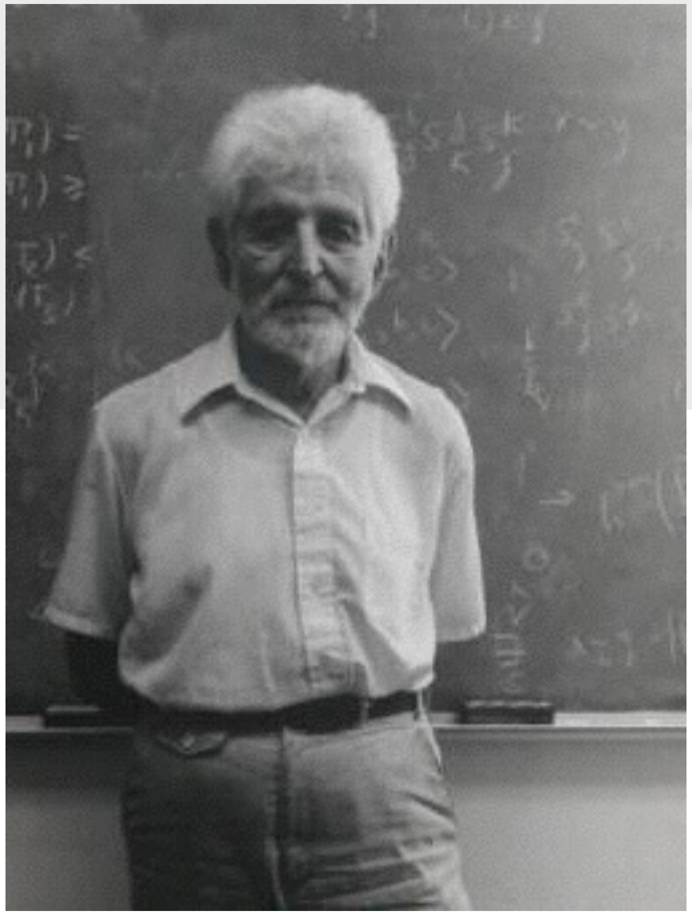
Craig Interpolation Theorems and Database Applications

Balder ten Cate
LogicBlox & UC Santa Cruz

November 7, 2014, UC Berkeley - Logic Colloquium

Craig Interpolation



- **William Craig (1957)**: For all first-order formulas ϕ, ψ, if $\phi \vDash \psi$, then there is a first-order formula χ with $\mathrm{Voc}(\chi) \subseteq \mathrm{Voc}(\phi) \cap \mathrm{Voc}(\psi)$ and $\phi \vDash \chi \vDash \psi$.
  Moreover the formula χ in question can effectively constructed from a proof of $\phi \vDash \psi$.

- Various extensions and variations (e.g., Lyndon interpolation, many-sorted interpolation, Otto interpolation).

- Van Benthem (2008): "*Craig's Theorem is about the last significant property of first-order logic that has come to light*."

Outline

1. An interpolation theorem for FO formulas with relational access restrictions
2. Effective interpolation for the guarded negation fragment of FO

Relational Access Restrictions

- A **database** is a (finite) relational structure over some schema $S = \{R_1, \ldots, R_n\}$
- **Relational access restrictions**: restrictions on the way we can access the relations $R_1, \ldots, R_n$.

