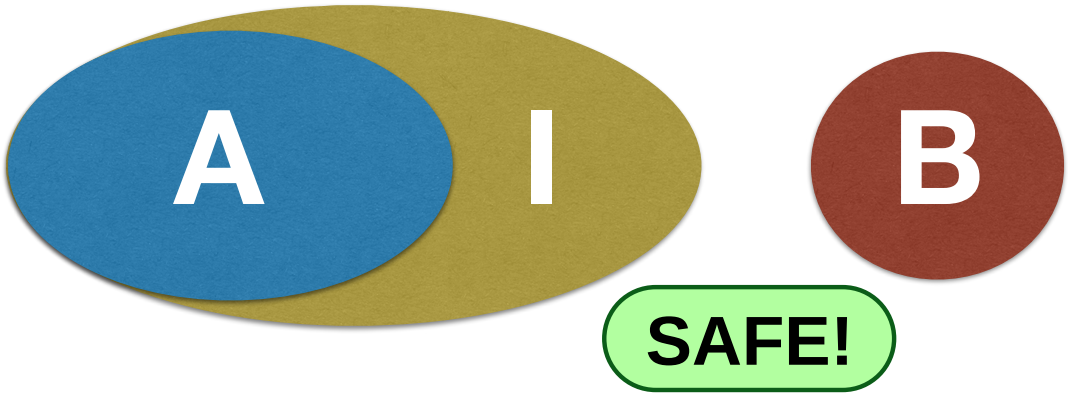
Uniform Interpolants and Model Completions in Formal Verification of Infinite-State Systems

Alessandro Gianola

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy

iPRA 2022 - FLoC 2022

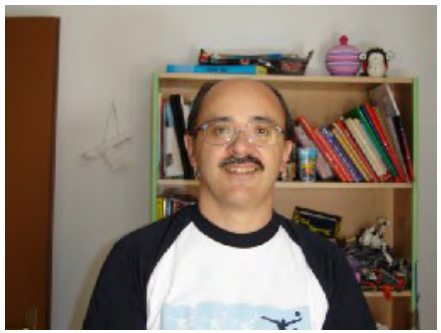
August 11, 2022



Acknowledgments

Diego Calvanese

Silvio Ghilardi



Marco Montali

Andrey Rivkin

Outline

Outline

Topic of the Talk

- Stronger form of (Craig) Interpolation: **Uniform Interpolation**.

Topic of the Talk

- Stronger form of (Craig) Interpolation: **Uniform Interpolation**.
- We focus on **Quantifier-free** Uniform Interpolants (UIs).

Topic of the Talk

- Stronger form of (Craig) Interpolation: **Uniform Interpolation**.
- We focus on **Quantifier-free** Uniform Interpolants (UIs).
- UIs are a 'weak form' of Quantifier Elimination (QE):

Topic of the Talk

- Stronger form of (Craig) Interpolation: **Uniform Interpolation**.
- We focus on **Quantifier-free** Uniform Interpolants (UIs).
- UIs are a 'weak form' of Quantifier Elimination (QE):
  - The UI of a quantified formula $\exists \underline{e}\, \phi(\underline{e}, \underline{x})$ is the ***strongest*** formula **implied** by $\exists \underline{e}\, \phi(\underline{e}, \underline{x})$;

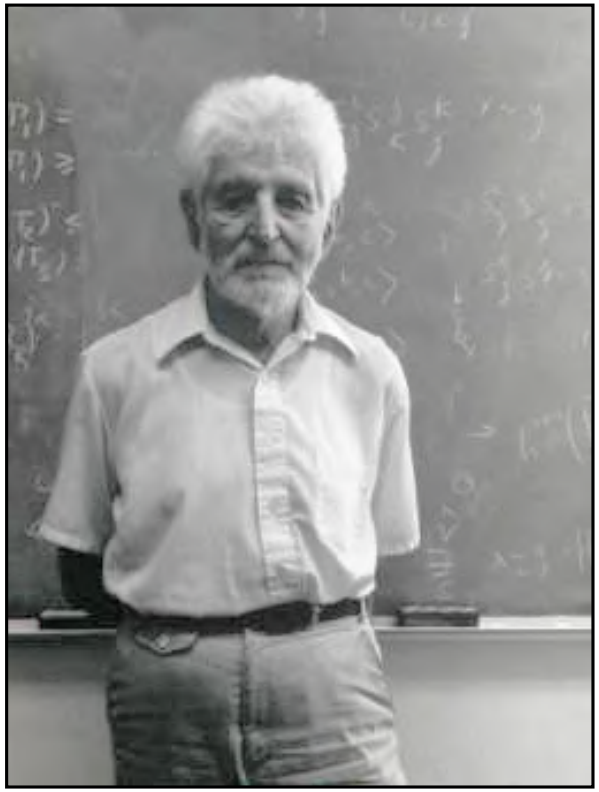
Topic of the Talk

- Stronger form of (Craig) Interpolation: **Uniform Interpolation**.
- We focus on **Quantifier-free** Uniform Interpolants (UIs).
- UIs are a 'weak form' of Quantifier Elimination (QE):
  - The UI of a quantified formula $\exists \underline{e}\, \phi(\underline{e}, \underline{x})$ is the ***strongest*** formula **implied** by $\exists \underline{e}\, \phi(\underline{e}, \underline{x})$;
  - UIs are strictly related to proper QE in **model completions**.

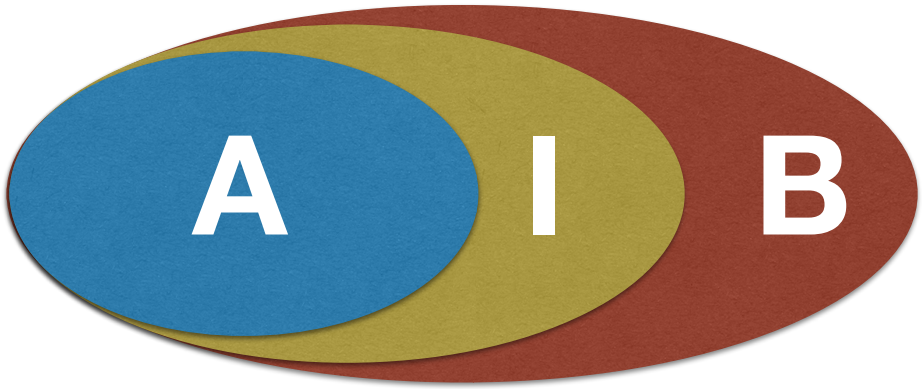
Topic of the Talk

- Stronger form of (Craig) Interpolation: **Uniform Interpolation**.
- We focus on **Quantifier-free** Uniform Interpolants (UIs).
- UIs are a 'weak form' of Quantifier Elimination (QE):
  - The UI of a quantified formula $\exists \underline{e}\, \phi(\underline{e}, \underline{x})$ is the ***strongest*** formula **implied** by $\exists \underline{e}\, \phi(\underline{e}, \underline{x})$;
  - UIs are strictly related to proper QE in **model completions**.
- Applications to **verification of infinite-state systems**, where UIs are used to compute **sets of reachable states** in a *precise* way.

Ordinary Interpolation



William Craig



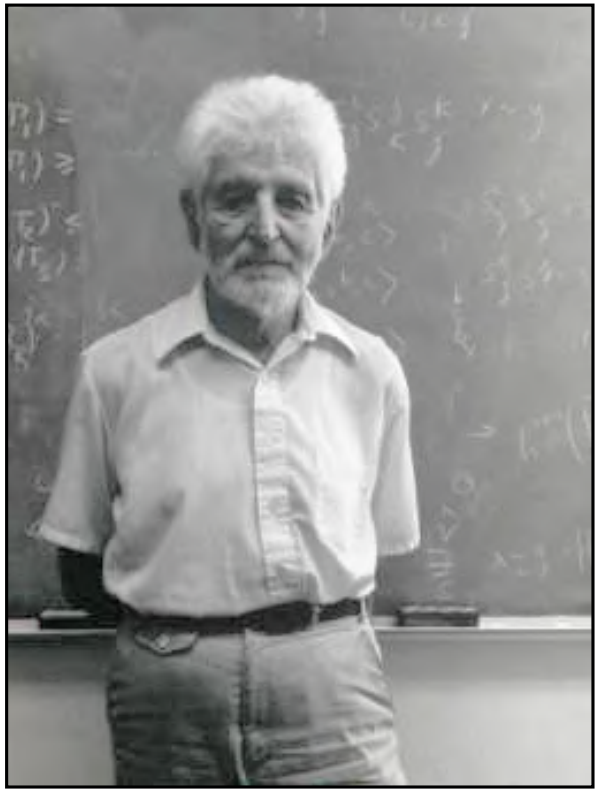
$$A \rightarrow B$$

$$\Rightarrow$$

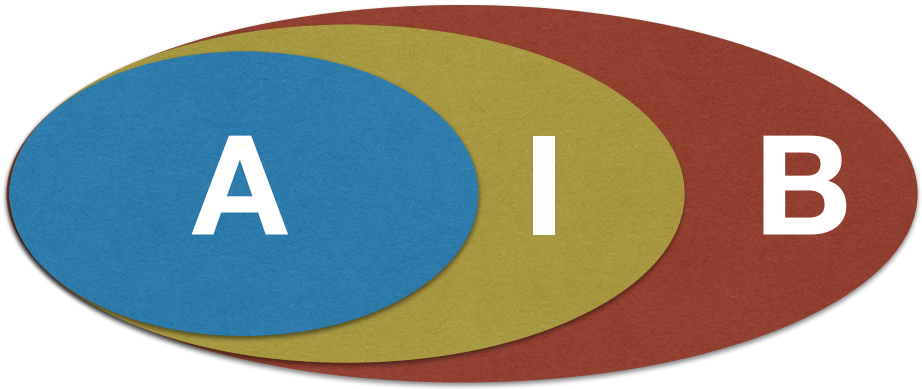
$$A \rightarrow I \qquad I \rightarrow B$$

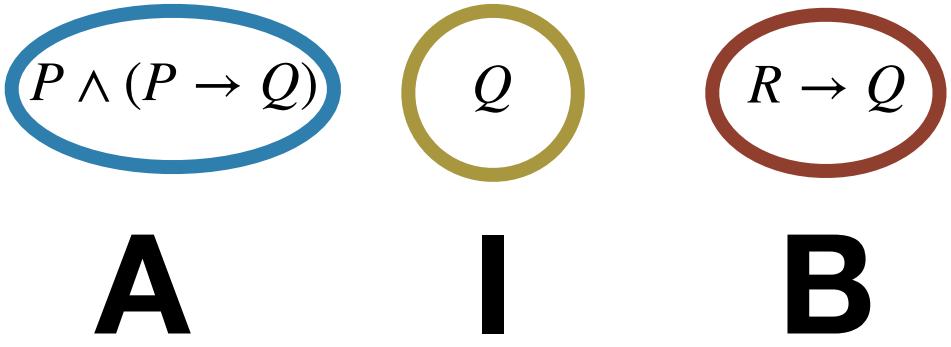
I has only symbols in common with both **A** and **B**

Ordinary Interpolation



William Craig





Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Ordinary Interpolation)

Given a pair of L-formulae $(\phi(\underline{x}, \underline{y}), \psi(\underline{x}, \underline{z}))$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, an ***ordinary interpolant*** of (ϕ, ψ) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Ordinary Interpolation)

Given a pair of L-formulae $(\phi(\underline{x}, \underline{y}), \psi(\underline{x}, \underline{z}))$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, an ***ordinary interpolant*** of (ϕ, ψ) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

- Every pair of L-formulae (ϕ, ψ) has an ordinary interpolant $\implies$ T admits ***(L)-ordinary (Craig) interpolation*** [Cra57].

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Ordinary Interpolation)

Given a pair of L-formulae $(\phi(\underline{x}, \underline{y}), \psi(\underline{x}, \underline{z}))$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, an ***ordinary interpolant*** of (ϕ, ψ) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

- Every pair of L-formulae (ϕ, ψ) has an ordinary interpolant $\implies$ T admits ***(L)-ordinary (Craig) interpolation*** [Cra57].
- ***Reverse* ordinary interpolant** for a pair (ϕ, ψ): ordinary interpolant for the pair $(\phi(\underline{x}, \underline{y}), \neg\psi(\underline{x}, \underline{z}))$ [McM03, McM06].

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Ordinary Interpolation)

Given a pair of L-formulae $(\phi(\underline{x},\underline{y}),\psi(\underline{x},\underline{z}))$ s.t. $\phi(\underline{x},\underline{y}) \vdash_T \psi(\underline{x},\underline{z})$, an ***ordinary interpolant*** of (ϕ,ψ) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, satisfying:

- $\phi(\underline{x},\underline{y}) \vdash_T \phi'(\underline{x})$;
- $\phi'(\underline{x}) \vdash_T \psi(\underline{x},\underline{z})$.

- Every pair of L-formulae (ϕ,ψ) has an ordinary interpolant $\implies$ T admits ***(L)-ordinary (Craig) interpolation*** [Cra57].
- ***Reverse* ordinary interpolant** for a pair (ϕ,ψ): ordinary interpolant for the pair $(\phi(\underline{x},\underline{y}),\neg\psi(\underline{x},\underline{z}))$ [McM03, McM06].
  - In this case, $\phi \wedge \psi$ is T-*inconsistent* and a reverse interpolant $\phi'(\underline{x})$ is s.t. $\phi \vdash_T \phi'$ and $\phi' \wedge \psi$ is T-*inconsistent*.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Uniform Interpolation)

Given an L-formula ϕ, a ***uniform interpolant (UI)*** of ϕ (w.r.t. $\underline{y}$) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, and satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- for any further L-formula $\psi(\underline{x}, \underline{z})$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, we have $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Uniform Interpolation)

Given an L-formula ϕ, a ***uniform interpolant (UI)*** of ϕ (w.r.t. $\underline{y}$) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, and satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- for any further L-formula $\psi(\underline{x}, \underline{z})$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, we have $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

- Every L-formula $\phi(\underline{x}, \underline{y})$ has UI $\implies$ T admits ***(L)-uniform interpolation***.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Uniform Interpolation)

Given an L-formula ϕ, a ***uniform interpolant (UI)*** of ϕ (w.r.t. $\underline{y}$) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, and satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- for any further L-formula $\psi(\underline{x}, \underline{z})$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, we have $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

- Every L-formula $\phi(\underline{x}, \underline{y})$ has UI $\implies$ T admits ***(L)-uniform interpolation***.
- UIs are ordinary interpolants as well, but **only** depend on ϕ and are ***independent*** of ψ.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, FO quantifier-free, etc.) of the language of a theory T.

Definition (Uniform Interpolation)

Given an L-formula ϕ, a ***uniform interpolant (UI)*** of ϕ (w.r.t. $\underline{y}$) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, and satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- for any further L-formula $\psi(\underline{x}, \underline{z})$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, we have $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

- Every L-formula $\phi(\underline{x}, \underline{y})$ has UI $\implies$ T admits ***(L)-uniform interpolation***.
- UIs are ordinary interpolants as well, but **only** depend on ϕ and are ***independent*** of ψ.
- If they exist, UIs are **unique**, up to T-equivalence.

Ordinary vs Uniform Interpolation

Let L be a fragment (propositional, **FO quantifier-free**, etc.) of the language of a theory T.

Definition (Uniform Interpolation)

Given an L-formula ϕ, a ***uniform interpolant (UI)*** of ϕ (w.r.t. $\underline{y}$) is an L-formula $\phi'(\underline{x})$ where **only** the $\underline{x}$ occur, and satisfying:

- $\phi(\underline{x}, \underline{y}) \vdash_T \phi'(\underline{x})$;
- for any further L-formula $\psi(\underline{x}, \underline{z})$ s.t. $\phi(\underline{x}, \underline{y}) \vdash_T \psi(\underline{x}, \underline{z})$, we have $\phi'(\underline{x}) \vdash_T \psi(\underline{x}, \underline{z})$.

Quantifier-free (ordinary) interpolants – quantifier-free uniform interpolants

Outline

Symbolic Transition System

- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).

Symbolic Transition System

- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).
- The content of $\underline{x}$ determines the **global state** of the system.

Symbolic Transition System

- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).
- The content of $\underline{x}$ determines the **global state** of the system.
- Usually, state formulae are **quantifier-free**, where the $\underline{x}$ occur free.

Symbolic Transition System

- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).
- The content of $\underline{x}$ determines the **global state** of the system.
- Usually, state formulae are **quantifier-free**, where the $\underline{x}$ occur free.
- **Initial states** $Init$: state formula $I(\underline{x})$.

Symbolic Transition System

- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).
- The content of $\underline{x}$ determines the **global state** of the system.
- Usually, state formulae are **quantifier-free**, where the $\underline{x}$ occur free.
- **Initial states** $Init$: state formula $I(\underline{x})$.
- **Transitions** T: formula $\tau(\underline{x}, \underline{x}')$, connecting the **current state** $\underline{x}$ to the **next state** $\underline{x}'$.

Symbolic Transition System

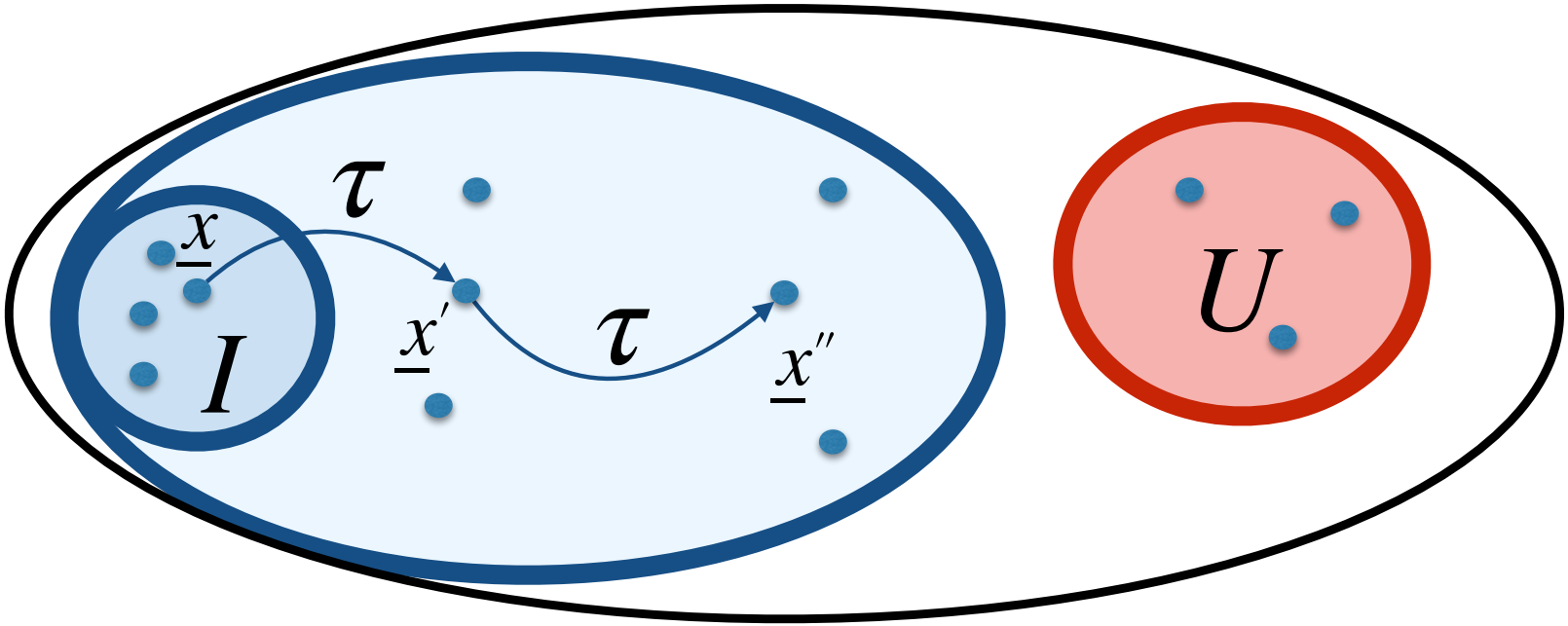
- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).
- The content of $\underline{x}$ determines the **global state** of the system.
- Usually, state formulae are **quantifier-free**, where the $\underline{x}$ occur free.
- **Initial states** $Init$: state formula $I(\underline{x})$.
- **Transitions** T: formula $\tau(\underline{x}, \underline{x}')$, connecting the **current state** $\underline{x}$ to the **next state** $\underline{x}'$.
- **Symbolic Transition System** TS: $\langle(\Sigma, T), I(\underline{x}), \tau(\underline{x}, \underline{x}')\rangle$, with (Σ, T) a FO theory (runs of the system live in their models)

Symbolic Transition System

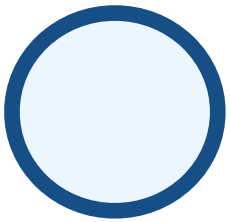
- **Set of states** S: formula $\phi(\underline{x})$, called **state formula**, where $\underline{x}$ is a tuple of variables, called **state variables** (e.g., registers of a RAM machine).
- The content of $\underline{x}$ determines the **global state** of the system.
- Usually, state formulae are **quantifier-free**, where the $\underline{x}$ occur free.
- **Initial states** $Init$: state formula $I(\underline{x})$.
- **Transitions** T: formula $\tau(\underline{x}, \underline{x}')$, connecting the **current state** $\underline{x}$ to the **next state** $\underline{x}'$.
- **Symbolic Transition System** TS: $\langle(\Sigma, T), I(\underline{x}), \tau(\underline{x}, \underline{x}')\rangle$, with (Σ, T) a FO theory (runs of the system live in their models)
- If the $\underline{x}$ range over an infinite domain, TS formalizes an **infinite-state system**.

Symbolic Model Checking

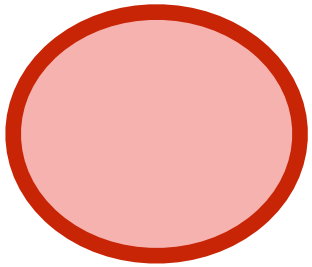
Safety Problem: is a set of **undesired states** $U(\underline{x})$ (state formula) *reachable* through the transitions $\tau(\underline{x}, \underline{x}')$ from a set of initial states $I(\underline{x})$?



Forward Reachability

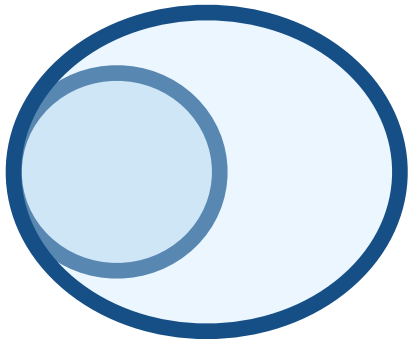


$I(\underline{x})$
initial
configurations



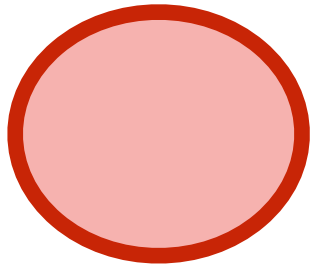
$U(\underline{x})$
unsafe
configurations

Forward Reachability



$I(\underline{x})$

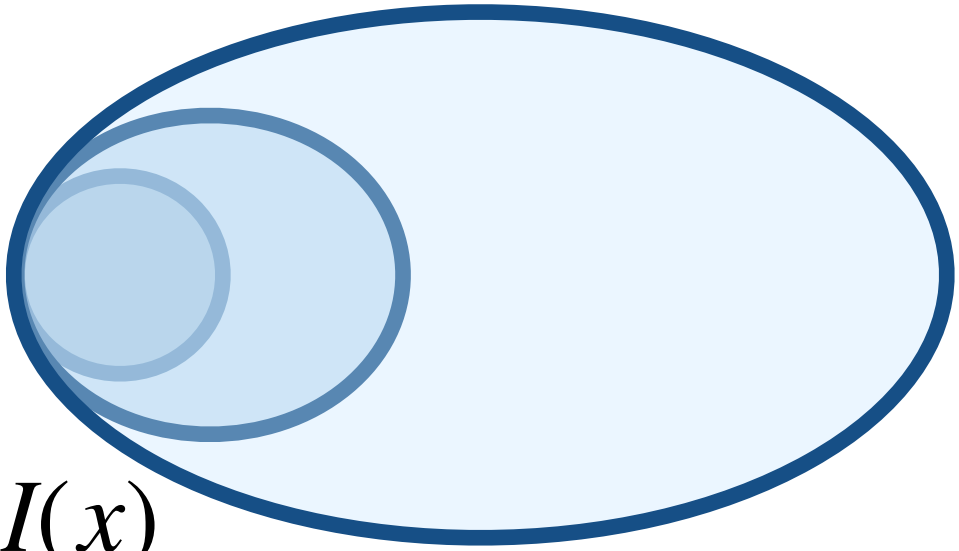
initial configurations



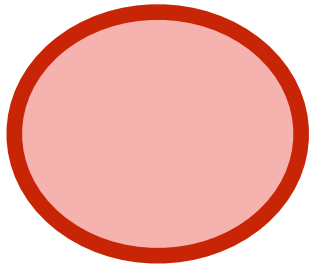
$U(\underline{x})$

unsafe configurations

Forward Reachability

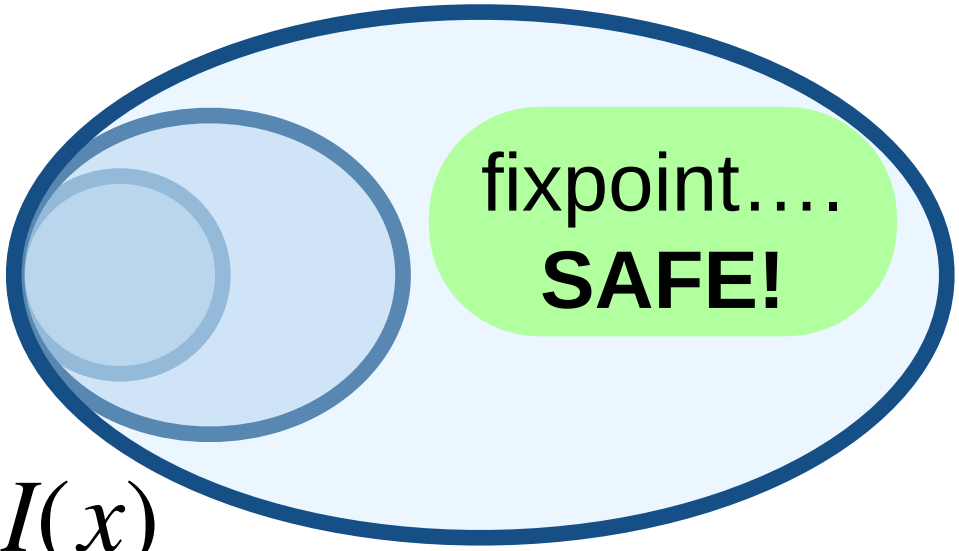


$I(\underline{x})$
initial
configurations

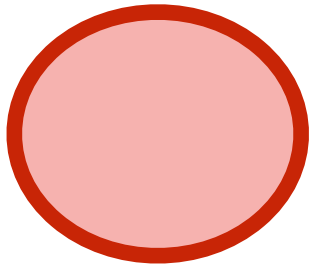


$U(\underline{x})$
unsafe
configurations

Forward Reachability

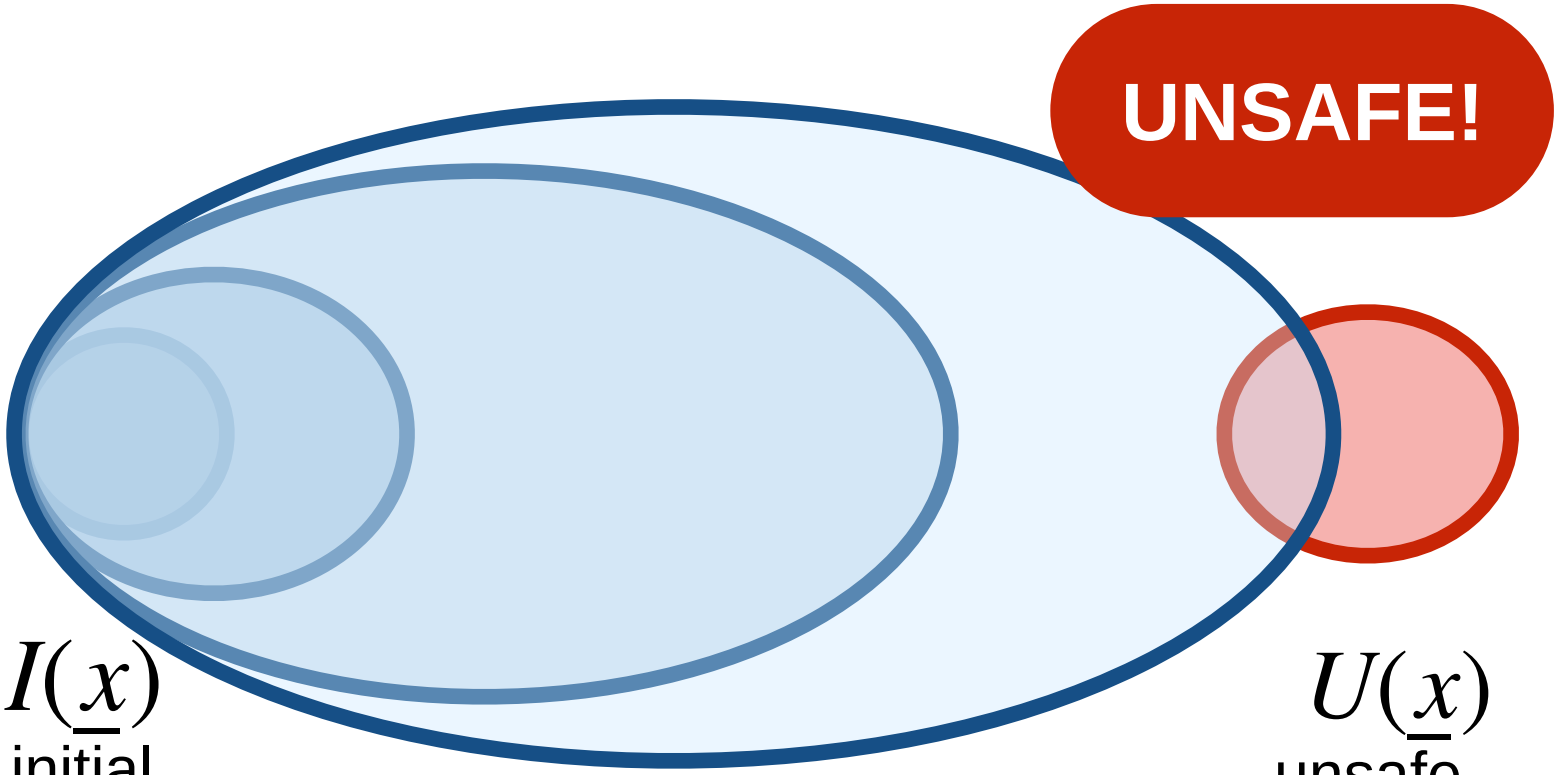


$I(\underline{x})$
initial
configurations



$U(\underline{x})$
unsafe
configurations

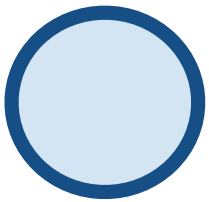
Forward Reachability



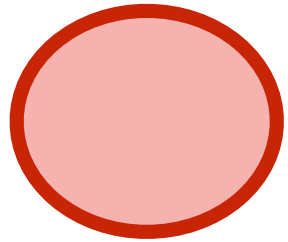
Direct Image Computation

The key ingredient is to compute *iteratively* **Direct Images** of a state formula ϕ through transitions τ

$$Dir(\tau, \phi) :\equiv \exists \underline{x}(\phi(\underline{x}) \wedge \tau(\underline{x}, \underline{x}'))$$



