2nd International Workshop on Graph Transformation and Visual Modeling Techniques

Application of Attribute NCE Graph Grammars to Syntactic Editing of Tabular Forms

Tomokazu ARITA (Nihon Univ.),
Kiyonobu TOMIYAMA (Nihon Univ.),
Kensei TSUCHIDA (Toyo Univ.),
Takeo YAKU (Nihon Univ.)

Terms

- **Tabular forms** with **syntax**
- **Graph grammars** for tabular form syntax
- **Attribute** : used for drawing
- **NCE** : Neighborhood Controlled Embedding

  (cf. NLC : Node Labeled Controlled) :

  A type of embedding mechanisms of right hand side graph into host graph for **graph rewriting**
- **Syntactic editing** : editing defined by sequence of rewriting rules

Types of Visual Data

(e.g. Power Point)

- Text
- Organization Charts
- **Table(Our Target)**
- Charts, Graphs
- Images

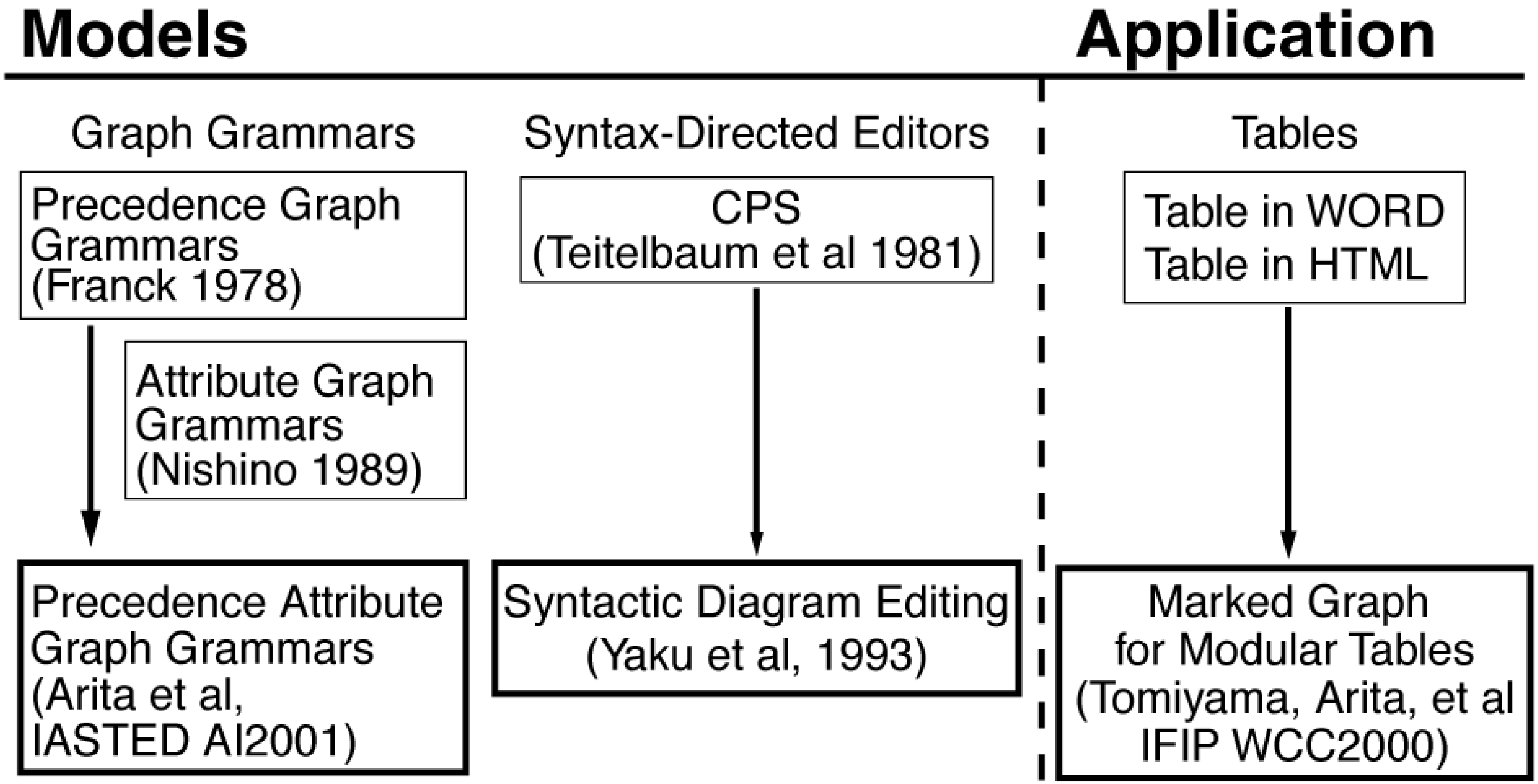
Why Graph Grammars ?

- Tabular Forms have syntax
- To determine the scope of rewriting by cell insertion, deletion etc.
- Without graph grammars, often to collapse whole structure of tables

1. Introduction

- Background
- Related Works
- Motivation
- Purpose
- Results

Background


Models
Application
Graph Grammars
Syntax-Directed Editors
Tables
Precedence Graph Grammars (Franck 1978)
CPS (Teitelbaum et al 1981)
Table in WORD
Table in HTML
Attribute Graph Grammars (Nishino 1989)
Precedence Attribute Graph Grammars (Arita et al, IASTED AI2001)
Syntactic Diagram Editing (Yaku et al, 1993)
Marked Graph for Modular Tables (Tomiyama, Arita, et al IFIP WCC2000)

Related Works

- Related works for **syntactic editing methods are CPS, DIAGEN(Minas et al.)** and so on.

Motivation

- To investigate whether graph grammars can effectively formalize table editing.

