Graph Transformations Using Relational Calculus

溝口, 佳寛

https://doi.org/10.11501/3063856

出版情報：九州大学, 1992, 博士（理学）, 論文博士
バージョン：
権利関係：

Graph Transformations Using Relational Calculus

1992年

溝口佳寛

(1)

Graph Transformations Using Relational Calculus

1992

Yoshihiro Mizoguchi

溝口佳寛

Abstract

Graph transformations are very common in many area of Computer Science and related fields, especially they are used in many different kinds of non-numeric computation. Since the early beginnings of Computer Science graphical representations have played a fundamental role to explain complex situations on an intuitive level. On the other hand, many people believe that graphs are only of limited use for internal machine representation because classical graph theory offers only little help and more graph algorithms are highly inefficient in general.

Though most graph algorithms are globally defined the basic idea of graph transformations is of local nature. This means that the transformation from one graph into another one is based on local changes where only certain subgraphs are transformed and the complement remains unchanged. By these features, several researches about concurrent and distributed computation based on graph transformations have been developed. Algebraic specification techniques based on term rewriting should be extended efficiently to graph transformations. So it can be applied to the area such as production systems, logical programming and analysis of programs. A general graph transformation system provides not only an efficient way to develop useful tools but also graph transformation approaches provide a new insight of algorithms about such as a decidability problem of graph properties. In this way, there are various applications of graph transformations.

This thesis is a collection of the theory about foundations of graph transformations. The research goal of the foundations is a comparative study, unification of the concepts developed in the different approaches and finding out new insights for applications.

It is well-known the "algebraic categorical approach" to graph grammars devel-

oped by Ehrig as a first algebraic fundamental theory about graph transformations. Recently, Raoult, Kennaway and Löwe proposed other graph structures and different but very closed formalizations of graph transformations. They proved the existence theorem of pushout which is a basic notation for transformations. In this thesis, we propose a categorical framework unifying several graph structures and analyze various properties about graph transformations. Further, we implement a graph transformation system based on a symbolical object which is called a graph term and consider their applications.

In Chapter 2, we summarize a theory of relational calculus which is a theoretical foundation in this thesis. Especially, the properties about partial functions and pushouts which play fundamental roles of graph transformations are investigated not only in the category of set and functions but also in the more general categories including toposes which are models of constructive logic.

In Chapter 3, we present a categorical framework to consider several graph structures universally. We show an existence condition of pushouts in the categories. That is, graphs in which out-edges from a node is ordered, graphs in which out-edges from a node is not ordered, and graphs in which do not have multiple edges between the same nodes are treated uniformly and proved the conditions of existence of pushouts.

In Chapter 4, we developed a theory about the category of simple graphs in which pushouts always exist so it is an adequate category for a model of graph transformations. We formalized the graph transformation and investigate the properties using relational calculus. A sufficient condition for two transformations to commute is shown. Further, we show the critical pair lemma holds like term rewriting system when we restrict some rewriting rules and matchings.

In Chapter 5, we introduce a graph rewriting system based on a model introduced in Chapter 4, using a symbolical object called a graph term. We precisely defined parallel rewritings for symbolical objects and show a sufficient condition for that a parallel rewriting is sequential simulatable. A problem finding out regular expressions of recognized languages from a graph expressing a finite automaton is solved using graph rewritings. Especially, we show those parallel rewritings which are not simulated

by sequential rewritings also lead a collect answer of the problem.

In Chapter 6, we propose a more general framework for graph transformations which is an extension of the theory discussed in Chapter 4. A hyper graph structure which has a label with arity for each hyper edge is included as an example. We extend our theory to the general framework and proved that pushouts always exist.

The thesis concludes with Chapter 7 in which the main results of the thesis, further related research topics, and remaining open problems are discussed.

Acknowledgments

I would like to express my sincere appreciation to Professor Yasuo Kawahara for his valuable advice, many constructive suggestions, and continual encouragement during the course of this study. I wishes to thank Professor Setsuo Arikawa for his helpful comments and encouragement.

I am grateful to Professor Toru Hitaka and Professor Nagata Furukawa for their valuable suggestions and criticisms.

I thank Professor Koichi Niijima for his kind support and encouragement. I also thank Professor Satoru Miyano, Hiroshi Ohtsuka, Masaya Nohmi, Satoru Kumamoto and Takayoshi Shoudai for their useful suggestions and helpful comments during discussions.

Finally, I am especially grateful to my parents and my wife Kyoko for their constant supports and encouragements.

Contents

Chapter 1

Introduction

There are many researches [Cou86, CM91, EKL90, EKR90, Ken87, Ken90, LE90, Miz92b, OH91, Rao84] on graph transformations which have a lot of applications including software specification, data bases, analysis of concurrent systems, developmental biology and many others. In these one of the advantages of categorical graph rewritings is to produce a universal reduction despite how to execute algorithms for applying production rules.

At first, we introduce two illustrative examples of application of graph transformations. The first example is a small relational data base of the information processing system of West Germany's police [Ger83, LE90]. A simple data base is expressed by a graph in Fig. 1.1. Nodes consists of special nodes PD(Person Data) and CD(Case Data), person's nodes which have an edge to PD and case's nodes which have an edge to CD. Edges represent relations between persons and cases. For example, edges s means "Person p is suspected in case c". A situation related over three persons or cases expressed by a hyper edge. Operations of the data base, such as adding a person(case), adding a relation of persons and cases, deleting a person(case) are given by graph transformation rules. For example, an operation deleting a person can be modeled by Fig. 1.2. The desired effect of object deletion is not only the removal of the object but also the elimination of all relations referring to this object (cf. Fig. 1.3). The graphical intuitional understanding is easy, but for discussing about properties about effects of a combination of operations we need a precise mathematical model of

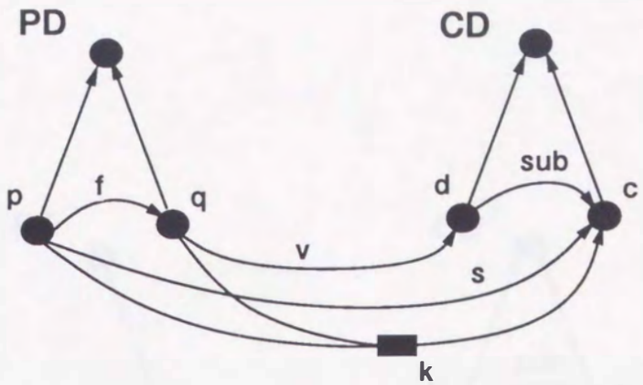


Figure 1.1: Small Police Data Base

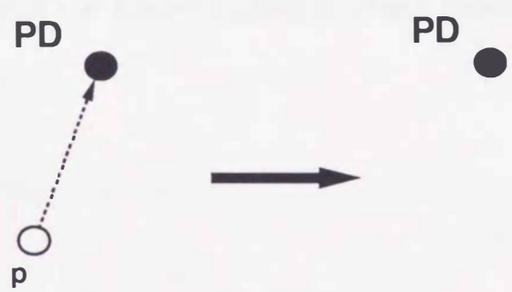


Figure 1.2: Graph Transformation Rule for Object Deletion

graph transformations in which we can get the desired object from the operation rules.

The second example is an application of graph transformations to a network reliability analysis [OH91]. A network model is a labeled graph in which each edge has a probability of connected states called a reliability (cf. Fig. 1.4). The set of three fundamental reliability-preserving transformation rules [KRB77, SW85] is shown in Fig. 1.5. They are serial reduction, parallel reduction and Δ-Y reduction, where P_{ij}, P_{ij}^k is the reliability of the edge between vertices i and j, $\bar{P_{ij}^k} = 1 - P_{ij}^k$, $x = P_{12} + P_{23}P_{31} - P_{12}P_{23}P_{31}$, $y = P_{23} + P_{31}P_{12} - P_{12}P_{23}P_{31}$, and $z = P_{31} + P_{12}P_{23} - P_{12}P_{23}P_{31}$. We can calculate the reliability between two points applying these rules (cf. Fig. 1.6). There exist problems about convergence and termination of transformations like a term rewriting system. Since these properties vary with a model of transformations. We need a widely applica-

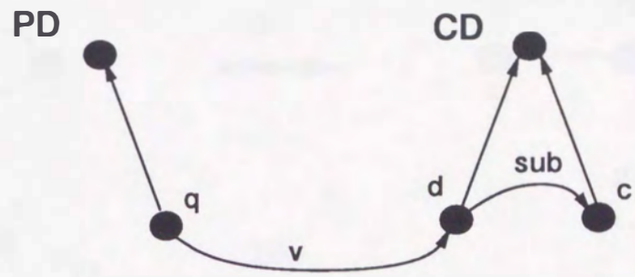


Figure 1.3: Intended Effect of Object Deletion

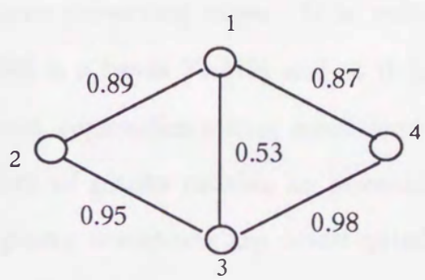


Figure 1.4: An Example of Networks

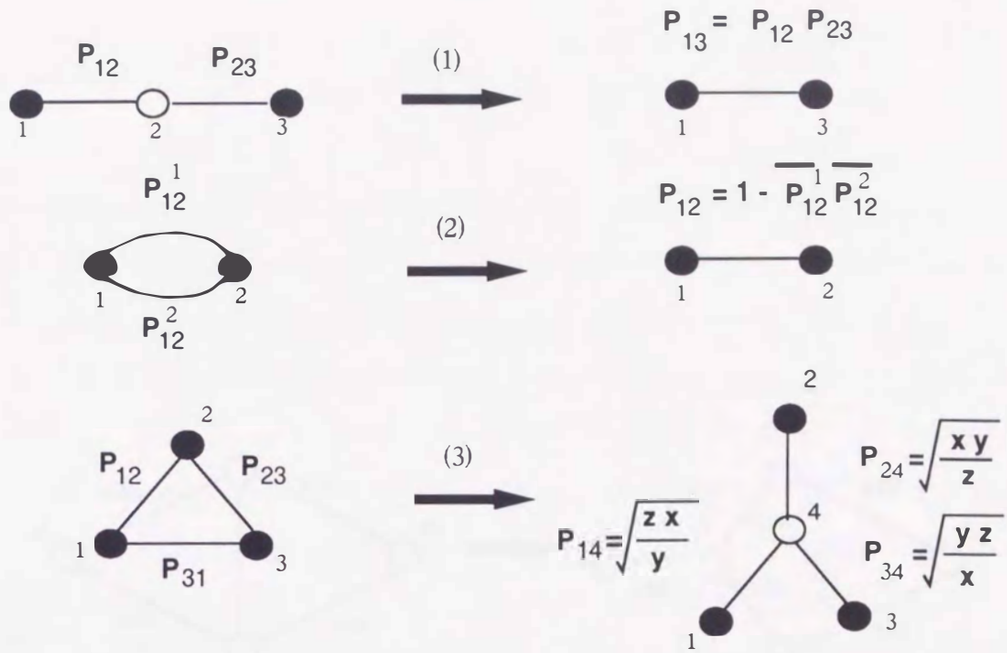


Figure 1.5: Network Transformation Rules

ble mathematical model of graph transformations which have good enough conditions to discuss about rewriting properties.

Next, we review theories of mathematical model of graph transformations. Ehrig et al. [ER80, EKL90] developed "algebraic approach" to graph grammars for a wide class of graphs and functions preserving edges. It is well-known that the category of graphs in [ER80, EKL90] is a topos [Gol79] and so it has pushouts [ML72, page 65]. In their double pushout approaches gluing conditions for existence of pushout-complements in the category of graphs provide an essential mean of controlling the semantics of rules. The gluing conditions are investigated by Ehrig and Kreowski [EK79] and Kawahara[Kaw90]. As the category of graphs is considered as a functor category over the category of sets and functions, it becomes a topos and has various useful properties. The existence theorem of pushout complements in a topos including the category of graph was generally proved by Kawahara[Kaw90].

Raoult [Rao84] and Kennaway [Ken87] proposed another formalization of graph rewritings by using a single pushout and regarding production rules as partial functions

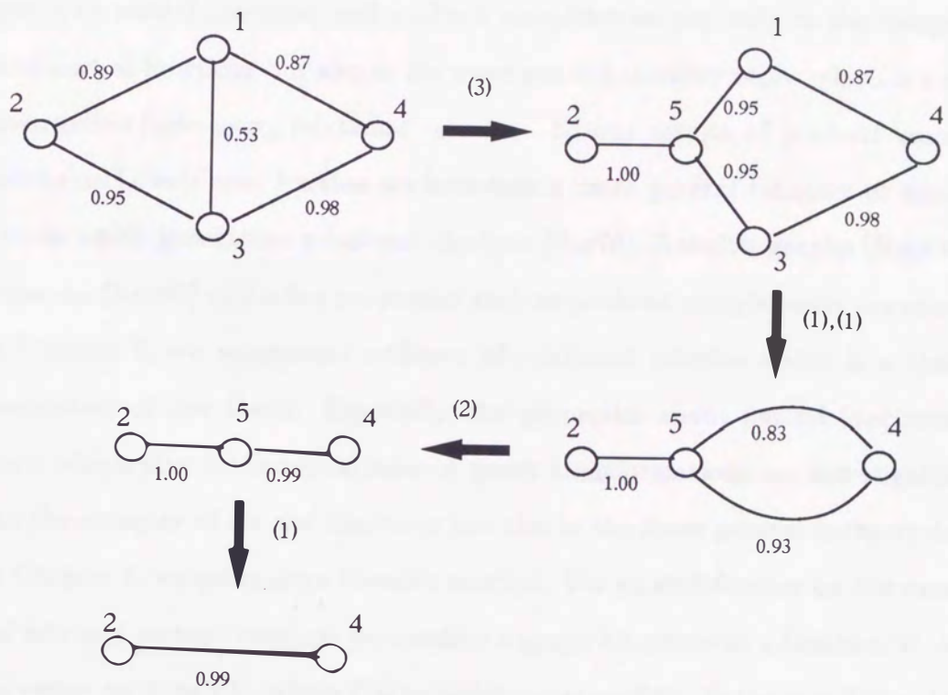


Figure 1.6: An Example of Network Reliability Calculation

preserving graph structures. They show the condition for existing pushouts in their models. We introduce a general framework of graph transformations which includes their formalization as an example. Our result of the condition for existing of pushouts contains theirs and we give another model which is pushout complete.

Recently Löwe [Löw89, LE90, EKL90] studied rewritings based on a single pushout in Sig-algebras and proved pushout completeness for restricted signatures with monadic operator symbols only. His category of Sig-algebras can be considered as a functor category over the category of sets and functions, so it is also a topos. We investigate the properties of partial functions and pushout completeness not only in the category of sets and partial functions but also in the more general category topos which is a model of constructive logic using relational calculus. So our results of pushout completeness contains Löwe's one. Further we introduce a more general category of relational structures which generalizes relational algebras [Bar70], Raoult's graphs [Rao84] and hypergraphs [Ken90] and a few properties such as pushout completeness are studied.

In Chapter 2, we summarize a theory of relational calculus which is a theoretical foundation of this thesis. Especially, the properties about partial functions and pushouts which play fundamental roles of graph transformations are investigated not only in the category of set and functions but also in the more general category topos.

In Chapter 3, we generalizes Raoult's method. For an endofunctor on the category **Pfn** of sets and partial functions we consider a graph structure as a function $V \to TV$ from a vertex set V to TV, where T is an endofunctor on **Pfn**. In the case $TV = V^*$ the structure is the same as Raoult's one. We prove an existence theorem of pushouts in our general settings avoiding many kind of conditional checks and case divisions by using the relational calculus. Our main result on the existence theorem of pushouts produces a modification of Raoult's result [Rao84, Proposition 5] which lacked a condition. A counter example to his result is given.

In Chapter 4 we treat the category of (simple) graphs (with or without labeled edges) and partial functions preserving graph structures, and present a new formalization of graph rewritings by using a primitive pushout construction in the category. Our graph rewritings can be always executed without any gluing conditions, only if a

graph has a matching to a given rewriting rule. Moreover our formalization of graph rewritings generalizes Ehrig's graph derivations [EK79, EKR90] and Raoult's graph rewritings [Rao84] in a reasonable sense. For a pair of partial functions from a common set into graphs a primitive pushout square is constructed, which shows the category of graphs and partial morphisms has pushouts. Moreover we give a more general sufficient condition for two graph rewritings to commute and a critical pair lemma of graph rewriting systems. We compare our approach with another approaches by Ehrig[EKL90], Löwe[LE90], Kennaway[Ken90] and Okada[OH91]. Some examples related to graph rewritings are listed. In the last of the chapter, we state how to develop our formalization of graph rewritings for graphs with labelled edges which contains Raoult's graphs[Rao84].

In Chapter 5, we introduce new notations for graphs and graph terms and the interpretation of graph terms are given. Using the notions, we define not only simple reduction but also a parallel reduction. The symbolical notation of graphs and reduction rules is helpful to define a reading region and a writing region precisely. These facts contribute to clear the concept of sequentially simulatable parallel reductions. We have a sufficient condition that holds parallel reductions are sequentially simulatable. Some properties about sequentially simulatable reduction are discussed. Turner's SK-reduction machine [Tur79] guarantees every parallel reduction induce a correct result. We define a graph rewriting rules corresponding to SK-reductions, we reconfirm the fact of parallel reductions in our framework. For a meaningful example of non sequentially simulatable reductions, we show a graph reduction system which solves equations of regular expressions. The reduction rules are not sequentially simulatable, but it is proved that every parallel reduction leads to the right solution of the equations.

In Chapter 6, we propose a more general framework for graph transformations which contains the category defined in Chapter 4. A hyper graph structure which have a label with arity for each hyper edge is included as an example. We also showed in the general framework that pushouts always exist using the properties of partial functions. Finally, we compare our category of relational structures together with partial morphisms to a category of partial morphisms in the sense [RR88, Ken90] and show that they are

isomorphic choosing a certain admissible class of monomorphisms.

In Chapter 7, we concludes our researches and discuss about several problems for developing the research.

Chapter 2

Relational calculus

In this chapter we summarize a theory of relational calculus which is a theoretical foundation of this thesis. In Section 2.1, we state basic notations and fundamental properties of relation in the category of sets and functions. In the category of sets and partial functions, we give a simple proof of existence of pushouts using relational calculus and several properties of pushout squares are showed. In Section 2.2, we extend the notions for topos which is a model of constractive logic and investigate the properties of partial functions and pushouts which play fundamental roles of graph transformations.

2.1 Basic notations

A *relation* α of a set A into another set B is a subset of the cartesian product $A \times B$ and denoted by $\alpha : A \rightharpoonup B$. The *inverse relation* $\alpha^\sharp : B \rightharpoonup A$ of α is a relation such that $(b, a) \in \alpha^\sharp$ if and only if $(a, b) \in \alpha$. he *composite* $\alpha\beta : A \rightharpoonup C$ of $\alpha : A \rightharpoonup B$ followed by $\beta : B \rightharpoonup C$ is a relation such that $(a, c) \in \alpha\beta$ if and only if there exists $b \in B$ with $(a, b) \in \alpha$ and $(b, c) \in \beta$.

As a relation of a set A into a set B is a subset of $A \times B$, the inclusion relation, union, intersection and difference of them are available as usual and denoted by $\sqsubseteq$, $\sqcup$, $\sqcap$ and $-$, respectively. The *identity relation* $\mathrm{id}_A : A \rightharpoonup A$ is a relation with $\mathrm{id}_A = \{(a, a) \in A \times A \mid a \in A\}$ (the diagonal set of A).

The followings are the basic properties of relations and indicate that the totality of sets and relations forms a category **Rel** with involution (or shortly I-category).

Proposition 2.1.1 (I-category) *Let $\alpha, \alpha' : A \multimap B$, $\beta, \beta' : B \multimap C$ and $\gamma : C \multimap D$ be relations. Then,*

(1) $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ (associative),

(2) $id_A\alpha = \alpha id_B = \alpha$ (identity),

(3) $\alpha^{\sharp\sharp} = \alpha$, $(\alpha\beta)^\sharp = \beta^\sharp\alpha^\sharp$ (involutive),

(4) *If $\alpha \sqsubseteq \alpha'$ and $\beta \sqsubseteq \beta'$, then $\alpha\beta \sqsubseteq \alpha'\beta'$ and $\alpha^\sharp \sqsubseteq \alpha'^\sharp$* (monotone).

The distributive law for relations is trivial but indispensable in our relational calculus.

Proposition 2.1.2 (Distributive Law) *The distributive law $\alpha(\bigsqcup_{\lambda\in\Lambda}\beta_\lambda)\gamma = \bigsqcup_{\lambda\in\Lambda}\alpha\beta_\lambda\gamma$ holds for relations $\alpha : A \multimap B$, $\beta_\lambda : B \multimap C$ $(\lambda \in \Lambda)$ and $\gamma : C \multimap D$.*

Proposition 2.1.3 (Law of Puppe-Calenko) *If $\alpha : A \multimap B$, $\beta : B \multimap C$ and $\gamma : A \multimap C$ are relations, then $\alpha\beta \sqcap \gamma \sqsubset \alpha(\beta \sqcap \alpha^\sharp\gamma)$.*

A *partial function* f of a set A into a set B is a relation $f : A \multimap B$ with $f^\sharp f \sqsubseteq \mathrm{id}_B$ and it is denoted by $f : A \to B$. A *(total) function* f of a set A into a set B is a relation $f : A \multimap B$ with $f^\sharp f \sqsubseteq \mathrm{id}_B$ and $\mathrm{id}_A \sqsubseteq ff^\sharp$, and it is also denoted by $f : A \to B$. Clearly a function is a partial function. A function $f : A \to B$ is injective if $ff^\sharp = \mathrm{id}_A$ and surjective if $f^\sharp f = \mathrm{id}_B$. Note that the identity relation id_A of a set A is a function. The readers easily understand our definitions of partial functions and (total) functions are coincide with ordinary ones.

A singleton set $\{*\}$ is denoted by 1 and the maximum relation from a set A into 1 by $\Omega_A : A \multimap 1$, that is, $\Omega_A = \{(a, *) | a \in A\}$. For a partial function $f : A \to B$ a relation $f^\sharp\Omega_A : B \to 1$ and $f\Omega_B : A \to 1$ corresponds to the image $\mathrm{Im}(f)$ and $\mathrm{dom}(f)$ of f respectively.

Proposition 2.1.4 *Let $\alpha, \beta : A \rightharpoonup B$ be relations. If $f : X \to A$ and $g : Y \to B$ are partial functions, then $f(\alpha \sqcap \beta)g^\sharp = f\alpha g^\sharp \sqcap f\beta g^\sharp$ and $f(\alpha - \beta)g^\sharp = f\alpha g^\sharp - f\beta g^\sharp$.* ■

Given a relation $\alpha : A \rightharpoonup B$, the *domain* $d(\alpha) : A \rightharpoonup A$ of α is a relation defined by $d(\alpha) = \alpha\alpha^\sharp \sqcap \mathrm{id}_A$. A partial function $f : A \to B$ is a function if and only if $d(f) = \mathrm{id}_A$.

The following propositions are useful for manipulating domains of partial functions.

Lemma 2.1.5 *Let $f : A \to B$, $g : A \to B$, $h : B \to C$ be partial functions.*

(1) *f is a total function if and only if $f\Omega_B = \Omega_A$.*

(2) *If $f \subset g$ and $f\Omega_B = g\Omega_B$ then $f = g$.*

(3) *If $\alpha \sqsupset \beta$, then $f(\alpha - \beta) = f\alpha - f\beta$, where $\alpha, \beta : B \rightharpoonup C$.*

Proposition 2.1.6 *Let $\alpha : A \rightharpoonup B$ and $\beta : B \rightharpoonup C$ be relations and $f : A \to B$ a partial function. Then*

(1) $\mathrm{d}(\alpha) \subset \mathrm{d}(\alpha')$ *iff* $\mathrm{dom}(\alpha) \subset \mathrm{dom}(\alpha')$ *iff* $\alpha\Omega_B \subset \alpha'\Omega_B$.

(2) *$d(\alpha\beta)d(\alpha) = d(\alpha\beta)$ (or $d(\alpha\beta) \sqsubseteq d(\alpha)$),*

(3) *$d(f\beta)f = fd(\beta)$.*

■

Proposition 2.1.7 *Let $\alpha : A \rightharpoonup A$, $\theta : B \rightharpoonup B$ be relations and let $f : A \to B$ be a partial function. If $\theta \sqsubseteq f^\sharp \alpha f$, then $\theta = f^\sharp f \theta f^\sharp f$.* ■

2.2 Category of partial functions

We denote the category of sets and functions by **Set** and the category of sets and partial functions by **Pfn**. Both of **Set** and **Pfn** have all small limits and colimits, so in particular, they have pushouts [ML72, Rao84, Ken87, Miz92b]. Note that **Pfn** is equivalent to the category of sets with a base point (a selected element) and base point preserving functions.

Fact 2.2.1 *The category* **Pfn** *has pushouts.*

Let

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

be a pushout square in **Pfn**. *For any functions* $x : B \to S$, $y : C \to S$ *satisfying* $fx = gy$, *there exists a unique function* $t : D \to S$ *such that* $ht = x$ *and* $ht = y$, *where* $t = h^\sharp x \cup k^\sharp y$.

There are many proofs of existence of pushouts in **Pfn** [Rao84, Löw89, Miz92b, KMb]. In this paper, we introduce a simple proof using relational calculus as follows.

Let $B+C$ be the disjoint union of B and C and $i_B : B \to B+C$ and $i_C : C \to B+C$ inclusion functions, where $i_B^\sharp i_B \cup i_C^\sharp i_C = \mathrm{id}_{B+C}$, $i_B i_C^\sharp = \Phi_{BC}$ and $i_C i_B^\sharp = \Phi_{CB}$. An injective function $i : \mathrm{dom}(f) \cap \mathrm{dom}(g) \to A$ is a function which satisfies $i^\sharp i = \mathrm{d}(f) \cap \mathrm{d}(g)$. Let $\tilde{f} = ifi_B$, $\tilde{g} = igi_C$, $\theta = \tilde{f}^\sharp \tilde{g} \cup \tilde{g}^\sharp \tilde{f}$, and $\rho = \cup_{n \geq 0} \theta^n$. Since $\rho : B + C \to B + C$ is an equivalence relation (i.e. $\mathrm{id}_{B+C} \subset \rho$, $\rho^\sharp \subset \rho$ and $\rho\rho \subset \rho$), there exist a surjective function $e : B + C \to E$ with $ee^\sharp = \rho$. Let $\gamma_0 = \cup\Gamma$ where $\Gamma = \{\gamma : E \to 1 : fi_B e\gamma = gi_C e\gamma\}$. There exists an injective function $e_0 : D \to E$ with $\gamma_0 = e_0^\sharp \Omega_D$. We define $h = i_B e e_0^\sharp$ and $k = i_C e e_0^\sharp$ and prove that they make a pushout square in **Pfn** using following lemma.

$$\begin{array}{ccccccccc} & & & & B & & & & \\ & & & \nearrow^f & \downarrow{\scriptstyle i_B} & & & & \\ \mathrm{dom}(f) \cap \mathrm{dom}(g) & \xrightarrow{i} & A & & B + C & \xrightarrow{e} & E & \xleftarrow{e_0} & D \\ & & & \searrow_g & \uparrow{\scriptstyle i_C} & & & & \\ & & & & C & & & & \end{array}$$

Lemma 2.2.2 (1) $\tilde{f}e = \tilde{g}e$.

(2) $\mathrm{d}(fh) \subset \mathrm{d}(f) \cap \mathrm{d}(g)$ *and* $\mathrm{d}(gk) \subset \mathrm{d}(f) \cap \mathrm{d}(g)$.

(3) $fh = gk$

(4) *If a partial function* $\tilde{t} : B + C \to S$ *satisfies* $\tilde{f}\tilde{t} = \tilde{g}\tilde{t}$, *then* $ee^\sharp\tilde{t} = \tilde{t}$.

(5) $h^\sharp h \cup k^\sharp k = id_D$.

(6) *If partial functions* $x : B \to S$ *and* $y : C \to S$ *satisfies* $fx = gy$, *then there exists a unique partial function* $t : D \to S$ *such that* $ht = x$ *and* $kt = y$ *where* $t = h^\sharp x \cup k^\sharp y$.

(7) $\gamma_0 = \Omega_E - \{e^\sharp i_B^\sharp f^\sharp(f\Omega_B - g\Omega_C) \cup e^\sharp i_C^\sharp g^\sharp(g\Omega_C - f\Omega_B)\}$

(8) $(B - f(A)) \subset \mathrm{dom}(h)$ *and* $(C - g(A)) \subset \mathrm{dom}(k)$.

(*Proof.*) (1) Since $\tilde{f}$ and $\tilde{g}$ are total functions and $\tilde{f}^\sharp\tilde{g} \cup \tilde{g}^\sharp\tilde{f} \subset ee^\sharp$, we have $\tilde{f}e \subset \tilde{g}\tilde{g}^\sharp\tilde{f}e \subset \tilde{g}ee^\sharp e \subset \tilde{g}e$. Similary we have $\tilde{g}e \subset \tilde{f}e$ and so $\tilde{f}e = \tilde{g}e$.

