On the Complexity of Computing Strahler Numbers

Moses Ganardi, Markus Lohrey

RPTU Kaiserslautern-Landau, Universität Siegen

STACS 2026

Strahler numbers

Strahler numbers

For a binary tree t, the Strahler number (aka Horton-Strahler number) $\mathrm{st}(t)$ is defined as follows:

$$\mathrm{st}(\bullet) = 0$$

$$\mathrm{st}(t_1 \overset{\bullet}{\wedge} t_2) = \begin{cases} \max\{\mathrm{st}(t_1), \mathrm{st}(t_2)\} & \text{if } \mathrm{st}(t_1) \neq \mathrm{st}(t_2) \\ \mathrm{st}(t_1) + 1 & \text{if } \mathrm{st}(t_1) = \mathrm{st}(t_2) \end{cases}$$

Strahler numbers

For a binary tree t, the Strahler number (aka Horton-Strahler number) $\mathrm{st}(t)$ is defined as follows:

$$\mathrm{st}(\bullet) = 0$$

$$\mathrm{st}(t_1 \overset{\bullet}{\wedge} t_2) = \begin{cases} \max\{\mathrm{st}(t_1), \mathrm{st}(t_2)\} & \text{if } \mathrm{st}(t_1) \neq \mathrm{st}(t_2) \\ \mathrm{st}(t_1) + 1 & \text{if } \mathrm{st}(t_1) = \mathrm{st}(t_2) \end{cases}$$

Alternative characterization: $\mathrm{st}(t)$ is the height of the largest perfect binary tree that can be embedded into t.

Strahler numbers

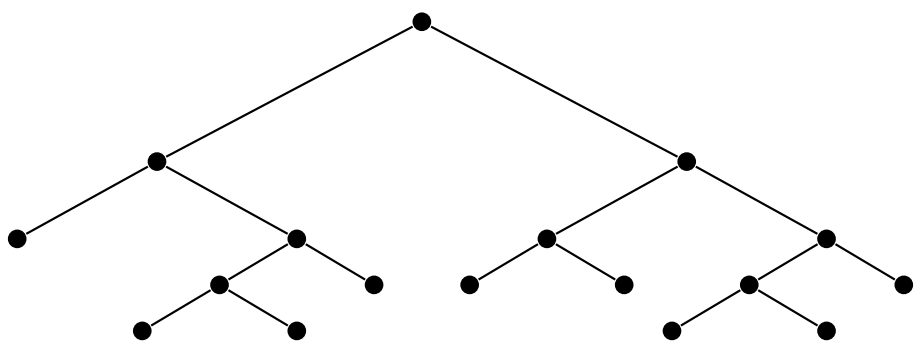
For a binary tree t, the Strahler number (aka Horton-Strahler number) $\mathrm{st}(t)$ is defined as follows:

$$\mathrm{st}(\bullet) = 0$$

$$\mathrm{st}(t_1 \overset{\bullet}{\wedge} t_2) = \begin{cases} \max\{\mathrm{st}(t_1), \mathrm{st}(t_2)\} & \text{if } \mathrm{st}(t_1) \neq \mathrm{st}(t_2) \\ \mathrm{st}(t_1) + 1 & \text{if } \mathrm{st}(t_1) = \mathrm{st}(t_2) \end{cases}$$

Alternative characterization: $\mathrm{st}(t)$ is the height of the largest perfect binary tree that can be embedded into t.

Example:



Strahler numbers

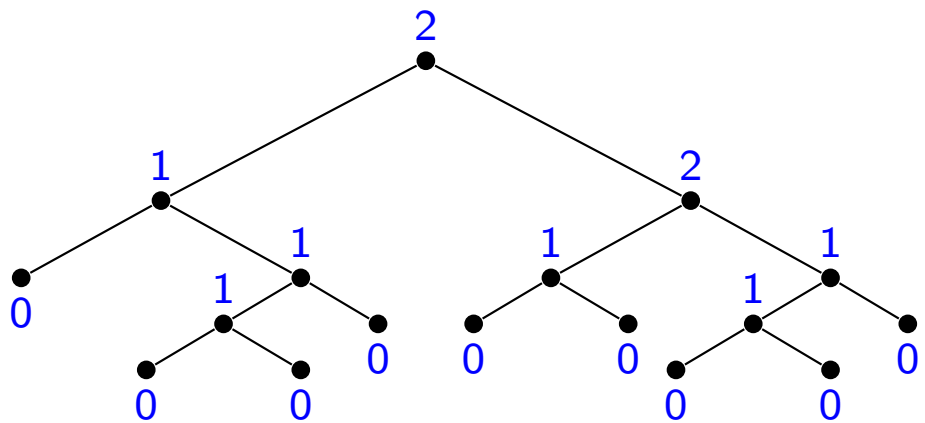
For a binary tree t, the Strahler number (aka Horton-Strahler number) $\mathrm{st}(t)$ is defined as follows:

$$\mathrm{st}(\bullet) = 0$$

$$\mathrm{st}(t_1 \overset{\bullet}{\wedge} t_2) = \begin{cases} \max\{\mathrm{st}(t_1), \mathrm{st}(t_2)\} & \text{if } \mathrm{st}(t_1) \neq \mathrm{st}(t_2) \\ \mathrm{st}(t_1) + 1 & \text{if } \mathrm{st}(t_1) = \mathrm{st}(t_2) \end{cases}$$

Alternative characterization: $\mathrm{st}(t)$ is the height of the largest perfect binary tree that can be embedded into t.

