Guarded Negation

Balder ten Cate

based on joint work with Luc Segoufin, Vince Barany and Martin Otto
(STACS 11, ICALP 11, VLDB 2012)

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]

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: $\{R, P_1, \ldots, P_n\}$
  - Formulas: $\phi(x) := P_i(x) \mid \phi(x) \wedge \psi(x) \mid \neg\phi(x) \mid \exists y(Rxy \wedge \phi(y))$

shorthand: $\Diamond\phi$

Modal logic

- (Basic) modal logic: a small fragment of FO
  - Signature: $\{R, P_1, ..., P_n\}$
  - Formulas: $\phi(x) := P_i(x) \mid \phi(x) \wedge \psi(x) \mid \neg\phi(x) \mid \exists y(Rxy \wedge \phi(y))$

shorthand: $\Diamond\phi$

- Very well behaved (decidable for satisfiability, has Craig interpolation, ..)
  - many extensions, such as the modal mu-calculus, are decidable too.

Modal logic

- (Basic) modal logic: a small fragment of FO
  - Signature: $\{R, P_1, ..., P_n\}$
  - Formulas: $\phi(x) := P_i(x) \mid \phi(x) \wedge \psi(x) \mid \neg\phi(x) \mid \exists y(Rxy \wedge \phi(y))$

shorthand: $\Diamond\phi$

- 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

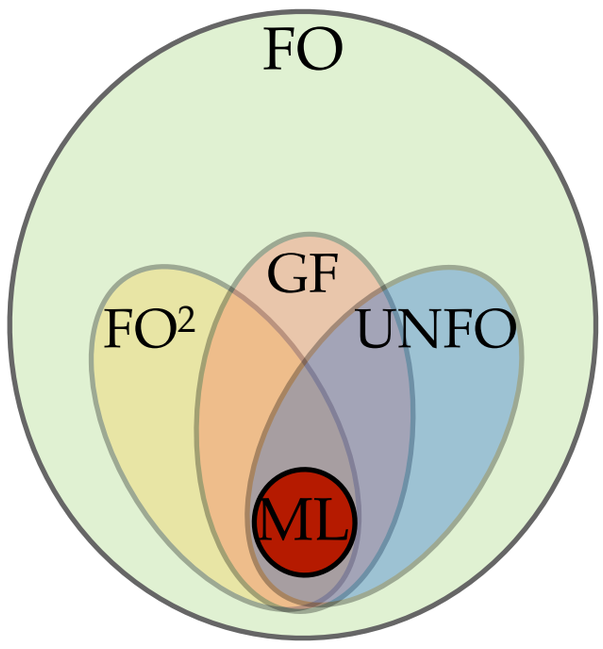
- "Syntactic explanations":
  - Modal formulas only need two variables (FO^2) [Graedel-Kolaitis-Vardi 1997]
  - Modal formulas only use guarded quantification patterns (GFO) [Andreka-van Benthem-Nemeti 1998, Graedel 1999]

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]

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]



Unary Negation

Unary Negation

- Unary Negation FO (UNFO):
  - $\phi ::= R(\mathbf{x}) \mid x = y \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x \phi \mid \neg\phi(x)$
  - Only allow negation of formulas in one free variable.
  - NB. The universal quantifier is treated as a defined connective.

Unary Negation

- Unary Negation FO (UNFO):
  - $\phi ::= R(\mathbf{x}) \mid x = y \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x \phi \mid \neg \phi(x)$
  - Only allow negation of formulas in one free variable.
  - NB. The universal quantifier is treated as a defined connective.

- Fixed-Point Extension (UNFP):
  - $\phi ::= \ldots \mid [\mathrm{LFP}_{Z,z}\, \phi(Z,\mathbf{Y},z)](x)$ (where Z occurs only positively in ϕ)
  - NB. No first-order parameters (i.e., only one free first-order variable).

Unary Negation

- Unary Negation FO (UNFO):
  - $\phi ::= R(\mathbf{x}) \mid x = y \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x \phi \mid \neg\phi(x)$
  - Only allow negation of formulas in one free variable.
  - NB. The universal quantifier is treated as a defined connective.

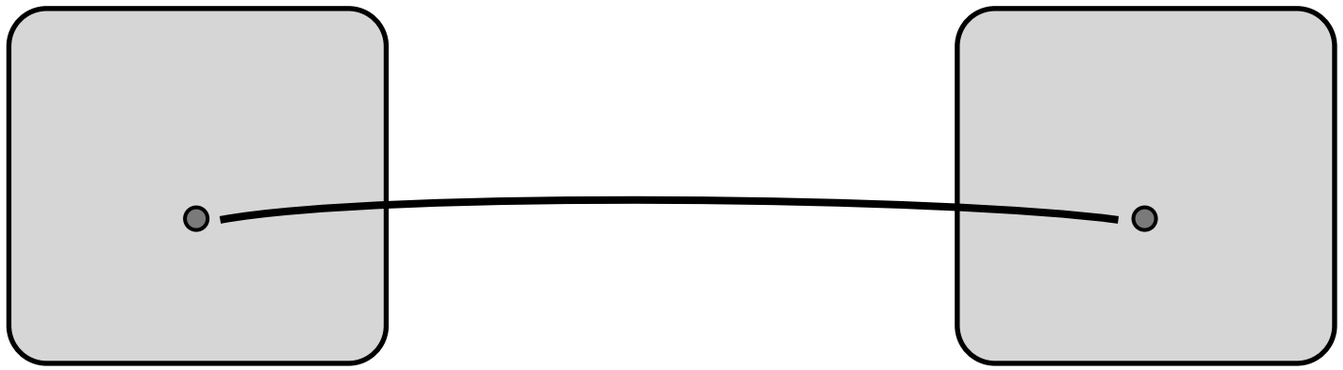
- Fixed-Point Extension (UNFP):
  - $\phi ::= \ldots \mid [\mathrm{LFP}_{Z,z}\, \phi(Z,\mathbf{Y},z)](x)$ (where Z occurs only positively in ϕ)
  - NB. No first-order parameters (i.e., only one free first-order variable).

