# Query Inseparability for Description Logic Knowledge Bases

Elena Botoeva[1] Roman Kontchakov[2] Vladislav Ryzhikov[1]
Frank Wolter[3] Michael Zakharyaschev[2]

[1]Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

[2]Department of Computer Science, Birkbeck, University of London, UK

[3]Department of Computer Science, University of Liverpool, UK

July 21
KR 2014
Vienna

# Query Answering Over Knowledge Bases

**Category**

▷*Product*
- ▷*Shoes*
  - ⋆*Sandals*
    - ⋆*Heeled*
    - ⋆*Platform*
    - ⋆*Classic*
  - . . .
- ▷*Clothing*
- . . .

**Size** Choose size ▽ | **Color** Choose color ▽ | **Brand** Choose brand ▽

**Sandals: (3 products found)**

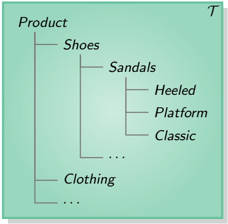
heel_sand
*Heeled*
€79

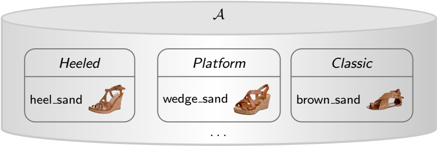
wedge_sand
*Platform*
€69

brown_sand
*Classic*
€50

# Query Answering Over Knowledge Bases

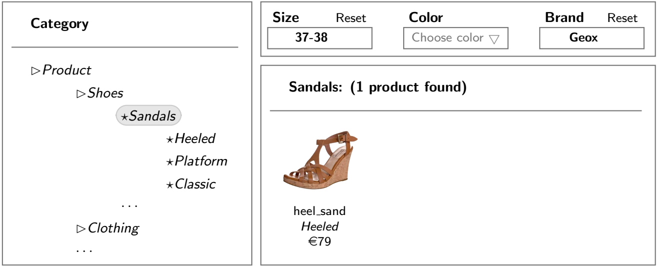

Viewed as a knowledge base $(\mathcal{T}, \mathcal{A})$ and a query $\mathbf{q}$:



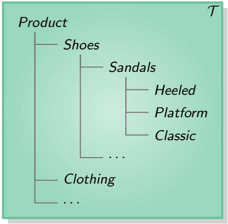



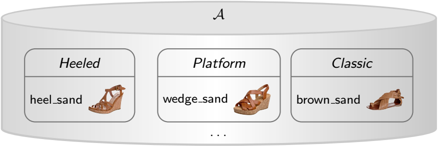

$$\mathbf{q}(x) \leftarrow \mathit{Sandals}(x)$$

# Query Answering Over Knowledge Bases



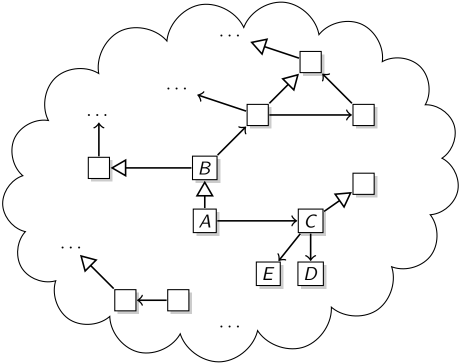

Viewed as a knowledge base $(\mathcal{T}, \mathcal{A})$ and a query **q**:



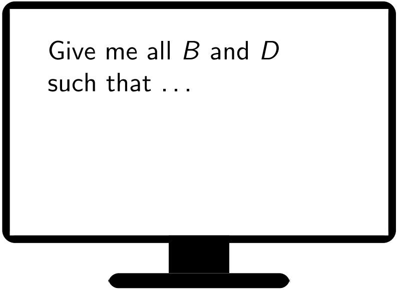




$$\mathbf{q}(x) \leftarrow \begin{array}{c} \mathit{Sandals}(x), \mathit{hasSize}(x, 37), \mathit{hasBrand}(x, \mathrm{geox}) \\ \vee \\ \mathit{Sandals}(x), \mathit{hasSize}(x, 38), \mathit{hasBrand}(x, \mathrm{geox}) \end{array}$$