(2) $fi_Be\gamma_0 = fi_Be(\cup_{\gamma\in\Gamma}\gamma) = \cup_{\gamma\in\Gamma}(fi_Be\gamma) = \cup_{\gamma\in\Gamma}(gi_Ce\gamma) = gi_Ce(\cup_{\gamma\in\Gamma}\gamma) = gi_Ce\gamma_0$. So $fh\Omega_D = fi_Bee_0^\sharp\Omega_D = gi_Cee_0^\sharp\Omega_D\ gh\Omega_D$ and $\mathrm{d}(fh) = \mathrm{d}(gk) \subset \mathrm{d}(f) \cap \mathrm{d}(g)$.

(3) Since $\mathrm{d}(fh) = \mathrm{d}(gk) \subset \mathrm{d}(f) \cap \mathrm{d}(g) = i^\sharp i$, $fh = fi_Bee_0^\sharp = i^\sharp ifi_Bee_0^\sharp = i^\sharp igi_Bee_0^\sharp = gk$.

(4) Since $\tilde{f}^\sharp\tilde{g}\tilde{t} = \tilde{f}^\sharp\tilde{f}\tilde{t} \subset \tilde{t}$ and $\tilde{g}^\sharp\tilde{f}\tilde{t} \subset \tilde{t}$, we have $\theta\tilde{t} \subset \tilde{t}$. If $\theta^{n-1}\tilde{t} \subset \tilde{t}$ then $\theta^n\tilde{t} \subset \theta\theta^{n-1}\tilde{t} \subset \theta\tilde{t} \subset \tilde{t}$. So we have $\theta^n\tilde{t} \subset \tilde{t}$ for any $n = 1, 2, \cdots$, and $\tilde{t} \subset \rho\tilde{t} \subset \tilde{t}$.

(5) $h^\sharp h \cup k^\sharp k = e_0e^\sharp(i_B^\sharp i_B \cup i_C^\sharp i_C)ee_0^\sharp = e_0e^\sharp ee_0^\sharp = \mathrm{id}_D$.

(6) If there exist a partial function $t : D \to S$ with $ht = x$ and $kt = y$, then $t = (h^\sharp h \cup k^\sharp k)t = h^\sharp x \cup k^\sharp y$. Let $\tilde{t} = i_B^\sharp x \cup i_C^\sharp y$. It is easy to check $\tilde{t}$ is a partial function. Since $\tilde{f}\tilde{t} = ifi_B(i_B^\sharp x \cup i_C^\sharp y) = ifx = igy = igi_C(i_B^\sharp x \cup i_C^\sharp y) = \tilde{g}\tilde{t}$, we have $ee^\sharp\tilde{t} = \tilde{t}$ by (4). Since $t = e_0e^\sharp\tilde{t}$, $t^\sharp t = \tilde{t}^\sharp ee_0^\sharp e_0e^\sharp\tilde{t} = \tilde{t}^\sharp ee^\sharp\tilde{t} = \tilde{t}^\sharp\tilde{t} \subset \mathrm{id}_S$. So t is a partial function. Before we show $ht = x$ and $kt = y$, we check $\mathrm{d}(e^\sharp\tilde{t}) \subset \mathrm{d}(e_0^\sharp)$ that is $e_0^\sharp e_0e^\sharp\tilde{t} = e^\sharp\tilde{t}$. Let $\gamma = e^\sharp\tilde{t}\Omega_S$. Since $fi_Be\gamma = fi_Bee^\sharp\tilde{t}\Omega_S = fi_B\tilde{t}\Omega_S = gi_C\tilde{t}\Omega_S = gi_Cee^\sharp\tilde{t}\Omega_S = gi_Ce\gamma$, we have $\gamma \subset \gamma_0$, $\mathrm{d}(e^\sharp\tilde{t}) \subset \mathrm{d}(e_0^\sharp)$ and $e_0^\sharp e_0e^\sharp\tilde{t} = e^\sharp\tilde{t}$. Finaly we have $ht = h(h^\sharp x \cup k^\sharp y) = i_Bee_0^\sharp(e_0e^\sharp i_B^\sharp x \cup e_0e^\sharp i_C^\sharp y) = i_Bee_0^\sharp e_0e\tilde{t} = i_Bee^\sharp\tilde{t} = i_B\tilde{t} = i_B(i_B^\sharp x \cup i_C^\sharp y) = x$ and similary $kt = y$.

(7) Let $\gamma_1 = \Omega_E - \{e^\sharp i_B^\sharp f^\sharp(f\Omega_B - g\Omega_C) \cup e^\sharp i_C^\sharp g^\sharp(g\Omega_C - f\Omega_B)\}$. Since $\gamma_1 \cap e^\sharp i_B^\sharp f^\sharp(f\Omega_B - g\Omega_C) = \phi$, we have $fi_Be\gamma_1 \cap (f\Omega_B - g\Omega_C) = \phi$ by Proposition 2.1.3 and $fi_Be\gamma_1 \subset$

$f\Omega_B \cap g\Omega_C = i^\sharp i\Omega_A$. So $fi_B e\gamma_1 = i^\sharp i fi_B e\gamma_1$. Similarry $gi_C e\gamma_1 = i^\sharp i gi_C e\gamma_1$. So we have $fi_B e\gamma_1 = gi_C e\gamma_1$ and $\gamma_1 \subset \gamma_0$.

Let $\gamma : E \rightharpoonup 1$ be an arbitrary relation satisfying $fi_B e\gamma = gi_C e\gamma$. Then $\gamma \cap e^\sharp i_B^\sharp f^\sharp(f\Omega_B - g\Omega_C) \subset e^\sharp i_B^\sharp f^\sharp(fi_B e\gamma \cap (f\Omega_B - g\Omega_C)) = e^\sharp i_B^\sharp f^\sharp(gi_C e\gamma \cap (f\Omega_B - g\Omega_C))$ $\subset e^\sharp i_B^\sharp f^\sharp(g\Omega_C \cap (f\Omega_B - g\Omega_C)) = \phi$ and we obtain $\gamma \subset \Omega_E - e^\sharp i_B^\sharp f^\sharp(f\Omega_B - g\Omega_C)$. Similarry $\gamma \subset \Omega_E - e^\sharp i_C^\sharp g^\sharp(g\Omega_C - f\Omega_B)$. So we have $\gamma \subset \gamma_1$ and $\gamma_0 \subset \gamma_1$.

(8) Since $\rho(i_B^\sharp f^\sharp\Omega_A \cup i_C^\sharp g^\sharp\Omega_A) \subset (i_B^\sharp f^\sharp\Omega_A \cup i_C^\sharp g^\sharp\Omega_A)$, we have $i_B\rho(i_B^\sharp f^\sharp\Omega_A \cup i_C^\sharp g^\sharp\Omega_A) \subset f^\sharp\Omega_A$.

Since

$$\begin{aligned} h\Omega_D &= i_B e\gamma_0 \\ &= i_B e\Omega_E - i_B e\{e^\sharp i_B^\sharp f^\sharp(f\Omega_B - g\Omega_C) \cup e^\sharp i_C^\sharp g^\sharp(g\Omega_C - f\Omega_B)\} \\ &\supset i_B e\Omega_E - i_B\rho\{i_B^\sharp f^\sharp\Omega_A \cup i_C^\sharp g^\sharp\Omega_A\} \\ &\supset \Omega_B - f^\sharp\Omega_A, \end{aligned}$$

we obtain $\mathrm{dom}(h) \supset (B - f(A))$. Similarly we have $\mathrm{dom}(k) \supset (C - g(A))$.

■

Proposition 2.2.3 *Let a square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

be a pushout in **Pfn** *and let* $t : X \to C$ *be a function. Then the composite* $tk : X \to D$ *is a function if and only if* $\mathrm{Im}(t) \sqcap \mathrm{Im}(g) \sqsubseteq \mathrm{dom}(k) \sqcap \mathrm{Im}(g)$. ■

Proposition 2.2.4 *Let a square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

be a pushout in **Pfn**. *If* g *is a total injective function, then* h *is a total injective function.* ■

Lemma 2.2.5 *Let A, B, C and X be sets with $B \subset A$ and $B \subset C$, $i : B \to A$ and $j : B \to C$ inclusion functions, $g : A \to X$ a function. If $gg^\sharp\Omega_A \subset ii^\sharp\Omega_A$, $X \cap (C - B) = \phi$ and the square*

$$\begin{array}{ccc} A & \xrightarrow{f} & C \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ X & \xrightarrow[k]{} & Y \end{array}$$

is a pushout in the category **Pfn**, *then $Y \cong (X - g(A)) \cup g(B) \cup (C - B)$, where $f = i^\sharp j$.*

■

2.3 Relational calculus in toposes

In this section we state a part of the relational calculus in elementary toposes, that is, some basic properties on (binary) relations and partial functions. For the details of relational calculus the reader is referred to M.S. Calenko [Cal77, CGR84] and Y. Kawahara [Kaw73, Kaw90].

Let $\mathbf{E}$ be an elementary topos [Gol79, Joh77]. A morphism f of $\mathbf{E}$ is denoted by $f : X \to Y$ when it has domain X and codomain Y. The composite of two morphisms $f : X \to Y$ and $g : Y \to Z$ of $\mathbf{E}$ is written as $fg : X \to Z$. A span (f, g) in $\mathbf{E}$ is a pair of morphisms $x : Z \to X$ and $y : Z \to Y$ in $\mathbf{E}$ with a common domain. Given a span (f, g) in $\mathbf{E}$ there exists a unique morphism $h : Z \to X \times Y$ in $\mathbf{E}$ such that $hp = f$ and $hq = g$, where $p : X \times Y \to X$, $q : X \times Y \to Y$ are projections of a product $X \times Y$. We denote such a unique h by $x \top y$. By the basic property of toposes every morphism $f : X \to Y$ can be uniquely decomposed into a composite of an epimorphism $e(f) : X \to I$ followed by a monomorphism $m(f) : I \to Y$ up to isomorphisms. The subsequent arguments of this paper will be carried out in a fixed elementary topos $\mathbf{E}$.

A *relation* α from an object X to another object Y in $\mathbf{E}$, denoted by $\alpha : X \multimap Y$, is a subobject of the product $X \times Y$. Every span (f, g) with $f : R \to X$ and $g : R \to Y$ induces a relation $[m(f \top g)] : X \multimap Y$, which will be simply denoted by $[f, g]$. Recall that the subobject $[m(f \top g)]$ of $X \times Y$ presents an equivalence class of a monomorphism

$m(f \top g)$. It is trivial that each relation α can be written as $\alpha = [f, g]$ with some span (f, g). We usually identify a morphism $f : X \to Y$ with a relation $[\mathrm{id}_X, f]$, where id_X is the identity morphism of X. Remark that $f = [\mathrm{id}_X, f] = [\mathrm{id}_X \top f]$ since $\mathrm{id}_X \top f$ is a monomorphism.

The composite $\alpha\beta (= \alpha \cdot \beta) : A \multimap C$ of a relation $\alpha : A \multimap B$ followed by a relation $\beta : B \multimap C$ is defined as follows: Assume that $\alpha = [f, g]$ and $\beta = [h, k]$. Construct a pullback

$$\begin{array}{ccc} \cdot & \xrightarrow{h'} & \cdot \\ {\scriptstyle g'}\downarrow & & \downarrow{\scriptstyle g} \\ \cdot & \xrightarrow[h]{} & B \end{array}$$

and define $\alpha\beta = [h'f, g'k]$. Also the inverse $\alpha^\sharp : B \multimap A$ of α is defined by $\alpha^\sharp = [g, f]$. These definitions of the composition and the inverse of relations are well-defined. It is easy to see that $[f, g] = [f, \mathrm{id}][\mathrm{id}, g] = f^\sharp g$.

These definitions are natural extensions of composite of functions and relations in the category **Set** of sets and functions.

Example 2.3.1 *A relation $\alpha : A \multimap B$ in* **Set** *is a subset of $A \times B$. The inverse relation $\alpha^\sharp : B \multimap A$ is a subset $\{(b, a) \in B \times A \mid (a, b) \in \alpha\}$. The composite $\alpha \cdot \beta : A \multimap C$ of two relations $\alpha : A \multimap B$ and $\beta : B \multimap C$ is a subset $\{(a, c) \mid (a, b) \in \alpha$ and $(b, c) \in \beta$ for some $b \in B\}$. A function $f : X \to Y$ is considered as a relation $\{(x, y) \in X \times Y \mid y = f(x)\}$ from X to Y.*

Since the set $\mathbf{Rel}(A, B)$ of all relations from A to B coincides with the set of all subobjects of $A \times B$, the ordering of relations in $\mathbf{Rel}(A, B)$ is the same as the ordering $\sqsubseteq$ of subobjects of $A \times B$. Note that $[f, g] \sqsubseteq [f', g']$ if and only if there exist an epimorphism e and a morphism s such that $ef = sf'$ and $eg = sg'$.

Let $f : X \to Y$ be a morphism of $\mathbf{E}$ and $xf = yf$ a pullback. Then $f^\sharp f = [f, f] \sqsubseteq [\mathrm{id}_Y, \mathrm{id}_Y] = \mathrm{id}_Y$ since $f \top f = f(\mathrm{id}_Y \top \mathrm{id}_Y)$, and $\mathrm{id}_X = [\mathrm{id}_X, \mathrm{id}_X] \sqsubseteq [x, y] = [\mathrm{id}_Y, f][\mathrm{id}_Y, f] = ff^\sharp$ since there exists a unique morphism z such that $\mathrm{id}_X \top \mathrm{id}_X = z(x \top y)$. We recall that a relation $\alpha : X \multimap Y$ satisfies $\alpha^\sharp \alpha \sqsubseteq \mathrm{id}_Y$ and $\mathrm{id}_X \sqsubseteq \alpha\alpha^\sharp$ if and only if there exists a unique morphism $f : X \to Y$ such that $\alpha = [\mathrm{id}_X, f]$.

The terminal object of $\mathbf{E}$ will be denoted by 1 and its initial object by 0. The maximum relation $\Theta_{XY} : X \rightarrowtail Y$ is $[\mathrm{id}_{X\times Y}]$ and the minimum relation $0_{XY} : X \rightarrowtail Y$ is $[i_{X\times Y}](=[i_X, i_Y])$, where $i_X : 0 \to X$ is a unique morphism from initial object 0. In particular we write $\Omega_X = \Theta_{X1}$ (i.e. $\Omega_X = [\mathrm{id}_X, !_X](=!_X)$, where $!_X : X \to 1$ is unique morphism into terminal object 1).

We now state five fundamental properties of relations in an elementary topos without proof:

2.3.2 (I-category) *Let $\alpha, \alpha' : A \rightarrowtail B$, $\beta, \beta' : B \rightarrowtail C$ and $\gamma : C \rightarrowtail D$ be relations. Then*

(a) $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ *(associative),*

(b) $id_A\alpha = \alpha id_B = \alpha$ *(identity),*

(c) $\alpha^{\sharp\sharp} = \alpha$, $(\alpha\beta)^\sharp = \beta^\sharp\alpha^\sharp$ *(involutive),*

(d) *If $\alpha \sqsubseteq \alpha'$ and $\beta \sqsubseteq \beta'$, then $\alpha\beta \sqsubseteq \alpha'\beta'$ and $\alpha^\sharp \sqsubseteq \alpha'^\sharp$ (monotone).*

2.3.3 (Heyting Algebra) $\mathbf{Rel}(A, B)$ *is a Heyting algebra for all objects A and B. That is, it is a lattice with the minimum element $0_{A,B}$, the maximum element $\Theta_{A,B}$, and pseudo-complements.*

The infimum and the supremum of relations $\alpha, \beta : X \rightarrowtail Y$ are written as $\alpha \sqcap \beta$ and $\alpha \sqcup \beta$, respectively. Also the pseudo-complement of $\alpha : X \rightarrowtail Y$ relative to $\beta : X \rightarrowtail Y$ is denoted by $\alpha \Rightarrow \beta$.

2.3.4 (Law of Puppe-Calenko) *If $\alpha : A \rightarrowtail B, \beta : B \rightarrowtail C$ and $\gamma : A \rightarrowtail C$ are relations, then $\alpha\beta \sqcap \gamma \sqsubseteq \alpha(\beta \sqcap \alpha^\sharp\gamma)$ is valid.*

2.3.5 (Rationality) *For each relation $\alpha : A \rightarrowtail B$ there exists a pair of morphisms $f : X \to A$ and $g : X \to B$ such that $\alpha = f^\sharp g$ and $ff^\sharp \sqcap gg^\sharp = id_X$.*

2.3.6 (Distributive Law) *Let Λ be a set. If $\mathbf{E}$ has coproducts of all Λ-indexed families of objects, then the distributive law $\alpha(\sqcup_{\lambda\in\Lambda}\beta_\lambda) = \sqcup_{\lambda\in\Lambda}\alpha\beta_\lambda$ holds for relations $\alpha : A \rightarrowtail B$ and $\beta_\lambda : B \rightarrowtail C$ $(\lambda \in \Lambda)$.*

The following is an elementary result of relations deduced from the last fundamental properties.

Lemma 2.3.7 *Let $\alpha : X \multimap Y$, $\beta : X \multimap Z$ be relations in* $\mathbf{E}$ *and V,W objects of* $\mathbf{E}$. *Then*

(1) $\alpha 0_{YV} = 0_{XV}$. *and* $0_{WX}\alpha = 0_{WY}$.

(2) $\Omega_X \Omega_Y^\sharp = \Theta_{XY}$.

(3) $\alpha\Omega_Y \sqsubseteq \beta\Omega_Z$ *if and only if* $\alpha \sqsubseteq \beta\beta^\sharp\alpha$.

(4) $\alpha \sqsubseteq \alpha\alpha^\sharp\alpha$.

(5) *If* $\alpha^\sharp\beta = 0_{YZ}$ *and* $\alpha\Omega_Y \sqsubseteq \beta\Omega_Z$, *then* $\alpha = 0_{XY}$.

(6) *If* $\alpha\Omega_Y = 0_{X1}$, *then* $\alpha = 0_{XY}$.

Proof. (a) Let $\alpha = [f, g]$. Since $\mathbf{E}$ is a cartesian closed category with finite limits, the square

$$\begin{array}{ccc} 0 & \xrightarrow{\mathrm{i}} & \cdot \\ {\scriptstyle \mathrm{id}_0}\downarrow & & \downarrow{\scriptstyle g} \\ 0 & \xrightarrow[\mathrm{i}_Y]{} & Y \end{array}$$

is a pullback. Then $\alpha 0_{YV} = [\mathrm{i}f, \mathrm{id}_0\mathrm{i}] = [\mathrm{i}_X, \mathrm{i}_V] = 0_{XV}$.

(b) Let $p : X \times Y \to X$ and $q : X \times Y \to Y$ be projections. Then the square $p!_X = q!_Y$ is a pullback and so $\Omega_X\Omega_Y^\sharp = [p, q] = [\mathrm{id}_{X\times Y}] = \Theta_{XY}$.

(c) Firstly assume $\alpha \sqsubseteq \beta\beta^\sharp\alpha$. Then $\alpha\Omega_Y \sqsubseteq \beta\beta^\sharp\alpha\Omega_Y \sqsubseteq \beta\Omega_Z$. Secondly assume $\alpha\Omega_Y \sqsubseteq \beta\Omega_Z$. Then we have $\alpha = \alpha \sqcap \alpha\Theta_{YY} = \alpha \sqcap \alpha\Omega_X\Omega_Y^\sharp$ (by (b)) $\sqsubseteq \alpha \sqcap \beta\Omega_Z\Omega_Y^\sharp = \alpha \sqcap \beta\Theta_{ZY}$ (by (b)) $\sqsubseteq \beta(\beta^\sharp\alpha \sqcap \Theta_{ZY})$ (by Law of Puppe-Calenko) $= \beta\beta^\sharp\alpha$.

(d) is immediate from (c) since $\alpha\Omega_Y = \alpha\Omega_Y$.

(e) It follows from (c) that $\alpha \sqsubseteq \beta\beta^\sharp\alpha$. Thus the result is obvious.

(f) is a particular case of (b) when $\beta = 0_{XY}$. ∎

A relation $f : X \multimap Y$ in $\mathbf{E}$ is called a *partial function* if it satisfies $f^\sharp f \sqsubseteq \mathrm{id}_Y$. We denote the category of objects in $\mathbf{E}$ and partial functions in $\mathbf{E}$ by $\mathbf{Pfn}(\mathbf{E})$.

Proposition 2.3.8 *Let $f, g : X \to Y$ be partial functions in* $\mathbf{E}$ *. Then*

(1) $ff^\sharp f = f$.

(2) *If* $f \sqsubseteq g$ *and* $f\Omega_Y = g\Omega_Y$, *then* $f = g$.

Proof. (a) follows from $f \sqsubseteq ff^\sharp f$ (by 2.3.7(d)) $\sqsubseteq f$ since $f^\sharp f \sqsubseteq \mathrm{id}_Y$.

(b) Assume $f \sqsubseteq g$ and $f\Omega_Y = g\Omega_Y$. Then we have $g \sqsubseteq ff^\sharp g$ (by 2.3.7(c)) $\sqsubseteq fg^\sharp g$ (by $f \sqsubseteq g$) $\sqsubseteq f$ (by $g^\sharp g \sqsubseteq \mathrm{id}_Y$). ■

Let X be an object in $\mathbf{E}$. A relation $p : X \rightharpoonup X$ is called a *guard function* on X if it satisfies $p \sqsubseteq \mathrm{id}_X$. Let $\mathbf{G}(X)$ be the set of all guard functions of X, that is, $\mathbf{G}(X) = \{p : X \rightharpoonup X \mid p \sqsubseteq \mathrm{id}_X\}$. It is clear that $\mathbf{G}(X)$ is a Heyting algebra as a subalgebra of $\mathbf{Rel}(X, X)$.

Proposition 2.3.9 *Let $p, q : X \rightharpoonup X$ be guard functions on an object X of* $\mathbf{E}$ *. Then*

(1) $pp = p$, $pq = qp = p \sqcap q$ *and* $p^\sharp = p$.

(2) *If* $p\Omega_X \sqsubseteq q\Omega_X$, *then* $p \sqsubseteq q$.

(3) *If* $p\Omega_X = q\Omega_X$, *then* $p = q$.

(4) $p\Omega_X \sqcap q\Omega_X = (p \sqcap q)\Omega_X$.

Proof. (a) simply follows from the following short computations: $pp \sqsubseteq p \sqsubseteq pp^\sharp p \sqsubseteq pp$, $pq \sqsubseteq p \sqcap q = (p \sqcap q)(p \sqcap q) \sqsubseteq pq$, and $p \sqsubseteq pp^\sharp p \sqsubseteq p^\sharp$.

(b) Assume $p\Omega_X \sqsubseteq q\Omega_X$. First note that a guard function is a partial function and $p \sqcup q$ is also a guard function. We have $(p \sqcup q)\Omega_X = p\Omega_X \sqcup q\Omega_X$ (by Distributive Law) $= q\Omega_X$ and hence $q = p \sqcup q$ follows from 2.3.8(b).

(c) is a corollary of (b).

(d) follows from $(p \sqcap q)\Omega_X \sqsubseteq p\Omega_X \sqcap q\Omega_X \sqsubseteq p(\Omega_X \sqcap p^\sharp q\Omega_X)$ (by Law of Puppe-Calenko) $= pp^\sharp q\Omega_X = pq\Omega_X = (p \sqcap q)\Omega_X$. ■

Throughout this paper the negation operator will be used only for guard functions. That is, for a guard function q on Y, $\neg q$ denotes the negation of q in $\mathbf{G}(Y)$, i.e. $\neg q = (q \Rightarrow 0_{YY}) \sqcap \mathrm{id}_Y$. For example, $\neg \mathrm{id}_Y = 0_{YY}$ and $\neg 0_{YY} = \mathrm{id}_Y$.

In the rest of the paper we will assume that all of the morphisms, relations, partial functions and guard functions are those in a fixed topos $\mathbf{E}$.

Lemma 2.3.10 *Let $\alpha : X \multimap Y$ be a relation and $q : Y \to Y$ a guard function. Then $\alpha q = 0_{XY}$ if and only if $\alpha(\neg q) = \alpha$.*

Proof. Assume $\alpha q = 0$. Then we have $\alpha^\sharp \alpha \sqcap \mathrm{id}_Y \sqsubseteq \neg q$ from $(\alpha^\sharp \alpha \sqcap \mathrm{id}_Y) \sqcap q \sqsubseteq \alpha^\sharp \alpha q = 0$. Hence $\alpha = \alpha \sqcap \alpha \sqsubseteq \alpha(\alpha^\sharp \alpha \sqcap \mathrm{id}_Y) \sqsubseteq \alpha(\neg q) \sqsubseteq \alpha$ and so $\alpha = \alpha(\neg q)$. Conversely assume $\alpha = \alpha(\neg q)$. Then it is immediate that $\alpha q = \alpha(\neg q) q = 0$. ∎

For every relation $\alpha : X \multimap Y$, there exists a monomorphism $i : D \to X$ in $\mathbf{E}$ such that $i^\sharp \Omega_D = \alpha \Omega_Y$ $(= i^\sharp i \Omega_X)$ from Rationality. Such a monomorphism i is called a *domain monomorphism* of α. It is trivial that $i^\sharp i$ is a guard function on X, that is, $i^\sharp i \sqsubseteq \mathrm{id}_X$. We define the kernel $k(\alpha)$ of α to be a guard function $k(\alpha) = \neg(i^\sharp i)$ on X and the domain $d(\alpha)$ of α to be a guard function $d(\alpha) = \neg k(\alpha)$ on X. Note that the definitions of kernels and domains do not depend on the choice of domain monomorphisms, that is, $i^\sharp i = \alpha \alpha^\sharp \sqcap \mathrm{id}_X$.

Proposition 2.3.11 *Let $\alpha, \beta : X \multimap Y$ be relations. Then*

(1) *$d(\alpha) = \neg k(\alpha)$, $k(\alpha) = \neg d(\alpha)$ and $k(\alpha) \sqcap d(\alpha) = 0_{XX}$.*

(2) *$k(id_X) = 0_{XX}$, $d(id_X) = id_X$, $k(0_{XY}) = id_X$ and $d(0_{XY}) = 0_{XX}$.*

(3) *If p is a guard function on X, then $k(p) = \neg p$ and $d(p) = \neg\neg p$.*

(4) *If $f : X \to Y$ is a morphism in $\mathbf{E}$, then $k(f) = 0_{XX}$ and $d(f) = id_X$.*

(5) *If $\alpha \sqsubseteq \beta$, then $k(\beta) \sqsubseteq k(\alpha)$ and $d(\alpha) \sqsubseteq d(\beta)$.*

Proof. The statements (a) - (d) are trivial.

(e) Let i and j be domain monomorphisms of α and β, respectively. Then $k(\alpha) = \neg(i^\sharp i)$ and $k(\beta) = \neg(j^\sharp j)$. The statement follows from $\alpha \Omega_Y \sqsubseteq \beta \Omega_Y \Rightarrow i^\sharp i \Omega_Y \sqsubseteq j^\sharp j \Omega_Y$ $\Rightarrow i^\sharp i \sqsubseteq j^\sharp j$ (by 2.3.9(c)) $\Rightarrow k(\beta) \sqsubseteq k(\alpha) \Rightarrow d(\alpha) \sqsubseteq d(\beta)$. ∎

Note that $k(\alpha) \sqcup d(\alpha) \neq \mathrm{id}_X$ in general, because of the basic properties of Heyting algebras.

Lemma 2.3.12 *Let $\alpha : X \multimap Y$ be a relation and $p : X \multimap X$ a guard function. Then the following statements are equivalent:*

(1) $p\alpha = 0_{XY}$.

(2) $(\neg p)\alpha = \alpha$.

(3) $p \sqsubseteq k(\alpha)$.

(4) $\neg\neg p \sqsubseteq k(\alpha)$.

(5) $(\neg\neg p)\alpha = 0_{XY}$.

Proof. Let i be a domain monomorphism of α. Then $\alpha\Omega_Y = i^\sharp i\Omega_X$. The proof follows from the following equivalences: $\alpha = (\neg p)\alpha \Leftrightarrow \alpha^\sharp = \alpha^\sharp(\neg p)$ (by applying $\sharp$ and 2.3.9(a)) $\Leftrightarrow \alpha^\sharp p = 0$ (by 2.3.10) $\Leftrightarrow$ (a) $p\alpha = 0 \Leftrightarrow pi^\sharp i\Omega_X = p\alpha\Omega_Y = 0$ (by 2.3.7(c)) $\Leftrightarrow$ $p \sqcap i^\sharp i = pi^\sharp i = 0 \Leftrightarrow$ (c) $p \sqsubseteq \neg(i^\sharp i) = k(\alpha) \Leftrightarrow$ (d) $\neg\neg p \sqsubseteq \neg(i^\sharp i) = k(\alpha)$ (by $p \sqsubseteq \neg\neg p$) $\Leftrightarrow$ (e) $(\neg\neg p)\alpha = 0$. ∎

Theorem 2.3.13 *Let $\alpha : X \multimap Y$ be a relation. Then*

(1) *$k(\alpha)\alpha = 0_{XY}$, that is, $k(\alpha)$ is the maximum element of a set $\{p \in G(X) \mid p\alpha = 0_{XY}\}$.*

(2) *$d(\alpha)\alpha = \alpha$.*

(3) *$k(\alpha) = id_X$ $(ord(\alpha) = 0_{XX})$ if and only if $\alpha = 0$.*

(4) *For every relation $\tau : W \multimap X$, $\tau\alpha = 0_{WY}$ if and only if $\tau k(\alpha) = \tau$.*

Proof. (a) and (b) easily follow from 2.3.12.

(c) From 2.3.11(b) it suffices to show that $d(\alpha) = 0$ implies $\alpha = 0$. But if $d(\alpha) = 0$ then $\alpha = d(\alpha)\alpha$ (by (b)) $= 0$.