$I(\underline{x})$



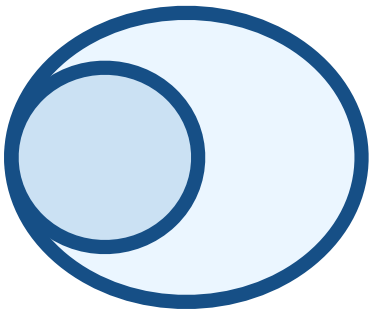
$U(\underline{x})$

Direct Image Computation

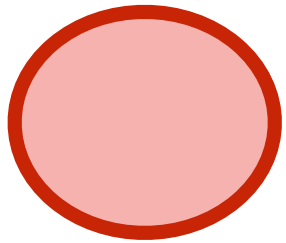
The key ingredient is to*iteratively* compute **Direct Images** of a state formula ϕ through transitions τ

$$Dir(\tau, \phi) :\equiv \exists \underline{x}(\phi(\underline{x}) \wedge \tau(\underline{x}, \underline{x}'))$$

$$\exists \underline{x}(I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}'))$$



$I(\underline{x})$



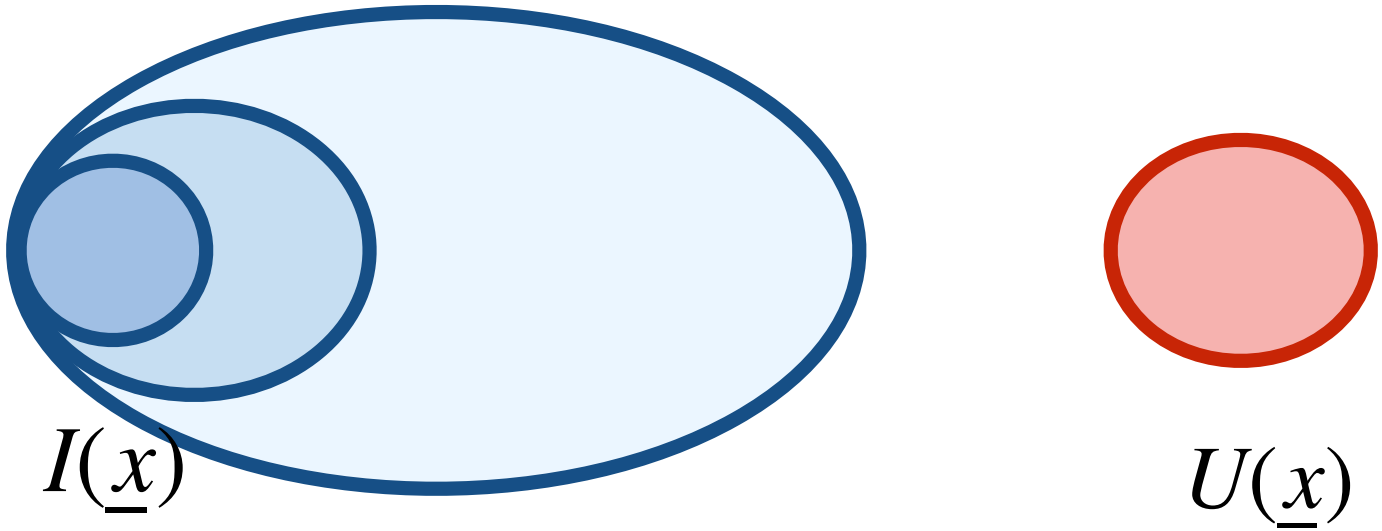
$U(\underline{x})$

Direct Image Computation

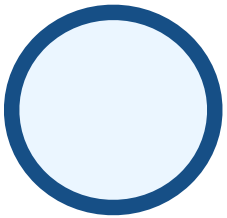
The key ingredient is to *iteratively* compute **Direct Images** of a state formula ϕ through transitions τ

$$Dir(\tau, \phi) :\equiv \exists \underline{x}(\phi(\underline{x}) \wedge \tau(\underline{x}, \underline{x}'))$$

$$\exists \underline{x}, \underline{x}'(I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}') \wedge \tau(\underline{x}', \underline{x}''))$$

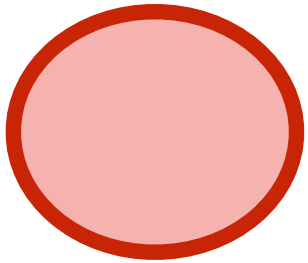


Backward Reachability



$I(\underline{x})$

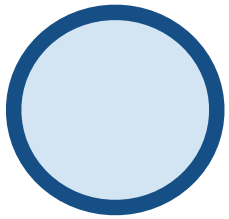
initial configurations



$U(\underline{x})$

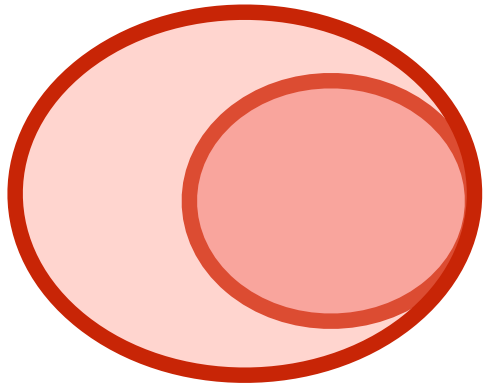
unsafe configurations

Backward Reachability



$I(\underline{x})$

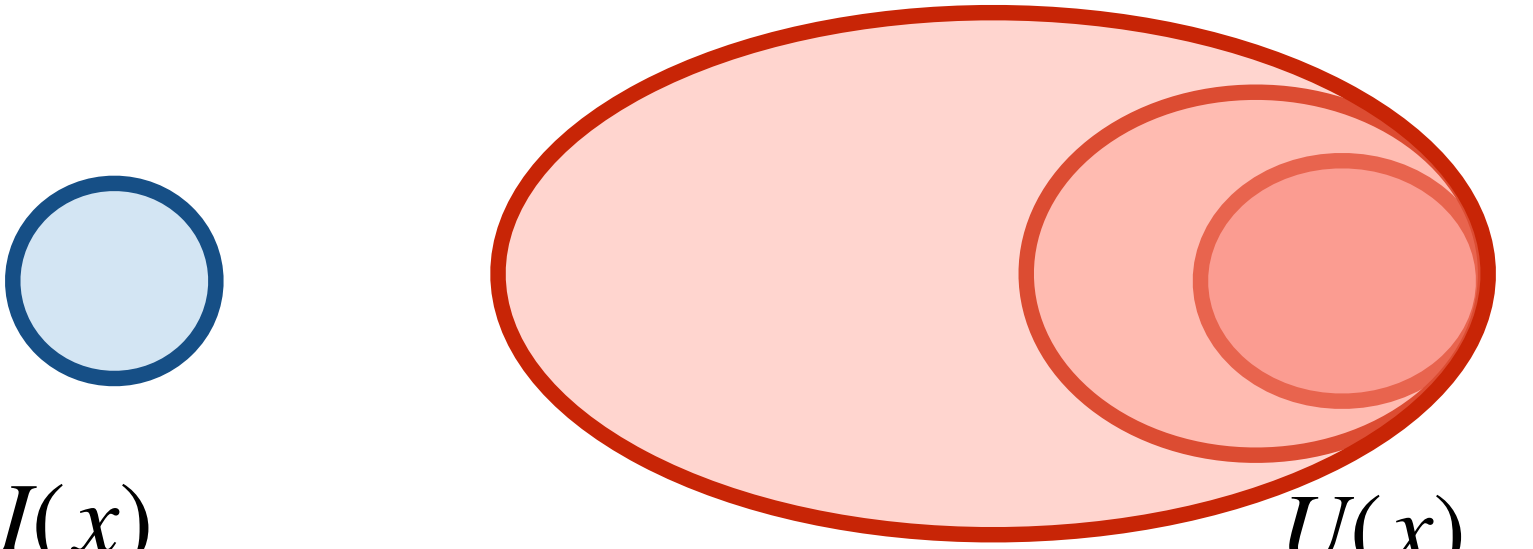
initial configurations



$U(\underline{x})$

unsafe configurations

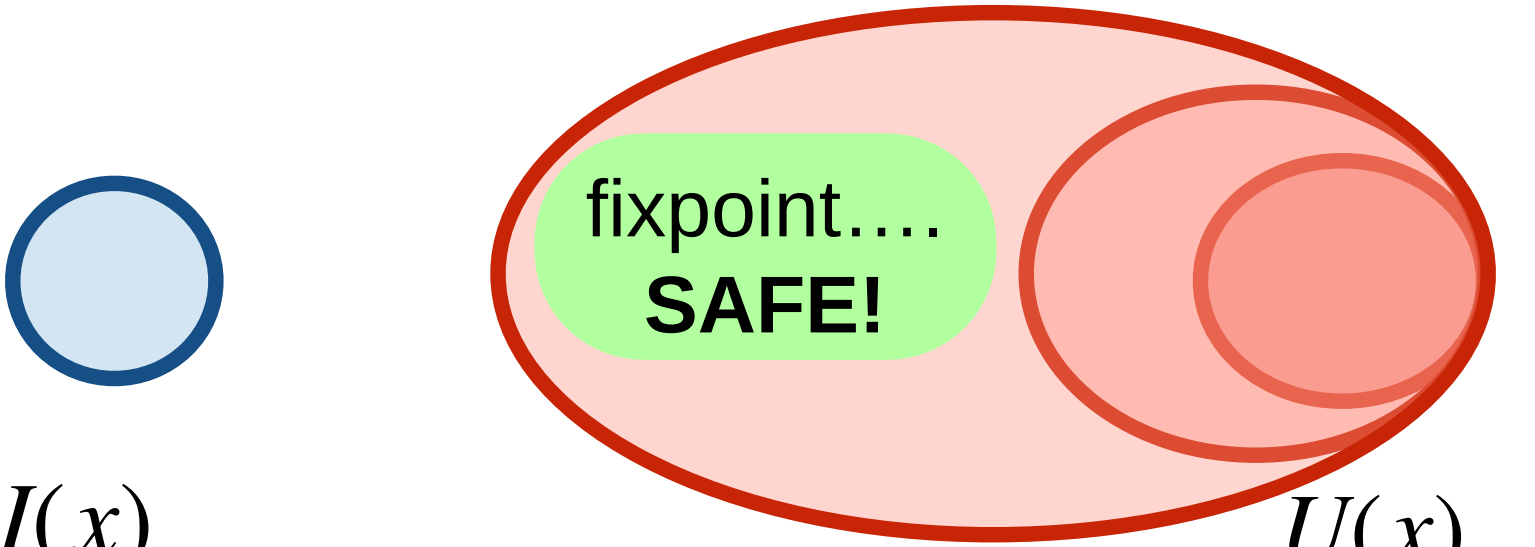
Backward Reachability



$I(\underline{x})$
initial
configurations

$U(\underline{x})$
unsafe
configurations

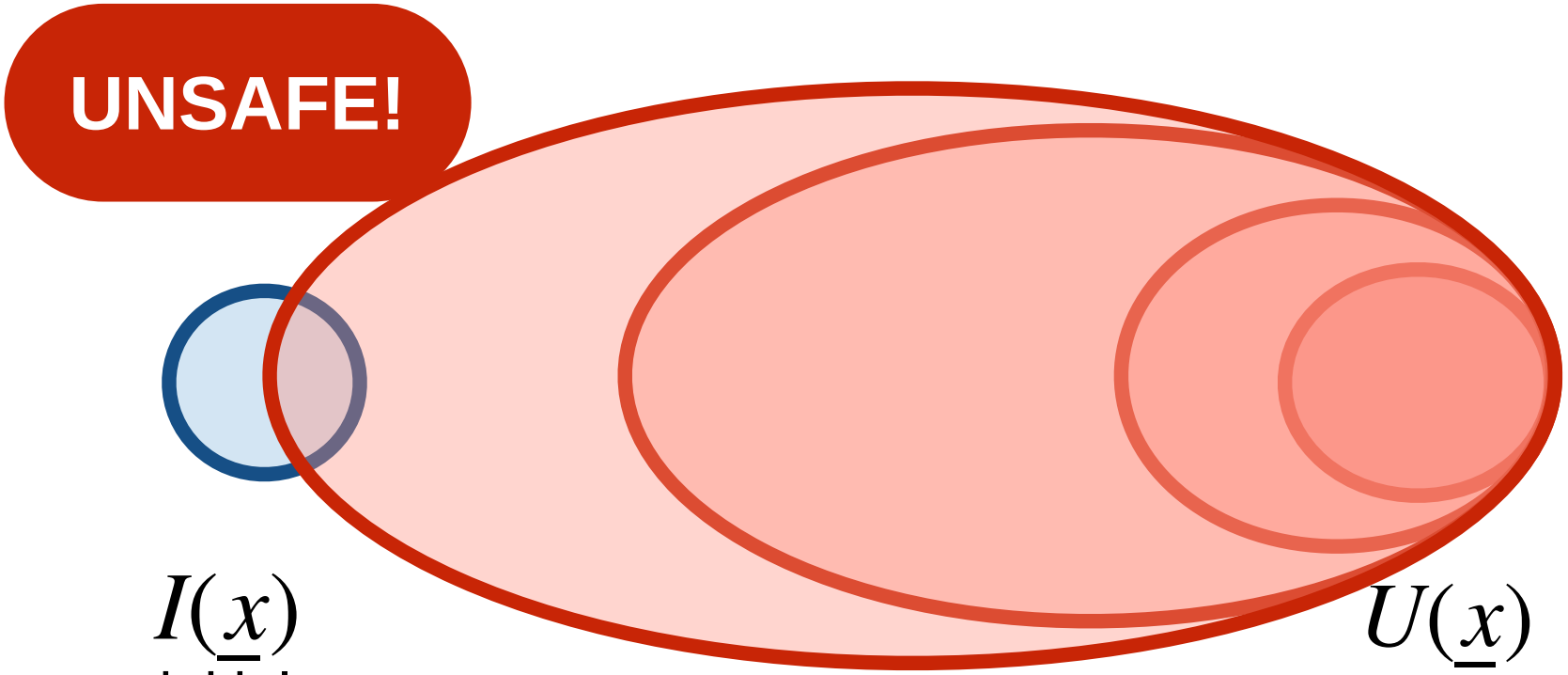
Backward Reachability



$I(\underline{x})$
initial
configurations

$U(\underline{x})$
unsafe
configurations

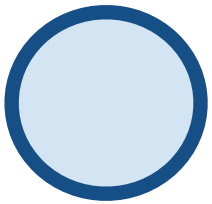
Backward Reachability



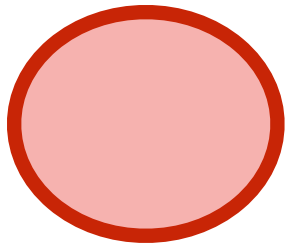
Preimage Computation

The key ingredient is to *iteratively* compute **Direct Images** of a state formula ϕ through transitions τ

$$Pre(\tau, \phi) :\equiv \exists \underline{x}'(\tau(\underline{x}, \underline{x}') \wedge \phi(\underline{x}'))$$



$I(\underline{x})$



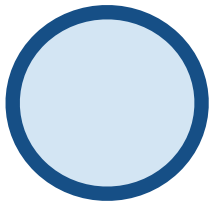
$U(\underline{x})$

Preimage Computation

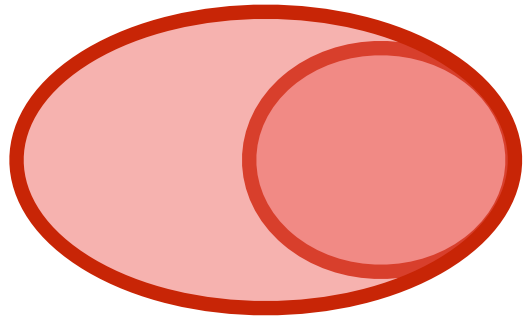
The key ingredient is to *iteratively* compute **Direct Images** of a state formula ϕ through transitions τ

$$Pre(\tau, \phi) :\equiv \exists \underline{x}'(\tau(\underline{x}, \underline{x}') \wedge \phi(\underline{x}'))$$

$$\exists \underline{x}'(\tau(\underline{x}, \underline{x}') \wedge U(\underline{x}'))$$



$I(\underline{x})$



$U(\underline{x})$

The Problem of Quantifiers (I)

- State formulae: quantifier-free

The Problem of Quantifiers (I)

- State formulae: quantifier-free
  - **Example**:

$$\phi(x_{\texttt{applicant}}) :\equiv User(x_{\texttt{applicant}}, John) \wedge x_{\texttt{applicant}} \neq \texttt{undef}$$

The Problem of Quantifiers (I)

- State formulae: quantifier-free
  - **Example**:

$$\phi(x_{\texttt{applicant}}) :\equiv User(x_{\texttt{applicant}}, John) \wedge x_{\texttt{applicant}} \neq \texttt{undef}$$

- Transition formulae may be **(existentially) quantified** formulae: e.g., a transition whose guard *queries* the content of a relational database.

The Problem of Quantifiers (I)

- State formulae: quantifier-free
  - **Example**:

$$\phi(x_{\texttt{applicant}}) :\equiv User(x_{\texttt{applicant}}, John) \wedge x_{\texttt{applicant}} \neq \texttt{undef}$$

- Transition formulae may be **(existentially) quantified** formulae: e.g., a transition whose guard *queries* the content of a relational database.
  - **Example**:

$$\tau(x_{\texttt{applicant}}, x'_{\texttt{applicant}}) :\equiv \exists uid, name\, (x_{\texttt{applicant}} = \texttt{undef} \wedge User(uid, name) \wedge x'_{\texttt{applicant}} = uid)$$

The Problem of Quantifiers (I)

- State formulae: quantifier-free
  - **Example**:

$$\phi(x_{\texttt{applicant}}) :\equiv User(x_{\texttt{applicant}}, John) \wedge x_{\texttt{applicant}} \neq \texttt{undef}$$

- Transition formulae may be **(existentially) quantified** formulae: e.g., a transition whose guard *queries* the content of a relational database.
  - **Example**:

$$\tau(x_{\texttt{applicant}}, x'_{\texttt{applicant}}) :\equiv \exists uid, name\, (x_{\texttt{applicant}} = \texttt{undef} \wedge User(uid, name) \wedge x'_{\texttt{applicant}} = uid)$$

- Transitions may introduce **new variables** wrt state variables $\underline{x}$

The Problem of Quantifiers (I)

- State formulae: quantifier-free
  - **Example**:

$$\phi(x_{\texttt{applicant}}) :\equiv User(x_{\texttt{applicant}}, John) \wedge x_{\texttt{applicant}} \neq \texttt{undef}$$

- Transition formulae may be **(existentially) quantified** formulae: e.g., a transition whose guard *queries* the content of a relational database.
  - **Example**:

$$\tau(x_{\texttt{applicant}}, x'_{\texttt{applicant}}) :\equiv \exists uid, name\, (x_{\texttt{applicant}} = \texttt{undef} \wedge User(uid, name) \wedge x'_{\texttt{applicant}} = uid)$$

- Transitions may introduce **new variables** wrt state variables $\underline{x}$
  - **Example**: uid and $name$ new w.r.t. the state variable $x_{\texttt{applicant}} \implies$ data values non-deterministacally taken from a DB.

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!
- The problem of quantifiers affects the *effectiveness* of the search:

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!
- The problem of quantifiers affects the *effectiveness* of the search:
  - **Loop invariant** of the procedure: 'a set of states is described by a state formula'

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!
- The problem of quantifiers affects the *effectiveness* of the search:
  - **Loop invariant** of the procedure: 'a set of states is described by a state formula'
  - During the search, the tail of quantifiers can **grow dramatically**, affecting also termination and the performance!

The Problem of Quantifiers (II)

- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!
- The problem of quantifiers affects the *effectiveness* of the search:
  - **Loop invariant** of the procedure: 'a set of states is described by a state formula'
  - During the search, the tail of quantifiers can **grow dramatically**, affecting also termination and the performance!
- How to solve the **problem of quantifiers**?

The Problem of Quantifiers (II)

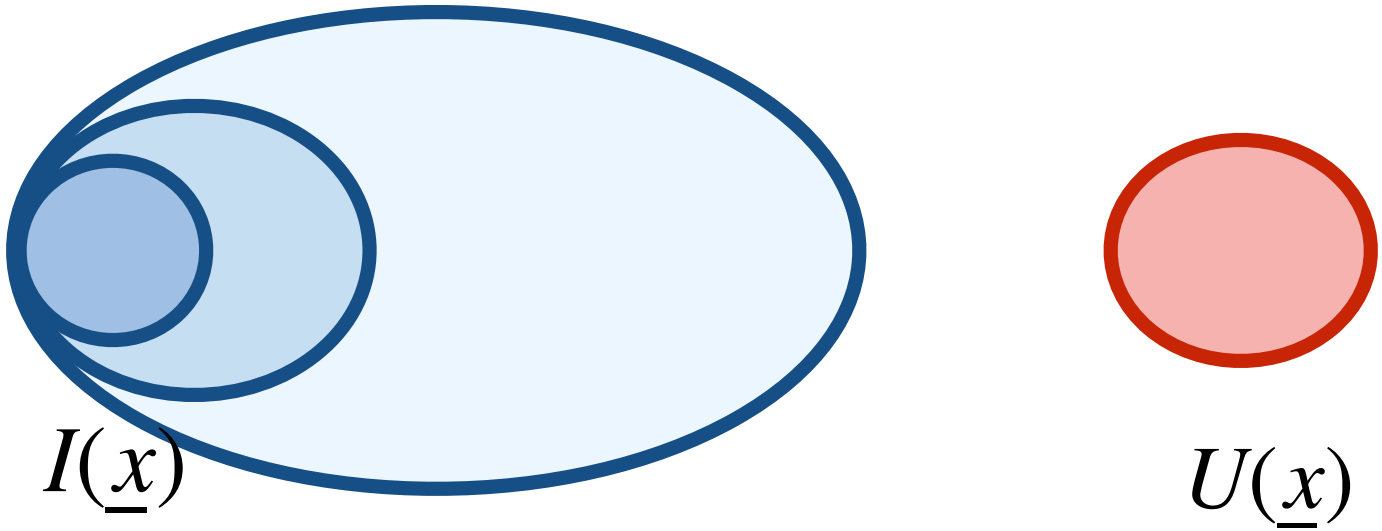
- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!
- The problem of quantifiers affects the *effectiveness* of the search:
  - **Loop invariant** of the procedure: 'a set of states is described by a state formula'
  - During the search, the tail of quantifiers can **grow dramatically**, affecting also termination and the performance!
- How to solve the **problem of quantifiers**?
  - **Ordinary Interpolation** [McM06]: *over-approximation* of reachable states!

The Problem of Quantifiers (II)

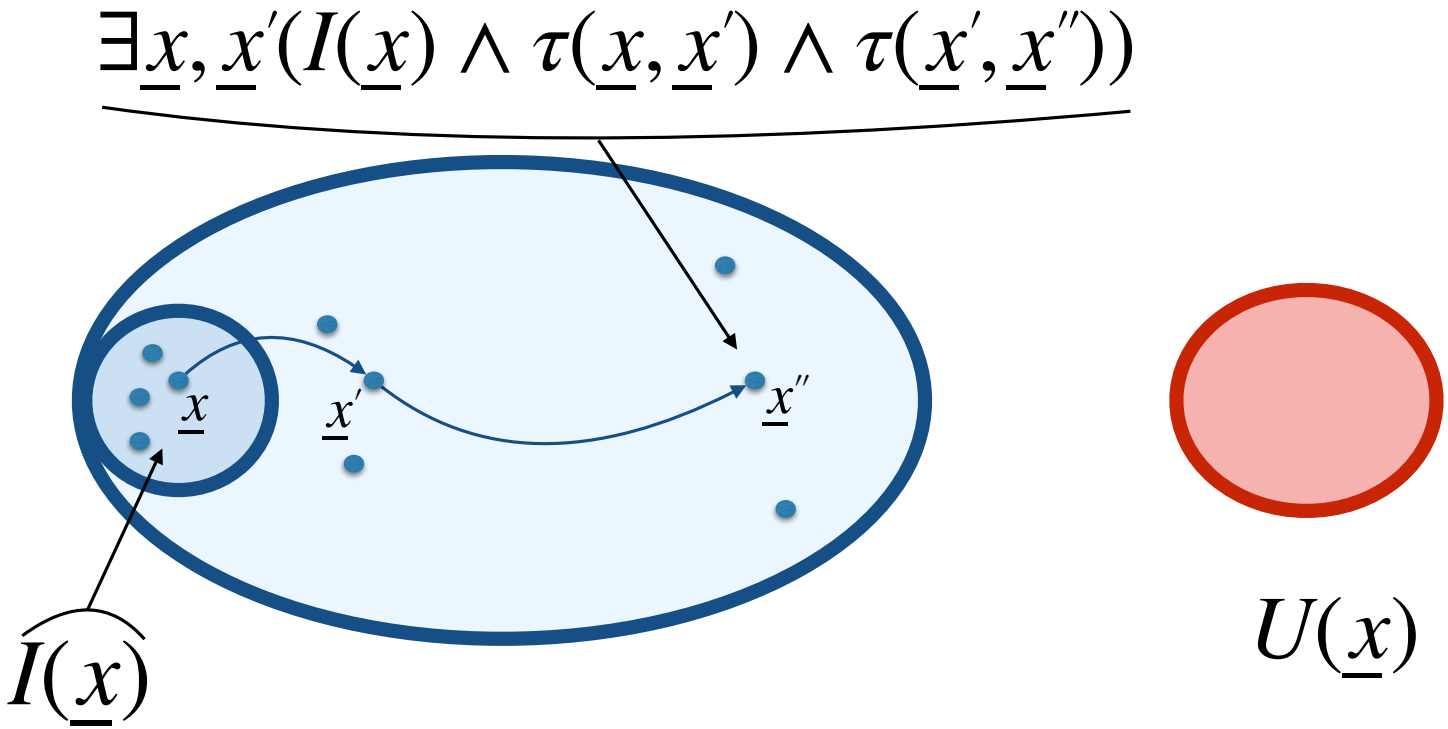
- Forward and Backward Reachability compute sets of (reachable) states by computing **(direct or pre-)images** through transitions:
  - **Images** introduce **quantifiers**, because of quantifiers in τ;
  - Images are **sets of states**, so they should be state formula $\implies$ Images should be **quantifier-free**!
- The problem of quantifiers affects the *effectiveness* of the search:
  - **Loop invariant** of the procedure: 'a set of states is described by a state formula'
  - During the search, the tail of quantifiers can **grow dramatically**, affecting also termination and the performance!
- How to solve the **problem of quantifiers**?
  - **Ordinary Interpolation** [McM06]: *over-approximation* of reachable states!
  - **Quantifier Elimination**: computationally expensive, not available for generic first-order theories

(Reverse) Ordinary Interpolation

$$\exists \underline{x}, \underline{x}'(I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}') \wedge \tau(\underline{x}', \underline{x}''))$$

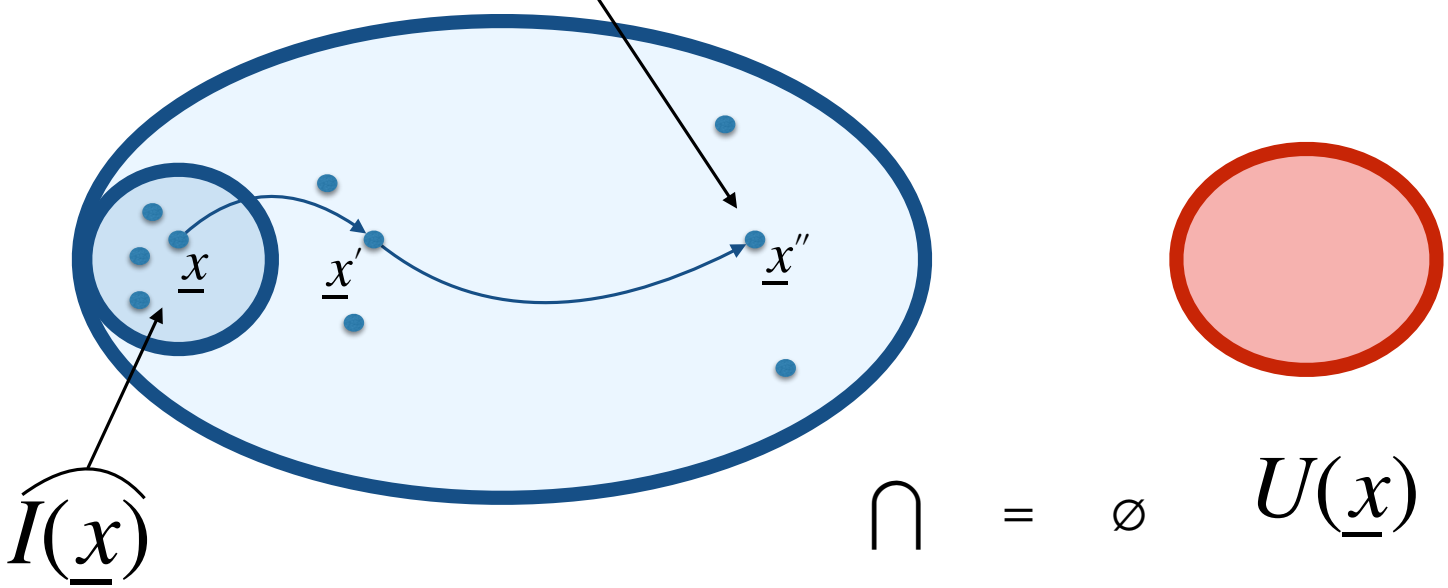


(Reverse) Ordinary Interpolation



(Reverse) Ordinary Interpolation

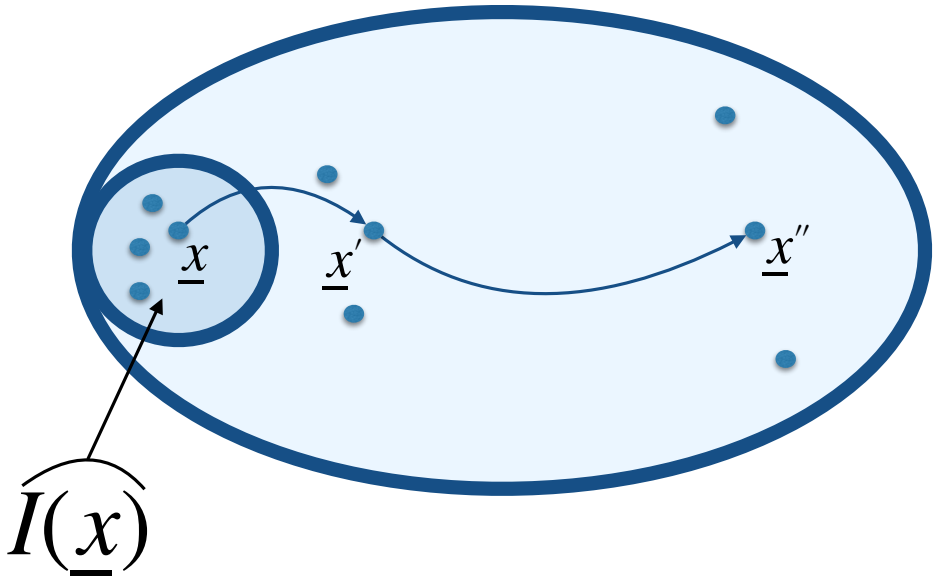
$$\exists \underline{x}, \underline{x}'(I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}') \wedge \tau(\underline{x}', \underline{x}'') \wedge U(\underline{x}'')) \models \bot$$

