Graphs of Bounded Treewidth can be Canonized in AC^1

Fabian Wagner*

CSR 2011

*Institut Theoretische Informatik, Universität Ulm, Germany
fabian.wagner@uni-ulm.de

June 14-18, 2011

1. Graph Isomorphism (GI)

Graph: $G_i = (V, E_i)$, vertices $V = \{1, \ldots, n\}$, edges $E_i \subseteq V \times V$

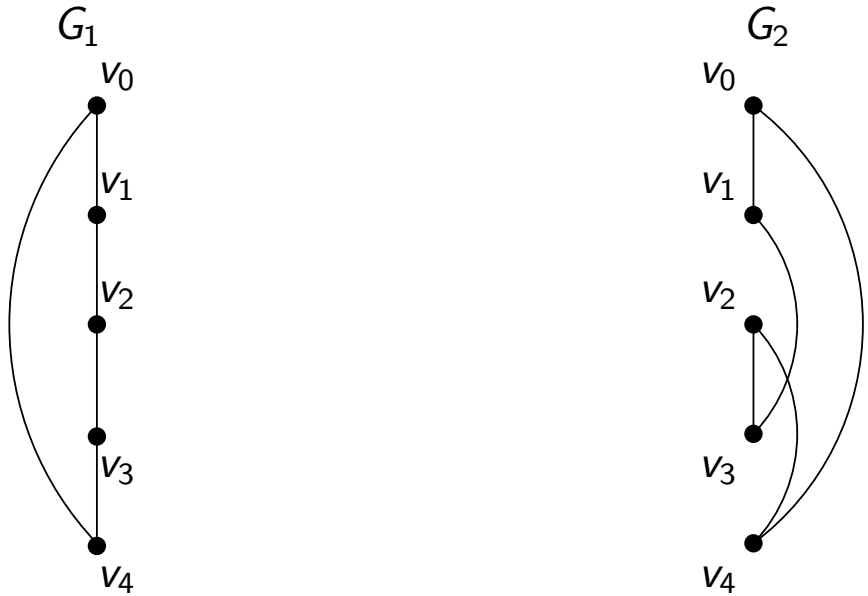
Isomorphism: $G_1 \cong G_2$ iff $\exists \phi \in S_n$ such that $\forall u, v \in V$:

$$(u, v) \in E_1 \quad \Leftrightarrow \quad (\phi(u), \phi(v)) \in E_2$$

Graph Isomorphism: $\mathsf{GI} = \{(G_1, G_2) \mid G_1 \cong G_2\}$

Complete invariant: $f : \mathcal{G} \mapsto \Sigma^*$ s.t. $f(G_1) = f(G_2) \Leftrightarrow G_1 \cong G_2$

Canonical form: $f : \mathcal{G} \mapsto \mathcal{G}$ s.t. $f(G_1) = f(G_2) \cong G_1 \Leftrightarrow G_1 \cong G_2$



1. Graph Isomorphism (GI)

Graph: $G_i = (V, E_i)$, vertices $V = \{1, \ldots, n\}$, edges $E_i \subseteq V \times V$

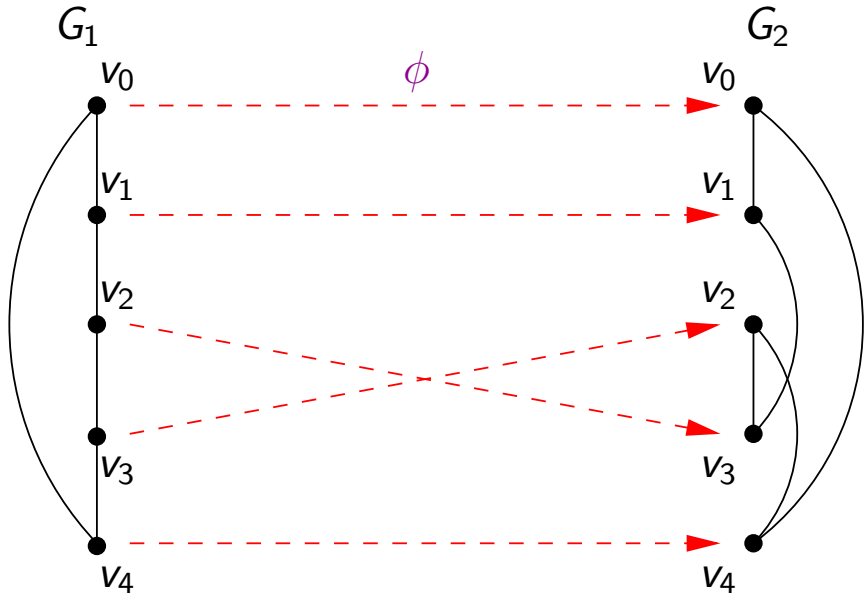
Isomorphism: $G_1 \cong G_2$ iff $\exists \phi \in S_n$ such that $\forall u, v \in V$:

$$(u, v) \in E_1 \quad \Leftrightarrow \quad (\phi(u), \phi(v)) \in E_2$$

Graph Isomorphism: $\mathsf{GI} = \{(G_1, G_2) \mid G_1 \cong G_2\}$

Complete invariant: $f : \mathcal{G} \mapsto \Sigma^*$ s.t. $f(G_1) = f(G_2) \Leftrightarrow G_1 \cong G_2$

Canonical form: $f : \mathcal{G} \mapsto \mathcal{G}$ s.t. $f(G_1) = f(G_2) \cong G_1 \Leftrightarrow G_1 \cong G_2$



1. Graph Isomorphism (GI)

Graph: $G_i = (V, E_i)$, vertices $V = \{1, \ldots, n\}$, edges $E_i \subseteq V \times V$

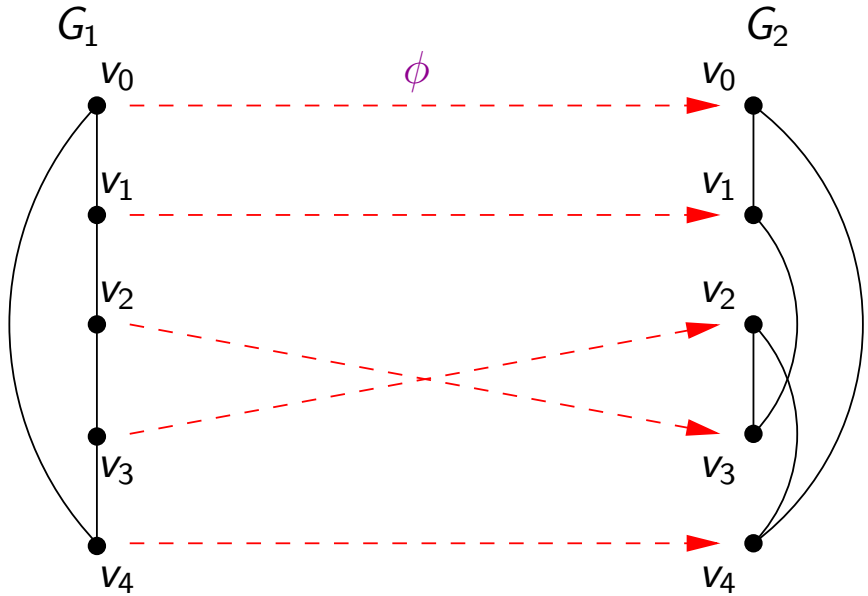
Isomorphism: $G_1 \cong G_2$ iff $\exists \phi \in S_n$ such that $\forall u, v \in V$:

$$(u, v) \in E_1 \quad \Leftrightarrow \quad (\phi(u), \phi(v)) \in E_2$$

Graph Isomorphism: $\mathsf{GI} = \{(G_1, G_2) \mid G_1 \cong G_2\}$

Complete invariant: $f : \mathcal{G} \mapsto \Sigma^*$ s.t. $f(G_1) = f(G_2) \Leftrightarrow G_1 \cong G_2$

