Complexity of Regular Functions

Eric Allender [1]   **Ian Mertz** [2]

Rutgers University

[1] *allender@rutgers.edu*

[2] *iwmertz@gmail.com*

March 2, 2015

Overview

Next up...

In the beginning...

Problem: find the minimum cost path in a graph

In the beginning...

Problem: find the minimum cost path in a graph

New twist: alternative costs ("edge e has weight 10, or 15 if you go through edge f at some point")

In the beginning...

Problem: find the minimum cost path in a graph

New twist: alternative costs ("edge e has weight 10, or 15 if you go through edge f at some point")

A *weighted automaton* returns the smallest value of an accepting path.

In the beginning...

Problem: find the minimum cost path in a graph

New twist: alternative costs ("edge e has weight 10, or 15 if you go through edge f at some point")

A *weighted automaton* returns the smallest value of an accepting path.

...but to calculate this, we need non-determinism, and we still don't get to deal with some types of alternative costs.

In the beginning...

Problem: find the minimum cost path in a graph

New twist: alternative costs ("edge e has weight 10, or 15 if you go through edge f at some point")

A *weighted automaton* returns the smallest value of an accepting path.

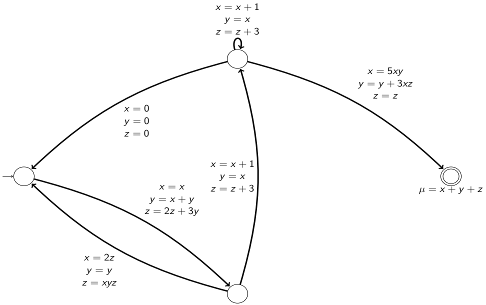
...but to calculate this, we need non-determinism, and we still don't get to deal with some types of alternative costs.

How do we make this model work?

Cost Register Automata (CRA)

Definition[Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]

A *cost-register automaton* is a deterministic finite-state automaton augmented with a finite set of registers that store elements of an algebraic domain. A computation step consists of consuming the next input symbol, transitioning to a new state based on that input symbol, and updating each register based on a function over the algebra.



We know a lot about CRAs

We know a lot about CRAs

- Equivalence:

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers
  [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost:

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost: usually polytime [Alur, Freilich, Raghothaman]

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost: usually polytime [Alur, Freilich, Raghothaman]
- Evaluation:

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost: usually polytime [Alur, Freilich, Raghothaman]
- Evaluation: ....

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost: usually polytime [Alur, Freilich, Raghothaman]
- Evaluation: ....

The punchline: we don't know how quickly we can even evaluate CRAs.

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost: usually polytime [Alur, Freilich, Raghothaman]
- Evaluation: ....

The punchline: we don't know how quickly we can even evaluate CRAs.

It is known that some situations give exponential output, while others are certainly within P, but we don't know *where* in P they are.

We know a lot about CRAs

- Equivalence: usually polytime in states and exponential in registers [Alur, D'Antoni, Deshmukh, Raghothaman, Yuan]
- Min-Cost: usually polytime [Alur, Freilich, Raghothaman]
- Evaluation: ....

The punchline: we don't know how quickly we can even evaluate CRAs.

It is known that some situations give exponential output, while others are certainly within P, but we don't know *where* in P they are.

Our results

Where in P they are.

Next up...

The classes

NC^1: problems that can be decided by uniform poly-size log-depth boolean circuits with AND and OR gates having fan-in 2

The classes

NC^1: problems that can be decided by uniform poly-size log-depth boolean circuits with AND and OR gates having fan-in 2

$\#\mathsf{NC}^1$: functions that can be represented by uniform poly-size log-depth arithmetic circuits over $\mathbb{N}$ with $+$ and $\times$ gates having fan-in 2

The classes

NC^1: problems that can be decided by uniform poly-size log-depth boolean circuits with AND and OR gates having fan-in 2

$\#\mathsf{NC}^1$: functions that can be represented by uniform poly-size log-depth arithmetic circuits over $\mathbb{N}$ with $+$ and $\times$ gates having fan-in 2

