Dynamic descriptive complexity of FO-definable modifications

Nils Vortmeier

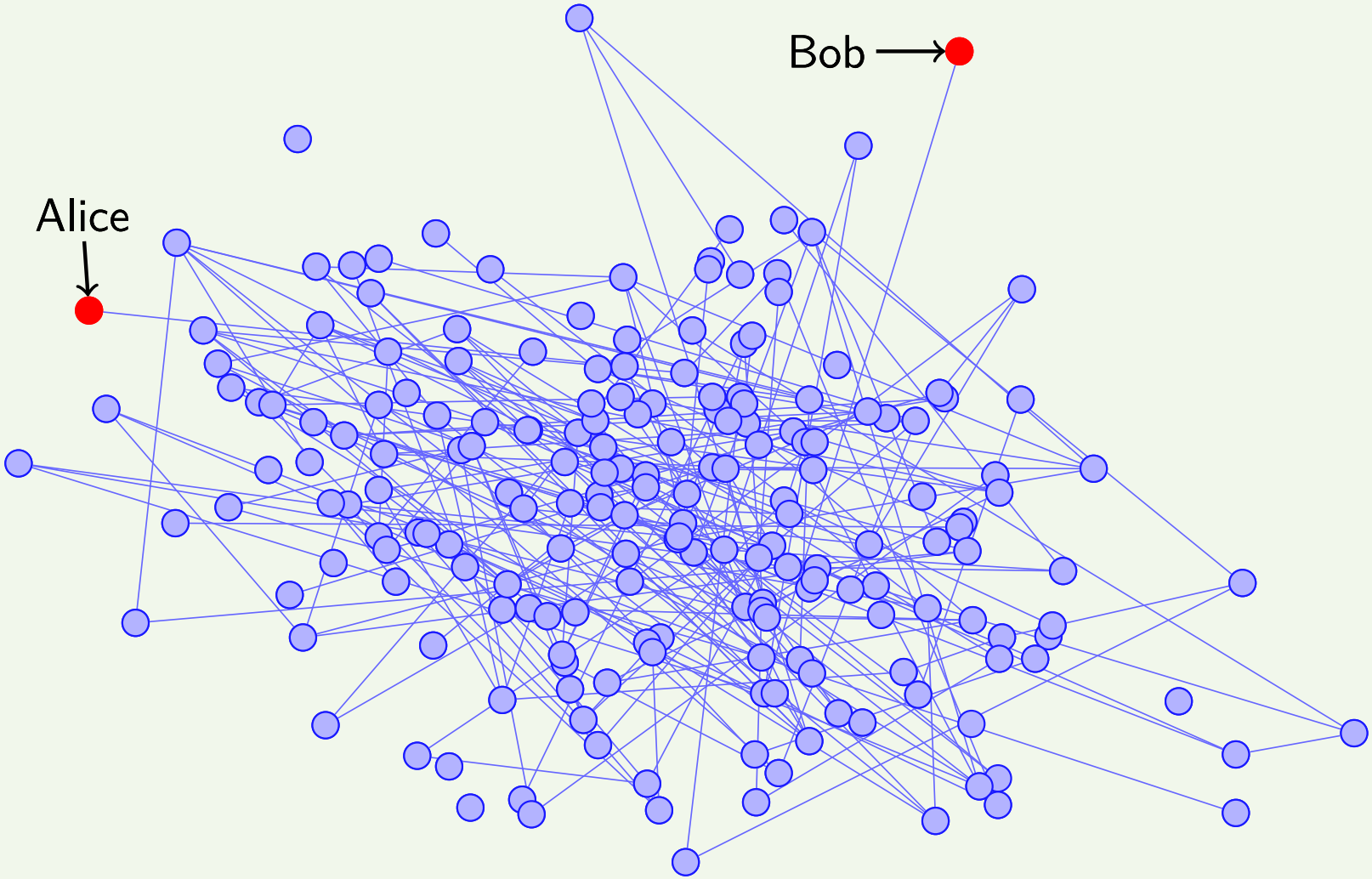
Joint work with
Thomas Schwentick Thomas Zeume

technische universität
dortmund

Highlights, September 2016

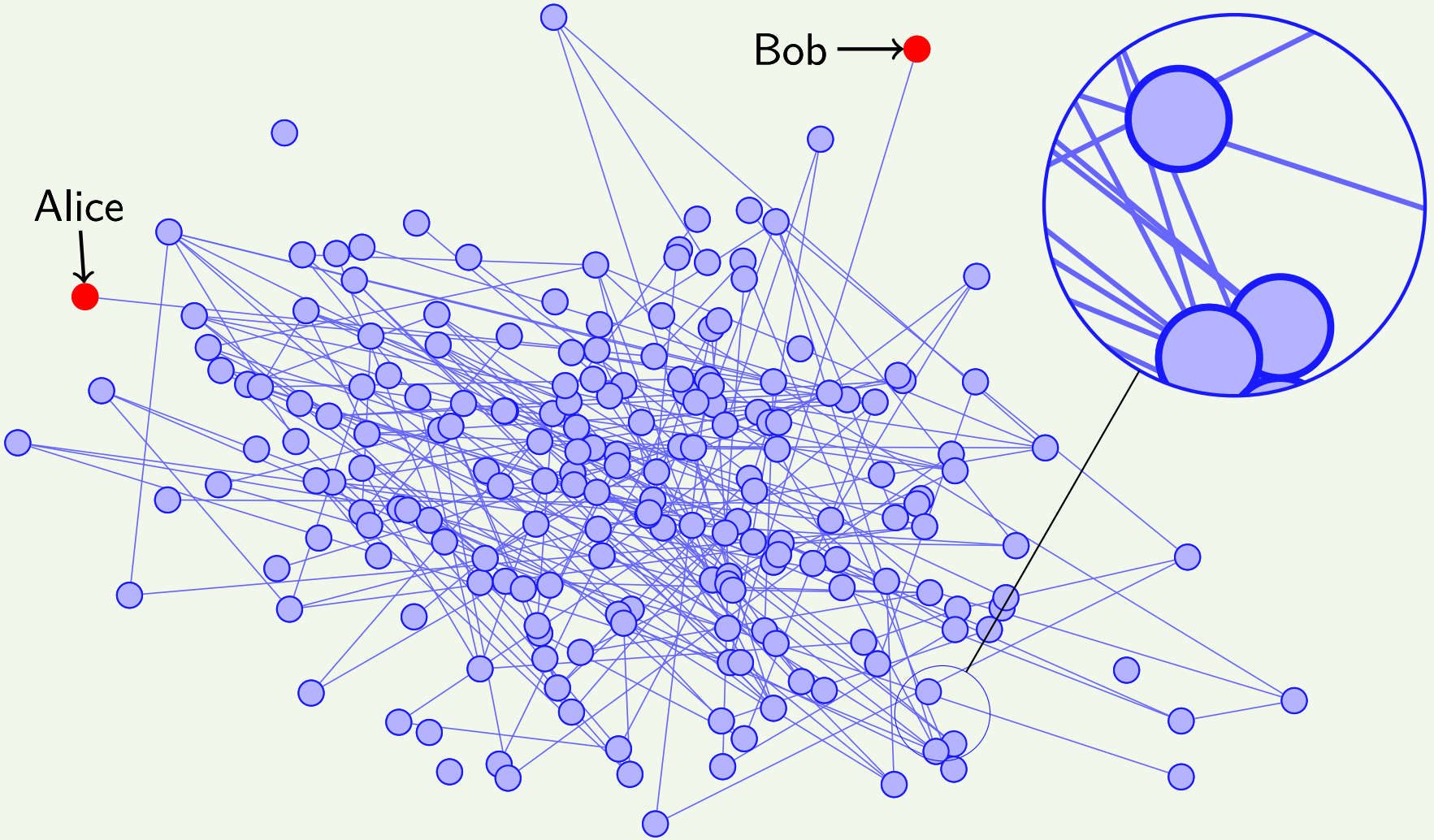
Social networks

Does Alice know someone who knows someone . . . who knows Bob?



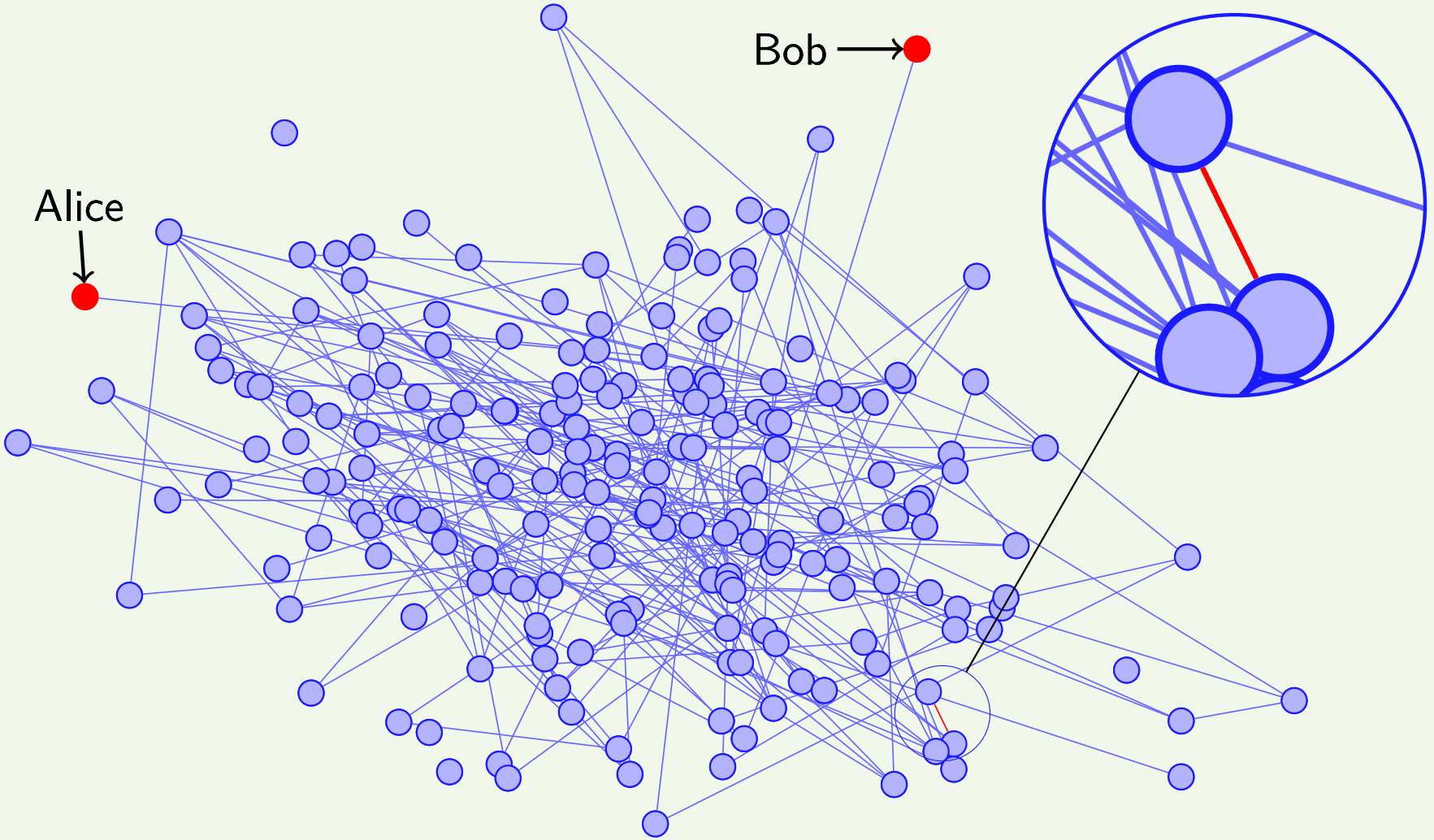
Social networks

Does Alice know someone who knows someone . . . who knows Bob?



