Expressive Power of Property Graph Constraint Languages

Steven Sailly
steven.sailly<at>ip-paris.fr

SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris

Séminaire Jeunes Chercheurs - LIGM
2025-11-05

Joint work with Stefania Dumbrava, Nadime Francis, and Victor Marsault

Outline

Property graph

We fix countably infinite and pairwise distinct sets $\mathcal{O}$, $\mathcal{L}$, $\mathcal{K}$ and $\mathcal{V}$ of objects, labels, keys and values, respectively.

A *property graph* is a tuple $G = (V, E, \eta, \lambda, \pi)$ such that:

- $V \subset \mathcal{O}$ is a finite set of vertices,
- $E \subset \mathcal{O}$ is a finite set of edges with $V \cap E = \emptyset$,
- $\eta : E \to V \times V$ is the incidence function assigning source and target vertices to each edge.
- $\lambda : V \cup E \to 2^{\mathcal{L}}$ is a labeling function such that $\lambda(o)$ is finite, for each $o \in V \cup E$
- $\pi : (V \cup E) \times \mathcal{K} \rightharpoonup \mathcal{V}$ is a partial function that assigns a value to the properties of a vertex or edge; we write $u.k$ as a shorthand for $\pi(u, k)$.

Property graph

We fix countably infinite and pairwise distinct sets $\mathcal{O}$, $\mathcal{L}$, $\mathcal{K}$ and $\mathcal{V}$ of objects, labels, keys and values, respectively.
A *property graph* is a tuple $G = (V, E, \eta, \lambda, \pi)$ such that:

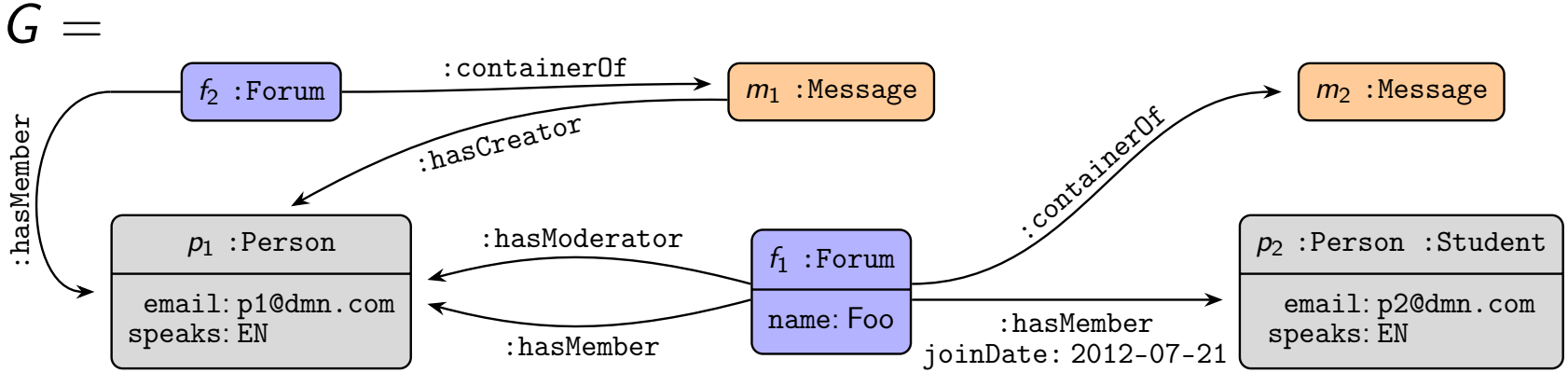
- $V \subset \mathcal{O}$ is a finite set of vertices,
- $E \subset \mathcal{O}$ is a finite set of edges with $V \cap E = \emptyset$,
- $\eta : E \to V \times V$ is the incidence function assigning source and target vertices to each edge.
- $\lambda : V \cup E \to 2^{\mathcal{L}}$ is a labeling function such that $\lambda(o)$ is finite, for each $o \in V \cup E$
- $\pi : (V \cup E) \times \mathcal{K} \rightharpoonup \mathcal{V}$ is a partial function that assigns a value to the properties of a vertex or edge; we write $u.k$ as a shorthand for $\pi(u, k)$.

Property graph

We fix countably infinite and pairwise distinct sets $\mathcal{O}$, $\mathcal{L}$, $\mathcal{K}$ and $\mathcal{V}$ of objects, labels, keys and values, respectively.
A *property graph* is a multilabeled multigraph with key-value pairs on vertices and edges.

Property graph

An example



Property graph instance based on the LDBC SNB Schema [Ang+20].

- $\lambda(p_2) = \{\texttt{Person}, \texttt{Student}\}$
- $f_1.\texttt{name} = \text{Foo}$
- for all $k \in \mathcal{K}$, $m_1.k$ is undefined

Conjunctive Queries

aka CQ

We fix Vars a countably infinite set of variables.
A *conjunctive query* is a tuple $Q(\overline{x}) = (V_Q, E_Q, \eta_Q, \lambda_Q)$ where:

- $V_Q \subset$ Vars is a finite set of vertex variables,
- $E_Q \subset$ Vars is a finite set of edge variables with $V_Q \cap E_Q = \emptyset$,
- $\eta_Q : E_Q \to V_Q \times V_Q$ is a function that assigns source and target vertices to each edge,
- $\lambda_Q : V_Q \cup E_Q \to 2^{\mathcal{L}}$ is a labeling function,
- $\overline{x}$ is a tuple of variables from V_Q and E_Q.

CQ + property map with an empty domain = property graph

Conjunctive Queries

aka CQ

We fix Vars a countably infinite set of variables.
A *conjunctive query* is a tuple $Q(\overline{x}) = (V_Q, E_Q, \eta_Q, \lambda_Q)$ where:

- $V_Q \subset$ Vars is a finite set of vertex variables,
- $E_Q \subset$ Vars is a finite set of edge variables with $V_Q \cap E_Q = \emptyset$,
- $\eta_Q : E_Q \to V_Q \times V_Q$ is a function that assigns source and target vertices to each edge,
- $\lambda_Q : V_Q \cup E_Q \to 2^{\mathcal{L}}$ is a labeling function,
- $\overline{x}$ is a tuple of variables from V_Q and E_Q.

CQ + property map with an empty domain = property graph

Conjunctive Queries

Homomorphism, match

Let $Q(\overline{x}) = (V_Q, E_Q, \eta_Q, \lambda_Q)$ be a CQ and $G = (V, E, \eta, \lambda, \pi)$ be a graph. A *homomorphism* from Q to G is a function $h : V_Q \cup E_Q \to V \cup E$ such that:

1. for all $u \in V_Q$, $h(u) \in V$;
2. for all $e \in E_Q$, $h(e) \in E$ and $\eta(h(e)) = (h(u_1), h(u_2))$, where $(u_1, u_2) = \eta_Q(e)$;
3. for all $o \in V_Q \cup E_Q$, $\lambda_Q(o) \subseteq \lambda(h(o))$.

A *match* for Q over a graph G is a homomorphism h from Q to G.
The *answer* of $Q(x_1, \ldots, x_k)$ over G is:
$Q(G) = \{(h(x_1), \ldots, h(x_k)) \mid h \text{ is a match for } Q \text{ over } G\}$.

Conjunctive Queries

Homomorphism, match

Let $Q(\overline{x}) = (V_Q, E_Q, \eta_Q, \lambda_Q)$ be a CQ and $G = (V, E, \eta, \lambda, \pi)$ be a graph. A *homomorphism* from Q to G is a function $h : V_Q \cup E_Q \to V \cup E$ such that:

1. for all $u \in V_Q$, $h(u) \in V$;
2. for all $e \in E_Q$, $h(e) \in E$ and $\eta(h(e)) = (h(u_1), h(u_2))$, where $(u_1, u_2) = \eta_Q(e)$;
3. for all $o \in V_Q \cup E_Q$, $\lambda_Q(o) \subseteq \lambda(h(o))$.

A *match* for Q over a graph G is a homomorphism h from Q to G.

The *answer* of $Q(x_1, \ldots, x_k)$ over G is:
$Q(G) = \{(h(x_1), \ldots, h(x_k)) \mid h \text{ is a match for } Q \text{ over } G\}$.

Conjunctive Queries

Homomorphism, match

Let $Q(\overline{x}) = (V_Q, E_Q, \eta_Q, \lambda_Q)$ be a CQ and $G = (V, E, \eta, \lambda, \pi)$ be a graph. A *homomorphism* from Q to G is a function $h : V_Q \cup E_Q \to V \cup E$ such that:

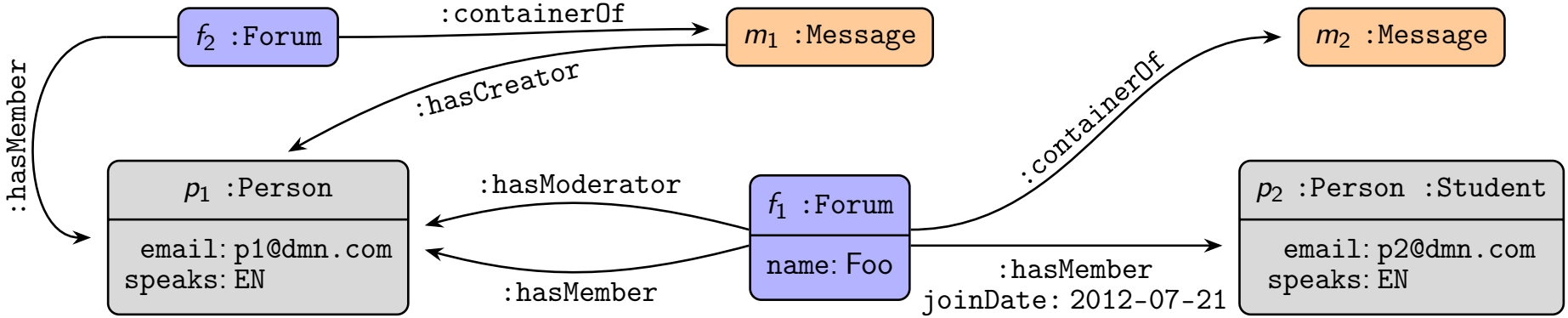
1. for all $u \in V_Q$, $h(u) \in V$;
2. for all $e \in E_Q$, $h(e) \in E$ and $\eta(h(e)) = (h(u_1), h(u_2))$, where $(u_1, u_2) = \eta_Q(e)$;
3. for all $o \in V_Q \cup E_Q$, $\lambda_Q(o) \subseteq \lambda(h(o))$.

A *match* for Q over a graph G is a homomorphism h from Q to G. The *answer* of $Q(x_1, \ldots, x_k)$ over G is:
$Q(G) = \{(h(x_1), \ldots, h(x_k)) \mid h \text{ is a match for } Q \text{ over } G\}$.

Conjunctive Queries

Examples

$G =$



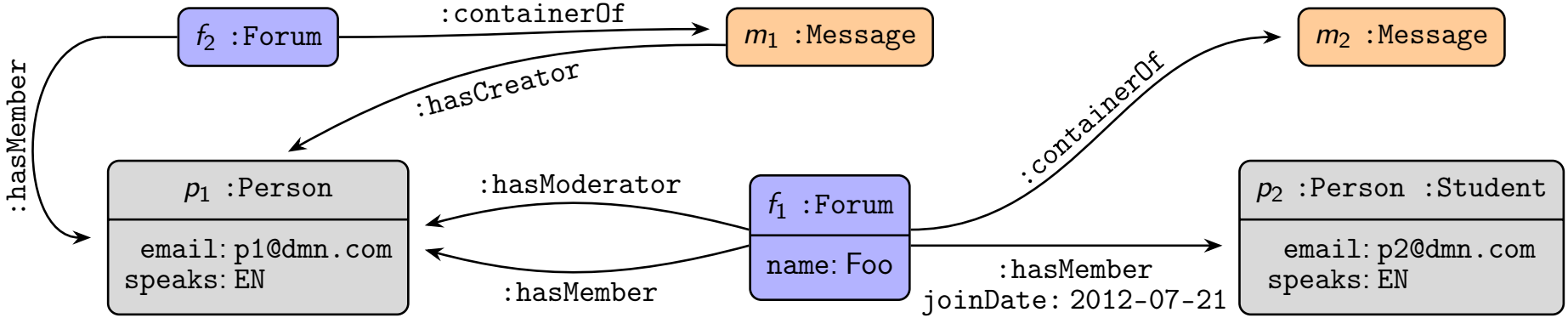
$Q_1 =$ x :Forum $\longrightarrow$ y :Person

