Incremental query evaluation

Thomas Schwentick

Luminy, April 2019

technische universität dortmund

Lehrstuhl Logik in der Informatik

Dynamic Complexity: Recent and Complex Updates

Thomas Schwentick

(with some borrowed slides from Nils Vortmeier and Thomas Zeume)

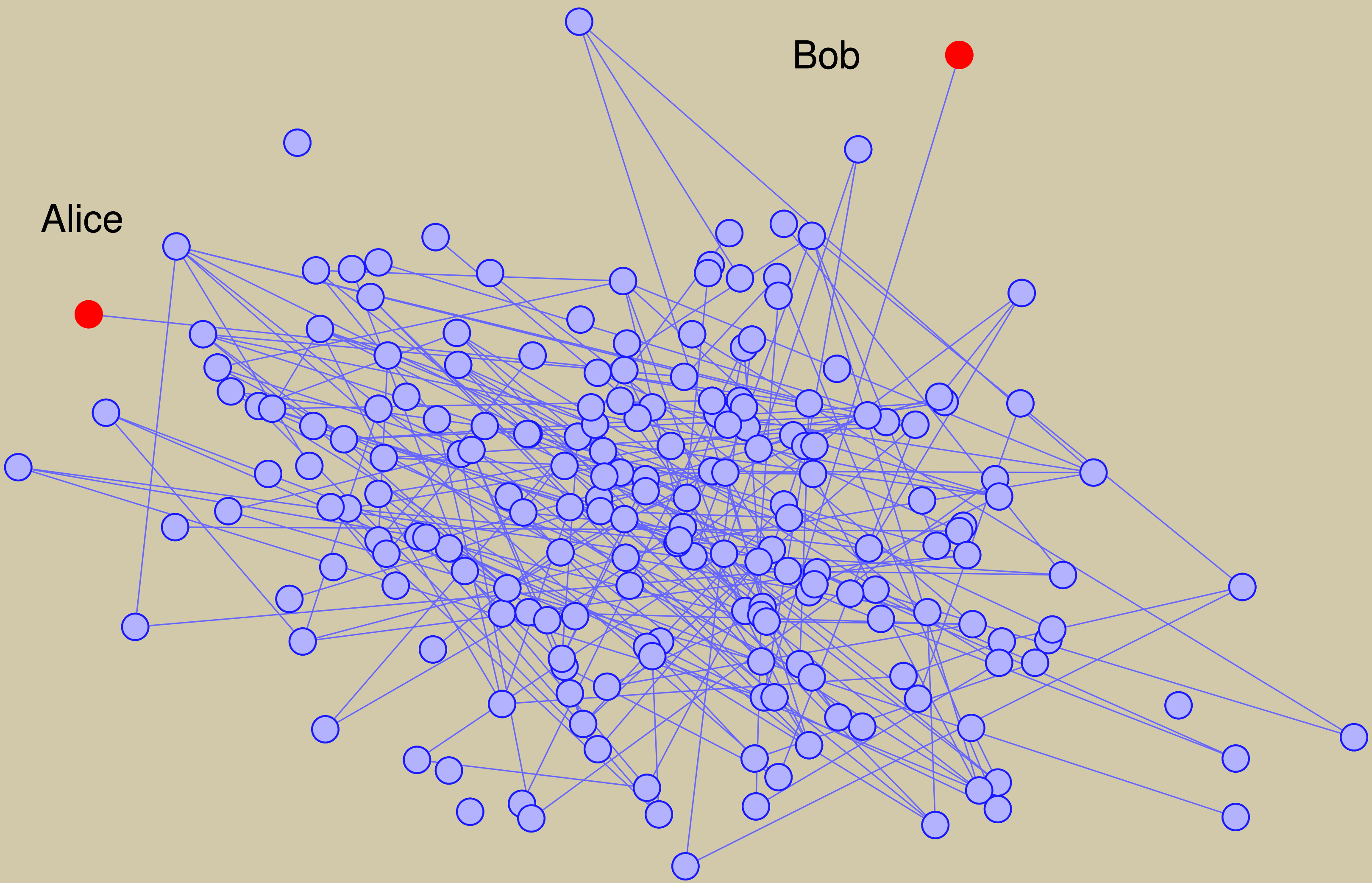
Luminy, April 2019

technische universität dortmund

Lehrstuhl Logik in der Informatik

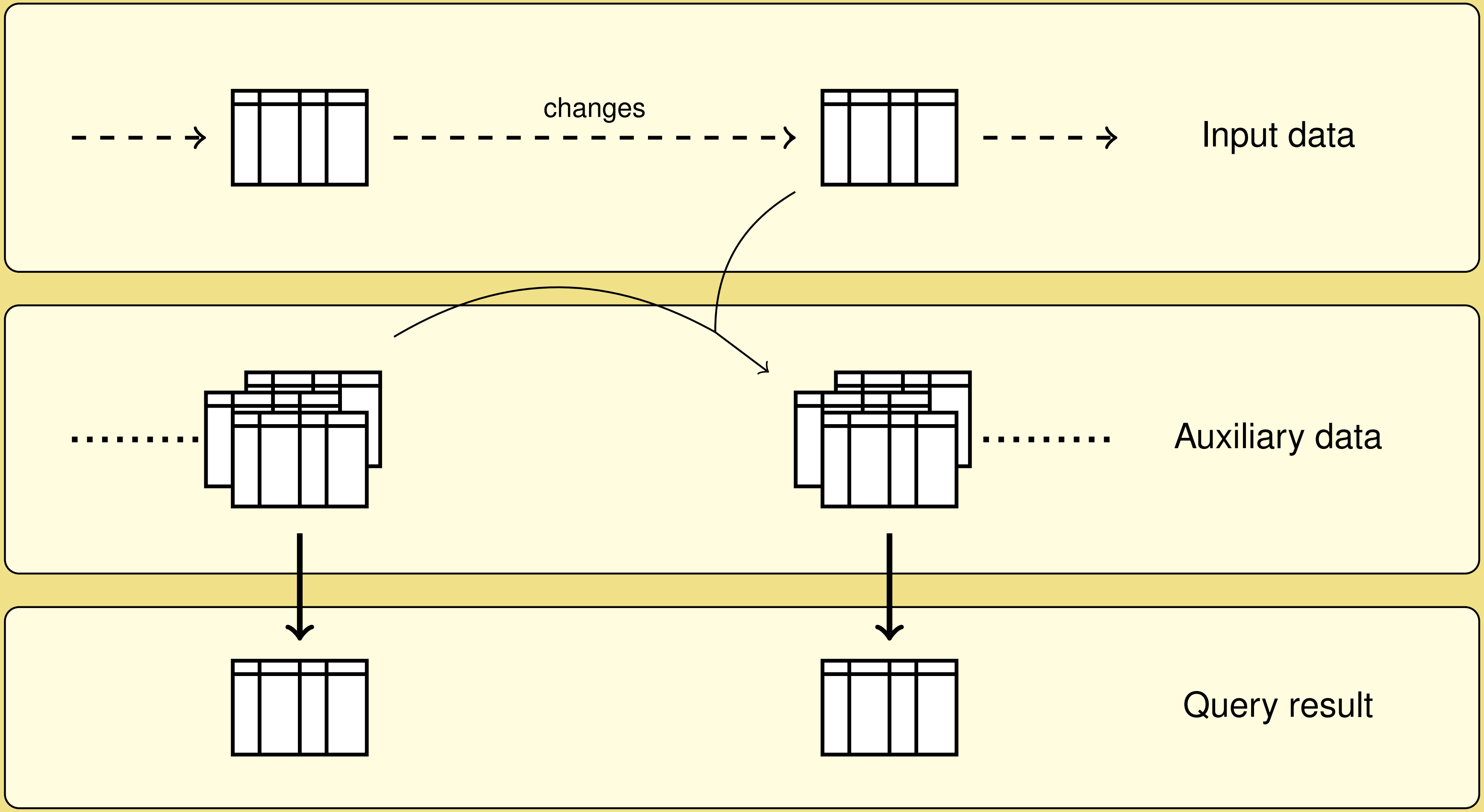
Dynamic Reachability in Practice: Social Networks

Example



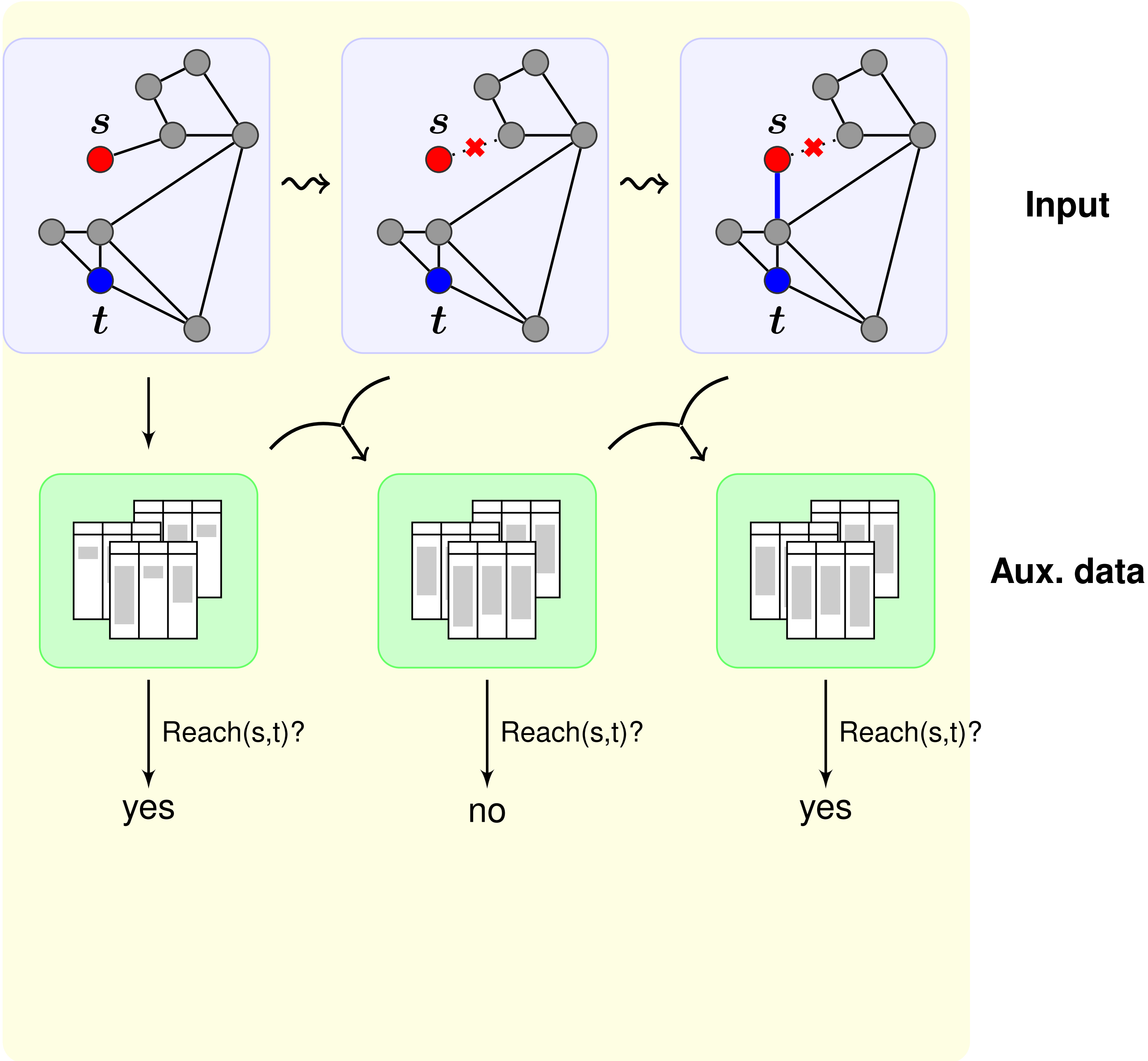
The Dynamic Setting

Dynamic Evaluation of a query



- **DynFO**: Auxiliary relations are updated using first-order logic

The Dynamic Setting: Reachability



The Dynamic Setting: Algorithms vs. Query Languages

- The field of *Dynamic Algorithms* studies algorithms that *maintain* a (graph) property faster than deciding it from scratch