First Example: View-Based Query Reformulation

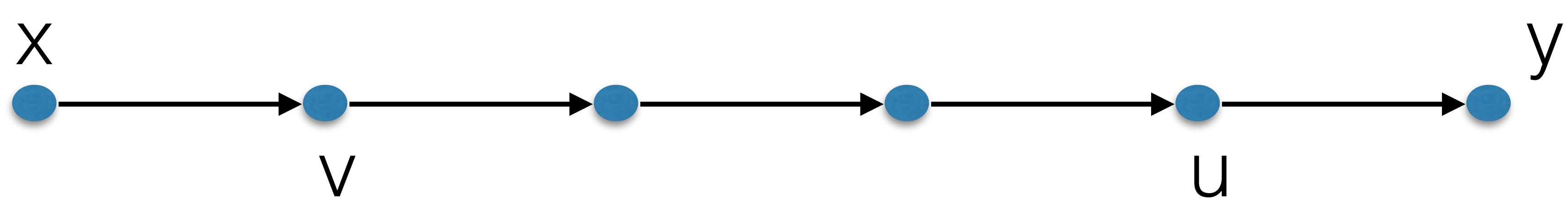
- **Road network database**: Road(x,y)
- **Views**:
  - $V_2(x,y)$ = "∃ path of length 2 from x to y" = $\exists u\ \mathrm{Road}(x,u) \wedge \mathrm{Road}(u,y)$
  - $V_3(x,y)$ = "∃ path of length 3 from x to y" = $\exists u,v\ \mathrm{Road}(x,u) \wedge \mathrm{Road}(u,v) \wedge \mathrm{Road}(v,y)$
  - …
- **Observation**: V_4 can be expressed in terms of V_2.
- **Puzzle** (Afrati'07): can V_5 be expressed (in FO logic) in terms of V_3 and V_4?

Solution to the puzzle

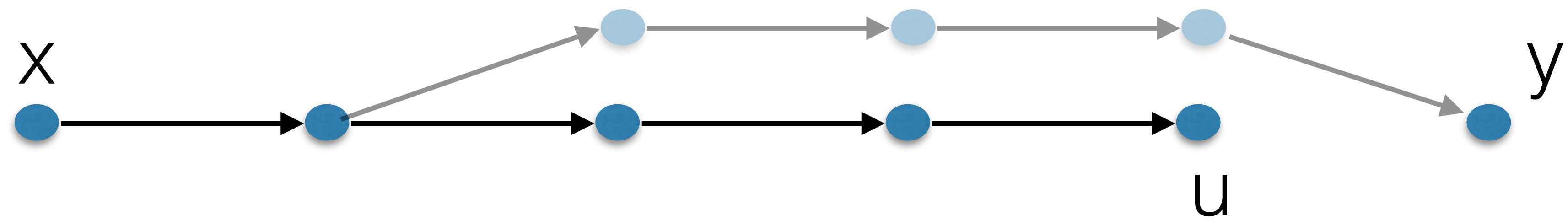
$$V_5(x,y) \Leftrightarrow \exists u \, (\, V_4(x,u) \wedge \forall v \, (\, V_3(v,u) \rightarrow V_4(v,y) \,) \,)$$

Proof:

[⟹]



[⟸]



Why this Example is Important

- A **conjunctive query** (CQ) is a FO formula built up using only $\wedge$, $\exists$.
  - Conjunctive queries are the most common type of database queries.
  - Every positive-existential FO formula is equivalent to a union of CQs.

- Remarkable fact:
  - V_3, V_4 and V_5 are all defined by CQs over the base relation (Road).
  - V_5 is definable in terms of V_3 and V_4 but not by means of a CQ.

Classic Results

Querying using views has been around since the 1980s. E.g.,

- **Theorem** (Levy Mendelzon Sagiv Srivastava ’95): there is an effective procedure to decide whether a conjunctive query is rewritable as a conjunctive query over a given set of conjunctive views.

- **Open problem** (Nash, Segoufin, Vianu ‘10): is there an effective procedure to decide if a conjunctive query is answerable on the basis of a set of conjunctive views (a.k.a., is determined by the views)? if so, in what language can we express the rewriting?

NB: The Beth definability theorem (1953) tell us that, if a FO query is answerable on the basis of a set of FO views, then, it has a FO rewriting.

Access Restrictions

- View-Based Query Reformulation:
  - *Can I reformulate Q as a query using only $V_1, \ldots, V_n$?*
  - *In other words, is Q equivalent to a query that only uses the symbols V1, …, Vn (relative to the theory consisting of the view definitions)?*
- Query Reformulation w.r.t. Access Methods (more refined):
  - *Can I find a plan to evaluate Q using only allowed access methods (possibly relative to some theory)?*
  - First theory work by Chang and Li '01, followed by work of Nash, Ludaescher, Deutsch, …

Access Methods

- **Access method**: a pair (R,X) where R is an n-ary relation and $X \subseteq \{1, \ldots, n\}$ is a set of "input positions"
  - *Relation R can be accessed if specific values are provided for the positions in X.*
- **Examples**:
  - (Yellowpages(name,city,address,phone#), {1,2})
  - (R,∅) means **free (unrestricted) access** to R.
  - (R,{1, …, n}) means only **membership tests** for specific tuples.
- There may be any number of access methods for a given relation. The allowed access methods for a relation can be assumed to be an upwards closed set.

