The Complexity of Composition

New Approaches to Depth and Space

Ian Mertz

Final Oral Examination
University of Toronto

August 16, 2022

Composition

When can we save by doing two separate things together?

Composition

When can we save by doing two separate things together?

Non-computational example: running all your errands in one trip

Composition

Composition theorem:

Composition

Composition theorem:

- functions f and g

Composition

Composition theorem:

- functions f and g
- *composition* $f \circ g$

Composition

Composition theorem:

- functions f and g
- *composition* $f \circ g$
- *complexity measure* $s(\cdot)$

Composition

Composition theorem:

- functions f and g
- *composition* $f \circ g$
- *complexity measure* $s(\cdot)$

Central goal: determine whether

$$s(f \circ g) \approx s(f) + s(g)$$

or

$$s(f \circ g) \ll s(f) + s(g)$$

Composition and lower bounds

What happens if $s(f \circ g) \approx s(f) + s(g)$?

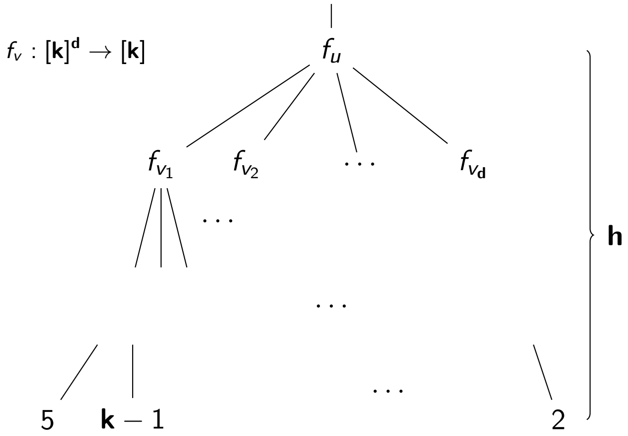
Composition and lower bounds

What happens if $s(f \circ g) \approx s(f) + s(g)$?

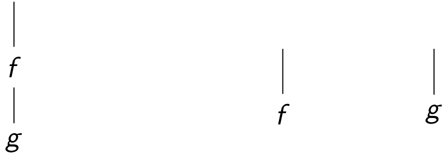
General idea: going from easier functions $f_1 \ldots f_m$ to a harder function $F = f_1 \circ \ldots \circ f_m$

Composition and lower bounds

Tree Evaluation Problem (TreeEval := $\mathsf{TreeEval}_{k,d,h}$)



Composition and lower bounds



"Two kinds of composition" at work:

- *sequential composition*: between layers
- *parallel composition*: within layers

Complexity measures

What does "save" mean? The measure matters!

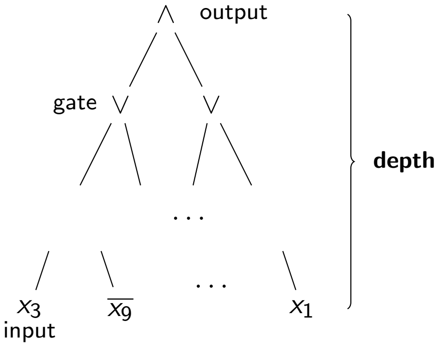
Complexity measures

What does "save" mean? The measure matters!

Errands example: can save yourself the *time* it takes to go downtown twice, but cannot save any *money* on the bills themselves.

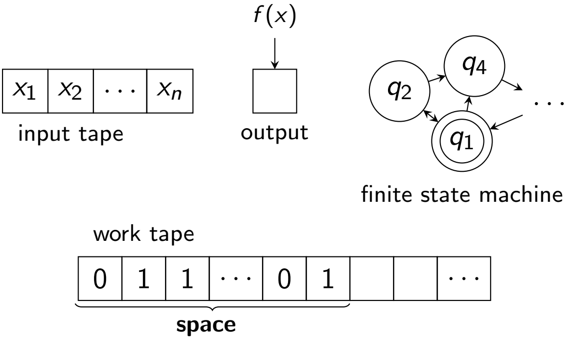
Complexity measures

Model 1: formulas



Complexity measures

Model 2: Turing Machines



Complexity measures

time efficient (P): Turing Machines with time $n^{O(1)}$

Complexity measures

time efficient (P): Turing Machines with time $n^{O(1)}$

depth efficient (NC^1): formulas of depth $O(\log n)$

Complexity measures

time efficient (P): Turing Machines with time $n^{O(1)}$

depth efficient (NC^1): formulas of depth $O(\log n)$
space efficient (L): Turing Machines with space $O(\log n)$

