Algebraic Approaches to Graph Grammars

Phawade Ramchandra Babasaheb

Contents

Chapter 1

Introduction

The main idea of graph grammars is the rule based modification of graphs where each application of a graph rule leads to a graph transformation step. Graph grammars can be used on one hand to generate graph languages in analogy to the idea to generate string languages by Chomsky grammars in formal language theory. On the other hand graphs can be used to model the states of all kinds of systems which allows to use graph transformations to model the state changes of these systems.

This allows us to apply graph grammars and graph transformation systems to a wide range of fields in computer science like formal language theory, compiler construction, distributed systems modelling, logical and functional programming, AI, visual modeling etc.

The wide applicability is due to the fact that graphs are a very natural way of explaining complex situations on an intuitive level. e.g. visualization of software and hardware architectures, data and control programs, evolution diagrams of nondeterministic processes etc. Like the token game for Petri nets, a graph transformation brings dynamism since it can describe the evolution of graph structures. Therefore, graph transformations become attractive as a modeling and programming paradigm for complex-structured software and graphical interfaces. In particular, graph rewriting is promising as a comprehensive framework in which the transformation of all these very different structures can be modeled and studied in a uniform way.

1.1 What is Graph transformation?

In fact, graph transformation has at least three different roots
– from Chomsky grammars on strings to graph grammars
– from term rewriting to graph rewriting
– from textual description to visual modeling.

Altogether we use the notion graph transformation to comprise the concepts of graph grammars and graph rewriting. In any case, the main idea of graph transformation is the rule-based modification of graphs as shown in the following figure.

The core of a rule or production p=(L, R) is a pair of graphs (L, R), called left hand side L and right hand side R. Applying the rule p=(L, R) means to find a match of L in the source graph and to replace L by R leading to target graph of graph transformation. The main problem is how to connect R with the context in the target graph. In fact, there are three different solutions how to handle this problem leading to different graph transformation approaches, which are summarized below.

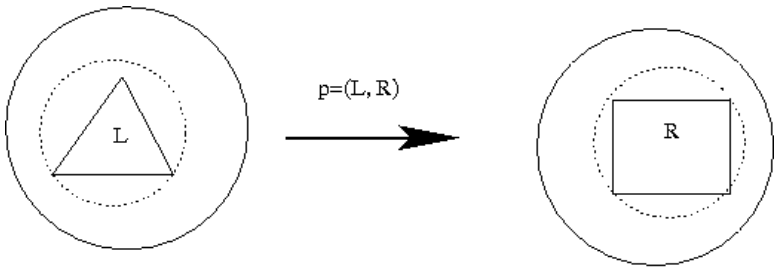


Figure Rule based Modification of Graphs

1.2 Overview of Different Approaches

1. The node label replacement approach,mainly developed by Rozenberg, Engelfriet and Janssens, allows replacing a single node as left hand side L by an arbitrary graph R. The connection of R. The connection of R with the context is determined by embeding rules depending on node labels.

2. The hyperedge replacement approach, mainly developed by Habel, Kreowsky and Drewes, has as L a labeled hyperedge, which is replaced by an arbitrary hypergraph R with designated attachment nodes corresponding to the nodes of L. The gluing of R with the context at corresponding attachment nodes leads to the target graph.

3. The algebraic approaches are based on pushout and pullback constructions in the category of graphs, where pushouts are used to model the gluing of graphs. Algebraic approach is an unified approach to graph transformations, simplifying proofs in complex situations also.

1.3 Aims of the report

The aim of this survey report is to study two algebraic approaches ,Double Pushout Approach (DPO) and Single Pushout Approach (SPO) to graph transformation. We have studied some of the main results from theory of parallelism, using these two approaches. We have studied chapter 3 and chapter 4 of Handbook of graph grammars and Computing by Graph Transformation :volume1;foundations.

In section 2.1 of this report we introduce basic concepts of DPO Approach, starting with category of graphs, pushouts, gluing conditions ,definitions of production, direct derivation, and derivation. In section 2.2 we present concepts and results concerning parallelism and independence of graph transformations . The main results are Local Church Rosser Theorem and the Parallelism Theorem, analysis and synthesis constructions. Section 2.3 presents various models of computations for graph grammars on different levels of abstractions, i.e., various categories having graphs as objects and graph derivations as arrows. All such models are built from the concrete ones by imposing suitable equivalences on graphs and derivations.

In section 3.1 of chapter 2 we present graph transformation using SPO approaches, introducing and using the concepts of single pushout construction, deletion by co-equalizer , and some properties of pushouts. Also the translation of DPO rules to SPO and vice-versa under suitable conditions. In section 3.2 we present the main results of independence and parallelism as in the case of DPO Approach.

Finally in chapter 4 we conclude the report.

Chapter 2

DPO approach to graph transformations

2.1 Graph transformations based on DPO Approaches

Definition 2.1.1. (Graph, Graph morphism)

Labeled Graph:

Given two fixed alphabets Ω_V and Ω_e for node and edge labels, respectively, a labeled graph over Ω_V and Ω_E is a tuple $G =< G_V, G_E, s^G, t^G, lv^G, le^G >$, where G_V is a set of vertices(or nodes), G_E is a set of edges (or arcs), $s^G, t^G : G_E \to G_V$ are the source and target functions, and $lv^G : G_V \to \Omega_V$ and $le^G : G^E \to \Omega_E$ are the node and the edge labeling functions, respectively.

Graph morphism:

A graph morphism $f : G \to G'$ is a pair $f =< f_V : G_V \to G'_V, f_E : G_E \to G'_E >$ of functions which preserve sources, targets, and labels i.e., which satisfies $f_V \circ t^G = t^{G'} \circ f_E$, $f_V \circ s^G = s^{G'} \circ f_E$, $lv^{G'} \circ f_V = lv^G$, $le^{G'} \circ f_E = le^G$.

Graph isomorphism:

A graph morphism f is an isomorphism if both f_V and f_E are bijections. If there is an isomorphism from graph G to graph H, then we write $G \cong H$.

Automorphism:

An automorphism of graph G is an isomorphism $\Phi : G \to G$; it is non-trivial if $\Phi \neq id_G$.

The category having labeled graphs as objects and graph isomorphisms as arrows is called $\mathcal{GRAPH}$.

Definition 2.1.2. (production, graph grammar)

A graph production:

A graph production $p : (L \leftarrow K \rightarrow R)$ is composed of a production name p and a pair of injective graph morphisms $l : K \to L$ and $r : K \to R$. The graphs L, K and R are called the left-hand side(lhs), the interface, and the right-hand side(rhs) of p, respectively.

Graph grammar:

A graph grammar G is a pair $G =< p : L \leftarrow K \rightarrow R, G_o >$ where the first component is a family of productions indexed by production names in P and G_o is the start graph.

A production is depicted in the following figure.

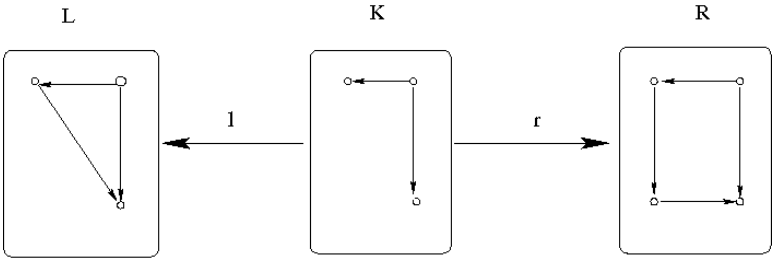


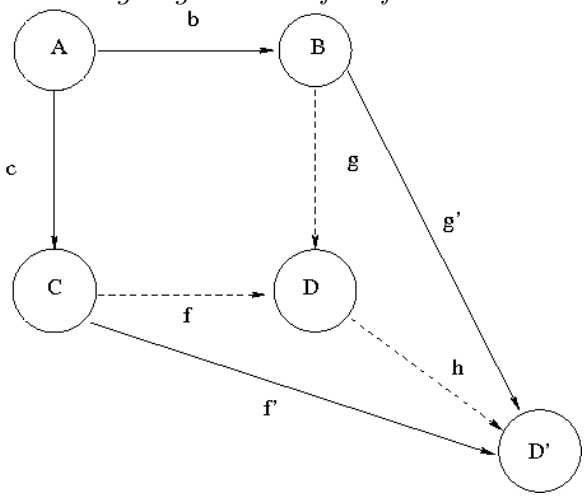
Figure :A production p with L, K and R

Definition 2.1.3. (pushout, pushout complement)

pushout:

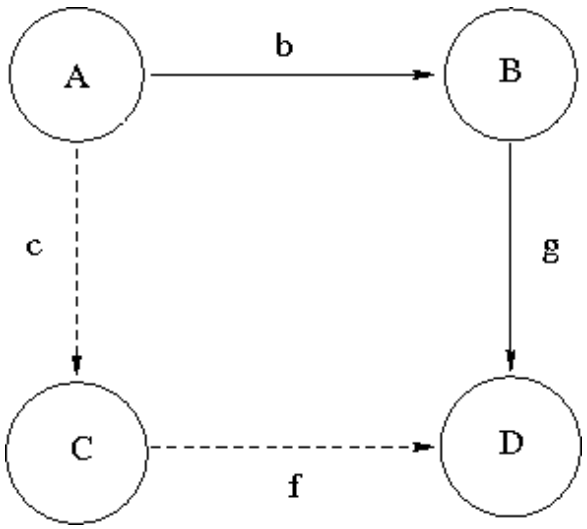
Given a category $\mathcal{C}$ and two arrows $b : A \to B$, $c : A \to C$ of $\mathcal{C}$, a triple $< D, g : B \to D, f : C \to D >$ as in the diagram below is called a pushout of $< b, c >$ if following conditions are satisfied,
1) commutativity: $g \circ b = f \circ c$, and
2) universal property: for all objects D' and arrows $g' : B \to D'$ and $f' : C \to D'$, with $g' \circ b = f' \circ c$, there exists a unique arrow $h : D \to D'$ such

that $h \circ g = g'$ *and* $h \circ f = f'$.



pushout complement:

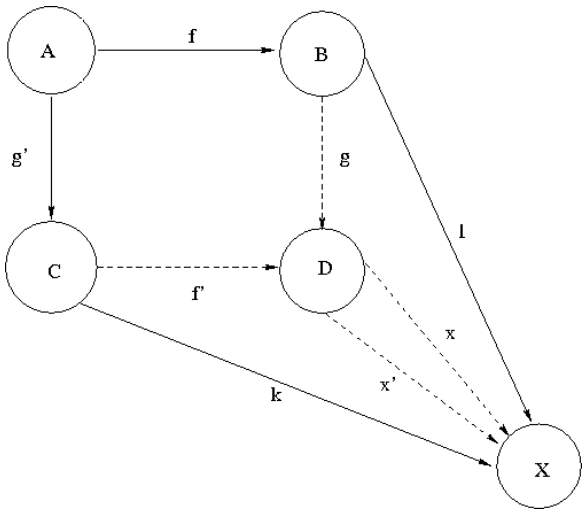
Given arrows $b : A \to B$ *and* $g : B \to D$ *a pushout complement of* $< b, g >$ *is a triple* $< C, c : A \to C, f : C \to D >$ *such that* $< D, g, f >$ *is a pushout of* $< b, c >$. *In this case* C *is called a pushout complement object of* $< b, g >$.



In category $\mathcal{SET}$ which have sets as objects and total functions as arrows, pushout of two arrows always exists.

Lemma 2.1.4. *pushout of two arrows always exists in category* $\mathcal{SET}$.

Proof. In $\mathcal{SET}$, pushout $< C, f' : C \to D, g' : A \to C >$ over $f : A \to B$ and $g : A \to C$ is the quotient set $D = B \uplus C|_{\equiv}$ of the disjoint union of B and C, modulo the equivalence relation $\equiv$, which is the least equivalence relation such that for all $a \in A$ $(f(a) \equiv g(a))$ and $f'(c) = [c]$ for $c \in C$ and $g'(b) = [b]$ for $b \in B$.



Consider the diagram given above. To prove that D is a pushout we have to prove following properties.
1)commutativity: $g' \circ f = f' \circ g$, and
Let us assume that $a \in A$ and $f(a) = b$, $g(a) = c$.
therefore $(b \equiv c)$, by construction of D.
$f' \circ g(a) = f'(g(a))$
$= f'(c)$
$= \{b, c\}$, by construction of f'
$= g'(b)$, by construction of g'

$= g'(f(a))$ by assumption
$= g' \circ f(a)$.

2)universal property: for all objects X and arrows $l : B \to X$ and $k : C \to X$, with $k \circ g = l \circ f$, there exists a unique arrow $x : D \to X$ such that $x \circ f' = k$ and $x \circ g' = l$.

To prove above property first of all we construct morphism $x : D \to X$ and then prove it's uniqueness. We remember that elements of D are equivalence classes. So we have to consider two cases for any $d \in D$.

case 1. $|d| = 1$.
Assume that $d = \{c\}$ i.e. by construction c was not an image of any element of A under g. In this case $[c] = \{c\}$ and $\{c\}$ is mapped to $k(c)$ i.e. $k(c) = \{c\}$.

case 2. $|d| > 1$.
In this case there exist atleast one element a in A such that $g(a) = c$ and by construction $\exists b \in B$ such that $f(a) = b$. Therefore $[b] = [c] = \{b, c\}$.
therefore $x(\{b, c\}) = x([b]) = x([c]) = (k(g(a)) = l(f(a))$.

First, we prove $x \circ f' = k$.
In case 1, there do not exist any element $a \in A$ such that $f(a) = c$, therefore $[c] = \{c\}$, and by construction of x $x(f'(c) = x([c]) = x(\{c\}) = k(c)$.
In case 2, there exist atleast one element a in A such that $g(a) = c$ and by construction $\exists b \in B$ such that $f(a) = b$. Therefore $[b] = [c] = \{b, c\}$.
thereofore by construction of x, $x(f'(c) = x([c] = x(\{b, c\}) = k(g(a)) = k(c)$ as required.

Second, we prove $x \circ g' = l$.
In case 1, there do not exist any element $a \in A$ such that $g(a) = b$, therefore $[b] = \{b\}$, and by construction of x $x(g'(b) = x([b]) = x(\{b\}) = l(b)$.
In case 2, there exist atleast one element a in A such that $f(a) = b$ and by construction $\exists c \in C$ such that $g(a) = c$. Therefore $[c] = [b] = \{b, c\}$.
thereofore by construction of x, $x(g'(b)) = x([b]) = x(\{b, c\}) = l(f(a)) = l(b)$ as required.

Third, we prove uniqueness of above constructed morphism $x : D \to X$.
Suppose that above morphism is not unique i.e. there exists another mor-

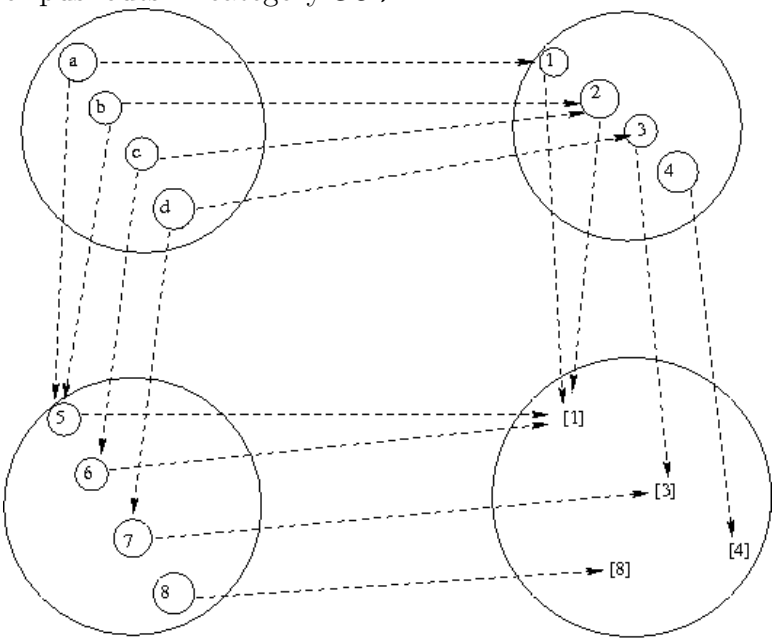
phism $x' : D \to X$, such that $x' \circ f' = k$, and $x' \circ g' = l$. If x and x' are not the same mappings, it means that $\exists d \in D$ such that $x(d) \neq x'(d)$, i.e. $\exists s, s' \in X$ such that $x(d) = s$ and $x'(d) = s'$ and $s \neq s'$.

case 1. $|d| = 1$
Assume that $d = \{c\}$.
$k(c) = x'(f'(c)) = x'([c]) = x'(c) = s$.
and also $k(c) = x(f'(c)) = x([c]) = x(c) = s'$, a contradiction.

case 2. $|d| > 1$
Without loss of generality assume that $d = \{b, c\}$, and $f(a) = b$,$g(a) = c$.
$k(c) = x'(f'(c)) = x'([c]) = x'(\{b, c\}) = s'$.
and also $k(c) = x(f'(c)) = x'([c]) = x'(\{b, c\}) = s$, a contradiction.

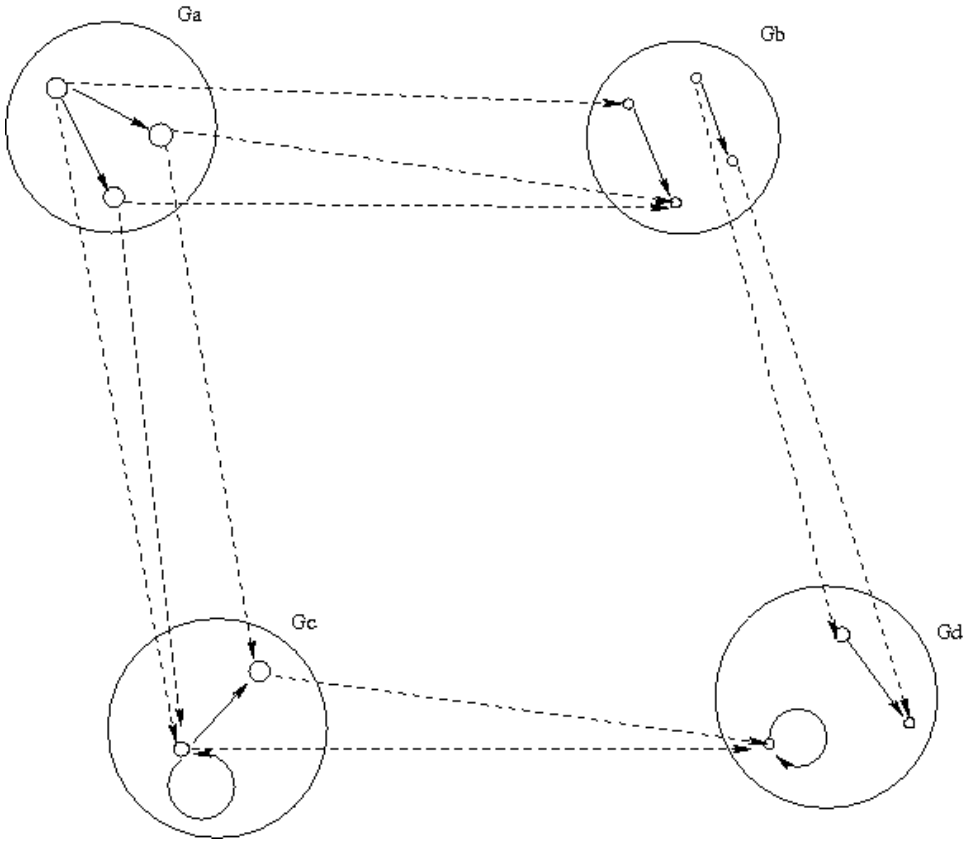
□

We provide an example in following diagram to illustrate the construction of pushouts in category $\mathcal{SET}$.



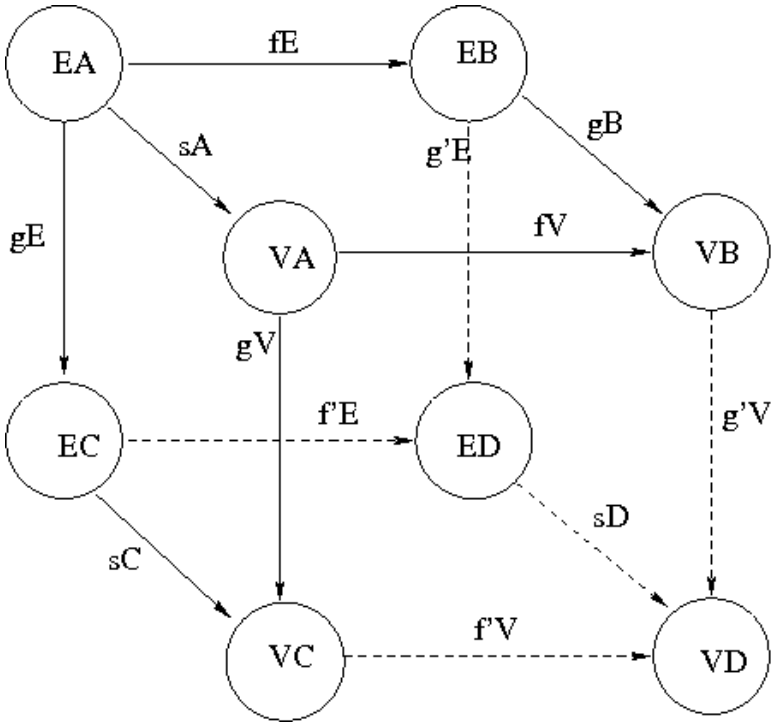
(1,5), (2,5), (2,6), (3,7) are in equivalnece classes of disjoint union of B and C.
[1]=[2]=[5]=[6]={1,2,5,6}, [3]=[7]={3,7}, [4]={4}, [8]={8}

In category $\mathcal{GRAPH}$ the pushout of two arrows always exists as well: It can be constructed componentwise (as a pushout in the $\mathcal{SET}$) for the nodes and for the edges, and the source, target, and labeling mappings are uniquely determined. The following diagram shows one example.