GapNC^1: functions that can be represented by the difference of two $\#\mathsf{NC}^1$ functions

The classes

NC^1: problems that can be decided by uniform poly-size log-depth boolean circuits with AND and OR gates having fan-in 2

$\#NC^1$: functions that can be represented by uniform poly-size log-depth arithmetic circuits over $\mathbb{N}$ with $+$ and $\times$ gates having fan-in 2

$GapNC^1$: functions that can be represented by the difference of two $\#NC^1$ functions

L: problems that can be decided by Turing machines with only a log-size work tape

The classes

NC^1: problems that can be decided by uniform poly-size log-depth boolean circuits with AND and OR gates having fan-in 2

$\#NC^1$: functions that can be represented by uniform poly-size log-depth arithmetic circuits over $\mathbb{N}$ with $+$ and $\times$ gates having fan-in 2

$GapNC^1$: functions that can be represented by the difference of two $\#NC^1$ functions

L: problems that can be decided by Turing machines with only a log-size work tape

AC^1: problems that can be decided by uniform poly-size log-depth circuits with AND and OR gates having unbounded fan-in

The classes

NC^1: problems that can be decided by uniform poly-size log-depth boolean circuits with AND and OR gates having fan-in 2

$\#NC^1$: functions that can be represented by uniform poly-size log-depth arithmetic circuits over $\mathbb{N}$ with $+$ and $\times$ gates having fan-in 2

$GapNC^1$: functions that can be represented by the difference of two $\#NC^1$ functions

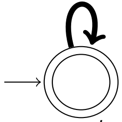
L: problems that can be decided by Turing machines with only a log-size work tape

AC^1: problems that can be decided by uniform poly-size log-depth circuits with AND and OR gates having unbounded fan-in

$$NC^1 \subseteq \#NC^1 \subseteq GapNC^1 \subseteq L \subseteq AC^1 \subseteq P$$

A natural barrier

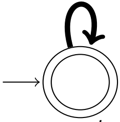
$$r_i = \prod_{j=1}^{k} r_j \ \forall i$$



$$\mu = \prod_{i=1}^{k} r_i$$

A natural barrier

$$r_i = \prod_{j=1}^{k} r_j \ \forall i$$

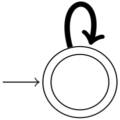


$$\mu = \prod_{i=1}^{k} r_i$$

We'll start with $r_i = c \ \forall i$ for some $c > 1$

A natural barrier

$$r_i = \prod_{j=1}^{k} r_j \ \forall i$$



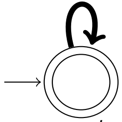
$$\mu = \prod_{i=1}^{k} r_i$$

We'll start with $r_i = c \ \forall i$ for some $c > 1$

After one step, $r_i = c^k$. Then $r_i = (c^k)^k = c^{k^2}$. Then c^{k^3}. Then...

A natural barrier

$$r_i = \prod_{j=1}^{k} r_j \ \forall i$$



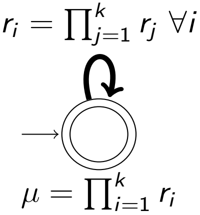
$$\mu = \prod_{i=1}^{k} r_i$$

We'll start with $r_i = c \ \forall i$ for some $c > 1$

After one step, $r_i = c^k$. Then $r_i = (c^k)^k = c^{k^2}$. Then c^{k^3}. Then...

Wait, this is c^{k^n}!

A natural barrier



We'll start with $r_i = c \ \forall i$ for some $c > 1$

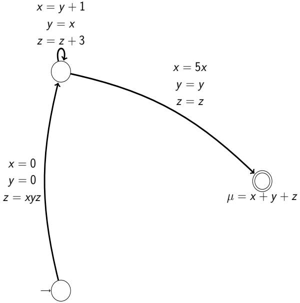
After one step, $r_i = c^k$. Then $r_i = (c^k)^k = c^{k^2}$. Then c^{k^3}. Then...

Wait, this is c^{k^n}! That takes $O(k^n)$ bits just to write down; no way we're getting that in P.

