Circuit Complexity of Regular Languages

Michal Koucky

Presented by,
Sunil K. S

April 13, 2012

Contents

Theme of Presentation

- Regular Languages
- Algebra: Monoids and Groups
- Relation between Regular Languages and Monoids
- Circuit complexity of Regular Languages
- Circuit complexity of Reg.Lang. in terms of Monoid Product

Algebraic Preliminaries: Monoids

- Monoid: A set M together with an associative binary operation that contains an identity element 1_M such that $\forall m \in M, m.1_M = 1_M.m = m$
- Represented as $(M, *, e)$
- Group: Monoid with an inverse element
  - Finite and infinite monoids
  - Group free monoids
  - Solvable and unsolvable groups
    A group G is solvable if it has a subnormal series

$$G = G_0 \geq G_1 \geq G_2 \geq \cdots \geq G_n = 1$$

where each quotient G_i/G_{i+1} is an abelian group.

Operations on Monoids

- Product over a monoid: $f : M^* \to M$ such that $f(m_1, m_2, \cdots, m_n) = m_1.m_2.\ldots.m_n$
- *a-word problem*: For $a \in M$, the language of words from M^* that multiply out to a.
- *word problem*: if not concerned about the choice of a.
- All *word problems* over M are regular languages.

Monoids as Recognizers

- *Morphism*: from $(M, ., e)$ to $(N, *, f)$ is a function $\phi : M \to N$ such that $u, v \in M, \phi(u.v) = \phi(u) * \phi(v)$ and $\phi(e) = f$.
  Eg: $len : \Sigma^* \to \mathbb{N}$ with $len(x) = |x|$
- Given a monoid $(M, ., e)$, a subset X of M and morphism $\phi : \Sigma^* \to M$, the *language defined* by X w.r.t ϕ is $\phi^{-1}(X)$
- $L \subseteq \Sigma^*$ can be *recognized* by M if there exists a morphism $\phi : \Sigma^* \to M$ and a subset $X \subseteq M$ so that $L = \phi^{-1}(M)$.

Monoids as Recognizers Cntd..

- A language is regular iff it can be recognized by some finite monoid (a variant of *Kleene's theorem*).
  - Let L is recognized by the monoid M via the morphism ϕ and $X \subseteq M$
  - Define $A_M = (M, \Sigma, \delta, e, X)$ where $\delta(m, a) = m.\phi(a), \forall m \in M, a \in \Sigma$
  - $\hat{\delta}(m, a_1 a_2 \cdots a_n) = m.\phi(a_1).\phi(a_2)....\phi(a_n)$
  - $\hat{\delta}(e, a_1 a_2 \cdots a_n) = e.\phi(a_1).\phi(a_2)....\phi(a_n) = \phi(a_1 a_2 \cdots a_n)$
  - Thus $L(A_M) = \{x | \phi(x) \in X\} = L$

Monoids as Recognizers Cntd..

- A language is regular iff it can be recognized by some finite monoid (a variant of *Kleene's theorem*).
  - Let L is recognized by the monoid M via the morphism ϕ and $X \subseteq M$
  - Define $A_M = (M, \Sigma, \delta, e, X)$ where $\delta(m, a) = m.\phi(a), \forall m \in M, a \in \Sigma$
  - $\hat{\delta}(m, a_1 a_2 \cdots a_n) = m.\phi(a_1).\phi(a_2)....\phi(a_n)$
  - $\hat{\delta}(e, a_1 a_2 \cdots a_n) = e.\phi(a_1).\phi(a_2)....\phi(a_n) = \phi(a_1 a_2 \cdots a_n)$
  - Thus $L(A_M) = \{x | \phi(x) \in X\} = L$
- Syntactic monoid: Minimal monoid $M(L)$ that recognize L.
- Syntactic morphism: $\nu_L : \Sigma^* \rightarrow M(L)$
- M_L is the monoid of state transformations generated by minimum state FSA recognizing L

Monoids and Automata

Automata to Monoid

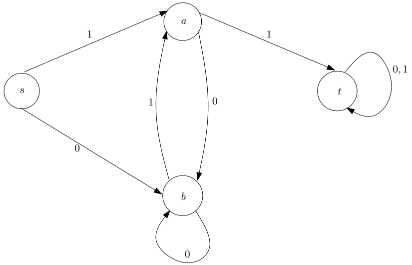


Figure: Automata

Inputs	0	1	00	01	10	11	000	001	010	011	100	101	110	111
s	b	a	b	a	b	r	b	a	b	r	b	a	r	r
a	b	r	b	a	r	r	b	a	b	r	r	r	r	r
b	b	a	b	a	b	r	b	a	b	r	b	a	r	r
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Same as			0				0	01	0	11	10	1	11	11