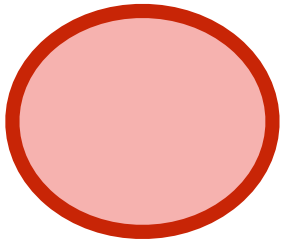


(Reverse) Ordinary Interpolation

$$A := I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}') \qquad B := \tau(\underline{x}', \underline{x}'') \wedge U(\underline{x}'')$$

$$A \wedge B := I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}') \wedge \tau(\underline{x}', \underline{x}'') \wedge U(\underline{x}'') \models \bot$$

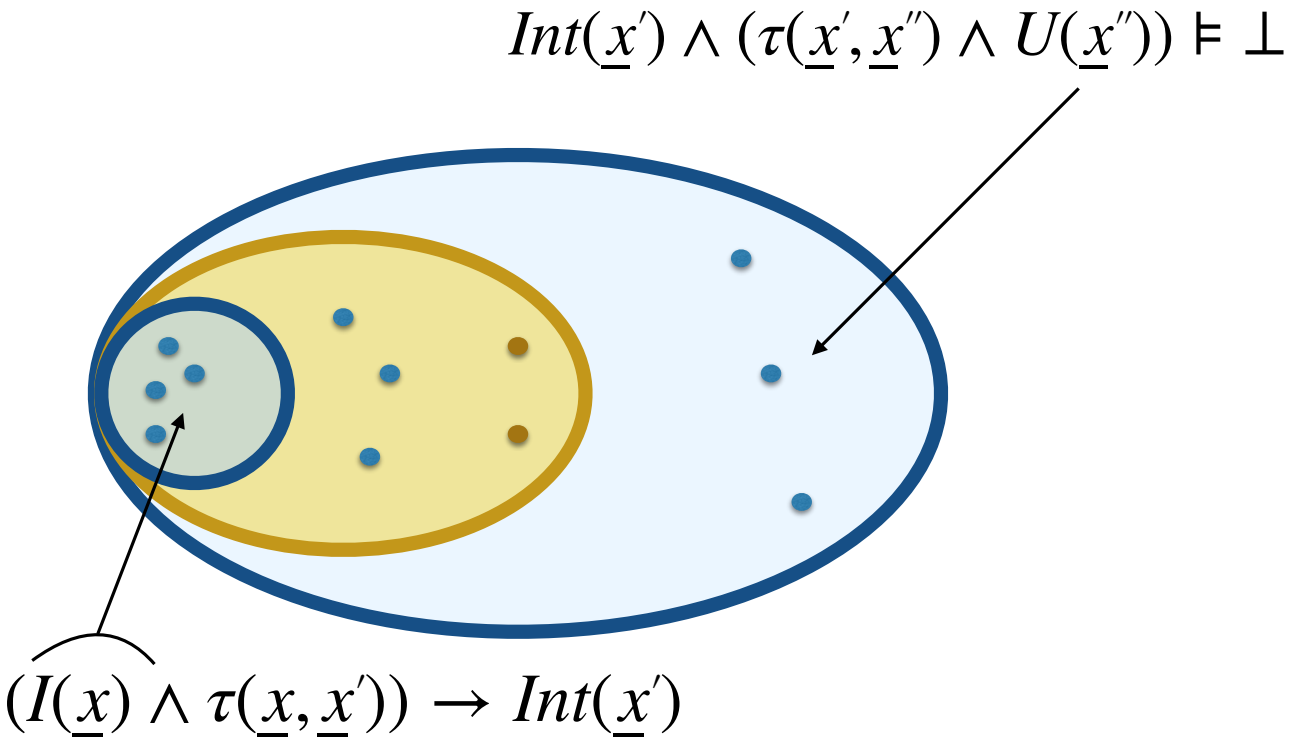




(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

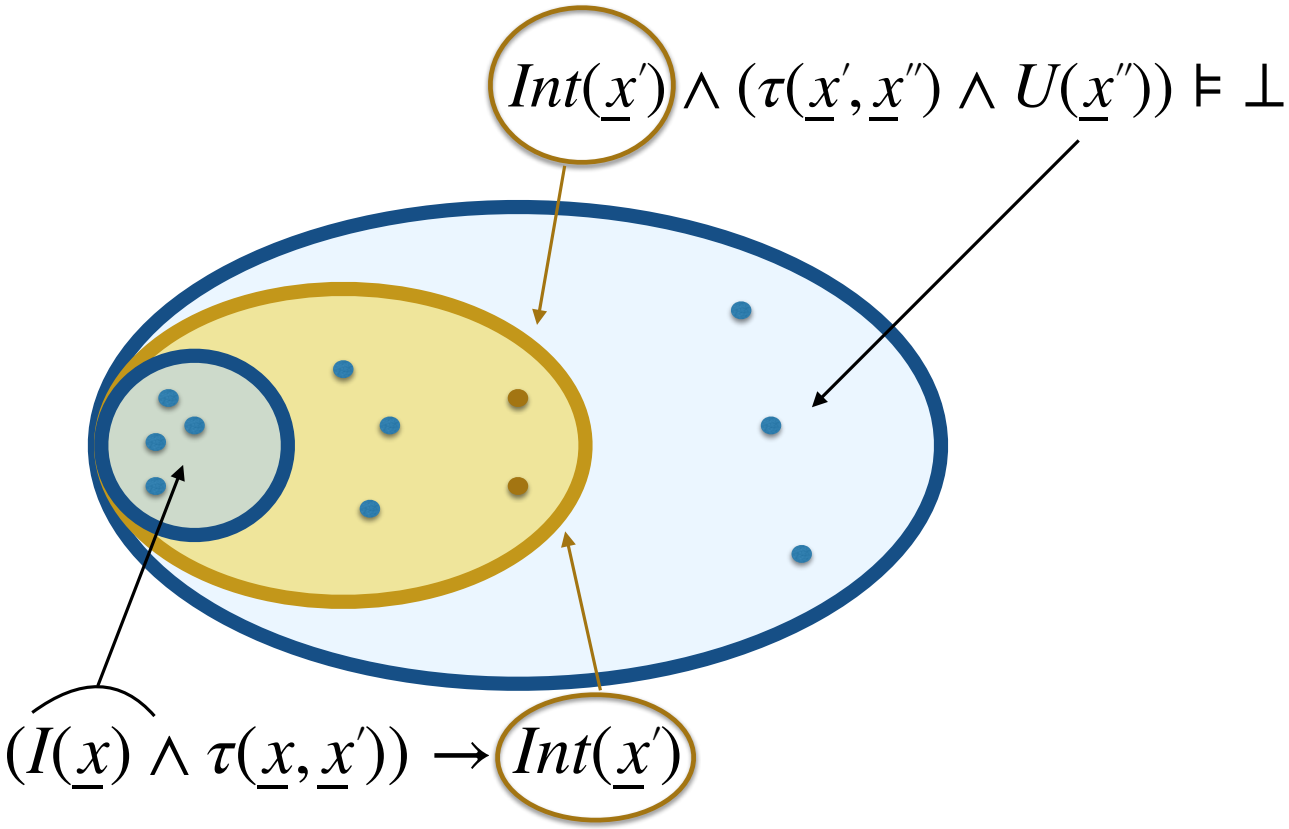
$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$



(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

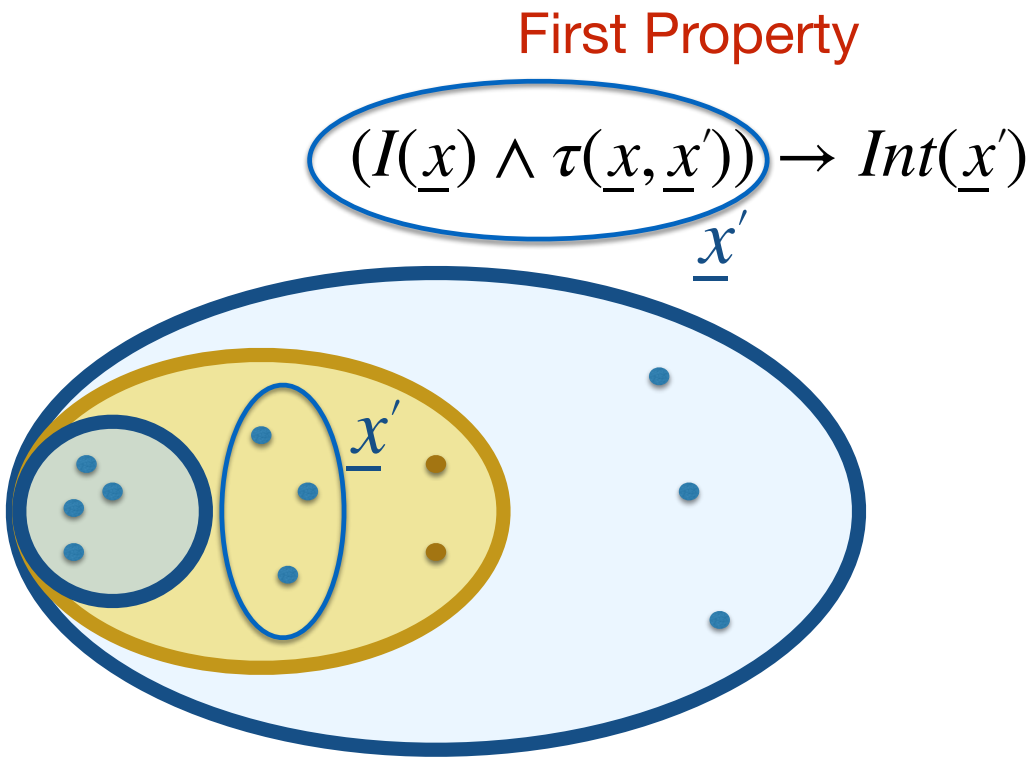
$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$



(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$



(Reverse) Ordinary Interpolation

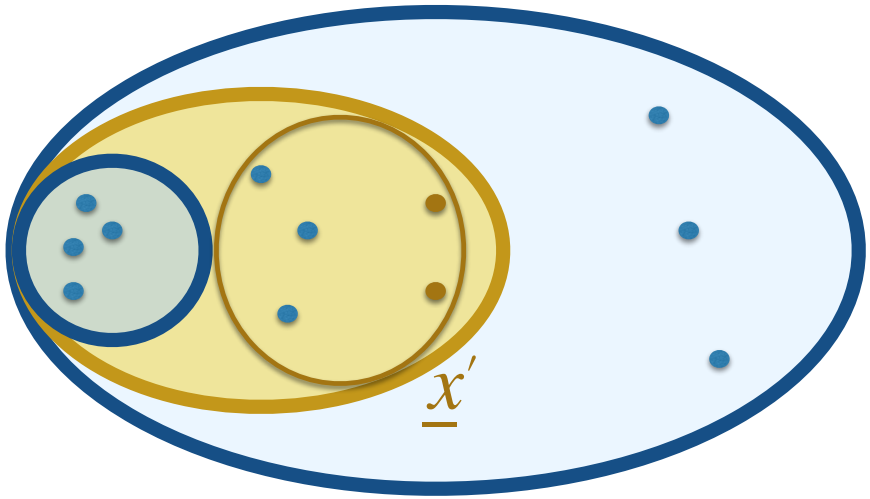
By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$

First Property

$$(I(\underline{x}) \wedge \tau(\underline{x}, \underline{x}')) \rightarrow Int(\underline{x}')$$

$\underline{x}'$



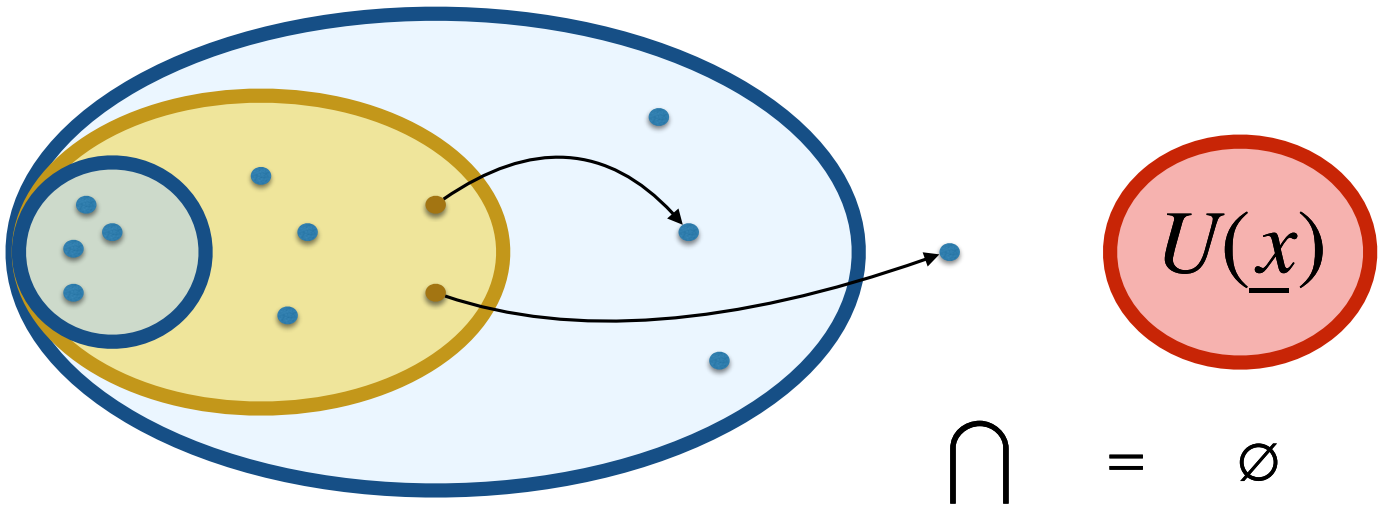
(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$

Second Property

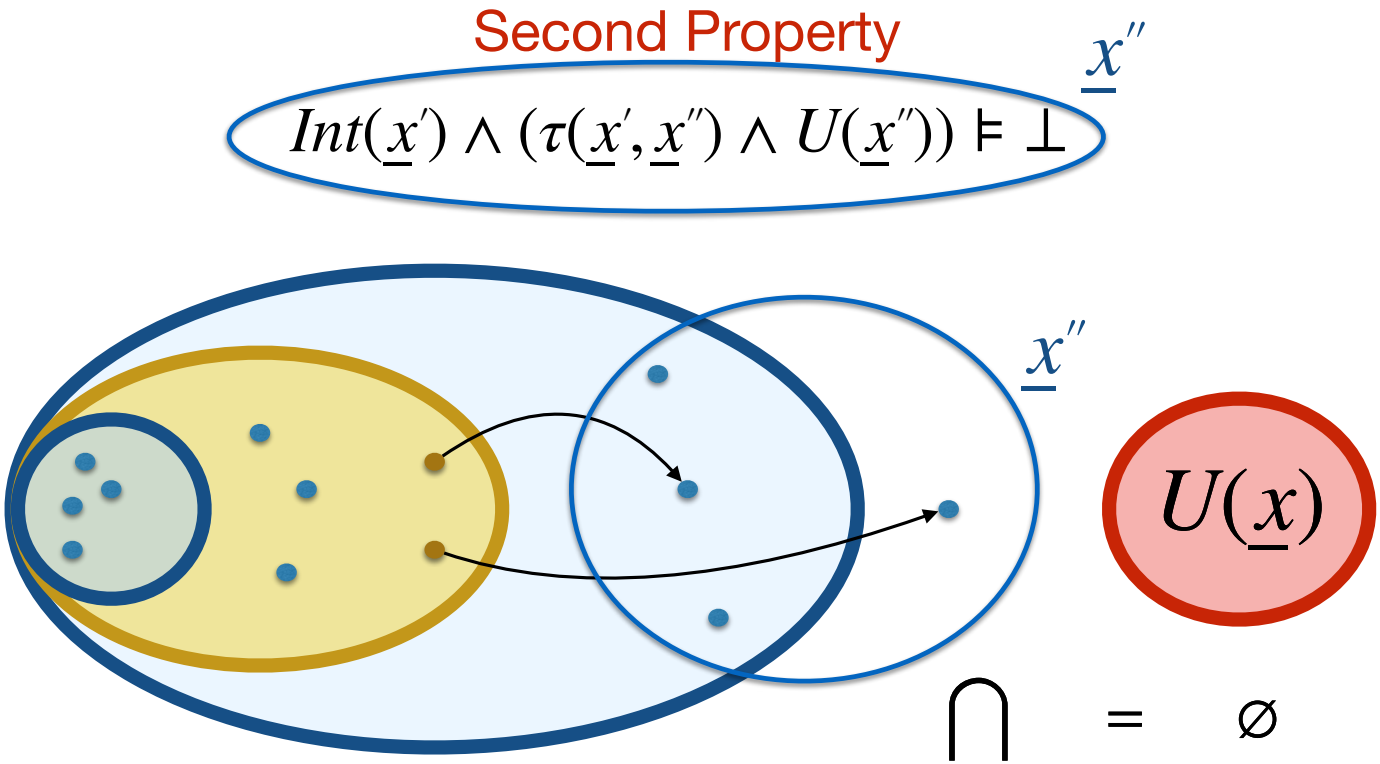
$$Int(\underline{x}') \wedge (\tau(\underline{x}', \underline{x}'') \wedge U(\underline{x}'')) \vDash \bot$$



(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

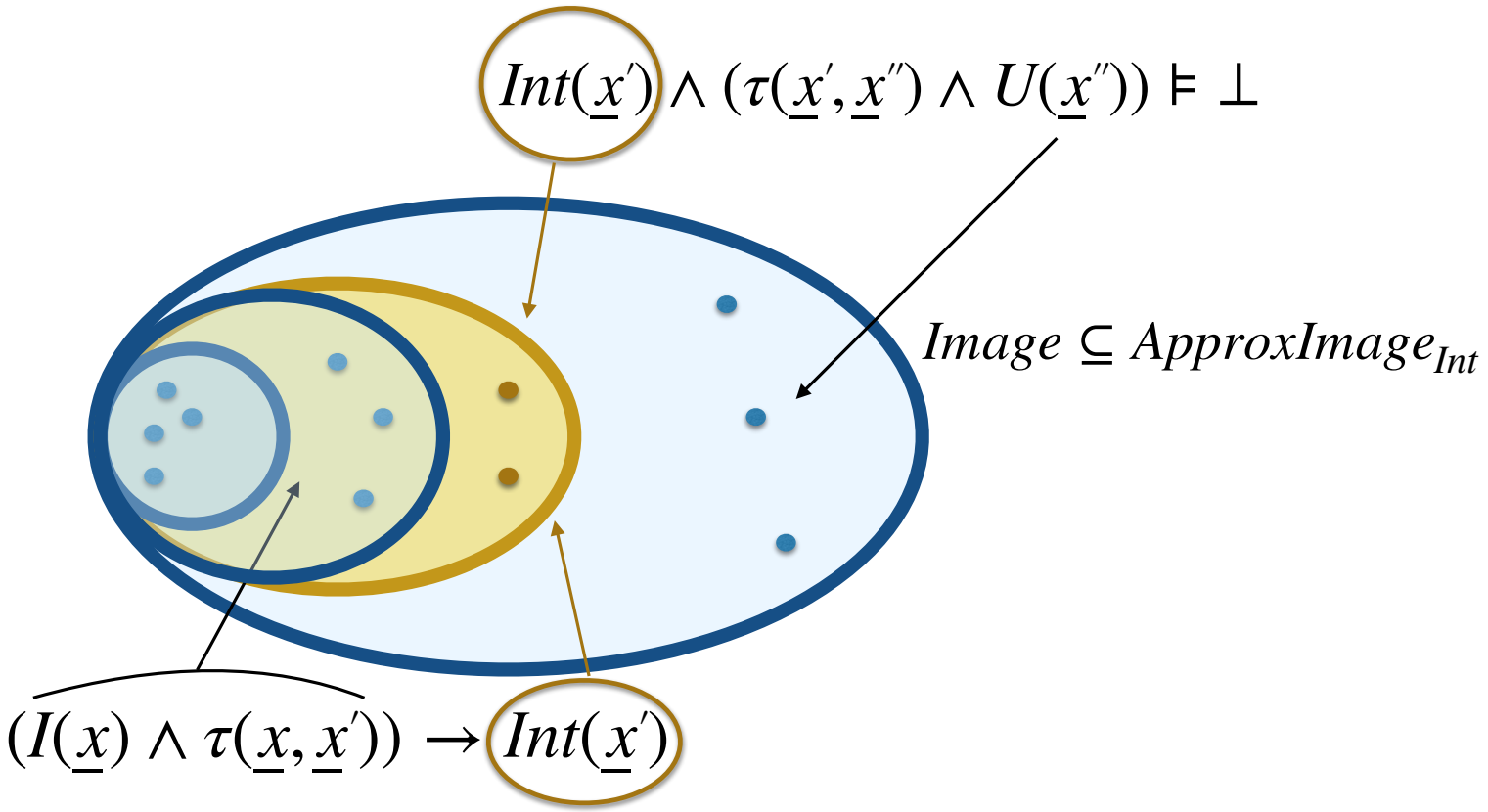
$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$



(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$



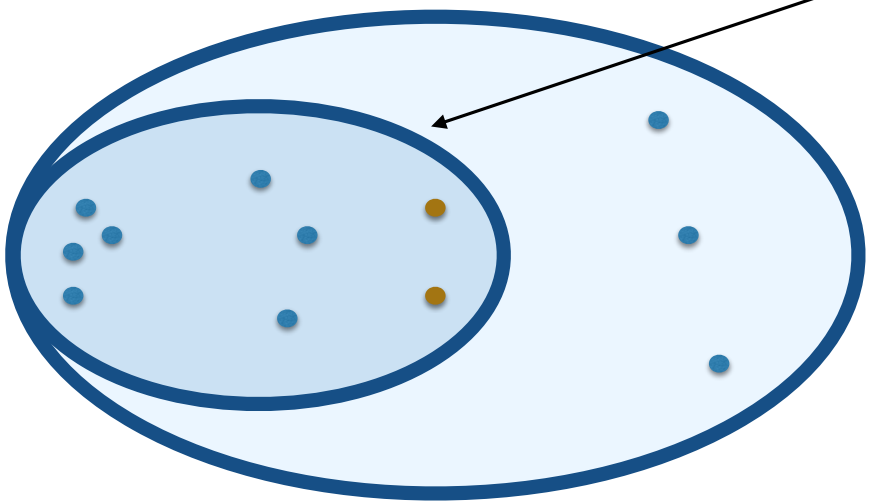
(Reverse) Ordinary Interpolation

By definition of **(reverse) ordinary interpolant**, there is a formula $Int(\underline{x}')$ (where only the common variables $\underline{x}'$ occur), such that in T:

$$A \rightarrow Int \text{ and } Int \wedge B \models \bot$$

First Iteration: $R(\underline{x}) \leftarrow I(\underline{x})$

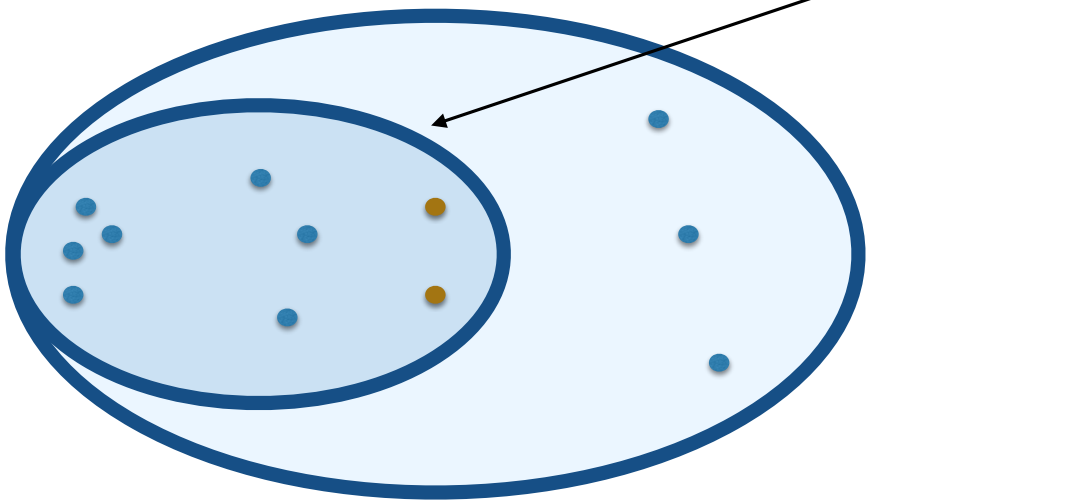
Update $R(\underline{x})$: $R(\underline{x}) \leftarrow Int(\underline{x}) \vee R(\underline{x})$



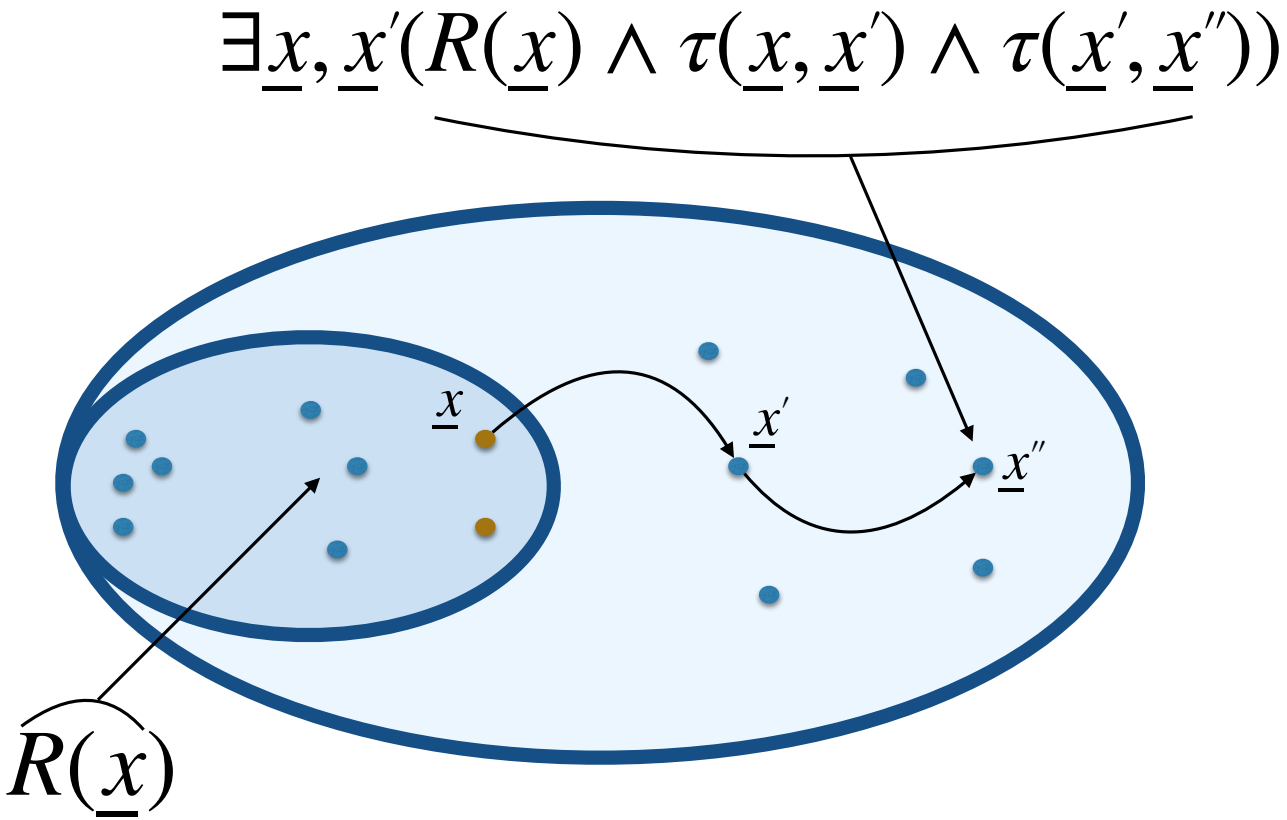
(Reverse) Ordinary Interpolation

First Iteration: $R(\underline{x}) \leftarrow I(\underline{x})$

Update $R(\underline{x})$: $R(\underline{x}) \leftarrow Int(\underline{x}) \vee R(\underline{x})$

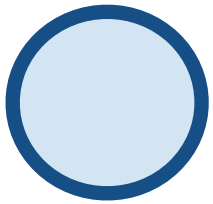


(Reverse) Ordinary Interpolation

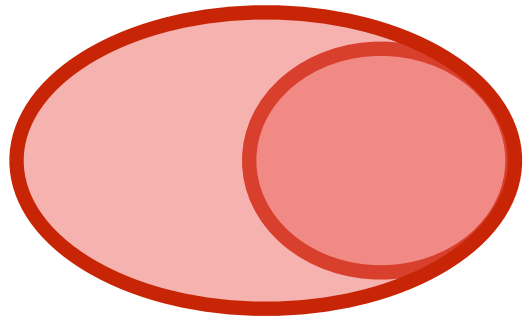


(Reverse) Ordinary Interpolation: Backward

$\exists \underline{x}'(\tau(\underline{x}, \underline{x}') \wedge U(\underline{x}'))$