Example:



Origin of the Strahler number

Invented in hydrology by Horton (1945) and Strahler (1952).

Origin of the Strahler number

Invented in hydrology by Horton (1945) and Strahler (1952).

Ershov 1958: For an arithmetic expression t, $\mathrm{st}(t)$ is the minimal number of registers needed to evaluate t.

Origin of the Strahler number

Invented in hydrology by Horton (1945) and Strahler (1952).

Ershov 1958: For an arithmetic expression t, $\mathrm{st}(t)$ is the minimal number of registers needed to evaluate t.

For more background, see the surveys by Viennot (1990) and Esparza, Luttenberger and Schlund (2014).

Applications of Strahler numbers

- Context-free grammars with finite index

  (Ginsburg, Spanier 1968)

- Parikh's theorem

  (Esparza, Ganty, Kiefer, Luttenberger 2011)

- Pushdown VASS

  (Atig, Ganty 2011; Bizière, Czerwiński 2025)

- Parity games

  (Daviaud, Jurdziński, Thejaswini 2020)

Computing Strahler numbers

How difficult is it to compute the Strahler number?

How difficult is it to compute the Strahler number?

Easy to compute in linear time on a RAM.

How difficult is it to compute the Strahler number?

Easy to compute in linear time on a RAM.

Easy to compute in space $\mathcal{O}(\log n \log\log n)$.

How difficult is it to compute the Strahler number?

Easy to compute in linear time on a RAM.

Easy to compute in space $\mathcal{O}(\log n \log\log n)$.

Observation

If the binary tree t has n leaves then $\mathrm{st}(t) \leq \log_2 n$. Hence, $\mathrm{st}(t)$ can be stored with $\mathcal{O}(\log\log n)$ bits.

How difficult is it to compute the Strahler number?

Easy to compute in linear time on a RAM.

Easy to compute in space $\mathcal{O}(\log n \log\log n)$.

Observation

If the binary tree t has n leaves then $\mathrm{st}(t) \leq \log_2 n$. Hence, $\mathrm{st}(t)$ can be stored with $\mathcal{O}(\log\log n)$ bits.

Recursive algorithm where the larger subtree is visited first.
$\rightarrow$ stack of height $\mathcal{O}(\log n)$ with $\log\log n$ bit numbers.

The stack can be stored with $\mathcal{O}(\log n)$ bits

Example: Assume that the current stack content is $s_1, s_2, \ldots, s_\ell$:

The stack can be stored with $\mathcal{O}(\log n)$ bits

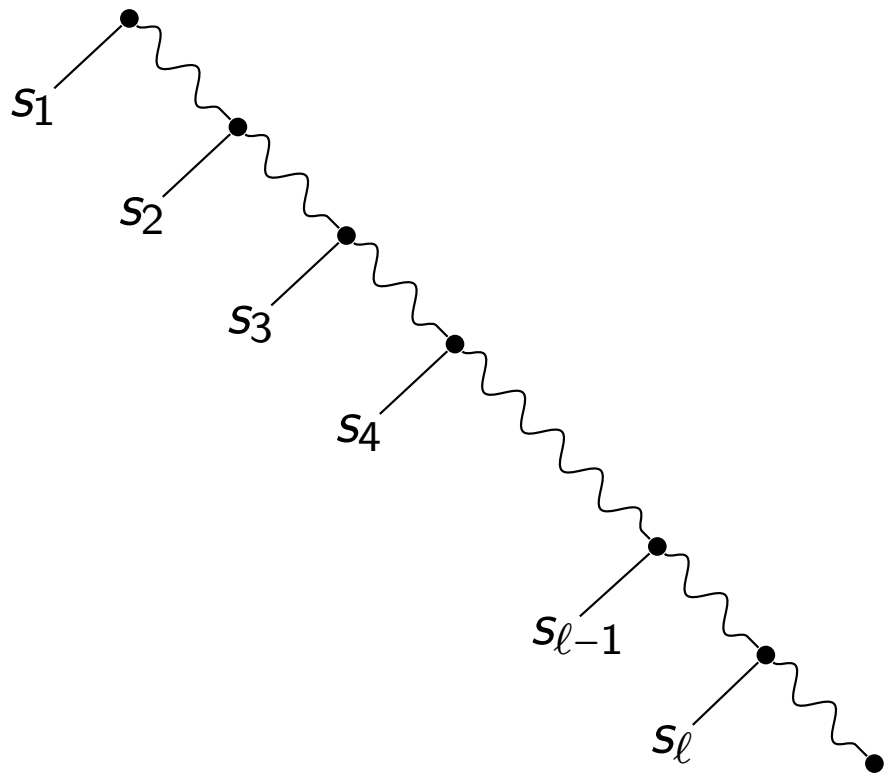
Example: Assume that the current stack content is $s_1, s_2, \ldots, s_\ell$:

If $i < j$ and $s_i < s_j$ then we can set $s_i := 0$ without changing the Strahler number of the tree.