Purpose

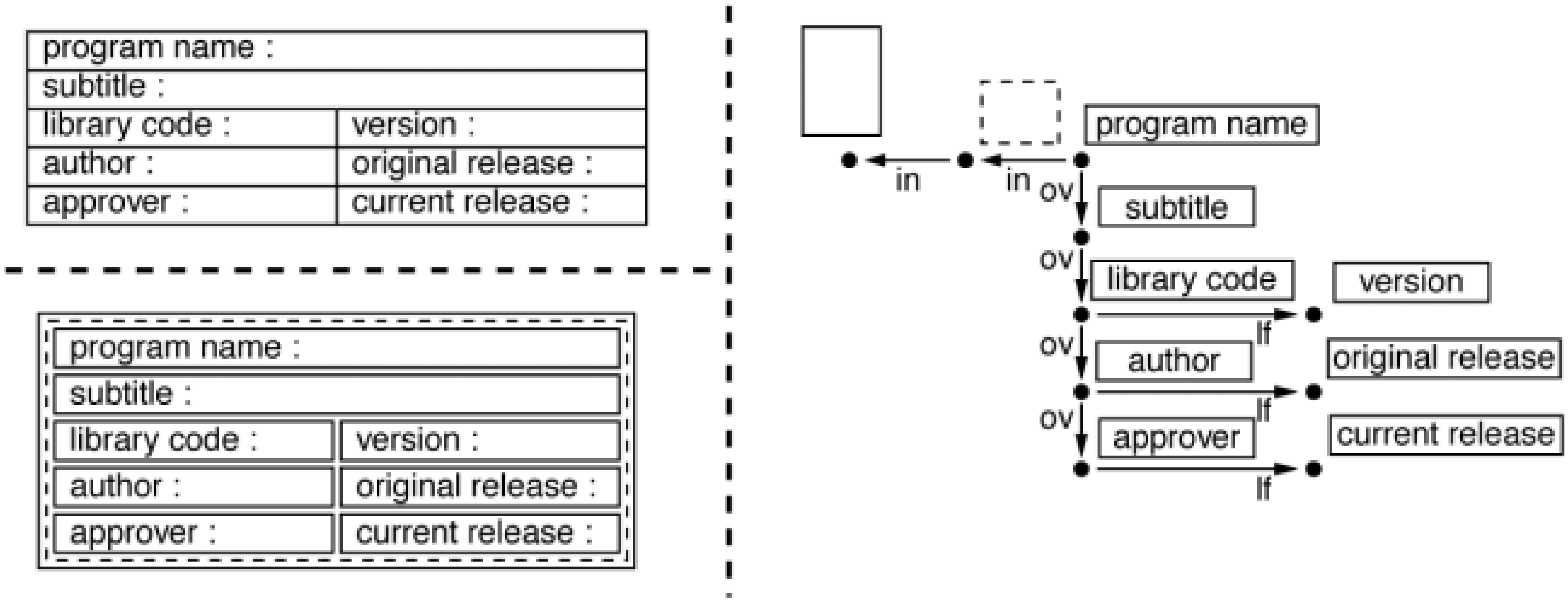
- To formalize editing by graph grammars
- To investigate validity of editing model
- To construct algorithms of editing by graph grammars

Results

- Definitions of insertion by rewriting rules of edNCE graph grammars [Definition 3.1, 3.2, 3.3, 3.4]
- Editing order doesn't influence the editing results [Proposition 3.5]
- Linear time algorithm for editing [Proposition 3.6]

2. Preliminaries

Modular Tabular Form and Its Corresponding Marked Graph



Tabular Form for Hiform

A documentation Language for Program Specification

- 17 types of forms based on ISO6592
- A collection of tabular forms

Project Code:	**A 5**
Program Name:	**Program Specification-1** p
Library Code:	Version:
Author:	Original Release:
Approver:	Current Release:
Problem Description:	
Problem Supplementary Information (Theoretical Principles, Methods and References):	
Problem Solution: 1.Conventions and Terminology 2.Principles and Algorithms	

REVIEW

edNCE Graph Grammar

[1987 Rozenberg et al]

Definition 2.1

An edNCE graph grammar : $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$,