Access Methods "Used" by a Formula

BindPatt(ϕ) is the set of access methods "used" by ϕ.

$$
\begin{array}{lcl}
\mathsf{BindPatt}(\top) = \mathsf{BindPatt}(x=y) & = & \emptyset \\
\mathsf{BindPatt}(R(t_1,\ldots,t_n)) & = & \{(R,\{1,\ldots,n\})\} \\
\mathsf{BindPatt}(\neg\varphi) & = & \mathsf{BindPatt}(\varphi) \\
\mathsf{BindPatt}(\varphi \wedge \psi) & = & \mathsf{BindPatt}(\varphi) \cup \mathsf{BindPatt}(\psi) \\
\mathsf{BindPatt}(\varphi \vee \psi) & = & \mathsf{BindPatt}(\varphi) \cup \mathsf{BindPatt}(\psi) \\
\mathsf{BindPatt}(\exists\vec{x}(R(t_1,\ldots,t_n) \wedge \varphi)) & = & \mathsf{BindPatt}(\varphi) \cup \{(R,\{i \mid t_i \notin \vec{x}\})\} \\
\mathsf{BindPatt}(\forall\vec{x}(R(t_1,\ldots,t_n) \to \varphi)) & = & \mathsf{BindPatt}(\varphi) \cup \{(R,\{i \mid t_i \notin \vec{x}\})\}
\end{array}
$$

- For example BindPatt($\forall y(Rxy \to Sxy)$) = { (R,{1}), (S,{1,2}) }
- A FO formula ϕ is executable if BindPatt(ϕ) consists of allowed access methods.
- **Fact**: Each executable FO formula admits a query plan, and, conversely, every formula that admits a query plan is equivalent to an executable FO formula.
    - Query plan = sequence of allowed accesses and / or relational algebra operations.

Motivation

- Query Reformulation w.r.t. Access Methods (again):
  - *Can I find a plan to evaluate Q using only allowed access methods (possibly relative to some theory)?*
- **Example**: In the road network example, $V_5(x,y)$ admits a first-order plan using only the access methods $(V_2,\emptyset)$ and $(V_3,\{1,2\})$.
- **Motivation**:
  - Answering queries using data behind webforms.
  - Query optimization (*if a relation R(x,y) is stored in order sorted on x, access method (R,{2}) is much more costly than access method (R,{1}).*)
  - …

- View-Based Query Reformulation:
  - *Can I reformulate Q as a query using only the views $V_1, \ldots, V_n$?*
- Query Reformulation w.r.t. Access Methods (more refined):
  - *Can I find a plan to evaluate Q using only allowed access methods (possibly relative to some theory)?*

- Enter Craig interpolation

View-Based Query Reformulation: Key concepts

- **Determinacy**: V_4 is determined by (or answerable from) V_2.
  - Whenever two structures, satisfying the view-definition theory, assign the same denotation to V_2, then they also assign the same denotation to V_4.
- **Query reformulations**: V_4 can be reformulated as a query over V_2 in FO
  - There is a FO formula that uses only V_2 and that is equivalent to V_4 (relative to the view definition theory).

V_4 is implicitly defined in terms of V_2

E.W.Beth

J. Groenendijk and M. Stokhof

$?xy.V_2(x,y) \vDash ?xy.V_4(x,y)$

View-Based Query Reformulation

- Given:
  - Base relations $R_1 \ldots R_n$,
  - View names $V_1 \ldots V_m$
  - View definition theory: $T = \{ \forall \mathbf{x}(V_1(\mathbf{x}) \leftrightarrow \psi_1(\mathbf{x})), \ldots \}$
  - Query Q over the base relations

- The following are equivalent:

1. Q is determined by $V_1 \ldots V_m$ (w.r.t. the theory T).
2. a certain FO implication $\theta_{T,Q}$ is valid
3. Q can be reformulated as a FO query over $V_1 \ldots V_m$. In fact, every Craig interpolant of $\theta_{T,Q}$ is such a reformulation.

