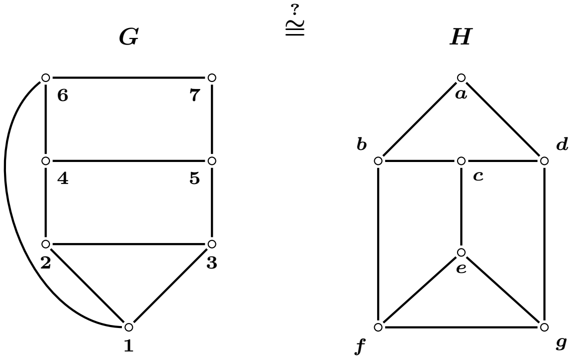
Dynamic Planar Graph Isomorphism is in DynFO

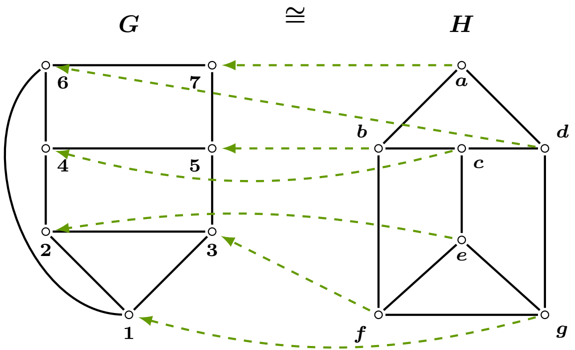
Samir Datta Asif Khan Felix Tschirbs
Nils Vortmeier Thomas Zeume

AlMoTh, 3 March 2026

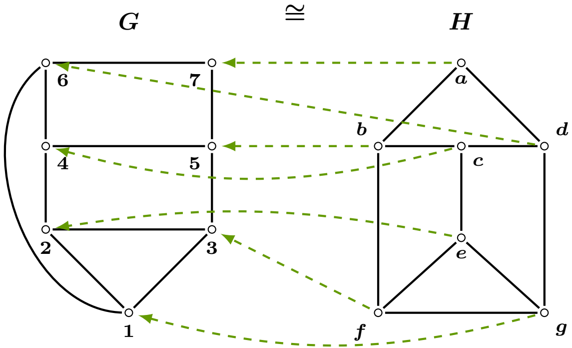
The Planar Graph Isomorphism Problem



The Planar Graph Isomorphism Problem



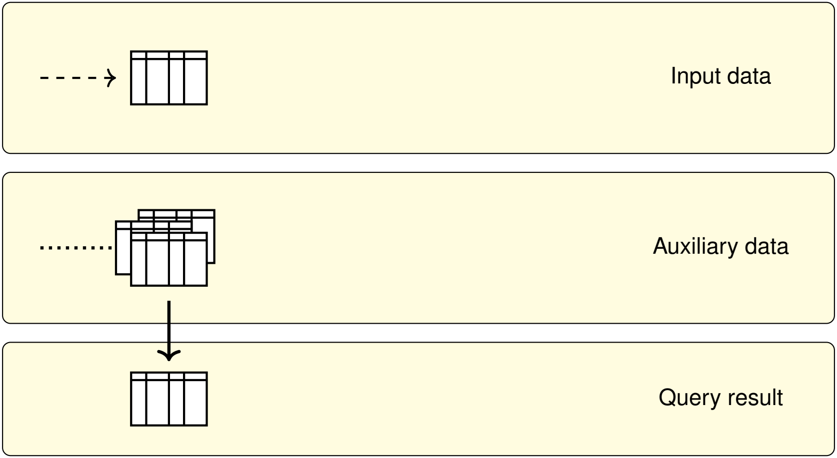
The Planar Graph Isomorphism Problem



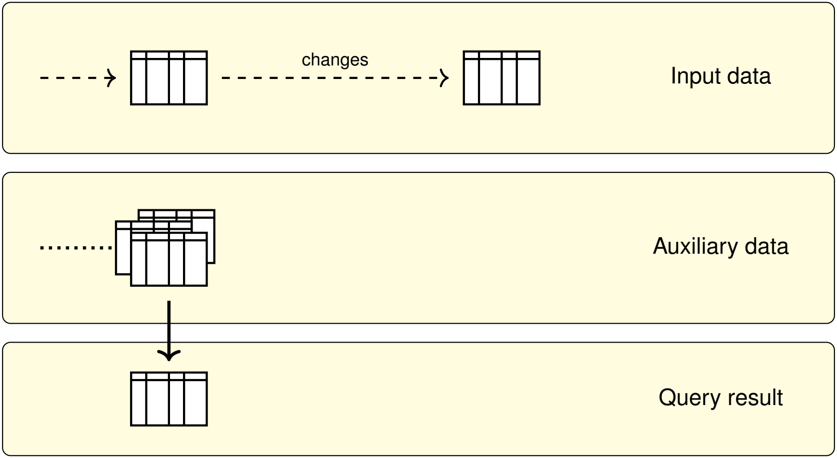
Isomorphism of planar graphs can be solved

- in linear time [Hopcroft and Wong 1974]
- with logarithmic space [Datta et al. 2022]

