Algebraic Graph Transformation and Model Transformation

Fernando Orejas

Royal Holloway University of London

on leave from Universitat Politècnica de Catalunya, Barcelona

1. Introduction
2. Specification of Model Transformations by Triple Graph Grammars
3. Implementing Model Transformations
4. Synchronizing Model Transformations

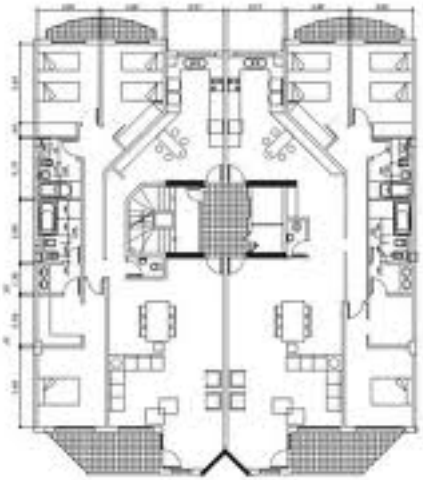
Introduction

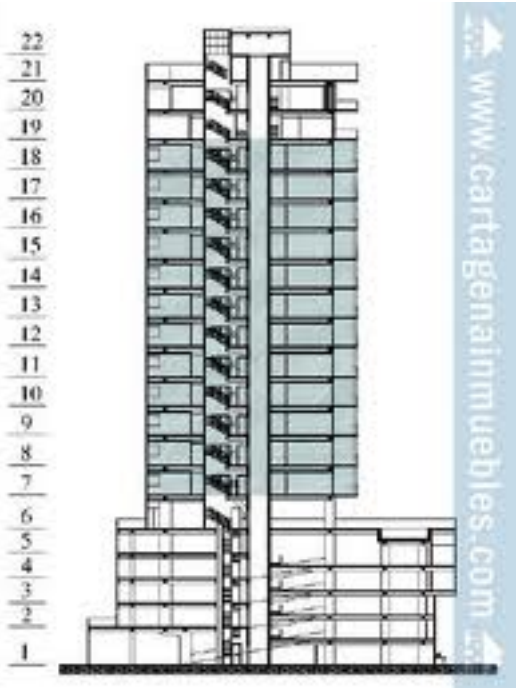
What is a model?

What is a model?

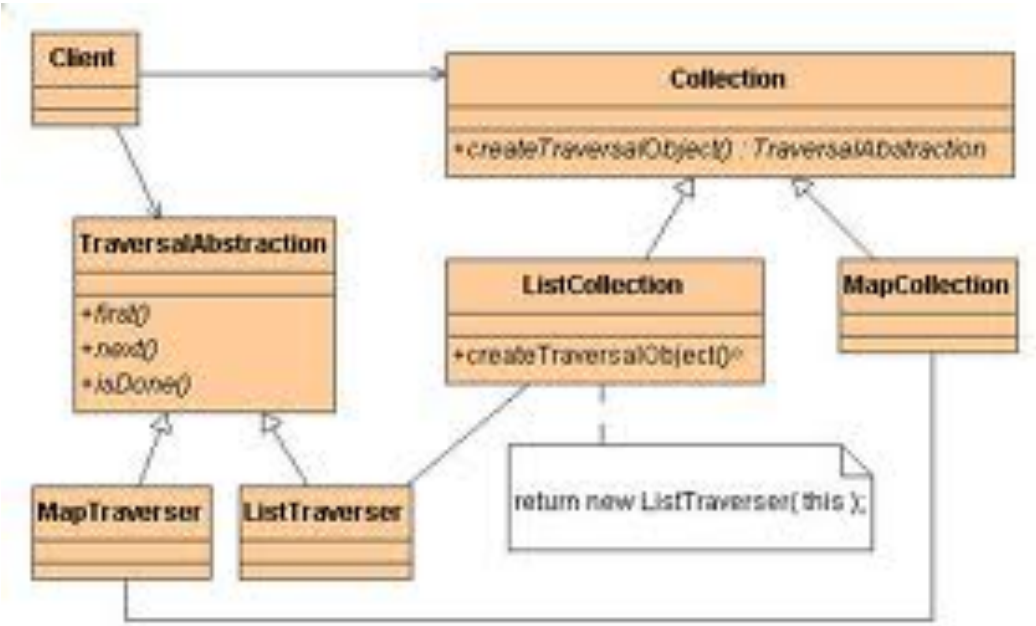
What is a model?

What is a model?

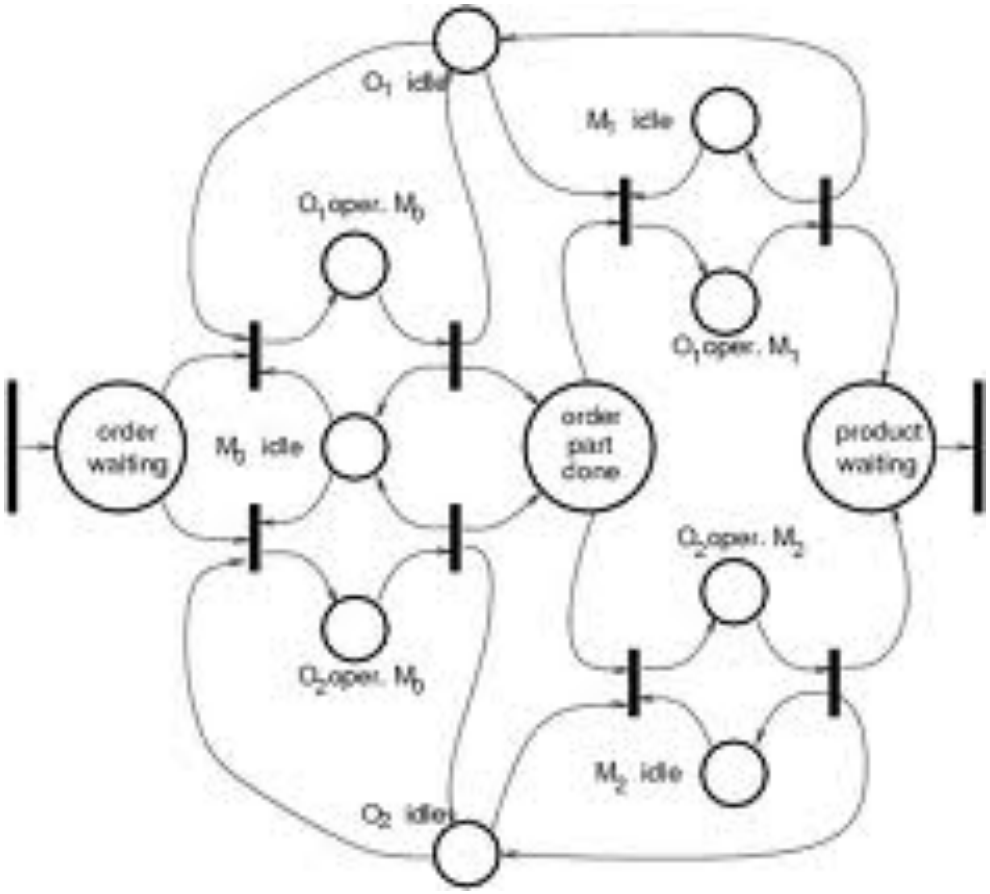




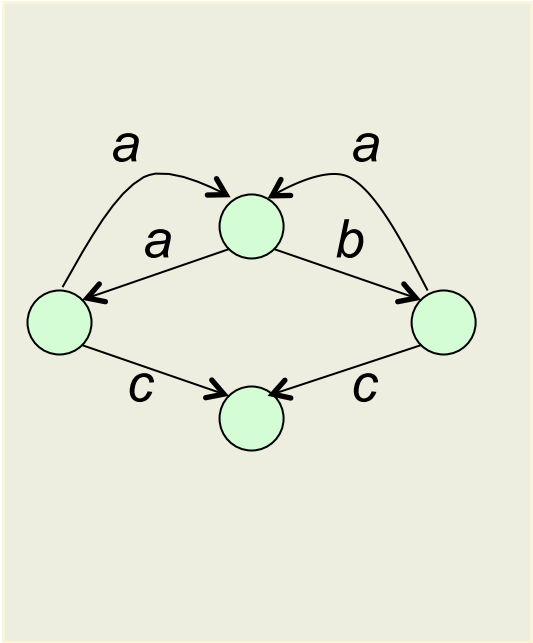
What is a model?



What is a model?

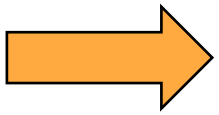


What is a model?

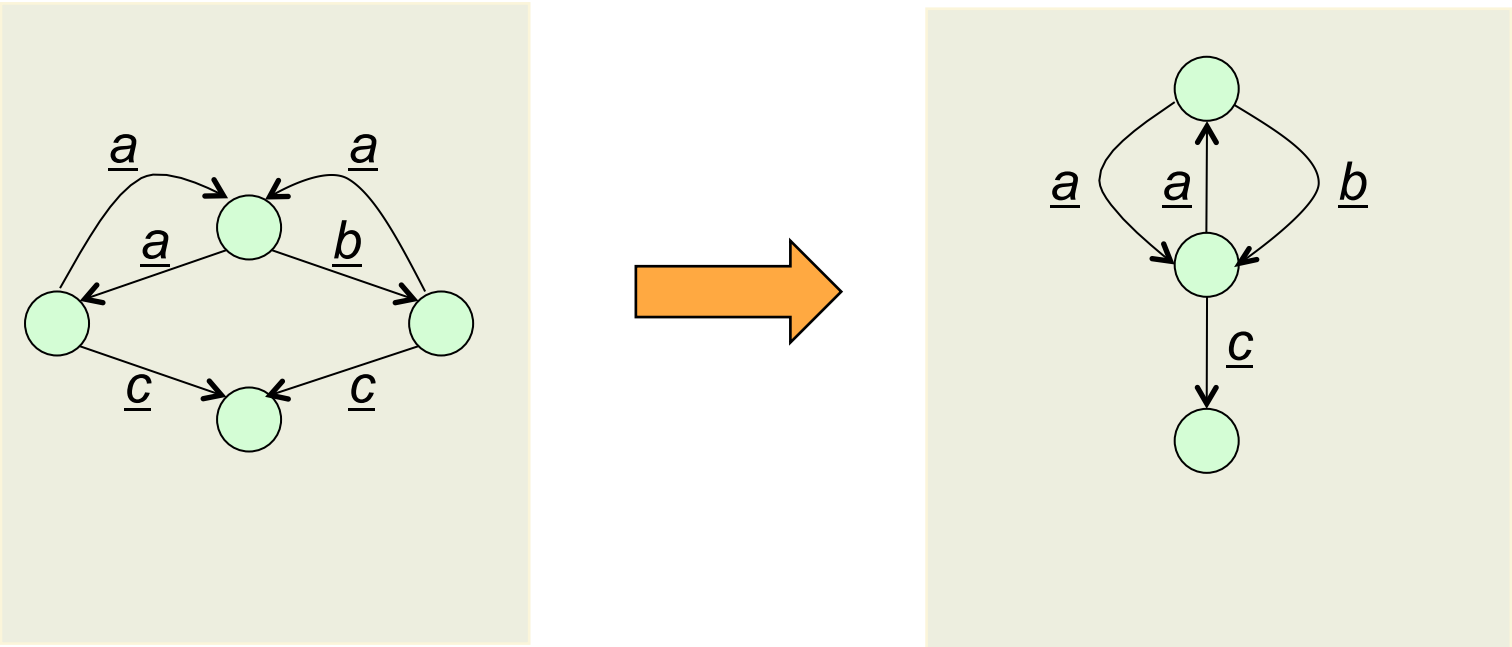


What is a model transformation?

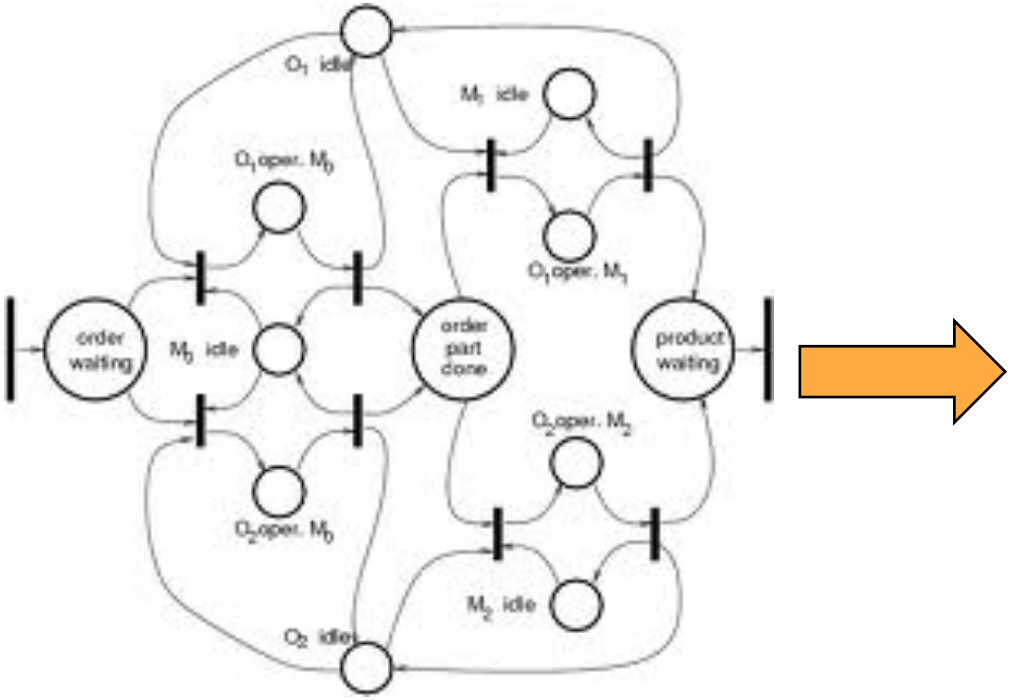
What is a model transformation?

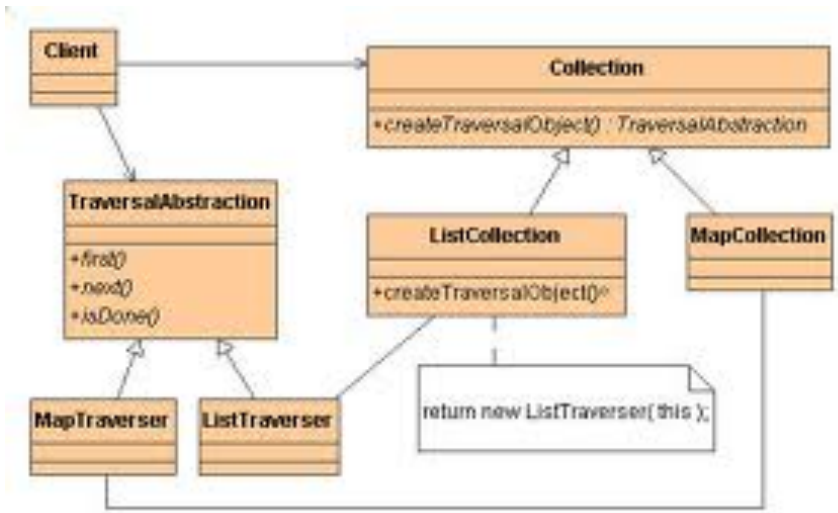


What is a model transformation?



What is a model transformation?





What is a model transformation?

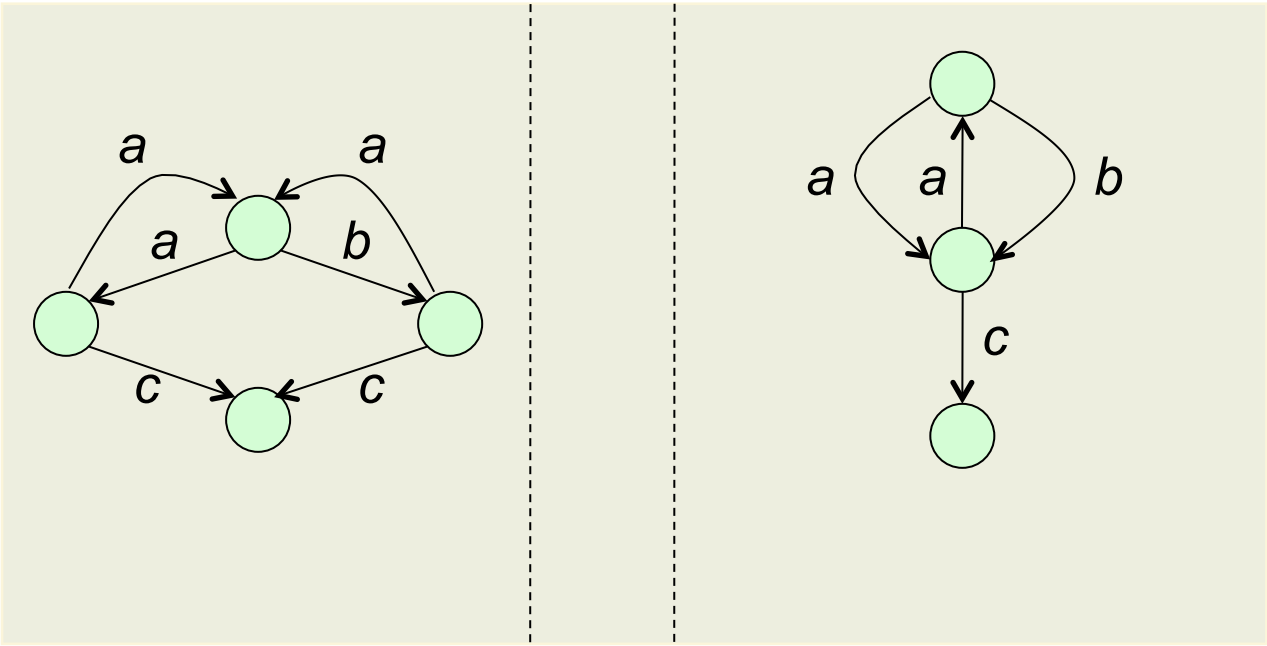
A model transformation is a procedure that given instances of a certain source model class it returns instances of a target model class.

