A Tight Meta-theorem for LOCAL Certification of MSO2 Properties within Bounded Treewidth

Eunjung Kim

SoC KAIST and DIMAG IBS, Daejeon, South Korea & CNRS, France

Jointwork with Linda Cook and Tomáš Masařík

LOGALG 2025, 19-21 November 2025, Vienna, Austria

Proof Labeling Scheme (PLS)

A *certificate* on a graph G is a mapping φ_G from $V(G)$ to $\{0,1\}^*$. The size of a certificate on G is $\max_{v \in V(G)} |\varphi_G(v)|$.

Proof Labeling Scheme

A pair (P, V) is said to be a *proof labeling scheme* for a graph property $\mathcal{P}$ if

- P is an algorithm which outputs a certificate φ_G for each graph G, and
- V is an algorithm whose outputs Yes or No on an input from $\{0,1\}^*$

for which the following holds:

- if $G \in \mathcal{P}$, then $\mathsf{V}(\varphi_G(v), \bigcup_{w \in N_G(v)} \varphi_G(w)) = \text{Yes}$ for every $v \in V(G)$, and
- if $G \notin \mathcal{P}$, then $\mathsf{V}(\varphi_G(v), \bigcup_{w \in N_G(v)} \varphi_G(w)) = \text{No}$ for some $v \in V(G)$.

Proof Labeling Scheme (PLS)

Intuitively, the prover P computes a certificate φ_G and assigns to each vertex $v \in V(G)$ the *local* certificate $\varphi_G(V)$.

After a one round of *synchronized* communication between neighboring vertices, the verifier V at v outputs Yes or No based on its computation on the instance consisting of $\varphi_G(v)$ and $(\varphi_G(w))_{w \in N_G(v)}$.

Each vertex is equipped with a unique *identifier* as a $\{0, 1\}$-string of length *poly*(n), where n is the number of vertices.

It is assumed that the underlying graph G is connected.

PLS: SIMPLE EXAMPLES

PLS for { all bipartite graphs }.

- P outputs a certificate $\varphi_G : V(G) \to \{red, blue\}$; φ_G is a proper 2-coloring of G if one exists, a random mapping otherwise.
- V at a vertex v outputs YES if $\varphi_G(w)$ differs from $\varphi_G(v)$ for every $w \in N_G(v)$; NO otherwise.

PLS for { all acyclic graphs }.

- If G is a tree, P chooses an arbitrary vertex r as the root and computes φ_G such that for each vertex v, $\varphi_G(v) = \text{dist}(v, r)$.
- V at v outputs YES if either $\text{dist}(v, r) = 0$ or the following holds:
    1. there is a unique neighbor w such that $\text{dist}(w, r) \leq \text{dist}(v, r)$ and it holds that $\text{dist}(w, r) = \text{dist}(v, r) + 1$, and
    2. for every other neighbor z, it holds that $\text{dist}(z, r) = \text{dist}(v, r) + 1$.

Known Results

Property	upper bound	ref
H-minor-free for small H	$O(\log n)$	BFP'21
planarity	$O(\log n)$	FFMRRT'21
bounded genus	$O(\log n)$	EL'22
treedepth at most k	$O(\log n)$	FBP'22
treewidth at most k	$O(\log^2 n)$	FMRT'22
cographs	$O(\log n)$	FMMRT'23
cliquewidth at most k	$O(\log^2 n)$	FMMRT'23

Our Main Result

Theorem (Cook, K. Masařík 2025)

There is an approximate PLS for graphs of treewidth at most k of size $O(\log n)$; i.e. there exists a computable function f for which P computes φ_G of size $O(\log n)$ such that

- *if* $\textsc{tw}(G) \leq k$*, then each V outputs* Yes*, and*
- *if* $\textsc{tw}(G) > f(k)$*, then some V outputs* No.

Theorem (Fraigniaud, Montealegre, Rapaport, Todinca 2022, Cook, K. Masařík 2025)

For each integer t and MSO_2*-sentence ϕ, there is a $O(\log n)$-size PLS for an* MSO_2*-definable property on graphs of bounded treewidth; with promise of* $\textsc{tw}(G) \leq t$*, the prover P computes φ_G such that*

- *if $G \models \phi$, then each V outputs* Yes*, and*
- *if $G \not\models \phi$, then some V outputs* No.

