VUES LOGIQUE ET GRAPHE DES FAITS ET DES REQUÊTES CONJONCTIVES

HAI933I

Rappels du cours précédent

FACTBASE

Vocabulary : $(\mathcal{P},\mathcal{C})$ where $\mathcal{P}$ is a finite set of predicates
$\mathcal{C}$ is a possibly infinite set of constants

[Arity of a predicate = its number of arguments]

$\mathcal{P}$ = { Prof/1, PHS/1, involvedIn/2, ...}
$\mathcal{C}$ = {Bob, #1, 456, ...}

Fact : a **ground** atom p(e1 ... ek) with p ∈ $\mathcal{P}$ and ei ∈ $\mathcal{C}$ [ground = no variables]

involvedIn(Bob,#1)

Factbase : usually a set of **ground atoms** on the vocabulary

F = { Prof(Bob), PHS(#1), involvedIn(Bob,#1) }

logically seen as the **conjunction** of these atoms

Prof(Bob) ∧ PHS(#1) ∧ involvedIn(Bob,#1)

Factbases can be extended to unknown values

An unknown value is logically seen as an **existentially quantified variable**
Then a factbase is logically seen as the existential closure of the conjunction of its atoms

Relational database

Movie

m_id	
m1	…
m2	…
?x	

Actor

a_id	
a	…
b	…
c	

Play

m_id	a_id
a	m1
a	m2
c	?x

Factbase

$$\{\ \mathbf{movie(m1), movie(m2), movie(}x\mathbf{), actor(a), actor(b), actor(c), play(a,m1), play(a,m2), play(c,}x\mathbf{)}\ \}$$

Logical formula assigned to the factbase

$$\exists x\ (\ \mathrm{movie}(m1) \wedge \mathrm{movie}(m2) \wedge \mathrm{movie}(x)\ \ \mathrm{actor}(a) \wedge \mathrm{actor}(b) \wedge \mathrm{actor}(c)\ \ \mathrm{play}(a,m1) \wedge \mathrm{play}(a,m2) \wedge \mathrm{play}(c,x)\ )$$

Conjunctive Queries (CQ)

$q(x) = \exists y\,(movie(y) \wedge play(x, y))$ *« find all those who play in a movie »*

$q() \;= \exists y\,(movie(y) \wedge play(b, y))$ *« does* b *play in a movie ? » (*b *is a constant)*

A **CQ** is an **existentially quantified conjunction of atoms**

The **free variables** are the **answer variables**

If closed formula: **Boolean CQ**

Simplified notation

$q(x) = \{\, movie(y), play(x,y) \,\}$

Rule notation

$ans(x) \leftarrow movie(y), play(x, y)$ classical Datalog notation

$movie(y), play(x, y) \rightarrow ans(x)$ alternative notation

Basic SQL queries (on relational databases)

SELECT ... FROM … WHERE *<equalities: restrictions and joins>*

Basic SPARQL (on RDF triples)

SELECT … WHERE *<basic graph pattern>*

ANSWERS TO A CONJUNCTIVE QUERY

- The answer to a BCQ Q in F is *yes* if F ⊨ Q

  yes = ()

- A tuple $(a_1, \ldots, a_k)$ of *constants* is an answer to $Q(x_1,\ldots,x_k)$ with respect to F

if $F \models Q[a_1,\ldots,a_k]$, where $Q[a_1,\ldots,a_k]$ is obtained from $Q(x_1,\ldots,x_k)$ by replacing each x_i by a_i.

- Let F and Q be seen as sets of atoms. A homomorphism *h* from Q to F is a mapping from *variables*(Q) to *terms*(F) such that $h(Q) \subseteq F$

$F \models Q()$ iff Q can be mapped by homomorphism to F

$(a_1, \ldots, a_k)$ is an answer to $Q(x_1,\ldots,x_k)$ on F iff there is a homomorphism from Q to F that maps each x_i to a_i

FRAGMENT EXISTENTIEL CONJONCTIF POSITIF : FOL($\exists$,$\wedge$)

Formules construites avec le quantificateur existentiel ($\exists$) et la conjonction ($\wedge$)

- Forme normalisée (« prénexe ») :

  $\exists x_1 \ldots \exists x_n (A_1 \wedge \ldots A_p)$ où les A_i sont des atomes
  et chaque x_j apparait dans un A_i

- Permettent de représenter des bases de faits (et bases de données relationnelles) et des requêtes conjonctives
- Pour des formules closes :

  f1 $\models$ *f2* ssi il existe un homomorphisme de *f1* dans *f2*

Dans ce cours :

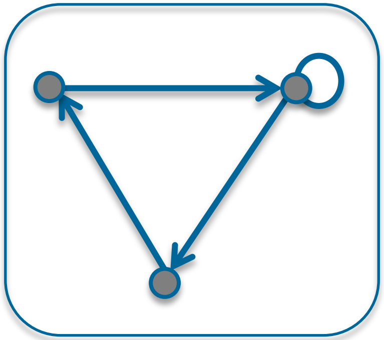
- Vision « graphe » de ces formules (=> homomorphisme de graphe)
- Notion de minimalité : peut-on supprimer des atomes en gardant une formule équivalente ? Deux formules équivalentes sont-elles « identiques » ?