- In *Dynamic Complexity* we think in terms of query languages

- In Databases (Theory) our main language is
  Relational Algebra $\equiv$ *Relational Calculus* $\approx$ *First-Order Logic*

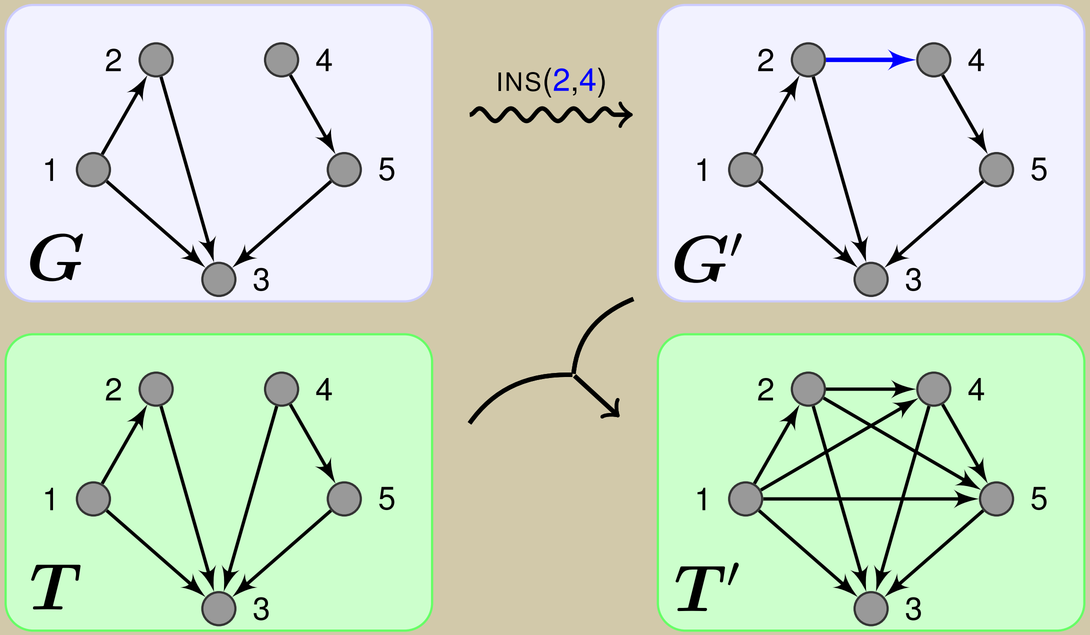
- *Incremental View Maintenance:* Update FO-query results as efficiently as possible

- Different angle: can the result of a query q that is expressible in a stronger query language $\mathcal{L}_1$ be updated with a weaker query language $\mathcal{L}_2$

- Two natural questions:
  (1) What expressive power is needed to *maintain* FO-expressible queries? ☞ later
  (2) Which queries that can not be expressed in FO, can be updated with FO? ☞ Main topic of this talk

Example 1: Reachability under Insertions

Example

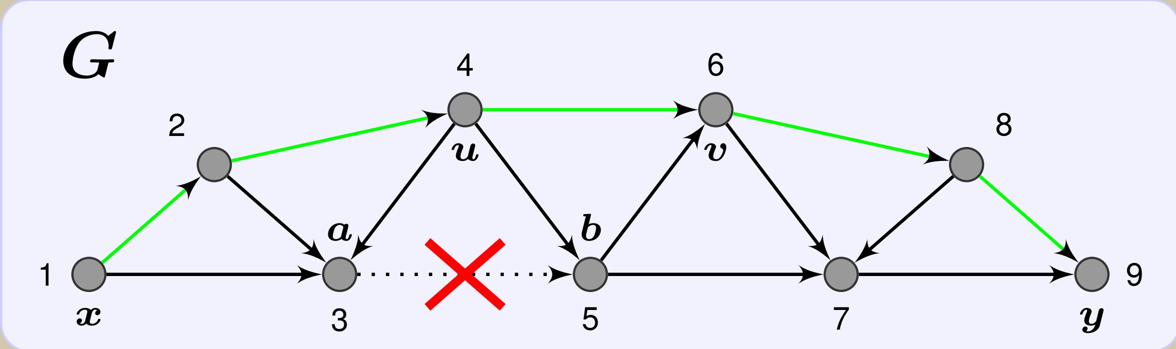


- **Idea:** store the transitive closure of the edge relation in a binary auxiliary relation T ☞[Dong, Su 93/95; Patnaik, Immerman 94/97]
- Update rule:
  on insert (u, v) **into** E
  update $T(x, y)$ **as** $T(x, y) \vee (T(x, u) \wedge T(v, y))$
  - determines the pairs (x, y) in T after insertion of (u, v) to E

- Transitive closure does not suffice for edge *deletions* [Dong, Libkin, Wong 95]

Example 2: Reachability in DAGs under Deletions

Example



- For *directed acyclic* graphs, Reachability can be maintained with first-order updates [Dong, Su 93/95; Patnaik, Immerman 94/97]
- Challenge: how to express, that there is still a path p from x to y after deleting edge (a, b)?

  Simple cases $E(x, y)$, $\neg T(x, a)$, $\neg T(b, y)$, $\ldots$

  Otherwise p must have a last node $u \neq y$ from which a can be reached

  $$\cdots \vee \exists u, v\big((u \neq a \vee v \neq y) \wedge T(x, u) \wedge E(u, v) \wedge T(v, y) \wedge T(u, a) \wedge \neg T(v, a)$$

Dynamic Complexity: Our Setting

- Databases in this talk: graphs (directed/undirected, possibly labelled)

- Change operations:
  - Simple changes:
    - Insertion of a single tuple: **insert** (u, v)
    - Deletion of a single tuple: **delete** (u, v)
  - Complex changes: later

- Set of nodes is fixed, for each computation
  - n = number of nodes

- Dynamic program:
  - One update formula per change operation and auxiliary relation
  - One output formula

- Initialisation:
  - Source of technical complications
  - We ignore it for this talk
  - We can always assume the nodes are numbers $1, \ldots, n$ and formulas can use a linear order $\leqslant$ on the nodes and addition and multiplication relations

Definition

- **DynFO** $\stackrel{\text{def}}{=}$ queries that can be maintained by first-order logic with auxiliary relations under the given change operations

Motivation and Goals

- **Why DynFO?**

- captures essentially what can be maintained in a relational database ☞ core SQL
- meaningful from a complexity theoretic point of view:
  - **FO**$(+,\times)$ $\equiv$ uniform **AC**^{0} $\equiv$ circuit families of bounded depth and poly size
- the most natural logic

- **General goals of our research:**

- Understand the expressive power of **DynFO**
  - Which queries are in **DynFO**?
    - General techniques for **DynFO** programs
  - Which queries are **not** in **DynFO**?
    - Methods for inexpressibility results?

- **What we learned:**

- In the dynamic setting, first-order logic is much more powerful than in the static setting
- Inexpressibility results are hard to get

Short History of Dynamic Complexity

2019

2015

2010

2005

2000

1995

1990

Complex Changes, Muddling

Reach in non-uniform **DynAC^0**$[\oplus]$

Reach in **DynFO**

Connections to static complexity classes

Lower bounds for small syntactic fragments

Reach in dynamic **TC^0**

Maintaining basic graph queries

Lower bounds for restricted Arity

Introduction of the setting

WWW N. Vortmeier

?

WWW A. Mukherjee

WWW E. Grädel

WWW S. Siebertz

WWW S. Datta

WWW R. Kulkarni

WWW W. Gelade

WWW M. Marquardt

WWW V. Weber

WWW T. Zeume

WWW L. Libkin

WWW L. Wong

WWW W. Hesse

WWW K. Etessami

WWW S. Patnaik

WWW N. Immerman

WWW J. Su

WWW R. Topor

WWW G. Dong

Contents

Methods for dynamic programs

- We will see various methods for upper bounds

- First we consider “traditional” methods
  - Ad-hoc programs for the problem at hand
  - Reductions

- Then we will have a look at more recent techniques
  - MSO-simulation
  - Muddling
  - Linear Algebra

Undirected Reachability in DynFO (1/3)

- We already know:

Theorem [Patnaik, Immerman 94/97]

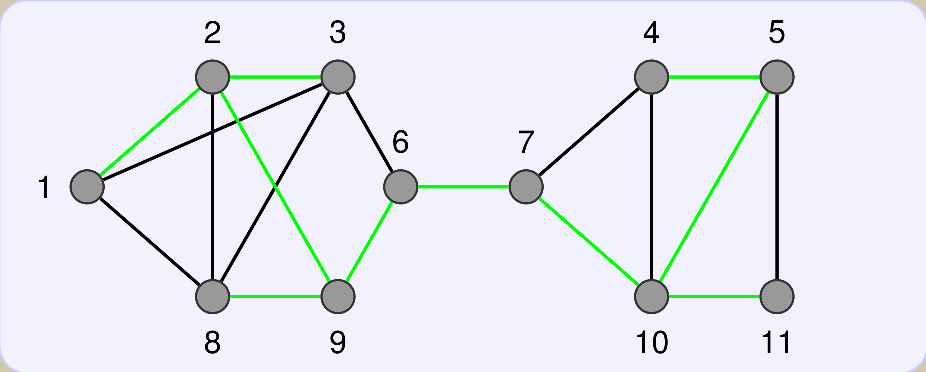
- ACYCLIC REACH ∈ **DynFO**

- As another restriction of REACH, we now consider SYM-REACH:
  - Reachability for undirected graphs
- There are several proofs for SYM-REACH ∈ **DynFO**
  - We look at the simplest and first proof by

[Patnaik, Immerman 94/97]

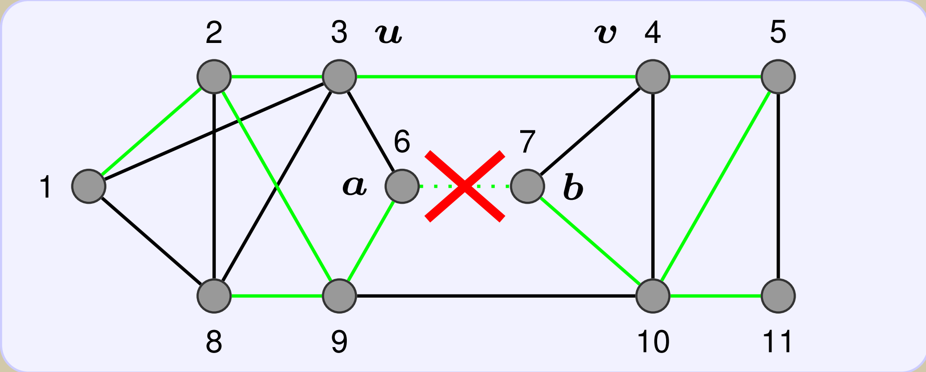
Undirected Reachability in DynFO (2/3)

Example: Insertion



- Basic idea: maintain a spanning forest F and its transitive closure T
- On arrival of a new edge, add it to F, if it connects two distinct components

Example: Deletion



- Deletion is, again, more tricky
- How to modify the spanning tree if an edge (a, b) is deleted but its component remains connected?
  - Determine nodes u and v in the subtrees of a and b, respectively, such that $(u, v) \in E$, and add (u, v) to F
- This can be done with
  - a more sophisticated relation T with all triples (d, e, g) for which there is a path in F from d to e through g
  - some order on the edges to choose (u, v) *uniquely*

Undirected Reachability in DynFO (3/3)

Theorem [Patnaik, Immerman 94/97]

- SYM-REACH $\in$ **DynFO**

- Is the ternary auxiliary relation T necessary? ☞ No

- <u>k-ary DynFO:</u> queries in **DynFO** that can be maintained with (at most) k-ary aux relations

Theorem [Dong, Su 95/98]

- SYM-REACH $\in$ binary **DynFO**
- SYM-REACH $\notin$ unary **DynFO**

Undirected Reachability under Complex Changes

- So far we only considered very simple change operations:
  - Insertion or deletion of a single tuple
  - No change of the universe/domain

- What about other kinds of changes?