A "standard language": QVT

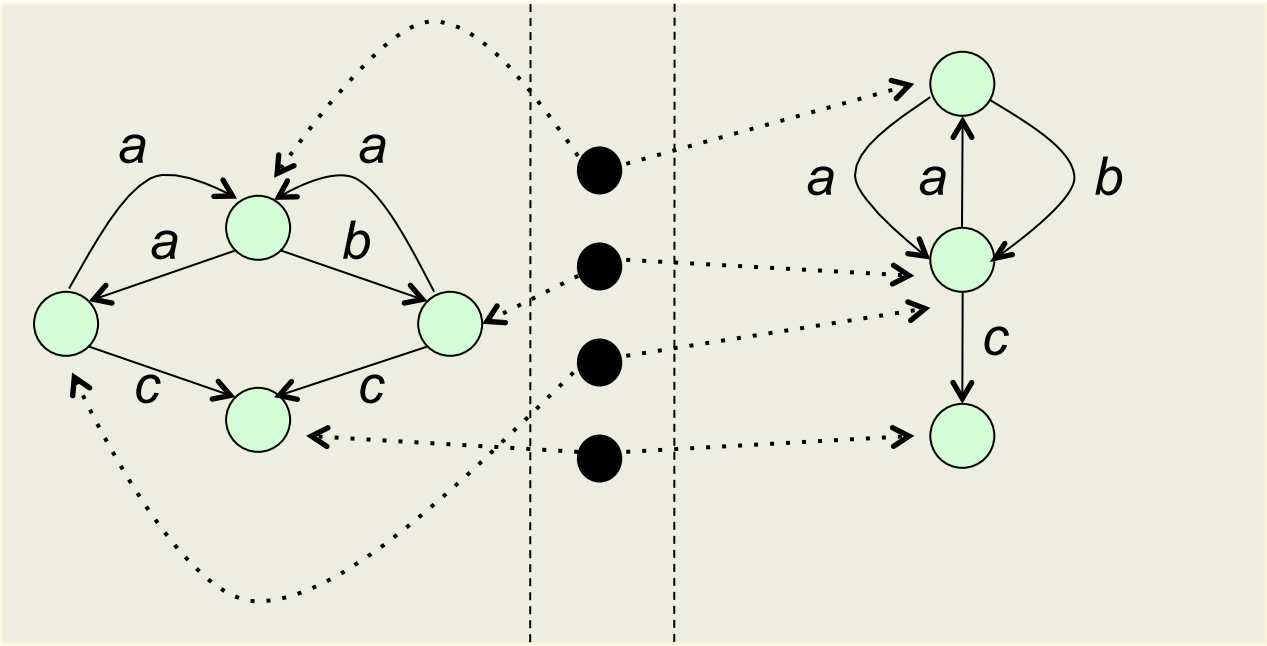
Kinds of Model Transformations:

- Monodirectional or Bidirectional
- Horizontal or Vertical
- Endogenous or exogenous
- From more abstract to more concrete
- From more concrete to more abstract

What is a model transformation?



What is a model transformation?



A model transformation is a set of triple graphs

$$G_S \xleftarrow{h_S} G_C \xrightarrow{h_T} G_T$$

Triple graph morphisms:

$$\begin{tikzcd}
M_S \arrow[d, "g_S"'] & M_C \arrow[l, "h_S"'] \arrow[r, "h_T"] \arrow[d, "g_C"'] & M_T \arrow[d, "g_T"'] \\
M_S & M_C \arrow[l, "h_S"'] \arrow[r, "h_T"] & M_T
\end{tikzcd}$$

Problems

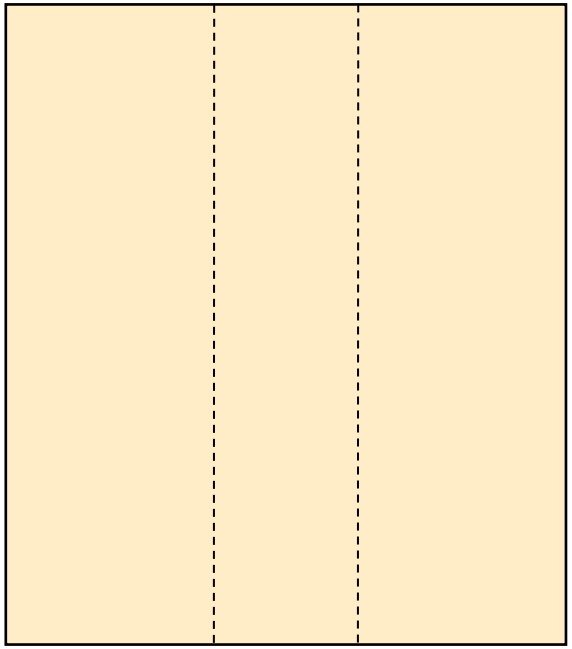
- Specification
- Implementation
- Synchronization

Specification of model Transformations by Triple Graph Grammars

(Many) Classes classes of model transformations can be naturally specified by triple graph grammars (TGGs).

A TGG consists of:

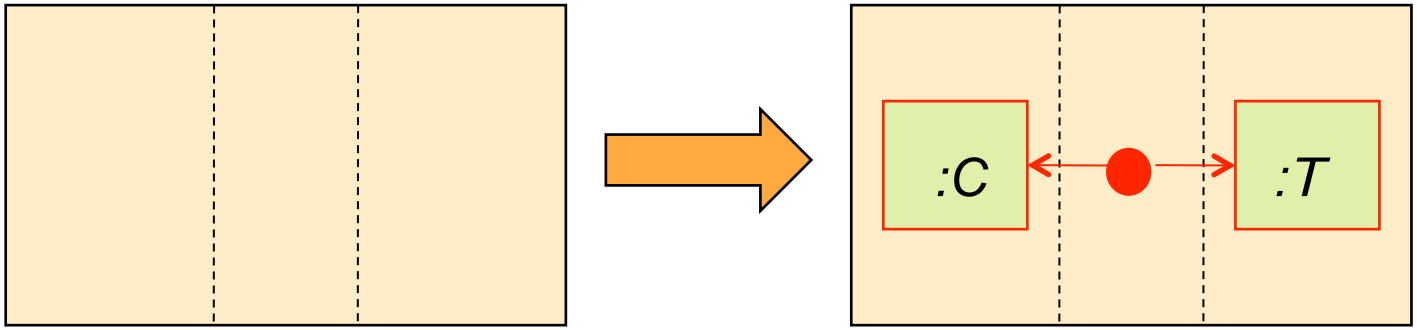
A starting triple graph



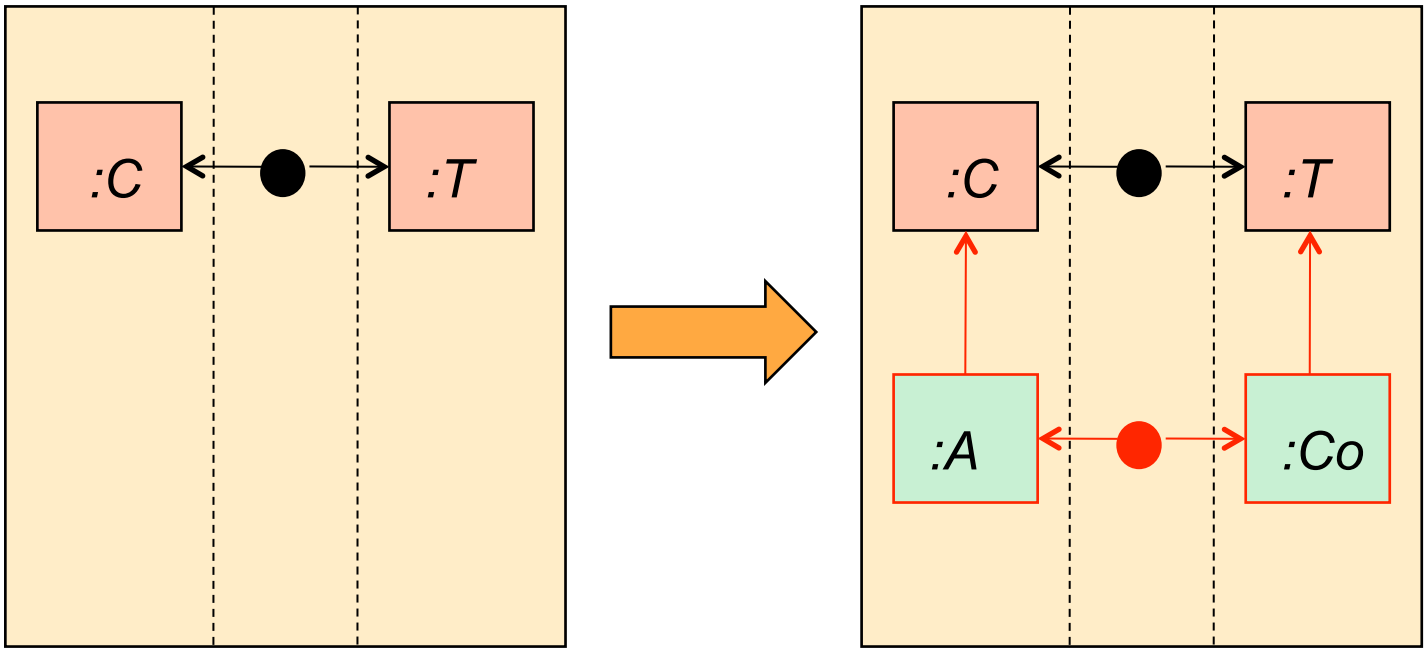
A triple graph transformation rule is:

$$p = L \longrightarrow R$$

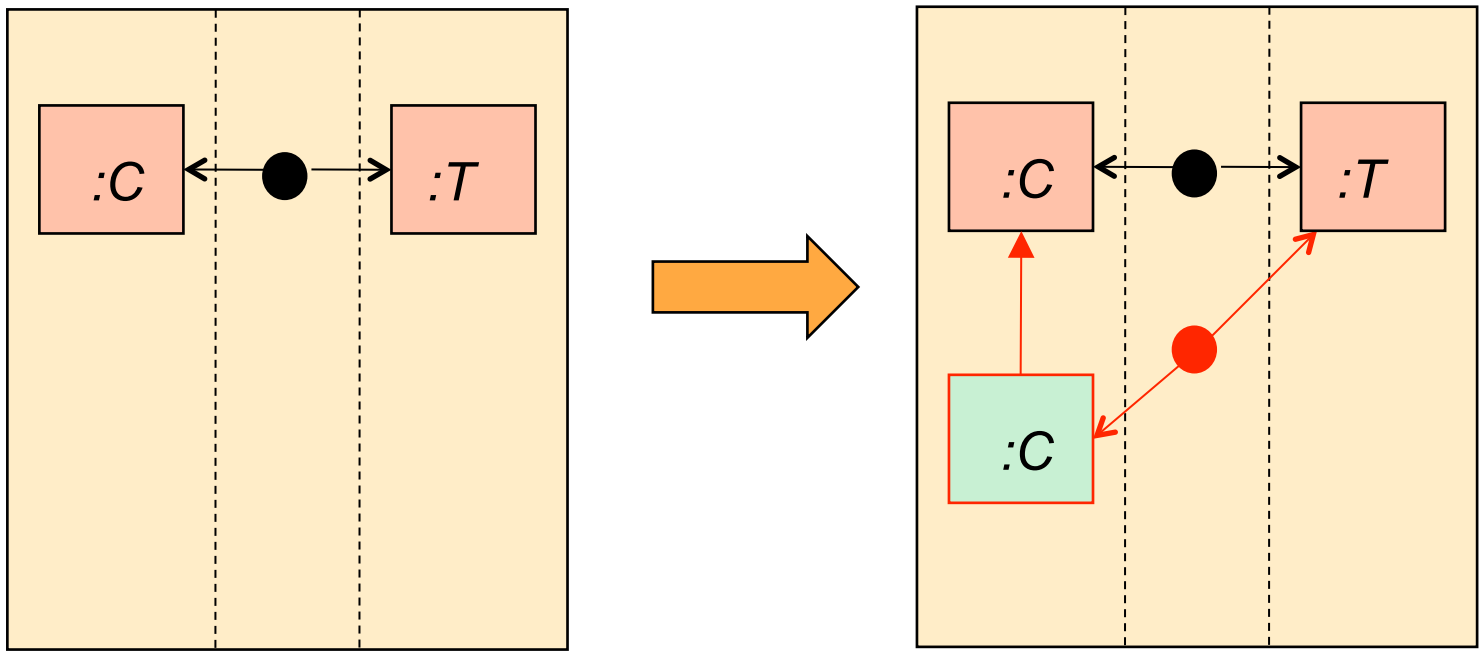
Transformation Rules



class-table creation



attribute-column creation

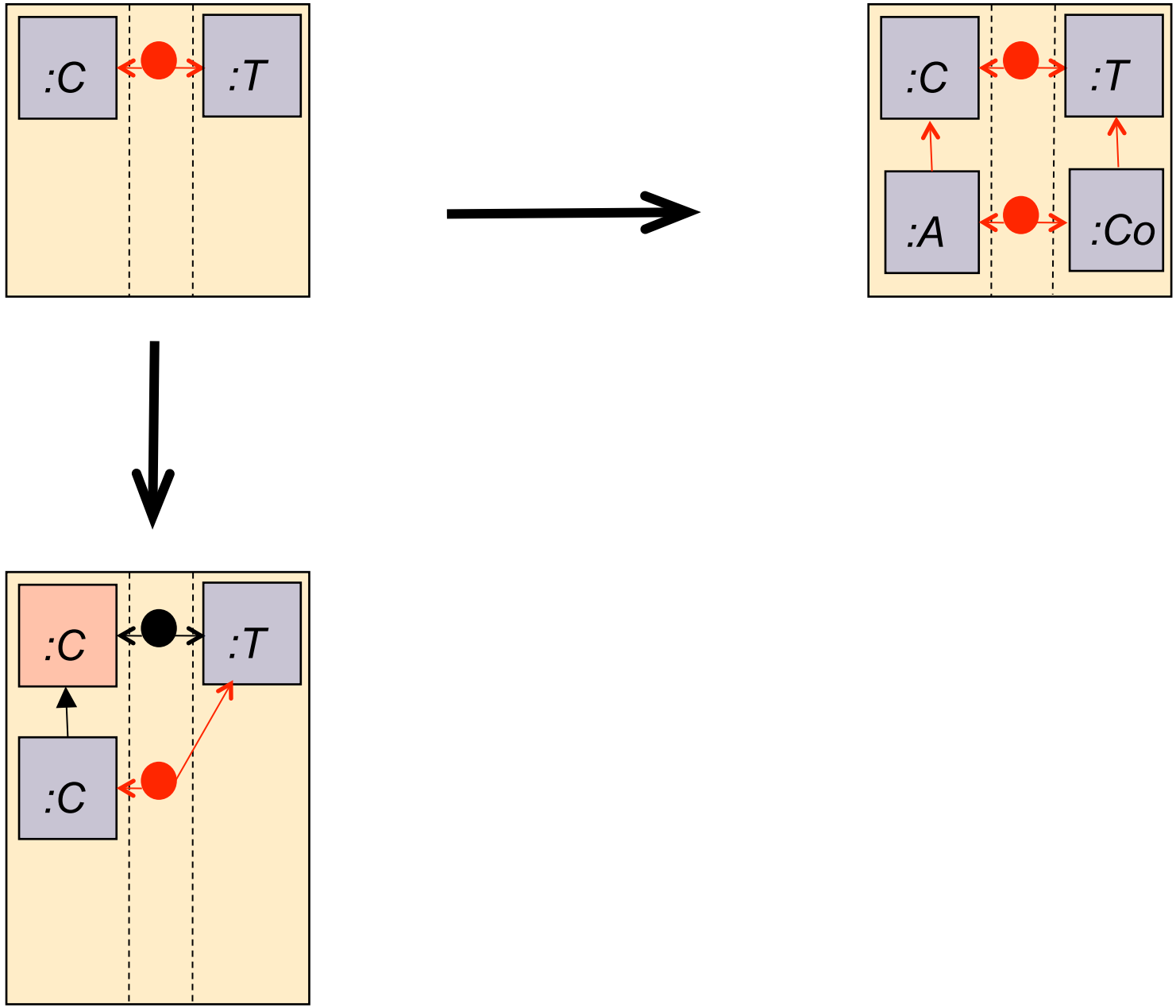


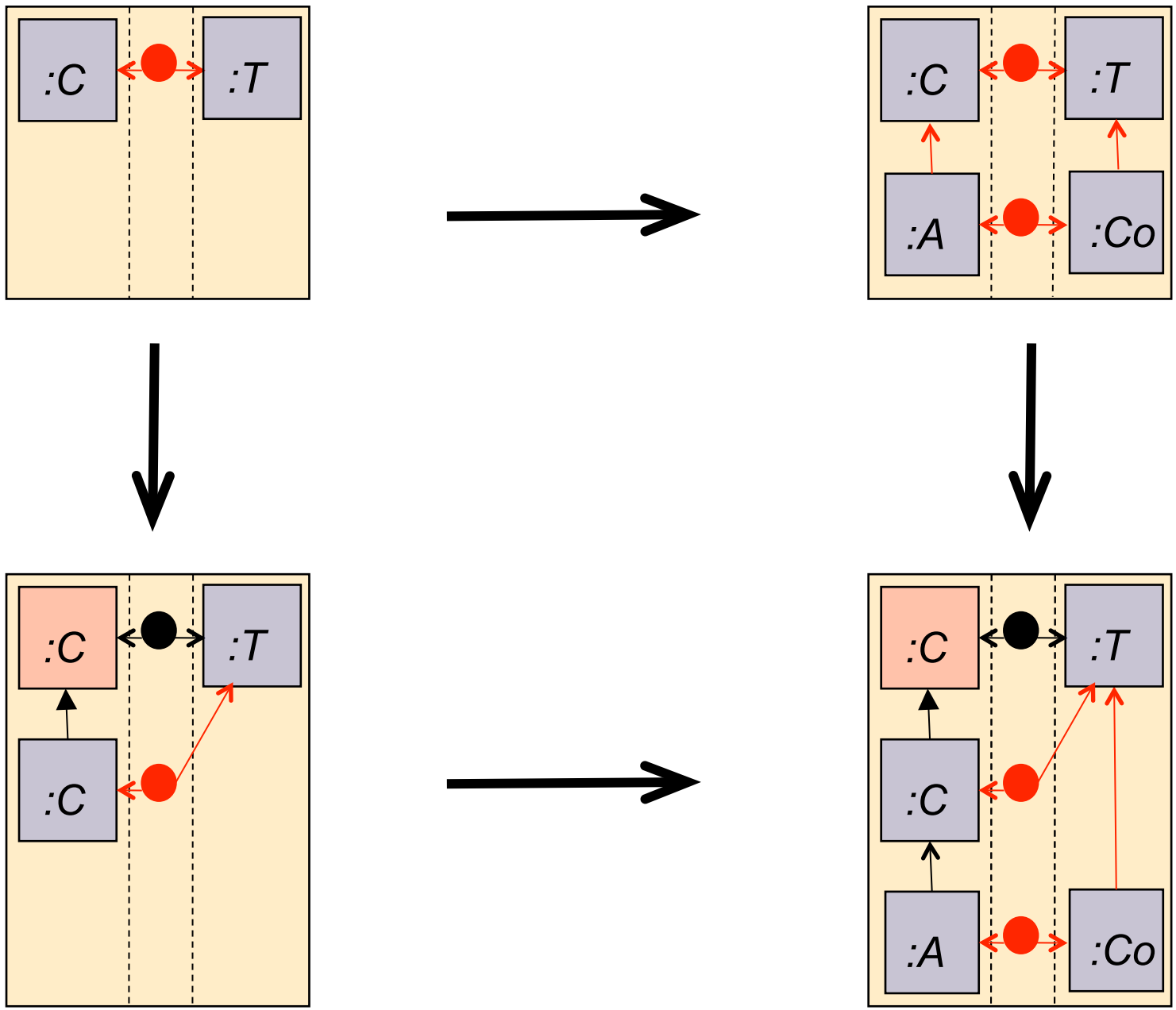
Subclass creation

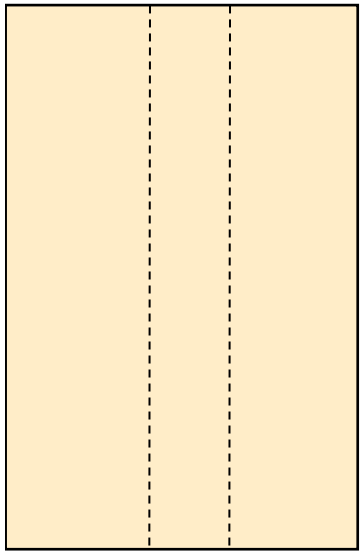
The model transformations specified by a TGG is the class of triple graphs that can be generated by the grammar.

