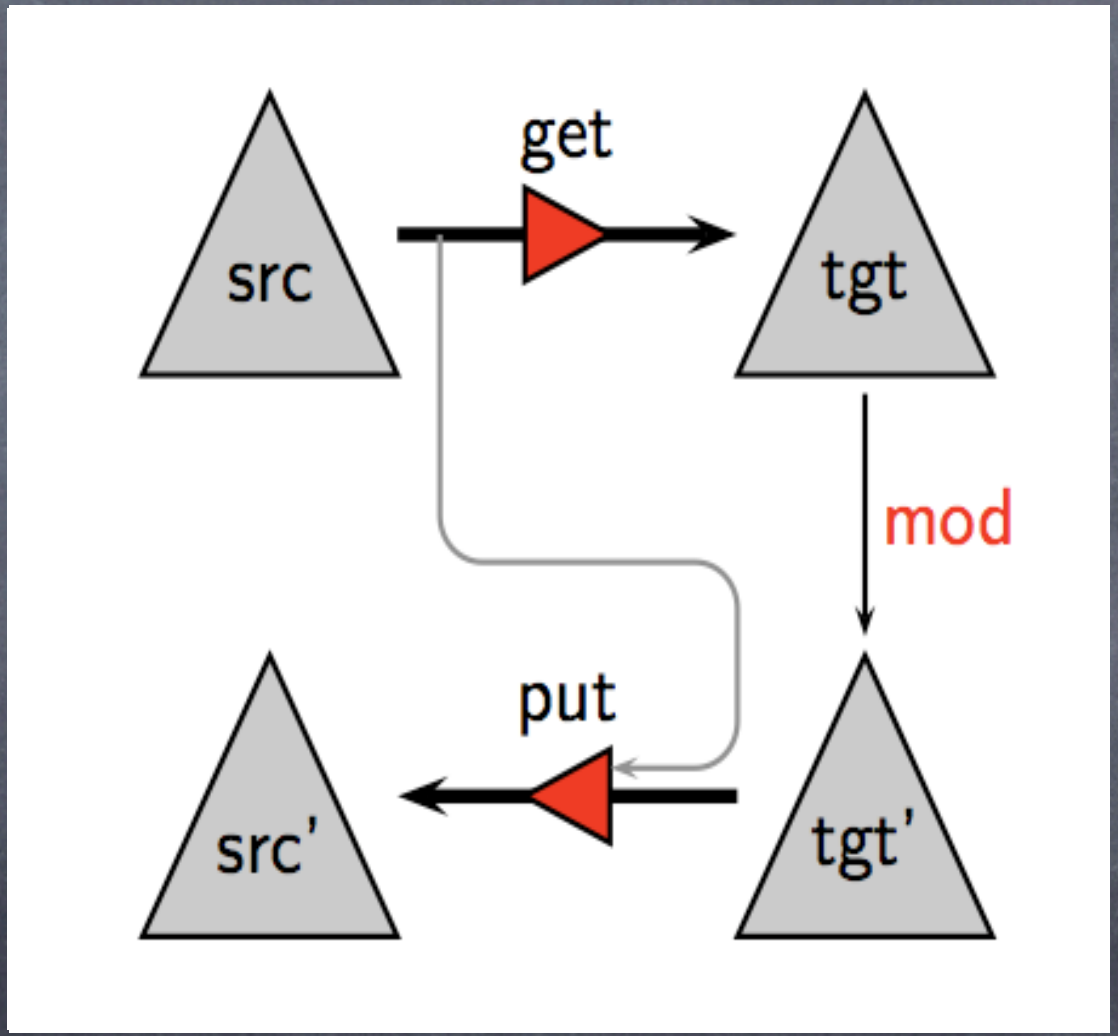
Can Graph Transformation be Bidirectionalized?

Zhenjiang Hu
National Institute of Informatics, Japan

Joint work with the BiG group members

H. Hidaka, K. Inaba, H. Kato, K. Matsuzaki, K. Nakano

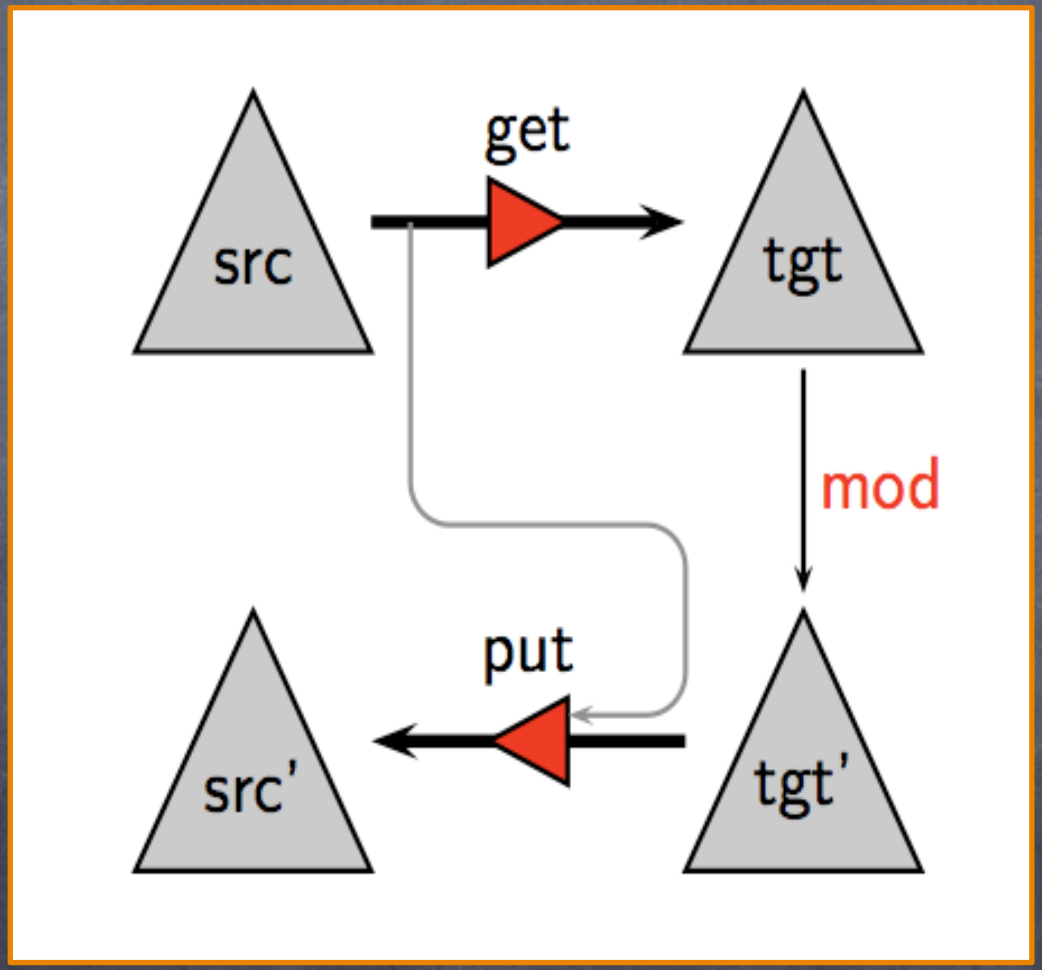
Bidirectional Transformation



onsists of a pair of computation **forward** and **backwa**

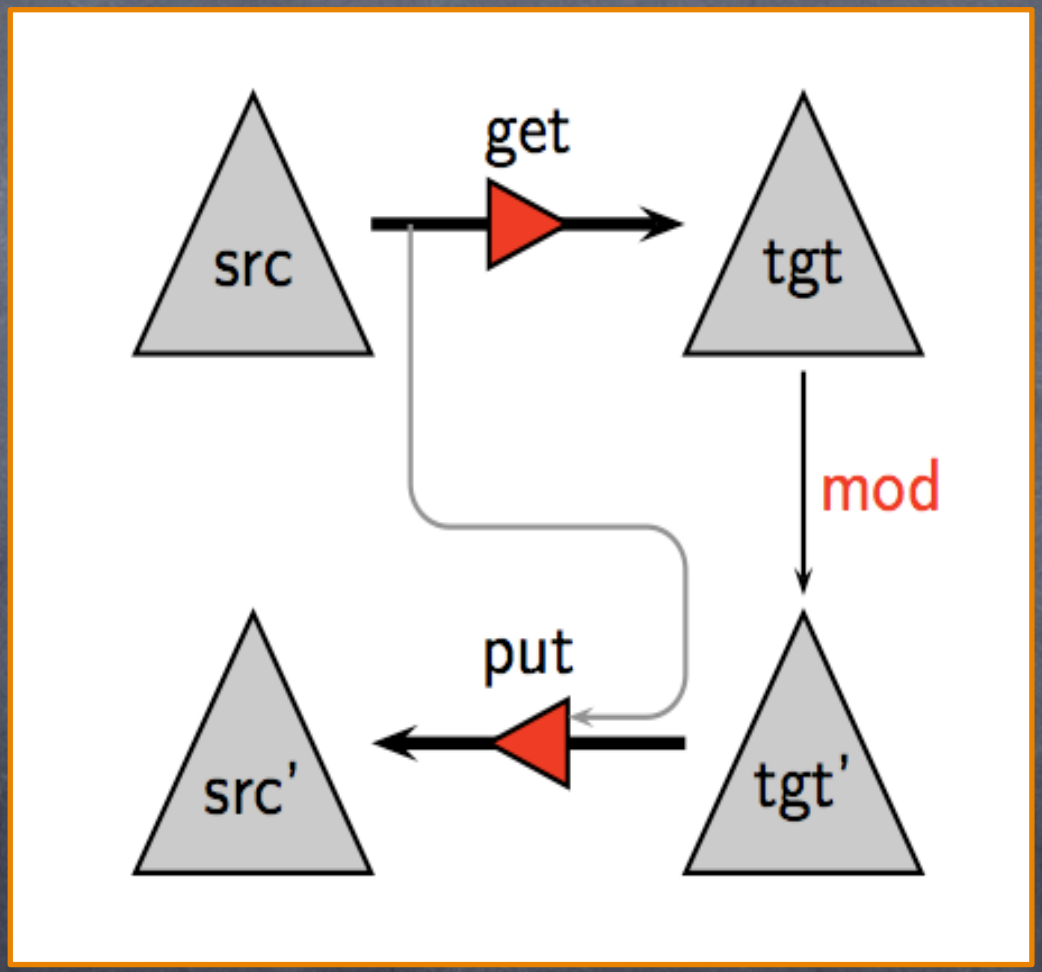
Stability

No Change on the view implies no change on the sou



Reflectivity

Changes on the view is reflected to the source

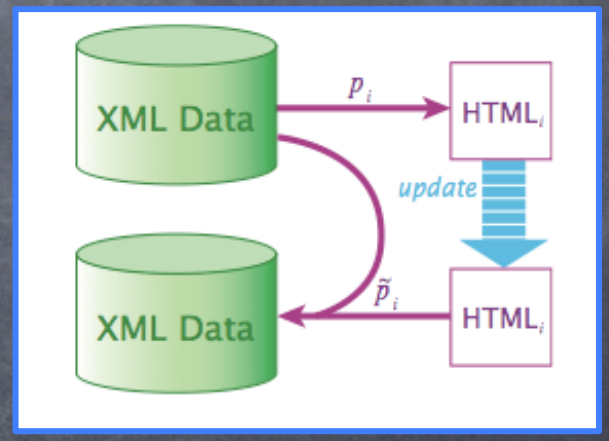


Pervasive but Calls for Language Support

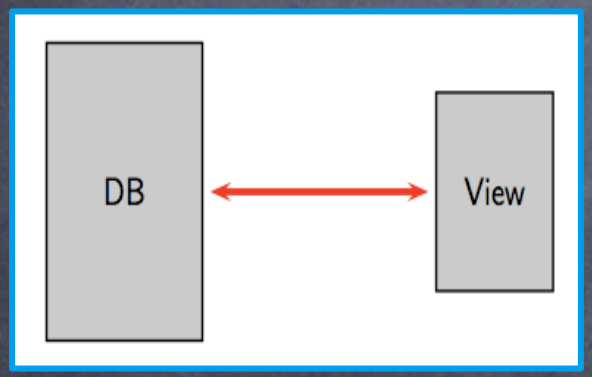
Data Synchronization (2006)