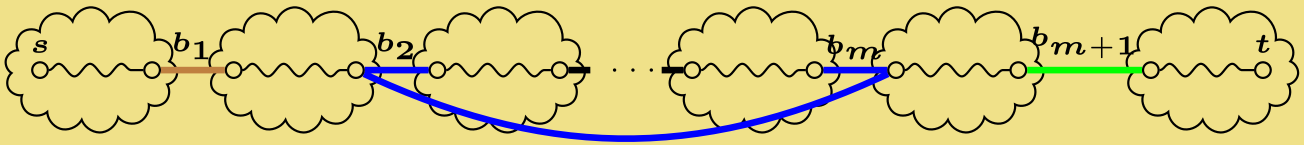
- "Arbitrary Changes"?
  - If the database can change arbitrarily in one step, only **FO**-properties can be maintained in **DynFO**

- What about complex changes that are *defined* by formulas $\psi(\vec{y})$? ☞[Patnaik, Immerman 94]
- ... aka core SQL updates

Theorem [S., Vortmeier, Zeume 17]

- Reachability is in **DynFO** for undirected graphs in the presence of
  - single-tuple insertions and deletions and
  - **FO**-defined insertions

- Technique relies on a "bridge bound"



- In a nutshell each long path between connected components has a shortcut

- For insertions defined by unions of conjunctive queries (UCQs), the number of bridges is small

→ Prototypical implementation works quite well

Reductions

- In Complexity, reductions are mostly used for lower bound results
- But of course, they can also yield upper bounds

- If for problems A, B and class $\mathcal{C}$ it holds
  - $A \leqslant B$,
  - $B \in \mathcal{C}$ and
  - $\mathcal{C}$ is closed under $\leqslant$

  then $A \in \mathcal{C}$

→ Under which reductions is **DynFO** closed?

- Two requirements for such reductions:
  - FO-expressible
  - One change with respect to A should yield only few changes with respect to B

- bfo-reductions ($\leqslant_{\text{bfo}}$):
  FO-definable and one change wrt A yields only $\mathcal{O}(1)$ changes wrt B

Regular Path Queries and Reachability

- Let R be a regular language over Σ
- The regular path query q_R over graph databases asks for all pairs (u, v), for which there is a path from u to v with label sequence in R

- Let $G = (V, E)$ with edge labels from alphabet Σ
- Let $\mathcal{A}$ be a NFA for R with unique initial and final states s and t

- Let the product graph $G \times \mathcal{A}$ have
  - node set $V \times Q$,
  - edge $(i, p) \to (j, q)$ if $i \xrightarrow{\sigma} j$ and $p \xrightarrow{\sigma} q$, for some $\sigma \in \Sigma$

- There is an R-path in G from u to v if and only if (v, t) is reachable from (u, s) in $G \times \mathcal{A}$
- And every change in G yields $\leqslant |Q|$ changes in $G \times \mathcal{A}$

→ If Reachability is in **DynFO**, then Regular Path Queries are in **DynFO** as well

Contents

Reachability under $\log n$ insertions (1/2)

Theorem [Vortmeier, Zeume 19 (unpublished)]

- Reachability is in **DynFO*** under $\log n$ insertions

  $*$: If formulas can use $+$ and $\times$

Proof idea

- The basic idea is
  (1) to compute Reachability for the $\log n$ affected nodes, and
  (2) to combine this information with the Reachability information for the rest of the graph

Proof idea (cont.)

- How can (1) be done?

- Reachability between two nodes x, y can be expressed by a monadic second-order (MSO) formula:

$$\forall X \big(X(x) \wedge \forall v \forall w (X(v) \wedge E(v, w) \rightarrow X(w)) \rightarrow X(y)\big)$$

- Quantification of X is a-priori restricted to the subset W of affected nodes of size $\log n$

➡ The second-order $\exists X$ quantification ☞ restricted to W! can be replaced by a first-order quantification $\exists x$ ☞ over all nodes!

  - Since one node of the graph carries $\log n$ bits of information
  - And this information can be decoded with the help of $+$ and $\times$

Reachability under $\log n$ insertions (1/2)

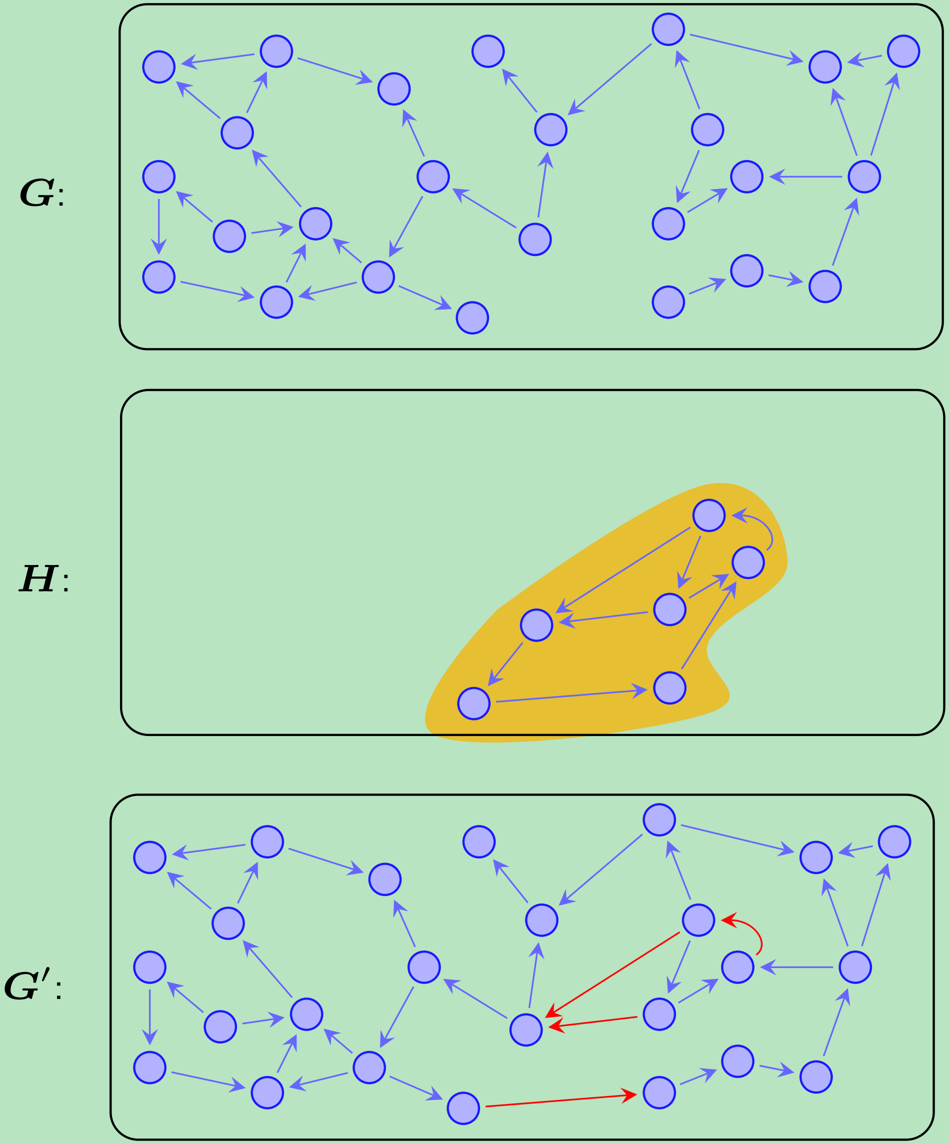
Proof idea (cont.)

- Assume that the transitive closure T_G of graph G is given

- After insertion of $\log n$ nodes
- ... let the graph H be defined on the effected nodes in the resulting graph G' ...
- ... with the newly inserted edges ...
- ... and additional edges for paths in G

- Compute transitive closure T_H of H
- Combine T_H with T_G to get $T_{G'}$

Proof idea (cont.)



Contents

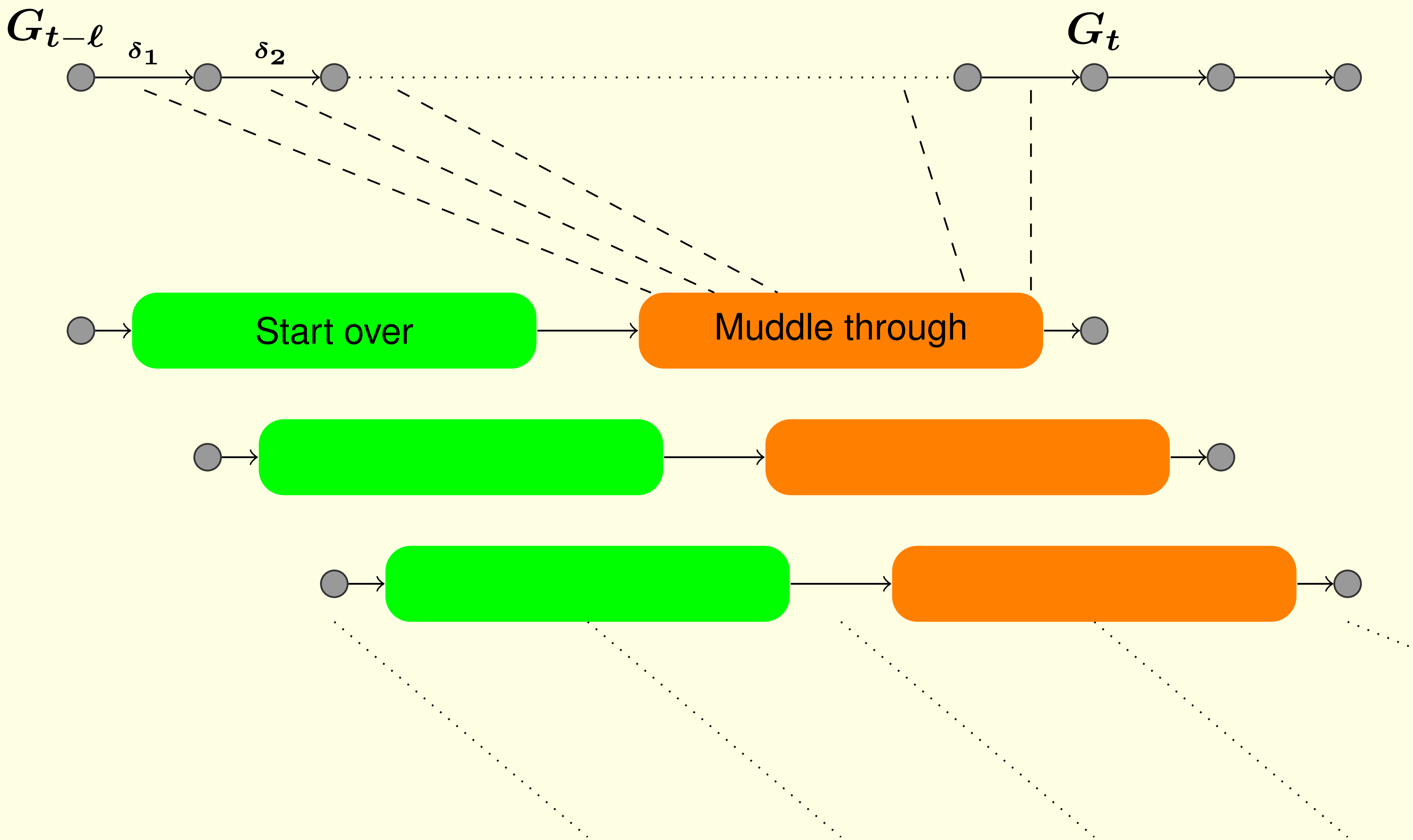
Muddling

- Basic idea of muddling:
  - To give the correct answer for graph G_t (at time t) do the following:

  (1) Compute a solution from scratch for $G_{t-\ell}$ for a suitable ℓ ☞ Start over

  (2) Update the computed solution for the ℓ changes between $t - \ell$ and t with constant speed-up ☞ Muddle through

Muddling: basic idea



Muddling Lemma

Muddling Lemma [Datta, Mukherjee, S., Vortmeier, Zeume 17]

- A query Q is in **DynFO**, if it has the following property:
  - From a graph G of size n
  - ... one can compute auxiliary relations in **AC**^{1} ...
  - ... with which the query can be maintained for $\log n$ change steps