- 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

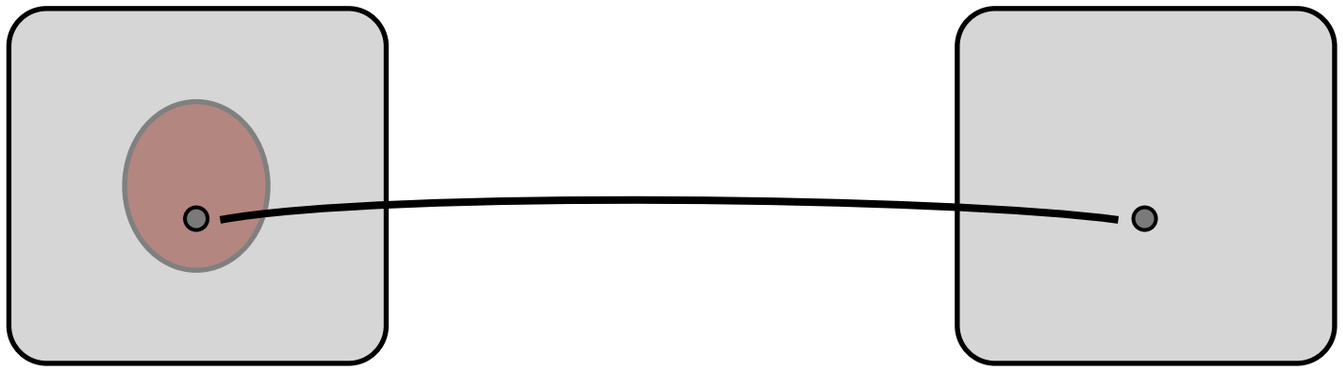
- Good model theory:
  - UNFO & UNFP have the tree-like model property (bounded tree-width)
  - UNFO has the finite model property
  - UNFO has Craig interpolation
  - UNFO can be characterized in terms of UN-bisimulation invariance.
- Decidable for satisfiability
  - 2ExpTime-complete, both on arbitrary 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.

UN-bisimulation game



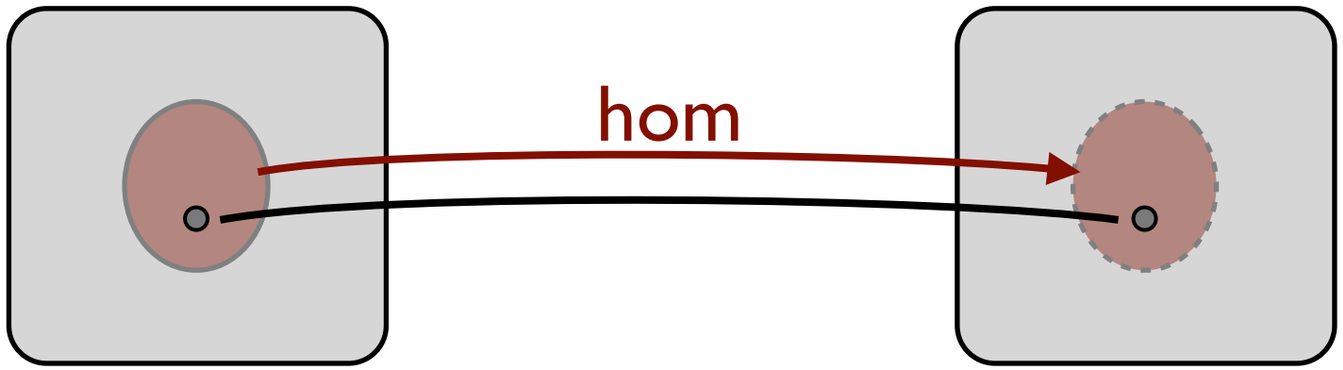
- The UN-bisimulation game:
  - Positions: pairs (a,b)
  - Moves:
    - Spoiler picks a finite set X in one of the structures.
    - Duplicator responds with a partial homomorphism h from X to the other structure (s.t. h(a)=b if a is in X).
    - Spoiler picks a pair (c,d) in h.

UN-bisimulation game



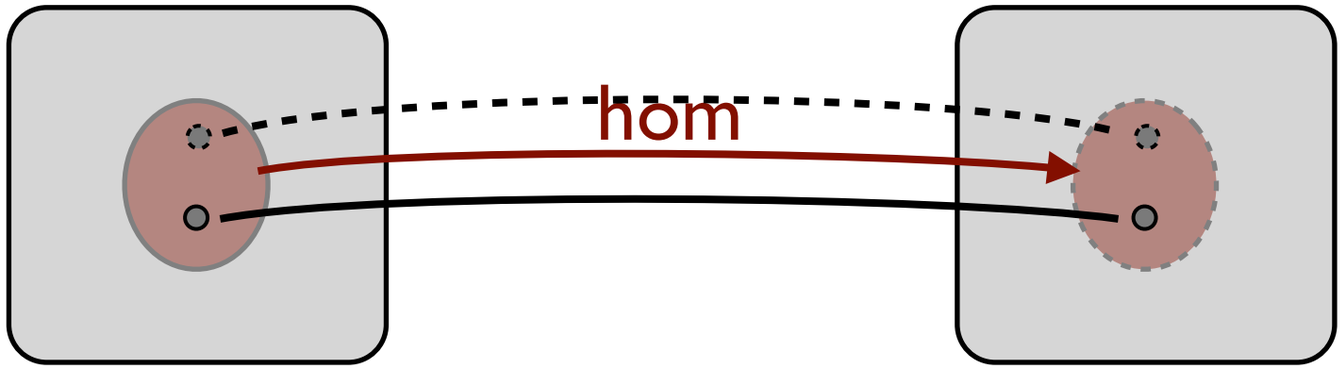
- The UN-bisimulation game:
  - Positions: pairs (a,b)
  - Moves:
    - Spoiler picks a finite set X in one of the structures.
    - Duplicator responds with a partial homomorphism h from X to the other structure (s.t. h(a)=b if a is in X).
    - Spoiler picks a pair (c,d) in h.