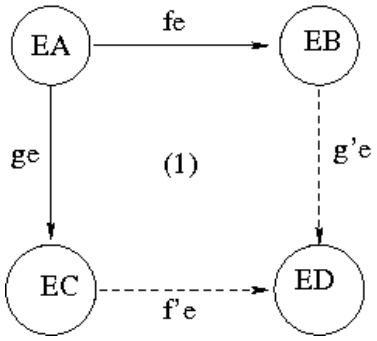


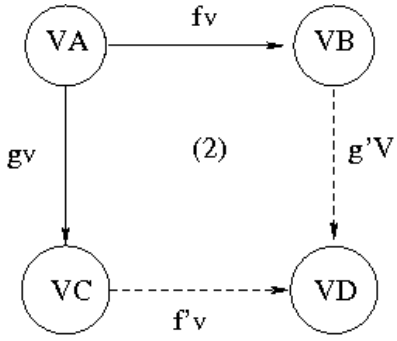
Lemma 2.1.5. *In category $\mathcal{GRAPH}$ the pushout of two arrows always exists.*

Proof. Consider the construction as shown in following diagram. We have built the cube using source functions only. A similar cube using target functions can be built. Proof for the other cube is same as the proof using source functions which is given below.

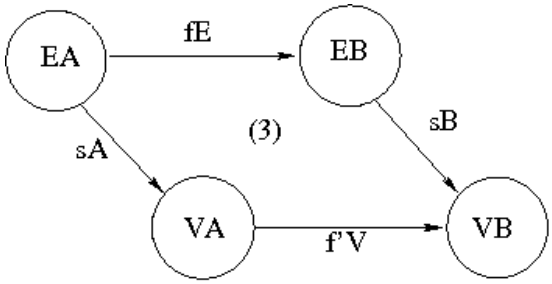


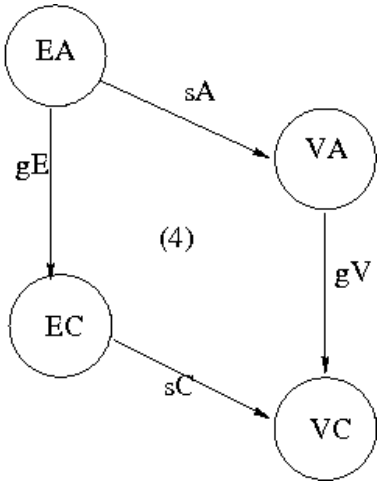
We have to show that all six squares labelled (1) to (6) commute. For squares (1),(2) shown in following diagram since we are considering only set of edges and vertices their commutativity follows by lemma 1.



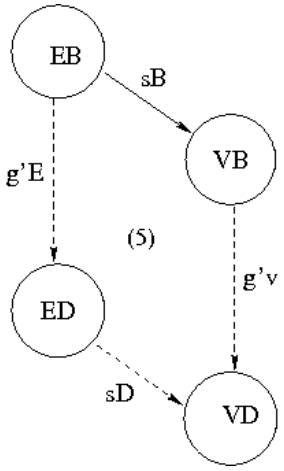


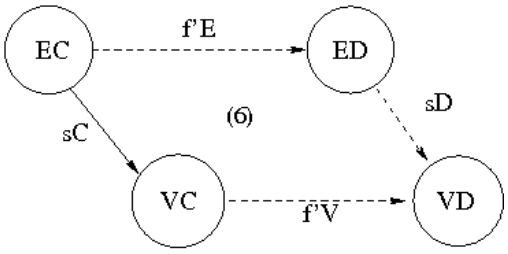
For squares (3),(4) shown in following diagram since we are considering only set of edges and vertices in the respective sets so their commutativity follows from the fact that souce and target are preserved by the morphisms which are already given and we are not considering any element of pushout graph or morphism constructed.





For squares (5),(6) as shown in the following diagram we have to prove $f'_V \circ s_C = s_D \circ f'_E$ and $s_D \circ g'_E = g'_V \circ s_B$



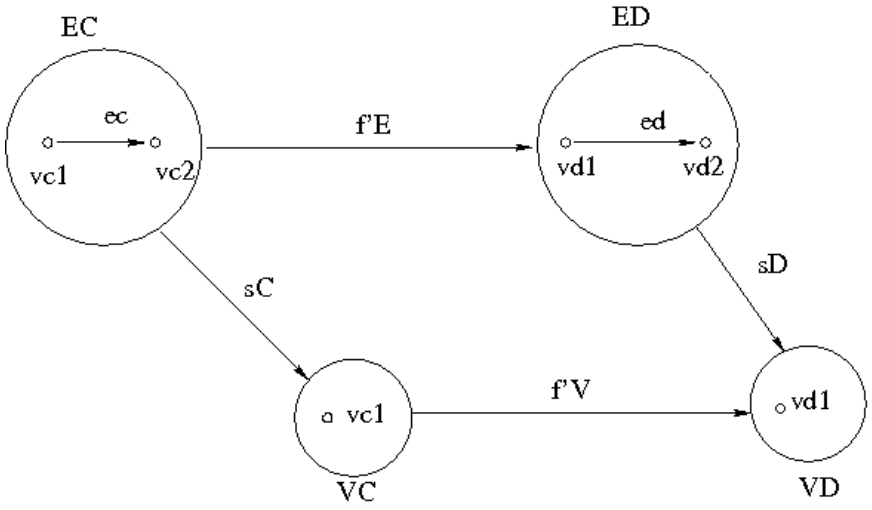


Now consider following paths, in the diagrams,
$f'_V \circ s_C \circ g_E = f'_V \circ g_V \circ s_A = g'_V \circ f_V \circ s_A = g'_V \circ s_B \circ f_E$
and $s_D \circ f'_E \circ g_E = s_D \circ g'_E \circ f_E = s_D \circ f'_E \circ g_E = f'_V \circ s_C \circ g_E$

But note that we also need to prove that $g'_V \circ f_V \circ s_A = s_D \circ g'_E \circ f_E$, so if we prove commutativity of diagram (5) , the commutativity of diagram (6) is also proved and vice versa.

To prove $f'_V \circ s_C = s_D \circ f'_E$, it is sufficient to prove that $f'_V \circ s_C \circ g_E = s_D \circ f'_E \circ g_E$.

Now let us assume that $f'_V \circ s_C \circ g_E \neq s_D \circ f'_E \circ g_E$. without loss of generality consider arbitrary edge $e_d \in D$ and other edges and vertices involved in it's construction as shown in the following diagram.



Here ec is an edge in graph C and vc1,vc2 are vertices in graph C

ed is an edge in graph D and vd1,vd2 are vertices in graph D

Now $(f'_V \circ s_C \circ g_E)(e_a) = (f'_V \circ s_C)(e_a) = f'_V(V_{c1}) = [V_{c1}] = \{V_{c1}, V_{b1}\}$ and $(s_D \circ f'_E \circ g_E)(e_a) = (s_D \circ f'_E)(e_c) = s_D([e_c]) = s_D(\{e_c, e_b\}) = \{V_{c1}, V_{b1}\}$ which is a contradiction. □

Lemma 2.1.6. (Gluing Condition) *Let $b : A \to B$ and $g : B \to D$ be two morphisms in category $\mathcal{GRAPH}$. Then there exists a pushout complement $< C, c : A \to C, f : C \to D >$ of $< b, g >$ if and only if the following conditions are satisfied:*
[**Dangling condition**] *No edge $e \in D_E - g_E(B_E)$ is incident to any node in $g_V(B_V - b_V(A_V))$;*
[**Identification condition**] *There is no x, $y \in B_V \cup B_E$ such that $x \neq y$, $g(x) = g(y)$ and $y \notin b(A_V \cup A_E)$.*

Proof. ($\Rightarrow$) We assume that both dangling condition and indentification condition are satisfied and prove existence of pushout complement.

1. First we construct C, the pushout complement object of $< b, g >$, and morphisms $c : A \to C$, $f : C \to D$ as follows.

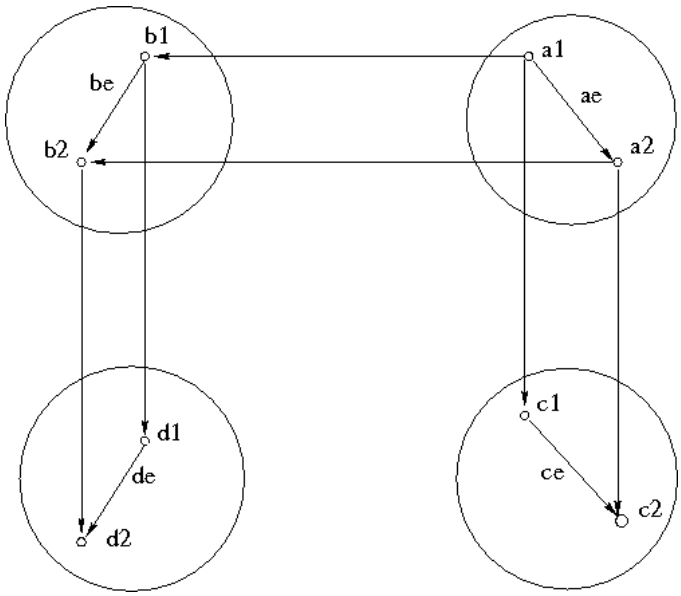
   Elements of C are the elements d in D for which $b^{-1}(g^{-1}(d))$ exists in A. For any element a in A c(a) is the element d in C, which is by construction is in C which satisfies the condition that $a = b^{-1}(g^{-1}(d))$. For any element c in C f(c) is element d in D such that $c(a) = b^{-1}(g^{-1}(d))$.

   c is a morphism and it preserve source, target and labels. i.e. c preserve following conditions:
   (1)$C_V \circ t^A = t^C \circ C_e$
   (2)$C_V \circ s^A = s^C \circ C_e$
   (3)$lv_C \circ C_V = lv^A$
   (4)$le_C \circ C_E = le^A$

   Clearly C is a morphism . If not then it means that there exist an element in a in A which is not mapped to any element of C. This is possibly only when stated condition $a = b^{-1}(g^{-1}(d))$ is not satisfied, i.e. $b^{-1}(g^{-1}(d))$ does not exist or when it exists it is not equal to a. As far as the former situation is considered this can only when there exist an element in B which is mapped to some element d in D via g but is itself not mapped by B. But this can not happen since d is the element in D which is preserved in C by construction. Therefore if such b in B exists it violates the dangling condition or identification condition.

   Consider the diagram given below. To prove condition (1), take any arbitrary edge ae in A. Since b and g are morphisms corresponding edges be and ce exists in B and D. Now lhs of condition (1) gives,

$C_V \circ t^A(ae) = C_V(a2) = c2$. and rhs of condition gives $t^C \circ C_e(ae) = t^C(ce) = d2 = c2$ by construction of elements of C. Similarly other 3 conditions are proved easily.



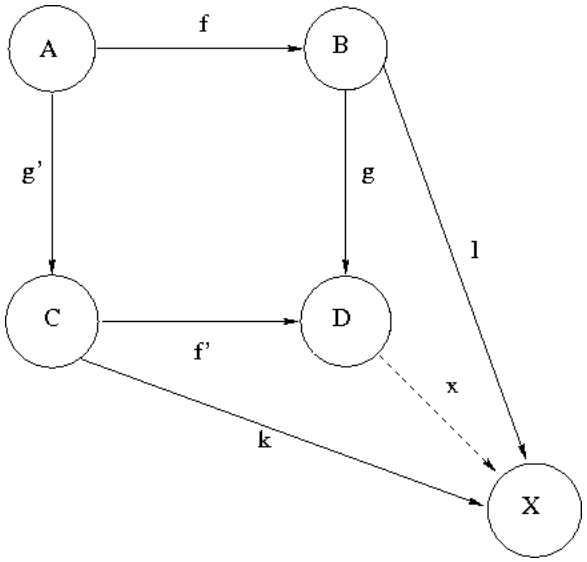
2. Second, we prove that D is pushout. To prove that D is a pushout, i.e. for any graph X as shown in the following diagram there exist a unique morphism $x : D \to X$ such that the shown diagram commutes.

   Here we first prove that the $g \circ b = f \circ c$
   without loss of generality we consider node a1 in the above diagram. For lhs $g(b(a1)) = g(b1) = d1$, while rhs is $f(c(a1)) = f(c1)$ by construction of C and c.
   $= d1$ by construction of C and f.

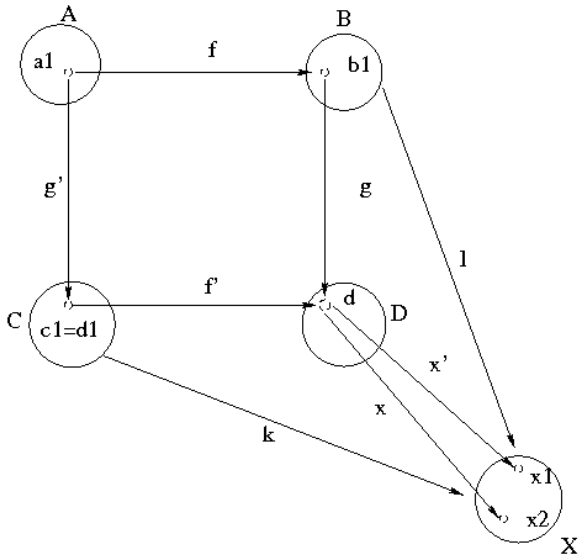
   Now, we prove that for any graph X as shown in the following diagram there exist a unique morphism $x : D \to X$ such that the above diagram commutes. Construct x as $x(d) = \{x1 \in X | b^{-1}(l^{-1}(x1)) = b^{-1}(g^{-1}(d))\}$ or $\{x1 \in X | c^{-1}(f^{-1}(x1)) = c^{-1}(k^{-1}(d))\}$.

   To prove that $x \circ f = k$, consider any element $c1 \in C$. $f(c1)$ is mapped to some element $d \in D$ which is same as $c1$ by construction of C.Now x maps that element to some $x1 \in X$. on the rhs $k(c1)$ is mapped to same $x1 \in X$, by construction of c, C and x.

Similarly $x \circ g = l$ is proved.



To prove that this morphism x constructed as above is unique, let us assume that there exist another morphism $x' \neq x$, such that $x' \circ f = k$ and $x' \circ g = l$ are satisfied, as shown in the diagram shown below.

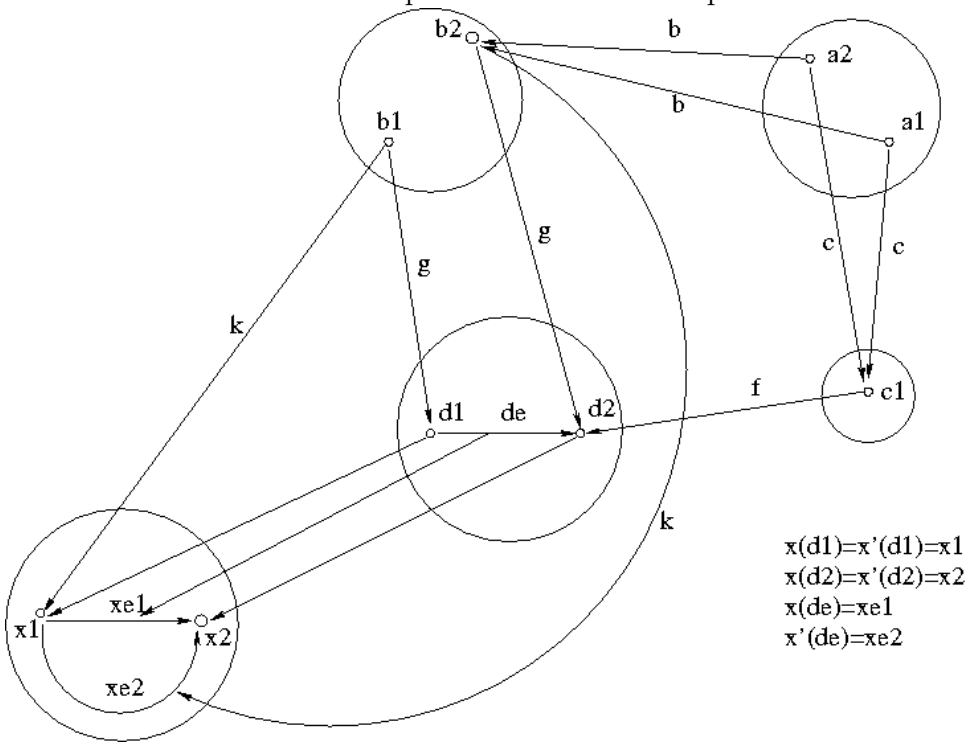


It means that there exist an element $d \in D$ such that $x(d) \neq x'(d)$ i.e. there exist $x1 \in X$ and $x2 \in X$ such that $x1 \neq x2$ but $x(d) = x1$ and $x'(d) = x2$.

without loss of generality let us assume that there exist elements $a1 \in A$ and $b1 \in B$ such that $b(a1) = b1$ and $g(b1) = d$. By the commutativity of diagram we know that $x \circ g = l = x' \circ g$.

but $x'(g(b1)) = x2 \neq x1 = x(g(b1))$, a contradiction.

($\Leftarrow$) We assume existence of pushout complement to prove that both dangling and identification conditions must be satisfied. Equivalently we prove this by if either of the conditions is not satisfied then pushout complement does not exist. It is sufficient to prove that D is not a pushout.



First if the dangling condition is not satisfied, i.e., there is an edge $de \in D_E - g_E(B_E)$ is incident to some node in $g_V(B_V - b_V(A_V))$; We prove our claim

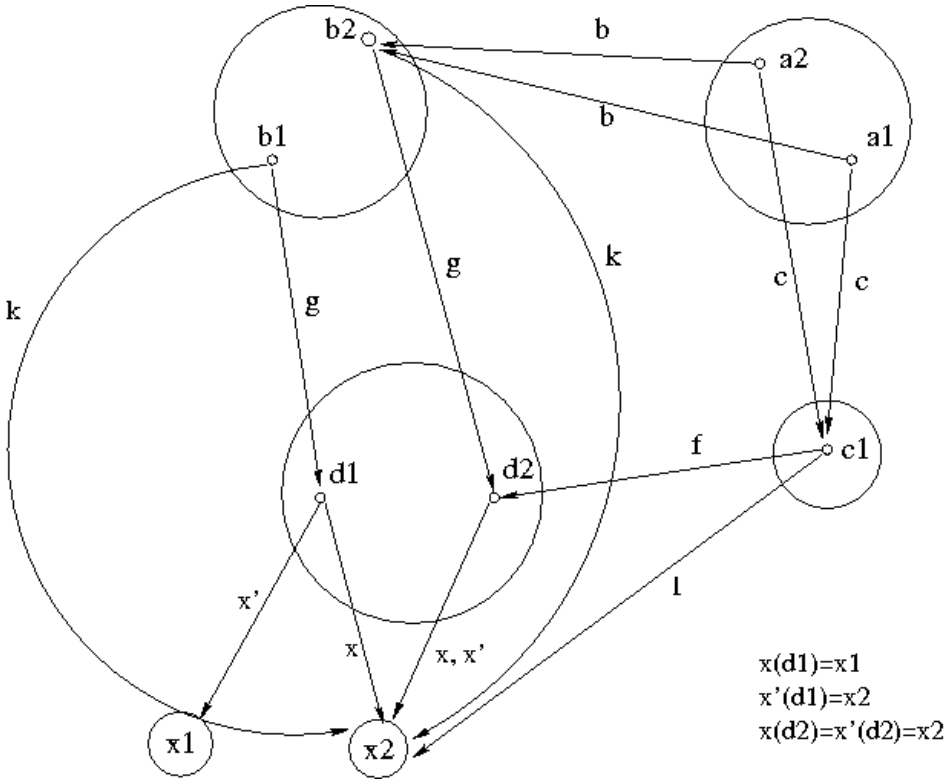
that D is not pushout by showing that universality property of pushouts is violated.

For that we construct another object X as shown in the following diagram, and construct two different morphisms $x, x' : D \to X$ such that $x \neq x'$.

In following diagram X satisfies the conditions that $k \circ b = l \circ c$ where k and l are morphisms $k : B \to X$ and $l : C \to X$

It is clear that x and x' satisfy conditions: $x \circ g = k$, $x' = k$, $x \circ f = l$, $x' \circ f = l$. But they differ as $x(de) = xe1 \neq xe2 = x'(de)$.

Second, if the identification condition is not satisfied. It implies that there are $b1$,$b2 \in B_V \cup B_E$ such that $b1 \neq b2$, $g(b1) = g(b2)$ and $b2 \notin b(A_V \cup A_E)$. Again prove our claim that D is not pushout by showing that universality property of pushouts is violated.



For that we construct another object X as shown in above diagram, and

construct two different morphisms $x, x' : D \to X$ such that $x \neq x'$.