$Q_1(G) = \{(h(x), h(y)) \mid h \text{ is a match for } Q_1 \text{ over } G\}$

Conjunctive Queries

Examples

$G =$



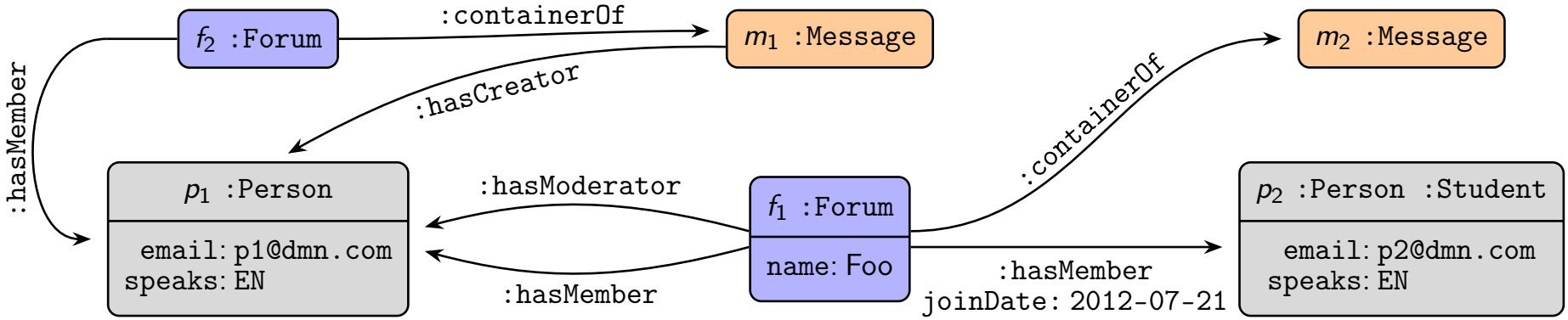
$Q_1 =$ x :Forum $\longrightarrow$ y :Person

$$Q_1(G) = \{(h(x), h(y)) \mid h \text{ is a match for } Q_1 \text{ over } G\}$$
$$= \{(f_1, p_1), (f_1, p_2), (f_2, p_1)\}$$

Conjunctive Queries

Examples

$G =$



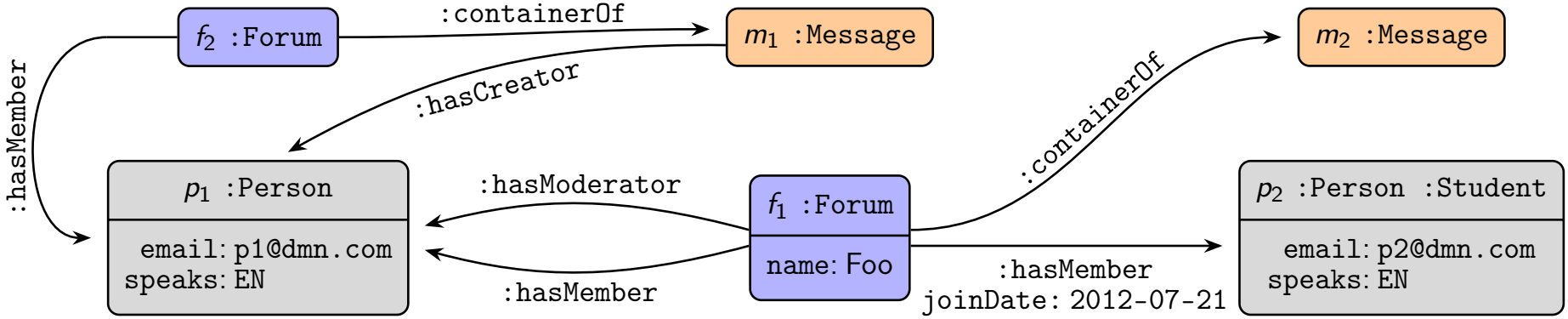
$Q_2 =$ x :Forum —:hasModerator→ y :Person

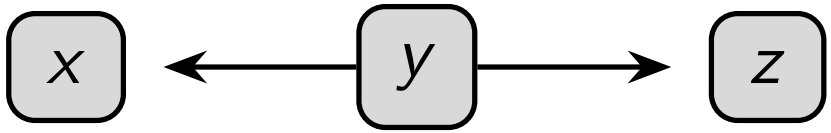
$Q_2(G) = \{(f_1, p_1)\}$

Conjunctive Queries

Examples

$G =$



$Q_3 =$ 

$$Q_3(G) = \{(p_1, f_1, p_1), (p_1, f_1, p_2), (p_1, f_1, m_2), (p_2, f_1, p_2), (p_2, f_1, p_1), (p_2, f_1, m_2), (m_2, f_1, m_2), (m_2, f_1, p_1), (m_2, f_1, p_2), (p_1, f_2, p_1), (p_1, f_2, m_1), (m_1, f_2, m_1), (m_1, f_2, p_1), (p_1, m_1, p_1)\}$$

Outline

Data predicates

Definition

Let $x, y \in \text{Vars}$, $a, b \in \mathcal{K}$, and $c \in \mathcal{V}$. *Data predicates* are:

- (in)equalities of identifiers: $x \doteq y$, $x \neq\!\!\!\doteq y$;
- (in)equalities of data values: $x.a \doteq y.b$, $x.a \neq\!\!\!\doteq y.b$;
- (in)equalities between a data value and a constant: $x.a \doteq c$, $x.a \neq\!\!\!\doteq c$;

Data predicates

Satisfaction

Let $G = (V, E, \eta, \lambda, \pi)$ be a graph, h be a match over G for some query Q, and p be a data predicate. *h satisfies p* ($h \models p$) if:

- $p \in \{x \doteq y, x \not\doteq y\}$: $x, y \in \mathrm{Dom}(h)$ and $h(x) = h(y)$ (resp. $h(x) \neq h(y)$);
- $p \in \{x.a \doteq c, x.a \not\doteq c\}$: $x \in \mathrm{Dom}(h)$, $(h(x), a) \in \mathrm{Dom}(\pi)$ and $\pi(h(x), a) = c$ (resp. $\pi(h(x), a) \neq c$);
- $p \in \{x.a \doteq y.b, x.a \not\doteq y.b\}$: $x, y \in \mathrm{Dom}(h)$, $(h(x), a), (h(y), b) \in \mathrm{Dom}(\pi)$ and $\pi(h(x), a) = \pi(h(y), b)$ (resp. $\pi(h(x), a) \neq \pi(h(y), b)$).

If $P = \{p_1, \ldots, p_n\}$, $h \models P \iff \forall i \in [n], h \models p_i$.

Given a query Q and a set of data predicates P, h is a *match* for (Q, P) if it is a match for Q and $h \models P$.

Data predicates

Satisfaction

Let $G = (V, E, \eta, \lambda, \pi)$ be a graph, h be a match over G for some query Q, and p be a data predicate. *h satisfies p* ($h \models p$) if all objects in p have a value and they are equal/different.

If $P = \{p_1, \ldots, p_n\}$, $h \models P \iff \forall i \in [n], h \models p_i$.

Given a query Q and a set of data predicates P, h is a *match* for (Q, P) if it is a match for Q and $h \models P$.

Data predicates

Satisfaction

Let $G = (V, E, \eta, \lambda, \pi)$ be a graph, h be a match over G for some query Q, and p be a data predicate. *h satisfies p* ($h \models p$) if all objects in p have a value and they are equal/different.

If $P = \{p_1, \ldots, p_n\}$, $h \models P \iff \forall i \in [n], h \models p_i$.

Given a query Q and a set of data predicates P, h is a *match* for (Q, P) if it is a match for Q and $h \models P$.

Data predicates

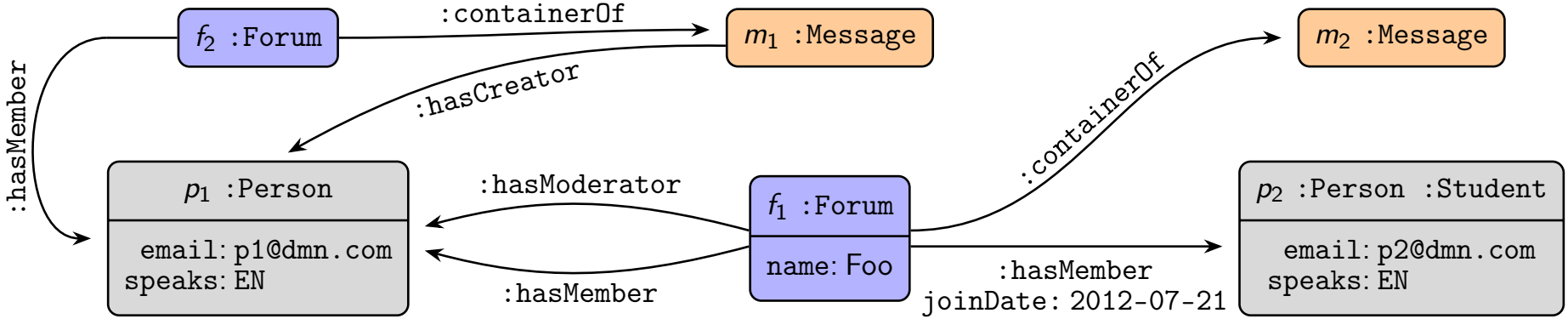
Satisfaction

Let $G = (V, E, \eta, \lambda, \pi)$ be a graph, h be a match over G for some query Q, and p be a data predicate. h *satisfies* p ($h \models p$) if all objects in p have a value and they are equal/different.
If $P = \{p_1, \ldots, p_n\}$, $h \models P \iff \forall i \in [n], h \models p_i$.
Given a query Q and a set of data predicates P, h is a *match* for (Q, P) if it is a match for Q and $h \models P$.

Data predicates

Examples

$G =$



$Q_1 =$ x :Forum $\longrightarrow$ y :Person

$P_1 = \{\mathrm{y.speaks} \doteq \mathrm{EN}\}$

$h_1 : \quad x \mapsto f_1$

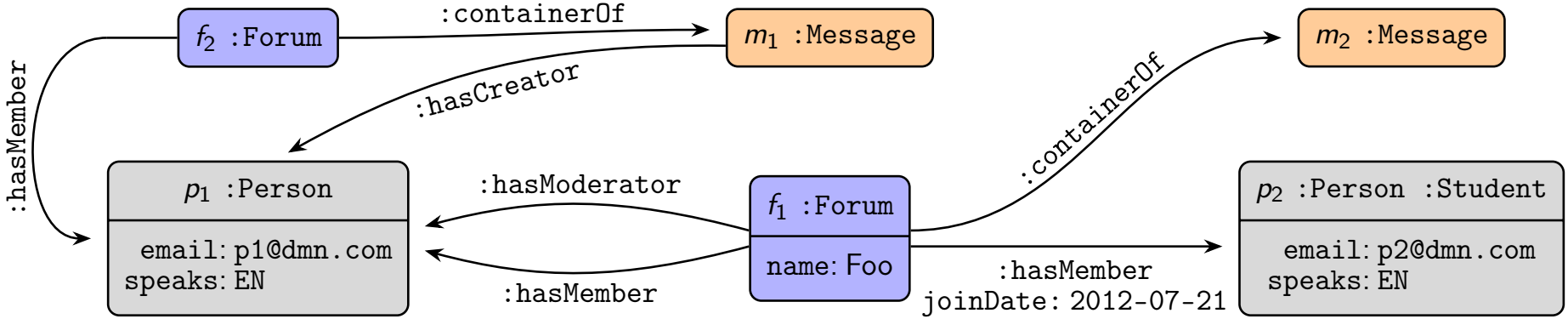
$\qquad\quad y \mapsto p_1$

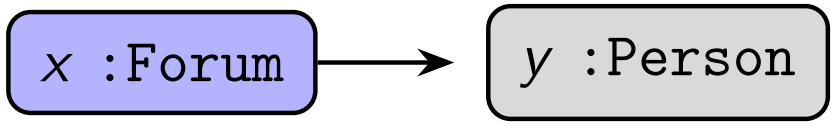
h_1 is a match for (Q_1, P_1).

Data predicates

Examples

$G =$



$Q_1 =$ 

$P_2 = \{\mathtt{x.name} \doteq \mathsf{Foo}, \mathtt{y.email} \doteq \mathtt{y.email}\}$

$h_2 : \quad x \mapsto f_2$

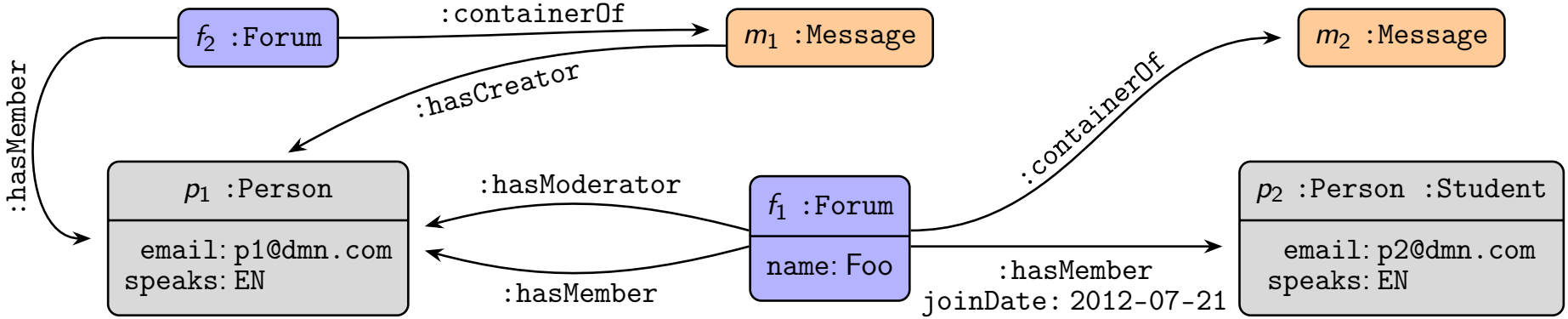
$\qquad\quad y \mapsto p_1$

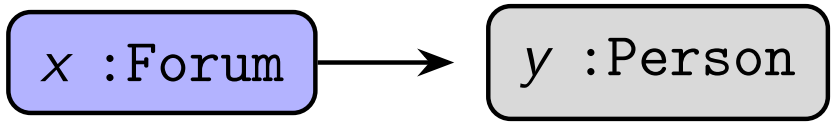
h_2 is not a match for (Q_1, P_2).