- **AC**^{1} is a complexity class based on circuits of logarithmic depth
- **LOGSPACE** $\subseteq$ **NL** $\subseteq$ **AC**^{1}

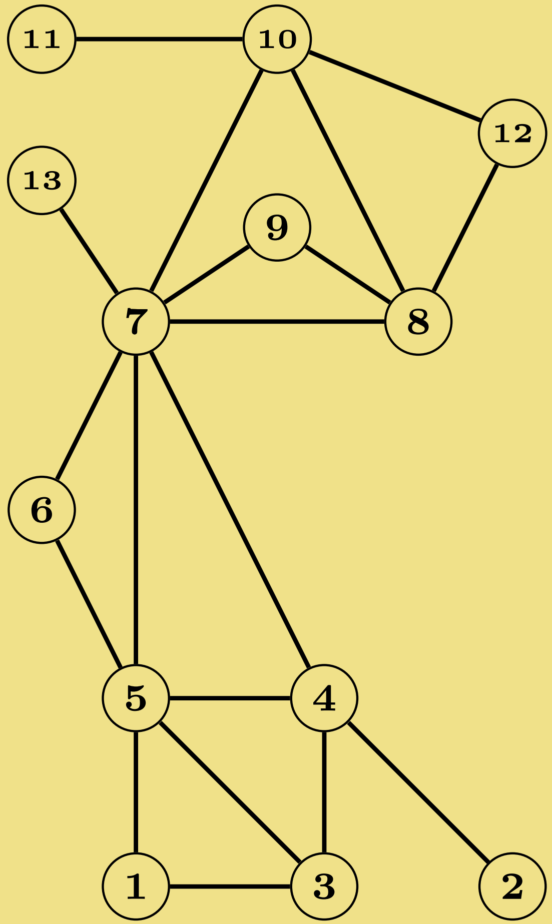
- **AC**^{1} can be characterised in terms of a limited fixed-point process:

  ☞ Immerman

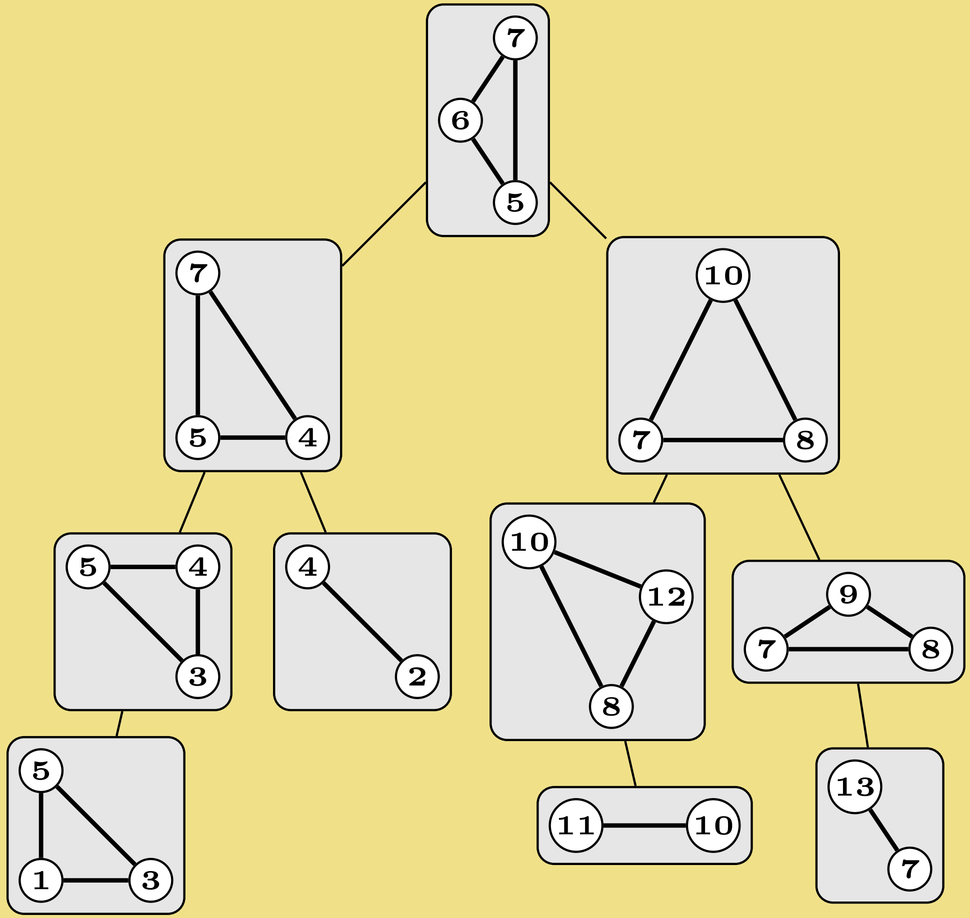
  - **AC**^{1} $=$ **IND**$(\log n)$,
    i.e., all queries that can be evaluated by $\mathcal{O}(\log n)$ many applications of the same **FO**-formula
  - Example: $\log n+1$ applications of
    $$\varphi(x,y) = E(x,y) \vee \exists z T(x,z) \wedge T(z,y)$$
    yield the transitive closure of a graph

Tree decompositions

An input graph ...



... and its tree decomposition



Application 1: 3-COL on bounded tree-width Graphs

Theorem [Datta, Mukherjee, S., Vortmeier, Zeume 17]

- 3-COL can be maintained in **DynFO** on graphs of bounded tree-width

- Tree decompositions can be computed in logarithmic space

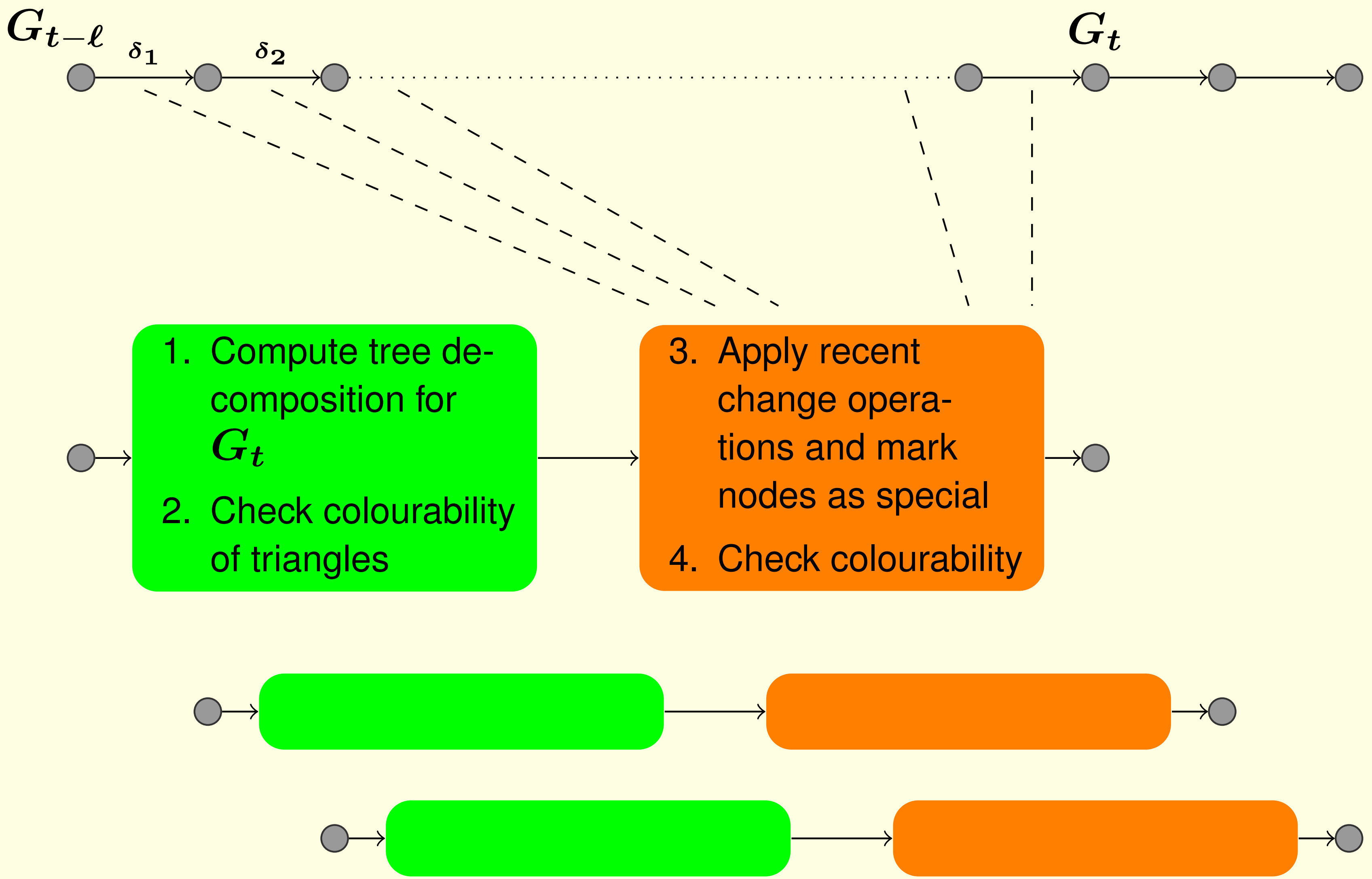
  [Elberfeld, Jakoby, Tantau 10]

- ...thus in **AC**^{1}
- ...thus in **IND**$(\log n)$

- Challenge: A small change of the graph might induce a big change of the tree decomposition

- Approach: use slightly outdated tree decomposition and muddle through for $\mathcal{O}(\log n)$ many "special" nodes

Application 1: Illustration



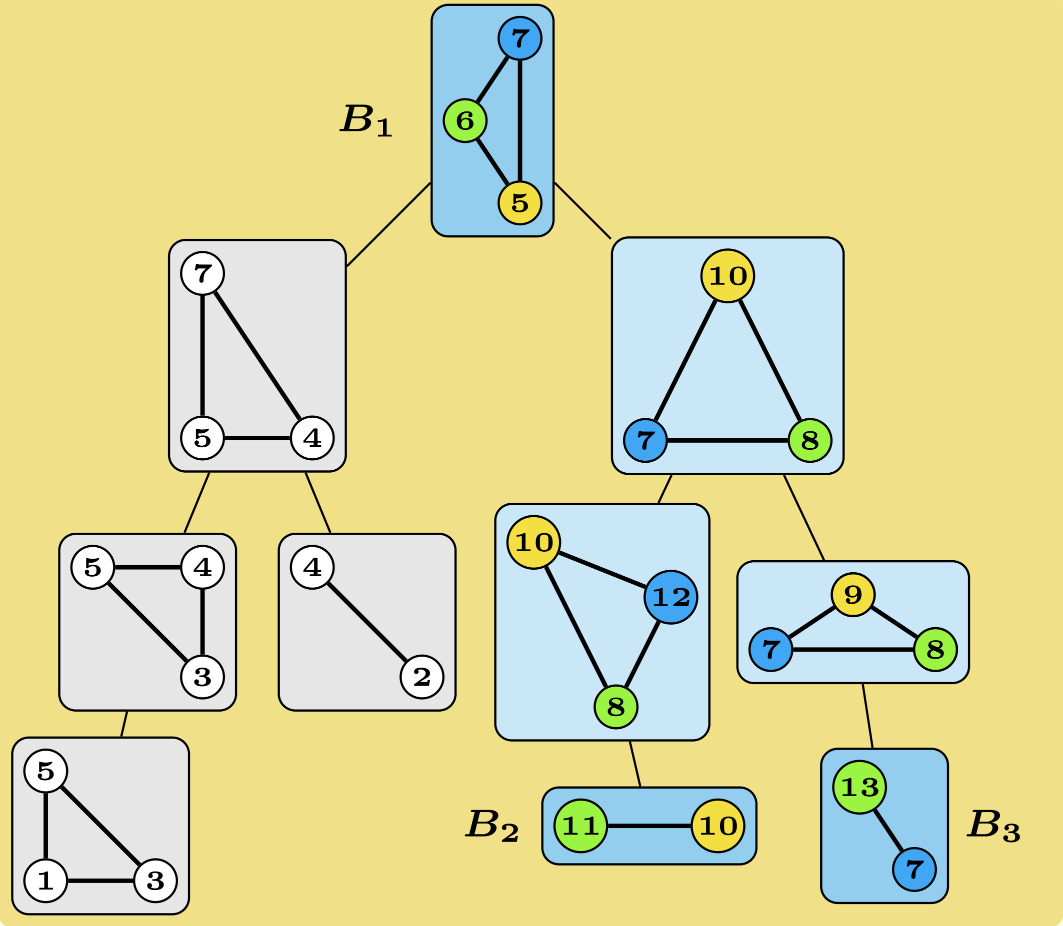
- Phase 1&2: $\frac{1}{2}\log n$ steps
- Phase 3: $\frac{1}{2}\log n$ steps
- Phase 4: 1 step

