TECHNISCHE UNIVERSITÄT DRESDEN

FOUNDATIONS OF DATABASES AND QUERY LANGUAGES

Lecture 8: Conjunctive Query Optimisation / First-Order Expressiveness

Markus Krötzsch

TU Dresden, 8 June 2015

Overview

1. Introduction | Relational data model
2. First-order queries
3. Complexity of query answering
4. Complexity of FO query answering
5. Conjunctive queries
6. Tree-like conjunctive queries
7. Query optimisation
8. Conjunctive Query Optimisation / First-Order Expressiveness
9. First-Order Expressiveness / Introduction to Datalog
10. Expressive Power and Complexity of Datalog
11. Implementation techniques for Datalog
12. Path queries
13. Constraints
14. Outlook: database theory in practice

See course homepage [⇒ link] for more information and materials

Review

There are many well-defined static optimisation tasks that are independent of the database
$\leadsto$ query equivalence, containment, emptiness

Unfortunately, all of them are undecidable for FO queries $\leadsto$
Slogan: “all interesting questions about FO queries are undecidable”

$\leadsto$ Let’s look at simpler query languages

Optimisation Futile for Conjunctive Queries

Optimisation is simpler for conjunctive queries

Conjunctive query containment – example:

$$Q_1 : \quad \exists x, y, z.\ R(x, y) \wedge R(y, y) \wedge R(y, z)$$
$$Q_2 : \quad \exists u, v, w, t.\ R(u, v) \wedge R(v, w) \wedge R(w, t)$$

Q_1 find R-paths of length two with a loop in the middle
Q_2 find R-paths of length three

$\rightsquigarrow$ in a loop one can find paths of any length
$\rightsquigarrow Q_1 \sqsubseteq Q_2$

Deciding Conjunctive Query Containment

Consider conjunctive queries $Q_1[x_1, \ldots, x_n]$ and $Q_2[y_1, \ldots, y_n]$.

A query homomorphism from Q_2 to Q_1 is a mapping μ
from terms (constants or variables) in Q_2 to terms in Q_1 such that:

- μ does not change constants, i.e., $\mu(c) = c$ for every constant c
- $x_i = \mu(y_i)$ for each $i = 1, \ldots, n$
- if Q_2 has a query atom $R(t_1, \ldots, t_n)$
  then Q_1 has a query atom $R(\mu(t_1), \ldots, \mu(t_n))$

Theorem (Homomorphism Theorem)

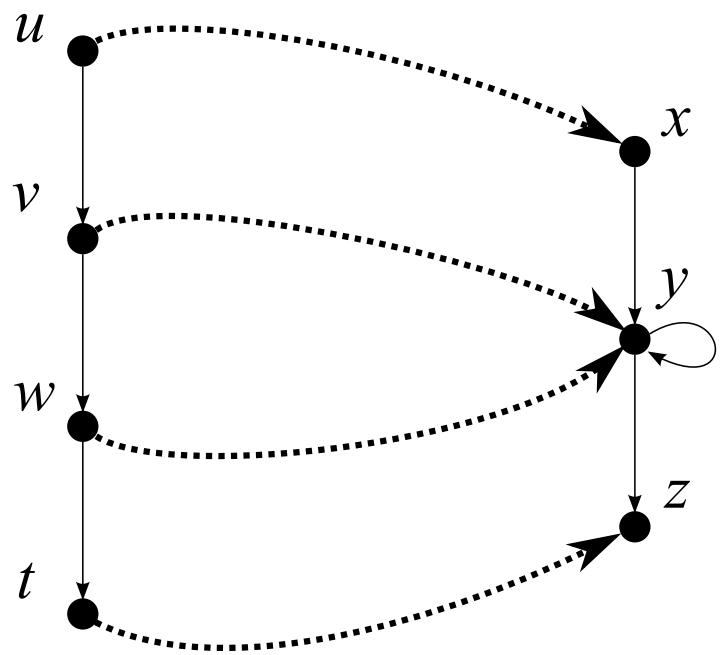
$Q_1 \sqsubseteq Q_2$ if and only if there is a query homomorphism $Q_2 \to Q_1$.

$\leadsto$ decidable (only need to check finitely many mappings from Q_2 to Q_1)

Example

$$Q_1 : \quad \exists x, y, z.\ R(x, y) \wedge R(y, y) \wedge R(y, z)$$

$$Q_2 : \quad \exists u, v, w, t.\ R(u, v) \wedge R(v, w) \wedge R(w, t)$$



Review: CQs and Homomorphisms

If $\langle d_1, \ldots, d_n \rangle$ is a result of $Q_1[x_1, \ldots, x_n]$ over database $\mathcal{I}$ then:

- there is a mapping ν from variables in Q_1 to the domain of $\mathcal{I}$
- $d_i = \nu(x_i)$ for all $i = 1, \ldots, m$
- for all atoms $R(t_1, \ldots, t_m)$ of Q_1, we find $\langle \nu(t_1), \ldots, \nu(t_m) \rangle \in R^{\mathcal{I}}$ (where we $\nu(c)$ take to mean c for constants c)

$\rightsquigarrow \mathcal{I} \models Q_1[d_1, \ldots, d_n]$ if there is such a homomorphism ν from Q_1 to $\mathcal{I}$

(Note: this is a slightly different formulation from the “homomorphism problem” discussed in a previous lecture, since we keep constants in queries here)

Proof of the Homomorphism Theorem

"$\Leftarrow$": $Q_1 \sqsubseteq Q_2$ if there is a query homomorphism $Q_2 \to Q_1$.

(1) Let $\langle d_1, \ldots, d_n \rangle$ be a result of $Q_1[x_1, \ldots, x_n]$ over database $\mathcal{I}$.

(2) Then there is a homomorphism ν from Q_1 to $\mathcal{I}$.

(3) By assumption, there is a query homomorphism $\mu : Q_2 \to Q_1$.

(4) But then the composition $\nu \circ \mu$ that maps each term t to $\nu(\mu(t))$ is a homomorphism from Q_2 to $\mathcal{I}$.

(5) Hence $\langle \nu(\mu(y_1)), \ldots, \nu(\mu(y_n)) \rangle$ is a result of $Q_2[y_1, \ldots, y_n]$ over $\mathcal{I}$.

(6) Since $\nu(\mu(y_i)) = \nu(x_i) = d_i$, we find that $\langle d_1, \ldots, d_n \rangle$ be a result of $Q_2[y_1, \ldots, y_n]$ over $\mathcal{I}$.

Since this holds for all results $\langle d_1, \ldots, d_n \rangle$ of Q_1, we have $Q_1 \sqsubseteq Q_2$.

(See board for a sketch showing how we compose homomorphisms here)

Proof of the Homomorphism Theorem

"⇒": there is a query homomorphism $Q_2 \to Q_1$ if $Q_1 \sqsubseteq Q_2$.