UN-bisimulation game



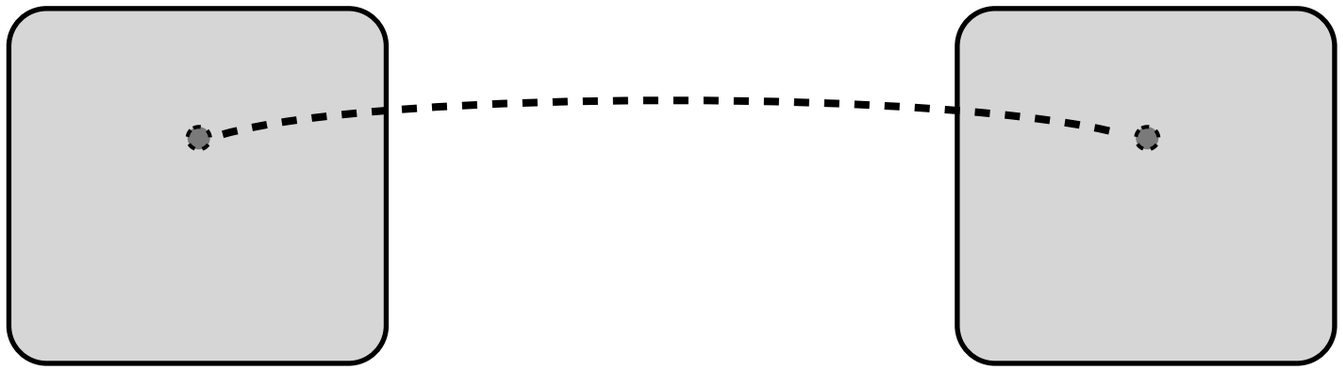
- The UN-bisimulation game:
  - Positions: pairs (a,b)
  - Moves:
    - Spoiler picks a finite set X in one of the structures.
    - Duplicator responds with a partial homomorphism h from X to the other structure (s.t. h(a)=b if a is in X).
    - Spoiler picks a pair (c,d) in h.

UN-bisimulation game



- The UN-bisimulation game:
  - Positions: pairs (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 if a is in X).
    - Spoiler picks a pair (c,d) in h.

UN-bisimulation game



- The UN-bisimulation game:
  - Positions: pairs (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 if a is in X).
    - Spoiler picks a pair (c,d) in h.

Unary Negation vs Guardedness

- What do UNFO & UNFP have that GFO & GFP do not have?
  - Allows unrestricted existential quantification, and therefore, includes Unions of Conjunctive Queries / monadic Datalog.
  - UNFO has Craig interpolation (which fails for GFO)

Unary Negation vs Guardedness

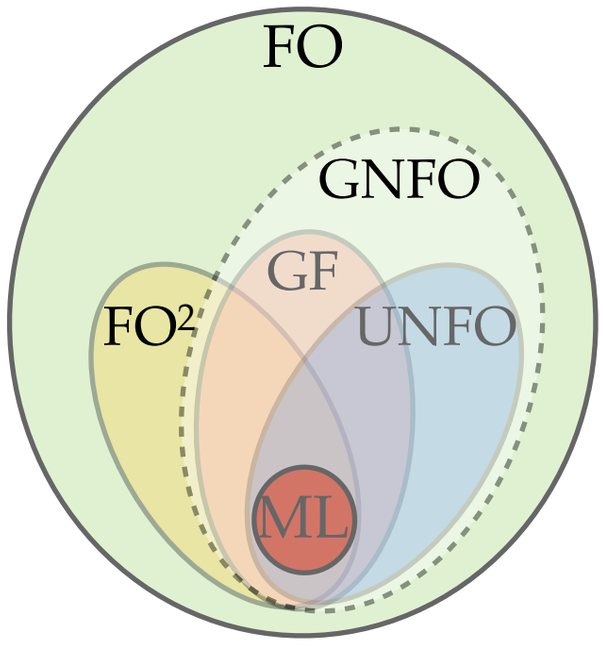
- What do UNFO & UNFP have that GFO & GFP do not have?
  - Allows unrestricted existential quantification, and therefore, includes Unions of Conjunctive Queries / monadic Datalog.
  - UNFO has Craig interpolation (which fails for GFO)

- A common generalization: guarded negation [Barany-tC-Segoufin '11].
  - Almost all results for unary negation have been lifted to guarded negation.
  - (Only exception: Craig interpolation)

Unary Negation vs Guardedness

- What do UNFO & UNFP have that GFO & GFP do not have?
  - Allows unrestricted existential quantification, and therefore, includes Unions of Conjunctive Queries / monadic Datalog.
  - UNFO has Craig interpolation (which fails for GFO)

- A common generalization: guarded negation [Barany-tC-Segoufin '11].
  - Almost all results for unary negation have been lifted to guarded negation.
  - (Only exception: Craig interpolation)



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

- 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.

