On the Computational Complexity of MapReduce

Ben Fish

Joint work with
Jeremy Kun, Ádám D. Lelkes, Lev Reyzin, and György Turán

UIC

October 7th, 2015

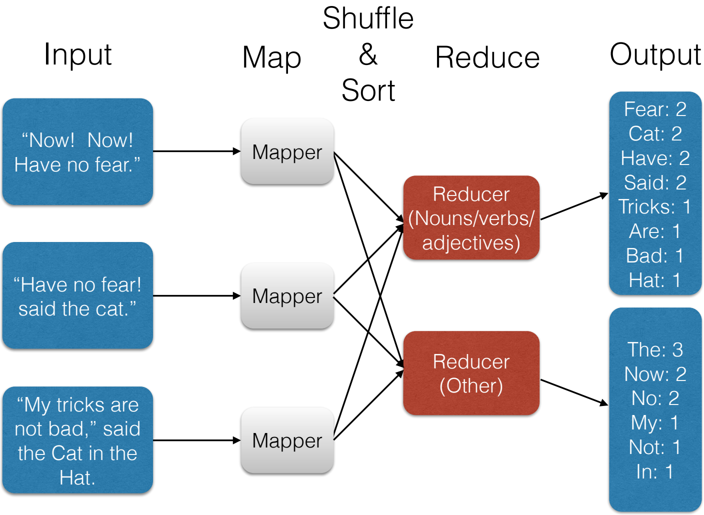
Intro

- MRC: MapReduce as complexity class
- Upper bounds for MRC
- Hierarchy theorem for MRC

Why complexity theory?

- Can answer whether more resources gives more power
- Containments between complexity classes solve lots of problems at once
- NP-hardness is evidence of a lack of a poly-time algorithm

MapReduce example



MapReduce - map, reduce functions

- Reducer $\sim$ processor
- Mapper $\sim$ which processor to send a given bit string

Definition (Slightly more formally)

Mapper $\mu : \langle k, v \rangle \to \langle k'_1, v'_1 \rangle, \ldots, \langle k'_s, v'_s \rangle$
Reducer $\rho : k, \langle v_1, \ldots, v_m \rangle \to \langle v'_1, \ldots, v'_m \rangle$

Definition (MRC machine)

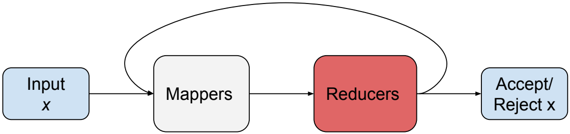
An MRC machine on R rounds is a list of alternating mappers and reducers $M_R = (\mu_1, \rho_1, \ldots, \mu_R, \rho_R)$.

Converting MapReduce into a decision problem framework

Recall every decision problem (yes/no answers) has an associated language L of strings corresponding with the yes answers of the decision problem.

Definition

An MRC machine M_R *accepts* a string if the reducers in the last round accept the string and *decides* a language L if $x \in L$ iff M accepts x.



MRC

Definition (MRC, informal, from Karloff et al. 2010)

A language L is in $\mathrm{MRC}[f(n), g(n)]$ if there is an MRC machine $M_R = (\mu_1, \rho_1, \ldots, \mu_R, \rho_R)$ that decides L and constant $c < 1$ such that for an input of size n,

1. $R = O(f(n))$
2. Each mapper and reducer takes $O(g(n))$ time
3. Each mapper and reducer takes $O(n^c)$ space
4. Each mapper outputs no more than $O(n^c)$ distinct keys

Definition

$$\mathrm{MRC}^0 := \bigcup_{k \in \mathbb{N}} \mathrm{MRC}[1, n^k].$$

Upper bounds

Theorem

$\mathrm{SPACE}(o(\log n)) \subseteq \mathrm{MRC}^0$.

Theorem (Warm-up)

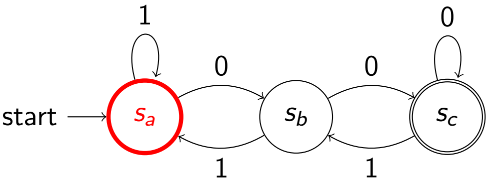
$\mathrm{REGULAR} \subsetneq \mathrm{MRC}^0$

Example

Checking whether a string contains a given regular expression is in MRC^0.