(1) Turn $Q_1[x_1, \ldots, x_n]$ into a database $\mathcal{I}_1$ in the natural way:
   - The domain of $\mathcal{I}_1$ are the terms in Q_1
   - For every relation R, we have $\langle t_1, \ldots, t_m \rangle \in R^{\mathcal{I}_1}$ exactly if $R(t_1, \ldots, t_m)$ is an atom in Q_1

(2) Then Q_1 has a result $\langle x_1, \ldots, x_n \rangle$ over $\mathcal{I}_1$

   (the identity mapping is a homomorphism – actually even an isomorphism)

(3) Therefore, since $Q_1 \sqsubseteq Q_2$, $\langle x_1, \ldots, x_n \rangle$ is also a result of Q_2 over $\mathcal{I}_1$

(4) Hence there is a homomorphism ν from Q_2 to $\mathcal{I}_1$

(5) This homomorphism ν is also a query homomorphism $Q_2 \to Q_1$.

Implications of the Homomorphism Theorem

The proof has highlighted another useful fact:

The following two are equivalent:

- Finding a homomorphism from Q_2 to Q_1
- Finding a query result for Q_2 over $\mathcal{I}_1$

$\rightsquigarrow$ all complexity results for CQ query answering apply

Theorem

Deciding if $Q_1 \sqsubseteq Q_2$ is NP-complete.

If Q_2 is a tree query (or of bounded treewidth, or of bounded hypertree width) then deciding if $Q_1 \sqsubseteq Q_2$ is polynomial (in fact LOGCFL-complete).

Note that even in the NP-complete case the problem size is rather small (only queries, no databases)

Application: CQ Minimisation

Definition

A conjunctive query Q is minimal if:

- for all subqueries Q' of Q (that is, queries Q' that are obtained by dropping one or more atoms from Q),
- we find that $Q' \not\equiv Q$.

A minimal CQ is also called a core.

It is useful to minimise CQs to avoid unnecessary joins in query answering.

CQ Minimisation the Direct Way

A simple idea for minimising Q:

- Consider each atom of Q, one after the other
- Check if the subquery obtained by dropping this atom is contained in Q

  (Observe that the subquery always contains the original query.)
- If yes, delete the atom; continue with the next atom

Example query $Q[v, w]$:

$$\exists x, y, z.R(a, y) \wedge R(x, y) \wedge S(y, y) \wedge S(y, z) \wedge S(z, y) \wedge T(y, v) \wedge T(y, w)$$

$\rightsquigarrow$ Simpler notation: write as set and mark answer variables

$$\{R(a, y), R(x, y), S(y, y), S(y, z), S(z, y), T(y, \bar{v}), T(y, \bar{w})\}$$

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$
$R(x,y)$	$R(x,y)$
$S(y,y)$	$S(y,y)$
$S(y,z)$	$S(y,z)$
$S(z,y)$	$S(z,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$
$T(y,\bar{w})$	$T(y,\bar{w})$

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	~~$R(a,y)$~~ ?
$R(x,y)$	$R(x,y)$
$S(y,y)$	$S(y,y)$
$S(y,z)$	$S(y,z)$
$S(z,y)$	$S(z,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$
$T(y,\bar{w})$	$T(y,\bar{w})$

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
$R(x,y)$	$R(x,y)$	
$S(y,y)$	$S(y,y)$	
$S(y,z)$	$S(y,z)$	
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
$R(x,y)$	~~$R(x,y)$~~ ?	
$S(y,y)$	$S(y,y)$	
$S(y,z)$	$S(y,z)$	
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	
$S(y,z)$	$S(y,z)$	
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	~~$S(y,y)$~~ ?	
$S(y,z)$	$S(y,z)$	
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
$S(y,z)$	$S(y,z)$	
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
$S(y,z)$	~~$S(y,z)$~~	?
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
$S(z,y)$	$S(z,y)$	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
$S(z,y)$	~~$S(z,y)$~~ ?	
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
~~$S(z,y)$~~	~~$S(z,y)$~~	Drop; map $S(z,y)$ to $S(y,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
~~$S(z,y)$~~	~~$S(z,y)$~~	Drop; map $S(z,y)$ to $S(y,y)$
$T(y,\bar{v})$	~~$T(y,\bar{v})$~~ ?	
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
~~$S(z,y)$~~	~~$S(z,y)$~~	Drop; map $S(z,y)$ to $S(y,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$	Keep (cannot map answer variable)
$T(y,\bar{w})$	$T(y,\bar{w})$	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
~~$S(z,y)$~~	~~$S(z,y)$~~	Drop; map $S(z,y)$ to $S(y,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$	Keep (cannot map answer variable)
$T(y,\bar{w})$	~~$T(y,\bar{w})$~~ ?	

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
~~$S(z,y)$~~	~~$S(z,y)$~~	Drop; map $S(z,y)$ to $S(y,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$	Keep (cannot map answer variable)
$T(y,\bar{w})$	$T(y,\bar{w})$	Keep (cannot map answer variable)

CQ Minimisation Example

$$\{R(a,y), R(x,y), S(y,y), S(y,z), S(z,y), T(y,\bar{v}), T(y,\bar{w})\}$$

Can we map the left side homomorphically to the right side?

$R(a,y)$	$R(a,y)$	Keep (cannot map constant a)
~~$R(x,y)$~~	~~$R(x,y)$~~	Drop; map $R(x,y)$ to $R(a,y)$
$S(y,y)$	$S(y,y)$	Keep (no other atom of form $S(t,t)$)
~~$S(y,z)$~~	~~$S(y,z)$~~	Drop; map $S(y,z)$ to $S(y,y)$
~~$S(z,y)$~~	~~$S(z,y)$~~	Drop; map $S(z,y)$ to $S(y,y)$
$T(y,\bar{v})$	$T(y,\bar{v})$	Keep (cannot map answer variable)
$T(y,\bar{w})$	$T(y,\bar{w})$	Keep (cannot map answer variable)

Core: $\exists y.R(a,y) \wedge S(y,y) \wedge T(y,v) \wedge T(y,w)$

CQ Minimisation

Does this algorithm work?

- Is the result minimal?
  Or could it be that some atom that was kept can be dropped later, after some other atoms were dropped?
- Is the result unique?
  Or does the order in which we consider the atoms matter?

Theorem

The CQ minimisation algorithm always produces a core, and this result is unique up to query isomorphisms (bijective renaming of non-result variables).

Proof: exercise

CQ Minimisation: Complexity

Even when considering single atoms, the homomorphism question is NP-hard:

Theorem

Given a conjunctive query Q with an atom A, it is NP-complete to decide if there is a homomorphism from Q to $Q \setminus \{A\}$.

Checking minimality is the dual problem, hence:

Theorem

Deciding if a conjunctive query Q is minimal (that is: a core) is coNP-complete.

However, the size of queries is usually small enough for minimisation to be feasible.

First-Order Query Expressiveness

Queries and Their Expressiveness

Recall:

- Syntax: a query expression q is a word from a query language (algebra expression, logical expression, etc.)
- Semantics: a query mapping $M[q]$ is a function that maps a database instance $\mathcal{I}$ to a database instance $M[q](\mathcal{I})$
- We only study generic queries, which are closed under bijective renaming (isomorphism of databases)

Definition

The expressiveness of a query language is characterised by the set of query mappings that it can express.
Given a query language $\mathcal{L}$, a query mapping M is $\mathcal{L}$-definable if there is a query expression $q \in \mathcal{L}$ such that $M[q] = M$.

We can study expressiveness for all query mappings over all possible databases, or we can restrict attention to a subset of query mappings or to a subset of databases.