Witness for treewidth at most t

Elimination Tree and Width

A rooted tree F is an *elimination tree* of a (simple) graph G if $V(F) = V(G)$ and for every $uv \in E(G)$, u is a strict ancestor of v or vice versa.

The *width* of an elimination tree F of G is defined as

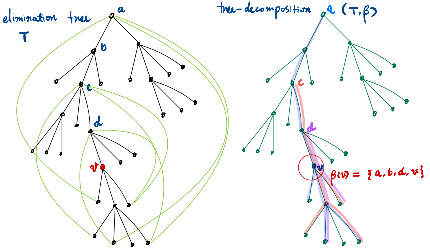
$$\textsc{width}(F) = \max_{v \in V(G)} \{w \in V(G) \mid w \text{ is an ancestor of } v \text{ and } w \text{ is adjacent with } F_v\} - 1$$

Elimination Tree and Width

A rooted tree F is an *elimination tree* of a (simple) graph G if $V(F) = V(G)$ and for every $uv \in E(G)$, u is a strict ancestor of v or vice versa.

The *width* of an elimination tree F of G is defined as

$$\textsc{width}(F) = \max_{v \in V(G)} \{w \in V(G) \mid w \text{ is an ancestor of } v \text{ and } w \text{ is adjacent with } F_v\} - 1$$



Witness for treewidth at most t

Elimination Tree and Width

A rooted tree F is an *elimination tree* of a (simple) graph G if $V(F) = V(G)$ and for every $uv \in E(G)$, u is a strict ancestor of v or vice versa.

The *width* of an elimination tree F of G is defined as

$$\textsc{width}(F) = \max_{v \in V(G)} \{w \in V(G) \mid w \text{ is an ancestor of } v \text{ and } w \text{ is adjacent with } F_v\} - 1$$

Arnborg 1985, Boadlaender et al. 1991, Bojańczyk and Pilipczuk 2016

A graph G has treewidth at most t if and only if G admits an elimination tree of width at most t.

Toy case and bottleneck for treewidth at most t

Toy case: suppose that G admits an elimination tree F of width at most t such that

every edge of F is an (actual) edge of G.

Certificate φ_G assigned by the prover

If G admits an elimination tree F of width at most t, the local certificate $\varphi_G(v)$ carries the following information for each $v \in V(G)$.

1. the distance $\text{dist}_F(v, root)$.
2. the list L_v of all strict ancestors of v with a neighbor in F_v, together with their distances to the root.

Toy case and bottleneck for treewidth at most t

Toy case: suppose that G admits an elimination tree F of width at most t such that

$$\text{every edge of } F \text{ is an (actual) edge of } G.$$

Verifier's algorithm at each $v \in V(G)$

Verifier V at each v checks the following.

1. Ensure that there is no neighbor u with $\mathrm{dist}_F(u, root) = \mathrm{dist}_F(v, root)$.
2. Ensure that there is a unique neighbor w with $\mathrm{dist}_F(w, root) = \mathrm{dist}_F(v, root) + 1$ (or $\mathrm{dist}_F(v, root) = 0$).
3. Whenever its neighbor z is an ancestor, ensure that z is the list L_v.
4. Ensure that $|L_v| \leq t$.
5. Ensure that $\bigcup_{w \text{ is a neighbor and strict descendant of } v} L_w \subseteq L_v \cup \{v\}$.

Toy case: suppose that G admits an elimination tree F of width at most t such that

every edge of F is an (actual) edge of G.

$\rightsquigarrow$ We cannot such an elimination tree in general.

Blanco, Cook, Hatzel, Hilaire, Illingworth, McCarty 2024

For every function f, there exists a graph G which does not admit a tree-decomposition (T, β) of width $f(\mathrm{tw}(G))$ such that T is a minor of G.

For an edge uv in the elimination treewith (v being the parent of u), we want to have a (u, v)-path in G so as to use it for “channeling” the oriented edge from u to v.

Toy case: suppose that G admits an elimination tree F of width at most t such that

every edge of F is an (actual) edge of G.

$\rightsquigarrow$ We cannot such an elimination tree in general.

Blanco, Cook, Hatzel, Hilaire, Illingworth, McCarty 2024

For every function f, there exists a graph G which does not admit a tree-decomposition (T, β) of width $f(\textsc{tw}(G))$ such that T is a minor of G.