Data predicates

Examples

$G =$



$Q_1 =$ 

$P_2 = \{\mathtt{x.name} \doteq \mathsf{Foo}, \mathtt{y.email} \doteq \mathtt{y.email}\}$

$h_3 : \quad x \mapsto f_1$

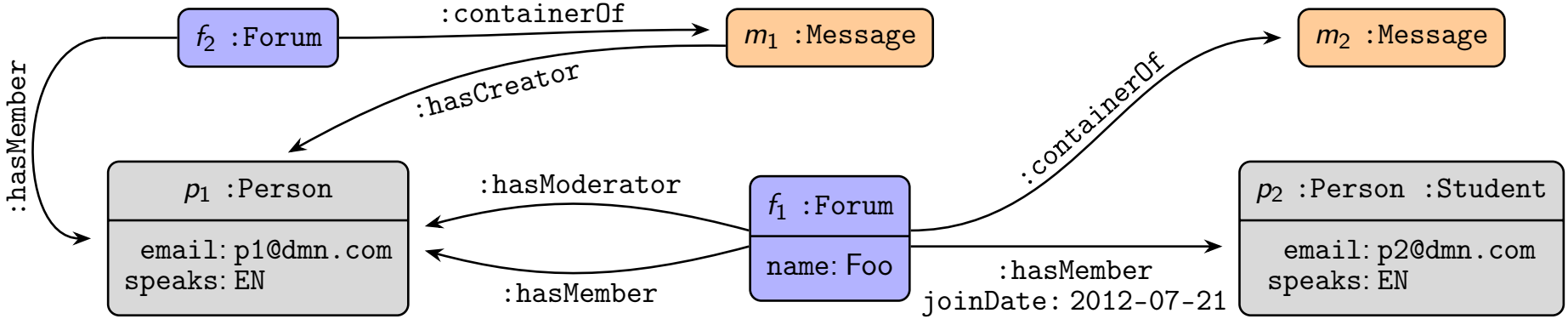
$\qquad z \mapsto p_1$

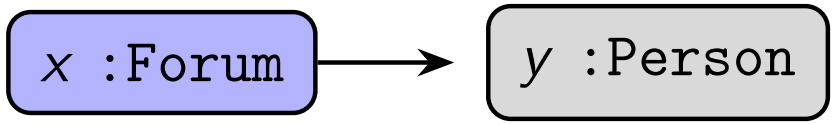
h_3 is not a match for (Q_1, P_2).

Data predicates

Examples

$G =$



$Q_1 =$ 

$P_2 = \{\mathtt{x.name} \doteq \mathsf{Foo}, \mathtt{y.email} \doteq \mathtt{y.email}\}$

$h_1 : \quad x \mapsto f_1$

$\qquad\quad y \mapsto p_1$

h_1 is a match for (Q_1, P_2).

Our cases of interest

Two cases of interest:

- $\mathrm{CQ}[\dot{=}]$: CQ with predicates of equality;
- $\mathrm{CQ}[\dot{=}, \dot{\neq}]$: CQ with predicates of equality and inequality.

Graph Generating Dependency

aka GGD [SFY20]

A *Graph Generating Dependency* (GGD) is a tuple $(Q_s(\overline{x}), C_s(\overline{x}) \Rightarrow Q_t(\overline{x}', \overline{y}), C_t(\overline{x}', \overline{y}))$ where:

- $\overline{x}$ and $\overline{y}$ are disjoint tuples of variables;
- $\overline{x}' \subseteq \overline{x}$;
- $Q_s, Q_t \in \mathrm{CQ}$ are the source/target queries;
- C_s, C_t are sets of data predicates.

A GGD is in nGGD if $|\overline{x}'| \leq n$.

Graph Generating Dependency

aka GGD [SFY20]

A *Graph Generating Dependency* (GGD) is a tuple $(Q_s(\overline{x}), C_s(\overline{x}) \Rightarrow Q_t(\overline{x}', \overline{y}), C_t(\overline{x}', \overline{y}))$ where:

- $\overline{x}$ and $\overline{y}$ are disjoint tuples of variables;
- $\overline{x}' \subseteq \overline{x}$;
- $Q_s, Q_t \in \mathrm{CQ}$ are the source/target queries;
- C_s, C_t are sets of data predicates.

A GGD is in $n\mathrm{GGD}$ if $|\overline{x}'| \leq n$.

Graph Functional Dependency

aka GFD [FWX16]

A *Graph Functional Dependency* (GFD) is a GGD where $\overline{y}$ is empty and $Q_t(\overline{x}')$ is trivial.
A GFD will be written as $(Q(\overline{x}), C_s(\overline{x}) \Rightarrow C_t(\overline{x}'))$.
A GFD is in $n\mathrm{GFD}$ if $|\overline{x}'| \leq n$.

Graph Functional Dependency

aka GFD [FWX16]

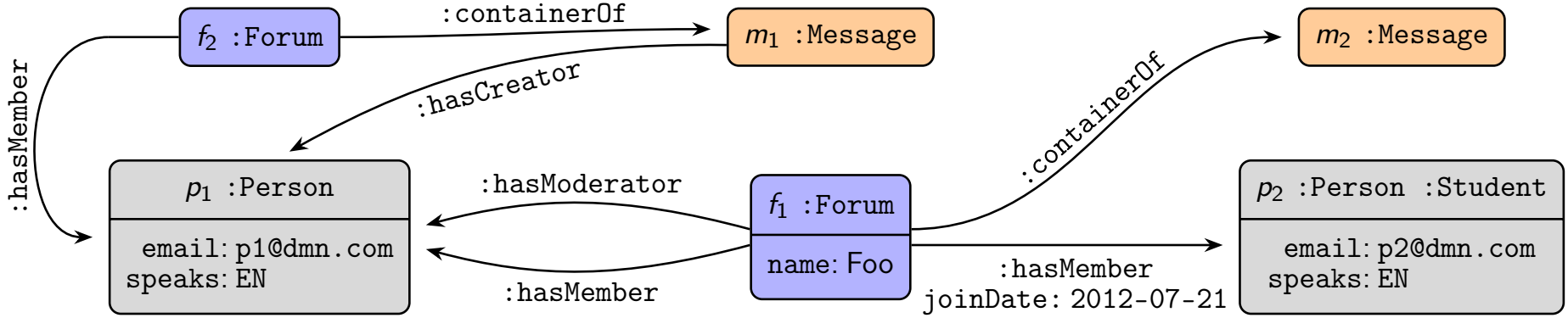
A *Graph Functional Dependency* (GFD) is a GGD where $\overline{y}$ is empty and $Q_t(\overline{x}')$ is trivial.
A GFD will be written as $(Q(\overline{x}), C_s(\overline{x}) \Rightarrow C_t(\overline{x}'))$.
A GFD is in $n\mathrm{GFD}$ if $|\overline{x}'| \leq n$.

Let φ be a GGD.
A graph G satisfies φ ($G \models \varphi$) if for all matches h_s of (Q_s, C_s) in G there exists a match h_t for (Q_t, C_t) over G that is consistent with h_s, i.e. $\forall x \in \overline{x}', h_s(x) = h_t(x)$.
Otherwise G *violates* φ ($G \not\models \varphi$).

GFDs & GGDs

Examples

$G =$



$$\text{GFD } \varphi_1 = \left((\mathtt{x}) \xleftarrow{\mathtt{:hasMember}} (\mathtt{z}) \xrightarrow{\mathtt{:hasMember}} (\mathtt{y}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\}\right)$$

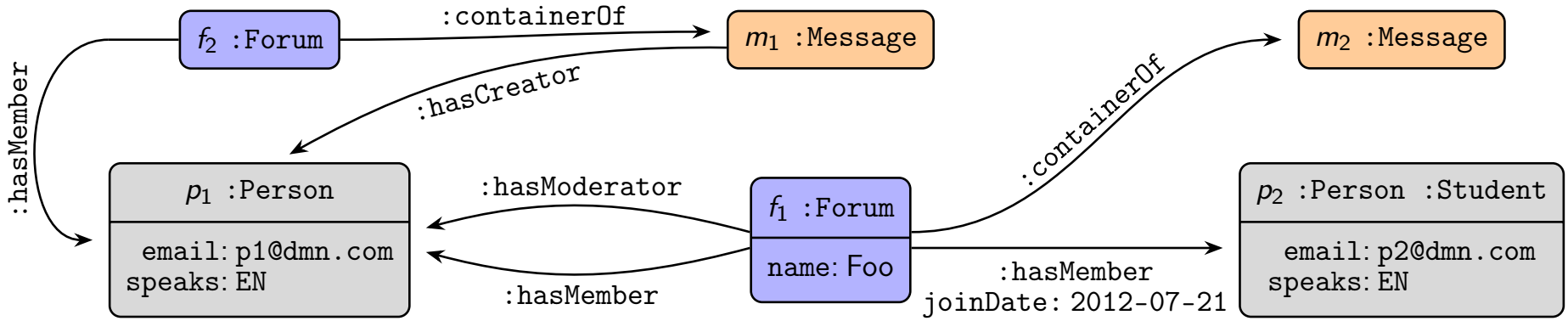
Two members of a same forum must speak the same language.

$G \models \varphi_1$

GFDs & GGDs

Examples

$G =$



$$\text{GGD } \varphi_2 = \big((\mathtt{z}) \xrightarrow{\mathtt{:containerOf}} (\mathtt{y}) \xrightarrow{\mathtt{:hasCreator}} (\mathtt{x}), \emptyset \Rightarrow (\mathtt{z}) \xrightarrow{\mathtt{:hasMember}} (\mathtt{x}), \emptyset\big)$$

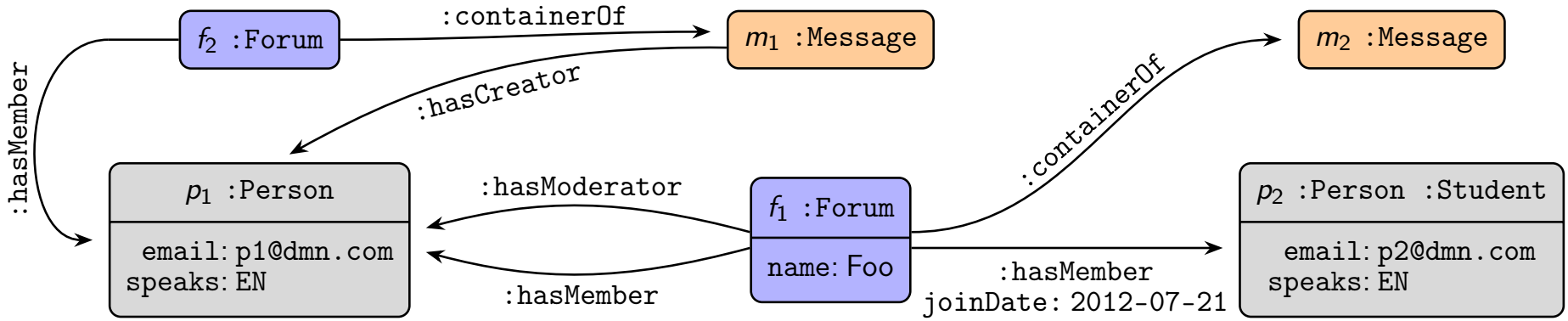
The creator of a message is a member of the forum where the message is posted.

$G \models \varphi_2$

GFDs & GGDs

Examples

$G =$



GFD $\varphi_3 = (\texttt{(x:Forum)}, \emptyset \Rightarrow \{\texttt{x.name} \doteq \texttt{x.name}\})$

A forum must have a name.

$G \not\models \varphi_3$

PG-Keys

[Ang+21]

A PG-Key is an expression of the form:

$$\texttt{FOR}\ x\ \texttt{WITHIN}\ Q_s(\overline{x}), C_s(\overline{x})\ \texttt{EXCLUSIVE} \mid \texttt{MANDATORY} \mid \texttt{SINGLETON}\ \overline{\nu}\ \texttt{WITHIN}\ Q_t(x, \overline{y}), C_t(x, \overline{y})$$

where:

- $\overline{x}$ and $\overline{y}$ are disjoint tuples of variables;
- $x \in \overline{x}$;
- $Q_s, Q_t \in \mathrm{CQ}$ are the source/target queries;
- C_s, C_t are sets of data predicates;
- EXCLUSIVE, MANDATORY and SINGLETON are *assertion keywords*;
- $\overline{\nu} = (\nu_1, \ldots, \nu_k)$ is a *key expression*, where for all $i \in [k]$, $\nu_i \in \{z, z.a\}$, where $z \in \{x\} \cup \overline{y}$ and $a \in \mathcal{K}$.