Boolean Query Mappings

A Boolean query mapping is a query mapping that returns “true” (usually a database with one table with one empty row) or “false” (usually an empty database).

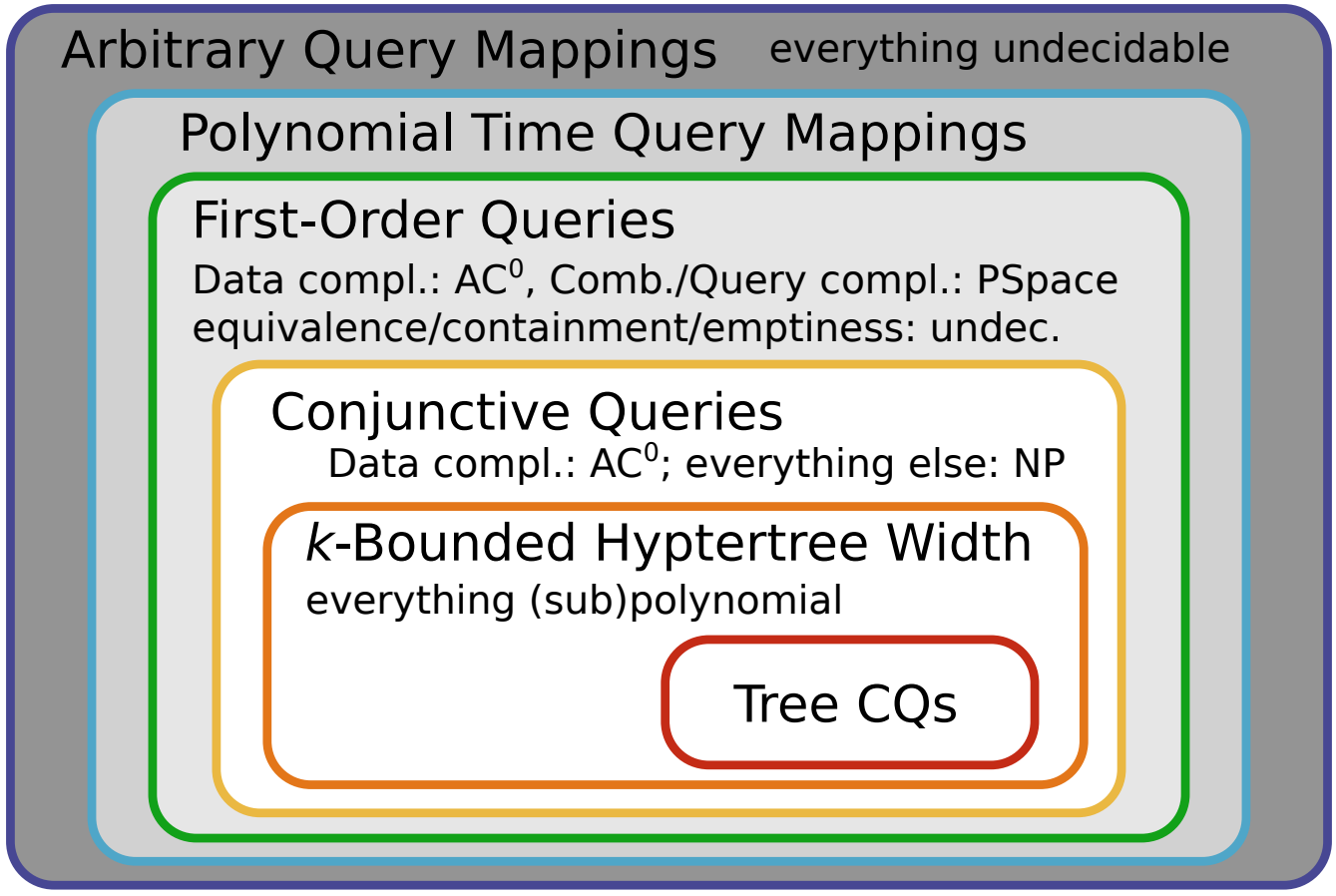
Every Boolean query mapping

- defines set of databases for which it is true
- defines a decision problem over the set of all databases
- could be decidable or undecidable
- if decidable, it may be characterised in terms of complexity

  Note: the “complexity of a mapping” is always “data complexity,” i.e., complexity w.r.t. the size of the input database; the mapping defines the decision problem and is fixed.

Expressivity vs. Complexity

All query mappings that can be expressed in first-order logic are of polynomial complexity, actually in AC^0.



The Limits of FO Queries

Are there polynomial query mappings that cannot be expressed in FO?

The Limits of FO Queries

Are there polynomial query mappings that cannot be expressed in FO?
$\rightsquigarrow$ yes!

We already knew this from previous lectures:

- We learned that $\mathrm{AC}^0 \subset \mathrm{NC}^1 \subseteq \ldots \subseteq \mathrm{P}$
- Hence, there is a problem X in NC^1 that is not in AC^0
- Therefore, the corresponding query mapping M_X is not FO-definable

$\mathrm{AC}^0 \subset \mathrm{NC}^1$ was first shown for the problem $X = \textsc{Parity}$:

- Input: finite relational structure $\mathcal{I}$
- Output: "true" if $\mathcal{I}$ has an even number of domain elements

The original proof is specific to this problem [Ajtai 1981].

Any Other FO-Undefinable Problems?

Yes, many.

Strong evidence from complexity theory:

- If any P-complete problem X were FO-definable,
- then every problem in P could be $\textsc{LogSpace}$-reduced to X
- and then solved in AC^0,
- hence every problem in P could be solved in $\textsc{LogSpace}$,
- that is, $\mathrm{P} = \mathrm{L}$.
- Most experts do not think that this is the case.

Therefore, one would expect all P-hard and similarly all NL-hard problems to not be FO-definable.

$\rightsquigarrow$ How can we see this more directly?

Proving FO-Undefinability

How to show that a query mapping is FO-definable?
$\rightsquigarrow$ Find an FO query that expresses a query mapping

How to show that a query mapping is not FO-definable?

Proving FO-Undefinability

How to show that a query mapping is FO-definable?
$\rightsquigarrow$ Find an FO query that expresses a query mapping

How to show that a query mapping is not FO-definable?
$\rightsquigarrow$ Not so easy . . . important tools:

- Ehrenfeucht-Fraïssé games
- Locality theorems

Ehrenfeucht-Fraïssé Games

A method for showing that certain finite structures cannot be distinguished by certain FO formulas

General idea:

- A game is played on two databases $\mathcal{I}$ and $\mathcal{J}$
- There are two players: the Spoiler and the Duplicator
- The players select elements from $\mathcal{I}$ and $\mathcal{J}$ in each round
- Spoiler wants to show that the two databases are different
- Duplicator wants make the databases appear to be the same

We will always play on finite structures without constant symbols

(remember that one can simulate constants by unary relations with one row)

Playing One Run of an EF Game

A single run of the game has a fixed number r of rounds

Spoiler starts each round, and Duplicator answers:

- Spoiler picks a domain element from $\mathcal{I}$ or from $\mathcal{J}$
- Duplicator picks an element from the other database ($\mathcal{J}$ or $\mathcal{I}$)

$\rightsquigarrow$ One element gets picked from each $\mathcal{I}$ and $\mathcal{J}$ per round