Canonical form: $f : \mathcal{G} \mapsto \mathcal{G}$ s.t. $f(G_1) = f(G_2) \cong G_1 \Leftrightarrow G_1 \cong G_2$

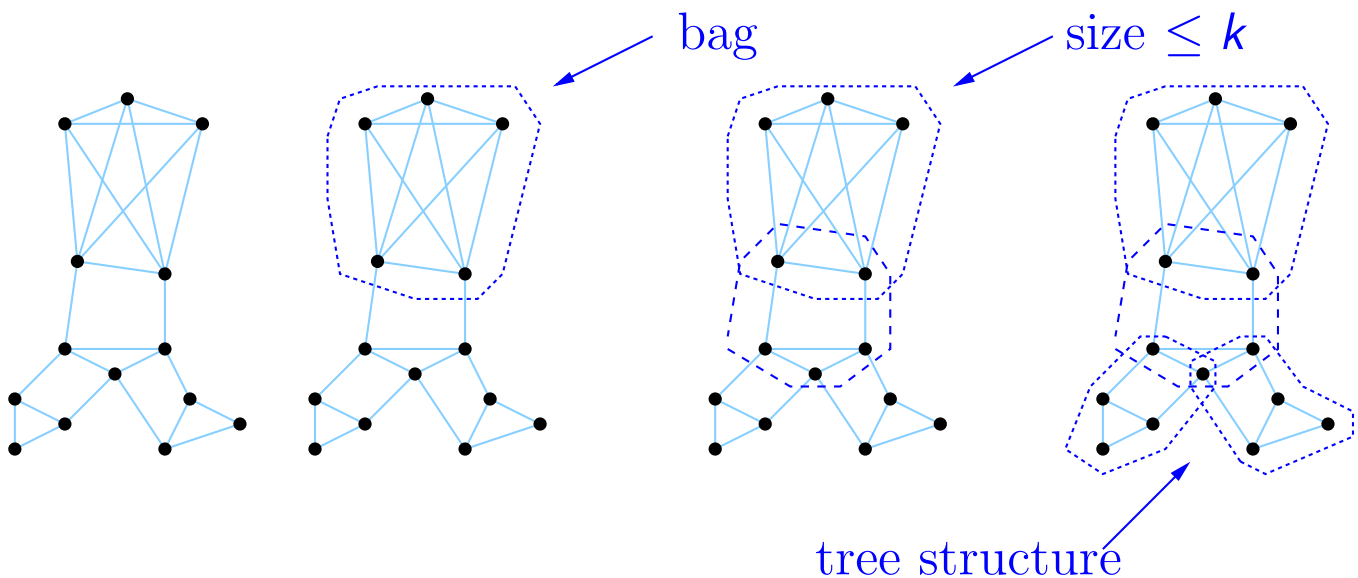


1. Bounded Treewidth Graphs

Tree decomposition of width k:

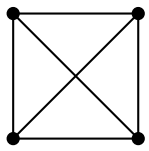
$D = (\{X_i \mid i \in I\}, T = (I, F))$

- every vertex in a bag X_i
- every edge in a bag X_i
- all bags which contain vertex v form a connected subtree
- treewidth $k-1$ $\Leftrightarrow$ bags have size $\leq k$

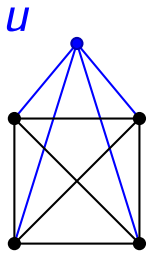


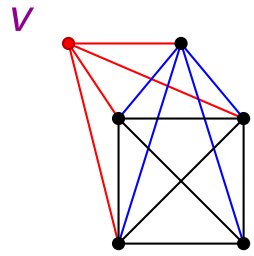
1. Bounded Treewidth graphs

k-tree:

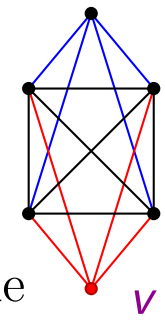


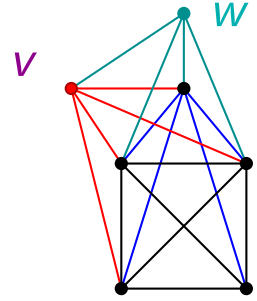
k-clique



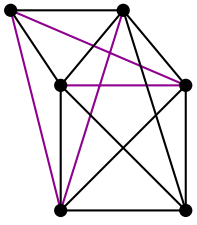


connect new vertex to a k-clique



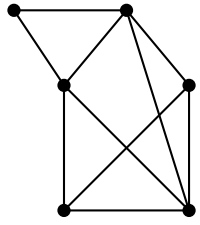


k-tree



subgraph

partial k-tree, treewidth k graph:



1. Results

Bounded Treewidth graph (BTW-graph)

Known Results:

- Isomorphism: BTW-graphs in P [Bod90]
- Complete invariant: BTW-graphs in TC^1 [GroVer06]
- Canonical labeling: BTW-graphs in TC^2 [KöbVer08]

New Results:

- Computation: tree decomposition in L [EJT10]
- Isomorphism: BTW-graphs in LogCFL [DTW10]
- Canonical labeling: BTW-graphs in AC^1

1. Results

Restricted classes of BTW-graphs

Known Results:

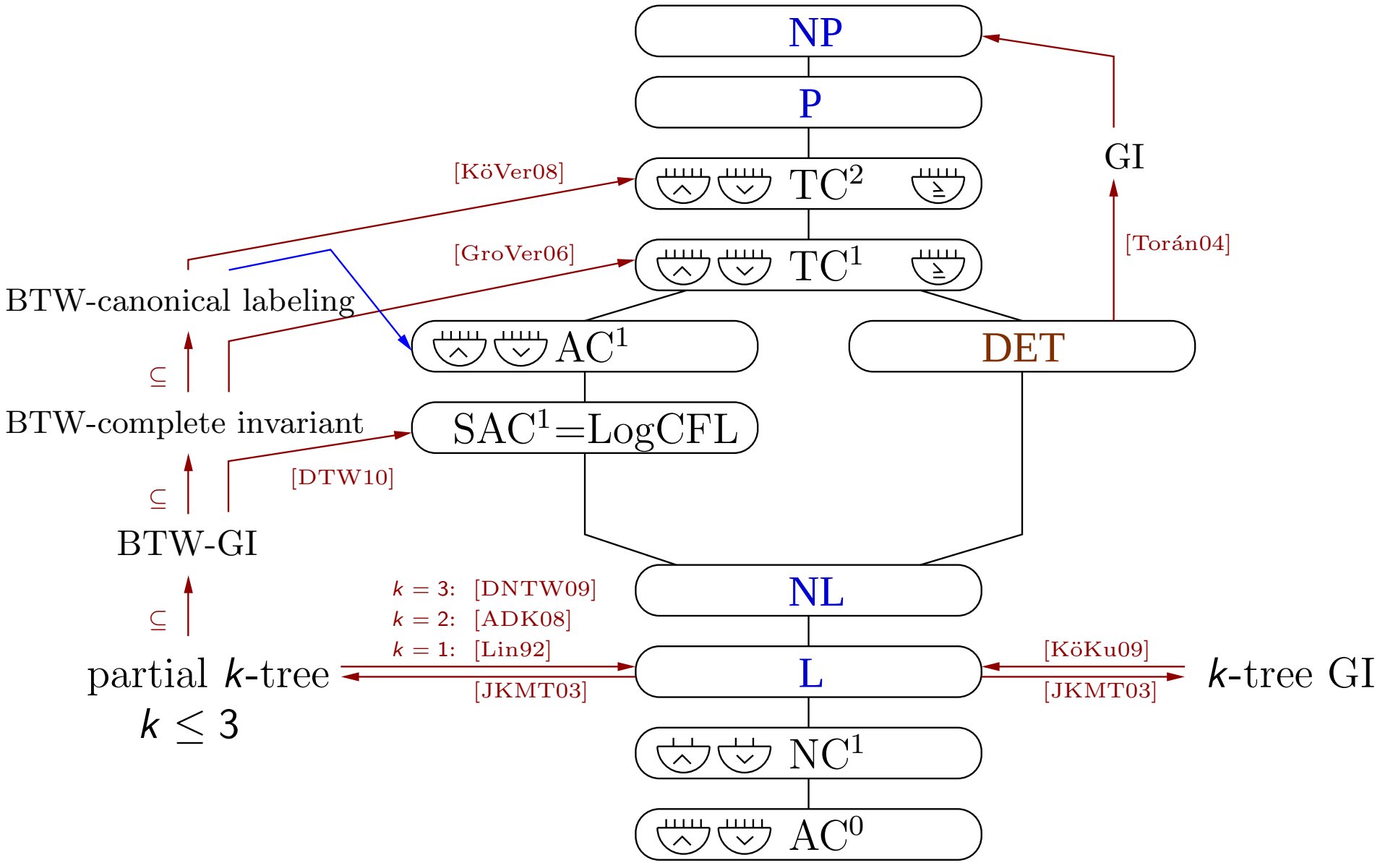
- Canonical labeling: BTDW-graphs in L [DTW10]
- Canonical labeling: k-trees in L [KöbKu09]
- Canonical labeling: partial 2-trees in L [ADK08]
- Canonical labeling: $K_{3,3}$- and K_5-free graphs in L [DNTW09]