An mPG-Key is a PG-Key that contains the assertion keyword MANDATORY.

PG-Keys

[Ang+21]

A PG-Key is an expression of the form:

$$\texttt{FOR}\ x\ \texttt{WITHIN}\ Q_s(\overline{x}), C_s(\overline{x})\ \texttt{EXCLUSIVE} \mid \texttt{MANDATORY} \mid \texttt{SINGLETON}\ \overline{\nu}\ \texttt{WITHIN}\ Q_t(x, \overline{y}), C_t(x, \overline{y})$$

where:

- $\overline{x}$ and $\overline{y}$ are disjoint tuples of variables;
- $x \in \overline{x}$;
- $Q_s, Q_t \in \mathrm{CQ}$ are the source/target queries;
- C_s, C_t are sets of data predicates;
- EXCLUSIVE, MANDATORY and SINGLETON are *assertion keywords*;
- $\overline{\nu} = (\nu_1, \ldots, \nu_k)$ is a *key expression*, where for all $i \in [k]$, $\nu_i \in \{z, z.a\}$, where $z \in \{x\} \cup \overline{y}$ and $a \in \mathcal{K}$.

An mPG-Key is a PG-Key that contains the assertion keyword MANDATORY.

PG-Keys

[Ang+21]

A PG-Key is an expression of the form:

$$\texttt{FOR}\ x\ \texttt{WITHIN}\ Q_s(\overline{x}), C_s(\overline{x})\ \texttt{EXCLUSIVE} \mid \texttt{MANDATORY} \mid \texttt{SINGLETON}\ \overline{\nu}\ \texttt{WITHIN}\ Q_t(x, \overline{y}), C_t(x, \overline{y})$$

where:

- $\overline{x}$ and $\overline{y}$ are disjoint tuples of variables;
- $x \in \overline{x}$;
- $Q_s, Q_t \in \mathrm{CQ}$ are the source/target queries;
- C_s, C_t are sets of data predicates;
- EXCLUSIVE, MANDATORY and SINGLETON are *assertion keywords*;
- $\overline{\nu} = (\nu_1, \ldots, \nu_k)$ is a *key expression*, where for all $i \in [k]$, $\nu_i \in \{z, z.a\}$, where $z \in \{x\} \cup \overline{y}$ and $a \in \mathcal{K}$.

An mPG-Key is a PG-Key that contains the assertion keyword MANDATORY.

Let $G = (V, E, \eta, \lambda, \pi)$ be a graph, Q be a query and $\overline{\nu} = (\nu_1, \ldots, \nu_k)$ be a key expression. A match h for Q over G *covers* $\overline{\nu}$ if, for all $i \in [k]$:

- if ν_i is a variable x, then $x \in \mathrm{Dom}(h)$;
- if ν_i is of the form $x.a$ then $x \in \mathrm{Dom}(h)$ and $(h(x), a) \in \mathrm{Dom}(\pi)$.

Satisfaction - Appetizer, round 1

Let $G = (V, E, \eta, \lambda, \pi)$ be a graph, Q be a query and $\overline{\nu} = (\nu_1, \ldots, \nu_k)$ be a key expression. A match h for Q over G *covers* $\overline{\nu}$ if all objects of $\overline{\nu}$ have a value.

Let G be a graph and φ be a PG-Key. A *witness* for φ is a pair (h_s, h_t) such that:

- h_s is a match for (Q_s, C_s) over G;
- h_t is a match for (Q_t, C_t) over G that covers $\overline{\nu}$ and that is consistent with h_s.

Let φ be a PG-KEY with assertion keyword $\alpha \in \{\texttt{MANDATORY}, \texttt{SINGLETON}, \texttt{EXCLUSIVE}\}$. A graph G *satisfies* φ ($G \models \varphi$) if G satisfies the condition corresponding to α:

- $\alpha = \texttt{MANDATORY}$: every suitable value for x must have a suitable valuation for $\overline{\nu}$;
- $\alpha = \texttt{SINGLETON}$: every suitable value for x has at most one suitable valuation for $\overline{\nu}$;
- $\alpha = \texttt{EXCLUSIVE}$: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

Otherwise we say that G violates φ, denoted as $G \not\models \varphi$.

Let φ be a PG-KEY with assertion keyword $\alpha \in \{\texttt{MANDATORY}, \texttt{SINGLETON}, \texttt{EXCLUSIVE}\}$. A graph G *satisfies* φ ($G \models \varphi$) if G satisfies the condition corresponding to α:

- $\alpha = \texttt{MANDATORY}$: every suitable value for x must have a suitable valuation for $\overline{\nu}$;
- $\alpha = \texttt{SINGLETON}$: every suitable value for x has at most one suitable valuation for $\overline{\nu}$;
- $\alpha = \texttt{EXCLUSIVE}$: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

Otherwise we say that G violates φ, denoted as $G \not\models \varphi$.

Let φ be a PG-KEY with assertion keyword $\alpha \in \{\texttt{MANDATORY}, \texttt{SINGLETON}, \texttt{EXCLUSIVE}\}$. A graph G *satisfies* φ ($G \models \varphi$) if G satisfies the condition corresponding to α:

- $\alpha = \texttt{MANDATORY}$: every suitable value for x must have a suitable valuation for $\overline{\nu}$;
- $\alpha = \texttt{SINGLETON}$: every suitable value for x has at most one suitable valuation for $\overline{\nu}$;
- $\alpha = \texttt{EXCLUSIVE}$: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

Otherwise we say that G violates φ, denoted as $G \not\models \varphi$.

Let φ be a PG-KEY with assertion keyword $\alpha \in \{\texttt{MANDATORY}, \texttt{SINGLETON}, \texttt{EXCLUSIVE}\}$. A graph G *satisfies* φ ($G \models \varphi$) if G satisfies the condition corresponding to α:

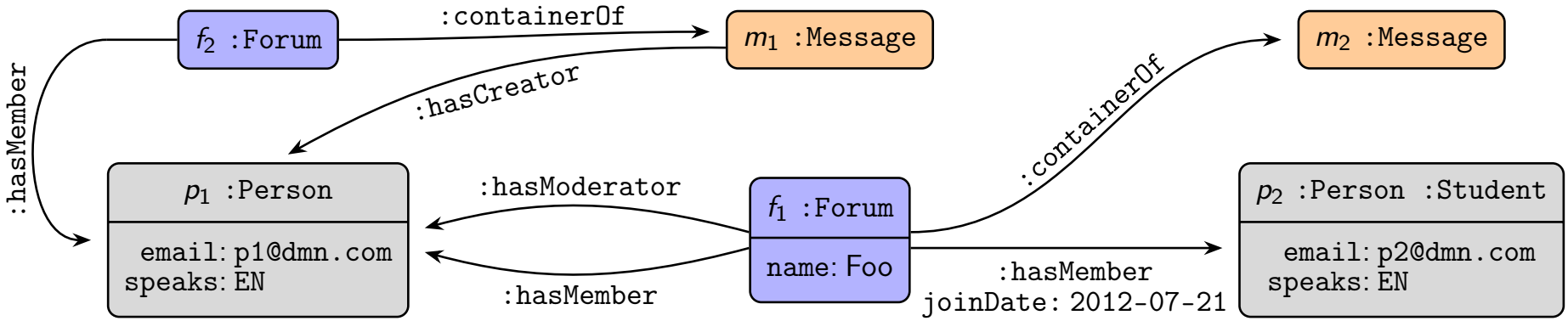
- $\alpha = \texttt{MANDATORY}$: for all matches h_s of (Q_s, C_s) over G, there exists h_t such that (h_s, h_t) is a witness for φ over G;
- $\alpha = \texttt{SINGLETON}$: for all witnesses (h_s, h_t) and (h'_s, h'_t) for φ over G, if $h_s(x) = h'_s(x)$ then $h_t(\overline{\nu}) = h'_t(\overline{\nu})$;
- $\alpha = \texttt{EXCLUSIVE}$: for all witnesses (h_s, h_t) and (h'_s, h'_t) for φ over G, if $h_t(\overline{\nu}) = h'_t(\overline{\nu})$ then $h_s(x) = h'_s(x)$.

Otherwise we say that G violates φ, denoted as $G \not\models \varphi$.

PG-Keys

Examples

$G =$



PG-Key φ_4 = `FOR z WITHIN (z:Message)`

`SINGLETON x WITHIN (z)` $\xrightarrow{\texttt{:hasCreator}}$ `(x)`

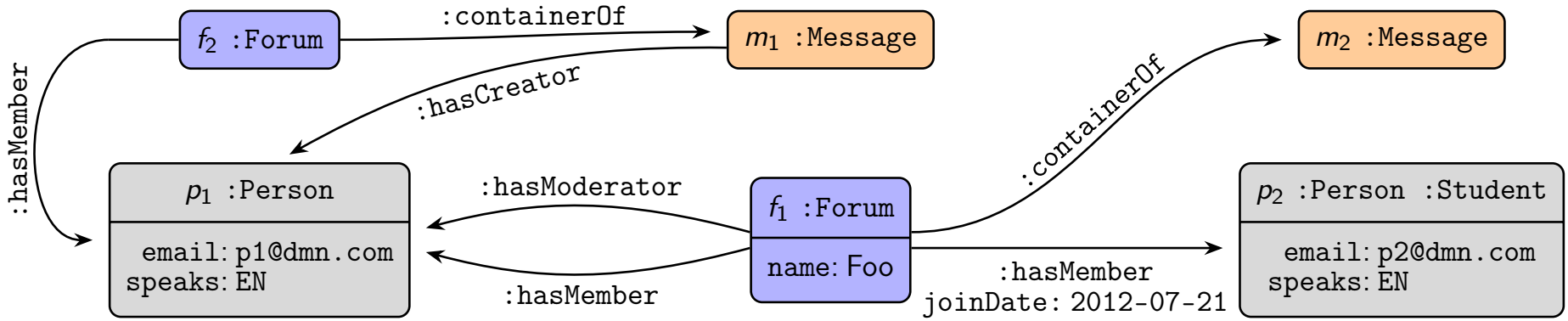
A message has at most one creator.

$G \models \varphi_4$

PG-Keys

Examples

$G =$



PG-Key φ_5 $=$ `FOR x WITHIN (:Forum)` $\xrightarrow{\texttt{x:hasMember}}$ `(:Person)`

`MANDATORY x.joinDate WITHIN ()` $\xrightarrow{\texttt{x}}$ `()`

We must know the date a person registered on a forum.

$G \not\models \varphi_5$

Outline

General results

Syntactic restrictions

For all query language:

- GFDs $\subseteq$ GGDs;
- mPG-Keys $\subseteq$ PG-Keys.

General results

GFDs vs. nGFDs

Lemma

Let $\mathscr{L} = \mathrm{CQ}$ and $\mathscr{P} = \{P_1, \ldots, P_k\}$, where $P_1, \ldots, P_k$ are predicate symbols. Then, GFDs and $n\mathrm{GFDs}$ are equivalent, where n is the maximal arity among $P_1, \ldots, P_k$.

General results

GFDs vs. nGFDs

Lemma

Let $\mathscr{L} = \mathrm{CQ}$ and $\mathscr{P} = \{P_1, \ldots, P_k\}$, where $P_1, \ldots, P_k$ are predicate symbols. Then, GFDs and $n\mathrm{GFDs}$ are equivalent, where n is the maximal arity among $P_1, \ldots, P_k$.

Proof Sketch.

Just split the GFD $(Q, C_s \Rightarrow \{d_1, \ldots, d_t\})$ into t GFDs: $\{(Q, C_s \Rightarrow \{d_1\}), \ldots, (Q, C_s \Rightarrow \{d_t\})\}$. □

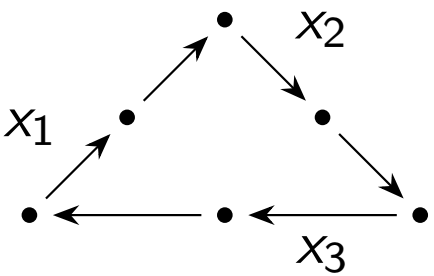
General results

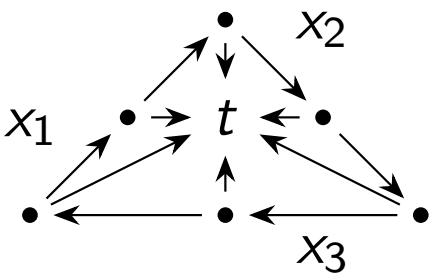
$n\mathrm{GGDs}$' hierarchy

Proposition

Let $n < m$. Then $n\mathrm{GGDs} \subsetneq m\mathrm{GGDs}$ for both $\mathrm{CQ}[\doteq]$ and $\mathrm{CQ}[\doteq, \neq]$.

Proof Sketch - with $n = 2, m = 3$.