☞ $\ell = 1 + \log n$

Application 1: More detail (1/2)

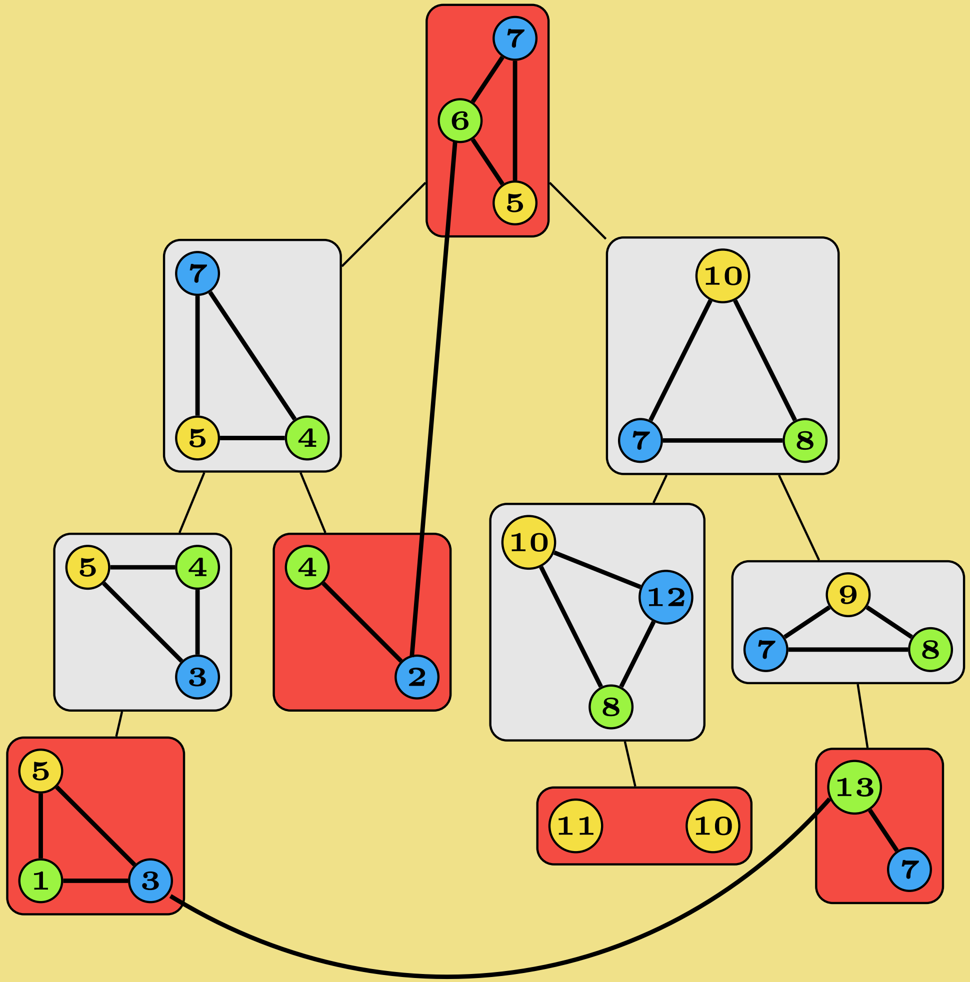
- Compute colourability information for all *triangles* of the decomposition
- *Triangle*: Three bags B_1, B_2, B_3
  - B_2 is in the subtree of B_1
  - B_3 is in the subtree of B_1
  - B_2 is no predecessor or descendant of B_3
- *Boundary:* All nodes in B_1, B_2, B_3

- Which colourings of boundaries of triangles can be extended to valid 3-colourings of the inner part of the induced graph? ☞ slightly simplified



Application 1: More detail (2/2)

- If v is affected by a change: declare one bag of v as **special** ☞ special nodes
- After $\log n$ changes: $\mathcal{O}(\log n)$ nodes are special
- Existentially quantify colouring C of special nodes
  → **MSO** on subgraph with $\mathcal{O}(\log n)$ nodes, again
- Check: C is a valid 3-colouring of the graph induced by the special nodes
- Use auxiliary relations to check that C can be extended for the whole graph



Theorem [Datta, Mukherjee, S., Vortmeier, Zeume 17]

- Every MSO-definable query can be maintained in **DynFO** on graphs of bounded tree-width

Application 2: Parameter-free definable changes

- Setting:
  - Fixed, finite set Δ of possible first-order definable change operations
    - No parameters

- Examples for parameter-free changes:
  - Delete all edges between blue and red nodes
  - Insert an edge between each green node x and yellow node y if they have a joint neigbour

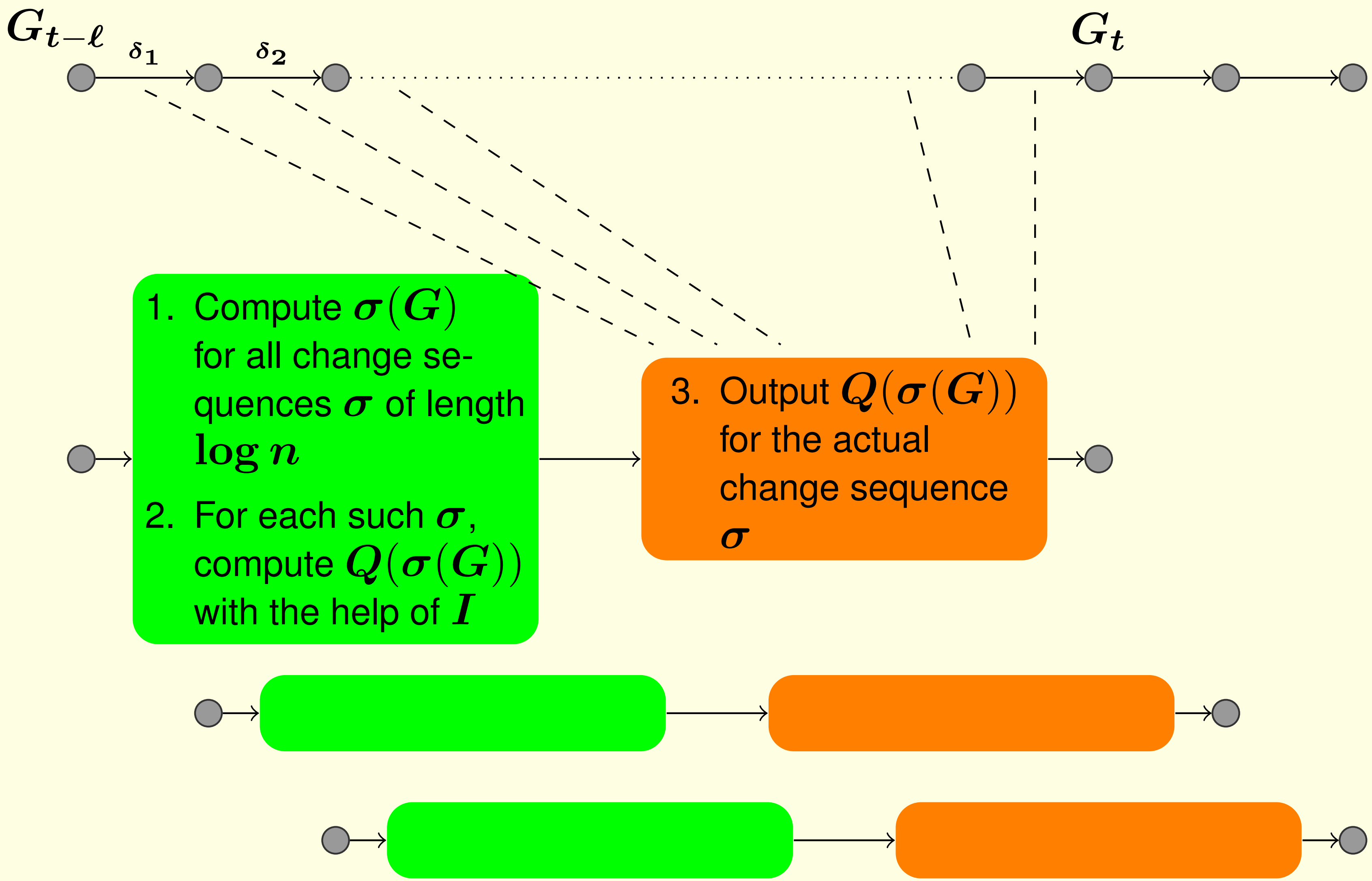
Theorem [Schwentick, Vortmeier, Zeume 17]

- In this setting, every **AC**^{1}-definable query* can be maintained in **DynFO**

- $*$: with suitable initialisation

- For a proof sketch, let Q be some **AC**^{1}-definable query
- Let I be a **IND**$(\log n)$-program for Q

Application 2: Illustration



- Phase 1: $\frac{1}{2}\log n$ steps
- Phase 2: $\frac{1}{2}\log n$ steps
- Phase 3: 1 step

☞ $\ell = 1 + \log n$

Contents

Results about Reachability

Conjecture [Patnaik, Immerman 97]

- Reachability is in **DynFO**

- **Reachability is in DynFO for ...**
  - acyclic graphs [Patnaik, Immerman 94/97]
  - undirected graphs
    [Patnaik, Immerman 94/97; Dong, Su 98, Grädel, Siebertz 12]
  - embedded planar graphs [Datta, Hesse, Kulkarni 14]

- **Reachability is in DynFO extended by ...**
  - counting quantifiers [Hesse 01]
  - modulo-2 counting quantifiers [Datta, Hesse, Kulkarni 14]

Theorem [Datta, Kulkarni, Mukherjee, TS, Zeume 15]

- Reachability is in **DynFO**

Theorem [Datta, Kulkarni, Mukherjee, Zeume 18]

- Reachability is in **DynFO**, even under under $\frac{\log n}{\log \log n}$ changes
  ☞ insertions and deletions!

Reachability in DynFO: Outline

Definition: REACH

Input: Directed Graph G

Result: All pairs (s, t) for which there is a path from s to t in G

Definition: FULLRANK

Input: $m \times m$-matrix A with values from $\{0, \ldots, m\}$

Question: Does A have full rank m?

Definition: FULLRANKMODP

Input: $m \times m$-matrix A with values from $\{0, \ldots, m\}$, prime $p \leqslant m^2$

Question: Does A have full rank m over $\mathbb{Z}_p$?