  $\rightarrow$ partial 3-trees are K_5-free

Open:

- Canonical labeling: partial 4-trees in L?

1. Complexity



2. BTW-Canonization: ingredients

- **Theorem:** $\{G \mid G \text{ has treewidth} \leq k\}$ in L [EJT10]

- for graph G, treewidth k,

  approximate tree decomposition: width $4k+3$, depth $O(\log n)$

  Theorem: approximate tree decomposition in L [EJT10]

- number of bags: $\leq n^{4k+3}$ bags of size $4k+3$

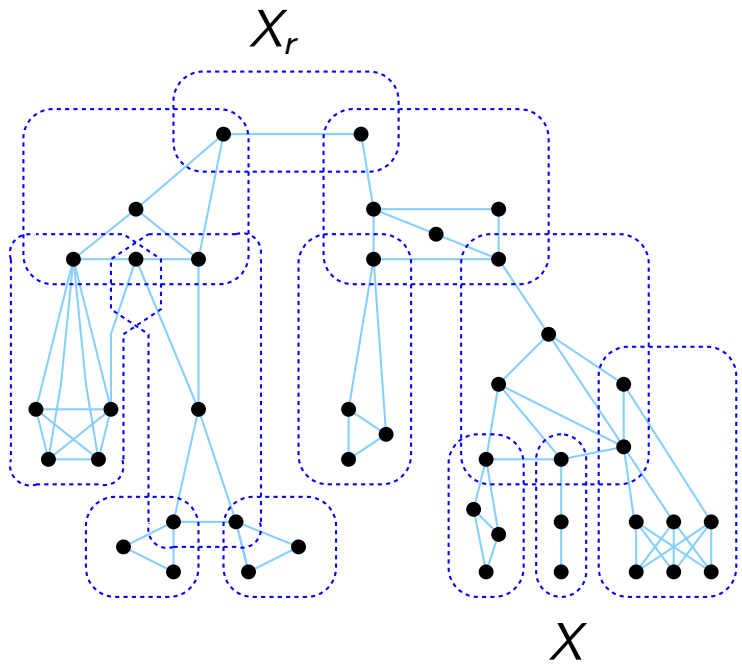
- pre-computation in L:
  - all possible bags,
  - all split components of bags w.r.t. fixed root X_r
  - all possible child bags w.r.t. fixed root X_r

2. BTW-Canonization: minimal description

$X = \{u, v, w\}$

$\sigma = \binom{u\ v\ w}{u\ w\ v}$

$$G[X]^\sigma = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

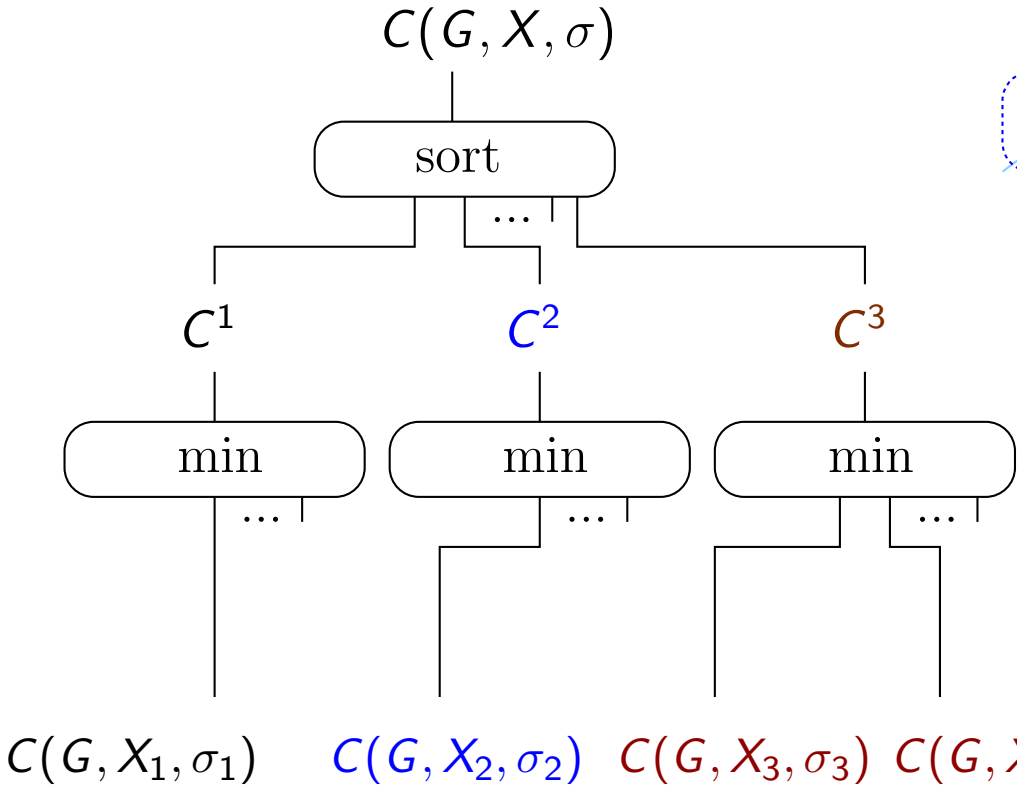


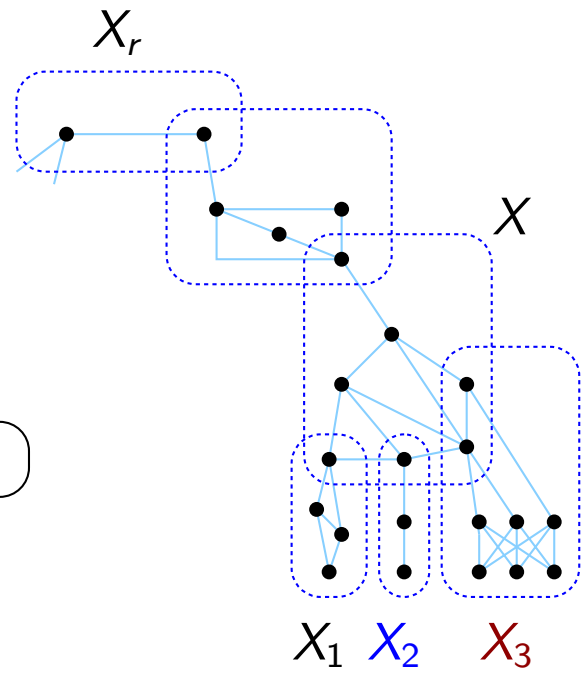
Definition: minimal description:

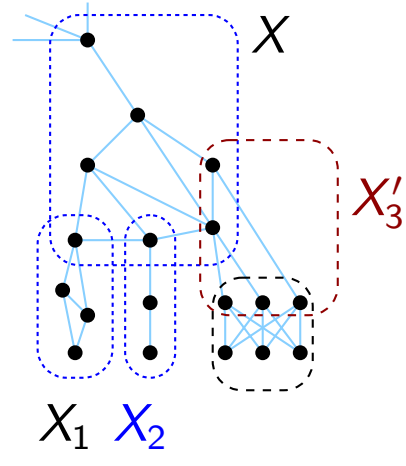
$C(G, X, \sigma) = \quad C_0(G, X, \sigma) \quad C_0'(G, X, \sigma) \quad [children]$

$C(G, X, \sigma) = \ 010\ 101\ 010\ 2^{(4k+3)^2-|X|^2} \quad uwv2^{(4k+3-|X|)\lceil \log n \rceil}\ [\ ]$

2. BTW-Canonization: inductive construction



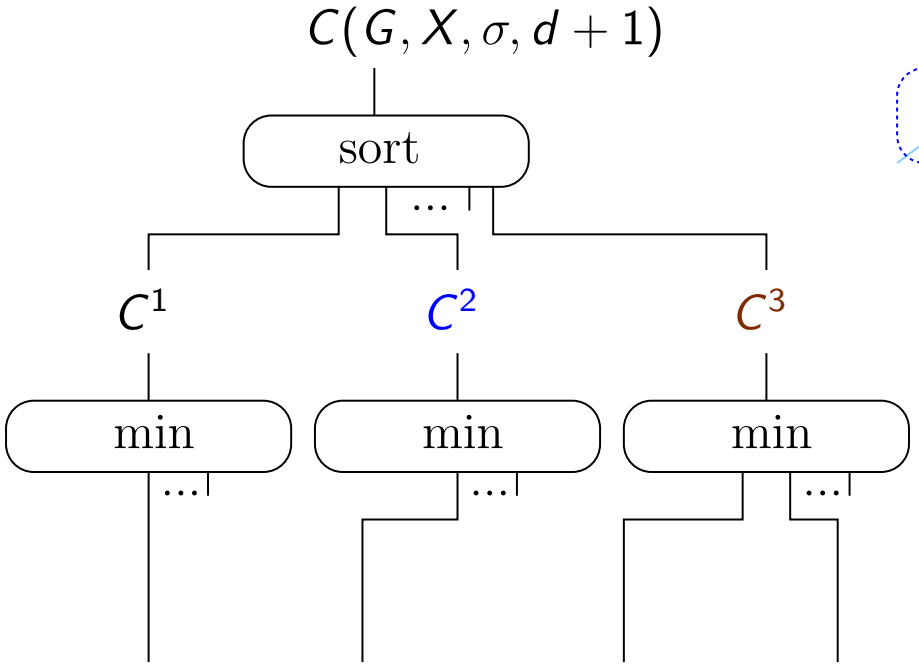


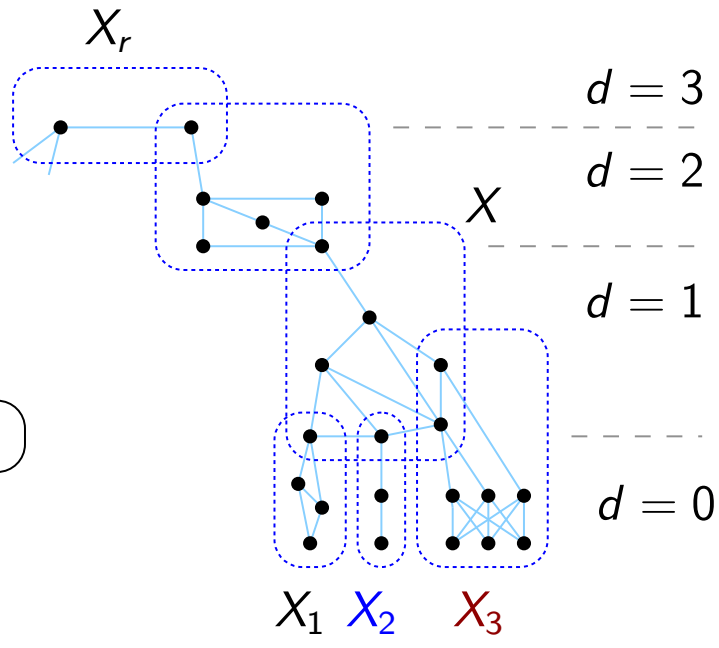


minimal description for i-th split component:

$C^i = \min_{X_i, \sigma_i} C(G, X_i, \sigma_i)[...]$

2. BTW-Canonization: depth parameter



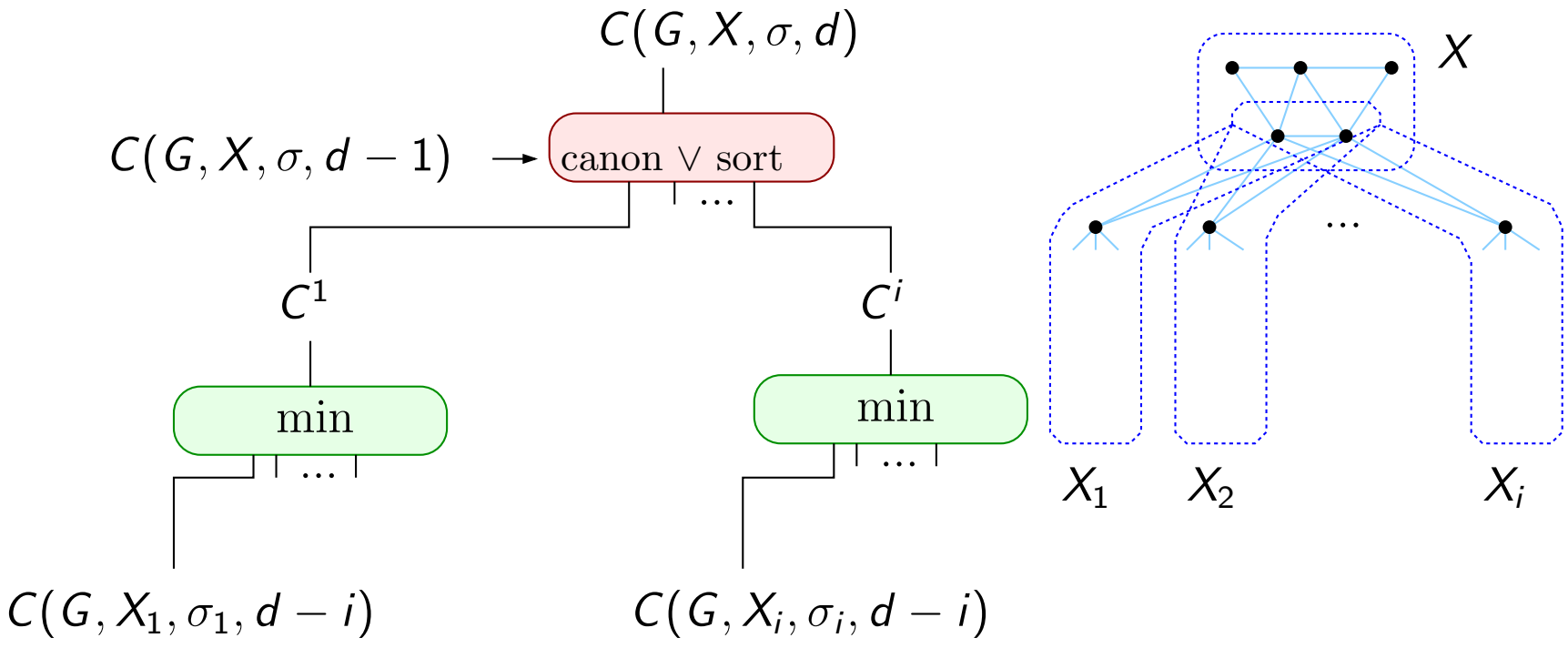


$C(G, X_1, \sigma_1, d)$ $C(G, X_2, \sigma_2, d)$ $C(G, X_3, \sigma_3, d)$ $C(G, X_3', \sigma_3', d)$

minimal description for i-th split component:

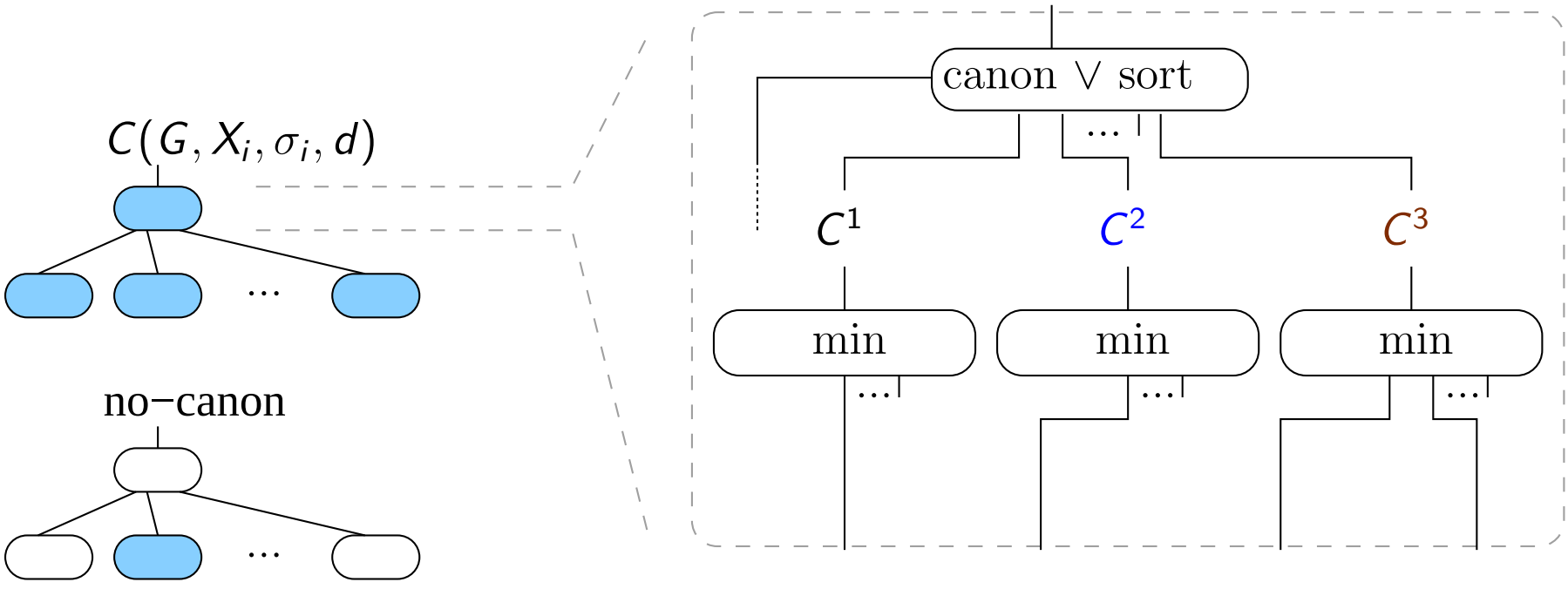
$C^i = min_{X_i, \sigma_i} C(G, X_i, \sigma_i, d)$

2. BTW-Canonization: circuit depth

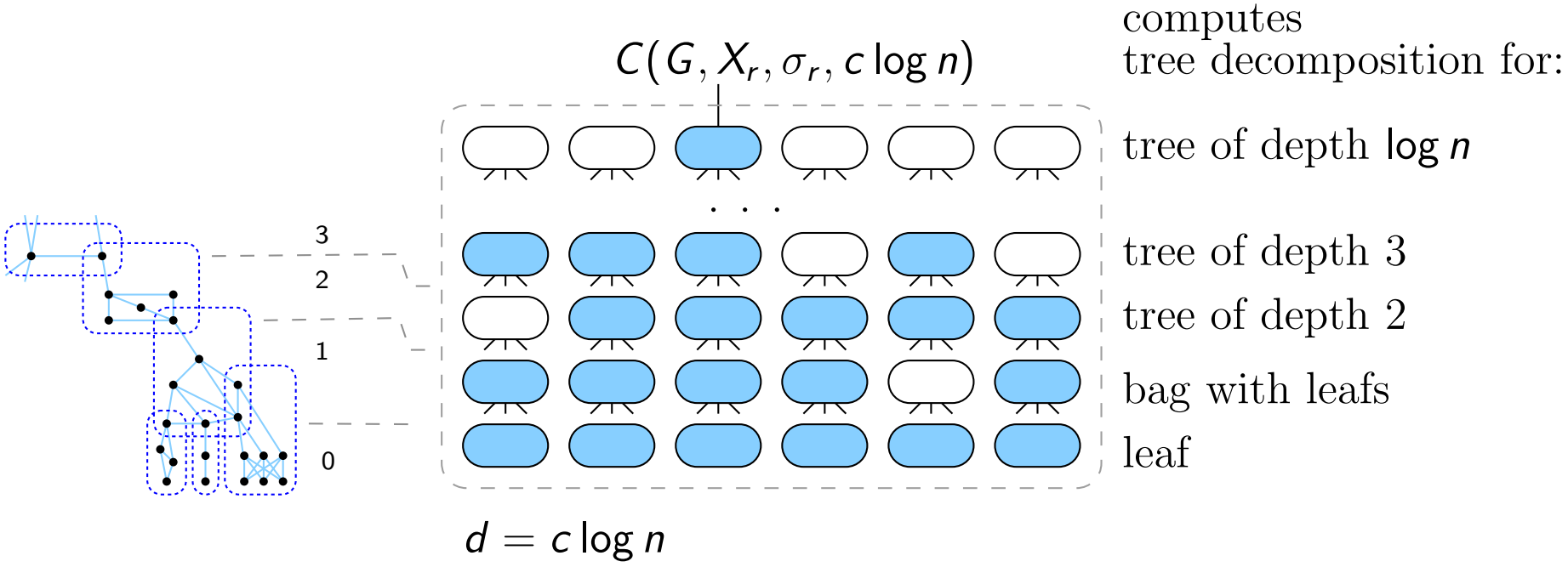


- minimum: first order definable, in AC^0 [Imm99]
- sort: i children of equal size $\leq n/i$:
  with $O(\log i)$-depth AC^1-circuits

