Universiteit Leiden

Proper Refinement of Datalog Clauses using Primary Keys

Siegfried Nijssen and Joost N. Kok

BNAIC-2003, Nijmegen

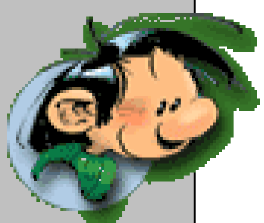
Introduction

- Inductive Logic Programming algorithm:

C: Set of Datalog clauses, initially empty

D: Database of facts *(Knowledge base)*

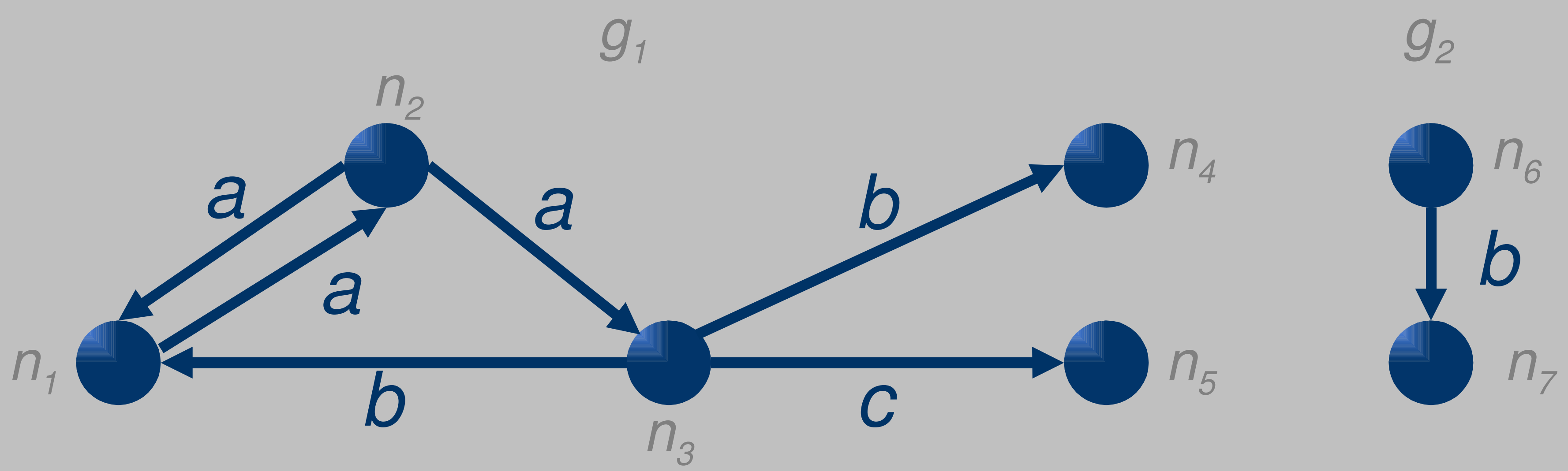
repeat



2. make clauses in *C* more specific (*downward refinement)*

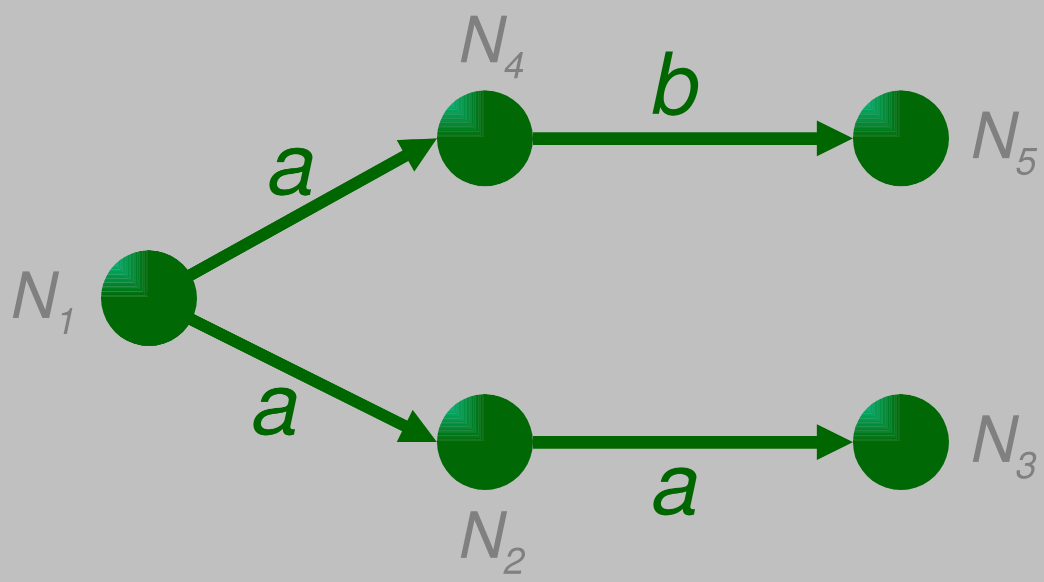
1. evaluate *C* against *D*