Structure of the proof

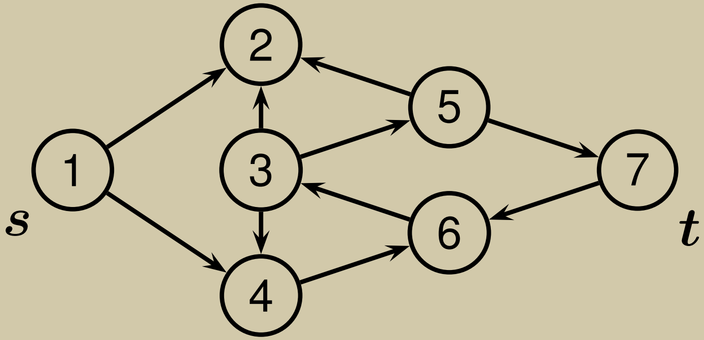
- We show:
  (1) REACH $\leqslant_{\text{btt}[+,\times]}$ FULLRANK
  (2) FULLRANK $\leqslant_{\text{bfo-tt}}$ FULLRANKMODP
  (3) FULLRANKMODP $\in$ **DynFO**$(+, \times)$
  (4) For domain independent Q:
  $Q \in$ **DynFO**$(+, \times) \Rightarrow Q \in$ **DynFO**

- Further ingredients:
  - **DynFO** is closed under $\leqslant_{\text{bfo-tt}}$-reductions
  - **DynFO**$(+, \times)$ is closed under $\leqslant_{\text{btt}[+,\times]}$-reductions
  - REACH is domain independent

- All steps (1)-(4) are relatively simple and build on previous work

Step 1: REACH $\leqslant_{\mathbf{btt}[+,\times]}$ FULLRANK (1/4)

Example



$$A_G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Example

$$(A_G)^2 = \begin{pmatrix} 1 & 2 & 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 2 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 1 & 1 & 2 \\ 0 & 1 & 2 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 & 1 \end{pmatrix}$$

$$(A_G)^8 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 57 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Step 1: REACH $\leqslant_{\text{btt}[+,\times]}$ FULLRANK (2/4)

Proof

- Let G be a graph with n vertices and A_G its adjacency matrix
- $(A_G)^i[s,t] \neq 0 \iff$ there is a path of length $\leqslant i$ from s to t

- **Important observation:** ☞ e.g.: [Laubner 11]
  - $I - \frac{1}{n}A_G$ is invertible and
    $$(I - \frac{1}{n}A_G)^{-1} = I + \sum_{i=1}^{\infty}(\frac{1}{n}A_G)^i$$
  - ➨ the s,t-entry of this matrix is zero $\iff$ t is **not** reachable from s

- $B \stackrel{\text{def}}{=} nI - A_G$ ☞ integer matrix

- **The following are equivalent:**
  - t is **not** reachable from s
  - $B^{-1}[s,t] = 0$
  - $Bx = e_t$ has a solution with $x[s] = 0$

- where column vector $e_t \stackrel{\text{def}}{=} \begin{cases} 1 & \text{in row } t \\ 0 & \text{otherwise} \end{cases}$

Proof (cont.)

$$\underbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}}_{B} \times \underbrace{\begin{pmatrix} x[1] \\ x[s] \\ \cdot \\ \cdot \\ x[n] \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}}_{e_t}$$

has a solution with $x[s] = 0$

$$\iff$$

$$\underbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & B & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{B'} \times \underbrace{\begin{pmatrix} x[1] \\ x[s] \\ \cdot \\ \cdot \\ x[n] \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}}_{e'_t}$$

has any solution

Step 1: REACH $\leqslant_{\text{btt}[+,\times]}$ FULLRANK (3/4)

Proof

$$\overbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & B & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}}^{B'} \times \overbrace{\begin{pmatrix} x[1] \\ x[s] \\ \cdot \\ \cdot \\ x[n] \end{pmatrix}}^{x} = \overbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}}^{e'_t}$$

$$\overbrace{\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & B & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}^{B'|e'_t}$$

Proof (cont.)

- $B'x = e'_t$ has any solution
  $\iff e'_t \in \text{RowSpace}(B')$
  $\iff \text{rank}(B') = \text{rank}(B'|e'_t)$
- Since B is invertible,
  $\text{rank}(B') = \text{rank}(B) = n$
- **Putting everything together:**
  t is reachable from s
  $\iff B'x = e'_t$ has **no** solution
  $\iff \text{rank}(B'|e'_t) = n + 1$
  $\iff B'|e'_t$ has full rank
- **Crucial:**
  - One edge change in G only yields one change in $B'|e'_t$
    → $\leqslant_{\text{btt}[+,\times]}$-reduction
  - All numbers in $B'|e'_t$ are $\leqslant n$

Step 1: REACH $\leqslant_{\text{btt}[+,\times]}$ FULLRANK (4/4)

- What does REACH $\leqslant_{\text{btt}[+,\times]}$ FULLRANK exactly mean?

- "fo" says that all parts are first-order definable:
  - $A_{s,t}$ is first-order definable from $G, s, t, +, \times$
  - REACH(G) is first-order definable from the query results FULLRANK$(A_{s,t})$, for all s, t

- "b" stands for "bounded expansion":
  - Each single edge change in G affects only a (constantly) bounded number of entries in $A_{s,t}$

- "tt" stands for "truth-table reduction":
  - For each pair s, t of nodes of G, one instance $A_{s,t}$ of FULLRANK is constructed

- "$[+, \times]$" basically indicates that the the nodes of G are numbers $1, \ldots, n$ of G and reduction can use addition and multiplication

✎ In general, more parameters possible...

Step 2: FULLRANK $\leqslant_{\text{bfo-tt}}$ FULLRANKMODP

Proof (cont.)

- Challenge: for the next step, numbers in matrices can become exponentially large

  ☞ cannot be handled over domain $\{0, \dots, m\}$

- **Claim:** The following are equivalent:
  - An $m \times m$-matrix A with values from $\{0, \dots, m\}$ has full rank m
  - For some prime $p \leqslant m^2$, A has full rank m over $\mathbb{Z}_p$

Step 3: FULLRANKMODP $\in$ DynFO (1/2)

Proof (cont.)

- It remains to maintain rank$(\boldsymbol{A})$ over $\mathbb{Z}_{\boldsymbol{p}}$ for primes $\boldsymbol{p} \leqslant \boldsymbol{m}^2$

Proof (cont.)

- **Idea:** Maintain a Gaussian elimination, i.e.:
  - an invertible matrix $\boldsymbol{U}$ and
  - a matrix $\boldsymbol{E}$ in reduced row-echelon form

  such that $\boldsymbol{U}\boldsymbol{A} = \boldsymbol{E}$ [Frandsen, Frandsen 09]

- **Reduced row-echelon form:**
  - The first non-zero (= *leading*) entry in every row is 1
  - The column of such a leading entry is all-zero otherwise
  - Rows are sorted in a “diagonal” fashion

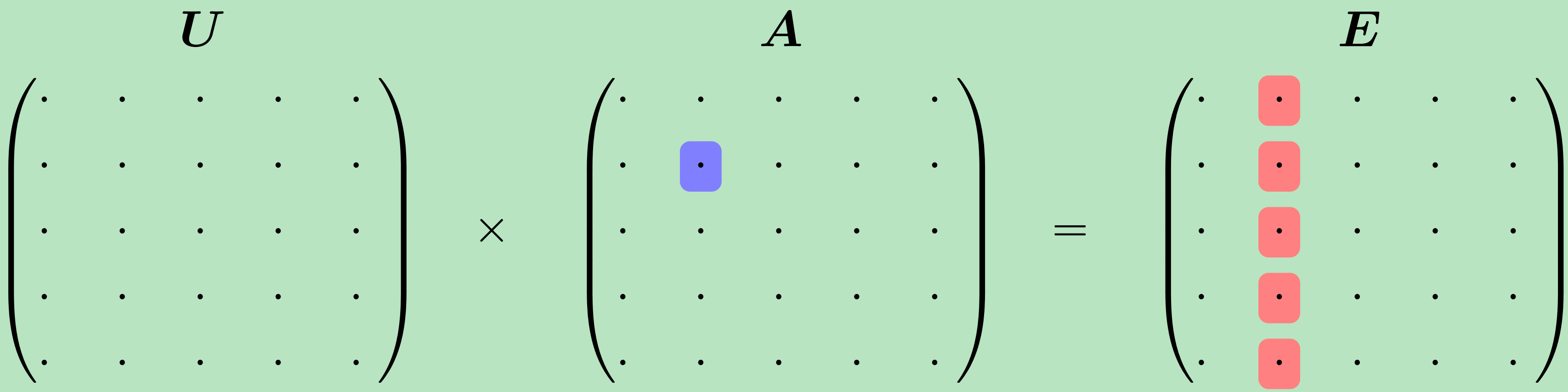
$$\begin{pmatrix} 1 & 4 & 0 & 2 & 0 & 2 \\ 0 & 0 & 1 & 3 & 0 & 4 \\ 0 & 0 & 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Thanks to
  - rank$(\boldsymbol{E}) =$ rank$(\boldsymbol{U}\boldsymbol{A}) =$ rank$(\boldsymbol{A})$, and
  - the structure of $\boldsymbol{E}$

  we get: rank$(\boldsymbol{A}) =$ number of non-zero rows of $\boldsymbol{E}$

Step 3: FULLRANKMODP $\in$ DynFO (2/2)

Proof (cont.)



- A change of $A[i, j]$ can only affect the j-th column of E
- To bring E back to reduced echelon form:
  - (i) If new leading entries occur in column j:
    - keep one with a maximum number of successive zeros in its row, and
    - set all other entries of column j to 0 by appropriate row operations
  - (ii) If a former leading entry of a row k is lost in column j (by the change in A or by (i))
    - Take the next non-zero-entry on row k
    - Clean its column by appropriate row operations
  - (iii) If needed: move the ($\leqslant 2$) rows whose leading entry has changed to their correct row positions (and adapt them so that their leading entries are 1)
  - (iv) Update U accordingly
- These update operations can be specified by first-order formulas

Step 4: From DynFO$(+, \times)$ to DynFO: Challenge

- Steps 1-3: REACH $\in$ **DynFO**$(+, \times)$

- How to obtain a **DynFO**-program P' from a **DynFO**$(+, \times)$-program P?

Proof idea

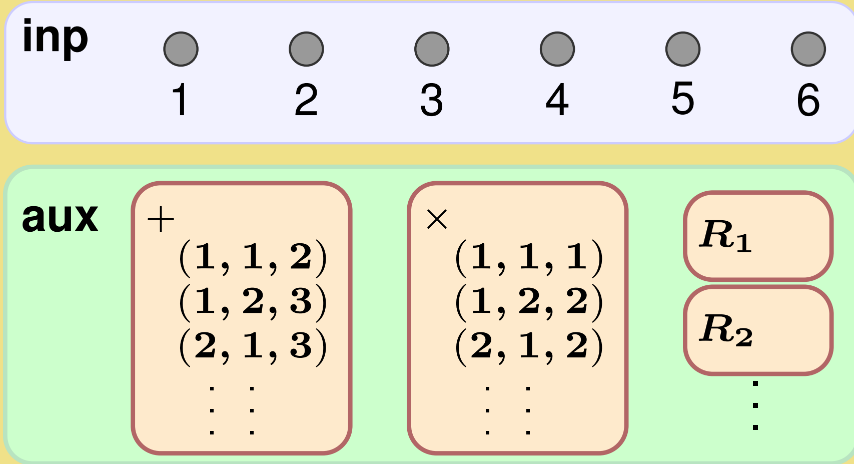
- Arithmetic for active elements can be built on the fly for the activated elements

[Etessami '98]

- We show that for domain independent queries, this approach can be extended to programs which use arithmetic for *all* elements from the very beginning

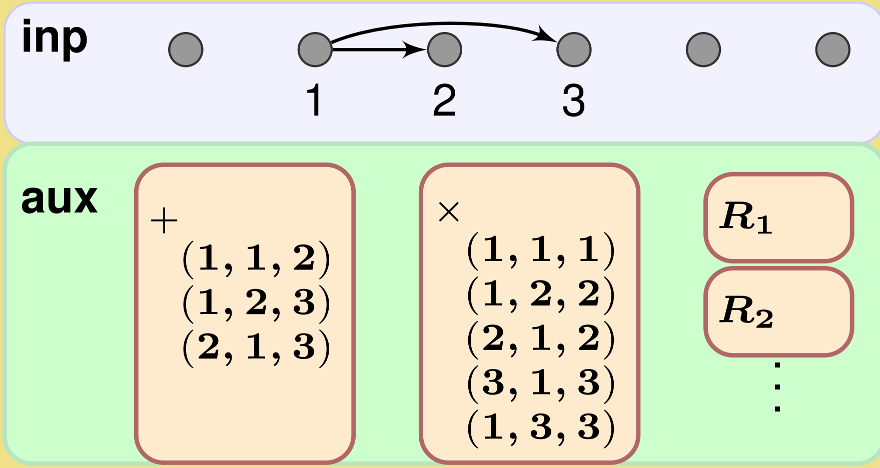
- *Domain independent:* invariant under adding isolated nodes