where

Σ : the alphabet of node labels,

$\Delta \subseteq \Sigma$: the alphabet of terminal node labels,

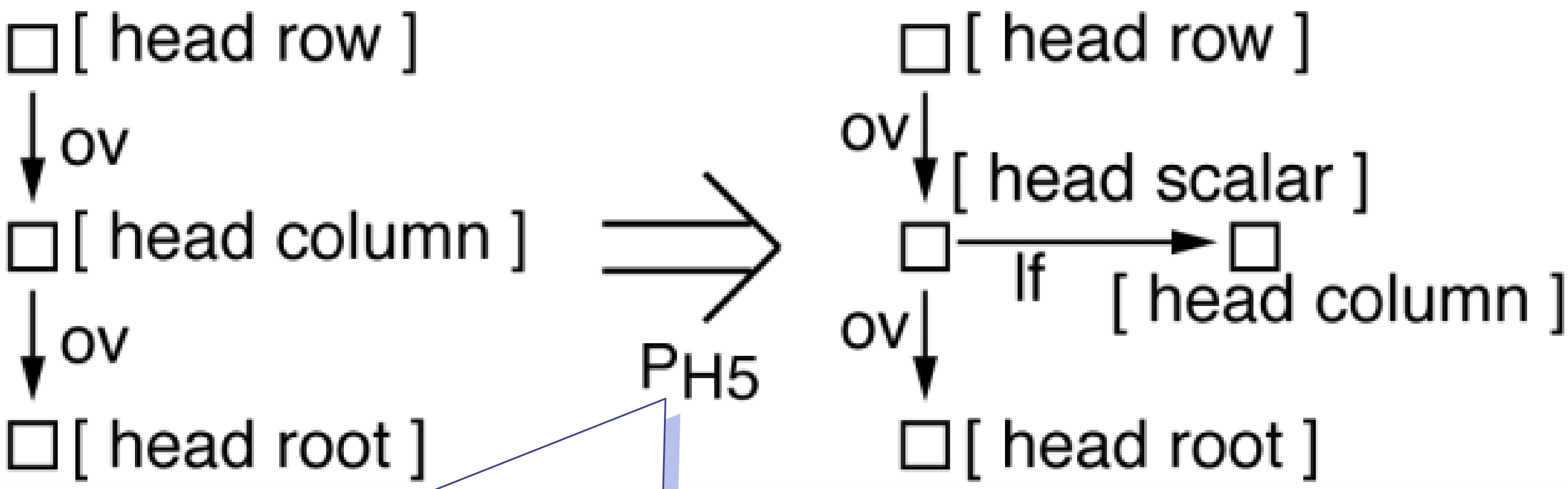
Γ : the alphabet of edge labels,

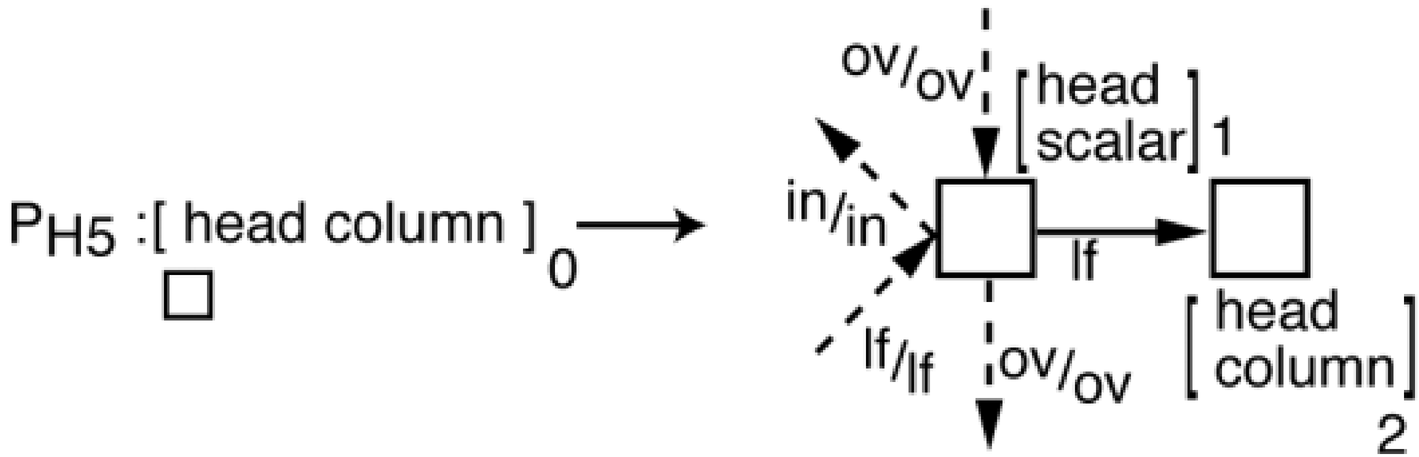
$\Omega \subseteq \Gamma$: the alphabet of final edge labels,

P : the finite set of productions,

$S \in \Sigma - \Delta$: the initial nonterminal.

Rewriting a graph by production





REVIEW

Composite Production Copy

Definition 2.2

[Adachi, Tsuchida, Yaku et al]

Let $G = (\ \Sigma,\ \Delta,\ \Gamma,\ \Omega,\ P,\ S\ )$ be an edNCE graph grammar.
Let $p_1 : X_1 \rightarrow (D_1, C_1)\ (D_1 = (V_{D_1}, E_{D_1}, \lambda_{D_1}))$
and $p_2 : X_2 \rightarrow (\ D_2, C_2\ )\ (D_2 = (V_{D_2}, E_{D_2}, \lambda_{D_2}))$
be production copies of G.
If $u \in V_{D_1}$ and $X_2 = \lambda_{D_1}(u)$, and D_1 and D_2 are disjoint,
then a composite production copy $p : X_1 \rightarrow (D, C)$
is defined as follows:
D is a graph as $V_D = \{V_{D_1}\text{-}\{u\}\} \cup V_{D_2}$ about nodes.
$C = \{\ (\sigma,\ \beta\ /\ \gamma,\ \omega,\ d) \in C_1 \mid \omega \in V_{D_1} - \{u\}\ \}$
$\cup \{\ (\sigma,\ \beta\ /\ \delta,\ y,\ d) \mid \exists\ \gamma \in \Gamma,\ (\sigma,\ \beta\ /\ \gamma,\ u,\ d\ ) \in C_1,$
$(\sigma,\ \gamma\ /\ \delta,\ y,\ d) \in C_2\ \}$
The composite production copy p composed by p_1 and p_2,
and denoted by $p_1 \circ p_2$.

$\square$

REVIEW

Confluence Property

Definition 2.3 [Rozenberg, 1997]

An edNCE graph grammar G=(Σ, Δ, Γ, Ω, P, S)
is <u>dynamically confluent</u>
if the following holds for every intermediate graph H generated by G:
if $H \Rightarrow_{u_1,p_1} H_1 \Rightarrow_{u_2,p_2} H_{12}$ and $H \Rightarrow_{u_2,p_2} H_2 \Rightarrow_{u_1,p_1} H_{21}$
$(p_1, p_2 \in P)$ are derivations of G with $u_1, u_2 \in V_H$ and $u_1 \neq u_2$,
then $H_{12} = H_{21}$. $\square$

$$\begin{tikzcd}
 & H_1 \arrow[dr, Rightarrow, "p_2"] & \\
H \arrow[ur, Rightarrow, "p_1"] \arrow[dr, Rightarrow, "p_2"'] & & H_{12} \arrow[d, equal] \\
 & H_2 \arrow[r, Rightarrow, "p_1"'] & H_{21}
\end{tikzcd}$$

REVIEW

Attribute NCE Graph Grammar

Definition2.4 [Arita et al, IASTED 2001]

An attribute NCE graph grammar :

$AGG = \langle G, Att, F \rangle$ where

1. $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$:

   an underlying graph grammar of AGG.

2. $Att = \bigcup_{Y \in \Sigma} Att(Y)$,

   $(Att(Y) = Inh(Y) \cup Syn(Y).)$

3. $F = \bigcup_{p \in P} F_p$:

   the set of semantic rules of AGG.

REVIEW

HNGG [Arita et al, IASTED2001]

Hiform Nested tabular form Graph Grammar :

$HNGG = (G_N, Att_N, F_N)$,

where

$G_N = (\Sigma_N, \Delta_N, \Gamma_N, \Omega_N, P_N, S_N)$ s.t.

Σ_N : node labels,

$\Delta_N \subseteq \Sigma$: for items of program specifcations,

$\Gamma_N = \{in, ov, lf\}$: for relations between items,

$\Omega_N = \Gamma_N$

P_N : the finite set of productions,

$S_N = [struct]$

$Att_N = \{x, y, width, height\}$

F_N : used for drawing tabular forms.