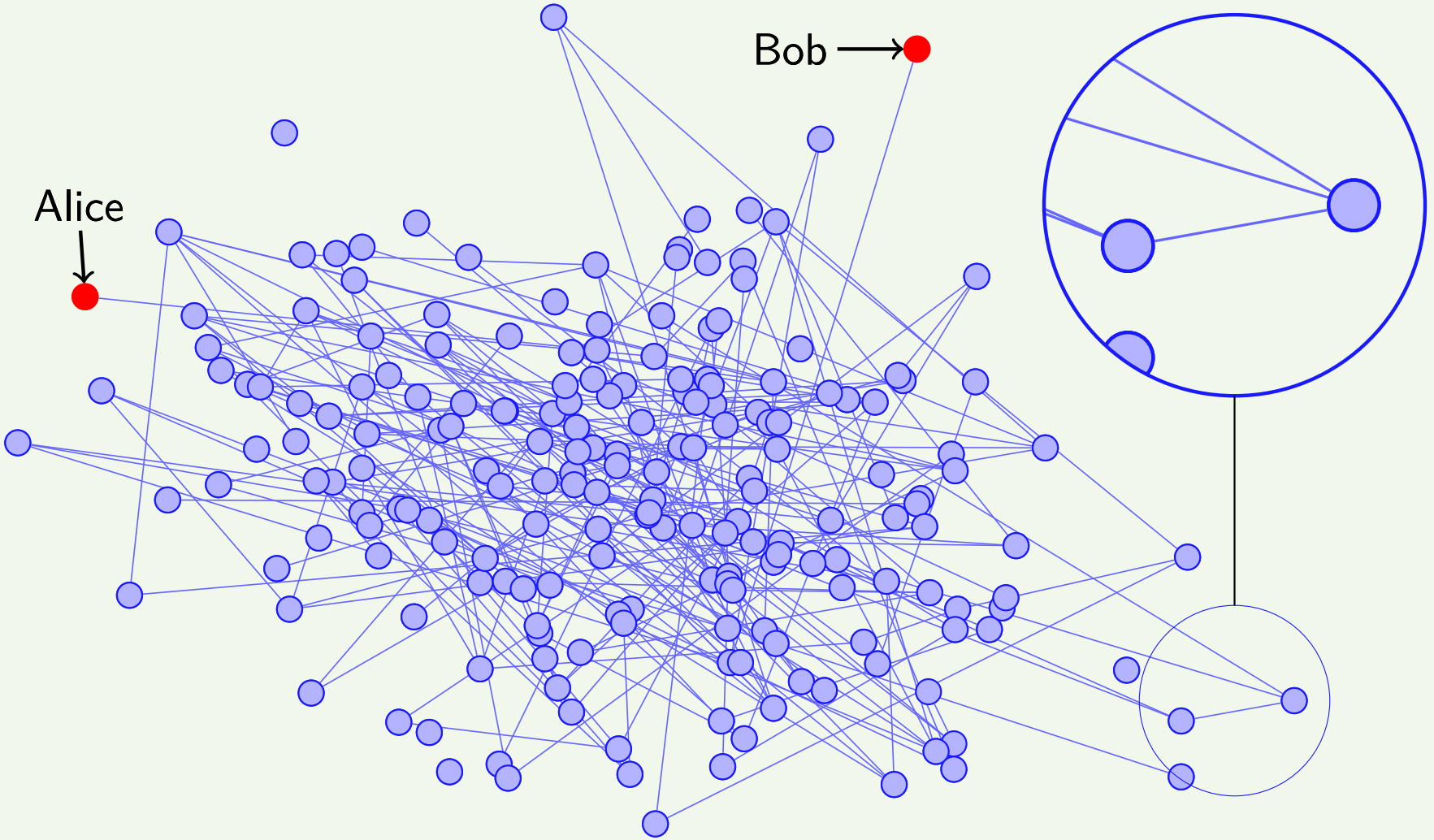
Social networks

Does Alice know someone who knows someone . . . who knows Bob?



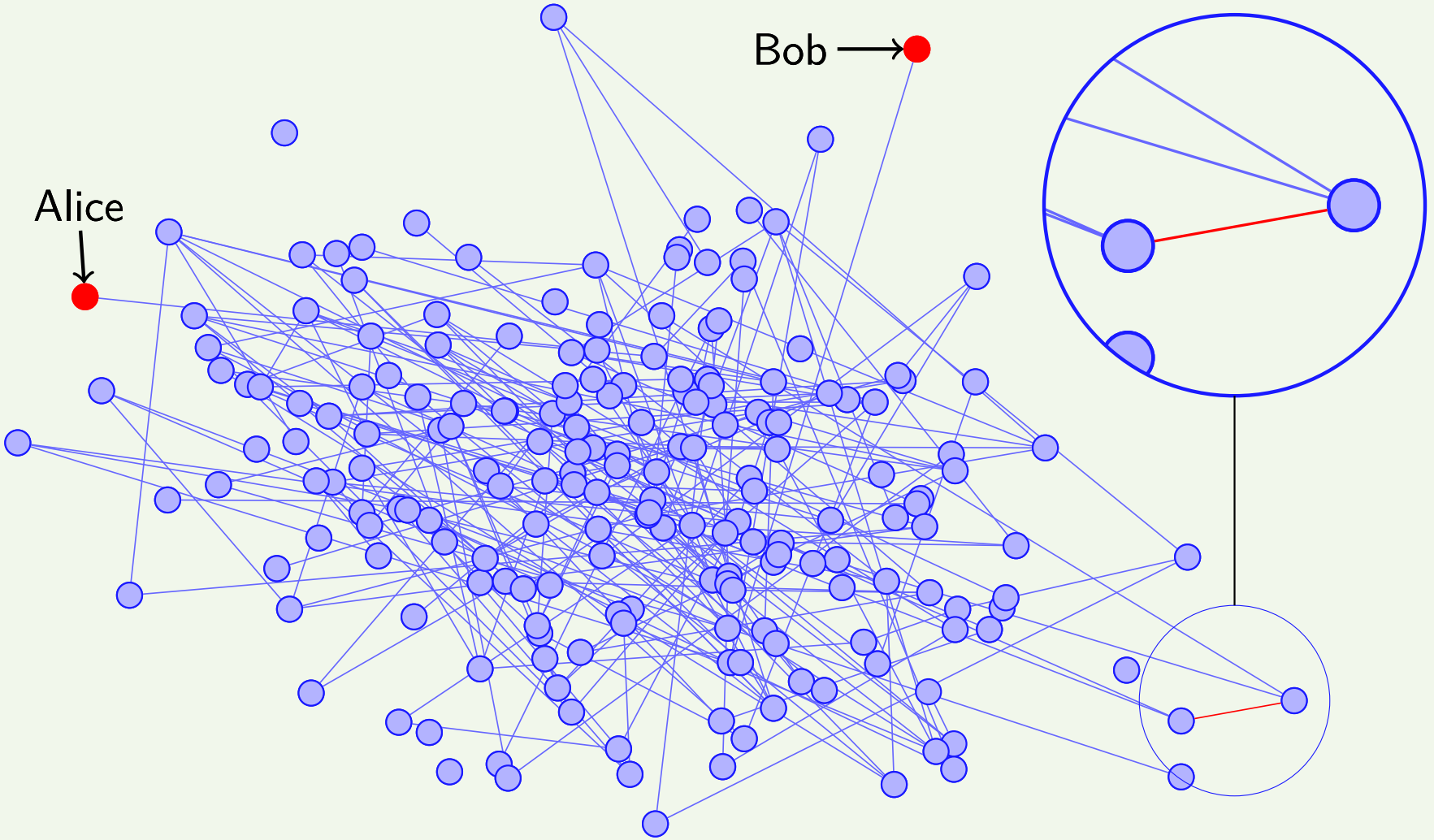
Social networks

Does Alice know someone who knows someone . . . who knows Bob?