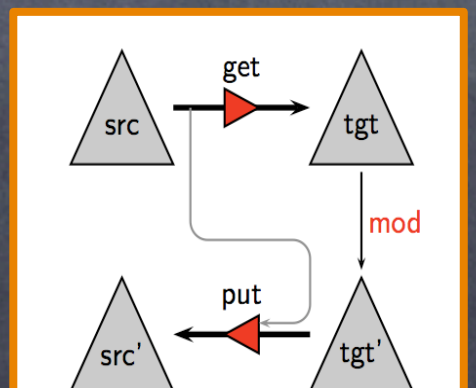
Roundtrip Engineering

Web Maintainer (2008)



View Updating (80')





Bidirectional Tree Transformation Languages

- Laguages for support data synchronization
  - University of Pennsyvania
  - Lens [POPL'06], Boomerang [ICFP'08], ...
- Language for document construction
  - University of Tokyo
  - X/Inv [PEPM'04,MPC'04], BX18 [ICFP'07,ESOP'

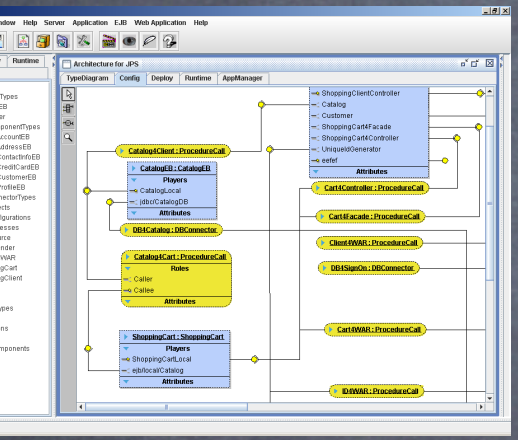
Can we design a language for bidirectional graph transformation (BiG)?

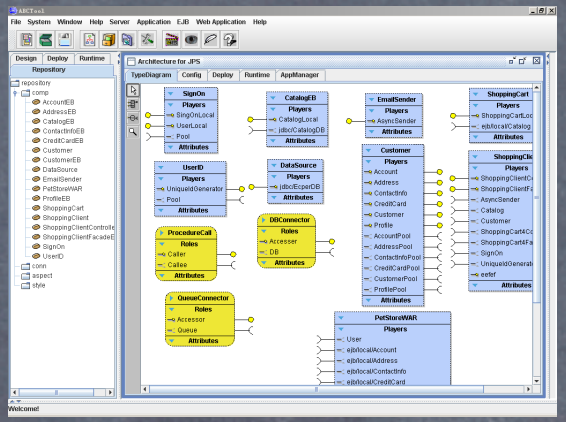
My Talk

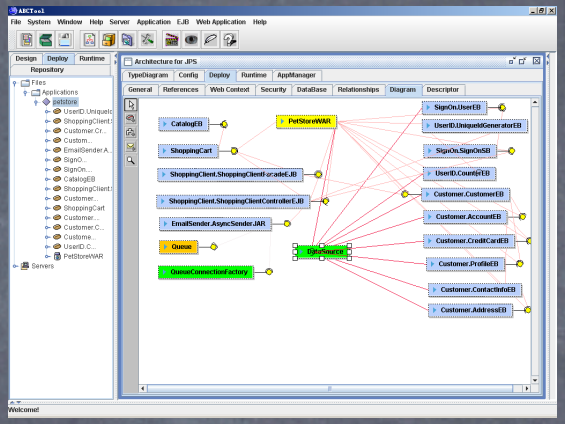
- We report our design and implementation of a bidirectional graph transformation languae UnQL+
- We demonstrate its application in evolutional software engineering
- We highlight some difficult problems

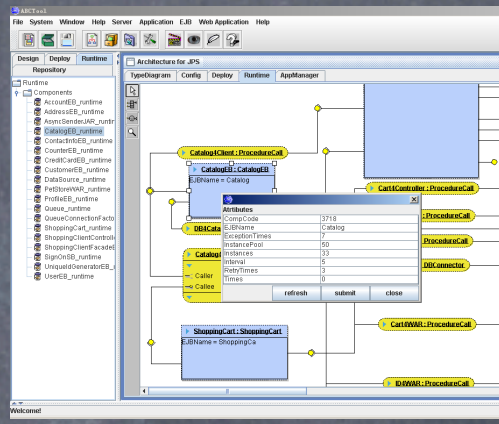
BiG is Important

Model-driven software development









Model-based Requirements Analysis

Model-based Software Design

Model-based Component Composition

Model-based Application Deployment

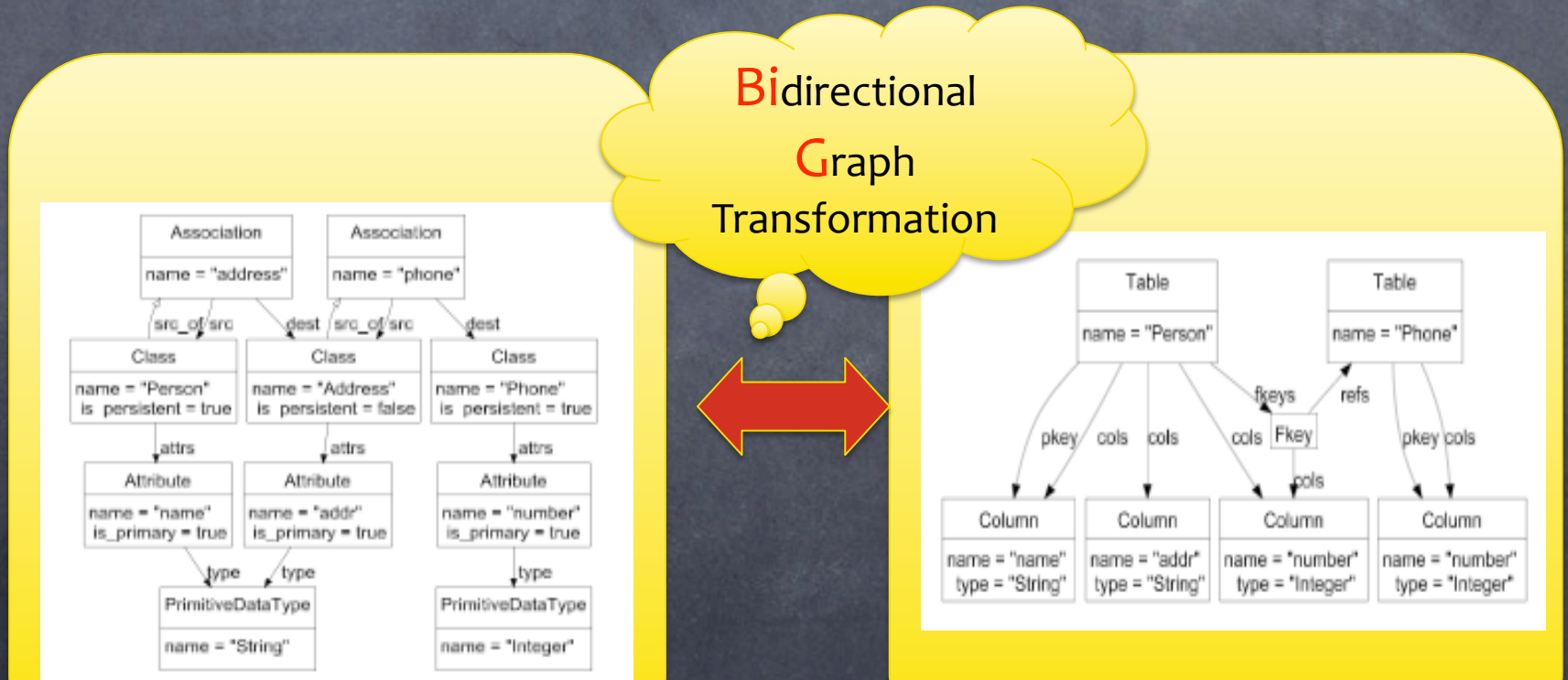
Model-based Testing

BiG is Important

Model-driven software development:

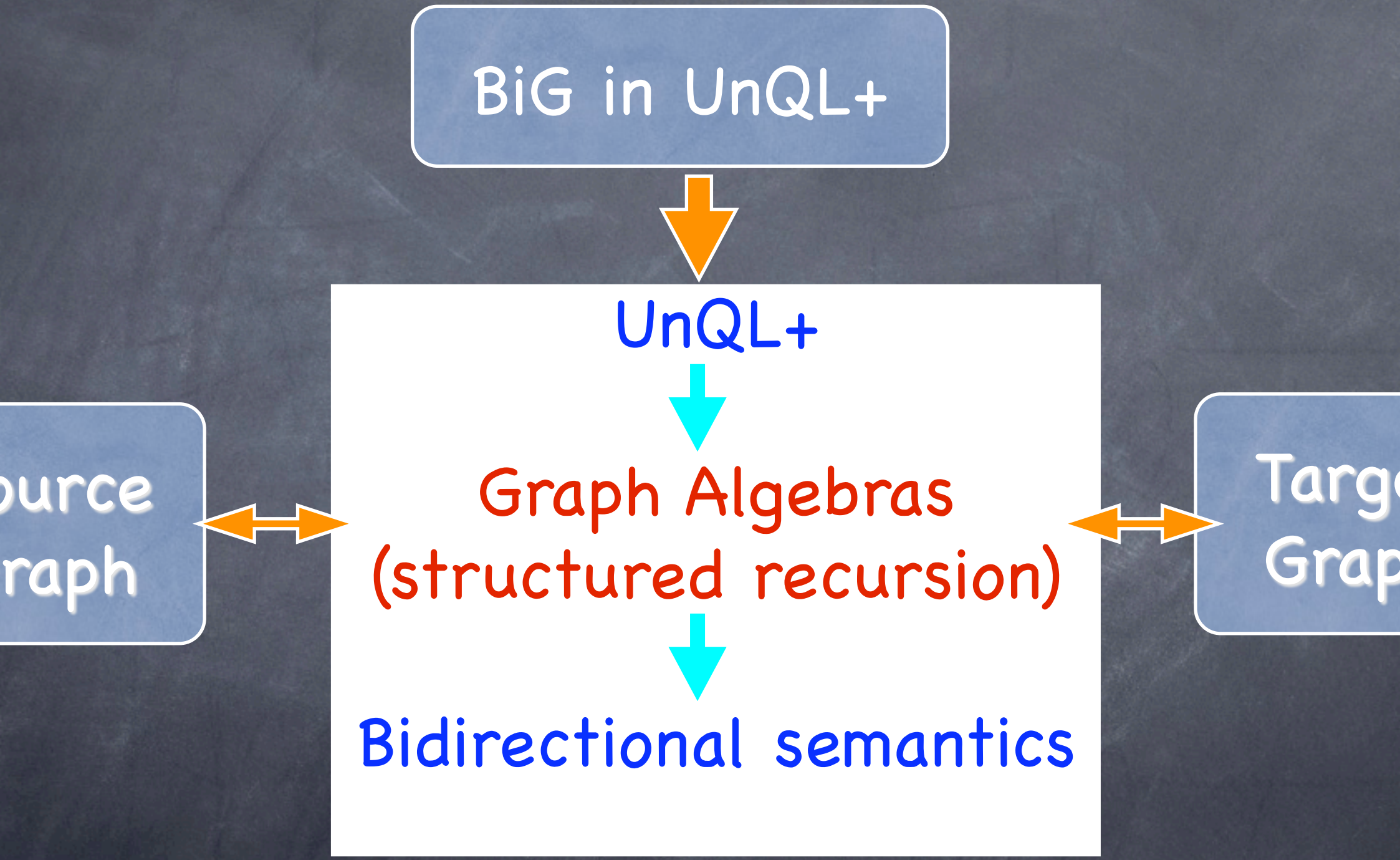
Models ==> Graphs

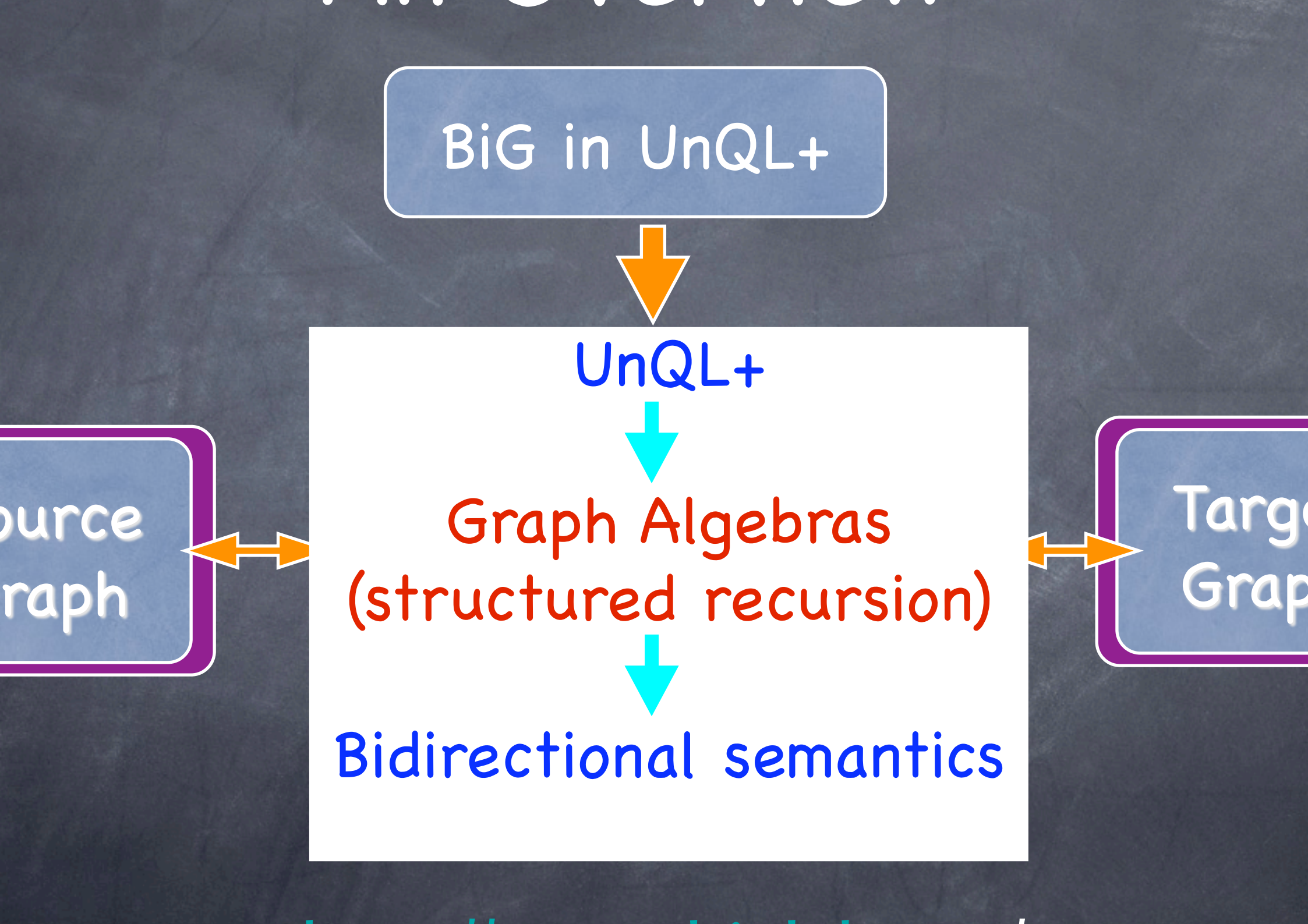
Model transformation ==> Graph transformations



BiG is Challenging

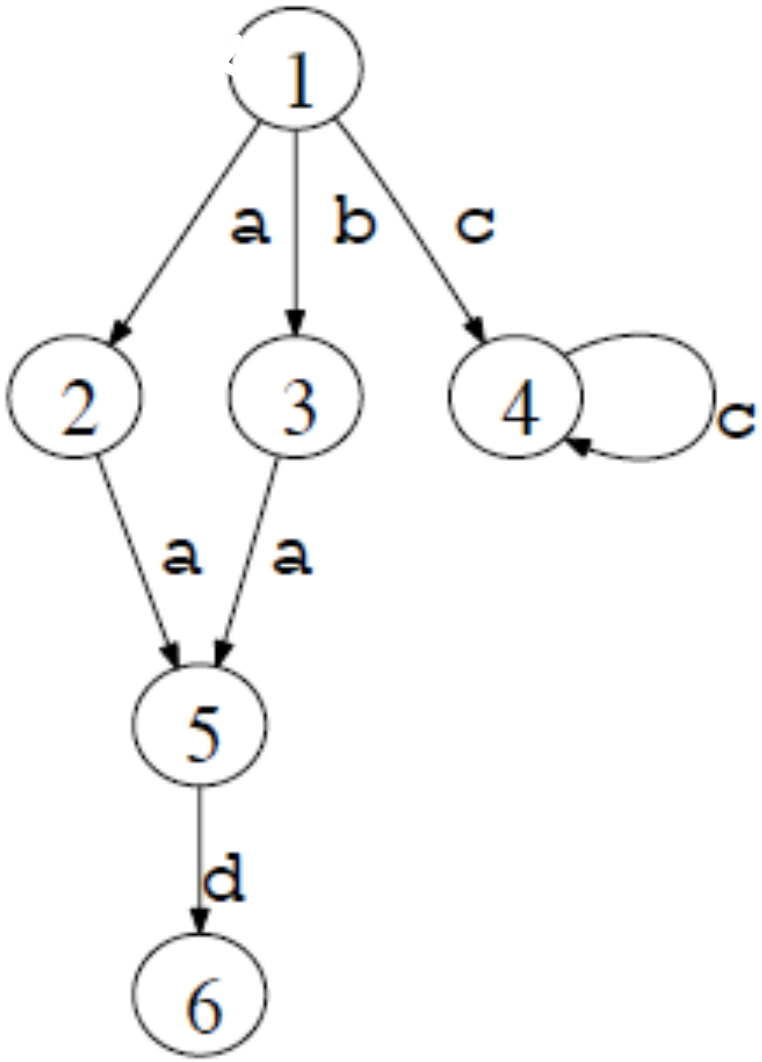
- How to deal with termination of graph transformation?
- How to deal with equality of two graphs?
- How to reflect changes on the view to the source?


BiG in UnQL+
UnQL+
Graph Algebras
(structured recursion)
Bidirectional semantics
ource
raph
Targ
Grap


BiG in UnQL+
UnQL+
Graph Algebras
(structured recursion)
Bidirectional semantics
ource
raph
Targ
Grap

Graph Model

Rooted Edged-Labeled Graph



$$
\begin{aligned}
G_{\text{root}} &= \{\mathtt{a} : \{\mathtt{a} : G_5\}, \mathtt{b} : \{\mathtt{a} : G_5\}, \mathtt{c} : G \\
G_5 &= \{\mathtt{d} : \{\}\} \\
G_4 &= \{\mathtt{c} : G_4\}.
\end{aligned}
$$