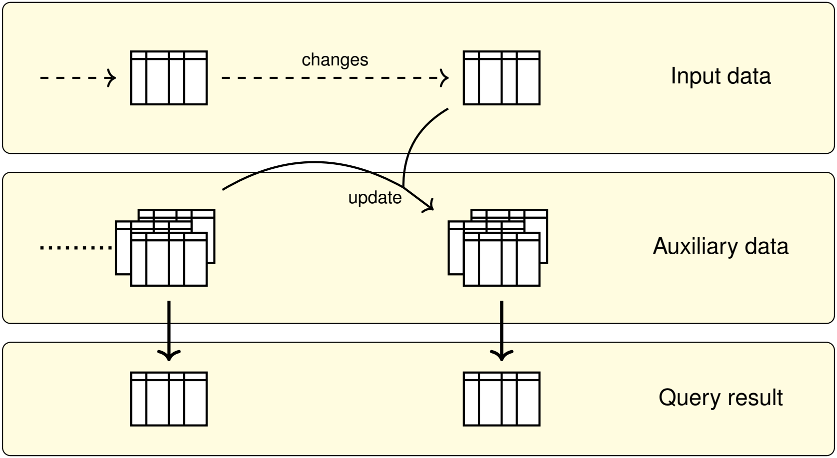
Setting of Dynamic Query Evaluation



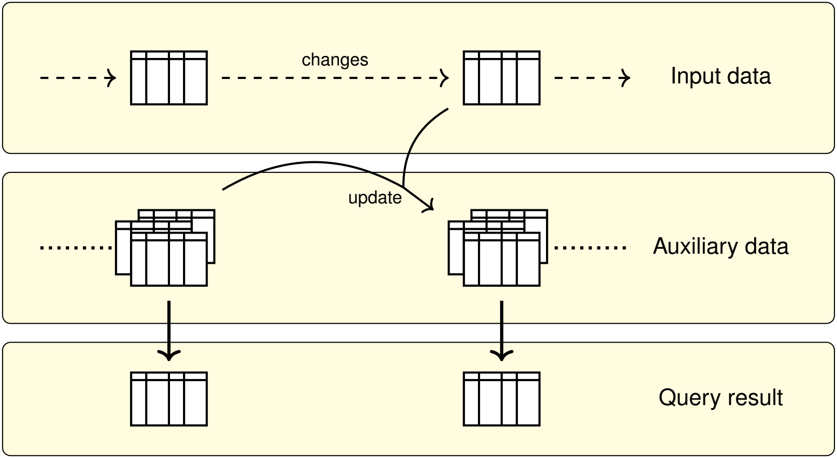
Setting of Dynamic Query Evaluation



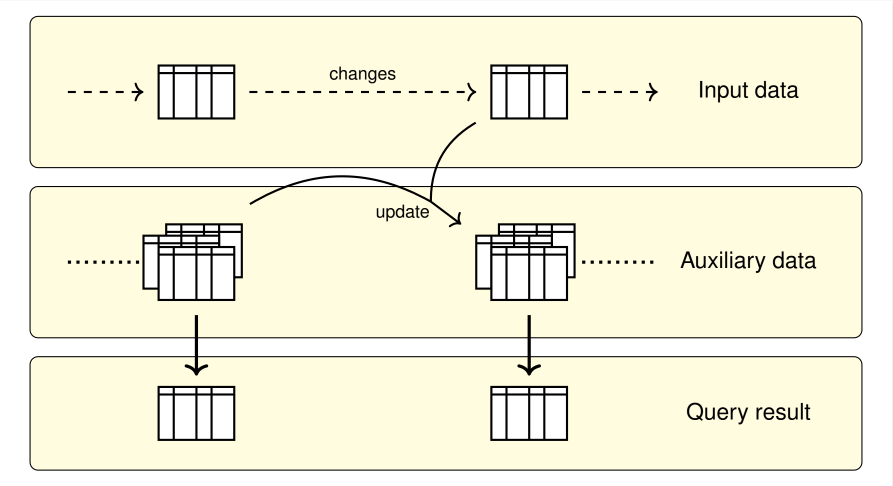
Setting of Dynamic Query Evaluation



Setting of Dynamic Query Evaluation



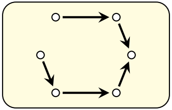
Setting of Dynamic Query Evaluation



- Descriptive approach: express update using logical formulas
  - input and auxiliary data is relational

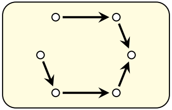
Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]

G



Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]

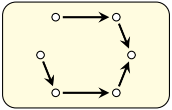
G



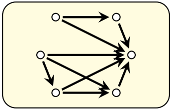
- **Goal**: Maintain transitive closure relation T of input graph G

Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]

G

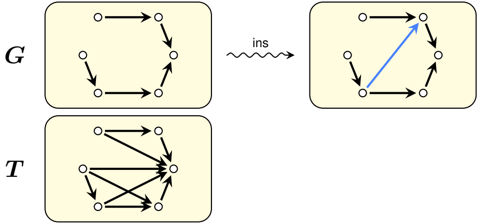


T



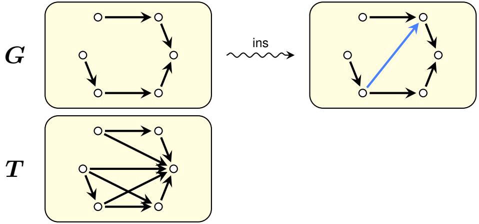
- **Goal**: Maintain transitive closure relation T of input graph G

Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



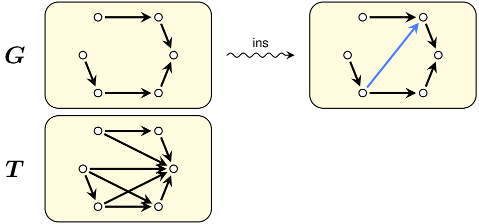
- **Goal**: Maintain transitive closure relation T of input graph G

Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



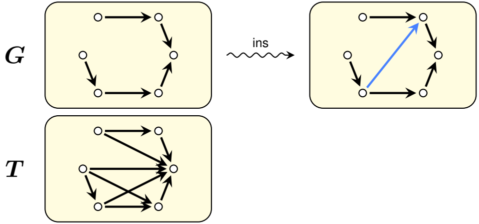
- **Goal**: Maintain transitive closure relation T of input graph G
- After insertion of edge (u, v): path $x \rightsquigarrow y$ exists

Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



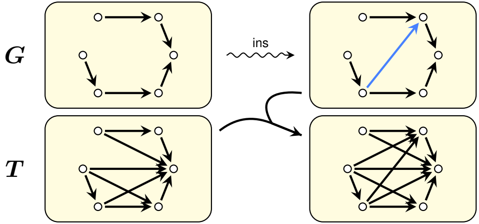
- **Goal**: Maintain transitive closure relation T of input graph G
- After insertion of edge (u, v): path $x \rightsquigarrow y$ exists
  1. if path $x \rightsquigarrow y$ existed before, or

Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



- **Goal**: Maintain transitive closure relation T of input graph G
- After insertion of edge (u, v): path $x \rightsquigarrow y$ exists
  1. if path $x \rightsquigarrow y$ existed before, or
  2. paths $x \rightsquigarrow u$ and $v \rightsquigarrow y$ existed before

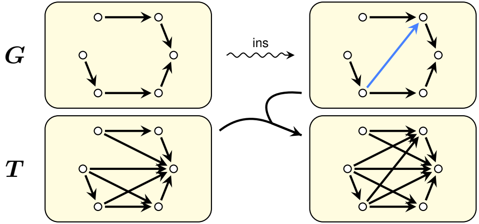
Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