3. Editing of Modular Tabular Forms

Production Instance

Definition 3.1

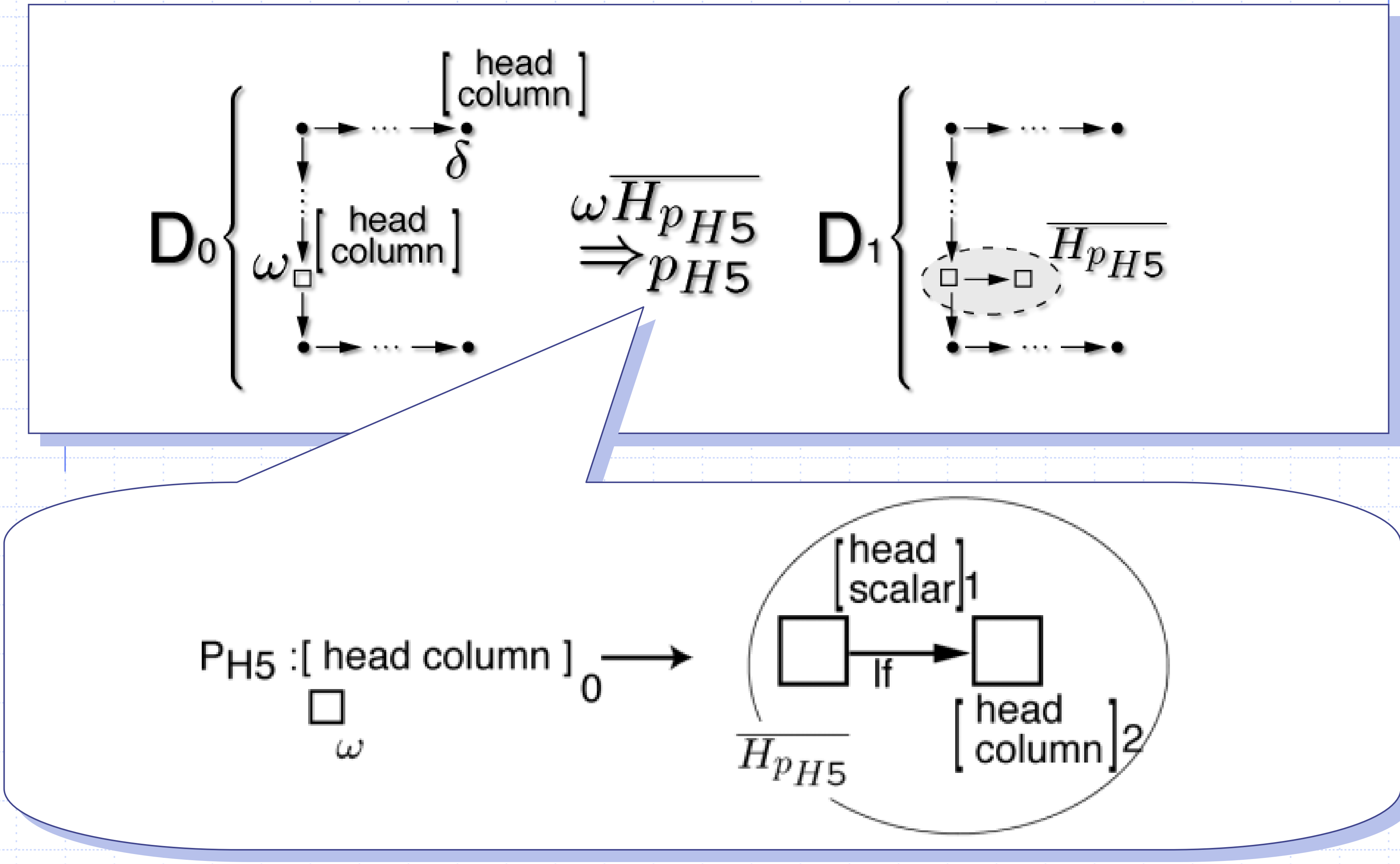
A <u>production instance</u> : $(\omega, p_i, \overline{H_{p_i}})$, where

1. $\omega \in V_{D_{i-1}}$: a node removed during the derivation $D_{i-1} \Rightarrow_{p_i} D_i$.
2. $p_i : X_{p_i} \to (H_{p_i}, C_{p_i}) \in P$
3. $\overline{H_{p_i}}$: an particular graph isomorphic to H_{p_i} during $D_{i-1} \Rightarrow_{p_i} D_i$.

We denote $D_{i-1} \overset{\omega \overline{H_{p_i}}}{\underset{p_i}{\Longrightarrow}} D_i$

if D_{i-1} is directly derived D_i by $(\omega, p_i, \overline{H_{p_i}})$.

A derivation by a production instance $(\omega, p_{H5}, \overline{H_{p_{H5}}})$) :



Insertable production

Definition 3.2

Consider $(\omega, p_i, \overline{H_{p_i}})$

A production q : <u>insertable</u> for p_i and $(\omega, p_i, \overline{H_{p_i}})$

$$\overset{\Longleftrightarrow}{def}$$

$\exists$ A production instance $(\omega, q, \overline{H_q})$

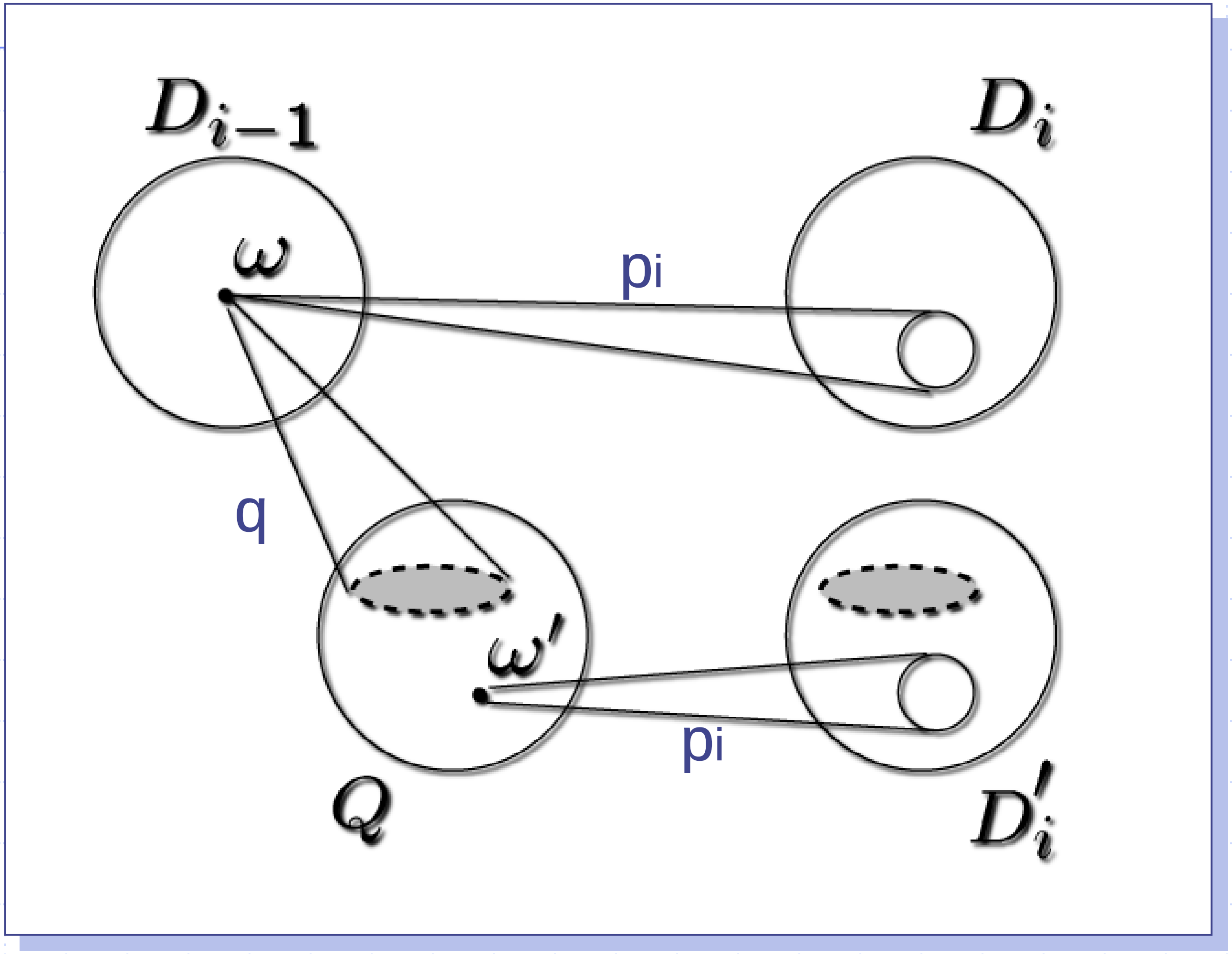
$$\text{s.t. } D_{i-1} \overset{\omega \overline{H_q}}{\Rightarrow_q} Q \overset{\omega' \overline{H_{p_i}}}{\Rightarrow_{p_i}} D'_i$$

