Dynamic Conjunctive Queries

Thomas Schwentick und Thomas Zeume

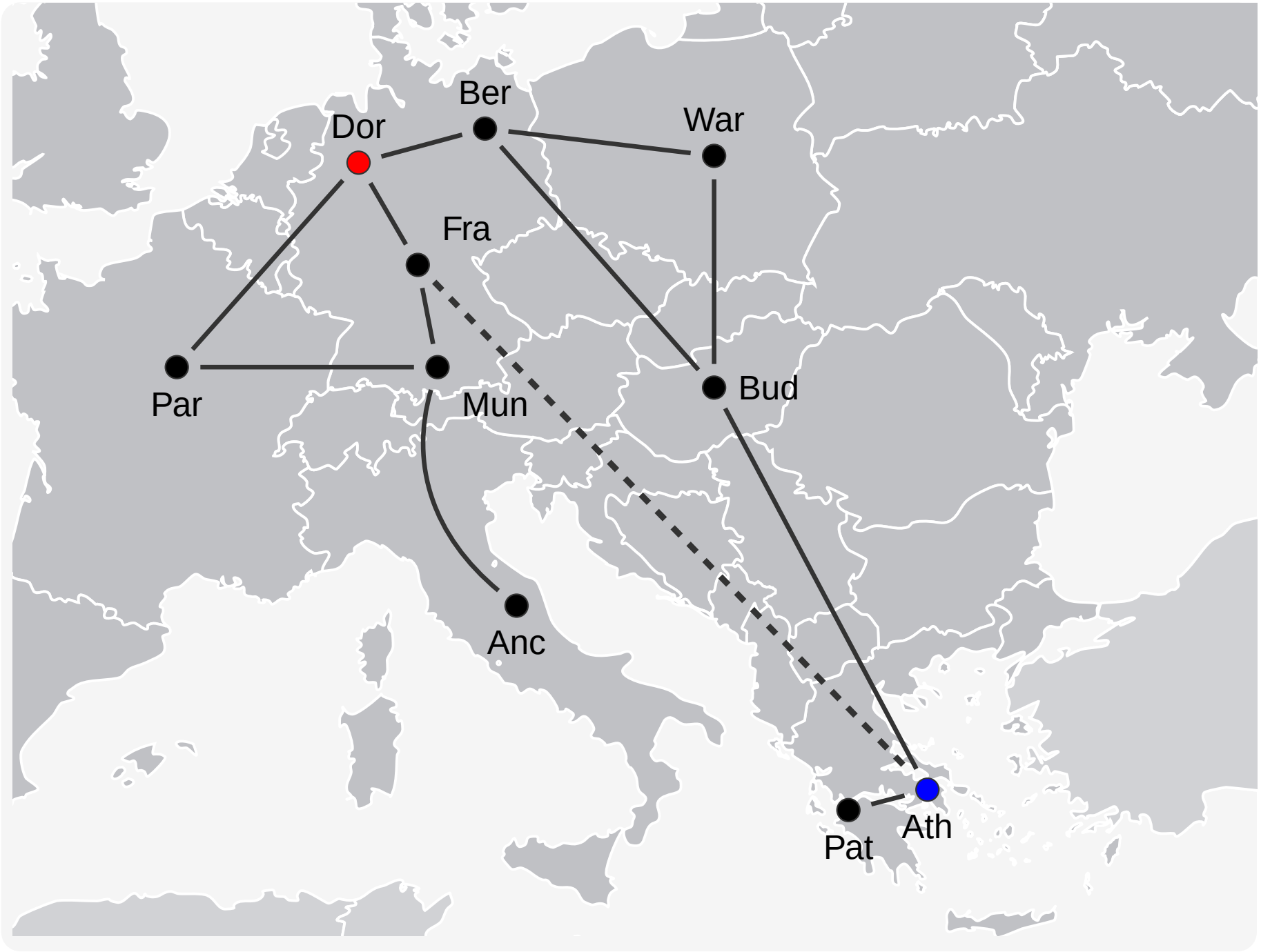
ICDT

Athens, March 2014

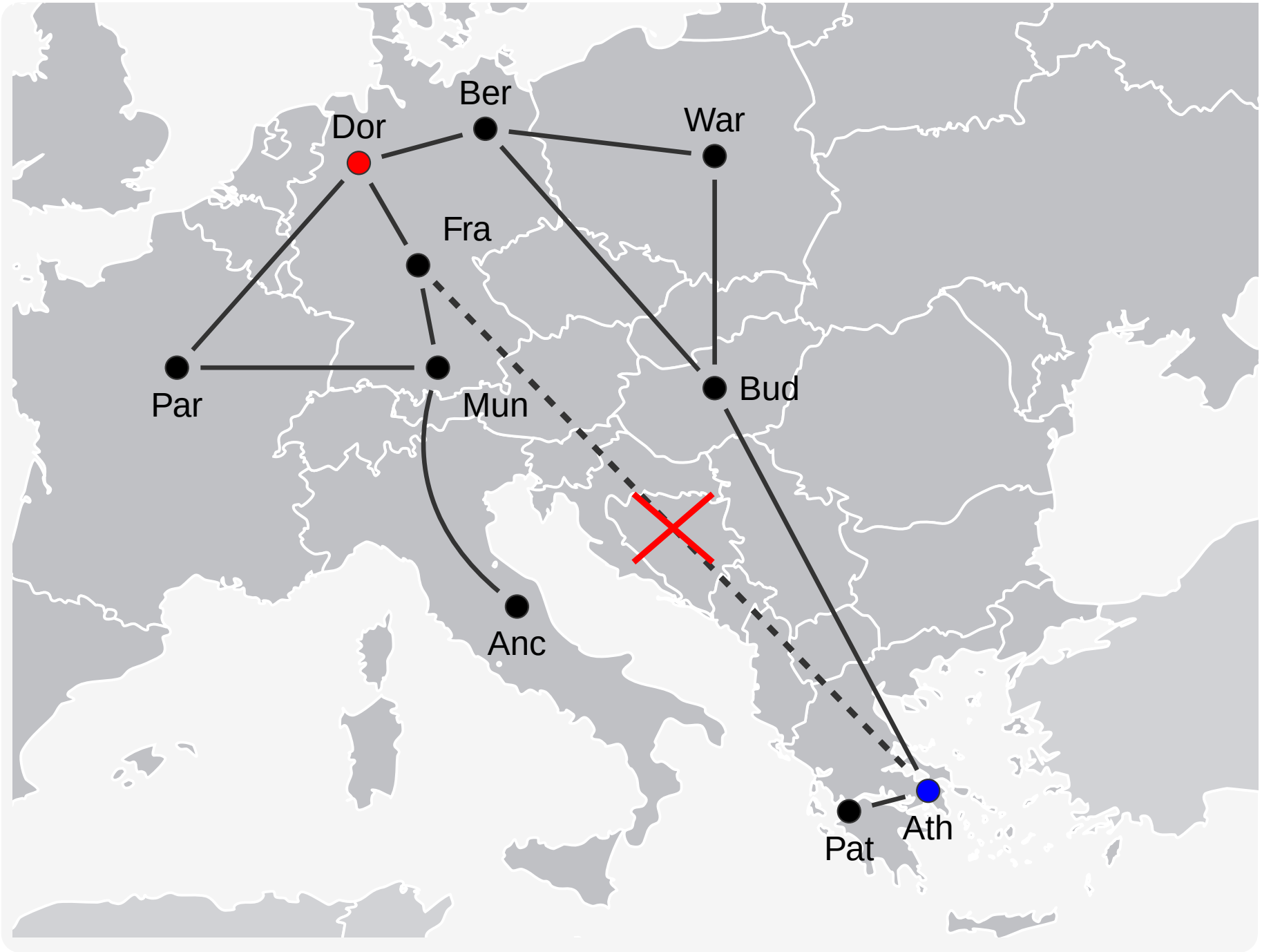
technische universität dortmund

Lehrstuhl Logik in der Informatik

How to go to Athens...



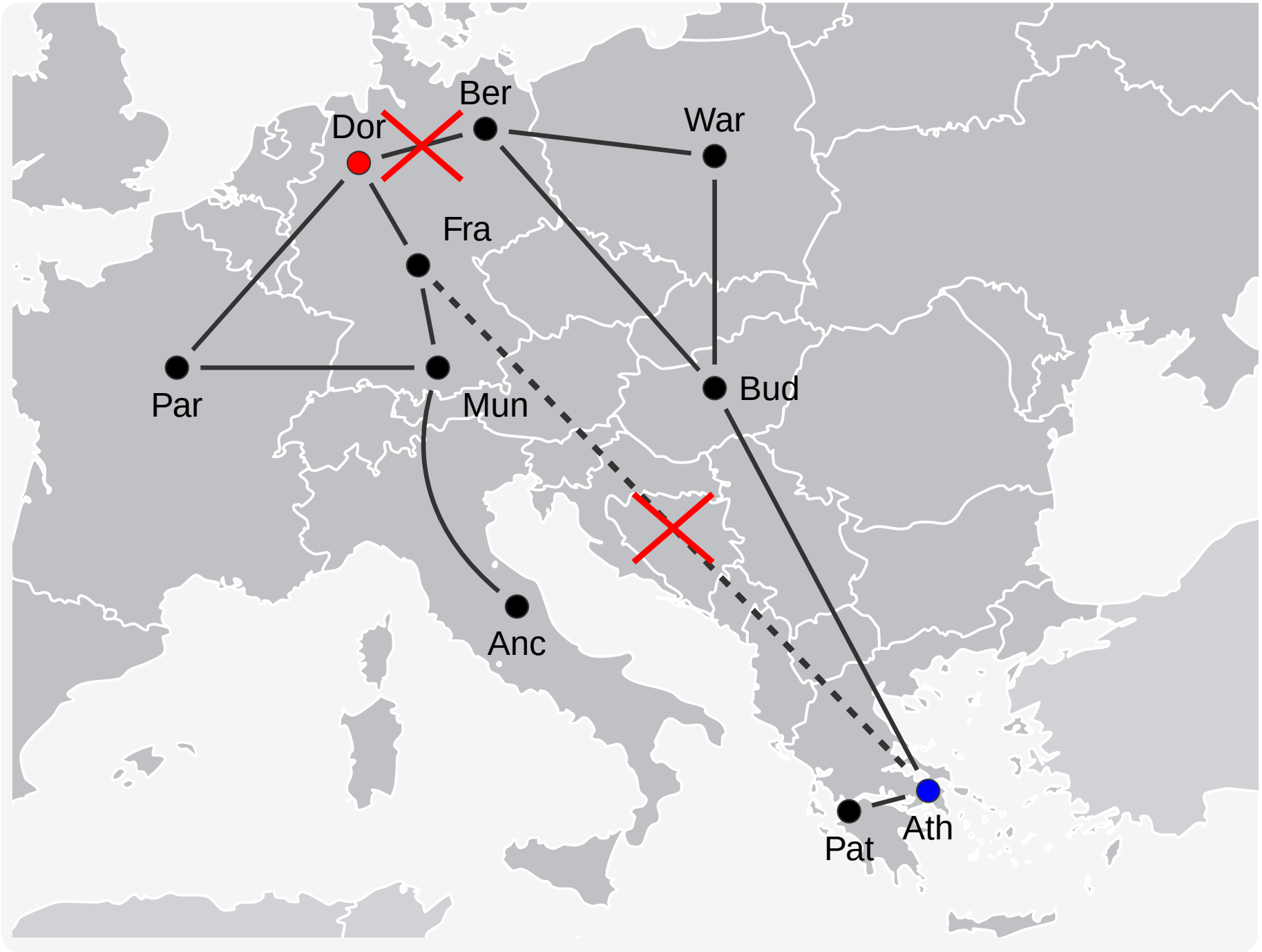
How to go to Athens...



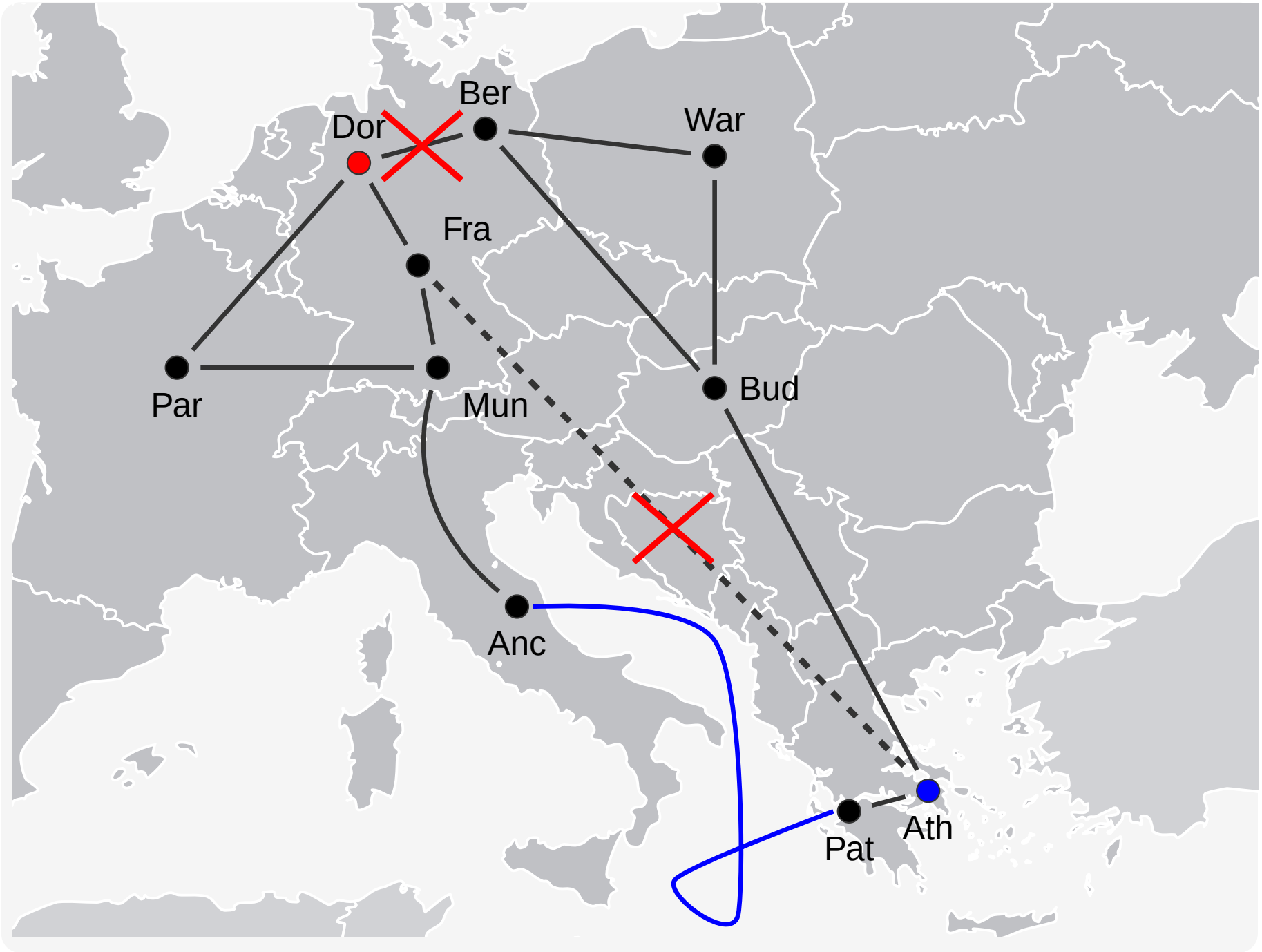
How to go to Athens...