G = (V, E, I, O)

where

V = {1,2,3,4,5,6}

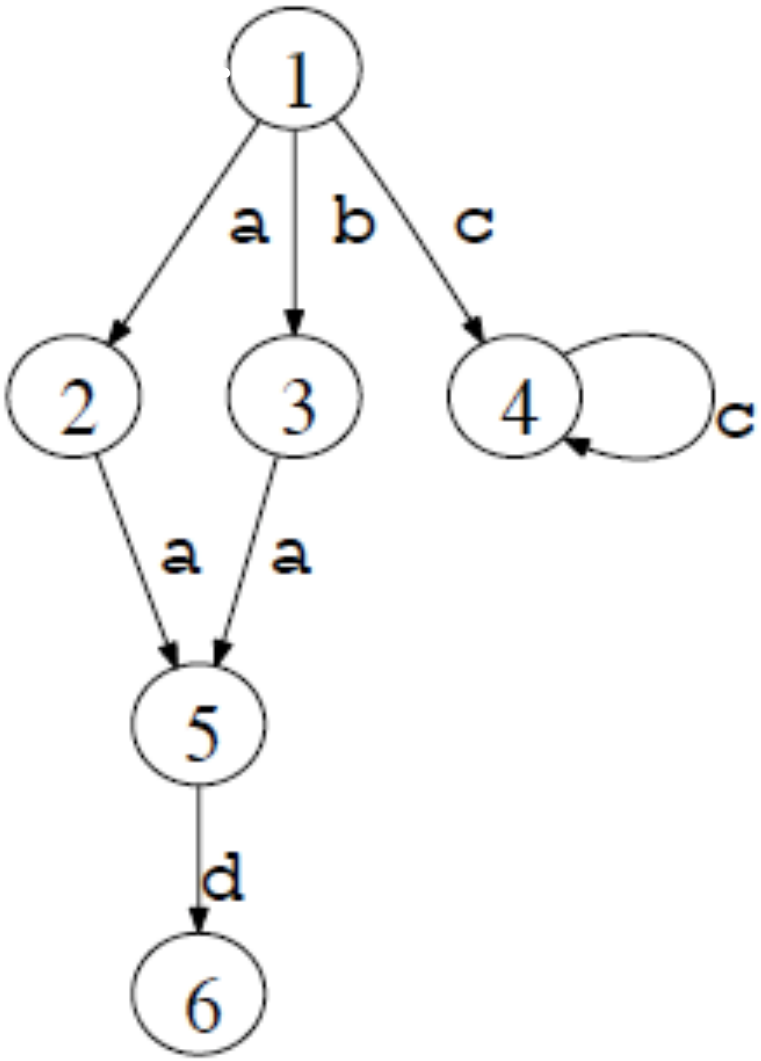
E = {(1,a,2), (1,b,3), (1,c,4),
(2,a,5), (3,a,5), (5,d,6)}