and

$$D'_i - (\overline{H_q} - \omega') \approx D'_i$$

for any D_{i-1}

An image of Definition 3.2



Syntactic Insertion of a production instance

Definition 3.3

For a production $q : X_q \to (H_q, C_q) \in P_N$,

An instance sequence S is obtained by insertion of a production instance $(\omega, q, \overline{H_q})$ into "valid" instance sequence $((\omega_1, p_1, \overline{H_{p_1}}), \cdots, (\omega, p_i, \overline{H_{p_i}}), \cdots, (\omega_n, p_n, \overline{H_{p_n}}))$

$\overleftrightarrow{def}$

1. q : insertable for $p_i : X_{p_i} \to (H_{p_i}, C_{p_i})$ and $(\omega, p_i, \overline{H_{p_i}})$,

   and $\cup_{i=1}^{n} \overline{H_{p_i}} \cap \overline{H_q} = \phi$.

2. S is of the form $((\omega_1, p_1, \overline{H_{p_1}}), \cdots, (\omega_{i-1}, p_{i-1}, \overline{H_{p_{i-1}}}),$

   $(\omega, q, \overline{H_q}), (\omega', p_i, \overline{H_{p_i}}), \cdots, (\omega'_n, p_n, \overline{H_{p_n}}))$

   and S is obtained by following algorithm:

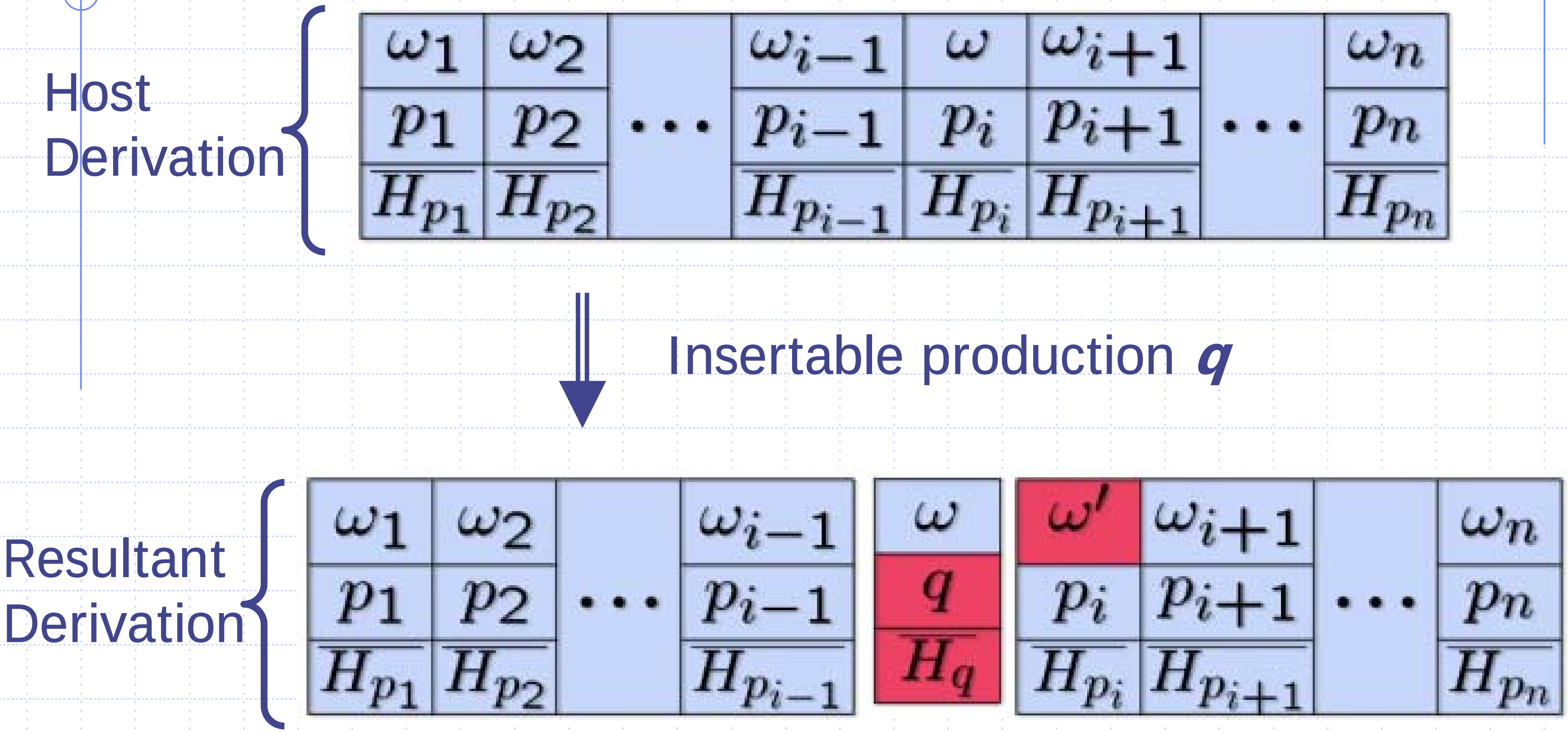
1. Trace a derivation with instance from $(\omega_n, p_n, \overline{H_{p_i}})$ back to $(\omega_{i-1}, p_{i-1}, \overline{H_{p_{i-1}}})$.
2. Apply the production instance $(\omega, q, \overline{H_q})$ to $(\omega_{i-1}, p_{i-1}, \overline{H_{p_{i-1}}})$
3. Apply the production instance sequence $((\omega', p_i, \overline{H_{p_i}}), (\omega'_{i+1}, p_{i+1}, \overline{H_{p_{i+1}}}), \cdots, (\omega'_n, p_n, \overline{H_{p_n}}))$ $\square$