Complexity measures

time efficient (P): Turing Machines with time $n^{O(1)}$

depth efficient (NC^1): formulas of depth $O(\log n)$
space efficient (L): Turing Machines with space $O(\log n)$

Known: $NC^1 \subseteq P$ and $L \subseteq P$

Complexity measures

time efficient (P): Turing Machines with time $n^{O(1)}$

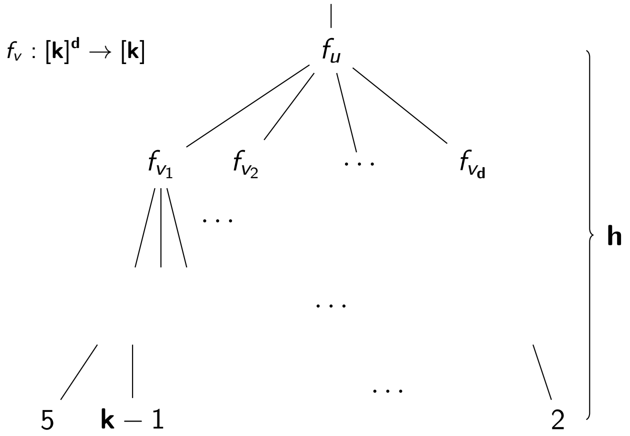
depth efficient (NC^1): formulas of depth $O(\log n)$
space efficient (L): Turing Machines with space $O(\log n)$

Known: $\mathrm{NC}^1 \subseteq \mathrm{P}$ and $\mathrm{L} \subseteq \mathrm{P}$

Central goal: show $\mathrm{NC}^1 \subsetneq \mathrm{P}$ and $\mathrm{L} \subsetneq \mathrm{P}$ using composition

Composition and lower bounds

$\text{TreeEval}_{k,d,h} \in \text{P}$: evaluating it bottom-up takes linear time



Composition and lower bounds

Conjecture 1 [KRW'95]: $\mathsf{TreeEval} \notin \mathsf{NC}^1$

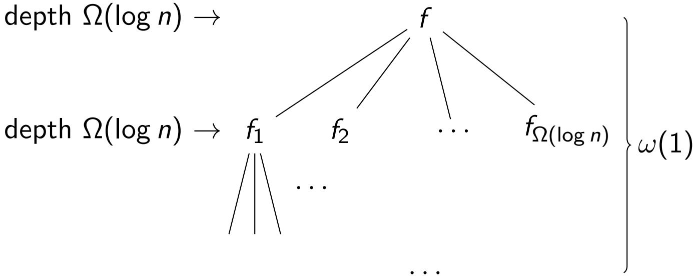
Conjecture 2 [CMWBS'12]: $\mathsf{TreeEval} \notin \mathsf{L}$

Composition and depth lower bounds

KRW conjecture [KRW'95]: $depth(f \circ g) \approx depth(f) + depth(g)$

Composition and depth lower bounds

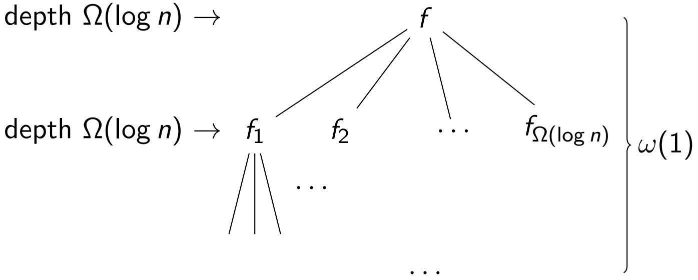
KRW conjecture [KRW'95]: $depth(f \circ g) \approx depth(f) + depth(g)$