©*Wirtschaftswoche*

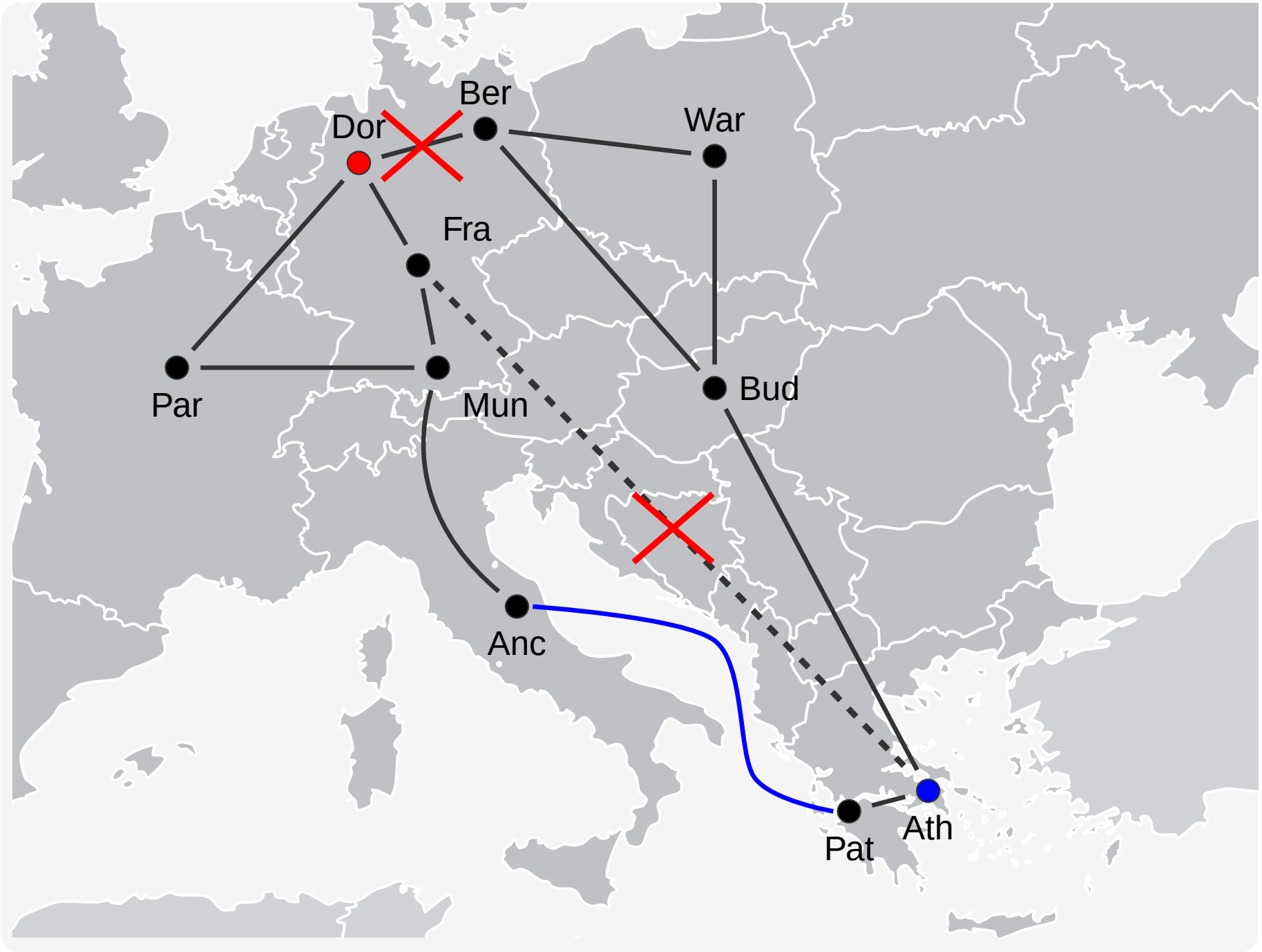
How to go to Athens...



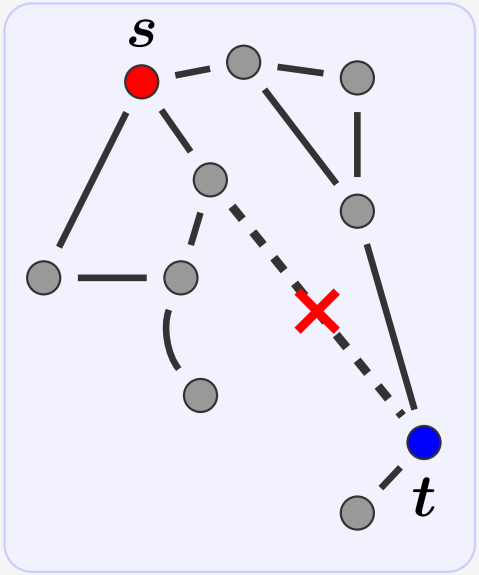
How to go to Athens...



How to go to Athens...

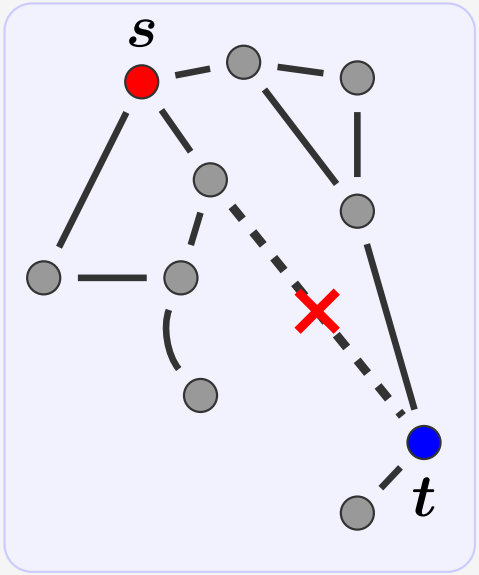


The Dynamic Approach

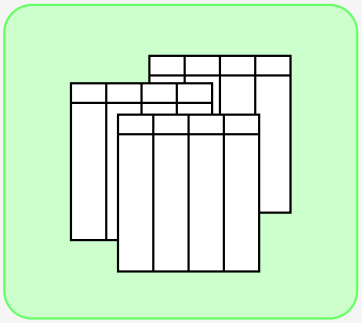


Input DB

The Dynamic Approach



Input DB

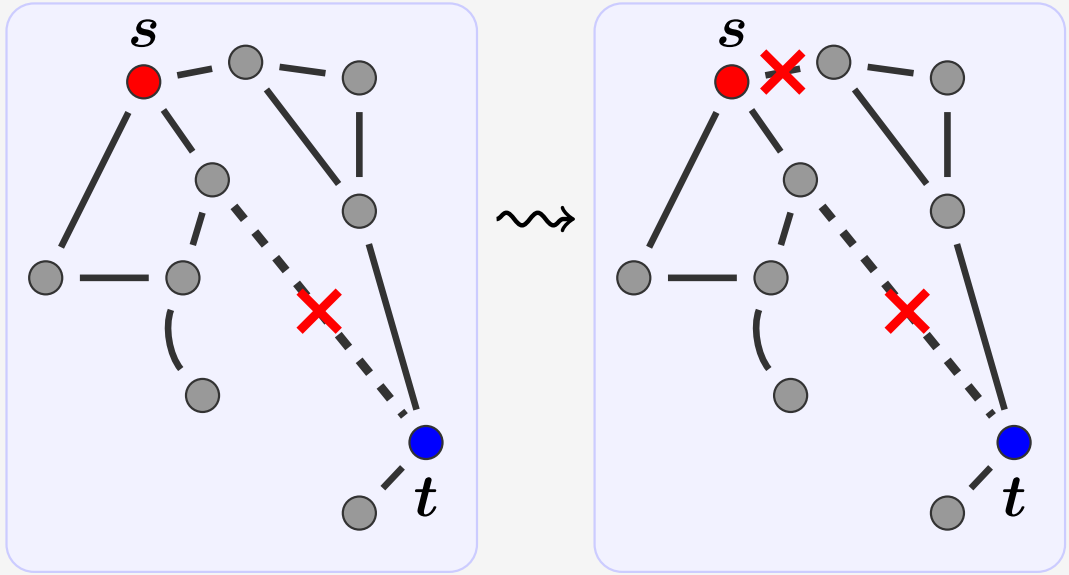


Auxiliary DB

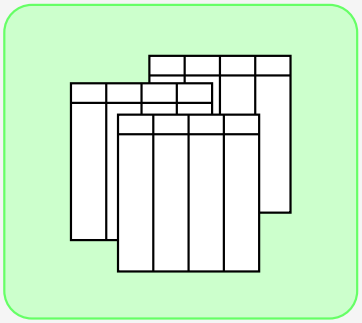
Reach(s,t)?

yes

The Dynamic Approach



Input DB

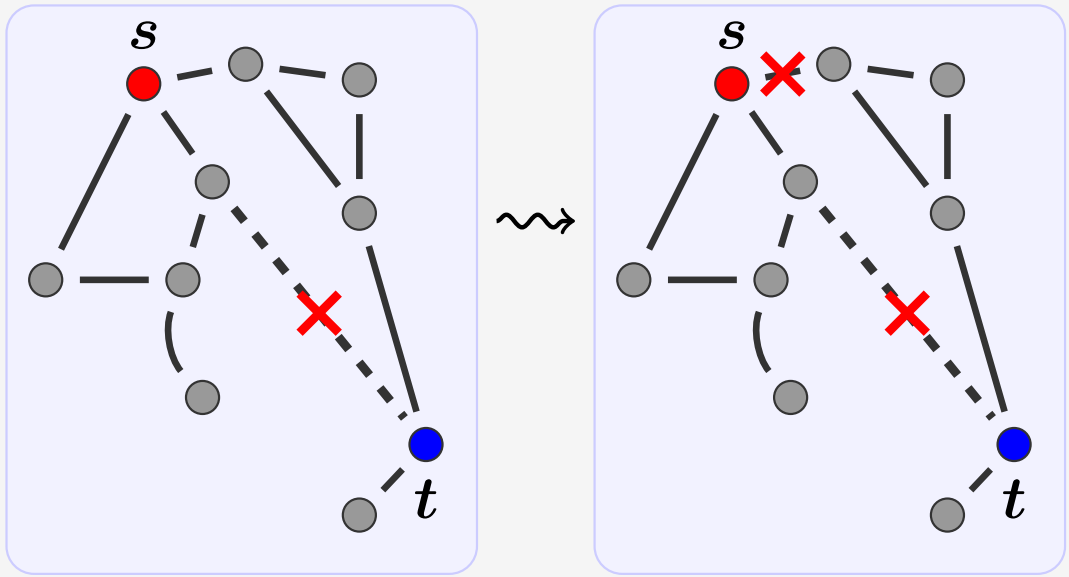


Auxiliary DB

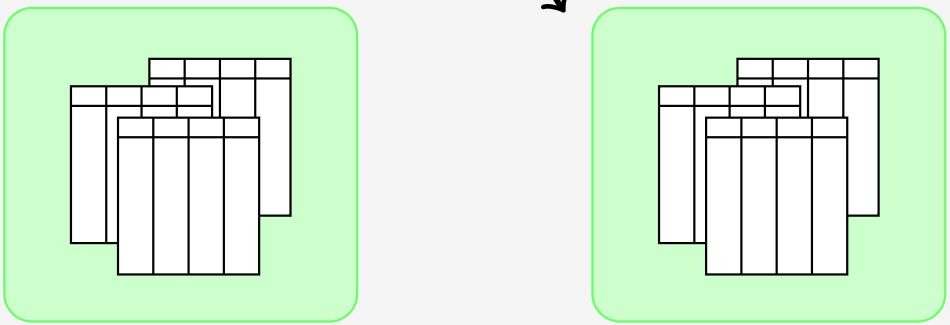
Reach(s,t)?

yes

The Dynamic Approach



Input DB



Auxiliary DB

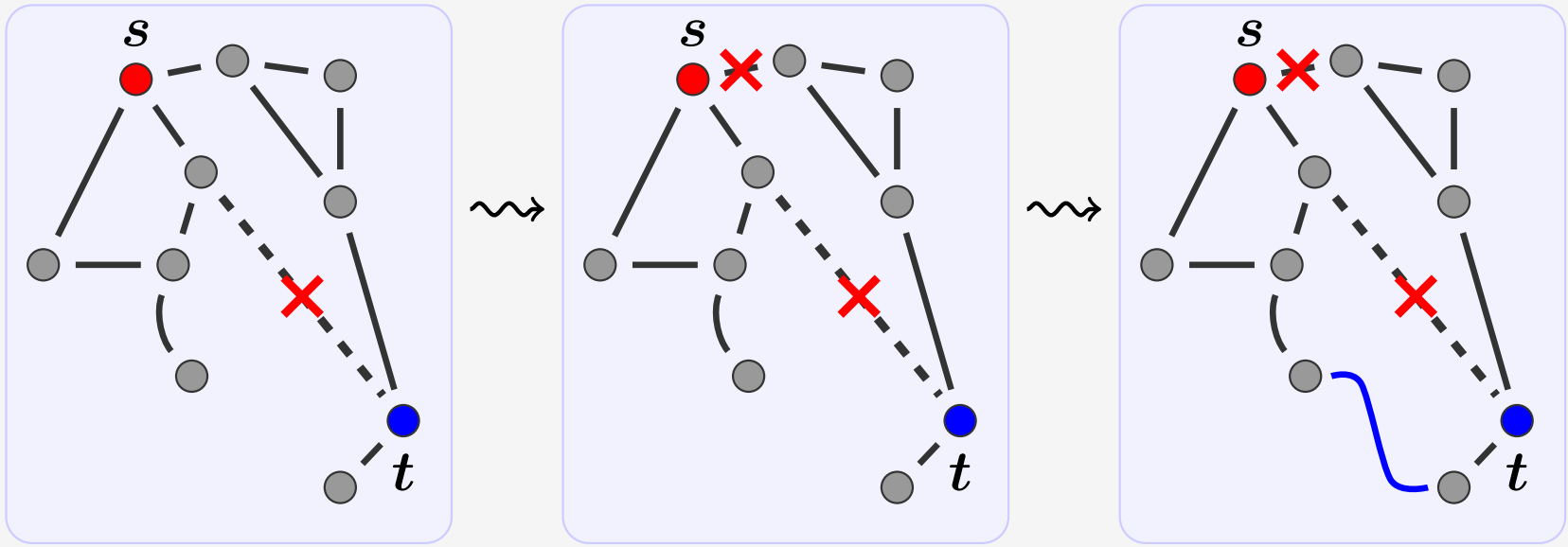
Reach(s,t)?

