Double-Pushout Graph Rewriting

Summary

- Applications of Graphs
- Graph rewriting
- Definition of a pushout
- Double-Pushout and rules
- Example
- Tools

Applications of graphs

- Visual Modeling and Specification
  - Representation of System States
  - Class/Object diagrams
  - Entity-relationship diagrams
  - Petri nets
- Model Transformation
  - Shift toward model driven code generation, analysis, computation and documentation
- Concurrency and Distribution
  - System behaviour represented by state changes is modeled by rule-based graph manipulations

- Software Development
  - Architectural structures and technical structures
  - Changes during software development process
  - Describe structural evolution in a rule-based way

What is a graph transformation?

The main idea of graph transformation is the rule based modification of graphs. The core of a rule (or production), $p = (L, R)$ is a pair of graphs (L, R) known as the left-hand side L and the right-hand side R. Applying the rule $p = (L, R)$ means

- to find a match L in the source graph and
- replacing L by R, thus leading to the target graph

Definition of a pushout

There are many approaches to graph transformations but here we discuss the algebraic approach which is based on pushout constructions. There are two variants of pushout:

- double-pushout
- single-pushout

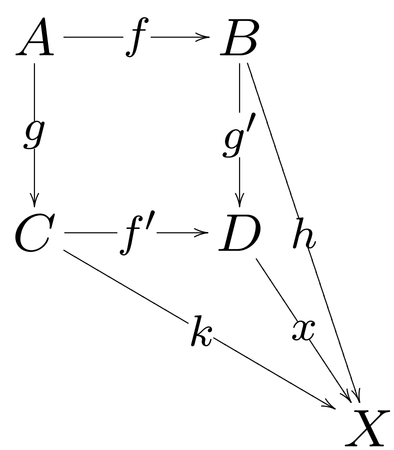
Definition: Given morphisms $f : A \to B$ and $g : A \to C$ in a category **C**, a pushout (D, f', g') over f and g is defined by

- a pushout object D and
- morphisms $f' : C \to D$ and $g' : B \to D$ with $f' \circ g = g' \circ f$

such that for all objects X and morphisms $h : B \to X$ and $k : C \to X$ with $k \circ g = h \circ f$ there is a unique morphism $x : D \to X$ such that $x \circ g' = h$ and $x \circ f' = k$

Definition of a pushout (continued)

Visually:



We write $D = B +_A C$ for the pushout object D, where D is the gluing of B and C via (A, f, g).

Double-pushout approach

Two gluing constructions are used to model a graph transformation.
In this approach a production is defined as $p = (L, K, R)$, where:

- L describes the preconditions of the rule
- R describes the postconditions of the rule
- K describes a graph part which has to exist to apply the rule
- $L \backslash K$ describes the part to be deleted
- $R \backslash K$ describes the part to be created

Double-pushout step by step process

Let m be the match of the left-hand side L in G such that m is structure preserving and let p be a production then:

- first create the context graph as $D := (G \backslash m(L)) \bigcup m(K)$ then
- then check for $gluing$ condition
  - $gluing\ condition$ - points in L which are source or target in $G \backslash L$ (S_B) known as the boundary point and points given by K known as the gluing points (S_K) must satisfy $S_B \subseteq S_K$
- add R through K to D, to get H as $H = R +_K D$

Since K is in L, R, D we express that there are morphisms $K \rightarrow L$, $K \rightarrow R$, $K \rightarrow D$ and pushout constructions $G = L +_K D$, $H = R +_K D$

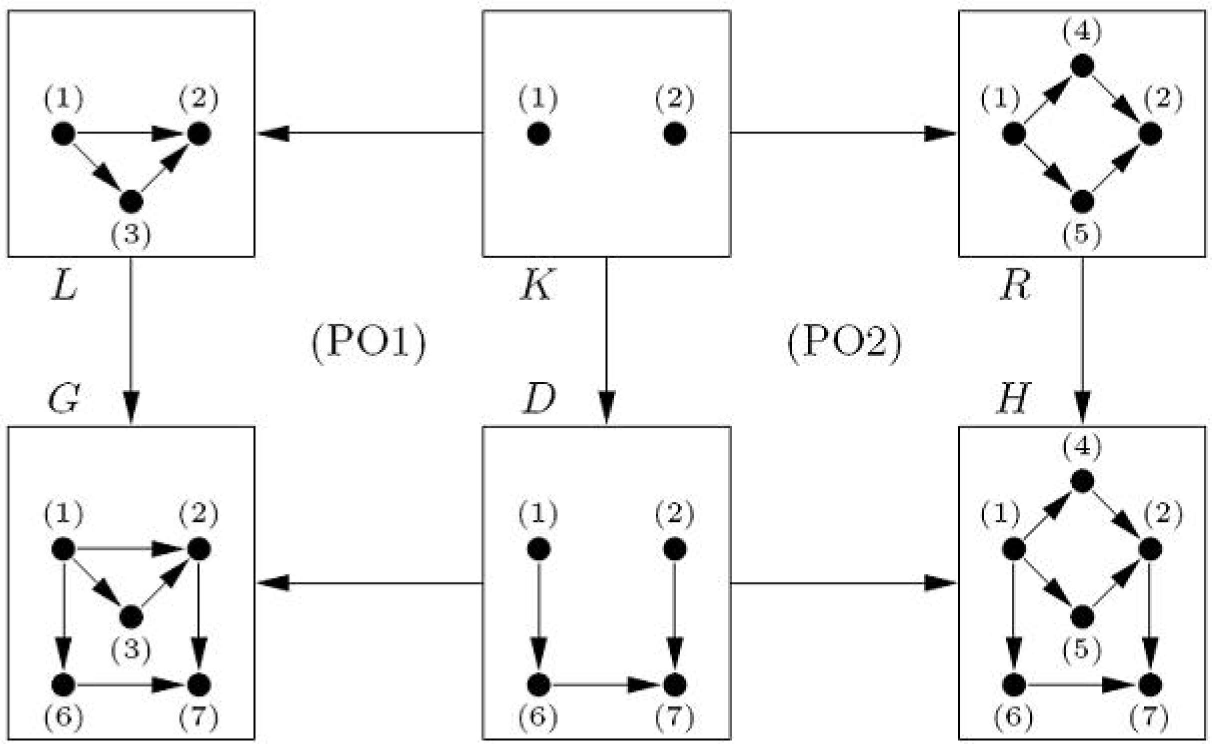
Double-pushout step by step process (continued)

$$\begin{array}{ccccc} L & \longleftarrow & K & \longrightarrow & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow & (2) & \downarrow \\ G & \longleftarrow & D & \longrightarrow & H \end{array}$$

(1) construct D such that the gluing of L and D via K is equal to G

(2) construct the gluing of R and D via K leading to the graph H

Simple Example



Tools

- AGG (http://tfs.cs.tu-berlin.de/agg/down.html)
- Progres (PRogrammed Graph REwriting Systems)
- Grace
- Fujaba

References:

- Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, Gabriele Taentzer. Introduction to the Algebraic Theory of Graph Grammars. In *Proc. Ehrig-1978*

- Hartmut Ehrig, Manfred Nagl, Grzegorz Rozenberg, Azriel Rosenfeld. Graph-Grammars and Their Application to Computer Science, 3rd International Workshop.

- Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, Michael Löwe.Algebraic Approaches to Graph Transformation, Part I: Basic Concepts and Double Pushout Approach. In *Collection Corradini-Montanari-Rossi-Ehrig-Heckel-Loewe-1997*

- Wolfram Kahl, Frank Derichsweiler. Declarative Term Graph Attribution for Program Generation.