$Q_{m\text{-cycle}} =$ 

$Q_{m\text{-middle}} =$ 

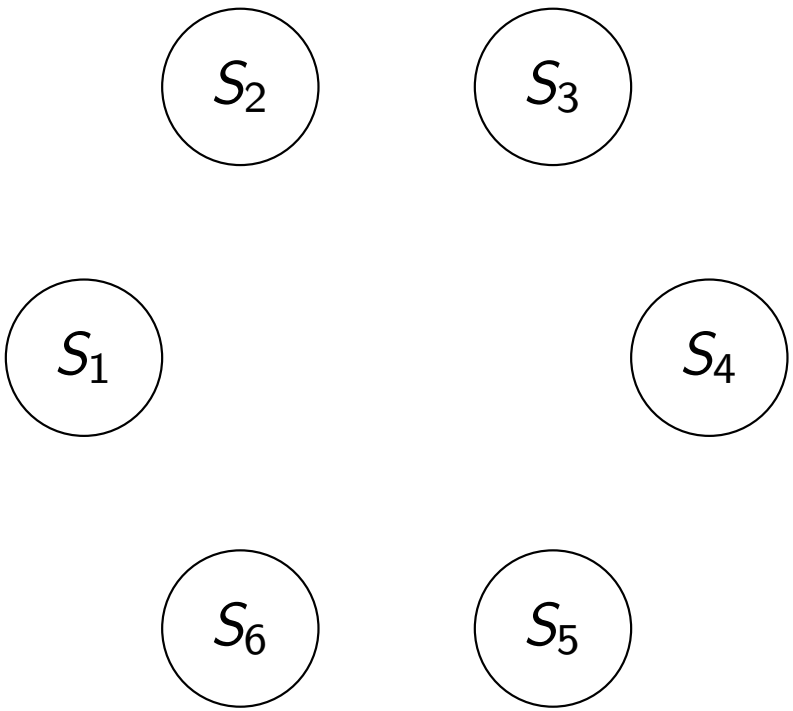
No set of $2\mathrm{GGDs}$ Ψ can express the $3\mathrm{GGD}$ $\varphi = (Q_{m\text{-cycle}}, \emptyset \Rightarrow Q_{m\text{-middle}}, \emptyset)$

General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:



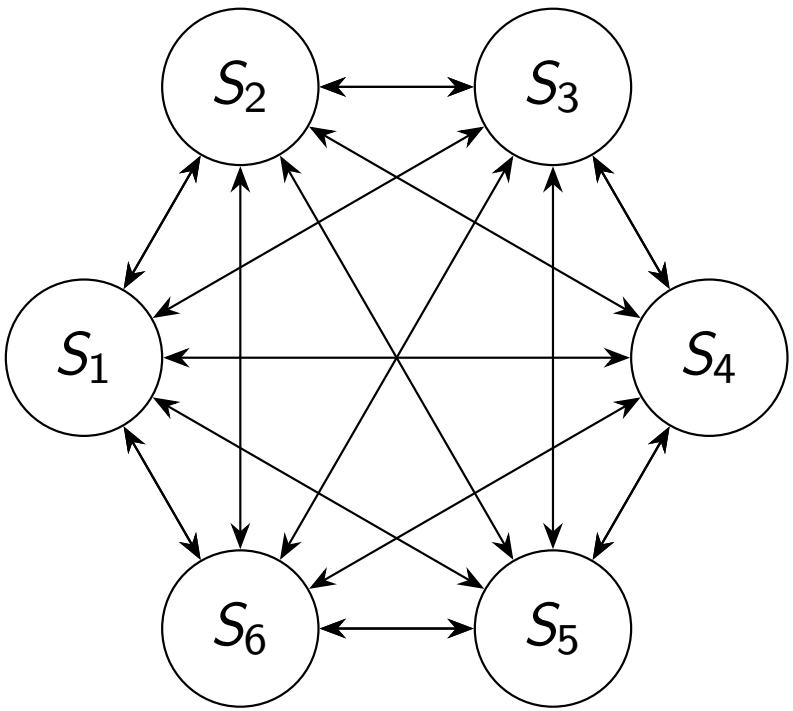
- For each $i \in [2m]$, S_i is a set of $2m + 2$ vertices.

General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

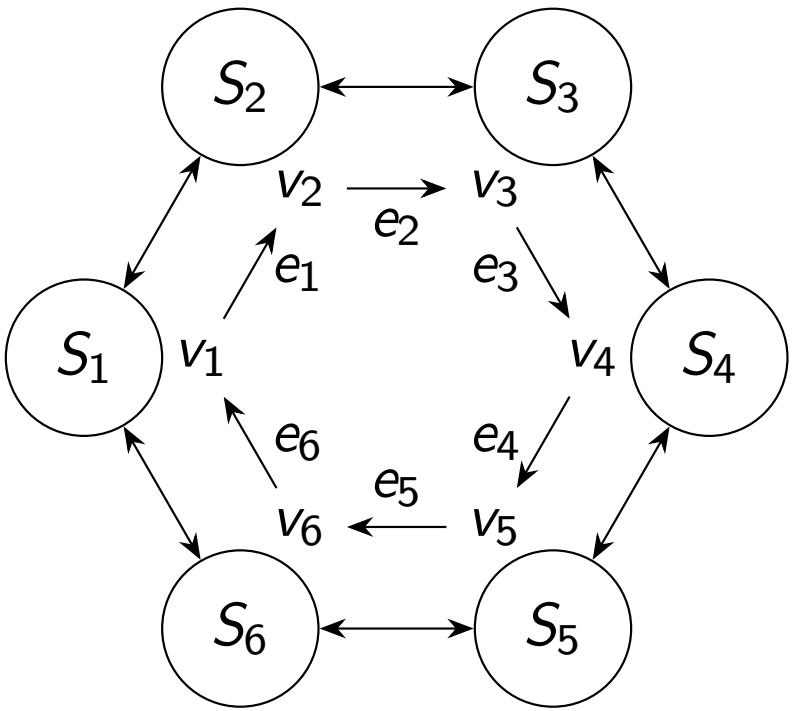
G_1 & G_2:



- For each $i \in [2m]$, S_i is a set of $2m + 2$ vertices.
- $S_1 \cup \cdots \cup S_{2m}$ is a clique.

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:



- For each $i \in [2m]$, S_i is a set of $2m + 2$ vertices.
- $S_1 \cup \cdots \cup S_{2m}$ is a clique.
- $v_1, \ldots, v_{2m}$ are vertices that do not occur in $S_1, \ldots, S_{2m}$ and that form a cycle.

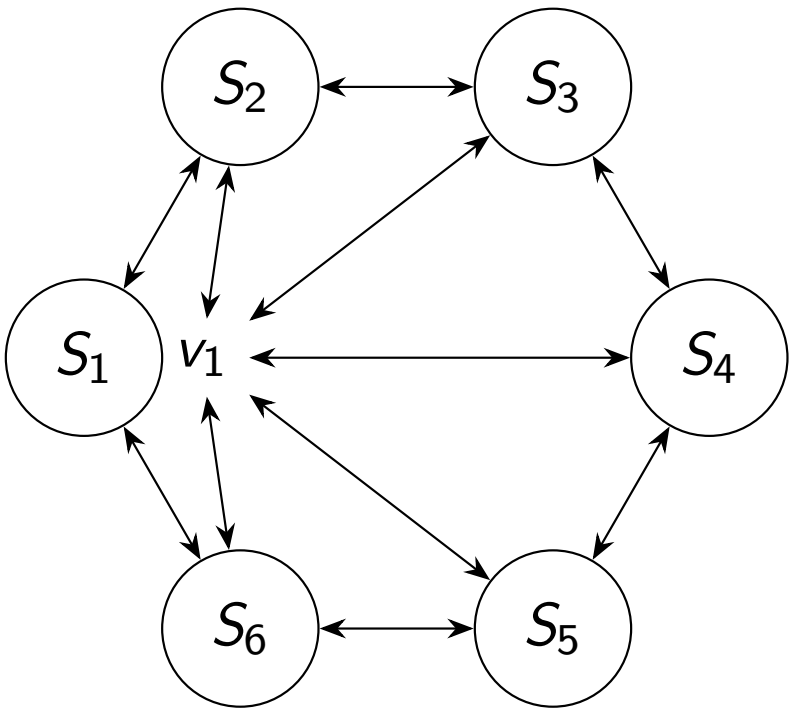
□

General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:



- For each $i \in [2m]$, S_i is a set of $2m + 2$ vertices.
- $S_1 \cup \cdots \cup S_{2m}$ is a clique.
- $v_1, \ldots, v_{2m}$ are vertices that do not occur in $S_1, \ldots, S_{2m}$ and that form a cycle.
- For each $i \in [2m]$, $S_i \cup \{v_j \mid i \neq j\}$ is a clique.

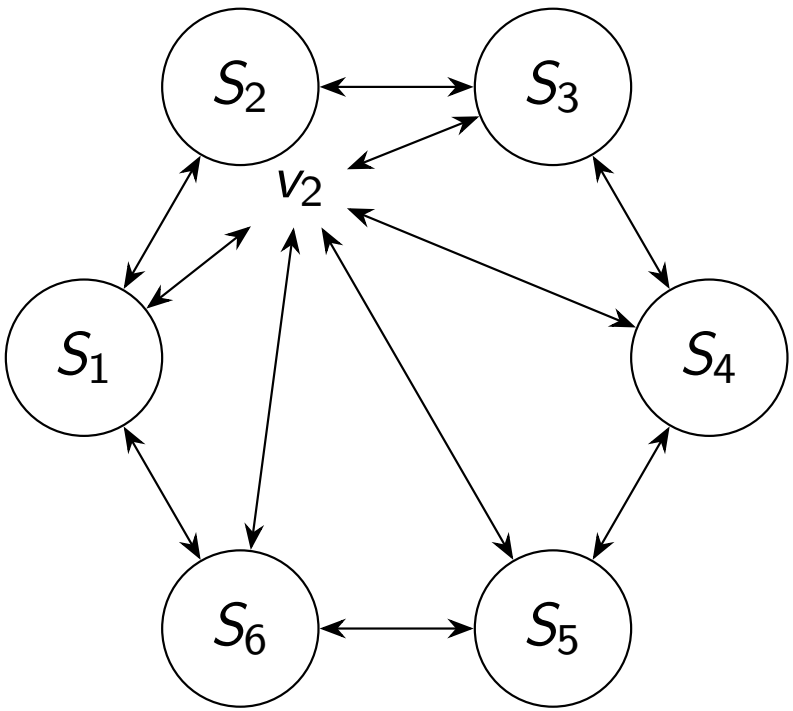
□

General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:



- For each $i \in [2m]$, S_i is a set of $2m + 2$ vertices.
- $S_1 \cup \cdots \cup S_{2m}$ is a clique.
- $v_1, \ldots, v_{2m}$ are vertices that do not occur in $S_1, \ldots, S_{2m}$ and that form a cycle.
- For each $i \in [2m]$, $S_i \cup \{v_j \mid i \neq j\}$ is a clique.

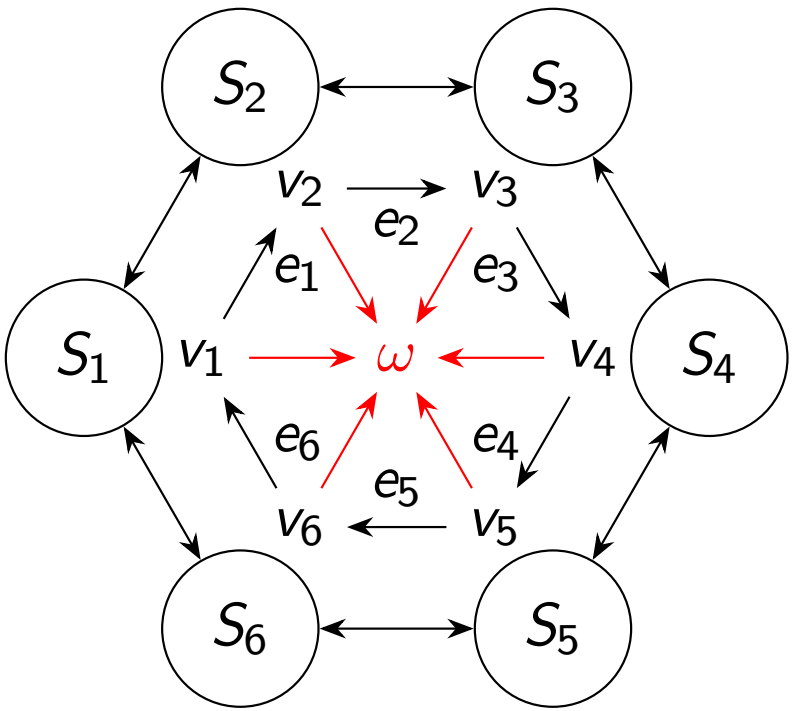
□

General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:



- For each $i \in [2m]$, S_i is a set of $2m + 2$ vertices.
- $S_1 \cup \cdots \cup S_{2m}$ is a clique.
- $v_1, \ldots, v_{2m}$ are vertices that do not occur in $S_1, \ldots, S_{2m}$ and that form a cycle.
- For each $i \in [2m]$, $S_i \cup \{v_j \mid i \neq j\}$ is a clique.
- In G_2, we add the vertex ω and its edges.