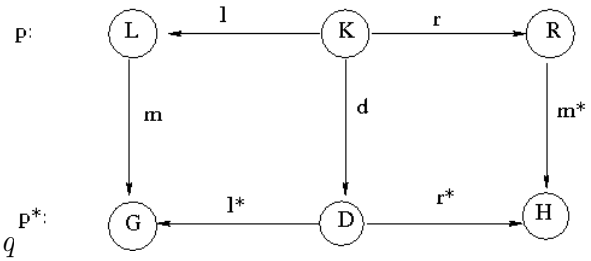
In above diagram object X satisfies the conditions that $k \circ b = l \circ c$ where k and l are morphisms $k : B \to X$ and $l : C \to X$. It is clear that x and x' satisfy conditions:$x \circ g = k$, $x' = k$, $x \circ f = l$, $x' \circ f = l$. But they differ as $x(d1) = x2 \neq x1 = x'(d1)$. □

We now give the definitions for direct derivation and sequential derivation. A sequential derivation is reflexive and transitive closure of a sequence of derivations.

Definition 2.1.7. (Direct derivation, Sequential derivation)

Direct Derivation:

Given a graph G, a graph production $p : (L \leftarrow K \rightarrow R)$, and a match $m : L \to G$, a direct derivation from G to H using p(based on m) exists if and only if following diagram can be constructed, where both squares are required to be pushouts in category $\mathcal{GRAPH}$.



Sequential Derivation:

A sequential derivation is either a graph G or a sequence of direct derivations $\rho = \{p_i : G_{i-1} \Rightarrow G_i\}_{i \in 1,..,n}$ such that p_i is a production of $\mathcal{G}$ for all $i \in \{1, ..., n\}$. In the first case sequential derivation is written as $G : G \Rightarrow^* G$ and in the latter case it is written as $\rho : G_0 \Rightarrow^* G_n$. If $\rho : G \Rightarrow^* H$ is a possible derivation, then graph G is called starting graph and G is called ending graph of derivation ρ. G is denoted by σ and H is denoted by τ. The length of sequential derivation ρ is is the number of direct derivations in ρ, if it is not identity and 0 otherwise. The sequential composition of two derivations ρ and ρ' is defined if and only if $\tau(\rho) = \sigma(\rho')$; in this case it is denoted by $\rho; \rho' : \sigma(\rho) \Rightarrow^* \tau(\rho')$.

2.2 Independence and Parallelism in the DPO approach

Parallel computations can be described in two different ways. In a sequential model, two parallel processes have to be modeled by interleaving arbitrarily their atomic actions, very similarly to the way multitasking is implemented on a single processor system. On the contrary, explicit parallelism means to have one processor per process, which allows actions to take place simultaneously.

Interleaving

In this approach two actions are concurrent i.e. potentially in parallel, if they may be performed in any order with the same result. In terms of graph transformations, the questions whether two actions(direct derivations) are concurrent or not can be asked from two different points of view.

Assume that a given graph represents a certain system state. The next evolution step of this state is obtained by the application of a production and the match are chosen non-deterministically from a set of possible alternatives. Clearly, each choice we make leads to a distinct derivation sequence, but the question remains, whether two of these sequences indeed model different computations or if we have only chosen one of two equivalent interleavings. In other words given two alternative direct derivations $H_1 \Leftarrow G \Rightarrow H_2$ where $p_1 : G \Rightarrow H_1$, $p_2 : G \Rightarrow H_2$ we ask ourselves if there are direct derivations $p_1 : H_2 \Rightarrow X$, $p_2 : H_1 \Rightarrow X$, showing that the two given direct derivations are not mutually exclusive, but each of them can instead be postponed after the application of the other, yielding the same result.

Given a derivation, intuitively two consecutive direct derivations are $G \Rightarrow H_1 \Rightarrow X$ where, $p_1 : G \Rightarrow H_1$, $p_2 : H_1 \Rightarrow X$ are concurrent if they can be performed in a different order, as in $G \Rightarrow H2 \Rightarrow X$ where, $p_2 : G \Rightarrow H_2$, $p_1 : H_2 \Rightarrow X$ without changing the result. The existence of two different orderings ensures that there is no causal dependency between these applications of p_1 and p_2.

We can say that two alternative derivations are concurrent if they are not mutually exclusive while two consecutive derivations are concurrent if they are not causally dependent.

Explicit Parallelism

Parallel productions can be represented more directly by parallel application of productions. Truly parallel applications of productions essentially requires to abstract from any possible application order, which implies that no intermediate graph is generated, i.e. a (true) pareallel production can be modeled by the application of single production, the parallel production.

Given productions $p_1 : L_1 \to R_1$ and $p_1 : L_1 \to R_1$, their *parallel production* is denoted by $p_1 + p_2 : L_1 + L_2 \to R_1 + R_2$. Intuitively $p_1 + p_2$ is just a disjoint union of p_1 and p_2. Direct derivations like $p_1 + p_2 : G \to X$ using parallel production $p_1 + p_2$ are called *parallel direct derivations*.

Now we give definitions of parallel independence and sequential independence along with examples.

Definition 2.2.1. (parallel independence)

Let $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m_2 : G \Rightarrow H_2$ be two direct derivations from the graph G as in figure 3.8. They are parallel independent if $m_1(L_1) \bigcap m_2(L_2) \subseteq m_1(l_1(K_1)) = m_2(l_2(K_2))$. This property can be formulated in categorical terms as follows:

There exists two graph morphisms $k_2 : L_1 \to D_2$ and $k_1 : L_2 \to D_1$ such that $l_2^* \circ k_2 = m_1$ and $l_1^* \circ k_1 = m_2$.

Two alternative direct derivations are parallel independent of each other, if each of them can still be applied after the other has been performed. This means that neither of $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m_2 : G \Rightarrow H_2$ delete elements of G which are also needed by the other direct derivation. In other words, the overlapping of two sides of p_1 and p_2 in G must be included in the intersection of the corresponding intersection graphs K_1 and K_2.

Following diagram shows categorical formulation of parallel independence.

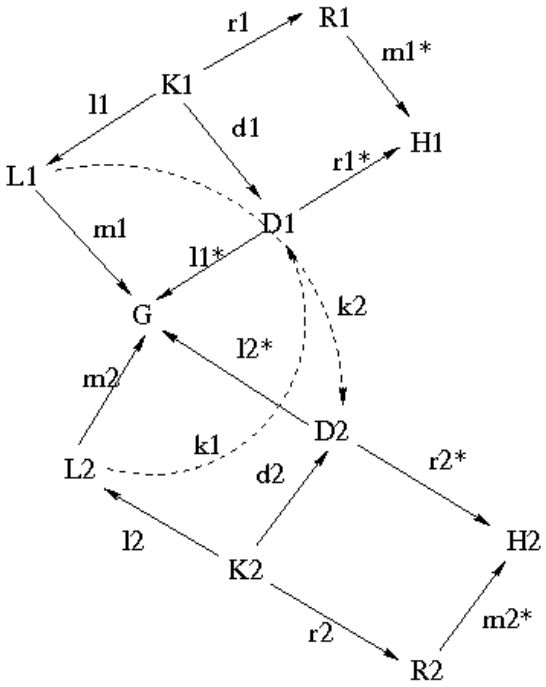


FIgure 3.8: Parallel independence of direct derivations

Definition 2.2.2. (sequential independence)

Given a two step derivation $p_1, m_1 : G \Rightarrow H_1$ *and* $p_2, m_2 : H_1 \Rightarrow H_2$ *as in figure 3.9. It is sequential independent iff* $m'_1(R_1) \bigcap m_2(L_2) \subseteq m_1^*(r_1(K_1)) = m_2(l_2(K_2))$. *This property can be formulated in categorical terms as follows:*

There exists two graph morphisms $k_2 : R_1 \to D_2$ *and* $k_1 : L_2 \to D_1$ *such that* $l_2^* \circ k_2 = m_1^*$ *and* $r_1^* \circ k_1 = m_2$.

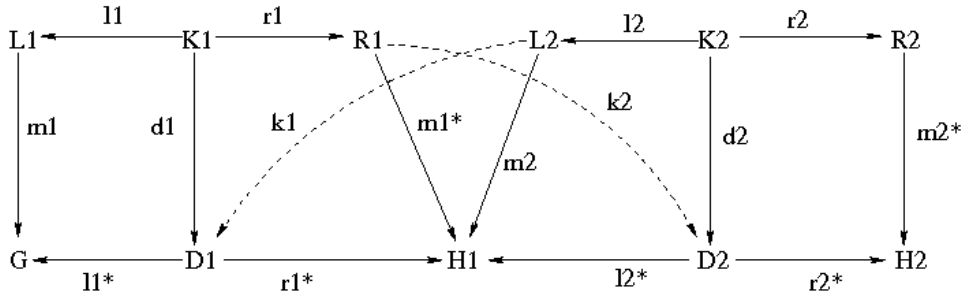


Figure 3.9 : sequential independent derivation

Two consecutive direct derivations $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m_2 : H_1 \Rightarrow X$ are sequentially independent if they may be swapped i.e., if p_2 can be applied to G and p_1 can be applied to the resulting graph. Therefore p_2 can at match m_2 can not delete anything that has been explicitly preserved by the application of p_1 at match m_1, and moreover it cannot use any element generated by p_1, which implies that the overlapping of R_1 and L_2 in H_1 must be included in the intersection of the interface graphs K_1 and K_2.

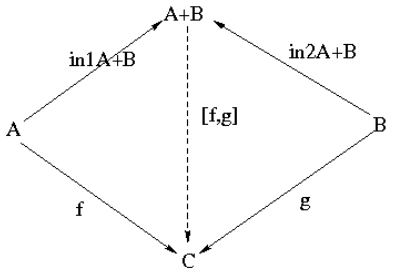
Definition 2.2.3. (coproducts)

Let $\mathcal{C}$ be a category and A, B be two objects of $\mathcal{C}$. A coproduct of $< A, B >$ is a triple $< A + B, in_1^{A+B}, in_2^{A+B} >$ where $A + B$ is an object and $in_1^{A+B} : A \to A + B$,$in_2^{A+B} : B \to A + B$ are arrows of $\mathcal{C}$, such that the following universal property holds:

for all pair of arrows $< f : A \to C, g : B \to C >$ there exists a unique arrow $[f, g] : A + B \to C$ such that $[f, g] \circ in_1^{A+B} = f$ and $[f, g] \circ in_2^{A+B} = g$.

In this case $A + B$ is called a coproduct object of $< A, B >$ and in_1^{A+B}, in_2^{A+B} are called injections; arrow $[f, g]$ is called copairing of f and g. One says that category $\mathcal{C}$ has coproducts if each pair of objects of $\mathcal{C}$ has a coproduct.

From the definition it is clear that the coproduct of two objects need not unique. Given any two coproducts of two objects they are not only isomorphic, but there exists *only one* isomorphism commuting with the injections.



Coproduct

Definition 2.2.4. (parallel productions, isomorphic parallel productions, parallel direct derivation)
Given a graph grammar $\mathcal{G}$,a parallel production (over $\mathcal{G}$)has the form $<(p_1, in_1), ...(p_k, in_k)>: (L \leftarrow K \rightarrow R)$, where $l : K \rightarrow L$, $K \rightarrow R$, where $k \geq 0$ $p_i : (L_i \leftarrow K_i \rightarrow R_i)$, where $l_i : K_i- > L_i$, $K_i- > R_i$ is a production of $\mathcal{G}$ for each $i \in \{1, ..., k\}$, L is a coproduct object of the graphs in $< L_1, ..., L_k >$ and similarly R and K are coproduct objects of the $< R_1, ..., R_k >$ and $< K_1, ..., K_k >$ respectively. Moreover, l and r are uniquely determined by the families of arrows $\{l_i\}_{i \leq k}$ and $\{r_i\}_{i \leq k}$, respectively. Finally, for each $i \in \{1, ..., k\}$, in^i denotes the triple of injections $< in^i_L : L_i \rightarrow L, in^i_K : K_i \rightarrow K, in^i_R : R_i \rightarrow R >$. All the parallel productions are recorded in the name of a parallel production.

A parallel production like above is proper if $k > 1$; the empty production is the only parallel production with $k = 0$, having the empty graph ϕ as left and right hand sides and as the interface. Each production $p : (L \leftarrow K \rightarrow R)$, where $l : K \rightarrow L, K \rightarrow R$ of $\mathcal{G}$ is identified with the parallel production $< (p, < id_L, id_K, id_R >) >: (L \leftarrow K \rightarrow R)$, where $l : K \rightarrow L, r : K \rightarrow R$.

Two parallel productions $\mathcal{G}$, $p =< (p_1, in_1), ...(p_k, in_k) >: (L \leftarrow K \rightarrow R)$, where $l : K \rightarrow L$, $r : K \rightarrow R$, and $q =< (q_1, \underline{i}n_1), ...(p_k, \underline{i}n_k) >: (L' \leftarrow K' \rightarrow R')$, where $l' : K' \rightarrow L'$, $r' : K' \rightarrow R'$, are isomorphic via π if $k = k'$ and π is a permutation of {1,...,k} such that and $p_i = q_{\pi_{(i)}}$ for each $i \in \{1, ..., k\}$; that is the component productions of $\mathcal{G}$ are the same, up to a permutation. We say that p and q are isomorphic if there is a π such that they are isomorphic via π.

The graph grammar with parallel productions $\mathcal{G}^+$ generated by grammar $\mathcal{G}$ has the same start graph of $\mathcal{G}$, and as productions all the parallel

productions over $\mathcal{G}$. A parallel direct derivation over $\mathcal{G}$ is a direct derivation over $\mathcal{G}^+$; it is proper or empty if the applied parallel production is proper or empty, respectively. A A parallel direct derivation over $\mathcal{G}$ is a sequential derivation over $\mathcal{G}^+$.

Definition 2.2.5. (span-isomorphic productions)

Two productions $p : (L \leftarrow K \rightarrow R)$ and $p' : (L' \leftarrow K' \rightarrow R')$ are span isomorphic if there are three morphisms m_L:$L \rightarrow L'$, m_R:$R \rightarrow R'$, m_K:$K \rightarrow K'$, such that they make the resulting square commutative.

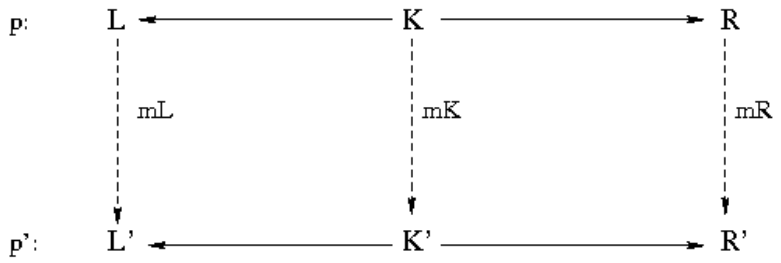


Figure : span–isomorphic productions

Proposition 2.2.6. (isomorphic parallel productions are span-isomorphic)

Let $p =< (p_1, in_1), ...(p_k, in_k) >: (L \leftarrow K \rightarrow R)$, where $l : K \rightarrow L$, $r : K \rightarrow R$, and $q =< (q_1, \underline{in}_1), ...(p_k, \underline{in}_k) >: (L' \leftarrow K' \rightarrow R')$, where $l' : K' \rightarrow L'$, $r' : K' \rightarrow R'$, be two parallel productions isomorphic via π. Then they are span-isomorphic.

Proof. By definition, we have for each $X \in \{L, K, R\}$, $< X, in_X^1, ..., in_X^k >$ and $< X', \underline{in}_X^1, ..., \underline{in}_X^k >$ are two coproducts of the same objects, and therefore there is a unique isomorphism $\phi_X : X \rightarrow X'$ which commutes with the injections. It is easy to see that these isomorphisms satisfy $\phi_L \circ l = l' \circ \phi_K$ and $\phi_R \circ r = r' \circ \phi_K$ □

Proposition 2.2.7. (applicability of parallel productions)
Let $q =< (p_1, in_1), ...(p_k, in_k) >: (L \leftarrow K \rightarrow R)$, *where* $l : K \rightarrow L$, $r : K \rightarrow R$, *and let* $m : L \rightarrow G$ *be a match for it. For each* $i \in \{1, ..., k\}$, *let* $m_i : L_i \rightarrow G$ *be the match of the i'th production in G induced by m, defined as* $m_i = m \circ in_L^i$. *Then* $< l, m >$ *satisfies the gluing condition , i.e., there is a parallel direct derivation* $q, m : G \Rightarrow H$, *iff for all* $i \in \{1, ..., k\}$ $< l_i, m_i >$ *satisfies the gluing condition, i.e.,* p_i *can be applied at match* m_i, *say with result* H_i, *and for each* $1 \leq i < j \leq k$, *direct derivations* $p_i, m_i : G \Rightarrow H_i$ *and* $p_j, m_j : G \Rightarrow H_j$ *are parallel independent.*

Proof. The parallel independence of the matches of the composing productions follows from the gluing conditions of the parallel direct derivations, and vice versa.

□

Lemma 2.2.8. (Butterfly lemma)
Let a_1 *and* a_2 *be injective graph morphisms. Then the square* (1) *of diagram*1 *is a pushout if and only if there are graphs* X_1, X_2 *and graph morphisms r, s, v, w, t and u, such that in the right part of the diagram*2 (2), (3), (4) *are pushouts,* (4) *is a pullback, and all triangles commute.*

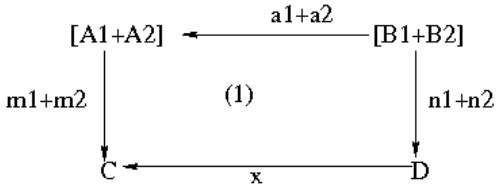


Diagram 1

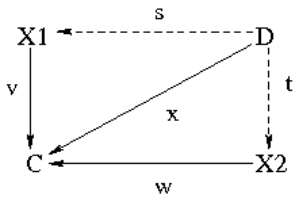


Diagram 3

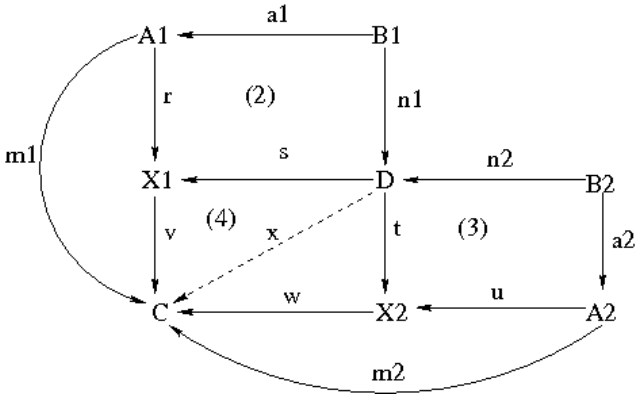


Diagram 2

Proof. ($\Leftarrow$)

Assuming that we have diagram2 , where (2), (3), (4) are pushouts, (4) is a pullback, and all triangles commute, and also that a_1 and a_2 be injective graph morphism. It is easy to see that Diagram 1 is a pushout.

($\Rightarrow$)Assume that we have diagram1 and we have to prove that (2), (3), (4) are pushouts, (4) is a pullback, and all triangles commute in diagram 2.

First we construct pushout (2), graph X_1 is constructed as the disjoint union of elements of $A1$ and D, in the usual way, in fact way X_1 is a part of graph C given in diagram 1. Here we use X's morphisms a_1 and n_1 only. morphism r maps any element a of $A1$ to equivalence class (a, d) where d an element of D is mapped under n_1 from the same element b of B_1, or to equivalence class (a) if it is not mapped under a_1. In either case it is mapped to same element of C. Similarly s is constructed.

Second we construct pushout (3) in similar fashion only difference is that elements of X_2 are made up of equivalences classes of A_2 and D only.
Since a_1 and a_2 injective (2) and (3) are unique.

Third we construct pushout (4) in the following way. We know that v and w are injective mappings from construction of (2) and (3). The fact that $v \circ s = t \circ w$ follows from the fact that $v \circ s = x = t \circ w$ i.e the triangles commute. Since s and t are injective (4) is unique.

Fourth we have to prove that (4) is a pullback i.e. we have to construct D and morphisms s and t given X_1, X_2, and morphisms v and w. as shown in the diagram3.
s exists and is unique follows the fact that x, v are injective morphisms by construction of pushout (4) since m_1, r and m_2, u are injective morphisms. Morphism t exists and is unique for the same reasons. □

Lemma 2.2.9. (analysis of a parallel direct derivation)

Let $\rho = (q : G \Rightarrow H)$ be a parallel direct derivation using the parallel production $q = p_1 + ... + p_k : (l : K \to L, r : K \to R)$. Then for each ordered partition $< I =< i_1, ..., i_n >, J =< j_1, ..j_m >>$ of $\{1, ..., k\}$ (i.e., $I \bigcup J = \{1, .., k\}$ and $I \bigcap J = \phi$) there is a constructive way to obtain a sequential independent derivation $\rho' = (q' : G \Rightarrow X, q'' : X \Rightarrow H)$, called

an analysis of ρ, where $q' = p_{i_1} + ... + p_{i_n}$, and $q' = p_{j_1} + ... + p_{j_m}$. Such a construction is in general not deterministic. if ρ and ρ' are as above , we shall write $\rho' \in ANAL_{I,J}(\rho)$ or $< \rho, \rho' > \in ANAL_{I,J}$.