Database of Facts



- $\{e(g_1,n_1,n_2,a), e(g_1,n_2,n_1,a), e(g_1,n_2,n_3,a),$
  $e(g_1,n_3,n_1,b), e(g_1,n_3,n_4,b), e(g_1,n_3,n_5,c),$
  $e(g_2,n_6,n_7,b)\}$

Clause



- $k(G) \leftarrow e(G,N_1,N_2,a),e(G,N_2,N_3,a),$
  $e(G,N_1,N_4,a),e(G,N_4,N_5,b)$

Evaluation of a clause

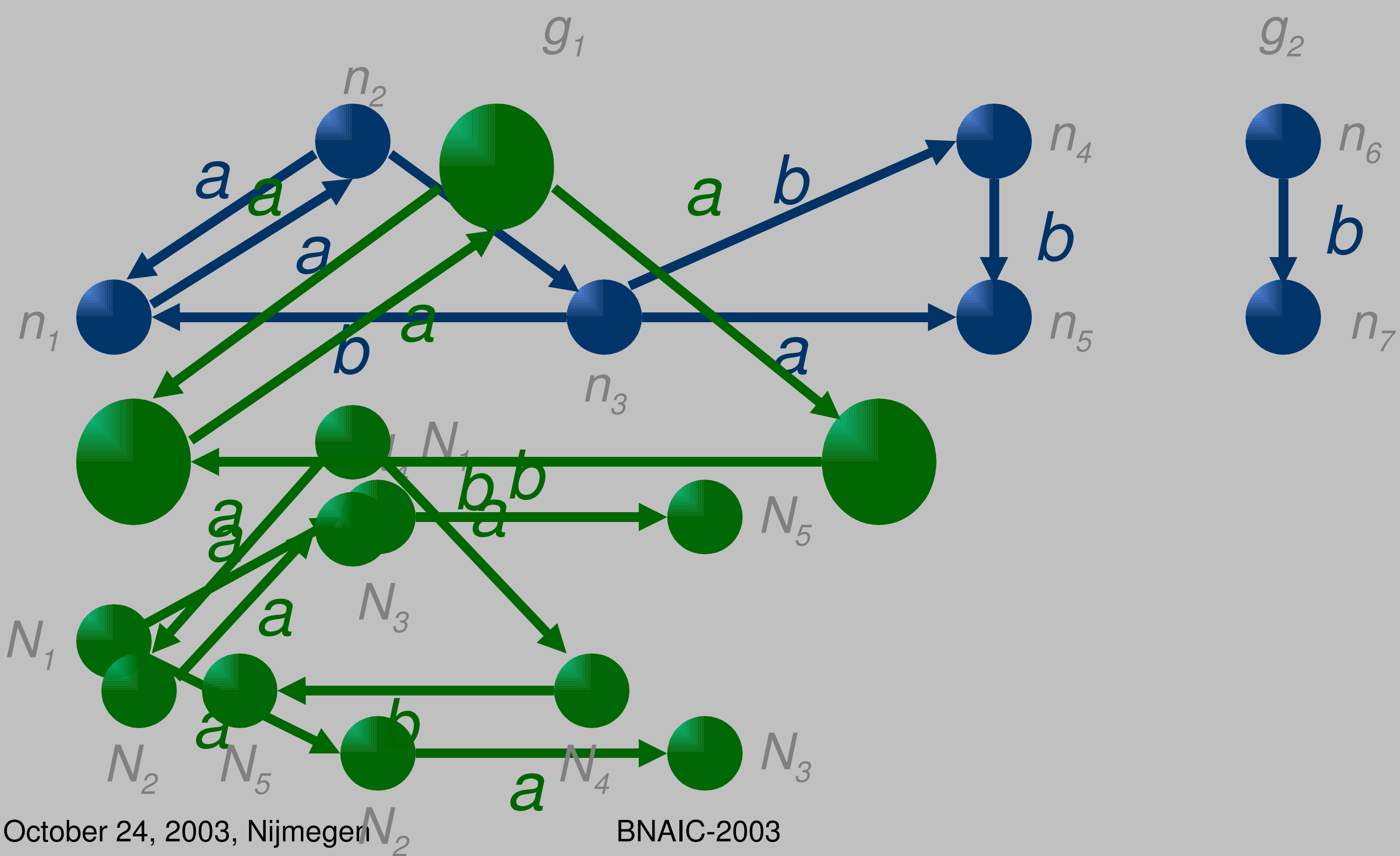
- θ-subsumption: $D \models C$ iff there is a substitution θ, $(C\theta) \subseteq D$
- Database D: $\theta=\{G/g_1, N_1/n_2, N_2/n_1, N_3/n_2, N_4/n_3, N_5/n_1\}$