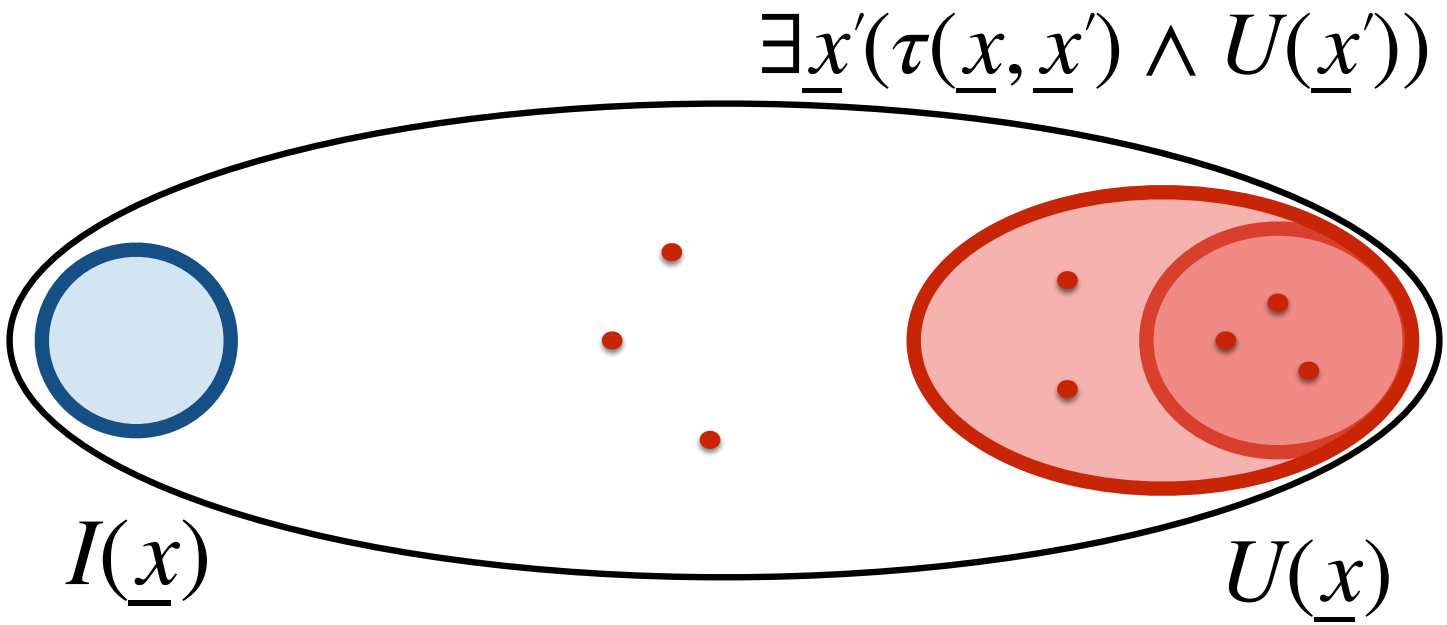
$I(\underline{x})$



$U(\underline{x})$

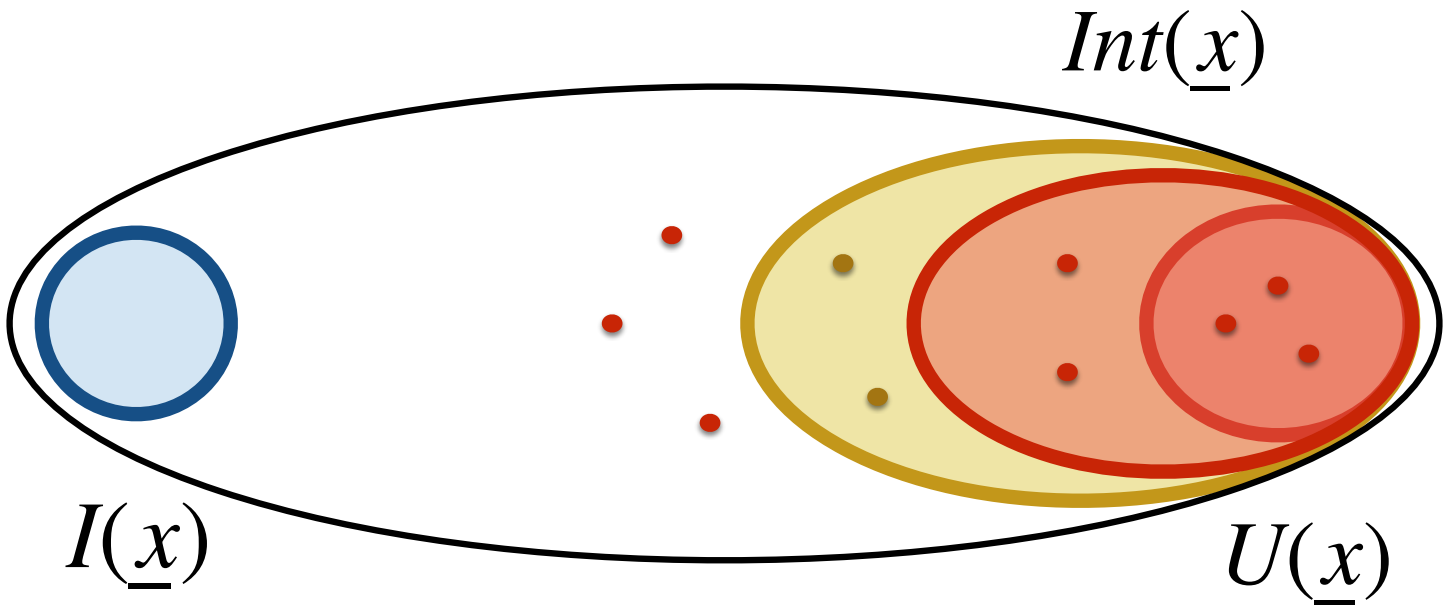
(Reverse) Interpolation: Backward

First Step: exact preimages



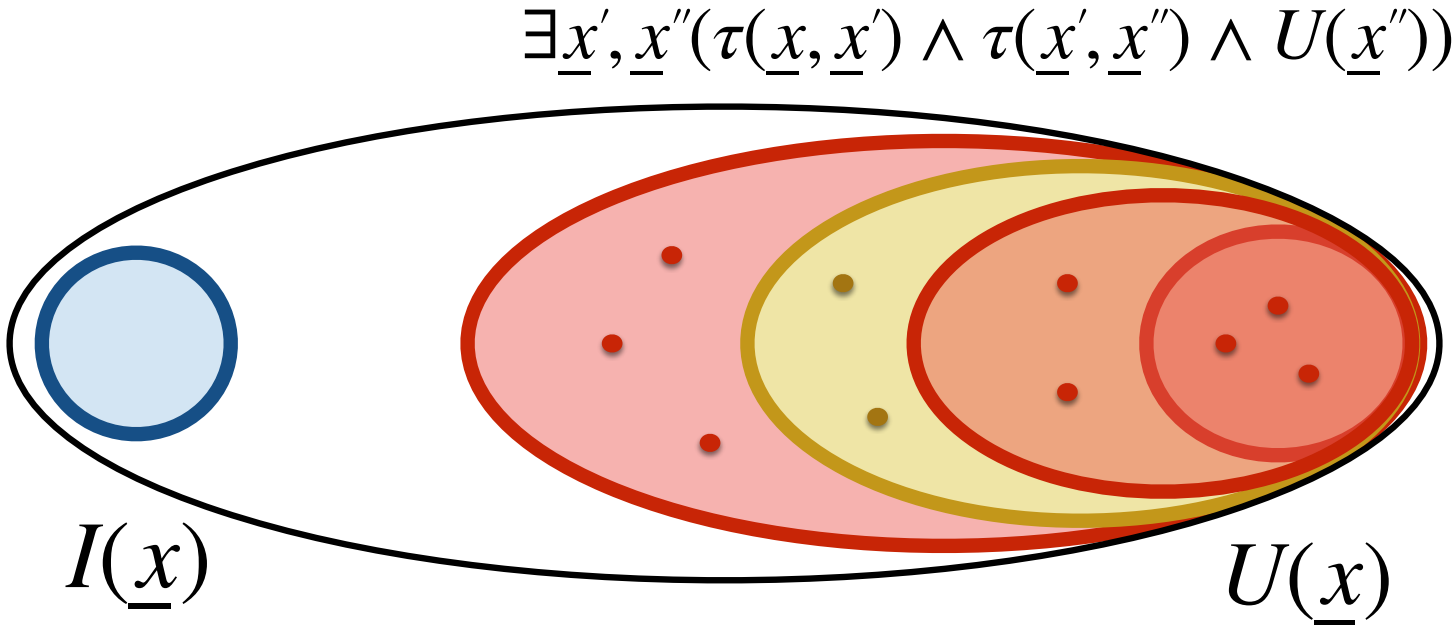
(Reverse) Ordinary Interpolation: Backward

First Step: over-approximated preimages



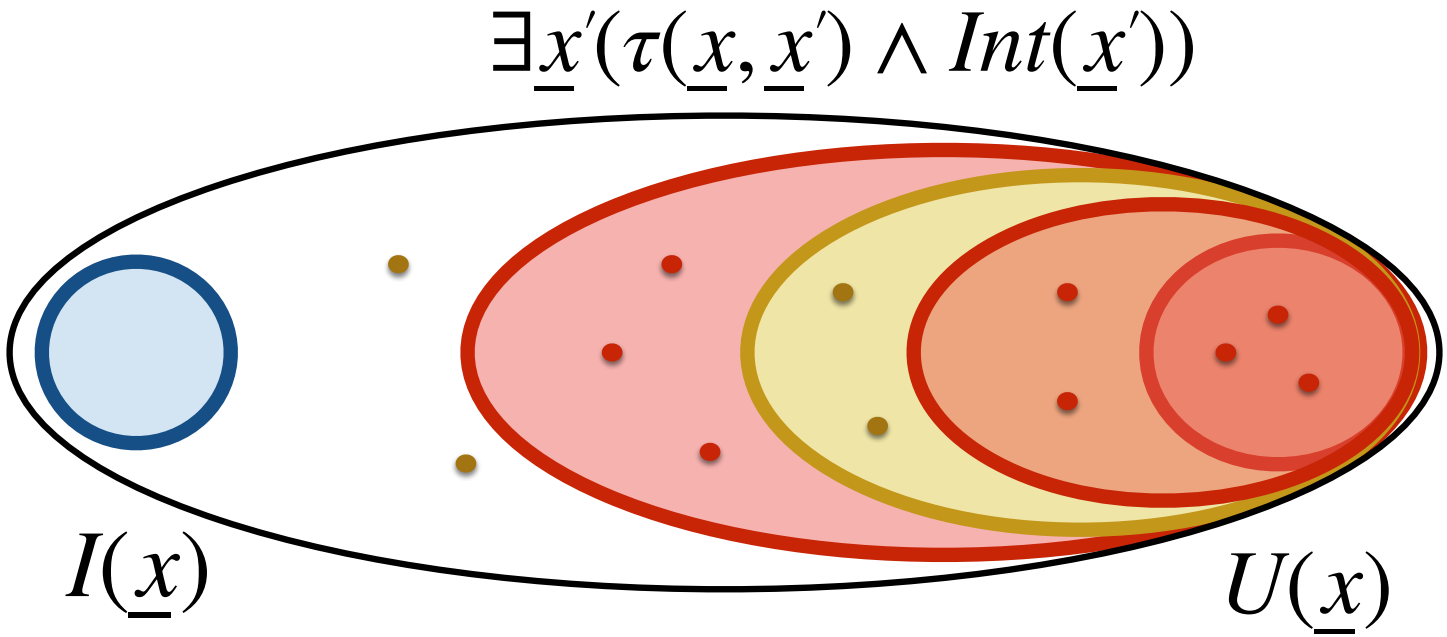
(Reverse) Ordinary Interpolation: Backward

Second Step: exact preimages



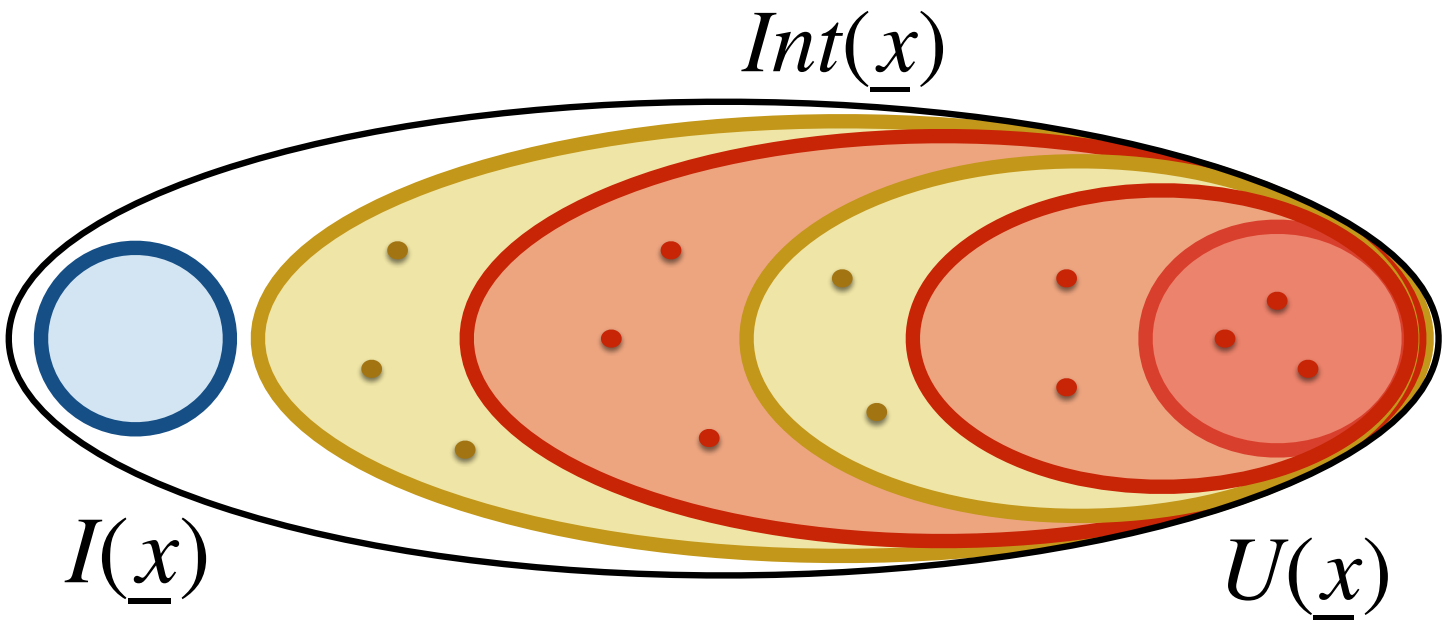
(Reverse) Ordinary Interpolation: Backward

The 'new' set of final states is given by Int

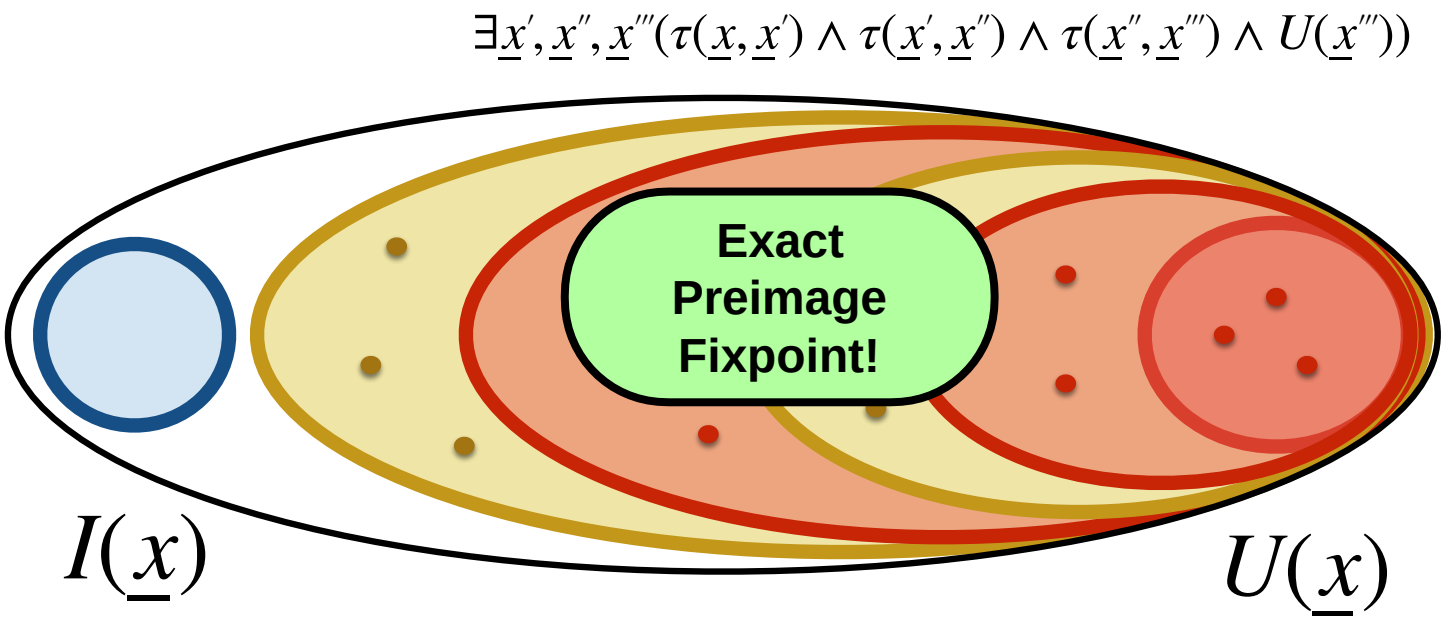


(Reverse) Ordinary Interpolation: Backward

Second Step: over-approximated preimages

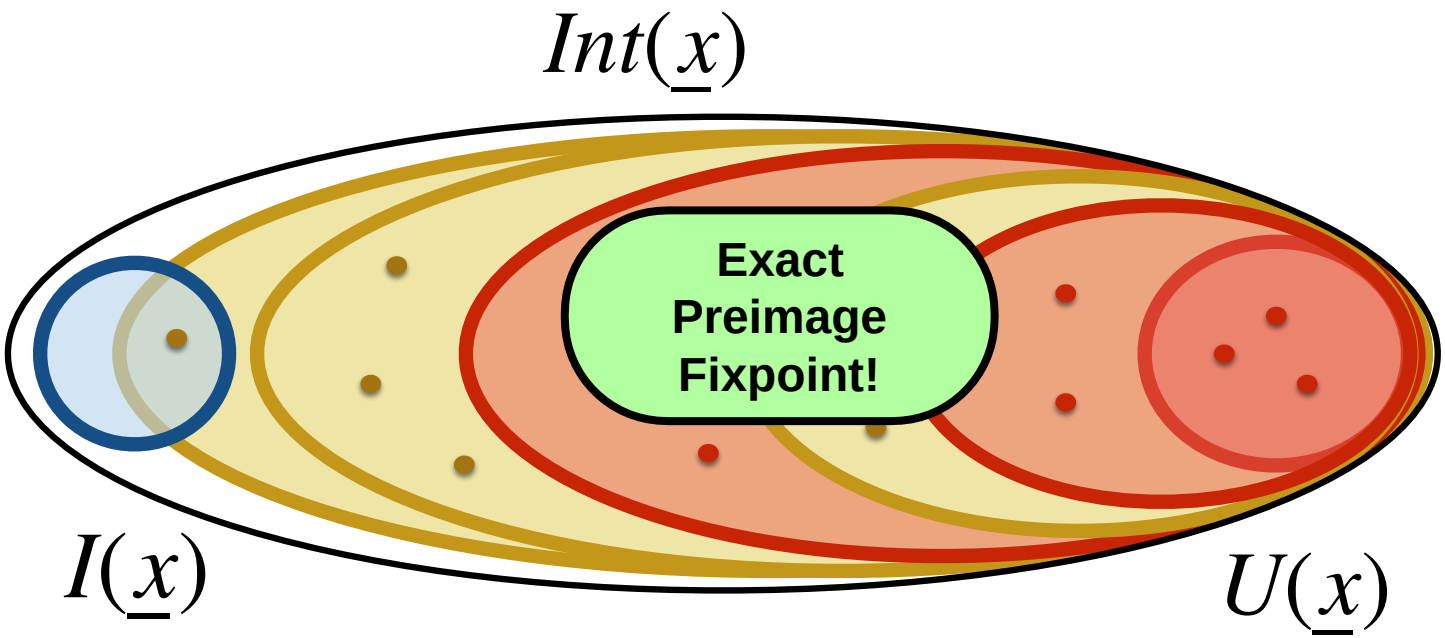


(Reverse) Ordinary Interpolation: Backward



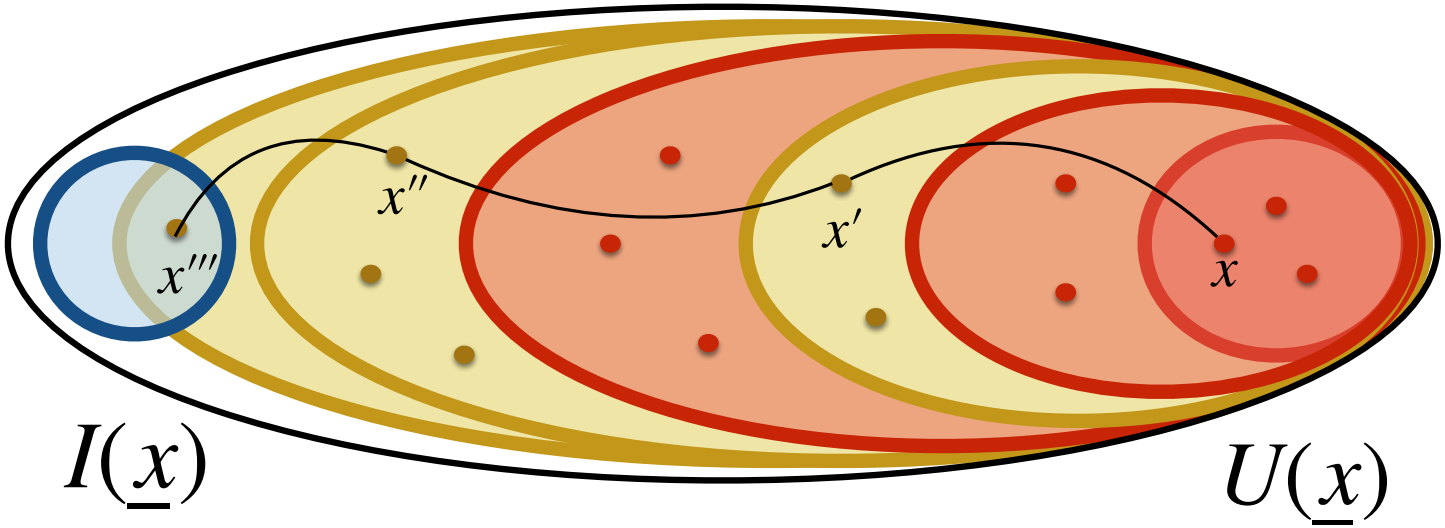
(Reverse) Ordinary Interpolation: Backward

Third Step: exact preimages. Fixpoint reached! The system is SAFE



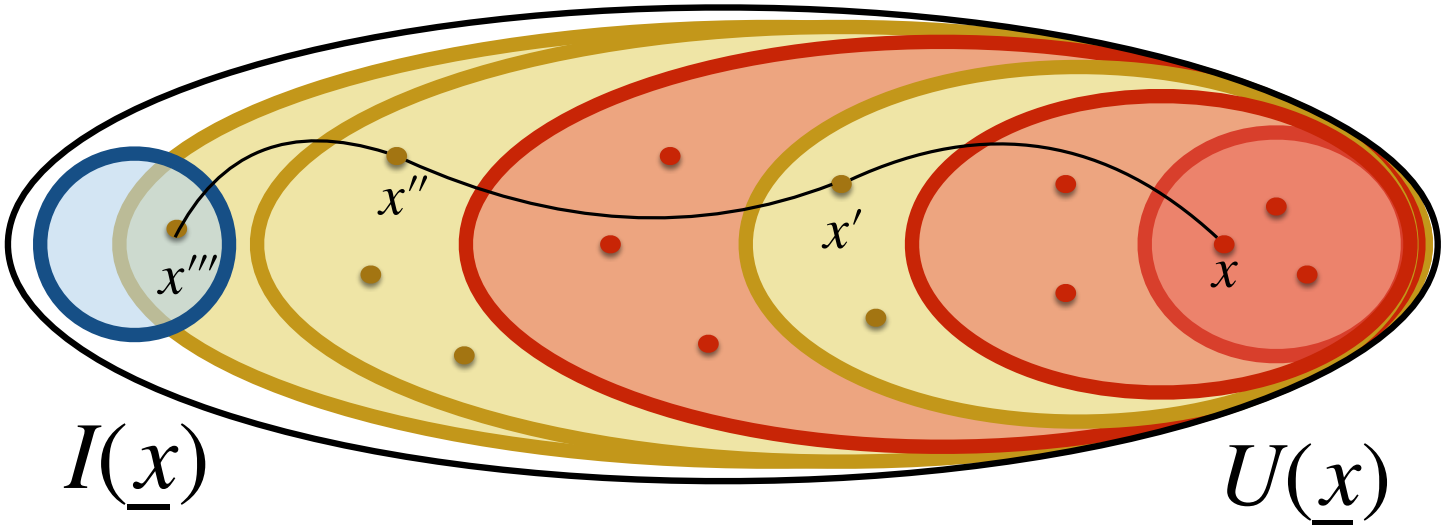
(Reverse) Ordinary Interpolation: Backward

Third Step: over-approximated preimages. Initial states intersected!

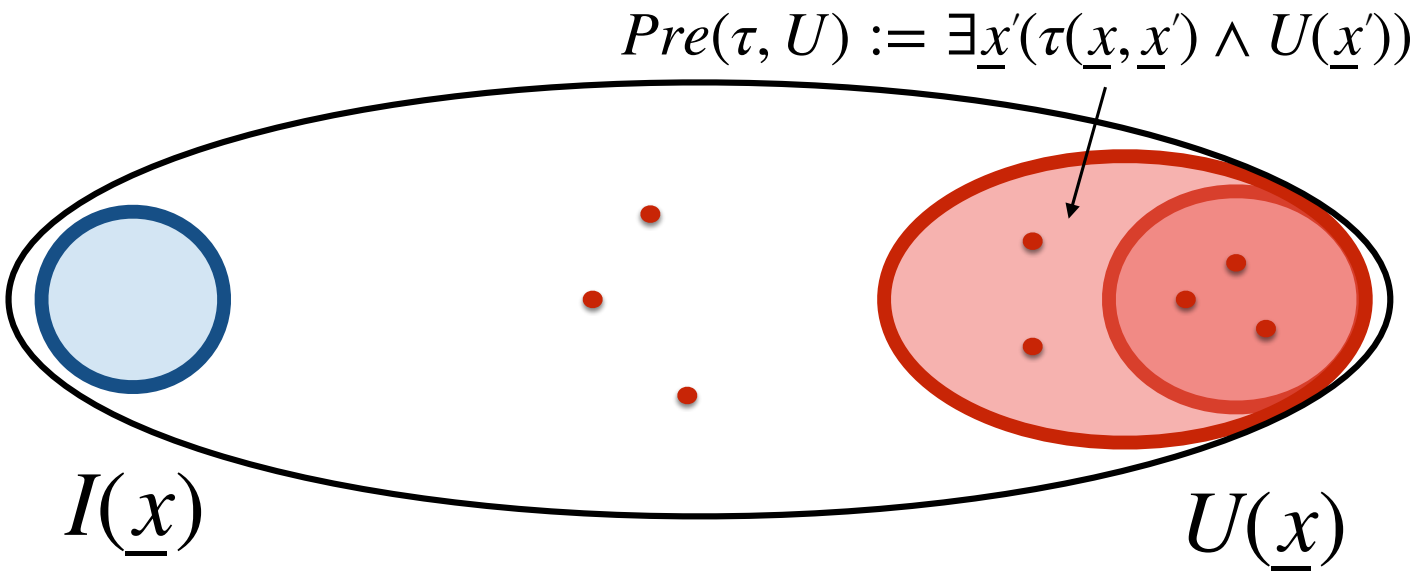


(Reverse) Ordinary Interpolation: Backward

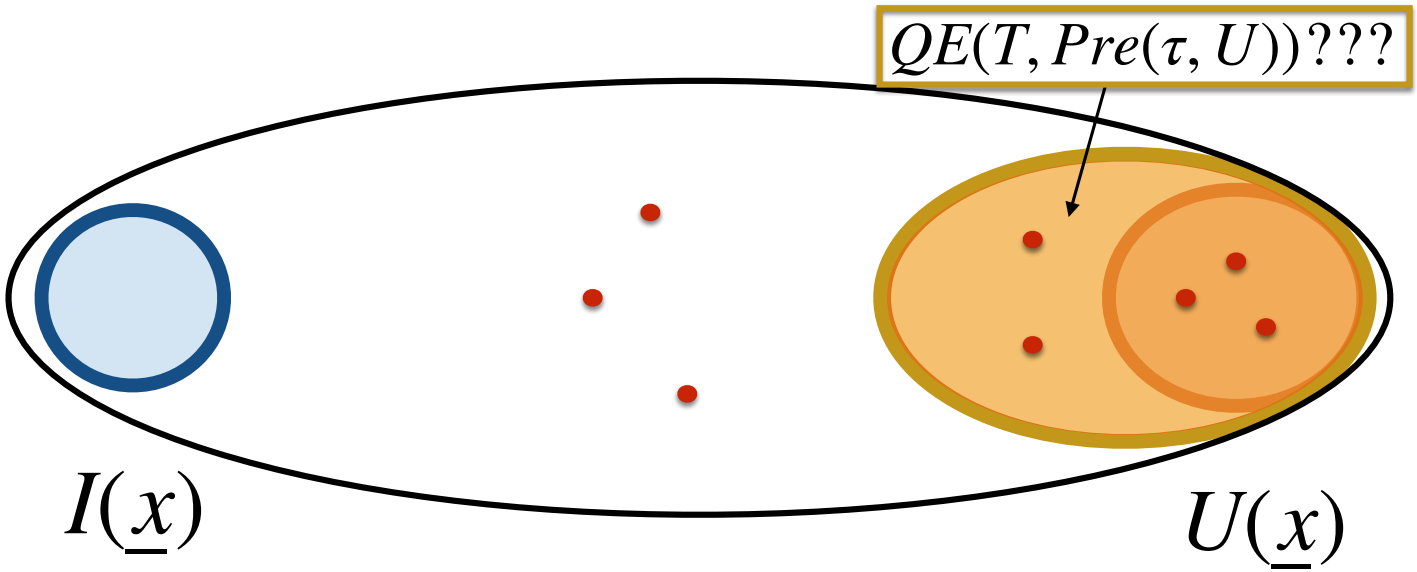
Spurious Trace! Refinement needed.



Exact Preimages? Toward Model Completions

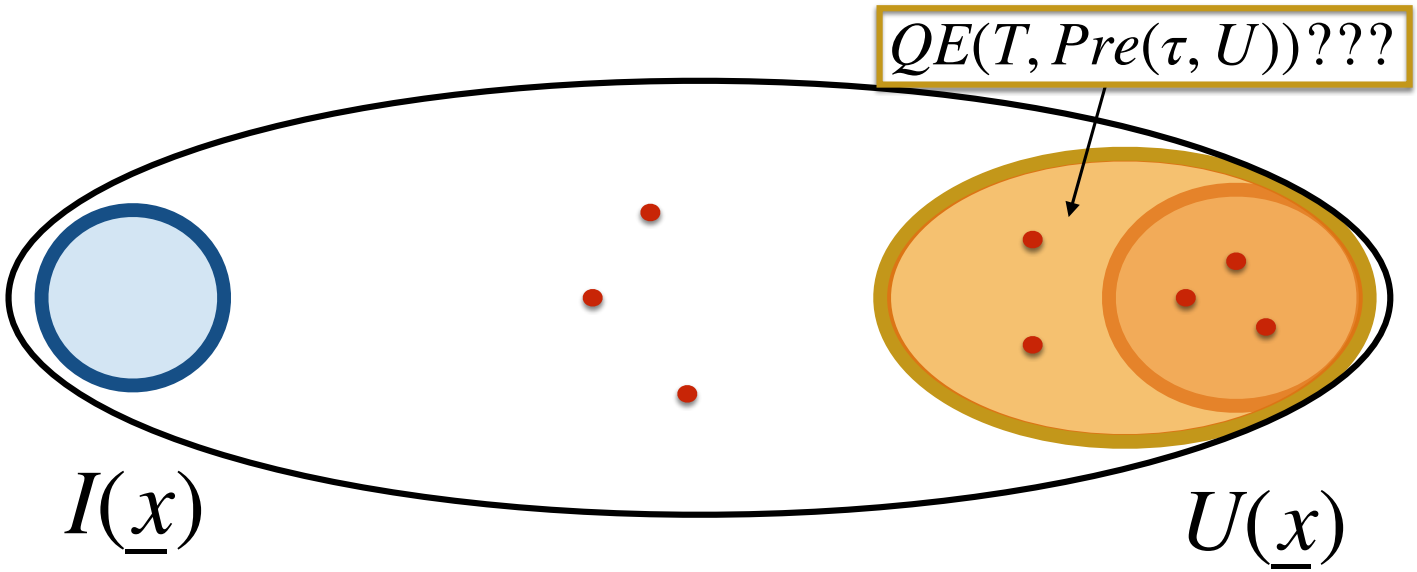


Exact Preimages? Toward Model Completions



Exact Preimages? Toward Model Completions

Every interesting T has QE?