$*$	T_0	T_1	T_{01}	T_{10}	T_{11}
T_0	T_0	T_{01}	T_{01}	T_{10}	T_{11}
T_1	T_{10}	T_{11}	T_1	T_{11}	T_{11}
T_{01}	T_0	T_{11}	T_{01}	T_{11}	T_{11}
T_{10}	T_{10}	T_1	T_1	T_{10}	T_{11}
T_{11}	T_{11}	T_{11}	T_{11}	T_{11}	T_{11}

$$T_{10} * T_{01} = T_{1001} = T_{100} * T_1 = T_{10} * T_1 = T_{101} = T1$$

and

$$T_{01} * T_{01} = T_{0101} = T_{010} * T_1 = T_0 * T_1 = T_{01}$$

Identity: T_λ such that $T_s * T_\lambda = T_\lambda * T_s = T_s$, for all input strings s.

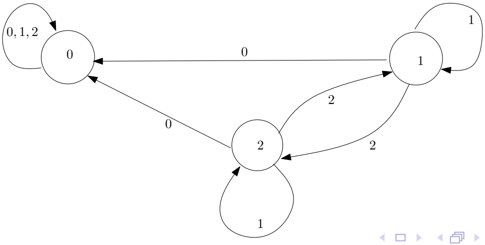
Monoid to Automata

Definition

Machine of a Monoid: If $[M, *]$ is a finite monoid, then the machine of M, denoted $m(M)$, is the state machine with state set M, input set M, and next-state function $t : M \times M \to M$ defined by $t(s, x) = s * x$.

Example

$[\mathbb{Z}_3, \times_3]$



Circuit Complexity

- Size of a circuit: Number of gates
- NC^0: Constant depth, bounded fan-in circuits
- AC^0: Constant depth, unbounded fan-in circuits
- $AC^0[q]$: AC^0 circuits with MOD_q gates
- ACC^0: AC^0 circuits with arbitrary MOD_q gates
- TC^0: Constant depth threshold circuits
- NC^1: Log depth, bounded fan-in, polynomial size circuits

- All regular languages are computable by linear size NC^1 circuits.
- Regular Languages in AC^0 and ACC^0 : Computable by almost linear size circuits.
- Existence of NC^1-complete languages
  Eg: *Boolean formula value problem (BFVP):* given a Boolean formula χ and values for the variables of χ, does χ evaluate to 1?

Reg. Lang. & Circuit Complexity

- All regular languages are computable by linear size NC^1 circuits.
- Regular Languages in AC^0 and ACC^0 : Computable by almost linear size circuits.
- Existence of NC^1-complete languages
  Eg: *Boolean formula value problem (BFVP):* given a Boolean formula χ and values for the variables of χ, does χ evaluate to 1?
- To separate ACC^0 and NC^1 it is suffices to prove that for some $\epsilon > 0$ an $\Omega(n^{1+\epsilon})$ lower bound on the circuit size of ACC^0 circuits which computing certain NC^1-complete functions.

Reg. Lang. & Circuit Complexity Cntd...

The relation between circuit complexity of regular language and the word problem over its syntactic monoid ML

- For $L \subseteq \Sigma^*$, $L^{=k}$ means $L \cap \Sigma^k$

Proposition

If a regular language L is computable by a circuit family of size $s(n)$ and depth $d(n)$ and for some $k \geq 0, \nu_L(L^{=k}) = M(L)$ then the product over its syntactic monoid $M(L)$ is computable by a circuit family of size $O(s(O(n)) + n)$ and depth $d(O(n)) + O(1)$

Proposition

If the product over a monoid M is computable by a circuit family of size $s(n)$ and depth $d(n)$ then the regular language with the syntactic monoid M is computable by a circuit family of size $s(n) + O(n)$ and depth $d(n) + O(1)$

Mapping the landscape

Theorem

All regular languages are computable by linear size NC^1 circuits.

- It is suffice to show that there are NC^1 circuits of linear size for the product of n elements over a fixed monoid M.
- Product of n elements $\Rightarrow$ product of $n/2$ elements (computing the product of adjacent pairs of elements in parallel).
- Final circuit have logarithmic depth and linear size.

Mapping the landscape Cntd...