Depth $\omega(1) \cdot \Omega(\log n)$

Composition and depth lower bounds

KRW conjecture [KRW'95]: $depth(f \circ g) \approx depth(f) + depth(g)$



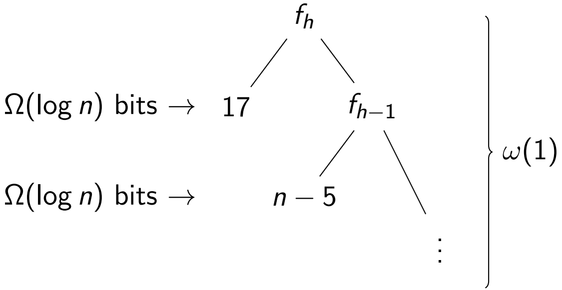
Depth $\omega(1) \cdot \Omega(\log n) \rightarrow \text{TreeEval}_{2,d,h} \notin \text{NC}^1$

Composition and space lower bounds

z-f conjecture [CMWBS'12]: computing f while remembering output of g requires space to compute f plus space to remember output of g

Composition and space lower bounds

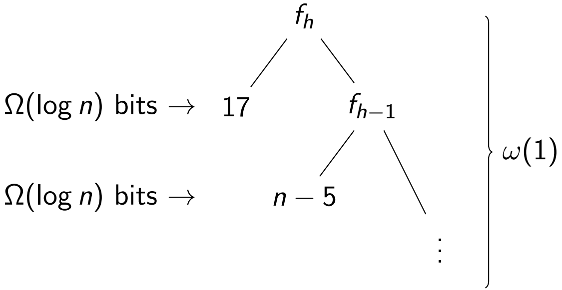
z-f conjecture [CMWBS'12]: computing f while remembering output of g requires space to compute f plus space to remember output of g



Space $\omega(1) \cdot \Omega(\log n)$

Composition and space lower bounds

z-f conjecture [CMWBS'12]: computing f while remembering output of g requires space to compute f plus space to remember output of g



Space $\omega(1) \cdot \Omega(\log n) \rightarrow \mathsf{TreeEval}_{k,2,h} \notin \mathsf{L}$

This thesis

Progress on both questions

This thesis

Progress on both questions...but in opposite directions!

This thesis

Progress on both questions...but in opposite directions!

Part I: getting closer to showing KRW conjecture is true

This thesis

Progress on both questions...but in opposite directions!

Part I: getting closer to showing KRW conjecture is true

Part II: unconditionally showing z-f conjecture is false

Part II

Space

Reusing space

Conjecture [CMWBS'12]: $\mathsf{TreeEval}_{k,2,h}$ requires space $\Omega(h \log k)$

Reusing space

Conjecture [CMWBS'12]: $\mathsf{TreeEval}_{k,2,h}$ requires space $\Omega(h \log k)$

Some interesting internal functions known to not be hard enough!

Reusing space

Conjecture [CMWBS'12]: $\mathsf{TreeEval}_{k,2,h}$ requires space $\Omega(h \log k)$

Some interesting internal functions known to not be hard enough!

[BoC'92] (rephrased): if every node of the $\mathsf{TreeEval}_{k,2,h}$ *instance computes either* $+$ *or* $\times$*, then we can solve this instance with space* $2h + 3\log k = O(h + \log k) = O(\log n)$*.*

Reusing space

Conjecture [CMWBS'12]: $\mathrm{TreeEval}_{k,2,h}$ requires space $\Omega(h \log k)$

Some interesting internal functions known to not be hard enough!

[BoC'92] (rephrased): if every node of the $\mathrm{TreeEval}_{k,2,h}$ *instance computes either* $+$ *or* $\times$*, then we can solve this instance with space* $2h + 3\log k = O(h + \log k) = O(\log n)$*.*

Proof explicitly refutes z-f conjecture for $f \in \{+, \times\}$:

$$space(z, f) = |z| \ll |z| + space(f)$$

Main contribution: upper bounds on TreeEval

[CM'20,21]: for any k, h, TreeEval$_{k,2,h}$ can be solved in space $O(h \log k / \log h) = o(h \log k)$

Main contribution: upper bounds on TreeEval

[CM'20,21]: for any k, h, TreeEval$_{k,2,h}$ can be solved in space $O(h \log k / \log h) = o(h \log k)$