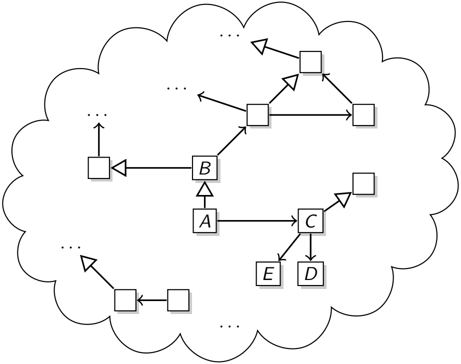
# Motivation: Module Extraction



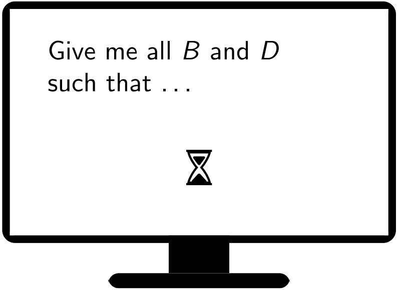

knowledge base $\mathcal{K}$



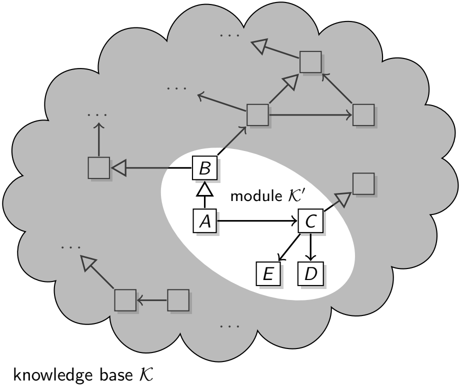
# Motivation: Module Extraction

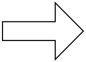



knowledge base $\mathcal{K}$

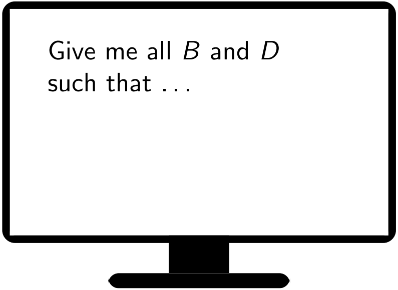


# Motivation: Module Extraction

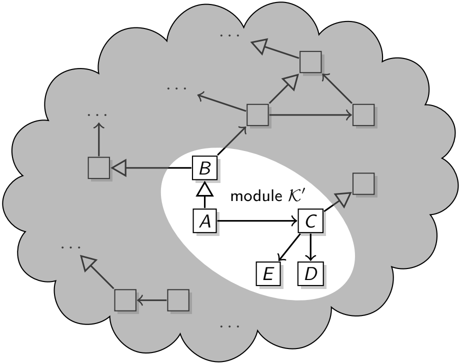


**Module Extraction**

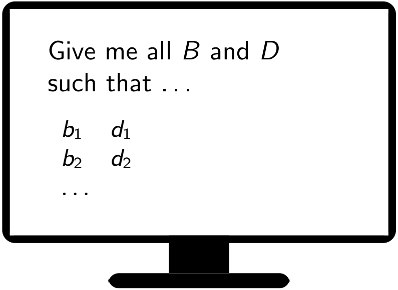


# Motivation: Module Extraction

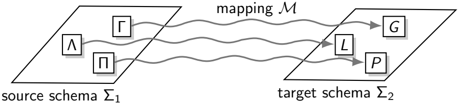


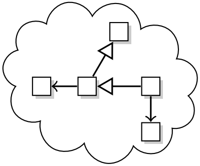

knowledge base $\mathcal{K}$

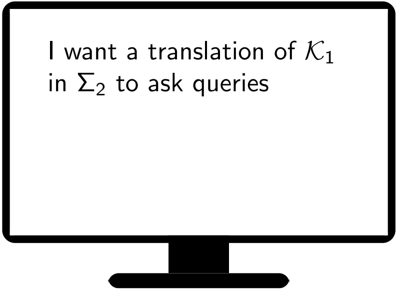
**Module Extraction**



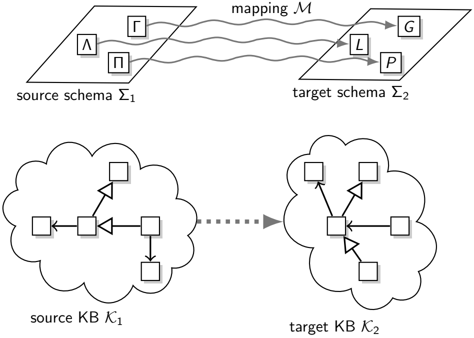
# Motivation: Knowledge Exchange



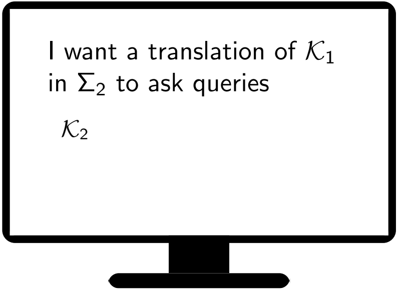



# Motivation: Knowledge Exchange




**Universal UCQ-solution**



# Σ-Query Entailment and Inseparability for KBs

- $\mathcal{K}_1$ **Σ-query entails** $\mathcal{K}_2$ if
$$\mathcal{K}_2 \models \mathbf{q}(\vec{a}) \text{ implies } \mathcal{K}_1 \models \mathbf{q}(\vec{a}),$$
for each **CQ** $\mathbf{q}(\vec{x})$ over $\Sigma$ and each tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{K}_2)$.

# $\Sigma$-Query Entailment and Inseparability for KBs

- $\mathcal{K}_1$ **$\Sigma$-query entails** $\mathcal{K}_2$ if

$$\mathcal{K}_2 \models \mathbf{q}(\vec{a}) \text{ implies } \mathcal{K}_1 \models \mathbf{q}(\vec{a}),$$

  for each **CQ** $\mathbf{q}(\vec{x})$ over $\Sigma$ and each tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{K}_2)$.

- $\mathcal{K}_1$ and $\mathcal{K}_2$ are **$\Sigma$-query inseparable**, $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_2$, if

$$\mathcal{K}_1 \ \Sigma\text{-query entails } \mathcal{K}_2 \quad \text{and} \quad \mathcal{K}_2 \ \Sigma\text{-query entails } \mathcal{K}_1$$

# $\Sigma$-Query Entailment and Inseparability for KBs

- $\mathcal{K}_1$ **$\Sigma$-query entails** $\mathcal{K}_2$ if

$$\mathcal{K}_2 \models \mathbf{q}(\vec{a}) \text{ implies } \mathcal{K}_1 \models \mathbf{q}(\vec{a}),$$

  for each **CQ** $\mathbf{q}(\vec{x})$ over $\Sigma$ and each tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{K}_2)$.

- $\mathcal{K}_1$ and $\mathcal{K}_2$ are **$\Sigma$-query inseparable**, $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_2$, if

$$\mathcal{K}_1 \ \Sigma\text{-query entails } \mathcal{K}_2 \quad \text{and} \quad \mathcal{K}_2 \ \Sigma\text{-query entails } \mathcal{K}_1$$

Then,

- $\mathcal{K}' \subseteq \mathcal{K}$ is a **$\Sigma$-module** of $\mathcal{K}$ if

$$\mathcal{K}' \equiv_\Sigma \mathcal{K}.$$

# Σ-Query Entailment and Inseparability for KBs

- $\mathcal{K}_1$ **Σ-query entails** $\mathcal{K}_2$ if
$$\mathcal{K}_2 \models \mathbf{q}(\vec{a}) \text{ implies } \mathcal{K}_1 \models \mathbf{q}(\vec{a}),$$
for each **CQ** $\mathbf{q}(\vec{x})$ over $\Sigma$ and each tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{K}_2)$.