  $\{e(g_1,n_1,n_2,a), e(g_1,n_2,n_1,a), e(g_1,n_2,n_3,a),$
  $e(g_1,n_3,n_1,b), e(g_1,n_3,n_4,b), e(g_1,n_3,n_5,c),$
  $e(g_2,n_6,n_7,b)\}$

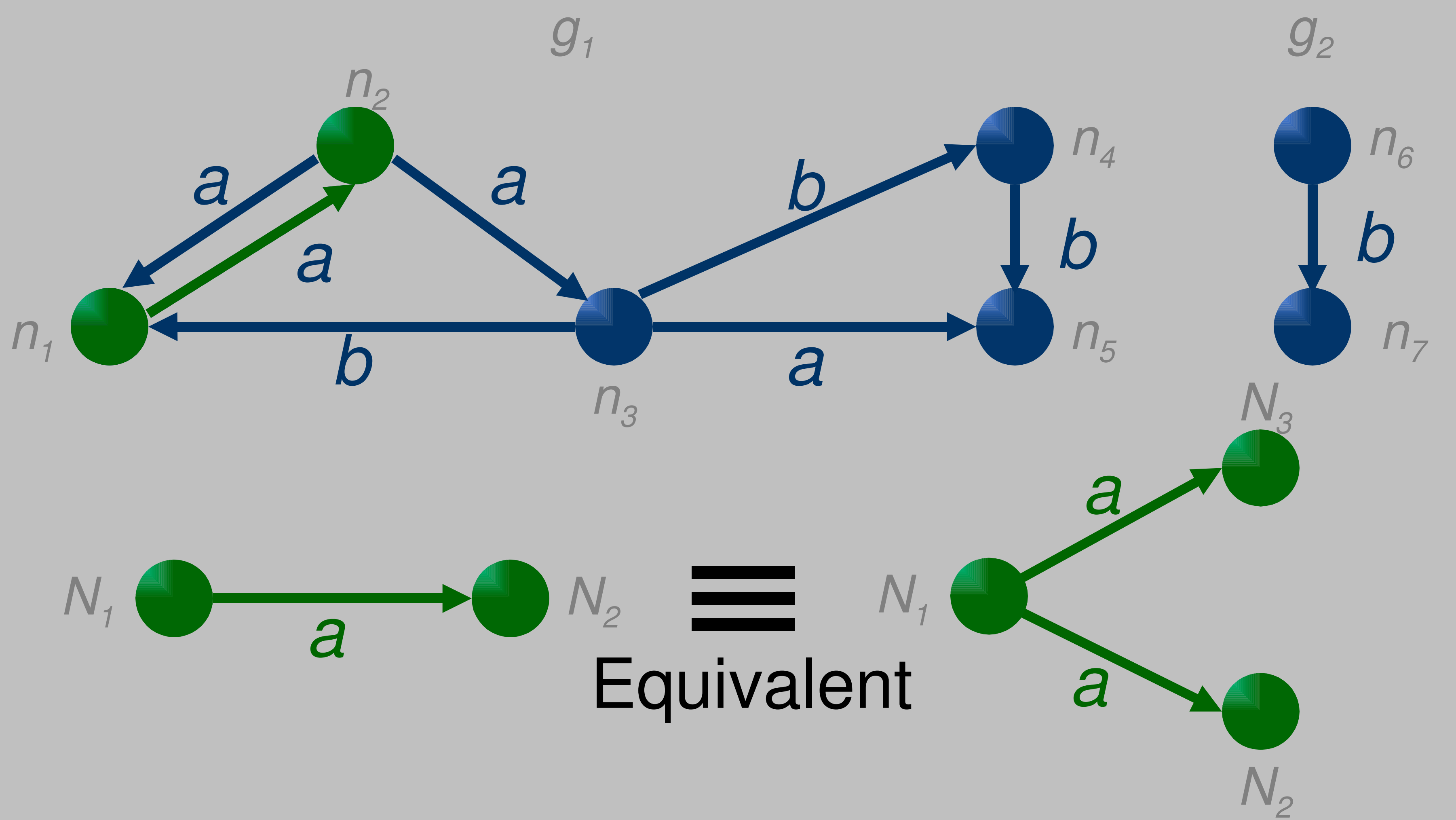
- Clause C:

  $k(G) \leftarrow e(G,N_1,N_2,a), e(G,N_2,N_3,a),$
  $e(G,N_1,N_4,a), e(G,N_4,N_5,b)$

Evaluation of a clause



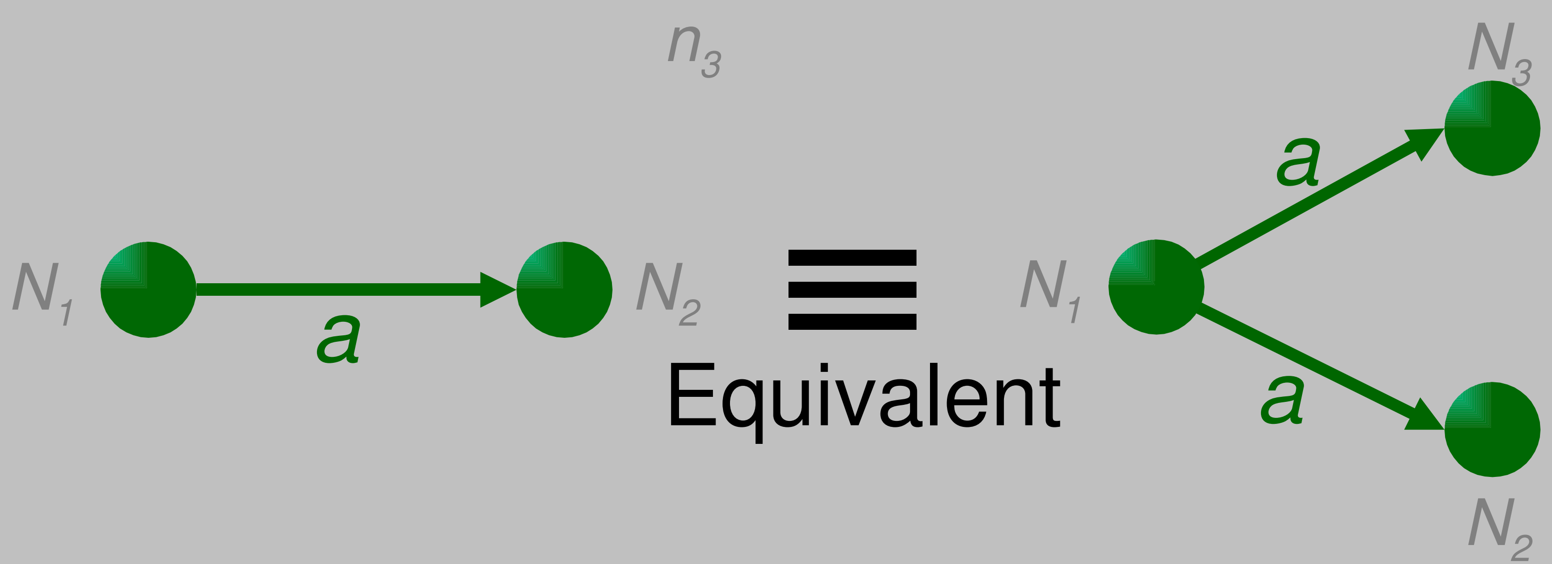
Evaluation of a clause



Evaluation of a clause

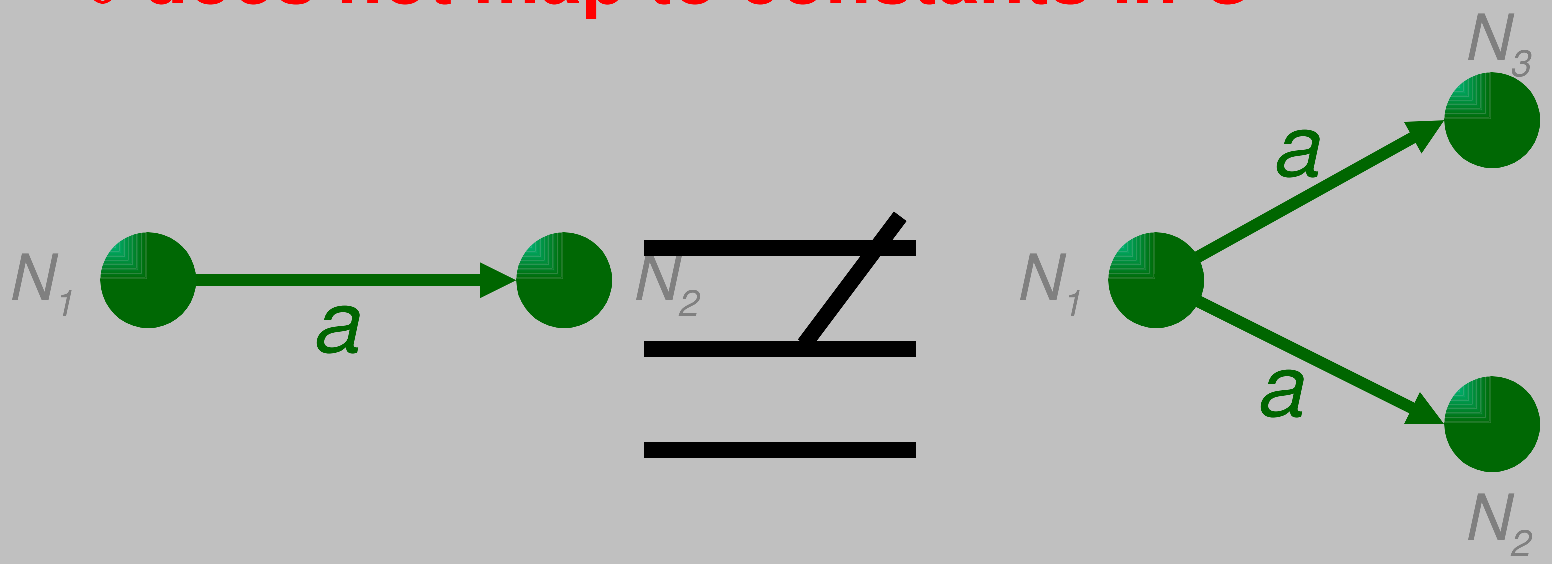
$$k(G) \leftarrow e(G,N_1,N_2,a)$$

$$k(G) \leftarrow e(G,N_1,N_2,a),\; e(G,N_1,N_3,a)$$



Evaluation of a clause

- OI-subsumption: $D \models C$ iff there is a substitution θ, $(C\theta) \subseteq D$, **while:**
  - **θ is injective**
  - **θ does not map to constants in C**



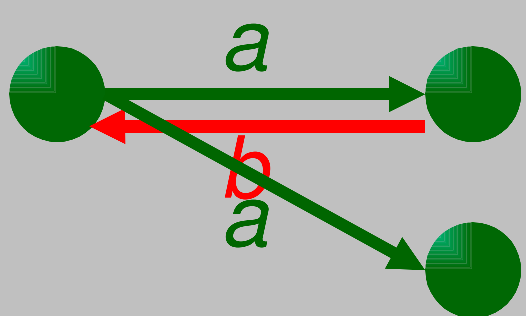
Clause Refinement - modes

- User defined Refinement using modes [Progol,Aleph, Warmr,Tilde,Farmer]