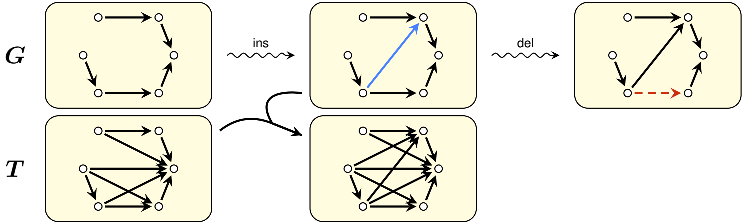
- **Goal**: Maintain transitive closure relation T of input graph G
- After insertion of edge (u, v): path $x \rightsquigarrow y$ exists
  1. if path $x \rightsquigarrow y$ existed before, or
  2. paths $x \rightsquigarrow u$ and $v \rightsquigarrow y$ existed before
- Update formula for T after edge insertion:

$$\phi^T_{\text{ins}}(x, y; u, v) = T(x, y) \vee \big(T(x, u) \wedge T(v, y)\big)$$

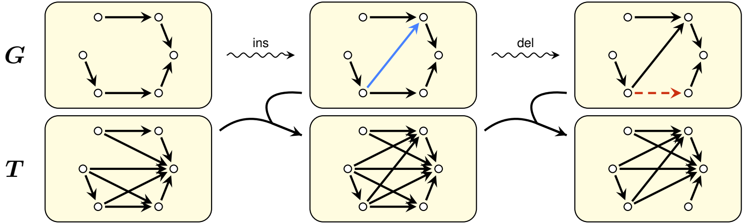
Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



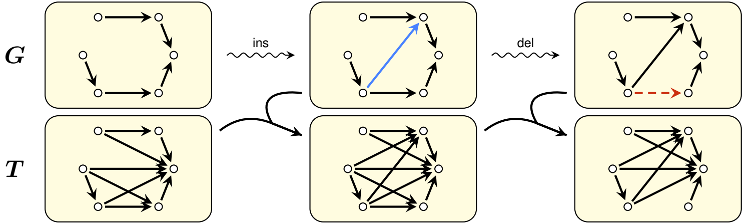
Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



- Update formula for T after edge deletion:

$$\phi^T_{\text{del}}(x, y; u, v) = T(x, y) \wedge \Big[\neg\big(T(x, u) \wedge T(v, y)\big) \vee \exists z \exists z' \big((z \neq u \vee z' \neq v) \wedge T(x, z) \wedge E(z, z') \wedge T(z', y) \wedge T(z, u) \wedge \neg T(z', u)\big)\Big]$$

Example: Reachability for acyclic graphs [Patnaik and Immerman 1997]



- Update formula for T after edge deletion:

$$\phi^{T}_{\text{del}}(x, y; u, v) = T(x, y) \wedge \Big[\neg\big(T(x, u) \wedge T(v, y)\big) \vee \exists z \exists z' \big((z \neq u \vee z' \neq v) \wedge T(x, z) \wedge E(z, z') \wedge T(z', y) \wedge T(z, u) \wedge \neg T(z', u)\big)\Big]$$

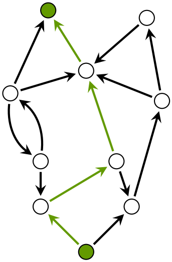
DynFO

Queries maintained using first-order update formulas

Prior DynFO maintainability results

Reachability is in **DynFO** for

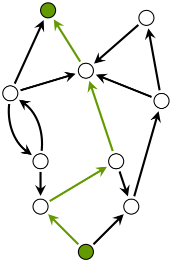
- acyclic graphs
  [Dong and Su 1995, Patnaik and Immerman 1997]
- undirected graphs
  [Patnaik and Immerman 1997, Dong and Su 1998, Grädel and Siebertz 2012]
- directed graphs
  [Datta, Kulkarni, Mukherjee, Schwentick, Zeume 2015]



Prior DynFO maintainability results

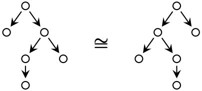
Reachability is in **DynFO** for

- acyclic graphs
  [Dong and Su 1995, Patnaik and Immerman 1997]
- undirected graphs
  [Patnaik and Immerman 1997, Dong and Su 1998, Grädel and Siebertz 2012]
- directed graphs
  [Datta, Kulkarni, Mukherjee, Schwentick, Zeume 2015]



Graph queries in **DynFO** that are relevant for this talk:

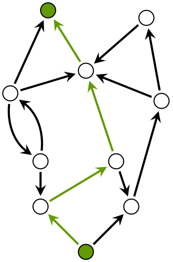
- Tree isomorphism
  [Etessami 1998]



Prior DynFO maintainability results

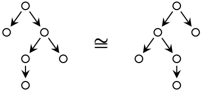
Reachability is in **DynFO** for

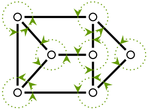
- acyclic graphs
  [Dong and Su 1995, Patnaik and Immerman 1997]
- undirected graphs
  [Patnaik and Immerman 1997, Dong and Su 1998, Grädel and Siebertz 2012]
- directed graphs
  [Datta, Kulkarni, Mukherjee, Schwentick, Zeume 2015]



Graph queries in **DynFO** that are relevant for this talk:

- Tree isomorphism
  [Etessami 1998]
- Graph planarity + maintenance of plane embedding
  [Datta, Khan, Mukherjee 2023]





Main result

Theorem

The dynamic graph isomorphism problem for planar graphs is in **DynFO**

- Allowed changes: inserting and deleting single edges
- The graph has to stay planar the whole time

Main result

Theorem

The dynamic graph isomorphism problem for planar graphs is in **DynFO**

- Allowed changes: inserting and deleting single edges
- The graph has to stay planar the whole time