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

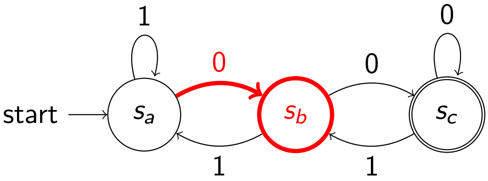


ρ_3

start	finish
s_a	
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

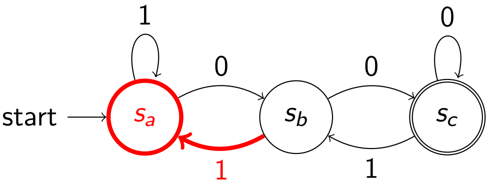


ρ_3

start	finish
s_a	
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

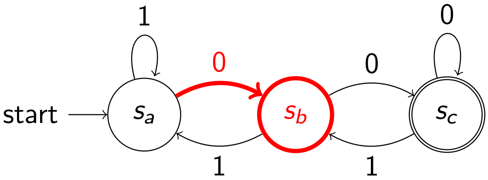


ρ_3

start	finish
s_a	
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

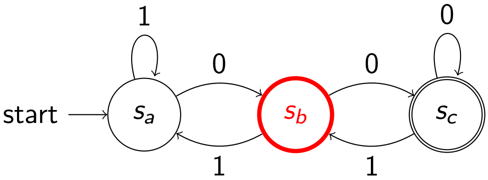


ρ_3

start	finish
s_a	s_b
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\,\underbrace{101}_{\rho_2}\,\underbrace{010}_{\rho_3}$

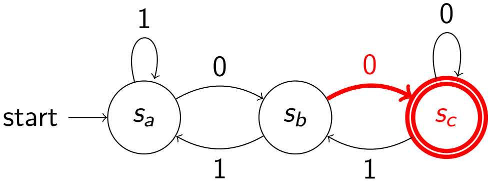


ρ_3

start	finish
s_a	s_b
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

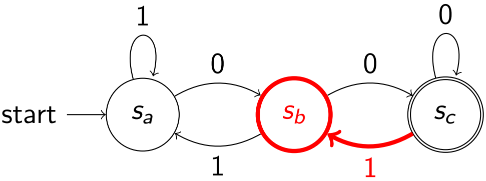


ρ_3

start	finish
s_a	s_b
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

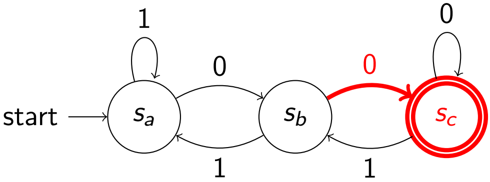


ρ_3

start	finish
s_a	s_b
s_b	
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

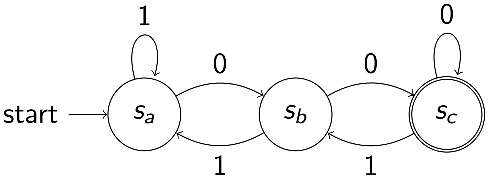


ρ_3

start	finish
s_a	s_b
s_b	s_c
s_c	

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$



ρ_3

start	finish
s_a	s_b
s_b	s_c
s_c	s_c

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

ρ_1

start	finish
s_a	s_c
s_b	s_c
s_c	s_c

ρ_2

start	finish
s_a	s_a
s_b	s_a
s_c	s_b

ρ_3

start	finish
s_a	s_b
s_b	s_c
s_c	s_c

Proof idea for upper bounds

Input: $\underbrace{000}_{\rho_1}\underbrace{101}_{\rho_2}\underbrace{010}_{\rho_3}$

ρ_1

start	finish
s_a	s_c
s_b	s_c
s_c	s_c

ρ_2

start	finish
s_a	s_a
s_b	s_a
s_c	s_b

ρ_3

start	finish
s_a	s_b
s_b	s_c
s_c	s_c

Second round:

$$s_a \xrightarrow{\rho_1} s_c \xrightarrow{\rho_2} s_b \xrightarrow{\rho_3} s_c$$

Extension to sub-logarithmic space

Theorem

$\mathrm{SPACE}(o(\log n)) \subseteq \mathrm{MRC}^0$.