The stack can be stored with $\mathcal{O}(\log n)$ bits

Example: Assume that the current stack content is $s_1, s_2, \ldots, s_\ell$:

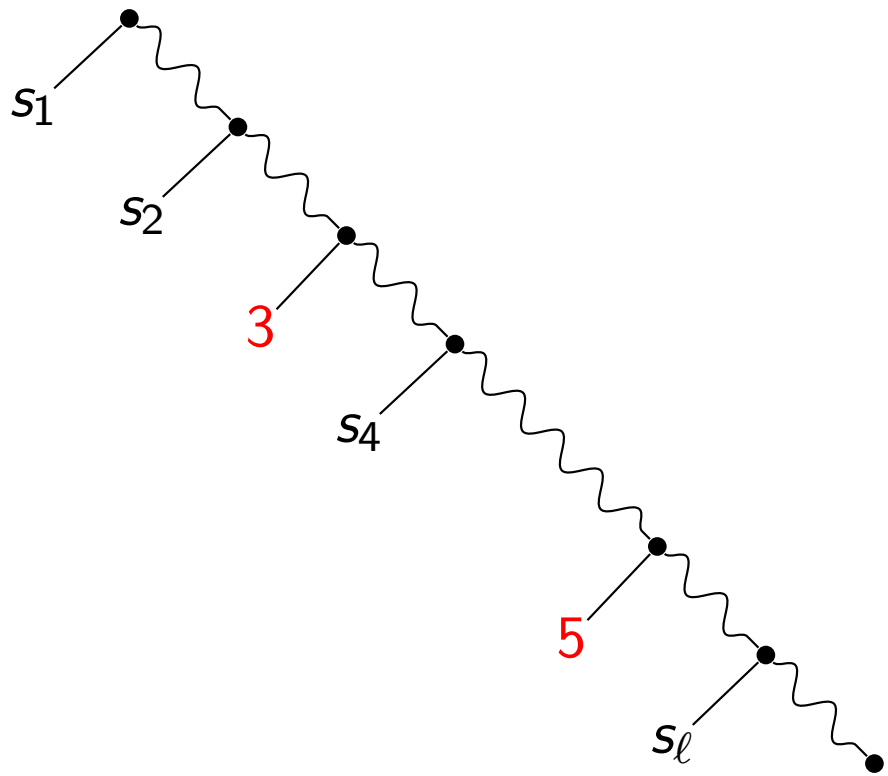
If $i < j$ and $s_i < s_j$ then we can set $s_i := 0$ without changing the Strahler number of the tree.



The stack can be stored with $\mathcal{O}(\log n)$ bits

Example: Assume that the current stack content is $s_1, s_2, \ldots, s_\ell$:

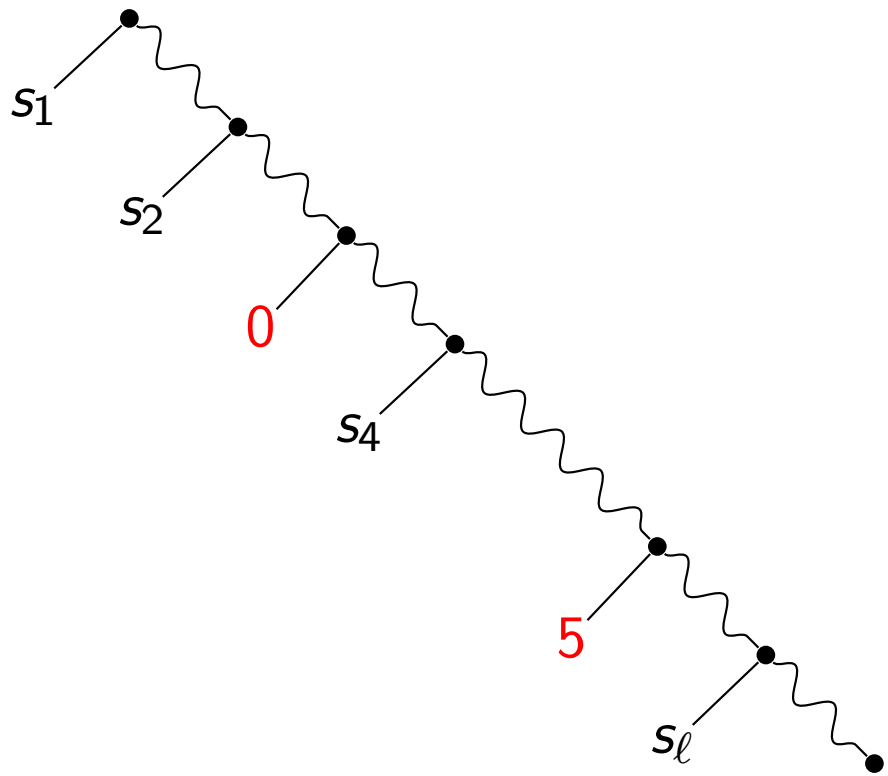
If $i < j$ and $s_i < s_j$ then we can set $s_i := 0$ without changing the Strahler number of the tree.