I = {(&,1)}

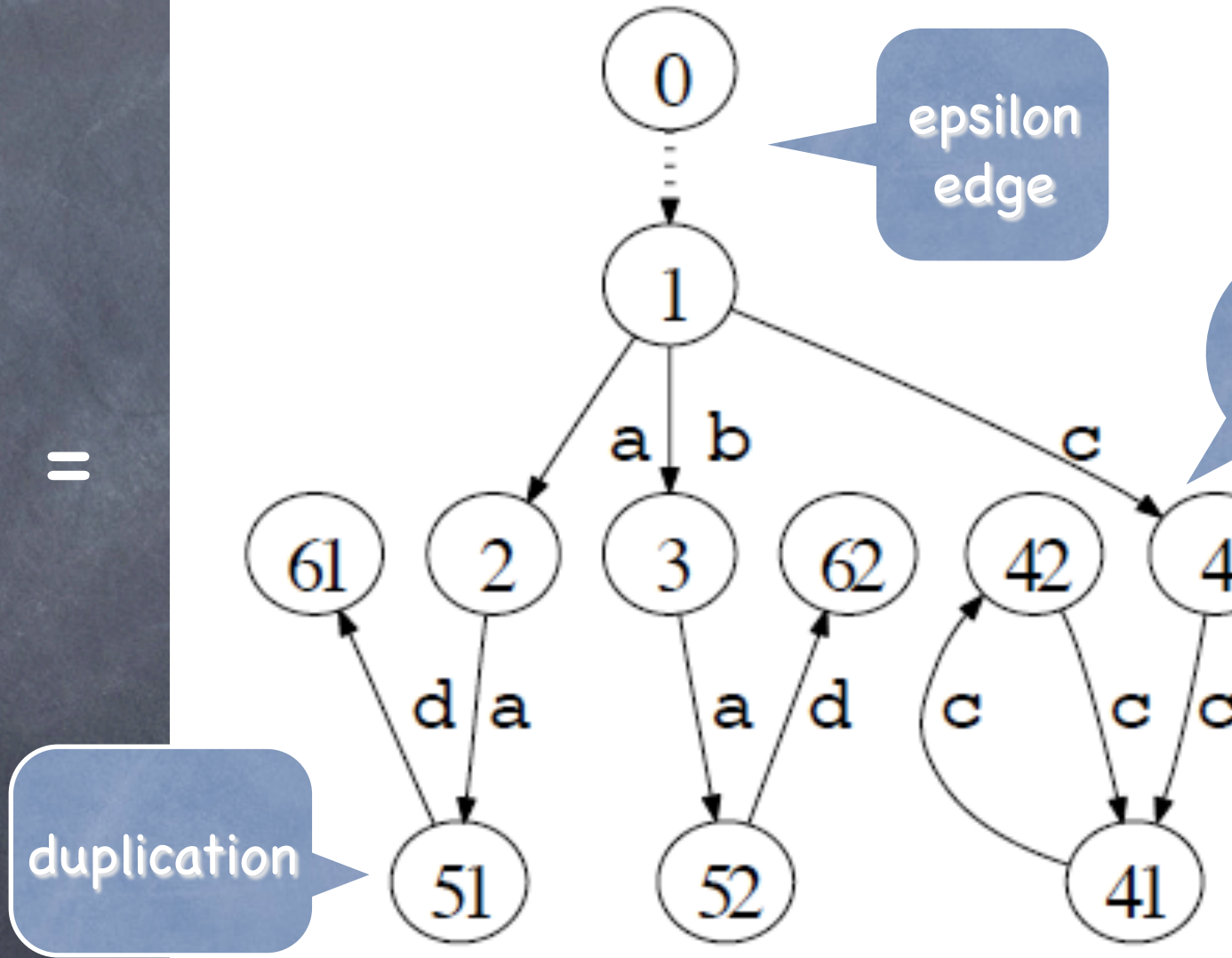
O = {}

Graph Model

Equivalence based on Bisimulation

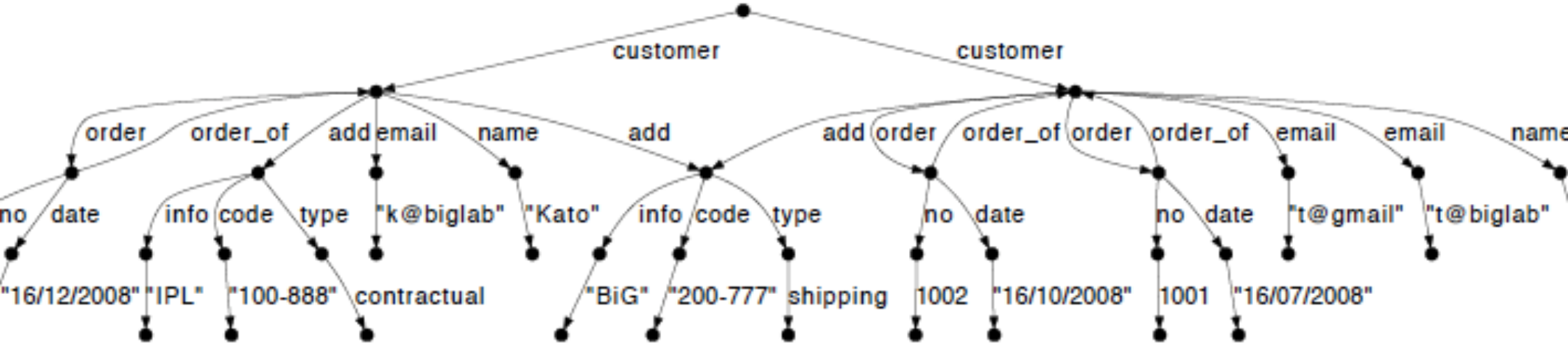


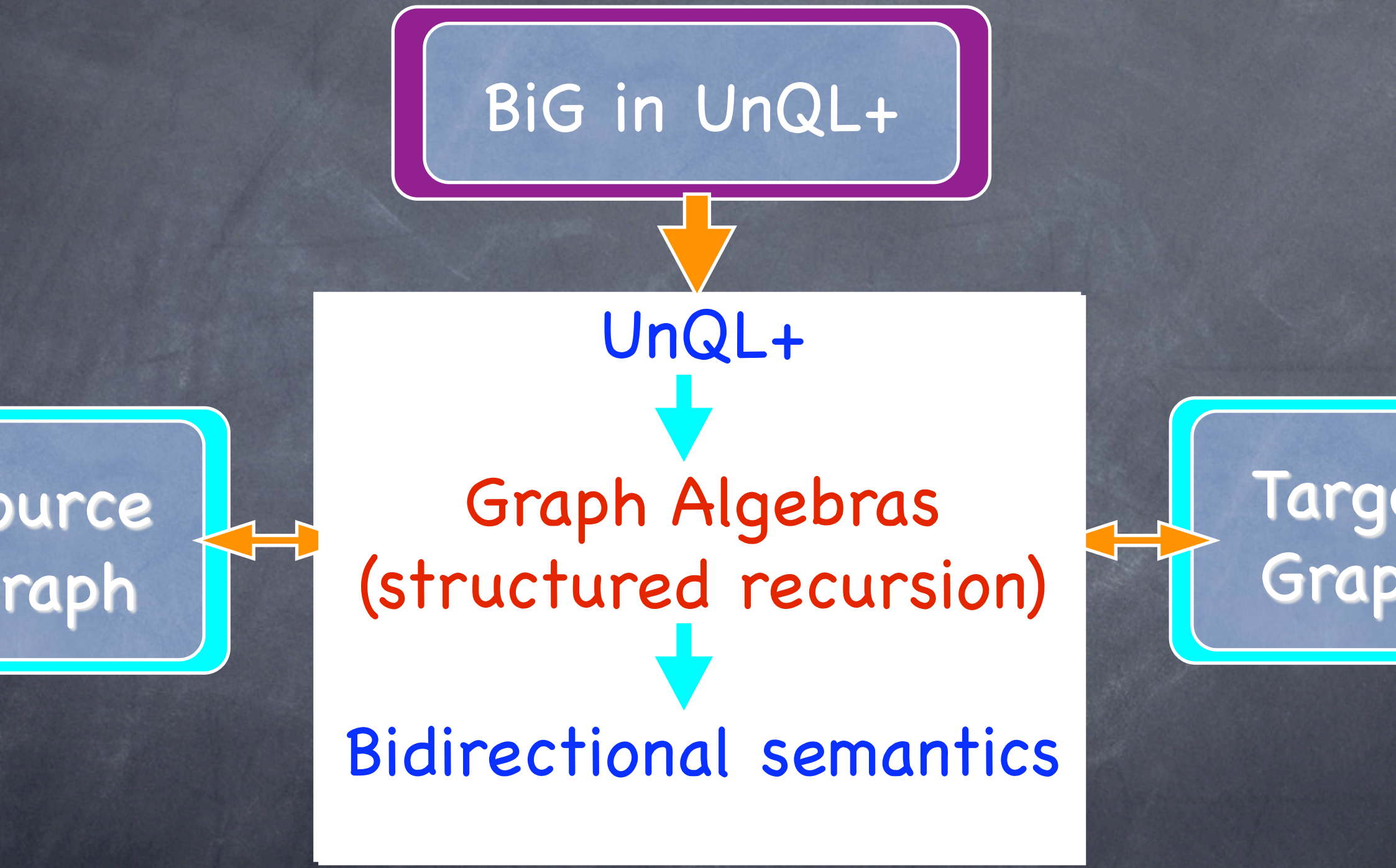
=



Graph Model

An Example: A Customer Graph



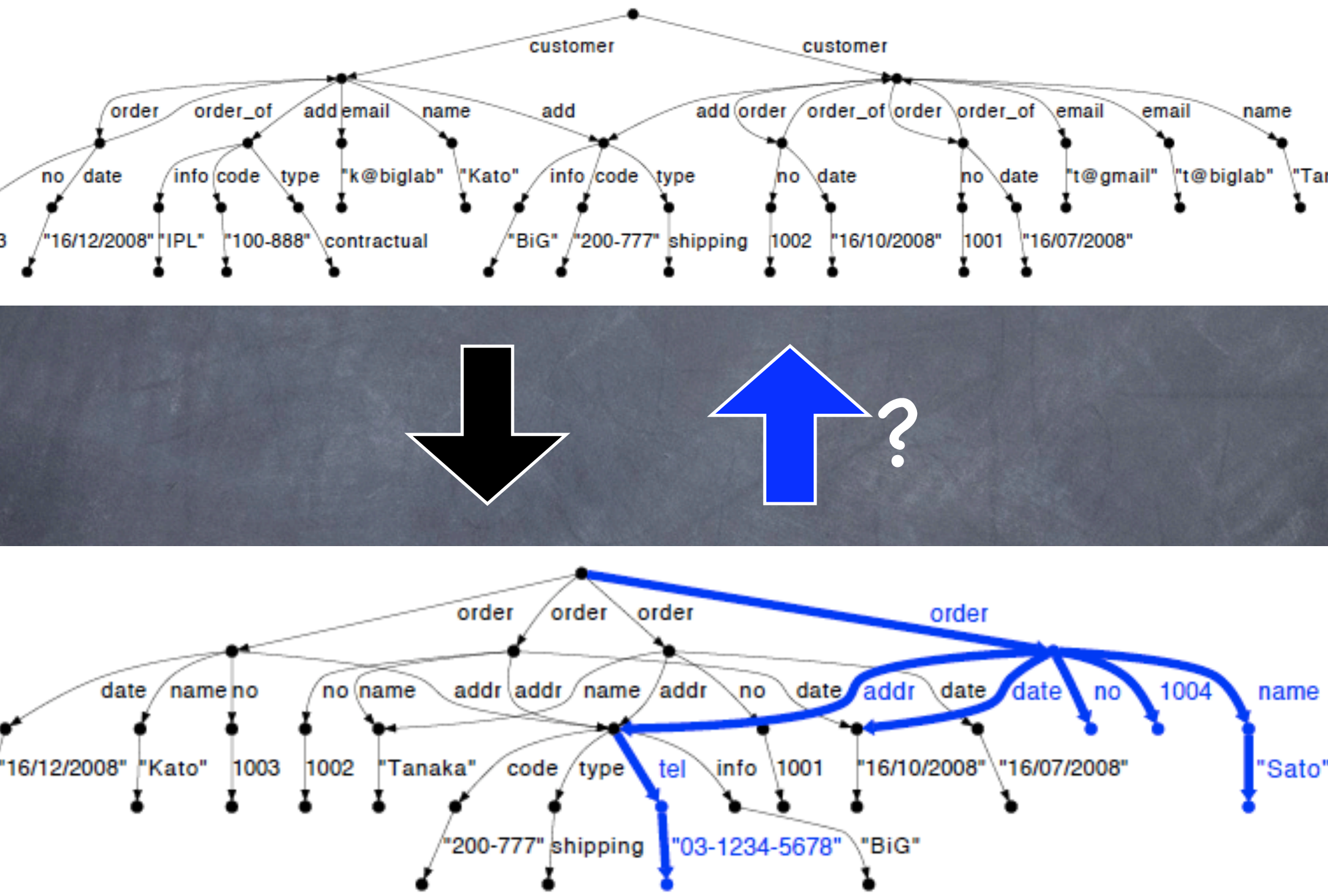

BiG in UnQL+
UnQL+
Graph Algebras
(structured recursion)
Bidirectional semantics
ource
raph
Targ
Grap

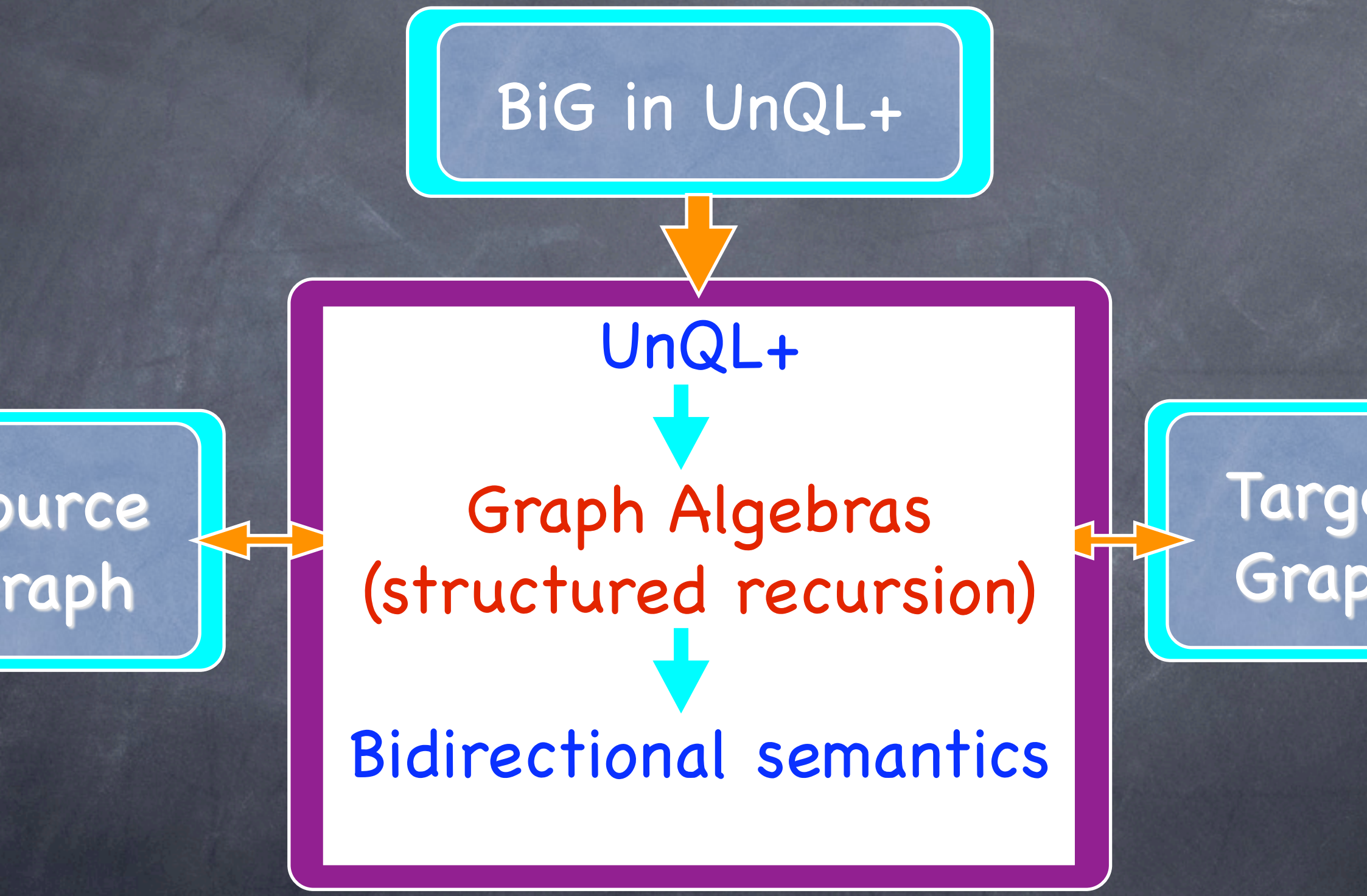
UnQL+

An extension of UnQL, a functional graph query language [Buneman et. al. VLDB'00].

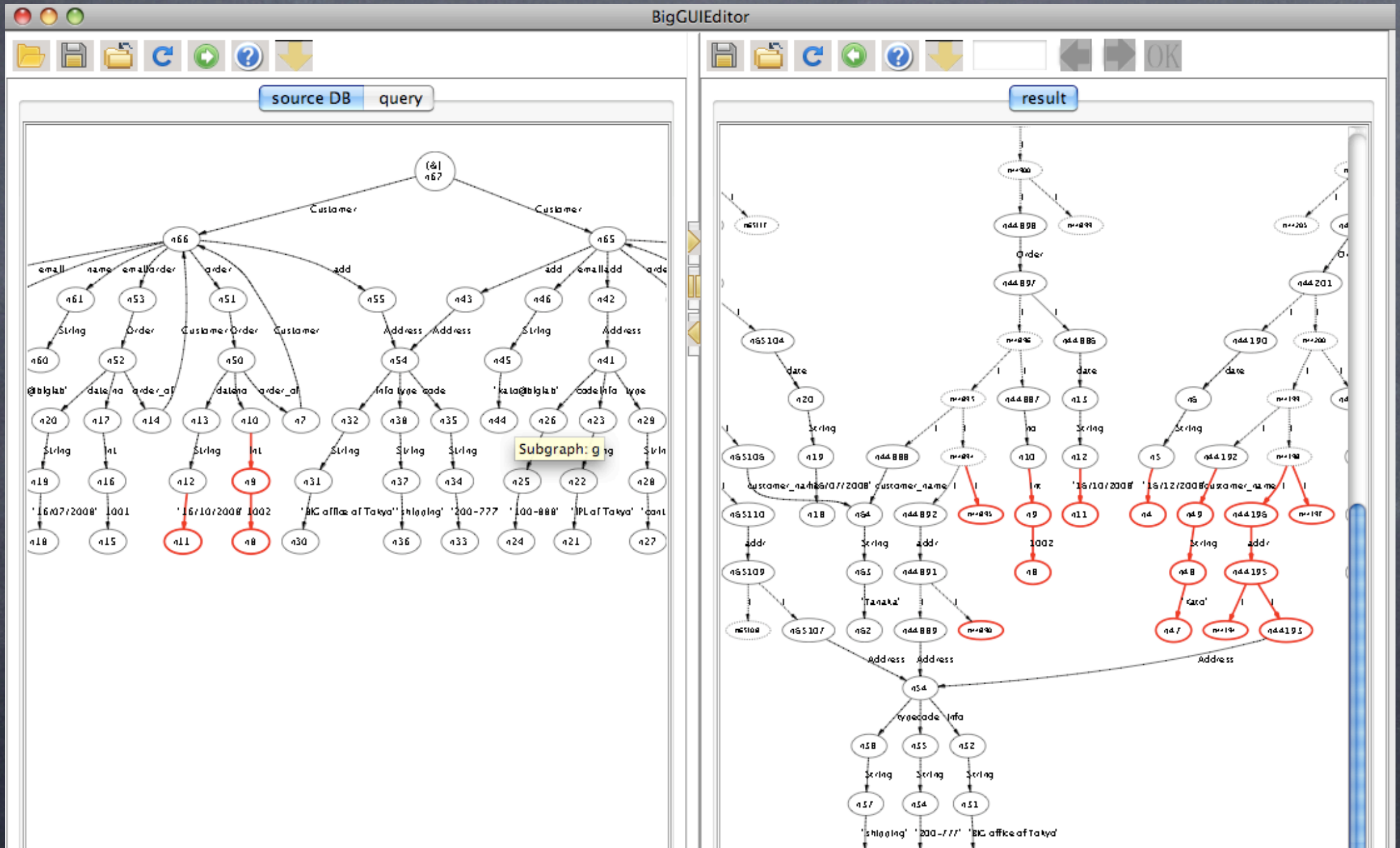
- select ... where ...
- replace ... by ... where ...