$\rightsquigarrow$ Run of game ends with two lists of elements:
$a_1, \ldots, a_r \in \Delta^{\mathcal{I}}$ and $b_1, \ldots, b_r \in \Delta^{\mathcal{J}}$

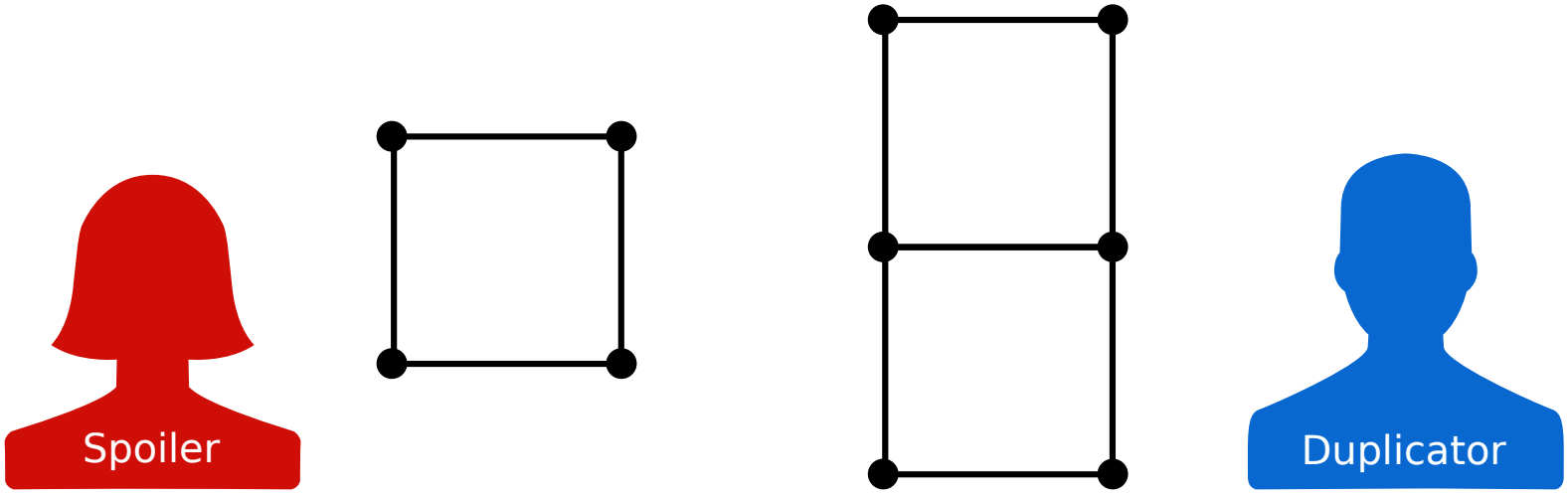
Duplicator wins the run if:

- For all indices i and j, we have $a_i = a_j$ if and only if $b_i = b_j$.
- For all lists of indices $i_i, \ldots, i_n$ and n-ary relation names R, we have $\langle a_{i_1}, \ldots, a_{i_n} \rangle \in R^{\mathcal{I}}$ if and only if $\langle b_{i_1}, \ldots, b_{i_n} \rangle \in R^{\mathcal{J}}$.

  "The substructures induced by the selected elements are isomorphic"

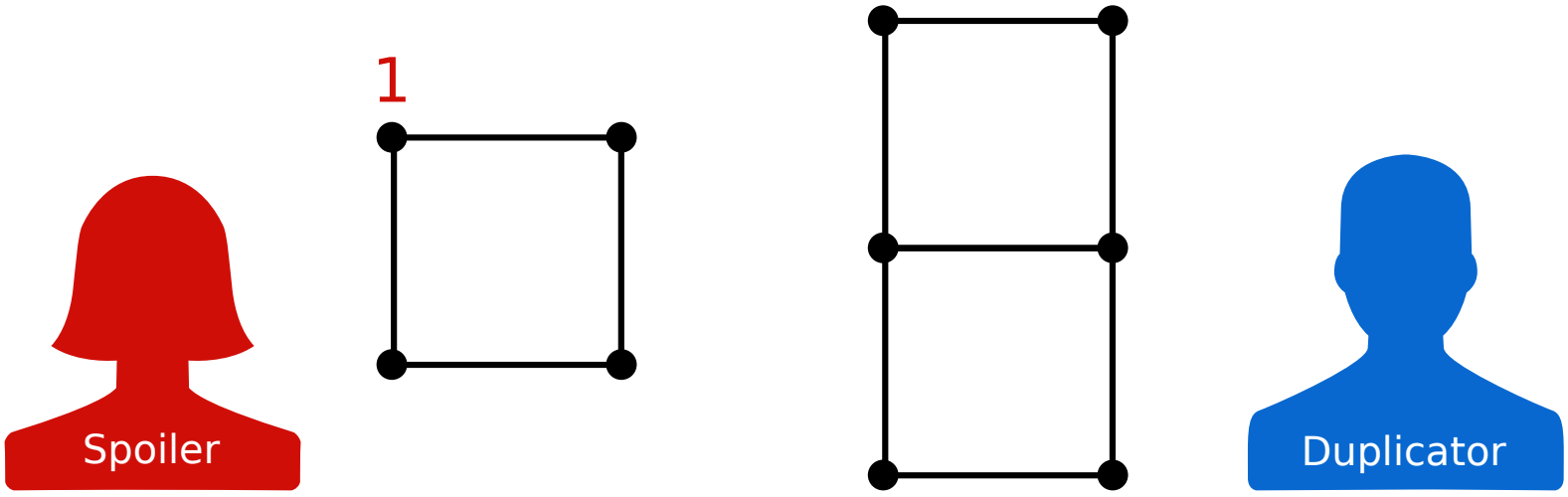
Otherwise Spoiler wins the run.