The stack can be stored with $\mathcal{O}(\log n)$ bits

Example: Assume that the current stack content is $s_1, s_2, \ldots, s_\ell$:

If $i < j$ and $s_i < s_j$ then we can set $s_i := 0$ without changing the Strahler number of the tree.



The stack can be stored with $\mathcal{O}(\log n)$ bits

Let the resulting sequence be

$$0, \ldots, 0, s_{i_1}, 0, \ldots, 0, s_{i_2}, \ldots, s_{i_{k-1}}, , 0, \ldots, 0, s_{i_k}, 0, \ldots, 0$$

The stack can be stored with $\mathcal{O}(\log n)$ bits

Let the resulting sequence be

$$0, \ldots, 0, s_{i_1}, 0, \ldots, 0, s_{i_2}, \ldots, s_{i_{k-1}}, , 0, \ldots, 0, s_{i_k}, 0, \ldots, 0$$

We have $s_{i_1} \geq s_{i_2} \geq \cdots \geq s_{i_{k-1}} \geq s_{i_k}$.

The stack can be stored with $\mathcal{O}(\log n)$ bits

Let the resulting sequence be

$$0, \ldots, 0, s_{i_1}, 0, \ldots, 0, s_{i_2}, \ldots, s_{i_{k-1}}, , 0, \ldots, 0, s_{i_k}, 0, \ldots, 0$$

We have $s_{i_1} \geq s_{i_2} \geq \cdots \geq s_{i_{k-1}} \geq s_{i_k}$.

We now use a delta encoding:

$$0 \cdots 0\, 1^{s_{i_1} - s_{i_2}} \# 0 \cdots 0\, 1^{s_{i_2} - s_{i_3}} \# \cdots 1^{s_{i_{k-1}} - s_{i_k}} \# 0 \cdots 0\, 1^{s_{i_k}} \# 0 \cdots 0$$

The stack can be stored with $\mathcal{O}(\log n)$ bits

Let the resulting sequence be

$$0, \ldots, 0, s_{i_1}, 0, \ldots, 0, s_{i_2}, \ldots, s_{i_{k-1}}, , 0, \ldots, 0, s_{i_k}, 0, \ldots, 0$$

We have $s_{i_1} \geq s_{i_2} \geq \cdots \geq s_{i_{k-1}} \geq s_{i_k}$.

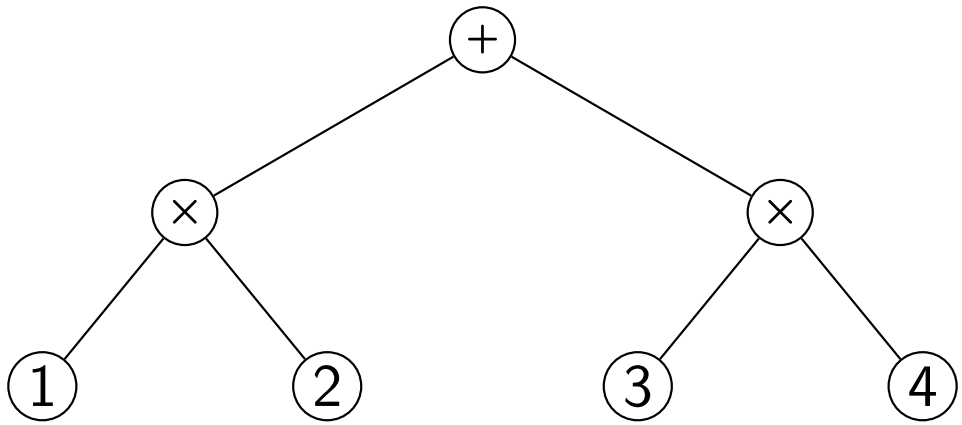
We now use a delta encoding:

$$0 \cdots 0\, 1^{s_{i_1} - s_{i_2}} \# 0 \cdots 0\, 1^{s_{i_2} - s_{i_3}} \# \cdots 1^{s_{i_{k-1}} - s_{i_k}} \# 0 \cdots 0\, 1^{s_{i_k}} \# 0 \cdots 0$$