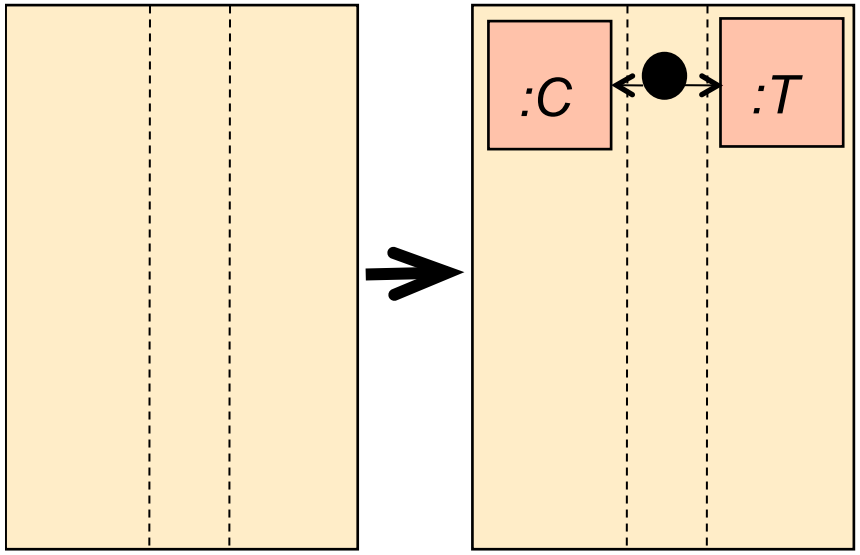
Rule application

$$\begin{tikzcd} L \arrow[r, "r"] \arrow[d, "m"'] & R \arrow[d, "m'"] \\ G \arrow[r] & H \end{tikzcd}$$

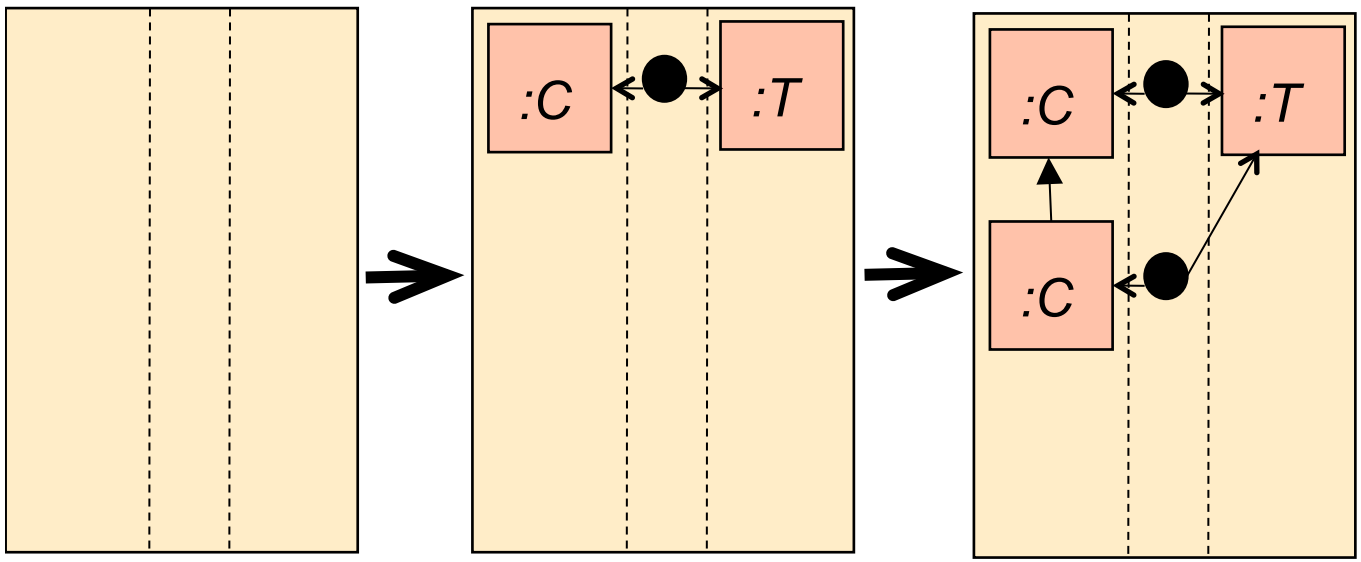

:C
:T
:C
:T
:A
:Co
:C
:T
:C


:C
:T
:C
:T
:A
:Co
:C
:T
:C
:C
:T
:C
:A
:Co



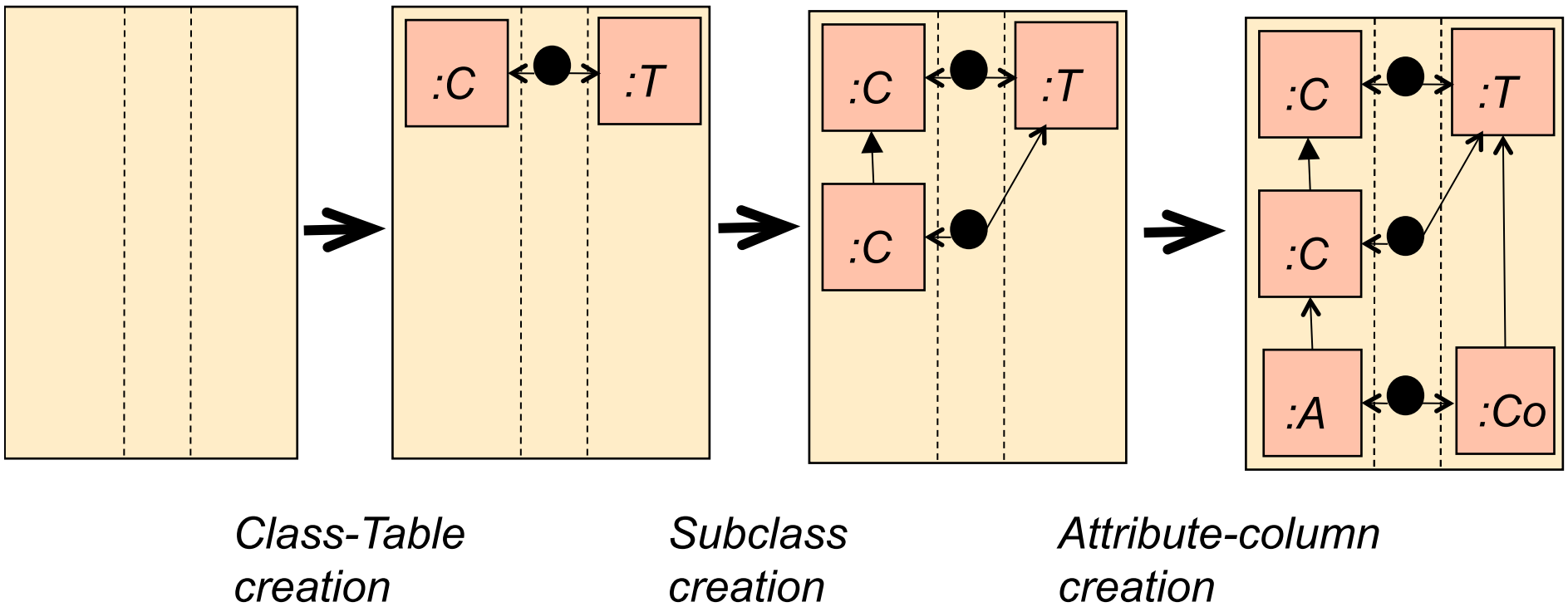


Class-Table creation

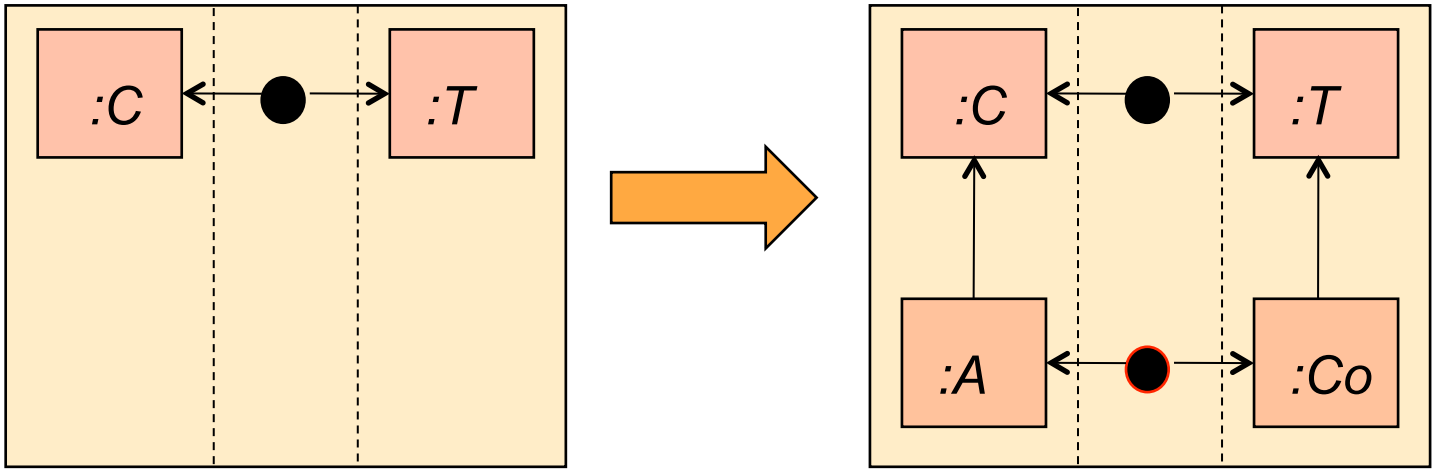


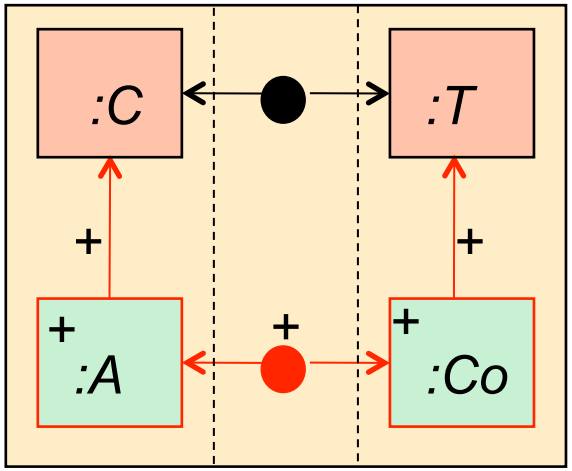
Class-Table creation

Subclass creation


:C
:T
:C
:T
:C
:C
:T
:C
:A
:Co
Class-Table creation
Subclass creation
Attribute-column creation

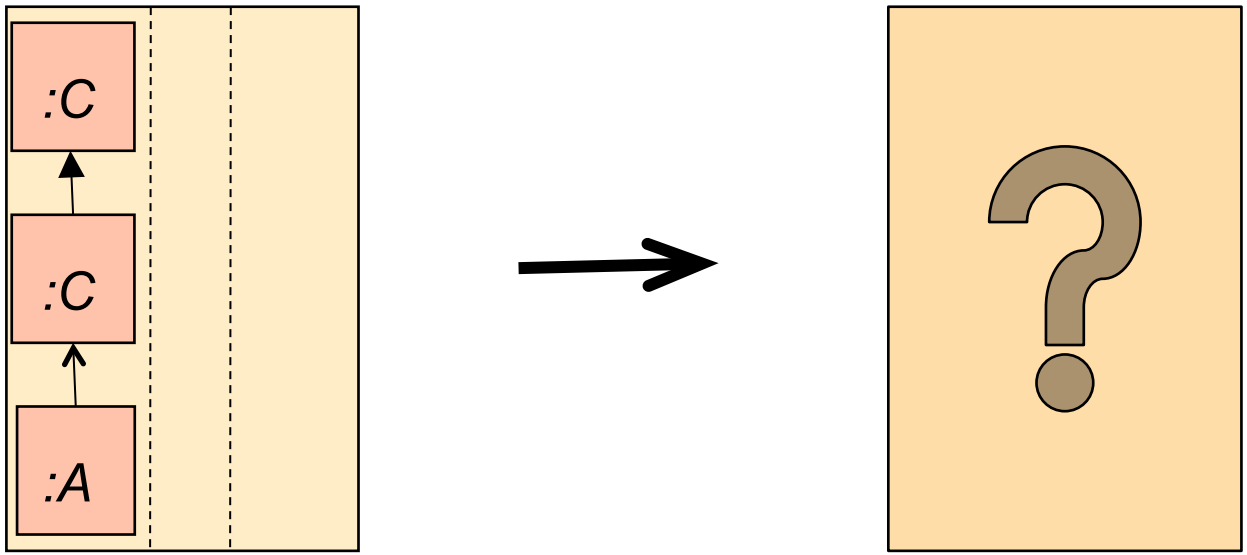
Compact Notation of Transformation Rules





Implementation

The problem:



Given a TGG rule

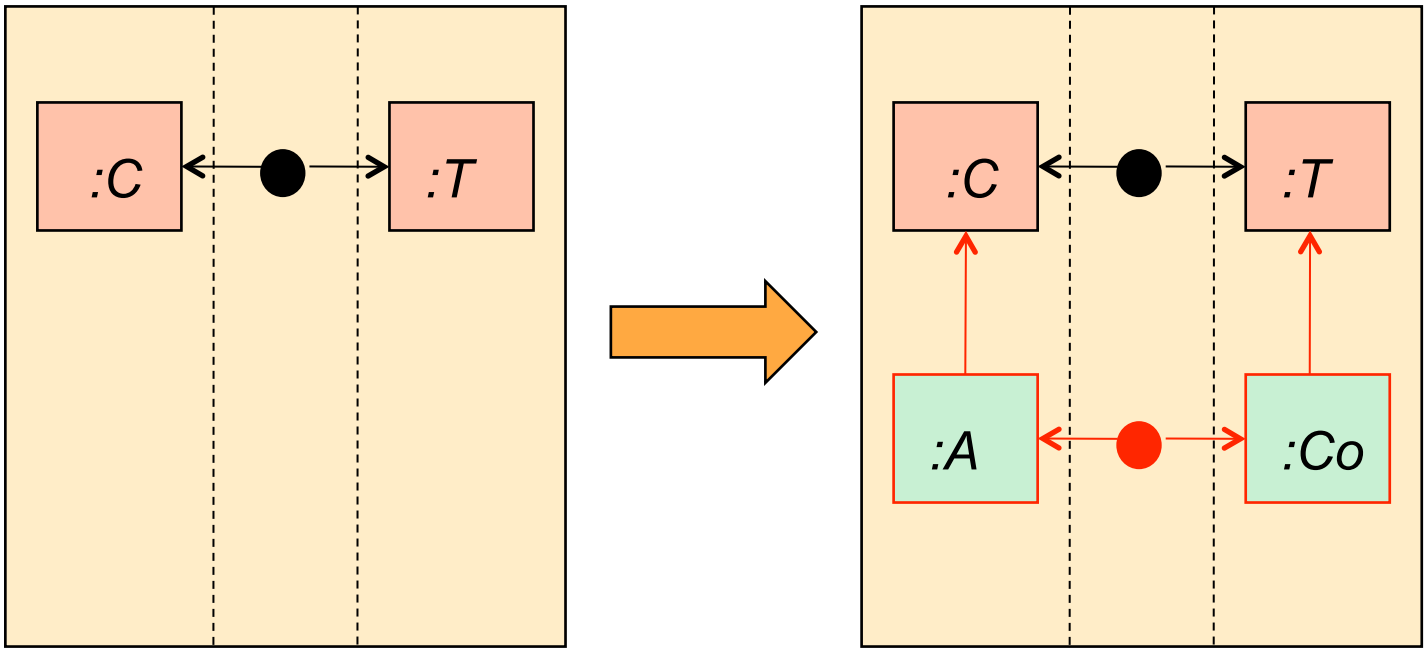
$$\begin{tikzcd}
L_S \arrow[d] & L_C \arrow[l] \arrow[r] \arrow[d] & L_T \arrow[d] \\
R_S & R_C \arrow[l] \arrow[r] & R_T
\end{tikzcd}$$

we can obtain the following operational rules

Source Rule

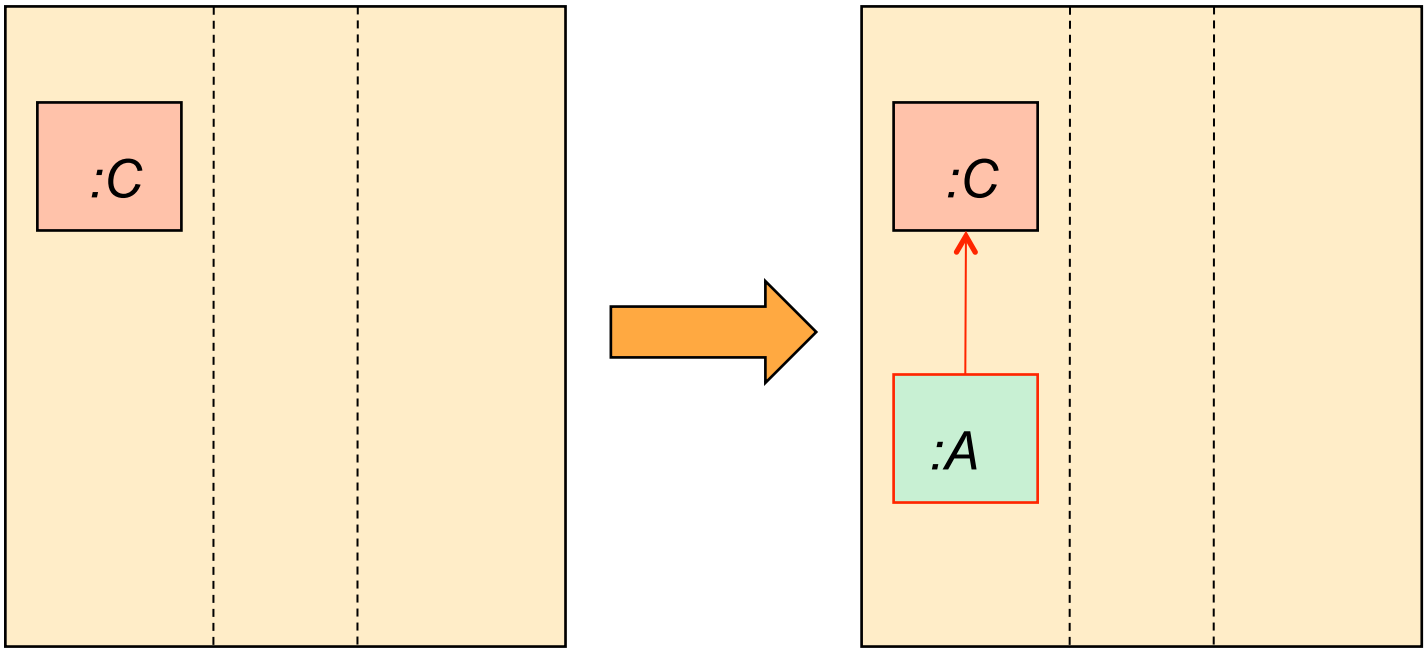
$$\begin{tikzcd}
L_S \arrow[d] & \emptyset \arrow[l] \arrow[r] \arrow[d] & \emptyset \arrow[d] \\
R_S & \emptyset \arrow[l] \arrow[r] & \emptyset
\end{tikzcd}$$