- On note G = (V,E) un graphe orienté où V est l'ensemble des sommets (*vertices*)
  et E est l'ensemble des arcs (*edges*)

- Un ensemble d'atomes avec un seul prédicat binaire et sans constantes peut être vu comme un graphe orienté

(et réciproquement)

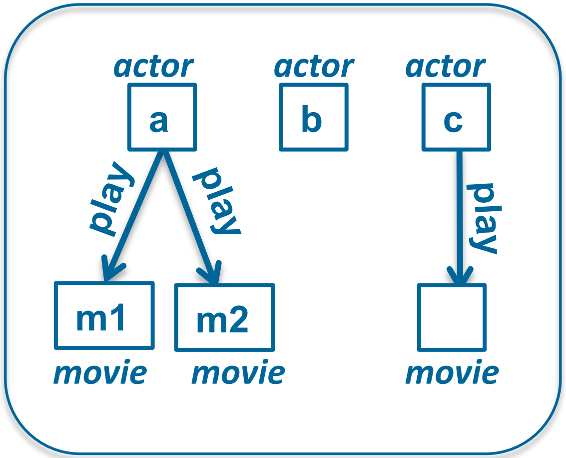
{ p(x,y), p(y,z), p(z,x), p(y,y) }



bijections : termes (variables ici) → V
atomes → E

- On étiquette : les arcs si on a plusieurs prédicats binaires
  les sommets si on a des constantes
- On peut ajouter un 2ème type d'étiquette pour représenter les prédicats unaires

movie(m1), movie(m2), movie(x), actor(a), actor(b), actor(c), play(a,m1), play(a,m2), play(c,x)



Et si on a des prédicats d'arité supérieure à 2 ?

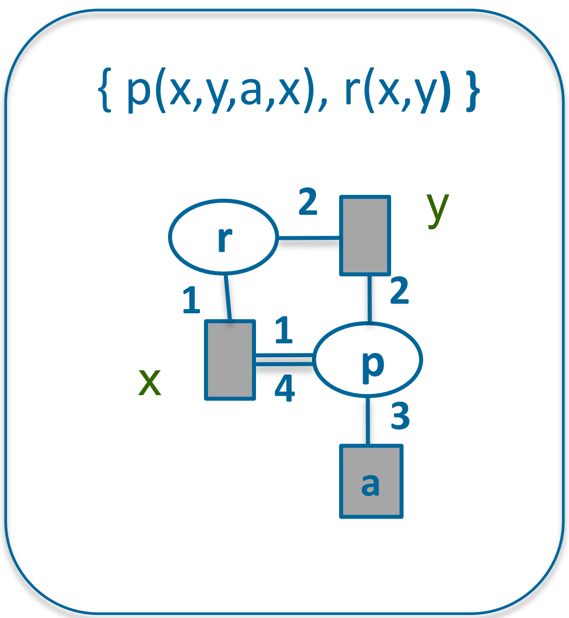
ENSEMBLE D'ATOMES ENCODÉ PAR UN GRAPHE

- À un ensemble d'atomes, on associe naturellement un hypergraphe « orienté ».

  La notion d'hyperarc généralise la notion d'arc : n-uplet (n > 0) de sommets
- Il est pratique de considérer le graphe d'incidence associé à l'hypergraphe.
  C'est un multi-graphe biparti non-orienté

  « biparti » : l'ensemble des sommets est partionné en 2 classes, tel qu'il n'y a aucun arc entre 2 sommets de la même classe