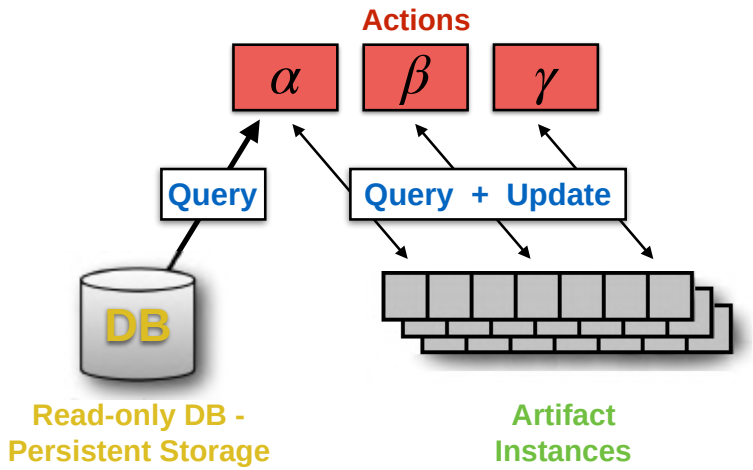


DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).

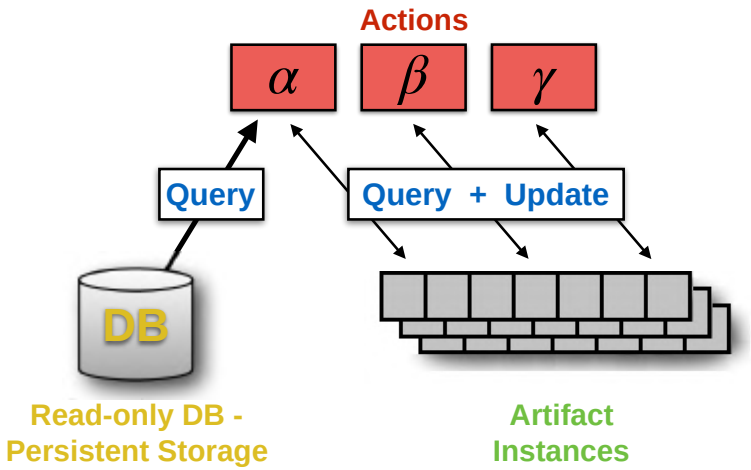
DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).
- Example: *Simple Artifact Systems (SAS)* [CGG$^+$20]



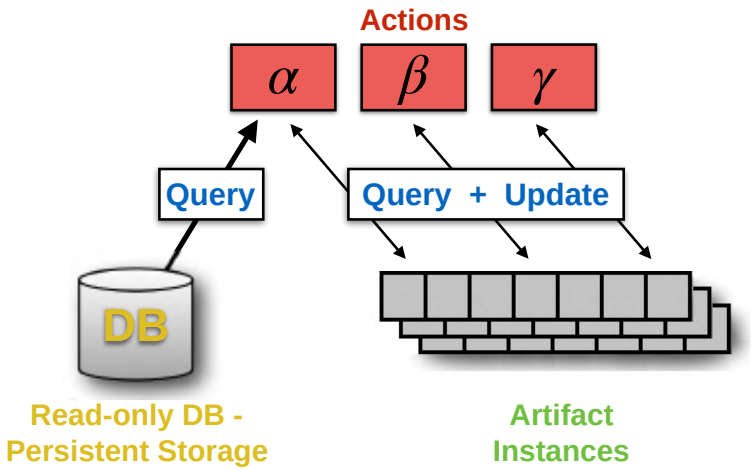
DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).
- Example: *Simple Artifact Systems (SAS)* [CGG$^+$20]
  - States: $\phi(\underline{x})$ (**quantifier-free**)



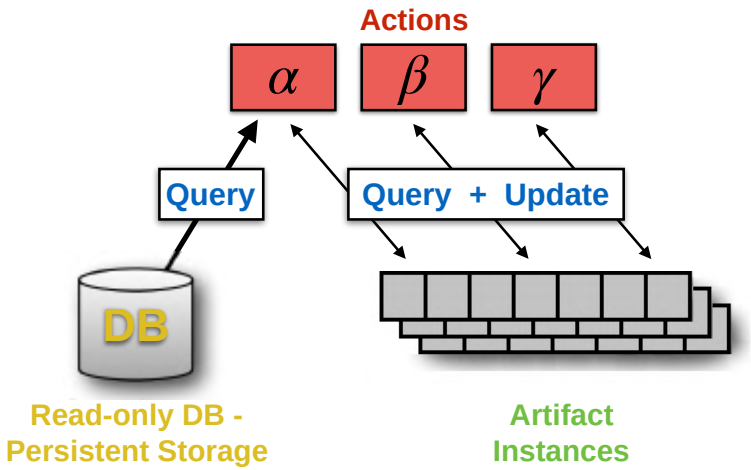
DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).
- Example: *Simple Artifact Systems (SAS)* [CGG$^+$20]
  - States: $\phi(\underline{x})$ (**quantifier-free**)
  - Tr: $\tau(\underline{x}, \underline{x}') \equiv \exists \underline{d}, \underline{i}, \underline{s}(G(\underline{x}, \underline{d}, \underline{i}, \underline{s}) \wedge U(\underline{x}, \underline{x}', \underline{d}, \underline{i}, \underline{s}))$ (**existential**)



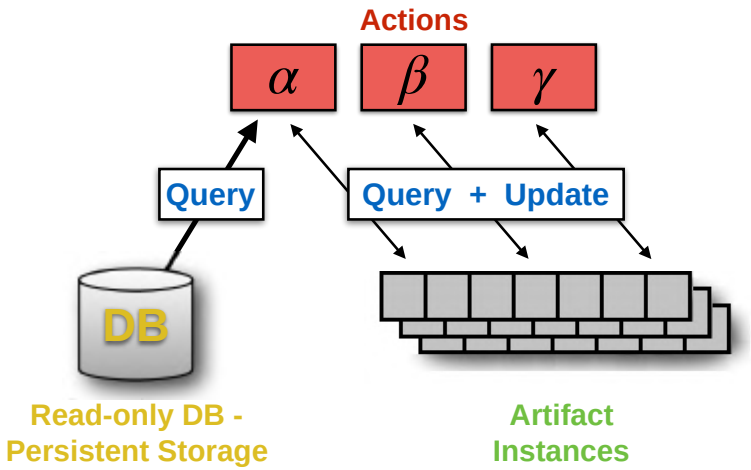
DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).
- Example: *Simple Artifact Systems (SAS)* [CGG$^+$20]
  - States: $\phi(\underline{x})$ (**quantifier-free**)
  - Tr: $\tau(\underline{x}, \underline{x}') \equiv \exists \underline{d}, \underline{i}, \underline{s}(G(\underline{x}, \underline{d}, \underline{i}, \underline{s}) \wedge U(\underline{x}, \underline{x}', \underline{d}, \underline{i}, \underline{s}))$ (**existential**)
  - $\underline{d}$: *Persistent Data* from DB;



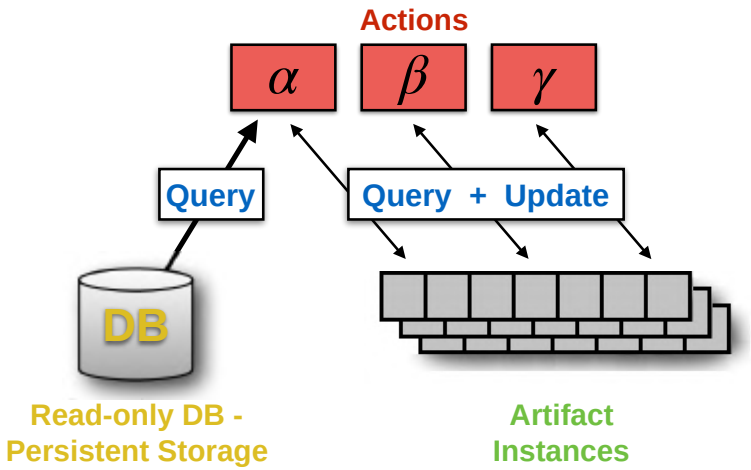
DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).
- Example: *Simple Artifact Systems (SAS)* [CGG$^+$20]
  - States: $\phi(\underline{x})$ (**quantifier-free**)
  - Tr: $\tau(\underline{x}, \underline{x}') \equiv \exists \underline{d}, \underline{i}, \underline{s}(G(\underline{x}, \underline{d}, \underline{i}, \underline{s}) \wedge U(\underline{x}, \underline{x}', \underline{d}, \underline{i}, \underline{s}))$ (**existential**)
  - $\underline{d}$: *Persistent Data* from DB;
  - $\underline{i}$: elements from *arithmetical domains* (integers or reals).



DB theories do not have QE

- Modeling and verifying **data-aware processes** $\Longrightarrow$ **combination of different theories** for defining the (Σ, T) of their TS, such as:
  (i) relational theories with key dependencies for the *database*;
  (ii) datatypes for elements from value domains (e.g., *numbers*).
- Example: *Simple Artifact Systems (SAS)* [CGG$^+$20]
  - States: $\phi(\underline{x})$ (**quantifier-free**)
  - Tr: $\tau(\underline{x}, \underline{x}') \equiv \exists \underline{d}, \underline{i}, \underline{s}(G(\underline{x}, \underline{d}, \underline{i}, \underline{s}) \wedge U(\underline{x}, \underline{x}', \underline{d}, \underline{i}, \underline{s}))$ (**existential**)
  - $\underline{d}$: *Persistent Data* from DB;
  - $\underline{i}$: elements from *arithmetical domains* (integers or reals).
  - $\underline{s}$: values of string type.



DB theories do not have QE

- *Persistent Data* from DB

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*
  - Again, $\mathcal{EUF}$ with multiple sorts (formalization of datatypes like strings).

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*
  - Again, $\mathcal{EUF}$ with multiple sorts (formalization of datatypes like strings).
  - $\mathcal{EUF}$ with multiple sorts ***does not admit QE*** (but has a model completion).

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*
  - Again, $\mathcal{EUF}$ with multiple sorts (formalization of datatypes like strings).
  - $\mathcal{EUF}$ with multiple sorts ***does not admit QE*** (but has a model completion).
- *Arithmetical values*

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*
  - Again, $\mathcal{EUF}$ with multiple sorts (formalization of datatypes like strings).
  - $\mathcal{EUF}$ with multiple sorts ***does not admit QE*** (but has a model completion).
- *Arithmetical values*
  - $\mathcal{LIA}$ and $\mathcal{LRA}$ (integer and real numbers).

DB theories do not have QE

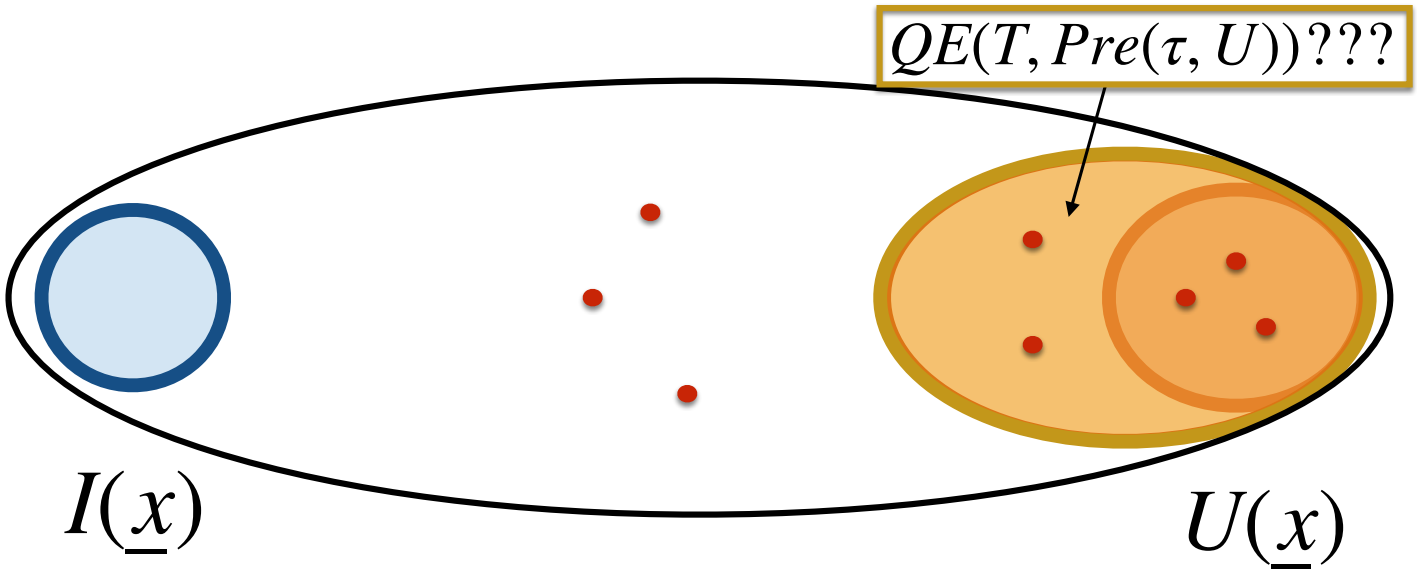
- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*
  - Again, $\mathcal{EUF}$ with multiple sorts (formalization of datatypes like strings).
  - $\mathcal{EUF}$ with multiple sorts ***does not admit QE*** (but has a model completion).
- *Arithmetical values*
  - $\mathcal{LIA}$ and $\mathcal{LRA}$ (integer and real numbers).
  - These theories admit QE (e.g., Cooper's algorithm for $\mathcal{LIA}$).

DB theories do not have QE

- *Persistent Data* from DB
  - $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations (algebraic formalization of relational DBs with key dependencies).
  - This theory ***does not admit QE*** (but it admits a *model completion*).
- *Values of string type*
  - Again, $\mathcal{EUF}$ with multiple sorts (formalization of datatypes like strings).
  - $\mathcal{EUF}$ with multiple sorts ***does not admit QE*** (but has a model completion).
- *Arithmetical values*
  - $\mathcal{LIA}$ and $\mathcal{LRA}$ (integer and real numbers).
  - These theories admit QE (e.g., Cooper's algorithm for $\mathcal{LIA}$).
- The **combination** of all these theories, clearly, ***does not admit QE***

Exact Preimages? Toward Model Completions

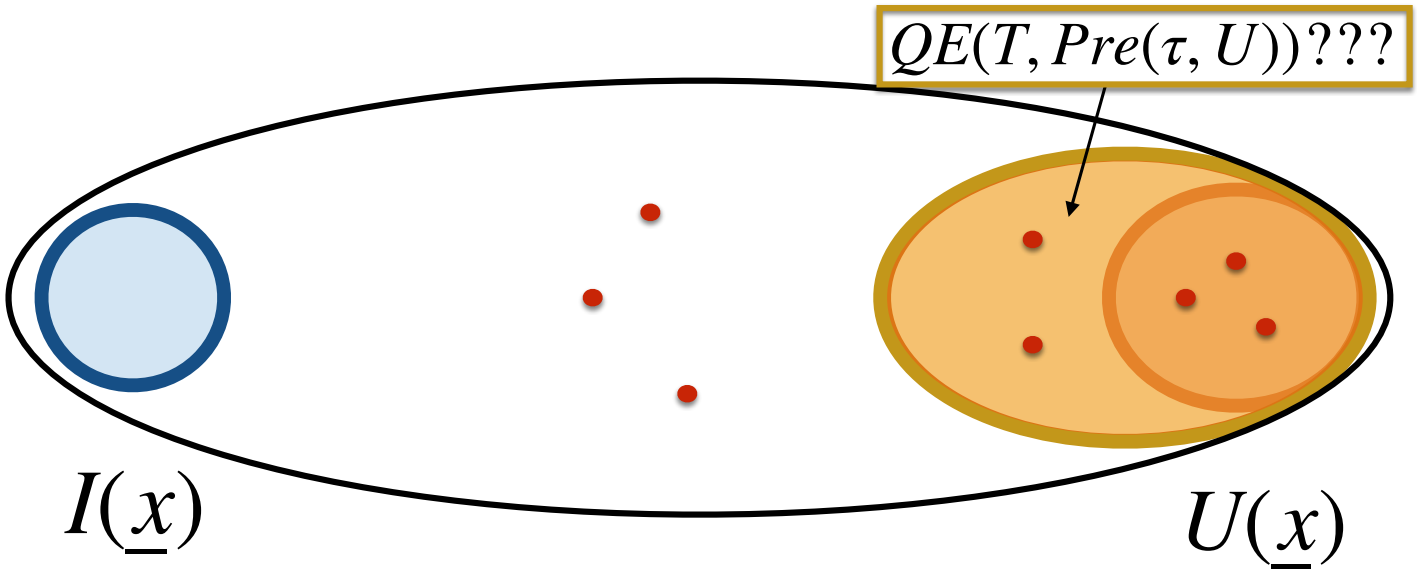
Theories used in the verification of data-aware processes do not admit QE.



Exact Preimages? Toward Model Completions

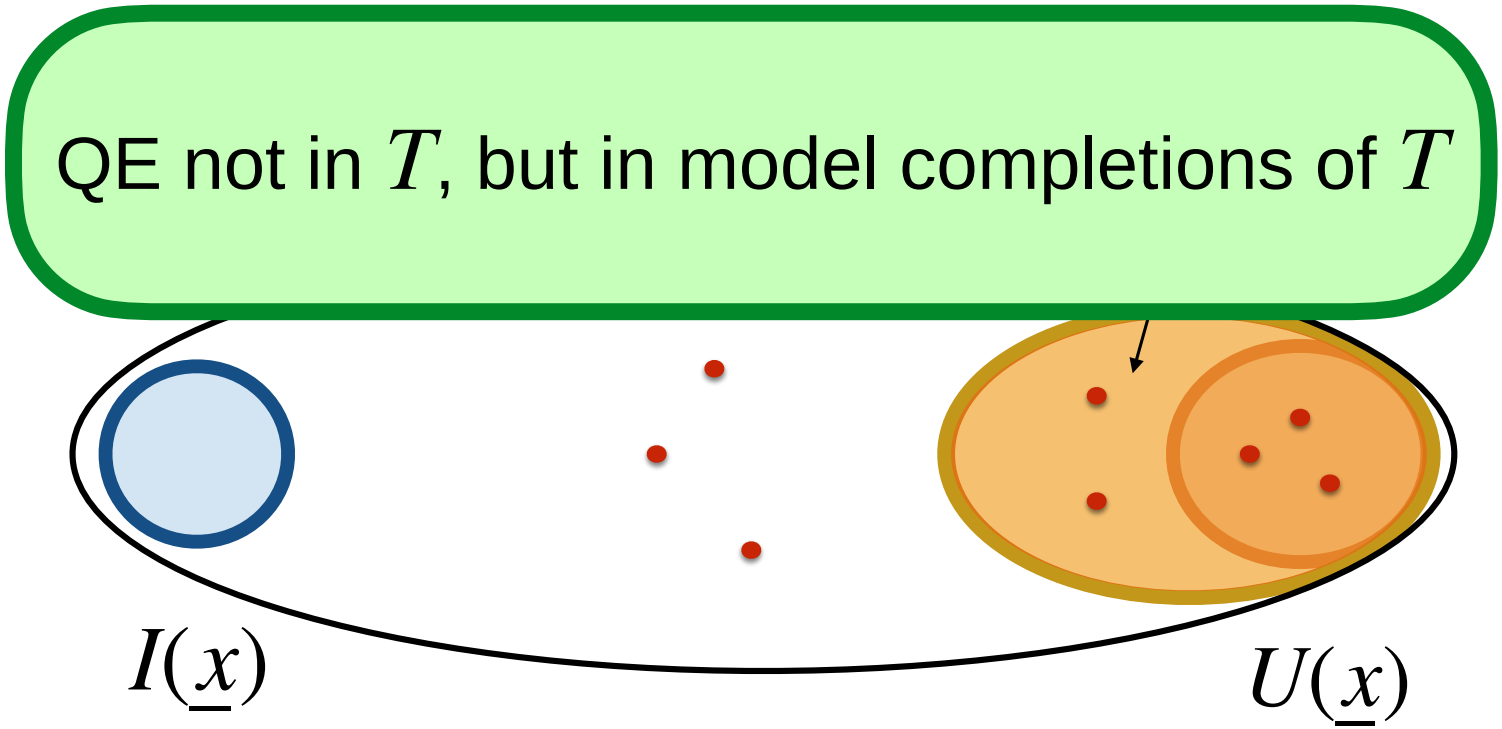
Theories used in the verification of data-aware processes do not admit QE.

What to do when QE is not available in T?



Exact Preimages? Toward Model Completions

Theories used in the verification of data-aware processes do not admit QE.



Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).
- Usually, QE is **computationally intractable** (for instance, arithmetics), or not available at all (DB theories).

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).
- Usually, QE is **computationally intractable** (for instance, arithmetics), or not available at all (DB theories).
- Methods for **symbol elimination** [KV09, JM09] (e.g., *ordinary interpolation*):

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).
- Usually, QE is **computationally intractable** (for instance, arithmetics), or not available at all (DB theories).
- Methods for **symbol elimination** [KV09, JM09] (e.g., *ordinary interpolation*):
  - quite *efficient*;

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).
- Usually, QE is **computationally intractable** (for instance, arithmetics), or not available at all (DB theories).
- Methods for **symbol elimination** [KV09, JM09] (e.g., *ordinary interpolation*):
  - quite *efficient*;
  - **over-approximate** states;

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).
- Usually, QE is **computationally intractable** (for instance, arithmetics), or not available at all (DB theories).
- Methods for **symbol elimination** [KV09, JM09] (e.g., *ordinary interpolation*):
  - quite *efficient*;
  - **over-approximate** states;
  - the computation is *not* exact.

Motivation: Recap

- Infinite-state model checking $\Longrightarrow$ sets of (*reachable*) states and transitions represented **symbolically**.
- *Precise* computations of the set of reachable states through Quantifier Elimination (QE).
- Usually, QE is **computationally intractable** (for instance, arithmetics), or not available at all (DB theories).
- Methods for **symbol elimination** [KV09, JM09] (e.g., *ordinary interpolation*):
  - quite *efficient*;
  - **over-approximate** states;
  - the computation is *not* exact.
- QE in richer theories (**model completions**) $\Leftrightarrow$ **uniform** interpolants (or, **covers** [GM08]): **tractable** in significant cases [CGG$^+$21].

Outline

QE and Finding Solutions to Equations

- **Eliminating quantifiers** $\Leftrightarrow$ **eliminating *existential* quantifiers** from primitive formulae.

QE and Finding Solutions to Equations

- **Eliminating quantifiers** $\Leftrightarrow$ **eliminating *existential* quantifiers** from primitive formulae.
- Logical counterpart of finding *witnesses*, i.e., ***solutions***, to systems of equations and/or disequalities expressed in logical form.

QE and Finding Solutions to Equations

- **Eliminating quantifiers** $\Leftrightarrow$ **eliminating *existential* quantifiers** from primitive formulae.
- Logical counterpart of finding *witnesses*, i.e., ***solutions***, to systems of equations and/or disequalities expressed in logical form.
- Model-theoretic algebra [Rob63]: powerful setting for formulating the problem in **algebraic terms**, exploiting tools from **model theory**.

QE and Finding Solutions to Equations

- **Eliminating quantifiers** $\Leftrightarrow$ **eliminating *existential* quantifiers** from primitive formulae.
- Logical counterpart of finding *witnesses*, i.e., ***solutions***, to systems of equations and/or disequalities expressed in logical form.
- Model-theoretic algebra [Rob63]: powerful setting for formulating the problem in **algebraic terms**, exploiting tools from **model theory**.
- **Model completions**: theories whose models *satisfy* all formulae that are *satisfiable in extended structures*.

Existentially Closed Models

- A quantifier-free formula with *parameters* in a model M is **solvable** if there is an extension M' of M where the formula is satisfied.

Existentially Closed Models

- A quantifier-free formula with *parameters* in a model M is **solvable** if there is an extension M' of M where the formula is satisfied.

- For instance, some *equations* do not have solutions in real numbers, but do have in the **extension** of complex numbers.

Existentially Closed Models

- A quantifier-free formula with *parameters* in a model M is **solvable** if there is an extension M' of M where the formula is satisfied.
- For instance, some *equations* do not have solutions in real numbers, but do have in the **extension** of complex numbers.
- A model M is **existentially closed** if *any solvable* quantifier-free formula already has a **solution** in M itself.

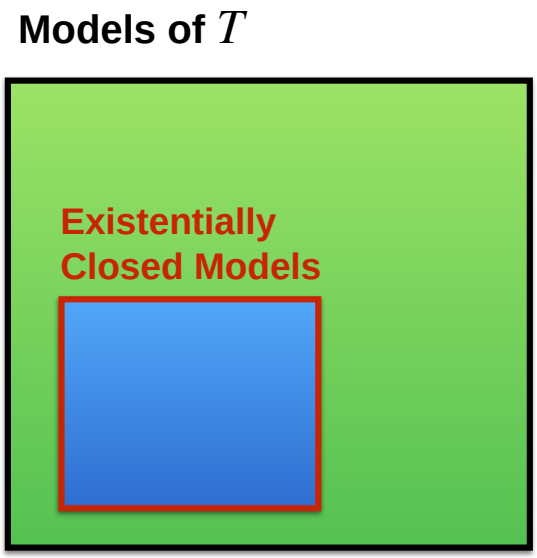
Existentially Closed Models

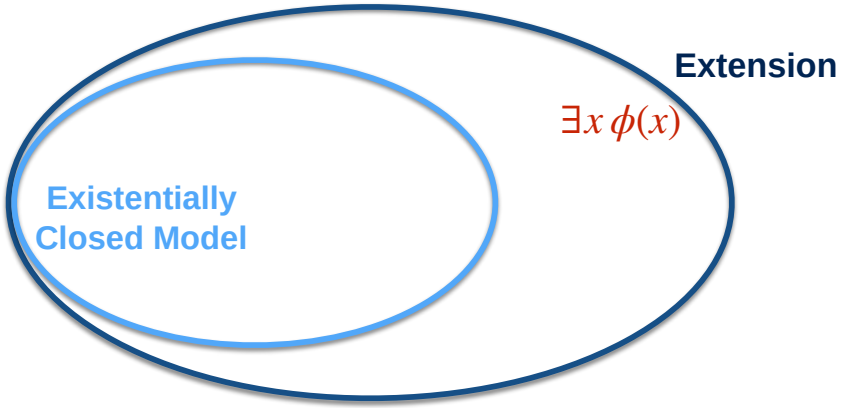
- A quantifier-free formula with *parameters* in a model M is **solvable** if there is an extension M' of M where the formula is satisfied.
- For instance, some *equations* do not have solutions in real numbers, but do have in the **extension** of complex numbers.
- A model M is **existentially closed** if *any solvable* quantifier-free formula already has a **solution** in M itself.
- In significant cases, existentially closed models of T are exactly the models of another FO theory T^*: the **model completion** of T.

Existentially closed models

Definition (Existentially closed model)

A model $\mathcal{M}$ of a Σ-theory T is said to be **existentially closed** for T if, for every extension $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N}$ is a model of T, every existential $\Sigma^{|\mathcal{M}|}$-sentence that holds in $\mathcal{N}$ holds also in $\mathcal{M}$.

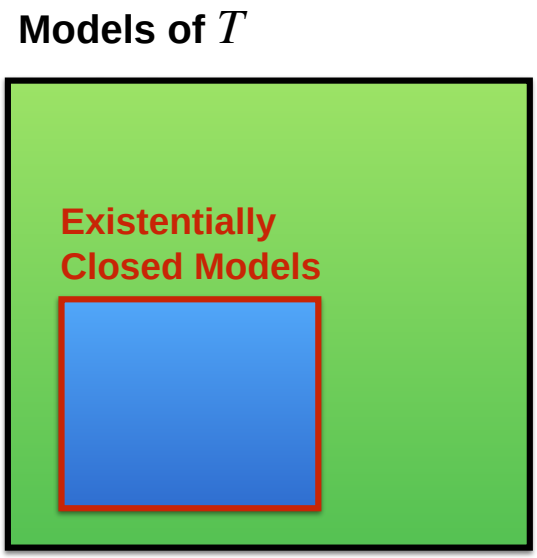


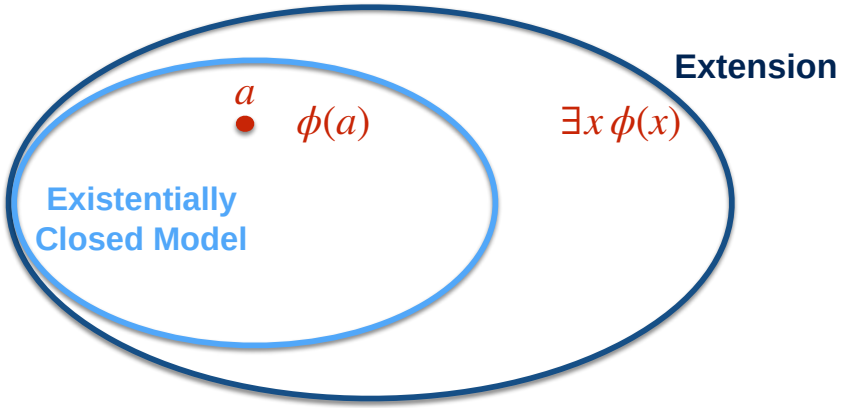


Existentially closed models

Definition (Existentially closed model)

A model $\mathcal{M}$ of a Σ-theory T is said to be **existentially closed** for T if, for every extension $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N}$ is a model of T, every existential $\Sigma^{|\mathcal{M}|}$-sentence that holds in $\mathcal{N}$ holds also in $\mathcal{M}$.

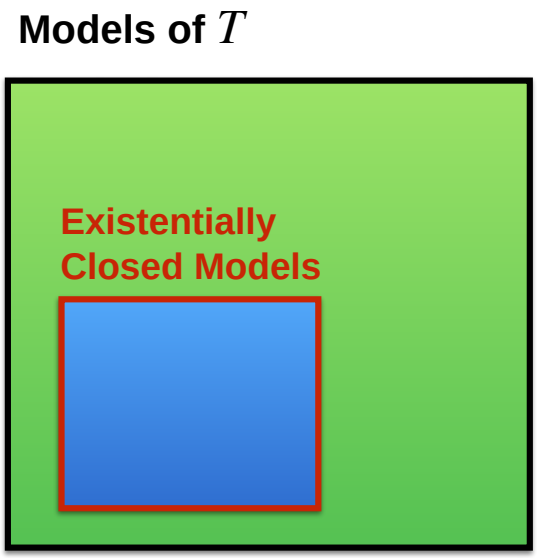


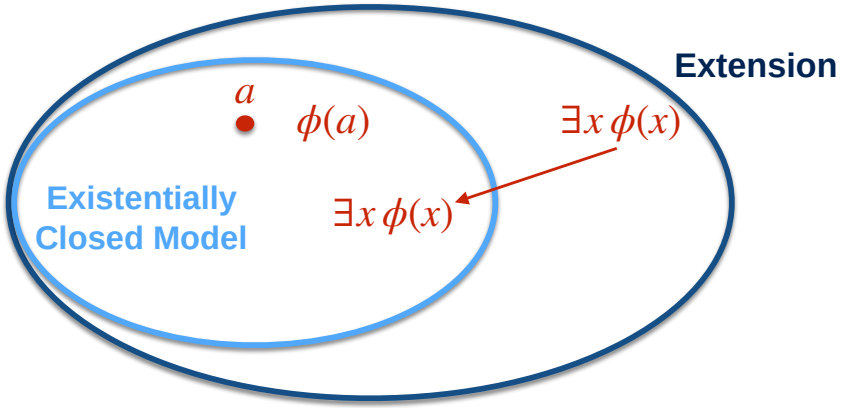


Existentially closed models

Definition (Existentially closed model)

A model $\mathcal{M}$ of a Σ-theory T is said to be **existentially closed** for T if, for every extension $\mathcal{N}$ of $\mathcal{M}$ such that $\mathcal{N}$ is a model of T, every existential $\Sigma^{|\mathcal{M}|}$-sentence that holds in $\mathcal{N}$ holds also in $\mathcal{M}$.