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):
  - $\phi ::= \ldots \mid \mu_{Z,\mathbf{z}}[\, \text{guarded}_\sigma(\mathbf{z}) \wedge \phi(\mathbf{Y}, Z, \mathbf{z})\,](\mathbf{x})$ (where Z occurs only positively in ϕ)
  - (greatest fixed points can be expressed as dual)

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):
  - $\phi ::= \ldots \mid \mu_{Z,\mathbf{z}}[\, \text{guarded}_\sigma(\mathbf{z}) \wedge \phi(\mathbf{Y}, Z, \mathbf{z})\,](\mathbf{x})$ (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:

  $q[\phi_1/U_1, ..., \phi_n/U_n]$
  (where q is a UCQ containing relations U1, ..., Un)

  - $\phi ::= R(\mathbf{x}) \mid x=y \mid \exists x\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(x) \mid G(\mathbf{xy}) \wedge \neg\phi(\mathbf{x})$
  - I.e., ϕ is built up from atoms using (i) UCQs, and (ii) guarded negation.

Normal form

- GN-Normal form for GNFO formulas:

  $q[\phi_1/U_1, ..., \phi_n/U_n]$
  (where q is a UCQ containing relations U1, ..., Un)

  - $\phi ::= R(\mathbf{x}) \mid x=y \mid \exists x\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(x) \mid G(\mathbf{xy}) \wedge \neg\phi(\mathbf{x})$
  - I.e., ϕ is built up from atoms using (i) UCQs, and (ii) guarded negation.

- GN-Normal form for GNFP formulas:

  - $\phi ::= R(\mathbf{x}) \mid x=y \mid q[\phi_1/U_1, ..., \phi_n/U_n] \mid \neg\phi(x) \mid G(\mathbf{xy}) \wedge \neg\phi(\mathbf{x})$

    $\mid \mu_{Z,\mathbf{z}}[\ \text{guarded}_\sigma(\mathbf{z}) \wedge \phi(\mathbf{Y}, Z, \mathbf{z})\ ](\mathbf{x})$

  - I.e., ϕ is built up from atoms using (i) UCQs, (ii) guarded negation, and (iii) guarded LFPs.

Normal form

- GN-Normal form for GNFO formulas:

  $q[\phi_1/U_1, ..., \phi_n/U_n]$
  (where q is a UCQ containing relations U1, ..., Un)

  - $\phi ::= R(\mathbf{x}) \mid x=y \mid \exists x\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi(x) \mid G(\mathbf{xy}) \wedge \neg\phi(\mathbf{x})$
  - I.e., ϕ is built up from atoms using (i) UCQs, and (ii) guarded negation.

- GN-Normal form for GNFP formulas:

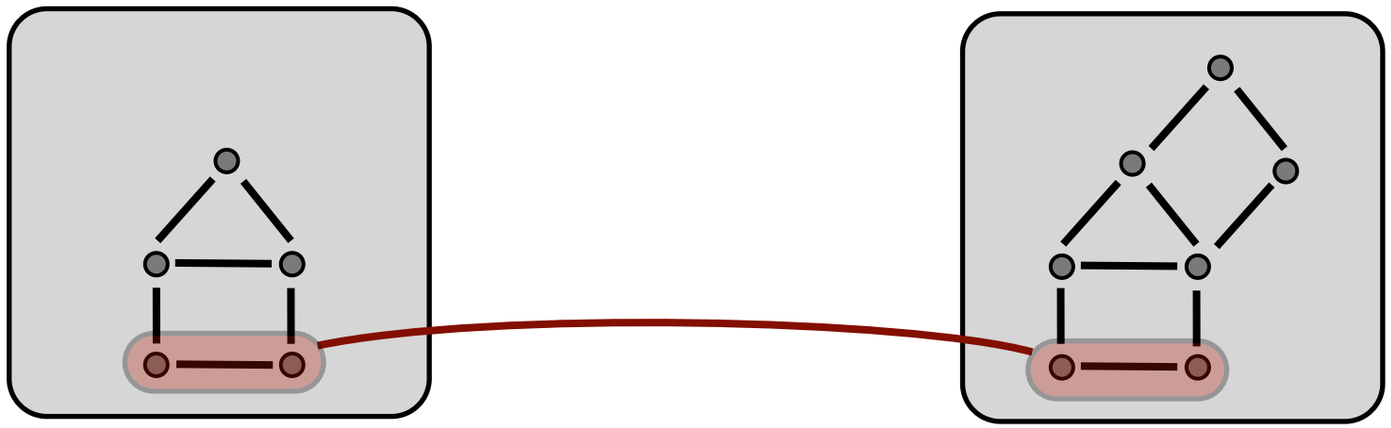
  - $\phi ::= R(\mathbf{x}) \mid x=y \mid q[\phi_1/U_1, ..., \phi_n/U_n] \mid \neg\phi(x) \mid G(\mathbf{xy}) \wedge \neg\phi(\mathbf{x})$

    $\mid \mu_{Z,\mathbf{z}}[\ \text{guarded}_\sigma(\mathbf{z}) \wedge \phi(\mathbf{Y}, Z, \mathbf{z})\ ](\mathbf{x})$
  - 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.