Can all regular languages be put into even smaller circuit class?

Can all regular languages be put into even smaller circuit class?
It is very unlikely: Barrington [1].

Mapping the landscape Cntd...

Can all regular languages be put into even smaller circuit class?
It is very unlikely: Barrington [1].

- Monoid M contains a non-solvable group $\Rightarrow$ the word problem over M is hard for NC^1 under projections.
- Projection:
  - Simple reduction: $w \in L$ to $w' \in L'$.
  - Each symbol of w' depends on at most one symbol of w.
  - The length of w' depends only on the length of w.
- Unless NC^1 collapses to smaller classes, NC^1 circuits are optimal for some regular languages.

Mapping the landscape Cntd...

Can all regular languages be put into even smaller circuit class?
It is very unlikely: Barrington [1].

- Monoid M contains a non-solvable group $\Rightarrow$ the word problem over M is hard for NC^1 under projections.
- Projection:
  - Simple reduction: $w \in L$ to $w' \in L'$.
  - Each symbol of w' depends on at most one symbol of w.
  - The length of w' depends only on the length of w.
- Unless NC^1 collapses to smaller classes, NC^1 circuits are optimal for some regular languages.

Theorem

Any regular language whose syntactic monoid contains a non-solvable group is hard for NC^1 under projections.

Mapping the landscape Cntd...

Theorem

If a language L has a group-free syntactic monoid $M(L)$ then L is in AC^0

- Regular languages with group-free syntactic monoids: *Star-free languages* or *non-counting languages*.
- Can be described by using only union, concatenation and complement operations.
- Proof (by Chandra) uses the characterization of counter-free regular languages by flip-flop automata of McNaughton and Papert [4].
- Showed that prefix product over *carry semi-group* is computable by AC^0 circuits.
- Carry semi-group:
  - Monoid with three elements P, R, S: $xP = x, xR = R, xS = S$ for any $x \in \{P, S, R\}$.

Mapping the landscape Cntd...

Theorem

If a monoid M contains a group then the product over M is not in AC^0

- Proof shows how the product over monoid with a group can be used to count number of ones in an input from $\{0,1\}^*$ modulo some constant $k \geq 2$.
- By the result of Furst, Saxe and Sipser [5] that cannot be done in AC^0.
- Hence Product over monoids containing groups cannot be done in AC^0.

Mapping the landscape Cntd...

- The language LENGTH(2) of words of even length:
    - Its syntactic monoid contains a group.
    - It is in AC^0

Theorem

A regular language is in AC^0 iff for every $k \geq 0$, the image of $L^{=k}$ under the syntactic morphism $\nu_L(L^{=k})$ does not contain a group.

- L is in AC^0 iff it can be described by a regular expression using operations union, concatenation and complement with the atom $\{a\}$ for every $a \in \Sigma$ and LENGTH(q) for every $q \geq 1$.

Mapping the landscape Cntd...

Theorem

If a syntactic monoid of a language contains only solvable groups then the language is computable by ACC^0 circuits

Example: PARITY of words from $\{0, 1\}^*$.

Mapping the landscape Cntd...

Theorem

If a syntactic monoid of a language contains only solvable groups then the language is computable by ACC^0 circuits

Example: PARITY of words from $\{0,1\}^*$.

- Regular Languages:
  - Some of them are complete for NC^1
  - Some of them are computable in AC^0
  - Otherwise they are in ACC^0
- TC^0 does not get assigned any languages unless it is equal to NC^1 or ACC^0.
- Proving regular language whose syntactic monoid contain non-solvable group is in TC^0 would collapse NC^1 to TC^0

Circuit size of regular languages

- All regular languages are computable by linear size NC^1 circuits.
- Can anything similar be said about regular languages in AC^0 or ACC^0?

Circuit size of regular languages

- All regular languages are computable by linear size NC^1 circuits.
- Can anything similar be said about regular languages in AC^0 or ACC^0?
- Th_2: Language over $\{0, 1\}$ of that contain at least two ones
  - Regular language
  - Can be computed by AC^0
  - Check all pairs of input positions: whether anyone of them contains two ones.
  - Circuit size: quadratic.
  - Ragde and Wigderson [6]
    - Th_k for up to poly-logarithmic k are computable by linear size AC^0 circuits.
    - Construction is based on perfect hashing

Circuit size of regular languages Cntd..