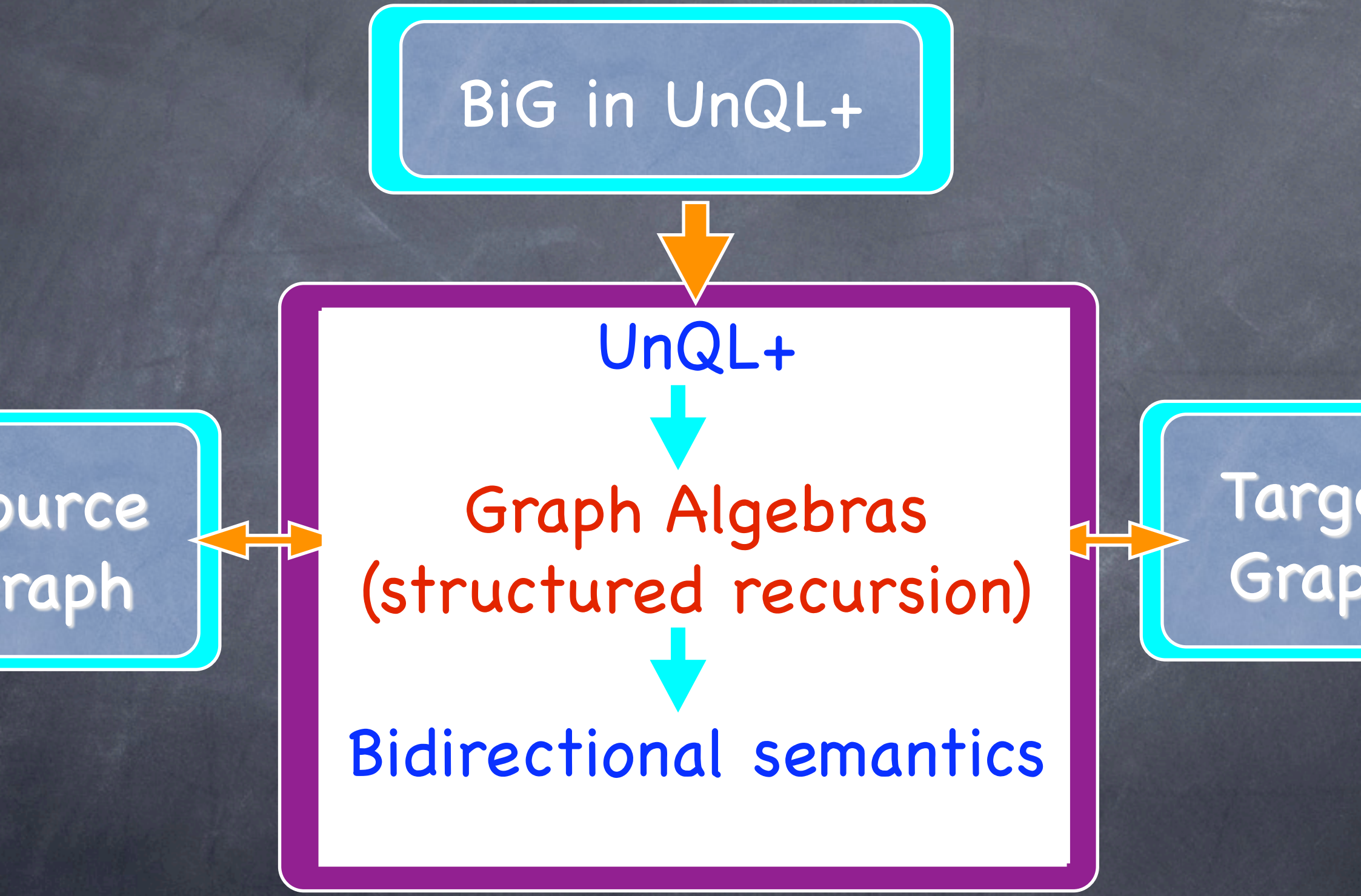
```
select {order : {date : $date,
                 no : $no,
                 name : $name,
                 addr : $a}}
where {customer.order : $o} in $customer_db,
      {order_of : $c, date : $date, no : $no} in $o,
      {add : $a, name : $name} in $c
```


customer
customer
order
order_of
add
email
name
add
add
order
order_of
order
order_of
email
email
name
no
date
info
code
type
"k@biglab"
"Kato"
info
code
type
no
date
no
date
"t@gmail"
"t@biglab"
"16/12/2008"
"IPL"
"100-888"
contractual
"BiG"
"200-777"
shipping
1002
"16/10/2008"
1001
"16/07/2008"
?
order
order
order
order
date
name
no
no
name
addr
addr
name
addr
no
date
addr
date
date
no
1004
name
"16/12/2008"
"Kato"
1003
1002
"Tanaka"
code
type
tel
info
1001
"16/10/2008"
"16/07/2008"
"Sato"
"200-777"
shipping
"03-1234-5678"
"BiG"


BiG in UnQL+
UnQL+
Graph Algebras
(structured recursion)
Bidirectional semantics
ource
raph
Targ
Grap

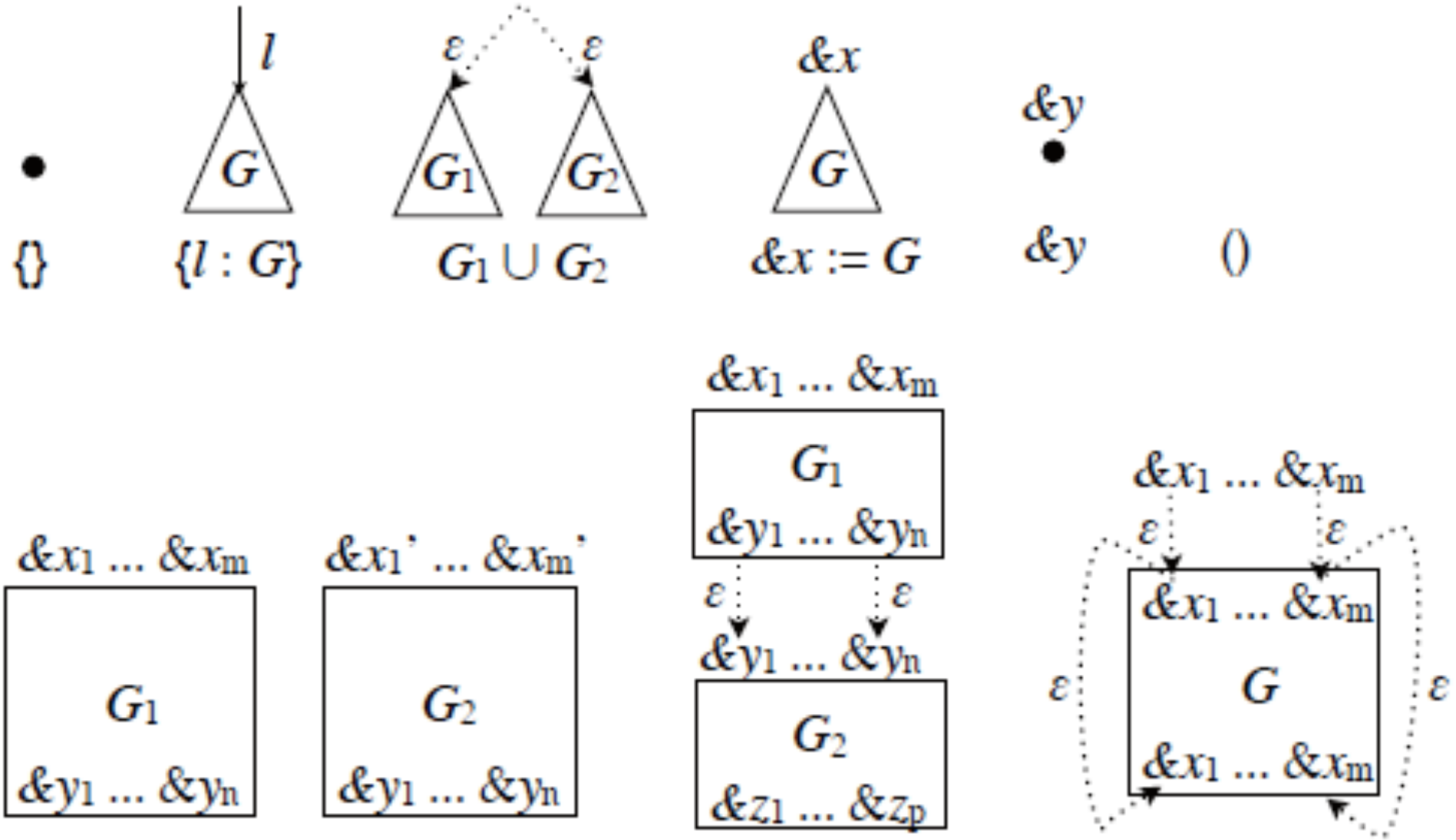
Simple Demos




BiG in UnQL+
UnQL+
Graph Algebras
(structured recursion)
Bidirectional semantics
ource
raph
Targ
Grap