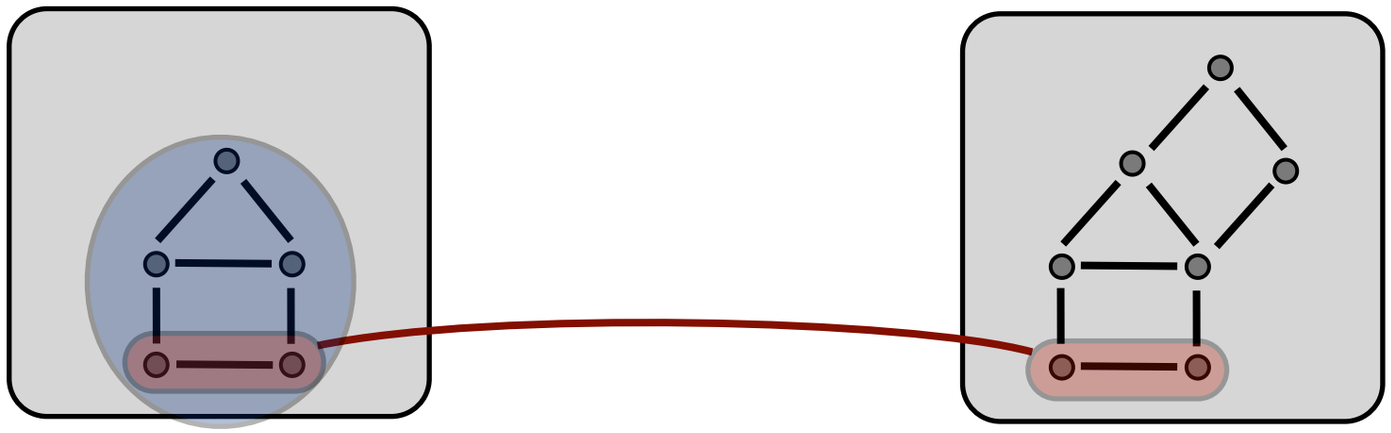
  - GNFO^k / GNFP^k: as above, but only use queries q with at most k variables.
  - GFO / GFP: as above, but only allow acyclic queries.

GN-Bisimulation Game



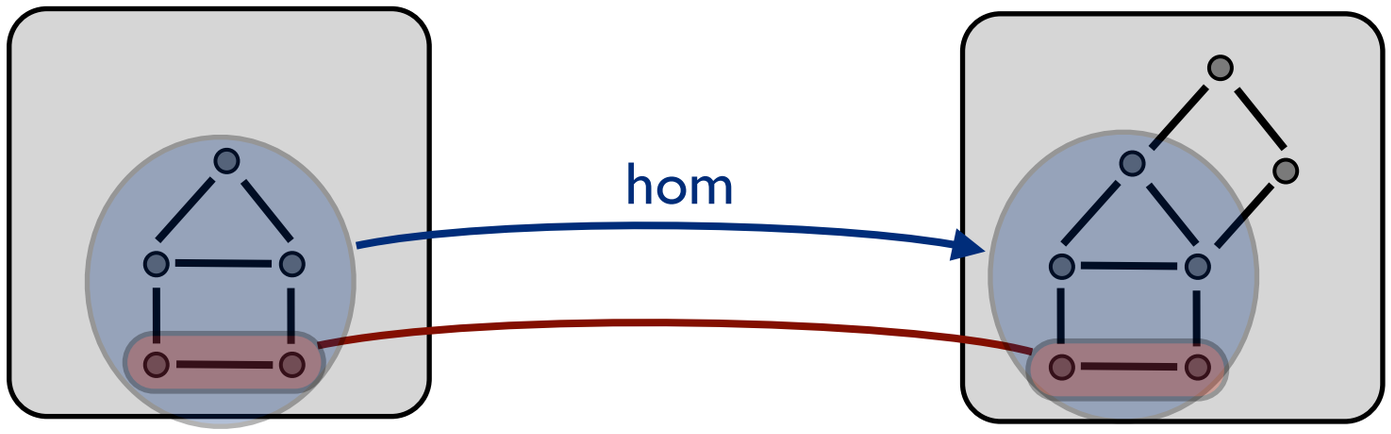
- The GN-bisimulation game:
  - Positions: pairs of guarded sets (**a**,**b**)
  - Moves:
    - Spoiler picks a finite set X in one of the structures.
    - Duplicator responds with a partial homomorphism h from X to the other structure, s.t. h(**a**)=**b**.
    - Spoiler picks guarded subsets (**c**,**d**)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

GN-Bisimulation Game



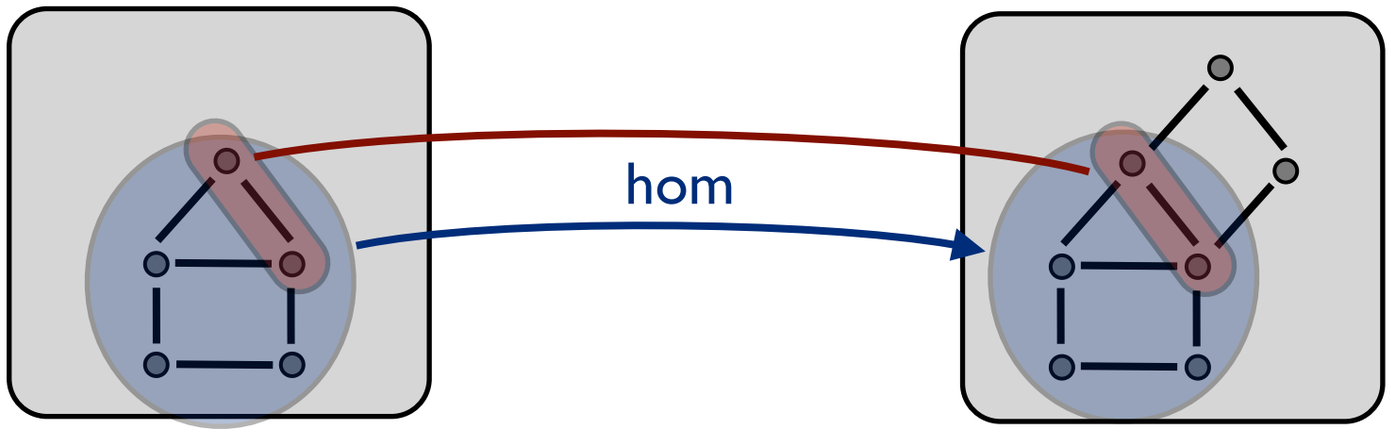
- The GN-bisimulation game:
  - Positions: pairs of guarded sets (**a**,**b**)
  - Moves:
    - Spoiler picks a finite set X in one of the structures.
    - Duplicator responds with a partial homomorphism h from X to the other structure, s.t. h(**a**)=**b**.
    - Spoiler picks guarded subsets (**c**,**d**)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

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.