(d) Let i be a domain monomorphism of α. Then $k(\alpha) = \neg(i^\sharp i)$. Hence $\tau k(\alpha) = \tau$ $\Leftrightarrow \tau(i^\sharp i) = 0$ (by 2.3.10 and $k(\alpha) = \neg(i^\sharp i)$) $\Leftrightarrow \tau\alpha\Omega_Y = \tau(i^\sharp i)\Omega_X = 0 \Leftrightarrow \tau\alpha = 0$. ∎

Corollary 2.3.14 *Let $\alpha : X \multimap Y$ be a relation and $p : X \multimap X$, $q : Y \multimap Y$ guard functions. Then the following statements are equivalent:*

(1) $p\alpha(\neg q) = 0_{XY}$.

(2) $(\neg p)\alpha(\neg q) = \alpha(\neg q)$.

(3) $p\alpha(\neg\neg q) = p\alpha$.

(4) $(\neg\neg p)\alpha(\neg q) = 0_{XY}$.

(5) $(\neg\neg p)\alpha(\neg\neg q) = (\neg\neg p)\alpha$.

(6) $p \sqsubseteq k(\alpha(\neg q))$.

Proof. From 2.3.12 it is easy to see (a) $\Leftrightarrow$ (b) $\Leftrightarrow$ (d) $\Leftrightarrow$ (f), and 2.3.10 implies (a) $\Leftrightarrow$ (c) and (d) $\Leftrightarrow$ (e). ■

Proposition 2.3.15 *Let A, B be objects in* **Pfn(E)**, *$A+B$ the coproduct of A and B in* **E**, *where $i_A : A \to A+B$ and $i_B : B \to A+B$ are inclusions and $i_A^\sharp i_A \sqcup i_B^\sharp i_B = id_{A+B}$, $i_A^\sharp i_B = \Theta_{AB}$ and $i_B^\sharp i_A = \Theta_{BA}$. Then, $A+B$ is also the coproduct of A and B in* **Pfn(E)**. *That is* **Pfn(E)** *has coproducts.* ■

We showed the pushout costruction in the category **Pfn(Set)** in Fact 2.2.1. Our method of Fact 2.2.1 and Lemma pfn:lemma:po using relational calculus needs only a properties about relations, so it is considered in the category **Pfn(E)**.

Proposition 2.3.16 *If* **E** *has the following properties:*

(1) *the set* **Sub**(A) *of subobjects of an object A is a complete lattice by inclusion,*

(2) *the distributive law (cf.* **2.1.2**$)$ *of relations holds,*

then **Pfn(E)** *has coequalizers. That is* **Pfn(E)** *is cocomplete.* ■

Proof. Let $f : A \to B$ and $g : A \to B$ be partial functions in **E**, $i : \mathrm{dom}(f) \sqcap \mathrm{dom}(g) \to B$ an inclusion. Since **E** is cocomplete there exist a coequalizer $e : B \to E$ in **E** . By condition (1), there exist a union of subobjects D where the inclusion $i_D : D \to E$ satisfies $fi_D^\sharp = gi_D^\sharp$. Let D_0 be the union and $\gamma : D_0 \to E$ an inclusion. It is easy to show $e\gamma^\sharp : B \to D_0$ is a coequalizer of f and g using similar relational calculation in Lemma 2.2.2. ■

Chapter 3

Graph structure over Pfn

In this chapter we present a categorical framework unifying several graph structures. For an endofunctor on the category **Pfn** we consider a graph structure as a function $V \to TV$ from a vertex set V to TV, where T is an endofunctor on **Pfn**. In the case $TV = V^*$ the structure is the same as Raoult's one [Rao84]. In section 3.1, we demonstrate many known graph structures are considered as examples and give the conditions for existence of pushouts in our general framework. The result on the existence theorem of pushouts produces a modification of Raoult's result [Rao84, Proposition 5] which lacked a condition. The amended proposition and a counter example to his result is given in section 3.2.

3.1 Graphs over Pfn

In this section, we introduce an abstract definition of a category which represents graphs and graph homomorphisms. Graph rewritings are defined by using a single pushout in the category. We prove a necessary and sufficient condition for existence of pushouts. Some concrete categories of graphs including Raoult's[Rao84] definition are shown.

Let $T : \mathbf{Pfn} \to \mathbf{Pfn}$ be an endofunctor. A *graph* constructed by T is a pair (A, a) of a set A and a total function $a : A \to TA$. A *graph morphism* $f : (A, a) \to (B, b)$ is a partial function $f : A \to B$ satisfying $fb = \mathrm{d}(f)aTf$. The *graph category* $G(T)$ is the

category of graphs and graph morphisms associated with T.

Lemma 3.1.1 *Let T : **Pfn** $\rightarrow$ **Pfn** be a functor, $a : A \rightarrow TA$, $b : B \rightarrow TB$ and $c : C \rightarrow TC$ total functions and $f : A \rightarrow B$ and $g : B \rightarrow C$ partial functions. If $fb = \mathrm{d}(f)aTf$ and $gc = \mathrm{d}(g)bTg$ then $fgc = \mathrm{d}(fg)aT(fg)$. That is $G(T)$ is in fact, a category.*

(*Proof.*) It follows from a simple relational computation:

$$\begin{aligned} fgc &= f\mathrm{d}(g)bTg \\ &= \mathrm{d}(fg)fbTg \qquad \text{(Lemma 2.1.5)} \\ &= \mathrm{d}(fg)\mathrm{d}(f)aTfTg \qquad (fb = \mathrm{d}(f)aTf) \\ &= \mathrm{d}(fg)aT(fg) \qquad (\mathrm{d}(fg)\mathrm{d}(f) = \mathrm{d}(fg)). \end{aligned}$$

■

Choosing a suitable functor T, we consider the several kinds of graph structures.

Example 3.1.2 (Kleene functor) We define the Kleene functor $* :$ **Pfn** $\rightarrow$ **Pfn** as follows. Let A, B be sets and $f : A \rightarrow B$ a partial function. $*A = A^*$ is the set of finite strings over A. We define $*f(= f^*) : A^* \rightarrow B^*$ as follows:

$$\begin{aligned} f^*(w) &= f(a_1)f(a_2)\cdots f(a_n) \qquad (\text{where } w = a_1a_2\cdots a_n \in (\mathrm{dom}(f))^*), \text{ and} \\ f^*(\varepsilon) &= \varepsilon. \end{aligned}$$

An object of $G(*)$ can be seen as a kind of directed graph and a morphism of $G(*)$ is a node-mapping which preserves out-edges but not in-edges. The category $G(*)$ is equivalent to what is considered by Raoult[Rao84].

Example 3.1.3 (Powerset functor) We define the power set functor $P :$ **Pfn** $\rightarrow$ **Pfn** as follows. Let A, B be sets and $f : A \rightarrow B$ a partial function. $P(A)$ is the set of all subsets of A and $Pf : P(A) \rightarrow P(B)$ is defined by $Pf(X) = f(X)$, for all $X \subset A$. An object of $G(P)$ is a kind of directed graph in which the out-edges of a node are not orderd and there cannot be multiple edges between the same nodes.

Example 3.1.4 A set N^A of functions from A to the set $N = \{0, 1, \ldots\}$ of natural numbers is defined by $N^A = \{f : A \to N | \Sigma_{x \in A} f(a) \text{ is finite.}\}$. We define the functor W : **Set** $\to$ **Set** as follows. Let A, B be sets and $f : A \to B$ a partial function. $W(A) = N^A$ and $Wf : N^A \to N^B$ is defined as $Wf(\alpha)(y) = \Sigma\{\alpha(x) | f(x) = y, x \in A\}$, $(\alpha \in N^A, y \in B)$. An objects of $G(W)$ are a type of edge-weighted directed graph.

We can treat labeled graph structure choosing a functor like next two examples.

Example 3.1.5 (L-labeled Kleene functor) We fix a set L of labels for edges. We define a functor $(L \times -)^*$: **Pfn** $\to$ **Pfn** as follows. For a set A, $(L \times A)^*$ is the set of finite strings of pairs of a label and an element of A. Other definition of the functor is similar to Example 3.1.2. An object of $G((L \times -)^*)$ is similar to the closed term hypergraph of Kennaway [Ken90].

Example 3.1.6 (L-labeled powerset functor) We similarly define a functor $P(L \times -)$: **Pfn** $\to$ **Pfn** like Example 3.1.3 and Example 3.1.5.

Theorem 3.1.7 *Let $f : (A, a) \to (B, b)$ and $g : (A, a) \to (C, c)$ be morphisms in $G(T)$. If the square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

is a pushout in **Pfn**, *then there exists a unique partial function*

$$d = (h^\sharp bTh) \cup (k^\sharp cTk)$$

such that $hd = \mathrm{d}(h)bTh$, $kd = \mathrm{d}(k)cTk$. When that is so, the square (2)

$$\begin{array}{ccc} (A, a) & \xrightarrow{f} & (B, b) \\ {\scriptstyle g}\downarrow & (2) & \downarrow{\scriptstyle h} \\ (C, c) & \xrightarrow[k]{} & (D, \delta) \end{array}$$

is a pushout in $G(T)$ if and only if $\delta = (h^\sharp bTh) \cup (k^\sharp cTk)$ and δ is a total function.

(*Proof.*) We first show $f\mathrm{d}(h)bTh = g\mathrm{d}(k)cTk$.

$$\begin{aligned} f\mathrm{d}(h)bTh &= \mathrm{d}(fh)fbTh \\ &= \mathrm{d}(fh)\mathrm{d}(f)aTfTh \\ &= \mathrm{d}(fh)aTfTh \\ &= \mathrm{d}(gk)aTgTk \\ &= \mathrm{d}(gk)\mathrm{d}(g)aTgTk \\ &= \mathrm{d}(gk)gcTk \\ &= g\mathrm{d}(k)cPk \end{aligned}$$

Since the square (1) is a pushout in **Pfn**, there exist a unique partial function $d : D \to TD$ such that $hd = \mathrm{d}(h)bTh$ and $hd = \mathrm{d}(k)cTk$, where $d = (h^\sharp bTh) \cup (k^\sharp cTk)$, by Fact 2.2.1. Assume the square (2) is a pushout. Graph morphisms h and k satisfys $h\delta = \mathrm{d}(h)bTh$ and $h\delta = \mathrm{d}(k)cTk$. Since $h^\sharp\mathrm{d}(h) = h^\sharp$, $k^\sharp\mathrm{d}(k) = k^\sharp$ and $(h^\sharp h \cup k^\sharp k) = \mathrm{id}_D$, we have $\delta = (h^\sharp h \cup k^\sharp k)d = (h^\sharp bTh) \cup (k^\sharp cTk)$. Conversely, assume $\delta = (h^\sharp bTh) \cup (k^\sharp cTk)$ is a total function. Let (S, s) be an object in $G(T)$, and $x : B \to S$, $y : C \to S$ morphisms in $G(T)$ satisfying $fx = gy$. Since the square (1) is a pushout in **Pfn**, there exist a unique partial function $t : D \to S$ such that $ht = x$ and $ht = y$, where $t = h^\sharp x \cup k^\sharp y$, by Fact 2.2.1. We only need to show that t is a graph morphism. Since $h^\sharp x\Omega_S = h^\sharp\mathrm{d}(h)\mathrm{d}(x)\Omega_B = h^\sharp\mathrm{d}(x)h\Omega_D$ and $k^\sharp y\Omega_S = k^\sharp\mathrm{d}(y)k\Omega_D$, we have $\mathrm{d}(t) = h^\sharp\mathrm{d}(x)h \cup k^\sharp\mathrm{d}(y)k$. Since

$$\begin{aligned} h^\sharp\mathrm{d}(x)h\delta Tt &= h^\sharp\mathrm{d}(x)h\delta Tt \\ &= h^\sharp\mathrm{d}(x)\mathrm{d}(h)bThTt \\ &= h^\sharp\mathrm{d}(x)\mathrm{d}(h)bTx \\ &= h^\sharp\mathrm{d}(h)\mathrm{d}(x)bTx \\ &= h^\sharp\mathrm{d}(h)xs \\ &= h^\sharp xs \end{aligned}$$

and $k^\sharp\mathrm{d}(y)k\delta Tt = k^\sharp ys$, we obtain $\mathrm{d}(t)\delta Tt = h^\sharp xs \cup k^\sharp ys = ts$. That is t is a graph morphism. So the square (2) is a pushout in $G(T)$.

■

Corollary 3.1.8 *Let* $f : (A,a) \to (B,b)$ *and* $g : (A,a) \to (C,c)$ *be morphisms in* $G(T)$. *If the square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

is a pushout in **Pfn**, *then the following conditions are equivalent:*

(1) $d = (h^\sharp bTh) \cup (k^\sharp cTk)$ *is a total function.*

(2) $b(\mathrm{dom}(h)) \subset \mathrm{dom}(Th)$ *and* $c(\mathrm{dom}(k)) \subset \mathrm{dom}(Tk)$

(3) $\mathrm{dom}(h) \subset \mathrm{dom}(bTh)$ *and* $\mathrm{dom}(k) \subset \mathrm{dom}(cTk)$

(*Proof.*) If $b^\sharp h\Omega_D \subset Th\Omega_{TD}$ then $h\Omega_D \subset bb^\sharp h\Omega_D \subset bTh\Omega_{TD}$. If $h\Omega_D \subset bTh\Omega_{TD}$ then $b^\sharp h\Omega_D \subset b^\sharp bTh\Omega_{TD} \subset Th\Omega_{TD}$. We have $c^\sharp k\Omega_D \subset Tk\Omega_{TD}$ if and only if $k\Omega_D \subset cTk\Omega_{TD}$. So we have shown that (2) and (3) are equivalent.

Next we show the equivalence of (3) and (1). Assume $h\Omega_D \subset bTh\Omega_{TD}$ and $k\Omega_D \subset cTk\Omega_{TD}$. Since $h^\sharp h \cup k^\sharp k = \mathrm{id}_D$ by Lemma 2.2.2, we have $\Omega_D \subset (h^\sharp h\Omega_D) \cup (k^\sharp k\Omega_D) \subset (h^\sharp bTh\Omega_{TD}) \cup (k^\sharp cTk\Omega_{TD}) \subset d\Omega_{TD}$. This means d is a total function. Conversely, assume d is a total function. Since $hd = \mathrm{d}(h)bTh$ means $\mathrm{d}(h) = \mathrm{d}(h) \cap \mathrm{d}(bTh)$, we have $\mathrm{d}(h) \subset \mathrm{d}(bTh)$. Similaly $\mathrm{d}(k) \subset \mathrm{d}(cTk)$.

■

We note that if $T = P$, $T = W$ and $T = P(L \times -)$, then $Tf : TA \to TB$ is a total function for any partial function $f : A \to B$. This property is very convenient for existence of pushouts.

Corollary 3.1.9 *The categories* $G(P)$, $G(W)$ *and* $G(P(L \times -))$ *have pushouts.*

3.2 Observations

In this section, we provide the proof of Raoult's proposition 5 in a view point of our framework.

Let $f : A \to B$ and $g : A \to C$ be partial functions. We define a relation $\Gamma_{(f,g)} : A \to 1$ by $\Gamma_{(f,g)} = \cup\{\alpha : A \to 1 | ff^\sharp\alpha = \alpha \text{ and } gg^\sharp\alpha = \alpha\}$, that is $\Gamma_{(f,g)}$ is the maximum relation satisfying $ff^\sharp\Gamma_{(f,g)} = \Gamma_{(f,g)}$ and $gg^\sharp\Gamma_{(f,g)} = \Gamma_{(f,g)}$.

Lemma 3.2.1 *Let*

$$\begin{array}{ccc} (A,a) & \xrightarrow{f} & (B,b) \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ (C,c) & \xrightarrow[k]{} & (D,d) \end{array}$$

be a pushout in $G(T)$. *Then* $\Gamma_{(f,g)} = fh\Omega_D (= gk\Omega_D)$.

(*Proof.*) It is obvious $ff^\sharp fh\Omega_D = fh\Omega_D$ and $gg^\sharp fh\Omega_D = fh\Omega_D$. Asuume a relation $\alpha : A \to 1$ satisfies $ff^\sharp\alpha = \alpha$ and $gg^\sharp\alpha = \alpha$. Since $f^\sharp\alpha : B \to 1$ and $g^\sharp\alpha : C \to 1$ are partial functions and D is a pushout, there exist a unique function $\beta : D \to 1$ such that $h\beta = f^\sharp\alpha$ and $k\beta = g^\sharp\alpha$ hold. We obtain $fh\Omega_D \supset ff^\sharp\alpha = \alpha$. So $fh\Omega_D$ is the maximum relation.

■

We note that $\Gamma_{(f,g)} = fh\Omega_D$ means $f^{-1}(\mathrm{dom}(h)) = \{a \in A | (a,1) \in \Gamma_{(f,g)}\}$. By Lemma 2.2.2, $\mathrm{dom}(h) = (B - f(A)) \cup f(A')$ where $A' = \{a \in A | (a,1) \in \Gamma_{(f,g)}\}$.

Lemma 3.2.2 *Under the situation of Theorem 3.1.7, consider the functor* $T - *$. *Then following five conditions are equivalent:*

(1) $b(f(A')) \subset \mathrm{dom}(Th) (= (\mathrm{dom}(h))^*)$,

(2) $c(g(A')) \subset \mathrm{dom}(Tk) (= (\mathrm{dom}(k))^*)$,

(3) $Tf(a(A')) \subset (\mathrm{dom}(h))^*$,

(4) $Tg(a(A')) \subset (\mathrm{dom}(k))^*$, *and*

(5) $a(A') \subset (A')^*$,

where $A' = \{a \in A | (a,1) \in \Gamma_{(f,g)}\}$.

(*Proof.*) Since $\mathrm{d}(f)aTf = fb$ and $\mathrm{d}(f)f = f$, we have $b^\sharp f^\sharp fh\Omega_B = (Tf)^\sharp a^\sharp fh\Omega_B$. This means $b(f(A')) = Tf(a(A'))$. So (1) and (3) are equivalent. (3) and (5) are equivalent by $(Tf)^{-1}(\mathrm{dom}(h)^*) = (A')^*$. Similarly, (2),(4) and (5) are equivalent, because of $\Gamma_{(f,g)} = gk\Omega_D$.

■

Using the last lemma, condition (2) in Corollary 3.1.8 is replaced to the form in the next proposition. The next proposition which was originally proved by Raoult [Rao84, Proposition 5] is a special case of Theorem 3.1.7.

Proposition 3.2.3 *Let the square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

is a pushout in **Pfn**. *A commutative diagram*

$$\begin{array}{ccc} (A,a) & \xrightarrow{f} & (B,b) \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ (C,c) & \xrightarrow[k]{} & (D,d) \end{array}$$

in $G(*)$ is a pushout in $G(*)$ if and only if the following conditions (1), (2) and (3) hold:

(1) $b(B - f(A)) \subset (\mathrm{dom}(h))^*$,

(2) $c(C - g(A)) \subset (\mathrm{dom}(k))^*$, *and*

(3) $a(A') \subset (A')^*$,

where $A' = \{a \in A | (a,1) \in \Gamma_{(f,g)}\}$.

The next example is a counter example of Raoult's proposition 5[Rao84] which lack a condition (3) of Proposition 3.2.3.

Example 3.2.4 Let $A = \{x_1 \to x_2, x_3\}$, $B = \{y_1 \to y_2\}$ and $C = \{z_1 \to z_2\}$ be graphs. Define graph morphisms $f : A \to B$ and $g : A \to C$ by $f(x_1) = y_1$, $f(x_2) =$

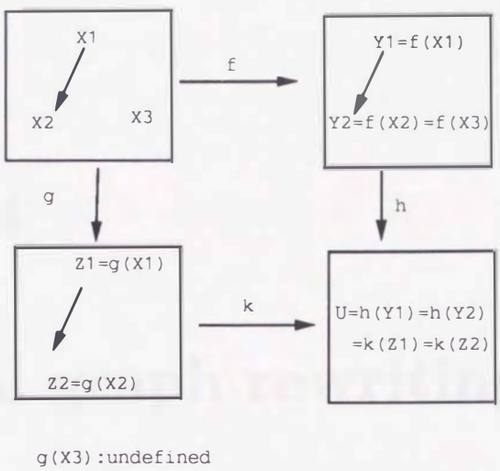


Figure 3.1:

$f(x_3) = y_2$, $g(x_1) = z_1$ and $g(x_2) = z_2$. The value of $g(x_3)$ is undefined(cf. Figure 3.1). It is easy to check $A' = \{x_1\}$. and the condition in Proposition 3.2.3(3) does not hold. Consider the pushout

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

in the category **Pfn**. Since D is a one point set, h and k are not graph morphisms.

Chapter 4

Relational graph rewritings

In this chapter, we treat the category of (simple) graphs (with or without labeled edges) and partial functions preserving graph structures, and we present a new formalization of graph rewritings by using a primitive pushout construction in the category. We show the main subjects of this chapter in Section 4.1. For a pair of partial functions from a common set into graphs a primitive pushout square is constructed, which shows the category of graphs and partial morphisms has pushouts. Moreover we give a more general sufficient condition for two graph rewritings to commute and a critical pair lemma of graph rewriting systems. In Section 4.2, we compare our approach with another approaches by Ehrig[EKL90], Löwe[LE90], Kennaway[Ken90] and Okada[OH91]. Some examples related to graph rewritings are listed in Section 4.3. In Section 4.4, we state how to develop our formalization of graph rewritings for graphs with labelled edges which contains Raoult's graphs[Rao84].

4.1 Rewritings for simple graphs

A *(simple) graph* $< A, \alpha >$ is a pair of a set A and a relation $\alpha : A \rightharpoonup A$. A *partial morphism* f of a graph $< A, \alpha >$ into a graph $< B, \beta >$, denoted by $f :< A, \alpha > \rightarrow < B, \beta >$, is a partial function $f : A \rightarrow B$ satisfying $d(f)\alpha f \sqsubseteq f\beta$. It is easily seen that a partial morphism among graphs is a partial function preserving edges on its domain of definitions.

Let $f :< A, \alpha >\rightarrow< B, \beta >$ and $g :< B, \beta >\rightarrow< C, \gamma >$ be partial morphisms of graphs. Since $d(f)\alpha f \sqsubseteq f\beta$ and $d(g)\beta g \sqsubseteq g\gamma$, we have $d(fg)\alpha fg = d(fg)d(f)\alpha fg$ (by 2.4(a)) $\sqsubseteq d(fg)f\beta g = fd(g)\beta g$ (by 2.4(b)) $\sqsubseteq fg\gamma$. Hence the composite of two partial morphisms of graphs is also a partial morphism of graphs. Thus we have the category $\mathbf{G}_p$ of (simple) graphs and partial morphisms between them.

The following theorem constructs a primitive pushout for a pair of partial functions from a common set into graphs.

Theorem 4.1.1 *If $< B, \beta >$ and $< C, \gamma >$ are graphs and if the square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

is a pushout in **Pfn**, *then $h :< B, \beta >\rightarrow< D, \delta >$ and $k :< C, \gamma >\rightarrow< D, \delta >$ are partial morphisms of graphs, where $\delta = h^\sharp\beta h \sqcup k^\sharp\gamma k$. Moreover, if $h' :< B, \beta >\rightarrow< D', \delta' >$ and $k' :< C, \gamma >\rightarrow< D', \delta' >$ are partial morphisms of graphs satisfying $fh' = gk'$, then there exists a unique partial morphism $t :< D, \delta >\rightarrow< D', \delta' >$ of graphs such that $h' = ht$ and $k' = kt$.*

(*Proof.*) First we see that $h :< B, \beta >\rightarrow< D, \delta >$ and $k :< B, \beta >\rightarrow< D, \delta >$ are partial morphisms of graphs. It simply follows from $d(h)\beta h \sqsubseteq hh^\sharp\beta h$ (by $d(h) = hh^\sharp \sqcap \mathrm{id}_B) \sqsubseteq h\delta$ (by $\delta = h^\sharp\beta h \sqcup k^\sharp\gamma k$). Next assume that $h' :< B, \beta >\rightarrow< D', \delta' >$ and $k' :< C, \gamma >\rightarrow< D', \delta' >$ are partial morphisms of graphs satisfying $fh' = gk'$. Then we have $d(h')\beta h' \sqsubseteq h'\delta'$ and $d(k')\gamma k' \sqsubseteq k'\delta'$. As (1) is a pushout in **Pfn**, there exists a unique partial function $t : D \rightarrow D'$ such that $h' = ht$ and $k' = kt$. It suffices to prove that $d(t)\delta t \sqsubseteq t\delta'$. But it follows from

$$\begin{aligned} d(t)\delta t &\sqsubseteq tt^\sharp(h^\sharp\beta h \sqcup k^\sharp\gamma k)t \quad (d(t) = tt^\sharp \sqcap \mathrm{id}_D) \\ &= t(t^\sharp h^\sharp\beta ht \sqcup t^\sharp k^\sharp\gamma kt) \quad (\text{by } \mathbf{(2.1.2)}) \\ &= t(h'^\sharp\beta h' \sqcup k'^\sharp\gamma k') \quad (h' = ht,\ k' = kt) \end{aligned}$$

$$
\begin{aligned}
&= t(h'^{\sharp}d(h')\beta h' \sqcup k'^{\sharp}d(k')\gamma k') \quad (h' = d(h')h',\ k' = d(k')k') \\
&= t(h'^{\sharp}h'\delta' \sqcup k'^{\sharp}k'\delta') \quad (d(h')\beta h' \sqsubseteq h'\delta',\ d(k')\gamma k' \sqsubseteq k'\delta') \\
&\sqsubseteq t(\delta' \sqcup \delta') \quad (h'^{\sharp}h' \sqsubseteq id_{D'},\ k'^{\sharp}k' \sqsubseteq id_{D'}) \\
&= t\delta'.
\end{aligned}
$$

This completes the proof.

■

Note that the graph $< D, \delta >$ is unique up to isomorphisms. The following is exactly a corollary of the last theorem.

Corollary 4.1.2 $\mathbf{G}_p$ *has pushouts.* ■

A partial morphism $f :< A, \alpha > \rightarrow < B, \beta >$ is said to be a *morphism* of graphs if $f : A \rightarrow B$ is a function. It is trivial that the composition of two morphisms of graphs is also a morphism of graphs and so one can consider the category **G**of graphs and morphisms between them.

Definition 4.1.3 A *rewriting rule* p is a triple of two graphs $< A, \alpha >$, $< B, \beta >$ and a partial function $f : A \rightarrow B$. (Note that f need not to be a partial morphism of graphs.) A *matching* to p is a morphism $g :< A, \alpha > \rightarrow < G, \xi >$ of graphs. Then construct a pushout

$$
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
{\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\
G & \xrightarrow[k]{} & H
\end{array}
$$

in **Pfn** and define $\eta = h^{\sharp}\beta h \sqcup k^{\sharp}(\xi - g^{\sharp}\alpha g)k$. The graph $< H, \eta >$ is the resultant graph after applying a production rule p along a matching g, and denoted by $< G, \xi > \Rightarrow_{p/g} < H, \eta >$. A square

$$
\begin{array}{ccc}
< A, \alpha > & \xrightarrow{f} & < B, \beta > \\
{\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\
< G, \xi > & \xrightarrow[k]{} & < H, \eta >
\end{array}
$$

is called the rewriting square for a rewriting rule p along a matching g. (Note that the rewriting square is not necessarily a pushout in the category of graphs and partial morphisms.)

Proposition 4.1.4 *Let* $g :< A, \alpha > \rightarrowtail < G, \xi >$ *be a matching to a rewriting rule* $p = (< A, \alpha >, < B, \beta >, f : A \rightarrow B)$. *If* $f :< A, \alpha > \rightarrowtail < B, \beta >$ *is a partial morphism of graphs, then the rewriting square*

$$\begin{array}{ccc} < A, \alpha > & \xrightarrow{f} & < B, \beta > \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ < G, \xi > & \xrightarrow[k]{} & < H, \eta > \end{array}$$

for p *along* g *is a pushout in* $\mathbf{G}$.

(*Proof.*) By the virtue of **4.1.1** it suffices to show that $\eta = h^\sharp \beta h \sqcup k^\sharp \xi k$. First note that $f^\sharp \alpha f \sqsubseteq f^\sharp f \beta \sqsubseteq \beta$ since $d(f)\alpha f \sqsubseteq f\beta$. Thus we have

$$\begin{aligned} \eta &= h^\sharp \beta h \sqcup k^\sharp (\xi - g^\sharp \alpha g) k \\ &\sqsupseteq h^\sharp f^\sharp \alpha f h \sqcup k^\sharp (\xi - g^\sharp \alpha g) k \\ &= k^\sharp g^\sharp \alpha g k \sqcup k^\sharp (\xi - g^\sharp \alpha g) k \\ &= k^\sharp \{ g^\sharp \alpha g \sqcup (\xi - g^\sharp \alpha g) \} k \\ &= k^\sharp \xi k. \end{aligned}$$

and

$$\begin{aligned} \eta &= h^\sharp \beta h \sqcup k^\sharp (\xi - g^\sharp \alpha g) k \sqcup k^\sharp \xi k \\ &= h^\sharp \beta h \sqcup k^\sharp \xi k. \end{aligned}$$

This completes the proof.

■

The last proposition suggests that our graph rewritings coincide with those of Raoult [Rao84] if a production rule is a partial morphism of graphs. It is easy to understand that analogous results to Raoult's work [Rao84] about the confluency and concurrency of graph rewritings are valid in our case. We state a general sufficient condition for two graph rewritings to commute (or to be strongly confluent).