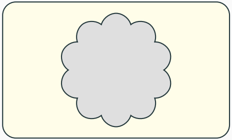
Theorem, rephrased

The dynamic graph isomorphism problem for planar graphs can be maintained by

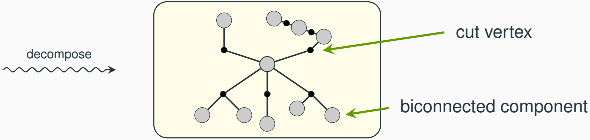
- $\mathbf{AC^0}$-circuits
- a dynamic constant-time parallel algorithm

with auxiliary data of polynomial size

Proof overview

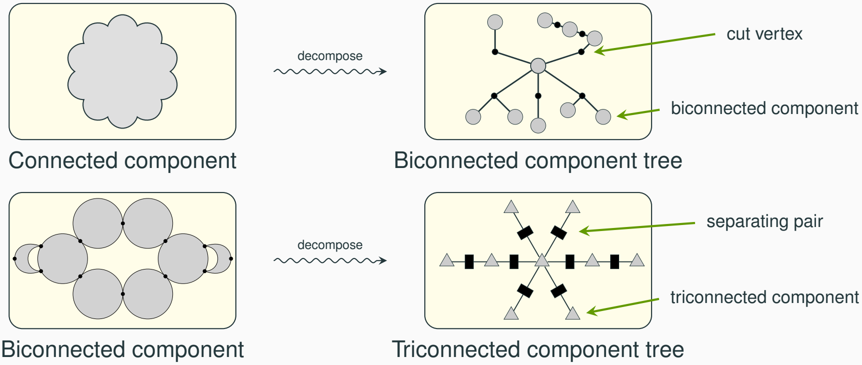


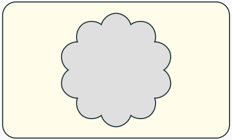
Connected component



Biconnected component tree

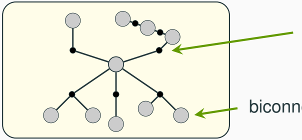
Proof overview



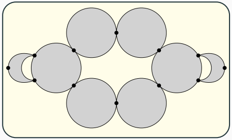


Connected component

decompose

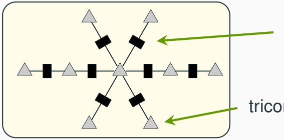


Biconnected component tree

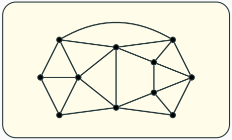


Biconnected component

decompose



Triconnected component tree



Triconnected component

solve

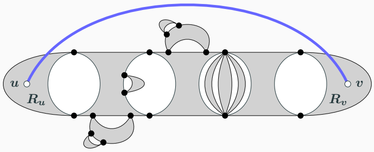
determine isomorphism via canonical embedding

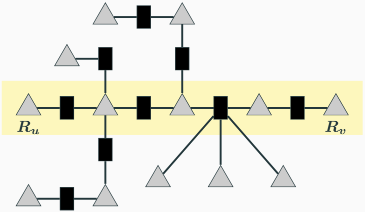
Effects of edge changes on the component trees

- Difficulty: biconnected and triconnected component trees may change drastically after an edge change

Effects of edge changes on the component trees

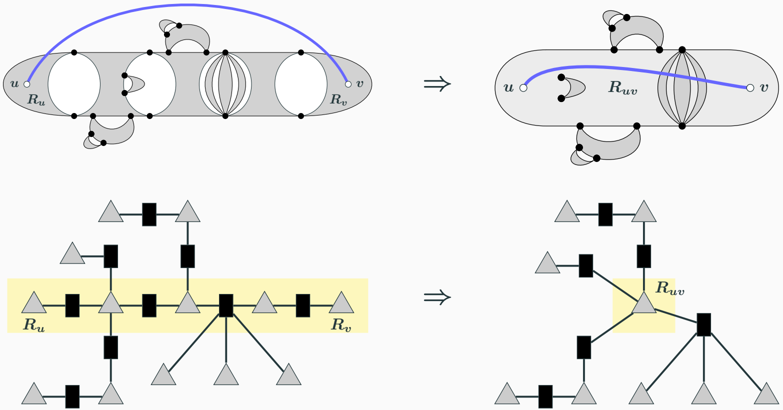
- Difficulty: biconnected and triconnected component trees may change drastically after an edge change





Effects of edge changes on the component trees

- Difficulty: biconnected and triconnected component trees may change drastically after an edge change



Technical contributions

The main technical challenges:

Technical contributions

The main technical challenges:

1. Maintaining embedding information for 3-connected components

Technical contributions

The main technical challenges:

1. Maintaining embedding information for $\mathbf{3}$-connected components

   ▷ Linear algebra techniques, extending e.g. [Datta, Mukherjee, V., Zeume 2018]

Technical contributions

The main technical challenges:

1. Maintaining embedding information for $\mathbf{3}$-connected components

    ▷ Linear algebra techniques, extending e.g. [Datta, Mukherjee, V., Zeume 2018]

2. Maintaining the component trees, including information like distances

Technical contributions

The main technical challenges:

1. Maintaining embedding information for 3-connected components

   ▷ Linear algebra techniques, extending e.g. [Datta, Mukherjee, V., Zeume 2018]

2. Maintaining the component trees, including information like distances

   ▷ Investigation of properties of combinatorial embeddings of planar graphs, building on [Datta, Khan, Mukherjee 2023]

Technical contributions

The main technical challenges:

1. Maintaining embedding information for $\mathbf{3}$-connected components

   ▷ Linear algebra techniques, extending e.g. [Datta, Mukherjee, V., Zeume 2018]

2. Maintaining the component trees, including information like distances

   ▷ Investigation of properties of combinatorial embeddings of planar graphs, building on [Datta, Khan, Mukherjee 2023]

3. Maintaining isomorphism of the component trees

Technical contributions

The main technical challenges:

1. Maintaining embedding information for 3-connected components

   ▷ Linear algebra techniques, extending e.g. [Datta, Mukherjee, V., Zeume 2018]

2. Maintaining the component trees, including information like distances

   ▷ Investigation of properties of combinatorial embeddings of planar graphs, building on [Datta, Khan, Mukherjee 2023]

3. Maintaining isomorphism of the component trees

   ▷ Generalizing Etessami's approach for tree isomorphism [Etessami 1998]

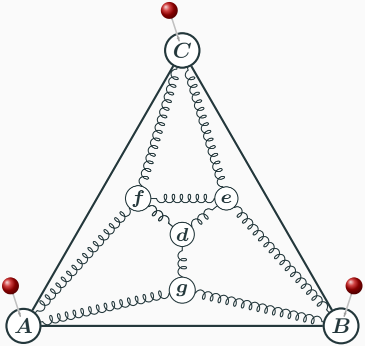
Maintaining isomorphism of 3-connected components