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.

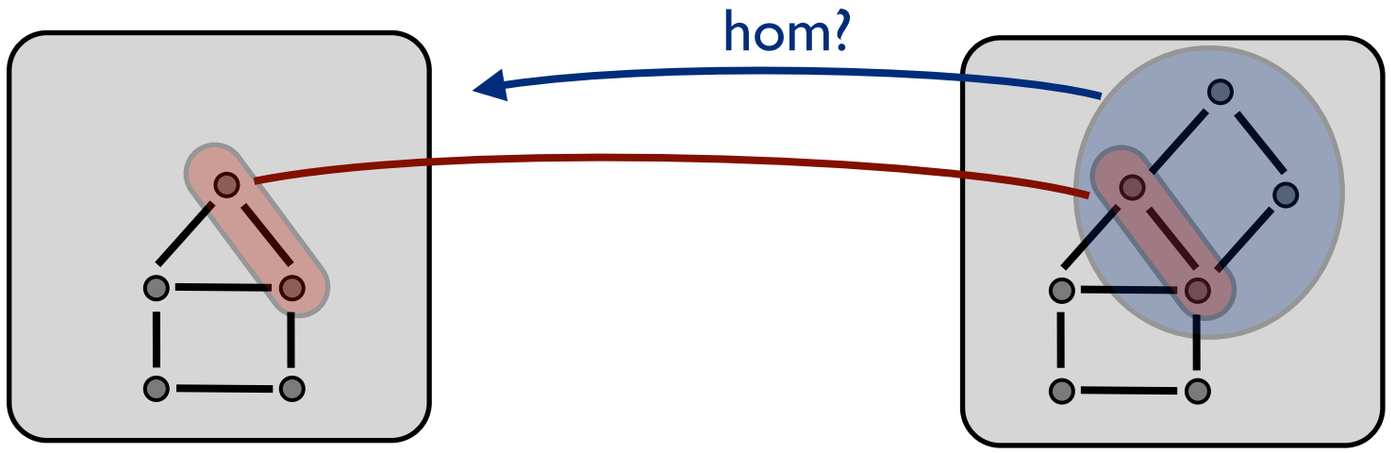
GN-Bisimulation Game

- 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.

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.

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.

GNFO and GNFP

- Good model theory
  - GNFO & GNFP have the tree-like model property (bounded tree-width)
  - GNFO has the finite model property (using [Barany-Gottlob-Otto 2010])
  - GNFO can be characterized in terms of GN-bisimulation invariance (cf. Martin Otto's talk)
- Decidable for satisfiability
  - 2ExpTime-complete, both on arbitrary structures and in the finite (using [Barany-Gottlob-Otto 2010; Barany-Bojanczyk 2011])
- Model checking:
  - GNFO: $P^{NP[\log^2]}$ - complete. GNFP: in $NP^{NP} \cap coNP^{NP}$ and P^{NP}-hard.
- Decidable boundedness problem for GNFO (using [Colcombet-Loeding; Blumensath-Otto-Weyer])

Three Examples

- Three “applications” of guarded negation.

1: Querying the Guarded Fragment

- Barany-Gottlob-Otto 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 $\phi \models q$.

1: Querying the Guarded Fragment

- Barany-Gottlob-Otto 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.

1: Querying the Guarded Fragment

- Barany-Gottlob-Otto 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 $\phi \wedge \neg q$.

1: Querying the Guarded Fragment

- Barany-Gottlob-Otto 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.

1: Querying the Guarded Fragment

- Barany-Gottlob-Otto 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.

1: Querying the Guarded Fragment

- Barany-Gottlob-Otto 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].

2: GNFO as a query language

- Many DB queries used in practice are not UCQs.
  - Many reasons, including the fact that some queries use negation.

2: GNFO as a query language

- Many DB queries used in practice are not UCQs.
  - Many reasons, including the fact that some queries use negation.

- We did a close analysis of standard benchmarks and workloads: most negation used in SQL queries is guarded negation.
  - Therefore, GNFO is of interest as a DB query language.

2: GNFO as a query language

- Many DB queries used in practice are not UCQs.
  - Many reasons, including the fact that some queries use negation.

- We did a close analysis of standard benchmarks and workloads: most negation used in SQL queries is guarded negation.
  - Therefore, GNFO is of interest as a DB query language.

- **Open-world query answering**: given an (incomplete) database D, a set of constraints T, and a (Boolean) query, is it the case that $D \wedge T \models q$?

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.
  - A tuple-generated dependency is frontier-guarded if the "exported variables" $\mathbf{x}$ co-occur in a single conjunct of ϕ [Baget, Mugnier, Rudolph, Thomazo '11]

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.
  - A tuple-generated dependency is frontier-guarded if the "exported variables" $\mathbf{x}$ co-occur in a single conjunct of ϕ [Baget, Mugnier, Rudolph, Thomazo '11]
  - Includes guarded tgds as a special case.

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.
  - A tuple-generated dependency is frontier-guarded if the "exported variables" $\mathbf{x}$ co-occur in a single conjunct of ϕ [Baget, Mugnier, Rudolph, Thomazo '11]
  - Includes guarded tgds as a special case.
  - Frontier-guarded tgds are expressible in GNFO as $\neg\exists \mathbf{x},\mathbf{y}(\phi(\mathbf{x},\mathbf{y}) \wedge \neg\exists \mathbf{z}\ \psi(\mathbf{y},\mathbf{z}))$.