+ old variable
- new variable
constant

- $T=\{k(G),e(G,N,N,L)\}$ $M=\{e(+,-,-,\#),e(+,+,-,\#)\}$

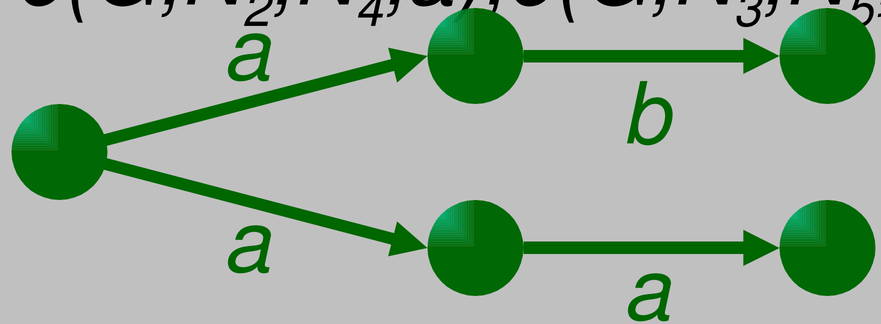
- $k(G) \leftarrow\ e(G,N_1,N_2,a)\ ,e(G,N_1,N_3,a)$



Using M only edge labeled trees can be constructed!

Clause Refinement - modes

- $k(G) \leftarrow e(G,N_1,N_2,a)$
  $k(G) \leftarrow e(G,N_1,N_2,a),e(G,N_1,N_3,a)$
- $k(G) \leftarrow e(G,N_1,N_2,a),e(G,N_1,N_3,a),$
  $e(G,N_2,N_4,a),e(G,N_3,N_5,b)$



- Complete & proper refinement is possible with OI-subsumption, not with θ-subsumption.

Refinement using Primary Keys

- Assume we know: between a pair of nodes there is at most one edge with one label
- How to incorporate this knowledge in the refinement operator?
- $M=\{e(+,-,-,\#),e(+,+,-,\#),e(+,+,+,\#)\}$
- These modes allow:
  - $k(G) \leftarrow e(G,N_1,N_2,a),\ e(G,N_1,N_2,b)$
  - $k(G) \leftarrow e(G,N_1,N_2,a),\ e(G,N_1,N_2,L_1)$
- Primary key: *{1,2,3}* (first 3 arguments of *e*)

Expressiveness
OI vs θ-subsumption

- $k(G) \leftarrow e(G,N_1,N_2,a),e(G,N_2,N_3,L),e(G,N_3,N_4,b)$

- For proper and complete refinement:

  OI is required

- Under OI: $L \neq a$, $L \neq b$

In an ideal situation...

- We have complete & proper refinement
- We are not required to use OI for all types (weak Object Identity)

Proper refinement using Primary Keys

- In many cases, this ideal situation exists for refinement using primary keys!

- $k(G) \leftarrow e(G,N_1,N_2,a),e(G,N_2,N_3,L),e(G,N_3,N_4,b)$

Proper refinement using Primary Keys

- $T=\{k(G),e(G,N,N,L),t(L,C)\}$
  $M=\{e(+,-,-,-),e(+,+,-,-),t(+,\#)\}$
  $K(p)=\{1,2,3\}$ $K(t)=\{1,2\}$
  $\mathrm{OI}=\{G,N\}$

- $k(G)\leftarrow e(G,N_1,N_2,L_1),t(L_1,a),$
  $e(G,N_1,N_3,L_2),t(L_2,a)$

Proper refinement using Primary Keys

- **Given** predicates, types, modes, primary keys and a partition of types into OI and non-OI
- **We prove** refinement is proper and complete if for every mode there is a primary key which does not include any non-OI *output*.

Conclusions

- **Higher performance for ILP algorithms**
  - primary keys restrict the search space efficiently
  - refinement is proper
- **Higher flexibility**
  - *weak OI* is more flexible than *full OI*

Clause refinement - other representation better?

Background knowledge:

$a(G,N_1,N_2) \leftarrow e(G,N_1,N_2,a)$

$b(G,N_1,N_2) \leftarrow e(G,N_1,N_2,b)$

$e(G,N_1,N_2) \leftarrow e(G,N_1,N_2,L)$

- $k(G) \leftarrow a(G,N_1,N_2),e(G,N_2,N_3),b(G,N_3,N_4)$

- $k(G) \leftarrow a(G,N_1,N_2),e(G,N_1,N_2)$