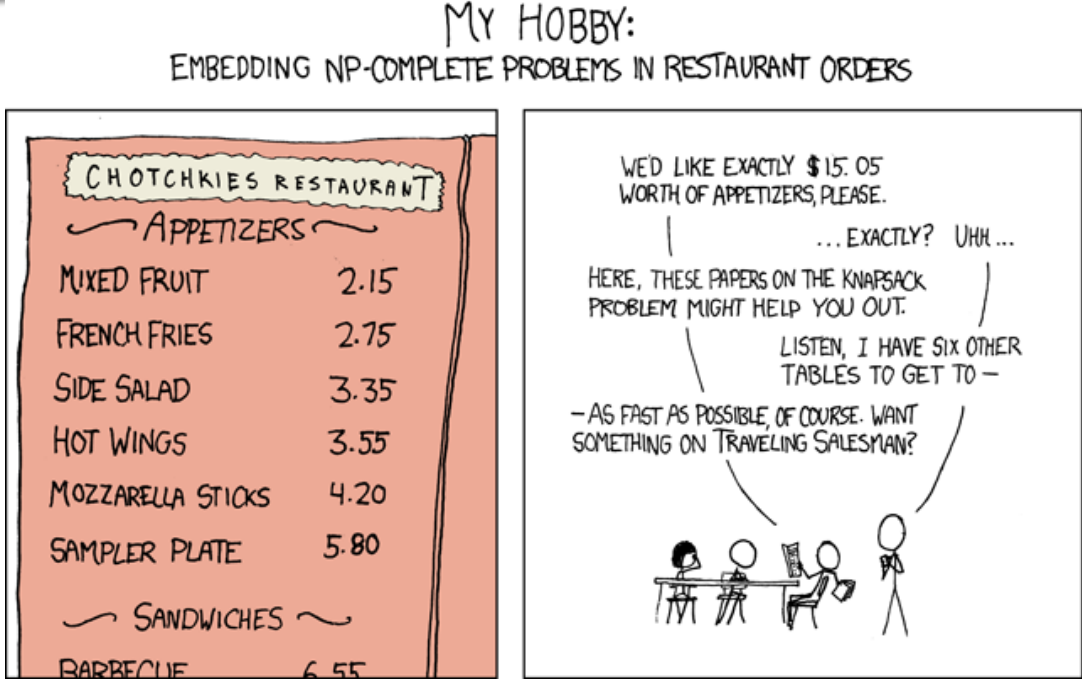
lamarr SECURITY RESEARCH

TU Graz SCIENCE PASSION TECHNOLOGY

Logic and Computability

Lecture 3

Combinational Equivalence Checking



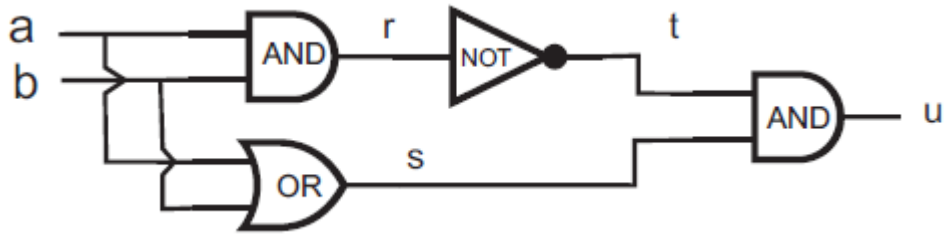
Bettina Könighofer
bettina.koenighofer@lamarr.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at

https://xkcd.com/287/

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

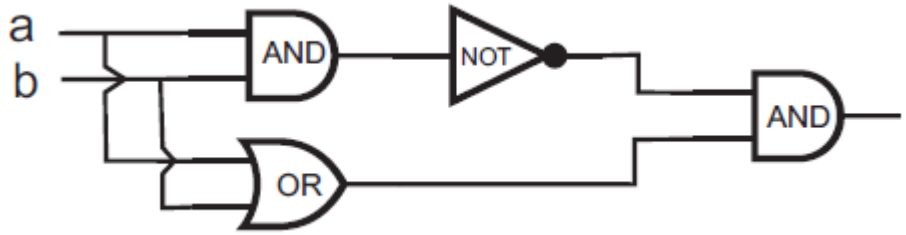
  - Exponentially large
  - → Not practicable!

- 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
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** $\Leftrightarrow$ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable**.