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.
  - A tuple-generated dependency is frontier-guarded if the "exported variables" $\mathbf{x}$ co-occur in a single conjunct of ϕ [Baget, Mugnier, Rudolph, Thomazo '11]
  - Includes guarded tgds as a special case.
  - Frontier-guarded tgds are expressible in GNFO as $\neg\exists \mathbf{x},\mathbf{y}(\phi(\mathbf{x},\mathbf{y}) \wedge \neg\exists \mathbf{z}\ \psi(\mathbf{y},\mathbf{z}))$.
- **Open-world query answering** ($D \wedge T \models q$) for frontier-guarded tgds and GNFO queries:

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.
  - A tuple-generated dependency is frontier-guarded if the "exported variables" $\mathbf{x}$ co-occur in a single conjunct of ϕ [Baget, Mugnier, Rudolph, Thomazo '11]
  - Includes guarded tgds as a special case.
  - Frontier-guarded tgds are expressible in GNFO as $\neg\exists \mathbf{x},\mathbf{y}(\phi(\mathbf{x},\mathbf{y}) \wedge \neg\exists \mathbf{z}\ \psi(\mathbf{y},\mathbf{z}))$.
- **Open-world query answering** ($D \wedge T \models q$) for frontier-guarded tgds and GNFO queries:
  - Reduces to GNFO satisfiability, and therefore 2ExpTime-complete (combined complexity).

2: GNFO as a query language

- **Tuple-generating dependencies** (tgds): an important class of DB constraints.
  - $\forall \mathbf{x},\mathbf{y}\ \phi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z})$ where ϕ, ψ are conjunctions of atoms.
  - A tuple-generated dependency is frontier-guarded if the "exported variables" $\mathbf{x}$ co-occur in a single conjunct of ϕ [Baget, Mugnier, Rudolph, Thomazo '11]
  - Includes guarded tgds as a special case.
  - Frontier-guarded tgds are expressible in GNFO as $\neg\exists \mathbf{x},\mathbf{y}(\phi(\mathbf{x},\mathbf{y}) \wedge \neg\exists \mathbf{z}\ \psi(\mathbf{y},\mathbf{z}))$.
- **Open-world query answering** ($D \wedge T \models q$) for frontier-guarded tgds and GNFO queries:
  - Reduces to GNFO satisfiability, and therefore 2ExpTime-complete (combined complexity).
  - NP-complete for data complexity (we identified a large PTime fragment).

3: GN-Datalog

3: GN-Datalog

- **Datalog with stratified negation**
  - Rules of the form $R(\mathbf{x}) :- (\neg)R_1(\mathbf{x_1}), ..., (\neg)R_n(\mathbf{x_n})$
    - where all variables must occur positively, and all negated relation symbols R_i belong to a lower "stratum" than R.
  - The program is evaluated in stages, according to the stratum of the relation symbols.

3: GN-Datalog

- **Datalog with stratified negation**
  - Rules of the form $R(\mathbf{x}) :- (\neg)R_1(\mathbf{x_1}), ..., (\neg)R_n(\mathbf{x_n})$
    - where all variables must occur positively, and all negated relation symbols R_i belong to a lower "stratum" than R.
  - The program is evaluated in stages, according to the stratum of the relation symbols.

- GN-Datalog ("Guarded-negation Datalog"):
  - There is an positive atom in the body containing all first-order variables in the head of the rule, and
  - For every negative atom in the body, there is a positive atom containing the same (and possibly more) first-order variables.

3: GN-Datalog

- **Datalog with stratified negation**
  - Rules of the form $R(\mathbf{x}) :- (\neg)R_1(\mathbf{x_1}), ..., (\neg)R_n(\mathbf{x_n})$
    - where all variables must occur positively, and all negated relation symbols R_i belong to a lower "stratum" than R.
  - The program is evaluated in stages, according to the stratum of the relation symbols.

- GN-Datalog ("Guarded-negation Datalog"):
  - There is an positive atom in the body containing all first-order variables in the head of the rule, and
  - For every negative atom in the body, there is a positive atom containing the same (and possibly more) first-order variables.
- GN-Datalog contains Monadic Datalog and Datalog LIT.

3: GN-Datalog

- **Datalog with stratified negation**
  - Rules of the form $R(\mathbf{x}) :- (\neg)R_1(\mathbf{x_1}), ..., (\neg)R_n(\mathbf{x_n})$
    - where all variables must occur positively, and all negated relation symbols R_i belong to a lower "stratum" than R.
  - The program is evaluated in stages, according to the stratum of the relation symbols.

- GN-Datalog ("Guarded-negation Datalog"):
  - There is an positive atom in the body containing all first-order variables in the head of the rule, and
  - For every negative atom in the body, there is a positive atom containing the same (and possibly more) first-order variables.
- GN-Datalog contains Monadic Datalog and Datalog LIT.
- **GN-Datalog is contained in GNFP.** Therefore, containment is decidable (2ExpTime-cmp). Boundedness (suitably defined) is decidable as well.

Decidability of GNFP Satisfiability

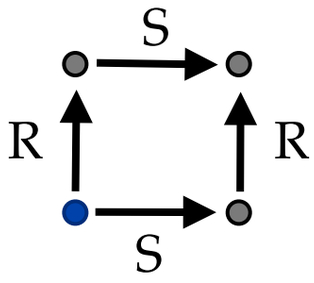
- **Thm**: (Finite) satisfiability of GNFP is 2ExpTime complete.
- Main ingredients of the proof:
  - Treeifications of cyclic conjunctive queries,
  - Rosati covers ([Barany-Gottlob-Otto 2010])
  - A reduction from GNFP to GFP.

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.

Treeification

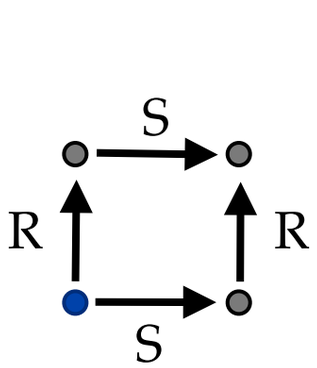
- **Treeification**:
  - If q is a CQ and τ is a schema, Λ^{τ}_{q} is the union of all minimal acyclic CQs over τ that imply q.