This is a word of length $\ell + s_{i_1} \leq \mathcal{O}(\log n)$. □

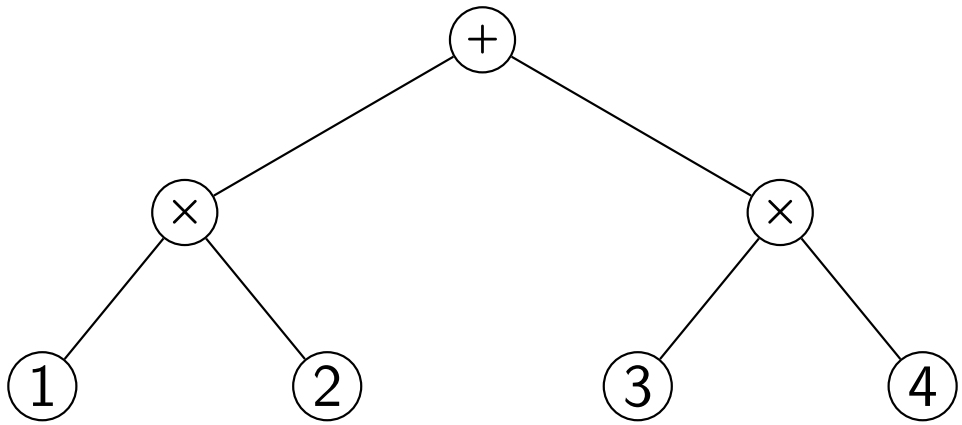
Broader context: Tree evaluation

Broader context: Tree evaluation



- Each leaf is labeled by an element $a \in A$.
- Each inner node carries a function $f\colon A^r \to A$, where r is the number of children.

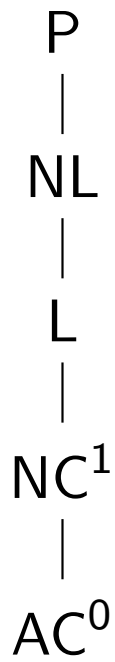
Broader context: Tree evaluation



- Each leaf is labeled by an element $a \in A$.
- Each inner node carries a function $f\colon A^r \to A$, where r is the number of children.

Goal: Understand the complexity of tree evaluation for various algebras.

Small complexity classes

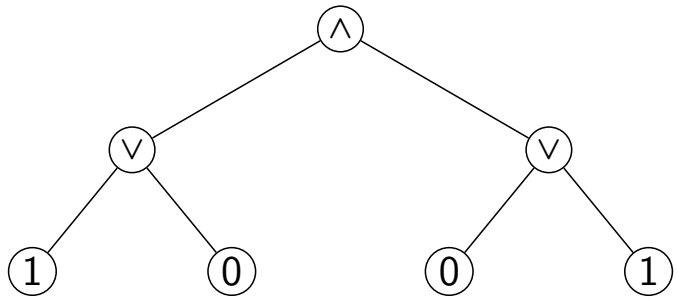


NC^1 = class of all problems that can be solved by a uniform family $(C_n)_{n\geq 0}$ of fan-in-2 boolean circuits such that C_n (= circuit for input length n) has

- size $\mathrm{poly}(n)$ and
- depth $\mathcal{O}(\log n)$.