Example: Run of a Two-Turn EF Game



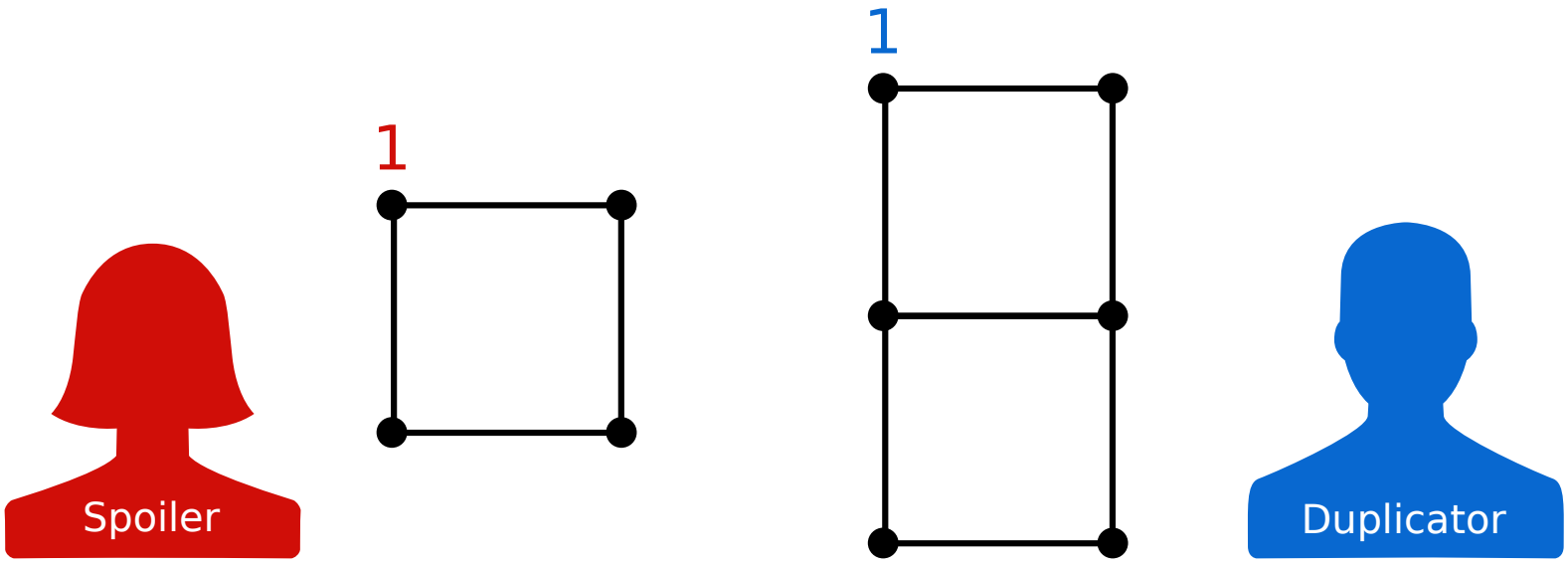
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Two-Turn EF Game



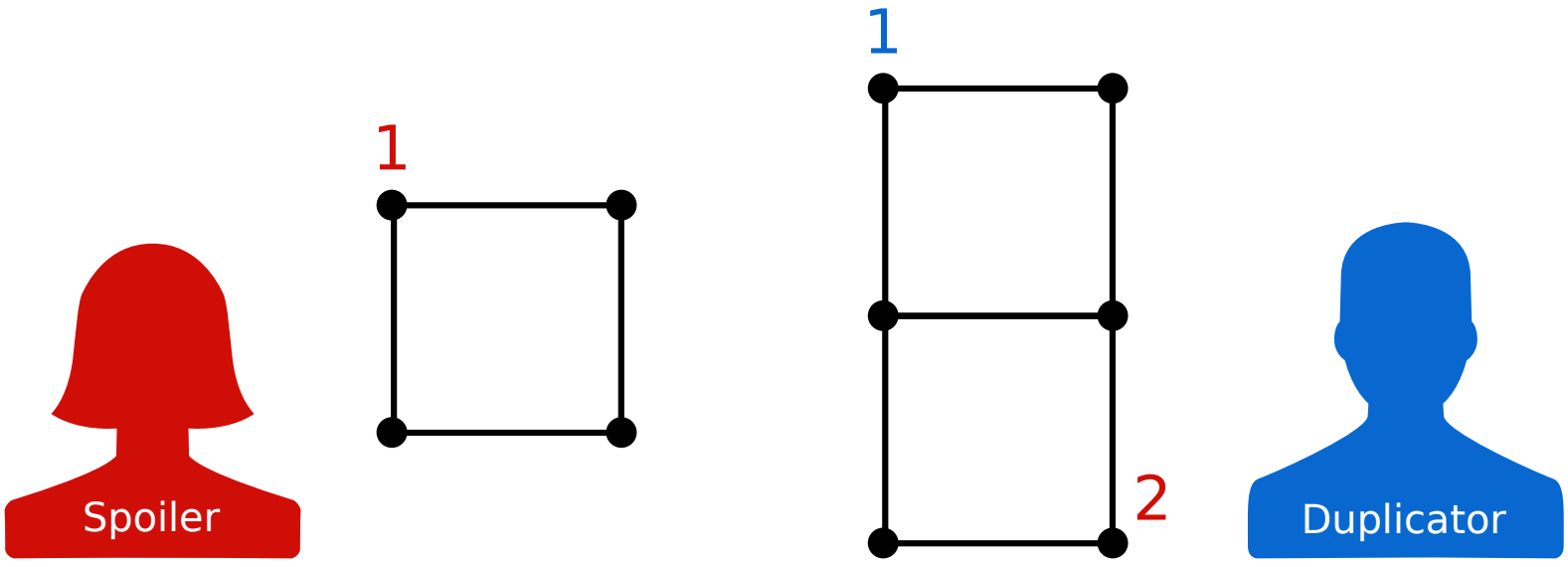
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Two-Turn EF Game



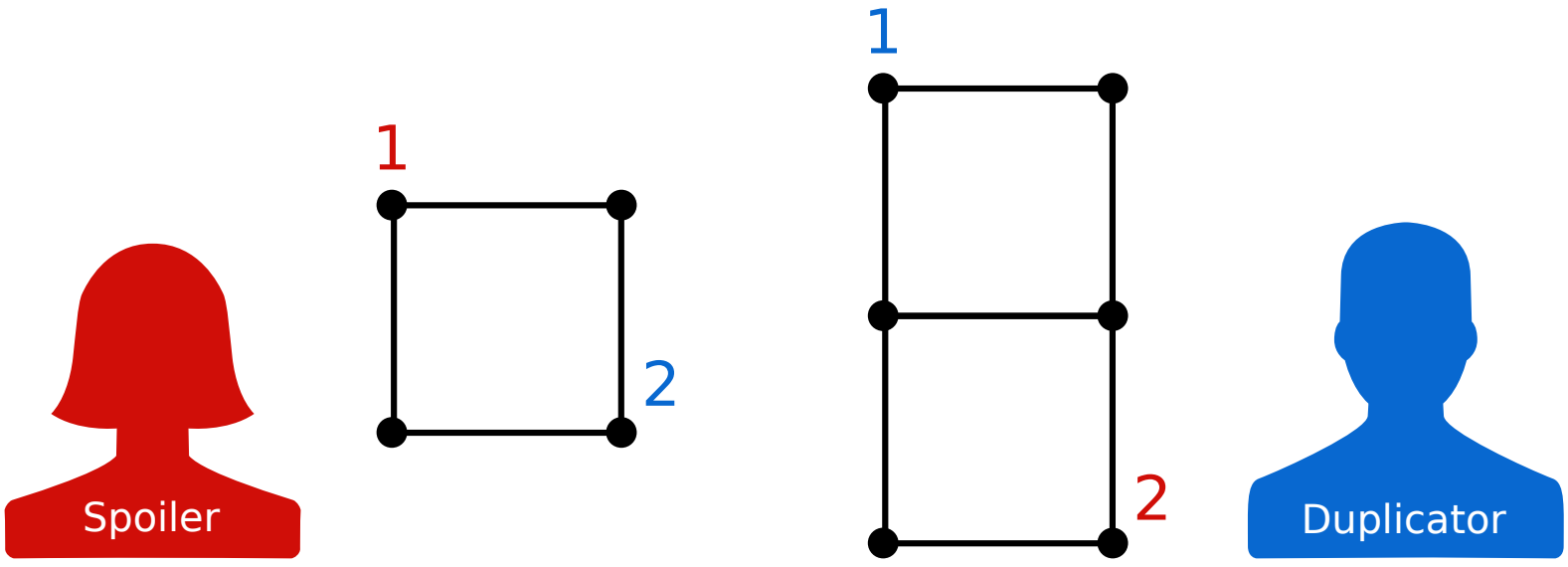
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Two-Turn EF Game