- $\mathcal{K}_1$ and $\mathcal{K}_2$ are **Σ-query inseparable**, $\mathcal{K}_1 \equiv_\Sigma \mathcal{K}_2$, if
$$\mathcal{K}_1 \text{ } \Sigma\text{-query entails } \mathcal{K}_2 \quad \text{and} \quad \mathcal{K}_2 \text{ } \Sigma\text{-query entails } \mathcal{K}_1$$

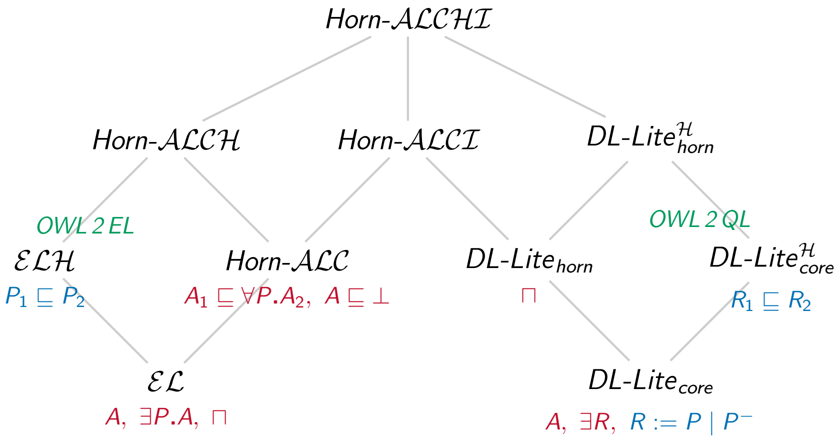
Then,

- $\mathcal{K}' \subseteq \mathcal{K}$ is a **Σ-module** of $\mathcal{K}$ if
$$\mathcal{K}' \equiv_\Sigma \mathcal{K}.$$

- $\mathcal{K}_2$ is a **universal UCQ-solution** for $\mathcal{K}_1$ under $\mathcal{M}$ if
$$\mathcal{K}_2 \equiv_{\Sigma_2} \mathcal{K}_1 \cup \mathcal{M}.$$

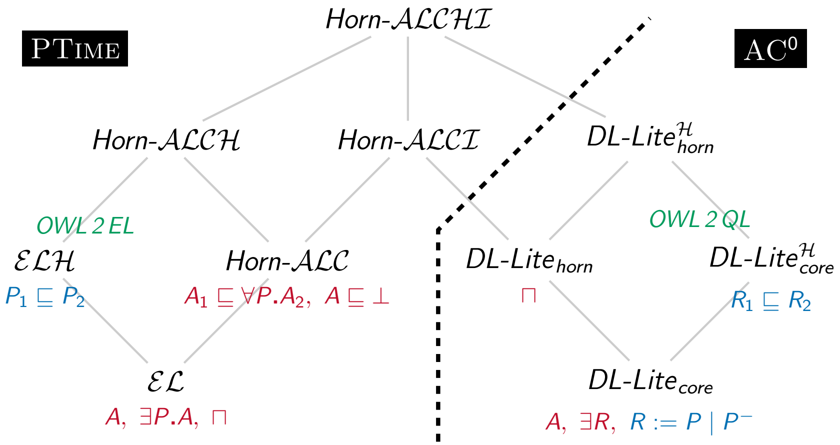
# Horn Description Logics

**Description Logics (DLs)** represent knowledge in terms of **concepts** (unary predicates) and **roles** (binary predicates).



# Horn Description Logics

**Description Logics (DLs)** represent knowledge in terms of **concepts** (unary predicates) and **roles** (binary predicates).




Data complexity of CQ-answering

# How We Tackle Σ-Query Entailment

We rely on two fundamental instruments:

1. **Materialisation**, as an abstract way to characterize all answers to CQs over a KB.

   A materialisation of a KB $\mathcal{K}$ is an interpretation $\mathcal{M}$ such that

   $$\mathcal{K} \models \mathbf{q}(\vec{a}) \quad \text{iff} \quad \mathcal{M} \models \mathbf{q}(\vec{a}),$$

   for each CQ $\mathbf{q}(\vec{x})$ and each tuple $\vec{a} \subseteq \mathsf{ind}(\mathcal{K})$.

2. **Reachability Games**, as a technique for obtaining upper-bounds.

# Materialisations

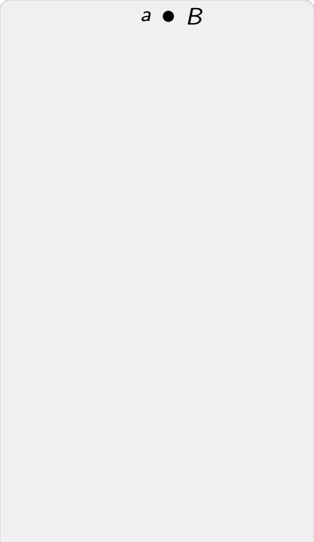
Horn DLs enjoy **materialisations** (chase, canonical models).

# Materialisations

Horn DLs enjoy **materialisations** (chase, canonical models).

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

$\mathcal{A} = \{B(a)\}$




Materialisation $\mathcal{M}$ of $\mathcal{K}$

# Materialisations

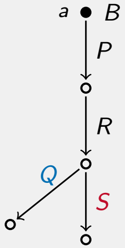
Horn DLs enjoy **materialisations** (chase, canonical models).

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

$\mathcal{A} = \{B(a)\}$

$\mathcal{T} = \{B \sqsubseteq \exists P.\exists R.(\exists S \sqcap \exists Q)$

$$\forall x.\ (B(x) \to \exists y, z, u, v.\ \ P(x,y), R(y,z), S(z,u), Q(z,v))$$




Materialisation $\mathcal{M}$ of $\mathcal{K}$

# Materialisations

Horn DLs enjoy **materialisations** (chase, canonical models).

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

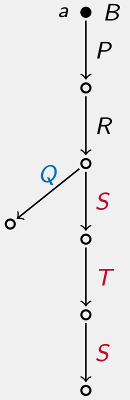
$\mathcal{A} = \{B(a)\}$

$\mathcal{T} = \{B \sqsubseteq \exists P.\exists R.(\exists S \sqcap \exists Q)$

$\exists S^{-} \sqsubseteq \exists T.\exists S$

$$\forall x.\ \big(B(x) \rightarrow \exists y, z, u, v.\ \ P(x, y), R(y, z), S(z, u), Q(z, v)\big)$$

$$\forall x.\ \big(\exists u.\ S(u, x) \rightarrow \exists y, z.\ \ T(x, y), S(y, z)\big)$$




Materialisation $\mathcal{M}$ of $\mathcal{K}$

# Materialisations

Horn DLs enjoy **materialisations** (chase, canonical models).