yes

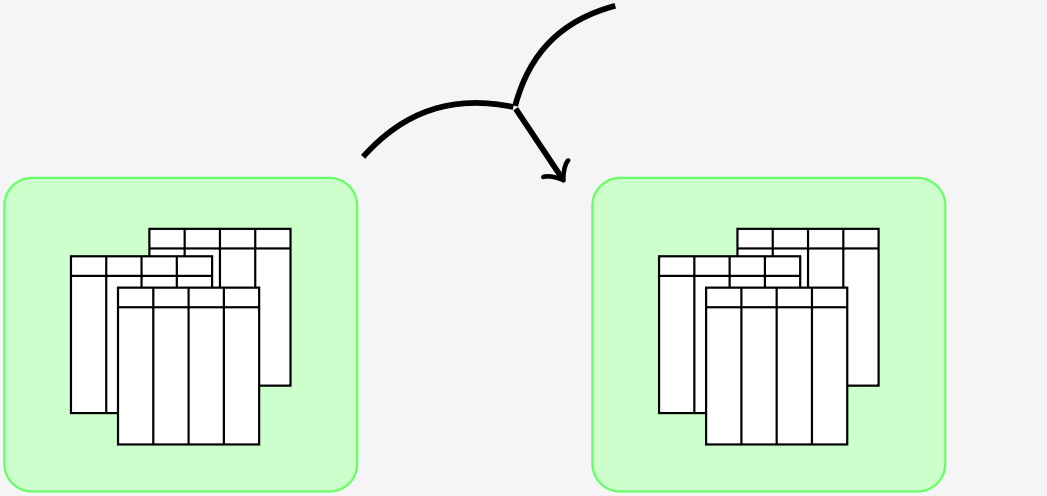
Reach(s,t)?

no

The Dynamic Approach



Input DB



Auxiliary DB

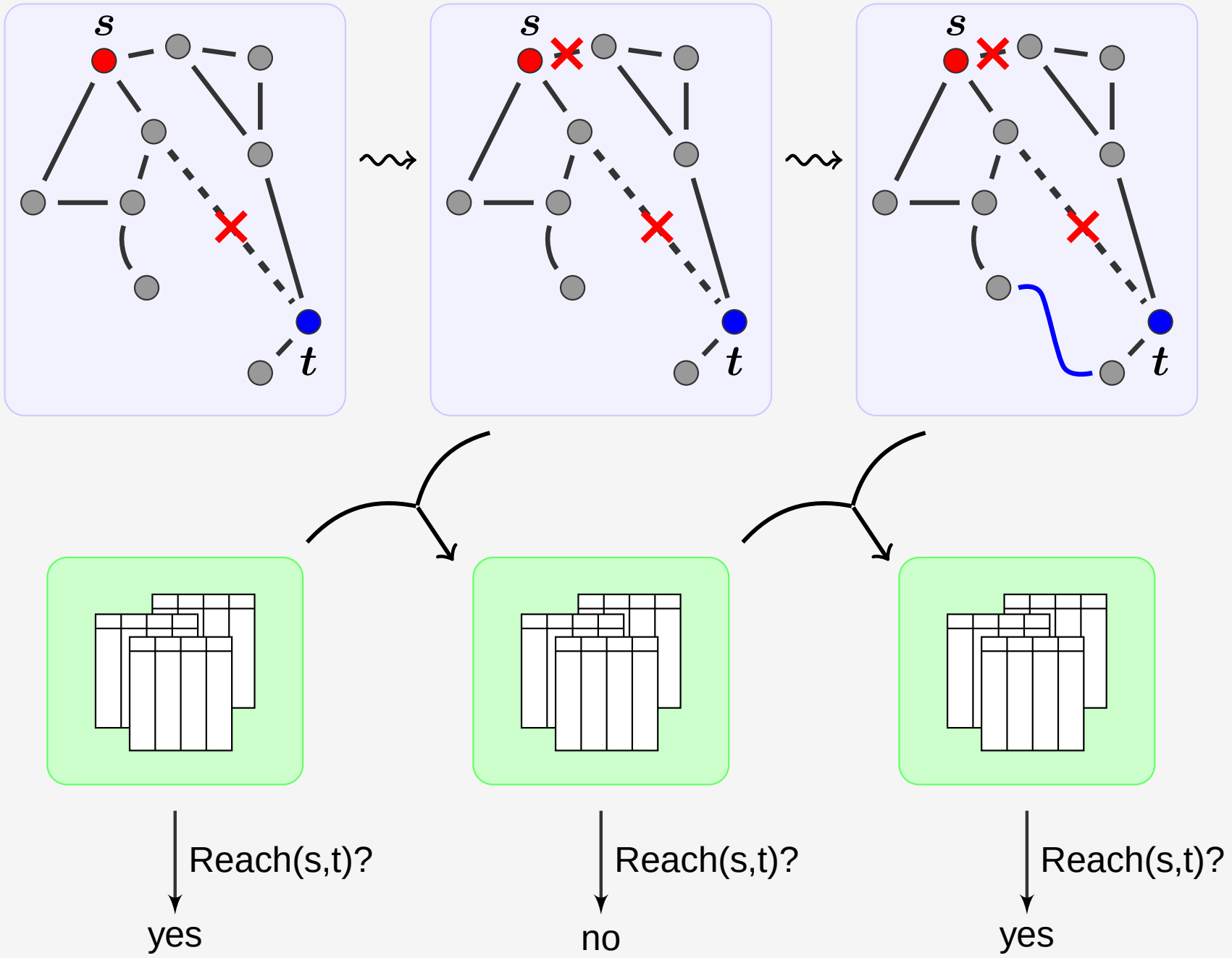
Reach(s,t)?

yes

Reach(s,t)?

no

The Dynamic Approach



Contents

Introduction

▷ **The Framework of Patnaik and Immerman**

Dynamic Conjunctive Queries

A Note on Semantics

Conclusion and Future Work

Framework of Patnaik and Immerman

Definition: Dynamic program

A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Framework of Patnaik and Immerman

Definition: Dynamic program

A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Framework of Patnaik and Immerman

Definition: Dynamic program

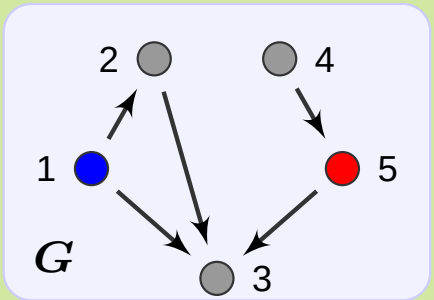
A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Example: Reachability under insertions



Framework of Patnaik and Immerman

Definition: Dynamic program

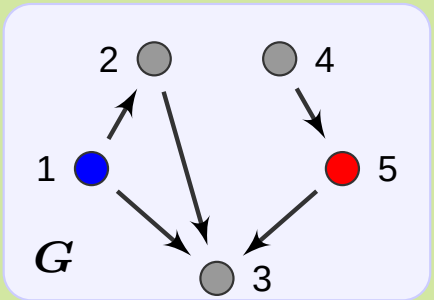
A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Example: Reachability under insertions



- Intention: Binary auxiliary relation T stores transitive closure

Framework of Patnaik and Immerman

Definition: Dynamic program

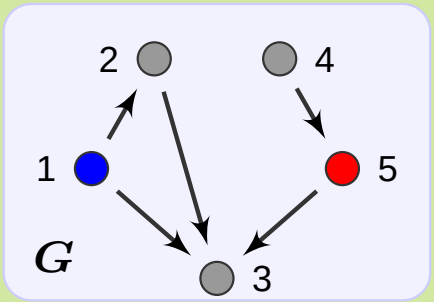
A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

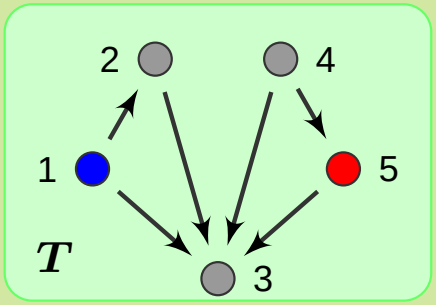
- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Example: Reachability under insertions





- Intention: Binary auxiliary relation T stores transitive closure
  - Initialization: $\textsc{Init}(T) = TC(G)$

Framework of Patnaik and Immerman

Definition: Dynamic program

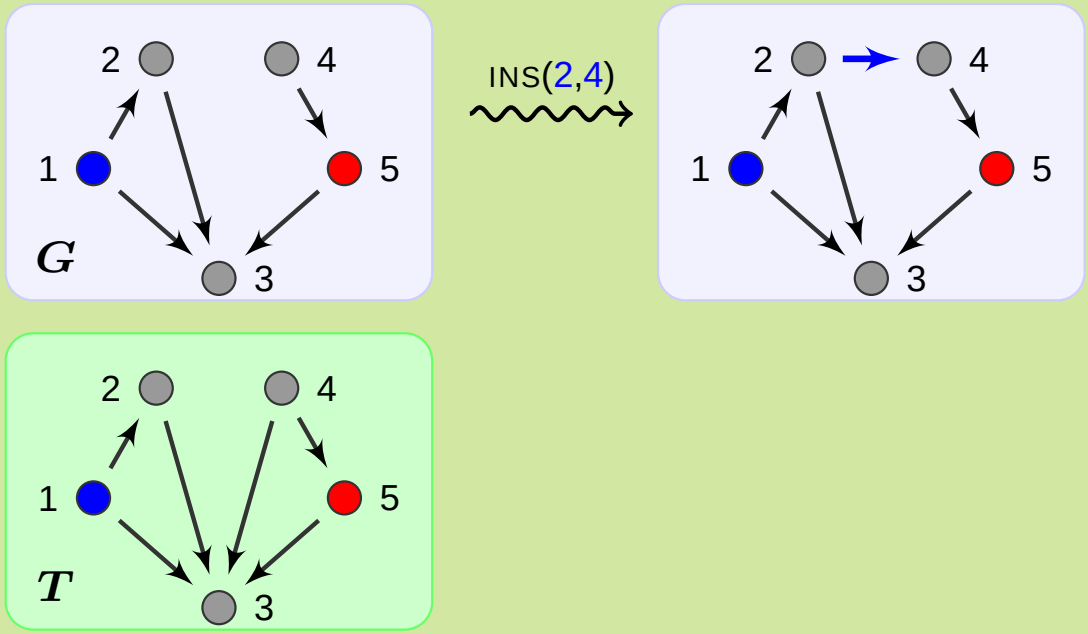
A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Example: Reachability under insertions



- Intention: Binary auxiliary relation T stores transitive closure
  - Initialization: $\textsc{Init}(T) = TC(G)$

Framework of Patnaik and Immerman

Definition: Dynamic program

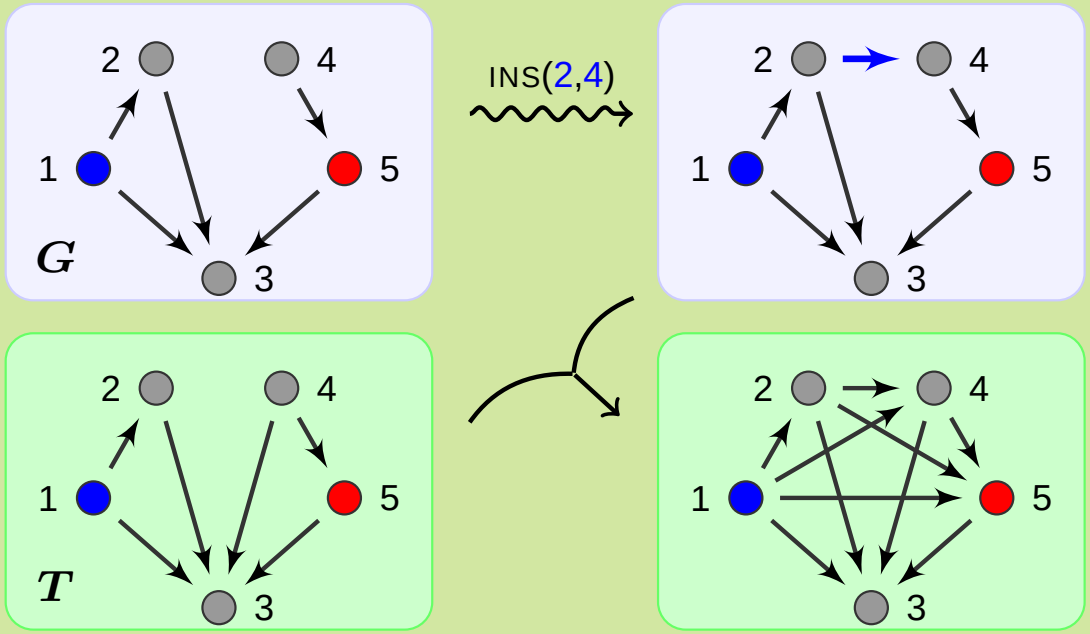
A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Example: Reachability under insertions



- Intention: Binary auxiliary relation T stores transitive closure
  - Initialization: $\textsc{Init}(T) = TC(G)$
  - Update formula $\phi^T_{\text{INS}}(u, v; x, y)$:

$$T(x, y) \vee (T(x, u) \wedge T(v, y))$$

Framework of Patnaik and Immerman

Definition: Dynamic program

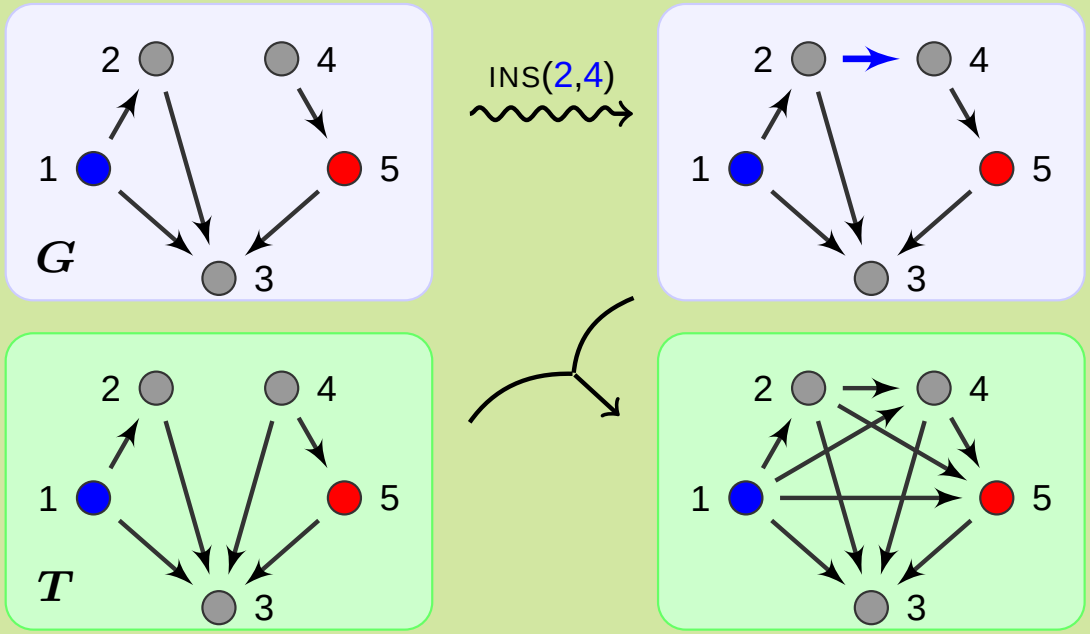
A dynamic program $\mathcal{P} = (\textsc{Init}, \Phi)$ describes how to

- initialize the auxiliary relations via initialization mapping $\textsc{Init}$
- update the auxiliary relations via first-order update formulas Φ

Definition: Query maintenance

$\mathcal{P}$ maintains a query if one of its auxiliary relations always stores the query result (i.e. after every possible modification to the input database)

Example: Reachability under insertions



- Intention: Binary auxiliary relation T stores transitive closure
  - Initialization: $\textsc{Init}(T) = TC(G)$
  - Update formula $\phi^T_{\text{INS}}(u, v; x, y)$:

$$T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- In this talk: Updates with fragments of first-order logic

Choosing a Precise Setting

- Is the size of the domain fixed or variable?
  - Here: Fixed domain (Patnaik/Immerman)

Choosing a Precise Setting

- Is the size of the domain fixed or variable?
  - Here: Fixed domain (Patnaik/Immerman)
- What is the initial database? How is the auxiliary data initialized?