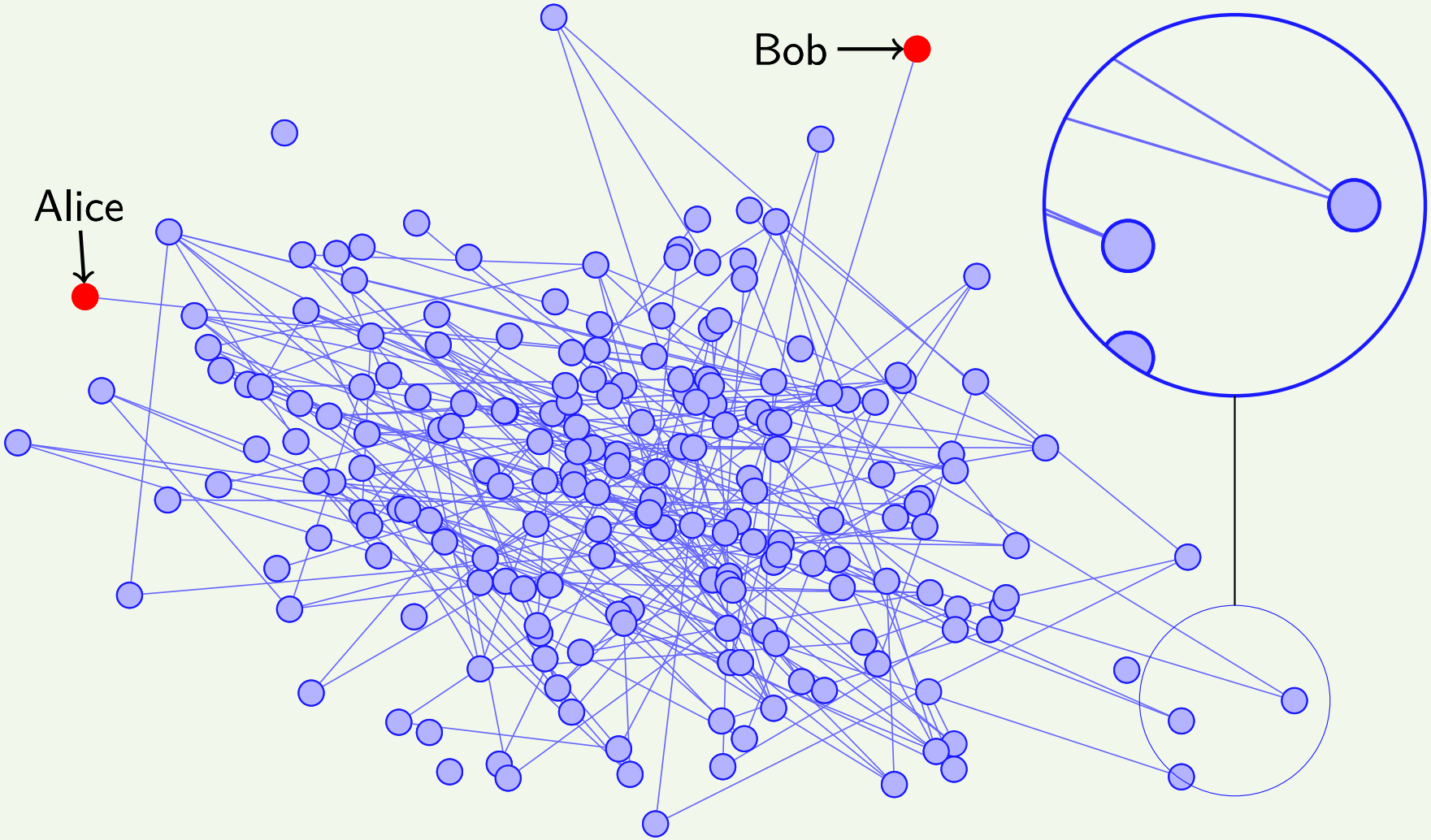
Social networks

Does Alice know someone who knows someone . . . who knows Bob?



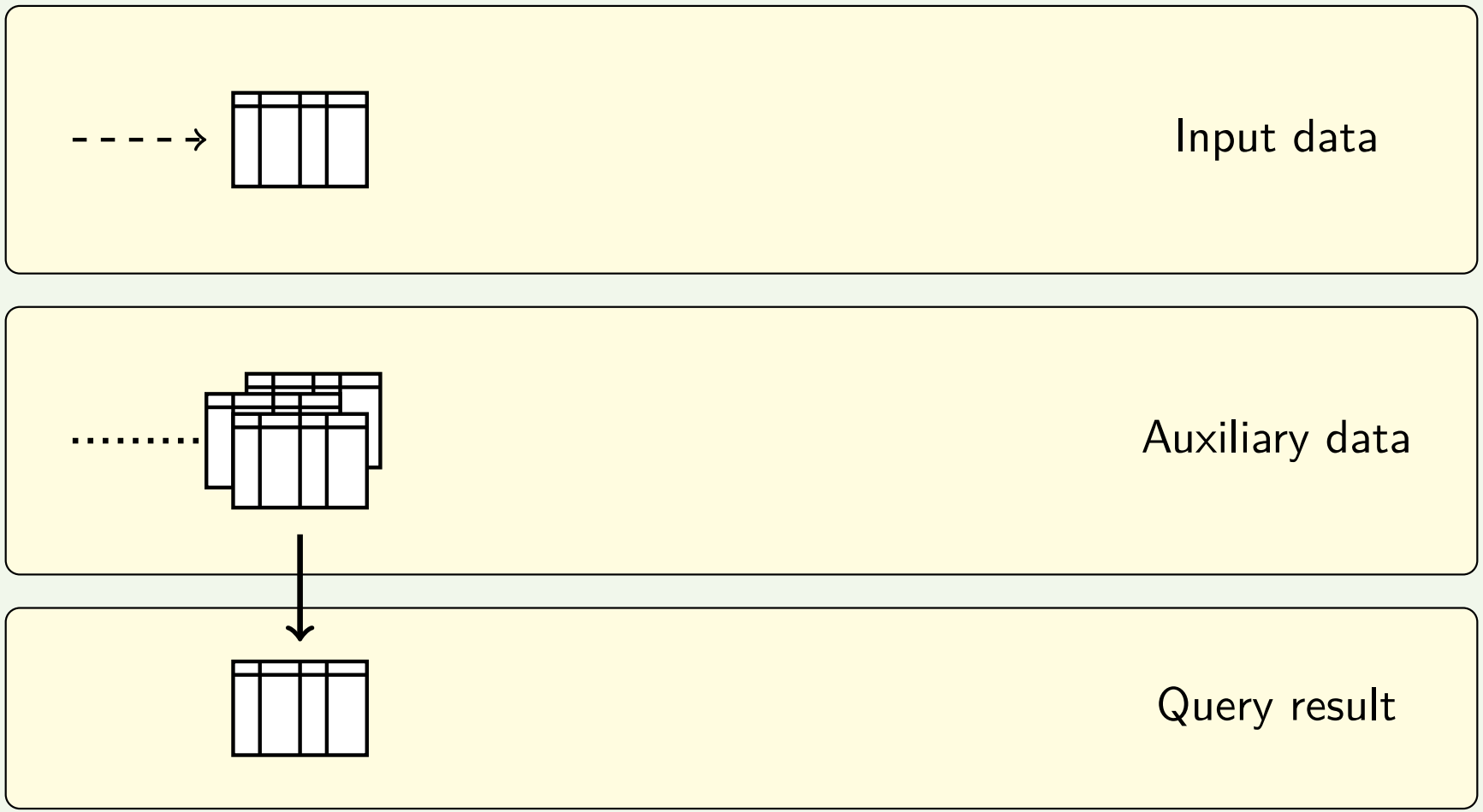
Social networks

Does Alice know someone who knows someone . . . who knows Bob?