What is going on?

- *From a proof of determinacy we are obtaining an actual reformulation.*
- This way of using interpolation to get explicit definitions from implicit ones goes right back to Craig's work.
- Same technique works for arbitrary theories T (not only view definitions).
- In principle this gives a method for finding query reformulations (but FO theorem proving is difficult).

- Can we do the same for the case with access methods?
- **Answer**: yes, using a suitable generalization of Craig interpolation.

Access Interpolation

- **Access interpolation theorem** (Benedikt, tC, Tsamoura, 2014): for all FO formulas ϕ, ψ, if $\phi \vDash \psi$, then there is a FO formula χ with BindPatt(χ) $\subseteq$ BindPatt(ϕ) $\cap$ BindPatt(ψ) and $\phi \vDash \chi \vDash \psi$. Moreover the formula χ in question can effectively constructed from a proof of $\phi \vDash \psi$ (in a suitable proof system).

- Can be further refined by distinguishing positive/negative uses of binding patterns.

- Generalizes many existing interpolation theorems (Lyndon, many-sorted interpolation, Otto interpolation).

- Gives rise to a way of testing “access-determinacy” and the existence of reformulations w.r.t. given access methods, as well as a method for finding such reformulations.

Mathematical Logic

- In set theory, a Δ_0-formula is a formula that only uses access method $(\in, \{2\})$
- In bounded arithmetic, we study formulas that only use access method $(\leq, \{2\})$.
- The access interpolation theorem generalizes an interpolation theorem for "$\leq$-persistent" formulas by Feferman (1967).
- Closely related: an interpolation theorem for the bounded fragment (equivalently, hybrid logic) by Areces, Blackburn and Marx (2001).

Summary

Querying under Access Restrictions

1. View-based query reformulation (simply restricting to a subset of the signature)

   This is the setting of the (projective) Beth theorem. We look for a proof of implicit definability (“determinacy”) and, from it, compute an explicit definition (“query reformulation”) using Craig interpolation.

2. Query reformulations given access methods (more refined)

   Same general technique applies, using a suitable adaptation of Craig interpolation: access interpolation.

Three Important Subtleties

1. Databases are finite structures. But Craig interpolation for first-order logic fails in the finite (and so does access interpolation).

2. For practical applications, we need effective algorithms. But first-order logic is undecidable (we cannot effectively decide if the implication $\theta_{T,Q}$ is valid).

3. For practical applications, we don't want just any query reformulation, we want one of low cost.

Solutions

- The solution for 1 and 2 is to **move to a fragment** of first-order logic that is (a) **decidable** and that has (b) the **finite model property**, and (c) Craig interpolation, while still being sufficiently expressive.
  - the guarded-negation fragment.

- The solution for 3 is to take into account a cost function.

Cost-sensitive Query Reformulation

- Every database management system has a cost-estimate function for query plans (expected execution time).
- We are looking for a proof of $\theta_{T,Q}$ such that the interpolant obtained from it constitutes a plan that has a low cost.
- Strategy: explore the space of possible proofs *guided by a (monotone) plan cost function.*
- Ongoing research (Michael Benedikt and others at Oxford, using the LogicBlox data management system).
  - cf. also [Benedikt, Leblay, Tsamoura PVLDB, 2014]

- Part 2: Guarded negation

  based on joint work with Luc Segoufin, Vince Barany and Martin Otto, Michael Benedikt, Michael vanden Boom (STACS 11, ICALP 11, VLDB 2012, MFCS 2013, LICS 2014)

Theme

- **Theme**: decidable fragments by restricting the use of negation.
  - Unary negation: allow only $\neg\phi(x)$ [tC & Segoufin STACS 11]
  - Guarded negation: allow also $G(\mathbf{x}) \wedge \neg\phi(\mathbf{x})$ [Barany, tC & Segoufin ICALP 11]