Proof. If I (or J) is empty the statement follows by taking $X = G(X = H), q'' = q(q = q')$, and an empty direct derivation as $q'(q'')$. Now let us assume that q is a proper parallel production, and that I and J are not empty. By the hypothesis on I and J, productions $q' + q''$ are clearly isomorphic via the permutation π defined as $\pi(x) = i_x$ if $1 \leq x \leq n$ and $\pi(x) = j_{x-n}$ if $n < x \leq n + m$. Thus since the left diagram of figure is a double pushout by the hypothesis, the right diagram is a double-pushout as well, where $(q' :: l_1 : K_1 \to L_1, r_1 : K_1 \to R_1)$ and $(q'' :: l_2 : K_2 \to L_2, r_2 : K_2 \to R_2)$.

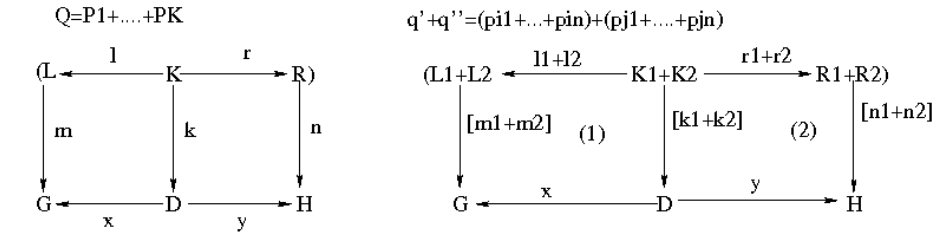


Figure 1: Splitting a parallel Derivation

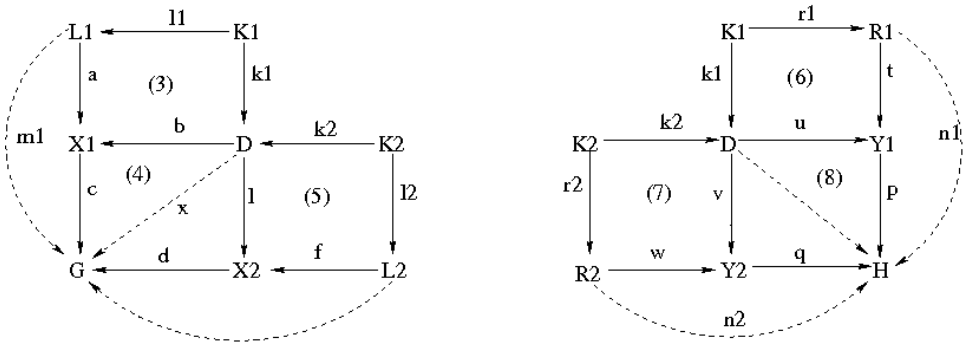


Figure 2: Application of Butterfly Lemma

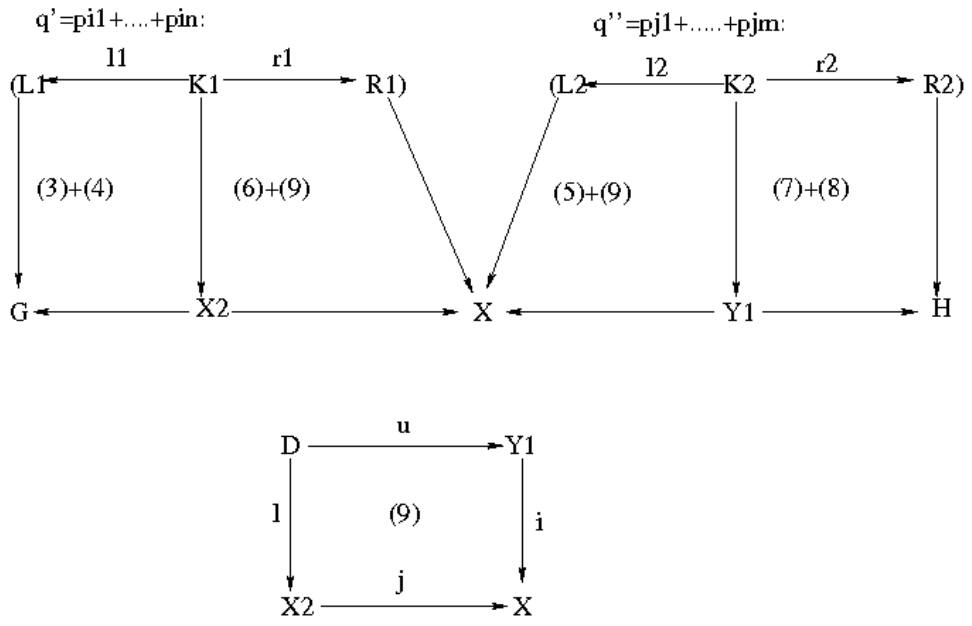


Figure 3: Building the sequential derivation

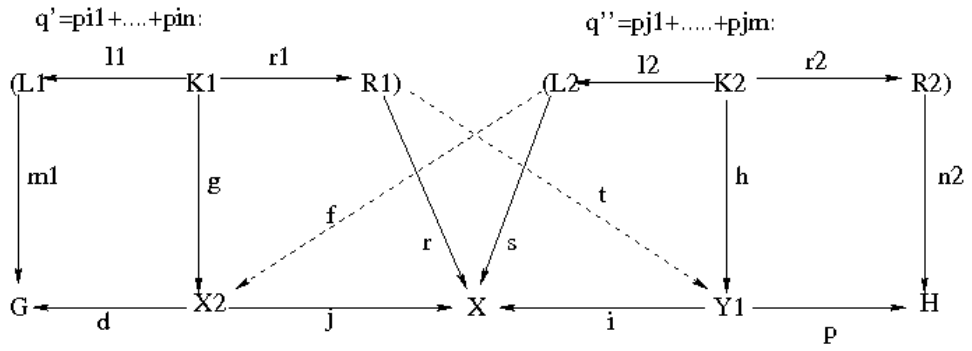


Figure 4: A sequential independent derivation

Therefore we can apply **Butterfly lemma** to (1) and (2), obtaining the

pushouts $(3) - (5)$ and $(6) - (8)$, respectively.

Now let (9) be the pushout of arrows l and u , where the pushout object is X. Then we can build the required derivation via q' and q'' as in the right part of the diagram, by using the fact that a square consisting of two adjacent squares is also a pushout. Such a derivation is sequential because arrows $f : L_2 \to X_2$ and $t : R_1 \to Y_1$ satisfy the conditions of definition of sequential independence.

□

Lemma 2.2.10. (synthesis of sequential independent derivations) *Let $\rho = (G \Rightarrow X \Rightarrow H)$, be a sequential independent derivation. where $q' : G \Rightarrow X$ and $q'' : X \Rightarrow H$. Then there is a constructive way to obtain a parallel derivation $\rho' = (q' + q'' : G \Rightarrow H)$ called a synthesis of ρ. Also this construction is in general not deterministic. if ρ and ρ' are as above , we shall write $\rho' \in SYNT(\rho)$.*

Proof. Let ρ be the sequential independent derivation of figure (4). We have to show that the commutative diagrams of figure (2) can be constructed, where squares from (3) to (8) have to be pushouts , and (4) and (8) need to be pullbacks. Then by two applications of butterfly lemma we obtain the parallel production in the right part of figure (1), as required.

The diagrams of figure (2) can be constructed as follows:

- square 5:

  $< D, k_2, l >$ is taken as a pushout complement of $< l_2, f >$. It is determined up to isomorphism because l_2 is injective. Thus sqaure 5 is pushout by construction. Morphism l is injective because l_2 is injective.

- Determining morphism $k_1 : K_1 \to D$

  k_1 is determined by showing that $g : K_1 \to X_2$ in figure (4) factorizes through morphism $l : D \to X_2$. i.e., there exists a morphism k_1 such that $l \circ k_1 = g$; then k_1 is unique since l is injective.

  We have to show that the image of K_1 in X_2 i.e. $g(K_1))$ is contained in the image of D in X_2 i.e.in $l(D)$. Since X_2 is the pushout object of $< l_2, k_2 >$, this is false only if $g(K_1)$ contains an item $z \in f(L_2)$ such that $z \notin f(l_2(K_2))$. This means that z is not in image of l then it is pushed by f alone from L_2 and which is not mapped by any element via

morphism l_2. Also $z \notin f(l_2(K_2))$ implies that $z \notin f(l(k_2))$, by property of pushout.

But since X is the pushout object of $< l_2, h >$ and $j \circ f = s$, this means that $j(z)$ is in $s(L_2)$ but not in $i(Y_1)$, which is absurd by the existence of a morphism $t : R_1 \to Y_1$ such that $i \circ t = r$.

- $< X_1, a, b >$ is taken as a pushout of l_1, k_1. Thus (3) is pushout by construction,and b is injective because so is l_1.

- Morphism x is defined as $x = d \circ l$.

- Morphism c is uniquely determined by condition $m_1 \circ l_1 = d \circ g = d \circ l \circ k_1 = x \circ k_1$, because (3) is a pushout. If not then let us assume that there is another morphism $c' : X_1 \to G$. It implies that $c(x_1) = g_1 \neq g_2 = c'(x_1)$, for $x_1 \in X_1$, and $g_1, g_2 \in G$, and $k \in K_1$
  $c(b(k_1(k))) = g_1$ and $c^1(b(k_1(k))) = g_2$ which implies
  $x(k_1(k)) = g_1$ and $x(k_1(k)) = g_2$
  which is a contradiction.

- square (4) is a pushout

  square (4) is a pushout by decomposition property of pushouts since (3) and (3) + (4) are pushouts. Also c is injective because l is injective.

- square (4) is a pullback

  since any pushout made of injective morphisms is a pullback in category $\mathcal{G}RAPH$

- Determining morphism u and it's uniqueness. Assuming this square (9) commutes which implies that u is injective since l,j,i are injective. Also it is clear that square (9) is a pushout because (5) and (5) + (9) are pushouts.

  To show this we have to show that $j \circ l$ factorizes through i. We have to show that the image of D in X i.e. $j(l(D))$ is contained in the image of Y_1 in X i.e.in $i(Y_1)$. If not then since X is the pushout object of $< l_2, h >$, there is an item $z \in j(l(z''))$ such that $z \in s(L_2)$ and $z \notin s(l_2(K_2))$. Let $z' \in L_2$ be such that $s(z') = z$ thus $z' \notin (l_2(K_2))$, then we have $j(f(z')) = s(z') = z = j(l(z''))$ which implies $f(z') = l(z')$ by injectivity of j. But since X_2 is the pushout object of $< l_1, k_2 >$, this implies that $z' \in l_2(K_2)$, which is a contradiction.

- square (7) is a pushout.

  $< Y_2, w, v >$ is taken as a pushout of $< k_2, r_2 >$. therefore (7) is a pushout and v is injective because r_2 is injective.

- Morphism y is defined as $y = p \circ u$.

- Morphism q is uniquely determined by condition $n_2 \circ r_2 = p \circ h = p \circ u \circ k_2 = y \circ k_2$, because (7) is a pushout.

- square (8) is a pushout. since (7) and (7) + (8) are pushouts. Since u is injective q is injective. Square (8) is injective because all it's morphisms are injective.

- Finally by pushout-pullback decomposition property which holds for category $\mathcal{GRAPH}$ with injective morphisms , square (8) is a pullback since (6) + (8) is a pushout, and morphisms r_1, u, v, q , p are injective then (6) is a pushout.

□

Theorem 2.2.11. (Parallelism)
Given (possibly) parallel productions $p_1 : L_1 \leftarrow K_1 \rightarrow R_1$ *where* $l_1 : K_1 \rightarrow L_1$,$r_1 : K_1 \rightarrow R_1$, *and* $p2 : L2 \leftarrow K_2 \rightarrow R2$, *where* $l_2 : K_2 \rightarrow L_2$, $l_2 : K_2 \rightarrow R_2$,
the following statements are equivalent:

1. *There are parallel direct derivations* $p_1 + p_2, m : G \Rightarrow X$

2. *There is a sequentially independent derivation* $G \Rightarrow H_1 \Rightarrow X$, *where* $p_1, m_1 : G \Rightarrow H_1$ *and* $p_2, m_2' : H_1 \Rightarrow X$.

Proof. $(1 \Rightarrow 2)$ follows from constructive proof given by lemma **(analysis of a parallel direct derivation)** .

$(2 \Rightarrow 1)$ follows from constructive proof given by lemma **(synthesis of sequential independent derivations)** .

□

Theorem 2.2.12. (Local Church Rosser)

1. *Let $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m_2 : G \Rightarrow H_2$ be two parallel direct derivations,as in Figure 3.8, and let $m'_2 = r_1^* \circ k_1$, where arrow $k_1 : L_2 \rightarrow D_1$ exists by parallel independence. Then morphisms $< l2, m2' >$ satisfy the gluing condition, and thus production p_2 can be applied to match m'_2. Moreover, derivation $G \Rightarrow H_1 \Rightarrow X$ is sequentially independent derivation , where $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m'_2 : H_1 \Rightarrow X$ is sequentially independent derivation.*

2. *Let $G \Rightarrow H_1 \Rightarrow H_2$ is a sequentially independent derivation , where $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m'_2 : H_1 \Rightarrow H_2$ as in Figure 3.9, and let $m'_2 = l_1^* \circ k_1$, where arrow $k_1 : L_2 \rightarrow D_1$ exists by sequential independence. Then morphisms $< l2, m2' >$ satisfy the gluing condition, and thus production p_2 can be applied to match m'_2. Moreover, direct derivations $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m_2 : G \Rightarrow Y$ are parallel independent.*

Proof. Let $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m_2 : G \Rightarrow Y$ be two parallel independent direct derivations
iff
(by **proposition 2.2.6**) there is a graph such that $p_1 + p_2, [m_1, m_2] : G \Rightarrow H$ is a parallel direct derivation,
iff
(by **Parallelism theorem)** ,there is a sequentially independent derivation $G \Rightarrow H_1 \Rightarrow X$ is sequentially independent derivation , where $p_1, m_1 : G \Rightarrow H_1$ and $p_2, m'_2 : H_1 \Rightarrow X$. □

2.3 Models of computation in the DPO Approach

Our aim is to study the operational behaviour of a grammar, i.e., to study formal frameworks where derivations can be analyzed and described, since derivations of grammar are intended to model computations of the system modeled by grammar.

A *model of computation* for a grammar is a mathematical structure which contains the relevant information concerning the potential behaviour of grammar, ,i.e. all about the possible derivations of the grammar. Since the basic operations on derivation is concatenation *categories* are very suitable to use as mathematical structure. A model of computation for a given

grammar is just a category of graphs as objects, and where every arrow is a derivation starting from the source graph and ending at the target graph. Here categorical operation of sequential composition corresponds to the concatenation of derivations.

A concrete model of computation for a grammar has all concrete graphs as objects and all derivations as arrows. But this model contains lot of redundant information. So we need abstract models where derivations and graphs differing only for insignificant aspects are identified. Such abstract models are defined by imposing an equivalence relation on derivations and graphs. Such relation equates those derivations and graphs which are indistinguishable by the chosen observation criterion. This equivalence should be a congruence wrt sequential composition. We get the abstract model corresponding to chosen observation criterion by taking the quotient category of concrete model with respect to the equivalence relation.

We study concrete model for a grammar in the DPO approach, two equivalence relations and their corresponding models. We refine them to obtain a third abstract model: the abstract, truly concurrent model, which satisfactorily composes the two equivalences.

2.3.1 Concrete model of computation in the DPO Approach

(We assume that a derivation is a possibly parallel derivation.)

Definition 2.3.1. The concrete model of computation

Given a graph grammar $\mathcal{G}$, the concrete model of computation for $\mathcal{G}$, denoted $\mathbf{Der}(\mathcal{G})$*, is the category having all graphs as objects and where $\rho : G \to H$ is an arrow iff ρ is a (parallel) derivation, $\sigma(\rho) = G$ and $\tau(\rho) = H$. Arrow composition is defined as the sequential composition of derivations, and the identity of object G is the identity derivation $G : G \Rightarrow^* G$.*

There is a drawback of concrete model.Consider any two step derivation $G_1 \Rightarrow G_2 \Rightarrow G_3$, where $p : G_1 \Rightarrow G_2$ and $q : G_2 \Rightarrow G_3$ are the two productions applied. Here between G_1 and G_3 we have one arrow for the two step derivation but, infinitely many isomorphic arrows, one for each isomorphic

copy of that derivation, which is obtained by changing arbitrarily the identity of nodes and edges in all the graphs of derivation, except the starting and ending ones.

2.3.2 Truly-concurrent model of computation in the DPO Approach

According to the parallelism theorem , two productions can be applied at parallel independent matches in a graph either at the same time, or one after the in any order, producing the same resulting graph. An observation mechanism which does not distinguish between two derivations where independent rewriting steps are performed in different order is considered to observe the *concurrent* behaviour of a grammar, instead of a sequential one. The corresponding equivalence is said to capture *true concurrency*.

Shift equivalence:

This equivalence is based on analysis and synthesis constructions. It equates two derivations if they are related by a finite number of applications of analysis and synthesis. The $shift$ refers to the basic operation of $shifting$ a production one step towards the left, if it is sequential independent from the previous direct derivation.

Definition 2.3.2. (shift equivalence)
If ρ and ρ' are two derivations such that $\rho' \in ANAL_{I,J}(\rho)$
$\rho_1 = \rho_2; \rho; \rho_3$
$\rho'_1 = \rho_2; \rho'; \rho_3$
and ρ_2 has length $n-1$ then we will write $\rho'_1 \in ANAL^n_{I,J}(\rho_1)$, indicating that ρ'_1 is an analysis of ρ_1 at step n.

Similarly $\rho' \in SYNT(\rho)$,
$\rho_1 = \rho_2; \rho; \rho_3$
$\rho'_1 = \rho_2; \rho'; \rho_3$
and ρ_2 has length $n-1$ then we will write $\rho'_1 \in SYNT^n(\rho_1)$, indicating that ρ'_1 is obtained from ρ_1 at step n via synthesis construction.

Now, let $ANAL$ be the union of all analysis relations, (i.e., $ANAL= \cup\{ANAL^n_{I,J} | n \in \aleph \wedge I, J \in \aleph^*\}$) and similarly let be $SYNT$ be the union of all synthesis relations, (i.e., $SYNT= \cup\{SYNT^n | n \in \aleph\}$). Furthermore,

let $SHIFT_0^0$ *be the relation defined as* $< \rho_1, \rho_2 > \in SHIFT_0^0$ *iff* ρ_1 *is an identity derivation* $\rho_1 : G \Rightarrow^* G$ *and* ρ^2 *is an empty derivation* $\rho_2 : G \Rightarrow^\phi G$ *such that the induced isomorphism is the identity* $id_G : G \Rightarrow^\phi G$.

The smallest equivalence relation on derivation on derivations containing $ANAL \cup SYNT \cup SHIFT_0^0$ *is called* $shift$ *equivalence and it is denoted by* $\equiv_{sh}$. *If* ρ *is a derivation, by* $[\rho]_{sh}$ *we denote the equivalence class containing all derivations shift-equivalent to* ρ.

From definition it is clear that $\rho \equiv_{sh} \rho'$ implies $\sigma(\rho) = \sigma(\rho')$ and $\tau(\rho) = \tau(\rho')$, because synthesis and analysis do not affect the extremes of a derivation.

The truly-concurrent model of computation for a graph grammar is, a category having concrete graphs as the objects; however it's arrows are equivalence classes of derivation modulo the shift equivalence.

Definition 2.3.3. (truly-concurrent model of computation)
Given a graph grammar $\mathcal{G}$, *the* $truly-concurrent$ *model of computation for* $\mathcal{G}$, *denoted* $\mathbf{Der}(\mathcal{G})/_{\mathbf{sh}}$, *is the category having graphs as objects and as arrows equivalences of derivations wrt the shift equivalence. More precisely*
$[\rho]_{sh} : G \to H$ *is an arrow of* $\mathbf{Der}(\mathcal{G})/_{\mathbf{sh}}$ *iff* $\sigma(\rho) = G$ *and* $\tau(\rho) = H$. *The arrow composition is defined as* $[\rho]_{sh}; [\rho]'_{sh} = [\rho; \rho']_{sh}$, *and the identity of* G *is the equivalence class* $[G]_{sh}$ *containing identity derivation* G.

Canonical derivations

In truly-concurrent model arrows are equivalences classes of derivations, so the natural question is to ask for some *standard representive* i.e., a derivation which can be characterised as the only one in the equivalence class which enjoys a certain property.

Canonical derivations are partial answer to above problem. Any given derivation can be converted to a $shift - equivalent$ canonical derivation where each production is applied as early as possible.

Definition 2.3.4. (shift relation, canonical derivation) *Let us write* $\rho' \in ANAL_i^n(\rho)$ *if* $\rho' \in ANAL_{I,J}^n(\rho)$, *where* $I =< i >$ *and* $J =< 1.., i-1, i+1, .., k >$; *i.e., when* ρ' *is obtained from derivation* ρ *by anticipating the application of the* $i'th$ *production of the* $n'th$ *direct derivation.*

Now for each pair of natural numbers $< n, j >$ *both greater than* 0, *the relation* $SHIFT_j^n$ *is defined as* $SHIFT_j^n = SYNT^{n-1} \circ ANAL_j^n$.

Moreover, for each $n > 0$, we write $< \rho_1, \rho_2 > \in SHIFT_0^n$ iff ρ_1 has length n, $\rho_1 = \rho; G \Rightarrow H$,where $q : G \Rightarrow H$ $\rho_2 = \rho; G \Rightarrow X \Rightarrow H$, where $q : G \Rightarrow X$ and $\phi : X \Rightarrow H$, an the double pushout with the isomorphism from X to H induced by the empty direct derivation $\phi : X \Rightarrow H$. Finally let $SHIFT_0^0$ be as defined earlier in definition of $shift$ equivalence.