Graph Algebra

Graph Constructors



Graph Algebra
Structured Recursion

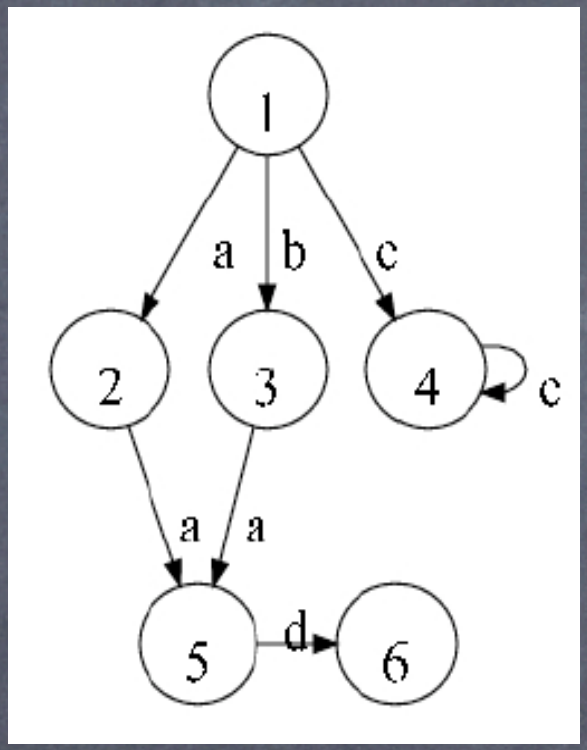
Structural Recursion:

$$\begin{aligned} f\ (\{\}) &= \{\} \\ f\ (\{l : g\}) &= l \odot f(g) \\ f\ (g_1 \cup g_2) &= f(g_1) \cup f(g_2) \end{aligned}$$

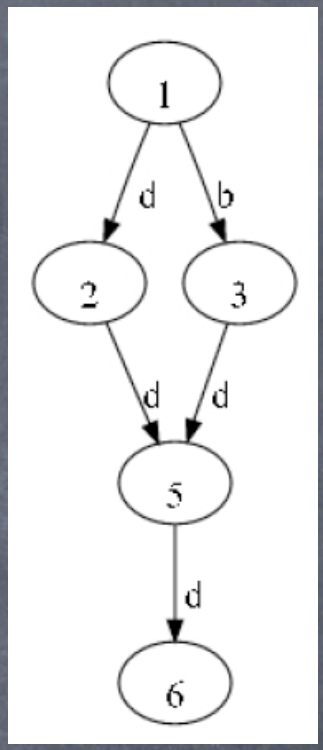
Or written as:

$$\text{sfun}\ f\ (\{l : g\}) = l \odot f(g)$$

Graph Algebra
Structured Recursion



a2d_xc



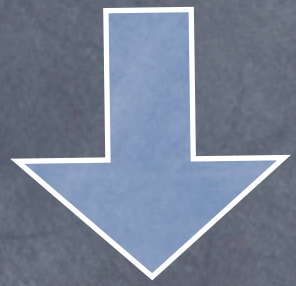
$$\begin{aligned} \text{sfun } a2d\_xc\ (\{l : g\}) \quad = \quad & \text{if } l = a \text{ then } \{d : a2d\_xc(g)\} \\ & \text{else if } l = c \text{ then } a2d\_xc(g) \\ & \text{else } \{l : a2d\_xc(g)\} \end{aligned}$$

Graph Algebra
UnCAL (Buneman et al. 2000)

$$
\begin{array}{lll}
::= & \{\} \mid \{l : e\} \mid e \cup e \mid \&x := e \mid \&y \mid () \mid e \oplus e \mid e \mathbin{@} e \mid \mathbf{cycle} & \\
\mid & \$g & \text{(* variable reference} \\
\mid & \mathbf{if}\ l = l\ \mathbf{then}\ e\ \mathbf{else}\ e & \text{(* conditional} \\
\mid & \mathbf{rec}(\lambda(\$l, \$g).e)(e) & \text{(* application of structural recursion} \\
::= & \$l & \text{(* label variable reference} \\
\mid & \mathtt{a} & \text{(* label } (a \in \mathit{Label})
\end{array}
$$

Desugaring [SAC'09]

Graph Transformation in UnQL+



Graph Transformation as composition of structured recursive functions

Bidirectional Semantics

Forward Evaluation

$\mathcal{F}[\![e]\!]\rho$ to produce a view graph

Backward Evaluation

$\mathcal{B}[\![e]\!](\rho, G')$ to produce a new environme

Bidirectional Semantics

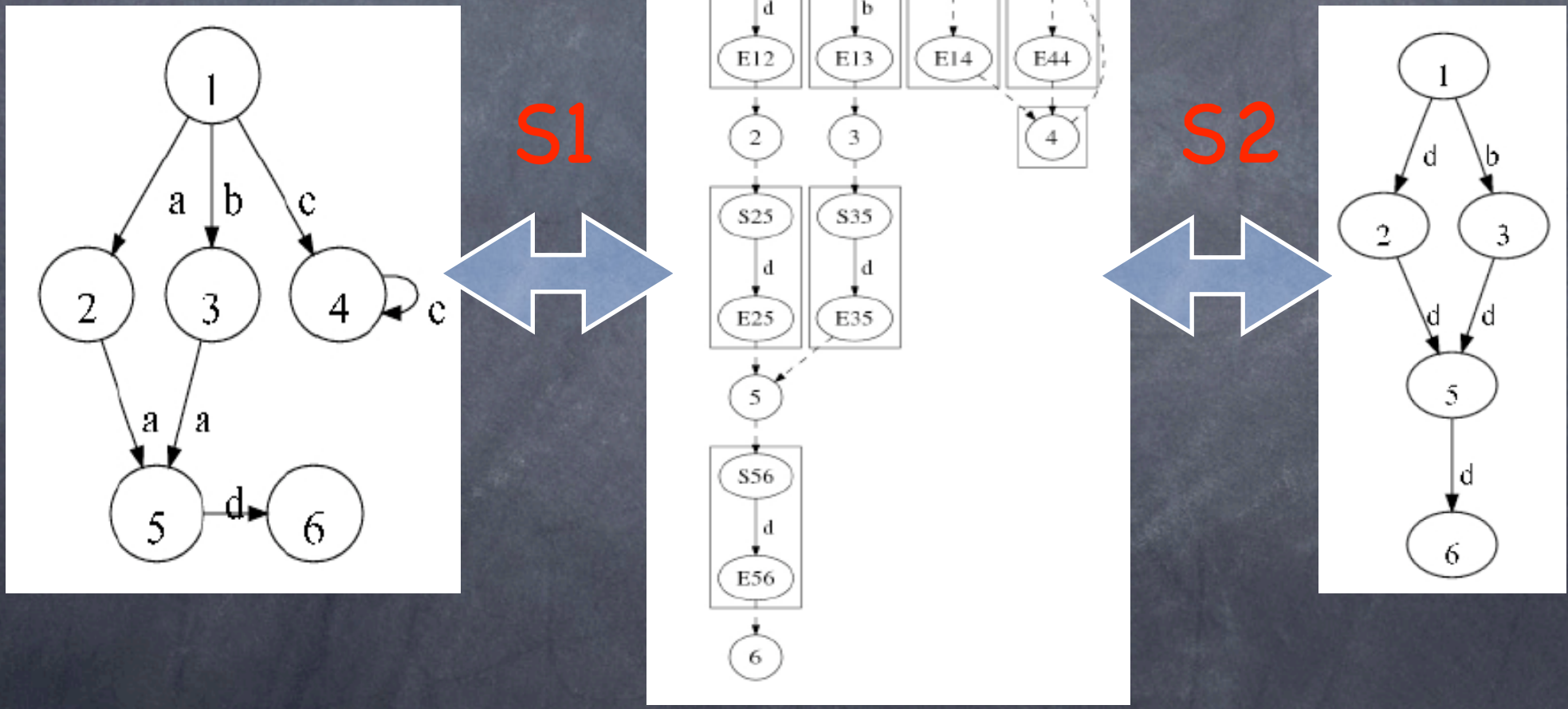
Bidirectional Properties

$$\frac{\mathcal{F}[\![e]\!]\rho = G}{\mathcal{B}[\![e]\!](\rho, G) = \rho}\,(\textsc{GetPut})$$

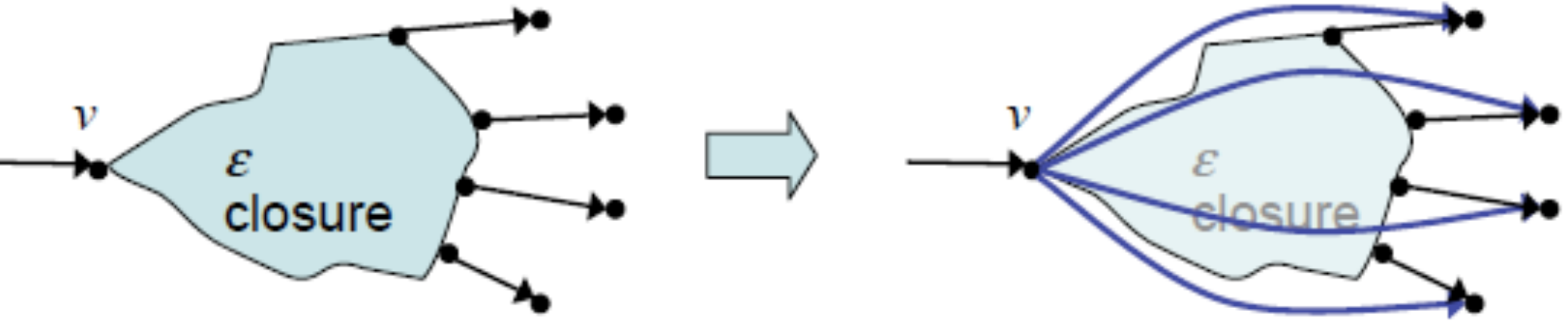
$$\frac{\mathcal{B}[\![e]\!](\rho, G') = \rho' \quad G' \in \mathrm{Range}(\mathcal{F}[\![e]\!])}{\mathcal{F}[\![e]\!]\rho' = G'}\,(\textsc{PutGet})$$

$$\frac{\mathcal{B}[\![e]\!](\rho, G') = \rho' \quad \mathcal{F}[\![e]\!]\rho' = G''}{\mathcal{B}[\![e]\!](\rho, G'') = \rho'}\,(\textsc{WPutGet})$$