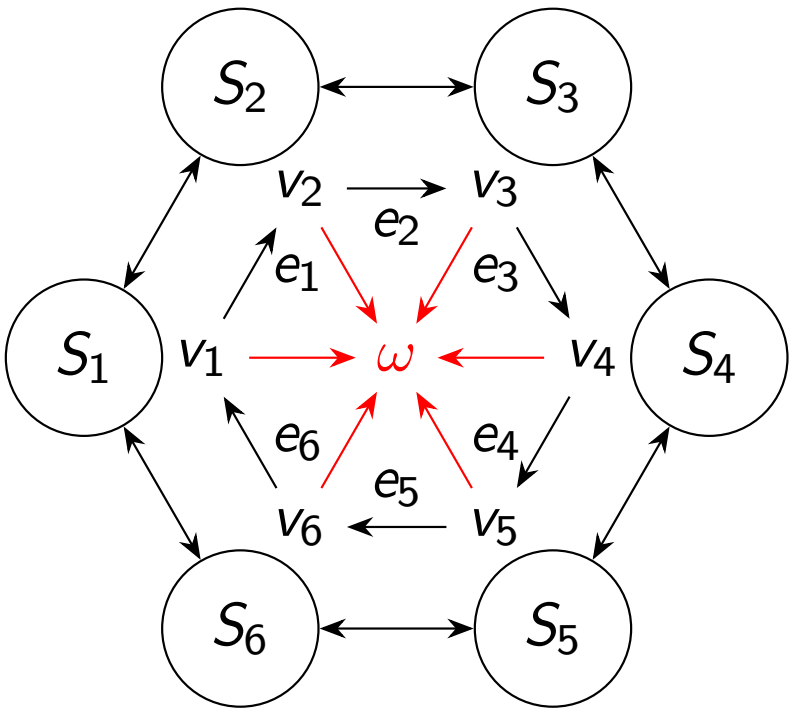
□

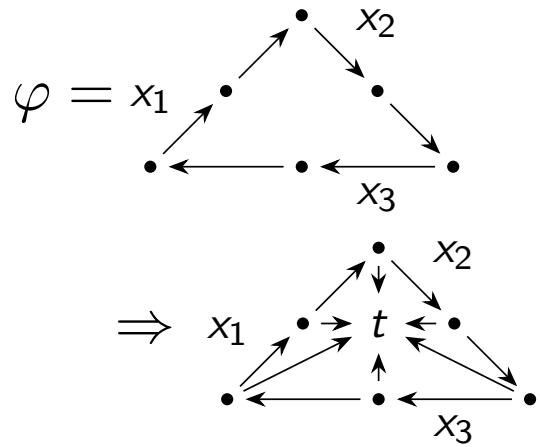
General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:





$G_1 \not\models \varphi$, $G_2 \models \varphi$:

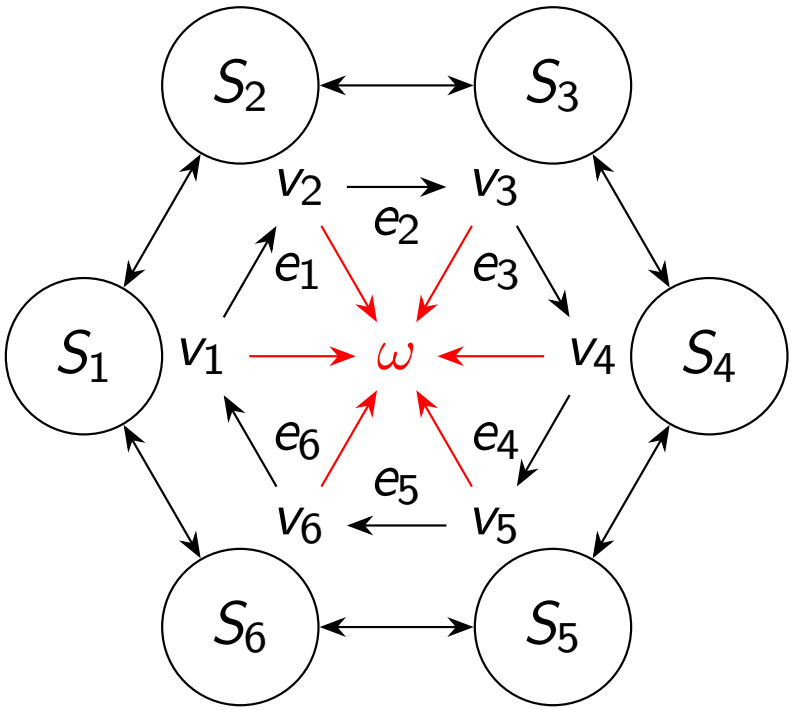
$h(x_1, x_2, x_3) = (e_2, e_4, e_6)$.

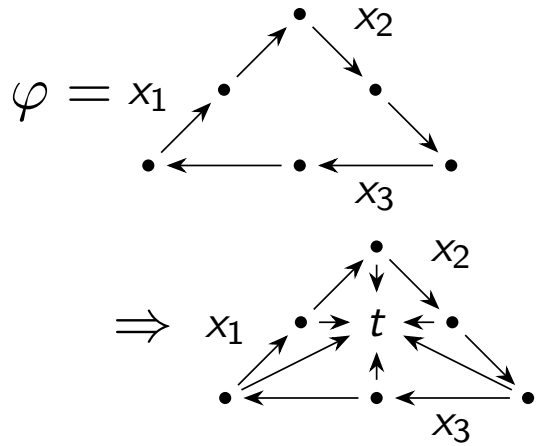
General results

nGGDs' hierarchy - cont'd

Proof Sketch - with $n = 2, m = 3$.

G_1 & G_2:





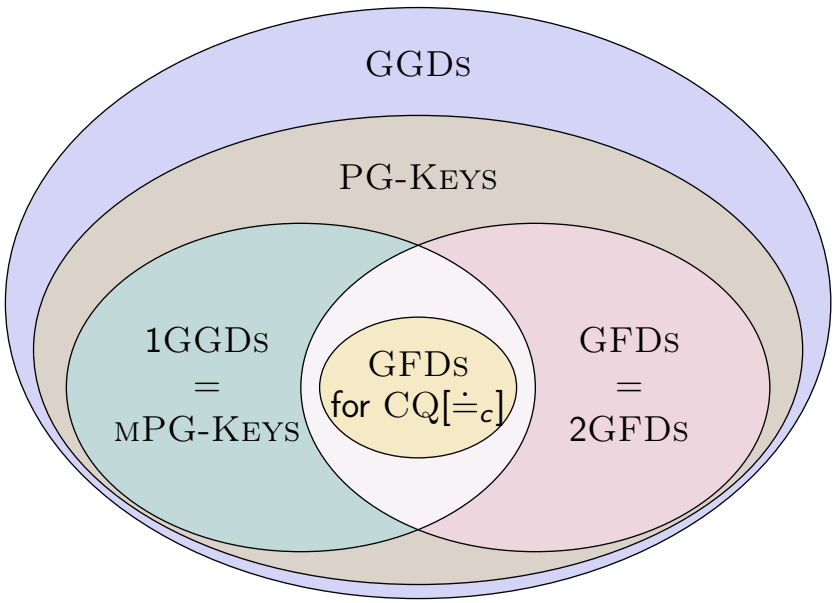
$G_1 \not\models \varphi$, $G_2 \models \varphi$:
$h(x_1, x_2, x_3) = (e_2, e_4, e_6)$.

With only two shared variables:
$G_2 \models \Psi \Rightarrow G_1 \models \Psi$.

□

$\mathrm{CQ}[\doteq]$

Spoiler

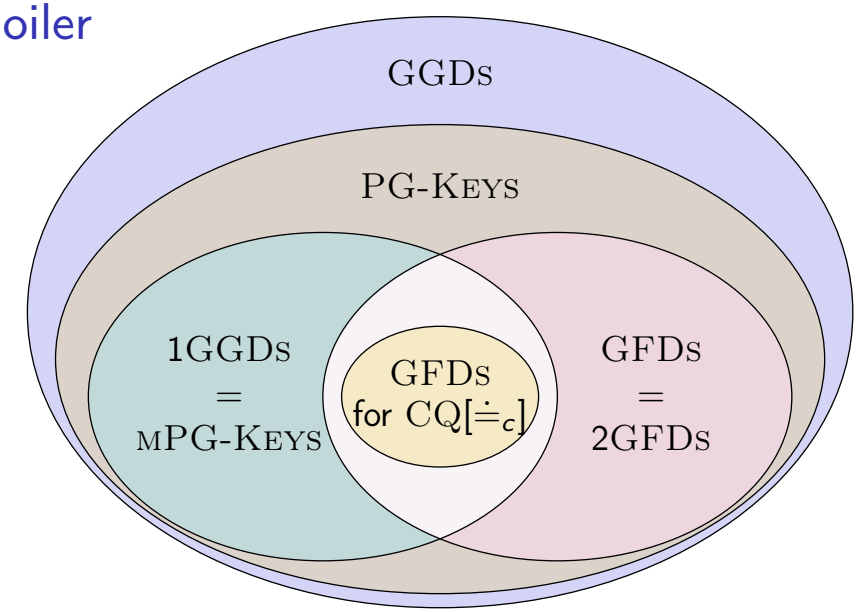


We already know that:

- $\mathrm{GFDs} \subseteq \mathrm{GGDs}$;
- $\mathrm{2GFDs} = \mathrm{GFDs}$ (since $\doteq$ is binary);
- GFDs for $\mathrm{CQ}[\doteq_c] \subseteq \mathrm{GFDs}$;
- $\mathrm{mPG\text{-}Keys} \subseteq \mathrm{PG\text{-}Keys}$.

$\mathrm{CQ}[\doteq]$

Spoiler



We already know that:

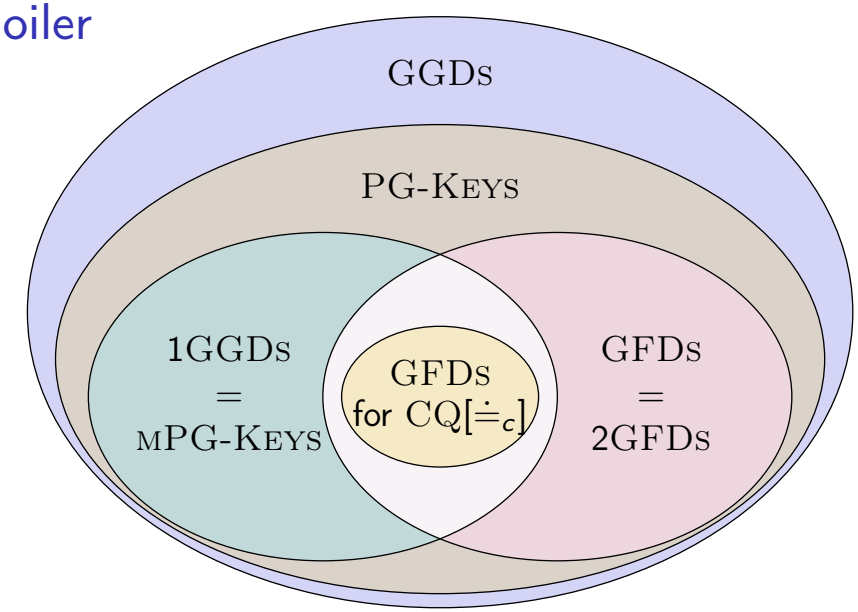
- $\mathrm{GFDs} \subseteq \mathrm{GGDs}$;
- $2\mathrm{GFDs} = \mathrm{GFDs}$ (since $\doteq$ is binary);
- GFDs for $\mathrm{CQ}[\doteq_c] \subseteq \mathrm{GFDs}$;
- $\textsc{m}\text{PG-Keys} \subseteq \text{PG-Keys}$.

What's next:

- $1\mathrm{GGDs} = \textsc{m}\text{PG-Keys}$
- $\text{PG-Keys} \subseteq \mathrm{GGDs}$
- $\text{PG-Keys} \neq \mathrm{GGDs}$ (even with connected queries)
- $\mathrm{GFDs} \subseteq \text{PG-Keys}$
- $\mathrm{GFDs} \neq 1\mathrm{GGDs}$
- $\text{PG-Keys} \neq \textsc{m}\text{PG-Keys}$
- GFDs for $\mathrm{CQ}[\doteq_c] \neq \mathrm{GFDs} \cap 1\mathrm{GGDs}$

$\mathrm{CQ}[\doteq]$

Spoiler



We already know that:

- $\mathrm{GFDs} \subseteq \mathrm{GGDs}$;
- $2\mathrm{GFDs} = \mathrm{GFDs}$ (since $\doteq$ is binary);
- GFDs for $\mathrm{CQ}[\doteq_c] \subseteq \mathrm{GFDs}$;
- $\textsc{mPG-Keys} \subseteq \textsc{PG-Keys}$.

