Efficient Divide-and-Conquer Simulations Of Symmetric FSAs

David Pritchard,
University of Waterloo

Aug. 29 '07
Automata @ Toronto, ON

Introduction

- Divide-and-conquer paradigm is used in both classical algorithms & parallel computing.
- We conisder a formal model for a divide-and-conquer process that resembles an FSA.
- Given a symmetric FSA (i.e., one unaffected by permutations of input) we show there exists a *divide-and-conquer automaton* to simulate it, such that the size of the state space used doesn't increase.

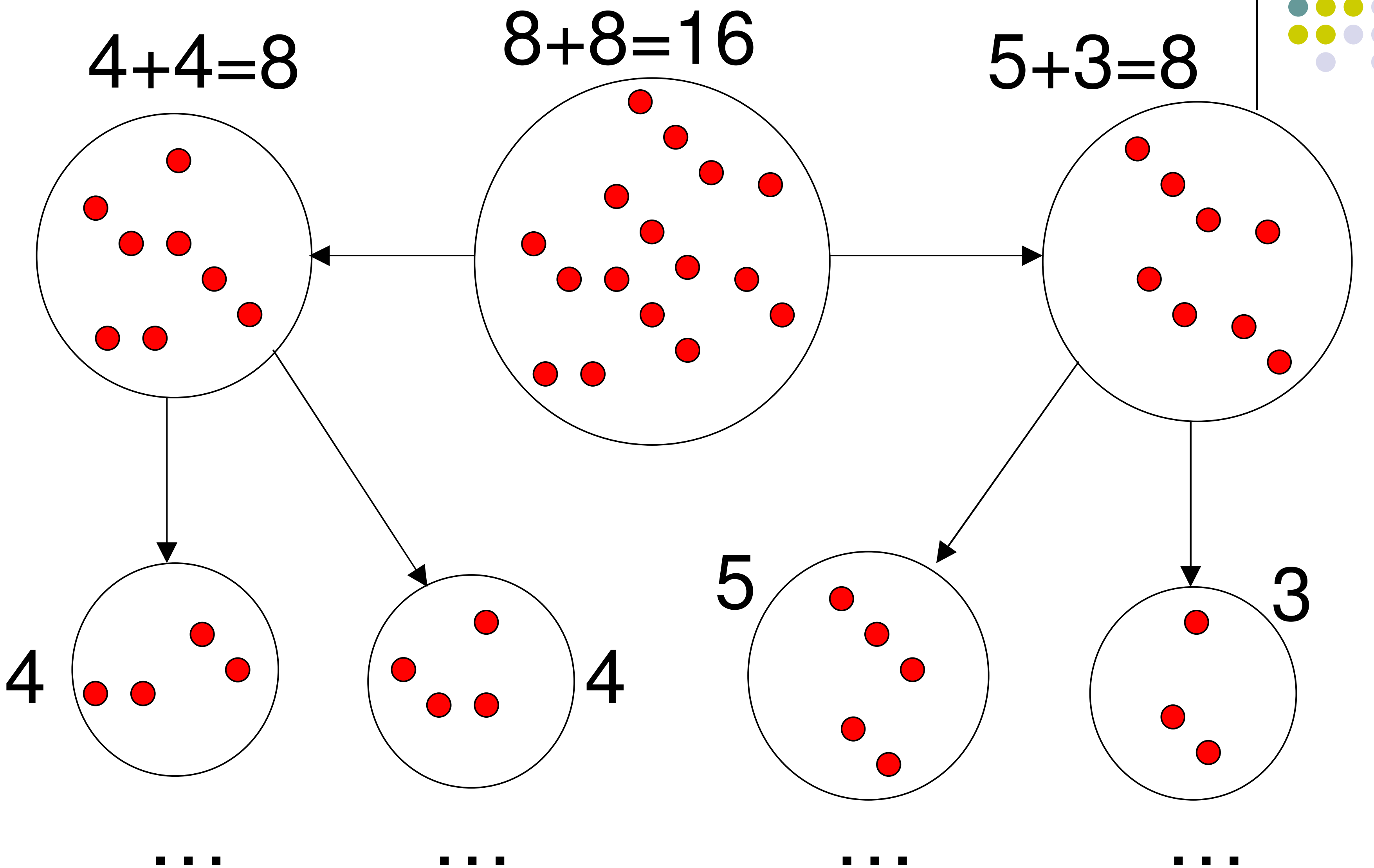
Overview

- Divide-and-conquer: example and definition
- Ladner and Fischer's divide-and-conquer simulation of a sequential FSA
- Motivation for new result
- Main lemma
- Construction/proof sketch
- Areas for future investigation

Definition: (Symmetric) Finite-State Automaton

- Input alphabet X, output alphabet O, state space Q, all finite. Initial state q_{o} in Q.
- Transition function f_σ:$Q \rightarrow Q$ for each σ in Σ.
- Post-processing function Π:$Q \rightarrow O$
  - generalizes set of accepting states
- On input string $\alpha\beta\gamma\ldots\omega$ start in state q_{o}, then apply f_α to current state, then f_β, etc.
- Output value is $\Pi(q)$ where q is final state.
- "Symmetric": permute input ⇨ same output.

Divide-and-Conquer Example



Elements of a Divide-and-Conquer Automaton (DCA)

- Inputs partitioned arbitrarily into 2 parts.
- Recurse until trivial case (e.g., one input).
- Intermediate results combined with a deterministic 2-input function.
- Post-processing function Π maps the final intermediate result to the output alphabet (analogue to set of accepting states).
- If all sets finite, and output is independent of how the division was performed, this is a *Divide-and-Conquer Automaton* (*DCA*).

[Ladner & Fischer '77] Functional Composition Idea