Two-Stage Bidirectionalization



32. Epsilon-edge Elimination



Bidirectional Semantics

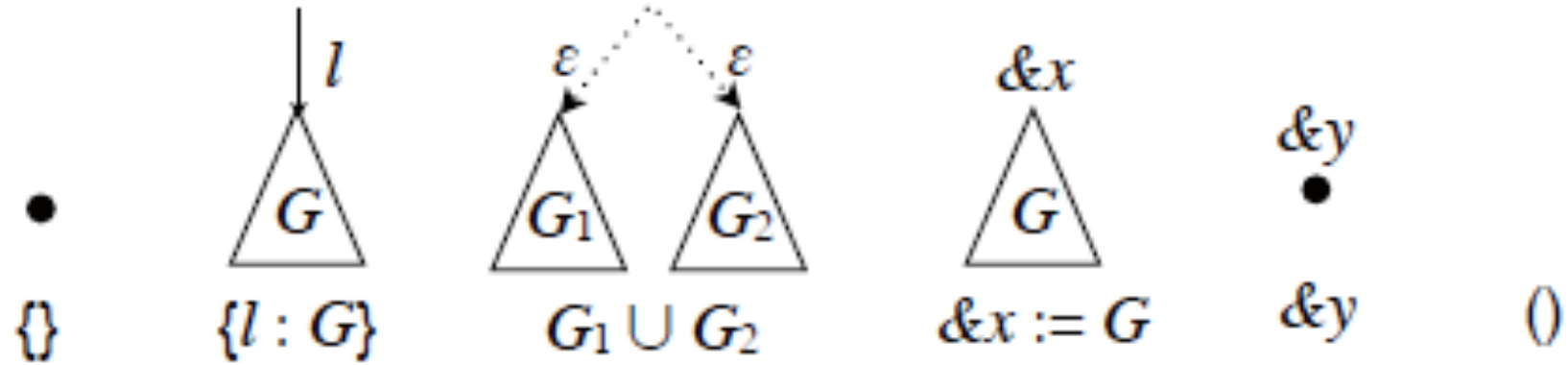
Variable

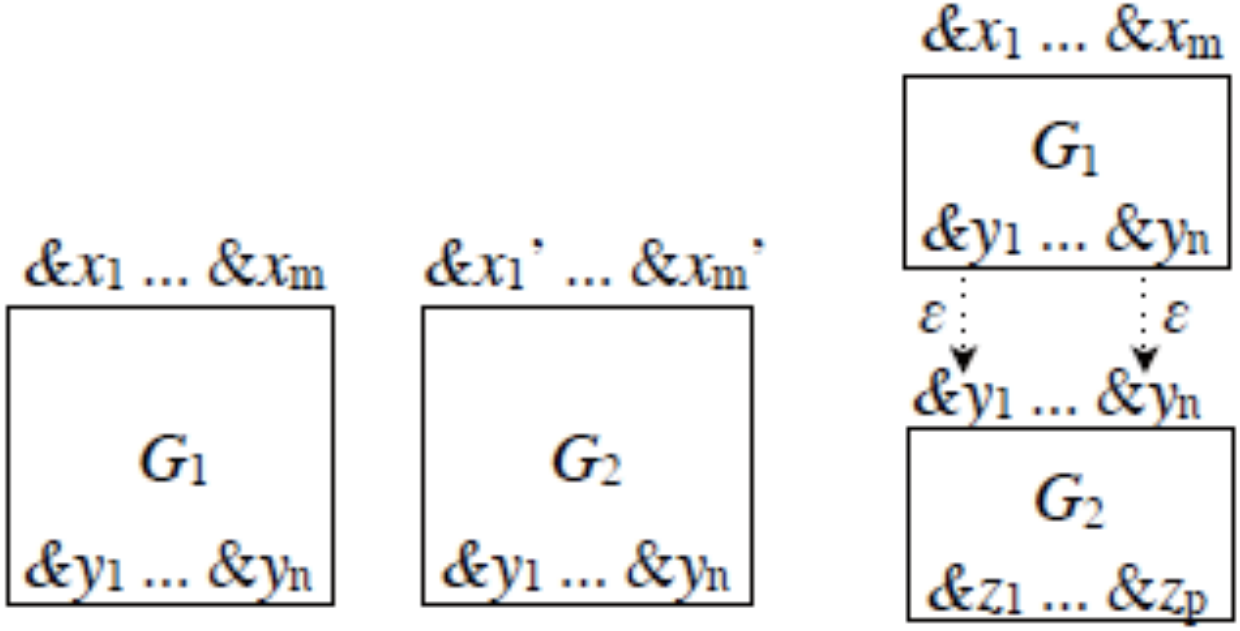
$$\mathcal{F}[\![(\$v)^p]\!]\rho = \rho(\$v)$$

$$\mathcal{B}[\![\$v]\!](\rho, G') = \rho[\$v \leftarrow G']$$

Bidirectional Semantics

Constructors





Bidirectional Semantics

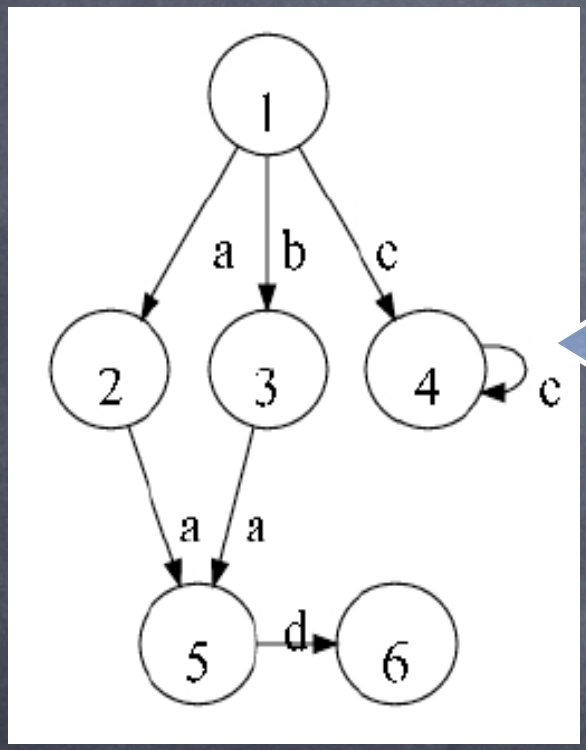
Condition

$$\mathcal{F}[\![(\textbf{if } l_1 = l_2 \textbf{ then } e_1 \textbf{ else } e_2)^p]\!]\rho = \begin{cases} \mathcal{F}[\![e_1]\!]\rho & \text{if } l_1\rho = l_2\rho \\ \mathcal{F}[\![e_2]\!]\rho & \text{otherwise} \end{cases}$$

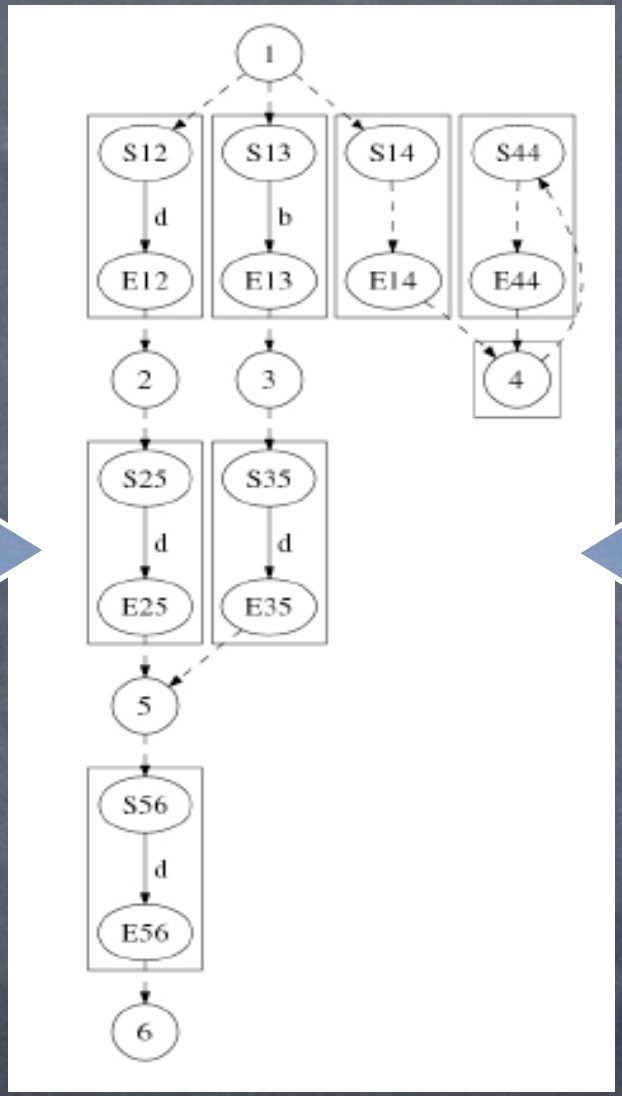
$$\mathcal{B}[\![\textbf{if } l_1 = l_2 \textbf{ then } e_1 \textbf{ else } e_2]\!](\rho, G') = \begin{cases} \rho_1' & \text{if } l_1\rho = l_2\rho \wedge l_1\rho_1' = l_2\rho_1' \\ \rho_2' & \text{if } l_1\rho \neq l_2\rho \wedge l_1\rho_2' \neq l_2\rho_2' \\ \textsc{Fail} & \text{otherwise} \end{cases}$$

$$\text{where } \rho_1' = \mathcal{B}[\![e_1]\!](\rho, G')$$
$$\rho_2' = \mathcal{B}[\![e_2]\!](\rho, G')$$

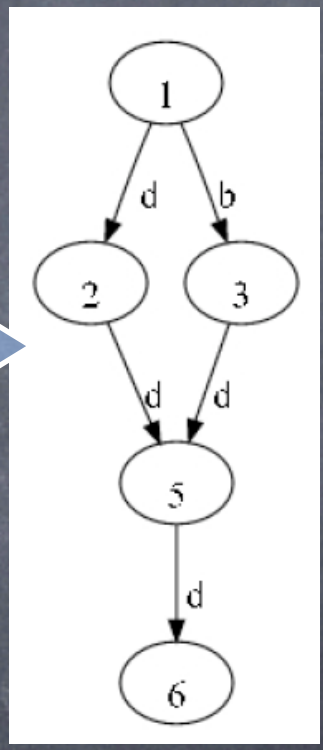
Bidirectionalizing Structured Recursion



S1



S2



$$\begin{array}{lll} \mathsf{sfun}\ \ a2d\_xc\ (\{l : g\}) & = & \mathsf{if}\ l = a\ \mathsf{then}\ \{d : a2d\_xc(g)\} \end{array}$$

Project Web Site

For more information, please visit the project page which contains all published papers as well as the source codes of the GRoudTram system.

Applications

- Many examples in the BiG website
- Applications in bidirectional model transformation (Software Engineering):
  - ASE'09, FSE'09, ICSE'09 (NIER track), models@runtime'09 (best paper)

1st GRACE International Meeting on Bidirectional Transformation

Bidirectional Transformations: A Cross-Discipline Perspective
-- GRACE meeting notes, state of the art, and outlook --,
International Conference on Model Transformation (ICMT 2009),
ETH Zurich, Switzerland, June 29-July 3 2009. LNCS 5563, Springer. pp.260-283.

Todo List

- How to type check graph transformation?
- How to type check bidirectional graph transformation (updability checking)?
- How to deal with selection of best updates (order on graph updating)?
- How to deal with high-order structural recursions (dealing join, zip, ...)?