Theorem 4.1.5 *Let* $p_\lambda = (< A_\lambda, \alpha_\lambda >, < B_\lambda, \beta_\lambda >, f_\lambda : A_\lambda \to B_\lambda)$ *be rewriting rules,* $g_\lambda :< A_\lambda, \alpha_\lambda > \to < G, \xi >$ *matchings to* p_λ *and* $< G, \xi > \Rightarrow_{p_\lambda/g_\lambda} < H_\lambda, \eta_\lambda >$ *for* $\lambda = 0, 1$. *If* $f_\lambda :< A_\lambda, \alpha_\lambda > \to < B_\lambda, \beta_\lambda >$ *is partial morphisms of graphs* $(\lambda = 0, 1)$ *and* $\mathrm{Im}(g_0) \sqcap \mathrm{Im}(g_1) \sqsubseteq \mathrm{dom}(k_0) \sqcap \mathrm{dom}(k_1)$. *then there exist matchings* $g'_\lambda :< A, \alpha_\lambda > \to < H_{1-\lambda}, \eta_{1-\lambda} >$ $(\lambda = 0, 1)$ *and a graph* $< H, \eta >$ *such that* $< H_{1-\lambda}, \eta_{1-\lambda} > \Rightarrow_{p_\lambda/g'_\lambda} < H, \eta >$ $(\lambda = 0, 1)$.

(*Proof.*) As f_0, g_0, f_1 and g_1 are partial morphisms of graphs, we can construct the following three pushouts in the category of graphs and partial morphisms between them by **4.1.2**:

$$\begin{array}{ccccc}
 & & < A_0, \alpha_0 > & \xrightarrow{f_0} & < B_0, \beta_0 > \\
 & & {\scriptstyle g_0}\downarrow & (0) & \downarrow{\scriptstyle h_0} \\
< A_1, \alpha_1 > & \xrightarrow{g_1} & < G, \xi > & \xrightarrow[k_0]{} & < H_0, \eta_0 > \\
{\scriptstyle f_1}\downarrow & (1) & \downarrow{\scriptstyle k_1} & (2) & \downarrow{\scriptstyle h'_0} \\
< B_1, \beta_1 > & \xrightarrow[h_1]{} & < H_1, \eta_1 > & \xrightarrow[h'_1]{} & < H, \eta >
\end{array}$$

Set $g'_\lambda = g_\lambda k_{1-\lambda}$ $(\lambda = 0, 1)$. Then we can deduces that g'_λ $(\lambda = 0, 1)$ is a function because of **2.2.3**, and so $g'_\lambda :< A_\lambda, \alpha_\lambda > \to < H_{1-\lambda}, \eta_{1-\lambda} >$ is a matching to p_λ $(\lambda = 0, 1)$. Since two squares (0)+(2) and (1)+(2) are pushouts in the category of graphs and partial morphisms between them, we have $< H_{1-\lambda}, \eta_{1-\lambda} > \Rightarrow_{p_\lambda/g'_\lambda} < H, \eta >$ $(\lambda = 0, 1)$ by means of **4.1.3**, which proves the theorem.

■

In the rest of this section we observe the critical pair lemma [KB70, Rao84, OH91] in our graph rewriting system. A basic idea using a single pushout construction was initiated by Raoult [Rao84]. Our approach is an extension of his method. We restrict no rewriting rules but a matching as an injective graph morphism. An essential point of our formulation is due to Proposition 2.2.4 stating that a pushout in our category of graphs preserves injective graph morphisms. In what follows, we assume that a rewriting rule is a morphism of graphs, and a matching is a (total) injective graph morphism.

Definition 4.1.6 Let $< A_\lambda, \alpha_\lambda >, < B_\lambda, \beta_\lambda >$ *and* $< I, 0 >$ be graphs, $(< A_\lambda, \alpha_\lambda >, < B_\lambda, \beta_\lambda >, f_\lambda : A_\lambda \to B_\lambda)$ a rewriting rule, and $i_\lambda :< I, 0 > \to < A_\lambda, \alpha_\lambda >$ an injective morphism of graphs $(\lambda = 0, 1)$, where 0 is the empty relation. A pair $(t_0 : S \to T_0, t_1 : S \to T_1)$ of morphisms defined by the following squares is called a critical pair formed from (f_0, f_1) by (i_0, i_1).

$$\begin{array}{ccccc}
< I, 0 > & \xrightarrow{i_0} & < A_0, \alpha_0 > & \xrightarrow{f_0} & < B_0, \beta_0 > \\
{\scriptstyle i_1}\downarrow & \mathrm{PO} & \downarrow{\scriptstyle s_0} & \mathrm{PO} & \downarrow{\scriptstyle h_0} \\
< A_1, \alpha_1 > & \xrightarrow{s_1} & < S, \delta > & \xrightarrow[t_0]{} & < T_0, \delta_0 > \\
{\scriptstyle f_1}\downarrow & \mathrm{PO} & \downarrow{\scriptstyle t_1} & & \\
< B_1, \beta_1 > & \xrightarrow[h_1]{} & < T_1, \delta_1 > & &
\end{array}$$

We note that if A_0 and A_1 are finite sets, then the set of critical pairs of (f_0, f_1) is finite. A function $f : I \to A$ is always a graph morphisms $f :< I, 0 > \to < A, \alpha >$ for any graph $< A, \alpha >$.

Lemma 4.1.7 *Let* $< G, \xi > \Rightarrow_{f_0/g_0} < H_0, \eta_0 >$ *and* $< G, \xi > \Rightarrow_{f_1/g_1} < H_1, \eta_1 >$ *be graph rewritings and a square*

$$\begin{array}{ccc}
I & \xrightarrow{i_0} & A_0 \\
{\scriptstyle i_1}\downarrow & PB & \downarrow{\scriptstyle g_0} \\
A_1 & \xrightarrow[g_1]{} & G
\end{array}$$

be a pullback in **Set**. *If* $(t_0 :< S, \sigma > \to < T_0, \tau_0 >, t_1 :< S, \sigma > \to < T_1, \tau_1 >)$ *is a critical pair formed from* (f_0, f_1) *by* (i_0, i_1) *and squares*

$$\begin{array}{ccc}
< A_\lambda, \alpha_\lambda > & \xrightarrow{f_\lambda} & < B_\lambda, \beta_\lambda > \\
{\scriptstyle s_\lambda}\downarrow & PO & \downarrow{\scriptstyle h_\lambda} \\
< S, \sigma > & \xrightarrow{t_\lambda} & < T_\lambda, \tau_\lambda > \\
{\scriptstyle s}\downarrow & PO & \downarrow{\scriptstyle h''_\lambda} \\
< G, \xi > & \xrightarrow[k'_\lambda]{} & < H_\lambda, \eta_\lambda >
\end{array}$$

are pushouts in $\mathbf{G}_p$, *then* $h_\lambda h''_\lambda = h'_\lambda$, *where* $s :< S, \sigma > \to < G, \xi >$ *is a unique*

morphism satisfying $s_\lambda s = g_\lambda$ *and a square*

$$\begin{array}{ccc} < A_\lambda, \alpha_\lambda > & \xrightarrow{f_\lambda} & < B_\lambda, \beta_\lambda > \\ {\scriptstyle g_\lambda}\downarrow & PO & \downarrow{\scriptstyle h'_\lambda} \\ < G, \xi > & \xrightarrow[k_\lambda]{} & < H_\lambda, \eta_\lambda > \end{array}$$

is a pushout in $\mathbf{G}_p$ $(\lambda = 0, 1)$.

(*Proof.*) Since the square

$$\begin{array}{ccc} < I, 0 > & \xrightarrow{i_0} & < A_0, \alpha_0 > \\ {\scriptstyle i_1}\downarrow & \text{PO} & \downarrow{\scriptstyle s_0} \\ < A_1, \alpha_1 > & \xrightarrow{s_1} & < S, \delta > \end{array}$$

is a pushout, there exist a unique morphism $s :< S, \sigma > \rightarrow < G, \xi >$ satisfying $s_\lambda s = g_\lambda$ $(\lambda = 0, 1)$. The composition $A_\lambda B_\lambda G H_\lambda$ of pushout squares $A_\lambda B_\lambda S T_\lambda$ and $S T_\lambda G H_\lambda$ is also a pushout, so we have $h_\lambda h''_\lambda = h'_\lambda$ $(\lambda = 0, 1)$.

■

We call a critical pair $(t_0 :< S, \sigma > \rightarrow < T_0, \tau_0 >, t_1 :< S, \sigma > \rightarrow < T_1, \tau_1 >)$ *confluent* if there exist rewriting rules f'_0, f'_1 such that two graphs $< T_0, \tau_0 >, < T_1, \tau_1 >$ are rewritten to a same graph $< T, \tau >$. That is there exist a graph $< T, \tau >$ and rewriting rules f_λ $(\lambda = 0, 1)$ such that a square

$$\begin{array}{ccc} . & \xrightarrow{f'_\lambda} & . \\ \downarrow & \text{PO} & \downarrow \\ < T_\lambda, \tau_\lambda > & \xrightarrow[t'_\lambda]{} & < T, \tau > \end{array}$$

is a pushout in $\mathbf{G}_p$ and $t_0 t'_0 = t_1 t'_1$.

Theorem 4.1.8 (Critical Pair Lemma) *A graph rewriting system is confluent, if every critical pair is confluent.*

(*Proof.*) Let a square

$$\begin{array}{ccc} < A_\lambda, \alpha_\lambda > & \xrightarrow{f_\lambda} & < B_\lambda, \beta_\lambda > \\ {\scriptstyle g_\lambda}\downarrow & \text{PO} & \downarrow{\scriptstyle h'_\lambda} \\ < G, \xi > & \xrightarrow[k_\lambda]{} & < H_\lambda, \eta_\lambda > \end{array}$$

be a pushout ($\lambda = 0, 1$). By Lemma 4.1.7, there exist a critical pair ($t_0 :< S, \sigma > \rightarrow < T_0, \tau_0 >, t_1 :< S, \sigma > \rightarrow < T_1, \tau_1 >$) and pushouts

$$\begin{array}{ccc} < A_\lambda, \alpha_\lambda > & \xrightarrow{f_\lambda} & < B_\lambda, \beta_\lambda > \\ {\scriptstyle s_\lambda}\downarrow & \text{PO} & \downarrow{\scriptstyle h_\lambda} \\ < S, \sigma > & \xrightarrow[k'_\lambda]{t_\lambda} & < T_\lambda, \tau_\lambda > \\ {\scriptstyle s}\downarrow & \text{PO} & \downarrow{\scriptstyle h''_\lambda} \\ < G, \xi > & \xrightarrow[k'_\lambda]{} & < H_\lambda, \eta_\lambda > \end{array}$$

in $\mathbf{G}_p$ satisfying $s_\lambda s = g_\lambda$ and $h_\lambda h''_\lambda = h'_\lambda$ ($\lambda = 0, 1$). Since every critical pair is confluent, there exists a graph $< T, \tau >$ and rewritings $< T_\lambda, \tau_\lambda > \Rightarrow_{f'_\lambda} < T, \tau >$. By making the following pushout

$$\begin{array}{ccc} < T_\lambda, \tau_\lambda > & \xrightarrow{t'_\lambda} & < T, \tau > \\ {\scriptstyle h''_\lambda}\downarrow & \text{PO} & \downarrow \\ < H_\lambda, \eta_\lambda > & \longrightarrow & < \widehat{H_\lambda}, \widehat{\eta_\lambda} > \end{array}$$

we have $< H_\lambda, \eta_\lambda > \Rightarrow_{f'_\lambda} < \widehat{H_\lambda}, \widehat{\eta_\lambda} >$ ($\lambda = 0, 1$). Since the composition

$$\begin{array}{ccc} < S, \sigma > & \xrightarrow{t_\lambda t'_\lambda} & < T, \tau > \\ {\scriptstyle s}\downarrow & & \downarrow \\ < G, \xi > & \longrightarrow & < \widehat{H_\lambda}, \widehat{\eta_\lambda} > \end{array}$$

of pushouts squares $ST_\lambda GH_\lambda$ and $T_\lambda TH_\lambda \widehat{H_\lambda}$ is also a pushout ($\lambda = 0, 1$) and $t_0 t'_0 = t_1 t'_1$, we have $< \widehat{H_0}, \widehat{\eta_0} > = < \widehat{H_0}, \widehat{\eta_0} >$. This completes the proof.

■

4.2 Observations

We first compare our category of graphs with Löwe's one [LE90]. Let $< A, \alpha >$ be a graph in our sense. We have two functions $i_\alpha p : \alpha \rightarrow A$ and $i_\alpha q : \alpha \rightarrow A$, where $i_\alpha : \alpha \rightarrow A \times A$ is an inclusion function and $p : A \times A \rightarrow A$ and $q : A \times A \rightarrow A$ are projections. Then $(\alpha, A, i_\alpha p, i_\alpha q)$ is naturally considered as a Sig-algebra with $Sig = \{s, t : E \rightarrow V\}$ in the sense of Löwe [LE90]. Exactly a graph in our sense corresponds to a Sig-algebra

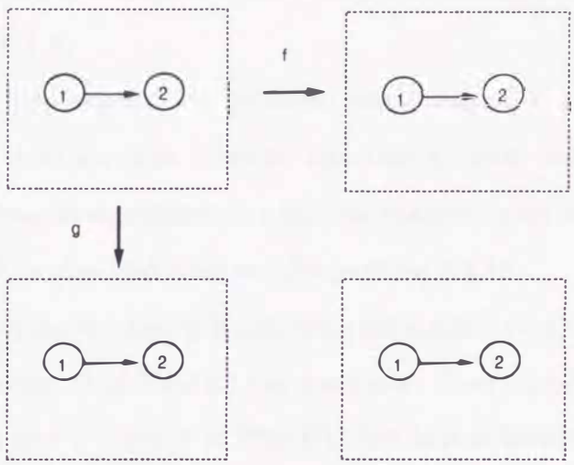


Figure 4.1:

(G_E, G_V, s^G, t^G) such that a function $(s^G, t^G) : G_E \to G_V \times G_V$ is injective. A partial Sig-algebra morphism from $< A, \alpha >$ to $< B, \beta >$ is a tuple $(< A', \alpha' >, i_f, t_f)$ of a subgraph $< A', \alpha' >$ of $< A, \alpha >$, an inclusion function $i_f :< A', \alpha' > \rightarrowtail < A, \alpha >$, and a (total) graph morphism $t_f :< A', \alpha' > \rightarrow < B, \beta >$. It corresponds to a notion of partial morphism [RR88, Ken90] over **G**. But **G**has pushouts which are not heredetary, so the category of partial morphisms constructed from **G**is not pushout complete [Ken90]. **Figure 4.1** illustrates a pushout which is not hereditary. Let $f :< A, \alpha > \rightarrowtail < B, \beta >$ be a partial graph morphism in our sense. We have the domain A' of partial function $f : A \to B$, an inclusion function $i_f : A' \to A$ and a function $t_f : A' \to B$ such that $f = i_f^\natural t_f$. Define α' by constructing a pullback

$$\begin{array}{ccc} \alpha' & \xrightarrow{i_{\alpha'}} & A' \times A' \\ \downarrow & \text{PB} & \downarrow{\scriptstyle i_f \times i_f} \\ \alpha & \xrightarrow[i_\alpha]{} & A \times A \end{array}$$

in **Set**. Sinse $d(f)\alpha f \sqsubseteq f\beta$, it follows by assumptions that $i_f :< A', \alpha' > \rightarrowtail < A, \alpha >$ and $t_f :< A', \alpha' > \rightarrow < B, \beta >$ are graph morphisms. But there may be many subgraphs $< A', \alpha^* >$ of $< A, \alpha >$ such that $t_f :< A', \alpha^* > \rightarrow < B, \beta >$ is a graph morphism. This is a difference between our partial graph morphisms and Löwe's graph morphisms. **Figure 4.1** indicates that Löwe's pushout construction is not closed under the subclass

of our graphs and so it is meaningful to prove the pushout completeness of the category $\mathbf{G}_p$ in our sense(cf. **4.1.2**).

Löwe [Löw89, LE90] showed the pushout completeness of the category of Sig-algebras whose signature contains monadic operator symbols only. In this case the category of Sig-algebras is equivalent to a functor category over **Set** which is a topos [Gol79]. So his result is also lead from our Proposition 2.3.16.

Kennaway [Ken90] showed that if **E** satisfying the condition (a) of Propositon 2.3.16 has hereditary pushouts, then **Pfn(E)** has pushouts. Every pushout square in **E** of Proposition 2.3.16 is also a pushout of **Pfn(E)**, that is it is hereditary.

Next we consider Ehrig's double pushout approach [EKL90] in our category **G**, that is, assume that the following two squares are pushouts in **G**and that m is an injective function.

$$\begin{array}{ccccc} < A, \alpha > & \xleftarrow{m} & < D, \delta > & \xrightarrow{f} & < B, \beta > \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle s} & & \downarrow{\scriptstyle h} \\ < G, \xi > & \xleftarrow[n]{} & < E, \varepsilon > & \xrightarrow[k]{} & < H, \eta > \end{array}$$

Then $\delta m \sqsubseteq m\alpha$, $\delta s \sqsubseteq s\varepsilon$, $\delta f \sqsubseteq f\beta$, $\xi = g^\sharp \alpha g \sqcup n^\sharp \varepsilon n$ and $\eta = h^\sharp \beta h \sqcup k^\sharp \varepsilon k$ by **4.1.1**. Since $nn^\sharp = \mathrm{id}_E$ by the pushout property it is easy to see that $s^\sharp \delta s \sqsubseteq \varepsilon$ and

$$\begin{aligned} n(\xi - g^\sharp \alpha g)n^\sharp &= (ng^\sharp \alpha g n^\sharp \sqcup nn^\sharp \varepsilon nn^\sharp) - ng^\sharp \alpha g n^\sharp \quad (\text{by } \mathbf{2.1.4}) \\ &= (ng^\sharp \alpha g n^\sharp \sqcup \varepsilon) - ng^\sharp \alpha g n^\sharp \quad (nn^\sharp = \mathrm{id}_E) \\ &= \varepsilon - ng^\sharp \alpha g n^\sharp \\ &\sqsubseteq \varepsilon. \end{aligned}$$

Hence $n(\xi - g^\sharp \alpha g)n^\sharp \sqcup s^\sharp \delta s \sqsubseteq \varepsilon$. Now put $\hat{\varepsilon} = n(\xi - g^\sharp \alpha g)n^\sharp \sqcup s^\sharp \delta s$. From $n^\sharp \varepsilon n - g^\sharp \alpha g = n^\sharp n(n^\sharp \varepsilon n - g^\sharp \alpha g)n^\sharp n$ (by **2.1.7**) $= n^\sharp(\varepsilon - ng^\sharp \alpha g n^\sharp)n$ (by **2.1.2**) and $nn^\sharp = \mathrm{id}_E$, we have

$$\begin{aligned} g^\sharp \alpha g \sqcup n^\sharp \hat{\varepsilon} n &= g^\sharp \alpha g \sqcup n^\sharp n(\xi - g^\sharp \alpha g)n^\sharp n \sqcup n^\sharp s^\sharp \delta s n \quad (\text{by } \mathbf{2.1.4}) \\ &= g^\sharp \alpha g \sqcup n^\sharp(\varepsilon - ng^\sharp \alpha g n^\sharp)n \sqcup n^\sharp s^\sharp \delta s n \\ &= g^\sharp \alpha g \sqcup (n^\sharp \varepsilon n - g^\sharp \alpha g) \sqcup n^\sharp s^\sharp \delta s n \\ &= g^\sharp \alpha g \sqcup n^\sharp \varepsilon n \sqcup n^\sharp s^\sharp \delta s n \\ &= g^\sharp \alpha g \sqcup n^\sharp \varepsilon n \quad (s^\sharp \delta s \sqsubseteq \varepsilon) \end{aligned}$$

$$= \xi.$$

Thus $\hat{\varepsilon} : E \rightharpoonup E$ is the least relation such that $s^\sharp \delta s \sqsubseteq \hat{\varepsilon}$ and $\xi = g^\sharp \alpha g \sqcup n^\sharp \hat{\varepsilon} n$. Hence it is reasonable to assume that $\varepsilon = \hat{\varepsilon}$. In this case we have

$$\begin{aligned} \eta &= h^\sharp \beta h \sqcup k^\sharp n(\xi - g^\sharp \alpha g) n^\sharp k \sqcup k^\sharp s^\sharp \delta s k \\ &= h^\sharp \beta h \sqcup k^\sharp n(\xi - g^\sharp \alpha g) n^\sharp k \sqcup h^\sharp f^\sharp \delta f h \quad (fh = sk) \\ &= h^\sharp \beta h \sqcup k^\sharp n(\xi - g^\sharp \alpha g) n^\sharp k \quad (f^\sharp \delta f \sqsubseteq \beta). \end{aligned}$$

This shows that $\hat{\varepsilon} : E \rightharpoonup E$ is the least relation ε which makes the above squares pushouts. In our category of graphs **G**the pushout complement is not always exist and not unique (cf. **4.3.1**). If there exists a pushout complement, our rewriting using single pushout is coincide to the rewriting using double pushout which uses the least pushout complement.

Finally we consider the boundary graphs (or B-graphs) due to Okada and Hayashi [OH91] in **G**. If a matching $g :< A, \alpha > \rightarrow < G, \xi >$ to $p = (< A, \alpha >, < B, \beta >, f : A \rightarrow B)$ is an injective morphism of graphs such that $deg(g(a)) = deg(a)$ for each $a \in A$ on which f is undefined, then the rewritings coincides with those of B-graphs.

4.3 Examples of graph rewritings

In this section a few examples related to graph rewritings are listed. The first example shows that pushout-complements are not unique in **G**.

4.3.1 Let $\alpha, \beta, \gamma : A \rightharpoonup A$ be relations with $\alpha \sqsubseteq \gamma \sqsubseteq \beta$. Then because of **4.1.1** the square

$$\begin{array}{ccc} < A, \alpha > & \xrightarrow{\mathrm{id}_A} & < A, \beta > \\ {\scriptstyle \mathrm{id}_A}\downarrow & & \downarrow{\scriptstyle \mathrm{id}_A} \\ < A, \gamma > & \xrightarrow[\mathrm{id}_A]{} & < A, \beta > \end{array}$$

is a pushout in the category of graphs and morphisms between them. Therefore the square is a pushout for any choice of γ satisfying $\alpha \sqsubseteq \gamma \sqsubseteq \beta$. The choice of $\hat{\varepsilon}$ in **4.2** means the most economical way to have pushout-complements.

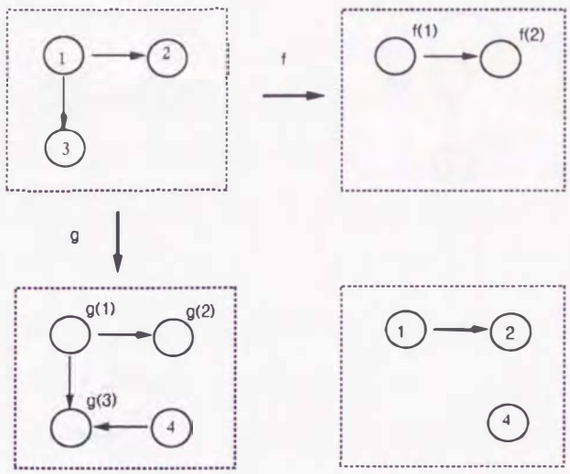


Figure 4.2:

Next we present two simple examples of graph rewritings to which conventional graph rewritings cannot be applied.

4.3.2 In **Figure 4.2** g is a neat morphism of graphs with respect to theories of Ehrig [EKL90], Raoult [Rao84] and ours. But f is not a morphism of graphs and it is not worth to be a rewriting rule in the sense of Raoult [Rao84]. On the other hand f means a fast production in the double pushout aproach of [EKL90] but unfortunately the necessary pushout-complement does not exist since the gluing condition is not satisfied. However we have the bottom right resultant graph by applying our formalization.

4.3.3 In **Figure 4.3** g is a morphism of graphs and f is a partial morphism of graphs in all theories of Ehrig [EKL90], Raoult [Rao84] and ours. However graph rewritings of Ehrig [EKL90] and Raoult [Rao84] does not work again because the gluing conditions are not valid. In this case the resultant graph given by our graph rewritings is one point graph without edges.

The final example indicates a reason why matchings must be morphisms of graphs in the definition **4.1.3** of graph rewritings.

4.3.4 Recall that matchings to rewriting rules are defined to be morphisms of graphs but not partial morphisms (Cf. **4.1.3**). We now observe what happens when matchings

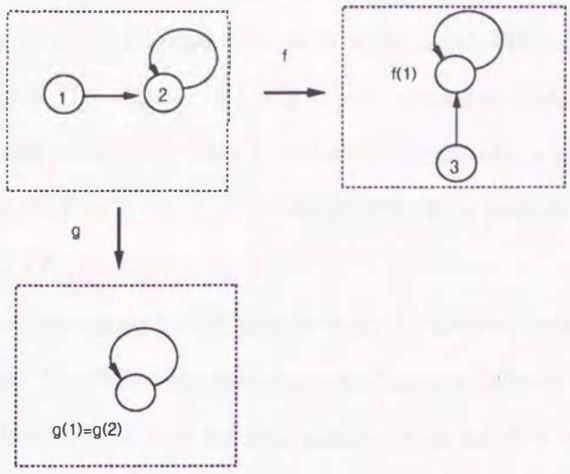


Figure 4.3:

are allowed to be partial morphisms of graphs. First we note that any couple of rewriting rules being partial morphisms of graphs commute, because rewriting squares are pushouts in the category of graphs and partial morphisms by **4.1.4**. Hence every set of rewriting rules consisting of partial morphisms of graphs is strongly confluent, which seems to exceed. Let $p = (< A, \alpha >, < B, \beta >, f : A \to B)$ and assume that $f(A) = B$ and there exists $a \in A$ such that f is undefined on a and a has no loops. (This rewriting rule p is not so special.) For any vertex v of an arbitrary graph $< G, \xi >$, define a matching $g :< A, \alpha > \to < G, \xi >$ such that $g(a) = v$ and undefined otherwise. Then g is in fact a partial morphism of graphs. The resultant graph H after applying p along g is a graph obtained by subtracting from G the vertex v and all edges connected with v. Thus this claims that any finite graph is reduced into the empty graph by iterating applications of p. Therefore these graph rewritings are nonsense.

4.4 Rewritings for graphs with labeled edges

In this section we first define graphs with labeled edges and partial morphisms between them, and a primitive pushout construction similar to **4.1.1** is stated for graphs with labeled edges. The readers may easily understand analogies with results in the section

4.1 are also valid in this case.

Let Σ be a set of labels. *A graph* $< A, \alpha >$ *with* Σ*-labelled edges* is a pair of a set A and a collection $\alpha = \{\alpha_\sigma : A \rightharpoonup A \mid \sigma \in \Sigma\}$ of relations indexed by Σ. A *partial morphism* f of a graph $< A, \alpha >$ with Σ -labeled edges into a graph $< B, \beta >$ with Σ-labelled edges, denoted by $f :< A, \alpha > \rightarrow < B, \beta >$, is a partial function $f : A \rightarrow B$ satisfying $d(f)\alpha_\sigma f \sqsubseteq f\beta_\sigma$ for all $\sigma \in \Sigma$.

Similarly we have the category of graphs with Σ-labelled edges and partial morphisms between them. The following theorem constructs a primitive pushout for a pair of partial functions from a common set into graphs with labeled edges.

Theorem 4.4.1 *If* $< B, \beta >$ *and* $< C, \gamma >$ *are graphs with* Σ*-labelled edges and if the square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

is a pushout in **Pfn**, *then* $h :< B, \beta > \rightarrow < D, \delta >$ *and* $k :< C, \gamma > \rightarrow < D, \delta >$ *are partial morphisms of graphs with* Σ*-labelled edges, where* $\delta_\sigma = h^\sharp \beta_\sigma h \sqcup k^\sharp \gamma_\sigma k$ *for each* $\sigma \in \Sigma$. *Moreover, if* $h' :< B, \beta > \rightarrow < D', \delta' >$ *and* $k' :< C, \gamma > \rightarrow < D', \delta' >$ *are partial morphisms of graphs with* Σ*-labelled edges satisfying* $fh' = gk'$, *then there exists a unique partial morphism* $t :< D, \delta > \rightarrow < D', \delta' >$ *of graphs with* Σ*-labelled edges such that* $h' = ht$ *and* $k' = kt$.

Similarly we have the following corollary from the last theorem.

Corollary 4.4.2 *The category of graphs with* Σ*-labelled edges and partial morphisms between them has pushouts.*

Remark. A graph $< A, \alpha >$ with $\alpha^\sharp = \alpha$ is just an undirected graph. Hence almost all results in this note are also valid for undirected graphs.

Example 4.4.3 Let $\Sigma = N$ be the set of natural numbers. A graph $< A, \alpha >$ with Σ-labelled edges satisfying the following conditions:

(1) α_i is a partial function for any $i \in N$, i.e. $\alpha_i \alpha_i^\sharp = \mathrm{id}_A$,

(2) $\alpha_j \sqsubseteq \alpha_i$ for any $i \leq j$ $(i, j \in N)$,

is equivalence to Raoult's graph $< A, \delta_A >$ where $\delta_A : A \to A^*(\delta_A(a) = \alpha_1(a) \cdots \alpha_i(a) \cdots)$. Since Raoult's graph morphisms are graph morphisms in **G**, Raoult's category of graphs is a subcategory of **G**. Raoult showed an sufficient condition for existence of pushouts which corresponds to the pushout closedness.