- Instead of simulating a DFA, we need to simulate a TM with a read-only input tape and a read/write work tape.
- Again, each processor computes, for all input states, what state the TM ends up in
- Now a 'state' consists of:
  - Work tape configuration ($2^{o(\log n)} \cdot o(\log n)$ configurations)
  - TM state (constant number of states)
  - Side of the input chunk the read head starts on (left/right)

Hierarchy theorem

Theorem

Suppose the Exponential Time Hypothesis holds. Then for every α, β there exist $\mu > \alpha$ and $\nu > \beta$ such that

$$\mathsf{MRC}[n^\alpha, n^\beta] \subsetneq \mathsf{MRC}[n^\mu, n^\beta]$$

and

$$\mathsf{MRC}[n^\alpha, n^\beta] \subsetneq \mathsf{MRC}[n^\alpha, n^\nu].$$

"Sufficiently more rounds or time per round gives you strictly more power."

ETH

Conjecture (Exponential Time Hypothesis, Impagliazzo, Paturi, and Zane)

3-SAT *is not in* $\mathrm{TIME}(2^{cn})$ *for some* $c > 0$.

Hierarchy proof via TISP

TISP$(f(n), g(n))$ is the class of languages solvable on a Turing machine using $f(n)$ time and $g(n)$ space.

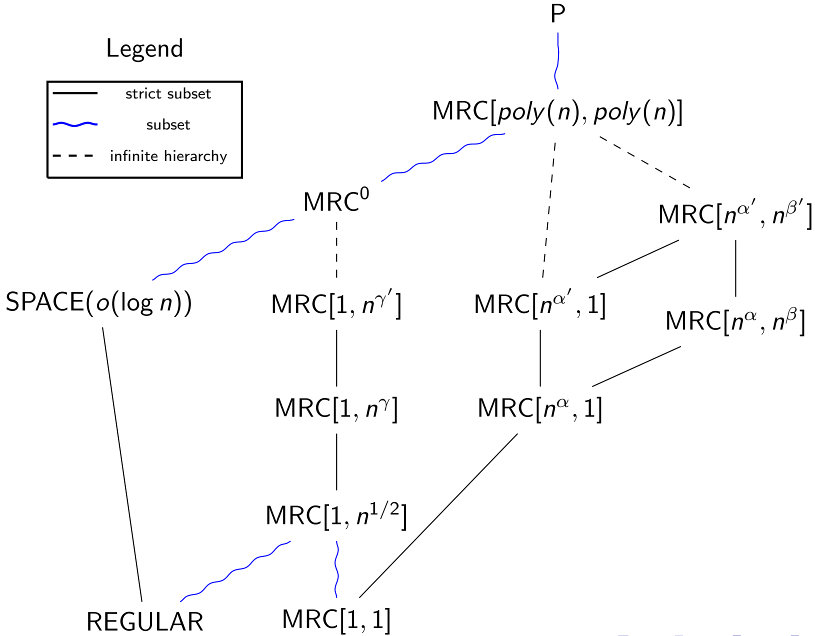
Lemma

For every $\alpha, \beta \in \mathbb{N}$ the following holds:

$$\mathsf{TISP}(n^\alpha, n) \subseteq \mathsf{MRC}[n^\alpha, 1] \subseteq \mathsf{MRC}[n^\alpha, n^\beta] \subseteq \mathsf{TISP}(n^{\alpha+\beta+2}, n^2).$$

The proof of the hierarchy theorem comes from the above and a padding/simulation argument to move the hardness guaranteed by ETH into the appropriate MRC class.

Summary (Assuming the ETH holds)



Open Problems

- Is it possible to remove the dependence on the ETH?
- Where does SPACE($\log(n)$) lie?
- Is (undirected) graph connectivity in MRC^0?
- Does $\mathrm{MRC}[poly(n), poly(n)] = P$?

Corollary

$$\begin{aligned}\mathrm{SPACE}(\log(n)) &\subseteq \mathrm{TISP}(\mathrm{poly}(n), \log n) \subseteq \mathrm{TISP}(\mathrm{poly}(n), n) \\ &\subseteq \mathrm{MRC}[\mathrm{poly}(n), 1] \subseteq \mathrm{MRC}[\mathrm{poly}(n), \mathrm{poly}(n)] \subseteq \mathrm{P}.\end{aligned}$$