- Allows us to simulate any FSA with a DCA.
- Output of FSA on input $\alpha\beta\gamma\ldots\omega$ is

$$\Pi(f_\omega(\ldots f_\gamma(f_\beta(f_\alpha(q_0)))\ldots))$$
$$= \Pi(f_\omega \circ \cdots \circ f_\gamma \circ f_\beta \circ f_\alpha(q_0))$$

- Composition ($\circ$) is associative; we can hence use D & C to compute composition of f's.
- In post-processing, $f_\omega \circ \cdots \circ f_\alpha \mapsto \Pi(f_\omega \circ \cdots \circ f_\alpha(q_0))$.
- # intermediate states: #{ f | $f : Q \to Q$} = $|Q|^{|Q|}$.

[Ladner & Fischer '77] Illustration

- For string $S = \alpha\beta\gamma\ldots\omega$ define $f_S = f_\omega \circ \cdots \circ f_\gamma \circ f_\beta \circ f_\alpha$

- Compute composition

$$f_S = f_{\lambda\mu\ldots\omega} \circ f_{\alpha\beta\ldots\kappa}$$

$$S = \alpha\beta\ldots\kappa\lambda\mu\ldots\omega$$

$$\alpha\beta\ldots\kappa \qquad f_{\alpha\beta\ldots\kappa}$$

$$\lambda\mu\ldots\omega \qquad f_{\lambda\mu\ldots\omega}$$

$$\ldots$$

$$\alpha$$

- Look up f_α, store as table

f_α

q_1	q_2	q_3	$\ldots$
$f_\alpha(q_1)$	$f_\alpha(q_2)$	$f_\alpha(q_3)$	$\ldots$

Motivation (1/2)

- *Symmetric Network Computation*, 2006 with Santosh Vempala.
  - Want FSA-based network computation in non-regular graphs with global symmetry and local symmetry.
- Each node is a copy of **the same symmetric** FSA; reads neighbours' states as inputs to compute next state during a transition.
  - asynchronous, probabilistic
  - Can elect leader, do biconnectivity. Firing squad?
  - Demo

Motivation (2/2)

- Look at one node
- We showed: {*symmetric functions computable by FSAs*} = {*symmetric functions computable by DCAs*} = {*ultimately periodic symmetric functions*}.
  - E.g., determine if at least 10 neighbours, or if number of purple neighbours is odd.
- cf. Ladner & Fischer: {*functions computable by FSAs*} = {*functions computable by DCAs*}.