Copyless CRAs (CCRAs)

Definition[Alur, Freilich, Raghothaman]

A *copyless CRA* is a CRA where for any transition, no register can be used more than once to update the registers on that transition.



Copyless CRAs (CCRAs)

We claim that this dodges the natural barrier from before.

Copyless CRAs (CCRAs)

We claim that this dodges the natural barrier from before.

Again start with all registers holding c. If you multiply them all together, you get c^k again.

Copyless CRAs (CCRAs)

We claim that this dodges the natural barrier from before.

Again start with all registers holding c. If you multiply them all together, you get c^k again.

But since you used up all your input variables, the rest can only be set to constants.

Copyless CRAs (CCRAs)

We claim that this dodges the natural barrier from before.

Again start with all registers holding c. If you multiply them all together, you get c^k again.

But since you used up all your input variables, the rest can only be set to constants.

So even if you repeat this n times, you just get c^{nk}, which can be written in $O(nk)$ bits.

Copyless CRAs (CCRAs)

We claim that this dodges the natural barrier from before.

Again start with all registers holding c. If you multiply them all together, you get c^k again.

But since you used up all your input variables, the rest can only be set to constants.

So even if you repeat this n times, you just get c^{nk}, which can be written in $O(nk)$ bits.

If we consider the algebraic degree of the functions we are representing, then CCRAs have $n^{O(1)}$-bounded algebraic degree, unlike the problematic exponential degree functions from before.

Next up...

Copyless monoids

Theorem

All functions computable by CCRAs over $(\mathbb{Z}, +)$ are computable in NC^1

Copyless monoids

Theorem

All functions computable by CCRAs over $(\mathbb{Z}, +)$ are computable in NC^1

CCRAs over $(\mathbb{Z}, +)$ are equivalent to CRAs with register updates of the form $r \leftarrow r' + c$

Copyless monoids

Theorem

All functions computable by CCRAs over $(\mathbb{Z}, +)$ are computable in NC^1

CCRAs over $(\mathbb{Z}, +)$ are equivalent to CRAs with register updates of the form $r \leftarrow r' + c$

In NC^1 we can build a constant-width graph with n layers, with an arrow from $v_{i,j}$ to $v_{i+1,l}$ if on the ith input variable there is a transition of the form $r_l \leftarrow r_j + c$.

Copyless monoids

Theorem

All functions computable by CCRAs over $(\mathbb{Z},+)$ are computable in NC^1

CCRAs over $(\mathbb{Z},+)$ are equivalent to CRAs with register updates of the form $r \leftarrow r' + c$

In NC^1 we can build a constant-width graph with n layers, with an arrow from $v_{i,j}$ to $v_{i+1,l}$ if on the ith input variable there is a transition of the form $r_l \leftarrow r_j + c$.

There is a unique path from the first layer to any chosen vertex v_i on the last layer (given that they all have indegree 1), and we can find it in NC^1. Tracing back along this path gives us all the constants that sum up to give the final value of r_i.

Copyless monoids

Theorem

All functions computable by CCRAs over $(\mathbb{Z}, +)$ are computable in NC^1

CCRAs over $(\mathbb{Z}, +)$ are equivalent to CRAs with register updates of the form $r \leftarrow r' + c$

In NC^1 we can build a constant-width graph with n layers, with an arrow from $v_{i,j}$ to $v_{i+1,l}$ if on the ith input variable there is a transition of the form $r_l \leftarrow r_j + c$.

There is a unique path from the first layer to any chosen vertex v_i on the last layer (given that they all have indegree 1), and we can find it in NC^1. Tracing back along this path gives us all the constants that sum up to give the final value of r_i.

Theorem

All functions computable by CCRAs over $(\Gamma^*, \circ)$ are computable in NC^1

Copyless over $\{+, \times\}$

Theorem

All functions computed by CCRAs over $(\mathbb{Z}, +, \times)$ are computable in GapNC^1

Copyless over $\{+, \times\}$

Theorem

All functions computed by CCRAs over $(\mathbb{Z}, +, \times)$ are computable in GapNC^1

Take a function f that we can compute, and its input x. In NC^1 we can map it to an arithmetic circuit that computes $f(x)$ by having gates that compute the register updates at every step of our computation.