Convert to CNF using **Tseitin** Encoding

Outline

- Algorithm - Decide equivalence of combinational circuits
  - Based on reduction to Satisfiability

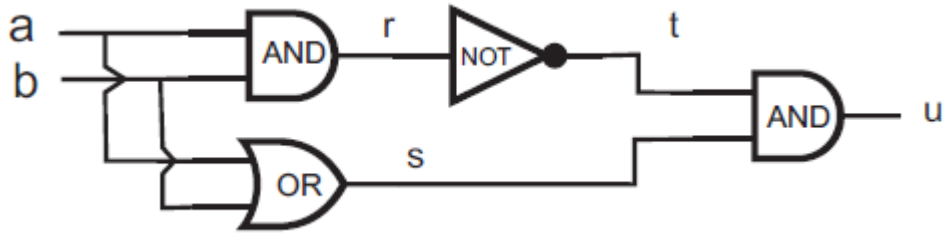
✓

- Translation of a Circuit into a Formula
- Relations between Satisfiability, Validity, Equivalence and Semantic Entailment
- Normal Forms
- Tseitin Encoding

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

Convert to CNF using **Tseitin** Encoding

Translation of a Circuit into a Formula



$$\varphi_1 = t \wedge s$$
$$= \neg r \wedge (a \vee b)$$
$$= \neg(a \wedge b) \wedge (a \vee b)$$

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

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** ⇔ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable.**

Convert to CNF using **Tseitin** Encoding

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** ⇔ 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: $(p \vee \neg q \vee r)$
- **Cube:** conjunction of literals
  - Example: $(\neg x \wedge y \wedge \neg z)$

Normal Forms

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

$$(a_1 \wedge a_2 \wedge \cdots \wedge a_n) \vee (b_1 \wedge \cdots \wedge b_m) \vee \cdots$$

where each a_i, b_j is a literal

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

$$(a_1 \vee a_2 \vee \cdots \vee a_n) \wedge (b_1 \vee \cdots \vee b_m) \wedge \cdots$$

where each a_i, b_j is a literal

DNF from Truth Table

Example:

p	q	r	$(r \vee q) \rightarrow (p \wedge \neg q)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$\neg p \wedge \neg q \wedge \neg r$

$\vee$

$p \wedge \neg q \wedge \neg r$

$\vee$

$p \wedge \neg q \wedge r$

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

CNF from Truth Table

Example:

p	q	r	$(p \vee \neg q) \rightarrow r$	
0	0	0	0	$p \vee q \vee r$
0	0	1	1	
0	1	0	1	$\wedge$
0	1	1	1	
1	0	0	0	$\neg p \vee q \vee r$
1	0	1	1	$\wedge$
1	1	0	0	$\neg p \vee \neg q \vee r$
1	1	1	1	

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

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** ⇔ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable.**

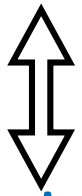
Convert to CNF using **Tseitin** Encoding

SAT Problem

- Given a formula, decide SAT / UNSAT
- Problem is NP - complete
  - $\mathbf{P \neq NP \Rightarrow}$ **worst-case exponential**
- Automated Tools: **"SAT Solver"**
  - Highly efficient for many practical problem instances
  - Require formula in CNF!
- Ways to obtain a CNF
  - Via truth tables → Exponential size
  - Tseitin Encoding → Produces **equisatisfiable formula** with **linear blowup**

Definition of Equisatisfiability

ϕ and ψ are *equisatisfiable*



either both satisfiable, or both unsatisfiable

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

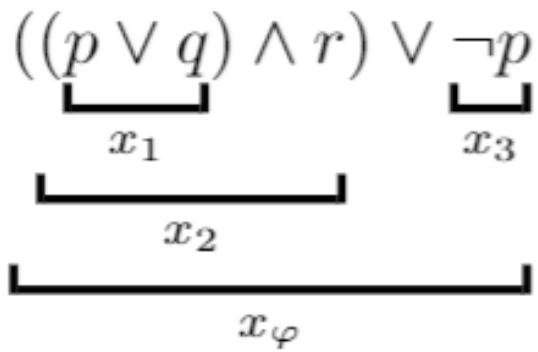
Tseitin Encoding

- 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

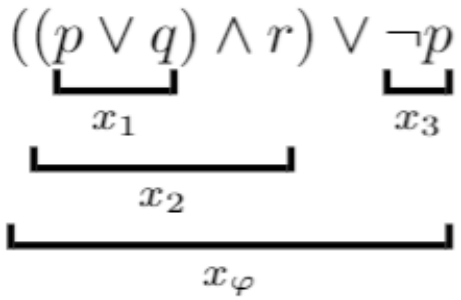
- Step 1
  - Assign new variables to all nodes in the parse tree / to each sub-formula