Model Completion

Definition (Model Completion [CK90])

Let T be a universal Σ-theory and let $T^\star \supseteq T$ be a further Σ-theory; we say that $T^\star$ is a model completion of T iff: $(\mathrm{C.i})$ every model of T can be embedded into a model of $T^\star$; .$(\mathrm{C.ii})$ $T^\star$ admits Quantifier Elimination.

Model Completion

Definition (Model Completion [CK90])

Let T be a universal Σ-theory and let $T^\star \supseteq T$ be a further Σ-theory; we say that $T^\star$ is a model completion of T iff: $(\mathrm{C.i})$ every model of T can be embedded into a model of $T^\star$; .$(\mathrm{C.ii})$ $T^\star$ admits Quantifier Elimination.

Remark

- *The models of T^* are exactly all the existentially closed models for T.*
- *T^* **admits QE** even when T does not!*

Model Completion

Definition (Model Completion [CK90])

Let T be a universal Σ-theory and let $T^\star \supseteq T$ be a further Σ-theory; we say that $T^\star$ is a model completion of T iff: $(\mathrm{C.i})$ every model of T can be embedded into a model of $T^\star$; .$(\mathrm{C.ii})$ $T^\star$ admits Quantifier Elimination.

Remark

- *The models of T^* are exactly all the existentially closed models for T.*
- *T^* **admits QE** even when T does not!*

The model completion of a theory, *if it exists*, is **unique**.

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$
- $x^2 + 1 = 0$ has no solution in $\mathbb{R}$, whereas in $\mathbb{C}$ there is an element i such that $i^2 + 1 = 0$;

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$
- $x^2 + 1 = 0$ has no solution in $\mathbb{R}$, whereas in $\mathbb{C}$ there is an element i such that $i^2 + 1 = 0$;

Notice: the solution of an equation is an existential formula in the signature of algebraic fields!

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$
- $x^2 + 1 = 0$ has no solution in $\mathbb{R}$, whereas in $\mathbb{C}$ there is an element i such that $i^2 + 1 = 0$;

Notice: the solution of an equation is an existential formula in the signature of algebraic fields!

$$\begin{array}{ll} \mathbb{C} & i : i^2 + 1 = 0 \quad \textbf{!!!} \\ \uparrow & \\ \mathbb{R} & \exists x(x^2 + 1 = 0) \, \textbf{???} \end{array}$$

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$
- $x^2 + 1 = 0$ has no solution in $\mathbb{R}$, whereas in $\mathbb{C}$ there is an element i such that $i^2 + 1 = 0$;

Notice: the solution of an equation is an existential formula in the signature of algebraic fields!

$$\begin{array}{ll} \mathbb{C} & i : i^2 + 1 = 0 \quad \textbf{!!!} \\ \uparrow & \\ \mathbb{R} & \exists x(x^2 + 1 = 0) \, \textbf{???} \end{array}$$

- $\mathbb{C}$ is an existentially closed model, whereas $\mathbb{R}$ is not.

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$
- $x^2 + 1 = 0$ has no solution in $\mathbb{R}$, whereas in $\mathbb{C}$ there is an element i such that $i^2 + 1 = 0$;

Notice: the solution of an equation is an existential formula in the signature of algebraic fields!

$$\begin{array}{ll} \mathbb{C} & i : i^2 + 1 = 0 \quad \textbf{!!!} \\ \uparrow & \\ \mathbb{R} & \exists x(x^2 + 1 = 0) \, \textbf{???} \end{array}$$

- $\mathbb{C}$ is an existentially closed model, whereas $\mathbb{R}$ is not.
- The existential quantifier can be eliminated in the theory of ***algebraically closed*** fields, but ***not*** in the theory of fields! $(\mathrm{C.i})$

A Mathematical Example

Consider the algebraic field $\mathbb{R}$ of real numbers, and the **algebraically closed** field $\mathbb{C}$ of complex numbers.

- $\mathbb{R} \hookrightarrow \mathbb{C}$, i.e., $\mathbb{C}$ is an extension of $\mathbb{R}$ (embedding) $(\mathrm{C.ii})$
- $x^2 + 1 = 0$ has no solution in $\mathbb{R}$, whereas in $\mathbb{C}$ there is an element i such that $i^2 + 1 = 0$;

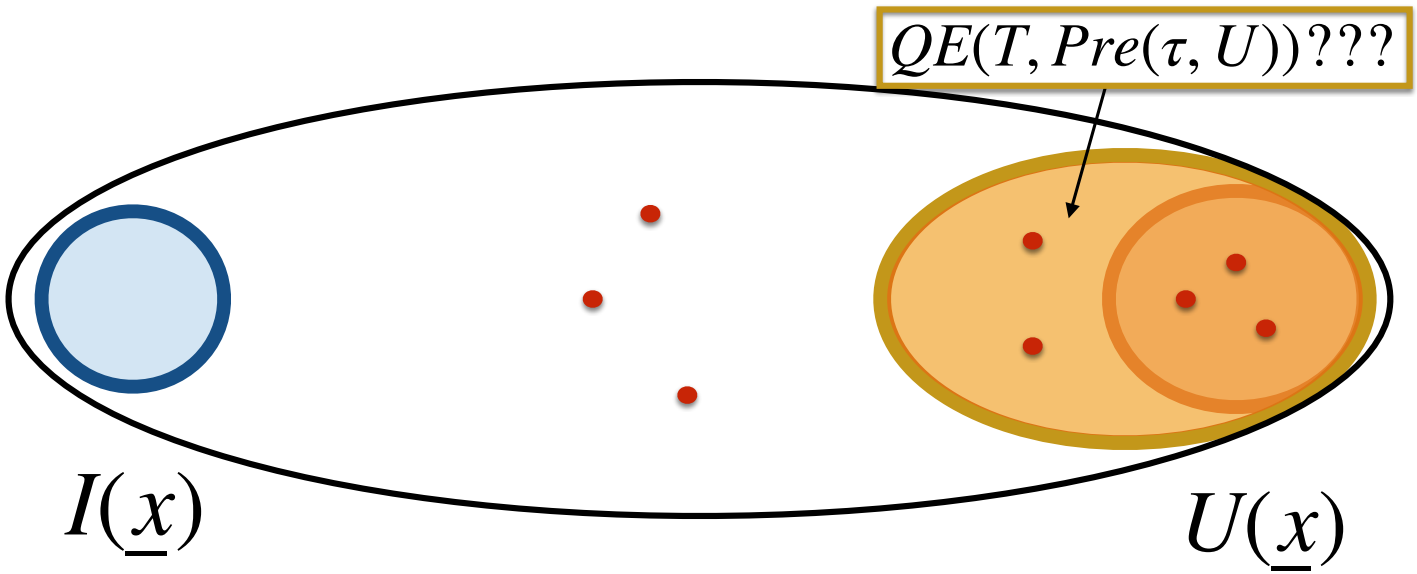
Notice: the solution of an equation is an existential formula in the signature of algebraic fields!

$$\begin{array}{ll} \mathbb{C} & i : i^2 + 1 = 0 \quad \textbf{!!!} \\ \uparrow & \\ \mathbb{R} & \exists x(x^2 + 1 = 0) \, \textbf{???} \end{array}$$

- $\mathbb{C}$ is an existentially closed model, whereas $\mathbb{R}$ is not.
- The existential quantifier can be eliminated in the theory of ***algebraically closed*** fields, but ***not*** in the theory of fields! $(\mathrm{C.i})$
- **Algebraically closed fields** are the model completion of **algebraic fields**

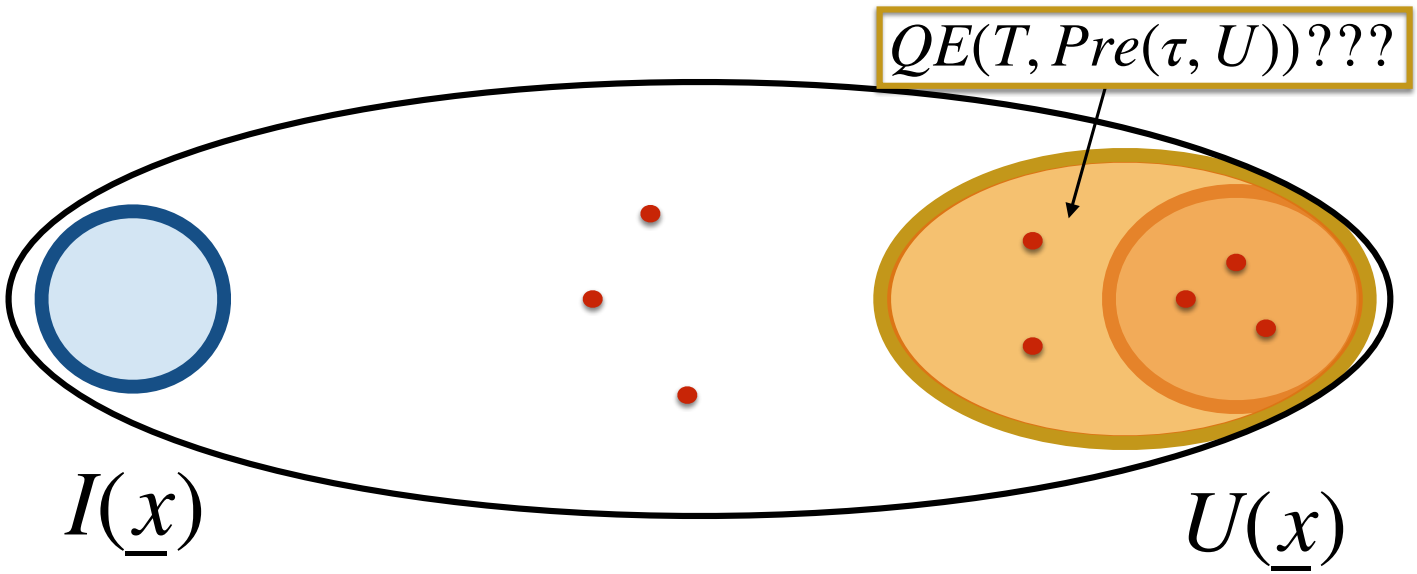
Exact Preimages? Toward Model Completions

- Theories used in the verification of data-aware processes (DB theories) do not admit QE.



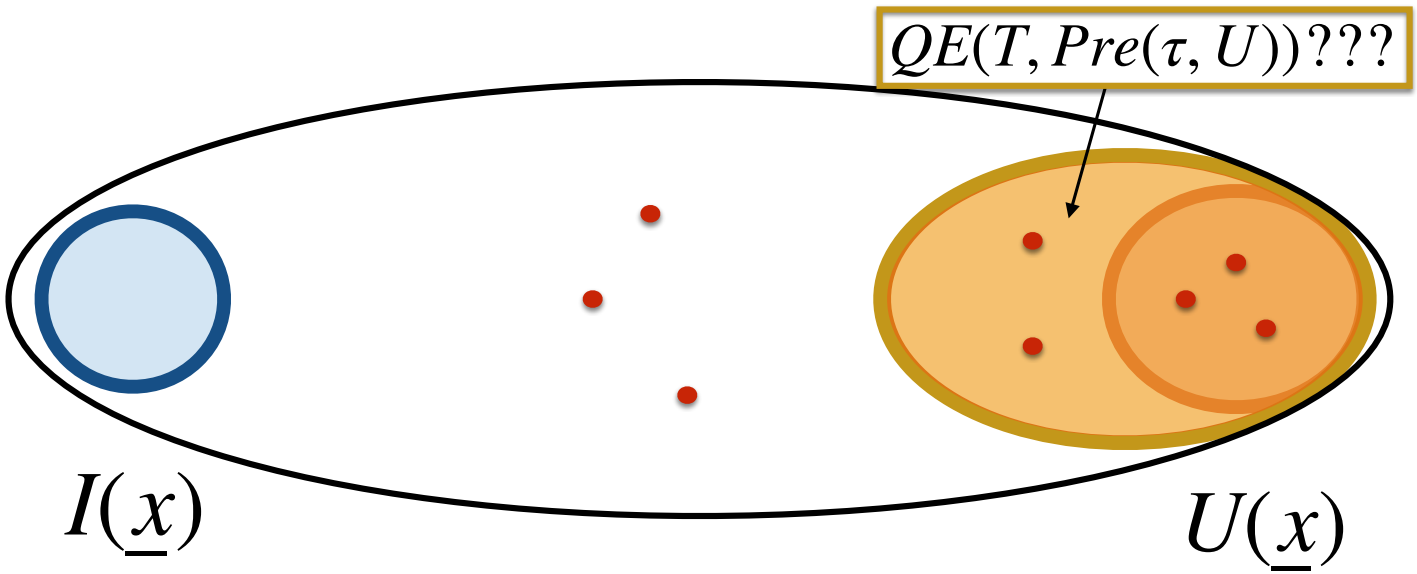
Exact Preimages? Toward Model Completions

- Theories used in the verification of data-aware processes (DB theories) do not admit QE.
- How to handle the quantifiers there?



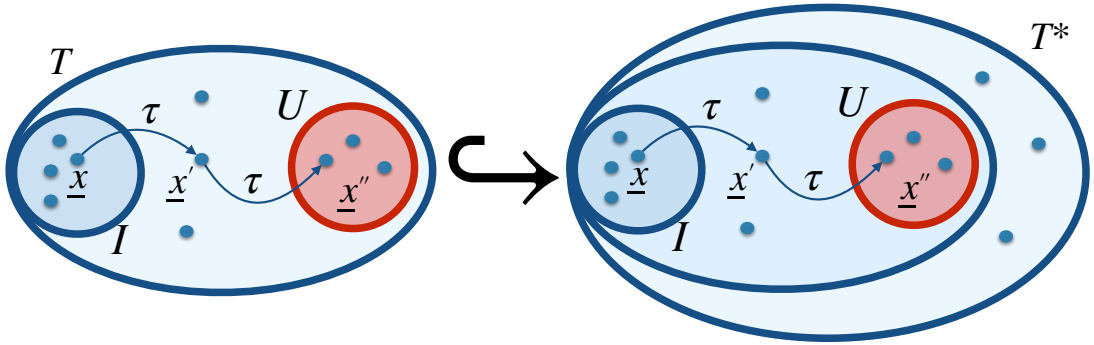
Exact Preimages? Toward Model Completions

- Theories used in the verification of data-aware processes (DB theories) do not admit QE.
- How to handle the quantifiers there?
- Is it possible to enrich T in a **'conservative way'** so as to get **computationally tractable** QE?



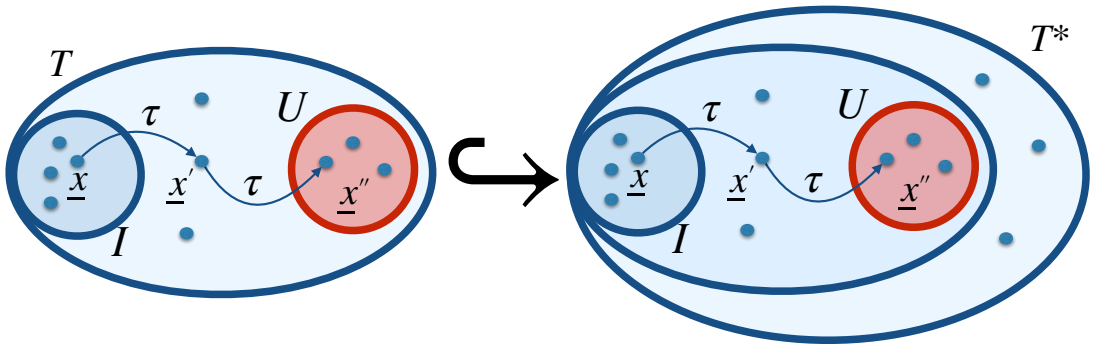
Solution: Model Completions

The answer is **YES**. Indeed, for a DB theory (Σ, T), there exists a **'minimal richer theory'** (Σ, T^*) that extends T and s.t. T^* *does* have QE. T^* is exactly the **model completion** of T.



Solution: Model Completions

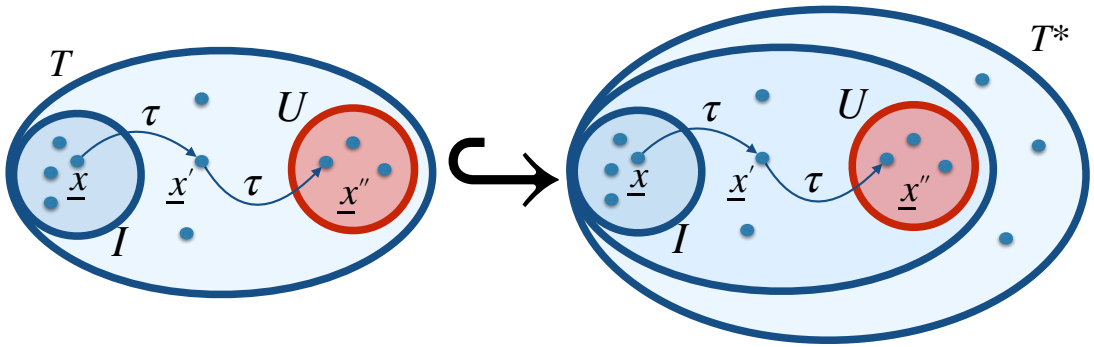
The answer is **YES**. Indeed, for a DB theory (Σ, T), there exists a **'minimal richer theory'** (Σ, T^*) that extends T and s.t. T^* *does* have QE. T^* is exactly the **model completion** of T.



- An **unsafe trace** in a model of T lifted into a model of T^* $(\mathrm{C.i})$. Here, we eliminate the quantifiers $(\mathrm{C.ii})$.

Solution: Model Completions

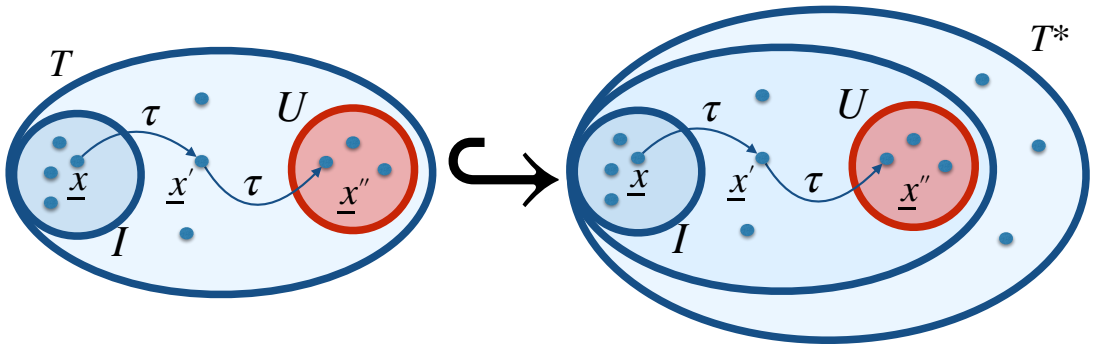
The answer is **YES**. Indeed, for a DB theory (Σ, T), there exists a **'minimal richer theory'** (Σ, T^*) that extends T and s.t. T^* *does* have QE. T^* is exactly the **model completion** of T.



- An **unsafe trace** in a model of T lifted into a model of T^* $(\mathrm{C.i})$. Here, we eliminate the quantifiers $(\mathrm{C.ii})$.
- An unsafe trace in a model of T^* is an unsafe trace in T $(T \subseteq T^*)$.

Solution: Model Completions

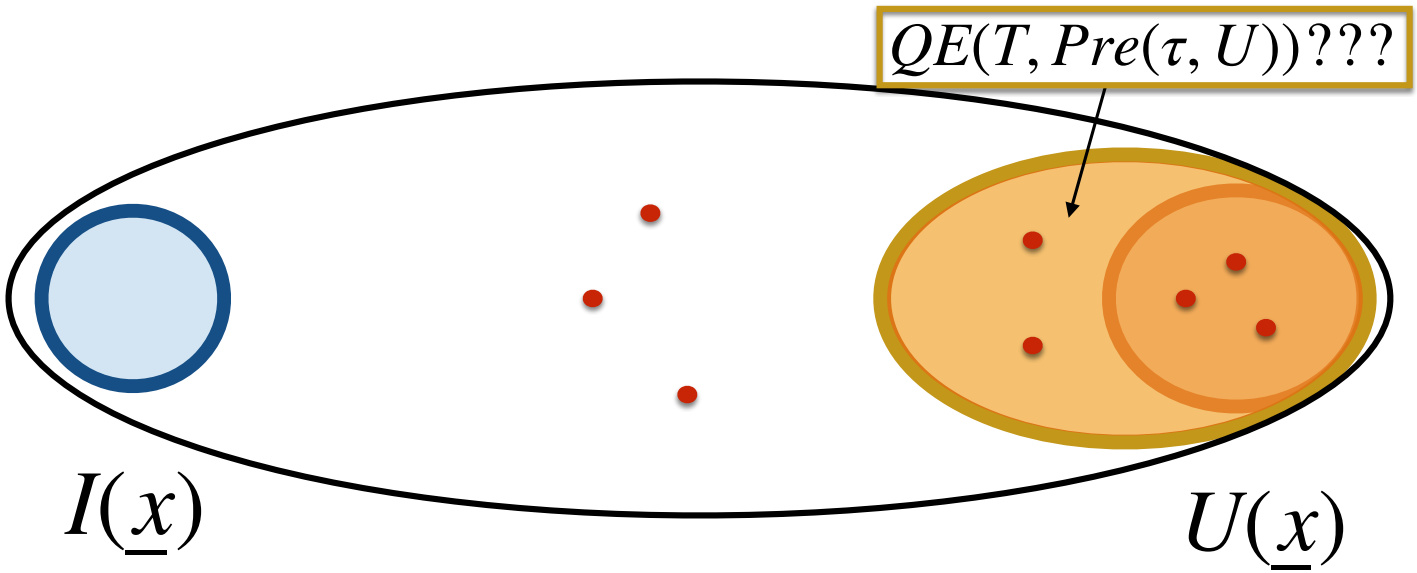
The answer is **YES**. Indeed, for a DB theory (Σ, T), there exists a **'minimal richer theory'** (Σ, T^*) that extends T and s.t. T^* *does* have QE. T^* is exactly the **model completion** of T.



- An **unsafe trace** in a model of T lifted into a model of T^* $(\mathrm{C.i})$. Here, we eliminate the quantifiers $(\mathrm{C.ii})$.
- An unsafe trace in a model of T^* is an unsafe trace in T $(T \subseteq T^*)$.
- Hence, detecting **safety** in T or in T^* are equivalent problems!

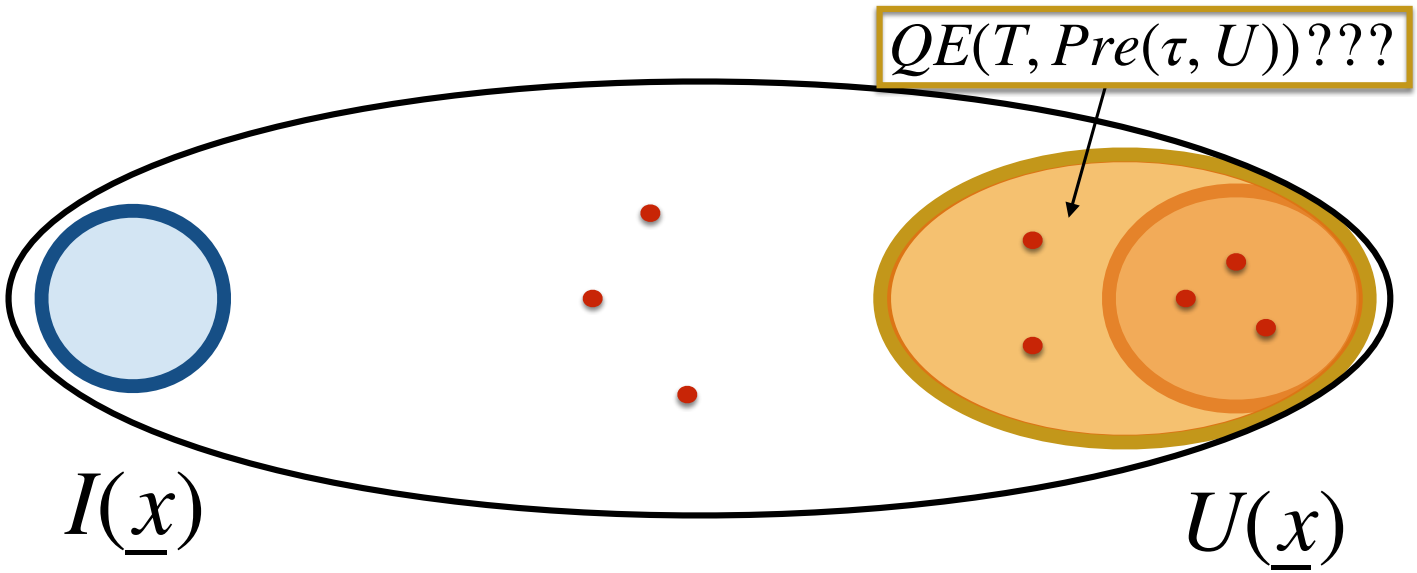
Exact Preimages? Model Completions

- Theories used in the verification of data-aware processes do not admit QE, but admit model completions!



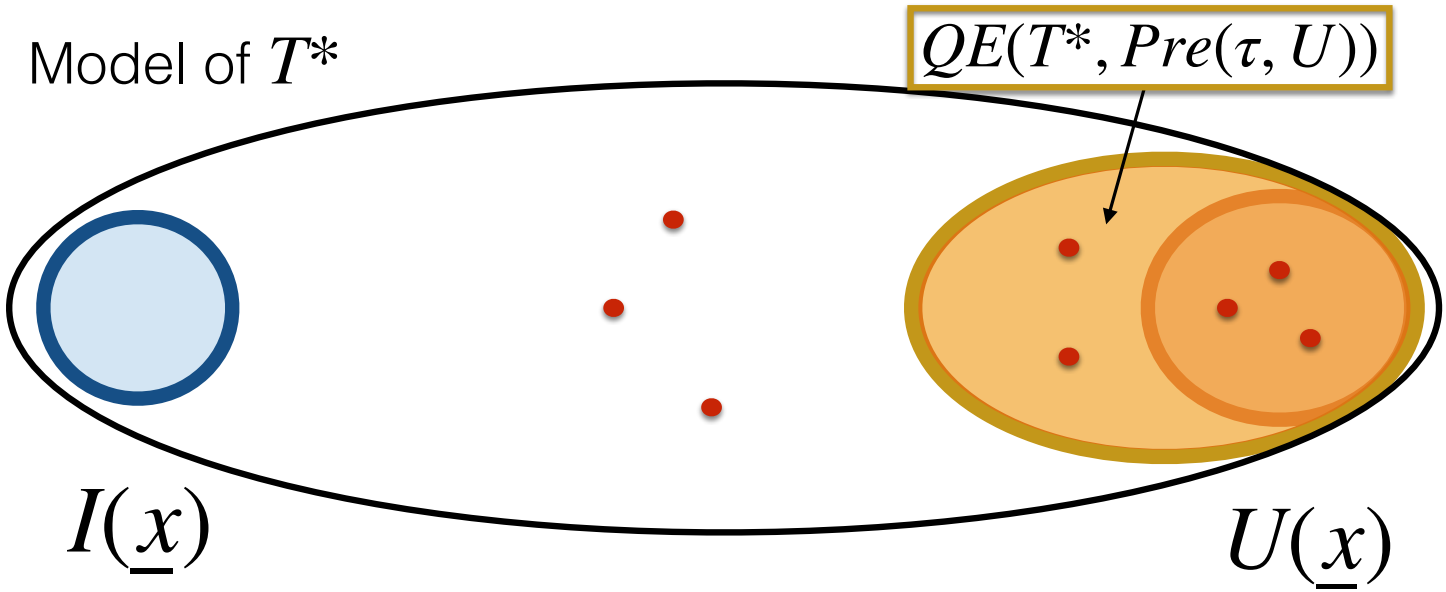
Exact Preimages? Model Completions

- Theories used in the verification of data-aware processes do not admit QE, but admit model completions!
- For instance, ***DB theories*** such as *combinations* of $\mathcal{EUF}$,$\mathcal{LIA}$,$\mathcal{LRA}$ have **model completions**.



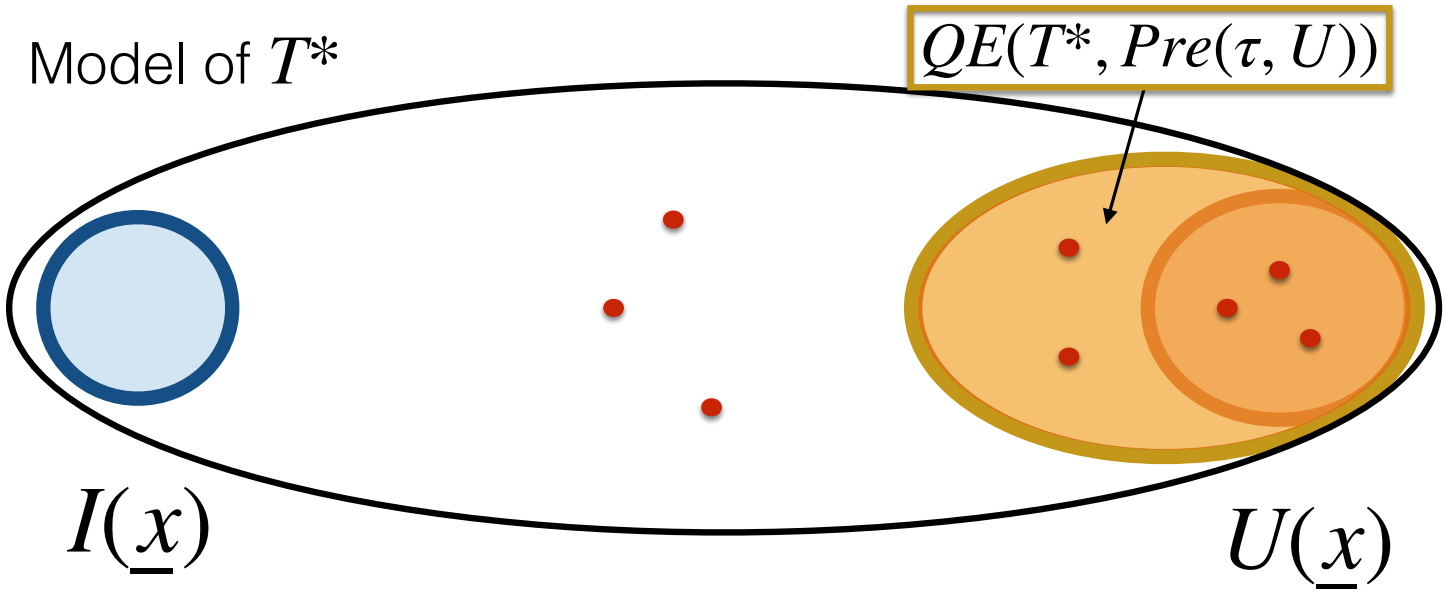
Exact Preimages? Model Completions

- Theories used in the verification of data-aware processes do not admit QE, but admit model completions!