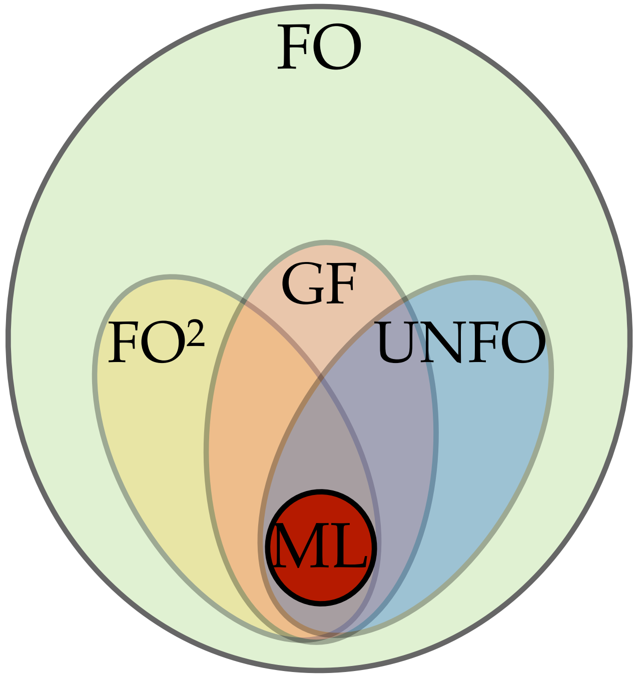
- Orthogonal to previous ways of “taming” FO:
  - We make no restriction on the number of variables
  - We make no restriction on quantifier alternation.
  - We allow unguarded quantification.

Modal logic

- (Basic) modal logic: a small fragment of FO
  - Signature: $\{R, P_1, \ldots, P_n\}$
  - shorthand: $\Diamond\phi$
  - Formulas: $\phi(x) := P_i(x) \mid \phi(x) \wedge \psi(x) \mid \neg\phi(x) \mid \exists y(Rxy \wedge \phi(y))$
- Very well behaved (decidable for satisfiability, has Craig interpolation, ..)
  - many extensions, such as the modal mu-calculus, are decidable too.
- "Why is modal logic so robustly decidable?" (Vardi '96)
  - tree model property,
  - translation into MSO (tree automata),
  - finite model property.

Why Modal Logic is so Robustly Decidable

- "Syntactic explanations":
  - Modal formulas only need two variables (FO^2) [Graedel-Kolaitis-Vardi 1997]
  - Modal formulas only use guarded quantification patterns (GFO) [Andreka-van Benthem-Nemeti 1998, Graedel 1999]
  - Modal formulas only use unary negation (UNFO) [tC-Segoufin 2011]



Guarded Fragment

(Andreka, van Benthem, Nemeti 1998)

- All quantification must be guarded.

  $\phi ::= R(x_1 \ldots x_n) \mid x{=}y \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{\exists y}.G(\mathbf{x,y,z}) \wedge \phi(\mathbf{x,y})$

- GF has become an extremely successful and well studied fragment of FO.
- Inherits all the good properties of modal logic (robust decidability, finite model property, …)
- Except Craig interpolation (cf. Hoogland and Marx 2002).

Unary Negation

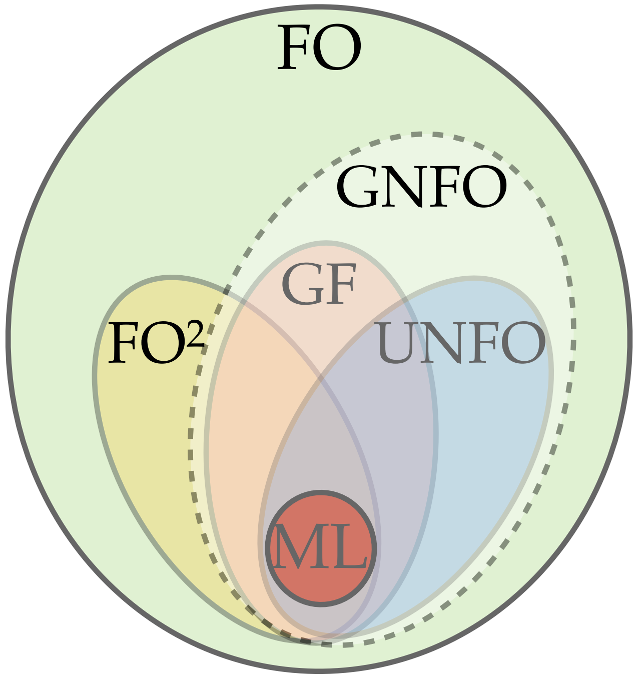
- Unary Negation FO (UNFO):
  - ϕ ::= $R(\mathbf{x})$ | $x = y$ | $\phi \wedge \phi$ | $\phi \vee \phi$ | $\exists x \phi$ | $\neg\phi(x)$
  - Only allow negation of formulas in one free variable.
  - NB. The universal quantifier is treated as a defined connective.
- Fixed-Point Extension (UNFP):
  - ϕ ::= ... | $[\mathrm{LFP}_{Z,z}\, \phi(Z,\mathbf{Y},z)](x)$ (where Z occurs only positively in ϕ)
- UNFO and UNFP generalizes many existing logics:
  - Modal logic, modal mu-calculus, various description logics,
  - Unions of conjunctive queries, monadic Datalog,
  - CTL*(X), Core XPath