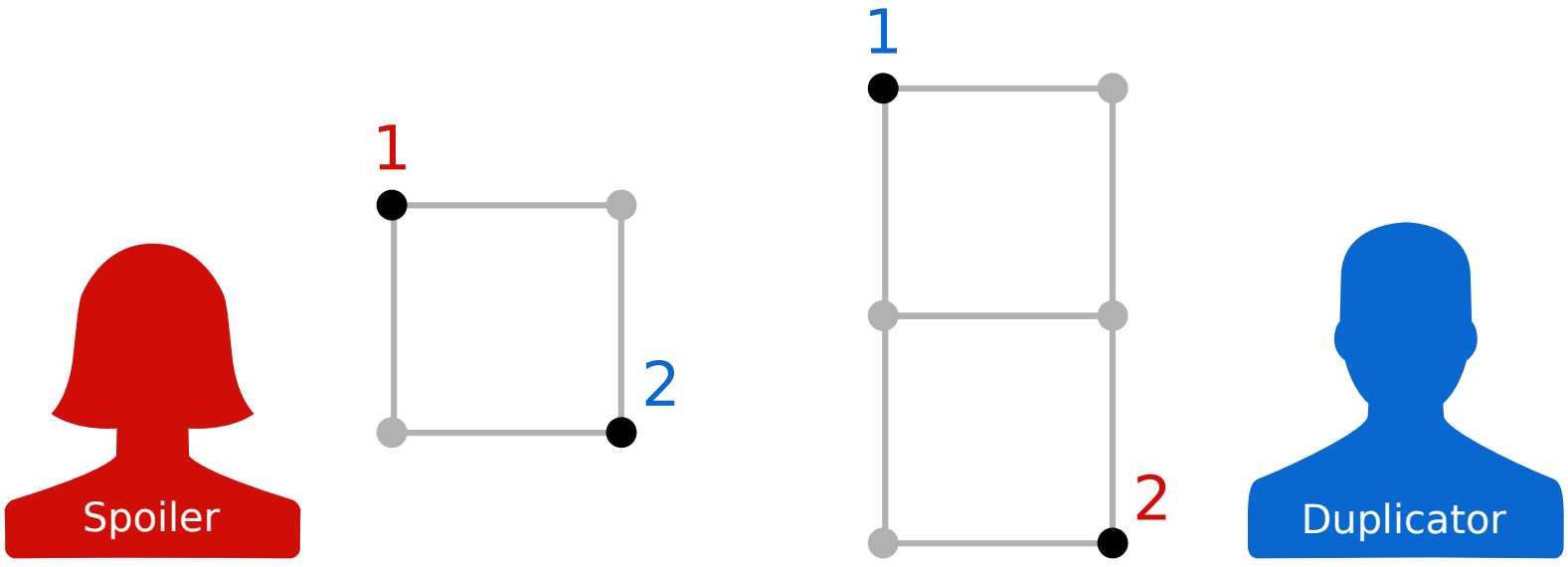
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Two-Turn EF Game



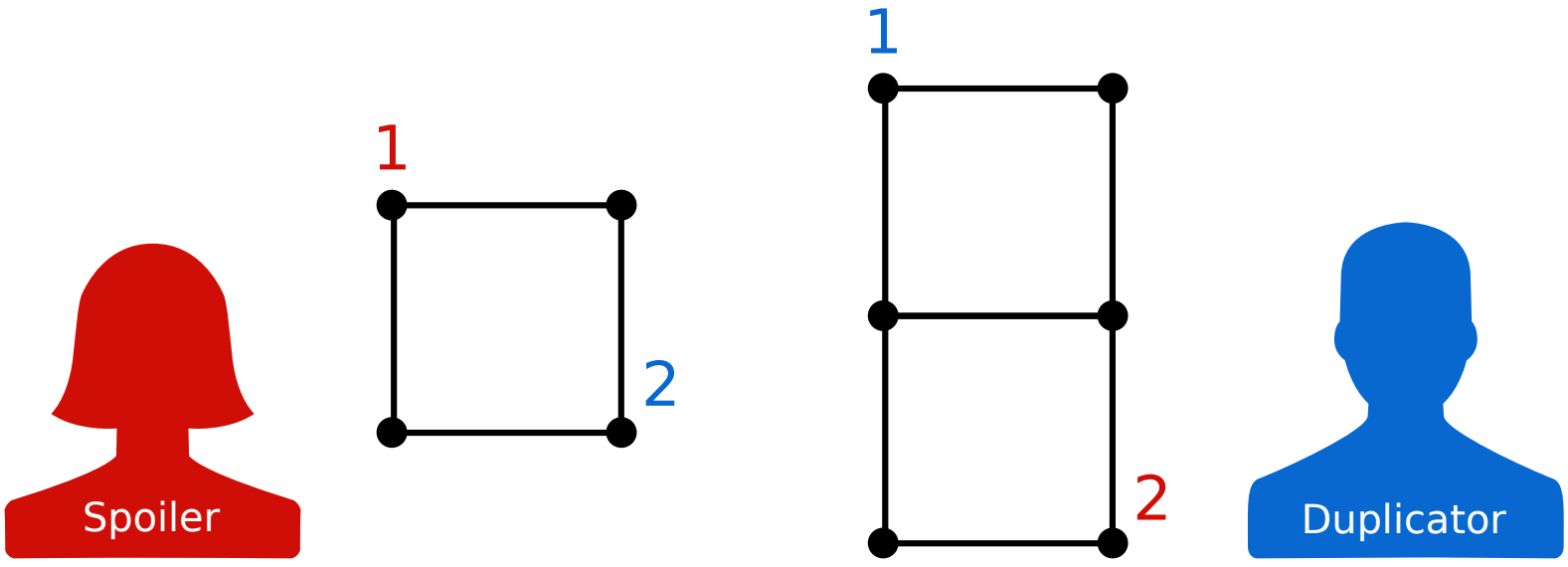
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Two-Turn EF Game