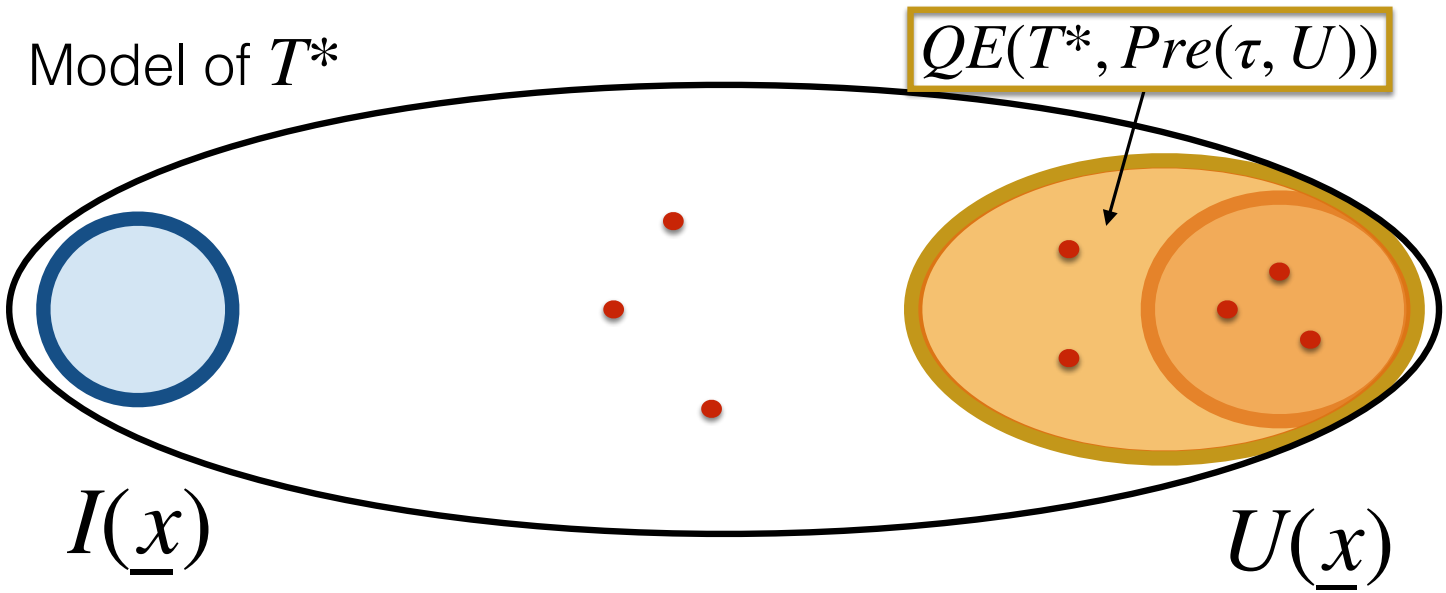
Exact Preimages? Model Completions

- Theories used in the verification of data-aware processes do not admit QE, but admit model completions!
- For instance, ***DB theories*** such as *combinations* of $\mathcal{EUF}$,$\mathcal{LIA}$,$\mathcal{LRA}$ have **model completions**.



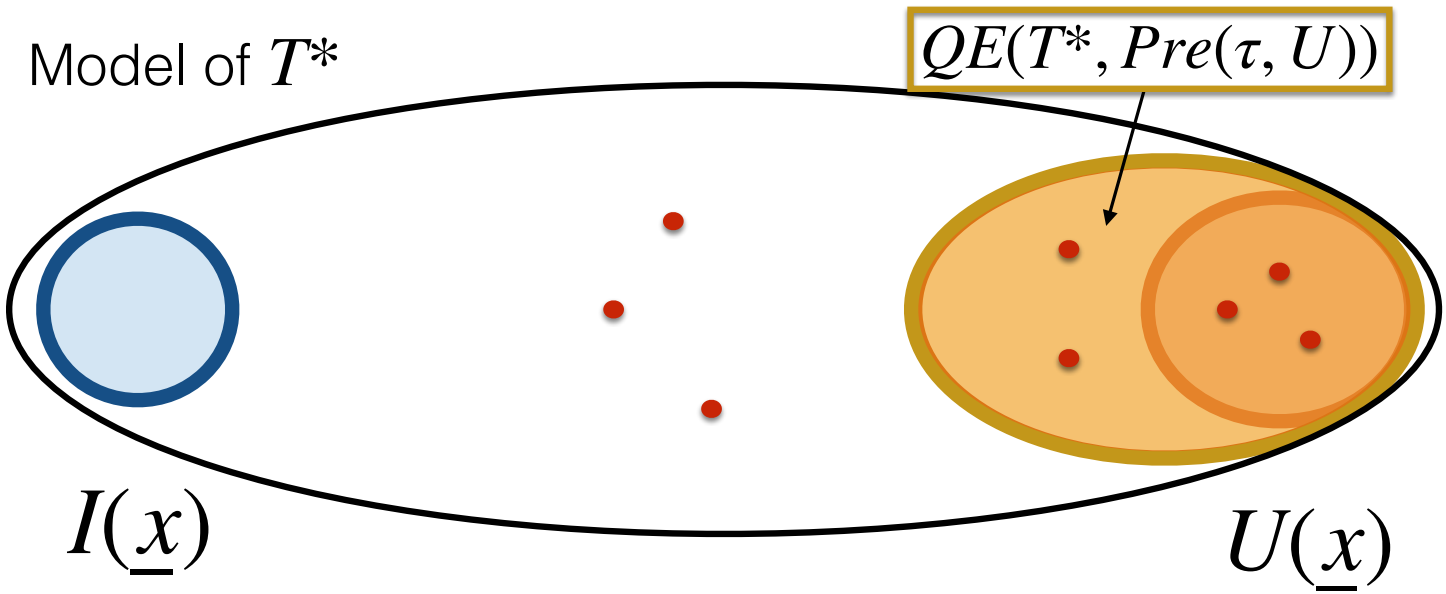
Exact Preimages? Model Completions

- The QE is performed in $T*$, and not in T!



Exact Preimages? Model Completions

- The QE is performed in $T*$, and not in T!
- **Exact Preimages!!!** No refinement needed.



Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a T-***cover*** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff

Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a T-**cover** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff

  (i) $\psi(\underline{y}) \in Res(\exists \underline{e}\, \phi) := \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \rightarrow \theta(\underline{y}, \underline{z})\}$,

Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a T-**cover** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff
  - (i) $\psi(\underline{y}) \in Res(\exists \underline{e}\, \phi) := \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \to \theta(\underline{y}, \underline{z})\}$,
  - (ii) $\psi(\underline{y})$ implies (modulo T) all the formulae in $Res(\exists \underline{e}\, \phi)$.

Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a T-***cover*** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff

  (i) $\psi(\underline{y}) \in Res(\exists \underline{e}\, \phi) := \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \rightarrow \theta(\underline{y}, \underline{z})\}$,

  (ii) $\psi(\underline{y})$ implies (modulo T) all the formulae in $Res(\exists \underline{e}\, \phi)$.

- T-**covers** and T-**quantifier-free UIs** are the same notion.

Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a T-***cover*** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff
  - (i) $\psi(\underline{y}) \in Res(\exists \underline{e}\, \phi) := \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \rightarrow \theta(\underline{y}, \underline{z})\}$,
  - (ii) $\psi(\underline{y})$ implies (modulo T) all the formulae in $Res(\exists \underline{e}\, \phi)$.
- T-**covers** and T-**quantifier-free UIs** are the same notion.
- Intuitively, it is the strongest formula implied by $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a ***T-cover*** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff
  (i) $\psi(\underline{y}) \in Res(\exists \underline{e}\, \phi) := \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \rightarrow \theta(\underline{y}, \underline{z})\}$,
  (ii) $\psi(\underline{y})$ implies (modulo T) all the formulae in $Res(\exists \underline{e}\, \phi)$.
- **T-covers** and **T-quantifier-free UIs** are the same notion.
- Intuitively, it is the strongest formula implied by $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.
- In the cover $\psi(\underline{y})$, the variables $\underline{e}$ have been 'eliminated', in some sense.

Covers [GM08]

Fix a theory T and an existential formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.

- A (qf) formula $\psi(\underline{y})$ is a T-***cover*** of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ iff
  (i) $\psi(\underline{y}) \in Res(\exists \underline{e}\, \phi) := \{\theta(\underline{y}, \underline{z}) \mid T \models \phi(\underline{e}, \underline{y}) \to \theta(\underline{y}, \underline{z})\}$,
  (ii) $\psi(\underline{y})$ implies (modulo T) all the formulae in $Res(\exists \underline{e}\, \phi)$.
- T-**covers** and T-**quantifier-free UIs** are the same notion.
- Intuitively, it is the strongest formula implied by $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.
- In the cover $\psi(\underline{y})$, the variables $\underline{e}$ have been 'eliminated', in some sense.
- But, in general, $\psi(\underline{y})$ does *not* imply $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$. Hence, usually $\psi(\underline{y})$ and $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$ are not T-equivalent.

Model Completions and UIs

Equivalence between **QE in model completions** and **Uniform Interpolants (or Covers)**

Model Completions and UIs

Equivalence between **QE in model completions** and **Uniform Interpolants (or Covers)**

Theorem (UIs and QE [CGG+21])

Let T be a universal theory. Then, T has a model completion T^* ***iff*** *T has uniform quantifier-free interpolation. T^* is axiomatized by the infinitely many sentences $\forall \underline{y}\,(\psi(\underline{y}) \rightarrow \exists \underline{e}\,\phi(\underline{e}, \underline{y}))$, where $\exists \underline{e}\,\phi(\underline{e}, \underline{y})$ is a primitive formula and ψ is a* ***UI*** *of it.*

Model Completions and UIs

Equivalence between **QE in model completions** and **Uniform Interpolants (or Covers)**

Theorem (UIs and QE [CGG+21])

Let T be a universal theory. Then, T has a model completion T^* ***iff*** *T has uniform quantifier-free interpolation. T^* is axiomatized by the infinitely many sentences $\forall \underline{y}\,(\psi(\underline{y}) \rightarrow \exists \underline{e}\,\phi(\underline{e},\underline{y}))$, where $\exists \underline{e}\,\phi(\underline{e},\underline{y})$ is a primitive formula and ψ is a* ***UI*** *of it.*

Remark

- *In T^*, thanks to the axioms, the UI $\psi(\underline{y})$* ***implies*** *$\exists \underline{e}\,\phi(\underline{e},\underline{y})$!*
- *Hence, the UI $\psi(\underline{y})$ is a quantifier-free formula that is T^***-equivalent** to the quantified formula $\exists \underline{e}\,\phi(\underline{e},\underline{y})$!*
- *The UI $\psi(\underline{y})$ is the formula that* ***eliminates the quantifiers*** *from $\exists \underline{e}\,\phi(\underline{e},\underline{y})$!*

Model Completions and UIs

Theorem (UIs and QE [CGG+21])

Let T be a universal theory. Then, T has a model completion T^* ***iff*** *T has uniform quantifier-free interpolation. T^* is axiomatized by the infinitely many sentences $\forall \underline{y}\,(\psi(\underline{y}) \rightarrow \exists \underline{e}\,\phi(\underline{e}, \underline{y}))$, where $\exists \underline{e}\,\phi(\underline{e}, \underline{y})$ is a primitive formula and ψ is a* ***UI*** *of it.*

Thus, **computing UIs** in a theory T
is **equivalent** to
eliminating quantifiers in its model completion T^*

Model Completions and UIs

Abraham Robinson

Model Completion = Existentially Closed Models

"*Equations are **solvable** in extensions*"

[not a quote, but still evocative]

'light' QE

=

strong interpolation

=

UIs

Outline

Uniform Interpolants in $\mathcal{EUF}$

- **Quantifier Elimination** in model completions: computing **exact images** when performing reachability, specifically for DB theories.

Uniform Interpolants in $\mathcal{EUF}$

- **Quantifier Elimination** in model completions: computing **exact images** when performing reachability, specifically for DB theories.
- QE in model completions **equivalent** to computing UI in the original DB theory.

Uniform Interpolants in $\mathcal{EUF}$

- **Quantifier Elimination** in model completions: computing **exact images** when performing reachability, specifically for DB theories.
- QE in model completions **equivalent** to computing UI in the original DB theory.
- $\mathcal{EUF} \implies$ formalize the basic theory of relational DBs with key dependencies.

Uniform Interpolants in $\mathcal{EUF}$

- **Quantifier Elimination** in model completions: computing **exact images** when performing reachability, specifically for DB theories.
- QE in model completions **equivalent** to computing UI in the original DB theory.
- $\mathcal{EUF} \implies$ formalize the basic theory of relational DBs with key dependencies.
- UIs in $\mathcal{EUF}$ can be computed via a **constrained version of the Superposition Calculus** with suitable application strategies;

Uniform Interpolants in $\mathcal{EUF}$

- **Quantifier Elimination** in model completions: computing **exact images** when performing reachability, specifically for DB theories.
- QE in model completions **equivalent** to computing UI in the original DB theory.
- $\mathcal{EUF} \implies$ formalize the basic theory of relational DBs with key dependencies.
- UIs in $\mathcal{EUF}$ can be computed via a **constrained version of the Superposition Calculus** with suitable application strategies;
- This computation is **tractable** for multi-sorted $\mathcal{EUF}$ with unary functions and n-ary relations, making **UIs** crucial in *verification of data-aware processes*.

Ingredients fo the UI algorithm for EUF

- Preprocessing: **Flattening** of terms/literals w.r.t. to $\underline{e}$ variables ($\underline{e}$-flattening).

Ingredients fo the UI algorithm for EUF

- Preprocessing: **Flattening** of terms/literals w.r.t. to $\underline{e}$ variables ($\underline{e}$-flattening).
  - **Example**: $f(t(y), e) = e$, where f is a function symbol and t a generic term, is an $\underline{e}$-flat literal.

Ingredients fo the UI algorithm for EUF

- Preprocessing: **Flattening** of terms/literals w.r.t. to $\underline{e}$ variables ($\underline{e}$-flattening).
  - **Example**: $f(t(y), e) = e$, where f is a function symbol and t a generic term, is an $\underline{e}$-flat literal.

- Subroutine $E(t, u)$: **unification** w.r.t. the variables $\underline{y}$ (the $\underline{e}$ variables are considered as **constants**). **Example**:

Ingredients fo the UI algorithm for EUF

- Preprocessing: **Flattening** of terms/literals w.r.t. to $\underline{e}$ variables ($\underline{e}$-flattening).
  - **Example**: $f(t(y), e) = e$, where f is a function symbol and t a generic term, is an $\underline{e}$-flat literal.

- Subroutine $E(t, u)$: **unification** w.r.t. the variables $\underline{y}$ (the $\underline{e}$ variables are considered as **constants**). **Example**:
  - $E(t, u)$ fails if $t \equiv e_i$ and $u \equiv e_j$ for $i \neq j$;

Ingredients fo the UI algorithm for EUF

- Preprocessing: **Flattening** of terms/literals w.r.t. to $\underline{e}$ variables ($\underline{e}$-flattening).
  - **Example**: $f(t(y), e) = e$, where f is a function symbol and t a generic term, is an $\underline{e}$-flat literal.

- Subroutine $E(t, u)$: **unification** w.r.t. the variables $\underline{y}$ (the $\underline{e}$ variables are considered as **constants**). **Example**:
  - $E(t, u)$ fails if $t \equiv e_i$ and $u \equiv e_j$ for $i \neq j$;
  - If $t := g(y_1)$ and $u := y_2$, we have $E(t, u) = \{t = u\}$;

Ingredients fo the UI algorithm for EUF

- Preprocessing: **Flattening** of terms/literals w.r.t. to $\underline{e}$ variables ($\underline{e}$-flattening).
  - **Example**: $f(t(y), e) = e$, where f is a function symbol and t a generic term, is an $\underline{e}$-flat literal.

- Subroutine $E(t, u)$: **unification** w.r.t. the variables $\underline{y}$ (the $\underline{e}$ variables are considered as **constants**). **Example**:
  - $E(t, u)$ fails if $t \equiv e_i$ and $u \equiv e_j$ for $i \neq j$;
  - If $t := g(y_1)$ and $u := y_2$, we have $E(t, u) = \{t = u\}$;
  - If $t := f(e, y_1, y_2)$ and $u := f(e, y_3, y_4)$, the subroutine $E(t, u)$ is defined *inductively*, i.e., $E(t, u) = E(y_1, y_3) \cup E(y_2, u_4)$, which, in turn, is equal to $E(t, u) = \{y_1 = y_3, y_2 = y_4\}$

Constrained Superposition Calculus

Rules of the Constrained Superposition Calculus (**constr-SC**) for computing UIs in $\mathcal{EUF}$ [CGG$^+$21]:

Constrained Superposition Calculus

Rules of the Constrained Superposition Calculus (**constr-SC**) for computing UIs in $\mathcal{EUF}$ [CGG+21]:

Superposition Right (Constrained)

$$\frac{l = r \parallel C \qquad s = t \parallel D}{s[r]_p = t \parallel C \cup D \cup E(s_{|p}, l)} \quad \text{if } l > r \text{ and } s > t$$

Superposition Left (Constrained)

$$\frac{l = r \parallel C \qquad s \neq t \parallel D}{s[r]_p \neq t \parallel C \cup D \cup E(s_{|p}, l)} \quad \text{if } l > r \text{ and } s > t$$

Reflection (Constrained)

$$\frac{t \neq u \parallel C}{\bot \parallel C \cup E(t, u)}$$

Demodulation (Constrained)

$$\frac{L \parallel C, \qquad l = r \parallel D}{L[r]_p \parallel C} \quad \text{if} \quad l > r,\ L_{|p} \equiv l \text{ and } C \supseteq D$$

Constrained Superposition Calculus

Rules of the Constrained Superposition Calculus (**constr-SC**) for computing UIs in $\mathcal{EUF}$ [CGG+21]:

Superposition Right (Constrained)

$$\frac{l = r \parallel C \qquad s = t \parallel D}{s[r]_p = t \parallel C \cup D \cup E(s_{|p}, l)} \quad \text{if } l > r \text{ and } s > t$$

Superposition Left (Constrained)

$$\frac{l = r \parallel C \qquad s \neq t \parallel D}{s[r]_p \neq t \parallel C \cup D \cup E(s_{|p}, l)} \quad \text{if } l > r \text{ and } s > t$$

Reflection (Constrained)

$$\frac{t \neq u \parallel C}{\bot \parallel C \cup E(t, u)}$$

Demodulation (Constrained)

$$\frac{L \parallel C, \qquad l = r \parallel D}{L[r]_p \parallel C} \quad \text{if } l > r,\ L_{|p} \equiv l \text{ and } C \supseteq D$$

- The calculus is equipped with appropriate **reduction application strategies**.

Constrained Superposition Calculus

Rules of the Constrained Superposition Calculus (**constr-SC**) for computing UIs in $\mathcal{EUF}$ [CGG+21]:

Superposition Right (Constrained)

$$\frac{l = r \parallel C \qquad s = t \parallel D}{s[r]_p = t \parallel C \cup D \cup E(s_{|p}, l)} \quad \text{if } l > r \text{ and } s > t$$

Superposition Left (Constrained)

$$\frac{l = r \parallel C \qquad s \neq t \parallel D}{s[r]_p \neq t \parallel C \cup D \cup E(s_{|p}, l)} \quad \text{if } l > r \text{ and } s > t$$

Reflection (Constrained)

$$\frac{t \neq u \parallel C}{\bot \parallel C \cup E(t, u)}$$

Demodulation (Constrained)

$$\frac{L \parallel C, \qquad l = r \parallel D}{L[r]_p \parallel C} \quad \text{if } \; l > r,\, L_{|p} \equiv l \text{ and } C \supseteq D$$

- The calculus is equipped with appropriate **reduction application strategies**.
- Each rule applies only if the subroutine E does not fail.

Termination, Correctness, and Complexity

The $\underline{e}$-flattening is needed for termination, but *not* sufficient: infinitely many $\underline{e}$-flat terms in principle generated during saturation.

Termination, Correctness, and Complexity

The $\underline{e}$-flattening is needed for termination, but *not* sufficient: infinitely many $\underline{e}$-flat terms in principle generated during saturation.

Proposition (Termination [CGG$^+$21])

Thanks to the ***application strategy****, the saturation of the initial set of $\underline{e}$-flat constrained literals always terminates after finitely many steps.*

Termination, Correctness, and Complexity

The $\underline{e}$-flattening is needed for termination, but *not* sufficient: infinitely many $\underline{e}$-flat terms in principle generated during saturation.

Proposition (Termination [CGG$^+$21])

Thanks to the ***application strategy****, the saturation of the initial set of $\underline{e}$-flat constrained literals always terminates after finitely many steps.*

Theorem (Correctness [CGG$^+$21])

If ***constr-SC*** *takes as input the primitive $\underline{e}$-flat formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$, then it gives as output the qf formula $\psi(\underline{y})$, which is a* ***(qf) uniform interpolant*** *of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.*

Termination, Correctness, and Complexity

The $\underline{e}$-flattening is needed for termination, but *not* sufficient: infinitely many $\underline{e}$-flat terms in principle generated during saturation.

Proposition (Termination [CGG+21])

Thanks to the ***application strategy****, the saturation of the initial set of $\underline{e}$-flat constrained literals always terminates after finitely many steps.*

Theorem (Correctness [CGG+21])

If ***constr-SC*** *takes as input the primitive $\underline{e}$-flat formula $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$, then it gives as output the qf formula $\psi(\underline{y})$, which is a* ***(qf) uniform interpolant*** *of $\exists \underline{e}\, \phi(\underline{e}, \underline{y})$.*

DB theory $\mathcal{EUF}$ with multiple sorts, unary functions and n-ary relations: UIs in polynomial time (with a quadratic bound).

Outline

The Problem of Combined Uniform Interpolation

- Verification of **data-aware processes** requires **combination of different theories** for formalizing the DB (e.g., $\mathcal{EUF}$, $\mathcal{LIA}$, $\mathcal{LRA}$).

The Problem of Combined Uniform Interpolation

- Verification of **data-aware processes** requires **combination of different theories** for formalizing the DB (e.g., $\mathcal{EUF}$, $\mathcal{LIA}$, $\mathcal{LRA}$).
- Important question: is it **possible** (and, if so, **under which conditions**) to **transfer** UIs from two theories T_1, T_2 to the **theory combination** $T_1 \cup T_2$?

The Problem of Combined Uniform Interpolation

- Verification of **data-aware processes** requires **combination of different theories** for formalizing the DB (e.g., $\mathcal{EUF}$, $\mathcal{LIA}$, $\mathcal{LRA}$).
- Important question: is it **possible** (and, if so, **under which conditions**) to **transfer** UIs from two theories T_1, T_2 to the **theory combination** $T_1 \cup T_2$?
- For disjoint-signature convex theories: **Yes**, under the same hypothesis for the transfer of *ordinary interpolation* – the **equality interpolating condition** [YM05].

The Problem of Combined Uniform Interpolation

- Verification of **data-aware processes** requires **combination of different theories** for formalizing the DB (e.g., $\mathcal{EUF}$, $\mathcal{LIA}$, $\mathcal{LRA}$).
- Important question: is it **possible** (and, if so, **under which conditions**) to **transfer** UIs from two theories T_1, T_2 to the **theory combination** $T_1 \cup T_2$?
- For disjoint-signature convex theories: **Yes**, under the same hypothesis for the transfer of *ordinary interpolation* – the **equality interpolating condition** [YM05].
- Using Beth definability: devise a **combined UI algorithm** that employs the **component UI algorithms**.

Equality Interpolating Condition

Sufficient and, in some sense, necessary condition for transferring **qf ordinary interpolants**:

Equality Interpolating Condition

Sufficient and, in some sense, necessary condition for transferring **qf ordinary interpolants**:

Definition ([YM05])

A convex universal theory T is *equality interpolating* iff for every pair y_1, y_2 of variables and for every pair of *constraints* $\delta_1(\underline{x}, \underline{z}_1, y_1)$, $\delta_2(\underline{x}, \underline{z}_2, y_2)$ such that $T \vdash \delta_1(\underline{x}, \underline{z}_1, y_1) \wedge \delta_2(\underline{x}, \underline{z}_2, y_2) \rightarrow y_1 = y_2$, **there exists** a term $t(\underline{x})$ such that $T \vdash \delta_1(\underline{x}, \underline{z}_1, y_1) \wedge \delta_2(\underline{x}, \underline{z}_2, y_2) \rightarrow y_1 = t(\underline{x}) \wedge y_2 = t(\underline{x})$.

Equality Interpolating Condition

Sufficient and, in some sense, necessary condition for transferring **qf ordinary interpolants**:

Definition ([YM05])

A convex universal theory T is *equality interpolating* iff for every pair y_1, y_2 of variables and for every pair of *constraints* $\delta_1(\underline{x}, \underline{z}_1, y_1)$, $\delta_2(\underline{x}, \underline{z}_2, y_2)$ such that $T \vdash \delta_1(\underline{x}, \underline{z}_1, y_1) \wedge \delta_2(\underline{x}, \underline{z}_2, y_2) \rightarrow y_1 = y_2$, **there exists** a term $t(\underline{x})$ such that $T \vdash \delta_1(\underline{x}, \underline{z}_1, y_1) \wedge \delta_2(\underline{x}, \underline{z}_2, y_2) \rightarrow y_1 = t(\underline{x}) \wedge y_2 = t(\underline{x})$.

Examples of universal **quantifier-free interpolating** and **equality interpolating** theories:

- $\mathcal{EUF}(\Sigma)$, given a signature Σ;
- recursive data theories;
- linear arithmetics.

Beth Definability and Equality Interpolating Condition

Equality interpolating can be characterized using Beth definability.

Beth Definability and Equality Interpolating Condition

Equality interpolating can be characterized using Beth definability.

Given a primitive formula $\exists \underline{z} \phi(\underline{x}, \underline{z}, y)$, we say that:

Beth Definability and Equality Interpolating Condition

Equality interpolating can be characterized using Beth definability.

Given a primitive formula $\exists \underline{z} \phi(\underline{x}, \underline{z}, y)$, we say that:

- $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *implicitly defines* y in T iff the following formula is T-valid: $\forall y \, \forall y' \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \wedge \exists \underline{z} \phi(\underline{x}, \underline{z}, y') \rightarrow y = y')$;

Beth Definability and Equality Interpolating Condition

Equality interpolating can be characterized using Beth definability.

Given a primitive formula $\exists \underline{z} \phi(\underline{x}, \underline{z}, y)$, we say that:

- $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *implicitly defines* y in T iff the following formula is T-valid: $\forall y \forall y' \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \wedge \exists \underline{z} \phi(\underline{x}, \underline{z}, y') \rightarrow y = y')$;
- $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *explicitly defines* y in T iff there is a term $t(\underline{x})$ such that the formula is T-valid: $\forall y \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \rightarrow y = t(\underline{x}))$;

Beth Definability and Equality Interpolating Condition

Equality interpolating can be characterized using Beth definability.

Given a primitive formula $\exists \underline{z} \phi(\underline{x}, \underline{z}, y)$, we say that:

- $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *implicitly defines* y in T iff the following formula is T-valid: $\forall y \forall y' \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \wedge \exists \underline{z} \phi(\underline{x}, \underline{z}, y') \rightarrow y = y')$;
- $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *explicitly defines* y in T iff there is a term $t(\underline{x})$ such that the formula is T-valid: $\forall y \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \rightarrow y = t(\underline{x}))$;
- a theory T has the ***Beth definability property*** *for primitive formulae* iff whenever a primitive formula $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *implicitly* defines the variable y then it also *explicitly* defines it.

Beth Definability and Equality Interpolating Condition

Equality interpolating can be characterized using Beth definability.

Given a primitive formula $\exists \underline{z} \phi(\underline{x}, \underline{z}, y)$:

- $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *implicitly defines* y in T iff the following formula is T-valid: $\forall y \forall y' \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \wedge \exists \underline{z} \phi(\underline{x}, \underline{z}, y') \rightarrow y = y')$;
- $\exists \underline{z} \phi(\underline{x}, \underline{z}, y)$ *explicitly defines* y in T iff there is a term $t(\underline{x})$ such that the formula is T-valid: $\forall y \, (\exists \underline{z} \phi(\underline{x}, \underline{z}, y) \rightarrow y = t(\underline{x}))$;
- T has the ***Beth definability property*** iff whenever $\exists \underline{z}\, \phi(\underline{x}, \underline{z}, y)$ *implicitly* defines the variable y then it also *explicitly* defines it.

Theorem (Key Theorem [BGR14])

*A convex theory T having quantifier-free interpolation is **equality interpolating** **iff** it has the **Beth definability property** for primitive formulae.*

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula is transformed into $\exists \underline{z}\, (\texttt{ExplDef}(\underline{z}, \underline{x}) \wedge \exists \underline{e}\, (\psi_1(\underline{x}, \underline{z}, \underline{e}) \wedge \psi_2(\underline{x}, \underline{z}, \underline{e})))$, where

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula is transformed into $\exists \underline{z}\, (\texttt{ExplDef}(\underline{z}, \underline{x}) \wedge \exists \underline{e}\, (\psi_1(\underline{x}, \underline{z}, \underline{e}) \wedge \psi_2(\underline{x}, \underline{z}, \underline{e})))$, where
  - ψ_i is a Σ_i-formula $(i = 1, 2)$

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula is transformed into $\exists \underline{z}\, (\texttt{ExplDef}(\underline{z}, \underline{x}) \wedge \exists \underline{e}\, (\psi_1(\underline{x}, \underline{z}, \underline{e}) \wedge \psi_2(\underline{x}, \underline{z}, \underline{e})))$, where
  - ψ_i is a Σ_i-formula $(i = 1, 2)$
  - $\texttt{ExplDef}(\underline{z}, \underline{x}) \equiv \bigwedge_{i=1}^{m} z_i = t_i(z_1, \ldots, z_{i-1}, \underline{x})$ (the term t_i is pure)

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula is transformed into $\exists \underline{z}\, (\texttt{ExplDef}(\underline{z}, \underline{x}) \wedge \exists \underline{e}\, (\psi_1(\underline{x}, \underline{z}, \underline{e}) \wedge \psi_2(\underline{x}, \underline{z}, \underline{e})))$, where
  - ψ_i is a Σ_i-formula $(i = 1, 2)$
  - $\texttt{ExplDef}(\underline{z}, \underline{x}) \equiv \bigwedge_{i=1}^{m} z_i = t_i(z_1, \ldots, z_{i-1}, \underline{x})$ (the term t_i is pure)
  - $\underline{x}$ are called *parameters*, $\underline{z}$ *defined variables* and $\underline{e}$ *existential variables*

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula is transformed into $\exists \underline{z}\, (\texttt{ExplDef}(\underline{z}, \underline{x}) \wedge \exists \underline{e}\, (\psi_1(\underline{x}, \underline{z}, \underline{e}) \wedge \psi_2(\underline{x}, \underline{z}, \underline{e})))$, where
  - ψ_i is a Σ_i-formula $(i = 1, 2)$
  - $\texttt{ExplDef}(\underline{z}, \underline{x}) \equiv \bigwedge_{i=1}^{m} z_i = t_i(z_1, \ldots, z_{i-1}, \underline{x})$ (the term t_i is pure)
  - $\underline{x}$ are called *parameters*, $\underline{z}$ *defined variables* and $\underline{e}$ *existential variables*
  - ψ_1, ψ_2 always contain the literals $e_i \neq e_j$ (for $i \neq j$).

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula transformed into a 'purified' formula, where (acyclic) recursive *explicit definitions* are isolated.
- Intuitively, for every $e_i \in \underline{e}$:
  - either e_i is **implicitly definable**, so, via *Beth definability*, can be made **explicitly defined**;
  - or e_i is **not implicitly definable**.