  « multi-graphe » : il peut y avoir plusieurs arêtes entre deux sommets



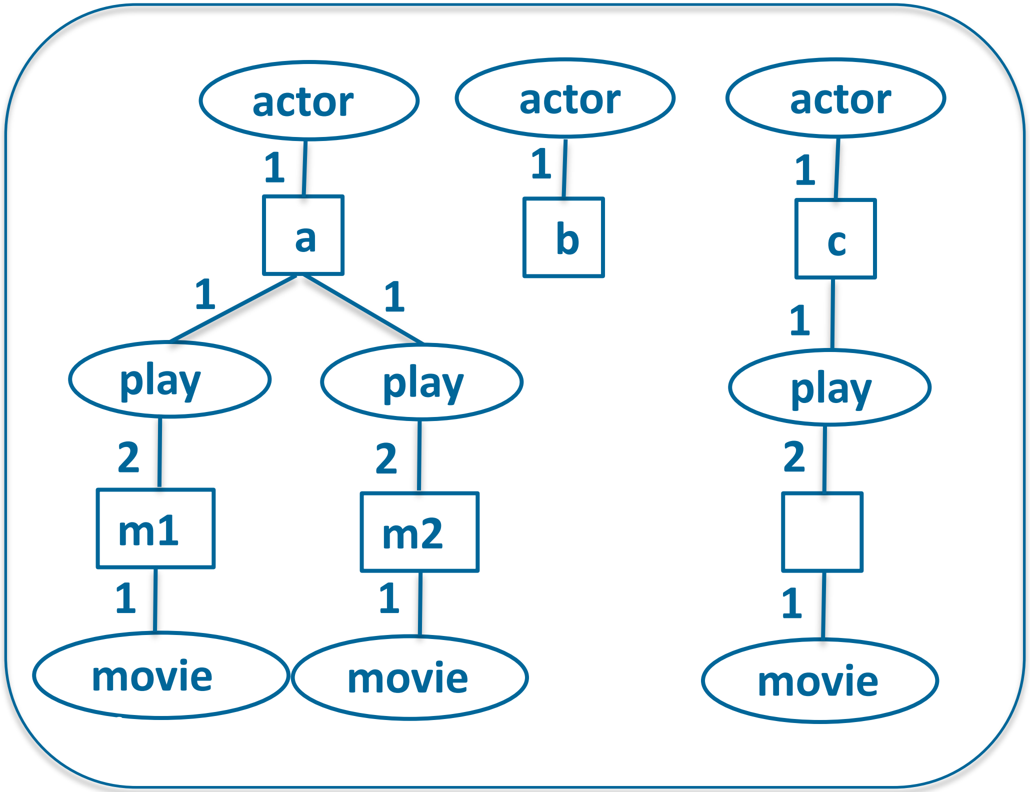
- 1 sommet par terme
  étiqueté par le terme si c'est une constante
- 1 sommet par atome
  étiqueté par le prédicat de l'atome
- les arêtes lient chaque sommet atome aux sommets qui représentent ses arguments
- les arêtes incidentes à un sommet atome sont totalement ordonnées (ce qu'on peut représenter par une numérotation)

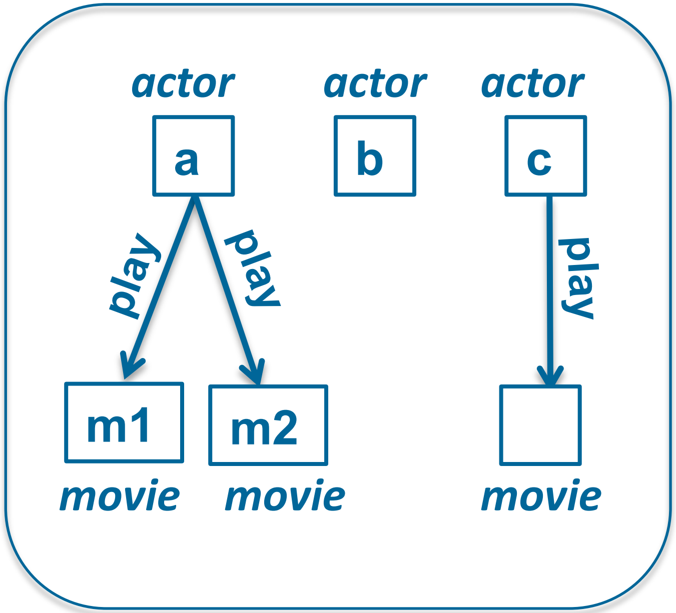
Plus précisément :

À un ensemble d'atomes F, on associe un (multi-)graphe biparti (V_T, V_A, E, label) tel que :

- V_T : ensemble des sommets termes
  (on a une bijection b_T de l'ensemble des termes de F vers V_T)
- **V_A** : ensemble des sommets atomes
  (on a une bijection b_A de l'ensemble des atomes de F vers V_A)
- **E** : multi-ensemble des arêtes : pour chaque atome A = p(t_1, ..., t_k) de F,
  on a k arêtes entre b_A(A) et chacun des b_T(t_i)
- **label** : fonction d'étiquetage qui vérifie :
  - tout sommet terme b_T(t) est étiqueté par t si t est une constante, sinon il n'est pas étiqueté
  - tout sommet atome b_A(A) avec A = p(t_1, ..., t_k) est étiqueté par p
    et chaque arête (b_A(A), b_T(t_i)) est étiquetée par i

movie(m1), movie(m2), actor(a), actor(b), actor(c),
play(a,m1), play(a,m2), movie(x), play(c,x)





graphe plus simple
(car prédicats d'arité ≤ 2)

graphe biparti associé à la vision « hypergraphe »

Isomorphism of sets of atoms / Graphs

- Let *f* and *g* be sets of atoms
  Isomorphism *h* from *f* to *g*: bijective mapping from *var(f)* to *var(g)*
  such that *h(f) = g*

  When *f* and *g* are isomorphic :
  we also say that *f* and *g* are ``equal up to a bijective variable renaming''

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs
  Isomorphism *h* from G_1 to G_2 : bijective mapping from V_1 to V_2
  such that for all vertices u,v in V_1,
  (u,v) is in E_1 if and only if (h(u),h(v)) is in E_2