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

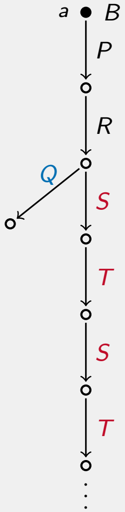
$\mathcal{A} = \{B(a)\}$

$\mathcal{T} = \{B \sqsubseteq \exists P.\exists R.(\exists S \sqcap \exists Q)$

$\exists S^{-} \sqsubseteq \exists T.\exists S$

$$\forall x.\ (B(x) \to \exists y, z, u, v.\ \ P(x, y), R(y, z), S(z, u), Q(z, v))$$

$$\forall x.\ (\exists u.\ S(u, x) \to \exists y, z.\ T(x, y), S(y, z))$$




Materialisation $\mathcal{M}$ of $\mathcal{K}$

# Materialisations

Horn DLs enjoy **materialisations** (chase, canonical models).

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

$\mathcal{A} = \{B(a)\}$

$\mathcal{T} = \{B \sqsubseteq \exists P.\exists R.(\exists S \sqcap \exists Q)$

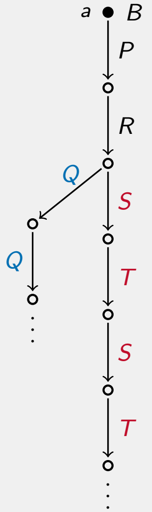
$\exists S^- \sqsubseteq \exists T.\exists S$

$\exists Q^- \sqsubseteq \exists Q\}$

$$\forall x.\ (B(x) \rightarrow \exists y, z, u, v.\ \ P(x,y), R(y,z), S(z,u), Q(z,v))$$

$$\forall x.\ (\exists u.\ S(u,x) \rightarrow \exists y, z.\ \ T(x,y), S(y,z))$$

$$\forall x.\ (\exists y.\ Q(y,x) \rightarrow \exists z.\ \ Q(x,z))$$




Materialisation $\mathcal{M}$ of $\mathcal{K}$

# Materialisations

Horn DLs enjoy **materialisations** (chase, canonical models).

Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$

$\mathcal{A} = \{B(a)\}$

$\mathcal{T} = \{B \sqsubseteq \exists P.\exists R.(\exists S \sqcap \exists Q)$

$\exists S^- \sqsubseteq \exists T.\exists S$

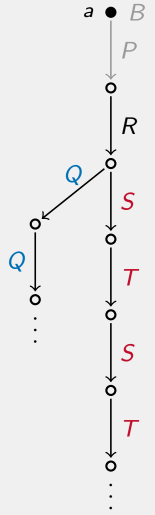
$\exists Q^- \sqsubseteq \exists Q\}$

$\Sigma = \{Q, R, S, T\}$

$$\forall x.\ (B(x) \rightarrow \exists y, z, u, v.\ \ P(x,y), R(y,z), S(z,u), Q(z,v))$$

$$\forall x.\ (\exists u.\ S(u,x) \rightarrow \exists y, z.\ \ T(x,y), S(y,z))$$

$$\forall x.\ (\exists y.\ Q(y,x) \rightarrow \exists z.\ \ Q(x,z))$$




Projection $\mathcal{M}^{\Sigma}$ of $\mathcal{M}$ on $\Sigma$

# Semantic Characterization of Σ-Query Entailment

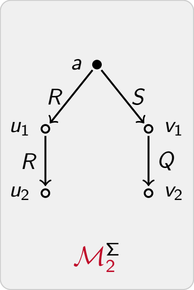
Assume KBs $\mathcal{K}_1$ and $\mathcal{K}_2$ with materialisations $\mathcal{M}_1$ and $\mathcal{M}_2$.

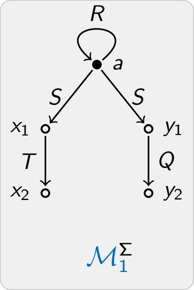
**Theorem**

*$\mathcal{K}_1$ Σ-query entails $\mathcal{K}_2$*
*iff*
*$\mathcal{M}_2$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_1$.*

# Homomorphisms as Games

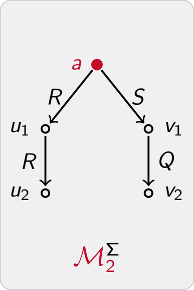
The problem of finding a homomorphism can be seen as a game.

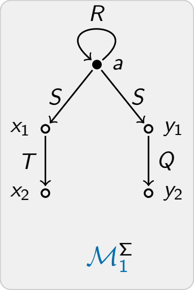


# Homomorphisms as Games

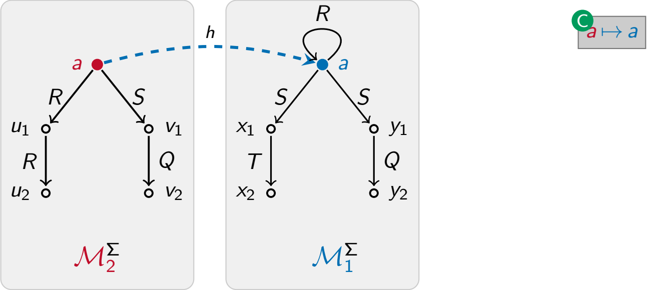
The problem of finding a homomorphism can be seen as a game.



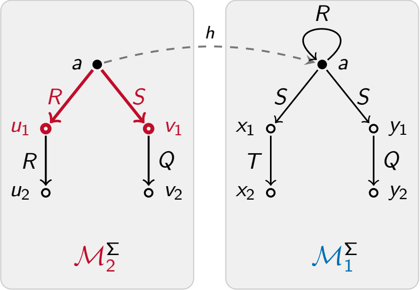
# Homomorphisms as Games

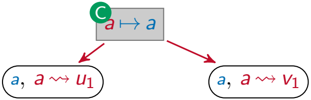
The problem of finding a homomorphism can be seen as a game.