What's next:

- ~~$1\mathrm{GGDs} = \textsc{mPG-Keys}$~~ (rewrite intuitively)
- $\textsc{PG-Keys} \subseteq \mathrm{GGDs}$: keyword `EXCLUSIVE` only
- ~~$\textsc{PG-Keys} \neq \mathrm{GGDs}$ (even with connected queries)~~
- $\mathrm{GFDs} \subseteq \textsc{PG-Keys}$
- ~~$\mathrm{GFDs} \neq 1\mathrm{GGDs}$~~ (induced subgraph & two shared variables)
- ~~$\textsc{PG-Keys} \neq \textsc{mPG-Keys}$~~ (duplicated graph)
- ~~GFDs for $\mathrm{CQ}[\doteq_c] \neq \mathrm{GFDs} \cap 1\mathrm{GGDs}$~~ (use $\doteq_d$)

$\mathrm{CQ}[\doteq]$

PG-Keys vs GGDs

We want to show that $\mathrm{PG\text{-}Keys} \subseteq \mathrm{GGDs}$.

$\mathrm{PG\text{-}Key}\ \varphi = \texttt{FOR}\ x\ \texttt{WITHIN}\ Q_s(\overline{x}), C_s(\overline{x})\ \texttt{EXCLUSIVE}\ \overline{\nu}$
$\texttt{WITHIN}\ Q_t(x, \overline{y}), C_t(x, \overline{y})$

$G \models \varphi$ iff for all witnesses (h_s, h_t), (h'_s, h'_t), if $h_t(\overline{\nu}) = h'_t(\overline{\nu})$ then $h_s(x) = h'_s(x)$[†].

$\mathrm{GGD}\ \psi = \big(Q_s(\overline{x}) \wedge Q_s(\overline{x}') \wedge Q_t(x, \overline{y}) \wedge Q_t(x', \overline{y}'),$
$C_s(\overline{x}) \cup C_s(\overline{x}') \cup C_t(x, \overline{y}) \cup C_t(x', \overline{y}') \cup \{\overline{\nu} \doteq \overline{\nu}'\}$
$\Rightarrow \emptyset, \{x \doteq x'\}\big)$

[†] intuitive version: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

We want to show that $\mathrm{PG\text{-}Keys} \subseteq \mathrm{GGDs}$.

$\mathrm{PG\text{-}Key}\ \varphi = \texttt{FOR}\ x\ \texttt{WITHIN}\ Q_s(\overline{x}), C_s(\overline{x})\ \texttt{EXCLUSIVE}\ \overline{\nu}$
$\texttt{WITHIN}\ Q_t(x, \overline{y}), C_t(x, \overline{y})$

$G \models \varphi$ iff for all witnesses (h_s, h_t), (h'_s, h'_t), if $h_t(\overline{\nu}) = h'_t(\overline{\nu})$ then $h_s(x) = h'_s(x)$[†].

$\mathrm{GGD}\ \psi = \big(Q_s(\overline{x}) \wedge Q_s(\overline{x}') \wedge Q_t(x, \overline{y}) \wedge Q_t(x', \overline{y}'),$
$C_s(\overline{x}) \cup C_s(\overline{x}') \cup C_t(x, \overline{y}) \cup C_t(x', \overline{y}') \cup \{\overline{\nu} \doteq \overline{\nu}'\}$
$\Rightarrow \emptyset, \{x \doteq x'\}\big)$

[†]intuitive version: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

We want to show that PG-Keys $\subseteq$ GGDs.

PG-Key $\varphi =$ FOR x WITHIN $Q_s(\overline{x}), C_s(\overline{x})$ EXCLUSIVE $\overline{\nu}$
WITHIN $Q_t(x, \overline{y}), C_t(x, \overline{y})$

$G \models \varphi$ iff for all witnesses (h_s, h_t), (h'_s, h'_t), if $h_t(\overline{\nu}) = h'_t(\overline{\nu})$ then $h_s(x) = h'_s(x)$[†].

GGD $\psi = \big(Q_s(\overline{x}) \wedge Q_s(\overline{x}') \wedge Q_t(x, \overline{y}) \wedge Q_t(x', \overline{y}'),$
$C_s(\overline{x}) \cup C_s(\overline{x}') \cup C_t(x, \overline{y}) \cup C_t(x', \overline{y}') \cup \{\overline{\nu} \doteq \overline{\nu}'\}$
$\Rightarrow \emptyset, \{x \doteq x'\}\big)$

[†]intuitive version: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

We want to show that PG-Keys $\subseteq$ GGDs.

PG-Key $\varphi =$ `FOR` x `WITHIN` $Q_s(\overline{x}), C_s(\overline{x})$ `EXCLUSIVE` $\overline{\nu}$
`WITHIN` $Q_t(x, \overline{y}), C_t(x, \overline{y})$

$G \models \varphi$ iff for all witnesses (h_s, h_t), (h'_s, h'_t), if $h_t(\overline{\nu}) = h'_t(\overline{\nu})$ then $h_s(x) = h'_s(x)$[†].

GGD $\psi = \big(Q_s(\overline{x}) \wedge Q_s(\overline{x}') \wedge Q_t(x, \overline{y}) \wedge Q_t(x', \overline{y}'),$
$C_s(\overline{x}) \cup C_s(\overline{x}') \cup C_t(x, \overline{y}) \cup C_t(x', \overline{y}') \cup \{\overline{\nu} \dot{=} \overline{\nu}'\}$
$\Rightarrow \emptyset, \{x \dot{=} x'\}\big)$

[†]intuitive version: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

We want to show that $\mathrm{PG\text{-}KEYS} \subseteq \mathrm{GGDS}$.

$\mathrm{PG\text{-}KEY}\ \varphi = \texttt{FOR}\ x\ \texttt{WITHIN}\ Q_s(\overline{x}), C_s(\overline{x})\ \texttt{EXCLUSIVE}\ \overline{\nu}$
$\texttt{WITHIN}\ Q_t(x, \overline{y}), C_t(x, \overline{y})$

$G \models \varphi$ iff for all witnesses (h_s, h_t), (h'_s, h'_t), if $h_t(\overline{\nu}) = h'_t(\overline{\nu})$ then $h_s(x) = h'_s(x)$[†].

$\mathrm{GGD}\ \psi = \big(Q_s(\overline{x}) \wedge Q_s(\overline{x}') \wedge Q_t(x, \overline{y}) \wedge Q_t(x', \overline{y}'),$
$C_s(\overline{x}) \cup C_s(\overline{x}') \cup C_t(x, \overline{y}) \cup C_t(x', \overline{y}') \cup \{\overline{\nu} \doteq \overline{\nu}'\}$
$\Rightarrow \emptyset, \{x \doteq x'\}\big)$

[†]intuitive version: two distinct suitable values for x cannot have a common suitable valuation for $\overline{\nu}$.

$\mathrm{CQ}[\dot{=}]$

PG-Keys vs GGDs - Example

PG-Key φ_6 = `FOR x WITHIN (x:Person)`
`EXCLUSIVE z WITHIN (z)` $\xrightarrow{\texttt{:hasCreator}}$ `(x)`

Two persons cannot write a same message.

Equivalent GGD:

$$\psi_6 = \big((x\,\texttt{:Person}) \wedge (x'\,\texttt{:Person}) \wedge (z) \xrightarrow{\texttt{:hasCreator}} (x) \wedge (z') \xrightarrow{\texttt{:hasCreator}} (x'), \{z \dot{=} z'\} \Rightarrow \emptyset, \{x \dot{=} x'\}\big)$$

$\mathrm{CQ}[\dot{=}]$

PG-Keys vs GGDs - Example

PG-Key φ_6 $=$ `FOR x WITHIN (x:Person)`
`EXCLUSIVE z WITHIN (z)` $\xrightarrow{\texttt{:hasCreator}}$ `(x)`

Two persons cannot write a same message.

Equivalent GGD:

$$\psi_6 = \big((x\,\texttt{:Person}) \wedge (x'\,\texttt{:Person}) \wedge (z) \xrightarrow{\texttt{:hasCreator}} (x) \wedge (z') \xrightarrow{\texttt{:hasCreator}} (x'), \{z \dot{=} z'\} \Rightarrow \emptyset, \{x \dot{=} x'\}\big)$$

PG-Key φ_6 $=$ `FOR x WITHIN (x:Person)`
`EXCLUSIVE z WITHIN (z)` $\xrightarrow{\texttt{:hasCreator}}$ `(x)`

Two persons cannot write a same message.

Equivalent GGD:

$$\psi_6 = \big((x\texttt{:Person}) \wedge (x'\texttt{:Person}) \wedge (z) \xrightarrow{\texttt{:hasCreator}} (x) \wedge (z') \xrightarrow{\texttt{:hasCreator}} (x'), \{z \dot{=} z'\} \Rightarrow \emptyset, \{x \dot{=} x'\}\big)$$

PG-Key φ_6 $=$ `FOR x WITHIN (x:Person)`
`EXCLUSIVE z WITHIN (z)` $\xrightarrow{\texttt{:hasCreator}}$ `(x)`

Two persons cannot write a same message.

Equivalent GGD:

$\psi_6 = \big((x\texttt{:Person}) \xleftarrow{\texttt{:hasCreator}} (z) \xrightarrow{\texttt{:hasCreator}} (x'\texttt{:Person})$
$\Rightarrow \emptyset, \{x \doteq x'\}\big)$

We want to show that $\mathrm{GFDs} \subseteq \mathrm{PG\text{-}Keys}$.
$\mathrm{GFD}\ \varphi = (Q_s(\overline{x}), C_s(\overline{x}) \Rightarrow C_t(\overline{x}'))$

We want to show that $\mathrm{GFDs} \subseteq \mathrm{PG\text{-}Keys}$.
2GFD $\varphi = (Q_s(x, y, \overline{z}), C_s(x, y, \overline{z}) \Rightarrow \{x.a \doteq y.b\})$

We want to show that $\mathrm{GFDs} \subseteq \mathrm{PG\text{-}Keys}$.
2GFD $\varphi = (Q_s(x, y, \overline{z}), C_s(x, y, \overline{z}) \Rightarrow \{x.a \doteq y.b\})$
$\Psi = \{\psi_1, \psi_2, \psi_3\}$ with:

We want to show that $\mathrm{GFDs} \subseteq \mathrm{PG\text{-}Keys}$.
2GFD $\varphi = (Q_s(x, y, \overline{z}), C_s(x, y, \overline{z}) \Rightarrow \{x.a \doteq y.b\})$
$\Psi = \{\psi_1, \psi_2, \psi_3\}$ with:

$$\psi_1 = \texttt{FOR}\ x\ \texttt{WITHIN}\ Q(x, y', \overline{z}'), C_s(x, y', \overline{z}')$$
$$\texttt{MANDATORY}\ x\ \texttt{WITHIN}\ Q(x, y, \overline{z}), C_s(x, y, \overline{z})\ \cup \{x.a \doteq y.b\}$$

For all matches h_1, h_2 for (Q, C_s) such that $h_1(x) = h_2(x)$, Ψ enforces that: $h_1(x.a) \doteq h_1(y.b)$ and $h_2(x.a) \doteq h_2(y.b)$;

$\mathrm{CQ}[\dot{=}]$

GFDs vs PG-Keys

We want to show that $\mathrm{GFDs} \subseteq \mathrm{PG\text{-}Keys}$.
2GFD $\varphi = (Q_s(x, y, \overline{z}), C_s(x, y, \overline{z}) \Rightarrow \{x.a \dot{=} y.b\})$
$\Psi = \{\psi_1, \psi_2, \psi_3\}$ with:

$\psi_1 =$ FOR x WITHIN $Q(x, y', \overline{z}'), C_s(x, y', \overline{z}')$
MANDATORY x WITHIN $Q(x, y, \overline{z}), C_s(x, y, \overline{z}) \cup \{x.a \dot{=} y.b\}$

$\psi_2 =$ FOR y WITHIN $Q(x', y, \overline{z}'), C_s(x', y, \overline{z}')$
MANDATORY y WITHIN (y), $\{y.b \dot{=} y.b\}$

For all matches h_1, h_2 for (Q, C_s) such that $h_1(x) = h_2(x)$, Ψ enforces that: $h_1(x.a) \dot{=} h_1(y.b)$ and $h_2(x.a) \dot{=} h_2(y.b)$; $h_1(y.b)$ is defined;

$\mathrm{CQ}[\doteq]$

GFDs vs PG-Keys

We want to show that GFDs $\subseteq$ PG-Keys.
2GFD $\varphi = (Q_s(x, y, \overline{z}), C_s(x, y, \overline{z}) \Rightarrow \{x.a \doteq y.b\})$
$\Psi = \{\psi_1, \psi_2, \psi_3\}$ with:

$\psi_1 =$ FOR x WITHIN $Q(x, y', \overline{z}'), C_s(x, y', \overline{z}')$
MANDATORY x WITHIN $Q(x, y, \overline{z}), C_s(x, y, \overline{z}) \cup \{x.a \doteq y.b\}$

$\psi_2 =$ FOR y WITHIN $Q(x', y, \overline{z}'), C_s(x', y, \overline{z}')$
MANDATORY y WITHIN (y), $\{y.b \doteq y.b\}$

$\psi_3 =$ FOR x WITHIN $Q(x, y', \overline{z}'), C_s(x, y', \overline{z}')$
SINGLETON $y.b$ WITHIN $Q(x, y, \overline{z}), C_s(x, y, \overline{z})$

For all matches h_1, h_2 for (Q, C_s) such that $h_1(x) = h_2(x)$, Ψ enforces that: $h_1(x.a) \doteq h_1(y.b)$ and $h_2(x.a) \doteq h_2(y.b)$; $h_1(y.b)$ is defined; $h_1(y.b) \doteq h_2(y.b)$.