Embedding 3-connected planar graphs

- We use Tutte's spring embedding [Tutte 1963]

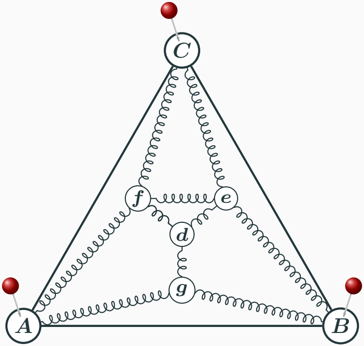
Embedding 3-connected planar graphs

- We use Tutte's spring embedding [Tutte 1963]
- Given a planar $\mathbf{3}$-connected graph:
  1. Pin three vertices to the plane
  2. Replace edges by springs, if they have at least one unpinned endpoint
  3. Wait until the non-pinned vertices become stationary



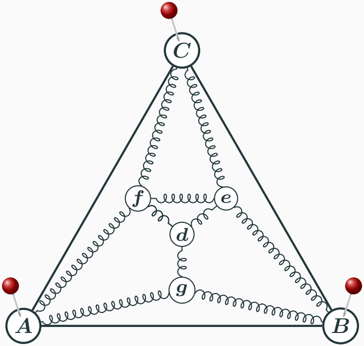
Embedding 3-connected planar graphs

- We use Tutte's spring embedding [Tutte 1963]
- Given a planar $\mathbf{3}$-connected graph:
  1. Pin three vertices to the plane
  2. Replace edges by springs, if they have at least one unpinned endpoint
  3. Wait until the non-pinned vertices become stationary
- The result gives a canonical planar embedding of the graph



Embedding 3-connected planar graphs

- We use Tutte's spring embedding [Tutte 1963]
- Given a planar $\mathbf{3}$-connected graph:
  1. Pin three vertices to the plane
  2. Replace edges by springs, if they have at least one unpinned endpoint
  3. Wait until the non-pinned vertices become stationary
- The result gives a canonical planar embedding of the graph



- Idea of our approach:
  1. Maintain the spring embedding for each possible choice of pinned vertices
  2. Determine whether $\mathbf{3}$-connected components are isomorphic by checking whether their spring embeddings are equal (for some choice of pinned vertices)

Maintaining the spring embedding

- Fixing vertices A, B, C to $(0,0), (0,1), (1,0)$ gives as equations for the x-coordinates

$$\begin{aligned} \deg(v) x_v &= \sum_{(v,w) \in E} x_w && \forall v \notin \{A, B, C\} \\ x_A &= 0 \\ x_B &= 0 \\ x_C &= 1 \end{aligned}$$

Maintaining the spring embedding

- Fixing vertices A, B, C to $(0,0), (0,1), (1,0)$ gives as equations for the x-coordinates

$$\begin{aligned} \deg(v) x_v &= \sum_{(v,w) \in E} x_w && \forall v \notin \{A, B, C\} \\ x_A &= 0 \\ x_B &= 0 \\ x_C &= 1 \end{aligned}$$

- ...which can be written as $\mathbf{T}\,\mathbf{x} = \mathbf{b}$, for some matrix $\mathbf{T}$ and $\mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$

Maintaining the spring embedding

- Fixing vertices A, B, C to $(0,0), (0,1), (1,0)$ gives as equations for the x-coordinates

$$\begin{aligned} \deg(v) x_v &= \sum_{(v,w)\in E} x_w && \forall v \notin \{A, B, C\} \\ x_A &= 0 \\ x_B &= 0 \\ x_C &= 1 \end{aligned}$$

- ...which can be written as $\mathbf{T}\,\mathbf{x} = \mathbf{b}$, for some matrix $\mathbf{T}$ and $\mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$

- Goal: maintain the inverse $\mathbf{T}^{-1}$
  - then obtain the embedding via $\mathbf{x} = \mathbf{T}^{-1}\,\mathbf{b}$

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes
- Matrix inverses do not exist modulo some small primes!

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes

- Matrix inverses do not exist modulo some small primes!
  - ▷ We maintain the entries modulo more small primes ☺
  - ▷ . . . and periodically recompute the information [Datta, Mukherjee, Schwentick, V., Zeume 2019]

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes
- Matrix inverses do not exist modulo some small primes!
  - ▷ We maintain the entries modulo more small primes ☺
  - ▷ ...and periodically recompute the information [Datta, Mukherjee, Schwentick, V., Zeume 2019]
- The inverse $\mathbf{T}^{-1}$ has to be updated after changing $\mathbf{T}$ using **FO**!

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes
- Matrix inverses do not exist modulo some small primes!
  - ▷ We maintain the entries modulo more small primes ☺
  - ▷ …and periodically recompute the information [Datta, Mukherjee, Schwentick, V., Zeume 2019]
- The inverse $\mathbf{T}^{-1}$ has to be updated after changing $\mathbf{T}$ using **FO**!
  - ▷ For small changes, the Sherman-Morrison-Woodbury (SMW) formula can be used [Henderson and Searle 1981]

$$(\mathbf{T}+\mathbf{U}\mathbf{V}^{\mathsf{T}})^{-1} = \mathbf{T}^{-1} - \mathbf{T}^{-1}\mathbf{U}(\mathbf{I}+\mathbf{V}^{\mathsf{T}}\mathbf{T}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathsf{T}}\mathbf{T}^{-1}$$

Challenges for maintaining the matrix inverse

- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes
- Matrix inverses do not exist modulo some small primes!
  - ▷ We maintain the entries modulo more small primes ☺
  - ▷ ... and periodically recompute the information [Datta, Mukherjee, Schwentick, V., Zeume 2019]
- The inverse $\mathbf{T}^{-1}$ has to be updated after changing $\mathbf{T}$ using **FO**!
  - ▷ For small changes, the Sherman-Morrison-Woodbury (SMW) formula can be used [Henderson and Searle 1981]

    $$(\mathbf{T} + \mathbf{U}\mathbf{V}^{\mathsf{T}})^{-1} = \mathbf{T}^{-1} - \mathbf{T}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^{\mathsf{T}}\mathbf{T}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathsf{T}}\mathbf{T}^{-1}$$
- Many $\mathbf{3}$-connected components may merge after an edge insertion!