Rule



attribute-column creation

Source Rule



attribute-column creation

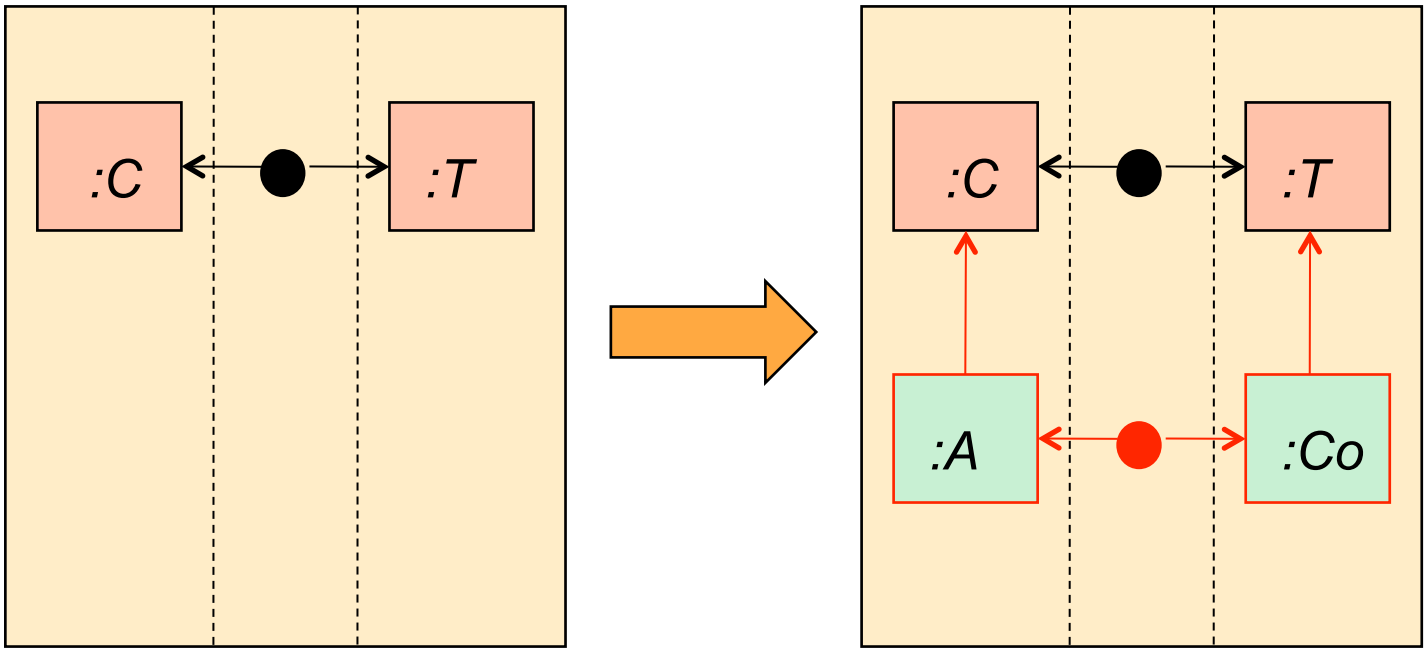
Given a TGG rule

$$\begin{tikzcd}
L_S \arrow[d] & L_C \arrow[l] \arrow[r] \arrow[d] & L_T \arrow[d] \\
R_S & R_C \arrow[l] \arrow[r] & R_T
\end{tikzcd}$$

Forward Rule

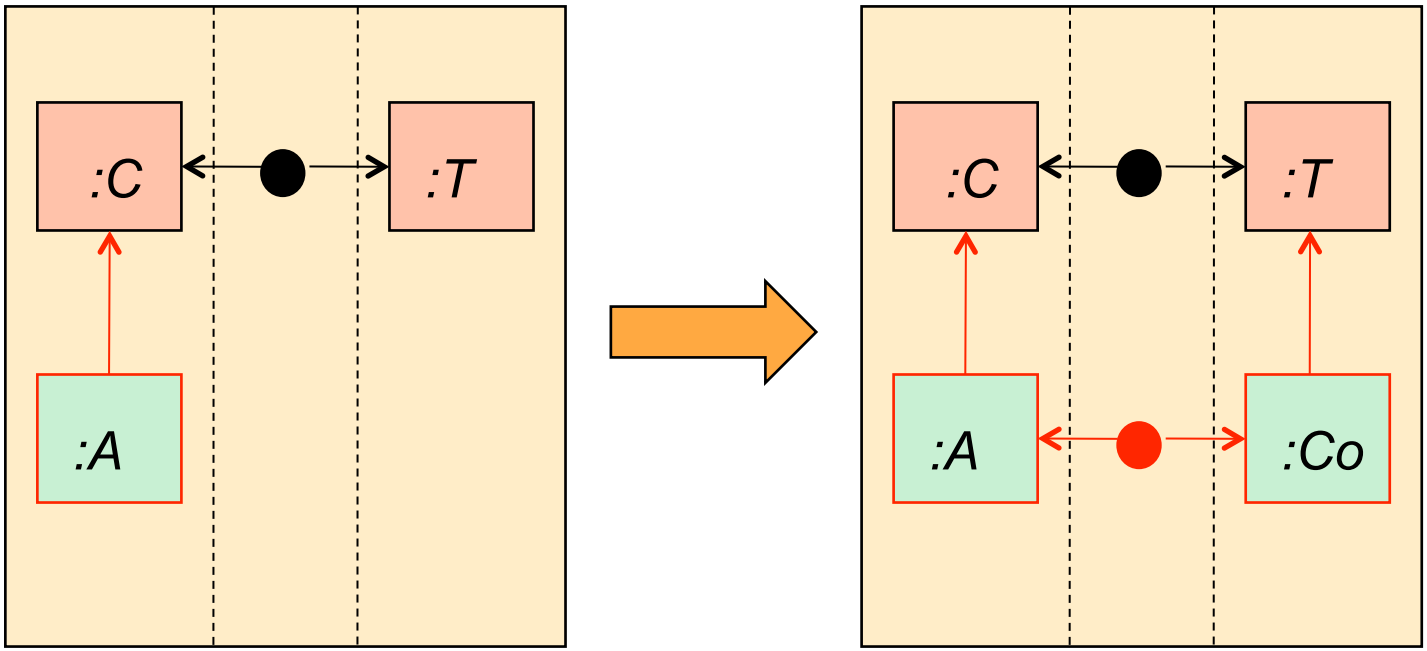
$$\begin{tikzcd}
R_S \arrow[d] & L_C \arrow[l] \arrow[r] \arrow[d] & L_T \arrow[d] \\
R_S & R_C \arrow[l] \arrow[r] & R_T
\end{tikzcd}$$

Rule



attribute-column creation

Forward rule



attribute-column creation

Fact:

$$G \Longrightarrow_{p} G'$$

If and only if

$$G \Longrightarrow_{p^S} G'' \Longrightarrow_{p^F} G'$$

Implementing Model Transformations:

$$(\emptyset, \emptyset, \emptyset) \Longrightarrow_{p_1} G_1 \Longrightarrow_{p_2} \cdots \Longrightarrow_{p_n} G$$

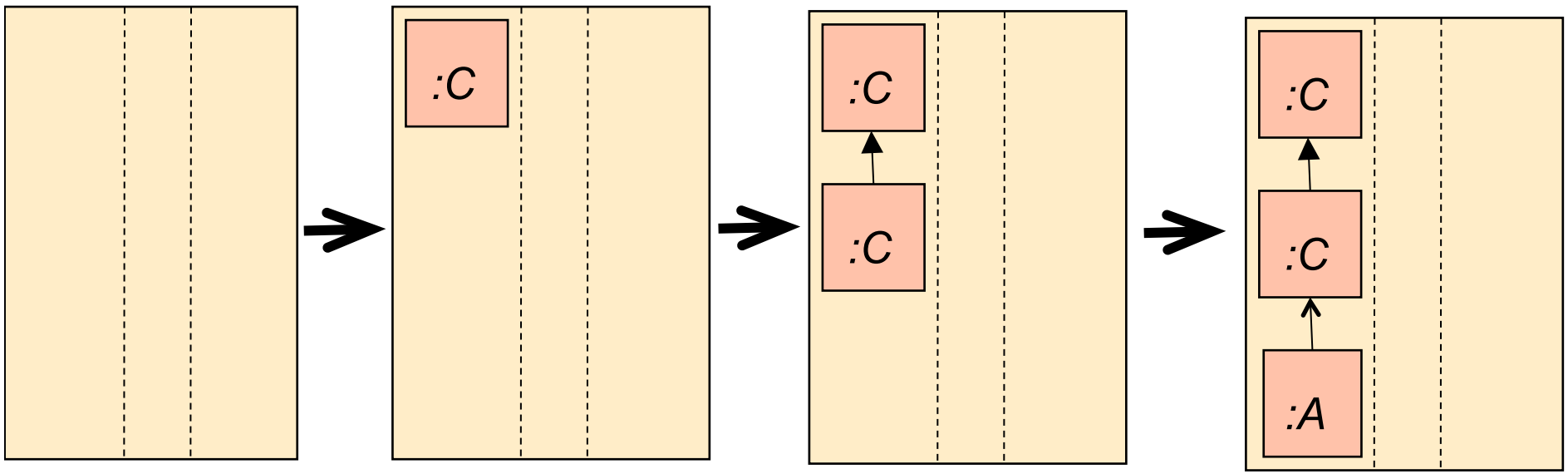
If and only if

$$(\emptyset, \emptyset, \emptyset) \Longrightarrow_{p_1^S} G'_1 \Longrightarrow_{p_1^F} G_1 \Longrightarrow_{p_n^S} \cdots G_{n-1} \Longrightarrow_{p_1^S} G'_n \Longrightarrow_{p_n^F} G$$

If and only if

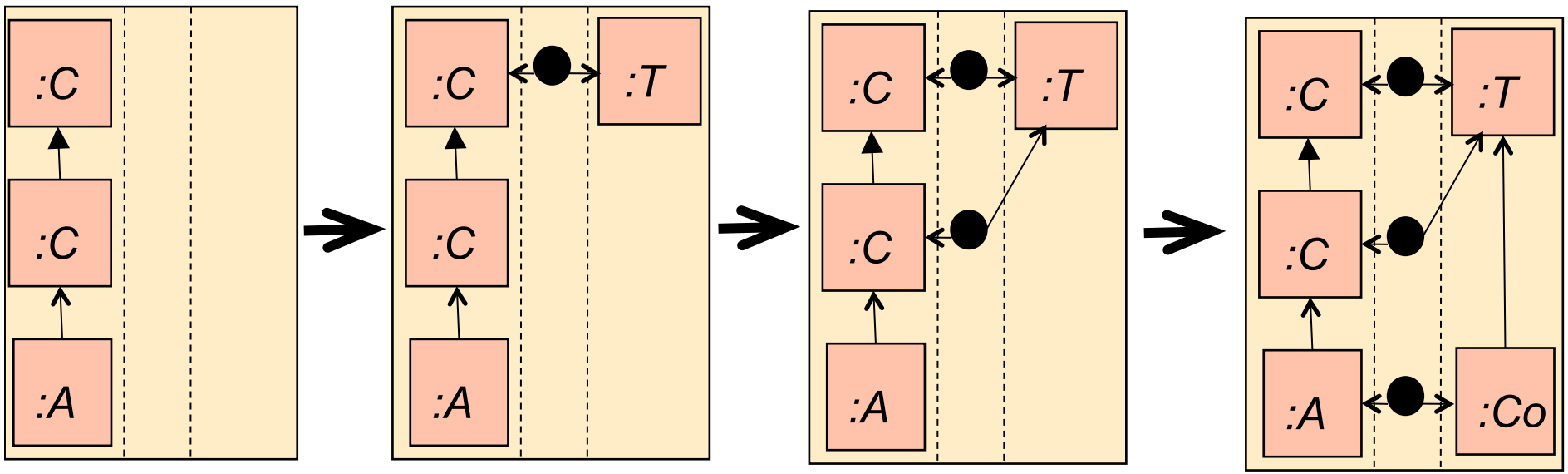
$$(\emptyset, \emptyset, \emptyset) \Longrightarrow_{p_1^S} G''_1 \Longrightarrow_{p_2^S} \cdots \Longrightarrow_{p_n^S} G' \Longrightarrow_{p_1^F} G'_1 \Longrightarrow_{p_2^F} \cdots \Longrightarrow_{p_n^F} G$$

Example



Rule 1^{S} *Rule 3*^{S} *Rule 2*^{S}

Example



Rule 1^F *Rule 3^F* *Rule 2^F*

Consistency of Forward Transformation Sequences

A forward transformation sequence

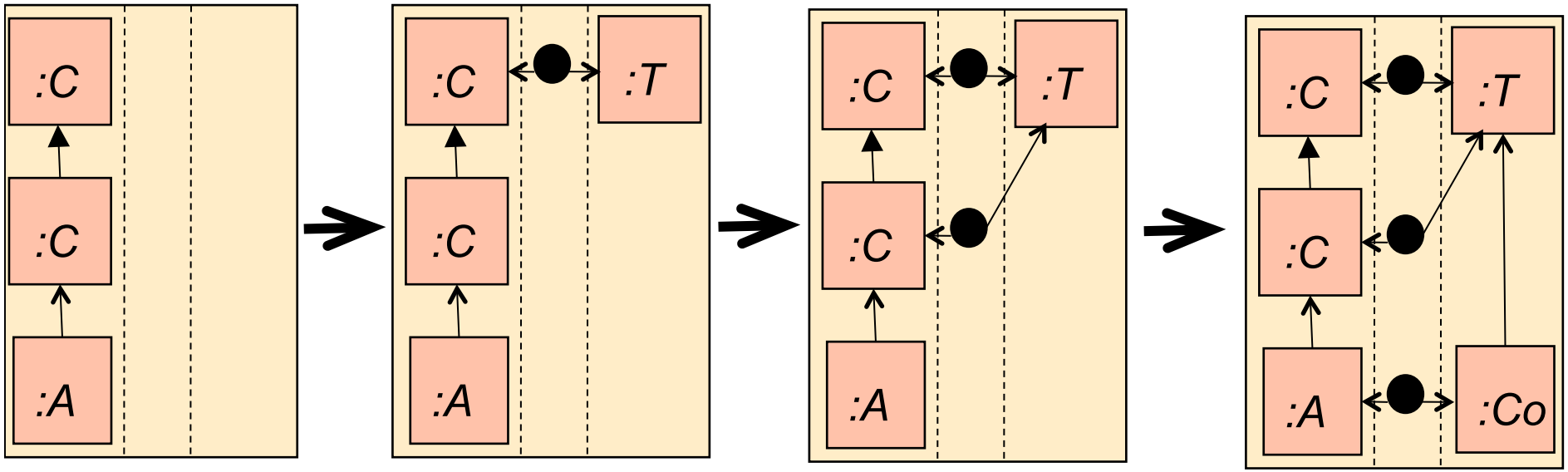
$$G \Longrightarrow_{p_1^F} G_1 \Longrightarrow_{p_2^F} \cdots \Longrightarrow_{p_n^F} G'$$

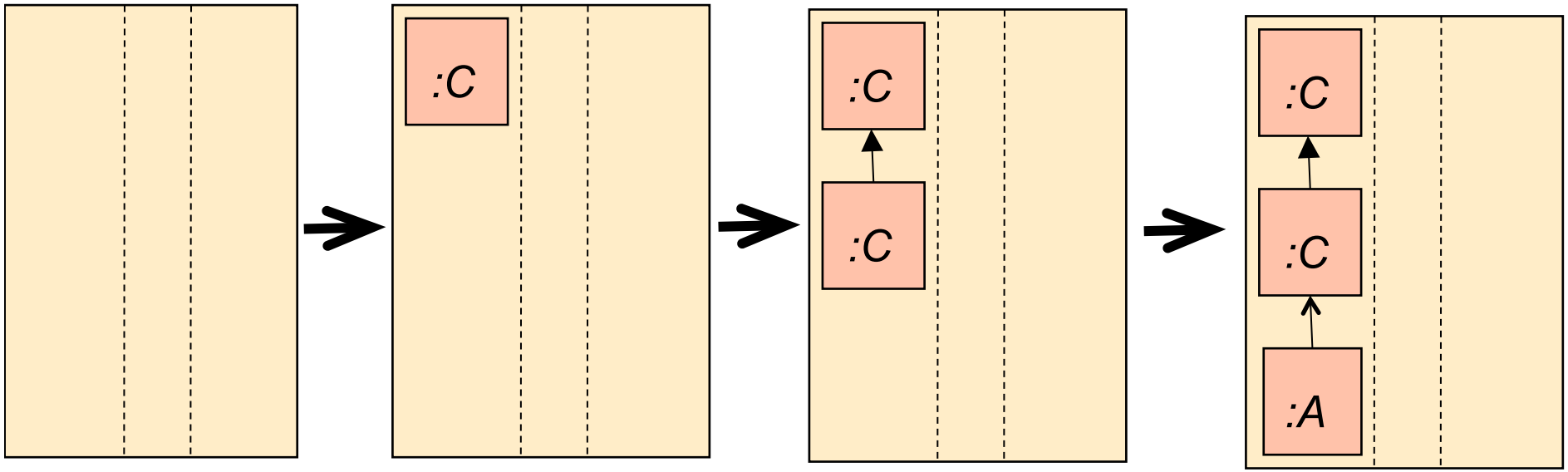
is source consistent if:

$$(\emptyset, \emptyset, \emptyset) \Longrightarrow_{p_1^S} G'_1 \Longrightarrow_{p_2^S} \cdots \Longrightarrow_{p_n^S} G$$

Theorem

Source consistent forward transformation sequences are correct and complete


:C
:C
:A
:C
:T
:C
:A
:C
:T
:C
:A
:C
:T
:C
:A
:Co


:C
:C
:C
:C
:C
:A

Problem

How to compute efficiently source consistent forward transformation sequences.

Answer

Using *forward translation rules*

Basic idea

Find the source derivation while constructing the forward derivation, by simulating the construction of the source model.