Aux-Initialization	Arbitrary initial structure	Empty initial structure
Arbitrary	Here	X
PolyTime	Weber/Schwentick	Patnaik/Immerman **DynFO**$^+$
Logical	Grädel/Siebertz	
FO	(only for properties in **FO**)	(e.g., **DynFO**)
$\mathcal{C}$	X	Patnaik/Immerman **DynFO**
empty	X	

Choosing a Precise Setting

- Is the size of the domain fixed or variable?
  - Here: Fixed domain (Patnaik/Immerman)
- What is the initial database? How is the auxiliary data initialized?

Aux-Initialization	Arbitrary initial structure	Empty initial structure
Arbitrary	Here	X
PolyTime	Weber/Schwentick	Patnaik/Immerman **DynFO**$^+$
Logical	Grädel/Siebertz	
FO	(only for properties in **FO**)	(e.g., **DynFO**)
$\mathcal{C}$	X	Patnaik/Immerman **DynFO**
empty	X	

- Choices motivated by long term goals
  - Understanding dynamics (as opposed to the influence of the initialization)
  - Lower bounds for dynamic complexity classes

Choosing a Precise Setting

- Is the size of the domain fixed or variable?
  - Here: Fixed domain (Patnaik/Immerman)
- What is the initial database? How is the auxiliary data initialized?

Aux-Initialization	Arbitrary initial structure	Empty initial structure
Arbitrary	Here	X
PolyTime	Weber/Schwentick	Patnaik/Immerman **DynFO**$^+$
Logical	Grädel/Siebertz	
FO	(only for properties in **FO**)	(e.g., **DynFO**)
$\mathcal{C}$	X	Patnaik/Immerman **DynFO**
empty	X	

- Choices motivated by long term goals
  - Understanding dynamics (as opposed to the influence of the initialization)
  - Lower bounds for dynamic complexity classes
- Our results hold for **FO**-initialization as well

Previous Work: An Overview

DynFO (First-order update formulas)

- reachability in acyclic and undirected graphs [Patnaik et al. ’94]
- context-free languages [Gelade et al. ’09]

Previous Work: An Overview

DynTC^0 (**FO** with Counting)

- reachability in directed graphs [Hesse '01]

DynFO (First-order update formulas)

- reachability in acyclic and undirected graphs [Patnaik et al. '94]
- context-free languages [Gelade et al. '09]

Previous Work: An Overview

DynTC^0 (**FO** with Counting)

- reachability in directed graphs [Hesse '01]

DynFO (First-order update formulas)

- reachability in acyclic and undirected graphs [Patnaik et al. '94]
- context-free languages [Gelade et al. '09]

DynProp (**FO** without Quantifiers)

- reachability in deterministic graphs [Hesse '03]
- regular languages [Gelade et al. '09]

Previous Work: An Overview

DynTC^0 (**FO** with Counting)

- reachability in directed graphs [Hesse '01]

DynFO (First-order update formulas)

- reachability in acyclic and undirected graphs [Patnaik et al. '94]
- context-free languages [Gelade et al. '09]

DynProp (**FO** without Quantifiers)

- reachability in deterministic graphs [Hesse '03]
- regular languages [Gelade et al. '09]

- Known lower bounds
  - ALT-REACH $\notin$ **DynProp** [Gelade et al. '09]
  - non-regular languages $\notin$ **DynProp** [Gelade et al. '09]
  - REACH $\notin$ unary **DynFO** [Dong and Su '98]
  - EQUALCARDINALITY $\notin$ **DynFO** with invariant initialization [Grädel et al. '12]
  - (+ similar results)

Previous Work: An Overview

DynTC^0 (**FO** with Counting)

- reachability in directed graphs [Hesse '01]

DynFO (First-order update formulas)

- reachability in acyclic and undirected graphs [Patnaik et al. '94]
- context-free languages [Gelade et al. '09]

?

DynProp (**FO** without Quantifiers)

- reachability in deterministic graphs [Hesse '03]
- regular languages [Gelade et al. '09]

- Known lower bounds
  - ALT-REACH $\notin$ **DynProp** [Gelade et al. '09]
  - non-regular languages $\notin$ **DynProp** [Gelade et al. '09]
  - REACH $\notin$ unary **DynFO** [Dong and Su '98]
  - EQUALCARDINALITY $\notin$ **DynFO** with invariant initialization [Grädel et al. '12]
  - (+ similar results)
- Question: What about other fragments of first-order logic?
  - Recall: (core) SQL $\approx$ first-order logic
- Goal of this talk: Typical restrictions of (core) SQL as update languages

Conjunctive Queries and their Extensions

- Most intensively studied query language: Conjunctive queries (**CQ**)

Example

$$\varphi(x) = \exists y_1 \exists y_2 \big(T(x, y_1) \wedge T(y_1, y_2)\big)$$

- General form:

$$\varphi(\vec{x}) = \exists \vec{y} \psi(\vec{x}, \vec{y})$$

where ψ is a conjunction of atoms

Conjunctive Queries and their Extensions

- Most intensively studied query language: Conjunctive queries (**CQ**)

Example

$$\varphi(x) = \exists y_1 \exists y_2 \big(T(x, y_1) \wedge T(y_1, y_2)\big)$$

- General form:

$$\varphi(\vec{x}) = \exists \vec{y} \psi(\vec{x}, \vec{y})$$

  where ψ is a conjunction of atoms

- Equivalent characterizations:
  - select-project-join queries in relational algebra
  - select-from-where queries in SQL

Conjunctive Queries and their Extensions

- Most intensively studied query language: Conjunctive queries (**CQ**)

Example

$$\varphi(x) = \exists y_1 \exists y_2 \big(T(x, y_1) \wedge T(y_1, y_2)\big)$$

- General form:

$$\varphi(\vec{x}) = \exists \vec{y} \psi(\vec{x}, \vec{y})$$

  where ψ is a conjunction of atoms

- Equivalent characterizations:
  - select-project-join queries in relational algebra
  - select-from-where queries in SQL
- Extensions
  - **UCQ**: Unions of **CQ**s

Conjunctive Queries and their Extensions

- Most intensively studied query language: Conjunctive queries (**CQ**)

Example

$$\varphi(x) = \exists y_1 \exists y_2 \big(T(x, y_1) \wedge T(y_1, y_2)\big)$$

- General form:

$$\varphi(\vec{x}) = \exists \vec{y} \psi(\vec{x}, \vec{y})$$

  where ψ is a conjunction of atoms

- Equivalent characterizations:
  - select-project-join queries in relational algebra
  - select-from-where queries in SQL
- Extensions
  - **UCQ**: Unions of **CQ**s
  - **CQ**$^\neg$: Same as **CQ**s but negated atoms are allowed
  - **UCQ**$^\neg$: Unions of **CQ**$^\neg$s

Conjunctive Queries and their Extensions

- Most intensively studied query language: Conjunctive queries (**CQ**)

Example

$$\varphi(x) = \exists y_1 \exists y_2 \big(T(x, y_1) \wedge T(y_1, y_2)\big)$$

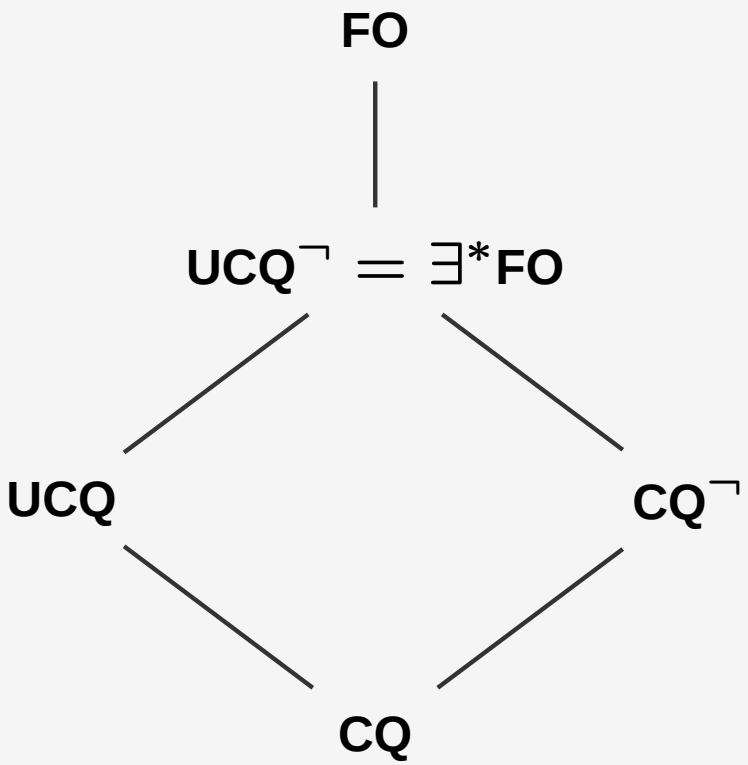
- General form:

$$\varphi(\vec{x}) = \exists \vec{y} \psi(\vec{x}, \vec{y})$$

  where ψ is a conjunction of atoms

- Equivalent characterizations:
  - select-project-join queries in relational algebra
  - select-from-where queries in SQL
- Extensions
  - **UCQ**: Unions of **CQ**s
  - **CQ**$^\neg$: Same as **CQ**s but negated atoms are allowed
  - **UCQ**$^\neg$: Unions of **CQ**$^\neg$s

- Known: Variants are pairwise different



Conjunctive Queries and their Extensions

- Most intensively studied query language: Conjunctive queries (**CQ**)

Example

$$\varphi(x) = \exists y_1 \exists y_2 \big(T(x, y_1) \wedge T(y_1, y_2)\big)$$

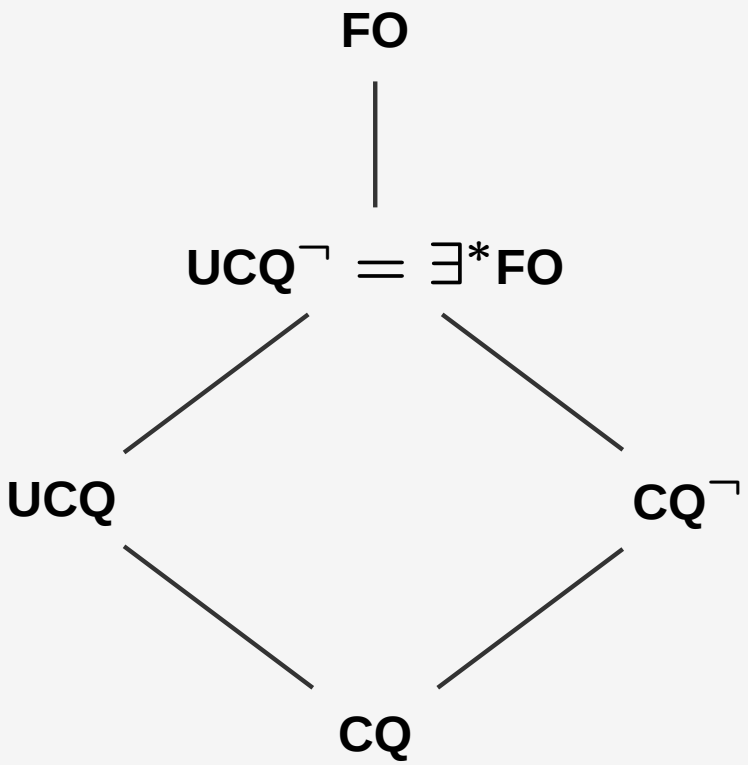
- General form:

$$\varphi(\vec{x}) = \exists \vec{y} \psi(\vec{x}, \vec{y})$$

  where ψ is a conjunction of atoms

- Equivalent characterizations:
  - select-project-join queries in relational algebra
  - select-from-where queries in SQL
- Extensions
  - **UCQ**: Unions of **CQ**s
  - **CQ**$^\neg$: Same as **CQ**s but negated atoms are allowed
  - **UCQ**$^\neg$: Unions of **CQ**$^\neg$s

- Known: Variants are pairwise different



- In this talk: Conjunctive queries (and extensions thereof) as update language
  - How powerful are they in the dynamic setting?
  - How do the dynamic variants relate to each other?

Contents

Introduction

The Framework of Patnaik and Immerman

▷ **Dynamic Conjunctive Queries**

A Note on Semantics

Conclusion and Future Work

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^{\neg}$, e.g.
  - reachability on acyclic graphs
  - context-free languages

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z, z)$$

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z, z)$$

- For simplicity: Insertions only

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

- Let R be a relation with

$$\vec{e} \in R \text{ if and only if } G + \vec{e} \models \psi$$

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

- Let R be a relation with

$$\vec{e} \in R \text{ if and only if } G + \vec{e} \models \psi$$

- Then $\phi^Q_{\text{INS}}(x_1, x_2) \stackrel{\text{def}}{=} \neg R(x_1, x_2)$

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

- Let R be a relation with

$$\vec{e} \in R \text{ if and only if } G + \vec{e} \models \psi$$

- Then $\phi^Q_{\text{INS}}(x_1, x_2) \stackrel{\text{def}}{=} \neg R(x_1, x_2)$

→ It remains to find an update formula for R

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

- Let R be a relation with

$$\vec{e} \in R \text{ if and only if } G + \vec{e} \models \psi$$

- Then $\phi^Q_{\text{INS}}(x_1, x_2) \stackrel{\text{def}}{=} \neg R(x_1, x_2)$

→ It remains to find an update formula for R

$$\phi^R_{\text{INS}}(y_1, y_2; x_1, x_2) = \exists z \Big(\neg E(z,z) \\ \wedge (y_1, y_2) \neq (z,z) \\ \wedge (x_1, x_2) \neq (z,z) \Big)$$

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

- Let R be a relation with

$$\vec{e} \in R \text{ if and only if } G + \vec{e} \models \psi$$

- Then $\phi^Q_{\text{INS}}(x_1, x_2) \stackrel{\text{def}}{=} \neg R(x_1, x_2)$

→ It remains to find an update formula for R

$$\phi^R_{\text{INS}}(y_1, y_2; x_1, x_2) = \exists z \Big(\neg E(z,z) \\ \wedge (y_1, y_2) \neq (z,z) \\ \wedge (x_1, x_2) \neq (z,z) \Big)$$

- For deletions:
  - Use an additional relation R' for $G - \vec{e}$
  - Update R and R' also under deletions

How Powerful are Dynamic Conjunctive Queries?

- Many queries maintainable in **DynFO** are already maintainable in **DynUCQ**$^\neg$, e.g.
  - reachability on acyclic graphs
  - context-free languages
- Furthermore:

Theorem

- Every first-order query can be maintained in **DynCQ**$^\neg$.

(Proof by) Example

- Question: How to remove $\forall$-quantifiers?
- Consider the graph query Q defined by

$$\forall z E(z,z)$$

- For simplicity: Insertions only
- Goal: Find update formula $\phi^Q_{\text{INS}}(x_1, x_2)$
- Idea: Look ahead.

