COMPSCI 501: Formal Language Theory

Lecture 31: Circuit Complexity

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

10 April 2019

What can we implement via circuits?

- A tool for attacking P = NP?
- Identify parallelizable computation vs. sequential bottleneck
- Alternative NP-completeness for SAT

From Sequential to Combinational

Circuits: AND, OR, NOT gates. Acyclic (why?)

Simplest computations: output = f(input). $f : \{0,1\}^n \to \{0,1\}$
May generalize to multiple output bits, $\{0,1\}^k$
$z = x + y$. Binary representation, combinational circuit.

How about:

```
s = 0
for i = 1 to n do
    s = s + a[i]
end for
```

Can view a[i] as inputs

Introduce successive copies of s and unroll:

$s_0 = 0 \wedge s_1 = s_0 + a_1 \wedge \ldots \wedge s_n = s_{n-1} + a_n$

Polynomial time algorithm $\implies$ polynomial size circuit

n stages: $\implies$ circuit depth matters (lecture on parallelism)

Circuit families and complexity

Turing machine can be fixed, variable-length input

Circuit is fixed-size $\implies$ family of circuits, one per input length

Def: A **circuit family** C is an infinite list of circuits, $(C_0, C_1, C_2, \ldots)$, where C_n has n input variables.

C decides a language A over $\{0,1\}$ if for every string w, $w \in A$ iff $C_n(w) = 1$. $(n = |w|)$

Can define:

- *size*, *size-minimal* circuits (no smaller equivalent)
- *size complexity* of a family: $f : \mathbb{N} \to \mathbb{N}$, $f(n) = size(C_n)$
- *depth*, *depth minimal*, *depth complexity* (measure of sequential-ness)

Circuit complexity of a language = size complexity of minimal circuit family

Time vs. Circuit Complexity

Theorem: Let $t : \mathbb{N} \to \mathbb{N}$ be a function with $t(n) \geq n$.
If $A \in$ TIME($t(n)$), then A has circuit complexity $O(t^2(n))$.

Consequence: if we were to find a language in NP with more than polynomial circuit complexity, that language can't be in P!

Assume running time $t(n)$
Build $t(n) \times t(n)$ *tableau* of configurations
Build circuit C_n from tableau.

- standardize accept: halt on leftmost cell, write blank
- encode head together with tape symbol: $|(Q \times \Gamma) \cup \Gamma| = k$
- one bit (wire) saying "cell[i,j] has symbol s" $\implies$ $kt^2(n)$ bits
- $bit[i,j,s]$ can be $bit[i-1,j,s]$ (keep previous)
  or come from cells $j-1/j+1$ in row $i-1$ (head move/write)
- cells in row 1 wired to input word

CIRCUIT-SAT is NP-complete

CIRCUIT=SAT $= \{\langle C \rangle | C$ is a satisfiable circuit$\}$

Clearly, it's in NP (witness = satisfying assignment, check in P)

Mapping reduction for any lanugage $A \in$ NP to *CIRCUIT-SAT*
$f(w) = \langle C \rangle$ with $w \in A \Leftrightarrow C$ is satisfiable

$A \in$ NP $\implies$ has poly-time verifier V with input $\langle x, c \rangle$

Construct circuit for V using tableau method.
Inputs for w filled in, inputs for c free.
C satisfiable iff a certificate exists $\Leftrightarrow w \in A$

Circuit building time/size: square in verifier size $\implies$ polynomial

3-SAT reduction from *CIRCUIT-SAT*

Express truth table for each gate as conjunction of implications:

$x = a$ AND b: (similar for OR, NOT)

$(a \wedge b \to x) \wedge (\neg a \wedge b \to \neg x) \wedge (a \wedge \neg b \to \neg x) \wedge (\neg a \wedge \neg b \to \neg x)$

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

Could simplify to $(\neg a \vee \neg b \vee x) \wedge (a \vee \neg x) \wedge (b \vee \neg x)$
(recall Tseitin transform)

Construction: linear time, linear-size formula ϕ
(one variable for each input and wire; $\leq$ 3 literals/clause)

ϕ precisely describes C's computation: satisfiable iff C is