Forward Translation Rules

All elements have a boolean translation attribute, meaning if the element *has been created* or not

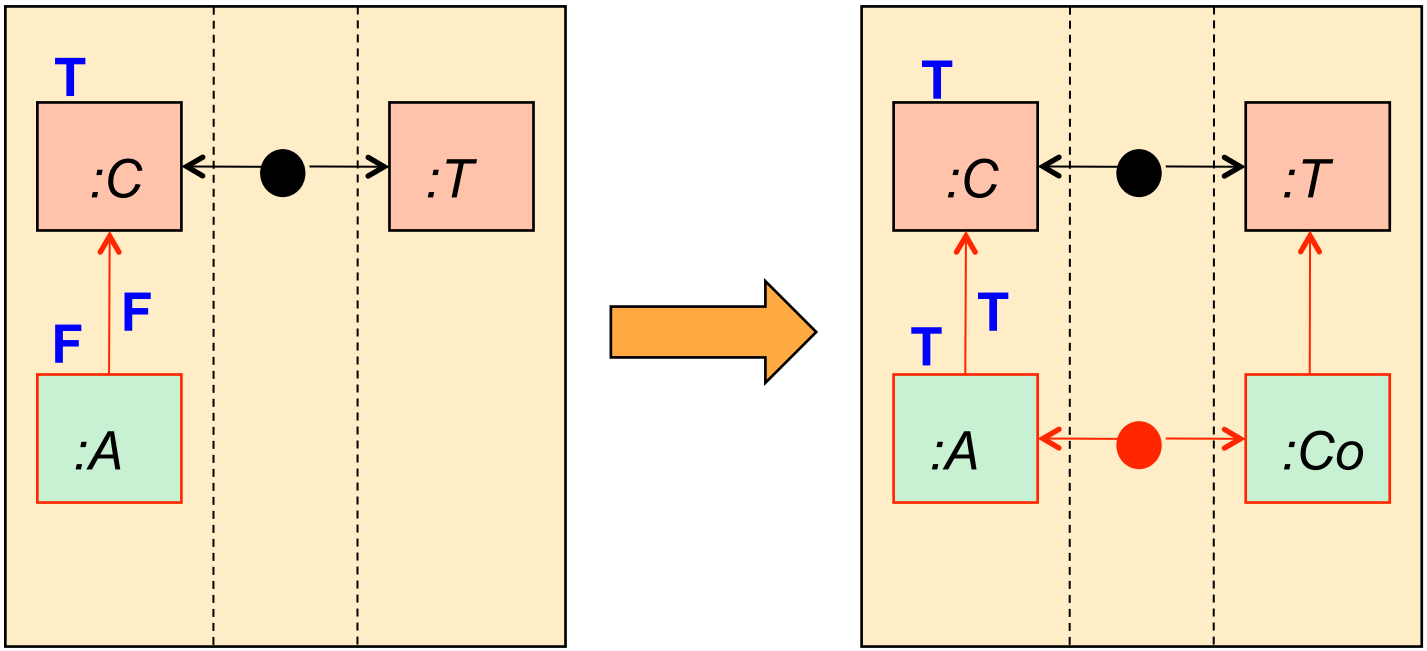
Forward translation rules are like forward rules but, on the left-hand side:

- The translation attribute of elements in $R_S \setminus L_S$ is false.
- The translation attribute of elements in L_S is true.

And, on the right hand side:

The translation attribute of all elements in R_S is True.

A Forward Translation Rule



attribute-column creation

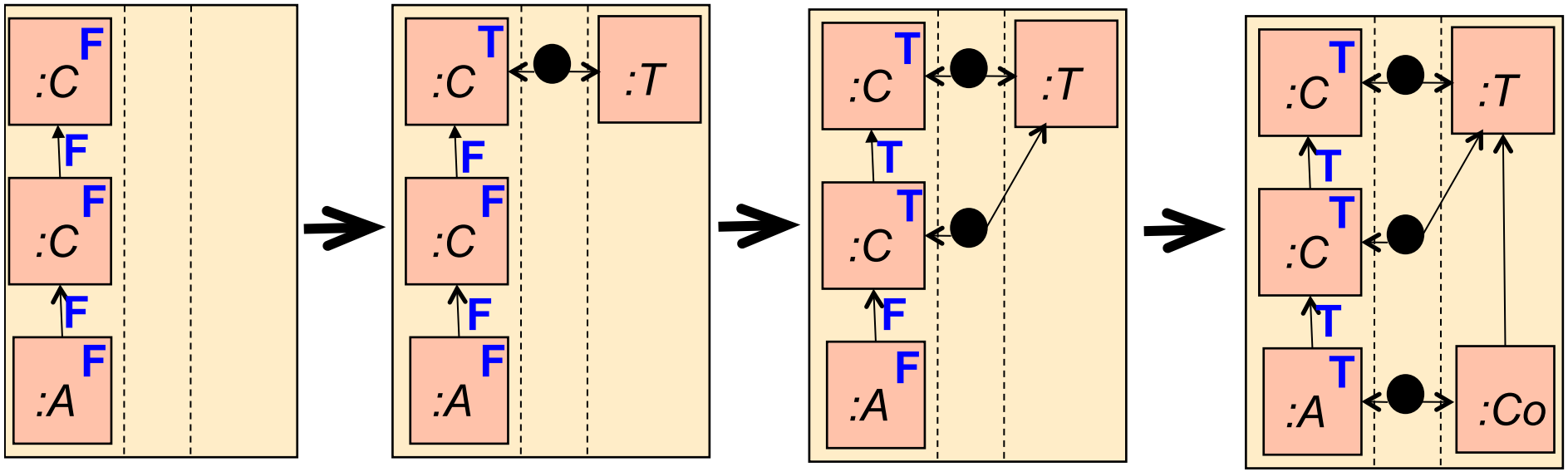
Applying Forward Translation Rules

Given a source model G_S, its associated triple graph with translation attributes is $\langle F(G_S), \emptyset, \emptyset \rangle$, where $F(G_S)$, is G_S with all the translation attributes set to False.

A forward translation sequence:

$$(F(G_S), \emptyset, \emptyset) \Longrightarrow_{p_1^{FT}} \ldots \Longrightarrow_{p_n^{FT}} (G'_S, H_C, H_T)$$

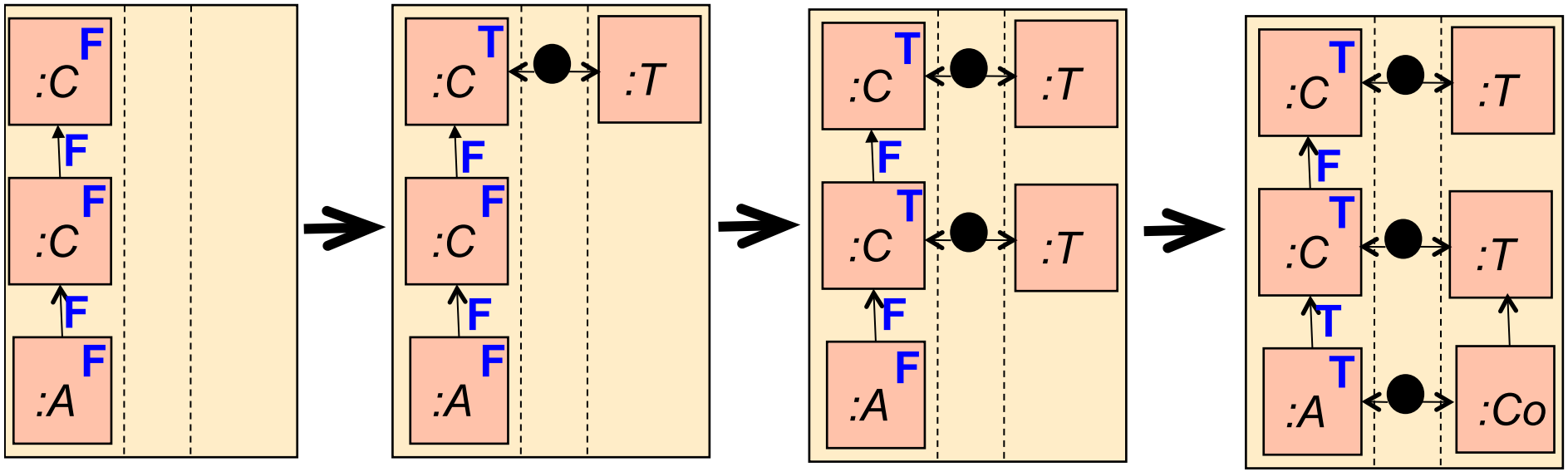
is complete if all translation attributes of G'_S are true.


:C
F
:C
F
:A
F
:C
T
:T
:A
:Co

Theorem

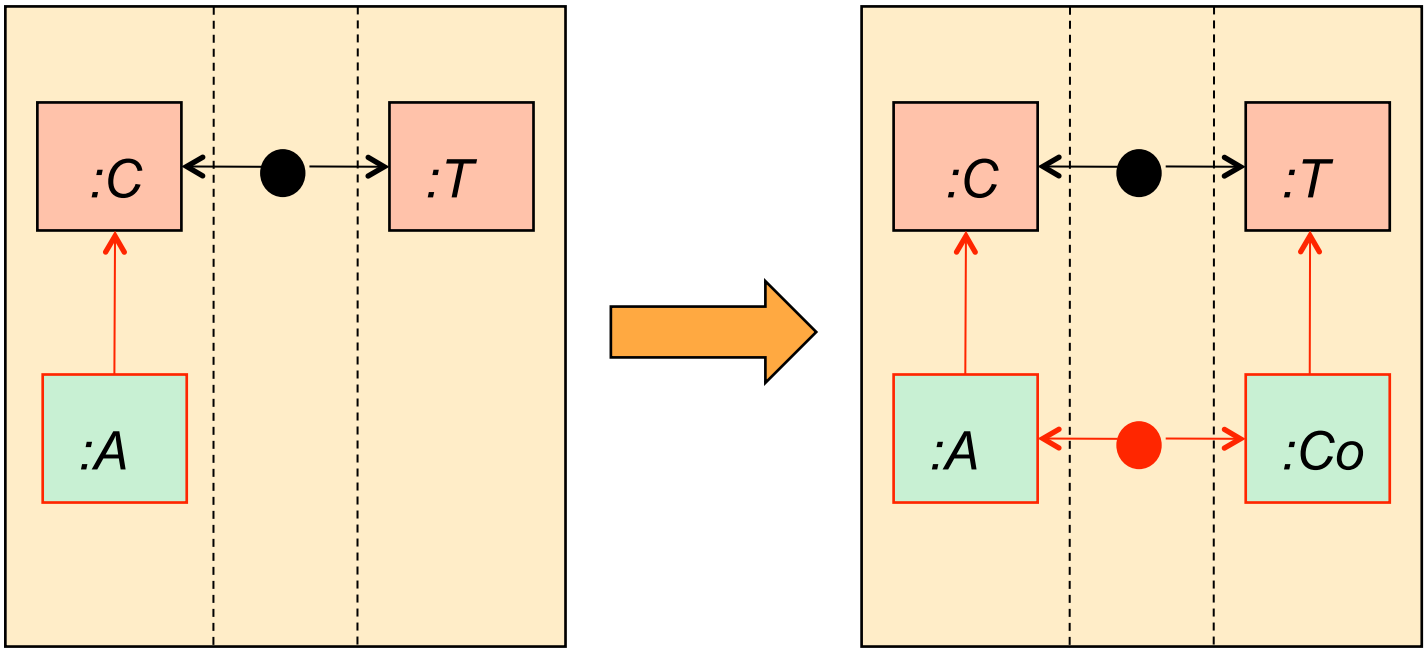
Complete translation sequences are correct and complete.

Moreover, if all rules in the given TGG create some source element, then translation sequences are finite.

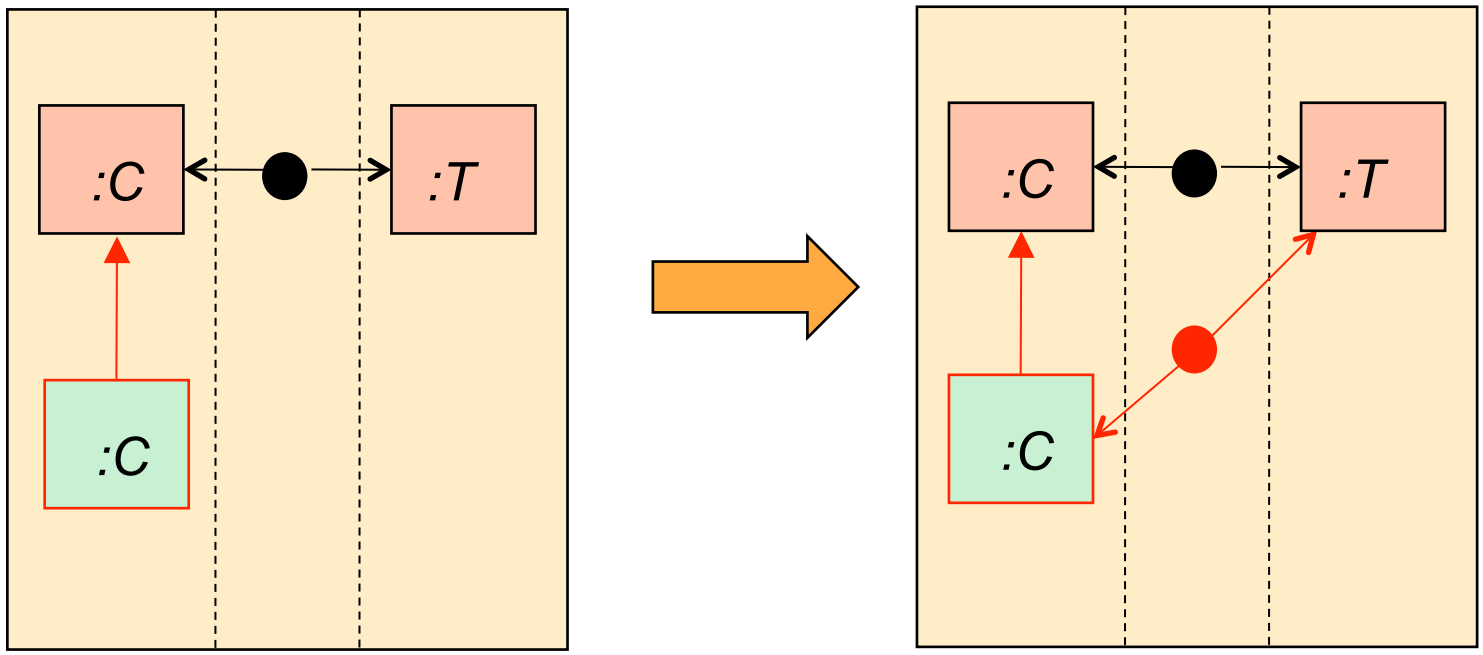

:C
F
:C
F
:A
F
:C
T
:T
:A
:Co

Conflict detection and elimination

We can detect conflicts using critical pairs.