- Rewrite formula to

$$\neg \underbrace{\exists z \neg E(z,z)}_{\psi}$$

- Let R be a relation with

$$\vec{e} \in R \text{ if and only if } G + \vec{e} \models \psi$$

- Then $\phi^Q_{\text{INS}}(x_1, x_2) \stackrel{\text{def}}{=} \neg R(x_1, x_2)$

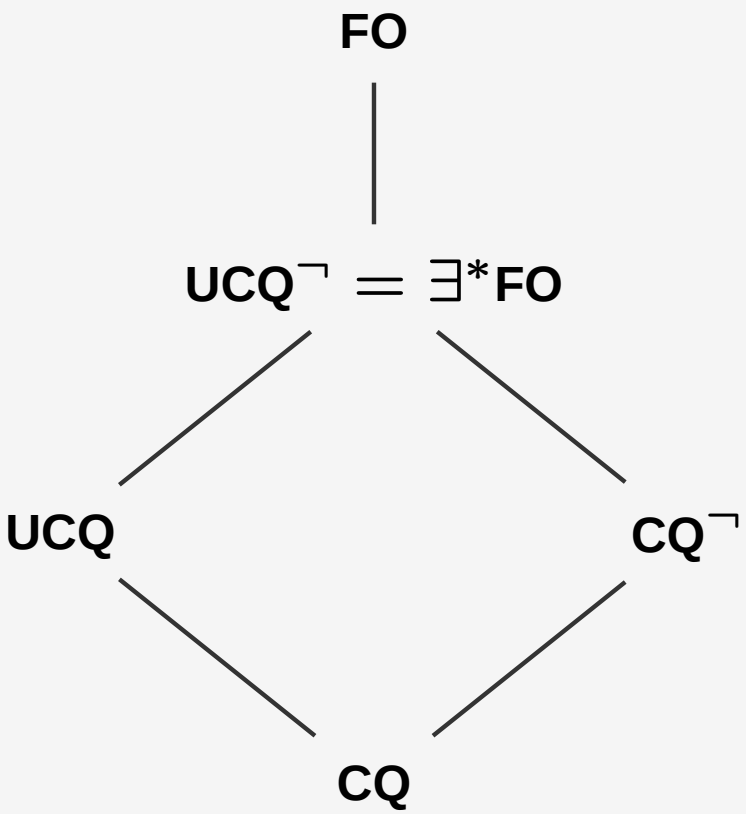
→ It remains to find an update formula for R

$$\phi^R_{\text{INS}}(y_1, y_2; x_1, x_2) = \exists z \Big(\neg E(z,z) \\ \wedge (y_1, y_2) \neq (z,z) \\ \wedge (x_1, x_2) \neq (z,z) \Big)$$

- For deletions:
  - Use an additional relation R' for $G - \vec{e}$
  - Update R and R' also under deletions
- Construction extends to full **FO**

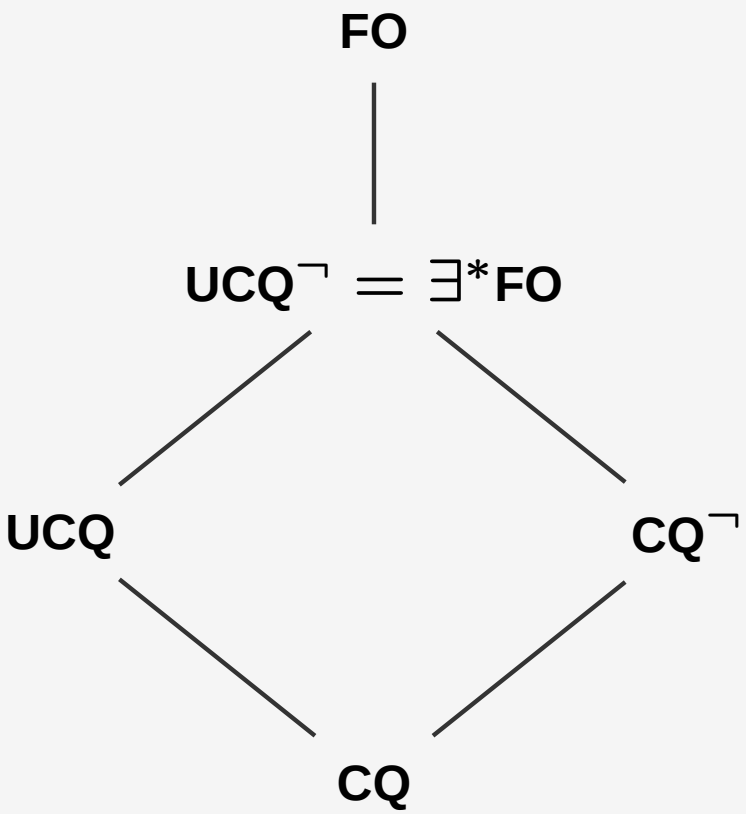
How do Variants of Dynamic Conjunctive Queries relate?

- Conjunctive queries in the static world

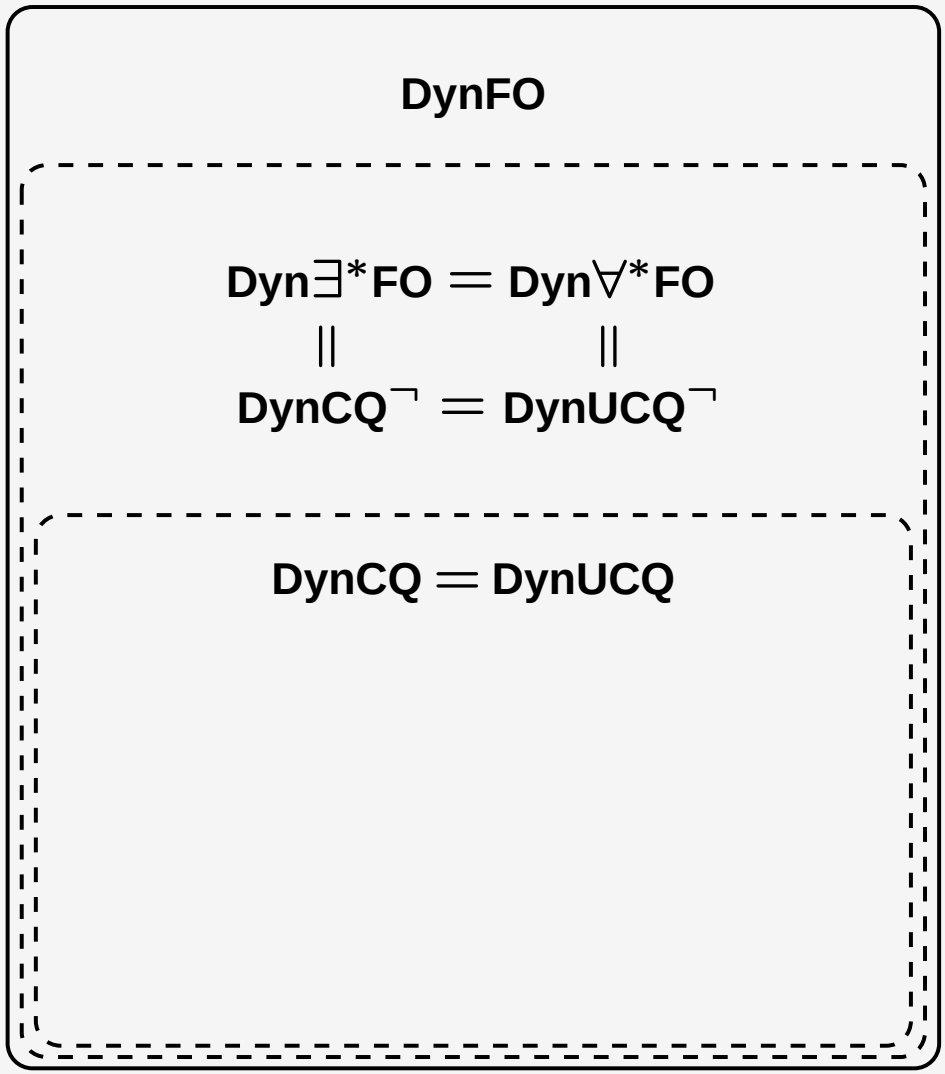


How do Variants of Dynamic Conjunctive Queries relate?

- Conjunctive queries in the static world

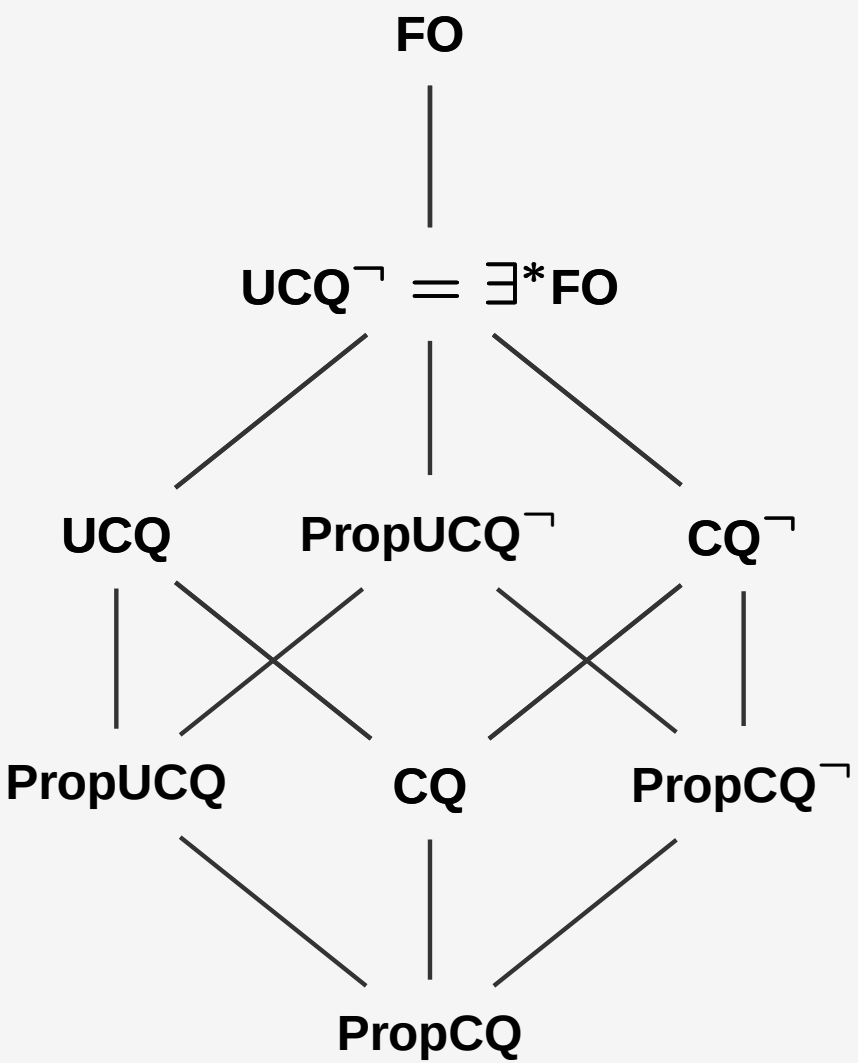


- Conjunctive queries in the dynamic world

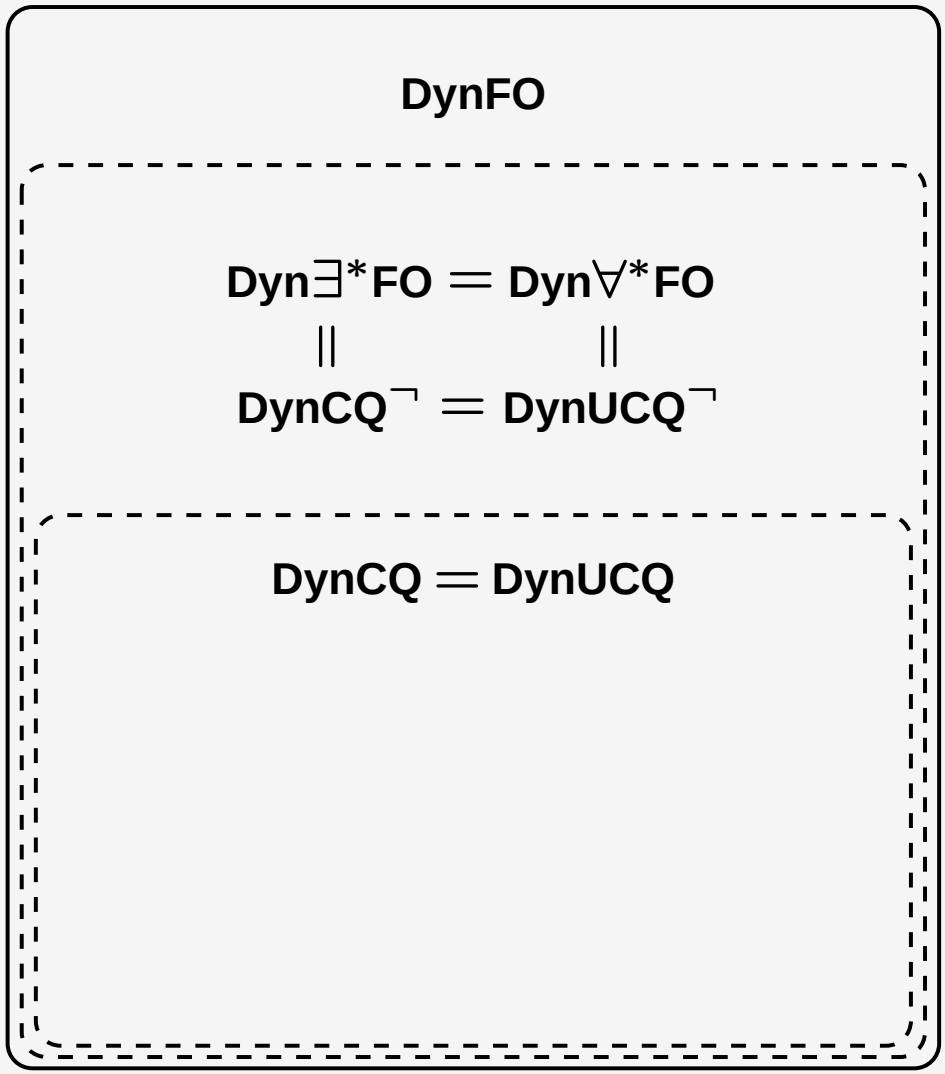


How do Variants of Dynamic Conjunctive Queries relate?