Challenges for maintaining the matrix inverse

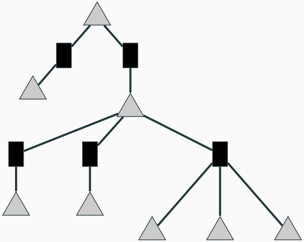
- The entries of the matrix inverse get very large!
  - ▷ We maintain the entries modulo small primes
- Matrix inverses do not exist modulo some small primes!
  - ▷ We maintain the entries modulo more small primes ☺
  - ▷ …and periodically recompute the information [Datta, Mukherjee, Schwentick, V., Zeume 2019]
- The inverse $\mathbf{T}^{-1}$ has to be updated after changing $\mathbf{T}$ using **FO**!
  - ▷ For small changes, the Sherman-Morrison-Woodbury (SMW) formula can be used [Henderson and Searle 1981]

    $$(\mathbf{T} + \mathbf{U}\mathbf{V}^\intercal)^{-1} = \mathbf{T}^{-1} - \mathbf{T}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^\intercal\mathbf{T}^{-1}\mathbf{U})^{-1}\mathbf{V}^\intercal\mathbf{T}^{-1}$$
- Many $\mathbf{3}$-connected components may merge after an edge insertion!
  - ▷ We maintain the inverses also for all $\mathbf{3}$-connected components that are the result of an edge insertion
    - ⋆ Then, in all cases only a constant number of matrices need to be combined
  - ▷ We maintain the result of some terms of the SMW formula

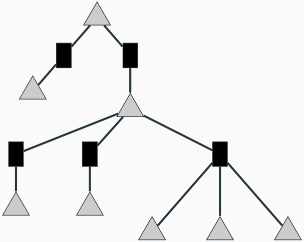
Maintaining isomorphism of 2-connected components

Auxiliary data for 2-connected components

- To maintain: Given two triconnected component trees, are the described graphs isomorphic?

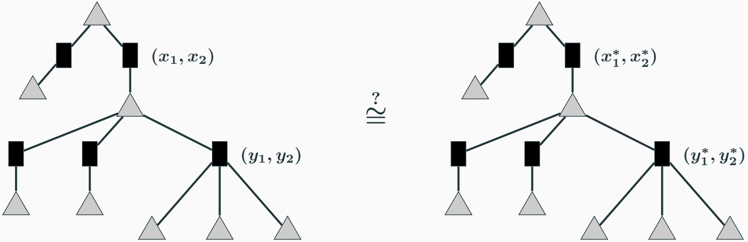


$\stackrel{?}{\cong}$



Auxiliary data for 2-connected components

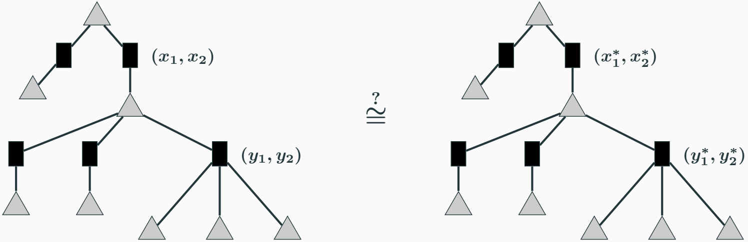
- To maintain: Given two triconnected component trees, are the described graphs isomorphic?



- We maintain implications of the following form as auxiliary data
  - If the graph of the subtree of (y_1, y_2) is isomorphic to the graph of the subtree of (y_1^*, y_2^*)
  - then so are the graphs of the subtree of (x_1, x_2) and of the subtree of (x_1^*, x_2^*)

Auxiliary data for 2-connected components

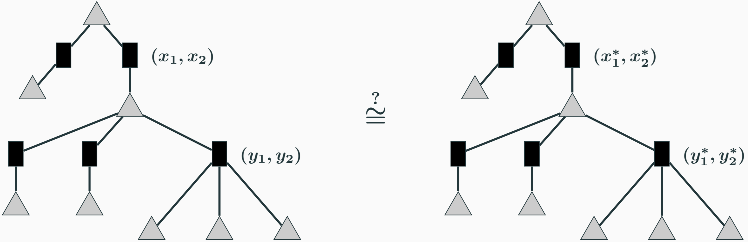
- To maintain: Given two triconnected component trees, are the described graphs isomorphic?



- We maintain implications of the following form as auxiliary data
  - If the graph of the subtree of (y_1, y_2) is isomorphic to the graph of the subtree of (y_1^*, y_2^*)
  - then so are the graphs of the subtree of (x_1, x_2) and of the subtree of (x_1^*, x_2^*)
- . . . and similar facts for triconnected component nodes

Auxiliary data for 2-connected components

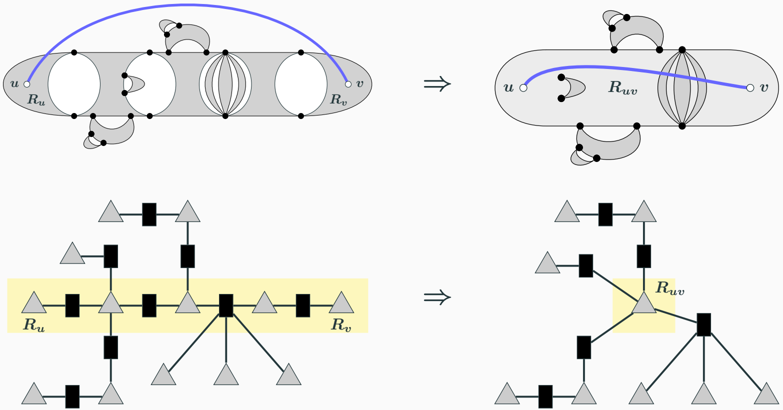
- To maintain: Given two triconnected component trees, are the described graphs isomorphic?