Tools: generalized [BoC'92] subroutine for arbitrary polynomials (instead of just $+$ or $\times$); complexity based on *degree*

Main contribution: upper bounds on TreeEval

[CM'20,21]: for any k, h, TreeEval$_{k,2,h}$ can be solved in space $O(h \log k / \log h) = o(h \log k)$

Tools: generalized [BoC'92] subroutine for arbitrary polynomials (instead of just $+$ or $\times$); complexity based on *degree*

$+$

recasting each internal TreeEval node as a polynomial

Main contribution: upper bounds on TreeEval

[CM'20,21]: for any k, h, TreeEval$_{k,2,h}$ can be solved in space $O(h \log k / \log h) = o(h \log k)$

Tools: generalized [BoC'92] subroutine for arbitrary polynomials (instead of just $+$ or $\times$); complexity based on *degree*

$+$

recasting each internal TreeEval node as a polynomial

$+$

time-space tradeoff to reduce the degree of these polynomials

Application: upper bounds on amortized non-uniform space

[P'17] (informally): every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have 2^{2^n-1} copies.

Application: upper bounds on amortized non-uniform space

[P'17] (informally): every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have 2^{2^n-1} copies.

[CM'22]: for any constant $\epsilon > 0$, every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have $2^{2^{\epsilon n}}$ copies.

Application: upper bounds on amortized non-uniform space

[P'17] (informally): every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have 2^{2^n-1} copies.

[CM'22]: for any constant $\epsilon > 0$, every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have $2^{2^{\epsilon n}}$ copies.

Tools: key polynomial subroutine we developed for [CM'20,21] as a one-shot algorithm applied to f

Application: upper bounds on amortized non-uniform space

[P'17] (informally): every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have 2^{2^n-1} copies.

[CM'22]: for any constant $\epsilon > 0$, every function f can be solved in time $O(n)$ and amortized non-uniform space $O(1)$, as long as we have $2^{2^{\epsilon n}}$ copies.

Tools: key polynomial subroutine we developed for [CM'20,21] as a one-shot algorithm applied to f

$$+$$

different sort of space-time tradeoff to reduce degree

Part I

Depth

Communication complexity

Another model of computation: communication complexity

- Alice receives $x \in \mathcal{X}$, Bob receives $y \in \mathcal{Y}$
- goal is to compute $F(x, y)$ together
- allowed to do any amount of computation on their own, charged for every bit exchanged

Communication complexity

Another model of computation: communication complexity

- Alice receives $x \in \mathcal{X}$, Bob receives $y \in \mathcal{Y}$
- goal is to compute $F(x, y)$ together
- allowed to do any amount of computation on their own, charged for every bit exchanged

[KW'90]: depth$(f) = cc(S_f)$ *for some related problem* S_f.

Communication complexity

Another model of computation: communication complexity

- Alice receives $x \in \mathcal{X}$, Bob receives $y \in \mathcal{Y}$
- goal is to compute $F(x, y)$ together
- allowed to do any amount of computation on their own, charged for every bit exchanged

[KW'90]: $depth(f) = cc(S_f)$ for some related problem S_f.

We *do* have composition-style results for communication!

Query-to-Communication Lifting

[RM'99,GPW'15]: for any F and for $g = IND_m$ (m suff. large),

$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Query-to-Communication Lifting

[RM'99,GPW'15]: for any F and for $g = IND_m$ (m suff. large),
$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Issues for KRW conjecture:

Query-to-Communication Lifting

[RM'99,GPW'15]: for any F and for $g = IND_m$ (m suff. large),
$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Issues for KRW conjecture:

- when moving to formula depth, only gives composition results for *monotone* formulas

Query-to-Communication Lifting

[RM'99,GPW'15]: for any F and for $g = IND_m$ *(m suff. large),*

$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Issues for KRW conjecture:

- when moving to formula depth, only gives composition results for *monotone* formulas
  - strong monotone lower bounds for sufficiently broad class of problems would also give general lower bounds

Query-to-Communication Lifting

[RM'99,GPW'15]: for any F and for $g = IND_m$ (m suff. large),
$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Issues for KRW conjecture:

- when moving to formula depth, only gives composition results for *monotone* formulas
  - strong monotone lower bounds for sufficiently broad class of problems would also give general lower bounds
- only works for $g = IND_m$, where m is a large polynomial

Query-to-Communication Lifting

[RM'99,GPW'15]: for any F and for $g = IND_m$ (m suff. large),
$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Issues for KRW conjecture:

- when moving to formula depth, only gives composition results for *monotone* formulas
  - strong monotone lower bounds for sufficiently broad class of problems would also give general lower bounds
- only works for $g = IND_m$, where m is a large polynomial

Main goal: more general g, starting with smaller m

Main contribution: better lifting

[LMMPZ'22] for any F and for $g = IND_m$ ($m \geq n^{1+\epsilon}$),

$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Main contribution: better lifting

[LMMPZ'22] for any F and for $g = IND_m$ ($m \geq n^{1+\epsilon}$),
$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Tools: "structure vs randomness" framework (earlier proofs)

Main contribution: better lifting

[LMMPZ'22] for any F and for $g = IND_m$ ($m \geq n^{1+\epsilon}$),
$$cc(F \circ g) \approx dt(F) \cdot \log m$$

Tools: "structure vs randomness" framework (earlier proofs)

$$+$$

"structure vs randomness" combinatorics to directly handle the random case (the bottleneck for previous proofs)

Application: proof complexity

[AM'20]: given $\tau \in \mathit{UNSAT}$*, it is* NP*-hard to approximate the size of the best Resolution or tree-like Resolution refutation of* τ*.*

Application: proof complexity

[AM'20]: given $\tau \in UNSAT$, it is NP*-hard to approximate the size of the best Resolution or tree-like Resolution refutation of τ.*

[GKMP'20]: given $\tau \in UNSAT$, it is NP-hard to approximate the size of the best Cutting Planes or tree-like Cutting Planes refutation of τ.

Application: proof complexity

[AM'20]: given $\tau \in UNSAT$, it is NP*-hard to approximate the size of the best Resolution or tree-like Resolution refutation of τ.*

[GKMP'20]: given $\tau \in UNSAT$, it is NP-hard to approximate the size of the best Cutting Planes or tree-like Cutting Planes refutation of τ.

Tools: non-approximability for Resolution or tree-like Resolution

Application: proof complexity

[AM'20]: given $\tau \in UNSAT$, it is NP*-hard to approximate the size of the best Resolution or tree-like Resolution refutation of τ.*

[GKMP'20]: given $\tau \in UNSAT$, it is NP-hard to approximate the size of the best Cutting Planes or tree-like Cutting Planes refutation of τ.

Tools: non-approximability for Resolution or tree-like Resolution

$$+$$

block-width and *graduated* lifting theorems using [LMMPZ'22]

Conclusions

Open problems

Open problems

1. *Directly improving our results* (better lifting parameters, better TreeEval algorithms)

Open problems

1. *Directly improving our results* (better lifting parameters, better TreeEval algorithms)
2. *Broadening and applying our results* (from lifting to KRW, better catalytic computing results)

Open problems

1. *Directly improving our results* (better lifting parameters, better TreeEval algorithms)
2. *Broadening and applying our results* (from lifting to KRW, better catalytic computing results)
3. *Structure of our results* (lifting and combinatorics, which classes catalytic techniques inherently lie in)

Works used

1. Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, Jiapeng Zhang. *Lifting with Sunflowers*. ITCS 2022.
2. Mika Göös, Sajin Koroth, Ian Mertz, Toniann Pitassi. *Automating Cutting Planes is NP-Hard*. STOC 2020.
3. Ian Mertz, Toniann Pitassi, Yuanhao Wei. *Short Proofs Are Hard to Find*. ICALP 2019.
4. James Cook, Ian Mertz. *Catalytic Approaches to the Tree Evaluation Problem*. STOC 2020.
5. James Cook, Ian Mertz. *Encodings and the Tree Evaluation Problem*. Technical note, 2021.
6. James Cook, Ian Mertz. *Trading Time and Space in Catalytic Branching Programs*. CCC 2022.

Thanks!