- Conjunctive queries in the static world

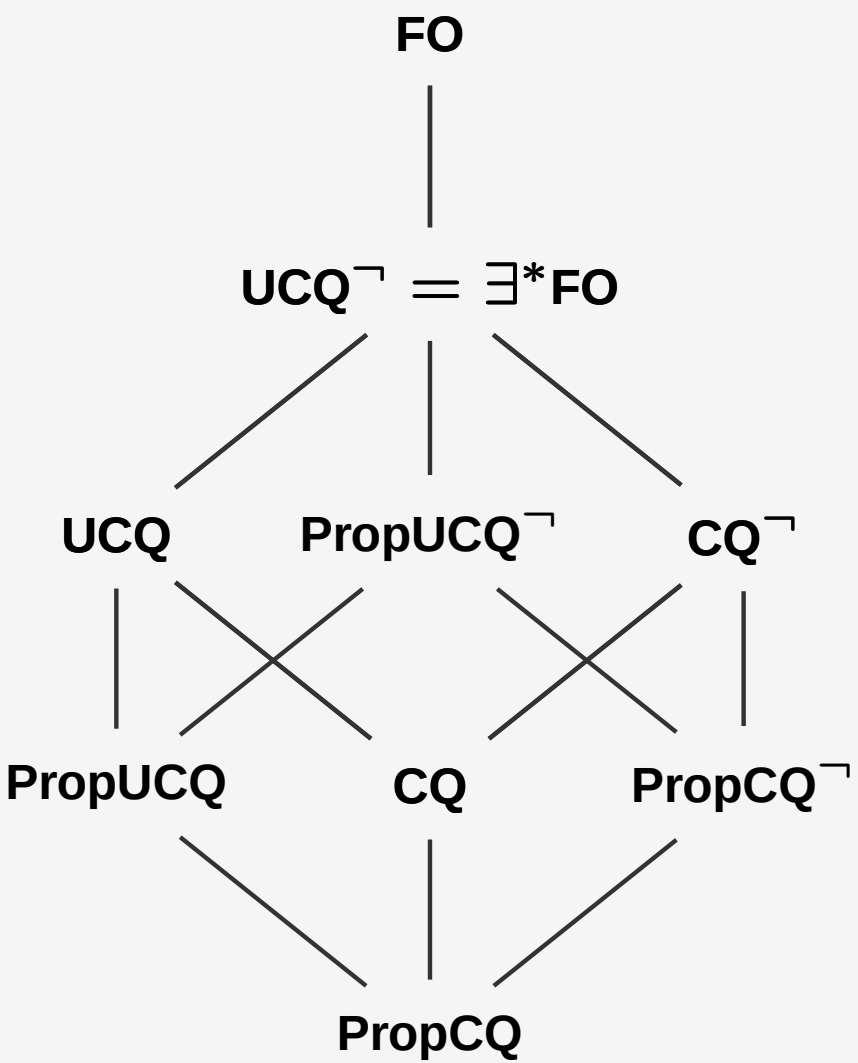


- Conjunctive queries in the dynamic world

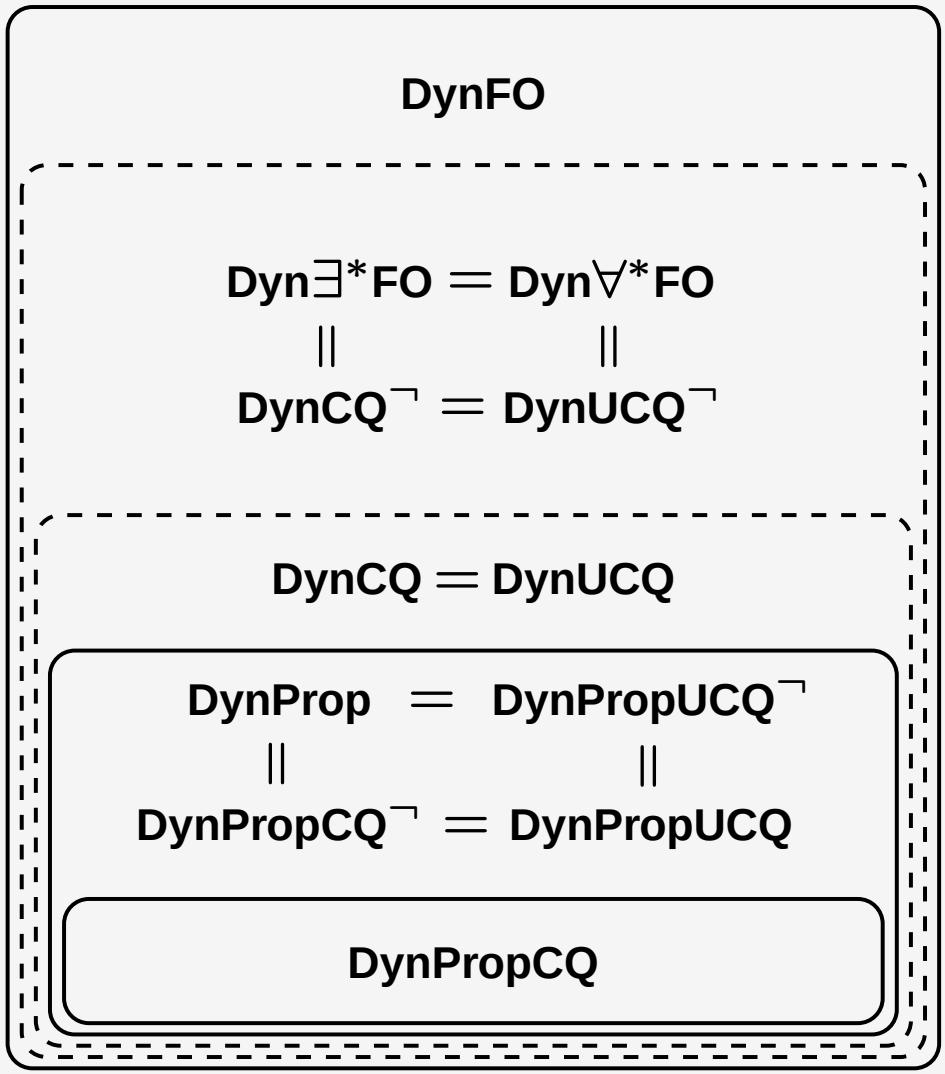


How do Variants of Dynamic Conjunctive Queries relate?

- Conjunctive queries in the static world

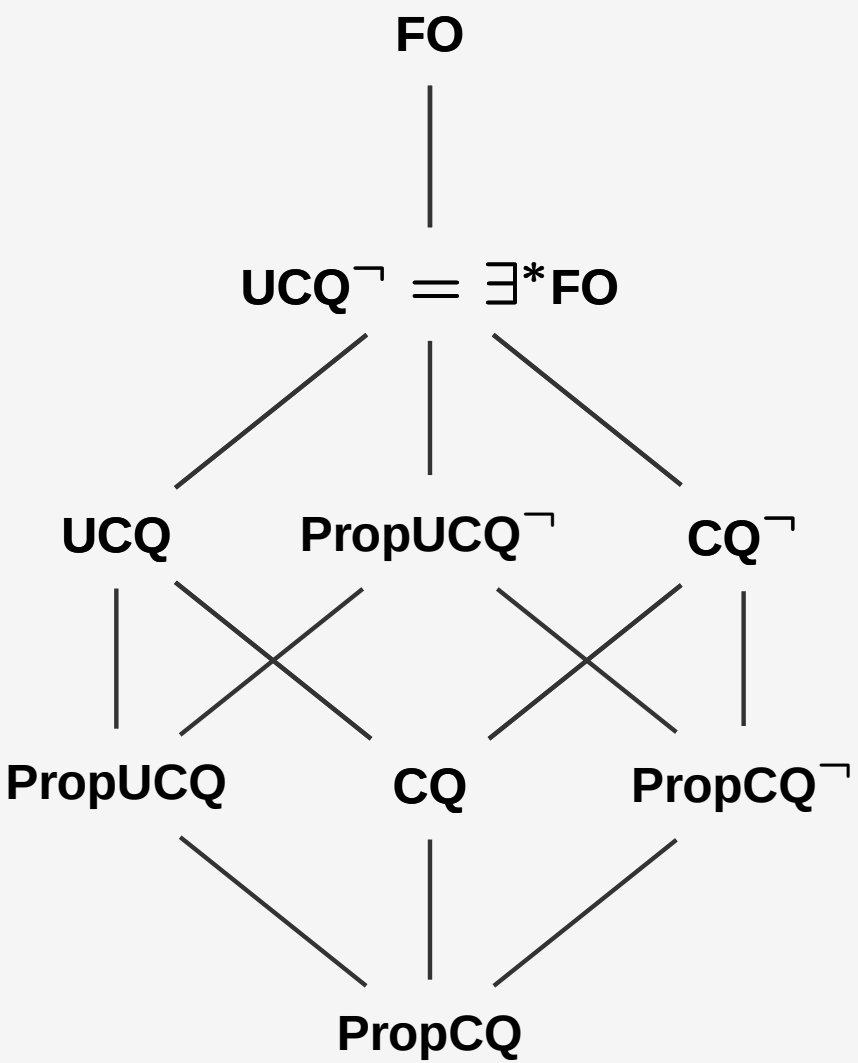


- Conjunctive queries in the dynamic world

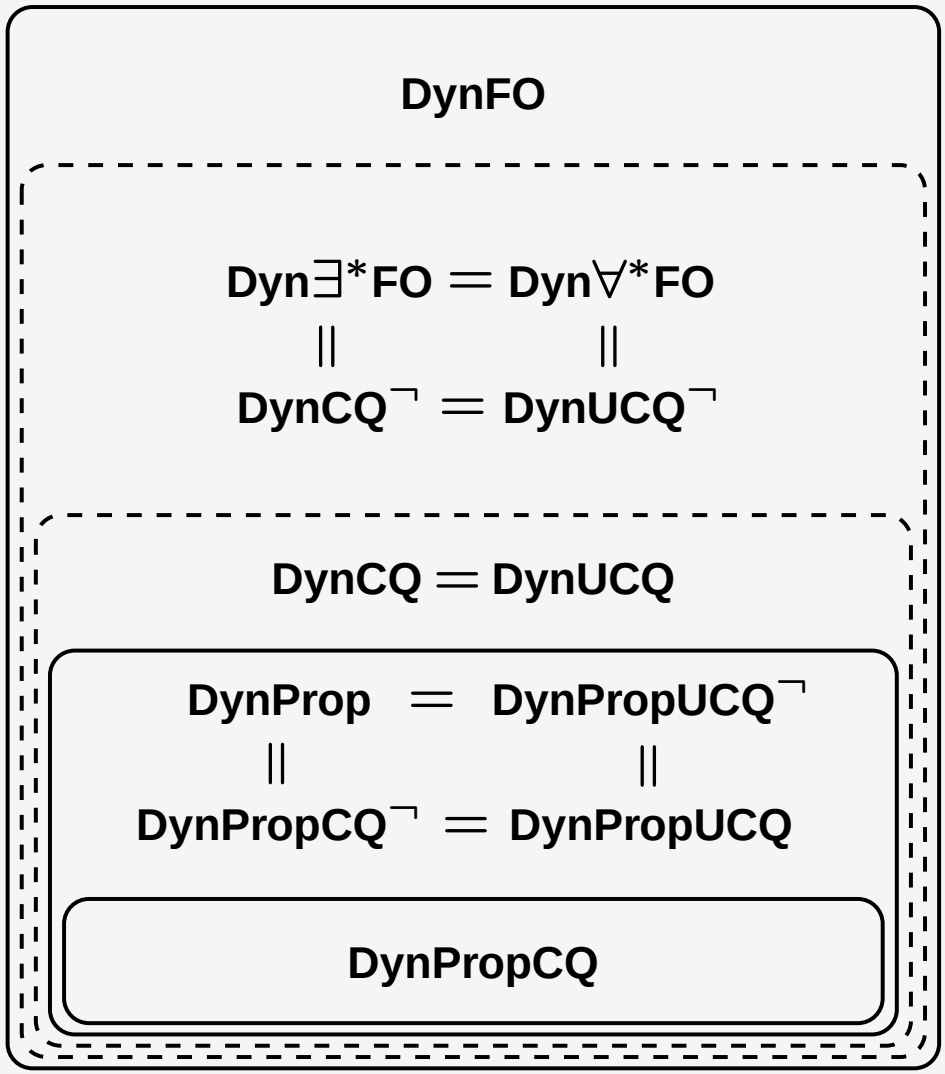


How do Variants of Dynamic Conjunctive Queries relate?

- Conjunctive queries in the static world



- Conjunctive queries in the dynamic world



- Solid lines are strict separations

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{\boldsymbol{T}}_{\text{INS}}(\boldsymbol{u},\boldsymbol{v};\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} \boldsymbol{Q}(\boldsymbol{x},\boldsymbol{y}) \vee (\boldsymbol{R}(\boldsymbol{x},\boldsymbol{u}) \wedge \boldsymbol{S}(\boldsymbol{v},\boldsymbol{y}))$$

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{\boldsymbol{T}}_{\text{INS}}(\boldsymbol{u},\boldsymbol{v};\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} \boldsymbol{Q}(\boldsymbol{x},\boldsymbol{y}) \vee (\boldsymbol{R}(\boldsymbol{x},\boldsymbol{u}) \wedge \boldsymbol{S}(\boldsymbol{v},\boldsymbol{y}))$$

- Assume for simplicity: $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ are non-empty.

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{\boldsymbol{T}}_{\text{INS}}(u, v; x, y) \stackrel{\text{def}}{=} \boldsymbol{Q}(x, y) \vee (\boldsymbol{R}(x, u) \wedge \boldsymbol{S}(v, y))$$

- Assume for simplicity: $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ are non-empty.
- Rewrite $\phi^{\boldsymbol{T}}_{\text{INS}}$ as

$$\exists z_1, z_2, z_3, z_4, z_5, z_6 \Big(\boldsymbol{Q}(z_1, z_2) \wedge \boldsymbol{R}(z_3, z_4) \wedge \boldsymbol{S}(z_5, z_6) \wedge \Big(\big((z_1, z_2) = (x, y) \vee \big((z_3, z_4) = (x, u) \wedge (z_5, z_6) = (v, y) \big) \big) \Big) \Big)$$

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{\boldsymbol{T}}_{\text{INS}}(u, v; x, y) \stackrel{\text{def}}{=} Q(x, y) \vee (R(x, u) \wedge S(v, y))$$

- Assume for simplicity: Q, R and S are non-empty.
- Rewrite $\phi^{\boldsymbol{T}}_{\text{INS}}$ as

$$\exists z_1, z_2, z_3, z_4, z_5, z_6 \Big(Q(z_1, z_2) \wedge R(z_3, z_4) \wedge S(z_5, z_6) \wedge \Big(\big((z_1, z_2) = (x, y) \vee \big((z_3, z_4) = (x, u) \wedge (z_5, z_6) = (v, y) \big) \big) \Big) \Big)$$

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{\boldsymbol{T}}_{\text{INS}}(u, v; x, y) \stackrel{\text{def}}{=} \boldsymbol{Q}(x, y) \vee (\boldsymbol{R}(x, u) \wedge \boldsymbol{S}(v, y))$$

- Assume for simplicity: $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ are non-empty.
- Rewrite $\phi^{\boldsymbol{T}}_{\text{INS}}$ as

$$\exists z_1, z_2, z_3, z_4, z_5, z_6 \Big(\boldsymbol{Q}(z_1, z_2) \wedge \boldsymbol{R}(z_3, z_4) \wedge \boldsymbol{S}(z_5, z_6) \wedge \Big(\big((z_1, z_2) = (x, y) \vee \big((z_3, z_4) = (x, u) \wedge (z_5, z_6) = (v, y) \big) \big) \Big) \Big)$$

→ Can be replaced by a fixed relation:

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} Q(x,y) \vee (R(x,u) \wedge S(v,y))$$

- Assume for simplicity: Q, R and S are non-empty.
- Rewrite ϕ^{T}_{INS} as

$$\exists z_1,z_2,z_3,z_4,z_5,z_6 \Big(Q(z_1,z_2) \wedge R(z_3,z_4) \wedge S(z_5,z_6) \wedge \Big((z_1,z_2) = (x,y) \vee \big((z_3,z_4) = (x,u) \wedge (z_5,z_6) = (v,y) \big) \Big) \Big)$$

→ Can be replaced by a fixed relation:

$$\exists z_1,z_2,z_3,z_4,z_5,z_6 \Big(Q(z_1,z_2) \wedge R(z_3,z_4) \wedge S(z_5,z_6) \wedge P(x,y,u,v,z_1,\ldots,z_6) \Big)$$

A Glimpse at the Proofs: How to remove Unions

Theorem

- Every query maintainable in **DynUCQ** can be maintained in **DynCQ** as well.