Chapter 5

Graph rewriting system using graph terms

In this chapter, we introduce a graph rewriting system based on a model introduced in Chapter 4, using a symbolical object called a graph term. In Section 5.1, new notations for graphs and graph terms and the interpretation of graph terms is given. In Section 5.2 we extend a graph term to contain a graph which have node label. We expand the graphs to infinite trees and regard a graph as an intermediate object between terms and infinite trees. Using the theory of infinite trees and finite automata, an expanded graph is charactereized as an infinite tree accepted by an tree automaton defined by the graph itself. In Section 5.3, we define using the notions not only simple rewritings but also parallel rewritings. The symbolical notation of graphs and rewriting rules is helpful to define a reading region and a writing region precisely which contribute to clear the concept of sequentially simulatable parallel rewritings. We show a sufficient condition that parallel rewritings are sequentially simulatable. Examples of graph rewriting systems which contain Turner's SK-reduction machine [Tur79] are given in Section 5.4. We reconfirm in our framework that the fact of parallel rewritings of the SK-reduction is sequentially simulatable. In Section 5.5, we show a graph rewriting system which solves equations of regular expressions as a meaningful example of non sequentially simulatable reductions. The rewriting rules are not sequentially simulatable, but it is proved that every parallel reduction leads to the right solution of

the equations. Examples of executions of the rewritings are given in Section 5.6.

5.1 Symbolic graphs and graph terms

We define the graph as a symbolical objects. A symbolical graph is defined a finite set of symbolical graph terms. The interpretation of these symbolical objects correspond to E-labeled graphs defined above. Let S be a *set of identifiers*.

A *graph term* is defined as follows:

(1) If $s \in S$, then s and $s[]$ are graph terms.

(2) If $s_0 \in S$, $e_1, \cdots, e_n \in E$ and $t_1, \cdots, t_n$ are graph terms, then $s_0[e_1 : t_1, \cdots, e_n : t_n]$ is a graph term.

A graph term is called *simple*, if it is defined by (1) or defined by (2) when every t_i is an element of S.

Definition 5.1.1 *For a graph term t, we define the root identifier $rt(t)$, the set $Id(t)$ of identifiers, the set $Int(t)$ of internal identifiers, the set ∂t of leaf identifiers, and the incidence relation $\xi_e(t) : Id(t) \rightharpoonup Id(t)$ for each $e \in E$.*

(1) *If $s \in S$, then $rt(s) = s$, $Id(s) = \{s\}$, $Int(s) = \phi$, $\partial s = \{s\}$, $\xi_e(s) = \phi$, $rt(s[]) = s$, $Id(s[]) = \{s\}$, $Int(s[]) = \{s\}$, $\partial s[] = \{s\}$, and $\xi_e(s[]) = \phi$.*

(2) *If $s_0 \in S$, $e_1, \cdots, e_n \in E$ and $t_1, \cdots, t_n$ are graph terms, then*

$$
\begin{aligned}
rt(s_0[e_1 : t_1, \cdots, e_n : t_n]) &= s_0 \\
Id(s_0[e_1 : t_1, \cdots, e_n : t_n]) &= \{s_0\} \cup \bigcup_{i=1}^{n} Id(t_i), \\
Int(s_0[e_1 : t_1, \cdots, e_n : t_n]) &= \{s_0\} \cup \bigcup_{i=1}^{n} Int(t_i), \\
\partial s_0[e_1 : t_1, \cdots, e_n : t_n] &= \bigcup_{i=1}^{n} \partial t_i, \\
\xi_e(s_0[e_1 : t_1, \cdots, e_n : t_n]) &= \{(s_0, rt(t_i)) | e = e_i\} \cup \bigcup_{i=1}^{n} \xi_e(t_i).
\end{aligned}
$$

For a finite set X of graph terms, the set $\{rt(x)|x \in X\}$ of all root identifiers of X is denoted by $Rt(X)$. An interpretation $|X|$ of a finite set X of graph terms is an E-labbeled graph $|X| =< Rt(X), \xi(X) >$, where $\xi_e(X) = \bigcup_{t \in X} \xi_e(t)$.

A finite set G of simple graph terms is a *symbolic graph* if it satisfies following conditions (1),(2) and (3):

(1) If $x \neq y$ then $rt(x) \neq rt(y)$ for any $x, y \in G$,

(2) If $x = s_0[e_1 : s_1, e_2 : s_2, \cdots, e_n : s_n] \in G$, there exists $y_i \in G$ such that $s_i = rt(y_i)$ for all $i = 1, \cdots, n$,

(3) If $x = s_0[e_1 : s_1, e_2 : s_2, \cdots, e_n : s_n] \in G$, then $e_i \neq e_j$ or $s_i \neq s_j$ for any $i \neq j$.

Since every element of a symbolic graph G has a different identifier, we can uniquely define the simple graph term $\exp_G(s)$ for $s \in Rt(G)$ satisfying $\exp_G(rt(x)) = x$ for all $x \in G$. A *pointed symbolic graph* is a pair $< rt(x), G >$ of a symbolic graph G and an element $rt(x) \in S$ for some $x \in G$.

Definition 5.1.2 *For a graph term t we define a finite set $G(t)$ of simple graph terms as follows:*

(1) *If $s \in S$, then $G(s) = \{s\}$ and $G(s[]) = \{s[]\}$.*

(2) *If $s_0 \in S$, $e_1, \cdots, e_n \in E$ and $t_1, \cdots, t_n$ are graph terms, then*

$$G(s_0[e_1 : t_1, \cdots, e_n : t_n]) = \{s_0[e_1 : rt(t_1), \cdots, e_n : rt(t_n)]\} \cup \bigcup_{i=1}^{n} G(t_i).$$

For any graph term t, a finite set $G(t)$ of graph terms is not always a symbolic graphs. So we restrict some more conditions for the graph term.

We define a *strict (graph) term* as follows:

(1) If $s \in S$, then s is a strict term,

(2) If $e_1, \cdots, e_n \in E$, $s_0 \in S - \cup_{i=1}^{n} Int(t_i)$ and $t_1, \cdots, t_n$ are strict terms such that $Int(t_i) \cap Int(t_j) = \phi$ for $i \neq j$, then $s_0[(e_1 : t_1), \cdots, (e_n : t_n)]$ is a strict term.

We note that if t is a strict term, then $G(t)$ is a symbolic graph.

Definition 5.1.3 *For a graph term t and a function $g : A \to B$ such that $Id(t) \subset A \subset S$ and $B \subset S$, we define a graph term $g^*(t)$ as follows:*

(1) *If $s \in S$, then $g^*(s) = g(s)$ and $g^*(s[]) = g(s)[]$.*

(2) *If $s_0 \in S$, $e_1, \cdots, e_n \in E$ and $t_1, \cdots, t_n$ are graph terms, then*

$$g^*(s_0[(e_1 : t_1), \cdots, (e_n : t_n)]) = g(s_0)[(e_1 : g(t_1)), \cdots, (e_n : g(t_n))].$$

The next proposition guarantees that we can construct the rewriting graphs using a symbolical substitution of graph terms. The element of the set ∂t operates a variables in the rule of term rewritings.

Proposition 5.1.4 *Let t be a strict term and let B, C and X be symbolic graphs with $B \subset G(t)$ and $Rt(B) \subset Rt(C)$, $i : Rt(B) \to Rt(G(t))$, $j : Rt(B) \to Rt(C)$ inclusion functions and $g : Rt(G(t)) \to Rt(X)$ a function. If $\exp_{G(g^*(t))}(g^*(Int(t))) \subset X$, $\exp_{G(t)}(\partial t) \subset B$, $gg^\sharp\Omega_{Rt(G(t))} \subset ii^\sharp\Omega_{Rt(G(t))}$, $g(Rt(X)) \cap (Rt(C) - Rt(B)) = \phi$ and the square*

$$\begin{array}{ccc} |G(t)| & \xrightarrow{f} & |C| \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ |X| & \xrightarrow[k]{} & <H, \eta> \end{array}$$

is a graph rewriting square, then then

$$<H, \eta> \cong |(G - \exp_{G(g(t))}(Int(t))) \cup \exp_{g(B)}(g(Rt(B) \cap Int(t))) \cup \exp_C(Rt(C) - Rt(B))|,$$

where $f = i^\sharp j$.

■

We note that if $B = G(t)$ then $gg^\sharp\Omega_{Rt(G(t))} \subset ii^\sharp\Omega_{Rt(G(t))}$. We only treat the case $B = G(t)$ in the graph reduction system defined in Section 5.3.

Definition 5.1.5 *Let G be a symbolic graph. we define a subset $P_G(x, y)$ of E^* for $x, y \in Rt(G)$ as follows:*

(1) *$\varepsilon \in Path_G(s, s)$ for any $s \in Rt(G)$,*

(2) $e_i \in Path_G(s, s_i)$ $(i = 1, 2, \cdots, n)$ *for a simple graph term* $s[e_1 : s_1, e_2 : s_2, \cdots, e_n : s_n] \in G$.

(3) *If* $a \in Path_G(x, s)$ *and* $b \in Path_G(s, y)$ *then* $ab \in Path_G(x, y)$ *for any* $s \in Rt(G)$.

We call an element of $P_G(x, y)$ *a path from* x *to* y.

Definition 5.1.6 *A pointed symbolic graph* $< s, G >$ *is* reachable *if* $Path_G(s, x) \neq \phi$ *for any* $x \in Rt(G)$.

Proposition 5.1.7 *If a pointed symbolic graph* $< s, G >$ *is reachable then there exists a g-term* t *satisfying* $SG(t) = G$ *and* $Rt(t) = s$.

■

Example 5.1.8 *Let* $E = \{a,\ b\}$ *and* $S = N$. *Then* $\{1[a:2,\ b:2],\ 2[a:1,\ b:2]\}$ *is a symbolic graph (cf. Fig. 5.1).*

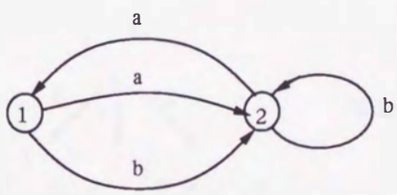


Figure 5.1: *An example of graph*

Example 5.1.9 *Let* E *be a one point set* $\{*\}$ *and* $S = N$*(a set of natural numbers). We omit to denote an element of* E.

- $G = \{0[1, 2], 1[2], 2[]\}$ *is a symbolic graph.*
- $Path_G(0, 1) = \{*\}$, $Path_G(0, 2) = \{*, **\}$, $Path_G(1, 0) = \phi$ *and* $Path_G(1, 2) = \{*\}$.
- $< 0, G >$ *is reachable.* $< 1, G >$ *is not reachable.*

- $Int(1[2]) = \{1\}$, $Rt(1[2]) = \{1,2\}$ *and* $G(1[2]) = \{1[2]\}$.
- $t = 0[1[2], 2[]]$ *is a strict term.* $G(t) = G$ *and* $rt(t) = s$.
- *graph term* $t = 0[1[2[]], 2[]]$ *is not strict but* $SG(t) = G$.
- *graph term* $u = 0[1[2], 2]$ *is strict and* $|t| = |u|$.
- *A set* $H = \{0[1], 0[2], 1[2]\}$ *of simple graph terms is not a symbolic graph and* $|H| = |\{0[1], 1[]\}|$.

Example 5.1.10 *General terms are expressed by graph term or symbolic graph as follows. We assume* $S = N$. *Let* $E = N + F + C$ *where* F *and* C *are a set of function symbols and a set of constant symbols, respectively. A constant(function)* a *is expressed* $\cdot \rightarrow^0 \cdot \rightarrow^a \cdot$, *because it is convenient to share it. We omit denoting a label in* N *when it is clear from the position of the term in the expressions. For example,* $0[x, y]$ *and* $0[3[f:4], 5]$ *means* $0[1:x, 2:y]$ *and* $0[1:3[f:4], 2:5]$, *respectively.*

Term $f(b, g(b, c))$ *is expressed by* $0[1[f:4], 2[9[b:11]], 3[5[g:8], 6[9], 7[10[c:12]]]]$.

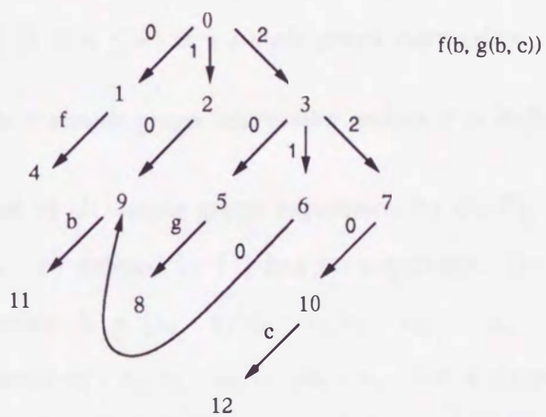


Figure 5.2: *An example of g-term expressing a term*

5.2 Graph terms and tree automata

In thie section, we extend a graph term to contain a graph which have node label. We expand the graphs to infinite trees and regard a graph as an intermediate object between terms and infinite trees. Using the theory of infinite trees and finite automata, an expanded graph is charactereized as an infinite tree accepted by an tree automaton defined by the graph itself.

Let Σ_N and Σ_E be sets. We call these sets the *set of labels for nodes* and the *set of labels for edges*, respectively. Let the set of identifiers S be $\mathcal{N}$ the set of natural numbers. We assume Spc is the set $\{(,),[,],:,\perp\}$ of special symbols. For a technical reason, we assume Σ_N and Σ_E do not intersect with Spc $(i.e. \Sigma_N \cap Spc = \phi$ and $\Sigma_E \cap Spc = \phi)$. Let k be a positive integer. The integer k bounds the maximum branches of graphs.

Definition 5.2.1 *We define simple graph expressions bounded to k branches as follows:*

(1) *For $s_0 \in S$ and $x \in \Sigma_N$, $(s_0 : x)$ is a simple graph expression.*

(2) *For $s_i \in S$ $(i = 0, \cdots, n)$, $a_i \in \Sigma_E$ $(i = 1, \cdots, n)$ and $x \in \Sigma_N$, $(s_0 : x)[a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n]$ $(1 \leq n \leq k)$ is a simple graph expression.*

(3) *Nothing else is a simple graph expression unless it is defined by (1) and (2).*

We denote the set of all simple graph expression by SGE_k. we say a simple graph expression $X = (s_0 : x)$ defined by (1) has no argument. On the other hand, for a simple graph expression $X = (s_0 : x)[a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n]$ defined by (2), we say X has n arguments $a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n$. For a simple graph expression X defined by (1) or (2), we call s_0 the *identifier* of X and denote it by $Id(X)$.

Definition 5.2.2 *A finite subset G of SGE_k is a graph bounded to k branches if it satisfies the following conditions:*

(1) *For any X, $Y \in G$ $(X \neq Y)$, $Id(X) \neq Id(Y)$,*

(2) *For any* $X \in G$ *and any argument* $a : s$ *of* X, *there exists* $Y \in G$ *such that* $s = Id(Y)$.

We denote by $Graph_k$ the set of all graphs bounded to k branches. The union set $\cup_k Graph_k$ is denoted by $Graph$.

For a graph G, we use the notation $Id(G)$ for the set $\{Id(X)|X \in G\}$. Since every element of G has a different identifier, we can define the function E_G from $Id(G)$ to G which satisfies $E_G(Id(X)) = X$ for any $X \in G$.

Example 5.2.3 *Let* $\Sigma_N = \{x,\ y\}$, $\Sigma_E = \{a,\ b\}$. $\{(1 : x)[a : 2,\ b : 2],\ (2 : y)[a : 1,\ b : 2]\}$ *is a graph bounded by 2 branches.*

Definition 5.2.4 *A pair* $< s,\ G >$ *of an element of* S *and a graph is a pointed graph if there exists an element* X *of* G *such that* $Id(X) = s$. *We denote the set of pointed graphs by* $Graph_k^*$

Definition 5.2.5 *We define a relation on* $Graph_k^*$. *For elements* $< s,\ G >$, $< t,\ H >$, $< s,\ G > \ \leq \ < t,\ H >$ *holds if there exists an injective map* $f : Id(G) \rightarrow Id(H)$ *satisfying the following conditions:*

(1) $f(s) = t$.

(2) *If* $X \in G$ *has arguments, then* X *and* $E_H(f(Id(X)))$ *have the same number of arguments.*

(3) *If* $X = (s_0 : x)[a_1 : s_1,\ a_2 : s_2,\ \cdots,\ a_n : s_n]$ *then there exist a bijection* $\tau : \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, n\}$, *such that* $E_H(f(Id(X))) = (f(s_0) : x)[a_{\tau(1)} : f(s_{\tau(1)}), a_{\tau(2)} : f(s_{\tau(2)}), \cdots, a_{\tau(n)} : f(s_{\tau(n)})]$.

It is easy to check that the relation $\leq$ is a preorder. We denote by $\sim$ the equivalence relation induced by the preorder $\leq$. That is, $X \sim Y$ if and only if $X \leq Y$ and $Y \leq X$. We denote by $Graph_k^*/\sim$ the quotient set of $Graph_k^*$ by the relation $\sim$.

Proposition 5.2.6 *For pointed graphs* $< s, G >$ *and* $< t, H >$, *if* $< s, G > \leq < t, H >$ *and* $E_G(s) = (s_0 : x)[a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n]$ *then* $< s_i, G > \leq < f(s_i), H >$ $(i = 1, 2, \cdots, n)$. ■

Let Σ_F be a set with $\Sigma_F \cap Spc = \phi$. We call Σ_F the set of function symbols. Generally every function symbol has an arity which determines the number of arguments. In this paper, we ignore arities. Any function symbol is assumed to have arbitrarily many but less than k arguments.

Definition 5.2.7 *We define* terms bounded to k branches *as follows:*

(1) *If f is an element of Σ_F, then f is a term.*

(2) *If $g \in \Sigma_F$ and f_1, f_2, $\cdots$, f_n $(1 \leq n \leq k)$ are terms, then $g[f_1, f_2, \cdots, f_n]$ is a term.*

(3) *Nothing else is a term unless it follows from a finite number of applications of (1) and (2).*

We denote by $Term_k$ the set of terms bounded to k branches.

In the rest of this section, we assume $\Sigma_E = \mathcal{N}$. We consider the relation between $Term_k$ and $Graph_k^*$. At the first step for embedding terms into $Graph_k^*$, we prepare the algorithm Emb.

It is easy to check the termination of the algorithm Emb using the structural induction of the definition of terms. We define the function $Embed : Term_k \rightarrow Graph_k^*/\sim$ by $Embed(t) =< s, G >$, where $Emb < S, t > = < s, G, S_O >$. The algorithm Emb has an ambiguity of choosing the identifier s from S_I, but it is not essential. If we have $Embed(t) =< s,\ G >$ by one choosing algorithm and $Embed(t) =< s',\ G' >$ by another choosing algorithm, it always holds $< s, G > \sim < s', G' >$. Furthermore, if $t \neq t'$ then $Embed(t) \not\sim Embed(t')$.

Let $\Sigma_k = \{1, 2, \cdots, k\}$ be a set which contains k symbols, and $T_k = \Sigma_k^*$ be the set of strings over Σ_k. We denote by ε an empty string. The set Σ_L is the set of labels. In this paper, we assume $\Sigma_L = \Sigma_F \cup \{\perp\}$. We note $\perp \notin \Sigma_F$ in the beginning of this paper. The infinite k-ary tree over Σ_L is the function $t : T_k \rightarrow \Sigma_L$. The set of all k-ary tree over Σ_L is denoted by $\Sigma_L^{T_k}$. The set Σ_L has the structure of flat semilattice. That is, Σ_L has an order relation $\leq$. For $x, y \in \Sigma_L$, it holds $x \leq y$ iff $x = \perp$. We can

---

Algorithm *Emb*

Input: $< S_I,\ t >$: a set of identifiers and a term

Output: $< s,\ G,\ S_O >$: an identifier, a graph and a set of identifiers.

If $t \in \Sigma_F$ **Then** choose an element $s \in S_I$
return $< s,\ \{(s : t)\},\ S - \{s\} >$

Else $t = g[f_1, f_2, \cdots, f_n]$
$< s_1, G_1, S_1 >= Emb < S, f_1 >$
$< s_2, G_2, S_2 >= Emb < S_1, f_2 >$
$\vdots$
$< s_n, G_n, S_n >= Emb < S_{n-1}, f_n >$
choose an element $s \in S_n$
return $< s, \{(s : g)[1 : s_1, 2 : s_2, \cdots, n : s_n]\}$
$\cup \bigcup_i G_i, S_n - \{s\} >$

---

extend this order into $\Sigma_L^{T_k}$ naturally. For elements t and s in $\Sigma_L^{T_k}$, we define $t \leq s$ iff $t(w) \leq s(w)$ for any $w \in T_k$.

Definition 5.2.8 *We define a function* $Inc : Term_k \rightarrow \Sigma_L^{T_k}$ *as follows:*

(1) *For* $t \in \Sigma_F$,

$$\begin{aligned} Inc(t)(\varepsilon) &= t \\ Inc(t)(mw) &= \bot \qquad (m \in \Sigma_k, w \in \Sigma_k^*). \end{aligned}$$

(2) *For* $t = g[f_1, f_2, \cdots, f_n]$,

$$\begin{aligned} Inc(t)(\varepsilon) &= g \\ Inc(t)(mw) &= \begin{cases} Inc(f_m)(w), & m \leq n \\ \bot, & n < m. \end{cases} \end{aligned}$$

It is easy to check that Inc is a monomorphism. We can induce an order $\leq$ in $Term_k$ from $\Sigma_L^{T_k}$ using the monomorphism.

Algorithm Exp

Input : $(i, < s, G >)$: a positive integer and a pointed graph.

If $i = 1$ **Then** $Exp(1, < s, G >) = x$
where $E_G(s) = (s_0 : x)[1 : s_1, \cdots, n : s_n]$.

Else $Exp(i, < s, G >)$
$= x[Exp(i-1, < s_1, G >), Exp(i-1, < s_2, G >),$
$\cdots, Exp(i-1, < s_n, G >)]$
where $E_G(s) = (s_0 : x)[1 : s_1, \cdots, n : s_n]$.

Proposition 5.2.9 *$Emb : Term_k \to Graph_k^*$ is an order preserving function.* ■

Lemma 5.2.10 (1) *For any directed sequence $(x_1 \leq x_2 \leq \cdots)$ of Σ_L, there exists a supremum $\bigvee_i x_i$.*

(2) *For any directed sequence $(t_1 \leq t_2 \leq \cdots)$ of $\Sigma_L^{T_k}$, there exists a supremum $\bigvee_i t_i$.*

■

Definition 5.2.11 *We define the subset $ITerm_k$ of $\Sigma_L^{T_k}$ as follows:*

$$ITerm_k = \{\bigvee_i Inc(t_i) | t_i \text{is a directed sequence of} Term_k\}$$

Since $(t \leq t \leq t \leq \cdots)$ is a directed sequence, using the monomorphism Inc, we can consider $Term_k$ as the subset of $ITerm_k$ naturally. At the first step for defining a function from $Graph_k^*$ into $ITerm_k$, we prepare a function $Exp : \mathcal{N} \times Graph_k^* \to Term_k$ using the algorithm Exp.

Proposition 5.2.12 *For a positive integer i and a pointed graph $< s, G >$, $Exp(i, < s, G >) \leq Exp(i+1, < s, G >)$. That is, $\{Exp(i, < s, G >) | 1 \leq i\}$ is a directed sequence.* ■

Definition 5.2.13 *The expanding function $Exp : Graph_k^* \to ITerm_k$ is defined by $Exp(< s, G >) = \bigvee_i Exp(i, < s, G >)$ for a pointed graph $< s, G >$.*

Proposition 5.2.14 (1) *For any pointed graph* $< s, G >$ *and* $< t, H >$, *if* $< s, G > \leq < t, H >$, *then* $Exp(< s, G >) \leq Exp(< t, H >)$. *So Exp is an order preserving function.*

(2) *For any pointed graph* $< s, G >$ *and* $< t, H >$, *if* $Exp(< s, G >) \leq Exp(< t, H >)$, *it does not always hold* $< s, G > \leq < t, H >$.

(*Proof*) (1) Let the order $< s, G > \leq < t, H >$ is determined by a function $f : Id(G) \to Id(H)$. We first note that for any element $X = (s_0 : x)[1 : s_1, 2 : s_2, \cdots, n : s_n]$ in G, it holds $E_H(f(Id(X))) = (f(s_0) : x)[1 : f(s_1), 2 : f(s_2), \cdots, n : f(s_n)]$ by the correspondence of edge labels. We show $Exp(i, < s, G >) = Exp(i, < t, H >)$ for any positive integer i using the induction on i. $Exp(1, < s, G >) = Exp(1, < t, H >)$ is trivial. By the hypothesis of the induction and Proposition 2.1.1, we have $Exp(i-1, < s_i, G >) = Exp(i-1, < f(s_i), H >)$. So we obtain $Exp(i, < s, G >) = Exp(i, < t, H >)$.

(2) Let $G = \{(1 : x)[1 : 2, 2 : 1], (2 : x)[1 : 1, 2 : 1]\}$ and $H = \{(1 : x)[1 : 1, 2 : 2], (2 : x)[1 : 1, 2 : 1]\}$ (cf.*Figure ref fig : gh.*). We assume a relation $< 1, G > \leq < 1, H >$ is determined a function $f : Id(G) \to Id(H)$. By the definition of the order, $f(1) = 1$. We compare the $X = (1 : x)[1 : 2, 2 : 1] \in G$ and $E_H(f(Id(X))) = (1 : x)[1 : 1, 2 : 2]$. By the correspondence of edge labels, an bijection $\tau : \{1, 2\} \to \{1, 2\}$ must be an identity function. But this bijection τ does not hold the condition of the order. So there is no map $f : Id(G) \to Id(H)$ which satisfies the order condition. That is $< s, G > \not\leq < t, H >$. Similarly we obtain $< t, H > \not\leq < s, G >$.

We show $Exp(i, < 1, G >) = Exp(i, < 1, H >) = Exp(i, < 2, G >) = Exp(i, < 2, H >)$ for any positive integer i using the induction on i. If $i = 1$ then $Exp(1, < 1, G >) = Exp(1, < 1, H >) = Exp(1, < 2, G >) = Exp(1, < 2, H >) = x$.

$Exp(i, < 1, G >) = x[Exp(i-1, < 2, G >), Exp(i-1, < 2, G >)]$

$Exp(i, < 2, G >) = x[Exp(i-1, < 1, G >), Exp(i-1, < 1, G >)]$

$Exp(i, < 1, H >) = x[Exp(i-1, < 1, H >), Exp(i-1, < 2, H >)]$

$Exp(i, < 2, H >) = x[Exp(i-1, < 1, H >), Exp(i-1, < 1, H >)]$

By the hypothesis of the induction, we have $Exp(i, < 1, G >) = Exp(i, < 1, H >) = Exp(i, < 2, G >) = Exp(i, < 2, H >)$. ■

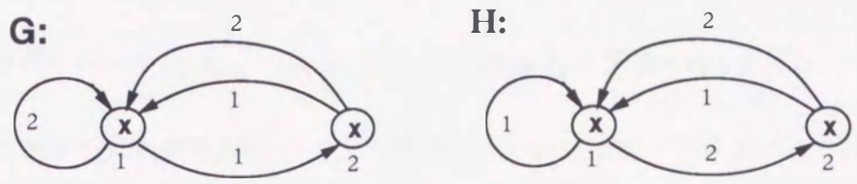


Figure 5.3:

Definition 5.2.15 *A finite k-ary tree automaton is a quintuple* $M = < S, \Sigma, \delta, s_0, F >$, *where*

(1) S *is a finite set of states,*

(2) δ *is a transite function* $\delta : S \times \Sigma \to S^k$, *where* S^k *is the k-fold product of* S,

(3) s_0 *is a state in* S *called the initial state,*

(4) F *is a subset of* S, *called the set of final states.*

Definition 5.2.16 *Let* $t : T_k \to \Sigma_F$ *be a k-ary tree over* Σ_F. *A run of* M *on* t *is a function* $r_t : T_k \to S$ *such that* $r_t(\varepsilon) = s_0$, $(r_t(1x), r_t(2x), \cdots, r_t(kx)) = \delta(r_t(x), t(x))$.

We note that r is defined uniquely by t. We say t is accepted by M if $r_t(T_k) \subset F$. We note this acceptance condition is the sense of $C_4 - acceptance$ [6].

The set $L(M)$ of k-ary tree over Σ is defined as $L(M) = \{t \in \Sigma_L^{T_k} | t$ is accepted by $M\}$. For a subset $L \subset \Sigma_L^{T_k}$, if $L = L(M)$ then we say L is accepted by M.

Definition 5.2.17 *A restricted k-ary tree automaton is a k-ary tree automaton* $M =< S, \Sigma, \delta, s_0, E >$ *which satisfies the following conditions:*

(1) $\Sigma = \Sigma_E \cup \{\perp\}$,

(2) S *contains special states* T *and* F,

(3) $E = S - \{F\}$,

(4) $\delta(F, a) =< F, F, \cdots, F >$ *for any* $a \in \Sigma$,

(5) $\delta(T, \perp) =< T, T, \cdots, T >$,

(6) *For* $\delta(s, a) =< s_1, s_2, \cdots, s_k >$, *if* $s_i = T$ *then* $s_j = T$ *for any* $j \geq i$.

(7) *For any* $s \in S$ *and* $a \in \Sigma$, *if* $\delta(s, a) =< s_1, s_2, \cdots, s_k > \neq < F, F, \cdots, F >$ *then* $s_i \neq F$ *for any* i.