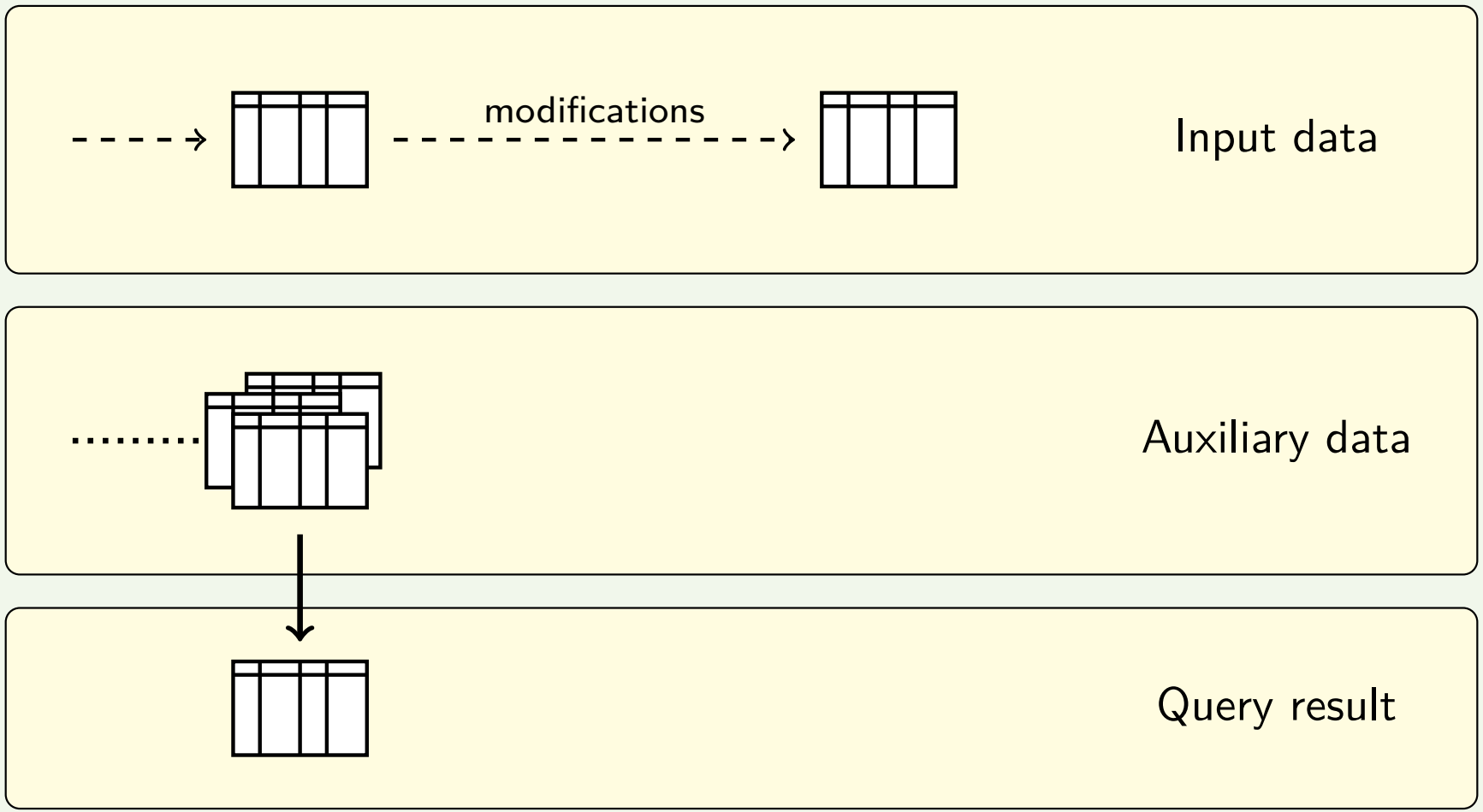
The Dynamic Setting

Dynamic Evaluation of a query



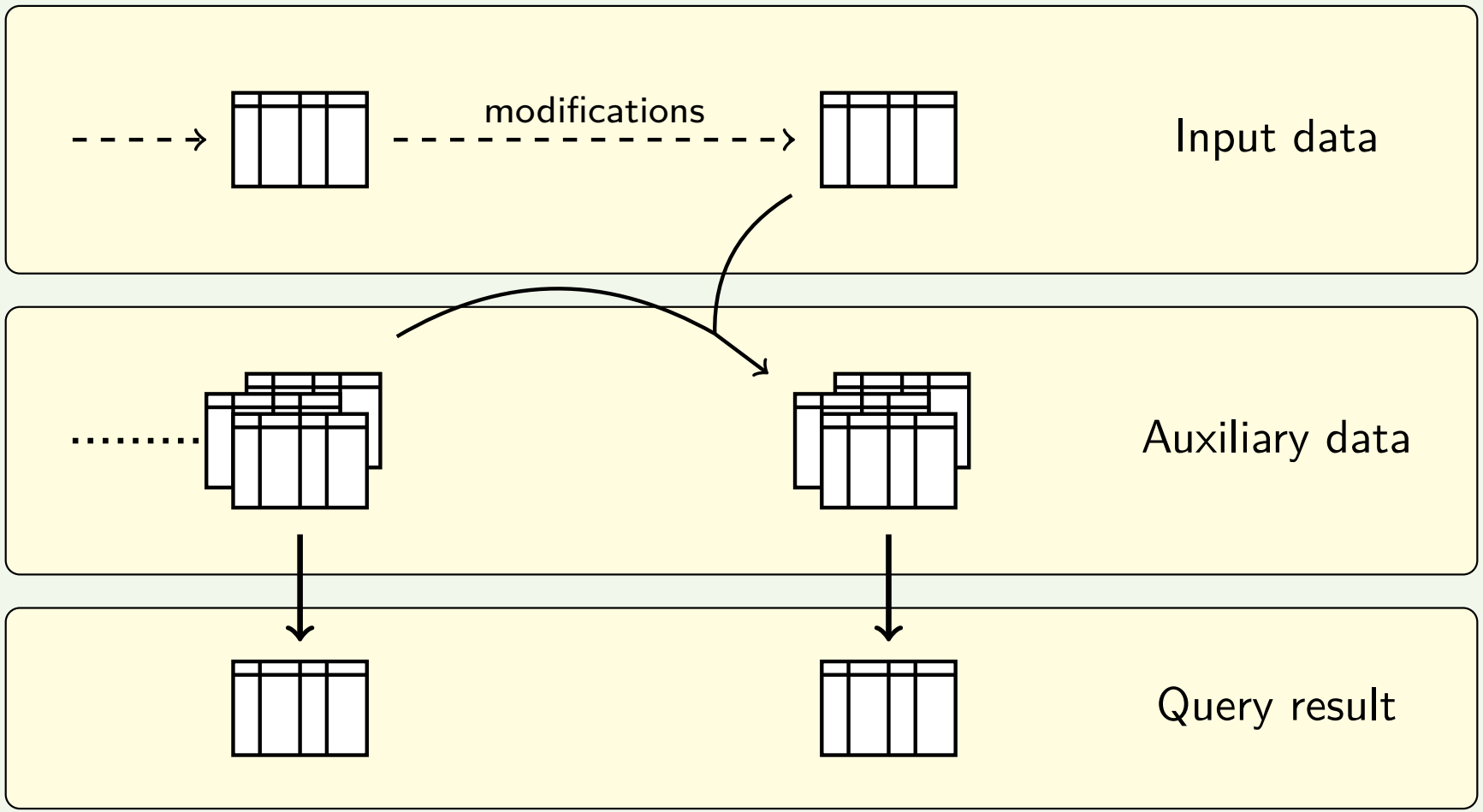
The Dynamic Setting

Dynamic Evaluation of a query



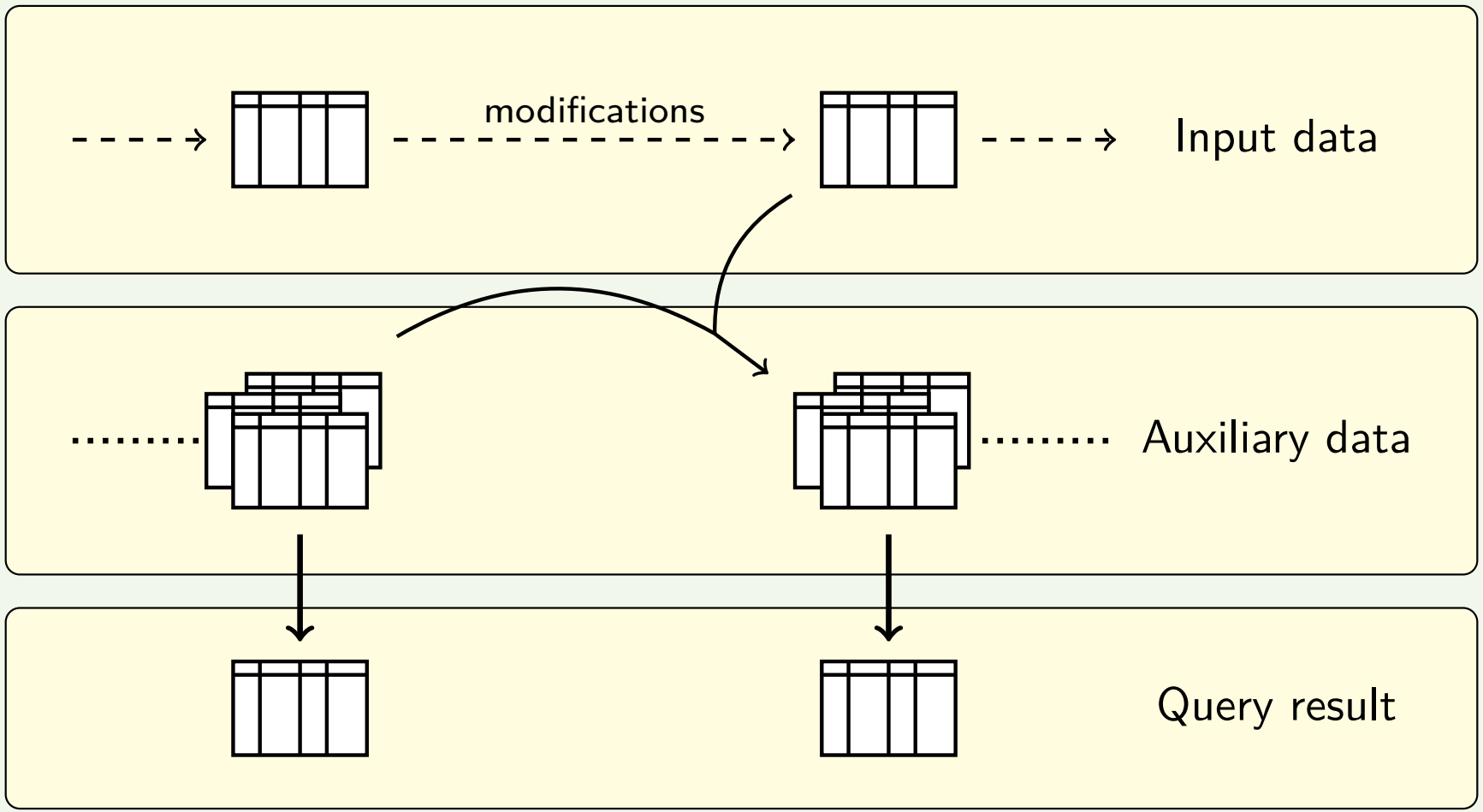
The Dynamic Setting

Dynamic Evaluation of a query



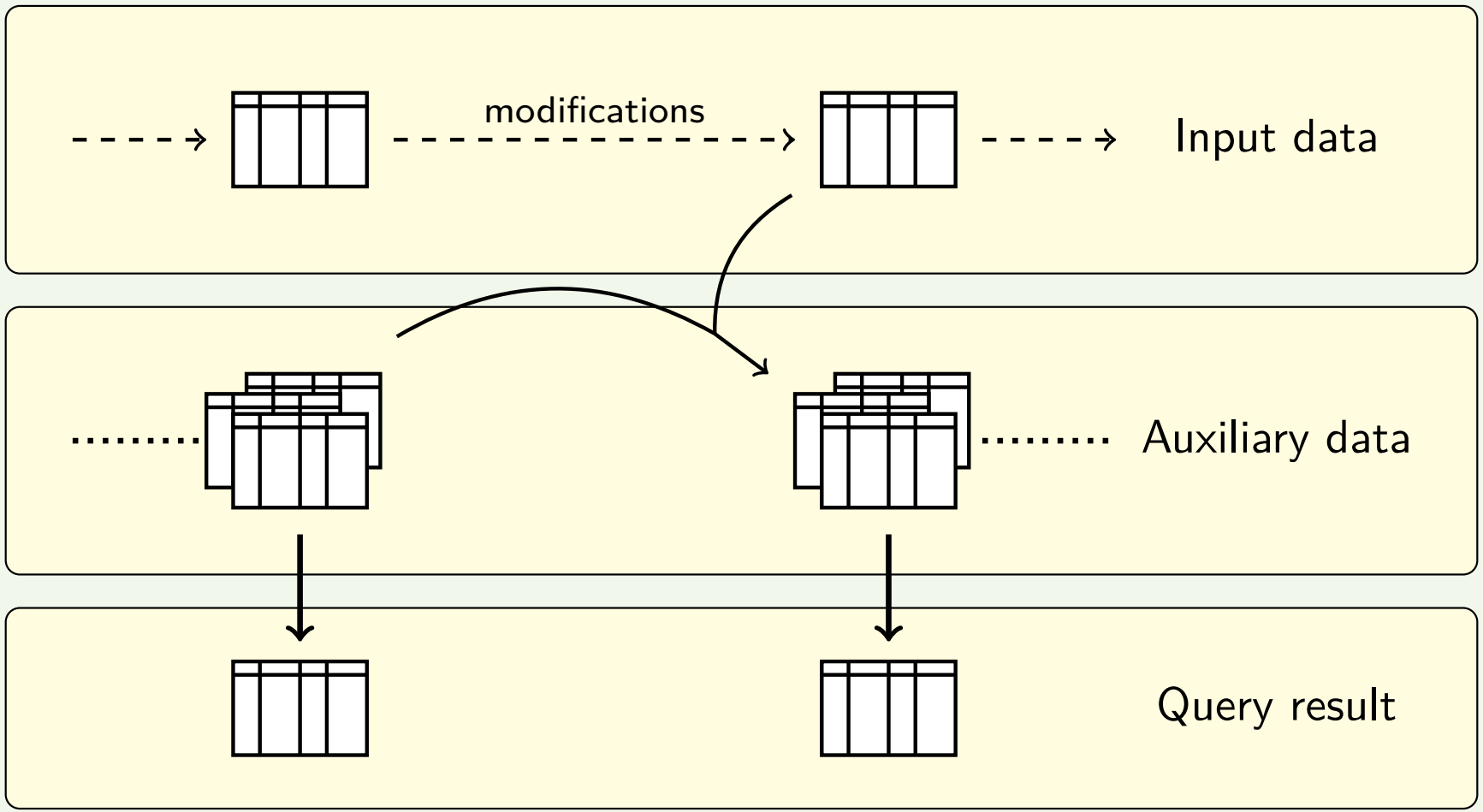
The Dynamic Setting

Dynamic Evaluation of a query



The Dynamic Setting

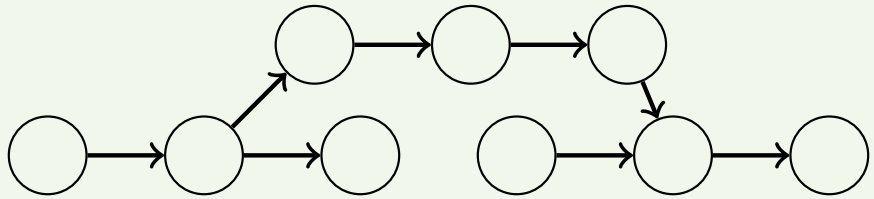
Dynamic Evaluation of a query



- Descriptive approach: Update auxiliary data by logical formulas

Example ([Patnaik and Immerman 1997])