(Proof by) Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} Q(x,y) \vee (R(x,u) \wedge S(v,y))$$

- Assume for simplicity: Q, R and S are non-empty.
- Rewrite ϕ^{T}_{INS} as

$$\exists z_1,z_2,z_3,z_4,z_5,z_6 \Big(Q(z_1,z_2) \wedge R(z_3,z_4) \wedge S(z_5,z_6) \wedge \Big((z_1,z_2) = (x,y) \vee \big((z_3,z_4) = (x,u) \wedge (z_5,z_6) = (v,y) \big) \Big) \Big)$$

→ Can be replaced by a fixed relation:

$$\exists z_1,z_2,z_3,z_4,z_5,z_6 \Big(Q(z_1,z_2) \wedge R(z_3,z_4) \wedge S(z_5,z_6) \wedge P(x,y,u,v,z_1,\ldots,z_6) \Big)$$

where $P(x,y,u,v,z_1,\ldots,z_6)$ is (**FO**-)initialized by the result of

$$(z_1,z_2) = (x,y) \vee \big((z_3,z_4) = (x,u) \wedge (z_5,z_6) = (v,y) \big)$$

Contents

Introduction

The Framework of Patnaik and Immerman

Dynamic Conjunctive Queries

▷ **A Note on Semantics**

Conclusion and Future Work

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{\boldsymbol{T}}_{\text{INS}}(u, v; x, y) \stackrel{\text{def}}{=} \boldsymbol{T}(x, y) \vee (\boldsymbol{T}(x, u) \wedge \boldsymbol{T}(v, y))$$

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- Also common in databases: Δ-Semantics

$$\phi^{T^+}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \neg T(x,y) \wedge T(x,u) \wedge T(v,y)$$
$$\phi^{T^-}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \bot$$

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- Also common in databases: Δ-Semantics

$$\phi^{T^+}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \neg T(x,y) \wedge T(x,u) \wedge T(v,y)$$
$$\phi^{T^-}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \bot$$

- For fragments closed under boolean operations:
  - From absolute to Δ-semantics:

$$\phi^{R^+}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge \neg R(\vec{x})$$
$$\phi^{R^-}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \neg\phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge R(\vec{x})$$

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- Also common in databases: Δ-Semantics

$$\phi^{T^+}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \neg T(x,y) \wedge T(x,u) \wedge T(v,y)$$
$$\phi^{T^-}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \bot$$

- For fragments closed under boolean operations:
  - From absolute to Δ-semantics:

$$\phi^{R^+}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge \neg R(\vec{x})$$
$$\phi^{R^-}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \neg\phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge R(\vec{x})$$

  - Similarly: Conversion from Δ-semantics to absolute semantics

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- Also common in databases: Δ-Semantics

$$\phi^{T^+}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \neg T(x,y) \wedge T(x,u) \wedge T(v,y)$$
$$\phi^{T^-}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \bot$$

- For fragments closed under boolean operations:
  - From absolute to Δ-semantics:

$$\phi^{R^+}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge \neg R(\vec{x})$$
$$\phi^{R^-}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \neg\phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge R(\vec{x})$$

  - Similarly: Conversion from Δ-semantics to absolute semantics

→ Semantics do not differ for fragments closed under boolean operations.

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- Also common in databases: Δ-Semantics

$$\phi^{T^+}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \neg T(x,y) \wedge T(x,u) \wedge T(v,y)$$
$$\phi^{T^-}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \bot$$

- For fragments closed under boolean operations:
  - From absolute to Δ-semantics:

$$\phi^{R^+}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge \neg R(\vec{x})$$
$$\phi^{R^-}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \neg\phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge R(\vec{x})$$

  - Similarly: Conversion from Δ-semantics to absolute semantics

→ Semantics do not differ for fragments closed under boolean operations.

- But: Conjunctive queries are not closed under negations.

An Alternative Semantics

- Upto now: Absolute Semantics
  - Auxiliary data defined from scratch

Example

$$\phi^{T}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} T(x,y) \vee (T(x,u) \wedge T(v,y))$$

- Also common in databases: Δ-Semantics

$$\phi^{T^+}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \neg T(x,y) \wedge T(x,u) \wedge T(v,y)$$
$$\phi^{T^-}_{\text{INS}}(u,v;x,y) \stackrel{\text{def}}{=} \bot$$

- For fragments closed under boolean operations:
  - From absolute to Δ-semantics:

$$\phi^{R^+}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge \neg R(\vec{x})$$
$$\phi^{R^-}_{\delta}(\vec{u};\vec{x}) \stackrel{\text{def}}{=} \neg\phi^{R}_{\delta}(\vec{u};\vec{x}) \wedge R(\vec{x})$$

  - Similarly: Conversion from Δ-semantics to absolute semantics

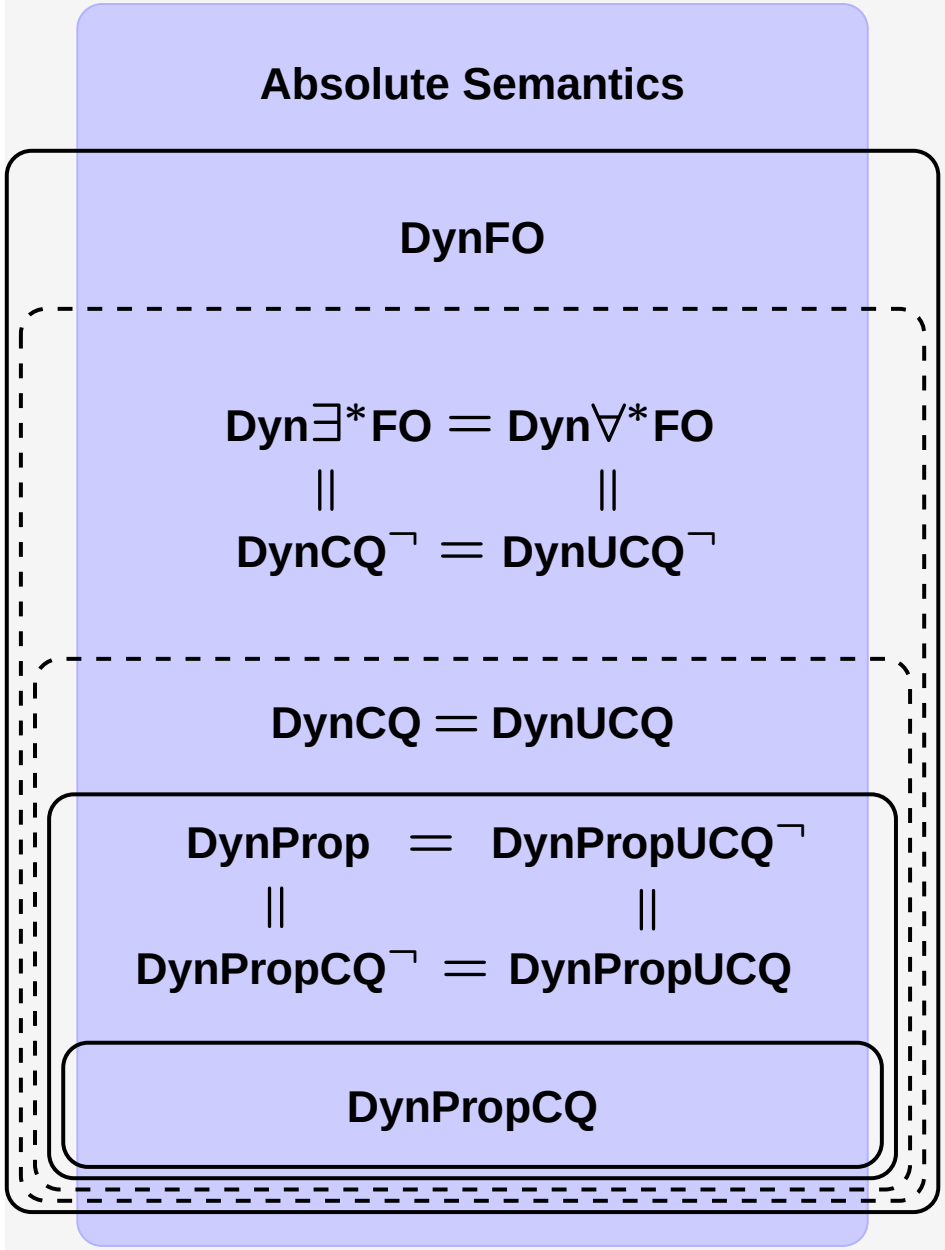
→ Semantics do not differ for fragments closed under boolean operations.

- But: Conjunctive queries are not closed under negations.

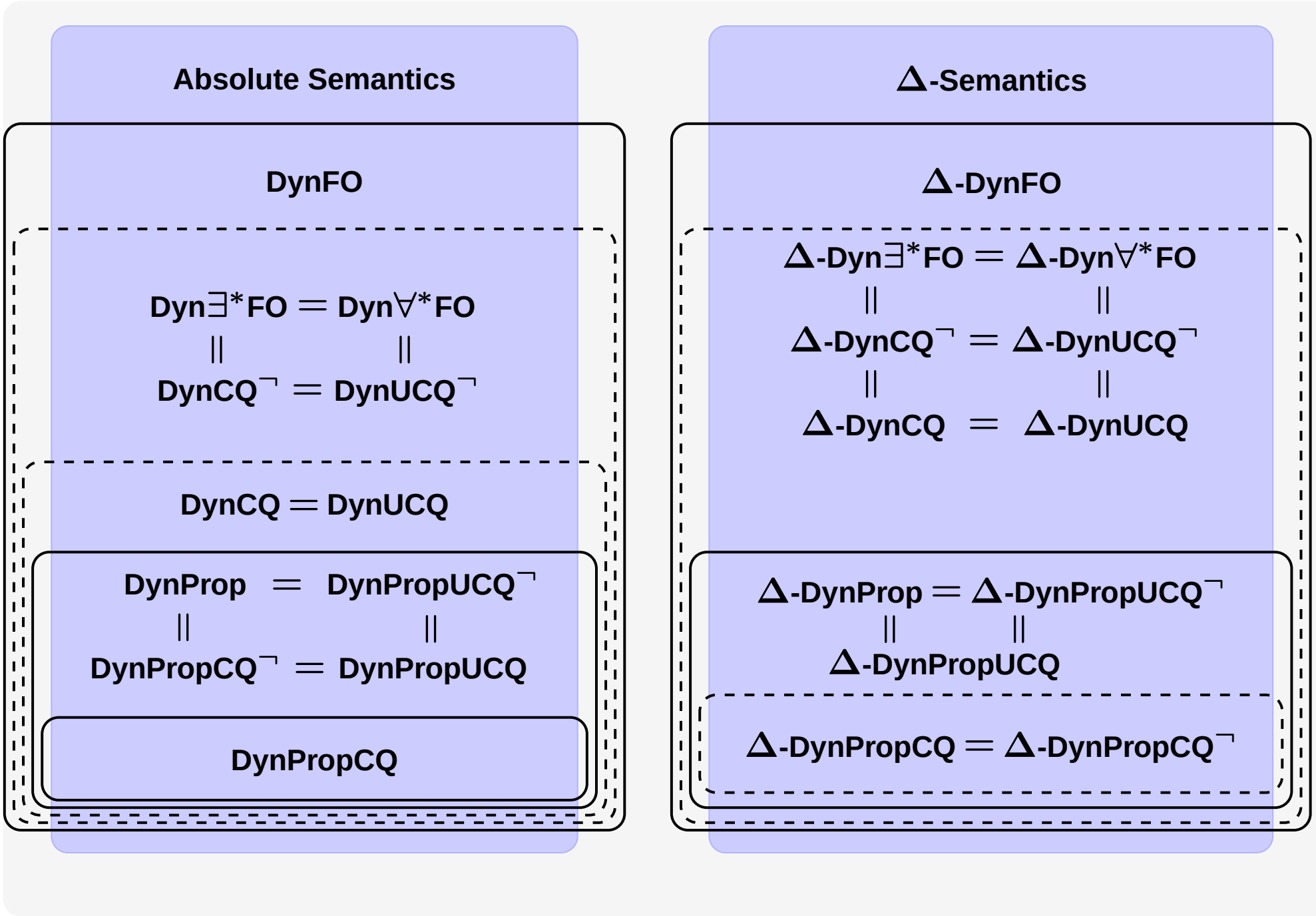
→ Questions:

  - Do variants of dynamic conjunctive queries under Δ-semantics collapse as well?
  - How do the semantics compare to each other?