(8) *For any* $s \in S$ $(s \neq F)$, *there exists only one element* $a_s \in \Sigma$, *such that* $\delta(s, a_s) \neq < F, F, \cdots, F >$ *and* $\delta(s, a) =< F, F, \cdots, F >$ *for any* $a \neq a_s$.

The set of all restricted k-ary tree automata is denoted by $RAut_k$.

For a restricted k-ary tree automaton M, there is a unique k-ary tree accepted by M. That is, $L(M)$ is a singleton set $\{t_M\}$.

Proposition 5.2.18 *There is a one-to-one correspondence between* $RAut_k$ *and* $Graph_k^*$. *Further, if a restricted k-ary tree automaton* M *corresponds to a pointed graph* $< s, G >$, *then the k-ary tree* t_M *accepted by* M *is equal to* $Exp(< s, G >)$. *That is, the tree accepted by* M *is* $Exp(< s, G >)$.

(*Proof*) First we define a function $\Psi : RAut_k \rightarrow Graph_k^*$. Let $M =< S, \Sigma, \delta, s_0, E >$ be a restricted k-ary tree automaton. For an element $s \in S$ $(s \neq F)$, there exists an element $a_s \in \Sigma$, such that $\delta(s, a_s) \neq < F, F, \cdots, F >$. We define a graph expression X_s for any $s \in S$. If $\delta(s, a_s) =< T, T, \cdots, T >$ then we put $X_s = (s : a_s)$. Otherwise we define $X_s = (s : a_x)[1 : s_1, \cdots, n : s_n]$ where $\delta(s, a_s) =< s_1, \cdots, s_k >$ and n is a maximum integer satisfying $s_n \neq T$. The pointed graph $\Psi(M) =< s, G >$ is defined by $s = s_0$ and $G = \{X_s | s \in S - \{T, F\}\}$.

It is easy to check $Exp(\Psi(M))$ is accepted by M.

Next we define a function $\Phi : Graph_k^* \rightarrow RAut_k$. Let $< s_0, G >$ be a pointed graph, $S = Id(G) \cup \{T, F\}$, and $s \in Id(G)$. If $E_G(s) = (s : a_s)$ then we define $\delta(s, a_s) =< T, T, \cdots, T >$ and $\delta(s, a) =< F, F, \cdots, F >$ for any $a \neq a_s$. If $E_G(s) = (s : a_s)[1 : s_1, \cdots, n : s_n]$ then we define $\delta(s, a_s) =< s_1, \cdots, s_n, T, \cdots, T >$ and $\delta(s, a) =< F, F, \cdots, F >$ for any $a \neq a_s$. The definition of $\delta(T, a)$ and $\delta(F, a)$ for $a \in \Sigma$ is

determined by conditions of a restricted tree automaton. So we complete the definition of $\Phi(< s_0, G >) =< S, \Sigma, \delta, s_0, E >$.

It is clear that $\Psi(\Phi(< s_0, G >)) =< s_0, G >$ for any pointed graph $< s_0, G >$ and $\Phi(\Psi(M)) = M$ for any restricted tree automaton M. ■

5.3 Graph rewriting system

A function $\sigma : D \to S$ is called an *assignment* for a strict term t if $Rt(G(t)) \subset D \subset S$. A procedure $Match$ is defined for an assignment σ for t and a pointed symbolic graph $< s, G >$, as follows:

Procedure *Match*

Input: a strict term t,
a pointed graph $< s, G >$,
and an assignment σ for t.

Case t **of**

$t = s$ $(s \in S)$:
{**If** $s = \sigma(t)$ **Then** success **Else** failure }

$t = s[]$ $(s \in S)$:
{**If** $\exp_G(s) = \sigma(t)[]$ **Then** success **Else** failure }

$t = s_0[e_1 : t_1, \cdots, e_n : t_n]$:
{**If** $\exp_G(s) = \sigma(t_0)[e_1 : s_1, \cdots, e_n : s_n]$ and
$Match(t_i, < s_i, G >, \sigma)$ succeeds
for any $i = 1, \cdots, n$.
Then success **Else** failure }

When the procedure $Match(t, \sigma, < s, G >)$ succeeds, we say a strict term t matches to a pointed symbolic graph $< s, G >$ by the assignment σ. In this case, we can consider $\sigma(t) \subset G$.

Example 5.3.1 *Let E be a one point set and $S = N$. We omit to denote an element of E. We consider a graph term $t = 0[1[2]]$, a symbolic graph $G = \{0[1], 1[0]\}$ and an assignment $\sigma = \{(0,0),(1,1),(2,0)\}$ for t. Then all of $Match(2, \sigma, < 0, G >)$, $Match(1[2], \sigma, < 1, G >)$ and $Match(t, \sigma, < 0, G >)$ succeed.*

Definition 5.3.2 *A* reduction rule *is a pair $T =< t_0, \{t_1, \cdots, t_n\} >$ of a strict term t_0 and a non empty finite set of simple graph terms $\{t_1, \cdots, t_n\}$ which satisfies $rt(t_i) \notin \partial t_0$ for $i = 1, \cdots, n$.*

Let $T =< t_0, T_0 >$ be a reduction rule. For a graph G and an assignment σ, if the conditions

(1) The procedure $Match(t_0, \sigma, < s, G >)$ succeeds for an element $s \in G$.

(2) $rt(\sigma(t_i)) \neq rt(\sigma(t_j))$ for any $t_i \neq t_j$ $(t_i, t_j \in T_0)$.

(3) $\sigma(rt(t_i)) \notin Rt(G)$ for any $rt(t_i) \notin Int(t_0)$.

satisfies then we can construct a symbolic graph

$$H = (G - \{\exp_G(Rt(\sigma(t_i)))|i = 1, \cdots, n\}) \cup \{\sigma(t_i)|i = 1, \cdots, n\}.$$

We say that the graph G is reduced to the graph H and denote $< G, s > \rightarrow_{T/\sigma} H$. We abbreviate $G \rightarrow_{T/\sigma} H$ when we need not concerning s.

Let $t = t_0$, $C = (G(t) - \{\exp_{G(t)}(rt(t_i))|i = 1, \cdots, n\}) \cup \{t_i|i = 1, \cdots, n\}$ and $B = (G(t) - \{\exp_{G(t)}(rt(t_i)))|i = 1, \cdots, n\}) \cup \{t_i|rt(t_i) \in Rt(G(t)), i = 1, \cdots, n\}$. The above conditions correspond to that of Proposition 5.1.4. So the interpretation graph $|H|$ is uniquely determined in the category of graphs.

We define the *reading region* $Rd(G \rightarrow_T H) = \sigma(Int(t_0))$ and the *writing region* $Wt(G \rightarrow_T H) = \{\sigma(rt(t_i))|i = 1, \cdots, n\}$.

Definition 5.3.3 A graph reduction system *is a triple $< S, E, R >$, where S is a set of identifiers, E is a label set for edges, and R is a set of reduction rules.*

We note that the set R of reduction rules may be an infinite set.

Next, we define the parallel reductions. Let $T^j =< t_0^j, T_0^j >$ $(j = 1, \cdots, m)$ are reduction rules. For a graph G if the conditions

(1) G is reducible to the graph H^j by reduction rules T^j $(< G, s^j > \rightarrow_{T^j/\sigma^j} H^j)$.

(2) For any $j_1 \neq j_2$, $Wt(G \rightarrow_{T^{j_1}/\sigma^{j_1}} H^{j_1}) \cap Wt(G \rightarrow_{T^{j_2}/\sigma j_2} H^{j_2}) = \phi$ $(j_1, j_2 = 1, \cdots, m)$.

satisfies then we define a symbolic graph

$$H = G - (\cup_j \{\exp_G(Rt(\sigma^j(t_i^j)))|i = 1, \cdots, n_j\}) \cup (\cup_j \{\sigma(t_i^j)|i = 1, \cdots, n_j\}).$$

We say that the graph G is parallel reduced to the graph H and denote $G \rightarrow_{\{T^1, \cdots, T^m\}} H$.

Roughly speaking, the condition of the parallel reduction is equivalent to the condition of the single reduction by a rule $T =< t_0, T_0 >$ where $t_0 = \cup_j t_0^j$ and $T_0 = \cup_j T_0^j$. The statement is not perfectly exact, because t_0 may not be a strict term.

Definition 5.3.4 *A parallel reduction* $G \rightarrow_{\{T^1, \cdots, T^m\}} H$ *is* sequentially simulatable *iff there exists an bijection* $\tau : \{1, \cdots, m\} \rightarrow \{1, \cdots, m\}$ *and graphs* $G_1, \cdots, G_{m-1}$ *such that* $G \rightarrow_{t^{\tau(1)}} G_1 \rightarrow_{t^{\tau(2)}} \cdots \rightarrow_{t^{\tau(m-1)}} G_{m-1} \rightarrow_{t^{\tau(m)}} H$.

Proposition 5.3.5 *For reduction rules* $T^j =< t_0^j, \{t_1^j, \cdots, t_{n_j}^j\} >$ $(j = 1, \cdots, m)$, *the parallel reduction* $G \rightarrow_{\{T^1, \cdots, T^m\}} H$ *is sequentially simulatable, if* $Rd(G \rightarrow_{T^{j_1}} H^{j_1}) \cap Rd(G \rightarrow_{T^{j_2}} H^{j_2}) = \phi$ *for any* $j_1 \neq j_2$.

(*Proof*) We show the special case of $m = 2$. Let $H_j = \{\sigma^j(t_i^j)|i = 1, \cdots, n_j\}$ and $E_j = \{\exp_G(rt((\sigma^j(t_i^j)))|i = 1, \cdots, n_j\}$. Since $Wt(G \rightarrow_{T^1} H^1) \cap Wt(G \rightarrow_{T^2} H^2) = \phi$, we have $E_2 \cap H_1 = \phi$ then $H_1 - E_2 = H_1$.

$$\begin{aligned} H &= G - (E_1 \cup E_2) \cup (H_1 \cup H_2) \\ &= (G - E_1 - E_2) \cup H_1 \cup H_2 \\ &= (G - E_1 - E_2) \cup (H_1 - E_2) \cup H_2 \\ &= ((G - E_1) \cup H_1) - E_2 \cup H_2 \end{aligned}$$

Let $G_1 = G - E_1 \cup H_1$. Since $Int(t_0^1) \cap Int(t_0^2) = \phi$ and $Match(t_0^2, \sigma^2, < s^2, G >)$ succeeds, the procedure $Match(t_0^2, \sigma^2, < s^2, G_1 >)$ succeeds. So we obtain $G \rightarrow_{T^1} G_1 \rightarrow_{T^2} H$.

The cases of $m \geq 2$ are similarly showed. ■

5.4 Examples of graph reduction systems

In this section we show three kinds of examples. These examples show interesting properties of graph reduction systems such as parallel computabilities.

Example 1

We can simulate a term rewriting system using the graph reduction systems easily. At first, we show that a famous example of term rewriting system, *append*, can be denoted by a graph reduction system.

Σ_N contains special symbols C, A, and N. C and A stand for function symbols *cons* and *append*, respectively. N expresses the Nil, the null list. We assume $\Sigma_E = \{1, 2\}$. We omit denoting elements of Σ_E when it is clear from the position of the term in the expressions. The reduction rules $(t_0, \{t_1, t_2, \ldots, t_n\})$ for *append* are defined as follows:

$$
\begin{array}{lll}
(T_1) & t_0 = 0[1[C:3], 2[x, 4[5[A:7], 6[8[N:9], y]]]] \\
 & t_1 = 2[x, y] \\
(T_2) & t_0 = 0[1[A:3], 2[4[5[C:7], 6[x, y]], z]] \\
 & t_1 = 0[5, 2] \\
 & t_2 = 2[x, 4] \\
 & t_3 = 4[1, 6] \\
 & t_4 = 6[y, z]
\end{array}
$$

We consider parallel executions of the calculation of *append*. For any reduction $G \rightarrow_{T_2} H$, $Rd(G \rightarrow_{T_2} H) = Wt(G \rightarrow_{T_2} H)$. By Proposition 9, any parallel reduction of

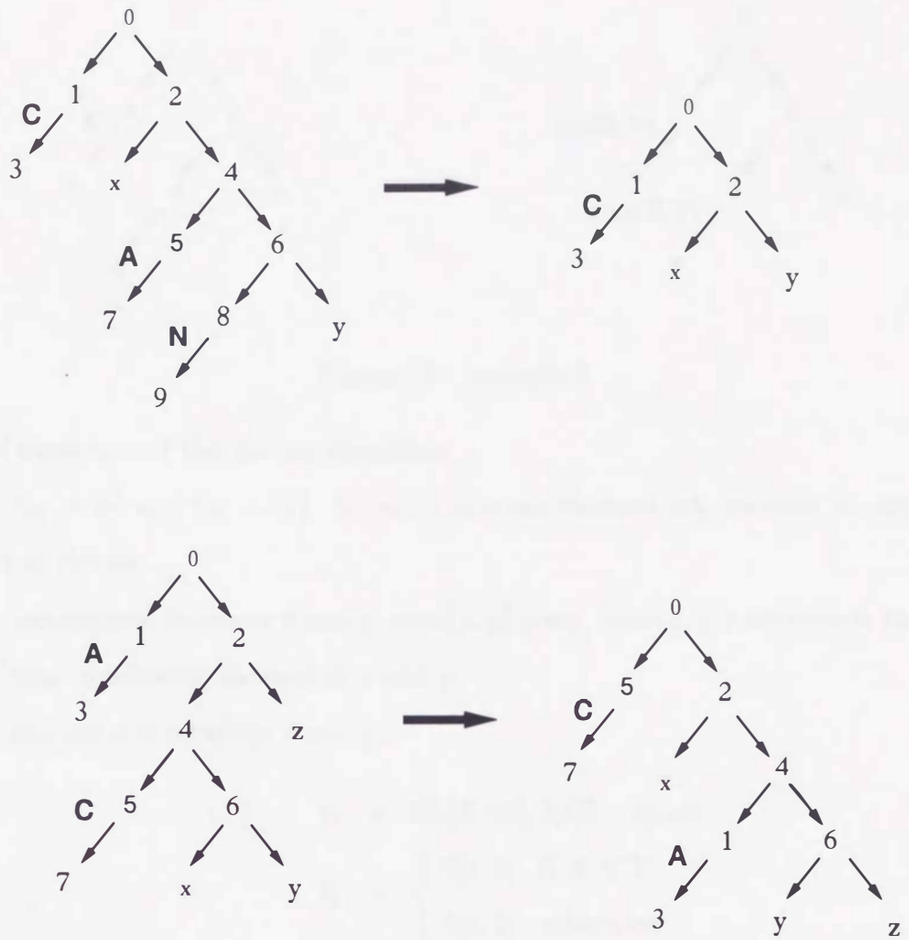


Figure 5.4: example 1

T_2 is sequentially simulatable. Further for any matching of T_1 and T_2, reading regions are disjoint. This fact guarantees any parallel reduction of the deduction system is sequentially simulatable.

Example 2

The second example is sorting. We put a sequence of numbers into the nodes of a graph which has only one edge. The system exchanges the numbers in two nodes adjoined by an edge, to increasing order. This algorithm is based on the famous bubble sorting. Using our argument of disjointness of reading regions, we guarantee the correctness of

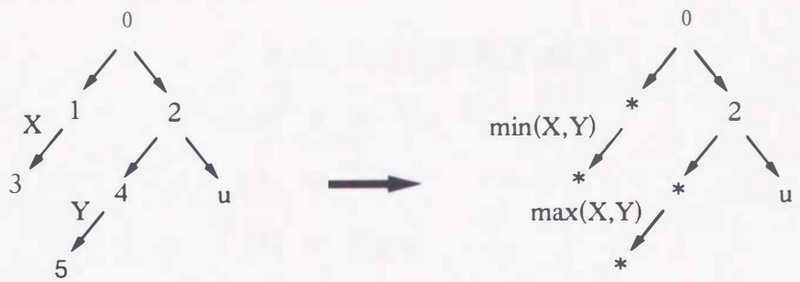


Figure 5.5: example 2

parallel execution of the sorting algorithm.

Let $\Sigma_N = \mathcal{N}$ and $\Sigma_E = \{1\}$. Since Σ_E is a one element set, we omit to denote an element of the set.

For any natural numbers x and y, $max(x, y)$ (resp. $min(x, y)$) represents the maximum (resp. minimum) element of x and y.

For any natural numbers x and y,

$$
\begin{aligned}
(T) \qquad t_0 &= 0[1[X:3], 2[4[Y:5], u]] \\
t_1 &= \begin{cases} 0[1,2] & \text{if } X \leq Y \\ 0[4,2] & \text{otherwise} \end{cases} \\
t_2 &= \begin{cases} 2[4,u] & \text{if } X \leq Y \\ 2[1,u] & \text{otherwise} \end{cases}
\end{aligned}
$$

For any reduction $G \rightarrow_T H$, $Rd(G \rightarrow_T H) = Wt(G \rightarrow_T H)$. Hence any parallel reduction of T is sequentially simulatable.

Example 3 (SK-reduction rules)

Let $E = \{1, 2, S, K, I, B, C, Y, P\}$ and $S = N \cup \{f, g, h, x, y\}$. We omit denoting 1, 2 of E when it is clear from the position of the term in the expressions. For example, $0[x, y]$ and $0[3[S:4], f]$ means $0[1:x, 2:y]$ and $0[1:3[S:4], 2:f]$, respectively. We define rewriting rules as follows. We write t and t_i's separately for a reduction rule $(t, \{t_1, \cdots, t_n\})$.

(1) Rule for $Sfgx \to fx(gx)$:

$$\begin{aligned} t &= 0[1[2[3[S:4], f], g], x] \\ t_1 &= 0[1, 2] \\ t_2 &= 1[f, x] \\ t_3 &= 2[g, x] \end{aligned}$$

(2) Rule for $Kxy \to x$:

$$\begin{aligned} t &= 0[1[2[3[K:4], x], y], z] \\ t_1 &= 0[x, z] \end{aligned}$$

(3) Rule for $Ix \to x$:

$$\begin{aligned} t &= 0[1[2[I:3], x], y] \\ t_1 &= 0[x, y] \end{aligned}$$

(4) Rule for $Bfgx \to f(gx)$:

$$\begin{aligned} t &= 0[1[2[3[B:4], f], g], x] \\ t_1 &= 0[f, 1] \\ t_2 &= 1[g, x] \end{aligned}$$

(5) Rule for $Cfgx \to fxg$:

$$\begin{aligned} t &= 0[1[2[3[C:4], f], g], x] \\ t_1 &= 0[1, g] \\ t_2 &= 1[f, x] \end{aligned}$$

(6) Rule for Yh:

$$\begin{aligned} t &= 0[1[Y:2], h] \\ t_1 &= 0[h, 0] \end{aligned}$$

Since reading reagions are disojoin for any two matchings, it always holds the condition of Proposition 5.3.5. We note that every parallel reduction of this example is sequential simulatable.

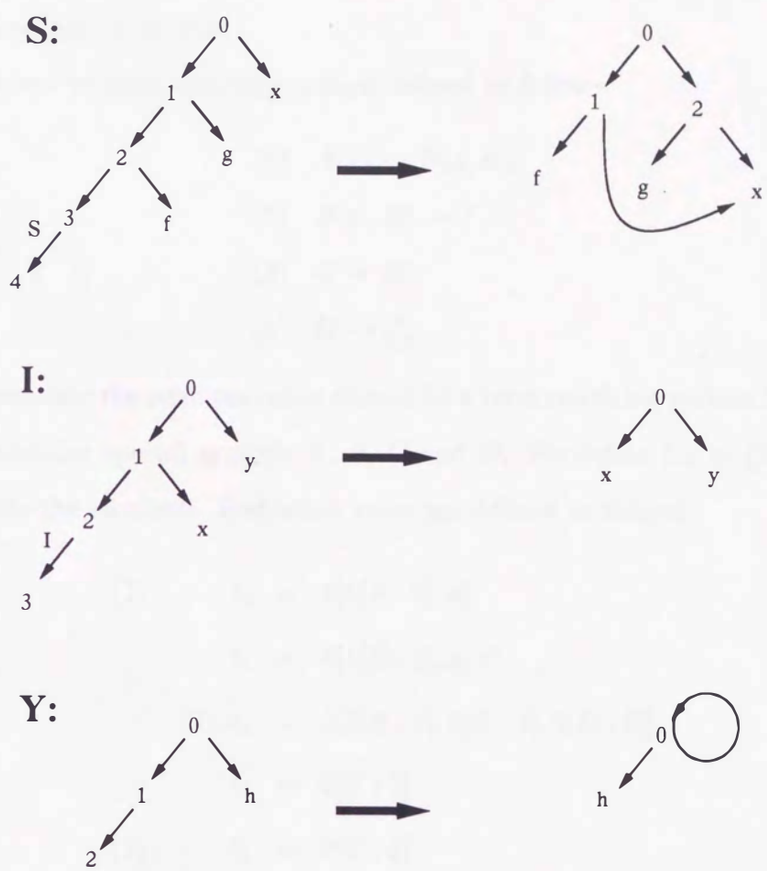


Figure 5.6: *SK-reduction rules*

$$A(C) \xrightarrow{(1)} B(C,C) \xrightarrow{(3)} B(C,D) \xrightarrow{(2)} F$$

Figure 5.7:

Example 4

The last example shows the difference of behaviors between a term rewriting system and a graph reduction system.

We consider the term rewriting system defined as follows:

$$\begin{aligned}
&(1)\quad A(x) \to B(x,x),\\
&(2)\quad B(C,D) \to F,\\
&(3)\quad C \to D,\\
&(4)\quad D \to C.
\end{aligned}$$

We can simulate the term rewriting system by a term rewriting system like Example 1. Let Σ_N contains special symbols A, B, C and D. We define $\Sigma_E = \{1, 2\}$, and we omit to denote the elements. Reduction rules are defined as follows:

$$\begin{aligned}
(T_1) \qquad t_0 &= 0[1[A:2], x]\\
t_1 &= 0[1[B:2], x, x]\\
(T_2) t_0 &= 0[1[B:2], 3[C:4], 5[D:6]]\\
t_1 &= 0[F:1]\\
(T_3) \qquad t_0 &= 0[C:1]\\
t_1 &= 0[D:1]\\
(T_4) \qquad t_0 &= 0[D:1]\\
t_1 &= 0[C:1]
\end{aligned}$$

In the term rewriting system, the term $A(C)$ has a normal form F. But the graph expression $A[C]$ can not be reduced to F. Because we do not reduce an only one argument in the graph expression $B[C,C]$, to D. We first forecast that the Church-Rosser property of a term rewriting system guarantees the correctness of simulation

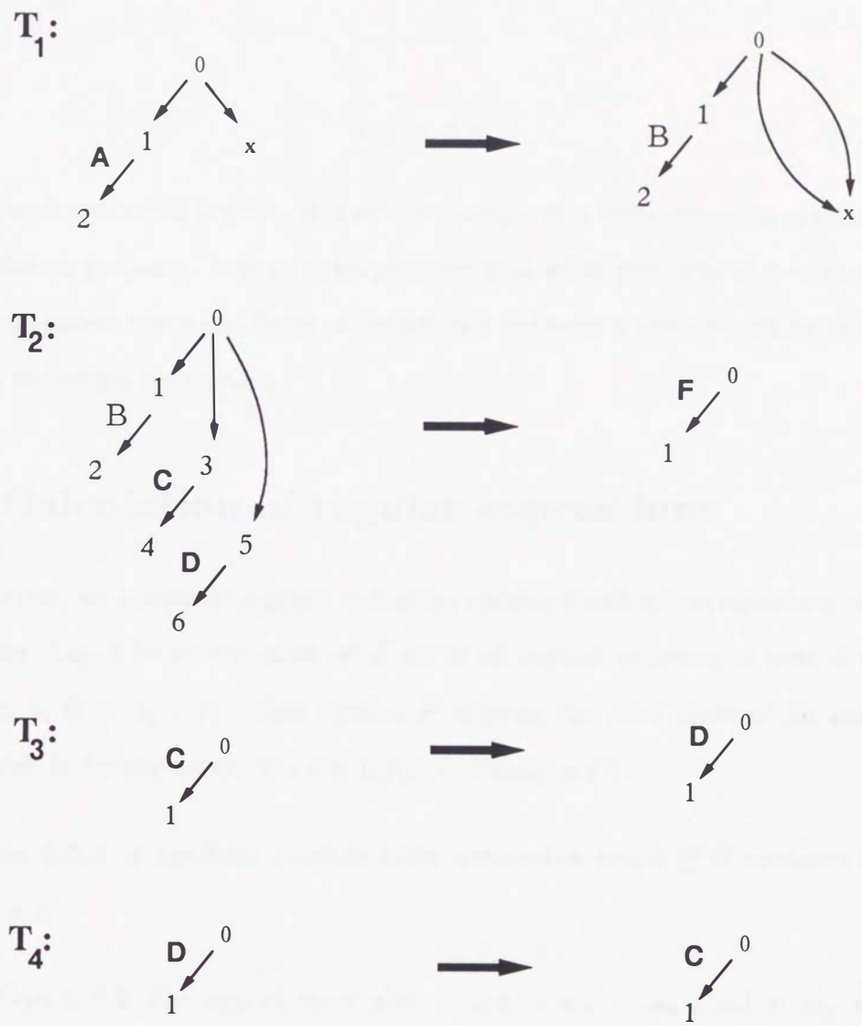


Figure 5.8: Example 4

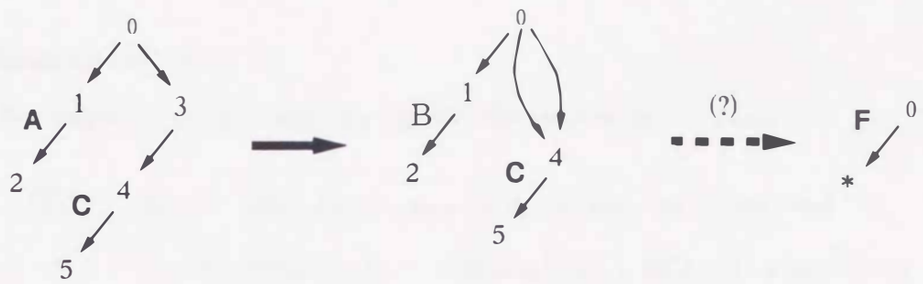


Figure 5.9:

using a graph reduction system. But above example of a term rewriting system has the Church-Rosser property. It is an open problem that what property of a term rewriting system guarantees the coincidence of behaviours between a term rewriting system and its graph reduction simulation.

5.5 Calculation of regular expressions

In this section, we construct a graph reduction system which solves equations of regular expressions. Let A be an alphabet set E set of all regular expressions over A and $S = \{F, x, y, x_i, y_i \ (i = 1, \cdots)\}$. This symbol F express the final state of an automaton graph which is defined later. We put $L_G(s) = Path_G(s, F)$.

Definition 5.5.1 *A symbolic graph G is an automaton graph iff G contains a simple graph term F.*

Proposition 5.5.2 *For any element $s[a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n] \in G$, $L_G(s) = \cup_i a_i L_G(s_i)$.*

For an element $s[a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n] \in G$, we can construct an equation of regular expressions, $W = \cup_i a_i W_i$, where W and W_i $(i = 1, 2, \cdots, n)$ stand for sets of regular expressions. Proposition 5.5.2 means that $L_G(s)$ and $L_G(s_i)$ $(i = 1, 2, \cdots, n)$ is a solution of the equation. We consider solving these equations constructed by G, using a graph reduction system.

Definition 5.5.3 *We define three kinds of reduction rules. Assume* $a, a_i, b_i \in E$.