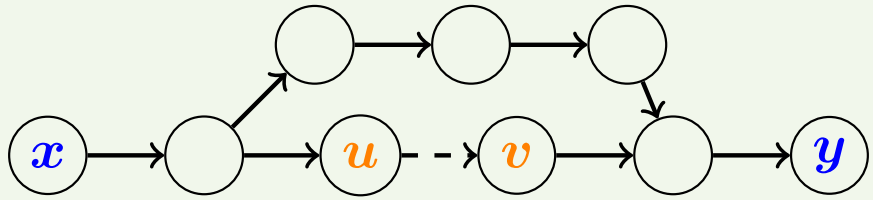
Reachability on acyclic graphs: single edge insertion



- Goal: Maintain the transitive closure in an auxiliary relation T

Example ([Patnaik and Immerman 1997])

Reachability on acyclic graphs: single edge insertion

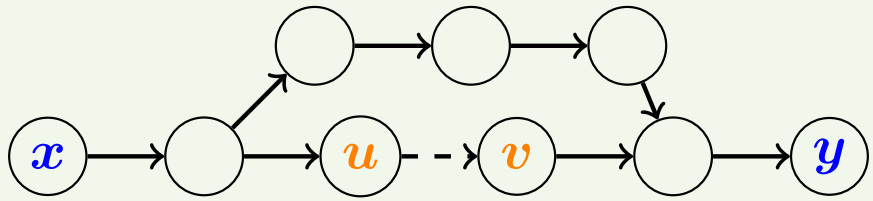


- Goal: Maintain the transitive closure in an auxiliary relation T
- Is $(x, y) \in T$ after an edge (u, v) is inserted?

$$\phi_{\text{ins}} = T(x, y) \vee \big(T(x, u) \wedge T(v, y)\big)$$

Example ([Patnaik and Immerman 1997])

Reachability on acyclic graphs: single edge insertion



- Goal: Maintain the transitive closure in an auxiliary relation T
- Is $(x, y) \in T$ after an edge (u, v) is inserted?

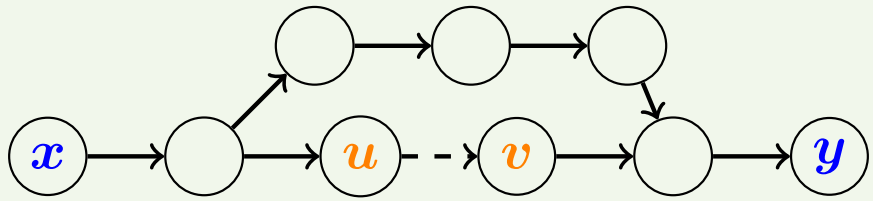
$$\phi_{\text{ins}} = T(x, y) \vee \big(T(x, u) \wedge T(v, y)\big)$$

- Formula for single edge deletion is slightly more complex

$$\begin{aligned}\phi_{\text{del}} = T(x, y) \wedge \Big(\Big(\neg T(x, u) \vee \neg T(v, y)\Big) \vee \\ \exists z \exists z' \Big(T(x, z) \wedge E(z, z') \wedge (z \neq u \vee z' \neq v) \wedge \\ T(z', y) \wedge T(z, u) \wedge \neg T(z', u)\Big)\Big)\end{aligned}$$

Example ([Patnaik and Immerman 1997])

Reachability on acyclic graphs: single edge insertion



- Goal: Maintain the transitive closure in an auxiliary relation T
- Is $(x, y) \in T$ after an edge (u, v) is inserted?

$$\phi_{\text{ins}} = T(x, y) \vee \big(T(x, u) \wedge T(v, y)\big)$$

- Formula for single edge deletion is slightly more complex

$$\begin{aligned}\phi_{\text{del}} = T(x, y) \wedge \Big(\big(\neg T(x, u) \vee \neg T(v, y)\big) \vee \\ \exists z \exists z' \Big(T(x, z) \wedge E(z, z') \wedge (z \neq u \vee z' \neq v) \wedge \\ T(z', y) \wedge T(z, u) \wedge \neg T(z', u)\Big)\Big)\end{aligned}$$

- Dynamic Program: Collection of first-order formulas defining auxiliary relations after modifications

How much can change in one step?

- Arbitrary modifications
  - often used in View Maintenance (for example [Gupta et al. 1993])
  - framework for Dynamic Descriptive Complexity defined in [Weber and Schwentick 2007, Hesse and Immerman 2002]
  - but: if database changes arbitrarily, old information might be useless

How much can change in one step?

- Arbitrary modifications
  - often used in View Maintenance (for example [Gupta et al. 1993])
  - framework for Dynamic Descriptive Complexity defined in [Weber and Schwentick 2007, Hesse and Immerman 2002]
  - but: if database changes arbitrarily, old information might be useless
- Insertion or deletion of one tuple
  - usual setting in Dynamic Descriptive Complexity [Dong et al. 1995, Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012, Gelade et al. 2012, Zeume and Schwentick 2014, Datta et al. 2015]
  - sometimes: insertion and deletion of constantly many tuples
  - strong results: Reachability in directed graphs can be maintained with first-order update formulas [Datta et al. 2015]

How much can change in one step?

- Arbitrary modifications
  - often used in View Maintenance (for example [Gupta et al. 1993])
  - framework for Dynamic Descriptive Complexity defined in [Weber and Schwentick 2007, Hesse and Immerman 2002]
  - but: if database changes arbitrarily, old information might be useless
- Insertion or deletion of one tuple
  - usual setting in Dynamic Descriptive Complexity [Dong et al. 1995, Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012, Gelade et al. 2012, Zeume and Schwentick 2014, Datta et al. 2015]
  - sometimes: insertion and deletion of constantly many tuples
  - strong results: Reachability in directed graphs can be maintained with first-order update formulas [Datta et al. 2015]
- Insertion or deletion of sets with some structural property
  - appears in [Dong et al. 1995]: insertion of sets closed under some operation
  - in [Dong and Pang 1997]: Reachability under deletions of "anti-chains" of edges/nodes

First-order modifications

- In practice, modifications to databases affect many tuples

First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language

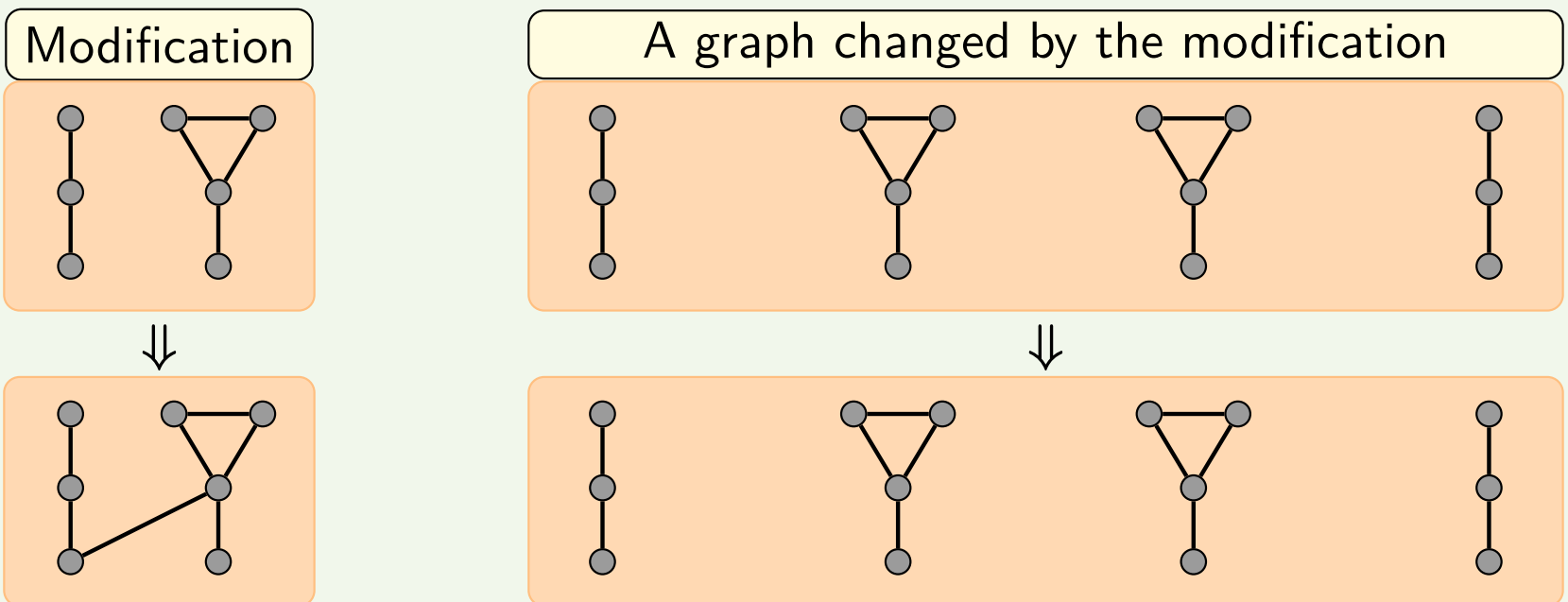
First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language
- Here: define modifications by first-order formulas
  - encouraged for example by [Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012]

First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language
- Here: define modifications by first-order formulas
  - encouraged for example by [Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012]