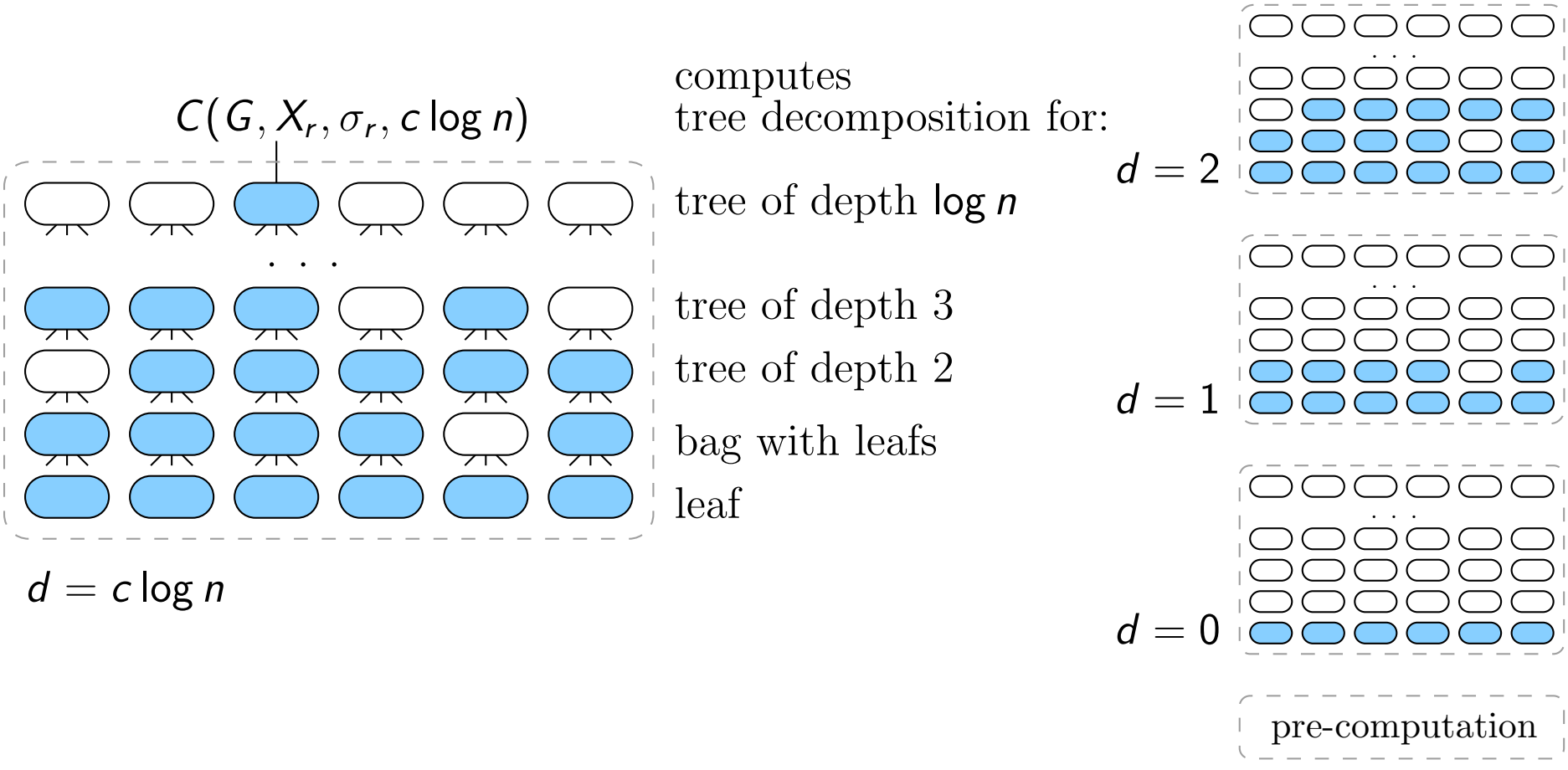
2. BTW-Canonization: example



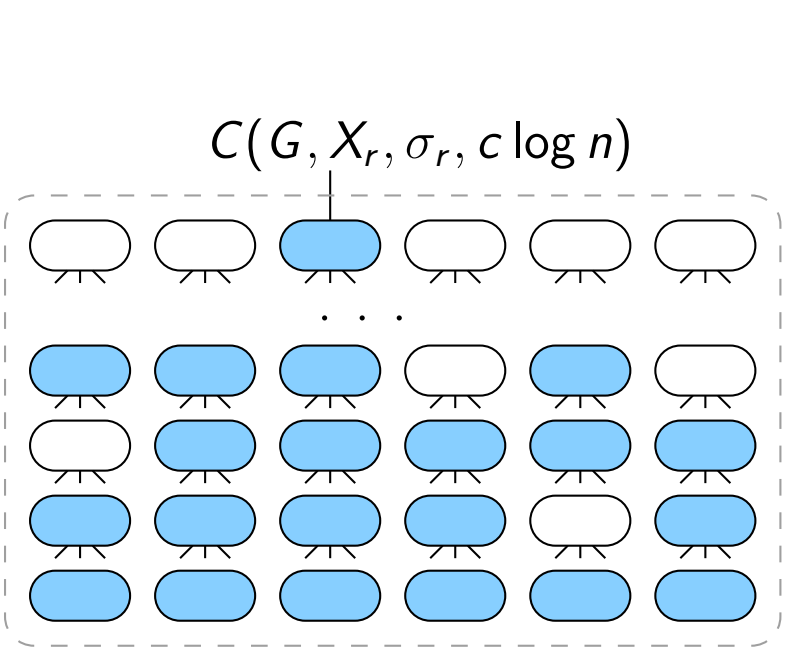
2. BTW-Canonization: example



2. BTW-Canonization: example



2. BTW-Canonization: example



computes
tree decomposition for:

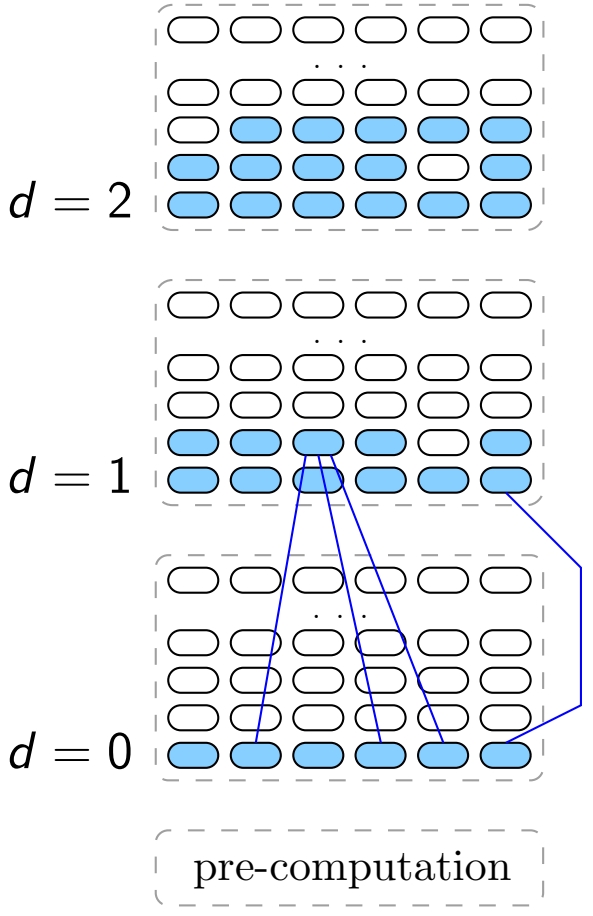
tree of depth $\log n$

tree of depth 3

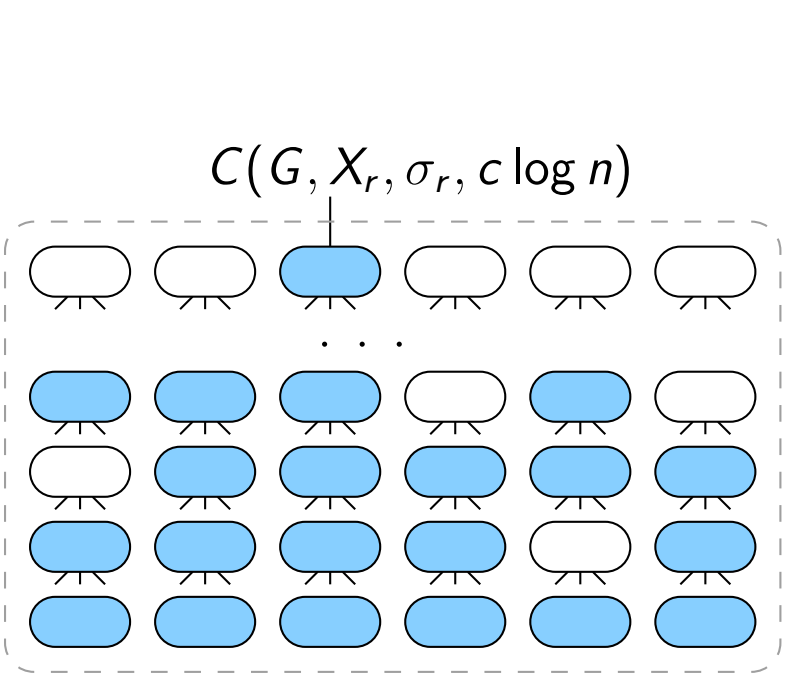
tree of depth 2

bag with leafs

leaf



2. BTW-Canonization: example



computes
tree decomposition for:

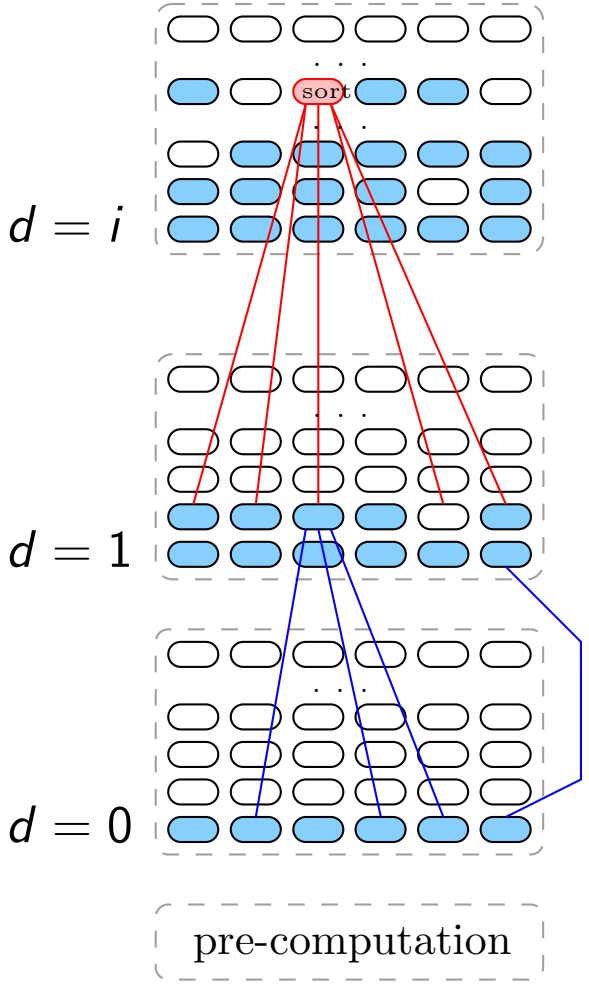
tree of depth $\log n$

tree of depth 3

tree of depth 2

bag with leafs

leaf



2. BTW-Canonization: algorithm

input: graph G, treewidth k

output: minimal description C

1: **for all** bags $X_r = \{v_1, \ldots, v_{4k+3}\} \subseteq V$ **in parallel do**

2: **for all** $\sigma \in Sym(X_r)$ **in parallel do**

3: $C = \min\{C, C(G, X_r, \sigma, c \log n)\}$ **recursively**

4: relabel vertices in C in increasing order in L [DLN08]

5: **return** C $\rightarrow$ complete invariant

return new labels $\rightarrow$ canonical form

- $C = 010101010222\ uvw222\ [0101010102222\ xyw222 \ldots]$
  $\rightarrow u = 1, v = 2, w = 3, x = 4, y = 5, \ldots$

2. BTW-Canonization: algorithm

input: graph G, treewidth k

output: canonical labeling

1: **for all** bags $X_r = \{v_1, \ldots, v_{4k+3}\} \subseteq V$ **in parallel do**
2: **for all** $\sigma \in Sym(X_r)$ **in parallel do**
3: $C = \min\{C, C(G, X_r, \sigma, c \log n)\}$ **recursively**
4: relabel vertices in C in increasing order in L [DLN08]
5: **return** C $\rightarrow$ complete invariant

return new labels $\rightarrow$ canonical form

- $C = 010101010222$ *uvw*222 [0101010102222 *xyw*$222 \ldots$]
  $\rightarrow u = 1, v = 2, w = 3, x = 4, y = 5, \ldots$

6. Tools for Canonization

Decomposition

tree decomposition [DTW10,Wag11]
tree distance decomposition [DTW10]
2-connected component tree [DLNTW09,DNTW09]
3-connected component tree [DLNTW09,DNTW09]
four-connected component tree [DNTW09]
colored tree [KK09]
tree representation of inverval graphs [KKLV10]

→ merge canons
→ analyse resources [Lin92]

Reduction

BTW-Canon $\leq$ BTDW-Canon [DTW10]

Canonize Pieces

constant size components [DTW10,Wag11]
3-connected planar components [DLN08]

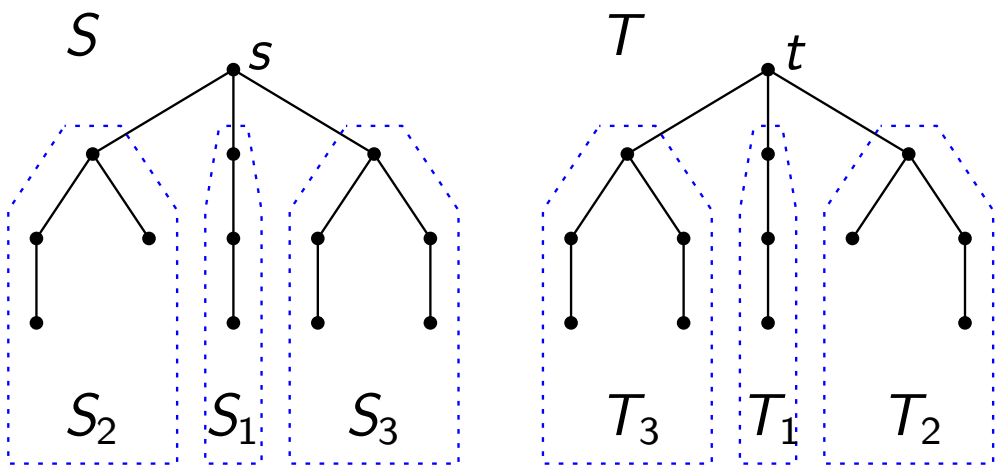
Computational models

AC^1-circuit [Wag11]
NAuxPDA [DTW10]
logspace machine [Lin92]

A. PlanarGI $\in$ L - Canonize Biconnected Components

Isomorphism order $S < T$ if [Lindell92]:

1. $|S| < |T|$ or
2. $|S| = |T|$ but $\#s < \#t$ or
3. $|S| = |T|$ and $\#s = \#t = k$ but $(S_1, \ldots, S_k) < (T_1, \ldots, T_k)$

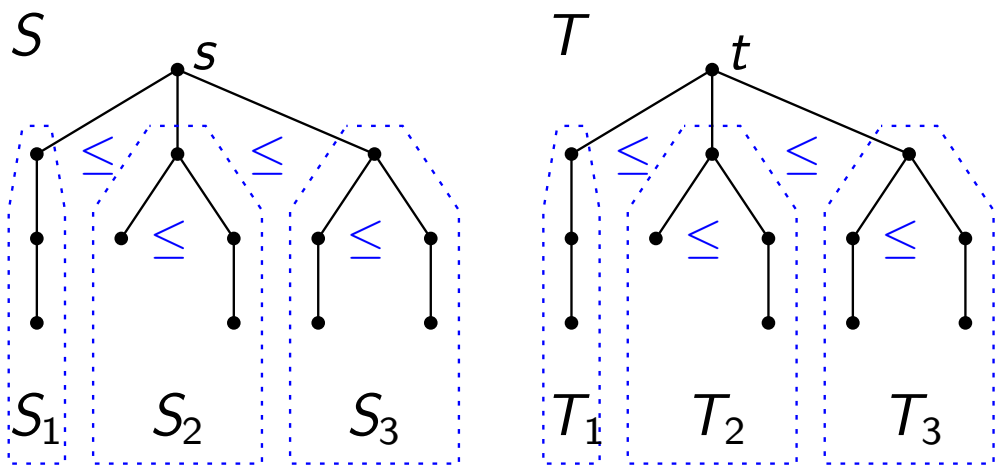


$canon(S) =$
$($ **(**$(())$**)** **(**$()()$**)** **(**$\ldots$**)** $)$

A. PlanarGI $\in$ L - Canonize Biconnected Components

Isomorphism order $S < T$ if [Lindell92]:

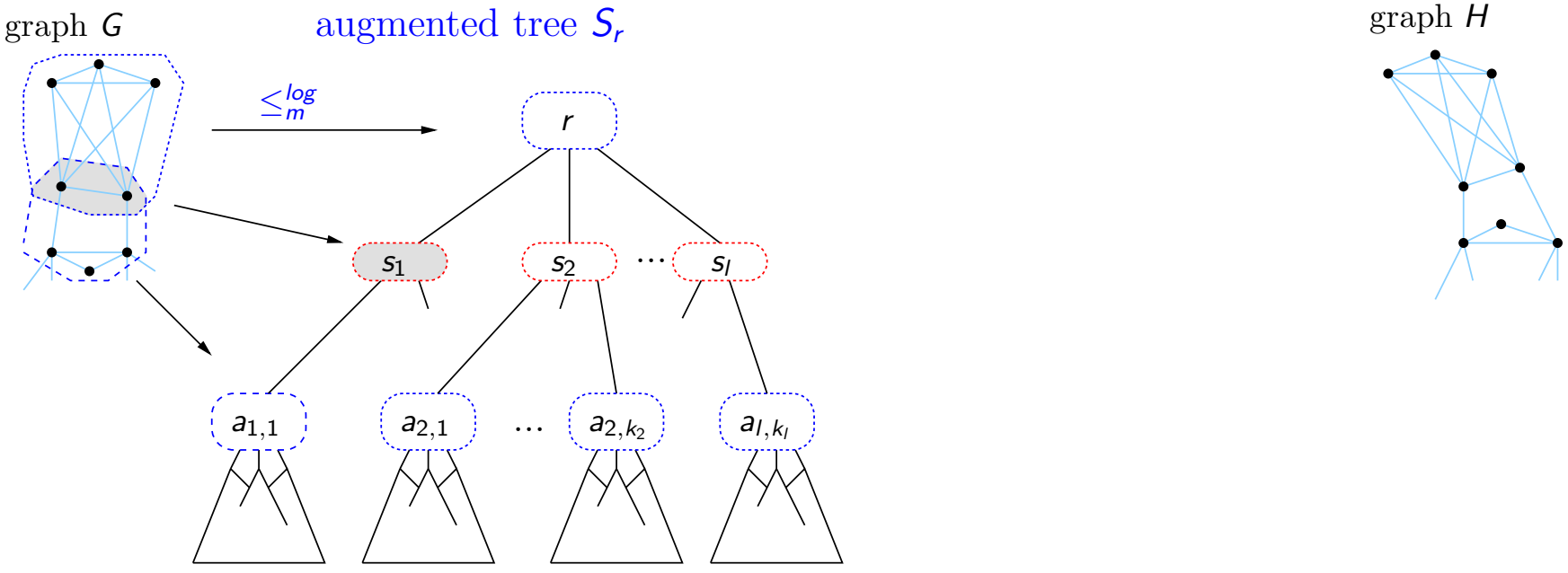
1. $|S| < |T|$ or
2. $|S| = |T|$ but $\#s < \#t$ or
3. $|S| = |T|$ and $\#s = \#t = k$ but $(S_1, \ldots, S_k) < (T_1, \ldots, T_k)$



$canon(S) =$
(**(**(()) **)** **(**()() **)** **(**. . . **)**)

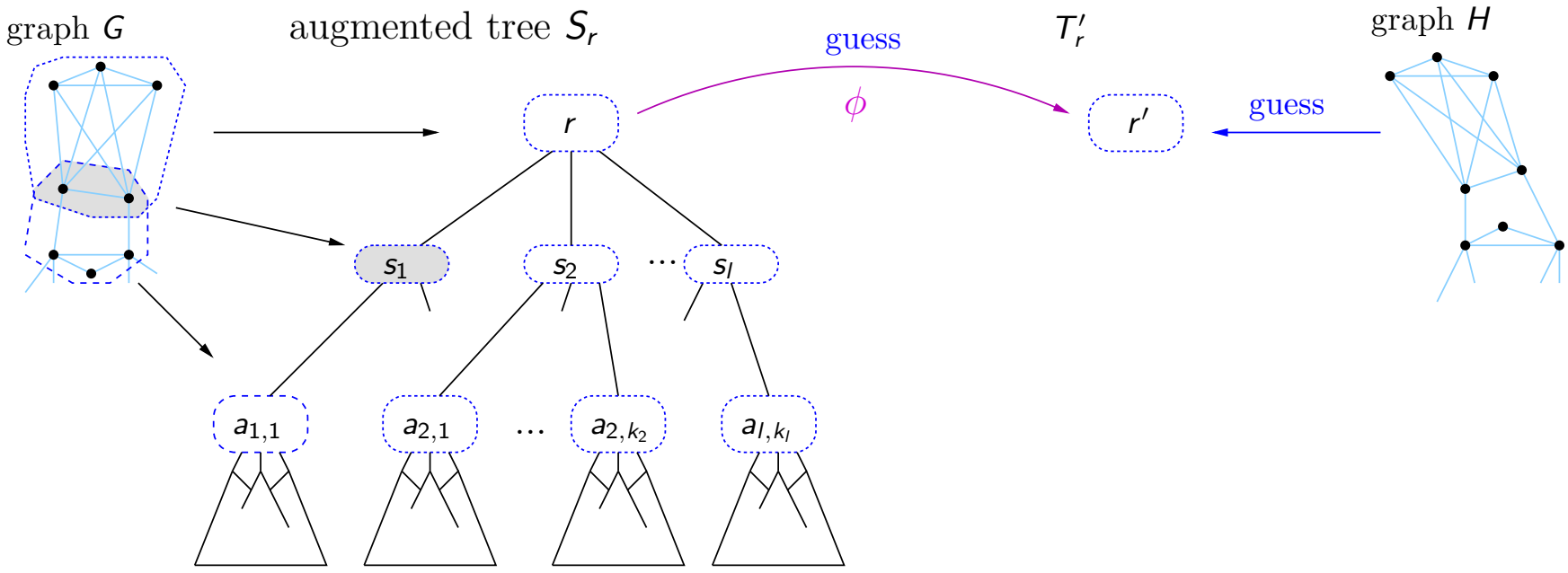
BTW-GI in LogCFL

- compute tree decomposition for G [Wanke94,GoLeSc02,EJT10]
- compute augmented tree S_r for G
- **guess** root r' in H, $\phi : X_r \mapsto X_{r'}$
- **if** r', ϕ not consistent **then reject**
- recursion: $(s_1 \leq_T \cdots \leq_T s_l) \leq_T (t_1 \leq_T \cdots \leq_T t_l)$
- return



BTW-GI in LogCFL

- compute tree decomposition for G [Wanke94,GoLeSc02,EJT10]
- compute augmented tree S_r for G
- **guess** root r' in H, $\phi : X_r \mapsto X_{r'}$
- **if** r', ϕ not consistent **then reject**
- recursion: $(s_1 \leq_T \cdots \leq_T s_l) \leq_T (t_1 \leq_T \cdots \leq_T t_l)$
- return



BTW-GI in LogCFL

- compute tree decomposition for G [Wanke94,GoLeSc02,EJT10]
- compute augmented tree S_r for G
- **guess** root r' in H, $\phi : X_r \mapsto X_{r'}$
- **if** r', ϕ not consistent **then reject**
- recursion: $(s_1 \leq_T \cdots \leq_T s_l) \leq_T (t_1 \leq_T \cdots \leq_T t_l)$
- return