(R) *Reducing rule for* x*:*

For any n, $1 \leq i \leq n$ *and any regular expressions* $(a_1, \cdots, a_n)$,

$$\begin{aligned} (T_1) \qquad t_0 &= x[a_1 : x_1, \cdots, a_{i-1} : x, a_i : x, a_{i+1} : x, \cdots, a_n : x_n] \\ t_1 &= x[(a_i^* a_1) : x_1, \cdots, (a_i^* a_{i-1}) : x_{i-1}, (a_i^* a_{i+1}) : x_{i+1}, (a_i^* a_n) : x_n] \end{aligned}$$

(E) *Expansion rule for* y*:*

For any n, m, $1 \leq i \leq m$ *and any regular expressions* $(a_1, \cdots, a_n, b_1, \cdots, b_m)$,

$$\begin{aligned} (T_1) \qquad t_0 &= x[b_1 : y_1, \cdots, b_{i-1} : y_{i-1}, b_i : y[a_1 : x_1, \\ &\qquad \cdots, a_n : x_n], b_{i+1} : y_{i+1}, \cdots, b_m : y_m] \\ t_1 &= x[b_1 : y_1, \cdots, b_{i-1} : y_{i-1}, (b_i a_1) : x_1, \\ &\qquad \cdots, (b_i a_n) : x_n, b_{i+1} : y_{i+1}, \cdots, b_m : y_m] \end{aligned}$$

(C) *Combination rule for* y*:*

For any n, $1 \leq i < j \leq n$ *and any regular expressions* $(a_1, \cdots, a_m)$,

$$\begin{aligned} (T_1) \qquad t_0 &= x[a_1 : y_1, \cdots, a_i : y, \cdots, a_j : y, \cdots, a_m : y_m] \\ t_1 &= x[a_1 : y_1, \cdots, (a_i + a_j) : y_i, \cdots, a_{j-1} : y_{j-1}, \\ &\qquad a_{j+1} : y_{j+1}, \cdots, a_m : y_m] \end{aligned}$$

Proposition 5.5.4 *For any automaton graph* G*, there exists an automaton graph* H *which is reduced from* G *applying a finite number of reduction rules such that any reduction rules can not apply to the graph* H*. That is, for a suitable application of reduction rules, there exists the reduction strategy to the normal form of the automaton graph.*

(*Proof*) At first, we note that we can not apply any reduction rules (R), (E) and (C) for a graph expression F. For another graph expression $x[a_1 : s_1, a_2 : s_2, \cdots, a_n : s_n]$, if x has an argument x itself, then first we apply the reduction rule (C) for x and next we

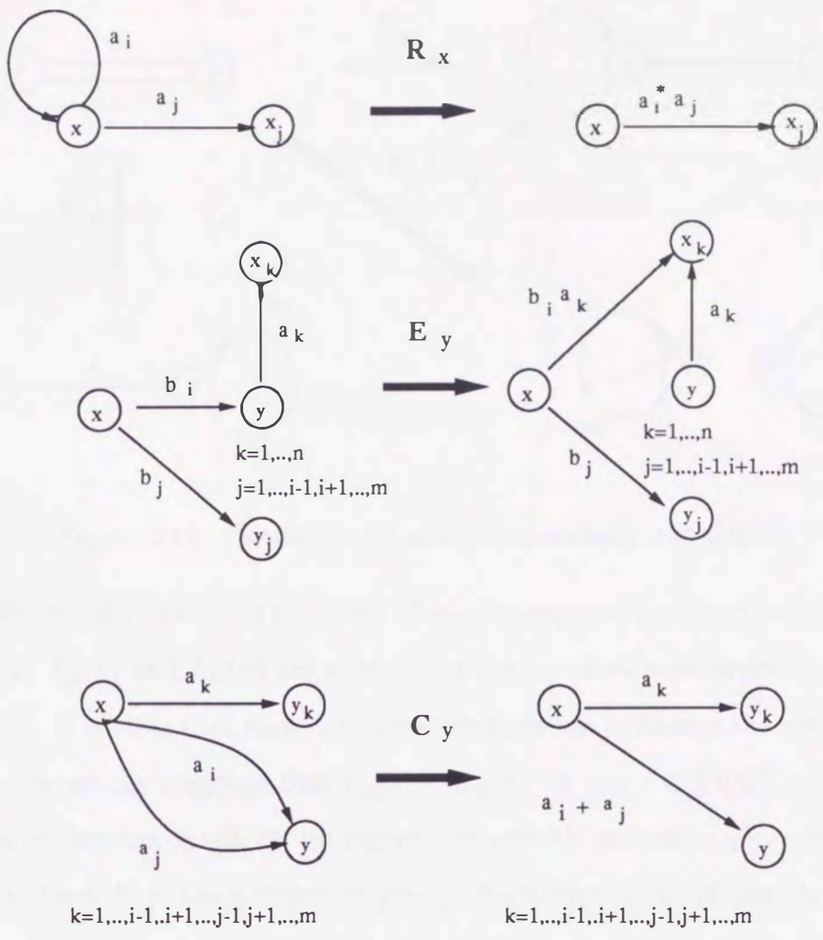


Figure 5.10: *Automaton graph reductions*

apply the reduction rule (R) for x. For any graph expression which has an argument x, we apply the reduction rule (C) for x and next apply the reduction rule (E) for x. Since x itself does not have an argument x, the result of reduction does not have an argument x. Then we can eliminate the all incoming edges to x. So we can not apply any rules for x. Since G has finite element, by applying these processes to all graph expressions in G, we can get the normal form of the automaton graph. That is, we can not apply any reduction rules for any elements of G. ■

Proposition 5.5.5 *For any reduction rules T, if $G \rightarrow_T H$, then $L_G(x) = L_H(x)$ (for any $x \in Rt(G)$).*

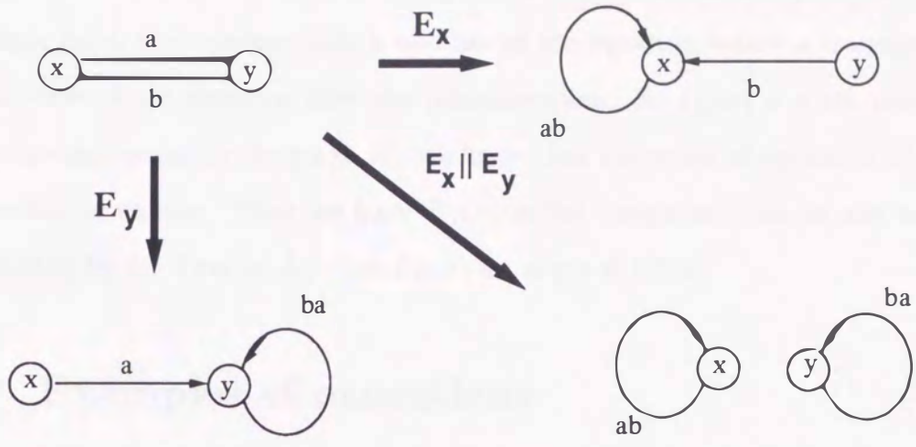


Figure 5.11: *Parallel is not always sequentially simulatable*

(*Proof*) We consider the set of equations of regular expressions, discussed after Proposition 5.5.2. $L_G(x)$ and $L_H(x)$ are solutions of the equation constructed by G and H, respectively. It is clear that each reduction rule does not exchange the solution of the equations. So we can conclude that $L_G(x) = L_H(x)$ for any $x \in Rt(G)$. ■

For an automaton graph G, we choose the suitable reduction procedure and get the normal form H of the automaton graph. Each element of H has the form x or $x[w : F]$. So we know that the set $L_H(x)$ easily ϕ or w respectively. That is, the procedure of reducing an automaton graph to the normal form is corresponds to the process of solving the equation of regular expressions constructed by the automaton graph.

For any reduction $G \rightarrow_R H$ and $G \rightarrow_C H$, there hold $Wt(G \rightarrow_R H) = Rd(G \rightarrow_R H)$ and $Wt(G \rightarrow_C H) = Rd(G \rightarrow_C H)$. Using the Proposition 5.3.5, we have a parallel reduction of (R) and (C) is a sequentially simulatable.

It is not always true that $Wt(G \rightarrow_E H) = Rd(G \rightarrow_E H)$. The next figure shows an example of a parallel reduction which is not sequentially simulatable.

Theorem 5.5.6 *For any parallel reduction $G \rightarrow_{\{t_1,t_2,\cdots,t_n\}} H$, it holds $L_G(x) = L_H(x)$ (for any $x \in Rt(G)$).*

(*Proof*) Reduction rules are corresponding to the transformation of equations. For any reduction rules, it is obvious that a solution of the equation before a transformation is a solution of the equation after the transformation. So $L_G(x)$ is a solution of the equation constructed by the graph H. We know that a solution of equations of regular expressions is unique. Then we have $L_G(x)$ is the unique solution of the equation constructed by H. That is, $L_G(x) = L_H(x)$ for any $x \in Id(G)$. ■

5.6 Examples of executions

Followings are examples of executions of graph reduction system solving equations of regular expressions discussed in Section 5.5.

The first example is solving a equation,

$$\begin{cases} X_1 &= bX_1 + aX_2 \\ X_2 &= bX_1 + aX_2 + \varepsilon. \end{cases}$$

We input equations directly. The system automatically translate an equation to the corresponding graph. For a practical reason, a displayed graph expression have a different form from an expression using in this paper. For example, a graph expression $1[b : 1, a : 2]$ is displayed as $(1.X1) = (b : (1.X1), a : (2.X2))$. By typing a reduction rule and an identifier, the system checks matching procedure for the reduction rule. And if it succeeds, the system reduce the graph applying the reduction rule. Applicable nodes for each reduction rules are always displayed before waiting for a command input.

```
$ (rules − test)
Graph Expression Test Program (rules).
Input Equations.  (Q = End)
X1 = b X1 + a X2 ;
X2 = b X1 + a X2 + e ;
Q
(2 .  X2) = (b:(1 .  X1), a:(2 .  X2), e:(0 .  F))
(1 .  X1) = (b:(1 .  X1), a:(2 .  X2))
(0 .  F) = ()
Combinable Nodes.
NIL
```

```
Expandable Nodes.
NIL
Reducible Nodes.
((2 .  X2) (1 .  X1))
Command (C [n]),(E [n]),(R [n]), A or Q) > (R 2)
(2 .  $) = (a*b:(1 .  X1), a*:(0 .  F))
(1 .  X1) = (b:(1 .  X1), a:(2 .  $))
(0 .  F) = ()
Combinable Nodes.
NIL
Expandable Nodes.
((1 .  X1))
Reducible Nodes.
((1 .  X1))
Command (C [n]),(E [n]),(R [n]), A or Q) > (E 1)
(2 .  $) = (a*b:(1 .  X1), a*:(0 .  F))
(1 .  X1) = (b:(1 .  X1), (a*b):(1 .  X1), aa*:(0 .  F))
(0 .  F) = ()
Combinable Nodes.
((1 .  X1))
Expandable Nodes.
NIL
Reducible Nodes.
((1 .  X1))
Command (C [n]),(E [n]),(R [n]), A or Q) > (C 1)
(2 .  $) = (a*b:(1 .  X1), a*:(0 .  F))
(1 .  X1) = (b+a(a*b):(1 .  X1), aa*:(0 .  F))
(0 .  F) = ()
Combinable Nodes.
NIL
Expandable Nodes.
NIL
Reducible Nodes.
((1 .  X1))
Command (C [n]),(E [n]),(R [n]), A or Q) > (R 1)
(2 .  $) = (a*b:(1 .  $), a*:(0 .  F))
(1 .  $) = ((b+a(a*b))*aa*:(0 .  F))
(0 .  F) = ()
Combinable Nodes.
NIL
Expandable Nodes.
((2 .  $))
Reducible Nodes.
NIL
Command (C [n]),(E [n]),(R [n]), A or Q) > (E 2)
(2 .  $) = (a*b((b+a(a*b))*aa*):(0 .  F), a*:(0 .  F))
(1 .  $) = ((b+a(a*b))*aa*:(0 .  F))
```

```
(0 .  F) = ()
Combinable Nodes.
((2 .  $))
Expandable Nodes.
NIL
Reducible Nodes.
NIL
Command (C [n]),(E [n]),(R [n]), A or Q) > (C 2)
(2 .  $) = (a*b((b+a(a*b))*aa*)+a*:(0 .  F))
(1 .  $) = ((b+a(a*b))*aa*:(0 .  F))
(0 .  F) = ()
Combinable Nodes.
NIL
Expandable Nodes.
NIL
Reducible Nodes.
NIL
```

The next example is solving a equation,

$$\begin{cases} X_1 &= aX_2 + bX_3 + \varepsilon \\ X_2 &= aX_3 + bX_1 + \varepsilon \\ X_3 &= (a+b)X_3. \end{cases}$$

By the command 'A', the system automatically apply the reduction rules according to the strategy discussed in Proposition 4.5. And it terminates with its normal form.

```
$ (rules-test)
Graph Expression Test Program (rules).
Input Equations.  (Q = End)
X1 = a X2 + b X3 + e;
X2 = a X3 + b X1 + e;
X3 = (a+b)X3;
Q
(3 .  X3) = (a+b:(3 .  X3))
(2 .  X2) = (a:(3 .  X3), b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (a:(2 .  X2), b:(3 .  X3), e:(0 .  F))
(0 .  F) = ()
Combinable Nodes.
```

```
NIL
Expandable Nodes.
NIL
Reducible Nodes.
((3 .  X3) (2 .  X2) (1 .  X1))
Command (C [n]),(E [n]),(R [n]),A or Q > A
(3 .  X3) = (a+b:(3 .  X3))
(2 .  X2) = (a:(3 .  X3), b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (a:(2 .  X2), b:(3 .  X3), e:(0 .  F))
(0 .  F) = ()
Reduce
(3 .  $) = ()
(2 .  X2) = (a:(3 .  $), b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (a:(2 .  X2), b:(3 .  $), e:(0 .  F))
(0 .  F) = ()
Expand.
(3 .  $) = ()
(2 .  X2) = (b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (a:(2 .  X2), e:(0 .  F))
(0 .  F) = ()
Reduce.
(3 .  $) = ()
(2 .  $) = (b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (a:(2 .  $), e:(0 .  F))
(0 .  F) = ()
Expand.
(3 .  $) = ()
(2 .  $) = (b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (ab:(1 .  X1), a:(0 .  F), e:(0 .  F))
(0 .  F) = ()
Combine.
(3 .  $) = ()
(2 .  $) = (b:(1 .  X1), e:(0 .  F))
(1 .  X1) = (ab:(1 .  X1), a+e:(0 .  F))
(0 .  F) = ()
Reduce.
(3 .  $) = ()
(2 .  $) = (b:(1 .  X1), e:(0 .  F))
(1 .  $) = ((ab)*(a+e):(0 .  F))
(0 .  F) = ()
Expand.
(3 .  $) = ()
(2 .  $) = (b((ab)*(a+e)):(0 .  F), e:(0 .  F))
(1 .  $) = ((ab)*(a+e):(0 .  F))
(0 .  F) = ()
Combine.
(3 .  $) = ()
```

```
(2 .  $) = (b((ab)*(a+e))+e:(0 .  F))
(1 .  $) = ((ab)*(a+e):(0 .  F))
(0 .  F) = ()
NIL
$
```

Chapter 6

Transformations of relational structures

In this chapter, we propose a more general framework for graph transformations which contains the category defined in Chapter 4. In Section 6.1, extended notions of relational graph structures and examples which contain a hyper graph structure which have a label with arity for each hyper edge are showed. We also proved in the general framework that pushouts alway exist using the properties of partial functions. In Section 6.2, the main results of this chapter is proved. We introduced a notion of partial morphisms of relational structures and show an existence theorem of primitive pushouts which enable us to more generally formalize rewritings of relational structures. In Section 6.3, we compare our category of relational structures together with partial morphisms to a category of partial morphisms in the sense of [RR88, Ken90] and show that they are isomorphic choosing a certain admissible class of monomorphisms.

6.1 Relational structures

In this section we introduce a basic notion of relational structures which generalizes relational algebras [Bar70], simple graphs[1] [MK91], labelled graphs[2] [MK91] and hy-

[1] Chapter 4, Section 4.1
[2] Chapter 4, Section 4.4

pergraphs [Ken90], and a few properties on the category of relational structures are studied.

A *frame* $< \omega : S \to R, \lambda : T \to R >$, on which relational structures are defined, consists of three functors $S, T, R : \mathbf{C} \to \mathbf{Set}$ and two natural transformations $\omega : S \to R$, $\lambda : T \to R$.

Definition 6.1.1 A relational structure $< a, \alpha >$ over a frame $< \omega : S \to R, \lambda : T \to R >$ is a pair of an object a of $\mathbf{C}$ and a relation $\alpha : aS \to aT$ such that $\alpha \cdot a\lambda \sqsubseteq a\omega$. A morphism f from a relational structure $< a, \alpha >$ into a relational structure $< b, \beta >$, denoted by $f :< a, \alpha > \to < b, \beta >$, is a morphism $f : a \to b$ in $\mathbf{C}$ satisfying $\alpha \cdot fT \sqsubseteq fS \cdot \beta$.

$$\begin{array}{ccc} aS & \xrightarrow{\alpha} & aT \\ & \searrow \quad \swarrow & \\ & a\omega \quad aR \quad a\lambda & \end{array} \qquad \begin{array}{ccc} aS & \xrightarrow{fS} & bS \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ aT & \xrightarrow[fT]{} & bT \end{array} \quad \square$$

Let $f :< a, \alpha > \to < b, \beta >$ and $g :< b, \beta > \to < c, \gamma >$ be morphisms of relational structures over a frame $< \omega : S \to R, \lambda : T \to R >$. As $\alpha \cdot fT \sqsubseteq fS \cdot \beta$ and $\beta \cdot gT \sqsubseteq gS \cdot \gamma$ we have

$$\alpha \cdot (fg)T = \alpha \cdot fT \cdot gT \sqsubseteq fS \cdot \beta \cdot gT \sqsubseteq fS \cdot gS \cdot \gamma = (fg)S \cdot \gamma,$$

which shows that the composite $fg :< a, \alpha > \to < c, \gamma >$ is a morphism of relational structures. Also it is clear that $\mathrm{id}_a :< a, \alpha > \to < a, \alpha >$, where id_a is the identity morphism of a. Thus relational structures and their morphisms over $< \omega : S \to R, \lambda : T \to R >$ form a category $\mathbf{C} < \omega : S \to R, \lambda : T \to R >$, called the category of relational structures and their morphisms.

When a functor $R : \mathbf{C} \to \mathbf{Set}$ is a constant functor into a singleton set $1 (= \{0\})$, that is, $aR = 1$ for any object a of $\mathbf{C}$, the condition $\alpha \cdot a\lambda \sqsubseteq a\omega$ is void. In this case we write $\mathbf{C} < S, T >$ for $\mathbf{C} < \omega : S \to R, \lambda : T \to R >$.

Now some examples of relational structures are given in the following:

Example 6.1.2 (Relational algebras [Bar70]) Let $T : \mathbf{Set} \to \mathbf{Set}$ be a functor. A relational T-algebra $< A, \alpha >$ is a pair of a set A and a relation $\alpha : A \to A$. A

T-algebra homomorphism f from a T-algebra $< A, \alpha >$ into a T-algebra $< B, \beta >$ is a function $f : A \to B$ such that $\alpha \cdot f \sqsubseteq fT \cdot \beta$.

$$\begin{array}{ccc} AT & \xrightarrow{fT} & BT \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A & \xrightarrow[f]{} & B \end{array}$$

It is trivial that the category of relational T-algebras and T-algebra homomorphisms coincides with the category **Set** $< T, Id >$ of relational structures, where Id denotes the identity functor of **Set**.

Example 6.1.3 (Simple graphs) A (simple) graph $< A, \alpha >$ is a pair of a set A and a relation $\alpha : A \rightharpoonup A$. A graph homomorphism f from a graph $< A, \alpha >$ into a graph $< B, \beta >$ is a function $f : A \to B$ such that $\alpha \cdot f \sqsubseteq f \cdot \beta$.

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A & \xrightarrow[f]{} & B \end{array}$$

It is obvious that the category of graphs and graph homomorphisms is identical with the category **Set** $< Id, Id >$ of relational structures.

Example 6.1.4 (Labelled graphs) Let Σ be a set of labells. A Σ-labelled graph $< A, \alpha >$ is a pair of a set A and a family $\alpha = \{\alpha_\sigma : A \rightharpoonup A \mid \sigma \in \Sigma\}$ of relations. A Σ-labelled graph homomorphism f from a Σ-labelled graph $< A, \alpha >$ into a Σ-labelled graph $< B, \beta >$ is a function $f : A \to B$ such that $\alpha_\sigma \cdot f \sqsubseteq f \cdot \beta_\sigma$ for any $\sigma \in \Sigma$.

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle\alpha_\sigma}\downarrow & & \downarrow{\scriptstyle\beta_\sigma} \\ A & \xrightarrow[f]{} & B \end{array}$$

A Σ-copower functor $\Sigma \cdot (-) :$ **Set** $\to$ **Set** is a functor which assigns to each set A a coproduct(disjoint union) $\Sigma \cdot A = \coprod_{\sigma \in \Sigma} A$ of Σ copies of A and to each function $f : A \to B$ a function $\Sigma \cdot f : \Sigma \cdot A \to \Sigma \cdot B$ mapping σ-th component A into σ-th component B by $f : A \to B$ for any $\sigma \in \Sigma$. Then it is easy to check that the category of Σ-labelled graphs and Σ-labelled graph homomorphisms is equal to the category **Set** $< \Sigma \cdot (-), Id >$ of relational structures.

Example 6.1.5 (Hyper graphs [Cou89, Ken90]) Let Λ be a set of function symbols with an arity function $\mathbf{arity} : \Lambda \to \mathbf{N}$, where $\mathbf{N}$ is the set of all positive integers. For a set A we denote by A^+ the set of all nonempty tuples of members of A and by $\mathit{Asize} : A^+ \to \mathbf{N}$ the size function of tuples. A hyper graph $< A, \alpha >$ over Λ is a pair of a set A and a relation $\alpha : \Lambda \to\!\!\!\!\!\!- \; A^+$ such that $\alpha \cdot \mathit{Asize} \sqsubseteq \mathbf{arity}$.

$$\begin{array}{ccc} \Lambda & \xrightarrow{\alpha} & A^+ \\ & \searrow \quad \swarrow & \\ & \mathbf{arity} \quad \mathbf{N} \quad \mathit{Asize} & \end{array}$$

For a function $f : A \to B$ a function $f^+ : A^+ \to B^+$ is defined by a trivial way [Rao84]. Thus we have a functor $(-)^+ : \mathbf{Set} \to \mathbf{Set}$, called nonempty Kleene functor. A homomorphism f from a hyper graph $< A, \alpha >$ into a hyper graph $< B, \beta >$ is a function $f : A \to B$ such that $\alpha \cdot f^+ \sqsubseteq \beta$.

$$\begin{array}{ccc} \Lambda & \xrightarrow{\mathrm{id}_\Lambda} & \Lambda \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A^+ & \xrightarrow[f^+]{} & B^+ \end{array}$$

A functor $\mathbf{const}_\Lambda : \mathbf{Set} \to \mathbf{Set}$ denotes the constant functor with values Λ and id_Λ. The arity function $\mathbf{arity} : \Lambda \to \mathbf{N}$ can be seen as a natural transformation $\mathbf{arity} : \mathbf{const}_\Lambda \to \mathbf{const}_N$. At last the category of hyper graphs over Λ is identical with the category $\mathbf{Set} < \mathbf{arity} : \mathbf{const}_\Lambda \to \mathbf{const}_N, \mathbf{size} : (-)^+ \to \mathbf{const}_N >$ of relational structures.

Example 6.1.6 Let $< A, \alpha >$ be a relational structure of $\mathbf{Set} < \mathbf{const}_1, (-)^+ >$, where $1 = \{0\}$. When $< A, \alpha >$ satisfies that for each $a \in A$ there exists exactly one $w \in A^*$ such that $(0, aw) \in \alpha$, it can be regarded as a graph in the sense of Raoult [Rao84]. (Where A^* is the set of all finite strings of A including the empty string.) Let $< B, \beta >$ be another such relational structure and $f :< A, \alpha > \to\!\!\!\!\!\!- < B, \beta >$ a morphism of $\mathbf{Set} < \mathbf{const}_1, (-)^+ >$.

$$\begin{array}{ccc} 1 & \xrightarrow{\mathrm{id}_1} & 1 \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ A^+ & \xrightarrow[f^+]{} & B^+ \end{array}$$

It is easy to verify that a condition $\alpha \cdot f^+ \sqsubseteq \beta$ is the same as a condition for graph morphisms in [Rao84]. Thus the category of graphs treated by Raoult is a subcategory of $\mathbf{Set} < \mathbf{const}_1, (-)^+ >$.

In the rest of this section we assume that a fixed frame $< \omega : S \to R, \lambda : T \to R >$ is given and we will write $\mathbf{C} < \omega, \lambda >$ for $< \omega : S \to R, \lambda : T \to R >$. The forgetful functor $V : \mathbf{C} < \omega, \lambda > \to \mathbf{C}$ is defiend by a trivial way.

Theorem 6.1.7 (1) *The forgetful functor* $V : \mathbf{C} < \omega, \lambda > \to \mathbf{C}$ *creates colimits.*

(2) *If* $R : \mathbf{C} \to \mathbf{Set}$ *preserves limits, then* $V : \mathbf{C} < \omega, \lambda > \to \mathbf{C}$ *creates limits.*

Proof. (a) It is sufficient to show that V creates coequalizers and coproducts. Let $f :< x, \alpha_x > \to < y, \alpha_y >$ and $g :< x, \alpha_x > \to < y, \alpha_y >$ be morphisms in $\mathbf{C}$, $e : y \to u$ a coequalizer of f and g in $\mathbf{Set}$ and $\alpha_u = eS^\sharp \cdot \alpha_y \cdot eT$. We show $e :< y, \alpha_y > \to < u, \alpha_u >$ is a coequalizer of f and g in $\mathbf{C}$.

Since $\alpha_u \cdot u\lambda = eS^\sharp \cdot \alpha_y \cdot eT \cdot u\lambda = eS^\sharp \cdot \alpha_y \cdot y\lambda \cdot eR \sqsubseteq eS^\sharp \cdot y\omega \cdot eR = eS^\sharp \cdot eS \cdot u\omega \sqsubseteq u\omega$, we have $< u, \alpha_u >$ is a relational structure. By $\alpha_y \cdot eT \sqsubseteq eS \cdot eS^\sharp \cdot \alpha_y \cdot eT \sqsubseteq eS \cdot \alpha_u$, we obtain $e :< y, \alpha_y > \to < u, \alpha_u >$ is a morphism of $\mathbf{C}$.

Let $p :< y, \alpha_y > \to < v, \alpha_v >$ is a morphism with $fp = gp$. Since e is a coequalizer in $\mathbf{Set}$, there exists a unique morphism $t : u \to v$ such that $et = p$. By $\alpha_u \cdot tT = eS^\sharp \cdot \alpha_y \cdot eT \cdot tT \sqsubseteq eS^\sharp \cdot pS \cdot \alpha_v = eS^\sharp \cdot eS \cdot tS \cdot \alpha_v \sqsubseteq tS \cdot \alpha_v$, we have $t :< u, \alpha_u > \to < v, \alpha_v >$ is a morphsim of $\mathbf{C}$. So $e :< y, \alpha_y > \to < u, \alpha_u >$ is a coequalizer of f and g in $\mathbf{C}$.

Let $x + y$ be a coproduct of x and y in $\mathbf{Set}$ with inclusions $i_x : x \to x + y$ and $i_y : y \to x + y$. It is similarly easy to show that $< x + y, i_x S^\sharp \cdot \alpha_x \cdot i_x T \sqcup i_y S^\sharp \cdot \alpha_y \cdot i_y T >$ is a coproduct of $< x, \alpha_x >$ and $< y, \alpha_y >$ in $\mathbf{C}$ with inclusions i_x and i_y.

Then V creates coequalizers and coproducts so it creates colimits.

(b) Let $x \times y$ be a product of x and y in $\mathbf{Set}$ and $p_x : x \times y \to x$, $p_y : x \times y \to y$ projections. Let $\alpha_{x \times y} = (p_x S \cdot \alpha_x \cdot p_x T^\sharp) \sqcap (p_y S \cdot \alpha_y \cdot p_y T^\sharp)$. It is trivial that $\alpha_{x \times y} \cdot p_x T \sqsubseteq p_x S \cdot \alpha_x$ and $\alpha_{x \times y} \cdot p_y T \sqsubseteq p_y S \cdot \alpha_y$.

We note that if R preserves limits then $(p_x R \cdot p_x R^\sharp) \sqcap (p_y R \cdot p_y R^\sharp) = \mathrm{id}_{(x \times y)R}$. So

we have

$$\begin{aligned}
\alpha_{x\times y}\cdot(x\times y)\lambda &\sqsubseteq (p_xS\cdot\alpha_x\cdot p_xT^\sharp\cdot(x\times y)\lambda)\sqcap(p_yS\cdot\alpha_y\cdot p_yT^\sharp\cdot(x\times y)\lambda)\\
&\sqsubseteq (p_xS\cdot\alpha_x\cdot x\lambda\cdot p_xR^\sharp)\sqcap(p_yS\cdot\alpha_y\cdot y\lambda\cdot p_yR^\sharp)\\
&\sqsubseteq (p_xS\cdot x\omega\cdot p_xR^\sharp)\sqcap(p_yS\cdot y\omega\cdot p_yR^\sharp)\\
&= ((x\times y)\omega\cdot p_xR\cdot p_xR^\sharp)\sqcap((x\times y)\omega\cdot p_yR\cdot p_yR^\sharp)\\
&= (x\times y)\omega((p_xR\cdot p_xR^\sharp)\sqcap(p_yR\cdot p_yR^\sharp))\\
&= (x\times y)\omega.
\end{aligned}$$

These show that $< x\times y, \alpha_{x\times y} >$ is a relational structure and p_x and p_y are morphisms of $\mathbf{C}$.

Let $f_x :< u,\alpha_u > \rightarrow < x,\alpha_x >$ and $f_y :< u,\alpha_u > \rightarrow < y,\alpha_y >$ be morphsims of $\mathbf{C}$. Since $x\times y$ is a product in **Set** there exists morphisms $t : u \rightarrow x\times y$ such that $tp_x = f_x$ and $tp_y = f_y$. We have $\alpha_u\cdot tT \sqsubseteq \alpha_u\cdot tT\cdot p_xT\cdot p_xT^\sharp = \alpha_u\cdot f_xT\cdot p_xT^\sharp \sqsubseteq fS\cdot\alpha_x\cdot p_xT^\sharp$ $= tS\cdot p_xS\cdot\alpha_x\cdot p_xT^\sharp$ and similarly $\alpha_u\cdot tT \sqsubseteq tS\cdot p_yS\cdot\alpha_y\cdot p_yT^\sharp$. So we obtain $\alpha_u\cdot tT$ $\sqsubseteq tS\cdot((p_xS\cdot\alpha_x\cdot p_xT^\sharp)\sqcap(p_yS\cdot\alpha_y\cdot p_yT^\sharp)) = tS\cdot\alpha_{x\times y}$. These indicates that V creates products.

It is similarly to show that V creates equalizers. So V creates limits. ■

The following is a standard corollary of the last theorem.