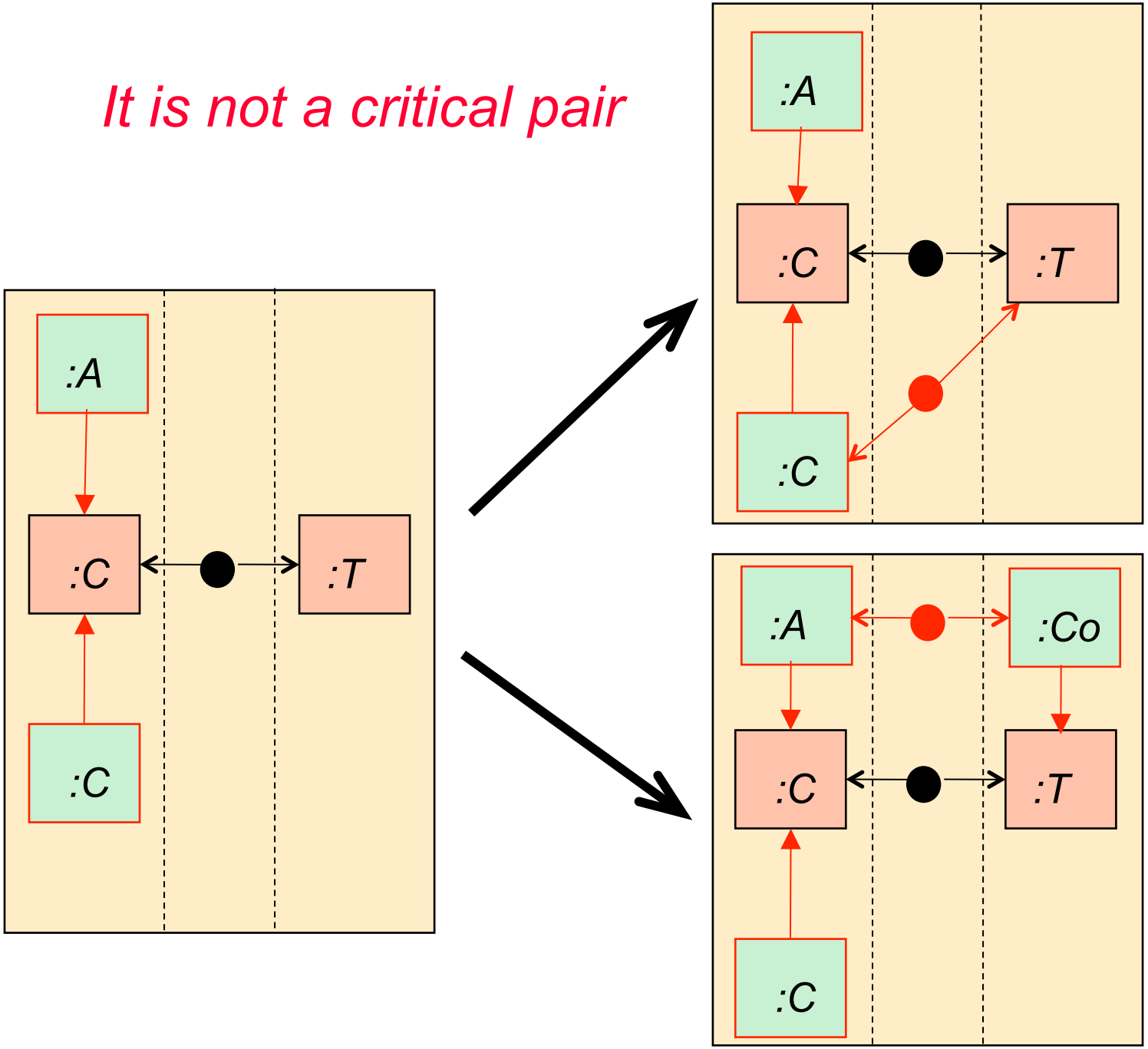


attribute-column creation

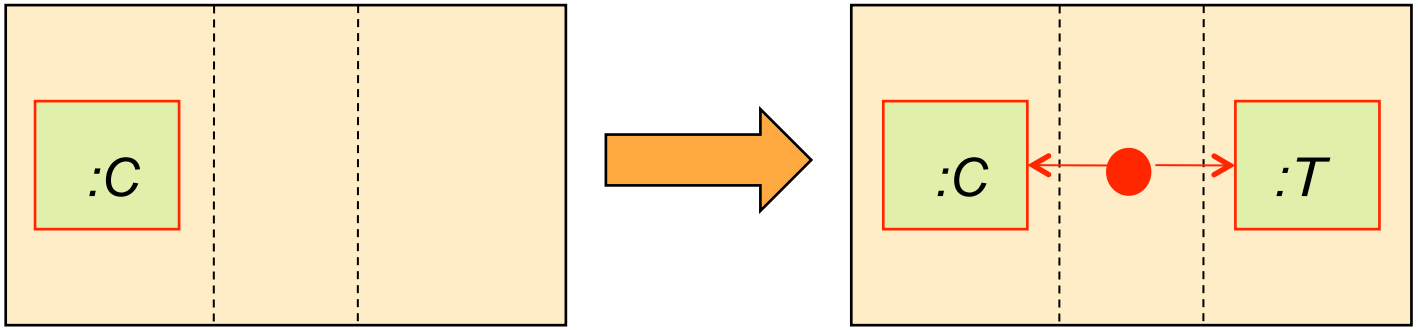


Subclass creation

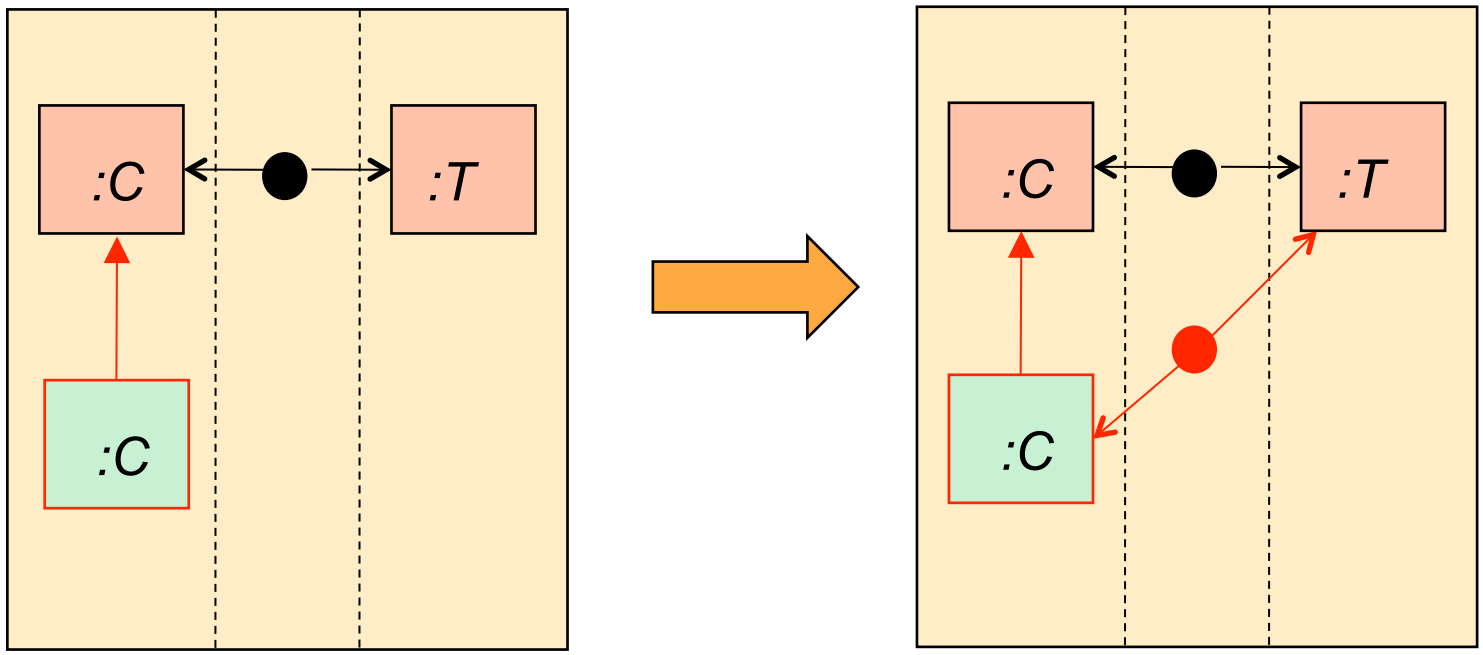
It is not a critical pair



Transformation Rules

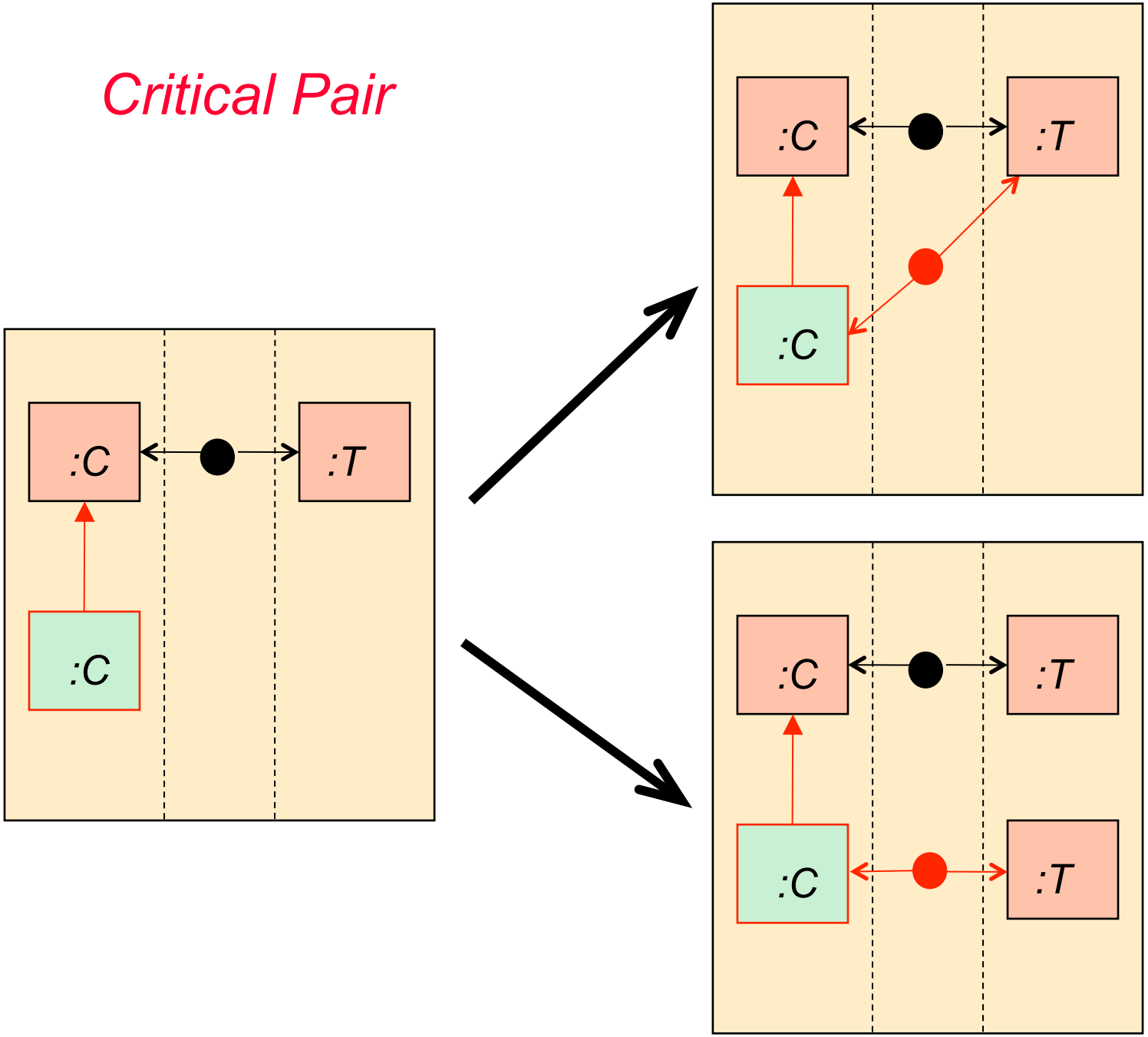


class-table creation



Subclass creation

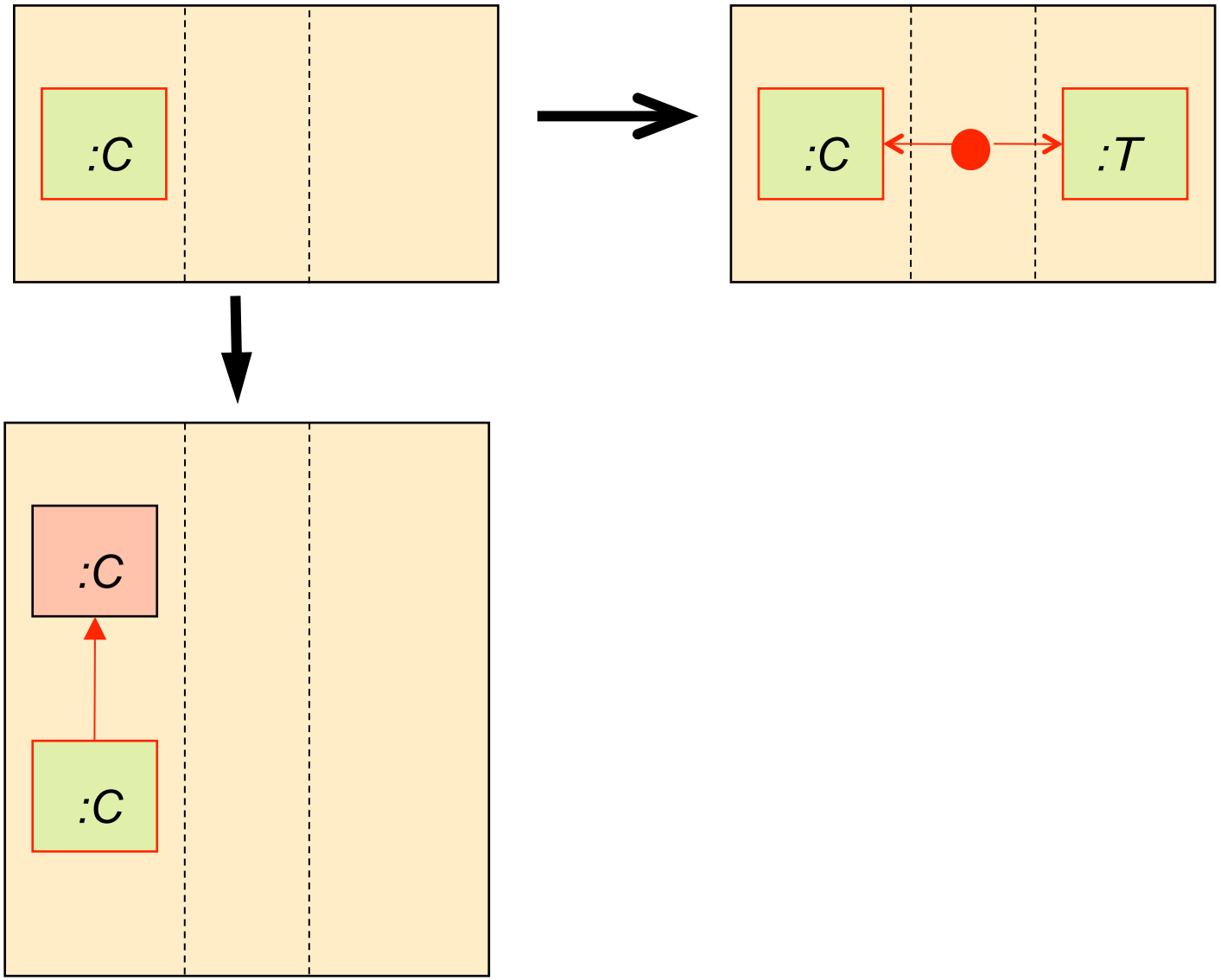
Critical Pair



Conflict detection and elimination

We can eliminate conflicts using *filter NACs*.

Filter NACs



Theorem

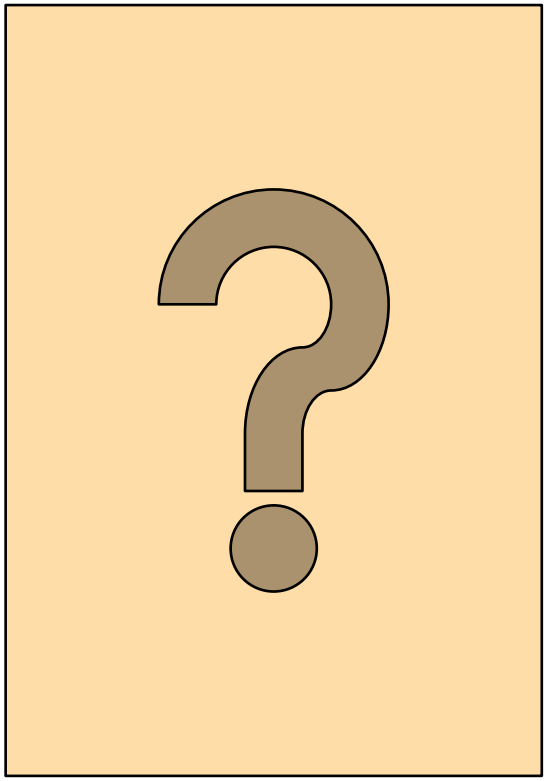
Filter NACs preserve completeness.