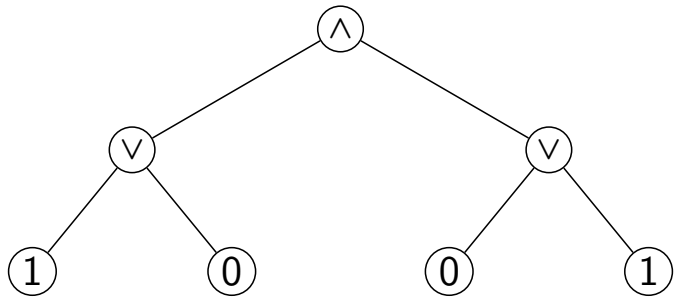
Broader context: Tree evaluation

Boolean formulas

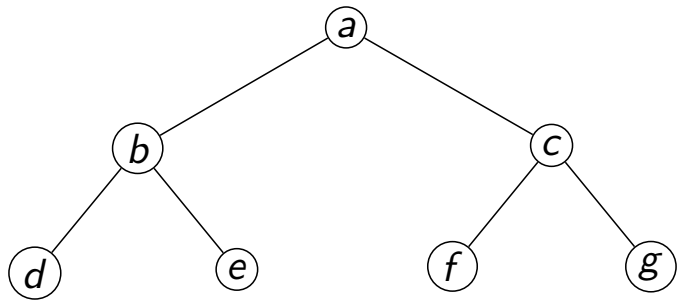


Broader context: Tree evaluation

Boolean formulas

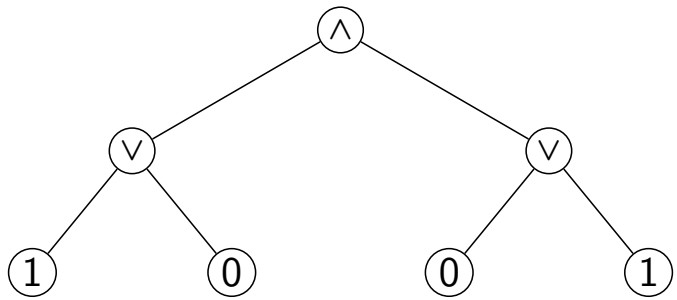


Acceptance of a (fixed) tree automaton

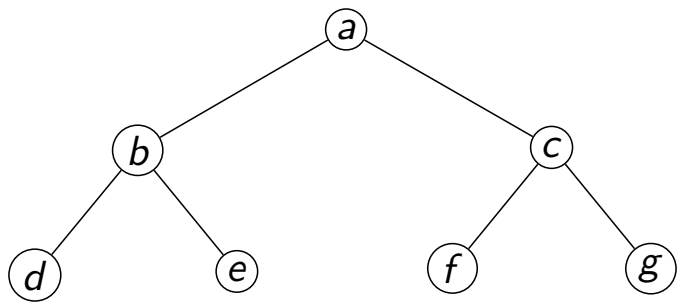


Broader context: Tree evaluation

Boolean formulas



Acceptance of a (fixed) tree automaton



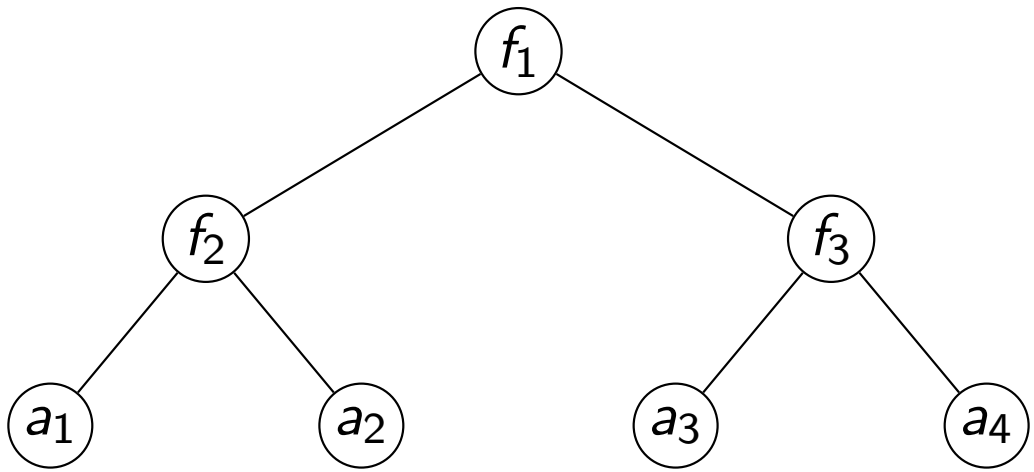
→ NC^1-complete assuming parenthesis representations $(1 \vee 0) \wedge (0 \vee 1)$.

(Buss 1987, Lohrey 2001)

Broader context: Tree evaluation

"The tree evaluation" problem

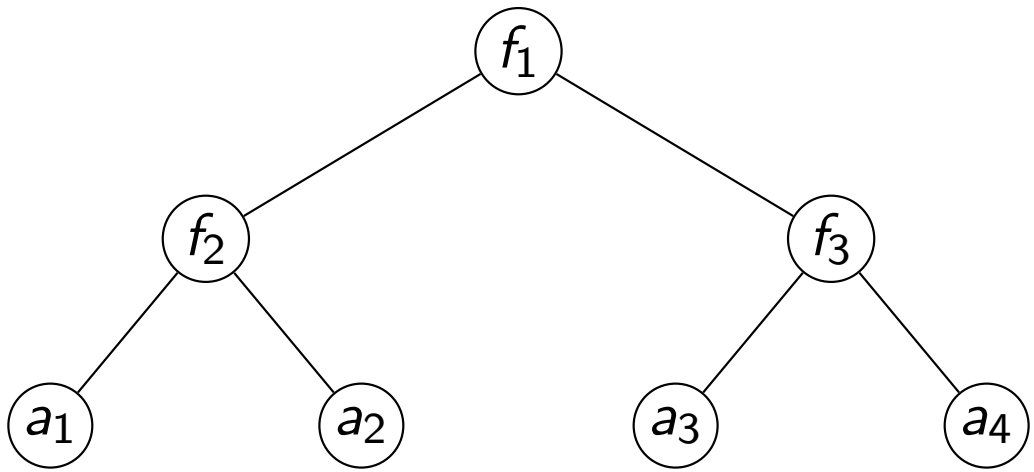
Evaluate a complete binary tree of height h over a domain of size k