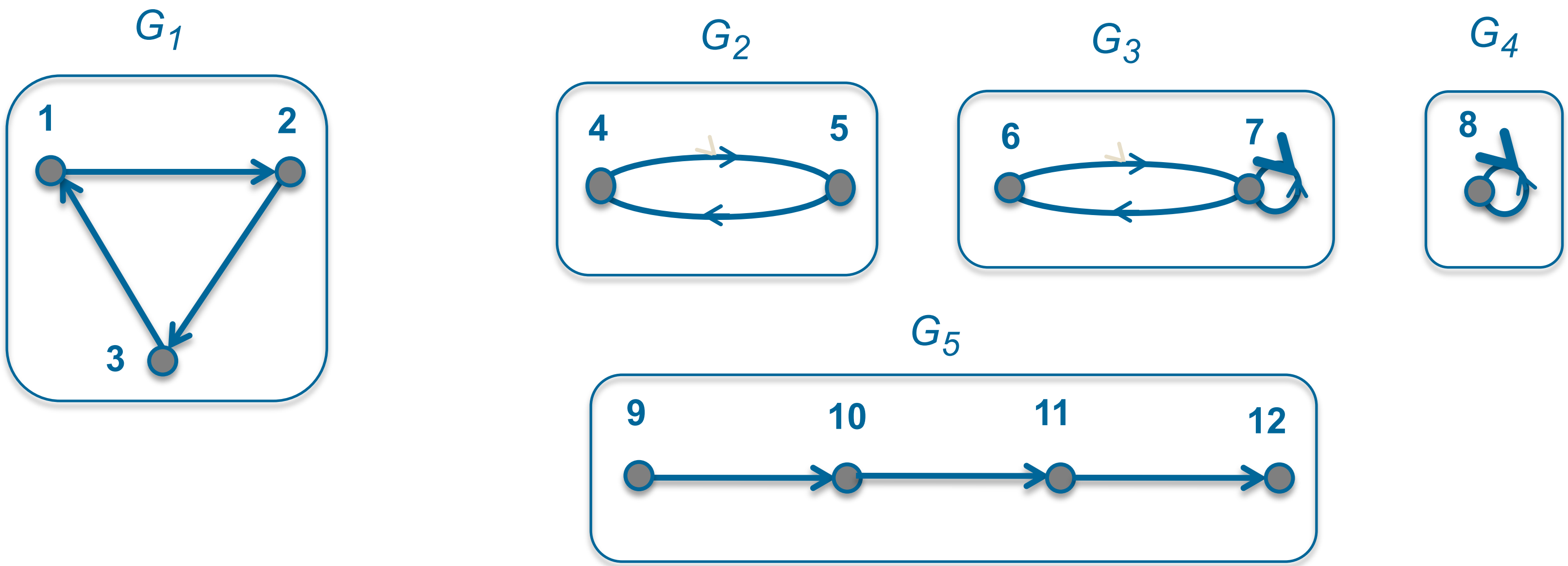
  If the graphs are labeled:
  label(u) = label(h(u)) for all u in V_1
  label ((u,v)) = label(h(u),h(v)) for all (u,v) in E_1

Les transformations précédentes assurent que deux ensembles d'atomes sont transformés en des graphes isomorphes **ssi** ils sont isomorphes

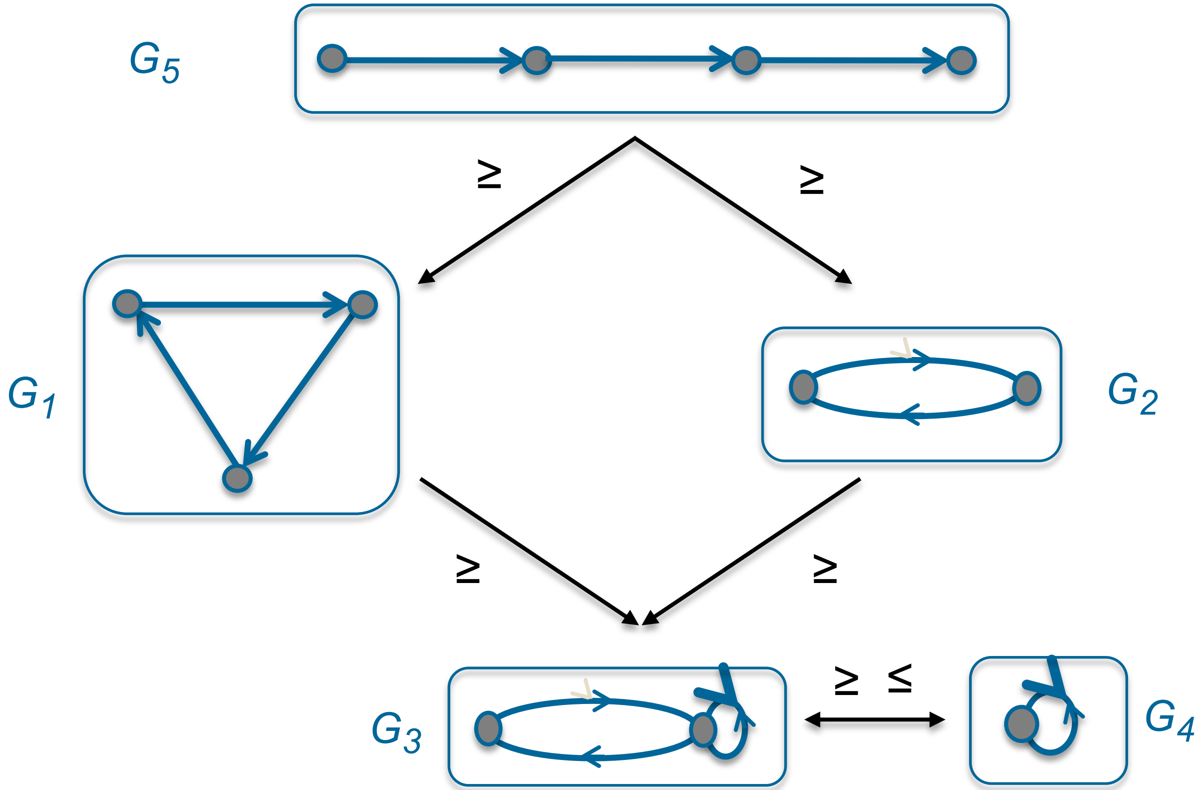
Graph homomorphisms (1)