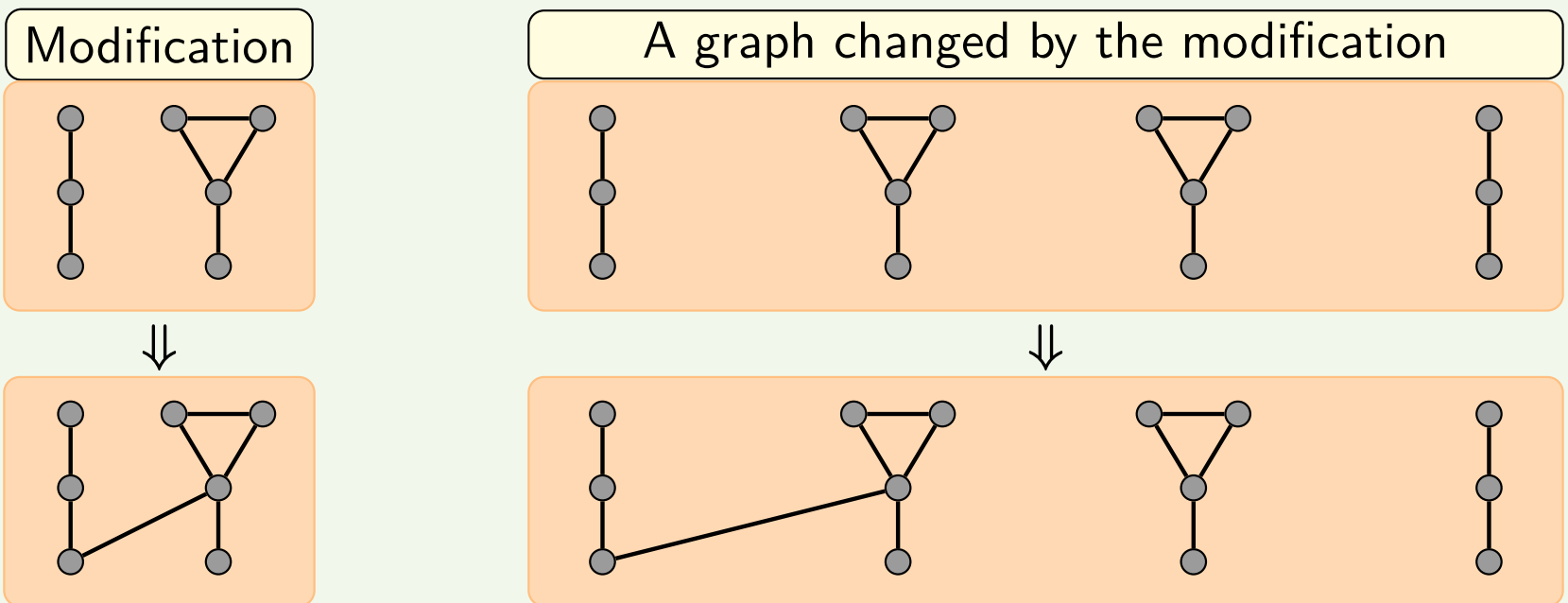
Example: modifying graphs



First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language
- Here: define modifications by first-order formulas
  - encouraged for example by [Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012]

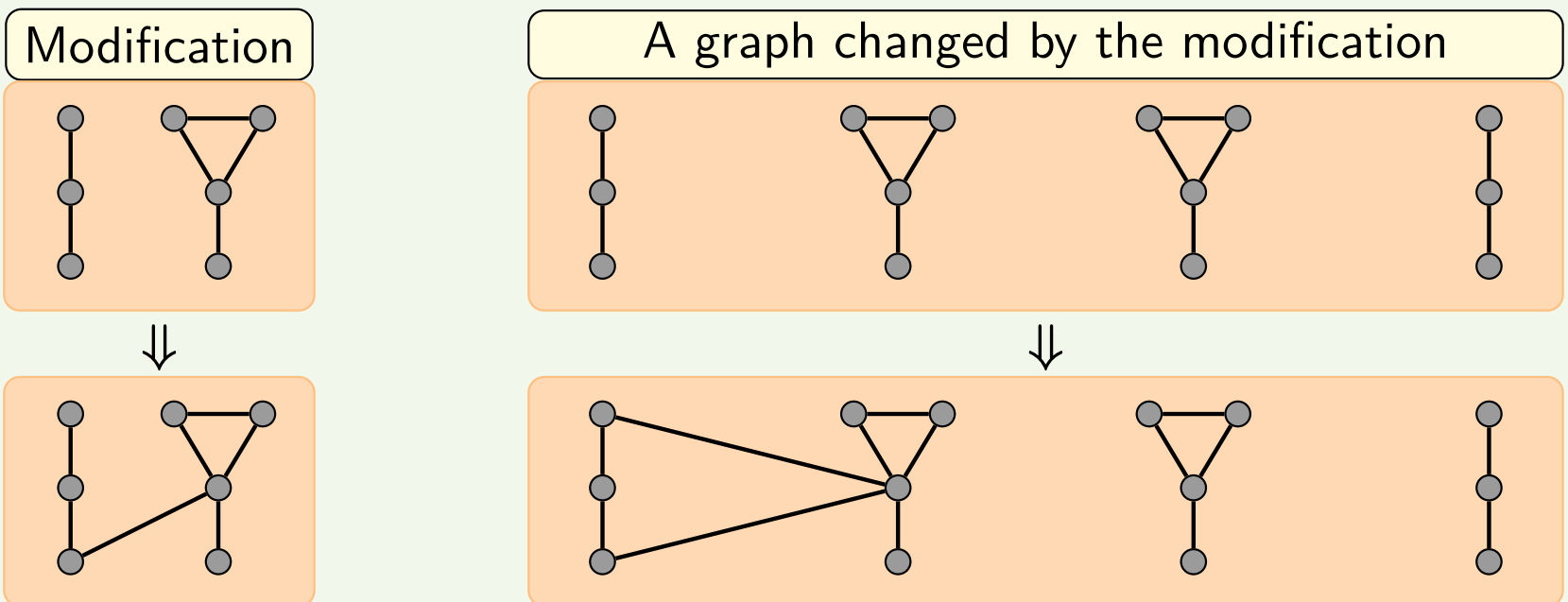
Example: modifying graphs



First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language
- Here: define modifications by first-order formulas
  - encouraged for example by [Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012]

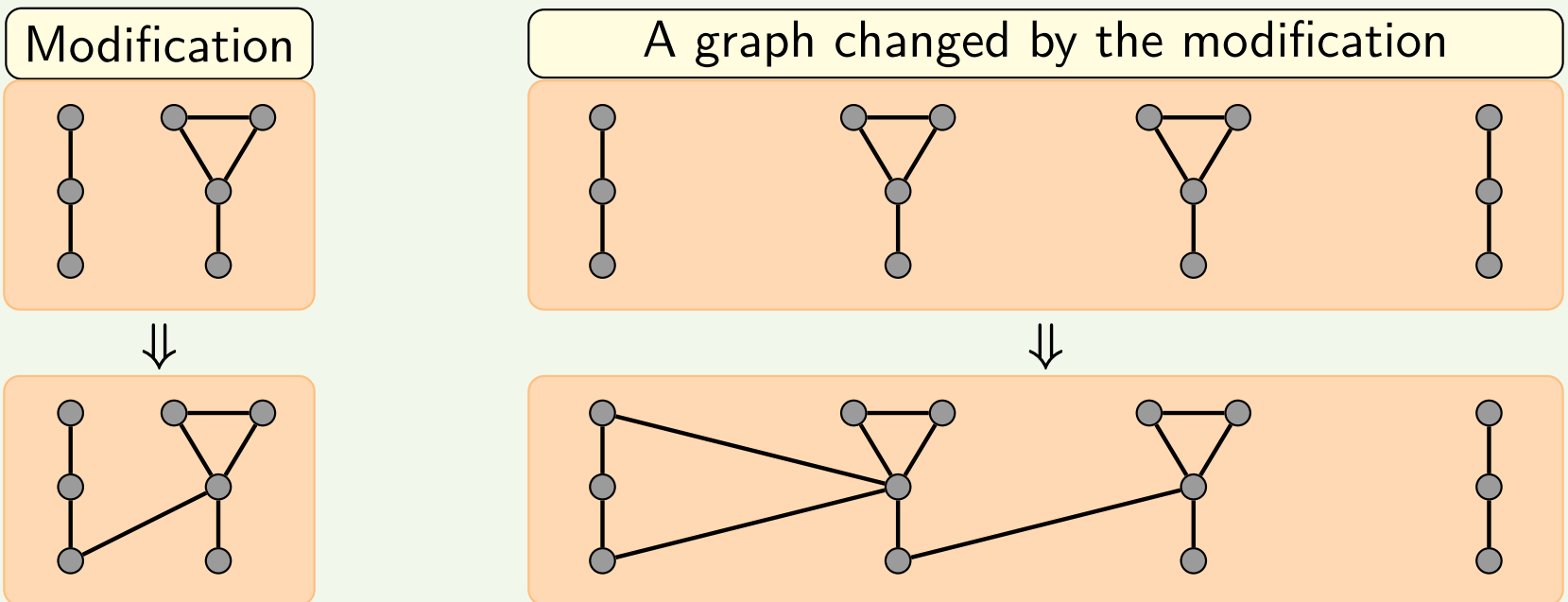
Example: modifying graphs



First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language
- Here: define modifications by first-order formulas
  - encouraged for example by [Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012]

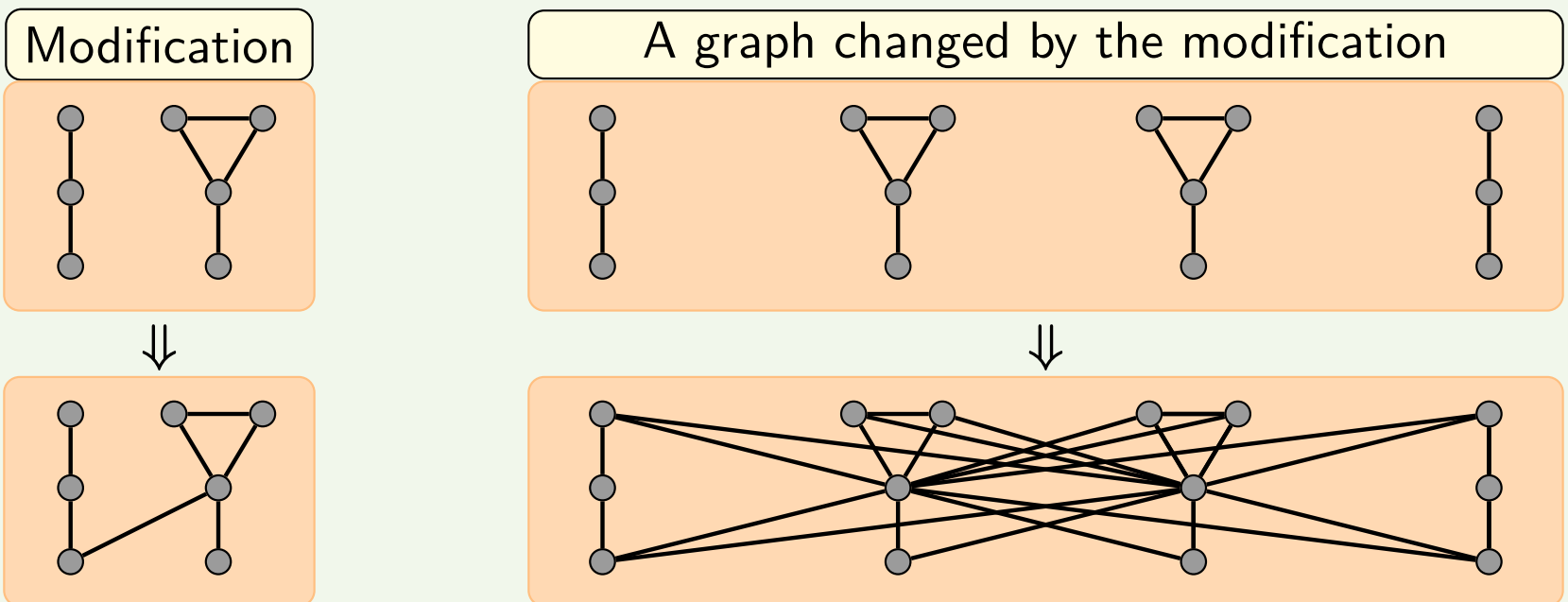
Example: modifying graphs



First-order modifications

- In practice, modifications to databases affect many tuples
- Often they are defined in some Data Manipulation Language
- Here: define modifications by first-order formulas
  - encouraged for example by [Patnaik and Immerman 1997, Etessami 1998, Grädel and Siebertz 2012]