Statement of New Result

- Until now all FSA-to-DCA conversions entail an exponential increase in the state space size (i.e., from $|Q|$ to $|Q|^{|Q|}$).

- New contribution: a way to convert a **symmetric** FSA to a DCA **without any increase** in size of state space.

Main Lemma (1/3)

- State q of FSA <u>inaccessible</u> if no string S has

$$f_S(q_o) = q.$$

- States q, q' are <u>indistinguishable</u> if for all S,

$$\Pi(f_S(q)) = \Pi(f_S(q')).$$

- If an FSA has no inaccessible states and no indistinguishable pairs, it is <u>irredundant</u>.
  - Given an FSA, straightforward to construct an equivalent FSA that is irredundant. E.g., delete states unreachable from q_o in transition graph.
  - Merge indistinguishable pairs similarly.

Main Lemma (2/3)

- Main lemma: irredundant symmetric FSAs have commuting transition functions.
  - Symmetry is a black-box property; add to it the innocent-looking "white-box" property of irredundancy and we get a "white-box" result (commuting transition functions).
- We then obtain a simple D&C construction with a reasonably short proof of correctness.

Main Lemma (3/3)

In a symmetric irredundant FSA, f_σ's commute.

- Say input symbols σ, σ' have $f_\sigma(f_{\sigma'}(q)) \neq f_{\sigma'}(f_\sigma(q))$
- By distinguishability some string S has

$$\Pi(f_S(f_\sigma(f_{\sigma'}(q)))) \neq \Pi(f_S(f_{\sigma'}(f_\sigma(q)))).$$

- By accessibility some string T has $q = f_T(q_0)$.
- ⁂ $\Pi(f_S(f_\sigma(f_{\sigma'}(f_T(q_0))))) \neq \Pi(f_S(f_{\sigma'}(f_\sigma(f_T(q_0))))).$
- But this says that outputs on inputs $T\sigma'\sigma S$ and $T\sigma\sigma' S$ differ, contradicting symmetry. $\square$

Construction, Proof Idea (1/2)

- Given: symmetric irredundant FSA.
- For each state q fix a <u>*representative string*</u> $r[q]$ that brings FSA to state q from q_o,

$$f_{r[q]}(q_o)=q$$

- Given any other string S with $f_S(q_o)=q$, S and $r[q]$ are interchangeable at start of input.
- Using commutativity of f's, we obtain

$$f_S = f_{r[q]}$$

i.e., interchangeable *anywhere* in input.

Construction, Proof Idea (2/2)

Definition of the DCA to simulate the FSA

- DCA intermediate state space = FSA state space; its size could only have decreased when redundancy was removed.
- Base case: map input character σ to $f_\sigma(q_0)$.
- Combining: map pair (q, q') to $f_{r[q']}(q)$.
- Post-processing: use same Π as FSA did.
  - Correct & independent of how dividing performed due to interchangeability.

Areas for Future Investigation

- Ladner-Fischer result extends to stochastic or nondeterministic automata. Seems for sure "main lemma" fails and no "small" DCA exist but we yet lack a convincing counterexample.
- Suppose FSA has implicit transition function: k-bit binary states/inputs, transition function is a poly-time Turing machine. Our transformation takes $exp(k)$ time to ensure irredundancy, can one do better?
- Thanks for listening!

References

- Preprint: http://arxiv.org/abs/0708.0580
- R. E. Ladner and M. J. Fischer. Parallel prefix computation. J. ACM, 27(4):831–838, 1980. Preliminary version appeared in Proc. 6th International Conf. Parallel Processing, pages 218–223, 1977.
- D. Pritchard and S. Vempala. Symmetric network computation. In Proc. 18th SPAA, pages 261–270, 2006.
  - Model demo: http://www.math.uwaterloo.ca/~dagpritc/fssga.html