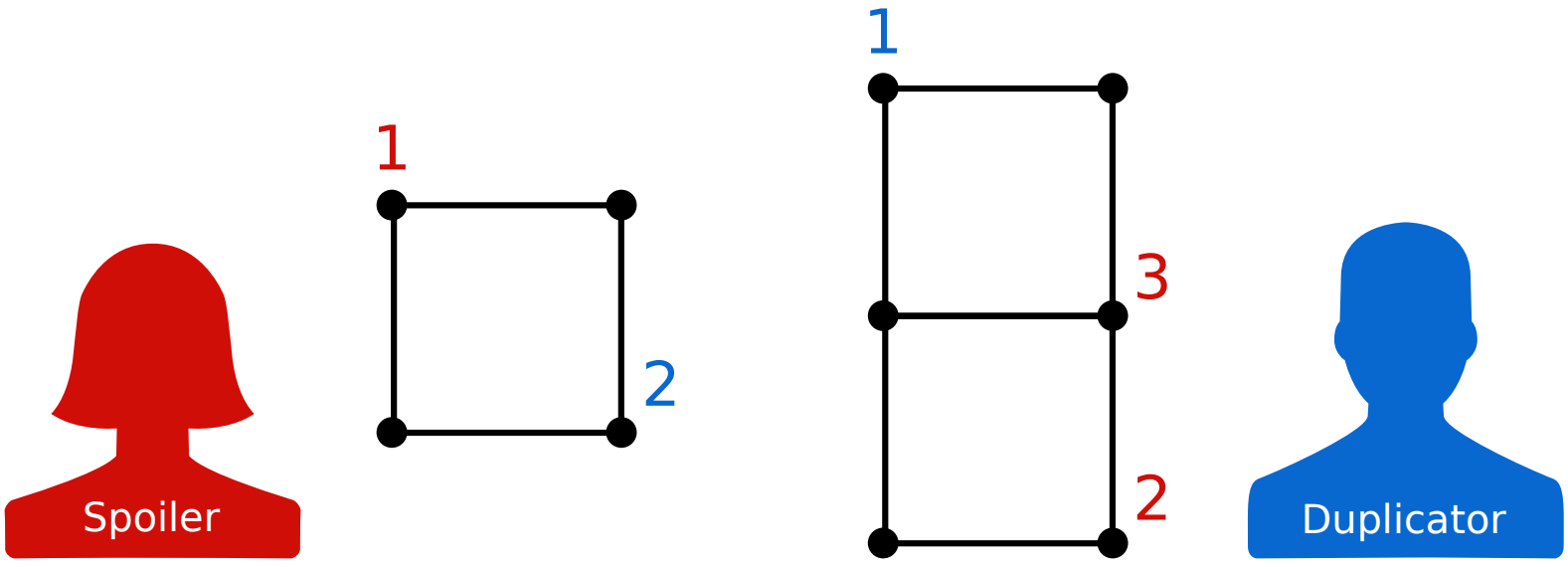
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Three-Turn EF Game



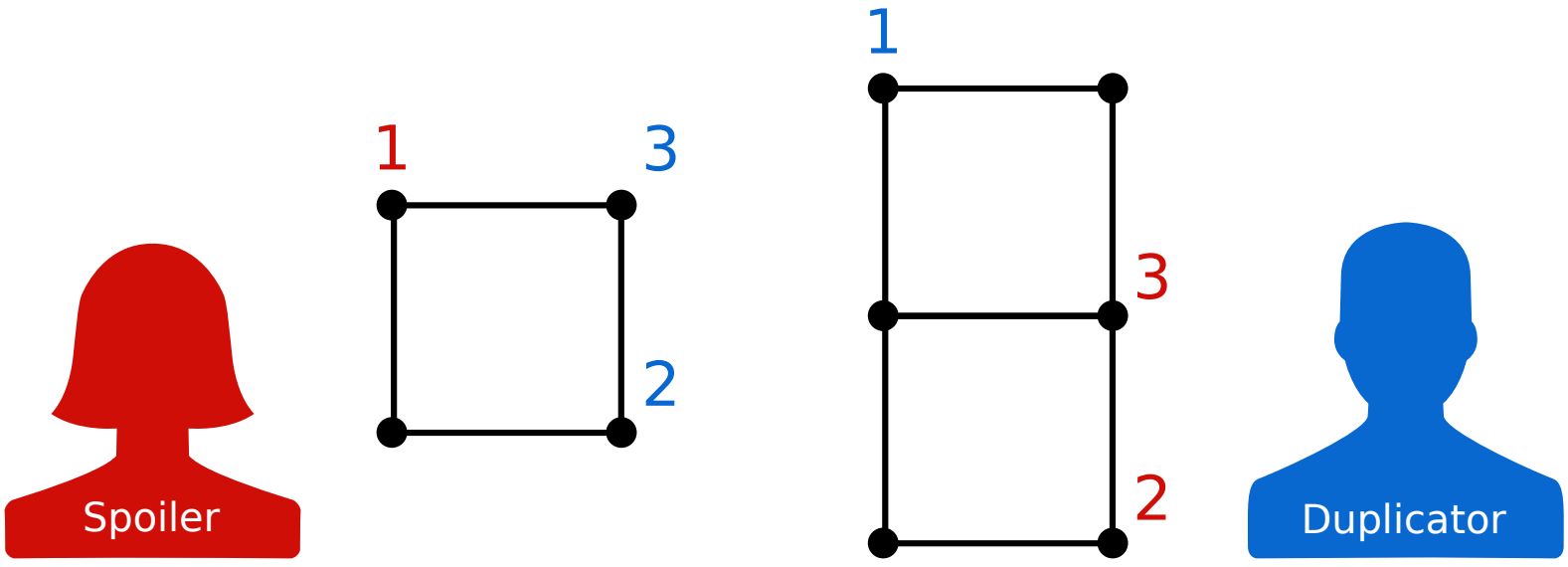
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Three-Turn EF Game



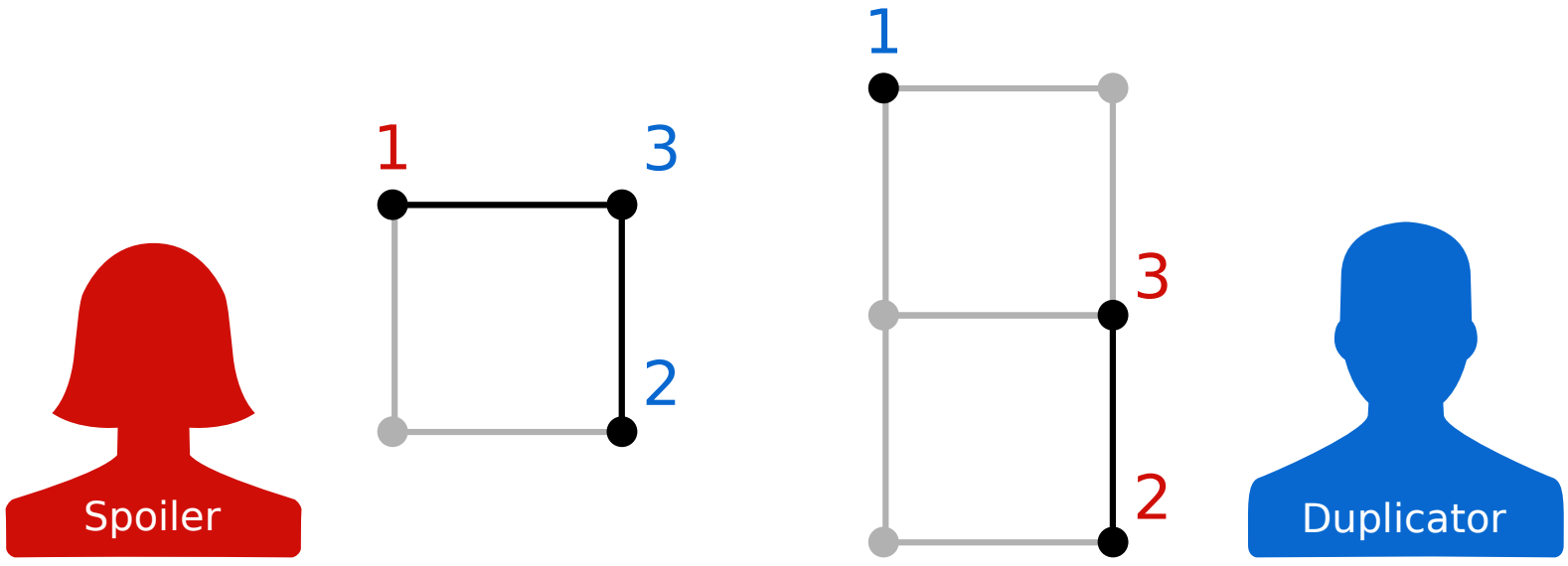
- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Three-Turn EF Game