Relation $<_{sh}$ is defined as the union of relations $SHIFT_j^n$ for each $n, j \in \aleph$. The transitive and reflexive closure of $<_{sh}$, denoted $\leq_{sh}$, is called the $shift$ relation. A derivation is called canonical iff it is minimal wrt $\leq_{sh}$.

$\rho_1 <_{sh} \rho_2$ iff ρ_1 is obtained from ρ_2 either by removing the last direct derivation if it is empty or by moving the application of a single production one step towards left.

It is important to note that for each $n \in \aleph$, relation $SHIFT_0^n$ is deterministic, in the sense that $< \rho_1, \rho_2 >, < \rho_1', \rho_2 > \in SHIFT_0^n$ implies $\rho_1 = \rho_1'$.

Proposition 2.3.5. (properties of the shift relation)

1. *The smallest equivalence relation containing the $<_{sh}$ relation is exactly relation $\equiv_{sh}$ introduced in definition of* **shift equivalence**.

2. *Relation $<_{sh}$ is well-founded, i.e., there is no infinite chain $\rho_1, \rho_2, ...$ of derivations such that $\rho_{i+1} <_{sh} \rho_i$ for all $i \in \aleph$. Thus for each derivation ρ there exists a canonical (i.e., minimal wrt $\leq_{sh}$) derivation ρ' such that $\rho' \leq_{sh} \rho$.*

The drawbacks of truly-concurrent model are

1. It still contains too many arrows.Consider any two step derivation $G_1 \Rightarrow G_2 \Rightarrow G_3$, where $p : G_1 \Rightarrow G_2$ and $q : G_2 \Rightarrow G_3$ are the two productions applied. However ,since the objects of the category are graphs , looking at the arrows starting from G_1 one may still find the infinitely many arrows corresponding to the application of same productions p and q, but ending at different isomorphic copies of G_3.

   Also consider the pushout of two arrows , which is unique upto isomorphism only, which implies that we can have application of same production yielding infinitely many results.

2. canonical derivations are not unique, since in general analysis and synthesis constructions are not deterministic.

2.3.3 Requirements for capturing representation independence

Graph grammars are mostly used as a formalism for specifying the evolution of certain systems. In general two isomorphic graphs are considered as representing the same state. In fact, such a state is determined by the topological structures of graph and by labels only, while the true identity of nodes and edges is considered as representation detail only.

Therefore we need an equivalence which equates all isomorphic graphs. Thus we reason in terms of *abstract graphs* i.e., of isomorphic classes of concrete graphs: Given a production $p : G \Rightarrow H$ applied at match m there is only one abstract graph $[H]$ such that $p, m : [G] \Rightarrow [H]$.

Above solution is sufficient if we need to consider only the set of graphs generated by the grammar. But for the semantics which associates with grammar all possible derivations still we have too much information dependent information. For example we derive $[H]$ from $[G]$ by application of production p at match m, this can be done via many distinct context graphs D in the double-pushout construction. Thus the problem is that we find too many distinct derivations which should not be considered as distinct system evolutions.

Therefore we need a suitable equivalence class of derivations also i.e., *abstract derivations*. The $shift$ *equivalence* is not representation independent because it relates only those derivations which are starting and ending at the same concrete graphs.

Three requirements for capturing representation independence and also to capture the semantics of computation are as follows:

1. well defined equivalence
2. equivalence allowing for sequential composition
3. uniqueness of canonical derivations

we use $\sim$ to denote an equivalence relation on derivations, and $[\rho]$ to denote an abstract derivation , which is an equivalence class of derivations modulo $\sim$.

The notion of abstract derivation should be consistent with notion of abstract graph, i.e., any abstract derivation should start from an abstract graph and should end in an abstract graph.

Definition 2.3.6. (well defined equivalence)
For each abstract derivation $[\rho]$*, define* $\sigma([\rho]) = \{\sigma(\rho')|\rho' \in [\rho]\}$*,and similarly* $\tau([\rho]) = \{\tau(\rho')|\rho' \in [\rho]\}$*. Then the equivalence* $\sim$ *is well defined if for each derivation* $\rho : G \Rightarrow^* H$ *we have that* $\sigma([\rho] = [\sigma(\rho)]$*, and* $\tau([\rho] = [\tau(\rho)]$*.*

If $\sim$ is well defined then for each derivation $\rho : G \Rightarrow^* H$ we can write $[\rho] : [G] \Rightarrow^* [H]$.

The second requirement is that the equivalence is a congruence with respect to sequential composition.

Definition 2.3.7. (equivalence allowing for sequential composition)
Given two abstract derivations $[\rho]$ *and* $[\rho']$ *such that* $\tau([\rho] = [\sigma(\rho')]$*, their sequential composition* $[\rho];[\rho']$ *is defined iff foe all* $[\rho_1], [\rho_2] \in [\rho]$ *and* $[\rho'_1], [\rho'_2] \in [\rho']$ *such that* $\tau([\rho_1] = [\sigma(\rho'_1)]$ *and* $\tau([\rho_2] = [\sigma(\rho'_2)]$*, one has that*
$\rho_1; \rho'_1 \sim \rho_2; \rho'_2$*. In this case*
$[\rho]; [\rho']$ *is by definition the abstract derivation* $[\rho; \rho'_1]$*.*
An equivalence $\sim$ *for sequential composition iff* $[\rho]; [\rho']$ *is defined for all* $[\rho]$ *and* $[\rho']$ *such that* $\tau([\rho] = \sigma([\rho'])$*.*

Each abstract derivation has a unique abstract canonical derivation.

Definition 2.3.8. (uniqueness of canonical derivations)
Equivalence $\sim$ *enjoys uniqueness of canonical derivations iff for each pair of equivalent derivations* $\rho \sim \rho'$ *and for each pair* $< \rho_c, \rho'_c >$ *of canonical derivations,* $\rho \equiv_{sh} \rho_c$ *and* $\rho' \equiv_{sh} \rho'_c$ *implies that* $\rho_c \sim \rho'_c$*.*

2.3.4 Equivalence $\equiv_0$

Let $\rho : G_0 \Rightarrow^* G_n$ and $\rho' : G'_0 \Rightarrow^* G'_n$ be two derivations having the same length as shown in the following figure. They are isomorphic (written $\rho \equiv_0 \rho'$) if there exists a family of isomorphisms $\{\phi_{G_0} : G_0 \to G'_0, \phi_{X_i} : X_i \to X'_i\}$ with $X \in \{L, K, R, G, D\}$ and $i \in \{1, ..., n\}$ between corresponding graphs of the two derivations, such that all those squares commute, which are obtained by composing two corresponding morphisms $X \to Y$ and $X' \to Y'$ of the two derivations and the morphisms ϕ_X and ϕ_Y relating the source and the target graphs.

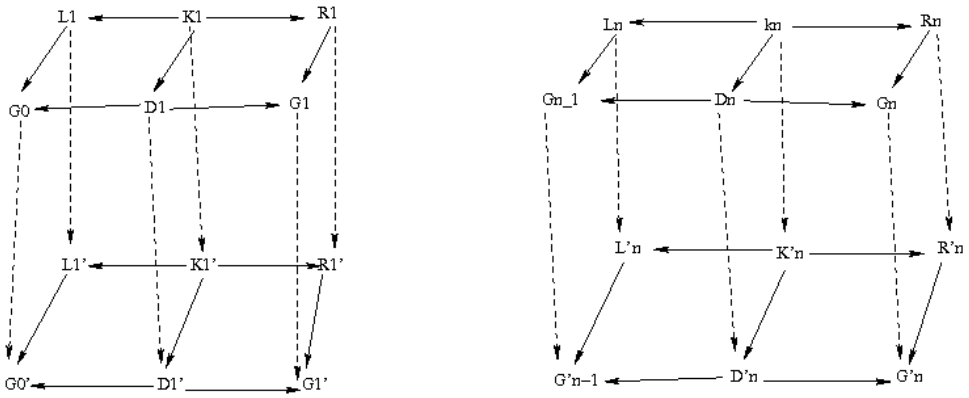


figure : Isomorphisms of derivations

The drawback of $\equiv_0$ is that equivalence induced by the isomorphisms of derivations identifies too many derivations. The productions in the corresponding direct derivations need to be only span-isomorphic. As a result derivations which can hardly be considering as differing in the representation detail may be identified.

For example, if a grammar $\mathcal{G}$ has two span-isomorphic productions p and p', two direct derivations using p and p' would be considered as equivalent. To give a more specific example, consider two parallel productions $q = p_1 + ... + p_k$ and $q' = p'_1 + ... + p'_k$, may happen to be span-isomorphic even if some of the involved sequential productions are distinct or if $k \neq k'$.

A more specific example is as follows: Following figure shows grammar production rules for the client-server system. In figure S denotes server, C denotes client. The various states they could be in are denoted by edges. A loop labeled *idle* on server indicates that server is ready to accept requests from client. A loop labeled *req* on client indicates that client issued a request, while a loop labeled *job* indicates client is doing some internal activity. An edge from client to server labeled *busy* means server is processing client's request. For prdoctions REQ models the situation where client sends request to server, REL models the situation where server finished client's request and ready to accept further requests, SER models the situation when server

switches state from *idle* to processing some client's request.

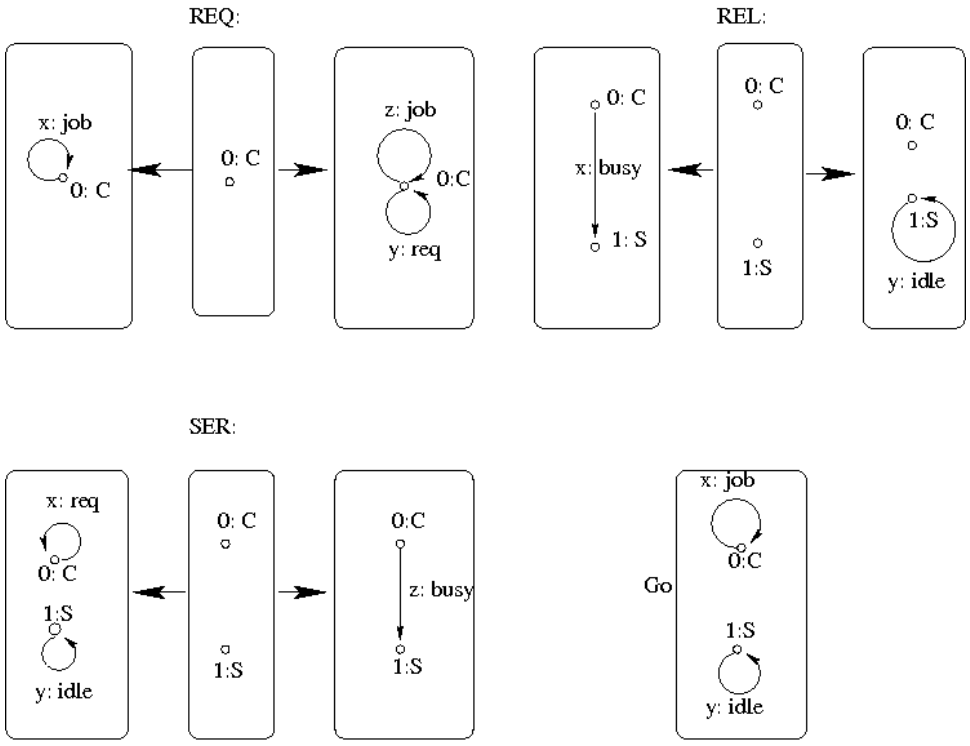


Figure : productions and start graph of the grammar modeling a client−server system.

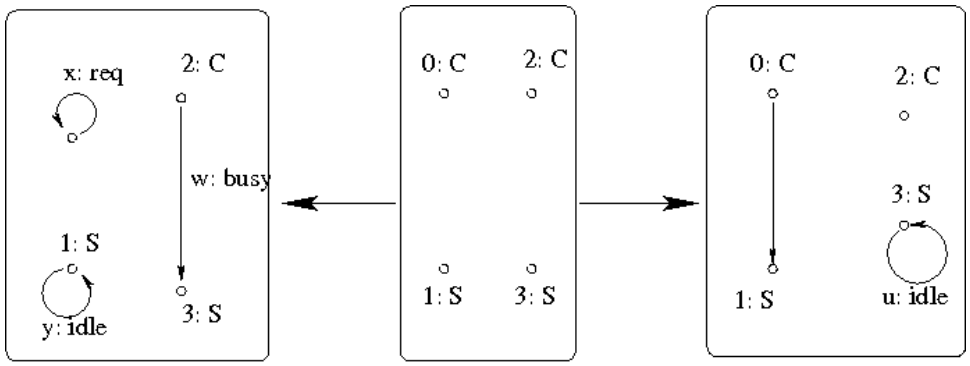


Figure : additional production SYNCHR:

Suppose that grammar client-server is extended with the production SYN-CHR which models the situation where the service of a request starts exactly when another service terminates. This production is shown in the above figure.

This production is span-isomorphic to the parallel production SER+REL, therefore these two productions would be identified by $\equiv_0$. But these derivations are very different, i.e., SER+REL can be sequentialized in any order, while other is sequential derivation and not decomposable.

2.3.5 Equivalence $\equiv_1$

Definition 2.3.9. (equivalence $\equiv_1$)
Let $\rho : G_0 \Rightarrow^* G_n$ and $\rho' : G'_0 \Rightarrow^* G'_n$ be two derivations such that $\rho \equiv_0 \rho'$ as in the figure given above. Then they are $1-$equivalent (written $\rho \equiv_1 \rho'$) if

1. *There exists a family of permutations $< \pi_1, ..., \pi_n >$ such that for each $i \in \{1, ..., n\}$, productions q_i and q'_i are isomorphic via π_i;*

2. *For each $i \in \{1, ..., n\}$, the family of isomorphisms $\{\phi_{L_i} : L_i \to L'_i, \phi_{K_i} : K_i \to K'_i, \phi_{R_i} : R_i \to R'_i, \}$ that exists by definition of $\equiv_0$ is exactly the family of isomorphisms between corresponding graphs of q_i and q'_i which is induced by permutations π_i according to the result that* **isomorphic parallel productions are span-isomorphic**.

we say that ρ and ρ' are $1-$equivalent via $< \pi_i >_{i \leq n}$ if $< \pi_i >_{i \leq n}$ is the family of permutations of point 1 above. Given a derivation ρ, it's equivalence class with respect to $\equiv_1$ is denoted by $[\rho]_1$, and is called a $1-$abstract derivation.

Lemma 2.3.10. (analysis, synthesis and shift preserve $\equiv_1$)

Let $\rho \equiv_1 \rho'$ via $< \pi_i >_{i \leq n}$ and let ϕ_{G_0} and ϕ_{G_n} be the isomorphisms between their starting and their ending graphs.

1. *If $\rho_1 \in ANAL^i_j(\rho)$ then there is at least one derivation $\rho_2 \in ANAL^i_{\pi_i(j)}(\rho')$. Moreover for each $\rho_2 \in ANAL^i_{\pi_i(j)}(\rho)'$ it holds that $\rho_2 \equiv_1 \rho_1$.*

2. *If* $\rho_1 \in SYNT^i(\rho)$ *then there is at least one derivation* $\rho_2 \in SYNT^i(\rho')$. *Moreover for each* $\rho_2 \in SYNT^i(\rho')$. *it holds that* $\rho_2 \equiv_1 \rho_1$.

3. *If* $\rho_1 \in SHIFT^i_j(\rho)$ *then there is at least one derivation* $\rho_2 \in SHIFT^i_{\pi_i(j)}(\rho')$. *Moreover for each* $\rho_2 \in SHIFT^i_{\pi_i(j)}(\rho')$ *, it holds that* $\rho_2 \equiv_1 \rho_1$.

   Furthermore in all these three cases, the isomorphisms relating the starting and ending graphs of the 1-equivalent derivations ρ_2 *and* ρ_1 *are exactly isomorphisms* ϕ_{G_0} *and* ϕ_{G_n}.

Proof. **Analysis proof**

We assume that ρ and ρ' are parallel direct derivations which implies that $\rho \equiv_1 \rho'$ via π. This assumption is made for simplicity as we will have to deal with parallel derivations of length 1. It can be generalised to equivalent derivations of arbitrary length. Also existence of $\rho_2 \in ANAL^i_{\pi_i(j)}(\rho)'$ is guaranteed by **Analysis of parallel direct derivations** lemma.

Two direct derivations $\rho = p_1 + p_2 + ... + p_k : G \Rightarrow H$ and $\rho' = p'_1 + p'_2 + ... + p'_k : G' \Rightarrow H'$ are transformed into corresponding derivations $\eta = (p_j + q_2 : G \Rightarrow H)$ and $\eta' = (p'_{\pi_{(j)}} + q'_2 : G' \Rightarrow H')$ using lemma **Analysis of parallel direct derivations**, where $q_2 = p_1 + ... + p_{j-1} + p_{j+1} + ... + p_k$ and $q'_2 = p'_1 + ... + p'_{\pi_{(j-1)}} + p'_{\pi_{(j+1)}} + ... + p'_k$.

Suppose $p_j = (l_1 : K_1 \to L\ , l_1 : K_1 \to R)$,
$q_2 = (l_2 : K_2 \to L\ , l_2 : K_2 \to R)$,
$p'_{\pi_{(j)}} = (l'_1 : K'_1 \to L'_1, l'_1 : K'_1 \to R'_1)$,
$q'_2 = (l'_2 : K'_2 \to L'_2, l'_2 : K'_2 \to R'_2)$,
We know that $p_j = p'_{\pi_{(j)}}$ i.e., we have $X = X'$ for $X \in \{L, K, R, l, r\}$.

Let π_1 be the unique permutation on $\{1\}$ and π_2 be the bijective mapping between the productions of q_2 and q'_2 induced by π.

Let us assume that the derivations η and η' have the same shape of the right double-pushout of figure **splitting a parallel production**. Then $(1) + (2)$ is derivation η and similarly we can have another diagram for η' having the derivation $(1') + (2')$. We also have that η and η' are related by family of isomorphisms $\{\phi_X : X \to X' | X$ is a graph of $\eta\ \}$ making the diagram commutative.

Now we apply **butterfly lemma** on the double-pushouts $(1) + (2)$ and $(1') + (2')$ corresponding to η and η' to get pushouts (3) to (8) and $(3') + (8')$ like in figure **Application of butterfly lemma**. And also construct (9)

and (9′) corresponding to η and η' respectively. Like in figure **Building the sequential derivation**.

Now all the corresponding graphs of two families are related by the isomorphisms relating η and η', except X_1, X_2, Y_1, Y_2 and X. Isomorphism $\phi_{X_1} : X_1 \to X_1'$ is uniquely determined since (3) and (3′) are pushouts. Similarly ϕ_{X_1}, ϕ_{X_2}, ϕ_{Y_1}, ϕ_{Y_2},ϕ_X are also determined.

From isomorphisms constructed as above combined with π_1, π_2, it follows that two sequential derivations corresponding to η and η' are 1-equivalent.

(Synthesis proof)

We assume that ρ and ρ' have length 2, therefore $\rho \equiv_1 \rho^1$ via $< \pi_1, \pi_2 >$. By hypothesis $\rho_1 \in SYNT^i(\rho)$ therefore ρ is sequential independent.

• To show the existence of $\rho_2 \in SYNT^i(\rho')$ by **synthesis of sequential independent derivation** lemma, it is sufficient to show that ρ' is sequential independent.

Let ρ' be a sequential derivation like in figure **a sequential independent derivation**. We have that $\rho \equiv_1 \rho^1$, therefore there exists a family of morphisms relating the corresponding graphs of ρ and ρ' so that everything commutes, making ρ' sequentially independent.

Pushouts (3) to (8) and (3′) + (8′) like in figure **Application of butterfly lemma** constructed for η and η' respectively induce isomorphisms between corresponding graphs by **synthesis of sequential independent derivation** lemma. Since all the corresponding graphs are either pushouts or pushout complement objects, the morphisms $\phi_D : D \to D'$, $\phi_{X_1} : X_1 \to X_1'$, $\phi_{Y_2} : Y_2 \to Y_2'$ are uniquely determined.

From isomorphisms constructed as above combined with $< \pi_1, \pi_2 >$ it above it follows that η and η' are 1-equivalent.

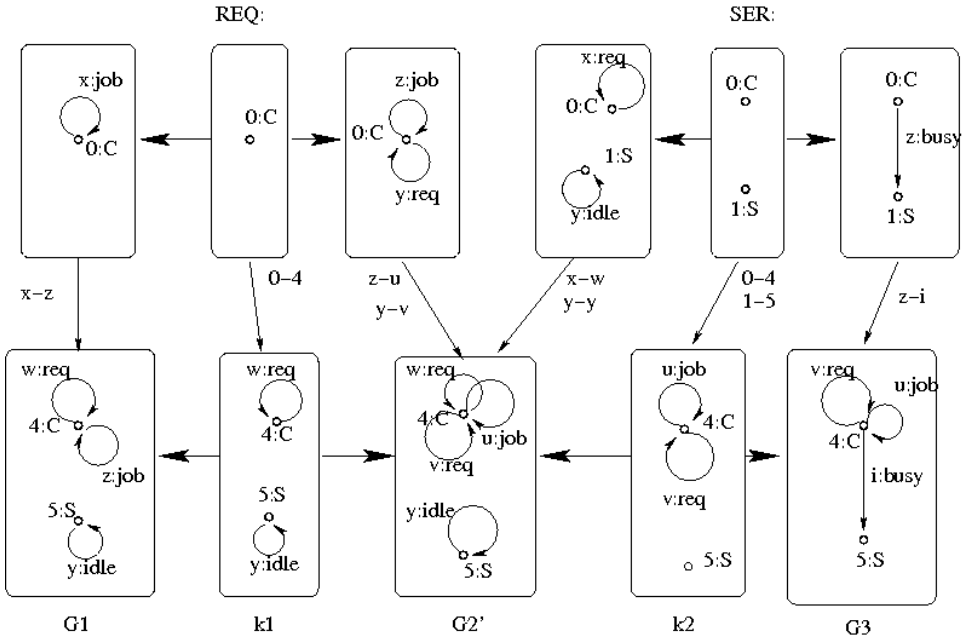
(Shift proof)
If $j \geq 1$ proof immediately follows from (1) and (2) above as $SHIFT_j^i = SYNT^{i-1} \circ ANAL_j^i$ by definition of $SHIFT$.