# Homomorphisms as Games

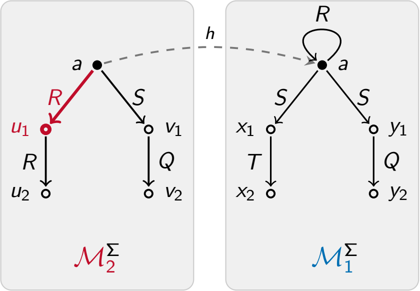
The problem of finding a homomorphism can be seen as a game.

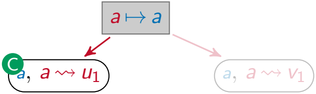


# Homomorphisms as Games

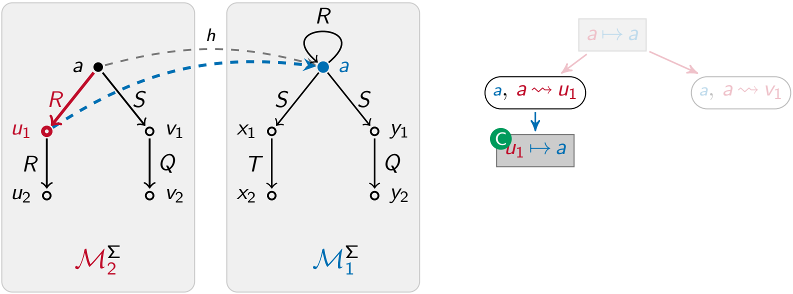
The problem of finding a homomorphism can be seen as a game.



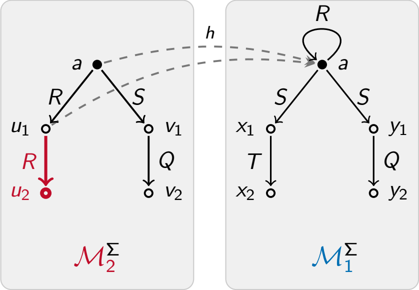
# Homomorphisms as Games

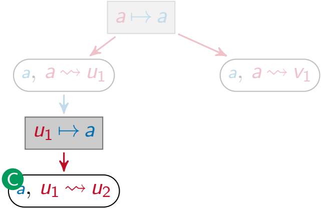
The problem of finding a homomorphism can be seen as a game.



# Homomorphisms as Games

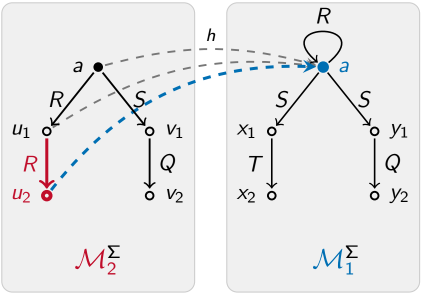
The problem of finding a homomorphism can be seen as a game.

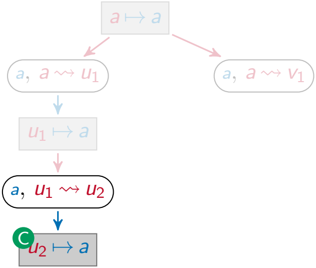


# Homomorphisms as Games

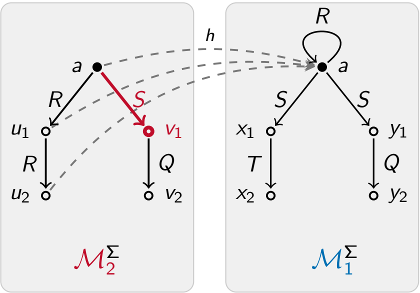
The problem of finding a homomorphism can be seen as a game.

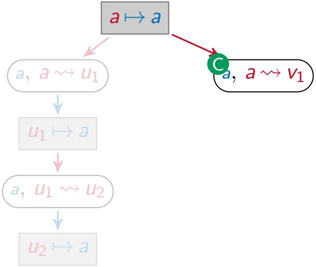


# Homomorphisms as Games

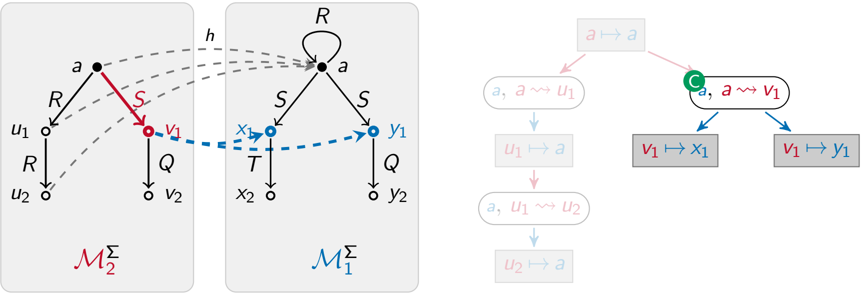
The problem of finding a homomorphism can be seen as a game.



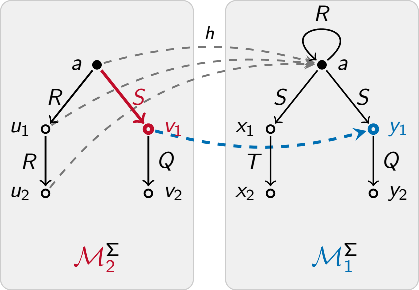
# Homomorphisms as Games

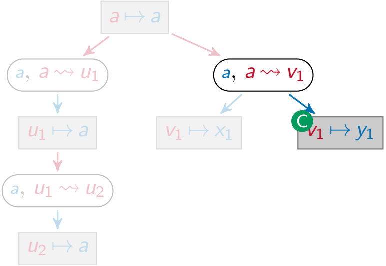
The problem of finding a homomorphism can be seen as a game.



# Homomorphisms as Games

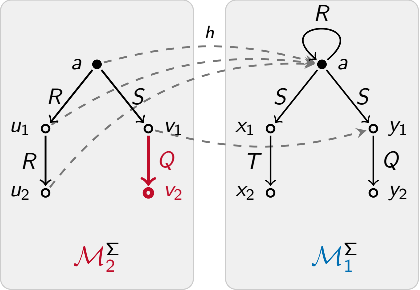
The problem of finding a homomorphism can be seen as a game.

