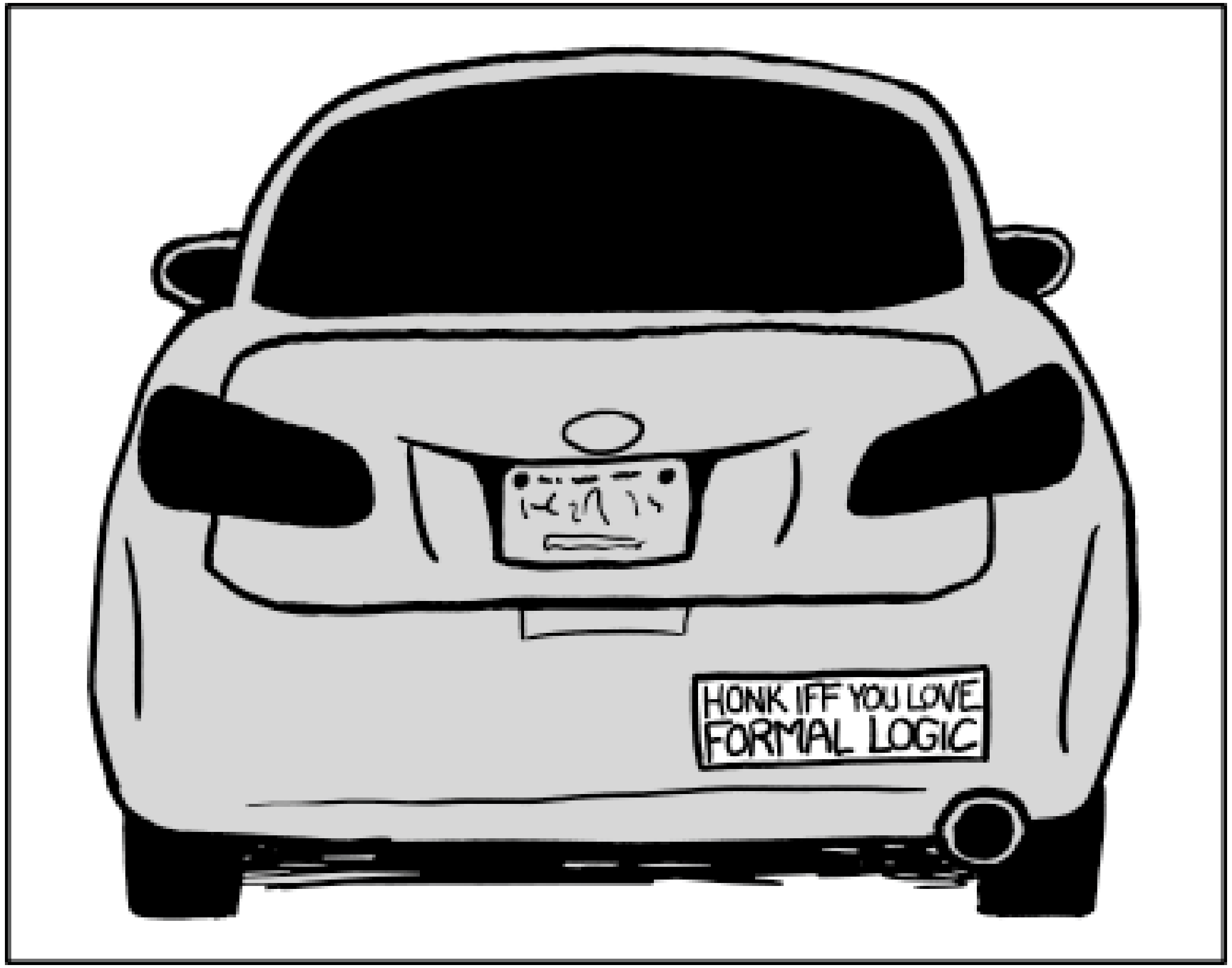
Circuits are **equivalent** $\Leftrightarrow$ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable**.

Convert to CNF using **Tseitin** Encoding

Learning Targets

- Explain the algorithm to check for equivalence based on the reduction to SAT
- Understand the notions between satisfiability, validity, equivalence and semantic entailment
- Understand the CNF and DNF normal form
  - Construct them using truth tables
- Apply Tseitin‘s algorithm to construct formulas in CNF
  - Understand the concept of equisatisfiability

Thank You



https://xkcd.com/1033/