- **Illustration of DynFO$(+, \times)$:**



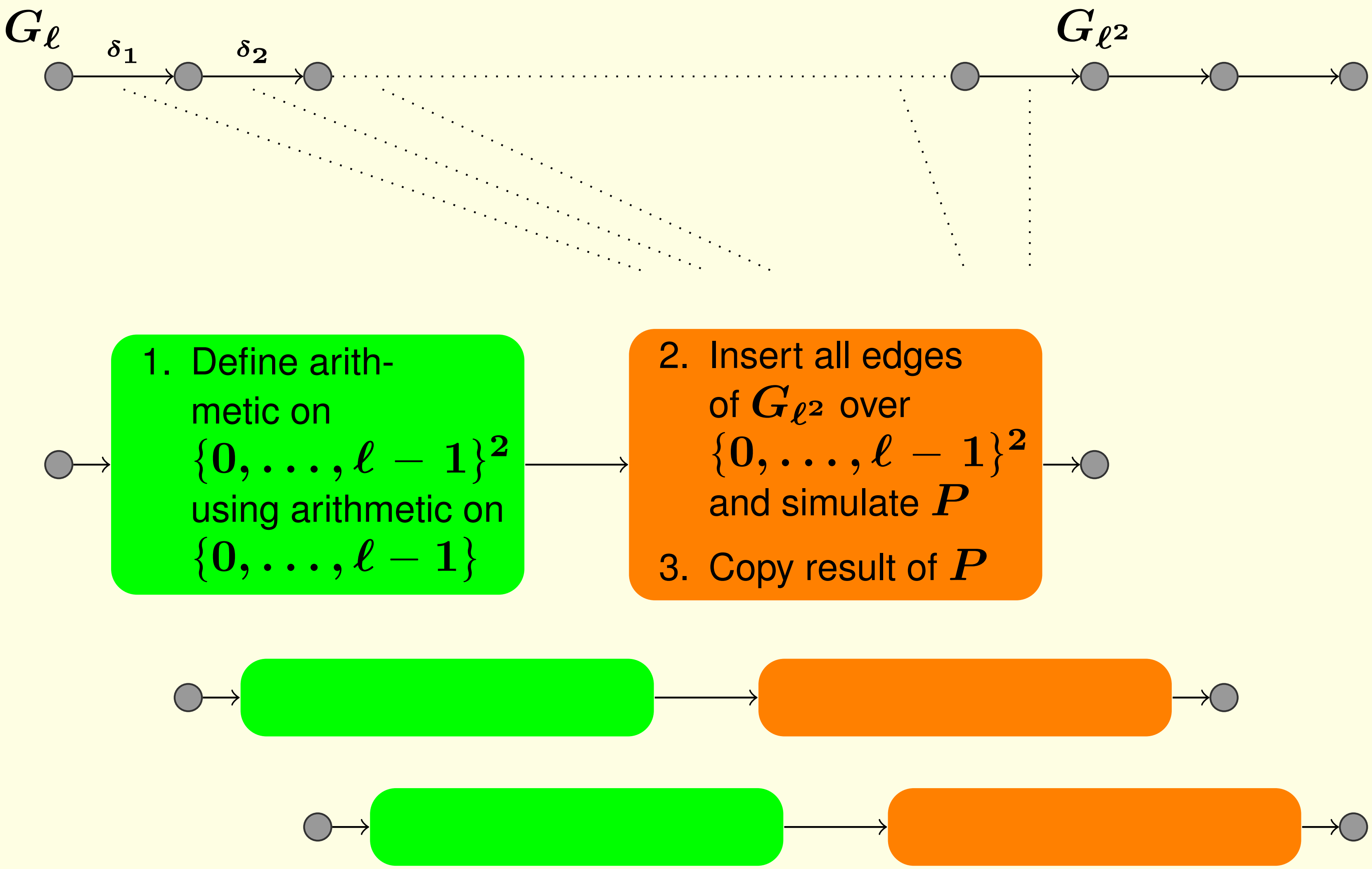
  - Updates can use arithmetic from the very beginning

- **Illustration of DynFO:**



  - Initially, updates can not use arithmetic

Step 4: From DynFO$(+,\times)$ to DynFO: Illustration



- To be done for all $\ell < \sqrt{n}$
- G_ℓ denotes the first graph in the computation, in which $\geqslant \ell$ elements are *activated*

Contents

Lower bounds: a sad state

- Easy observation: $q \in$ **DynFO** $\Rightarrow q \in$ **PTIME**
  - Just insert the tuples of $\mathcal{D}$ into an empty database one by one, and compute all updates

- So far there are no other general lower bound results for **DynFO**
- We cannot rule out that: **DynFO** $=$ **P**

- Most existing lower bounds apply to
  - auxiliary relations of bounded arity or
  - restricted logics or
  - both...

Reachability is not in unary DynFO (1/2)

Theorem [Dong Su 95/98]

- REACH $\notin$ unary **DynFO**

- unary **DynFO**: Update programs with unary auxiliary relations

Proof sketch

- Proof by contradiction with a locality argument
- Assume there is a unary dynamic program for REACH with m unary aux relations and a rule

  on delete (u, v) **from** E
  update $Q(x, y)$ **as** $\varphi(u, v, x, y)$

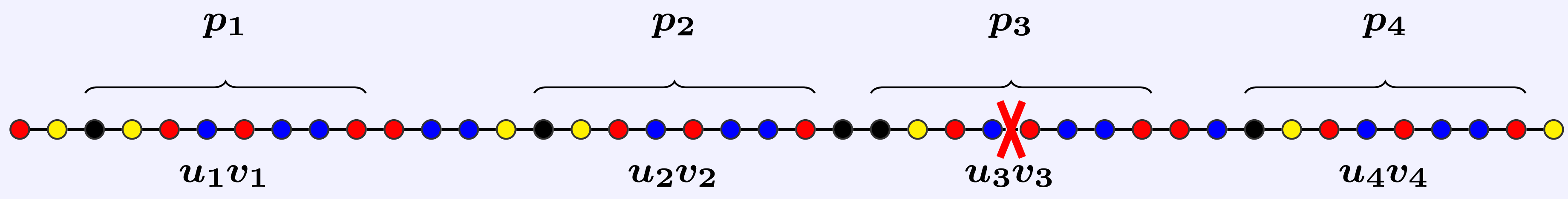
  with φ of quantifier-depth k

- The aux relations induce, for each node, one of 2^m colours

- Consider a graph consisting of a sufficiently long path with $\geqslant 4(2 \cdot 4^k + 2)2^{m(2 \cdot 4^k + 2)}$ nodes

Reachability is not in unary DynFO (2/2)

Example



Proof sketch (cont.)

- Since the path is long enough, there must exist four disjoint subpaths of length $2 \cdot 4^k + 2$ each with identical color (relations) sequence

- Let $(u_1, v_1), \ldots, (u_4, v_4)$ be the innermost edges of these paths
- After deletion of (u_3, v_3),
  - u_2 is still reachable from v_1, but
  - u_4 is no longer reachable from v_1

- The 4^k-neighborhoods of (v_1, u_3, v_3, u_2) and (v_1, u_3, v_3, u_4) are isomorphic

➡ $\varphi(u_3, v_3, v_1, u_2) \equiv \varphi(u_3, v_3, v_1, u_4)$ by Gaifman's Theorem

➡ After deletion of (u_3, v_3), the program gives the same answer for (v_1, u_2) and (v_1, u_4)

➡ The program is wrong with respect to either (v_1, u_2) or (v_1, u_4), the desired contradiction

Dynamic programs with quantifier-free formulas

- Hesse initiated the study of dynamic programs with quantifier-free update formulas [Hesse 03]

Definition

- **DynProp:**
  - Queries that can be maintained in **DynFO** with quantifier-free formulas and aux **relations**
- **DynQF:**
  - Queries that can be maintained in **DynFO** with quantifier-free formulas and aux **functions** (and relations)

✎ **DynQF** formulas can use "if-then-else"-terms

- Quantifier-free update formulas? Isn't that extremely weak?

Theorem [Hesse 03]

- Reachability is in **DynProp** for deterministic graphs ☞no quantifiers, aux relations

Theorem [Hesse 03]

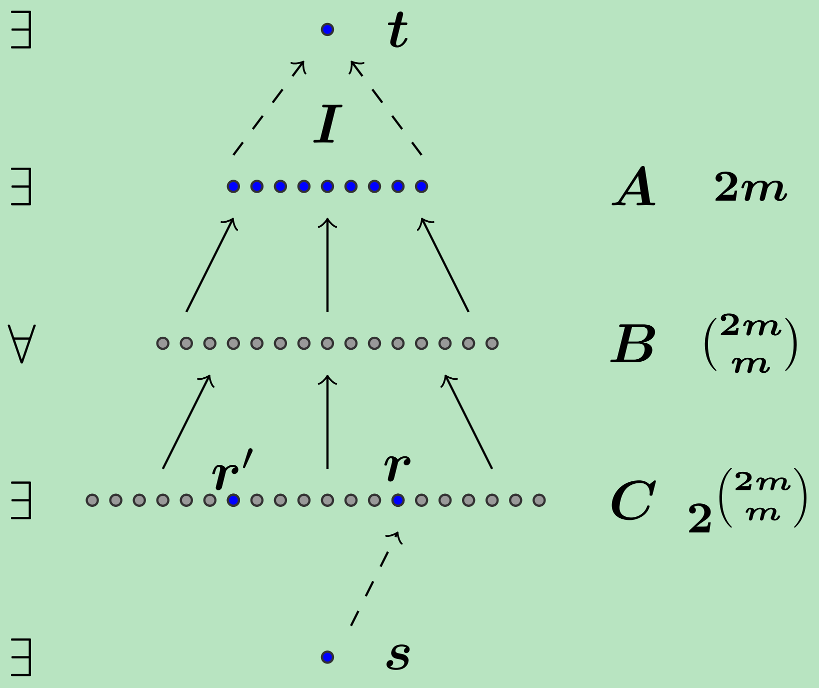
- Reachability is in **DynQF** for undirected graphs ☞ no quantifiers, unary aux functions & relations

Alternating Reachability is not in DynProp

Theorem [Gelade, Marquardt, Schwentick 08/12]

- Alternating Reachability $\notin$ **DynProp**

Proof idea



- A: $2m$ existential nodes $v_1, \ldots, v_{2m}$
- B: one universal node per size-m-subset of A
- C: one existential node per subset of B

Proof idea (cont.)

- Assume: $\mathcal{P}$ is a **DynProp** program for Alternating Reachability ☞ and let m be large enough
- There are $> 2^{2^m}$ nodes in C
- There are $< 2^{2^m}$ isomorphism types for tuples $(s, t, v_1, \ldots, v_{2m}, r)$ if m is sufficiently large with respect to $\mathcal{P}$

➨ There are $r \neq r'$ in C with the same tuple type ☞ together with $s, t, v_1, \ldots, v_{2m}$

➨ There is a set $I \subseteq A$ such that insertion of all edges (u, t), $u \in I$, makes t (alternatingly) reachable from exactly one of r and r'

- However, after adding either (s, r) or (s, r') the tuples $(s, t, v_1, \ldots, v_{2m}, r)$ and $(s, t, v_1, \ldots, v_{2m}, r')$ still have the same type

➨ Contradiction

Some Further Inexpressibility Results

Theorem [Gelade, Marquardt, Schwentick 08/12]

- **FO** $\nsubseteq$ **DynProp**

Theorem [Zeume, Schwentick 13]

- REACH $\notin$ binary **DynProp**

Theorem [Zeume 14]

- If only edge insertions are allowed:
  - $\boldsymbol{k}$-CLIQUE can be maintained in $(\boldsymbol{k}-\mathbf{1})$-ary **DynProp**
  - $\boldsymbol{k}$-CLIQUE $\notin$ $(\boldsymbol{k}-\mathbf{2})$-ary **DynProp**

Contents

Conclusion

- **DynFO** is far more powerful than expected
- Upper bound results might be even "practical"

- Lower bounds for **DynFO** seem hopeless

- A lot remains to be done
  - Applications of the Reachability result
  - Implementations
  - Further exploration of linear algebra approaches
  - ...