For an edge uv in the elimination treewith (v being the parent of u), we want to have a (u, v)-path in G so as to use it for "channeling" the oriented edge from u to v.

Oriented path system witnessing an elimination tree

Let F be an elimination tree of G. A collection $\mathcal{P}$ of directed path system is said to witness F if, for every vertex u and its parent v in F, there is some (u, v)-path of G oriented from u toward v in $\mathcal{P}$

The *congestion of* $\mathcal{P}$ is

$$\max_{v \in V(G)} \# \text{ of oriented paths in } \mathcal{P} \text{ using } v \text{ as a start or internal vertex.}$$

Key technical lemma [Cook, K. Masařík 2024 + Bojańczyk, Pilipczuk 2015]

For any graph G of treewidth at most t, there exists

- an elimination tree F of G of width at most $f(t)$, and
- an oriented path system $\mathcal{P}$ of congestion $f(t)$ witnessing F.

Suppose that G admits an elimination tree F of width $f(t)$ and an oriented path system $\mathcal{P}$ of congestion at most $f(t)$ witnessing F.

Certificate φ_G assigned by the prover

$\varphi_G(v)$ carries the following information for each $v \in V(G)$.

1. the distance $\mathrm{dist}_F(v, root)$.
2. the parent u of v.
3. the list L_v of all strict ancestors of v with a neighbor in F_v, together with their distances to the root.
4. channel for (s,t)-path in $\mathcal{P}$: when v is on an oriented path $P \in \mathcal{P}$ as a start or internal vertex: the start s and final vertex t of P (together with their distances to the root), the predecessor and successor of v, the list L_s of the start vertex.

Verifier's algorithm at each $v \in V(G)$: channel works properly

Ensure the channel for (s,t)-path P in $\mathcal{P}$ is working properly

1. Ensure that the (s,t)-channel is proper: $\mathrm{dist}_F(s, root) = \mathrm{dist}_F(t, root) - 1$.
2. Ensure that the predecessor / successor of v are neighbors of v and their channel information is consistent with what v knows.
3. If $v = s$ for some (s,t)-channel, ensure there is only one such channel.
4. If v has s as a predecessor, ensure $\mathrm{dist}_F(s, root)$ matches what v knows.
5. If v has t as a successor, ensure $\mathrm{dist}_F(t, root)$ matches what v knows.
6. If v has a neighbor which carries an (s,t)-channel with $v = t$, ensure that $\mathrm{dist}_F(s, root) = \mathrm{dist}_F(v) - 1$.

Verifier's algorithm at each $v \in V(G)$: elimination tree

Assuming that the channel for (s,t)-path P in $\mathcal{P}$ is working properly, the presumed elimination tree is verified.

1. Ensure that there is no neighbor u with $\mathrm{dist}_F(u, root) = \mathrm{dist}_F(v, root)$.
2. Ensure that there is a unique target of a channel starting with v (or $\mathrm{dist}_F(v, root) = 0$).
3. Whenever its neighbor (including its "parent" as a target of a channel starting with v) z is an ancestor, ensure that z is the list L_v
4. Ensure that $|L_v| \le t$.
5. Ensure that $\bigcup_{w \text{ is a neighbor and strict descendant of } v} L_w \subseteq L_v \cup \{v\}$.

Key technical lemma

For any graph G of treewidth at most t, there exists an elimination tree F of G of width at most $f(t)$, and an oriented path system $\mathcal{P}$ of congestion $f(t)$ witnessing F.

The proof relies on some key technical results from Bojańczyk and Pilipczuk (2016): sane tree-decomposition and some consequence of Simon's factorization forest theorem.

Sane tree-decomposition

If G has treewidth at most t, then it admits a sane tree-decomposition (T, β) of width at most t. That is, for every node t with parent t',

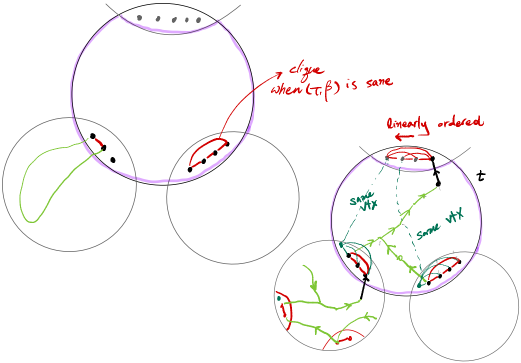
- $\beta(t) \setminus \beta(t') \neq \emptyset$,
- $Y_t := \bigcup_{b \in T_t} \beta(b) \setminus \beta(t')$ is connected, and
- every vertex in $\beta(t) \cap \beta(t')$ is adjacent with some vertex of Y_t.