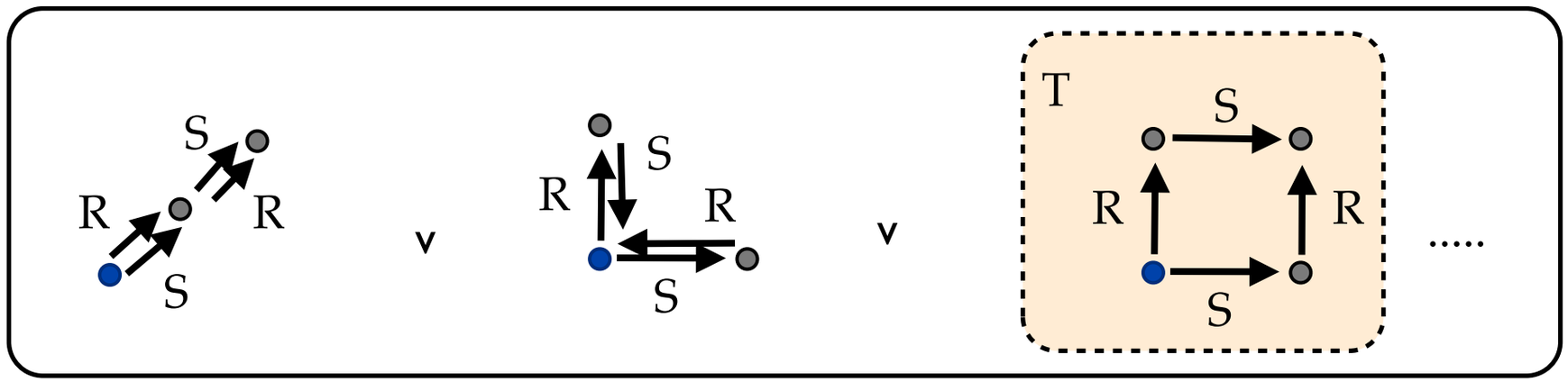
A (cyclic) CQ

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.



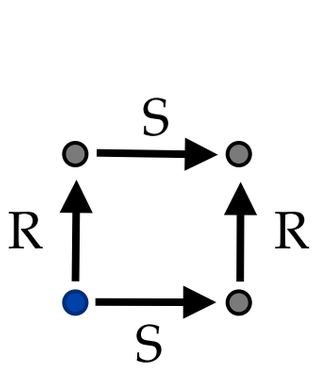
A (cyclic) CQ



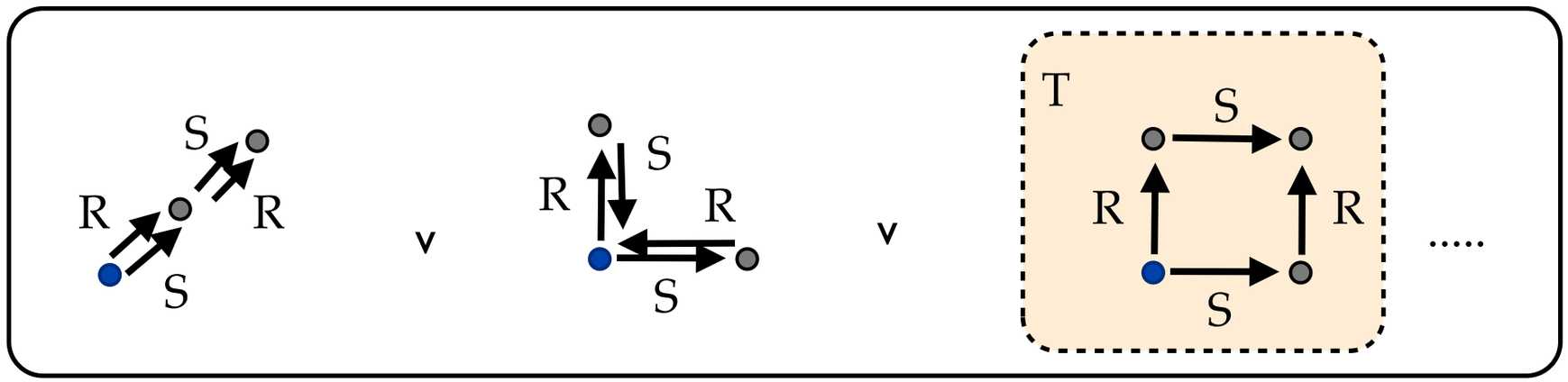
Its treeification

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 acyclic structures, q and Λ^{τ}_{q} are equivalent.

Proof sketch

- **Theorem**. (Finite) satisfiability for GNFP reduces to (finite) satisfiability for GFP.
- Proof sketch:

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.

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

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.

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).

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 acyclic, we have that $M^* \models \phi$.

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 acyclic, 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**:
  - GNFO: common generalization of GFO and conjunctive queries.
  - GNFP: further extension with least fixed point operator.
  - embeds many existing logics

Summary

- **Guarded negation logic**:
  - GNFO: common generalization of GFO and conjunctive queries.
  - GNFP: further extension with least fixed point operator.
  - embeds many existing logics
- Well behaved both (finite) model theoretically and computationally.

Summary

- **Guarded negation logic**:
  - GNFO: common generalization of GFO and conjunctive queries.
  - GNFP: further extension with least fixed point operator.
  - embeds many existing logics
- Well behaved both (finite) model theoretically and computationally.
- Open question:
  - Craig interpolation for GNFO
  - Finite model-theoretic characterization of GNFO in terms of GN-bisimulations (characterization for $GNFO^k$ was shown by Martin Otto)