Broader context: Tree evaluation

"The tree evaluation" problem

Evaluate a complete binary tree of height h over a domain of size k

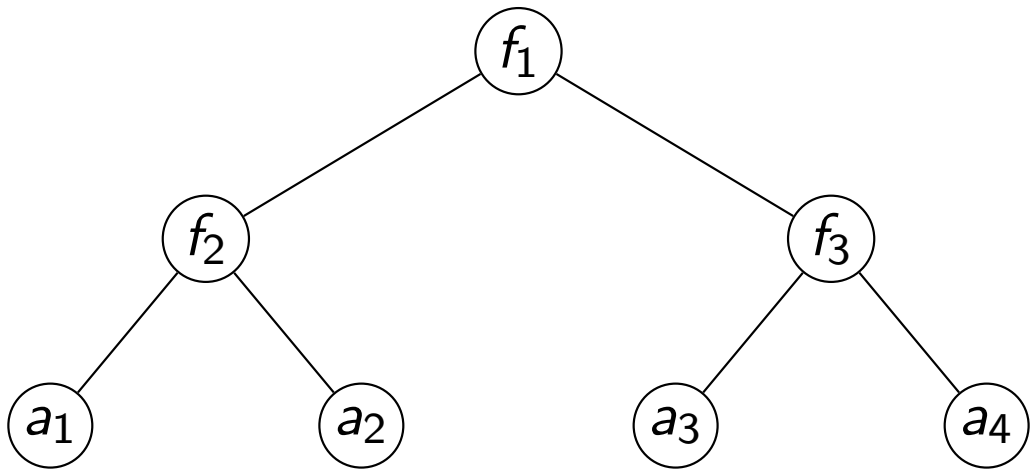


Straightforward: $\mathcal{O}(h \log k)$ space

Broader context: Tree evaluation

"The tree evaluation" problem

Evaluate a complete binary tree of height h over a domain of size k



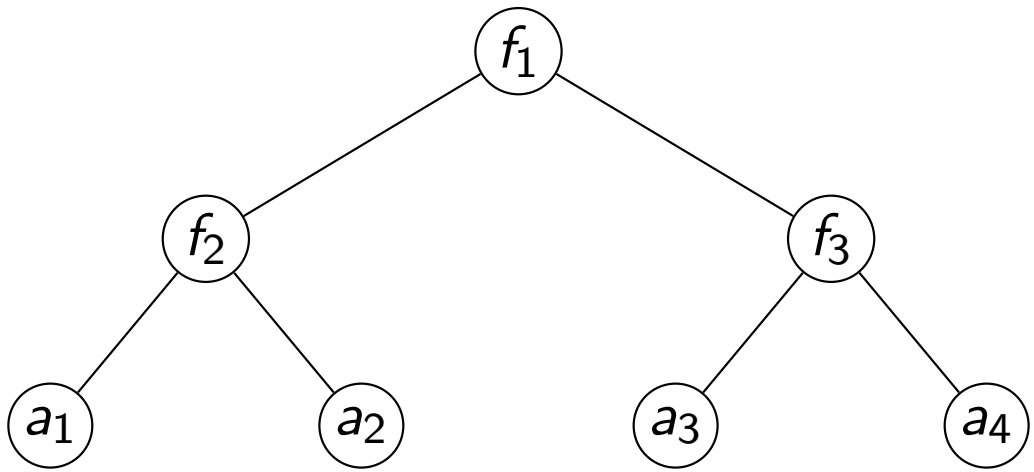
Straightforward: $\mathcal{O}(h \log k)$ space

Cook-Mertz (2024): $\mathcal{O}(h \log \log k + \log k)$ space

Broader context: Tree evaluation

"The tree evaluation" problem

Evaluate a complete binary tree of height h over a domain of size k



Straightforward:	$\mathcal{O}(h \log k)$ space
Cook-Mertz (2024):	$\mathcal{O}(h \log\log k + \log k)$ space
Williams (2025):	$\mathrm{TIME}[t] \subseteq \mathrm{SPACE}[\sqrt{t \log t}]$

Our contributions

The complexity of computing the Strahler number

Theorem

Computing the Strahler number of a binary tree t is NC^1-complete if the tree t is given by a parenthesis expression.

The complexity of computing the Strahler number

Theorem

Computing the Strahler number of a binary tree t is NC^1-complete if the tree t is given by a parenthesis expression.

NC^1 = Fan-in-2 boolean circuits of poly-size and log-depth

The complexity of computing the Strahler number

Theorem

Computing the Strahler number of a binary tree t is NC^1-complete if the tree t is given by a parenthesis expression.

NC^1 = Fan-in-2 boolean circuits of poly-size and log-depth

Further results on Strahler number computations on context-free grammars.

Proof idea for the NC^1 upper bound

1. G, Lohrey 2018: Compute in NC^1 an $\mathcal{O}(\log |t|)$-depth recursive decomposition of t into subtrees and contexts (subtrees in which a smaller subtree is replaced by placeholder x).

Proof idea for the NC^1 upper bound

1. G, Lohrey 2018: Compute in NC^1 an $\mathcal{O}(\log |t|)$-depth recursive decomposition of t into subtrees and contexts (subtrees in which a smaller subtree is replaced by placeholder x).

$$\begin{aligned} \text{subtree } s &\longrightarrow n_s = \mathsf{st}(s) \in \mathbb{N} \\ \text{context } c(x) &\longrightarrow f_c : \mathbb{N} \to \mathbb{N} \end{aligned}$$

Proof idea for the NC^1 upper bound

1. G, Lohrey 2018: Compute in NC^1 an $\mathcal{O}(\log|t|)$-depth recursive decomposition of t into subtrees and contexts (subtrees in which a smaller subtree is replaced by placeholder x).

$$\begin{aligned} \text{subtree } s &\longrightarrow n_s = \mathrm{st}(s) \in \mathbb{N} \\ \text{context } c(x) &\longrightarrow f_c : \mathbb{N} \to \mathbb{N} \end{aligned}$$

This function f_c can be described by two numbers $\ell_c \leq h_c$:

$$f_c(n) = \begin{cases} h_c & \text{if } n < \ell_c \\ h_c + 1 & \text{if } \ell_c \leq n \leq h_c \\ n & \text{if } n > h_c \end{cases}.$$

Proof idea for the NC^1 upper bound

1. G, Lohrey 2018: Compute in NC^1 an $\mathcal{O}(\log|t|)$-depth recursive decomposition of t into subtrees and contexts (subtrees in which a smaller subtree is replaced by placeholder x).