- We maintain implications of the following form as auxiliary data
  - If the graph of the subtree of (y_1, y_2) is isomorphic to the graph of the subtree of (y_1^*, y_2^*)
  - then so are the graphs of the subtree of (x_1, x_2) and of the subtree of (x_1^*, x_2^*)
- . . . and similar facts for triconnected component nodes
- . . . and some more technical auxiliary data

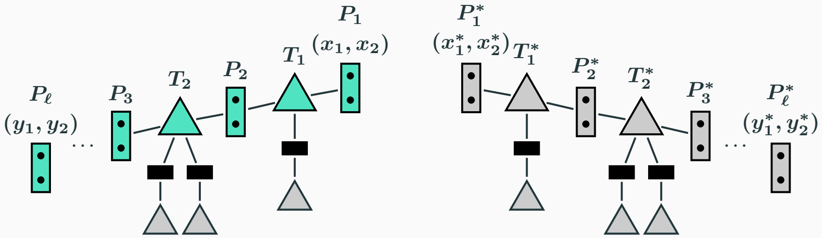
Recall: Effects of edge changes on the component trees

- Triconnected component trees may change drastically after an edge change



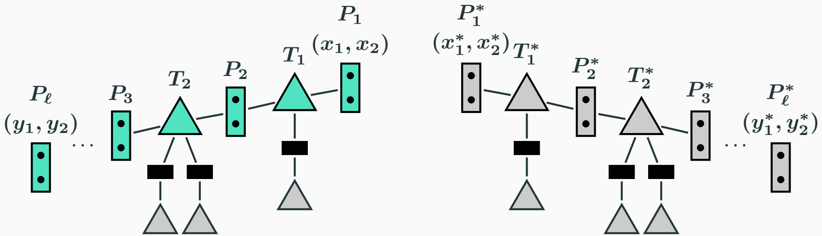
Maintenance of the auxiliary data: a central case

- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



Maintenance of the auxiliary data: a central case

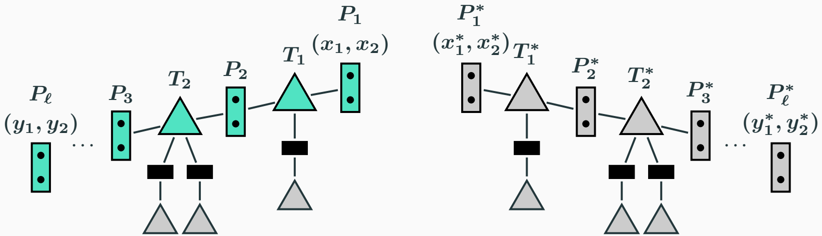
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path

Maintenance of the auxiliary data: a central case

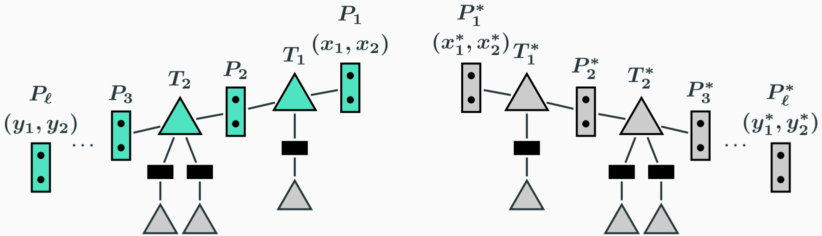
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph

Maintenance of the auxiliary data: a central case

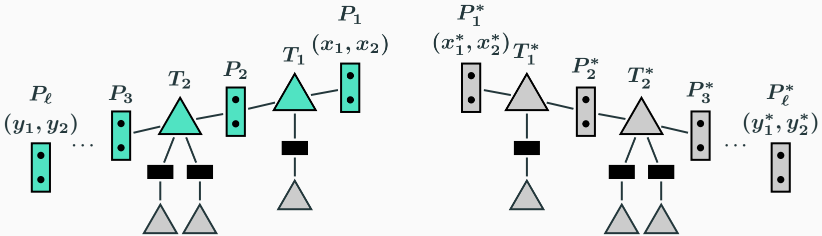
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices

Maintenance of the auxiliary data: a central case

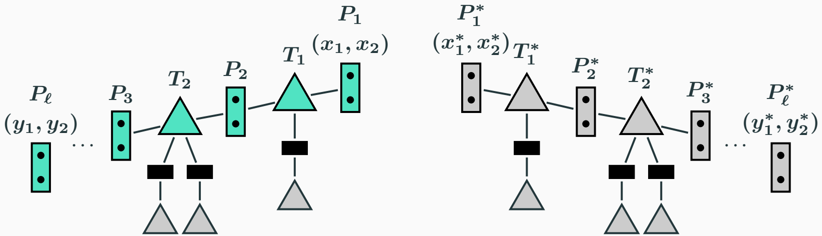
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices
  - for every separating pair of T_i that is not on the unfurled path

Maintenance of the auxiliary data: a central case

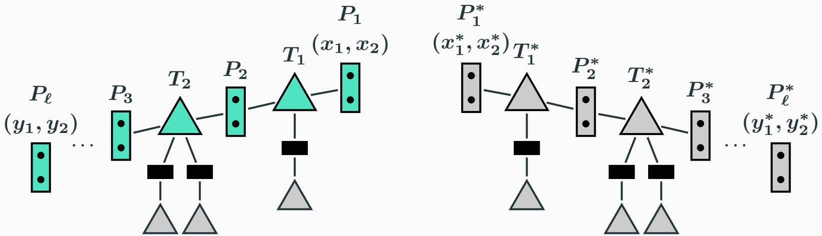
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices
  - for every separating pair of T_i that is not on the unfurled path
    - use the isomorphism from T_i to T_i^* to identify the corresponding separating pair of T_i^*

Maintenance of the auxiliary data: a central case

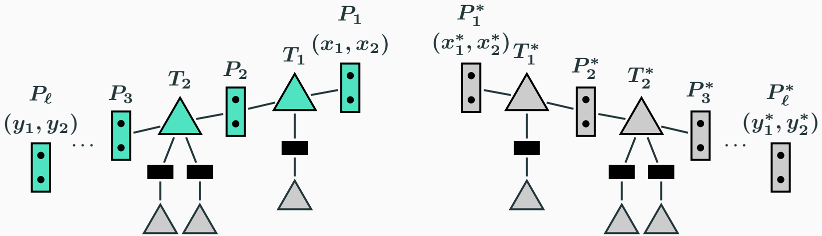
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices
  - for every separating pair of T_i that is not on the unfurled path
    - use the isomorphism from T_i to T_i^* to identify the corresponding separating pair of T_i^*
    - check in auxiliary data whether the subtrees of the separating pairs are isomorphic