Computing Combined UIs

- Every Σ_i-theory T_i from now on is ***convex, stably infinite, equality interpolating, universal and admitting a model completion*** T_i^*.
- Given a Σ_1-theory T_1 and a Σ_2-theory T_2: compute a $T_1 \cup T_2$-**UI** for $\exists \underline{e}\, \phi(\underline{x}, \underline{e})$ (Initial Formula).
- **Preprocessing**: the Initial Formula transformed into a 'purified' formula, where (acyclic) recursive *explicit definitions* are isolated.
- Intuitively, for every $e_i \in \underline{e}$:
  - either e_i is **implicitly definable**, so, via *Beth definability*, can be made **explicitly defined**;
  - or e_i is **not implicitly definable**.
- The **combined algorithm discovers** implicitly definable variables and eliminates them via explicit definability (via Beth definability): the obtained formula is called *terminal*.

Transfer of UIs

Given a *terminal* formula, the $T_1 \cup T_2$-UI is modularly computed by:

- computing the T_1-UI of the Σ_1-pure part of the formula.
- computing the T_2-UI of the Σ_2-pure part of the formula.
- **unfolding** the (recursive) explicit definitions.

Transfer of UIs

Given a *terminal* formula, the $T_1 \cup T_2$-UI is modularly computed by:

- computing the T_1-UI of the Σ_1-pure part of the formula.
- computing the T_2-UI of the Σ_2-pure part of the formula.
- **unfolding** the (recursive) explicit definitions.

Theorem ([CGG+22])

*Let T_1, T_2 be FO theories with the aforementioned hypotheses. Then $T_1 \cup T_2$ admits a model completion too. UIs in $T_1 \cup T_2$ can be effectively **computed** using the translating procedure above an the previous proposition.*

Implicit vs Explicit Definability

Evert Willem Beth

Hidden implicit definitions must be **discovered**, because in theory combination they can propagate information from one theory to the other!

"Hidden implicit facts always need to be made explicit"

[Again, not a citation, but a good synthesis]

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.
- **Convexity** hypothesis cannot be eliminated. Counterexample:

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.
- **Convexity** hypothesis cannot be eliminated. Counterexample:
  - $T_1 \cup T_2$, where

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.
- **Convexity** hypothesis cannot be eliminated. Counterexample:
  - $T_1 \cup T_2$, where
    - $T_1 := \mathcal{IDL}$ (integer difference logic, which are integer numbers with successor and predecessor, 0 and the strict order $<$)

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.
- **Convexity** hypothesis cannot be eliminated. Counterexample:
  - $T_1 \cup T_2$, where
    - $T_1 := \mathcal{IDL}$ (integer difference logic, which are integer numbers with successor and predecessor, 0 and the strict order $<$)
    - $T_2 := \mathcal{EUF}(\Sigma_f)$, where Σ_f has only one unary fresh function symbol f (different from the symbols of T_1)

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.
- **Convexity** hypothesis cannot be eliminated. Counterexample:
  - $T_1 \cup T_2$, where
    - $T_1 := \mathcal{IDL}$ (integer difference logic, which are integer numbers with successor and predecessor, 0 and the strict order $<$)
    - $T_2 := \mathcal{EUF}(\Sigma_f)$, where Σ_f has only one unary fresh function symbol f (different from the symbols of T_1)
  - the formula $\exists e\,(0 < e \wedge e < x \wedge f(e) = 0)$ does **not** have a UI in $T_1 \cup T_2$.

How crucial are the hypotheses?

- **Equality interpolating** is a **necessary condition** for UI transfer: **minimal combinations** with uninterpreted symbols.
- **Convexity** hypothesis cannot be eliminated. Counterexample:
  - $T_1 \cup T_2$, where
    - $T_1 := \mathcal{IDL}$ (integer difference logic, which are integer numbers with successor and predecessor, 0 and the strict order $<$)
    - $T_2 := \mathcal{EUF}(\Sigma_f)$, where Σ_f has only one unary fresh function symbol f (different from the symbols of T_1)
  - the formula $\exists e\,(0 < e \wedge e < x \wedge f(e) = 0)$ does **not** have a UI in $T_1 \cup T_2$.
- **Stable Infiniteness** is already crucial for combined satisfiability (cf. the **Nelson-Oppen method**!).

Outline

Conclusions

- **Uniform Interpolants**: important tool in **formal verification**

Conclusions

- **Uniform Interpolants**: important tool in **formal verification**
  - Compute ***exact images***

Conclusions

- **Uniform Interpolants**: important tool in **formal verification**
  - Compute ***exact images***
  - 'Weak form' of QE that is **computationally tractable**

Conclusions

- **Uniform Interpolants**: important tool in **formal verification**
  - Compute ***exact images***
  - 'Weak form' of QE that is **computationally tractable**

- UIs have strict relationships with **model completions**: in these theories, UIs *eliminate quantifiers* in a proper sense.

Conclusions

- **Uniform Interpolants**: important tool in **formal verification**
  - Compute ***exact images***
  - 'Weak form' of QE that is **computationally tractable**
- UIs have strict relationships with **model completions**: in these theories, UIs *eliminate quantifiers* in a proper sense.
- For significant theories (DB with keys), UIs can be **computed using well-established techniques** like the Superposition Calculus.

Conclusions

- **Uniform Interpolants**: important tool in **formal verification**
  - Compute ***exact images***
  - 'Weak form' of QE that is **computationally tractable**
- UIs have strict relationships with **model completions**: in these theories, UIs *eliminate quantifiers* in a proper sense.
- For significant theories (DB with keys), UIs can be **computed using well-established techniques** like the Superposition Calculus.
- **Modular combined algorithms** for UI computation: applications to data-aware processes.

Future Directions

Several open directions:

- Using UIs for verification tasks beyond **safety** (e.g. *liveness*, *fairness*);

Future Directions

Several open directions:

- Using UIs for verification tasks beyond **safety** (e.g. *liveness*, *fairness*);
- Compute UIs using variants of Superposition Calculus in case of **sophisticated background theories**.

Future Directions

Several open directions:

- Using UIs for verification tasks beyond **safety** (e.g. *liveness*, *fairness*);
- Compute UIs using variants of Superposition Calculus in case of **sophisticated background theories**.
- **UI transfer** properties for proper **non-disjoint signatures combinations**.

Future Directions

Several open directions:

- Using UIs for verification tasks beyond **safety** (e.g. *liveness*, *fairness*);
- Compute UIs using variants of Superposition Calculus in case of **sophisticated background theories**.
- **UI transfer** properties for proper **non-disjoint signatures combinations**.
- **Uniform Interpolants** in SMT solvers

References I

R. Bruttomesso, S. Ghilardi, and S. Ranise.
Quantifier-free interpolation in combinations of equality interpolating theories.
ACM Trans. Comput. Log., 15(1):5:1–5:34, 2014.

D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, and A. Rivkin.
SMT-based verification of data-aware processes: a model-theoretic approach.
Mathematical Structures in Computer Science, 2020.

D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, and A. Rivkin.
Model completeness, uniform interpolants and superposition calculus.
Journal of Automated Reasoning, 2021.

D. Calvanese, S. Ghilardi, A. Gianola, M. Montali, and A. Rivkin.
Combination of uniform interpolants via Beth definability.
Journal of Automated Reasoning, 2022.

C.-C. Chang and J. H. Keisler.
Model Theory.
North-Holland Publishing Co., Amsterdam-London, third edition, 1990.

References II

W. Craig.

Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory.

Journal of Symbolic Logic, 22:269–285, 1957.

S. Gulwani and M. Musuvathi.

Cover algorithms and their combination.

In *Proc. of ESOP, Held as Part of ETAPS*, pages 193–207, 2008.

Ranjit Jhala and Rupak Majumdar.

Software model checking.

ACM Comput. Surv., 41(4):21:1–21:54, 2009.

L. Kovács and A. Voronkov.

Interpolation and symbol elimination.

In *Proc. of CADE*, pages 199–213, 2009.

References III

K. L. McMillan.

Interpolation and SAT-Based Model Checking.

In *Proceedings of CAV 2003*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003.

K. L. McMillan.

Lazy Abstraction with Interpolants.

In *Proceedings of CAV 2006*, volume 4144 of *LNCS*, pages 123–136. Springer, 2006.

A. Robinson.

Introduction to model theory and to the metamathematics of algebra.

Studies in logic and the foundations of mathematics. North-Holland, 1963.

G. Yorsh and M. Musuvathi.

A combination method for generating interpolants.

In *Proc. of CADE-20*, LNCS, pages 353–368. 2005.


Uniform
Interpolants
Model
Completions
Formal
Verification
Thanks for your attention!

Example of Computation of **constr-SC**

Input formula: $f(e, y_1) = y_2 \wedge f(e, y_3) = y_4$

Example of Computation of **constr-SC**

Input formula: $f(e, y_1) = y_2 \wedge f(e, y_3) = y_4$

Apply *Superposition Right* in root position, with:

- $l \equiv f(e, y_1)$,
- $r \equiv y_2$,
- $s \equiv f(e, y_3)$,
- $t \equiv y_4$

C and D are empty. We get $E(s, l) \equiv \{y_1 = y_3\}$

Example of Computation of **constr-SC**

Input formula: $f(e, y_1) = y_2 \wedge f(e, y_3) = y_4$

Apply *Superposition Right* in root position, with:

- $l \equiv f(e, y_1)$,
- $r \equiv y_2$,
- $s \equiv f(e, y_3)$,
- $t \equiv y_4$

C and D are empty. We get $E(s, l) \equiv \{y_1 = y_3\}$

So, we produce the clause $y_1 = y_3 || \{y_2 = y_4\}$, i.e., $y_1 = y_3 \rightarrow y_2 = y_4$.

.

Example of Computation of **constr-SC**

Input formula: $f(e, y_1) = y_2 \wedge f(e, y_3) = y_4$

Apply *Superposition Right* in root position, with:

- $l \equiv f(e, y_1)$,
- $r \equiv y_2$,
- $s \equiv f(e, y_3)$,
- $t \equiv y_4$

C and D are empty. We get $E(s, l) \equiv \{y_1 = y_3\}$

So, we produce the clause $y_1 = y_3 || \{y_2 = y_4\}$, i.e., $y_1 = y_3 \rightarrow y_2 = y_4$.

$y_1 = y_3 \rightarrow y_2 = y_4$ is the **UI** of the **input formula**.

Combined Algorithm: an Example

Let T_1 be $\mathcal{EUF}(\Sigma)$ and T_2 be linear real arithmetic.

Combined Algorithm: an Example

Let T_1 be $\mathcal{EUF}(\Sigma)$ and T_2 be linear real arithmetic.

Covers are computed in real arithmetic by **quantifier elimination**, whereas for $\mathcal{EUF}(\Sigma)$ one can apply the **superposition-based algorithm** from [CGG$^+$21].

Combined Algorithm: an Example

Let T_1 be $\mathcal{EUF}(\Sigma)$ and T_2 be linear real arithmetic.

Covers are computed in real arithmetic by **quantifier elimination**, whereas for $\mathcal{EUF}(\Sigma)$ one can apply the **superposition-based algorithm** from [CGG$^+$21].

Consider the formula:

$$\exists e_1 \cdots \exists e_4 \begin{pmatrix} e_1 = f(x_1) \ \wedge \ e_2 = f(x_2) \ \wedge \\ \wedge \ f(e_3) = e_3 \ \wedge \ \ f(e_4) = x_1 \ \wedge \\ \wedge \ x_1 + e_1 \leq e_3 \ \wedge \ e_3 \leq x_2 + e_2 \ \wedge \ e_4 = x_2 + e_3 \end{pmatrix}$$

Combined Algorithm: an Example

Let T_1 be $\mathcal{EUF}(\Sigma)$ and T_2 be linear real arithmetic.

Covers are computed in real arithmetic by **quantifier elimination**, whereas for $\mathcal{EUF}(\Sigma)$ one can apply the **superposition-based algorithm** from [CGG+21].

Consider the formula:

$$\exists e_1 \cdots \exists e_4 \begin{pmatrix} e_1 = f(x_1) \ \wedge \ e_2 = f(x_2) \ \wedge \\ \wedge \ f(e_3) = e_3 \ \wedge \ \ f(e_4) = x_1 \ \wedge \\ \wedge \ x_1 + e_1 \leq e_3 \ \wedge \ e_3 \leq x_2 + e_2 \ \wedge \ e_4 = x_2 + e_3 \end{pmatrix}$$

Applying the combined algorithm, we get:

$$\begin{aligned} &[x_2 = 0 \ \wedge \ f(x_1) = x_1 \ \wedge \ x_1 \leq 0 \ \wedge \ x_1 \leq f(0)] \ \vee \\ &\vee \ [x_1 + f(x_1) < x_2 + f(x_2) \ \wedge \ x_2 \neq 0] \ \vee \\ &\vee \begin{bmatrix} x_2 \neq 0 \ \wedge \ x_1 + f(x_1) = x_2 + f(x_2) \ \wedge \ f(2x_2 + f(x_2)) = x_1 \ \wedge \\ \wedge \, f(x_1 + f(x_1)) = x_1 + f(x_1) \end{bmatrix} \end{aligned}$$

Artifact-Centric Systems

Artifact-Centric Systems: process-centric paradigm + data
(**artifact** = *lifecycle* + *information model*).

Artifact-Centric Systems

Artifact-Centric Systems: process-centric paradigm + data (**artifact** = *lifecycle* + *information model*).
They **can** be formalized using three components:

Artifact-Centric Systems

Artifact-Centric Systems: process-centric paradigm + data (**artifact** = *lifecycle* + *information model*).
They **can** be formalized using three components:

- *a read-only database (DB)*;

Artifact-Centric Systems

Artifact-Centric Systems: process-centric paradigm + data (**artifact** = *lifecycle* + *information model*).
They **can** be formalized using three components:

- *a read-only database (DB)*;
- *an artifact working memory (e.g., artifact variables + artifact relations)*;

Artifact-Centric Systems

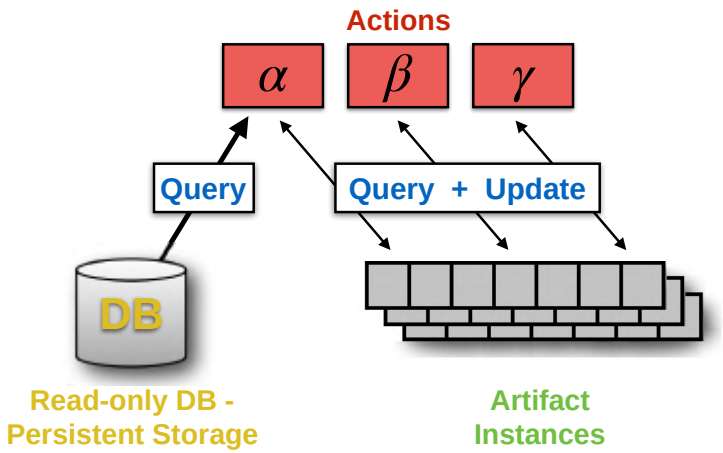
Artifact-Centric Systems: process-centric paradigm + data (**artifact** = *lifecycle* + *information model*).
They **can** be formalized using three components:

- *a read-only database (DB)*;
- *an artifact working memory (e.g., artifact variables + artifact relations)*;
- *actions* (also called *services*).

Artifact-Centric Systems

Artifact-Centric Systems: process-centric paradigm + data (**artifact** = *lifecycle* + *information model*).
They **can** be formalized using three components:

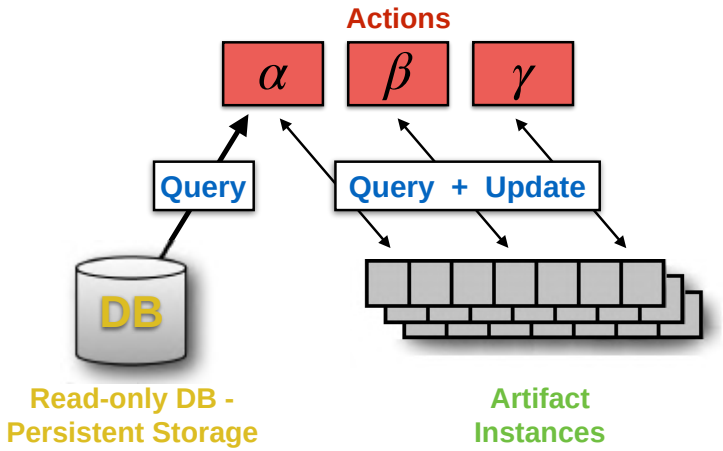
- *a read-only database (DB)*;
- *an artifact working memory (e.g., artifact variables + artifact relations)*;
- *actions* (also called *services*).



Artifact-Centric Systems

Artifact-Centric Systems: process-centric paradigm + data (**artifact** = *lifecycle* + *information model*).
They **can** be formalized using three components:

- *a read-only database (DB)*;
- *an artifact working memory (e.g., artifact variables + artifact relations)*;
- *actions* (also called *services*).



Artifact-Centric Systems $\Longrightarrow$ Array-based Systems $\Longrightarrow$
SMT-based tool **Model Checker Modulo Theories (MCMT)**

DB schemas

DB schemas: *read-only DB* of Artifact-Centric Systems, incorporating primary keys and foreign keys dependencies

DB schemas

DB schemas: *read-only DB* of Artifact-Centric Systems, incorporating primary keys and foreign keys dependencies

Definition

A DB schema is a pair (Σ, T), where:

- Σ is a *DB signature*, that is, a finite multi-sorted signature with equality, unary functions, n-ary relations and constants;
- T is a *DB theory*, that is, a set of universal Σ-sentences.

DB schemas

DB schemas: *read-only DB* of Artifact-Centric Systems, incorporating primary keys and foreign keys dependencies

Definition

A DB schema is a pair (Σ, T), where:

- Σ is a *DB signature*, that is, a finite multi-sorted signature with equality, unary functions, n-ary relations and constants;
- T is a *DB theory*, that is, a set of universal Σ-sentences.

In a *basic DB schema*, T is empty. $G(\Sigma)$: characteristic graph capturing the dependencies induced by functions over sorts.

DB schemas

DB schemas: *read-only DB* of Artifact-Centric Systems, incorporating primary keys and foreign keys dependencies

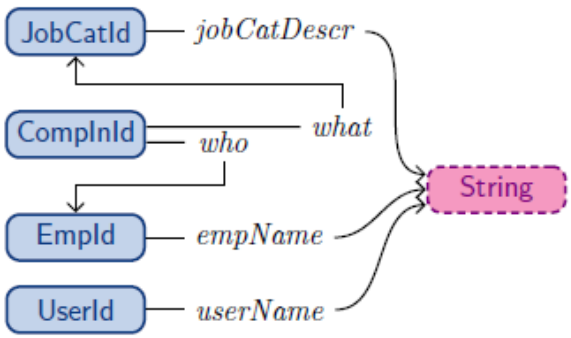
Definition

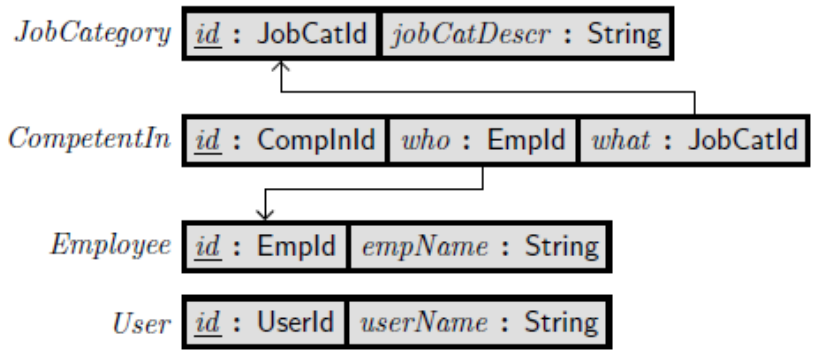
A DB schema is a pair (Σ, T), where:

- Σ is a *DB signature*, that is, a finite multi-sorted signature with equality, unary functions, n-ary relations and constants;
- T is a *DB theory*, that is, a set of universal Σ-sentences.

In a *basic DB schema*, T is empty. $G(\Sigma)$: characteristic graph capturing the dependencies induced by functions over sorts.

Example:





Array-based Artifact-Centric Systems: a simplified version

A **SAS** (**Simple Artifact Systems**) is a tuple $\mathcal{S} = \langle \Sigma, T, \underline{x}, \iota(\underline{x}), \tau(\underline{x}, \underline{x}') \rangle$, where:

- (Σ, T) is a DB schema;
- $\underline{x}$ are individual FO variables representing the current state;
- ι is a Σ-formula representing the initialization;
- $\tau(\underline{x}, \underline{x}')$ is a Σ-formula representing the transitions from the **current state** $\underline{x}$ to the **new state** $\underline{x}'$.

Array-based Artifact-Centric Systems: a simplified version

A **SAS** (**Simple Artifact Systems**) is a tuple $\mathcal{S} = \langle \Sigma, T, \underline{x}, \iota(\underline{x}), \tau(\underline{x}, \underline{x}') \rangle$, where:

- (Σ, T) is a DB schema;
- $\underline{x}$ are individual FO variables representing the current state;
- ι is a Σ-formula representing the initialization;
- $\tau(\underline{x}, \underline{x}')$ is a Σ-formula representing the transitions from the **current state** $\underline{x}$ to the **new state** $\underline{x}'$.

Array-based Artifact-Centric Systems: a simplified version

A **SAS** (**Simple Artifact Systems**) is a tuple $\mathcal{S} = \langle \Sigma, T, \underline{x}, \iota(\underline{x}), \tau(\underline{x}, \underline{x}') \rangle$, where:

- (Σ, T) is a DB schema;
- $\underline{x}$ are individual FO variables representing the current state;
- ι is a Σ-formula representing the initialization;
- $\tau(\underline{x}, \underline{x}')$ is a Σ-formula representing the transitions from the **current state** $\underline{x}$ to the **new state** $\underline{x}'$.

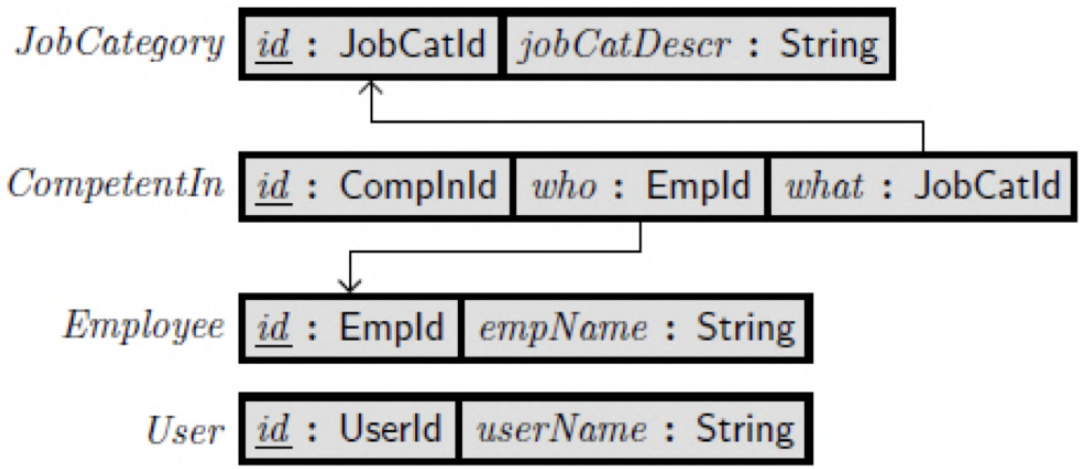
Individual variables change their value over the time, according to the *transitions* formula!

A simple example

Job Hiring Process:

$$\iota := (\mathsf{Applicant} = undef \wedge \mathsf{JobPos} = undef)$$

$$\tau := \exists \mathsf{UserID}, \mathsf{JobID} \left(\begin{array}{l} \mathsf{UserID} \neq undef \wedge \mathsf{JobID} \neq undef \wedge \mathsf{Applicant} = undef \wedge \\ \mathsf{JobPos} = undef \wedge \mathsf{Applicant}' := \mathsf{UserID} \wedge \mathsf{JobPos}' := \mathsf{JobID} \end{array} \right)$$

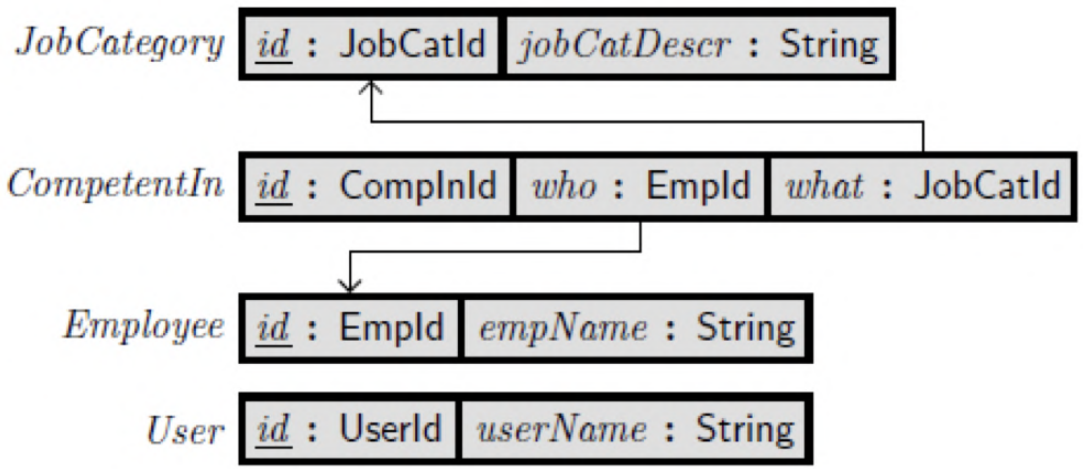


A simple example

Job Hiring Process:

$$\iota := (\mathsf{Applicant} = undef \wedge \mathsf{JobPos} = undef)$$

$$\tau := \exists \mathsf{UserID}, \mathsf{JobID} \left(\begin{array}{l} \mathsf{UserID} \neq undef \wedge \mathsf{JobID} \neq undef \wedge \mathsf{Applicant} = undef \wedge \\ \mathsf{JobPos} = undef \wedge \mathsf{Applicant}' := \mathsf{UserID} \wedge \mathsf{JobPos}' := \mathsf{JobID} \end{array} \right)$$

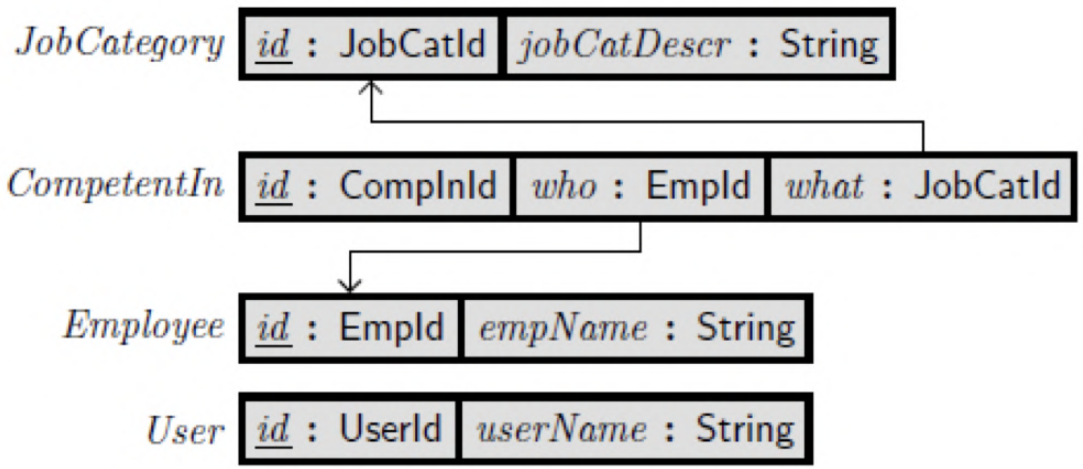


A simple example

Job Hiring Process:

$$\iota := (\mathsf{Applicant} = undef \wedge \mathsf{JobPos} = undef)$$

$$\tau := \exists \mathsf{UserID}, \mathsf{JobID} \left(\begin{array}{l} \mathsf{UserID} \neq undef \wedge \mathsf{JobID} \neq undef \wedge \mathsf{Applicant} = undef \wedge \\ \mathsf{JobPos} = undef \wedge \mathsf{Applicant}' := \mathsf{UserID} \wedge \mathsf{JobPos}' := \mathsf{JobID} \end{array} \right)$$



Verification of safety in a SAS $\mathcal{S}$

A *safety* formula for $\mathcal{S}$: *generic* quantifier-free formula $\upsilon(\underline{x}) \implies$ *undesired states* of $\mathcal{S}$.

Verification of safety in a SAS $\mathcal{S}$

A *safety* formula for $\mathcal{S}$: *generic* quantifier-free formula $\upsilon(\underline{x}) \implies$ *undesired states* of $\mathcal{S}$.

$\mathcal{S}$ is safe wrt υ **iff** in no model $\mathcal{M}$ of (Σ, T), for no $k \geq 0$ and for no assignment in $\mathcal{M}$ to $\underline{x}^0, \ldots, \underline{x}^k$ (1) is true ($\underline{x}^i$ are renamed copies of $\underline{x}$):

$$\iota(\underline{x}^0) \wedge \tau(\underline{x}^0, \underline{x}^1) \wedge \cdots \wedge \tau(\underline{x}^{k-1}, \underline{x}^k) \wedge \upsilon(\underline{x}^k) \tag{1}$$

Verification of safety in a SAS $\mathcal{S}$

A *safety* formula for $\mathcal{S}$: *generic* quantifier-free formula $\upsilon(\underline{x}) \implies$ *undesired states* of $\mathcal{S}$.

$\mathcal{S}$ is safe wrt υ **iff** in no model $\mathcal{M}$ of (Σ, T), for no $k \geq 0$ and for no assignment in $\mathcal{M}$ to $\underline{x}^0, \ldots, \underline{x}^k$ (1) is true ($\underline{x}^i$ are renamed copies of $\underline{x}$):

$$\iota(\underline{x}^0) \wedge \tau(\underline{x}^0, \underline{x}^1) \wedge \cdots \wedge \tau(\underline{x}^{k-1}, \underline{x}^k) \wedge \upsilon(\underline{x}^k) \tag{1}$$

Safety problem for $\mathcal{S}$: *given υ, decide if $\mathcal{S}$ is safe wrt υ.*

Verification of safety in a SAS $\mathcal{S}$

A *safety* formula for $\mathcal{S}$: *generic* quantifier-free formula $\upsilon(\underline{x}) \implies$ *undesired states* of $\mathcal{S}$.

$\mathcal{S}$ is safe wrt υ **iff** in no model $\mathcal{M}$ of (Σ, T), for no $k \geq 0$ and for no assignment in $\mathcal{M}$ to $\underline{x}^0, \ldots, \underline{x}^k$ (1) is true ($\underline{x}^i$ are renamed copies of $\underline{x}$):

$$\iota(\underline{x}^0) \wedge \tau(\underline{x}^0, \underline{x}^1) \wedge \cdots \wedge \tau(\underline{x}^{k-1}, \underline{x}^k) \wedge \upsilon(\underline{x}^k) \tag{1}$$

Safety problem for $\mathcal{S}$: *given υ, decide if $\mathcal{S}$ is safe wrt υ.*

Theorem (Soundness and Completeness)

Backward search is effective, correct and complete (the last one w.r.t. detecting unsafety) for the safety problems for SASs. If $G(\Sigma)$ is acyclic, backward search always terminates and it is a full decision procedure.