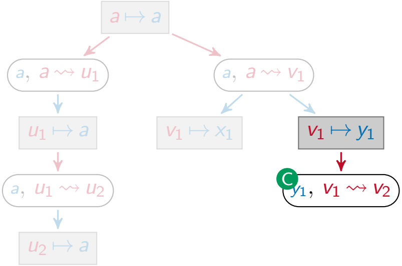


# Homomorphisms as Games

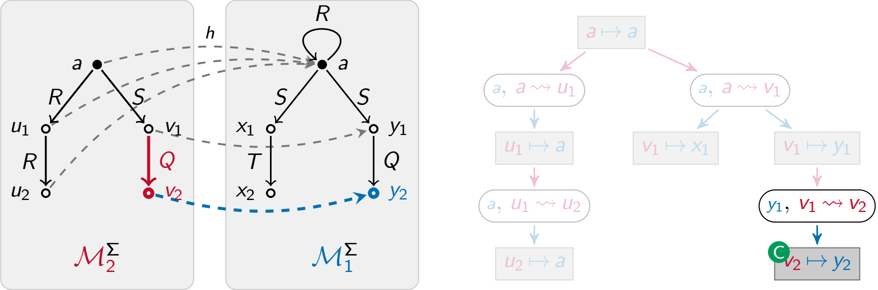
The problem of finding a homomorphism can be seen as a game.



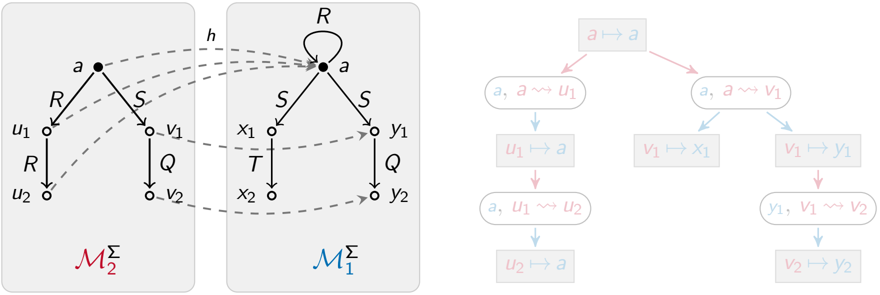
# Homomorphisms as Games

The problem of finding a homomorphism can be seen as a game.



# Homomorphisms as Games

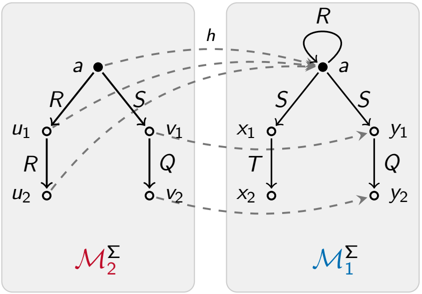
The problem of finding a homomorphism can be seen as a game.

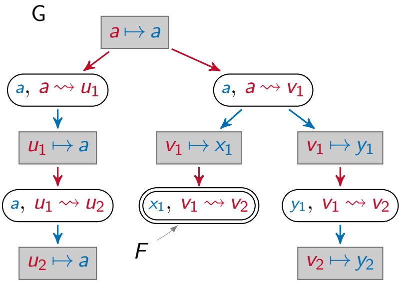



This game can be straightforwardly encoded as a **Reachability Game** $(\mathsf{G}, F)$.

# Homomorphisms as Games

The problem of finding a homomorphism can be seen as a game.

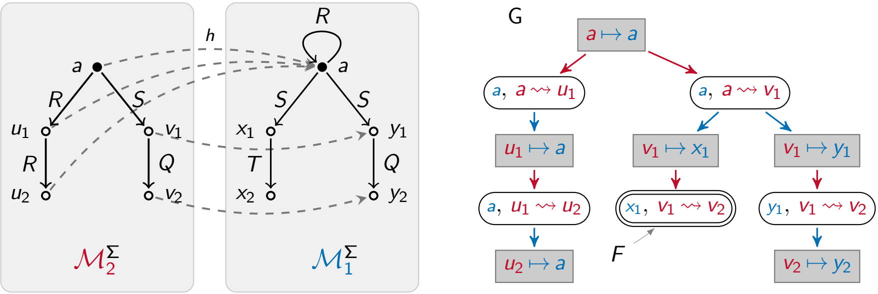



This game can be straightforwardly encoded as a **Reachability Game** $(\mathsf{G}, F)$.

# Homomorphisms as Games

The problem of finding a homomorphism can be seen as a game.



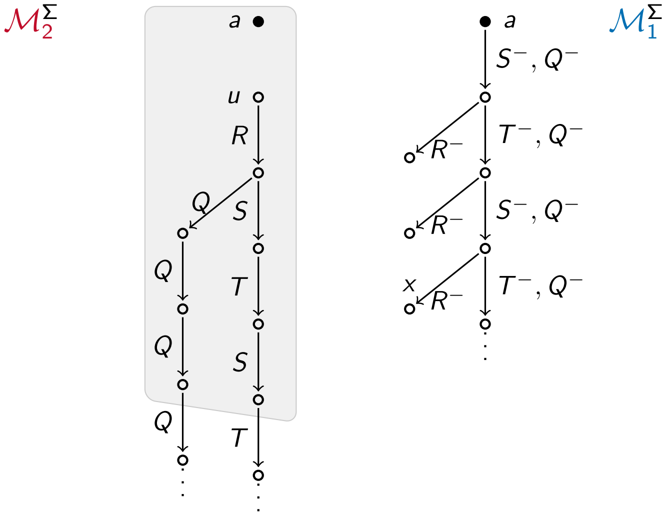

This game can be straightforwardly encoded as a **Reachability Game** $(G, F)$.