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs.

  Homomorphism *h* from G_1 to G_2: mapping from V_1 to V_2 s. t.
  for every edge (u,v) in E_1, (h(u),h(v)) is in E_2



Find the homomorphisms between these graphs



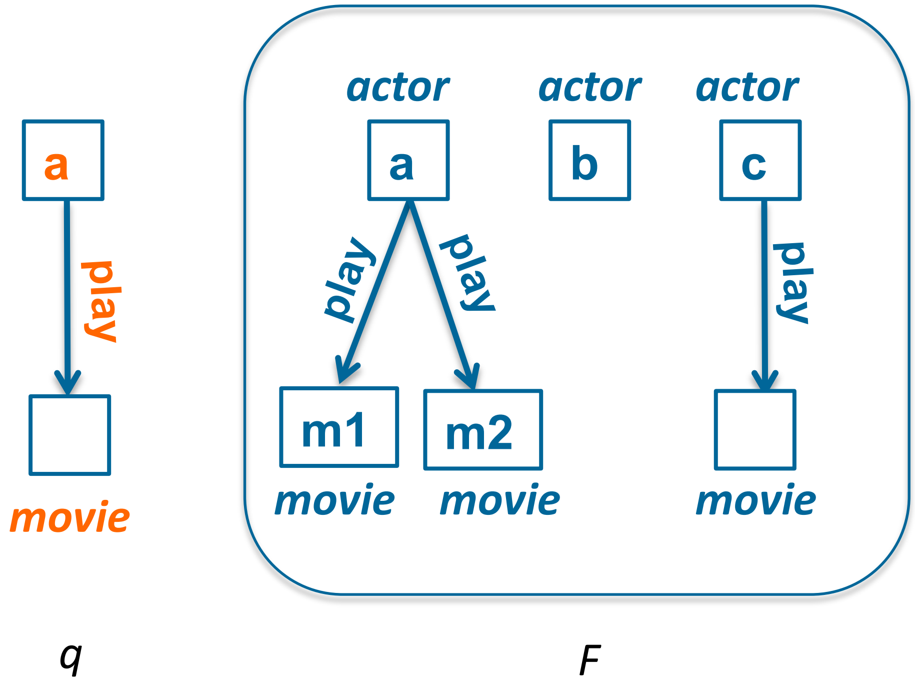
Here, $F \geq G$ means

« F maps to G by homomorphism »

GRAPH HOMOMORPHISMS (2)

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs.
  Homomorphism *h* from G_1 to G_2: mapping from V_1 to V_2 s. t.
  for every edge (u,v) in E_1, (h(u),h(v)) is in E_2

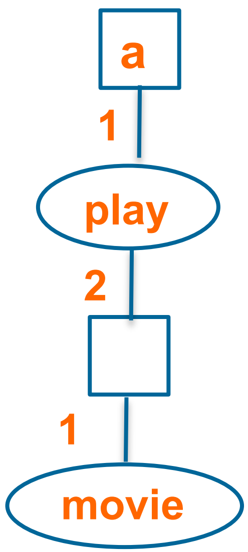
- If there are labels: they have to be ``kept'' as well