Copyless over $\{+, \times\}$

Theorem

All functions computed by CCRAs over $(\mathbb{Z}, +, \times)$ are computable in GapNC^1

Take a function f that we can compute, and its input x. In NC^1 we can map it to an arithmetic circuit that computes $f(x)$ by having gates that compute the register updates at every step of our computation.

Note that each outdegree is 1, so we have a formula. Using a nice result of [Buss et al], we can take such a circuit and turn it into a log-depth arithmetic formula that contains some boolean gates.

Copyless over $\{+, \times\}$

Theorem

All functions computed by CCRAs over $(\mathbb{Z}, +, \times)$ are computable in GapNC^1

Take a function f that we can compute, and its input x. In NC^1 we can map it to an arithmetic circuit that computes $f(x)$ by having gates that compute the register updates at every step of our computation.

Note that each outdegree is 1, so we have a formula. Using a nice result of [Buss et al], we can take such a circuit and turn it into a log-depth arithmetic formula that contains some boolean gates.

The boolean circuitry can all be replaced since $\mathrm{NC}^1 \subseteq \#\mathrm{NC}^1$, and so we get that $f \in \mathrm{GapNC}^1$ (we are working over $\mathbb{Z}$, so negative results are possible, and so we do not have $\#\mathrm{NC}^1$)

Copyless over $\{\max(min), +\}$ and $\{\max, \circ\}$

Similar tricks can give us two more results:

Copyless over $\{\max(\mathit{min}), +\}$ and $\{\max, \circ\}$

Similar tricks can give us two more results:

Theorem

All functions computed by CCRAs over $(\mathbb{N} \cup \{\infty\}, \max, +)$ are computable in $\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$ (meaning functions expressible as $g(f(x))$ for $f \in \mathsf{NC}^1$, $g \in \#\mathsf{NC}^1_{trop}$)

Copyless over $\{\max(min), +\}$ and $\{\max, \circ\}$

Similar tricks can give us two more results:

Theorem

All functions computed by CCRAs over $(\mathbb{N} \cup \{\infty\}, \max, +)$ are computable in $\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$ (meaning functions expressible as $g(f(x))$ for $f \in \mathsf{NC}^1$, $g \in \#\mathsf{NC}^1_{trop}$)

We do the same calculations as above, but since we do not yet know that $\#\mathsf{NC}^1_{trop} \in$GapNC^1, we just have that it is NC^1 reducible to $\#\mathsf{NC}^1_{trop}$.

Copyless over $\{\max(min), +\}$ and $\{\max, \circ\}$

Similar tricks can give us two more results:

Theorem

All functions computed by CCRAs over $(\mathbb{N} \cup \{\infty\}, \max, +)$ are computable in $\mathrm{NC}^1(\#\mathrm{NC}^1_{trop})$ (meaning functions expressible as $g(f(x))$ for $f \in \mathrm{NC}^1$, $g \in \#\mathrm{NC}^1_{trop}$)

We do the same calculations as above, but since we do not yet know that $\#\mathrm{NC}^1_{trop} \in$GapNC^1, we just have that it is NC^1 reducible to $\#\mathrm{NC}^1_{trop}$.

Theorem

All functions computed by CCRAs over $(\Gamma^*, \max, \circ)$ are computable in AC^1

Copyless over $\{\max(min), +\}$ and $\{\max, \circ\}$

Similar tricks can give us two more results:

Theorem

All functions computed by CCRAs over $(\mathbb{N} \cup \{\infty\}, \max, +)$ are computable in $\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$ (meaning functions expressible as $g(f(x))$ for $f \in \mathsf{NC}^1$, $g \in \#\mathsf{NC}^1_{trop}$)

We do the same calculations as above, but since we do not yet know that $\#\mathsf{NC}^1_{trop} \in$GapNC^1, we just have that it is NC^1 reducible to $\#\mathsf{NC}^1_{trop}$.