UNFO and UNFP

- Good model theory:
  - UNFO & UNFP have the tree-like model property (bounded tree-width)
  - UNFO has the finite model property
  - UNFO has Craig interpolation
  - UNFO can be characterized in terms of UN-bisimulation invariance.
- Decidable for satisfiability
  - 2ExpTime-complete, both on arbitrary structures and in the finite (via [Bojanczyk '02])
- Model checking:
  - UNFO model checking: $P^{NP[\log^2]}$ - complete (via [Schnoebelen '03])
  - UNFP model checking: in $NP^{NP} \cap coNP^{NP}$ and P^{NP}-hard.

Unary Negation vs Guardedness

- What do unary negation fragments have that guarded fragments don't?
  - Unrestricted existential quantification is allowed. Can express arbitrary Unions of Conjunctive Queries and monadic Datalog programs.
  - UNFO has Craig interpolation (which fails for GFO)
- A common generalization: guarded negation [Barany-tC-Segoufin '11].
  - All results for unary negation have been lifted to guarded negation.



Guarded Negation

- Guarded Fragment (GFO):

(

  - $\phi ::= R(\mathbf{x}) \mid x = y \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \exists \mathbf{x}\, G(\mathbf{xyz}) \wedge \phi(\mathbf{xy}) \mid \exists x\, \phi(x)$
  - Unrestricted use of negation; restricted use of quantification.

)

- Guarded Negation FO (GNFO):
  - $\phi ::= R(\mathbf{x}) \mid x{=}y \mid \exists x \phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(x) \mid G(\mathbf{xy}) \wedge \neg\phi(\mathbf{x})$
  - Restricted use of negation; unrestricted use of existential quantification.
- Fixed-point Extension (GNFP):
  - $\phi ::= \ \ldots \mid \mu_{Z,\mathbf{z}}[\ \mathrm{guarded}_\sigma(\mathbf{z}) \wedge \phi(\mathbf{Y}, Z, \mathbf{z})\ ](\mathbf{x})$ (where Z occurs only positively in ϕ)
  - (greatest fixed points can be expressed as dual)
- **Fact**: Every GFO / GFP sentence is equivalent to a GNFO / GNFP sentence.