Tseitin Encoding

- Step 2
  - Add new clauses for each new variable
  - Apply Tseitin Rewrite Rules:

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



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

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** ⇔ CNF($\varphi_1 \oplus \varphi_2$) is **unsatisfiable.**

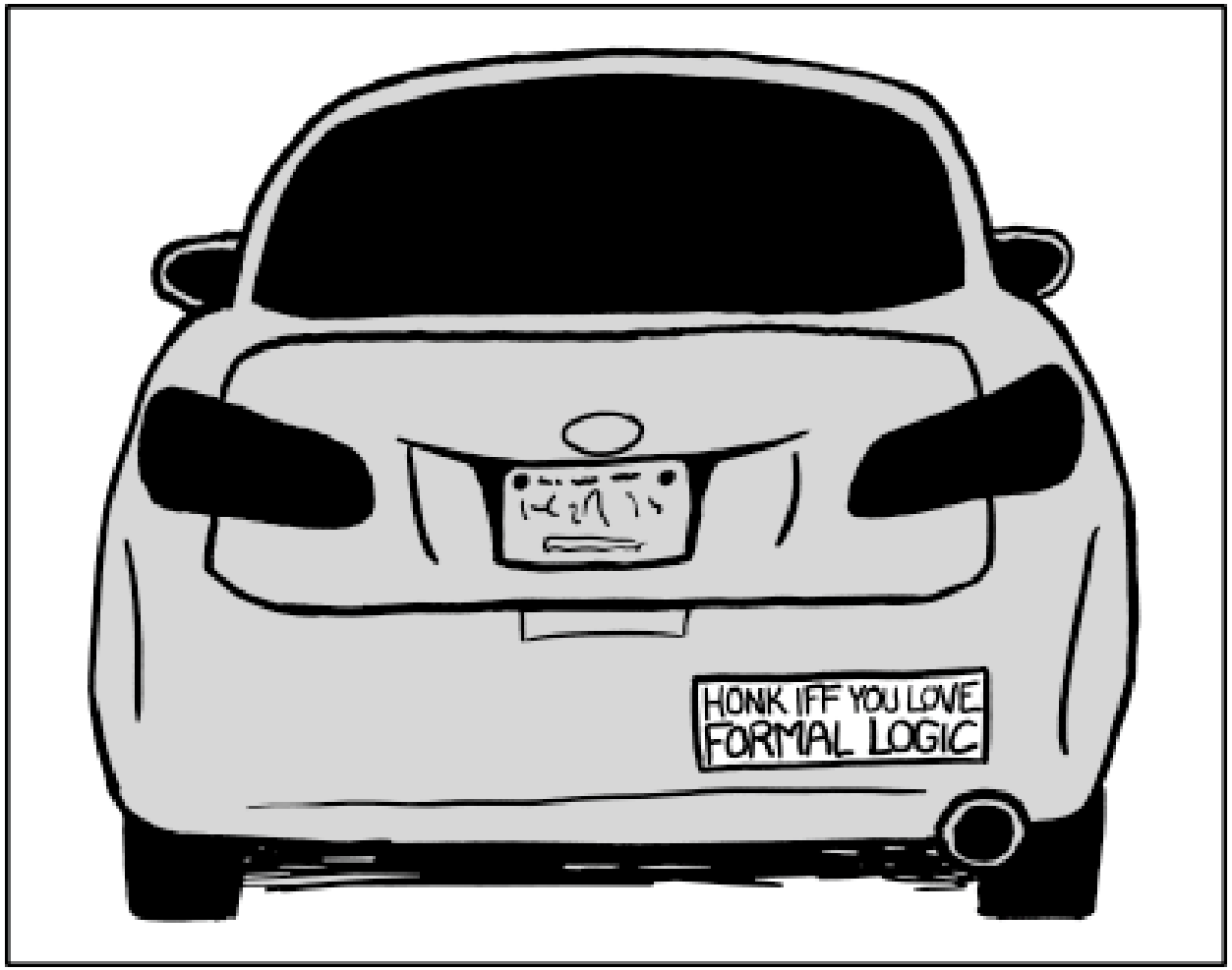
Convert to CNF using **Tseitin** Encoding

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

Lamarr SECURITY RESEARCH

Thank You



https://xkcd.com/1033/