Example: modifying graphs



Results

We can maintain with first-order update formulas:

Results

We can maintain with first-order update formulas:

Insertions

- Undirected Reachability under first-order insertions*
- Acyclic Reachability under quantifier-free first-order insertions

both: also with single edge deletions

*: with some technical restriction

Results

We can maintain with first-order update formulas:

Insertions

- Undirected Reachability under first-order insertions*
- Acyclic Reachability under quantifier-free first-order insertions

  both: also with single edge deletions

Modifications with restricted quantification

- Context-free languages under existential first-order modifications*

*: with some technical restriction

Results

We can maintain with first-order update formulas:

Insertions

- Undirected Reachability under first-order insertions*
- Acyclic Reachability under quantifier-free first-order insertions

  both: also with single edge deletions

Modifications with restricted quantification

- Context-free languages under existential first-order modifications*

*: with some technical restriction

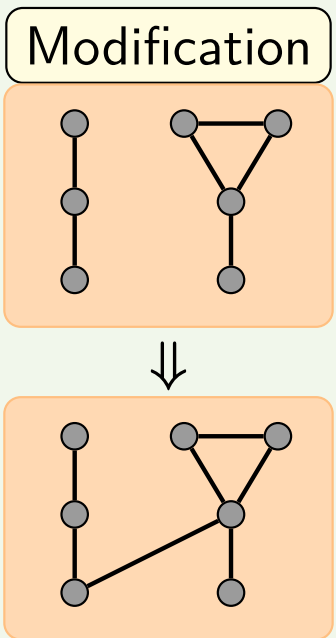
Parameter-free modifications

- All AC^1 queries on ordered databases under parameter-free first-order modifications

Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

Maintaining Reachability for the example modification

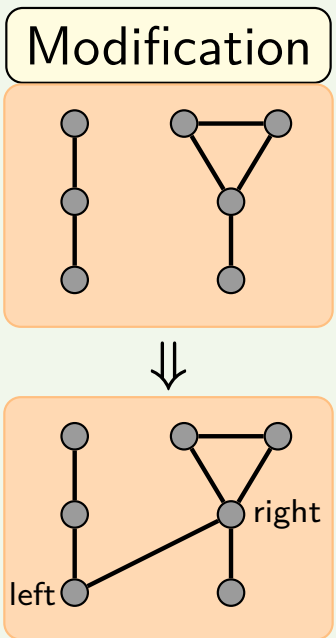


Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

Maintaining Reachability for the example modification

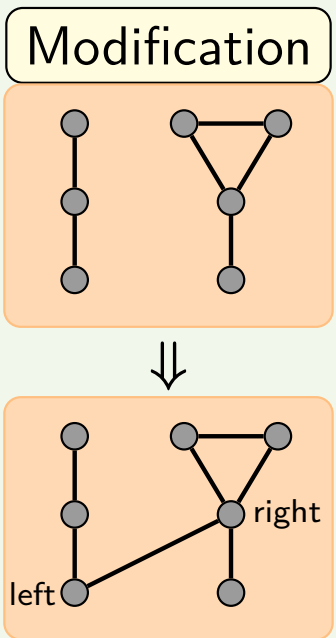
- Consider edges to be "directed" from "left" to "right"



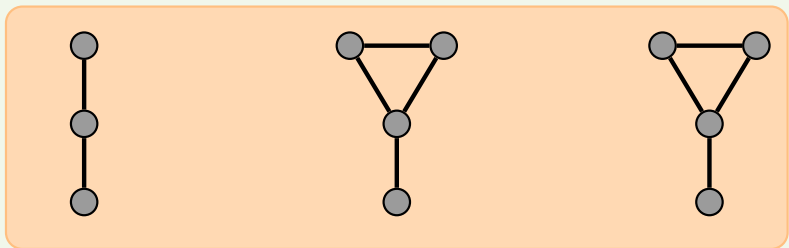
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

Maintaining Reachability for the example modification



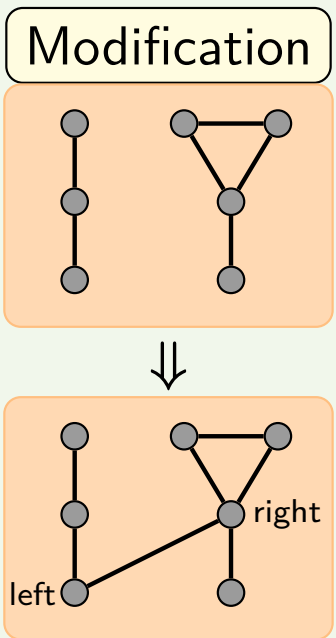
- Consider edges to be “directed” from “left” to “right”
- If two nodes are connected using two new edges “of the same direction”, there is a more direct new edge



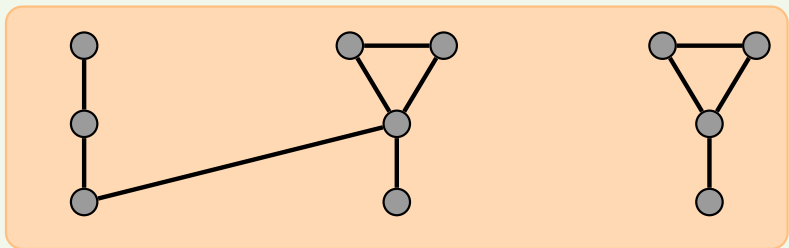
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

Maintaining Reachability for the example modification



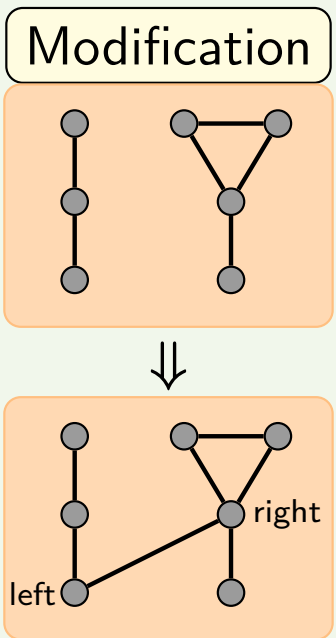
- Consider edges to be "directed" from "left" to "right"
- If two nodes are connected using two new edges "of the same direction", there is a more direct new edge



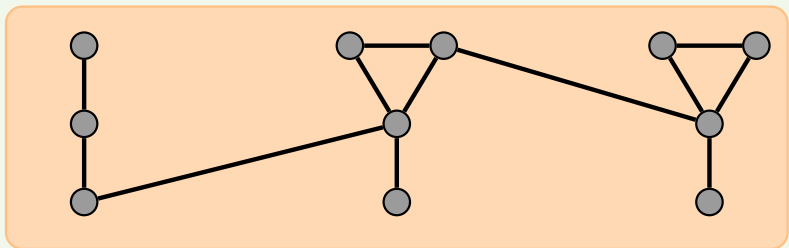
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

Maintaining Reachability for the example modification



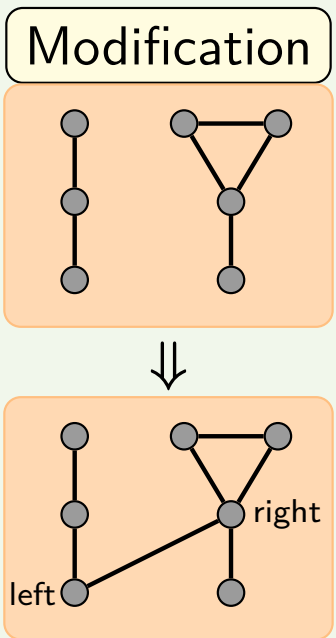
- Consider edges to be “directed” from “left” to “right”
- If two nodes are connected using two new edges “of the same direction”, there is a more direct new edge



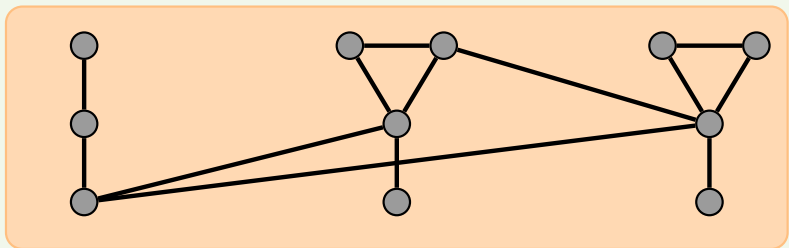
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

Maintaining Reachability for the example modification



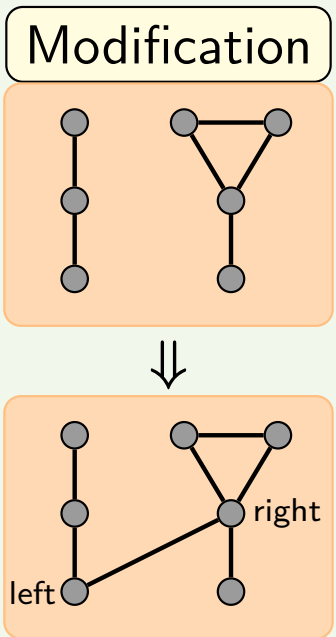
- Consider edges to be “directed” from “left” to “right”
- If two nodes are connected using two new edges “of the same direction”, there is a more direct new edge



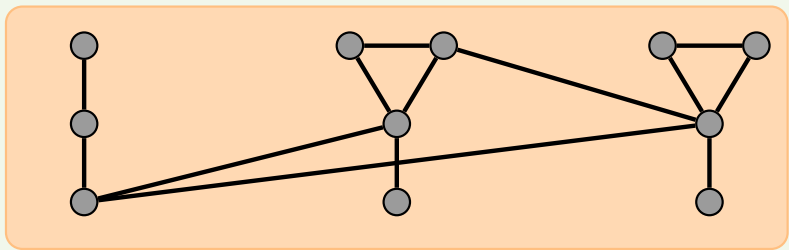
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