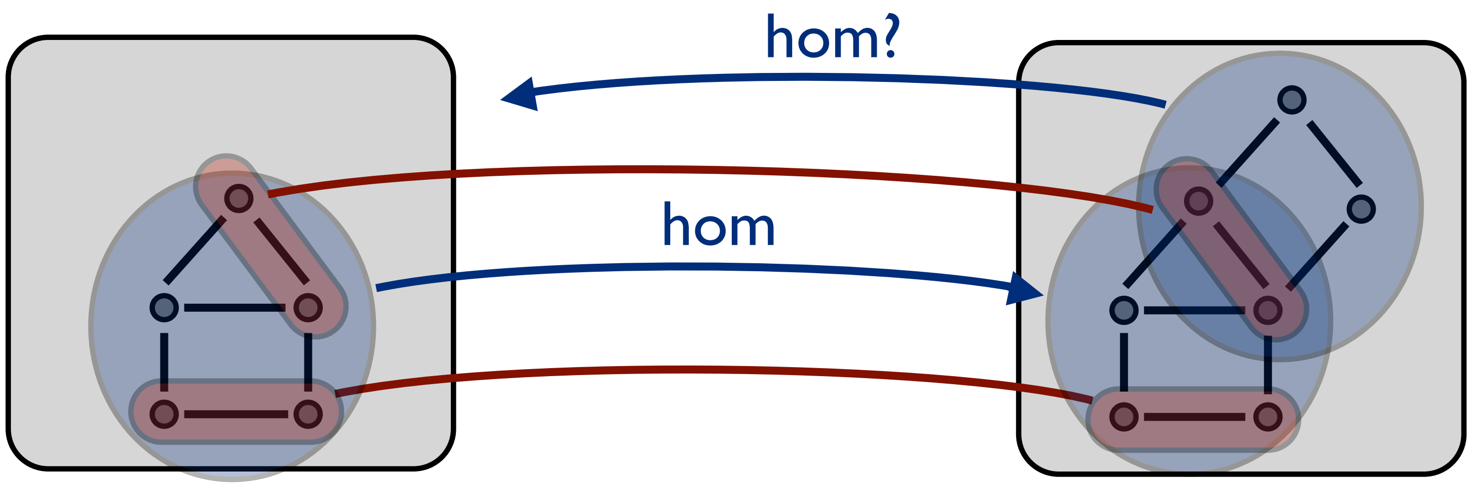
Normal form

- GN-Normal form for GNFO formulas:
  - $\phi ::= R(\mathbf{x}) \mid x{=}y \mid \exists x\phi \mid \phi\vee\phi \mid \phi\wedge\phi \mid \neg\phi(x) \mid G(\mathbf{xy})\wedge\neg\phi(\mathbf{x})$

    $q[\phi_1/U_1, \ldots, \phi_n/U_n]$
    (where q is a UCQ containing relations U1, ..., Un)
  - I.e., ϕ is built up from atoms using (i) UCQs, and (ii) guarded negation.
- GN-Normal form for GNFP formulas:
  - $\phi ::= R(\mathbf{x}) \mid x{=}y \mid q[\phi_1/U_1, \ldots, \phi_n/U_n] \mid \neg\phi(x) \mid G(\mathbf{xy})\wedge\neg\phi(\mathbf{x})$

    $\mid \mu_{Z,\mathbf{z}}[\ \text{guarded}_\sigma(\mathbf{z}) \wedge \phi(\mathbf{Y}, Z, \mathbf{z})\ ](\mathbf{x})$
  - I.e., ϕ is built up from atoms using (i) UCQs, (ii) guarded negation, and (iii) guarded LFPs.
- **Fact**: Every GNFO / GNFP formula is equivalent to one in GN-normal form.
  - GFO / GFP: as above, but only allow acyclic conjunctive queries.

GN-Bisimulation Game



- The GN-bisimulation game:
  - Positions: pairs of guarded sets (**a**,**b**)
  - Moves:
    - Spoiler picks a finite set X in one of the structures.
    - Duplicator responds with a partial homomorphism h from X to the other structure, s.t. h(**a**)=**b**.
    - Spoiler picks guarded subsets (**c**,**d**) in h.

Querying the Guarded Fragment

- Barany-Gottlob-Otto LICS 2010 ("Querying the guarded fragment"):
  - The following is 2ExpTime-complete and finitely controllable:

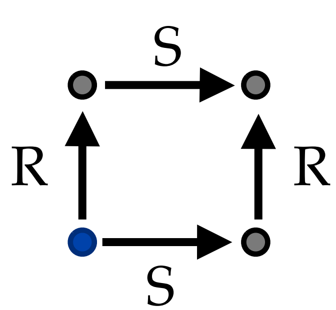
    Given a GFO-sentence ϕ and a (Boolean) UCQ q, test if $\phi \models q$.
- GNFO is a common generalization of GFO and UCQs.
  - The above question is equivalent to the (un)satisfiability of $\phi \wedge \neg q$.
  - Conversely, GNFO satisfiability reduces to querying the guarded fragment.
    - Replace all UCQs in the formula by fresh predicates, and "axiomatize" them (a la Scott normal form) using GFO sentences and negated CQs.
  - We show GNFP satisfiability is 2ExpTime-complete using the techniques from [Barany-Gottlob-Otto 2010] as well as [Barany-Bojanczyk 2011].