- edges denote a bi-directional binary predicate
- all edges are the same predicate

Example: Run of a Three-Turn EF Game



- edges denote a bi-directional binary predicate
- all edges are the same predicate

Winning the EF Game

The game is won by whoever has a winning strategy:
A player has a winning strategy if he/she can make sure that he/she will win, whatever the other player is doing

In other words:

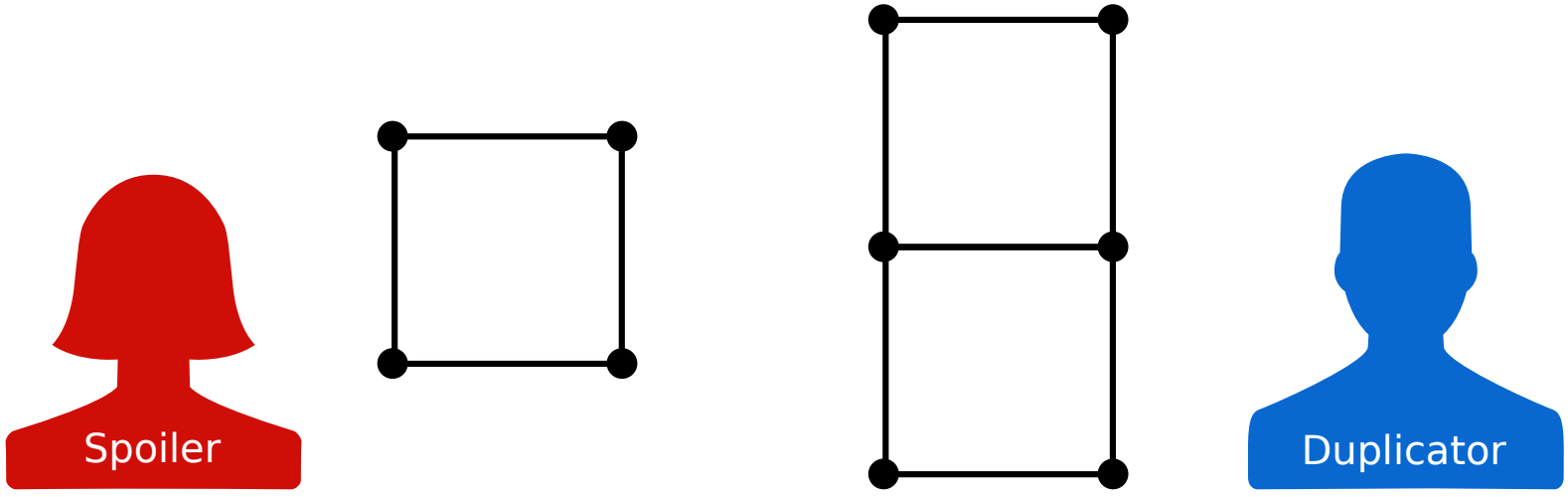
- Duplicator wins if he can duplicate any move that the spoiler makes.
- Spoiler wins if she can spoil any attempt to duplicate her moves.

We write $\mathcal{I} \sim_r \mathcal{J}$ if Duplicator wins the r-round EF game on $\mathcal{I}$ and $\mathcal{J}$.

Observation: given enough moves, the spoiler will always win, unless the structures are isomorphic

Example

Who wins the 2-round game?
Who wins the 3-round game?



- edges denote a bi-directional binary predicate
- all edges are the same predicate

Quantifier Rank

EF games characterise expressivity of FO formulae based on the nesting depth of quantifiers:

Definition

The quantifier rank of a FO formula is the maximal nesting level of quantifiers within the formula.

Examples:

- A formula without quantifiers has quantifier depth 0
- $\exists x.(C(x) \wedge \forall y.(R(x,y) \rightarrow x \approx y) \wedge \exists v.S(x,v))$ has quantifier depth 2

Definition

We write $\mathcal{I} \equiv_r \mathcal{J}$ if $\mathcal{I}$ and $\mathcal{J}$ satisfy the same FO sentences of rank r (or less).

Significance of EF Games

Theorem

For every r, $\mathcal{I}$ and $\mathcal{J}$, the following are equivalent:

- $\mathcal{I} \equiv_r \mathcal{J}$, that is, $\mathcal{I}$ and $\mathcal{J}$ satisfy the same FO sentences of rank r (or less).
- $\mathcal{I} \sim_r \mathcal{J}$, that is, the Duplicator wins the r-round EF game on $\mathcal{I}$ and $\mathcal{J}$.

Therefore, the following are equivalent:

- The query mapping M is FO-definable
- There is an FO sentence φ that defines M
- There is a number r such that, for every $\mathcal{I}$ accepted by M and every $\mathcal{J}$ not accepted by M, the Spoiler wins the r-round EF game on $\mathcal{I}$ and $\mathcal{J}$

Using EF Games to Show FO-Undefinability

How to show that a query mapping M can not be FO-defined:

- Let C_M be the class of all databases recognised by M
- Find sequences of databases $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots \in C_M$ and databases $\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \ldots \notin C_M$, such that $\mathcal{I}_i \sim_i \mathcal{J}_i$

$\rightsquigarrow$ for any formula φ (however large its quantifier rank r), there is a counterexample $\mathcal{I}_r \in C_M$ and $\mathcal{J}_r \notin C_M$ that φ cannot distinguish

Using EF Games to Show FO-Undefinability

How to show that a query mapping M can not be FO-defined:

- Let C_M be the class of all databases recognised by M
- Find sequences of databases $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \ldots \in C_M$ and databases $\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \ldots \notin C_M$, such that $\mathcal{I}_i \sim_i \mathcal{J}_i$

$\rightsquigarrow$ for any formula φ (however large its quantifier rank r), there is a counterexample $\mathcal{I}_r \in C_M$ and $\mathcal{J}_r \notin C_M$ that φ cannot distinguish

Problems:

- How to find such sequences of $\mathcal{I}_i$ and $\mathcal{J}_i$?
  $\rightsquigarrow$ No general strategy exists
- Given suitable sequences, how to show that $\mathcal{I}_i \sim_i \mathcal{J}_i$?
  $\rightsquigarrow$ Can be difficult, but doable for some special cases

Summary and Outlook

Perfect query optimisation is possible for conjunctive queries
$\rightsquigarrow$ Homomorphism problem, similar to query answering
$\rightsquigarrow$ NP-complete

Using this, conjunctive queries can effectively be minimised

FO-queries (and thus CQs) cannot express even all tractable query mappings $\rightsquigarrow$ FO-definability

Showing that a query is not FO-definable requires some creativity
$\rightsquigarrow$ Ehrenfeucht-Fraïssé Games as one approach

Next topics:

- How to really use EF games to get some results?
- If FO cannot express all tractable queries, what can?