References (1/2)

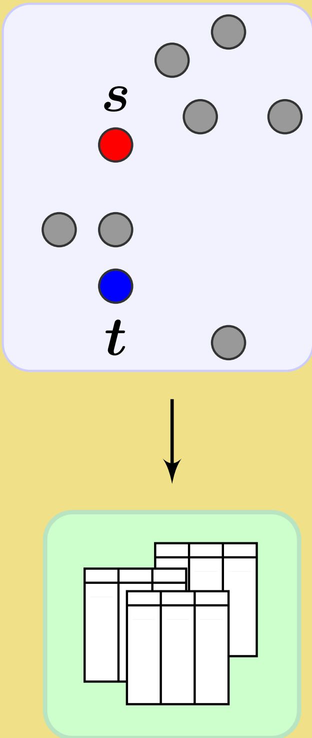
- Guozhu Dong and Jianwen Su. First-order incremental evaluation of datalog queries. In *DBPL 1993*, pages 295–308, 1993
- Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. In *PODS 1994*, pages 210–221, 1994
- Sushant Patnaik and Neil Immerman. Dyn-FO: A parallel, dynamic complexity class. *J. Comput. Syst. Sci.*, 55(2):199–209, 1997
- Guozhu Dong and Jianwen Su. Arity bounds in first-order incremental evaluation and definition of polynomial time database queries. *J. Comput. Syst. Sci.*, 57(3):289–308, 1998
- D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990
- Kousha Etessami. Dynamic tree isomorphism via first-order updates. In *PODS*, pages 235–243. ACM Press, 1998
- William Hesse. The dynamic complexity of transitive closure is in $DynTC^0$. In *ICDT 2001*, pages 234–247, 2001
- Bastian Laubner. *The structure of graphs and new logics for the characterization of Polynomial Time*. PhD thesis, Humboldt University of Berlin, 2011
- Gudmund Skovbjerg Frandsen and Peter Frands Frandsen. Dynamic matrix rank. *Theor. Comput. Sci.*, 410(41):4085–4093, 2009
- Samir Datta, William Hesse, and Raghav Kulkarni. Dynamic complexity of directed reachability and other problems. In *ICALP (1)*, pages 356–367, 2014

References (2/2)

- Samir Datta, Anish Mukherjee, Nils Vortmeier, and Thomas Zeume. Reachability and distances under multiple changes. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 120:1–120:14, 2018
- Samir Datta, Anish Mukherjee, Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. A strategy for dynamic programs: Start over and muddle through. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 98:1–98:14, 2017
- Thomas Schwentick, Nils Vortmeier, and Thomas Zeume. Dynamic complexity under definable changes. In *20th International Conference on Database Theory, ICDT 2017*, pages 19:1–19:18, 2017
- Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, and Thomas Zeume. Reachability is in dynfo. In *International Colloquium on Automata, Languages, and Programming, ICALP 2015, Proceedings, Part II*, pages 159–170, 2015
- Thomas Zeume and Thomas Schwentick. Dynamic conjunctive queries. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 38–49, 2014
- Thomas Zeume and Thomas Schwentick. On the quantifier-free dynamic complexity of reachability. In *Mathematical Foundations of Computer Science 2013*, pages 837–848, 2013
- Wouter Gelade, Marcel Marquardt, and Thomas Schwentick. The dynamic complexity of formal languages. *ACM Trans. Comput. Log.*, 13(3):19, 2012

Three initialisation settings

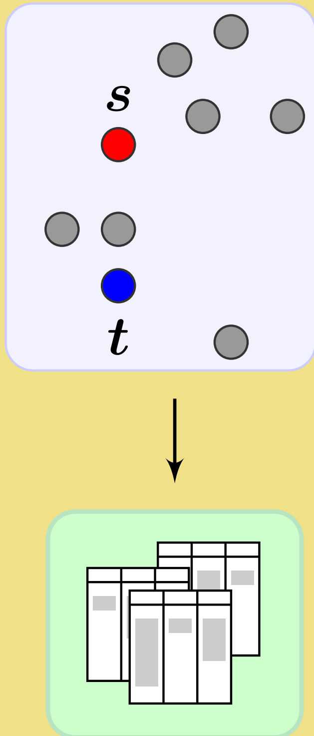
DynFO



- Start from **empty input** and **empty auxiliary data**

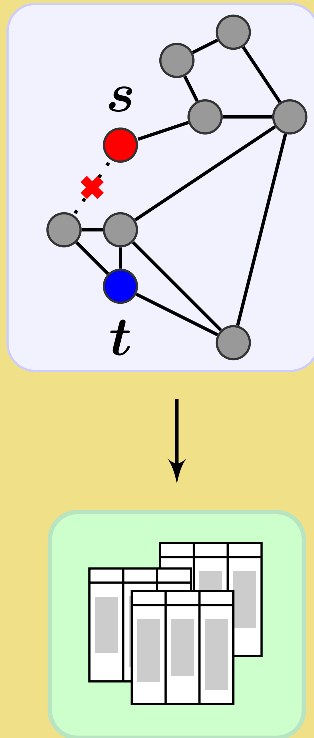
[Patnaik, Immerman 94/97]

DynFO$(+, \times)$



- Start from **empty input** and **precomputed auxiliary arithmetic relations** $+$ and $\times$
  - depending on the universe

✎ Other initialisations of the auxiliary relations possible

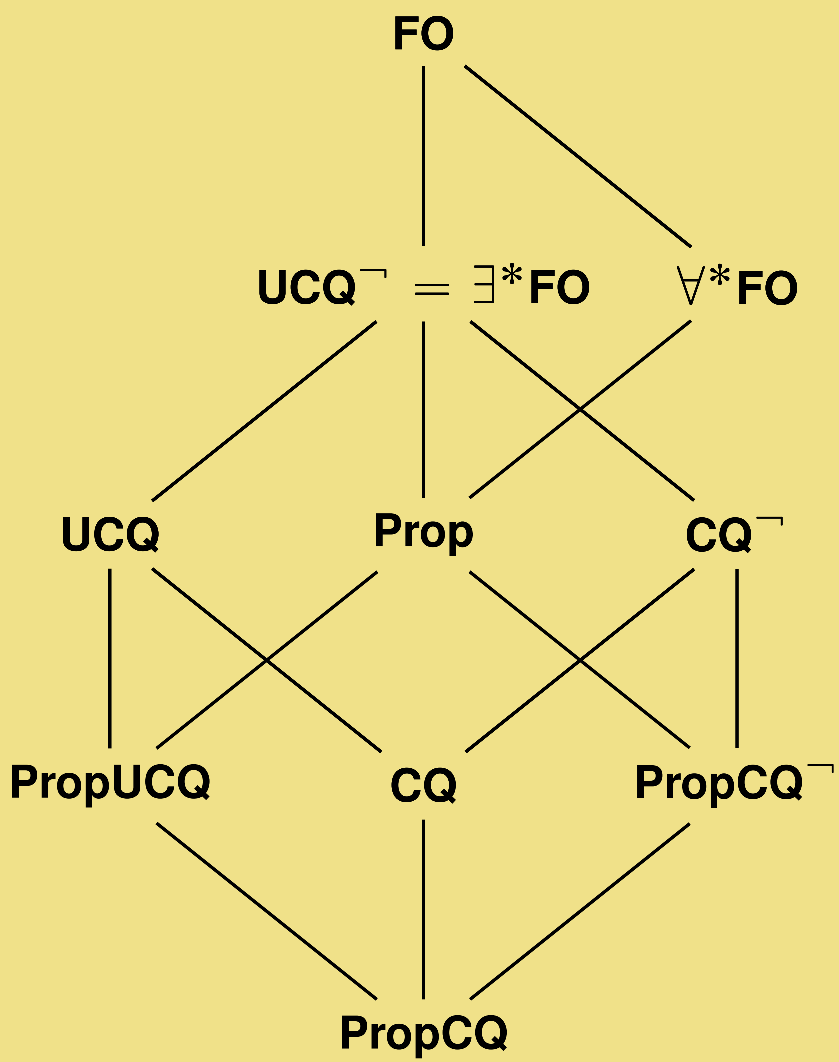


- Starts from **non-empty input** and **precomputed auxiliary data**
  - depending on the actual input

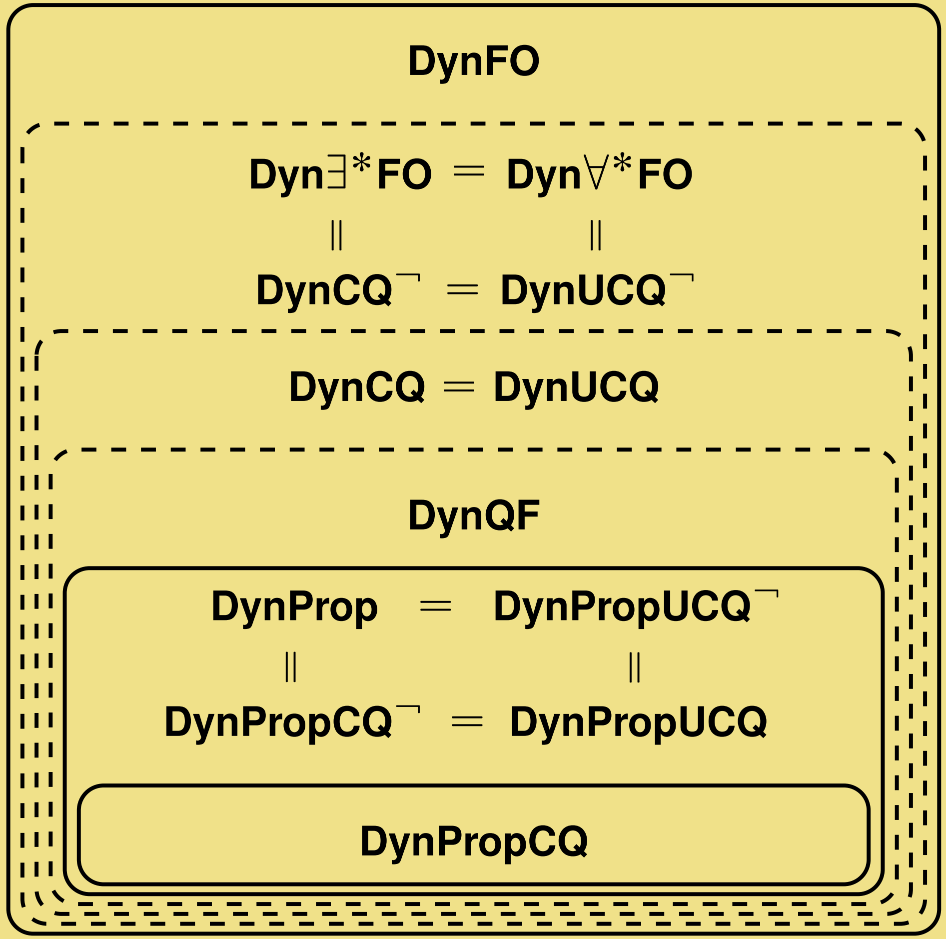
✎ Interesting, but not considered in this talk...

How do small fragments of DynFO relate?

- Small fragments in the static world



- Small fragments in the dynamic world



(non-empty input, PTIME aux) [Zeume, Schwentick 14]

- Many static classes coincide in the dynamic world
- Linear hierarchy of classes!
- Further: **FO** $\subseteq$ $\mathbf{DynCQ}^{\neg}$

Dynamic Complexity of Formal Languages

Theorem [Patnaik, Immerman 94/97]

- **Reg** $\subseteq$ **DynFO**
- All Dyck languages can be maintained in **DynFO**

Theorem [Hesse 03]

- **Reg** $\subseteq$ **DynQF**

Theorem [Gelade, Marquardt, TS 09/12]

- With respect to formal languages: **DynProp** $=$ **Reg**

Theorem [Gelade, Marquardt, TS 09/12]

- **CFL** $\subseteq$ **DynFO**
- All Dyck languages can be maintained in **DynQF**

Corollary

- **DynProp** $\subsetneq$ **DynQF**