Model Synchronización

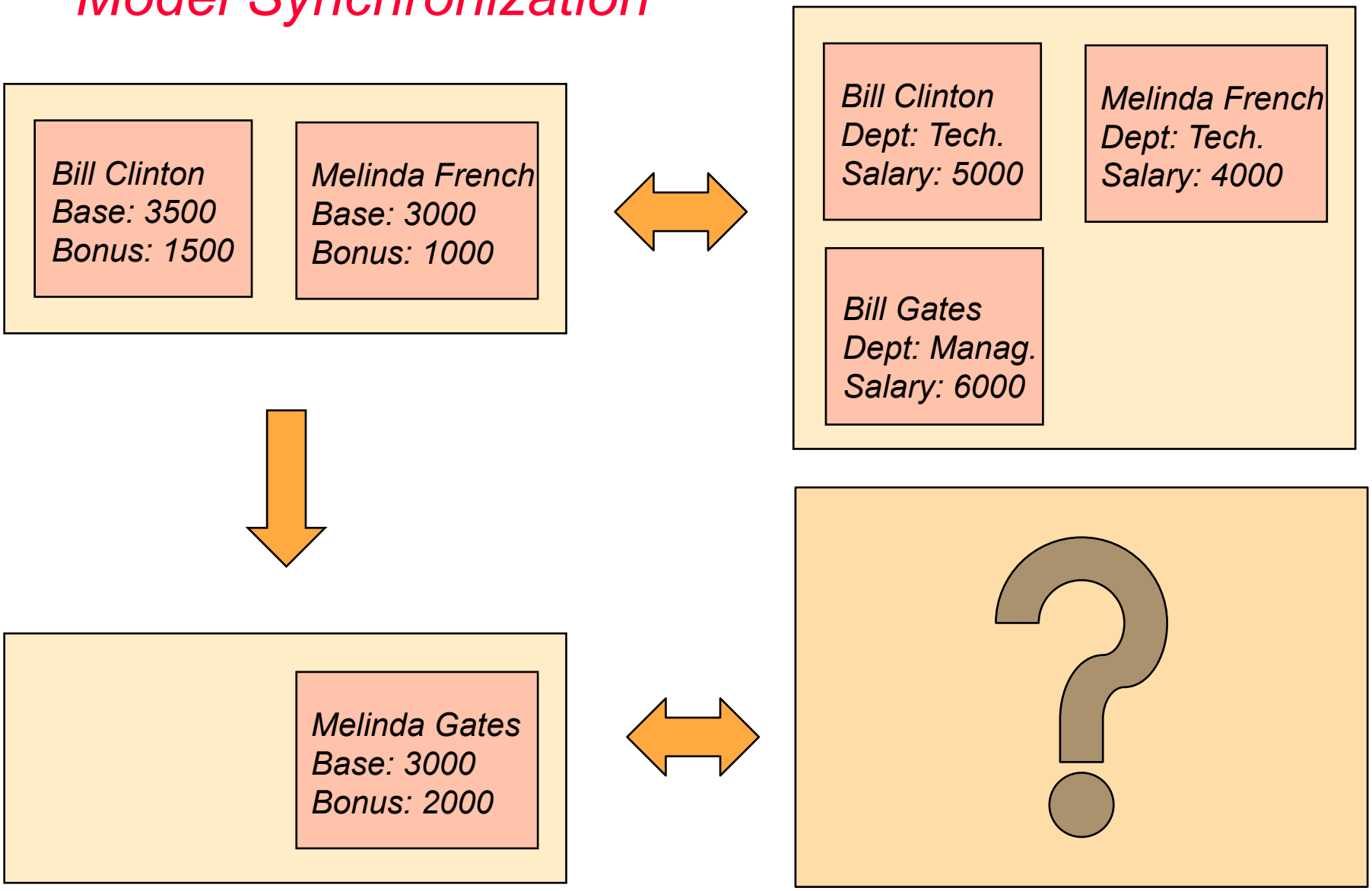
The problem

The problem

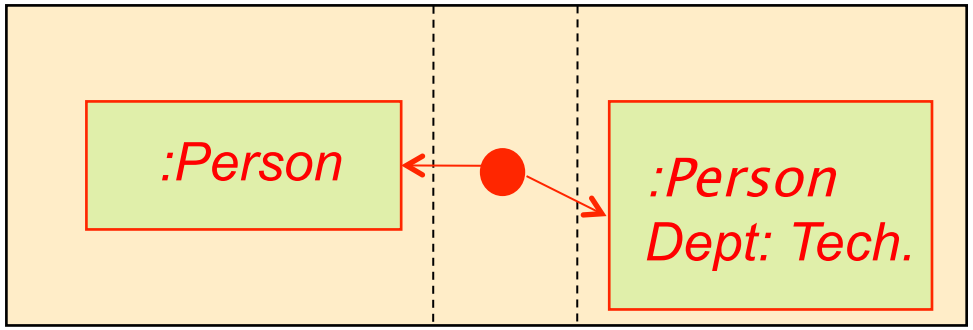
The problem

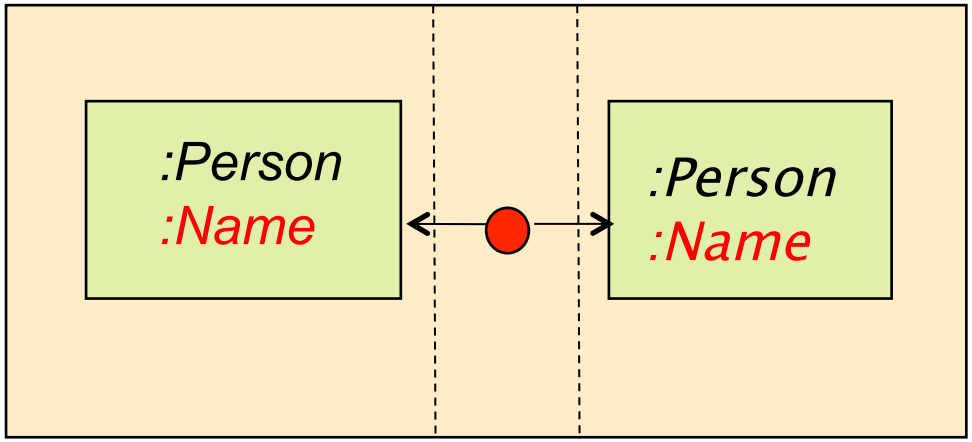


Model Synchronization

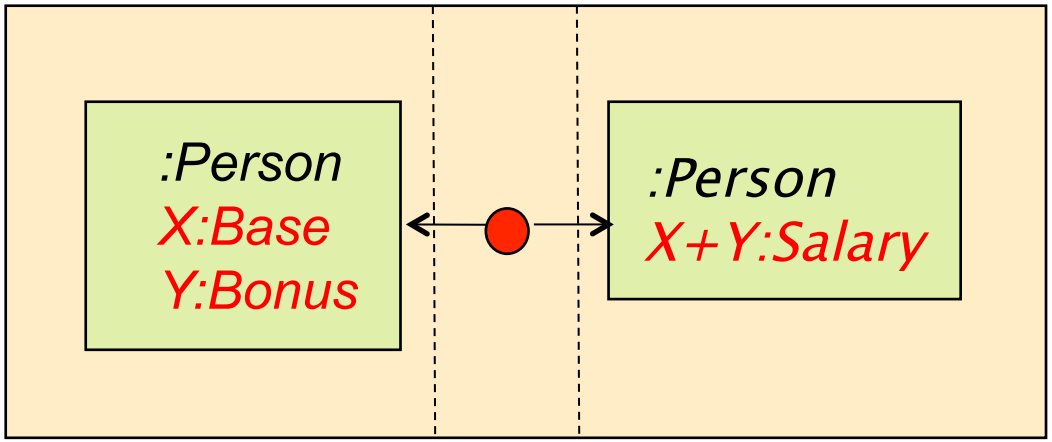


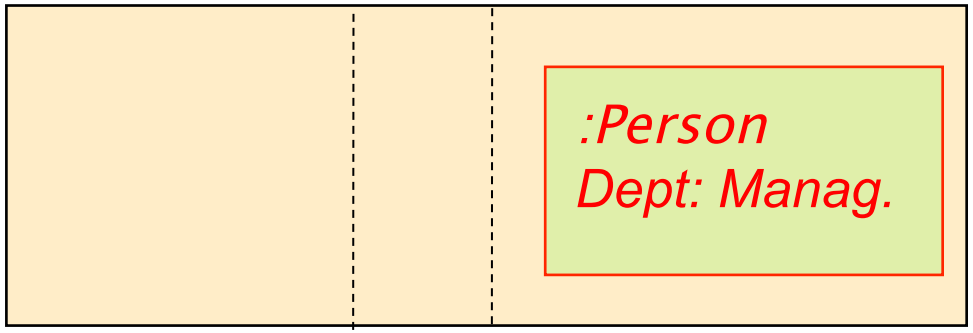
Transformation rules



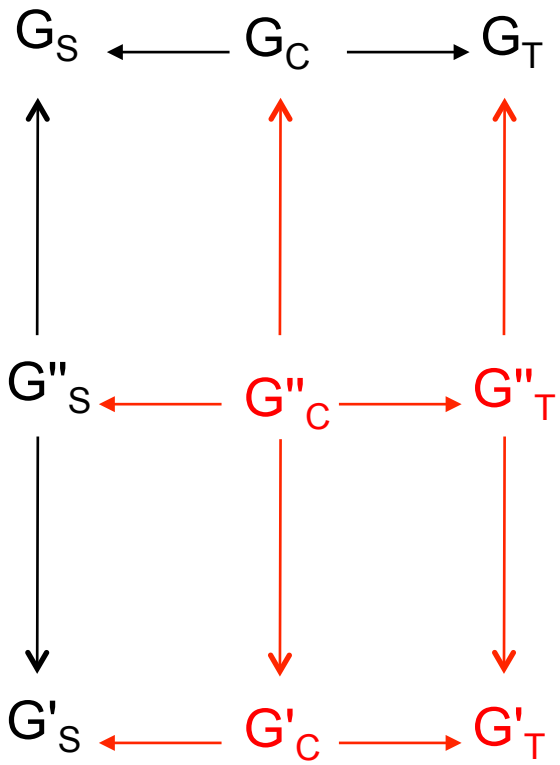


Transformation rules

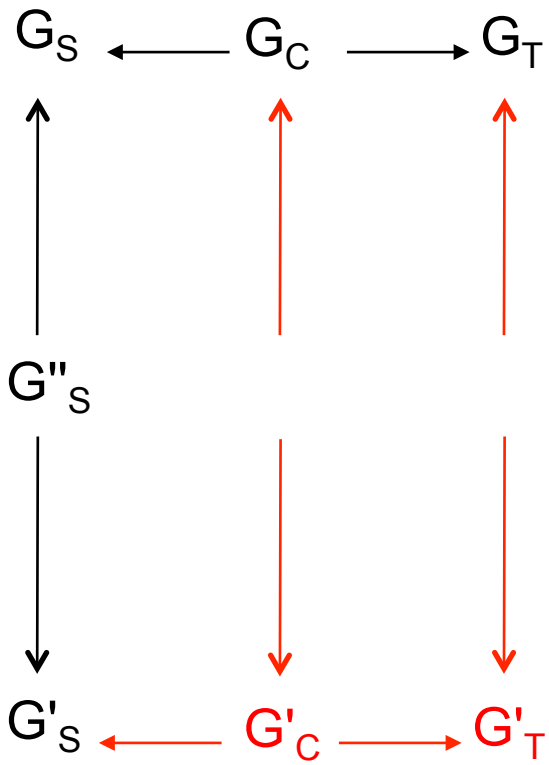




The problem

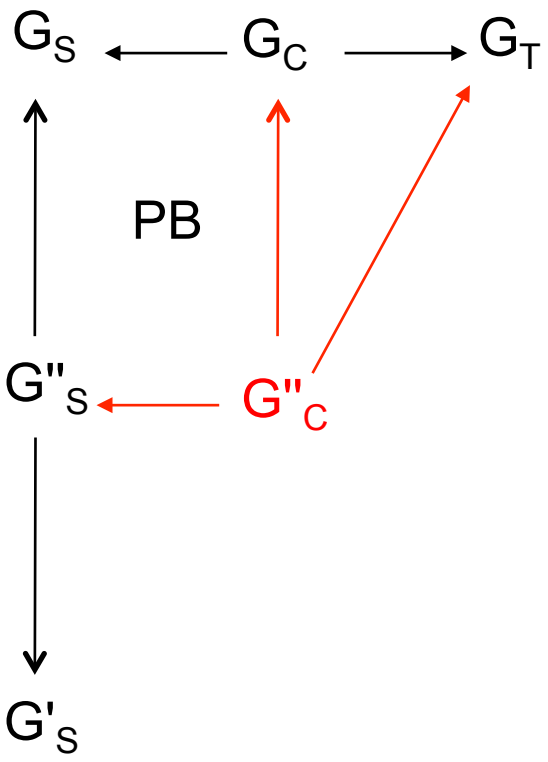


The problem

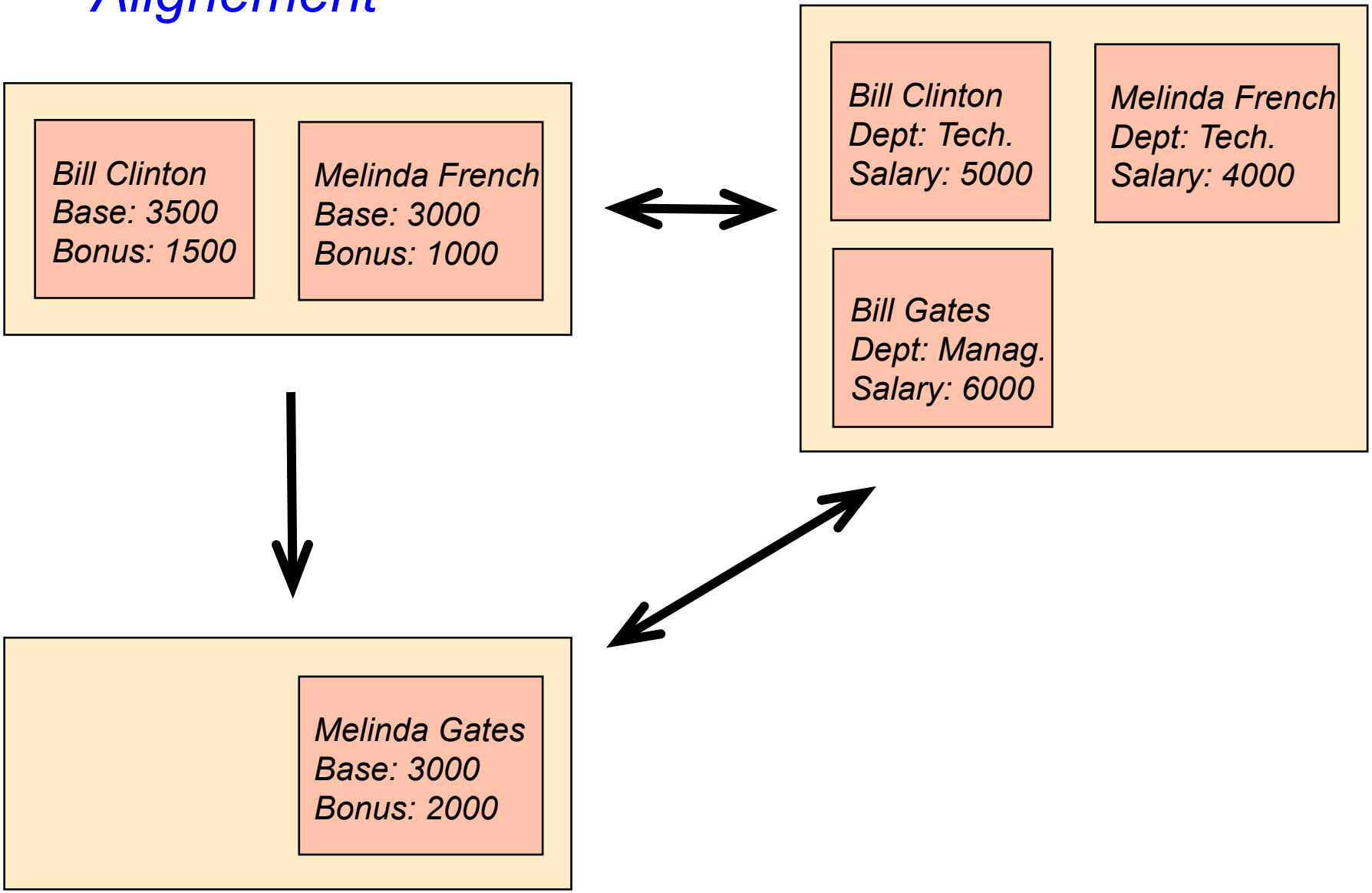


The solution

1. Alignement



Alignement

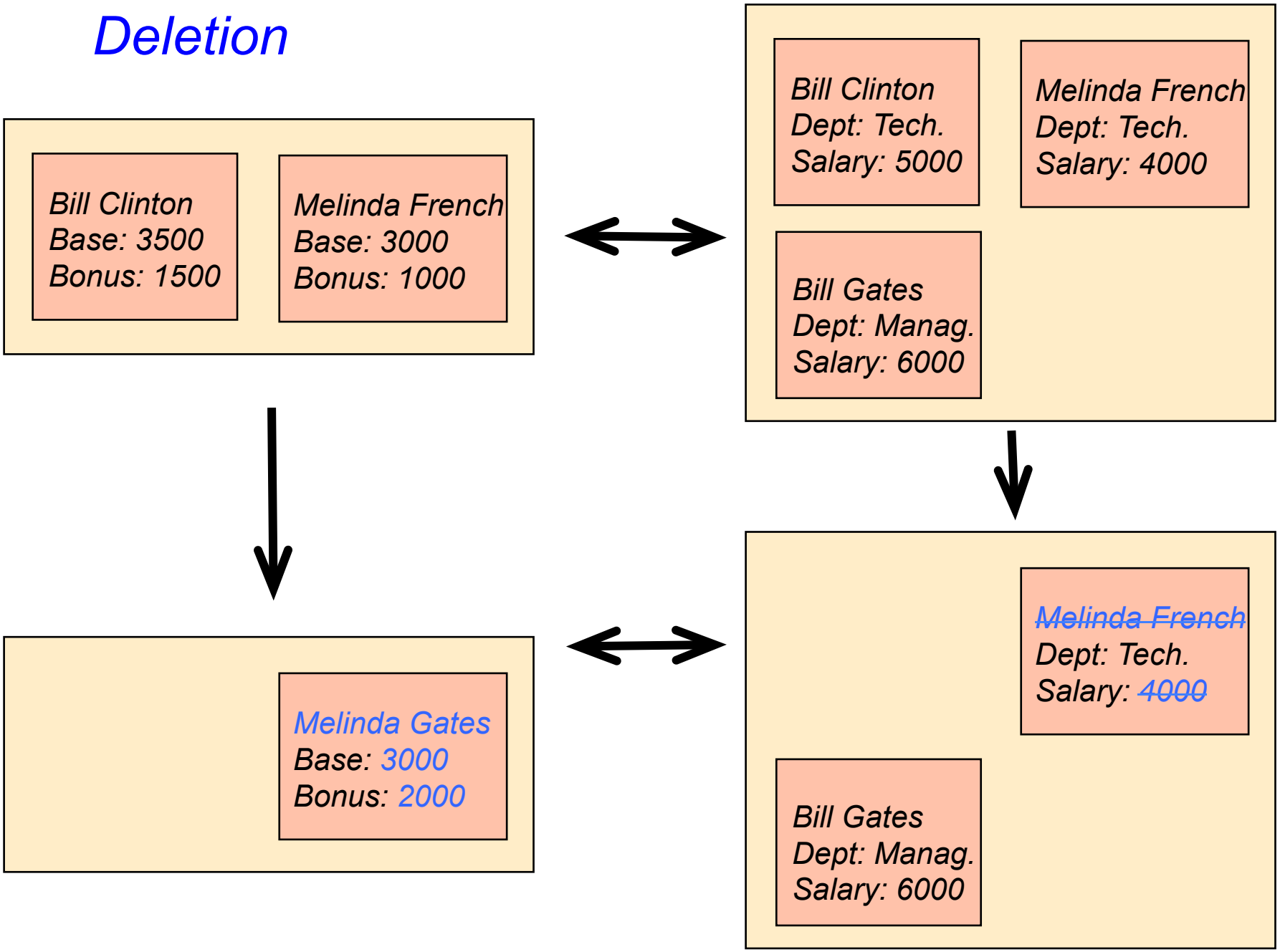

Bill Clinton
Base: 3500
Bonus: 1500
Melinda French
Base: 3000
Bonus: 1000
Bill Clinton
Dept: Tech.
Salary: 5000
Melinda French
Dept: Tech.
Salary: 4000
Bill Gates
Dept: Manag.
Salary: 6000
Melinda Gates
Base: 3000
Bonus: 2000

2. Deletion

Find maximal $G^0_S \leftarrow G^0_C \rightarrow G^0_T$ included in $G'_S \leftarrow G''_C \rightarrow G_T$

$$
\begin{tikzcd}
G_S \arrow[d] & G''_C \arrow[l] \arrow[r] \arrow[d] & G_T \arrow[d] \\
G^0_S \arrow[d] & G^0_C \arrow[l] \arrow[r] & G^0_T \\
G'_S & &
\end{tikzcd}
$$

Deletion

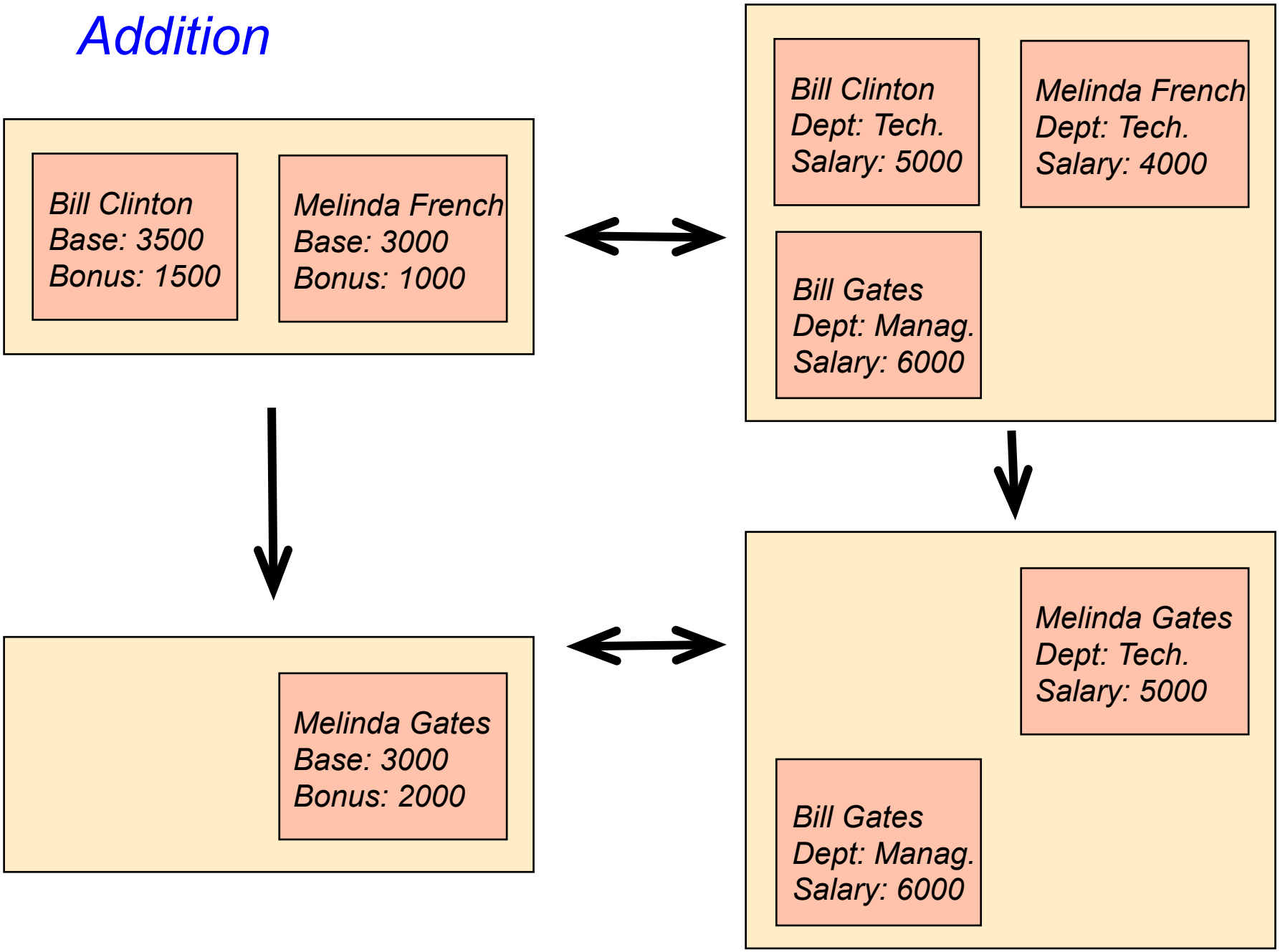

Bill Clinton
Base: 3500
Bonus: 1500
Melinda French
Base: 3000
Bonus: 1000
Bill Clinton
Dept: Tech.
Salary: 5000
Melinda French
Dept: Tech.
Salary: 4000
Bill Gates
Dept: Manag.
Salary: 6000
Melinda Gates
Base: 3000
Bonus: 2000
~~Melinda French~~
Dept: Tech.
Salary: ~~4000~~
Bill Gates
Dept: Manag.
Salary: 6000

3. Addition

Apply forward translation rules to obtain $G'_S \leftarrow G'_C \rightarrow G'_T$

$$
\begin{tikzcd}
G_S \arrow[d] & G_C \arrow[l] \arrow[r] \arrow[d] & G_T \arrow[d] \\
G''_S \arrow[d] & G^0_C \arrow[l] \arrow[r] \arrow[d] & G''_T \arrow[d] \\
G'_S & G'_C \arrow[l] \arrow[r] & G'_T
\end{tikzcd}
$$