<u>insertable by composite production copy</u> is similarly defined.

Notation

$$\begin{array}{|c|}\hline \omega \\ \hline p \\ \hline \overline{H_p} \\ \hline \end{array} \text{ denotes } (\omega, p, \overline{H_p}).$$

Example (application of insertable prodiction):



Syntactic Insertion of Graph

Definition 3.4

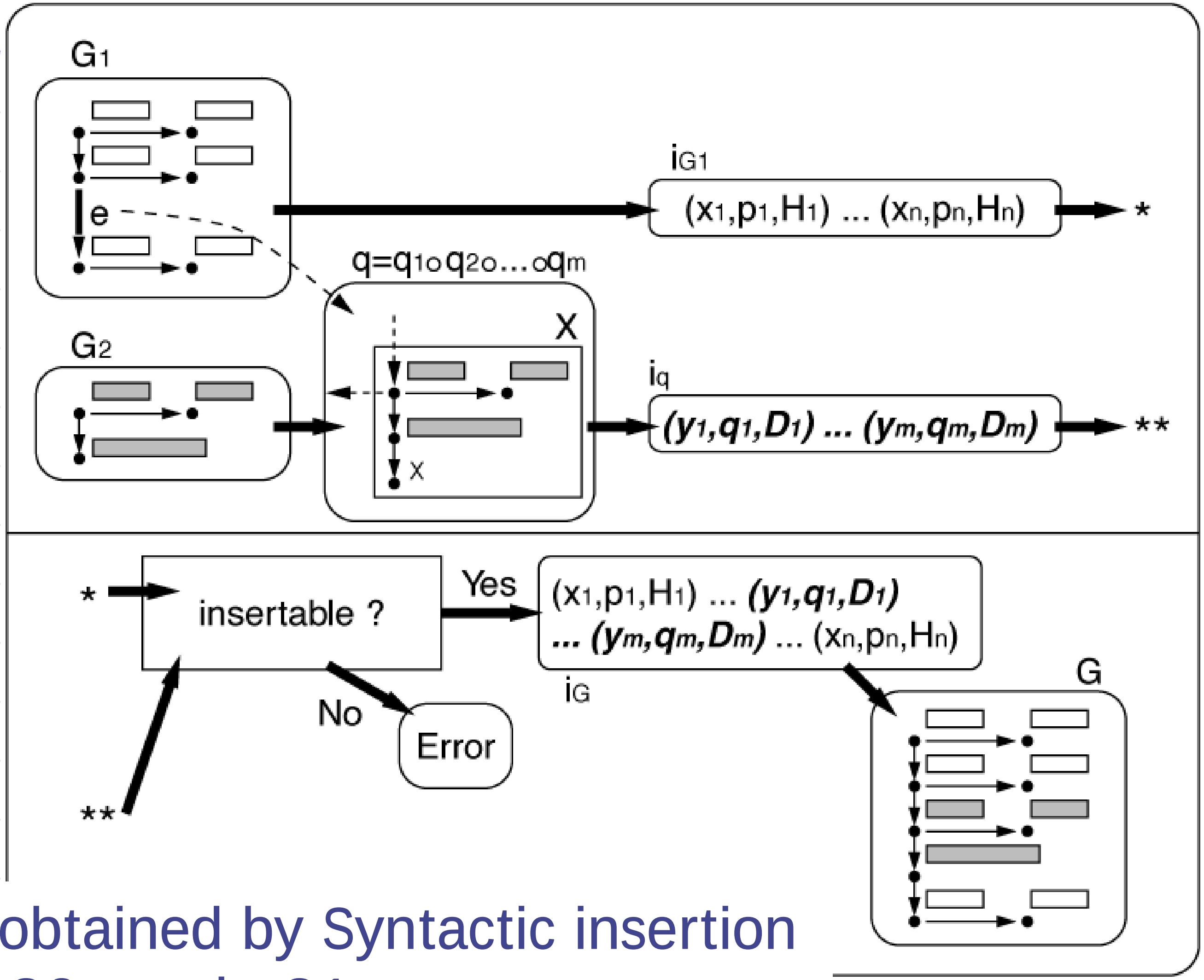
A graph H' is obtained by syntactic insertion of a graph A at an edge x in a graph H.

$\overset{\Longleftrightarrow}{def}$

1. A composite production copy q for A and x exists.
2. There exists an instance sequence i_q for q and an instance sequence i_H for H. An instance sequence S is obtained by insertion of i_q into i_H.
3. H' is derived by S. $\square$