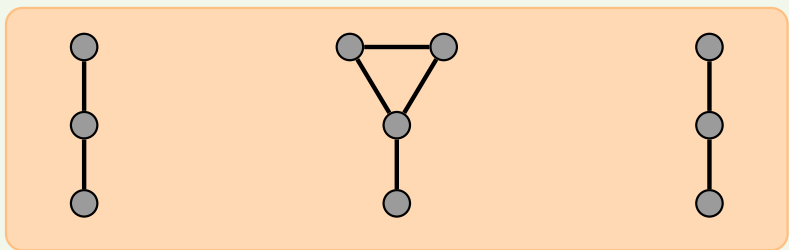
Maintaining Reachability for the example modification



- Consider edges to be "directed" from "left" to "right"
- If two nodes are connected using two new edges "of the same direction", there is a more direct new edge



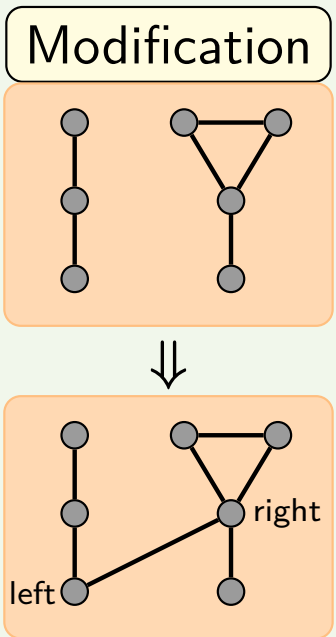
- Only two new edges are needed in every path



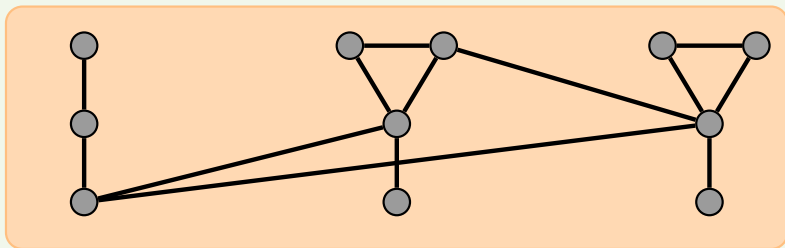
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

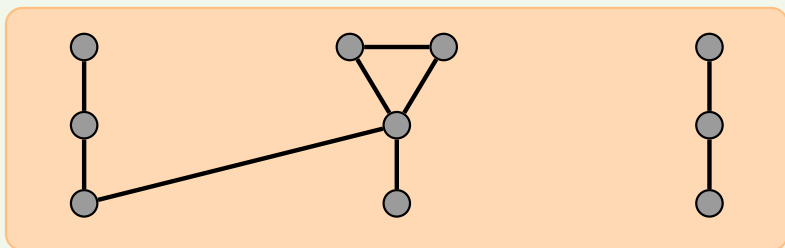
Maintaining Reachability for the example modification



- Consider edges to be "directed" from "left" to "right"
- If two nodes are connected using two new edges "of the same direction", there is a more direct new edge



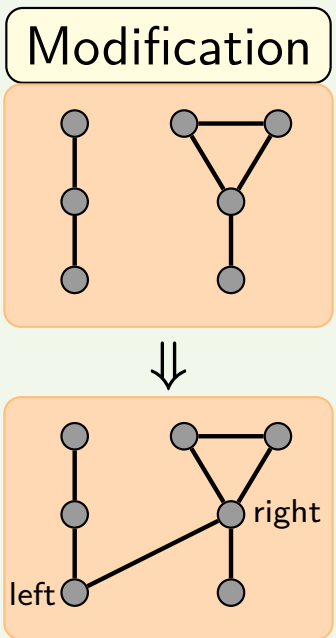
- Only two new edges are needed in every path



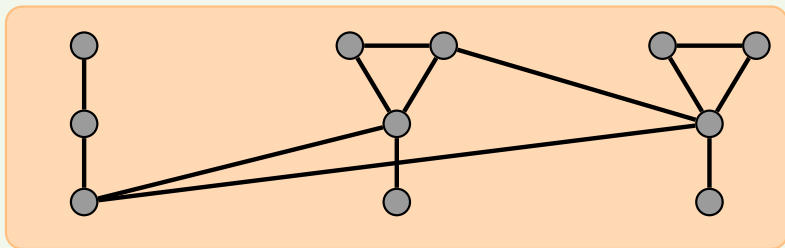
Maintaining Reachability in undirected graphs

- Proof idea: paths only need boundedly many new edges

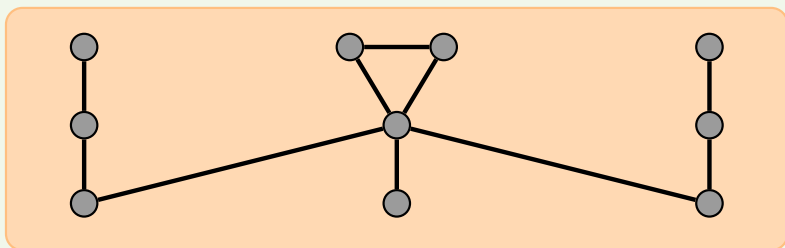
Maintaining Reachability for the example modification



- Consider edges to be "directed" from "left" to "right"
- If two nodes are connected using two new edges "of the same direction", there is a more direct new edge



- Only two new edges are needed in every path



Reachability under first-order deletions

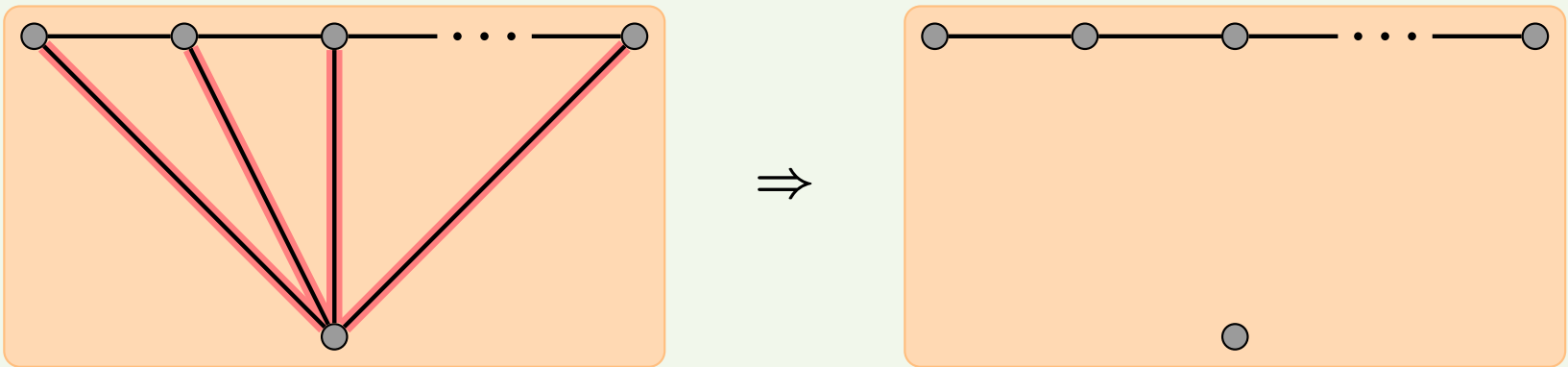
- For single edge deletions: maintain a spanning tree [Patnaik and Immerman 1997, Dong and Su 1998, Grädel and Siebertz 2012]

Reachability under first-order deletions

- For single edge deletions: maintain a spanning tree [Patnaik and Immerman 1997, Dong and Su 1998, Grädel and Siebertz 2012]

First-order deletions

How to maintain a spanning tree in the following situation?



Conclusion

- We study modifications of databases definable in (fragments) of first-order logic
  - Number of affected tuples is non-constant!

Conclusion

- We study modifications of databases definable in (fragments) of first-order logic
  - Number of affected tuples is non-constant!
- Interesting queries can be maintained under wide classes of modifications
  - Undirected Reachability under first-order insertions
  - Acyclic Reachability under quantifier-free insertions
  - Context-free languages under existential first-order modifications
  - All AC^1 queries on ordered databases under parameter-free first-order modifications

Conclusion

- We study modifications of databases definable in (fragments) of first-order logic
  - Number of affected tuples is non-constant!
- Interesting queries can be maintained under wide classes of modifications
  - Undirected Reachability under first-order insertions
  - Acyclic Reachability under quantifier-free insertions
  - Context-free languages under existential first-order modifications
  - All AC^1 queries on ordered databases under parameter-free first-order modifications
- For stronger modifications the techniques do not suffice

Conclusion

- We study modifications of databases definable in (fragments) of first-order logic
  - Number of affected tuples is non-constant!
- Interesting queries can be maintained under wide classes of modifications
  - Undirected Reachability under first-order insertions
  - Acyclic Reachability under quantifier-free insertions
  - Context-free languages under existential first-order modifications
  - All AC^1 queries on ordered databases under parameter-free first-order modifications
- For stronger modifications the techniques do not suffice
- To do: Show unmaintainability results for first-order modifications
  - Reachability under deletions
  - Reachability with quantifier-free update formulas

References I

Datta, S., Kulkarni, R., Mukherjee, A., Schwentick, T., and Zeume, T. (2015).

Reachability is in DynFO.

In *ICALP*, pages 159–170.

Dong, G. and Pang, C. (1997).

Maintaining transitive closure in first order after node-set and edge-set deletions.

Inf. Process. Lett., 62(4):193–199.

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.

References II

Dong, G., Su, J., and Topor, R. W. (1995).

Nonrecursive incremental evaluation of datalog queries.

Ann. Math. Artif. Intell., 14(2-4):187–223.

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.

Gelade, W., Marquardt, M., and Schwentick, T. (2012).

The dynamic complexity of formal languages.

ACM Trans. Comput. Log., 13(3):19.

References III

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.

Gupta, A., Mumick, I. S., and Subrahmanian, V. S. (1993).

Maintaining views incrementally.

In Buneman, P. and Jajodia, S., editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993.*, pages 157–166. ACM Press.

References IV

Hesse, W. and Immerman, N. (2002).

Complete problems for dynamic complexity classes.

In *17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings*, page 313. IEEE Computer Society.

Patnaik, S. and Immerman, N. (1997).

Dyn-FO: A parallel, dynamic complexity class.

J. Comput. Syst. Sci., 55(2):199–209.

Weber, V. and Schwentick, T. (2007).

Dynamic complexity theory revisited.

Theory Comput. Syst., 40(4):355–377.

References V

Zeume, T. and Schwentick, T. (2014).

Dynamic conjunctive queries.

In Schweikardt, N., Christophides, V., and Leroy, V., editors, *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014.*, pages 38–49. OpenProceedings.org.