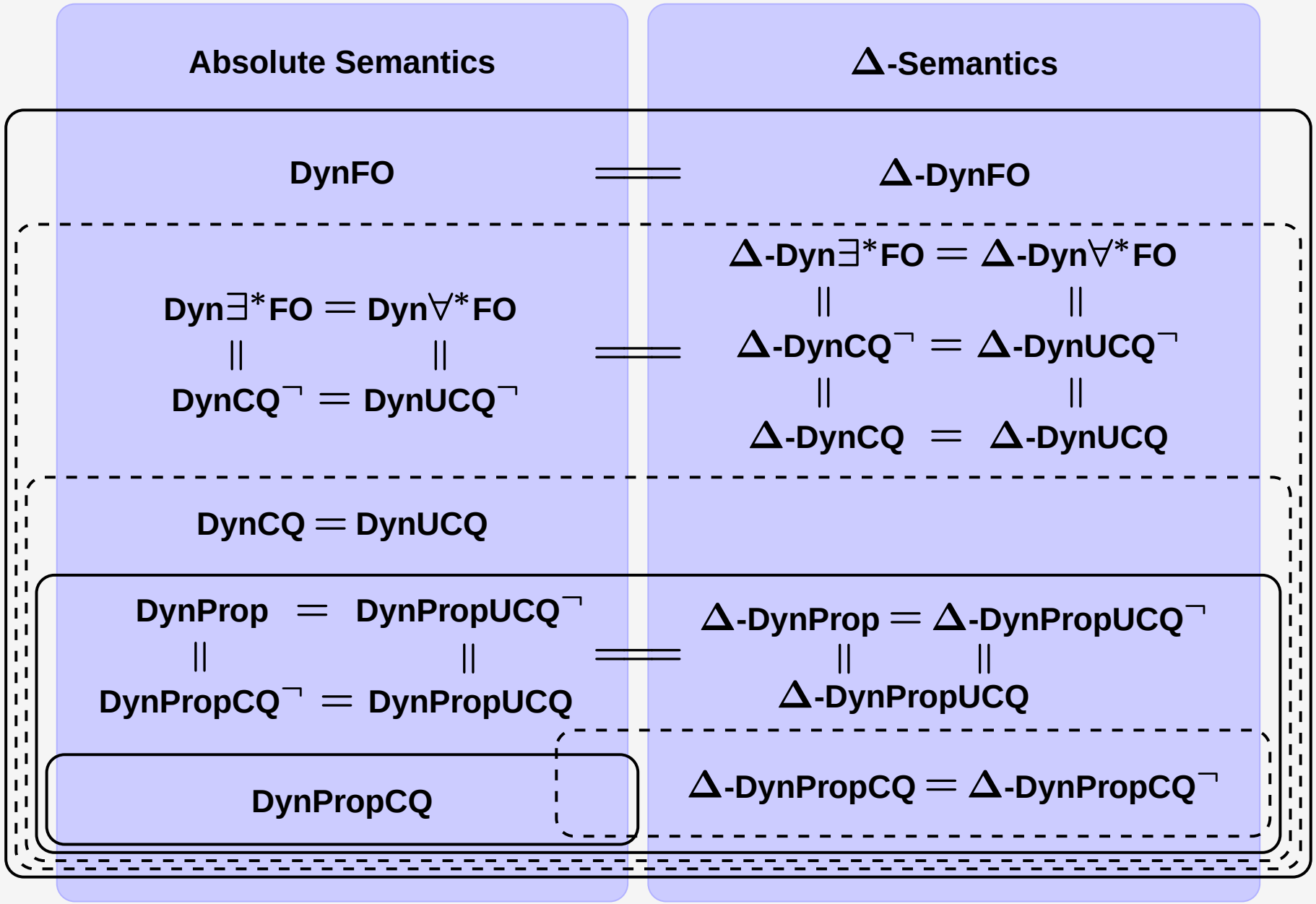
Absolute Semantics vs. Δ-Semantics



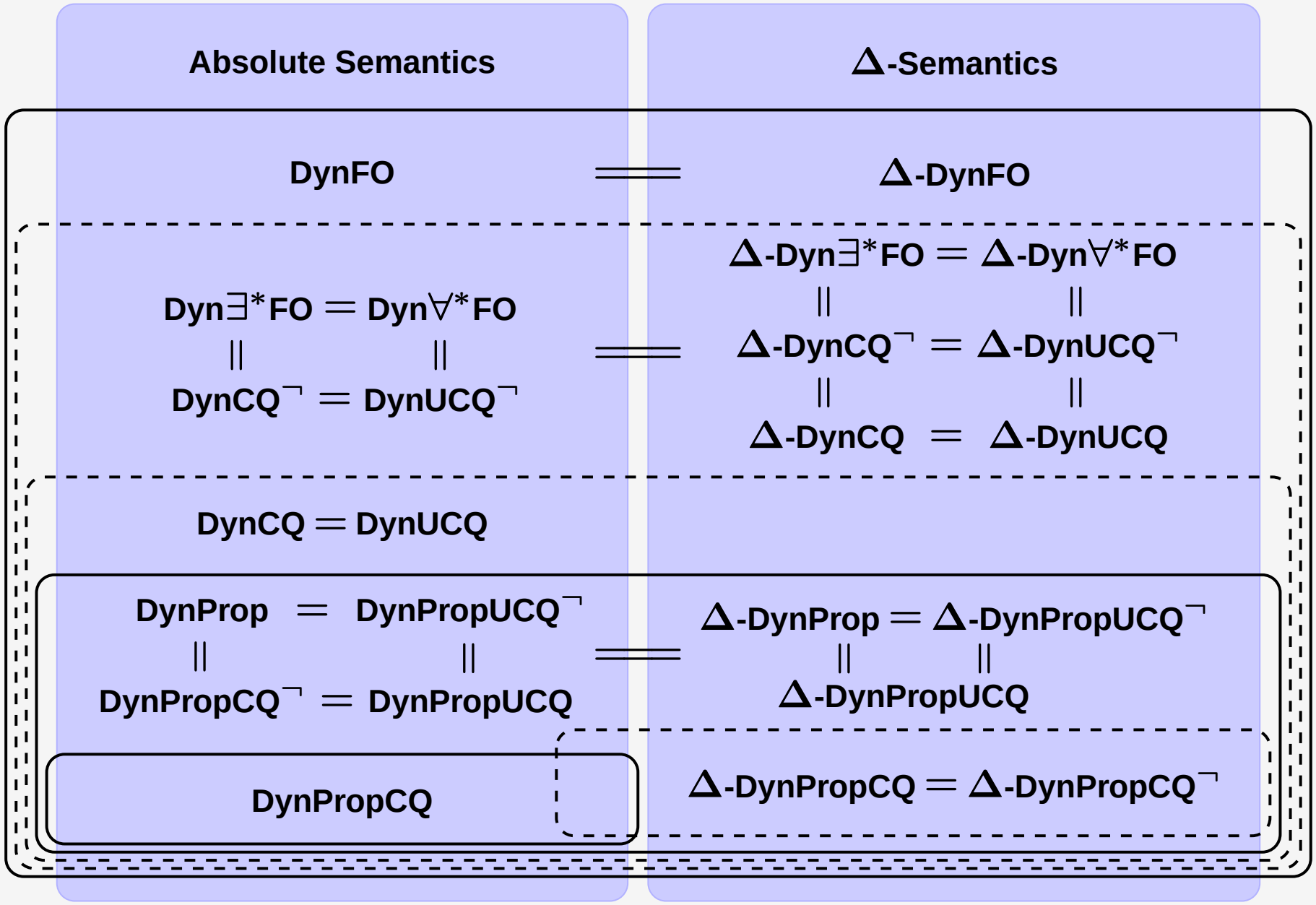
Absolute Semantics vs. Δ-Semantics



Absolute Semantics vs. Δ-Semantics



Absolute Semantics vs. Δ-Semantics



- Hence: Every first-order query is maintainable in **Δ-DynCQ**.

Contents

Introduction

The Framework of Patnaik and Immerman

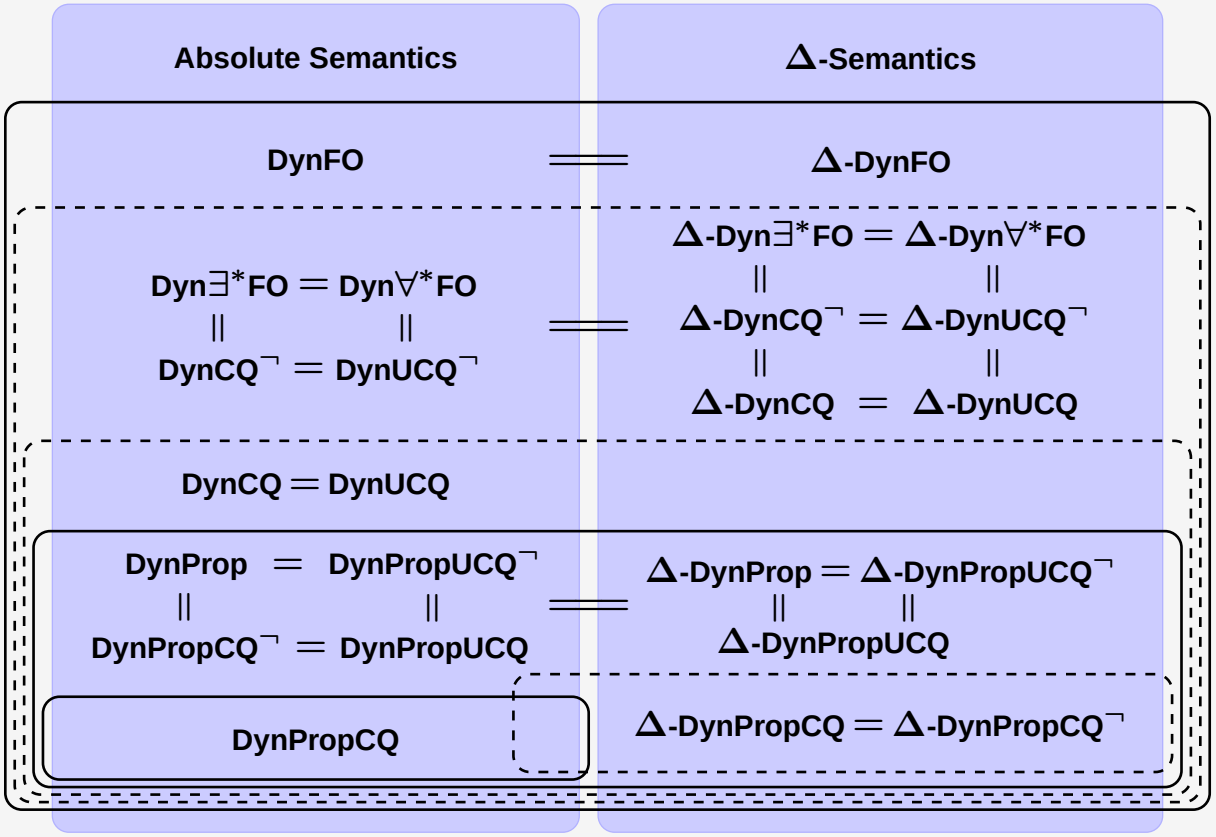
Dynamic Conjunctive Queries

A Note on Semantics

▷ **Conclusion and Future Work**

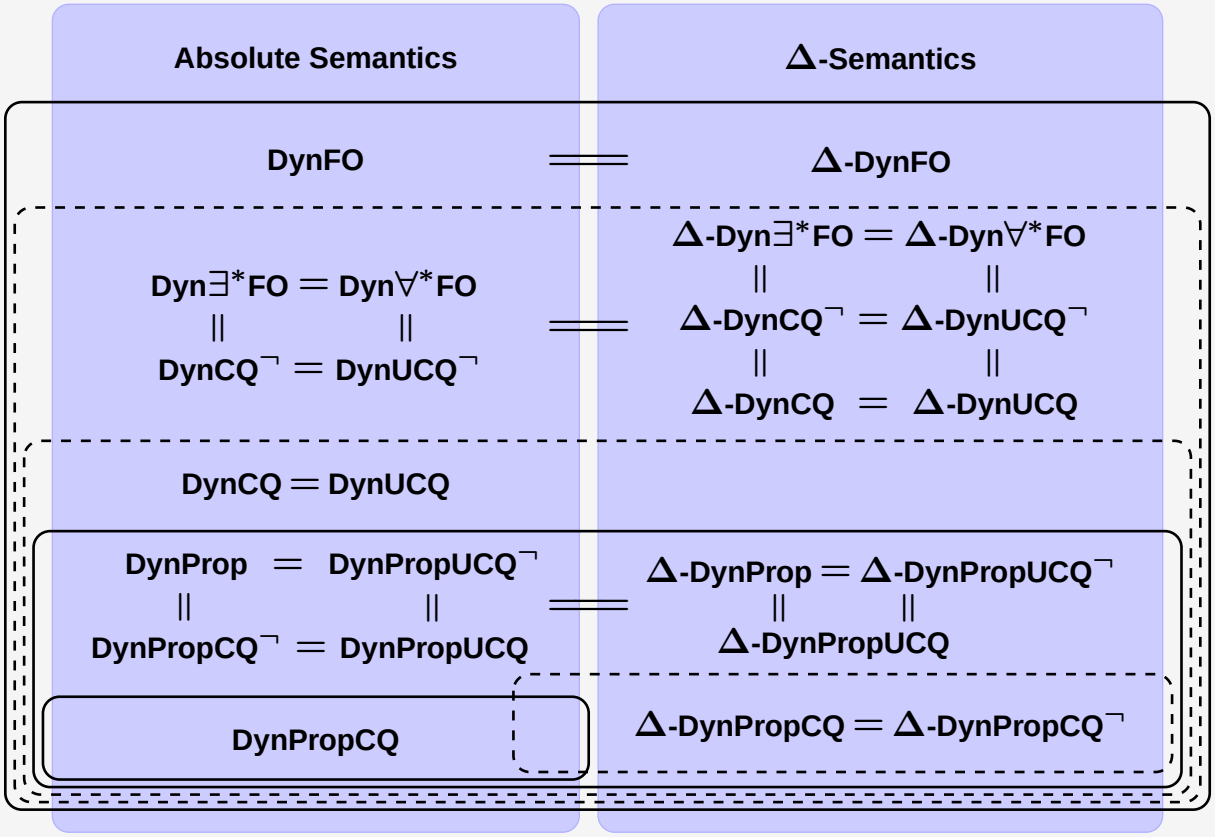
Conclusion and Future Work

- Main results
  - **FO $\subsetneq$ DynCQ$^\neg$**
  - Variants of dynamic conjunctive queries:



Conclusion and Future Work

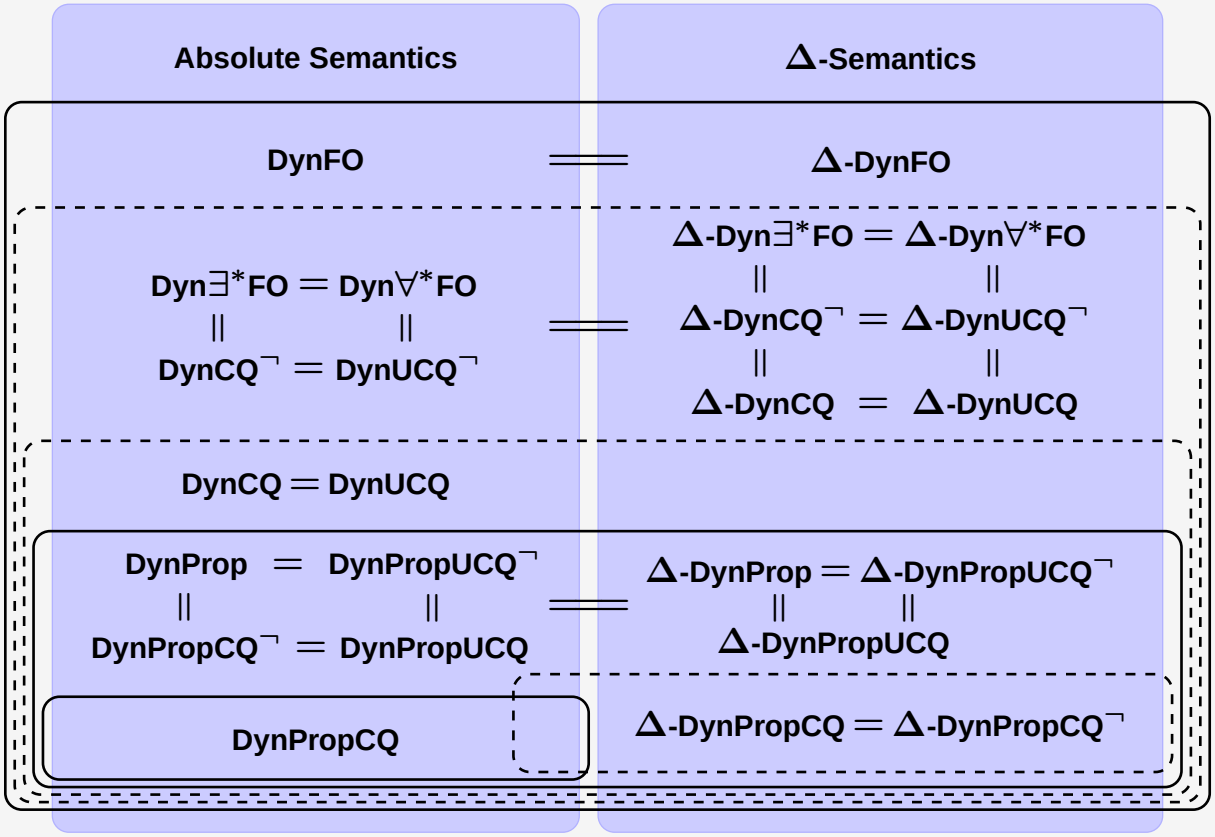
- Main results
  - **FO $\subsetneq$ DynCQ$^\neg$**
  - Variants of dynamic conjunctive queries:



- Work in progress and future work:
  - Can the remaining variants be separated?
  - Study dynamic graph queries and their fragments.

Conclusion and Future Work

- Main results
  - **FO $\subseteq$ DynCQ$^\neg$**
  - Variants of dynamic conjunctive queries:



- Work in progress and future work:
  - Can the remaining variants be separated?
  - Study dynamic graph queries and their fragments.

Thank You!

And thanks to the DFG for supporting this work.