Illustration of Definition 3.4

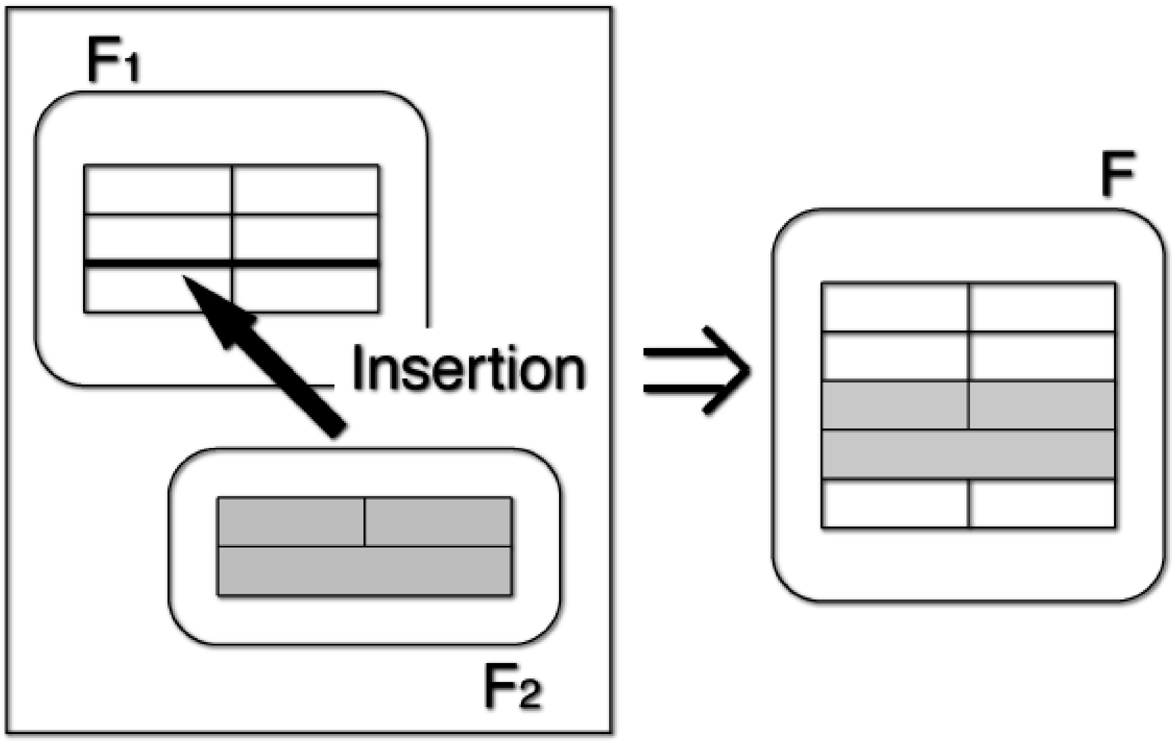
Insertion of G2 into G1 at edge e:



G is obtained by Syntactic insertion of G2 at e in G1

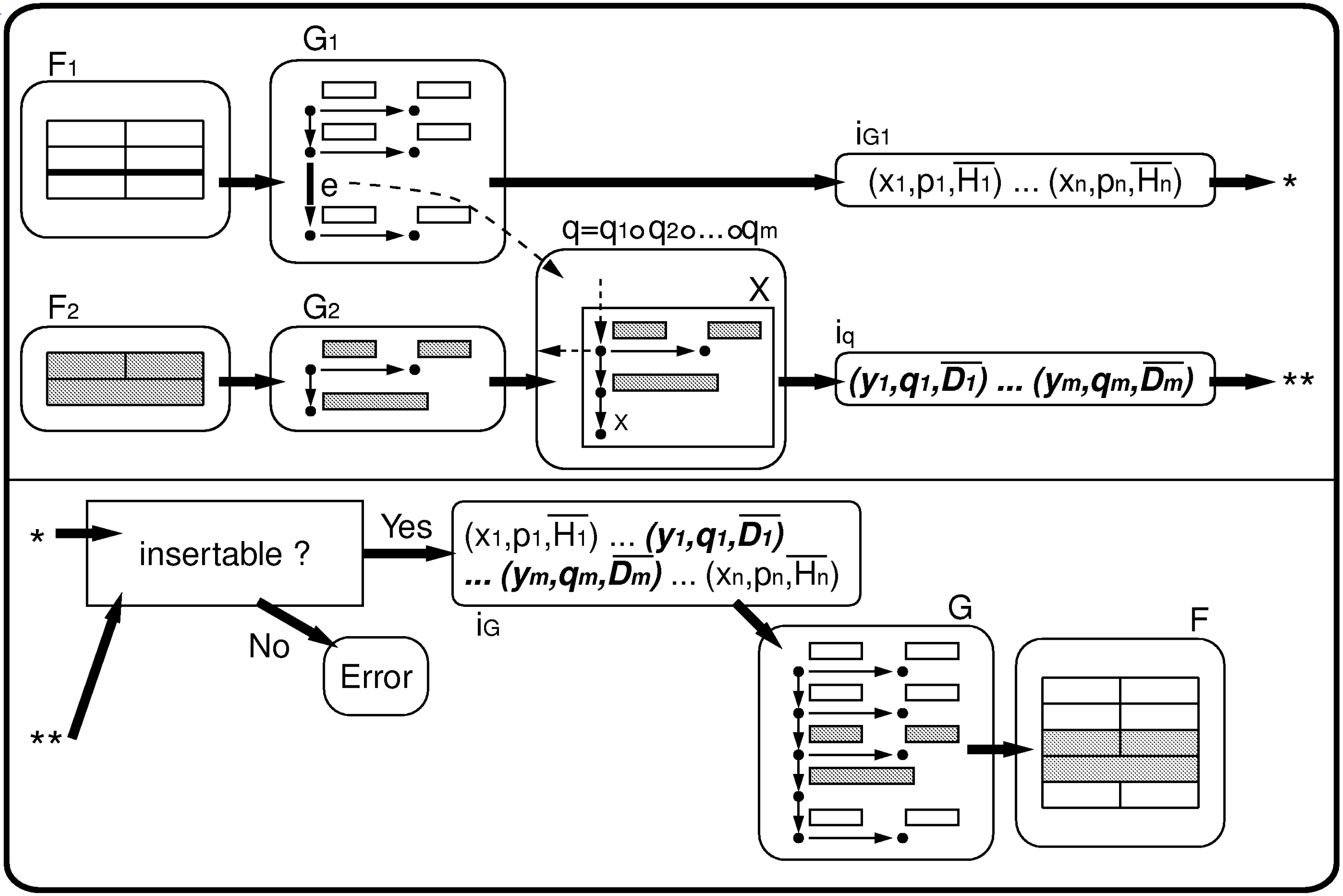
Consider following Form's insertion

Insertion of F2 into F1



An insertion process of a form by Def. 3.4.

Insertion of F2 into F1 at edge e.