Theorem

All functions computed by CCRAs over $(\Gamma^*, \max, \circ)$ are computable in AC^1

All functions computable by poly size, poly degree circuits over $(\Gamma^*, \max, \circ)$ lie in AC^1 [AJMV].

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \mathsf{max}, +)$:

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$:

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$:

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$:

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$: trivially L, probably better

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$: trivially L, probably better

$(\mathbb{Z}, +, \times c)$:

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$: trivially L, probably better

$(\mathbb{Z}, +, \times c)$: GapNC^1 (tight)

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$: trivially L, probably better

$(\mathbb{Z}, +, \times c)$: GapNC^1 (tight)

$(\Gamma^*, \max, \circ s)$:

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$: trivially L, probably better

$(\mathbb{Z}, +, \times c)$: GapNC^1 (tight)

$(\Gamma^*, \max, \circ s)$: AC^1

Non-copyless CRAs

$(\mathbb{N} \cup \{\infty\}, \max, +)$: haven't been able to beat P

$(\mathbb{Z}, +, \times)$: completely unbounded

$(\Gamma^*, \max, \circ)$: even more unbounded

$(\mathbb{N} \cup \{\infty\}, \max, +c)$: trivially L, probably better

$(\mathbb{Z}, +, \times c)$: GapNC^1 (tight)

$(\Gamma^*, \max, \circ s)$: AC^1

Moral of the story: replacing copyless with $\otimes c$ often gives the same result.

Next up...

A mandatory table and open problems

	Copyless $\{\otimes\}$	Copyless $\{\oplus, \otimes\}$	$\{\oplus, \otimes c\}$
$\{\mathbb{Z}, +, \times\}$		GapNC^1	GapNC^1
$\{\mathbb{N} \cup \{\infty\}, \max, +\}$	NC^1 (tight)	$\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$	L
$\{\Gamma^*, \max, \circ\}$	NC^1 (tight)	AC^1	AC^1

A mandatory table and open problems

	Copyless $\{\otimes\}$	Copyless $\{\oplus, \otimes\}$	$\{\oplus, \otimes c\}$
$\{\mathbb{Z}, +, \times\}$		GapNC^1	GapNC^1
$\{\mathbb{N} \cup \{\infty\}, \max, +\}$	NC^1 (tight)	$\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$	L
$\{\Gamma^*, \max, \circ\}$	NC^1 (tight)	AC^1	AC^1

As of now, most of the bounds are not necessarily tight.

A mandatory table and open problems

	Copyless $\{\otimes\}$	Copyless $\{\oplus, \otimes\}$	$\{\oplus, \otimes c\}$
$\{\mathbb{Z}, +, \times\}$		GapNC^1	GapNC^1
$\{\mathbb{N} \cup \{\infty\}, \max, +\}$	NC^1 (tight)	$\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$	L
$\{\Gamma^*, \max, \circ\}$	NC^1 (tight)	AC^1	AC^1

As of now, most of the bounds are not necessarily tight.

Goal: keep reducing the complexity of these problems, or prove their completeness.

(probably the first place to start is anything that is still listed as P, or L)

A mandatory table and open problems

	Copyless $\{\otimes\}$	Copyless $\{\oplus, \otimes\}$	$\{\oplus, \otimes c\}$
$\{\mathbb{Z}, +, \times\}$		GapNC^1	GapNC^1
$\{\mathbb{N} \cup \{\infty\}, \max, +\}$	NC^1 (tight)	$\mathsf{NC}^1(\#\mathsf{NC}^1_{trop})$	L
$\{\Gamma^*, \max, \circ\}$	NC^1 (tight)	AC^1	AC^1

As of now, most of the bounds are not necessarily tight.

Goal: keep reducing the complexity of these problems, or prove their completeness.

(probably the first place to start is anything that is still listed as P, or L)

If you are in computer-aided verification, there's a whole lot of literature on regular functions, so maybe these new bounds will make CRAs more attractive for using in algorithms?

That's all Folks!