If $j = 0$ then it means that $\pi_i(j) = 0$, then both ρ and ρ' are empty directions. In this case ρ_2 and ρ_1 are uniquely determined because $SHIFT_0^i$ is deterministic. If this construction is applicable to ρ then it is applicable to ρ' also giving 1-equivalent derivations.

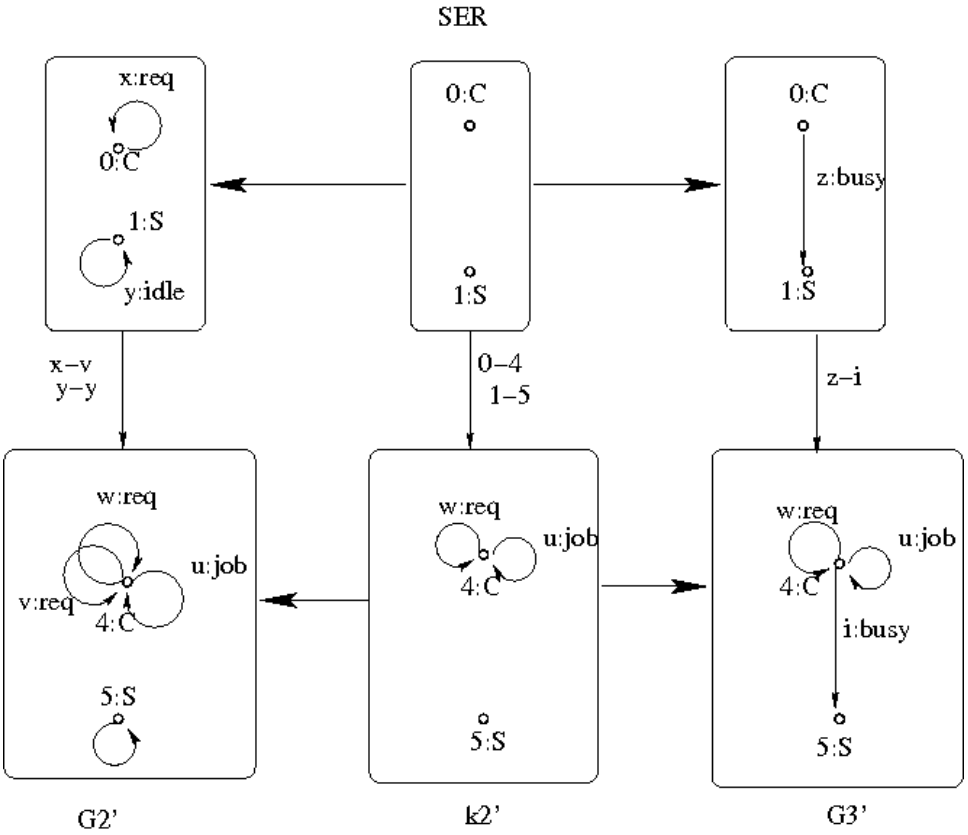
□

Theorem 2.3.11. **($\equiv_1$ enjoys uniqueness of canonical derivations)** *Let ρ and ρ' be two derivations such that $\rho \equiv_1 \rho'$, and let ρ_c, ρ'_c be two canonical derivations such that $\rho_c \equiv_{sh} \rho'_c$ and $\rho' \equiv_{sh} \rho'_c$. Then $\rho_c \equiv_1 \rho'_c$.*

Drawback of $\equiv_1$ model is that it does not allow for the sequential composition. We prove this by giving a counter example. Consider two derivations $d_1; d_2$ shown in figure(a), and d'_2 shown in figure(b), which are given below.



Figure(a): derivation d1; d2



Figure(b) : direct derivation d2'

We see that $d_2 \equiv_1 d_2'$,since productions applied SER are same, and we have family of isomorphisms $\phi_2 = \{\phi_{G_2'} : G_2' \to G_2', \phi_{K_2} : K_2 \to K_2', \phi_{G_3} : G_3 \to G_3'\}$ with $\phi_{G_2'}(w) = v, , \phi_{G_2'}(v) = w$, $\phi_{K_2}(v) = w$, $\phi_{G_3}(v) = w$, and for other edges and vertices, we use identity morphisms. Also $d_1 \equiv_1 d_1$ using family of morphisms $\phi_1 = \{id_{G_1}, id_{K_1}, id_{G_2'}\}$. We see that $\tau(d_1) = \sigma(d_2') = G_2'$ therefore $d_1; d_2'$ is defined.

So we have two derivations $d_1; d_2$ and $d_1; d_2'$ with $d_2 \equiv_1 d_2'$ and $d_1 \equiv_1 d_1$. but these two derivations are not 1-equivalent, since ϕ_1 and ϕ_2 do not agree on G_2'. These two derivations are semantically different also. In the first derivation $d_1; d_2$ first request will be served while in second derivation $d_1; d_2'$ second request is served first. Therefore there does not exist a family of isomorphisms between corresponding graphs of two derivations.

2.3.6 Equivalence $\equiv_3$

Definition 2.3.12. (standard isomorphisms)
A family s of standard isomorphisms in category $\mathcal{GRAPH}$ is a family of isomorphisms indexed by pairs of isomorphic graphs (i.e., $s = \{s(G, G') | G \cong G'\}$) satisfying the following conditions for each G, G' and $G'' \in |\mathcal{GRAPH}|$:

1. $s(G, G') : G \to G'$;
2. $s(G, G) = id_G$;
3. $s(G'', G') \circ s(G, G'') = s(G, G'')$.

Definition 2.3.13. (Equivalence $\equiv_3$)

Let s be an arbitrary but a fixed family of standard isomorphisms of category $\mathcal{GRAPH}$, and let ρ, ρ' be two derivations. we say that ρ and ρ' are $3-$equivalent (written $\rho \equiv_3 \rho'$) iff they are 1-equivalent, and moreover the isomorphisms ϕ_{G_0} and ϕ_{G_n}, relating their starting and ending graphs are standard. An equivalence class of derivations wrt $\equiv_3$ is denoted by $[\rho]_3$, and is called a $3-$abstract derivation.

Lemma 2.3.14. ($\equiv_3$ allows for sequential composition)

Let ρ ,ρ'_1,ρ_2 and ρ'_2 be four derivations such that $\rho_1 \equiv_3 \rho'_1$, $\rho_2 \equiv_3 \rho'_2$, $\tau(\rho_1) = \sigma(\rho_2)$ and $\tau(\rho'_1) = \sigma(\rho'_2)$. Then $\rho_1; \rho_2 \equiv_3 \rho'_1; \rho'_2$.

Proof. By hypothesis there are two families of isomorphisms making ρ_1 and ρ_2 $3-$*equivalent* to ρ'_1 and ρ'_2, respectively. The two families must agree on $\tau(\rho_1) = \sigma(\rho_2)$ since there is only one standard isomorphism between a given pair of isomorphic graphs. Thus union of these families provides a family of isomorphisms which makes 3-equivalent $\rho_1; \rho_2$ and $\rho'_1; \rho'_2$.

□

Proposition 2.3.15. (properties of equivalence $\equiv_3$)
Equivalence $\equiv_3$ is well defined, it allows for sequential composition , and it enjoys uniqueness of canonical derivations.

Proof. 1. Sequential composition is well defined by the proposition (**$\equiv_3$ allows for sequential composition**).

2. Uniqueness of canonical derivations follows from theorem (**$\equiv_1$ enjoys uniqueness of canonical derivations**) .

3. Analysis, synthesis and shift preserve $\equiv_1$ along with isomorphisms relating starting and the ending graph of 1-equivalent derivations , it follows that they preserve $\equiv_3$ also.

□

2.3.7 Abstract model of computation in the DPO Approach

Definition 2.3.16. (abstract model of computation)

Given a grammar $\mathcal{G}$, it's abstract model of computation $\mathbf{ADer}(\mathcal{G})$ *is defined as follows.*

The objects of $\mathbf{ADer}(\mathcal{G})$ *are abstract graphs, i.e., isomorphism classes of objects of $\mathcal{GRAPH}$. Arrows of* $\mathbf{ADer}(\mathcal{G})$ *are 3-abstract derivations, i.e., equivalence classes of (parallel) derivations of $\mathcal{G}$ with respect to equivalence $\equiv_3$. The source and target mapping are given by σ and τ, respectively; thus $[\rho]_3 : (\sigma[\rho]_3) \to \tau([\rho]_3)$. Composition of arrows is given by the sequential composition of the corresponding 3-abstract derivations.For each object $[G]$ the identity arrow is the 3-abstract derivation $[G : G \Rightarrow^* G]_3$ containing the identity derivation G.*

Category $\mathbf{ADer}(\mathcal{G})$ is equipped with an equivalence relation $\equiv_{sh}$ on arrows, defined as $[\rho]_3; [\rho']_3$ if $\rho \equiv_{sh} \rho'$.

Proposition 2.3.17. (category $\mathbf{ADer}(\mathcal{G})$ is well-defined)

Category $\mathbf{ADer}(\mathcal{G})$ *s well-defined. Moreover, equivalence $\equiv_{sh}$ is a congruence with respect to arrow compositions, i.e., if $[\rho_1]_3 \equiv_{sh} [\rho'_1]_3$ and $[\rho_2]_3 \equiv_{sh} [\rho'_2]_3$, then $[\rho_1]_3; [\rho_2]_3 \equiv_{sh} [\rho'_1]_3; [\rho'_2]_3$; (if the compositions are well defined).*

Proof. The two facts follow easily from the properties of equivalence $equiv_3$ summarized in proposition **properties of equivalence $\equiv_3$** and from the theorem (**$\equiv_1$ enjoys uniqueness of canonical derivations**) . □

Category $\mathbf{ADer}(\mathcal{G})/_{\underline{sh}}$ is equipped with an equivalence relation $\equiv_{\underline{sh}}$ on arrows, defined as $[\rho_3] \equiv_{\underline{sh}} [\rho']_3$ if $\rho \equiv_{sh} \rho'$.

2.3.8 Abstract, truly-concurrent model of computation in the DPO Approach

Definition 2.3.18. (Abstract, truly-concurrent model of computation)
The Abstract, truly−concurrent model of computation for a grammar $\mathcal{G}$ *is defined as category* $\mathbf{ADer}(\mathcal{G})/_{\underline{sh}}$*, having the same objects as* $\mathbf{ADer}(\mathcal{G})$ *and as arrows as equivalence classes of arrows of* $\mathbf{ADer}(\mathcal{G})$ *wrt equivalence* $\equiv_{\underline{sh}}$*.*

Chapter 3

SPO approach to graph transformations

3.1 Graph transformation based on SPO Approach

Definition 3.1.1. (partial graph morphism)
Let $G =< G_V, G_E, s^G, t^G, lv^G, le^G >$ *be a graph, where* G_V *is a set of vertices(or nodes),* G_E *is a set of edges (or arcs),* $s^G, t^G : G_E \to G_V$ *are the source and target functions, and* $lv^G : G_V \to \Omega_V$ *and* $le^G : G^E \to \Omega_E$ *are the node and the edge labeling functions, respectively.*

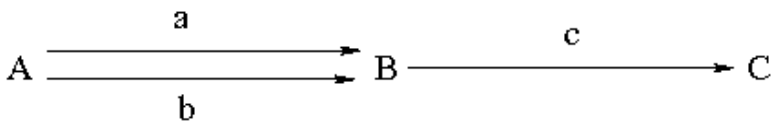
A **subgraph** *S of G , written* $S \subseteq G$ *or* $S \hookrightarrow G$ *is a graph with* $S_V \subseteq G_V$, $S_E \subseteq G_E$, $s^S = s^G|_{S_E}$, $le^S = le^G|_{S_E}$.

A **partial graph morphism** *g from G to H is a total graph morphism from some subgraph dom(g) of G to H, and dom(g) is called the domain of g.*

Definition 3.1.2. production, graph grammar
A **production** $p : (r : L \to R)$ *is composed of a production name p and of an injective partial graph morphism r, called the partial morphism. The graphs L, R are called the left-hand side(lhs), and the right-hand side(rhs) of p, respectively.*

A **graph grammar** *G is a pair* $G =< (p : r)_{p \in P}, G_o >$ *where the first component is a family of production morphisms indexed by production names in P and* G_o *is the* **start graph** *of the grammar.*

Definition 3.1.3. (co-equalizer)
Given a category $\mathcal{C}$ and two arrows $a : A \to B$ and $b : A \to B$ of C, a tuple $< C, c : B \to C >$ is called co-equalizer of $< a, b >$ if $c \circ a = c \circ b$ for all objects D and arrows $d : B \to D$, with $d \circ a = d \circ b$, there exists a unique arrow $u : C \to D$ such that $u \circ c = d$.



Proposition 3.1.4. (construction of specific co-equalizer in $\mathcal{GRAPH}^{\mathcal{P}}$)
Let $a, b : A \to B$ be two partial morphisms such that for each $x \in A$, if both a and b are defined on x then $a(x) = b(x)$. Then, the co-equalizer of a and b in $\mathcal{GRAPH}^{\mathcal{P}}$ is given by $< C, c >$ where,

- *$C \subseteq B$ is the largest subgraph of $[b(A) \cap a(A)] \cup [\overline{a(A)} \cap \overline{b(A)}]$, and*
- *$dom(c) = C$ and c is the identity morphism on it's domain.*

Proof. (proof that above construction indeed gives us a co-equalizer)

- commutativity $c \circ a = c \circ b$

  For each $x \in A$, if both a and b are defined on x then $a(x) = b(x)$ and c is the identity morphism on it's domain, therefore it follows that $c \circ a = c \circ b$.

- universal property

  For any other morphism $d : B \to D$ with $d \circ a = d \circ b$ we have $dom(d) \subseteq dom(c)$ from the first point of construction that c is the largest subgraph. Therefore the required unique arrow $u : C \to D$ is just the restriction of c to $dom(d)$.

□

Example 3.1.5. (deletion of elements by co-equalizer construction)

Following figure shows how specific co-equalizer constructed as above deletes elements. Morphism a is empty and b is a total morphism in $\mathcal{GRAPH}^{\mathcal{P}}$. vertex 2 is deleted because it has preimage under b but not under a, i.e., $2 \notin [b(A) \cap a(A)$ and also $2 \notin \overline{a(A)} \cap \overline{b(A)}$.

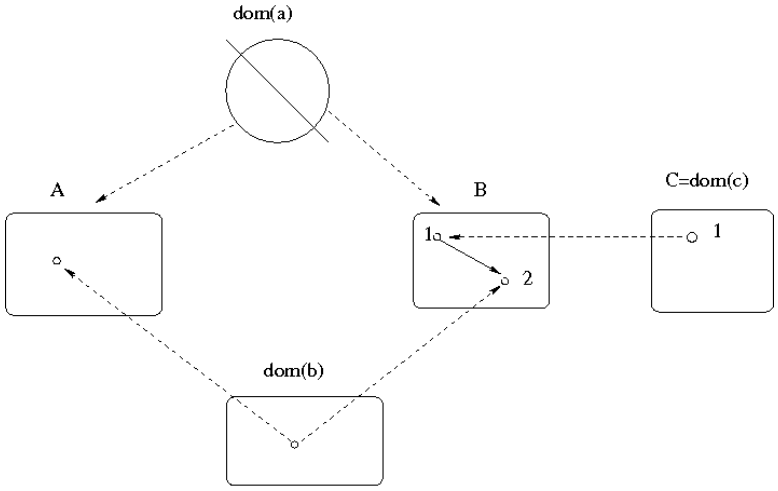


Figure: Deletion as a co–equalizer

Proposition 3.1.6. (pushout in $\mathcal{GRAPH}^{\mathcal{P}}$)

The pushout of two morphisms $b : A \to B$ and $c : A \to C$ in $\mathcal{GRAPH}^{\mathcal{P}}$ always exists and can be computed in three steps, as shown in figure 3:

1. [**gluing1**] *Construct the pushout $< C', A \to C', C \to C' >$ of the total morphisms $dom(c) \to C$ and $dom(c) \to A$ in $\mathcal{G}raph$.*

2. [**gluing2**] *Construct the pushout $< D, B \to D, C' \to D >$ of the total morphisms $dom(b) \to A \to C'$ and $dom(b) \to B$ in $\mathcal{G}raph$.*

3. [**deletion**] *Construct the co-equalizer $< E, D \to E >$ of the partial morphisms $A \to B \to D$ and $A \to C \to C' \to D$ in $\mathcal{GRAPH}^{\mathcal{P}}$.*

$< E, C \to C' \to D \to E, B \to D \to E >$ *is the pushout of b and c in $\mathcal{GRAPH}^{\mathcal{P}}$.*

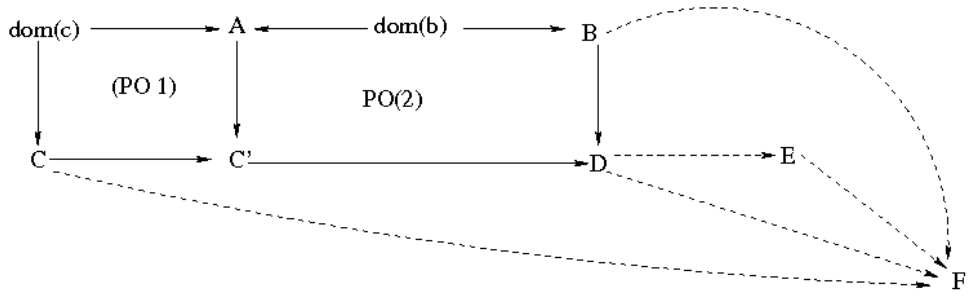


Figure 3: construction of pushout in SPO

Proof. Commutativity holds by construction. To prove uniqueness assume that there is $< F, C \to F, B \to F >$ with $A \to B \to F = A \to C \to F$. Then we have to show that there exists a unique morphism $E \to F$ such that, $B \to D \to E \to F = B \to F$ and $C \to C' \to D \to E \to F = C \to F$. □

Definition 3.1.7. (match, derivation)
A **match** *for $r : L \to R$ in some graph G is a total morphism $m : L \to G$. Given a production r and a match for r in a graph G, the* **direct derivation** *from G with r at m, written $G \Rightarrow^{r,m} H$, is the pushout of r and m in $\mathcal{GRAPH}^{\mathcal{P}}$ as shown in the following figure.*

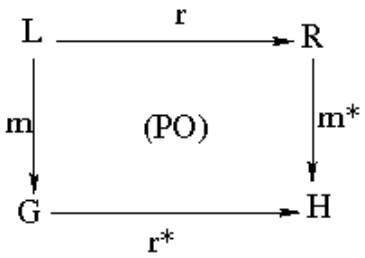


Figure 4: Direct derivation as pushout

A sequence of direct derivations of the form $\rho = (G_0 \Rightarrow^{r_1,m_1} .... \Rightarrow^{r_k,m_k} G_k)$ constitutes a **derivation** *from G_0 to G_k by $r_1...r_k$, briefly denoted as $G_0 \Rightarrow^* G_k$.*

The **graph language** *generated by a graph grammar $\mathcal{GRAPH}^{\mathcal{P}}$ is the set of all graphs G_k such that there is a derivation $G_0 \Rightarrow^* G_k$ using productions of $\mathcal{GRAPH}^{\mathcal{P}}$.*

Definition 3.1.8. (special matches)
Given a match $m : L \to G$ for a production $r : L \to R$. Then m is called **conflict free** *if $m(x) = m(y)$ implies $x, y \in dom(r)$ or $x, y \notin dom(r)$. It is called* **d-injective** *if $m(x) = m(y)$ implies $x, y \in dom(r)$ or $x = y$. Finally, m is* **d-complete** *if for each edge $e \in G_E$ with $s^G(e) \in m_V(L_V - dom(r)_V)$ or $t^G(e) \in m_V(L_V - dom(r)_V)$ we have $e \in m_E(L_E - dom(r)_V)$.*

Example 3.1.9. (special match, direct derivations as pushout construction)

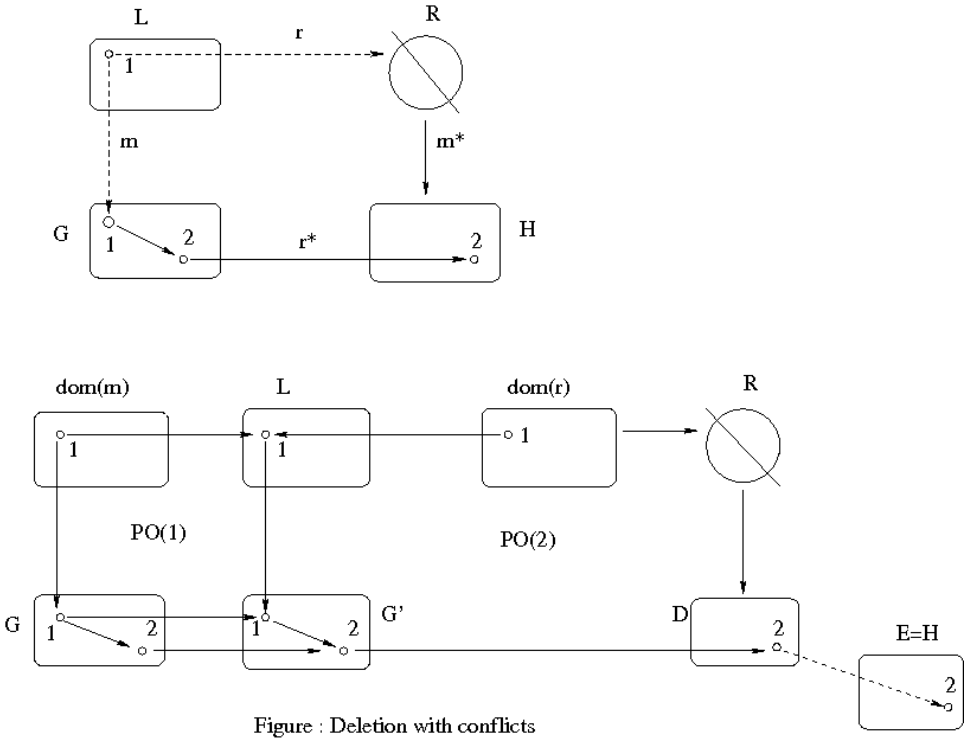


Figure : Deletion with conflicts

In the Diagram shown above, dangling edge is deleted as part of pushout construction, in D itself. It is important to note that it is not deleted as a part of co-equalizer construction but because deletion has got priority. Here the match is not **d-complete**, because dangling edge got implicitly deleted as a result of explicit deletion of it's source vertex 1.