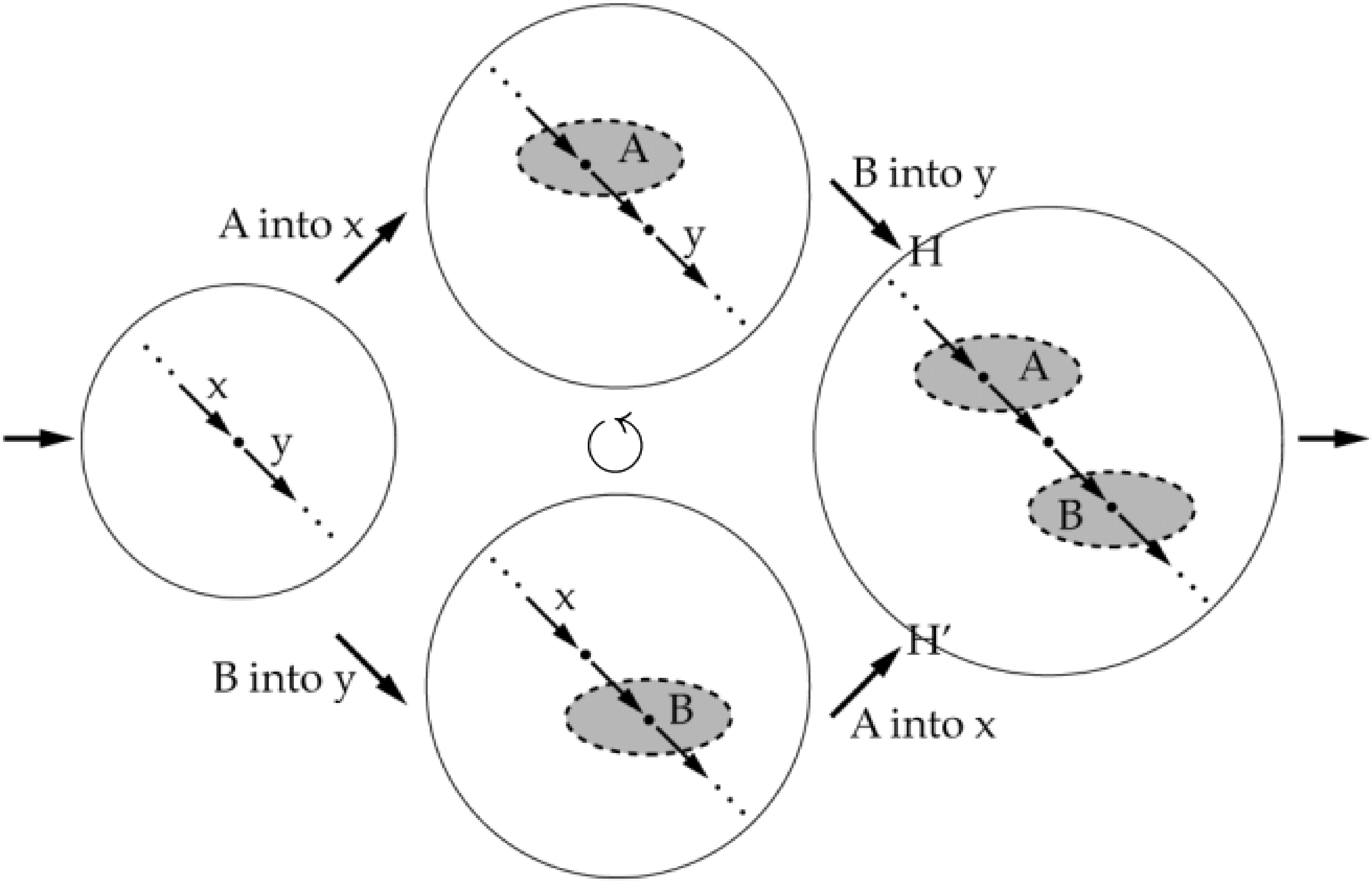


Proposition 3.6

Let H be the graph obtained from G by the insertion of graph a and b at edge x and y respectively in this order, in HNGG. H' be the graph obtained from G by the insertion of b and a at y and x respectively in this order, in HNGG.

$\Longrightarrow H = H'$.

Diagram of Proposition 3.5



Proposition 3.7

Insertion in HNGG is executed in linear time.

Algorithm of insertion

Step1. Construct instances on parsing graphs.

Step2. Syntactic insertion of instance at insert point.

Step3. Attribute evaluation based on an instance of step2.

Step4. Drawing a form based on values of attributes.

4. Conclusion

- **Summary**
  - We proposed a syntactic editing method for tabular forms, based on the attribute edNCE graph grammar.
  - The order of commands in editing doesn’t influence the editing results.
  - Linear time editing algorithm with attribute rules for primitive drawing.

Future Works

- Attribute rules for more sophisticated drawing
- Other edit manipulations representing a cell division manipulation, a cell combination manipulation and so on.
- We are now developing a tabular form editor system utilizing this approach.

Example (application of insertable prodiction):

Host Derivation

$$D_0 \begin{array}{|c|}\hline \omega_1 \\ \hline p_1 \\ \hline H_{p_1} \\ \hline \end{array} D_1 \begin{array}{|c|}\hline \omega_2 \\ \hline p_2 \\ \hline H_{p_2} \\ \hline \end{array} \cdots \begin{array}{|c|}\hline \omega_{i-1} \\ \hline p_{i-1} \\ \hline H_{p_{i-1}} \\ \hline \end{array} D_{i-1} \begin{array}{|c|}\hline \omega \\ \hline p_i \\ \hline H_{p_i} \\ \hline \end{array} D_i \begin{array}{|c|}\hline \omega_{i+1} \\ \hline p_{i+1} \\ \hline H_{p_{i+1}} \\ \hline \end{array} \cdots \begin{array}{|c|}\hline \omega_n \\ \hline p_n \\ \hline H_{p_n} \\ \hline \end{array} D_n$$

$\Downarrow$ Insertable production ***q***

Resultant Derivation

$$D_0 \begin{array}{|c|}\hline \omega_1 \\ \hline p_1 \\ \hline H_{p_1} \\ \hline \end{array} D_1 \begin{array}{|c|}\hline \omega_2 \\ \hline p_2 \\ \hline H_{p_2} \\ \hline \end{array} \cdots \begin{array}{|c|}\hline \omega_{i-1} \\ \hline p_{i-1} \\ \hline H_{p_{i-1}} \\ \hline \end{array} D_{i-1} \begin{array}{|c|}\hline \omega \\ \hline q \\ \hline H_q \\ \hline \end{array} Q \begin{array}{|c|}\hline \omega' \\ \hline p_i \\ \hline H_{p_i} \\ \hline \end{array} D'_i \begin{array}{|c|}\hline \omega_{i+1} \\ \hline p_{i+1} \\ \hline H_{p_{i+1}} \\ \hline \end{array} \cdots \begin{array}{|c|}\hline \omega_n \\ \hline p_n \\ \hline H_{p_n} \\ \hline \end{array} D'_n$$