Maintenance of the auxiliary data: a central case

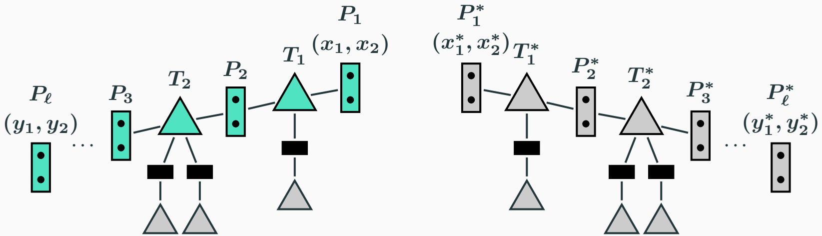
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices
  - for every separating pair of T_i that is not on the unfurled path
    - use the isomorphism from T_i to T_i^* to identify the corresponding separating pair of T_i^*
    - check in auxiliary data whether the subtrees of the separating pairs are isomorphic
- Additional data that we need for this:

Maintenance of the auxiliary data: a central case

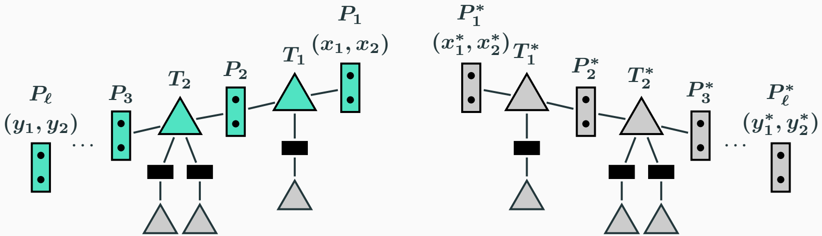
- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices
  - for every separating pair of T_i that is not on the unfurled path
    - use the isomorphism from T_i to T_i^* to identify the corresponding separating pair of T_i^*
    - check in auxiliary data whether the subtrees of the separating pairs are isomorphic
- Additional data that we need for this:
  - distances on the path

Maintenance of the auxiliary data: a central case

- Checks for updating the auxiliary data when a $\mathbf{3}$-connected component collapses:



- For every consecutive pair of separating pairs P_i, P_{i+1} on the unfurled path
  - existentially quantify the corresponding pairs P_i^*, P_{i+1}^* in the other graph
  - check that T_i, T_i^* are isomorphic when pinning the separating pair vertices
  - for every separating pair of T_i that is not on the unfurled path
    - use the isomorphism from T_i to T_i^* to identify the corresponding separating pair of T_i^*
    - check in auxiliary data whether the subtrees of the separating pairs are isomorphic
- Additional data that we need for this:
  - distances on the path
  - embedding information to consistently map vertices from P_i to P_i^*

Conclusion

Conclusion

- Dynamic planar graph isomorphism is in **DynFO**
  - Whether two planar graphs are isomorphic can be maintained using first-order update formulas
  - rephrased: . . . using a parallel constant-time algorithm with polynomial auxiliary data

Conclusion

- Dynamic planar graph isomorphism is in **DynFO**
  - Whether two planar graphs are isomorphic can be maintained using first-order update formulas
  - rephrased: …using a parallel constant-time algorithm with polynomial auxiliary data
- We only had a brief glimpse of the proof
  - just ideas and special cases
  - not mentioned at all: from isomorphism of biconnected components to isomorphism of connected components

Conclusion

- Dynamic planar graph isomorphism is in **DynFO**
  - Whether two planar graphs are isomorphic can be maintained using first-order update formulas
  - rephrased: …using a parallel constant-time algorithm with polynomial auxiliary data
- We only had a brief glimpse of the proof
  - just ideas and special cases
  - not mentioned at all: from isomorphism of biconnected components to isomorphism of connected components
- So far: we can solve the decision problem
  - Next step: maintaining a witnessing isomorphism

Datta, S., Khan, A., and Mukherjee, A. (2023).
Dynamic Planar Embedding Is in DynFO.
In Leroux, J., Lombardy, S., and Peleg, D., editors, *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:15, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Datta, S., Kulkarni, R., Mukherjee, A., Schwentick, T., and Zeume, T. (2015).
Reachability is in DynFO.
In Halldórsson, M. M., Iwama, K., Kobayashi, N., and Speckmann, B., editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part*

II, volume 9135 of *Lecture Notes in Computer Science*, pages 159–170. Springer.

Datta, S., Limaye, N., Nimbhorkar, P., Thierauf, T., and Wagner, F. (2022).
Planar graph isomorphism is in log-space.
ACM Trans. Comput. Theory, 14(2):8:1–8:33.

Datta, S., Mukherjee, A., Schwentick, T., Vortmeier, N., and Zeume, T. (2019).
A Strategy for Dynamic Programs: Start over and Muddle through.
Logical Methods in Computer Science, Volume 15, Issue 2.

Datta, S., Mukherjee, A., Vortmeier, N., and Zeume, T. (2018).
Reachability and distances under multiple changes.
In Chatzigiannakis, I., Kaklamanis, C., Marx, D., and Sannella, D., editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 120:1–120:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Dong, G. and Su, J. (1995).
Incremental and decremental evaluation of transitive closure by first-order queries.
Inf. Comput., 120(1):101–106.

References iv

Dong, G. and Su, J. (1998).
Arity bounds in first-order incremental evaluation and definition of polynomial time database queries.
J. Comput. Syst. Sci., 57(3):289–308.

Etessami, K. (1998).
Dynamic tree isomorphism via first-order updates.
In Mendelzon, A. O. and Paredaens, J., editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, pages 235–243. ACM Press.

Grädel, E. and Siebertz, S. (2012).
Dynamic definability.
In Deutsch, A., editor, *15th International Conference on Database Theory, ICDT '12, Berlin, Germany, March 26-29, 2012*, pages 236–248. ACM.

Henderson, H. and Searle, S. (1981).
On deriving the inverse of a sum of matrices.
SIAM Review, 23(1):53–60.

Hopcroft, J. E. and Wong, J. K. (1974).
Linear time algorithm for isomorphism of planar graphs (preliminary report).
In Constable, R. L., Ritchie, R. W., Carlyle, J. W., and Harrison, M. A., editors, *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1974, Seattle, Washington, USA*, pages 172–184. ACM.

Patnaik, S. and Immerman, N. (1997).
Dyn-FO: A parallel, dynamic complexity class.
J. Comput. Syst. Sci., 55(2):199–209.

Tutte, W. T. (1963).
How to draw a graph.
Proceedings of the London Mathematical Society, s3-13(1):743–767.