From a sane tree-decomposition of width at most t, we can construct an elimination tree F of width at most t together with an oriented path system witnessing F.

Fom sane tree-decomposition to Elimination tree

Observe that for each node t of sane (T, β)

- the graph $G[\beta(t) - \mathrm{adh}(t)]$ obtained by "torsofying with lower bags" makes each $\mathrm{adh}(t')$ a clique for each child t' of t.
- Build an elimination tree F by combining an elimination tree F_t as DFS tree for each the above "marginal graph" for each t.
- Choose the root of F_t as a neighbor of the "largest vertex in $\mathrm{adh}(t)$.

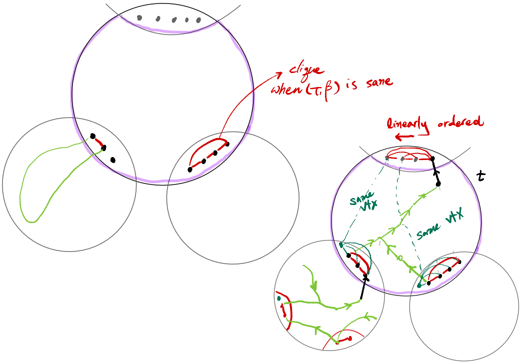


How to get a low-congestion path system

Fom sane tree-decomposition to Elimination tree

Observe that for each node t of sane (T, β)

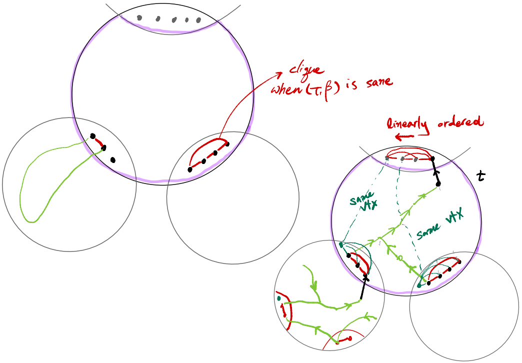
- the graph $G[\beta(t) - \mathrm{adh}(t)]$ obtained by "torsofying with lower bags" makes each $\mathrm{adh}(t')$ a clique for each child t' of t.
- Build an elimination tree F by combining an elimination tree F_t as DFS tree for each the above "marginal graph" for each t.
- Choose the root of F_t as a neighbor of the "largest vertex in $\mathrm{adh}(t)$.



FOM SANE TREE-DECOMPOSITION TO ELIMINATION TREE

Observe that for each node t of sane (T, β)

- the graph $G[\beta(t) - \mathrm{adh}(t)]$ obtained by "torsofying with lower bags" makes each $\mathrm{adh}(t')$ a clique for each child t' of t.
- Build an elimination tree F by combining an elimination tree F_t as DFS tree for each the above "marginal graph" for each t.
- Choose the root of F_t as a neighbor of the "largest vertex in $\mathrm{adh}(t)$.



How to get a low-congestion path system

From a sane tree-decomposition of width at most t, we can construct an elimination tree F of width at most t together with an oriented path system witnessing F.

Reducing the congestion needs much more work: core technical work done in Bojańczyk and Pilipczuk (2016); the main result was to MSO-transduce tree-decompositions of bounded width from graphs of bounded treewidth.

It was done with a nice combinatorial analysis of sane tree-decomposition, and Simon's factorization theorem applied to graphs of bounded pathwidth.

Our work for reducing the congestion builds on this.

Further Questions

Proof labeling scheme for the following properties are open, among others.

- $O(\log n)$-sized PLS for cliquewidth / rankwidth at most t? Open for linear cliquewidth at most t as well.
- $O(\log n)$-sized PLS for H-minor-freeness, for any fixed H?
  Known: H planar, of size at most K, H-minor-free being bounded genus, etc.