Automata Theory:
From Theory to Applications

Moshe Y. Vardi

Rice University

Automata Theory vs. Calculus

Calculus:

- *Theory*: Real and complex analysis
- *Applications*: problem solving, e.g, compute area under curve

Automata Theory:

- *Theory*: finite automata and regular expressions, properties of regular sets, context-free grammars, pushdown automata, etc.
- *Applications*: not in the automata course.

Automata Theory and Hanukkah Candles

"We light these lights
For the miracles and the wonders
For the redemption and the battles
That you made for our forefathers
. . .

During all eight days of Hanukkah
These lights are sacred
And we are not permitted to make
Ordinary use of them
But only to look at them
In order to express thanks
. . ."

Applications of Automata Theory

E. Rich: Automata, Computability, and Complexity–Theory and Applications

http://www.cs.utexas.edu/ ear/cs341/AutomataTheoryBook.pdf

- Programming language and compilers
- Software engineering
- Security
- Interactive games
- Natural-language processing
- Artificial intelligence
- ...

Example: Combinational-Circuit Equivalence

Combinational Circuit: directed acyclic graph, with nodes labeled as input nodes (in-degree 0), output nodes (out-degree 0), or logic gates (and, or, not, etc).

- Combinational circuit C with m inputs and m outputs, defines a Boolean function $f_C : \{0,1\}^m \rightarrow \{0,1\}^n$.

- Two combinational circuits C_1 and C_2 are *equivalent* if they define the same Boolean function.

Circuit-Equivalence Problem: Given two combinational circuits, decide if they are equivalent.

Motivation: Computer-aided design

Nonemptiness of DFAs

DFA– deterministic finite automata $A = (\Sigma, S, s_0, \delta, F)$

- S - state set
- $s_0 \in S$ - initial state
- $\delta : S \times \Sigma \to S$ - transition function
- $F \subseteq S$: final states

Emptiness: A is empty if $L(A) = \emptyset$.

From Automata to Graphs

DFA: $A = (\Sigma, S, s_0, \delta, F)$

Graph: $G_A = (S, E_A)$
$E_A = \{(s,t) : t \in \delta(s,a)$ for some $a \in \Sigma\}$.

Intuition: Delete labels from edges.

Theorem: $L(A) \neq \emptyset$ iff there is path in G_A from s_0 to F.

Corollary: Emptiness can be checked in linear time using breadth-first search.

Equivalence of DFAs

Equivalence: Two DFAs, A_1 and A_2 are equivalent iff $L(A_1) = L(A_2)$.

DFA-Equivalence Problem: Given two DFAs, A_1 and A_2, decide if they are equivalent.

In-equivalence Product:
Given $A_1 = (\Sigma, S^1, s^1, \delta^1, F^1)$ and $A_2 = (\Sigma, S^2, s^2, \delta^2, F^2)$, define their in-equivalence product A by taking the standard cross product, with final states $F = \big(F^1 \times (S^2 - F^2)\big) \cup \big((S^1 - F^1) \times F^2\big)$.

Theorem: A_1 and A_2 are equivalent iff $L(A) = \emptyset$

Corollary: Equivalence can be decided in quadratic time.

Circuits and Automata

Language of Circuit: Given a circuit C with inputs $x_1, \ldots, x_m$ and outputs $y_1, \ldots, y_n$, the *language* $L(C)$ is the set of words over $\{0, 1\}$ of length $m+n$ such that $(\mathbf{x}, \mathbf{y}) \in L(C)$ iff $\mathbf{y} = f_C(\mathbf{x})$.

Lemma: C_1 and C_2 are equivalent iff they have the same number of inputs and outputs, respectively, and $L(C_1) = L(C_2)$.

Key Observation: Language of a circuit is finite, therefore regular.

Automata-Theoretic Approach to Circuit Equivalence

Key idea: Given C_1 and C_2,

- Construct DFAs A_1 and A_2 such that $L(A_i) = L(C_i)$,
- Check that A_1 and A_2 are equivalent.

From circuits to automata:

- Sort circuit nodes topologically
- Generate DFA for each node using Boolean closure of regular languages
- Minimize DFAs at each step

Really?

Q: Is This A Theoretician's Dream?

Q: No. This is used daily! DFAs for circuits are called *binary decision diagrams*–**BDDs**.