$$\begin{aligned} \text{subtree } s &\longrightarrow n_s = \mathrm{st}(s) \in \mathbb{N} \\ \text{context } c(x) &\longrightarrow f_c : \mathbb{N} \to \mathbb{N} \end{aligned}$$

   This function f_c can be described by two numbers $\ell_c \leq h_c$:

$$f_c(n) = \begin{cases} h_c & \text{if } n < \ell_c \\ h_c + 1 & \text{if } \ell_c \leq n \leq h_c \\ n & \text{if } n > h_c \end{cases}.$$

2. Compute a $\mathcal{O}(\log|t|)$-depth fan-in-2 boolean circuit with gates of the form $[X \leq Y + o\ ?]$, where X and Y are among the n_s, ℓ_c, h_c and $o \in \mathbb{Z}$, $o \in \mathcal{O}(\log|t|)$.

Proof idea for the NC^1 upper bound

1. G, Lohrey 2018: Compute in NC^1 an $\mathcal{O}(\log|t|)$-depth recursive decomposition of t into subtrees and contexts (subtrees in which a smaller subtree is replaced by placeholder x).

$$\begin{aligned} \text{subtree } s &\longrightarrow n_s = \mathrm{st}(s) \in \mathbb{N} \\ \text{context } c(x) &\longrightarrow f_c : \mathbb{N} \to \mathbb{N} \end{aligned}$$

   This function f_c can be described by two numbers $\ell_c \leq h_c$:

$$f_c(n) = \begin{cases} h_c & \text{if } n < \ell_c \\ h_c + 1 & \text{if } \ell_c \leq n \leq h_c \\ n & \text{if } n > h_c \end{cases}.$$

2. Compute a $\mathcal{O}(\log|t|)$-depth fan-in-2 boolean circuit with gates of the form $[X \leq Y + o\ ?]$, where X and Y are among the n_s, ℓ_c, h_c and $o \in \mathbb{Z}$, $o \in \mathcal{O}(\log|t|)$.
3. Evaluate this circuit in NC^1.

Summary

Theorem

Computing the Strahler number of a binary tree t is NC^1-complete if the tree t is given by a parenthesis expression.

Open questions:

- Evaluating $\{\max, +\}$-trees with binary encoded numbers
- Shortest path problem in directed graphs with binary encoded weights
- Strahler numbers for unranked trees
- Pathwidth of trees

Strahler numbers and context-free grammars

Strahler numbers and context-free grammars

Gruska 1971

The following problem is undecidable:

- Input: A context-free grammar G in Chomsky normal form and $k \in \mathbb{N}$
- Question: Does every word in $L(G)$ have a derivation tree of Strahler number $\leq k$?

Strahler numbers and context-free grammars

Gruska 1971

The following problem is undecidable:

- Input: A context-free grammar G in Chomsky normal form and $k \in \mathbb{N}$
- Question: Does every word in $L(G)$ have a derivation tree of Strahler number $\leq k$?

G, Lohrey 2025

The following problem is P-complete:

- Input: A context-free grammar G in Chomsky normal form and $k \in \mathbb{N}$
- Question: Do all derivation trees of G have Strahler number $\leq k$?

Strahler numbers and context-free grammars

A derivation tree T is acyclic if no nonterminal appears more the once on a path from the root to a leaf.

Strahler numbers and context-free grammars

A derivation tree T is acyclic if no nonterminal appears more the once on a path from the root to a leaf.

G, Lohrey 2025

The following problem is PSPACE-complete:

- Input: A context-free grammar G in Chomsky normal form and $k \in \mathbb{N}$
- Question: Do all acyclic derivation trees of G have Strahler number at most k?

Strahler numbers and context-free grammars

A derivation tree T is acyclic if no nonterminal appears more the once on a path from the root to a leaf.

G, Lohrey 2025

The following problem is PSPACE-complete:

- Input: A context-free grammar G in Chomsky normal form and $k \in \mathbb{N}$
- Question: Do all acyclic derivation trees of G have Strahler number at most k?

Why are acyclic derivation trees interesting?

Strahler numbers and context-free grammars

Swernofsky, Wehar 2015

Intersection non-emptiness for DFAs and a single CFG is EXPTIME-complete.

Strahler numbers and context-free grammars

Swernofsky, Wehar 2015

Intersection non-emptiness for DFAs and a single CFG is EXPTIME-complete.

Lohrey, Rosowski, Zetzsche 2022

Intersection non-emptiness for group DFAs and a single CFG is PSPACE-complete.

Strahler numbers and context-free grammars

Swernofsky, Wehar 2015

Intersection non-emptiness for DFAs and a single CFG is EXPTIME-complete.

Lohrey, Rosowski, Zetzsche 2022

Intersection non-emptiness for group DFAs and a single CFG is PSPACE-complete.

Lohrey, Rosowski, Zetzsche 2022

Fix a constant $k \geq 1$. Intersection non-emptiness for group DFAs and a single CFG for which all acyclic derivation trees have Strahler number $\leq k$ is NP-complete.