However such encoding is impossible in practice:

1. Materialisations are infinite, in general;
2. Or of exponential size, even for $DL\text{-}Lite_{core}$.

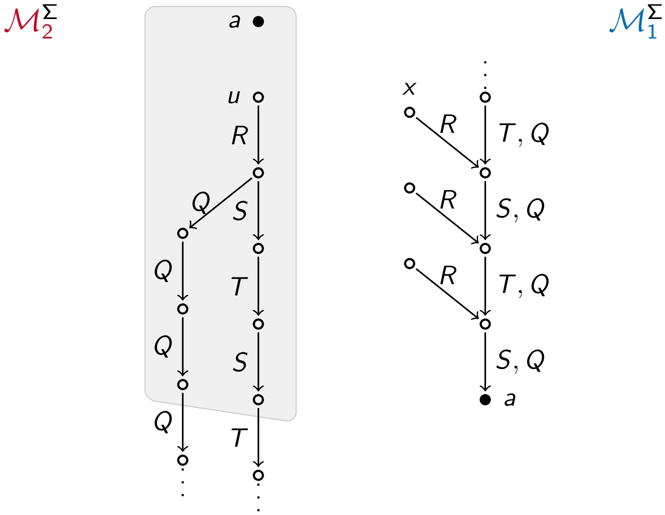
# Homomorphisms on Infinite Materialisations

$\mathcal{M}_2$ is finitely Σ-homomorphically embeddable into $\mathcal{M}_1$.



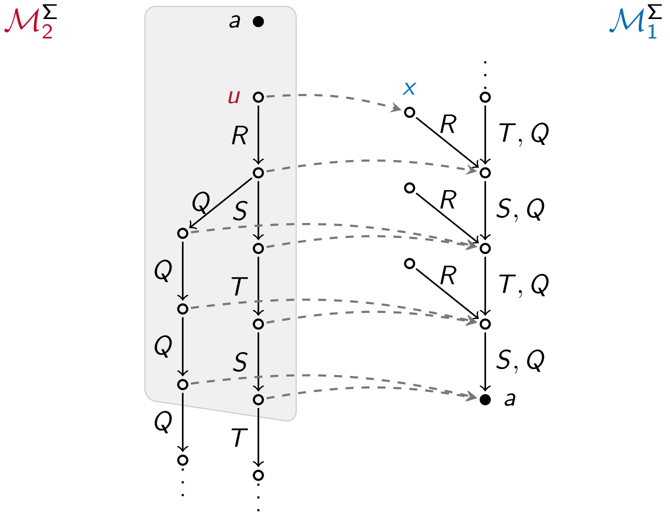
# Homomorphisms on Infinite Materialisations

$\mathcal{M}_2$ is finitely $\Sigma$-homomorphically embeddable into $\mathcal{M}_1$.



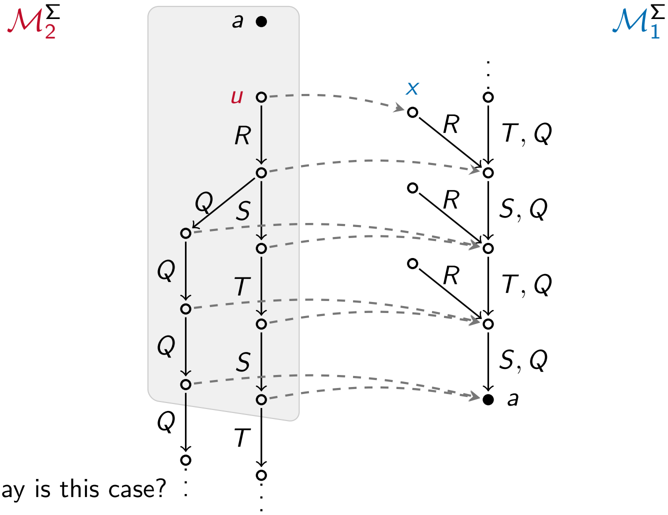
# Homomorphisms on Infinite Materialisations

$\mathcal{M}_2$ is finitely $\Sigma$-homomorphically embeddable into $\mathcal{M}_1$.



# Homomorphisms on Infinite Materialisations

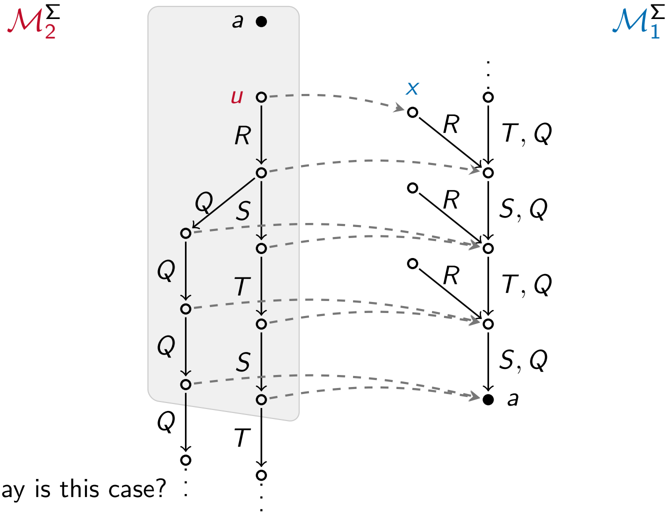
$\mathcal{M}_2$ is finitely $\Sigma$-homomorphically embeddable into $\mathcal{M}_1$.




How to play is this case?

# Homomorphisms on Infinite Materialisations

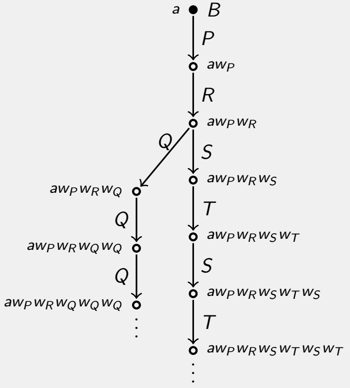
$\mathcal{M}_2$ is finitely $\Sigma$-homomorphically embeddable into $\mathcal{M}_1$.



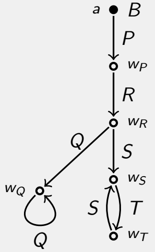

How to play is this case?

⟹ Instead of materialisations, we play on **finite generating structures**.