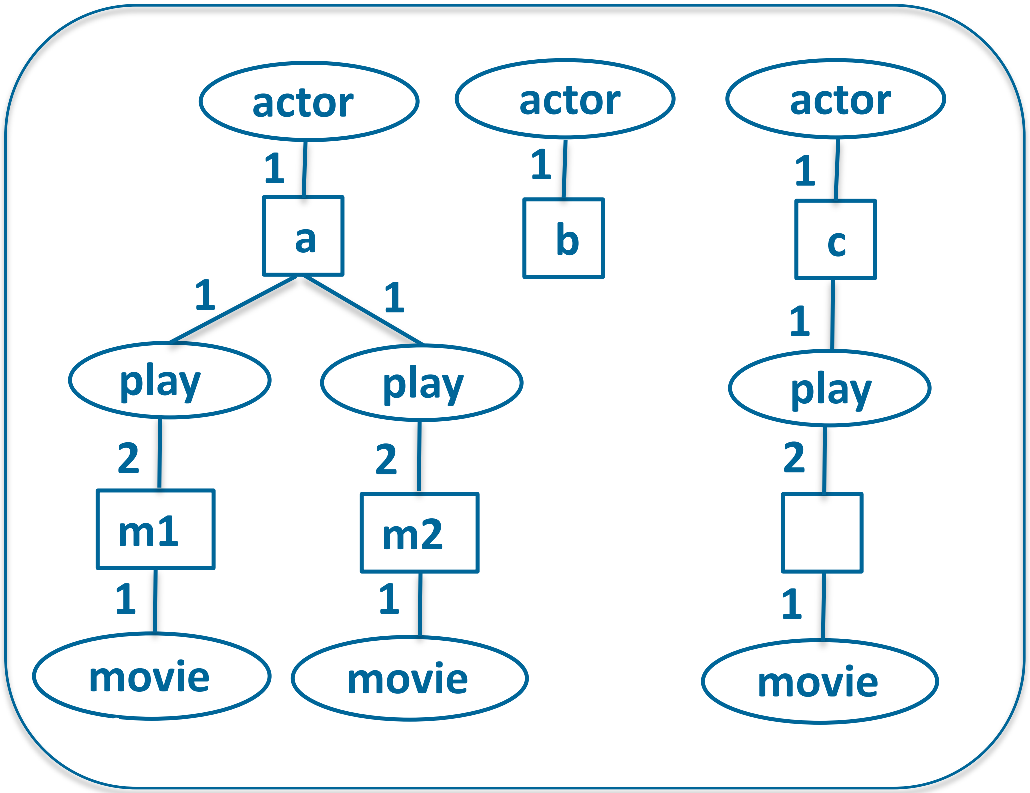
GRAPH HOMOMORPHISMS (3)

- Let $G_1=(V_1,E_1)$ to $G_2=(V_2,E_2)$ be classical graphs.
  Homomorphism *h* from G_1 to G_2: mapping from V_1 to V_2 s. t.
  for every edge (u,v) in E_1, (h(u),h(v)) is in E_2