CQ[$\dot{=}$]

GFDs vs PG-Keys - Example

GFD $\varphi_7 = (Q(\mathtt{x},\mathtt{y},\mathtt{z}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\})$

with $Q(\mathtt{x},\mathtt{y},\mathtt{z}) = ((\mathtt{x}) \xleftarrow{\texttt{:hasMember}} (\mathtt{z}) \xrightarrow{\texttt{:hasMember}} (\mathtt{y}))$

Every member of a same forum must speak the same language.

Equivalent set of PG-Keys: $\Psi_7 = \{\psi_{7,1}, \psi_{7,2}, \psi_{7,3}\}$ with:

GFD $\varphi_7 = (Q(\mathtt{x},\mathtt{y},\mathtt{z}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\})$
with $Q(\mathtt{x},\mathtt{y},\mathtt{z}) = ((\mathtt{x}) \xleftarrow{\texttt{:hasMember}} (\mathtt{z}) \xrightarrow{\texttt{:hasMember}} (\mathtt{y}))$
Every member of a same forum must speak the same language.
Equivalent set of PG-Keys: $\Psi_7 = \{\psi_{7,1}, \psi_{7,2}, \psi_{7,3}\}$ with:

GFD $\varphi_7 = (Q(\mathtt{x},\mathtt{y},\mathtt{z}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\})$
with $Q(\mathtt{x},\mathtt{y},\mathtt{z}) = ((\mathtt{x}) \xleftarrow{\texttt{:hasMember}} (\mathtt{z}) \xrightarrow{\texttt{:hasMember}} (\mathtt{y}))$
Every member of a same forum must speak the same language.
Equivalent set of PG-Keys: $\Psi_7 = \{\psi_{7,1}, \psi_{7,2}, \psi_{7,3}\}$ with:

CQ[$\dot{=}$]

GFDs vs PG-Keys - Example

GFD $\varphi_7 = (Q(\mathtt{x},\mathtt{y},\mathtt{z}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\})$
with $Q(\mathtt{x},\mathtt{y},\mathtt{z}) = ((\mathtt{x}) \xleftarrow{:\mathtt{hasMember}} (\mathtt{z}) \xrightarrow{:\mathtt{hasMember}} (\mathtt{y}))$
Every member of a same forum must speak the same language.
Equivalent set of PG-Keys: $\Psi_7 = \{\psi_{7,1}, \psi_{7,2}, \psi_{7,3}\}$ with:

$$\psi_{7,1} = \texttt{FOR x WITHIN } Q(\mathtt{x},\mathtt{y'},\mathtt{z'})$$
$$\texttt{MANDATORY x WITHIN } Q(\mathtt{x},\mathtt{y},\mathtt{z}), \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\}$$

Every x has a "speaks" key and some member y sharing a forum with x speaks the same language

CQ[$\dot{=}$]

GFDs vs PG-Keys - Example

GFD $\varphi_7 = (Q(\mathtt{x},\mathtt{y},\mathtt{z}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\})$
with $Q(\mathtt{x},\mathtt{y},\mathtt{z}) = ((\mathtt{x}) \xleftarrow{\texttt{:hasMember}} (\mathtt{z}) \xrightarrow{\texttt{:hasMember}} (\mathtt{y}))$
Every member of a same forum must speak the same language.
Equivalent set of PG-Keys: $\Psi_7 = \{\psi_{7,1}, \psi_{7,2}, \psi_{7,3}\}$ with:

$\psi_{7,1} =$ FOR x WITHIN $Q(\mathtt{x},\mathtt{y'},\mathtt{z'})$
MANDATORY x WITHIN $Q(\mathtt{x},\mathtt{y},\mathtt{z}), \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\}$

$\psi_{7,2} =$ FOR y WITHIN $Q(\mathtt{x},\mathtt{y},\mathtt{z})$
MANDATORY y WITHIN $(\mathtt{y}), \{\mathtt{y.speaks} \doteq \mathtt{y.speaks}\}$

Every x has a "speaks" key and some member y sharing a forum with x speaks the same language, every such member y has a "speaks" key

CQ[$\doteq$]

GFDs vs PG-Keys - Example

GFD $\varphi_7 = (Q(\mathtt{x},\mathtt{y},\mathtt{z}), \emptyset \Rightarrow \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\})$
with $Q(\mathtt{x},\mathtt{y},\mathtt{z}) = ((\mathtt{x}) \xleftarrow{\texttt{:hasMember}} (\mathtt{z}) \xrightarrow{\texttt{:hasMember}} (\mathtt{y}))$
Every member of a same forum must speak the same language.
Equivalent set of PG-Keys: $\Psi_7 = \{\psi_{7,1}, \psi_{7,2}, \psi_{7,3}\}$ with:

$\psi_{7,1} =$ FOR x WITHIN $Q(\mathtt{x},\mathtt{y'},\mathtt{z'})$
MANDATORY x WITHIN $Q(\mathtt{x},\mathtt{y},\mathtt{z}), \{\mathtt{x.speaks} \doteq \mathtt{y.speaks}\}$

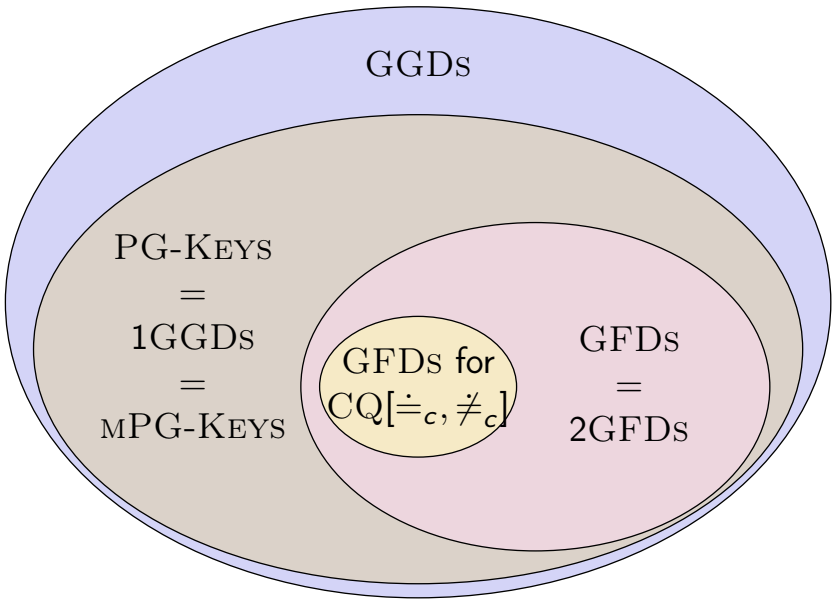
$\psi_{7,2} =$ FOR y WITHIN $Q(\mathtt{x},\mathtt{y},\mathtt{z})$
MANDATORY y WITHIN $(\mathtt{y}), \{\mathtt{y.speaks} \doteq \mathtt{y.speaks}\}$

$\psi_{7,3} =$ FOR x WITHIN $Q(\mathtt{x},\mathtt{y'},\mathtt{z'})$
SINGLETON y.speaks WITHIN $Q(\mathtt{x},\mathtt{y},\mathtt{z})$

Every x has a "speaks" key and some member y sharing a forum with x speaks the same language, every such member y has a "speaks" key, and all these members speak the same language.

$\mathrm{CQ}[\doteq, \neq]$

Spoiler



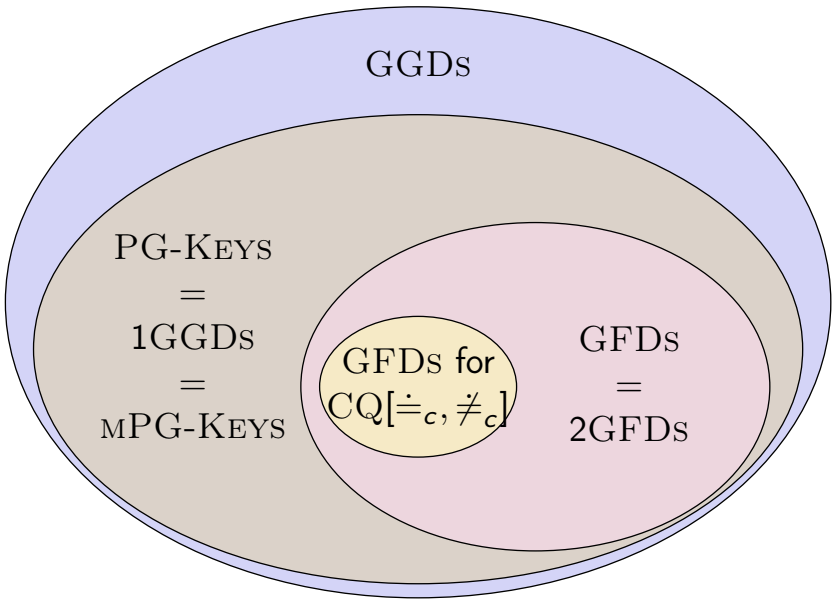
Other cases are similar.

What changes:

- $\mathrm{1GGDs} = \mathrm{mPG\text{-}Keys} = \mathrm{PG\text{-}Keys}$;
- $\mathrm{GFDs} \subsetneq \mathrm{1GGDs}$.

$\mathrm{CQ}[\doteq, \neq]$

Spoiler



Other cases are similar.

What changes:

- ~~1GGDs = mPG-Keys = PG-Keys~~: by an example;
- ~~GFDs $\subseteq$ 1GGDs~~.

$\mathrm{CQ}[\doteq, \neq]$

mPG-Keys vs PG-Keys - Example

$$\text{PG-Key } \varphi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\texttt{SINGLETON x WITHIN (x)} \xleftarrow{\texttt{:hasCreator}} \texttt{(z)}$$

Each message has a unique creator

$$\text{PG-Key } \varphi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\texttt{SINGLETON x WITHIN (x)} \xleftarrow{\texttt{:hasCreator}} \texttt{(z)}$$

Each message has a unique creator, i.e. a message cannot have two different creators.

$$\text{PG-Key } \varphi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\texttt{SINGLETON x WITHIN (x)} \xleftarrow{\texttt{:hasCreator}} \texttt{(z)}$$

Each message has a unique creator, i.e. a message cannot have two different creators.

$$\text{mPG-Key } \psi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\wedge\ \texttt{(z)} \xrightarrow{\texttt{:hasCreator}} \texttt{(x)}$$
$$\wedge\ \texttt{(z)} \xrightarrow{\texttt{:hasCreator}} \texttt{(x')}, \{\texttt{x} \neq \texttt{x'}\}$$
$$\texttt{MANDATORY z WITHIN (z)}, \{\texttt{z} \neq \texttt{z}\}$$

$$\text{PG-Key } \varphi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\texttt{SINGLETON x WITHIN (x)} \xleftarrow{\texttt{:hasCreator}} \texttt{(z)}$$

Each message has a unique creator, i.e. a message cannot have two different creators.

$$\text{mPG-Key } \psi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\wedge\ \texttt{(z)} \xrightarrow{\texttt{:hasCreator}} \texttt{(x)}$$
$$\wedge\ \texttt{(z)} \xrightarrow{\texttt{:hasCreator}} \texttt{(x')}, \{\texttt{x} \neq \texttt{x'}\}$$
$$\texttt{MANDATORY z WITHIN (z)}, \{\texttt{z} \neq \texttt{z}\}$$

Check for matches that do not share the same value for x.

$$\text{PG-Key } \varphi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\texttt{SINGLETON x WITHIN (x)} \xleftarrow{\texttt{:hasCreator}} \texttt{(z)}$$

Each message has a unique creator, i.e. a message cannot have two different creators.

$$\text{mPG-Key } \psi_8 = \texttt{FOR z WITHIN (z:Message)}$$
$$\wedge\ \texttt{(z)} \xrightarrow{\texttt{:hasCreator}} \texttt{(x)}$$
$$\wedge\ \texttt{(z)} \xrightarrow{\texttt{:hasCreator}} \texttt{(x')}, \{\texttt{x} \dot{\neq} \texttt{x'}\}$$
$$\texttt{MANDATORY z WITHIN (z)}, \{\texttt{z} \dot{\neq} \texttt{z}\}$$

Check for matches that do not share the same value for x.
Similar trick if the keyword is EXCLUSIVE.

References

[Ang+20] Renzo Angles et al. "The LDBC Social Network Benchmark". In: *CoRR* abs/2001.02299 (2020).

[Ang+21] Renzo Angles et al. "PG-Keys: Keys for Property Graphs". In: *SIGMOD Conference*. ACM, 2021, pp. 2423–2436.

[FWX16] Wenfei Fan, Yinghui Wu, and Jingbo Xu. "Functional Dependencies for Graphs". In: *SIGMOD Conference*. ACM, 2016, pp. 1843–1857.

[SFY20] Larissa Capobianco Shimomura, George Fletcher, and Nikolay Yakovets. "GGDs: Graph Generating Dependencies". In: *CIKM*. ACM, 2020, pp. 2217–2220.