# Finite Generating Structures

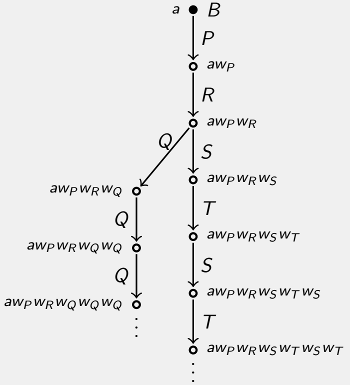



Materialisation $\mathcal{M}$

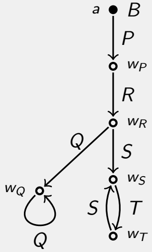



Generating structure $\mathcal{G}$ for $\mathcal{M}$

# Finite Generating Structures




Materialisation $\mathcal{M}$




Generating structure $\mathcal{G}$ for $\mathcal{M}$

[$\mathcal{EL}$, $\mathcal{ELH}$], [*DL-Lite*$_{core}$, *DL-Lite*$^{\mathcal{H}}_{horn}$]: generating structures of **polynomial** size
[*Horn*-$\mathcal{ALC}$, *Horn*-$\mathcal{ALCHI}$]: generating structures **exponential** size

# The Upper Bound

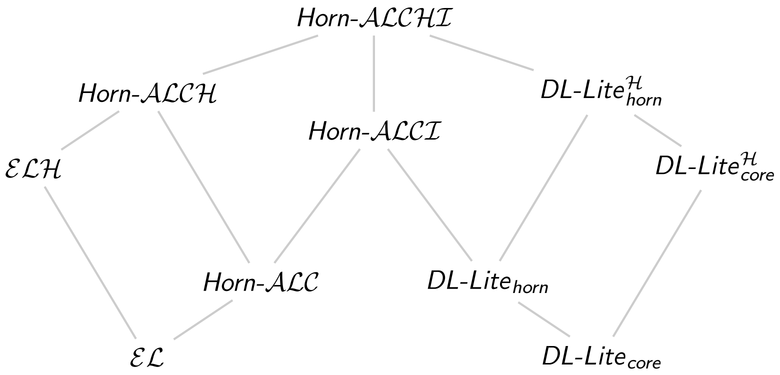
For KBs $\mathcal{K}_1$, $\mathcal{K}_2$, and a signature $\Sigma$, we construct a reachability game $G_\Sigma(\mathcal{G}_2, \mathcal{G}_1) = (\mathsf{G}, F)$ such that

Player 1 has a winning strategy against Player 2 in $G_\Sigma(\mathcal{G}_2, \mathcal{G}_1)$
iff
$\mathcal{M}_2$ is finitely $\Sigma$-homomorphically embeddable into $\mathcal{M}_1$.

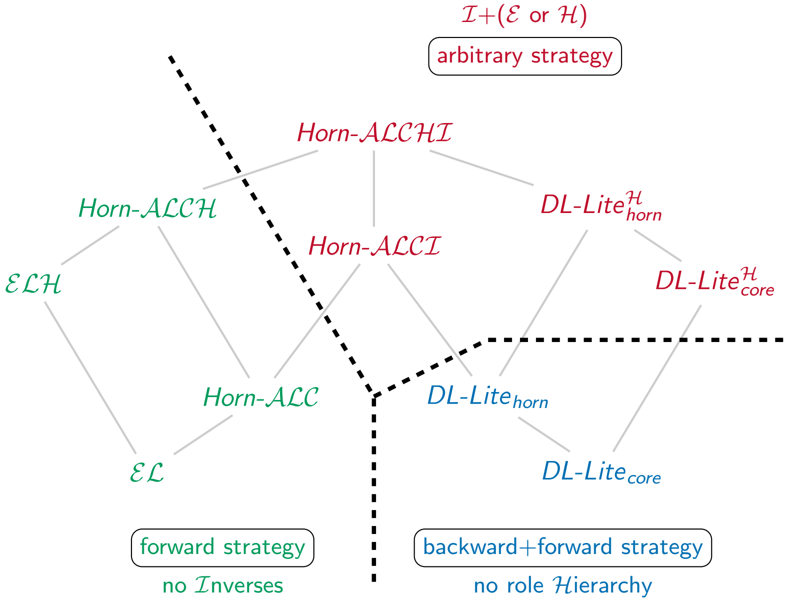
where the size of G is

- Polynomial in the size of $\mathcal{G}_2$ and $\mathcal{G}_1$,
  if the logic admits only **forward strategies** (without inverses)
- Polynomial in the size of $\mathcal{G}_2$ and $\mathcal{G}_1$,
  if the logic admits only **backward+forward strategies** (*DL-Lite* without $\mathcal{H}$)
- Exponential in the size of $\mathcal{G}_2$,
  if the logic admits **arbitrary strategies**.

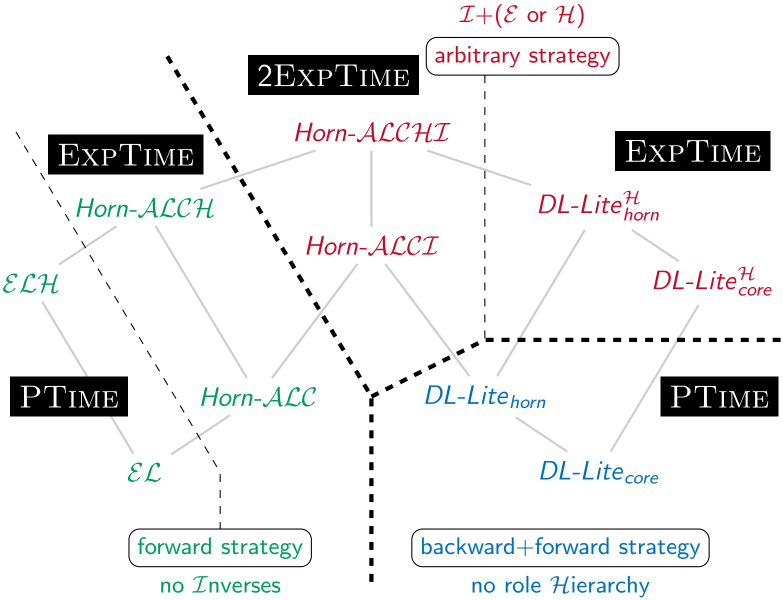
# The Summary of the Results



# The Summary of the Results



# The Summary of the Results



# Future Work

- approximate module extraction using forward strategies
- KB Inseparability for more expressive DLs:
  *Horn*-$\mathcal{SHIQ}$ and $\mathcal{ALC}$
- TBox Inseparability

Thank you
for your attention!