- If there are labels: they have to be ``kept'' as well



q



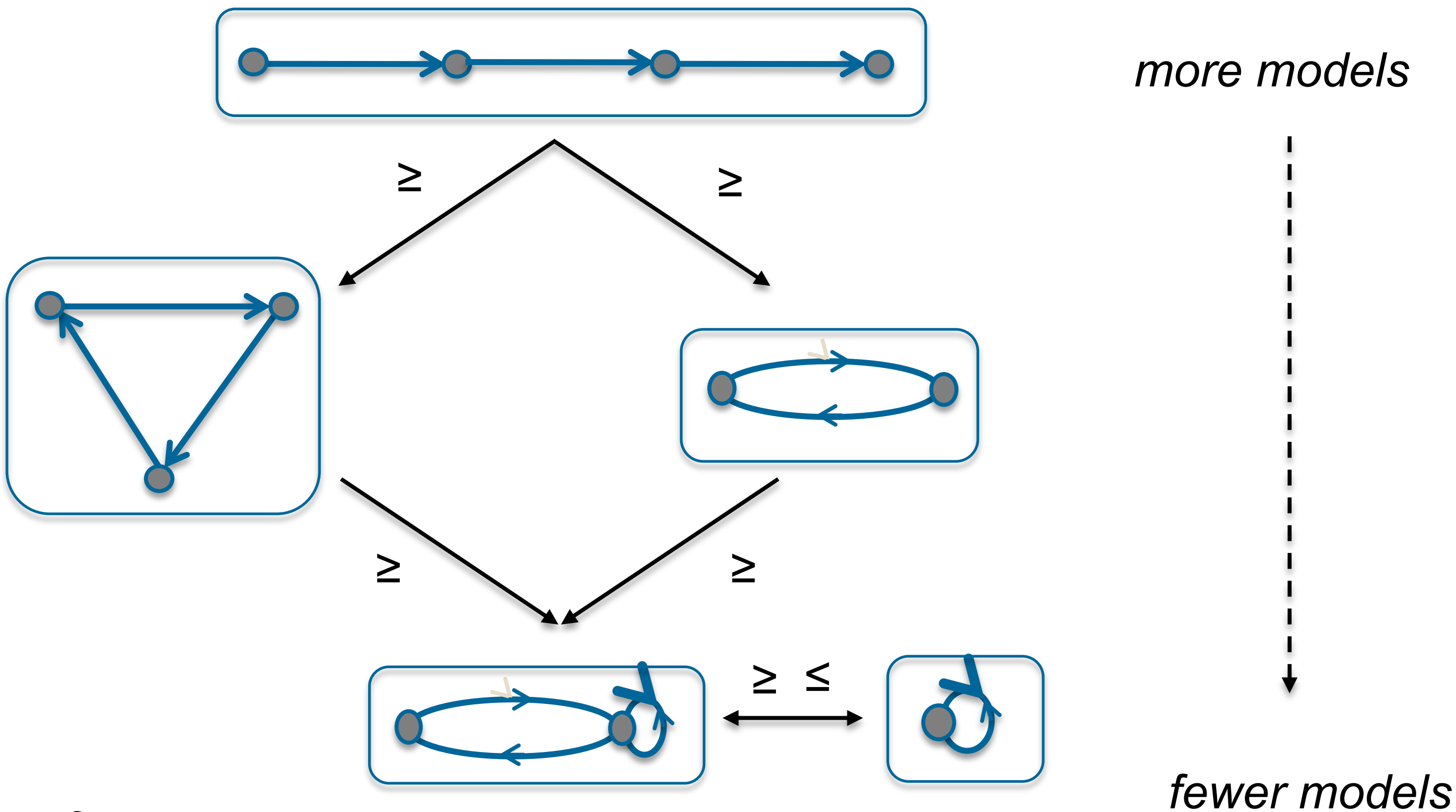
F

EXERCICE

Nous avons vu que nous pouvions encoder tout ensemble d'atomes par un multi-graphe biparti.

- Définir la notion d'homomorphisme naturellement associée avec ces graphes.

- Montrer que : pour tous ensembles d'atomes F1 et F2,
  il existe un homomorphisme de F1 vers F2 si et seulement si
  il existe un homomorphisme (de graphes) de *graphe*(F1) vers *graphe*(F2).



f ≥ *g*

f maps to *g* by homomorphism

g is « more specific » than *f*

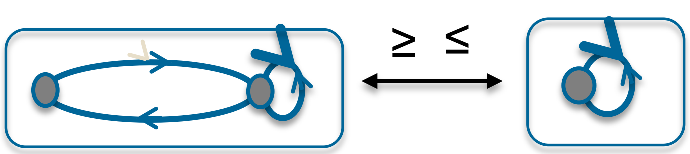
≥ is not an order but a preorder
(a reflexive and transitive relation)

Back to isomorphism

- Let f and g in FOL($\exists$,$\wedge$) seen as sets of atoms

- Isomorphism h from f to g: bijective mapping from $var(f)$ to $var(g)$ such that $h(f) = g$

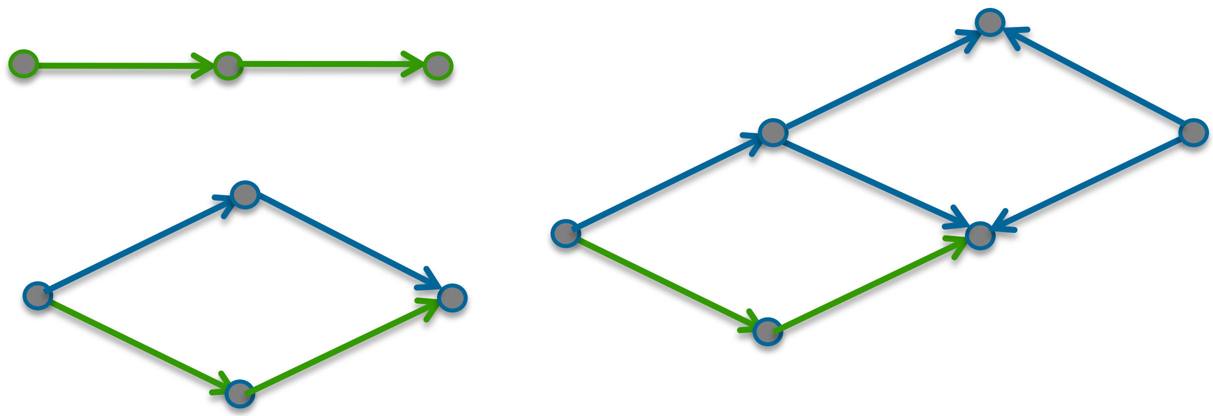
- Equivalent definition of isomorphism: homomorphism h from f to g such that $\boldsymbol{h^{-1}}$ is a homomorphism from g to f



One may have $f \geq g$ and $g \geq f$, but f and g are not isomorphic

EQUIVALENCE / CORE

- If $f \geq g$ and $g \geq f$, they are called (homomorphically) equivalent
- A **core** is a set of atoms that is not equivalent to any of its strict subsets
- Given f a set of atoms, the core of f is a minimal subset of f equivalent to f. It may happen that f has several cores, but they are all isomorphic. Hence, we can say « the » core of f.
- If f and g are equivalent, the core of f is isomorphic to the core of g.



Exercice : give an algorithmic scheme to compute the core of an atom set.