Addition

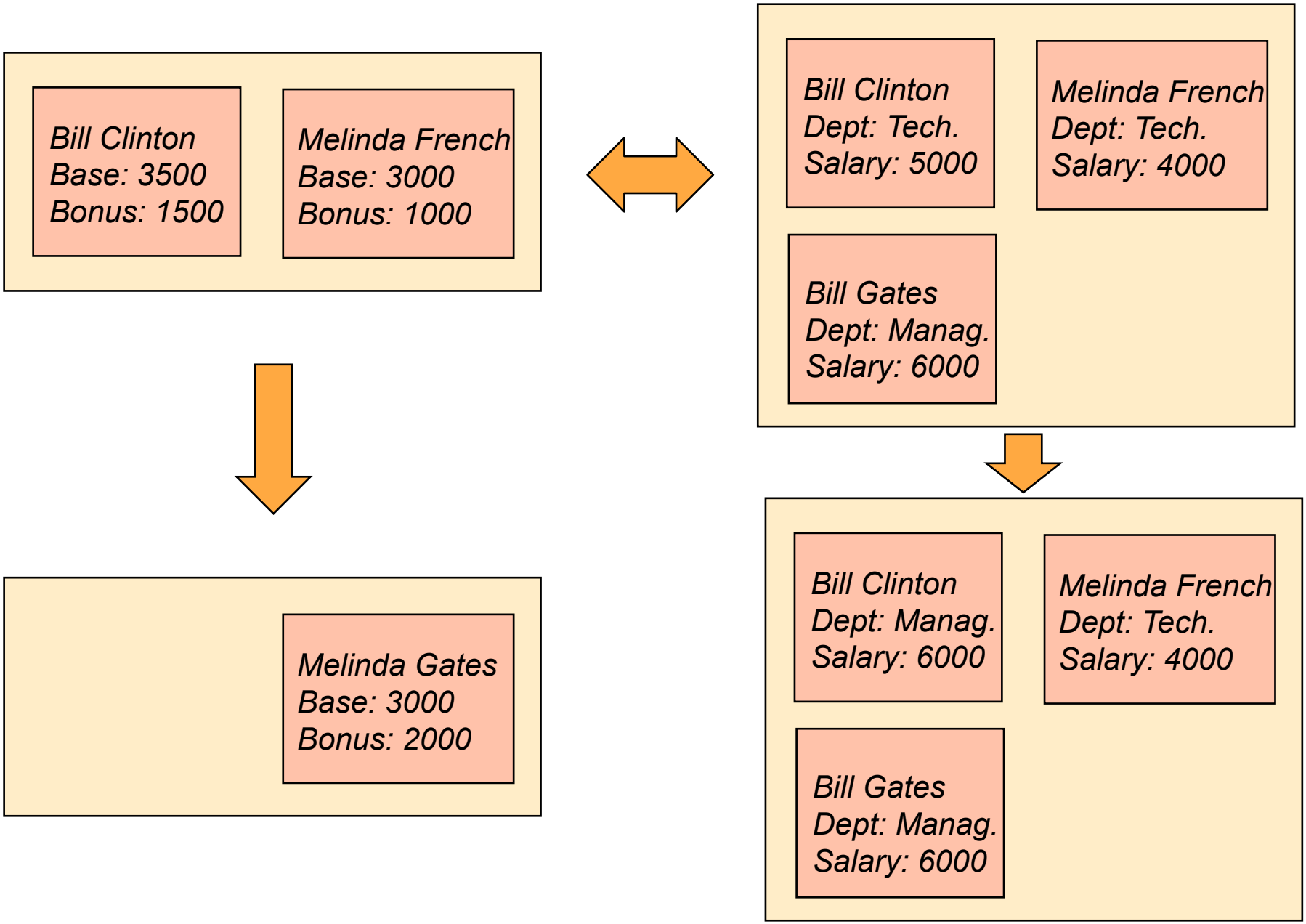

Bill Clinton
Base: 3500
Bonus: 1500
Melinda French
Base: 3000
Bonus: 1000
Bill Clinton
Dept: Tech.
Salary: 5000
Melinda French
Dept: Tech.
Salary: 4000
Bill Gates
Dept: Manag.
Salary: 6000
Melinda Gates
Base: 3000
Bonus: 2000
Melinda Gates
Dept: Tech.
Salary: 5000
Bill Gates
Dept: Manag.
Salary: 6000

Theorem

Model Synchronization is correct, complete and hipocratic

Concurrent Synchronization

Concurrent Syncronization

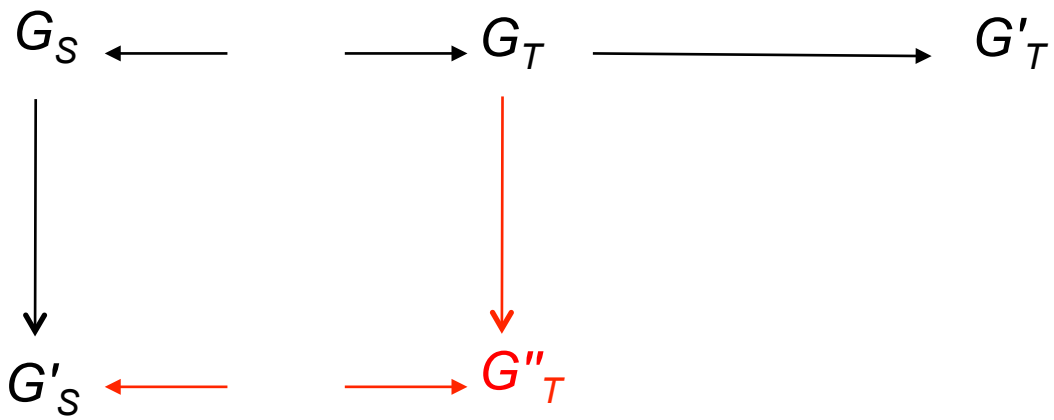


The Problem

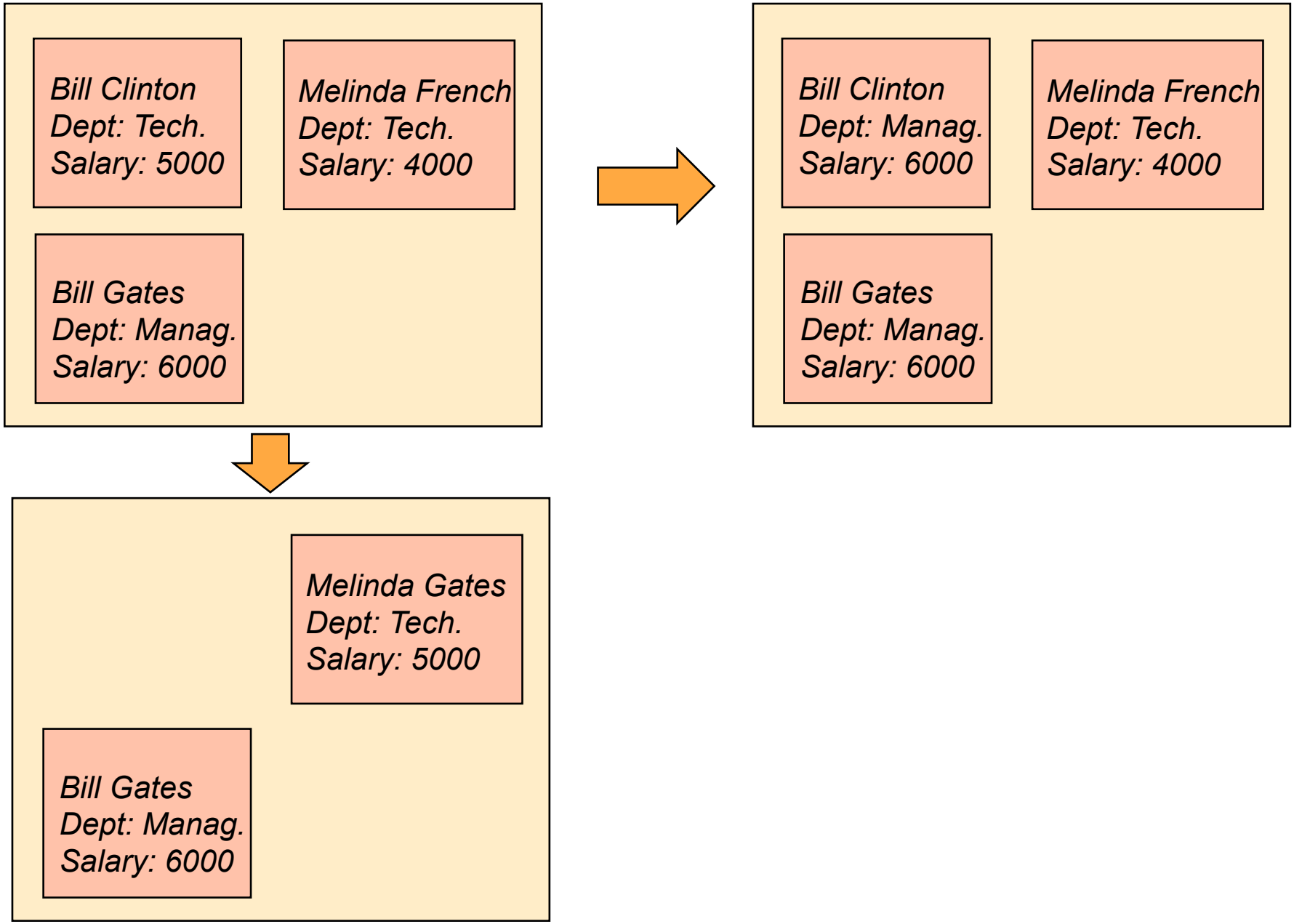
$$
\begin{tikzcd}
G_S \arrow[d] & G_T \arrow[l] \arrow[r] \arrow[d] \\
G'_S \arrow[d] & G'_T \arrow[l] \arrow[r] \arrow[d] \\
G''_S & G''_T \arrow[l] \arrow[r]
\end{tikzcd}
$$

The solution

1. Propagating source update



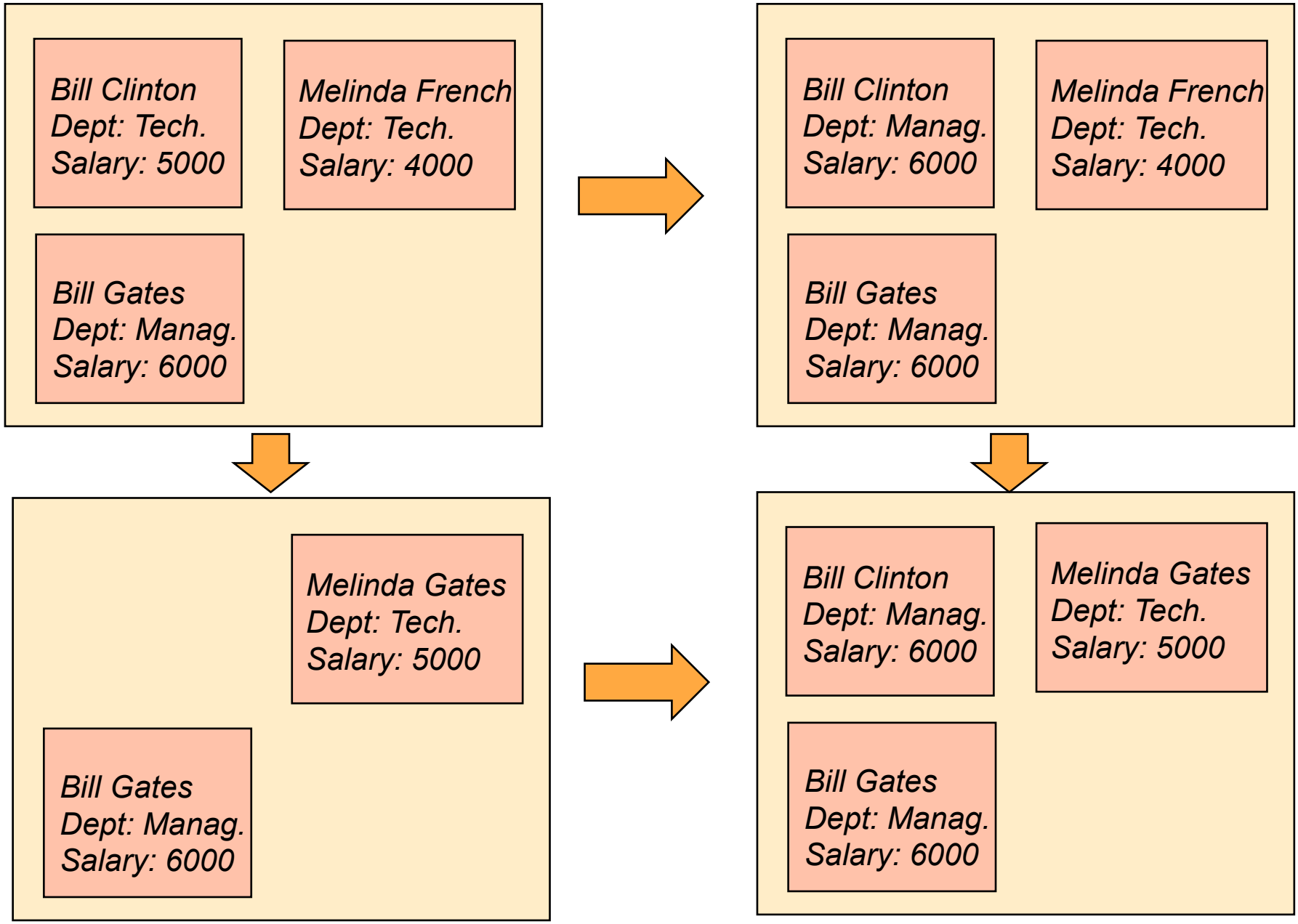
Propagating source update



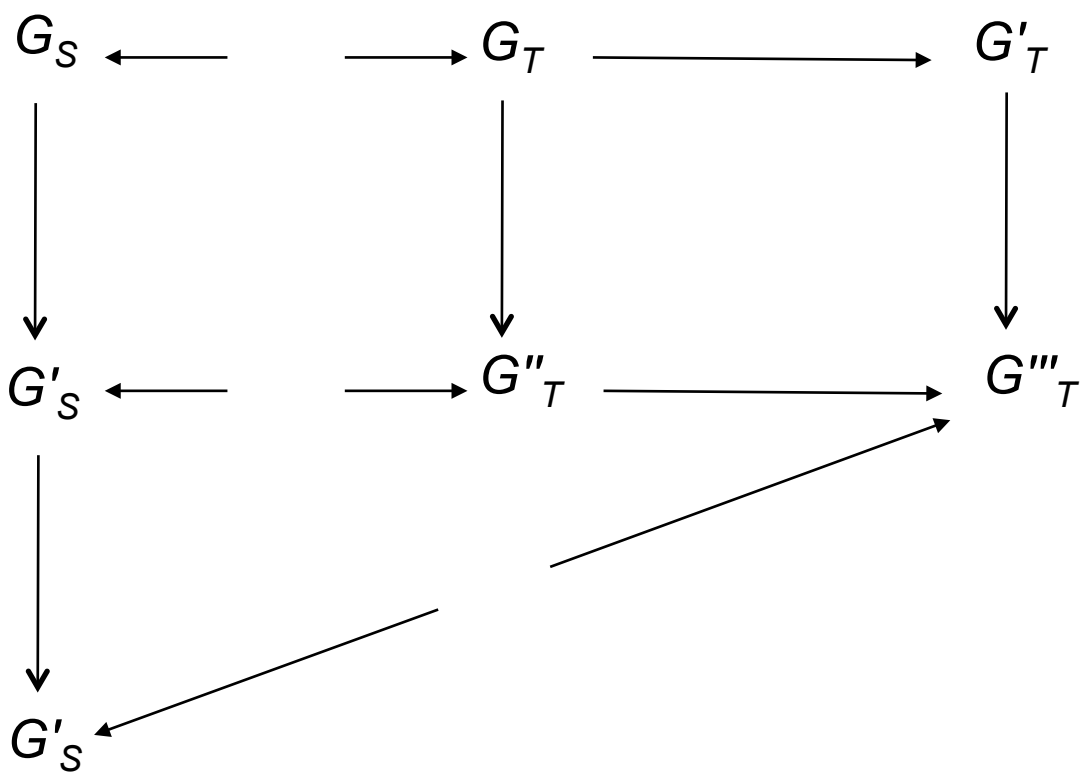
2. Conflict Resolution

$$
\begin{tikzcd}
G_S \arrow[d] & G_T \arrow[l] \arrow[r] \arrow[d] & G'_T \arrow[d] \\
G'_S & G''_T \arrow[l] \arrow[r] & G'''_T
\end{tikzcd}
$$

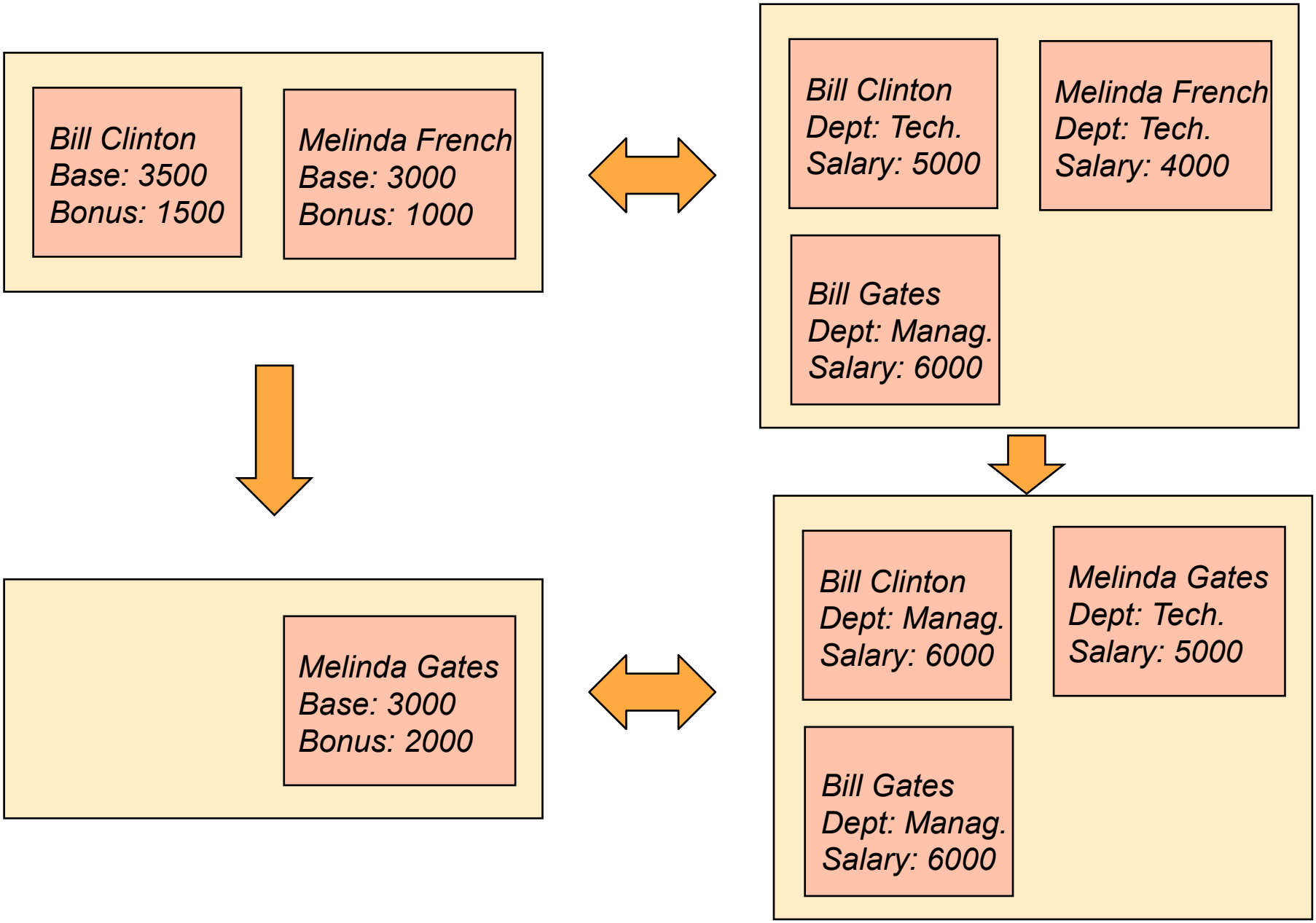
Conflict Resolution



3. Propagating target update



Propagating target update



Theorem

Concurrent synchronization is correct and complete.

Some problems

- Attributed graphs
- Correctness criteria for concurrent synchronization
- Multi-party concurrent synchronization

¡Thank You!