Lemma 3.1.10. (pushout properties)

If $(H, r* : G \to H, m* : R \to H)$ is the pushout of $r : L \to R$ and $m : G \to H$ in in $\mathcal{GRAPH}^{\mathcal{P}}$, then the following properties are fulfilled:

1. *pushouts preserve surjectivity, i.e. r surjective implies r^* surjective.*
2. *pushouts preserve injectivity, i.e. r injective implies r^* injective.*
3. *r^* and m^* are jointly surjective.*
4. *m conflict-free implies m^* total.*

Proof. 1. (proof of first property i.e., r surjective implies r^* surjective) Given that r is surjective means no new elements are added in G to construct resulting pushout graph.It is also given that m is total means all elements in L are mapped to G.

So there are 3 cases to consider ,

(a) element mapped by m but preserved from G . Consider an element $a \in L$ which is mapped to some $a^{'}$ in G via m and a'' in R via r, i.e., $e \to^m e^{'}$ and $e \to^r e''$, then by the pushout construction e'' is mapped to equivalence class $\{e^{'}, e''\}$ in H.

(b) element mapped by m but not mapped by r is deleted from G Consider an element $a \in L$ which is mapped to some $a^{'}$ in G via m and not mapped to any element in R via r, i.e.,$e \to^m e^{'}$ and by the pushout construction e'' is mapped to equivalence class $\{e^{'}, \phi\}$ in H,so by co-equalizer construction this element is deleted in H.

(c) element not mapped by m and not mapped by r in G, Consider such $x \in G$ as described above, then by pushout construction this element forms it's own equivalence in H and is preserved.

Finally to prove that r^* is surjective, Assume there exists an element $x \in H$ which is not an image of any element in G via r^* . Then it must came in H through $L \to R \to H$which means that x in an image of some element $b \in R$ which was not mapped by any element in L via r , which is a contradiction since r is surjective.

2. (proof of second property i.e., r injective implies r^* injective) Assume that r^* is not injective. i.e, there exists an element $h \in H$ s.t., $r^*(a) = r^*(b) = h$ and $a \neq b$ in G. as shown in the following diagram.

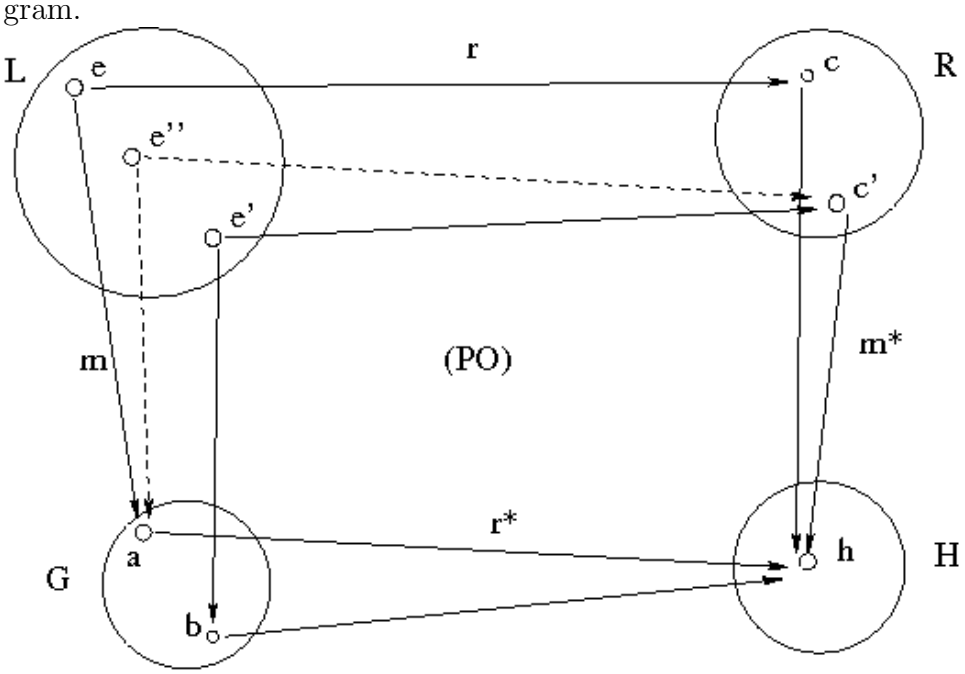


Figure: pushout property (2)

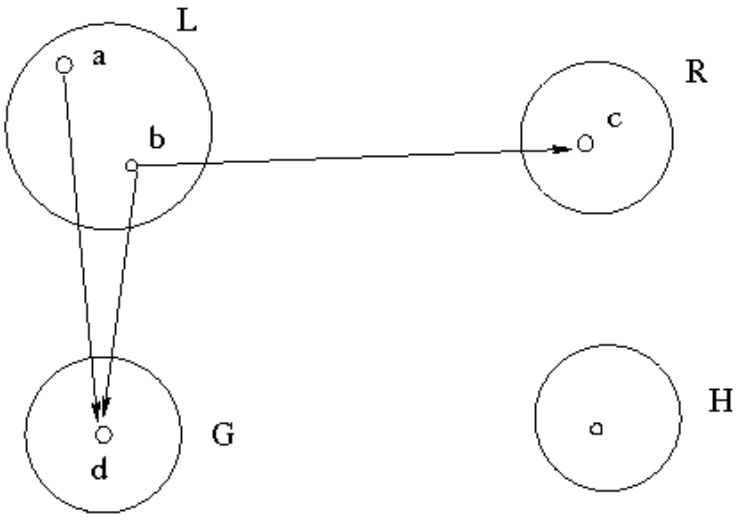
wlg assume that there are elements $e, e^{'} \in L$, $c, c^{'} \in R$ s.t. $m(a) = a$ and $m(e^{'}) = b$,and $m^*(c) = h = m^*(c^{'})$. Now observe that when we constructed H , (c, a) and $(c^{'}, b)$ form two different equivalence classes in H. The only way by which $(c, c', a, b) = h$ is that there must exist another element $e'' \in L$ to equate these two classes, i.e., $m(e'') = a$ and $r(e'') = c^{'}$ or $m(e'') = b$ and $r(e'') = c$ in either case we have r as non-injective which is a contradiction.

3. (proof of third property i.e., r^* and m^* are jointly surjective.) This property states that there does not exist $h \in H$ which not mapped either by r^* or m^* which is not possible by the construction of pushout

itself, since elements of H are equivalences classes made up from elements of R and G.

4. (proof of fourth property i.e., m conflict-free implies m^* total.) Assume that m^* is not total but m is conflict-free. Consider an element $c \in R$ which was not mapped under m^* in H. This means that c did not land up in it's equivalence class, i.e, it got deleted while construction of pushout. This is possible through only two cases:

   In first case c was mapped under r by some element $a \in L$ as shown in diag 1 of following figure, $r(a) = c$, but b was not mapped under r. and both of them mapped to same element $d \in G$.So, equivalences classes formed are (d, c) and (d, ϕ) leading to (d, c, ϕ) which means c was deleted as a result. but this case implies that m is not conflict free, since $m(a) = m(b) = d$ and $a \neq b$.



Vertex c is deleted beacuse it could not form it's own equivalence class in H

Figure : pushout property (4)

In second case c was not mapped by any element in L via r, then

either c was new element to be added, in that case it must have formed it's own equivalence class leading to contradiction that it was mapped under m^* or c was the element to be deleted again which is considered in first case as described above.

□

Definition 3.1.11. (Translation of double-pushout transformations) *If $r : L \to R$ is a production rule in SPO, $D(r) = (l : L_r \to L, r' : L_r \to R)$ denotes it's translation to the double-pushout rule, where l is the inclusion of L_r in L and r' is the domain restriction of r to L_r. Conversely for a double-pushout rule $p = (l : K \to L, r : K \to R)$, $S(p) : L \to R$ denotes it's translation to single-pushout rules, where $L_{S(p)} = l(K)$ and $S(p) = r \circ l^{-1}$.*

Theorem 3.1.12. (Embedding of the DPO approach)

If the object H is the result of transforming an object G with rule p at match m in the double-pushout framework, the translation of p to a single pushout rule i.e., $S(p)$, transforms G to H at the same match m in the single pushout setting.

Conversely, if G can be transformed to H with rule r at match m by a single pushout transformation, the translation of r to a double pushout rule , i.e., D(r), is applicable to G at m in the double pushout-framework iff m is d-injective and d-complete. In this case, the double pushout transformation of G with D(r) at m results in the same object H.

Proof. (**SPO $\Rightarrow$ DPO**)

L_s is preimage of $s : L \to R$ and G_{s_m} is preimage of $s_m : G \to H$.
We have $m(L_s) \subseteq G_{s_m}$, $m_s : R \to H$ is total, and $m(L - L_s) \subseteq G - G_{s_m}$ since m is d-injective and d-complete.

Consider following diagram.

we construct pushout (2) as follows. Interface graph $K = L_s$, context graph $D = G_{s_m}$, l is identity map from $L_s \to L$, l^* is identity map from $G_{s_m} \to G$, $k : K \to D$ is $(l^*)^{-1} \circ m \circ l$, since l, m, l^* are injective, k is also injective and total. It is easy to see that $< D, k, l^* >$ is pushout-complemet of $< K, l, m >$ and since l, m, are injective it is unique. Therefore (2) is pushout.

We construct pushout (3) as follows. r is identity map from $L_s \to R$, $r^* : G_{s_m} \to H$ is restriction of of s_m to G_{s_m}. It is easy to see that $< D, k, r^* >$

is pushout-complemet of $< K, l, m >$ and since r and k are injective it is unique. Therefore (3) is pushout.

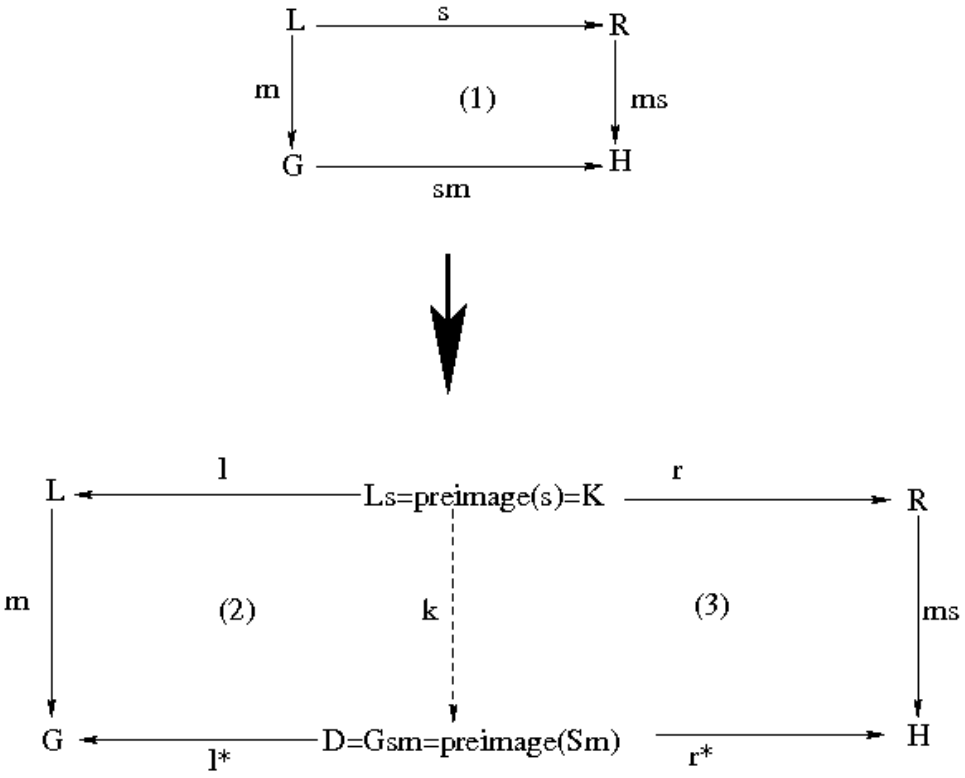


Figure: SPO to DPO construction

(**DPO** $\Rightarrow$ **SPO**)

Since we are starting with a DPO construction , we know that m satisfies gluing condition, which implies that m is d-injective and d-complete.Therefore m^* is total. Let $S(l, r)$ be the translation for s ,and $S(l^*, r^*)$ be the translation for s^* in pushout (3).

Now consider following diagram.

Construction of pushout (4) is trivial since, $dom(m) = L$, $id_L : L \to dom(m)$, $G^{'} = G$ and $id_G : G \to G^{'}$. Therefore $m^{'} = m$, Hence it is easy to see that (4) is a pushout.

We construct pushout (5) in the diagram as follows. $dom(s) = l(k)$; $s : L \rightarrow R$ is defined as for all $e \in l(k)$ $s(e) = r(l^{-1}(e))$; $l : dom(s) = l(K) \rightarrow L$; $r : dom(s) = l(K) \rightarrow R$; We construct g'' as pushout of $< L, r|_K : L \rightarrow R, m : L \rightarrow G >$ as shown in the pushout (6) where $r^*|_D$ is a map from $G \rightarrow G''$. It is clear that (5) is a pushout of total morphisms, $dom(s) \rightarrow K \rightarrow G'$ and $dom(s) \rightarrow R$. Graph G'' contains all the original elements of G and added elements from R.

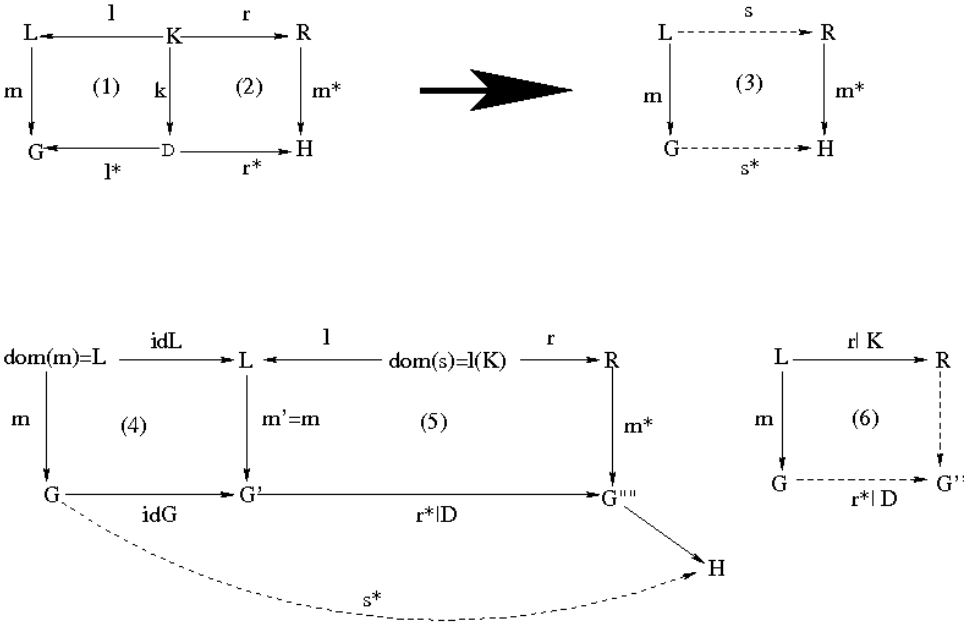


Figure : DPO to SPO construction

Now construct the co-equalizer $< H, G'' \rightarrow H >$ of total morphisms $L \rightarrow G \rightarrow G' \rightarrow G''$ and $L \rightarrow R \rightarrow G''$. Which will delete elements specified by pushout (1), Since elements coming through morphism $L \rightarrow G \rightarrow G' \rightarrow G''$ are not mapped by $L \rightarrow R \rightarrow G''$they will be deleted and will preserve elements as required since they will be mapped by both the morphisms considered, since elements to be preserved are specified by $l(K)$ which are mapped in both the morphisms, and all the component morphisms are total.

Morphism s^* is $G \rightarrow G^{'} \rightarrow G'' \rightarrow H$ as required for the pushout (3). $\square$

3.2 Parallelism in SPO approach

There are two ways to model concurrent computations, the interleaving and truly concurrent model.

3.2.1 Interleaving

In interleaving model two actions are concurrent i.e. potentially in parallel, if they may be performed in any order with the same result. Single actions are modeled by direct derivations. Again from two different points of view this notion of parallelism is studied, parallel independence and sequential independence.

Parallel independence

Parallel independence condition ensures that two alternative derivations are not mutually exclusive .

A direct derivation $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ does not affect $d_2 = (p_2, m_2 : G \Rightarrow H_2)$, if d_1 does not delete elements in G which are accessed by d_2. The vertices deleted from G by d_1 are those in $m_1(L_1 - dom(p_1))$. An edge will be deleted from G if it is in $m_1(L_1 - dom(r_1))$. or one of it's incident vertices is deleted.

Definition 3.2.1. (parallel independence)
Let $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ and $d_2 = (p_2, m_2 : G \Rightarrow H_2)$ be two alternative direct derivations. Then we say that d_2 is weakly parallel independent of d_1 iff $m_2(L_2) \cap m_1(L_1 - dom(p_1)) = \phi$. We call the derivations d_1 and d_2 parallel independent if they are mutually weakly parallel independent.

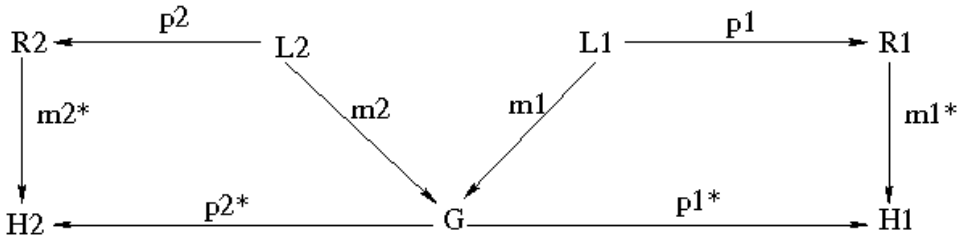


Figure : parallel inependence

Lemma 3.2.2. (categorical characterisation of weak parallel independence)
Let $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ and $d_2 = (p_2, m_2 : G \Rightarrow H_2)$ be two direct derivations. Then d_2 is weakly parallel independent of d_1 iff $p_1^* \circ m_2 : L_2 \Rightarrow H_1$ is a match for p_2.

Proof. ($\Rightarrow$) Assume that we have a match $p_1^* \circ m_2 : L_2 \Rightarrow H_1$ as described above then we have to prove that d_2 is weakly parallel independent of d_1 i.e., $m_2(L_2) \cap m_1(L_1 - dom(p_1)) = \phi$.

$m_2(L_2) \subseteq dom(p_1^*)$ by definition of composition. Now, $L_1 - dom(p_1$ contains elemets which are not mapped in R i.e, these are the elements which are deleted. Therefore by commutativity of pushouts $m_2(L_2) \cap m_1(L_1 - dom(p_1)) = \phi$.

To be more descriptive assume that there was such an element $c \in G$ such that $c \in m_2(L_2) \cap m_1(L_1 - dom(p_1))$, then by pushout commutativity of right pushout this element is deleted in H_1, but this is not possible since it is mapped to some element of L_2 via existence of m_2', which was needed by m_2 which is total morphism, leading to contradiction.

($\Leftarrow$)Assume that d_2 is weakly parallel independent of d_1 i.e., $m_2(L_2) \cap m_1(L_1 - dom(p_1)) = \phi$. and we have to prove that match $p_1^* \circ m_2 : L_2 \Rightarrow H_1$ is total.