- Size reduction of constant depth circuits computing regular languages
  - Let L be a regular language and the product over its syntactic monoid is computable by $O(n^k)$-size constant-depth circuits.
  - Divide an input $x \in M(L)^n$ into consecutive blocks of size $\sqrt{n}$ and compute product of each block in parallel.
  - Compute the product of the $\sqrt{n}$ products
  - Total size is $O(\sqrt{n}.n^{k/2}) = O(n^{(k+1)/2})$
  - Depth of the circuits only doubles.

Proposition

Let L be a regular language computable by a polynomial-size constant-depth circuits over arbitrary gates. If the product over its syntactic monoid $M(L)$ is computable by circuits of the same size then for every $\epsilon > 0$, there is a constant-depth circuit family of size $O(n^{1+\epsilon})$ that computes L.

Circuit size of regular languages Cntd..

Theorem

Let $g_0(n) = n^{1/4}$ and further for each $d = 0, 1, 2, \cdots$, $g_{d+1}(n) = g_d^*(n)$. Every regular languages L with a group-free syntactic monoid is computable by AC^0 circuits of depth $O(d)$ and size $O(n.g_d^2(n))$, for any $d \geq 0$.

$g^*(n) = min\{i : g^i(n) \leq 1\}$
$g^i(.)$ denotes $g(.)$ iterated i-times

- Chandra proved that almost all languages in AC^0 are computable by circuit families of almost linear size.
- True for product over group-free monoids

Theorem

Every regular languages L in AC^0 is computable by AC^0 circuits of depth $O(d)$ and size $O(n.g_d^2(n))$, for any $d \geq 0$.

Circuit size of regular languages Cntd..

Proof.

- $L \in AC^0 \Rightarrow \exists M$ (a group free monoid) and $k \geq 1$ such that all words of length divisible by k are mapped into M by the syntactic morphism of L.
- It is suffices to show that we can compute $\nu_L(w)$ for any $w \in \Sigma^n, n \geq 1$
  - Design a circuit:
    - Divide w into blocks $b_1, b_2, \cdots, b_m$ of length k and one block b of length at most k
    - For each b_i, compute the mapping $\nu_L(b_i)$ to obtain elements in M.
    - Compute the product $m' = \nu_L(b_1).\nu_L(b_2) \cdots \nu_L(b_m)$ using circuit of depth $O(d)$ and size $O(n.g_d^2(n))$
    - Compute $\nu_L(w) = m'.\nu_L(b)$
    - As k is a constant, the depth of the circuit will be $O(d)$ and size $O(n.g_d^2(n))$

□

Circuit size of regular languages Cntd..

Theorem

Every regular language L whose syntactic monoid contains only solvable groups is computable by ACC^0 circuits of size $O(n.g_i^2(n))$.

- Assuming that ACC^0 and NC^1 are different the above theorem indeed applies to all regular languages in ACC^0 .
- Proof is by an induction on the depth of the regular expression describing L.

Wires vs. gates

Theorem

Let L be a regular language.

- *If L is in AC^0 then for every $d \geq 0$ it is computable by AC^0 circuits using $O(ng_d^2(n))$ wires.*
- *If L is in ACC^0 and it is not hard for NC^1 then for every $d \geq 0$ it is computable by ACC^0 circuits using $O(ng_d^2(n))$ wires.*
- *If L is in ACC^0 then for every $\epsilon > 0$ it is computable by ACC^0 circuits using $O(n^{1+\epsilon})$ wires.*

Wires vs. gates Cntd..

Theorem

The class of regular languages computable by ACC^0 circuits using linear number of wires is a proper subclass of the languages computable by ACC^0 circuits using linear number of gates.

It is not known however whether the same is true for AC^0.

Open Problem

Are the classes of regular languages computable by AC^0 circuits using linear number of gates and liner number of wires different?

References

- D. A. Barrington. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. Journal of Computer and System Sciences, 38(1):150164,1989.
- A. K. Chandra, S. Fortune, and R. J. Lipton. Unbounded fan-in circuits and associative functions. Journal of Computer and System Sciences, 30:222234, 1985.
- H. Straubing. Families of recognizable sets corresponding to certain varieties of finite monoids. Journal of Pure and Applied Algebra, 15(3):305318, 1979.
- R. McNaughton and S.A. Papert. Counter-Free Automata. The MIT Press, 1971.
- M. Furst, J. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. Mathematical Systems Theory, 17:1327, 1984.
- P. Ragde and A. Wigderson. Linear-size constant-depth polylog-threshold circuits. Information Processing Letters, 39:143146, 1991.

Thank you all....!