Conclusion

- Le fragment logique existentiel conjonctif est fortement lié aux graphes / hypergraphes
  (donc aux modèles de données graphes, comme RDF)

- Les notions d'homomorphisme et de core sont fondamentales pour de nombreux problèmes sur les bases de faits et les requêtes

  Par exemple : optimisation de requêtes conjonctives
  - Déterminer si deux requêtes sont équivalentes
    - but : exécuter la plus simple pour le SGBD
  - Miminiser une requête : supprimer toutes ses redondances
    - but : accélérer l'évaluation de la requête

  [On parle de problèmes d'optimisation « statique » de requêtes, au sens où ils sont indépendants d'une base de données (ou base de faits) particulière]

Exercice 1 : inclusion de requêtes

Etant données deux requêtes conjonctives booléennes, Q1 et Q2, on dit que Q1 est **incluse** dans Q2 (notation Q1 $\sqsubseteq$ Q2) si l'ensemble des bases de faits qui répondent oui à Q1 est inclus dans l'ensemble des bases de faits qui répondent oui à Q2.

Utiliser les notions vues dans ce cours pour prouver que :

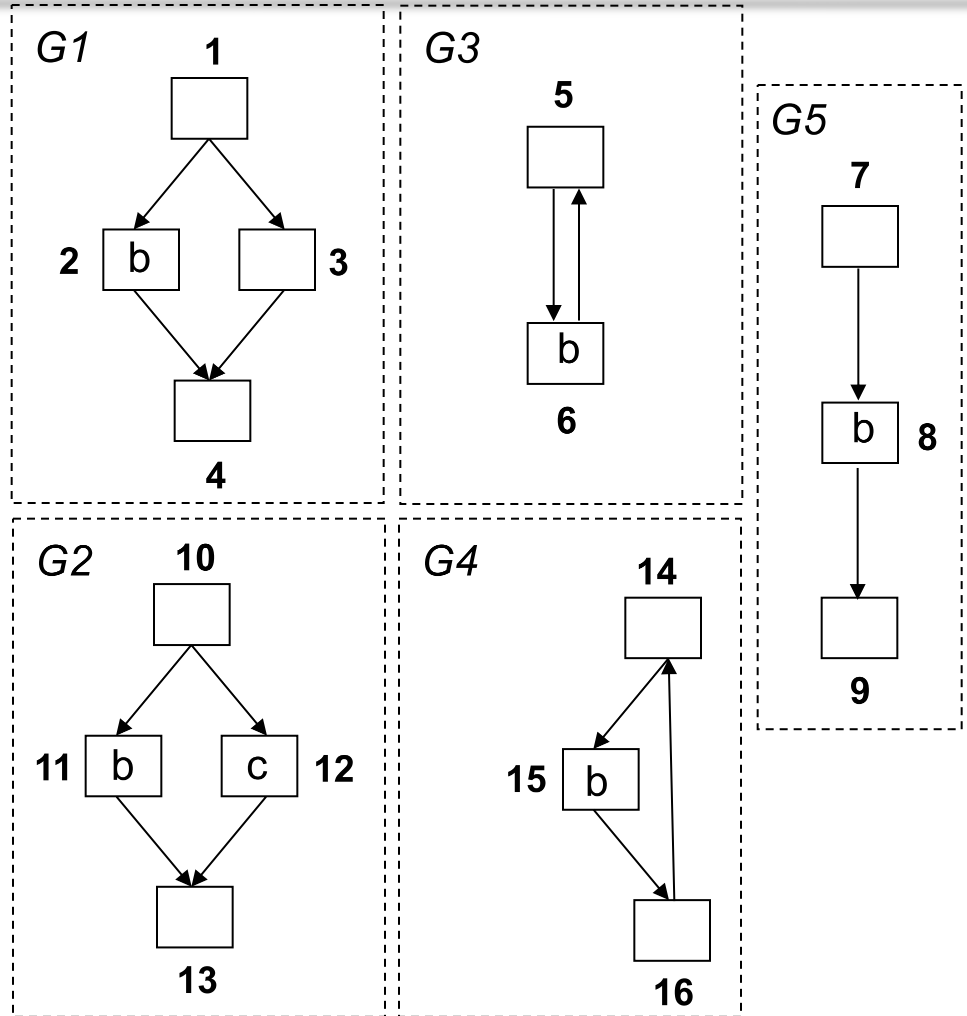
Q1 $\sqsubseteq$ Q2

ssi

il existe un homomorphisme de Q2 dans Q1

Question subsidiaire : pour des requêtes quelconques, Q1 est incluse dans Q2 si pour toute base de faits, l'ensemble des réponses à Q1 est inclus dans l'ensemble des réponses à Q2. Comment étendre le résultat précédent ?

Exercice 2 : Homomorphismes



1. En supposant qu'on n'ait que le prédicat binaire *p*, quelles sont les formules logiques associées à ces graphes ?
2. Les classifier par homomorphisme (en utilisant la vue logique ou graphe)