Let us consider **vertices** first. Vertex which explicitly deleted by p_1 i.e., which belongs to $m_1(L_1 - dom(p_1)$ is not there in H_1. So, it do not belong to $m_2(L_2)$ therefore it does not belong to $dom(p_1^*)$. Vertex which is not explicitly deleted by p_1 is preserved i.e., present in H_1 , and it belongs to $dom(p_1^*)$. This implies that each preimage of $v \in G$ under m_2 also have an image in H_1.

Now consider **edges**. Each edge e which is preserved by p_1, by following arguments for vertices $e \in dom(p_1^*)$. Each edge e which is implicitly deleted by p_1 i.e., $e \in G - m_1(L_1 - dom(p_1))$, i.e., an edge which is deleted because either source or target vertex was deleted i.e., $s^G(e) \in m_1(L_1 - dom(p_1))$ or $t^G \in m_1(L_1 - dom(p_1))$. But by definition of parallel independence $s^G(e) \notin m_1(L_1 - dom(p_1))$ or $t^G(e) \notin m_1(L_1 - dom(p_1))$ implying that $e \notin m_2(L_2)$ by graph morphisms. Therefore $e \in m_2(L_2)$ implies that $e \in dom(p_1^*)$. □

Sequential independence The condition of sequential independence ensures that two consecutive direct derivations are not causally dependent. A direct derivation is weakly sequential independent of the previous one

if it could already have been performed before that. Stronger sequential independence requires that additionally second will not delete anything which was needed by first one.

Definition 3.2.3. (sequential independence)
Let $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ and $d_2' = (p_2, m_2 : H_1 \Rightarrow X)$ be two consecutive direct derivations. Then we say that d_2' is weakly sequentially independent of d_1 if $m_2'(L_2) \cap m_1^*(R_1 - p_1(L_1)) = \phi$. If additionaly $m_2'(L_2 - dom(p_2)) \cap m_1^*(R_1) = \phi$, we say that d_2' is sequentially independent of d_1,and the derivation $(G \Rightarrow^{p_1,m_1} H_1 \Rightarrow^{p_2,m_2'} X)$ is called sequentially independent.

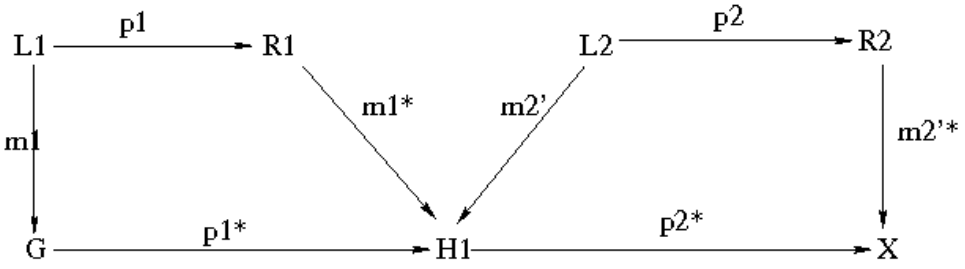


Figure : sequential independence

Lemma 3.2.4. (categorical characterisation of sequential independence)
Assume two direct derivations $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ and $d_2 = (p_2, m_2' : H_1 \Rightarrow X)$. Then d_2' is **weakly sequentially independent** *of d_1 iff there is a match $m_1 : L_2 \rightarrow G$for p_2 such that $m_2' = p_1^* \circ m_2$. The derivation d_2' is* **sequentially independent** *of d_1 iff d_2' is weakly sequentially independent of d_1 and d_1 is weakly parallel independent of the correspondingly existing derivation $d_2 = (p_2, m_2' : G \Rightarrow H_2)$.*

Proof. Let us divide this proof in four parts.

1. Assume that we have there is a match $m_1 : L_2 \rightarrow G$ for p_2 such that $m_2' = p_1^* \circ m_2$ we have to prove that d_2' is **weakly sequentially independent** of d_1 . i.e., $m_2'(L_2) \cap m_1^*(R_1 - p_1(L_1)) = \phi$.

   $m_2' = p_1^* \circ m_2$ ———— (1) (by assumption).
   $m_2' = p_1^*(m_2)$ ————(2) (from 1).

$m'_2 \subseteq dom(p^*_1)$ ——————(3) (be definition of function composition).
$p^*_1(G) \cap m^*_1(R_1 - p_1(L_1)) = \phi$ ——-(4) (by left pushout construction).
$p^*_1(m_2(L_2)) \cap m^*_1(R_1 - p_1(L_1)) = \phi$ ———(5)(by 3 and 4).
$m'_2(L_2) \cap m^*_1(R_1 - p_1(L_1)) = \phi$ (by 5 and 2).

2. Assume that d'_2 is **weakly sequentially independent** of d_1 . i.e., $m'_2(L_2) \cap m^*_1(R_1 - p_1(L_1)) = \phi$, we have to prove that a match $m_2 : L_2 \to G$ exists for p_2 such that $m'_2 = p^*_1 \circ m_2$ and it is total.

First we construct the required match m_2.

For the vertices and edges which are deleted from G either implicitly or explicitly by application of p_1 are not present in H_1 so they do not belong to $m'_2(L_2)$, so we do not have to bother about them. For a vertex $v \in G$ s.t. $v \in m_1(p_1(L_1))$, v is preserved , consider the situation given in the diag1 below.

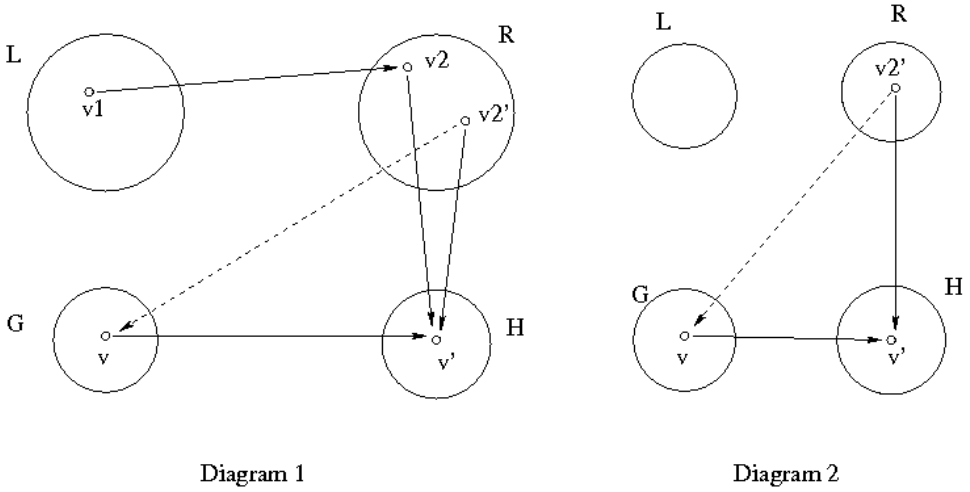


Diagram 1 Diagram 2

Figure : characterization of sequential independence

$p^*_1(v) = v' = m'_2(v'_2)$ for some $v'_2 \in L_2$ and $v' \in H_1$. Then $m_2(v'_2) = v$ i.e., $m_2(v'_2) = p^{*^{-1}}_1(m'_2(v'_2)))$.

For a vertex $v \in G$ s.t. $v \notin m_1(p_1(L_1))$, v is not mapped by m_1 but is simply copied to H_1 as shown in diag2 above. Then $m_2(v_2' = p_1^{*^{-1}}(m_2'(v_2'))$. For an edge or vertex which is generated by p_1 i.e., $v \in m_1^*(R_1 - p_1(L_1))$ and is in match of m_2'. but this violates the given condition therefore we do not have to consider this to construct map m_2.

By construction $m_2(L_2) \subseteq dom(p_1^*)$.

3. Assume that d_2' is **sequentially independent** of d_1 i.e.,

   (a) $m_2'(L_2) \cap m_1^*(R_1 - p_1(L_1)) = \phi$.

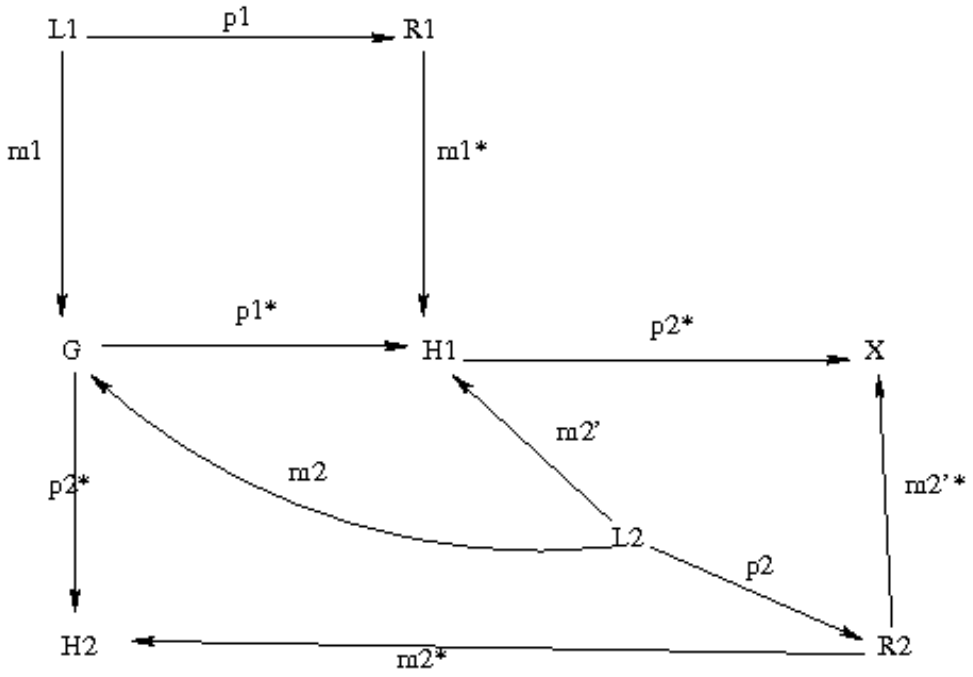
   (b) $m_2'(L_2 - dom(p_2)) \cap m_1^*(R_1) = \phi$.

   Then we have to prove that ,

   (a) d_2' is **weakly sequentially independent** of d_1.

   (b) d_1 is weakly parallel independent of the correspondingly existing derivation $d_2 = (p_2, m_2' : G \Rightarrow H_2)$. i.e., $m_1(L_1) \cap m_2(L_2 - dom(p_2)) = \phi$.

   First part follows by condition (a) of the assumption by definition of weak sequential indpendence. For the second part consider the following diagram described by above situation.



If second condition is not satisfied i.e., $m_1(L_1) \cap m_2(L_2 - dom(p_2)) \neq \phi$ then there must be some element $v \in m_1(L_1) \cap m_2(L_2 - dom(p_2))$ i.e., which means that v is needed by p_1 but deleted by p_2. This implies that v cannot be in domain of p_1^*. But by equality $m_2' = p_1^* \circ m_2$ the same element is deleted by m_2' i.e., $v \in m_2'(L_2 - dom(p_2))$. But we have that $v \in m_1(L_1) \Rightarrow v \in m_1^*(R_1)$ which violates (a) of our assumption.

4. Assume that

   (a) d_2' is **weakly sequentially independent** of d_1 i.e., $m_2'(L_2) \cap m_1^*(R_1 - p_1(L_1)) = \phi$.

   (b) d_1 is weakly parallel independent of the correspondingly existing derivation $d_2 = (p_2, m_2' : G \Rightarrow H_2)$. i.e., $m_1(L_1) \cap m_2(L_2 - dom(p_2)) = \phi$.

   Then we have to prove that , d_2' is **sequentially independent** of d_1 i.e.,

   (a) $m_2'(L_2) \cap m_1^*(R_1 - p_1(L_1)) = \phi$.

(b) $m'_2(L_2 - dom(p_2)) \cap m^*_1(R_1) = \phi$. i.e., p_2 do not delete any element which was generated by p_1.

First condition follows by (a) of the assumption. Assume that second condition is not satisfied then p_2 deletes some element generated by p_1. Let that element be v. Then $v \in m^*_1(R_1 - p_1(L_1))$ and $v \in m'_2(L_2 - dom(p_2))$ latter implies that $v \in m'_2(L_2)$. which clearly violates (a) of our assumption.

□

Lemma 3.2.5. (weak independence)
Given a direct derivation $d_1 = (p_1, m'_1 : G \Rightarrow H_1)$, the following statements are equivalent:

1. *There is a direct derivation $d_2 = (p_2, m'_2 : G \Rightarrow H_2)$ which is weakly parallel independent of d_1.*

2. *There is a direct derivation $d'_2 = (p_2, m'_2 : H_1 \Rightarrow X)$ which is weakly parallel independent of d_1. Up to isomorphism, a bijective correspondence between 1. and 2. above is given by by $m'_2 = p^*_1 \circ m_2$ and $m_2 = (p^*_1)^{-1} \circ m_1$.*

Proof. 1. and 2. are equivalent by two characterizations **(categorical characterisation of sequential independence)**, and **(categorical characterisation of weak parallel independence)** given above.

By **pushout properties** using pushouts preserve injectivity, i.e. p_1 injective implies p^*_1 injective we have $m_2 = (p^*_1)^{-1} \circ p^*_1 \circ m_2$. and $m'_2 = p^*_1 \circ (p^*_1)^{-1} \circ m_1$. □

Theorem 3.2.6. (Local Church Rosser) *Let $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ and $d_2 = (p_2, m_2 : G \Rightarrow H_2))$ be two direct derivations. Then the following statements are equivalent:*

1. *The direct derivations d_1 and d_2 are parallel independent.*

2. *There is a graph X and direct derivations $H_1 \Rightarrow^{p_2,m'_2} X$ and $H_2 \Rightarrow^{p_1,m'_1} X$ such that $G \Rightarrow^{p_1,m_1} H_1 \Rightarrow^{p_2,m'_2} X$ and $G \Rightarrow^{p_2,m_2} H_2 \Rightarrow^{p_1,m'_1} X$ are sequentially independent derivations.*

Up to isomorphism, a bijective correspondence between 1. and 2. above is given by by $m'_2 = p^*_1 \circ m_2$ and $m'_1 = p^*_2 \circ m_1$.

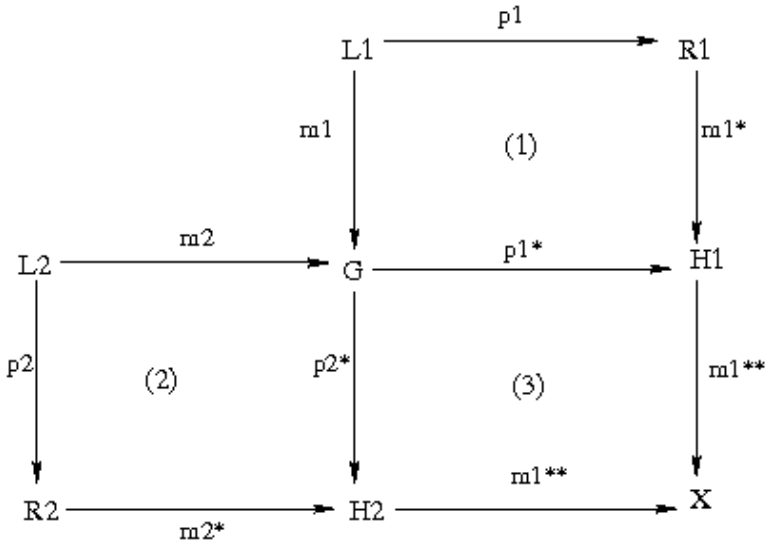


Figure : Local Church Rosser

Proof. Consider diagram given above. Subdiagrams (1) and (2) describes the pushouts corresponding derivations d_1 and d_2 respectively. Subdiagram (3) describes pushout for p_1^* and p_2^*. By composition of pushouts (2) and (3) we get pushout (2) + (3) which describes derivation for $H_1 \Rightarrow^{p_2,m_2'} X$ since we have $m_2' = p_1^* \circ m2$ by parallel independence of d_1 and d_2. Similarly we get derivation $H_2 \Rightarrow^{p_1,m_1'} X$ by combining (1) and (3). Now the equivalence of above two statements and isomorphism follows from **weak independence** lemma given above. □

3.2.2 Explicit Parallelism

Definition 3.2.7. (parallel production and derivations)
Given two productions $p_1 : L_1 \to R_1$ and $p_2 : L_2 \to R_2$, the parallel production $p_1 + p_2 : (L_1 + L_2 \to^p R_1 + R_2)$ is composed of the production name $p_1 + p_2$, i.e., the diagram above, and the associated partial morphism p. The graphs $(L_1 + L_2$ and $(R_1 + R_2$ (together with the corresponding injenctions $in_L^1, in_L^2, in_R^1, in_R^2$) are the coproducts of L_1, L_2 and R_1, R_2 respectively. The

partial morphism $L_1 + L_2 \to^p R_1 + R_2$ *is induced uniquely by the universal property of the coproduct* $L_1 + L_2$ *such that* $p \circ in_L^1 = in_R^1 \circ p_1$ *and* $p \circ in_L^2 = in_R^2 \circ p_2$. *The application of a parallel production* $p_1 + p_2$ *at a match* m *constitutes a* **direct parallel derivation**, *denoted by* $G \Rightarrow^{p_1+p_2,m} X$ *or* $(p_1 + p_2, m : G \Rightarrow X)$.

By referring to morphisms $m_1 = m \circ in_L^1$ and $m_2 = m \circ in_L^2$ we also write $p_1 + p_2, m_1 + m_2 : G \Rightarrow X$.

Theorem 3.2.8. (weak parallelism)
Given productions $p_1 : L_1 \to R_1$ *and* $p_2 : L_2 \to R_2$, *the following statements are equivalent:*

1. *There is a direct parallel derivation* $p_1 + p_2, m_1 + m_2 : G \Rightarrow X$ *such that* $p_2, m_2 : G \Rightarrow H_2$ *is weakly parallel independent of* $p_1, m_1 : G \Rightarrow H_1$.

2. *There is a weakly sequential independent derivation* $G \Rightarrow^{p_1,m_1} H_1 \Rightarrow^{p_2,m_2'} X$.

Up to isomorphism, a bijective correspondence between 1. *and* 2. *above is given by by* $m_2' = p_1^* \circ m_2$.

Proof. $(\mathbf{1}) \Rightarrow (\mathbf{2})$

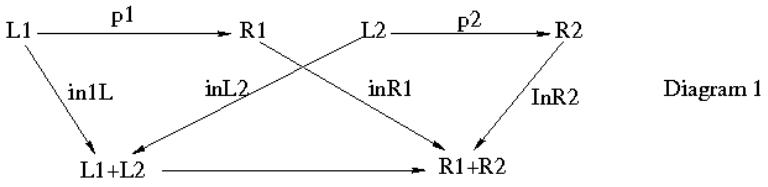


Diagram 1

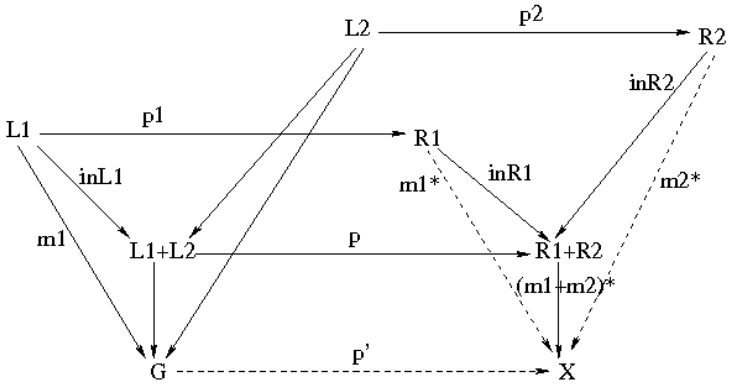


FIgure: weak parallelism

To construct a parallel derivation , in the first step we construct co-products for in_L^1 and in_L^2 as well as for in_R^1 and in_R^2 , then in second step we construct pushout of p and $m_1 + m_2$ i.e., to construct co-product of p_1, p_2, m_1, m_2 as shown in the diagram above.

We can construct co-product of p_1, p_2, m_1, m_2 in other way as shown in the figure, **Local Church Theorem**. Both of these co-product constructions coincide due to commutativity of co-products construction which ensures that all co-products can be iteratively obtained from composing partial co-products.

Therefore, the result of parallel derivation can be obtained from first constructing colimit(pushout) of p_1, m_1 and p_2, m_2 and second by constructing colimit(pushout) of p_1^* and p_2^* i.e., first to construct the direct derivations $d_1 = (p_1, m_1 : G \Rightarrow H_1)$ and $d_2 = (p_2, m_2 : G \Rightarrow H_2)$ represented by (1) and (2) as shown in the figure **Local Church Theorem**. but by assumption that d_2 is weakly parallel independent of d_1 we get a derivation $d_2' = (H_1 \Rightarrow^{p_2, m_2'} X)$ represented by (2) + (3). By **(categorical characterisation of sequential independence)** lemma , d_2' is weakly parallel

independent of d_1 as required.

$(\mathbf{2}) \Rightarrow (\mathbf{1})$

proof of this can be obtained by applying all the arguments in the reverse direction. □

Chapter 4

Conclusion

4.1 Summary

In this thesis we studied SPO and DPO approaches for graph transformations. As a part of comparison between these two approaches we saw how a DPO derivation can be embedded in SPO derivation and SPO to DPO under certain conditions. We also studied how these approaches are exploited to characterize and prove main results about parallel derivations.

4.2 Future Work

Both graph grammars and Petri nets are well-known specification formalisms for concurrent and distributed systems. Future work would be to study Petri net transformations using an algebraic or categorical approach.

.