Decidability of GNFP Satisfiability

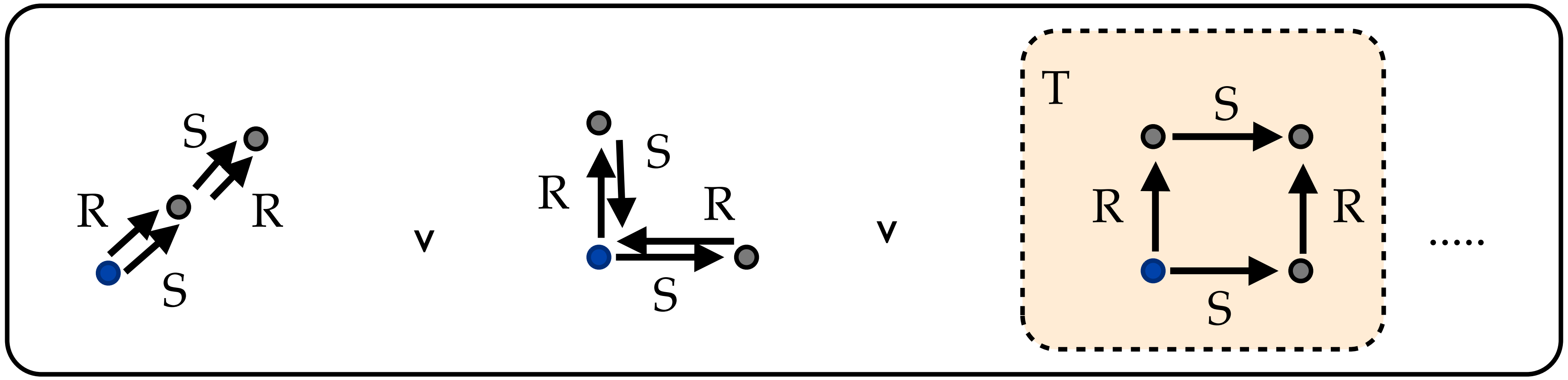
- **Thm**: (Finite) satisfiability of GNFP is 2ExpTime complete.
- Main ingredients of the proof:
  - Treeifications of cyclic conjunctive queries,
  - Rosati covers ([Barany-Gottlob-Otto 2010])
  - A reduction from GNFP to GFP.

Treeification

- **Treeification:**
  - If q is a CQ and τ is a schema, Λ^{τ}_{q} is the union of all minimal acyclic CQs over τ that imply q.



A (cyclic) CQ



Its treeification

  - The treeification Λ^{τ}_{q} can be expressed in GFO, and it “approximates” q:
    - By definition, Λ^{τ}_{q} implies q. Over tree-like structures, q and Λ^{τ}_{q} are equivalent.

Proof sketch

- **Theorem**. (Finite) satisfiability for GNFP reduces to (finite) satisfiability for GFP.
- Proof sketch:
  - Let ϕ be a GNFP sentence in GN-normal form.
  - Introduce a fresh relation symbol T of sufficiently large arity
  - Let $\eta(\phi)$ be the GFP formula obtained by replacing all CQs in ϕ by their $\tau \cup \{T\}$-treeification.
  - Every model of ϕ has an expansion satisfying ϕ' (interpret T as the total relation, and use the fact that τ-treeification includes the trivial T expansion).
  - Conversely, if $M \models \phi'$, then the (infinite) guarded unraveling $M^* \models \phi'$. Since M^* is tree-like, we have that $M^* \models \phi$.
  - In the finite case, instead of M^* we use a Rosati-cover (a finite approximation of M^*).

Summary

- **Guarded negation logic**:
  - Common generalization of guarded fragment and conjunctive queries.
  - Decidable (even when extended with fixed point operators).

- Satisfies the combination of properties that we were looking for:
  - Craig interpolation, decidable satisfiability problem, finite model property.

- **Corollary**: if a GNFO query is determined, given a collection of view definitions specified in GNFO, then there is a GNFO rewriting. Moreover, this holds also over finite structures, and is effective.