Corollary 6.1.8 (1) *If* $\mathbf{C}$ *has colimits, then* $\mathbf{C} < \omega,\lambda >$ *has colimits.*

(2) *If* $\mathbf{C}$ *has limits and* $R : \mathbf{C} \rightarrow$ **Set** *preserves limits, then* $\mathbf{C} < \omega,\lambda >$ *has limits.* □

Notice that every functor $T : \mathbf{C} \rightarrow$ **Set** naturally induces a natural transformation $T\lambda : T^+ \rightarrow \mathbf{const}_N$.

6.2 Partial morphisms of relational structures

In this section we introduce a notion of partial morphisms of relational structures to discuss single-pushout rewritings of relational structures and show an exsistence

thoerem (Cf. Theorem 6.2.4) of primitive pushouts, which enables us to more generally formalize rewritings of relational structures.

First we recall a notion of partial morphisms in a general cateogry. A class M of monomorphisms of a category $\mathbf{C}$ is called *admissible* [RR88, Ken87] if it contains all isomorphisms and is closed under composition and inverse images. (Exactly, M is closed under inverse images if $m \in M$ implies $n \in M$ whenever a square

$$\begin{array}{ccc} e & \xrightarrow{n} & b \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle k} \\ d & \xrightarrow[m]{} & a \end{array}$$

is a pullback in $\mathbf{C}$.) Let $\mathbf{C}$ be a category with inverse images of an admissible class M of monomorphisms of $\mathbf{C}$. A partial morphism $f : a \to b$ in $\mathbf{C}$ is a pair $(m : d \to a, f' : d \to b)$ of a monomorphism m in M and a morphism f' in $\mathbf{C}$. (Formally, it is an equivalence class of such pairs: $f_0 = (m_0 : d_0 \to a, f'_0 : d_0 \to b)$ is equivalent to $f_1 = (m_1 : d_1 \to a, f'_1 : d_1 \to b)$ if and only if there exists an isomorphism $i : d_0 \to d_1$ making the diagram

$$\begin{array}{ccccc} & & d_0 & & \\ & {\scriptstyle m_0}\swarrow & & \searrow{\scriptstyle f'_0} & \\ a & & \downarrow{\scriptstyle i} & & b \\ & {\scriptstyle m_1}\searrow & & \nearrow{\scriptstyle f'_1} & \\ & & d_1 & & \end{array}$$

commute.)

The composite of a partial morphism $f = (m : d \to a, f' : d \to b) : a \to b$ followed by a partial morphism $g = (n : e \to b, g' : e \to c) : b \to c$ is defined by $fg = (n'm, f''g')$, where a square (2) below is a pullback (an inverse image of $n \in M$).

$$\begin{array}{ccccc} p & \xrightarrow{n'} & d & \xrightarrow{m} & a \\ {\scriptstyle f''}\downarrow & (2) & \downarrow{\scriptstyle f'} & & \\ e & \xrightarrow[n]{} & b & & \\ {\scriptstyle g'}\downarrow & & & & \\ c & & & & \end{array}$$

Thus we have the category $Pfn(\mathbf{C}, M)$ of partial morphsims with respect to M.

A functor $S : \mathbf{C} \to \mathbf{Set}$ is *admissible* with respect to M if it transforms monomorphisms of M into injective functions and preserves inverse images of monomorphisms of M. Notice that every functor $S : \mathbf{C} \to \mathbf{Set}$ preverving pullbacks is admissible with respect to any admissible class of monomorphisms. Both products and copruducts of admissible functors are also admissible. Trivially the identity functor $Id : \mathbf{Set} \to \mathbf{Set}$, a copower functor $\Sigma \cdot (-) : \mathbf{Set} \to \mathbf{Set}$, (nonempty) Kleene functor $(-)^+ : \mathbf{Set} \to \mathbf{Set}$, and a constant functor $\mathbf{const}_A : \mathbf{Set} \to \mathbf{Set}$ are admissible (with respect to the class $Mon(\mathbf{Set})$ of all monomorphisms of $\mathbf{Set}$).

Lemma 6.2.1 *Let M be an admissible class of monomorphisms of a category $\mathbf{C}$.*

(1) *An admissible functor $S : \mathbf{C} \to \mathbf{Set}$ with respect to M can be extended into a functor $S : Pfn(\mathbf{C}, M) \to \mathbf{Pfn}$.*

(2) *If $S, T : \mathbf{C} \to \mathbf{Set}$ are admissible functors with respect to M and $\omega : S \to T$ is a natural transformation, then $fS \cdot b\omega \sqsubseteq a\omega \cdot fT$ holds for every partial morphism $f : a \to b$ in $Pfn(\mathbf{C}, M)$.*

Proof. (a) For a partial morphism $f = (m : d \to a, f' : d \to b)$ with $m \in M$ define a partial function $fS : aS \to bS$ by $fS = mS^\sharp \cdot f'S$. We will show that this definiton gives an extended functor $S : Pfn(\mathbf{C}, M) \to \mathbf{Pfn}$. To this end it suffices to prove that $(fg)S = fS \cdot gS$ for all partial morphisms $f : a \to b$ and $g : b \to c$. Assume that $f = (m : d \to a, f' : d \to b)$ and $g = (n : e \to b, g' : e \to c)$, where $m, n \in M$, and construct a pullback

$$\begin{array}{ccc} p & \xrightarrow{n'} & d \\ {\scriptstyle f''}\downarrow & & \downarrow{\scriptstyle f'} \\ e & \xrightarrow[n]{} & b \end{array}$$

in $\mathbf{C}$. Then we have $f'S \cdot nS^\sharp = n'S^\sharp \cdot f''S$, since S preserves inverse images of monomorphisms in M, and so

$$\begin{aligned} fS \cdot gS &= mS^\sharp \cdot f'S \cdot nS^\sharp \cdot g'S \\ &= mS^\sharp \cdot n'S^\sharp \cdot f''S \cdot g'S \end{aligned}$$

$$= (m'n)S^\sharp \cdot (f''g')S$$
$$= (fg)S.$$

(b) Assume that $f = (m : d \to a, f' : d \to b)$ with $m \in M$. From the naturality of $\omega : S \to T$ it follows that $f'S \cdot b\omega = d\omega \cdot f'T$ and $d\omega = mS \cdot a\omega \cdot mT^\sharp$ (by $mT \cdot mT^\sharp = \mathrm{id}_{dT}$). Hence

$$\begin{aligned} fS \cdot b\omega &= mS^\sharp \cdot f'S \cdot b\omega \\ &= mS^\sharp \cdot d\omega \cdot f'T \\ &= mS^\sharp \cdot mS \cdot a\omega \cdot mT^\sharp \cdot f'T \\ &\sqsubseteq a\omega \cdot mT^\sharp \cdot f'T \\ &= a\omega \cdot fT. \quad \square \end{aligned}$$

In what follows we assume that there is given a frame $< \omega : S \to R, \lambda : T \to R >$ in which functors $S, T, R : \mathbf{C} \to \mathbf{Set}$ are admissible with respect to an admissible class M of monomorphisms of $\mathbf{C}$.

Definition 6.2.2 Let $< a, \alpha >$ and $< b, \beta >$ be relational structures over $< \omega : S \to R, \lambda : T \to R >$. A partial morphism $f :< a, \alpha > \to < b, \beta >$ of relational structures is a partial morphism $f : a \to b$ of $Pfn(\mathbf{C})$ such that $d(fS) \cdot \alpha \cdot fT \sqsubseteq fS \cdot \beta$.

Assume that $f :< a, \alpha > \to < b, \beta >$ and $g :< b, \beta > \to < c, \gamma >$ are partial morphisms of relational structures over $< \omega : S \to R, \lambda : T \to R >$. As $d(fS) \cdot \alpha \cdot fT \sqsubseteq fS \cdot \beta$ and $d(gS) \cdot \beta \cdot gT \sqsubseteq gS \cdot \gamma$, we have

$$\begin{aligned} d((fg)S) \cdot \alpha \cdot (fg)T &= d(fS \cdot gS)d(fS) \cdot \alpha \cdot fT \cdot gT \\ &\sqsubseteq d(fS \cdot gS) \cdot fS \cdot \beta \cdot gT \\ &= fS \cdot d(gS) \cdot \beta \cdot gT \\ &\sqsubseteq fS \cdot gS \cdot \gamma \\ &= (fg)S \cdot \gamma. \end{aligned}$$

Hence the composite of partial morphisms is also a partial morphism and so relational structures and partial morphisms over $< \omega : S \to R, \lambda : T \to R >$ form a category

$Pfn(\mathbf{C}) < \omega : S \to R, \lambda : T \to R >$, called the category of relational structures and partial morphisms. When $\mathbf{C} = \mathbf{Set}$ we write $Pfn < \omega : S \to R, \lambda : T \to R >$ for $Pfn(\mathbf{Set}) < \omega : S \to R, \lambda : T \to R >$.

Lemma 6.2.3 *Let $x : B \to X$ and $y : C \to X$ be partial functions between sets and a square*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ C & \xrightarrow[k]{} & D \end{array}$$

a pushout in **Pfn**. *If a (total) function $z : D \to X$ satisfies $hz \sqsubseteq x$ and $kz \sqsubseteq y$, then $z = h^\sharp x \sqcup k^\sharp y$.*

Proof. First we show that $z' = h^\sharp x \sqcup k^\sharp y$ is a partial function, that is, $z'^\sharp z' \sqsubseteq \mathrm{id}_X$. But $hh^\sharp \sqsubseteq hzz^\sharp h^\sharp \sqsubseteq xx^\sharp$ because of the totality $\mathrm{id}_D \sqsubseteq zz^\sharp$ of z, and similarly $kk^\sharp \sqsubseteq yy^\sharp$ and $hk^\sharp \sqsubseteq xy^\sharp$. Hence we have

$$\begin{aligned} z'^\sharp z' &= x^\sharp hh^\sharp x \sqcup x^\sharp hk^\sharp y \sqcup y^\sharp kh^\sharp x \sqcup y^\sharp kk^\sharp y \\ &\sqsubseteq x^\sharp xx^\sharp x \sqcup x^\sharp xy^\sharp y \sqcup y^\sharp yx^\sharp x \sqcup y^\sharp yy^\sharp y \\ &\sqsubseteq \mathrm{id}_X \quad (\text{ by } x^\sharp x \sqsubseteq \mathrm{id}_X \text{ and } y^\sharp y \sqsubseteq \mathrm{id}_X \text{ }) \end{aligned}$$

On the other hand $h^\sharp h \sqcup k^\sharp k = \mathrm{id}_D$ since the above square is a pushout in **Pfn**, and so $z = h^\sharp hz \sqcup k^\sharp kz \sqsubseteq h^\sharp x \sqcup k^\sharp y = z'$. Therefore $z' \sqsubseteq zz^\sharp z' \sqsubseteq zz'^\sharp z' \sqsubseteq z$, which proves $z' = z$. □

Theorem 6.2.4 *Assume that the extended functor $S : Pfn(\mathbf{C}, M) \to \mathbf{Pfn}$ preserves pushouts. If $< b, \beta >$ and $< c, \gamma >$ are relational structures over $< \omega : S \to R, \lambda : T \to R >$ and if a square*

$$\begin{array}{ccc} a & \xrightarrow{f} & b \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle h} \\ c & \xrightarrow[k]{} & d \end{array}$$

is a pushout in $Pfn(\mathbf{C}, M)$, then $h :< b, \beta > \to < d, \delta >$ and $k :< c, \gamma > \to < d, \delta >$ are partial morphisms of relational structures, where $\delta = hS^\sharp \cdot \beta \cdot hT \sqcup kS^\sharp \cdot \gamma \cdot kT$.

Moreover, if $h' :< b, \beta >\rightarrow< d', \delta' >$ *and* $k' :< c, \gamma >\rightarrow< d', \delta' >$ *are partial morphisms of relational structures satisfying* $fh' = gk'$*, then there exists a unique partial morphism* $t :< d, \delta >\rightarrow< d', \delta' >$ *of relational structures such that* $h' = ht$ *and* $k' = kt$.

Proof. First we must see that $< d, \delta >$ is a relational structure, that is, $\delta \cdot d\lambda \sqsubseteq d\omega$. A square

$$\begin{array}{ccc} aS & \xrightarrow{fS} & bS \\ {\scriptstyle gS}\downarrow & & \downarrow{\scriptstyle hS} \\ cS & \xrightarrow[kS]{} & dS \end{array}$$

is a pushout in **Pfn** by the assumption and $hS \cdot d\omega \sqsubseteq b\omega \cdot hR$ and $kS \cdot d\omega \sqsubseteq c\omega \cdot kR$ by Lemma 6.2.1(b). Remark that $d\omega$ is a (total) function. Hence $d\omega = hS^\sharp \cdot b\omega \cdot hR \sqcup kS^\sharp \cdot c\omega \cdot kR$ by the last lemma and

$$\begin{aligned} \delta \cdot d\lambda &= hS^\sharp \cdot \beta \cdot hT \cdot d\lambda \sqcup kS^\sharp \cdot \gamma \cdot kT \cdot d\lambda \\ &\sqsubseteq hS^\sharp \cdot \beta \cdot b\lambda \cdot hR \sqcup kS^\sharp \cdot \gamma \cdot c\lambda \cdot kR \\ &\sqsubseteq hS^\sharp \cdot b\omega \cdot hR \sqcup kS^\sharp \cdot c\omega \cdot kR \\ &= d\omega. \end{aligned}$$

Next we see that $h :< b, \beta >\rightarrow< d, \delta >$ and $k :< b, \beta >\rightarrow< d, \delta >$ are partial morphisms of relational structures. It simply follows from

$$\begin{aligned} d(hS) \cdot \beta \cdot hT &\sqsubseteq hS \cdot hS^\sharp \cdot \beta \cdot hT \quad (\text{by } d(hS) = hS \cdot hS^\sharp \sqcap \mathrm{id}_{bS}) \\ &\sqsubseteq hS \cdot \delta \qquad\qquad (\text{by } \delta = hS^\sharp \cdot \beta \cdot hT \sqcup kS^\sharp \cdot \gamma \cdot kT). \end{aligned}$$

Finally assume that $h' :< b, \beta >\rightarrow< d', \delta' >$ and $k' :< c, \gamma >\rightarrow< d', \delta' >$ are partial morphisms ofs satisfying $fh' = gk'$. Then we have $d(h'S) \cdot \beta \cdot h'T \sqsubseteq h'S \cdot \delta'$ and $d(k'S) \cdot \gamma \cdot k'T \sqsubseteq k'S \cdot \delta'$. As (1) is a pushout in $\mathbf{Pfn}(\mathbf{C}, M)$, there exists a unique partial function $t : d \rightarrow d'$ such that $h' = ht$ and $k' = kt$. It suffices to prove that

$d(tS) \cdot \delta \cdot tT \sqsubseteq tS \cdot \delta'$. But it follows from

$$\begin{aligned}
d(tS) \cdot \delta \cdot tT &\sqsubseteq tS \cdot tS^\sharp \cdot (hS^\sharp \cdot \beta \cdot hT \sqcup kS^\sharp \cdot \gamma \cdot kT) \cdot tT \\
&= tS \cdot (tS^\sharp \cdot hS^\sharp \cdot \beta \cdot hT \cdot tT \sqcup tS^\sharp \cdot kS^\sharp \cdot \gamma \cdot kT \cdot tT) \\
&= tS \cdot (h'S^\sharp \cdot \beta \cdot h'T \sqcup k'S^\sharp \cdot \gamma \cdot k'T) \\
&= tS \cdot (h'S^\sharp \cdot d(h'S) \cdot \beta \cdot h'T \sqcup k'S^\sharp \cdot d(k'S) \cdot \gamma \cdot k'T) \\
&\sqsubseteq tS(h'S^\sharp \cdot h'S \cdot \delta' \sqcup k'S^\sharp \cdot k'S \cdot \delta') \\
&\sqsubseteq tS \cdot (\delta' \sqcup \delta') \\
&= tS \cdot \delta'.
\end{aligned}$$

This completes the proof. □

Note that a relational structure $< d, \delta >$ in the above theorem is unique up to isomorphisms. The following is exactly a corollary of the last theorem.

Corollary 6.2.5 *If the extended functor $S : Pfn(\mathbf{C}, M) \to \mathbf{Pfn}$ preserves pushouts, then the category $Pfn(\mathbf{C}) < \omega : S \to R, \lambda : T \to R >$ of relational structures and partial morphisms has pushouts.* □

The identity functor $Id : \mathbf{Pfn} \to \mathbf{Pfn}$, a copower functor $\Sigma \cdot (-) : \mathbf{Pfn} \to \mathbf{Pfn}$, and a constant functor $\mathbf{const}_\Lambda : \mathbf{Pfn} \to \mathbf{Pfn}$ preserve pushouts. Hence we obtain the following

Corollary 6.2.6 *The categories $Pfn < Id, Id >$, $Pfn < \Sigma \cdot (-), Id >$ and $Pfn <$ $\mathbf{arity} : \mathbf{const}_\Lambda \to \mathbf{const}_N, \mathbf{size} : (-)^+ \to \mathbf{const}_N >$ have pushouts.* □

6.3 A comparison theorem

This section is devoted to state that the category $Pfn(\mathbf{C}) < \omega, \lambda >$ of relational structures together with partial morphisms is isomorphic to a category of partial morphisms of $\mathbf{C} < \omega, \lambda >$ with respect to a certain admissible class of nomomorphsims of $\mathbf{C} < \omega, \lambda >$ in the sense of [Ken87, RR88].

Proposition 6.3.1 *A morphism* $m :< d, \delta > \rightarrow < a, \alpha >$ *in* $\mathbf{C} < \omega, \lambda >$ *is a monomorphism if and only if* $m : d \rightarrow a$ *is a monomorphism in* $\mathbf{C}$.

Proof. It is immediate that $m :< d, \delta > \rightarrow < a, \alpha >$ is a monomorphism if $m : d \rightarrow a$ is a monomorphism. We have to show its converse. Assume that $m :< d, \delta > \rightarrow < a, \alpha >$ is a monomorphism and two morphisms $x, y : c \rightarrow d$ in $\mathbf{C}$ satisfies $xm = ym$. Define $\gamma = xS \cdot \delta \cdot xT^\sharp \sqcap yS \cdot \delta \cdot yT^\sharp$. Then $x, y :< c, \gamma > \rightarrow < d, \delta >$ are morphisms of relational graph structures and so $x = y$ since $m :< d, \delta > \rightarrow < a, \alpha >$ is a monomorphism. □

A monomorphism $m :< d, \delta > \rightarrow < a, \alpha >$ in $\mathbf{C} < \omega, \lambda >$ is called full if $\delta = mS \cdot \alpha \cdot mT^\sharp$.

Lemma 6.3.2 *The class* F *of all full monomorphisms in* $\mathbf{C} < \omega, \lambda >$ *is admissible.*

Proof. Trivially F contains all isomorphisms im $\mathbf{C} < \omega, \lambda >$ and is closed under compsition. We show that F is closed under inverse images. Assume that a square

$$\begin{array}{ccc} < e, \varepsilon > & \xrightarrow{n} & < b, \beta > \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle h} \\ < d, \delta > & \xrightarrow[m]{} & < a, \alpha > \end{array}$$

is a pullback in $\mathbf{C} < \omega, \lambda >$ and $m :< d, \delta > \rightarrow < a, \alpha >$ is a full monomorphism. First note that $\delta = mS \cdot \alpha \cdot mT^\sharp$ and $\beta \sqsubseteq \beta \cdot hT \cdot hT^\sharp \sqsubseteq hS \cdot \alpha \cdot hT^\sharp$. Hence

$$\begin{aligned} kS \cdot \delta \cdot kT^\sharp &= kS \cdot mS \cdot \alpha \cdot mT^\sharp \cdot kT^\sharp \\ &= nS \cdot hS \cdot \alpha \cdot hT^\sharp \cdot nT^\sharp \\ &\sqsupseteq nS \cdot \beta \cdot nT^\sharp. \end{aligned}$$

On the other hand $\varepsilon = nS \cdot \beta \cdot nT^\sharp \sqcap kS \cdot \delta \cdot kT^\sharp$ by the pullback construction in Theorem 2.6 and so $\varepsilon = nS \cdot \beta \cdot nT^\sharp$. This complets the proof. □

Theorem 6.3.3 *The category* $Pfn(\mathbf{C}) < \omega, \lambda >$ *of relational structures together with partial morphisms between them is isomorphic to the category of partial morphisms*

$Pfn(\mathbf{C} < \omega, \lambda >, F)$ of $\mathbf{C} < \omega, \lambda >$ with respect to the class F of all full monomorphisms, that is,

$$Pfn(\mathbf{C}) < \omega, \lambda > \cong Pfn(\mathbf{C} < \omega, \lambda >, F).$$

Proof. Fisrt we construct a functor $\iota : Pfn(\mathbf{C}) < \omega, \lambda > \to Pfn(\mathbf{C} < \omega, \lambda >, F)$. Let $f :< a, \alpha > \to < b, \beta >$ be a partial morphism in $Pfn(\mathbf{C}) < \omega, \lambda >$. By the definition a partial morphism $f : a \to b$ in $Pfn(\mathbf{C})$ is represented as $f = (m : d \to a, f' : d \to b)$, where m is a monomorphism in $\mathbf{C}$ and f' is a morphism in $\mathbf{C}$. Define a relation $\delta : dS \to dT$ by $\delta = mS \cdot \alpha \cdot mT^\sharp$. Then $m :< d, \delta > \to < a, \alpha >$ is clearly a full monomorphism in $\mathbf{C} < \omega, \lambda >$. Also, since $d(fS) = mS^\sharp \cdot mS$, $mS \cdot mS^\sharp = \mathrm{id}_{dS}$ and $d(fS) \cdot \alpha \cdot fT \sqsubseteq fS \cdot \beta$, we obtain $\delta \cdot f'T \sqsubseteq f'S \cdot \beta$. Thus a functor $\iota : Pfn(\mathbf{C}) < \omega, \lambda > \to Pfn(\mathbf{C} < \omega, \lambda >, F)$ is defined by $\iota(m : d \to a, f' : d \to b) = (m :< d, \delta > \to < a, \alpha >, f' :< d, \delta > \to < b, \beta >)$. Obviously ι is an embedding. To prove that ι is an isomorphism of categories it suffices to see that if $f = (m :< d, \delta > \to < a, \alpha >, f' :< d, \delta > \to < b, \beta >)$ is a partial morphism in $Pfn(\mathbf{C} < \omega, \lambda >, F)$, then $f = (m : d \to a, f' : d \to a)$ is a partial morphism in $Pfn(\mathbf{C} < \omega, \lambda >, F)$. Now assume that $f = (m :< d, \delta > \to < a, \alpha >, f' :< d, \delta > \to < b, \beta >)$ is a partial morphism in $Pfn(\mathbf{C}) < \omega, \lambda >$. Finally we have

$$d(fS) \cdot \alpha \cdot fT = mS^\sharp \cdot mS \cdot \alpha \cdot mT^\sharp \cdot f'T = mS^\sharp \cdot \delta \cdot f'T \sqsubseteq mS^\sharp \cdot f'S \cdot \beta = fS \cdot \beta,$$

which completes the proof. □

Chapter 7

Conclusion

In this thesis, we proposed a general categorical framework for several graph structures and analyzed properties about graph transformations using relational calculus. Several graph structures such as graphs with ordered edges [Rao84], simple graphs, graphs with weighted edges and labeled graphs are abstracted to coalgebra-like objects. The result clarifies the relationship between these graph structures and gives an essential properties for pushout constructions. We obtained that when we choose a suitable functor which construct a graph structure we can formulate a graph transformation in which we are able to apply any rewriting rules without any restrictions such as gluing conditions.

We extended the theory of (binary) relations for the foundation of the theory of graph transformations. Especially, we studied the theory of partial functions and gave a general pushout constructions which is applicable to topos a model of constructive logic. Using relational calculus, simple calculations gave many valuable properties about graph transformations. Known results which have complicated proofs were reexamined by simple calculations and further we provided some insights and answers for advanced problems such as commutativity of two transformations and critical pair lemma.

For a particular example of calculations using graph transformations, we demonstrated the problem finding out the language accepted by a finite automaton which corresponds to solve equations of regular expressions. In the formulation of graph rewriting system using graph terms, we proved not only termination of rewritings but

also any permitted parallel rewritings lead the correct answer. The result gives us a new insight into parallel executions. We are inclined to consider only sequentially simulatable parallel executions, because it is difficult to consider the behavior of a non-sequential simulatable parallel executions. But precise theory of graph transformations gives a meaningful semantics of parallel executions. The example is a first trial for this parallel execution approach.

We are going to continue researching about relational structures and to produce a general graph rewriting system based on our theory of graph transformations as a kind of functional programming languages like DACTL[GKS87, GHK+88]. In the applications of the language, we will face another problems of our model. We will formulate these problems in our framework and solve it by choosing suitable graph structures and restricted conditions. A known problem is a treating the operations about node duplications which is difficult to characterize in pushout approaches. For this problem, Löwe introduced pullout a combination of pushouts and pullbacks approach but his method is so much special that it does not harmonize with pushout approach in his framework. We would like to try reducing these kinds of problems into a single pushout approach choosing a suitable relational structures.

Bibliography

[AM75] Michael A. Arbib and Ernest G. Manes. *Arrows, Structures, and Functors.* Academic Press, 1975.

[Bar70] M. Barr. Relational algebras. In *Report of the Mid West Category Seminar IV*, volume 137 of *Lecture Notes in Mathematics*, pages 39–55, Berlin, 1970. Springer.

[BHK90] D. Bolton, C. Hankin, and P. Kelly. Parallel object-oriented descriptions of graph reduction machines. *Future Generations Computer Systems*, 6:225–239, 1990.

[Bro90] P.M. van den Broek. Algebraic graph rewriting using a single pushout. In *TAPSOFT'91 Proceedings*, volume 493 of *Lecture Notes in Computer Science*, pages 90–102. Springer-Verlag, 1990.

[Cal77] M.S. Calenko. The structures of correspondence categories. *Soviet Math. Dokl.*, 18:1498–1502, 1977.

[CCM85] G. Cousineau, P-L. Curien, and M. Mauny. The categorical abstract machine. In *Proceedings of Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*, pages 50–64, 1985.

[CER79] V. Claus, H. Ehrig, and G. Rozenberg, editors. *Graph-grammars and their application to computer science*, volume 73 of *Lecture Notes in Computer Science*. Springer-Verlag, 1979.

[CGR84] M.S. Calenko, V.B. Gisin, and D.A. Raikov. Ordered categories with involution. *Dissertations Mathematics*, 227:111pp., 1984.

[Che86] P. Chenadec. *Canonical forms in finitely presented algebras.* Pitman, London, 1986.

[CM91] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. In *Proc. 17th Intern. workshop on graph-theoretic concepts in computer science*, volume 570 of *Lecture Notes in Computer Science*, pages 13–24, 1991.

[Cou86] B. Courcelle. A representation of graphs by algebraic expressions and its use for graph rewriting systems. In *Proc. 4th Intern. Workshop on graph grammars and their application to computer science*, volume 291 of *Lecture Notes in Computer Science*, pages 112–131, 1986.

[Cou89] B. Courcelle. The monadic second-order logic of graphs, II: Infinite graphs of bounded width. *Math. Systems Theory*, 21:187–221, 1989.

[Cou90] B. Courcelle. *Graph Rewritings: An algebraic and logic approach*, volume B of *Handbook of Theoretical Computer Science*, chapter 5. The MIT Press, 1990.

[Der82] D. Dershowitz. Ordering for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[DJ90] D. Dershowitz and J.-P. Jouannaud. *Rewrite System*, volume B of *Handbook of Theoretical Computer Science*, chapter 6. The MIT Press, 1990.

[EHR86] H. Ehrig, A. Habel, and B.K. Rosen. Concurrent transformations of relational structures. *Fundamenta Informaticae*, 9:13–50, 1986.

[EK79] H. Ehrig and H.J. Kreowski. Pushout-properties: An analysis of gluing constructions for graphs. *Math. Nachr.*, 91:135–149, 1979.

[EKL90] H. Ehrig, M. Korff, and M. Löwe. Tutorial introduction to the algebraic approach of graph grammars based on double and single pushouts. In *Proc. 4th Intern. Workshop on graph grammars and their application to computer science*, volume 532 of *Lecture Notes in Computer Science*, pages 24–37, 1990.

[EKR90] H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors. *Graph-grammars and their application to computer science*, volume 532 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[ENRR86] H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors. *Graph-grammars and their application to computer science*, volume 291 of *Lecture Notes in Computer Science*. Springer-Verlag, 1986.

[EPS73] H. Ehrig, M. Pfender, and H.J. Schneider. Graph grammars: An algebraic approach. In *Proc. IEEE Conf. SWAT '73*, pages 167–180, Iowa City, 1973.

[ER80] H. Ehrig and B.K. Rosen. Parallelism and concurrency of graph manipulations. *Theoretical Computer Science*, 11:247–275, 1980.

[Ger83] H. Gerster. Informationsystem der Polizei (INPOL). *Datenverarbeitung und Recht*, 12(1/2):19–71, 1983.

[GHK+88] J.R.W. Glauert, K Hammond, J.R. Kennaway, M.R. Sleep, G.W. Somner, N. Holt, M. Reeve, and I. Watson. Extensions to core dactl1. Report SYS-C88-01, University of East Anglia, 1988.

[GKS87] J.R.W. Glauert, J.R. Kennaway, and M.R. Sleep. Dactl: A computational model and compiler target language based on graph reduction. Report SYS-C87-03, University of East Anglia, 1987.

[Gol79] R. Goldblatt. *Topoi, The categorical analysis of logic*. North-Holland, Amsterdam, 1979.

[HP88] B. Hoffmann and D. Plump. Jungle evaluation for efficient term rewriting. In *Algebraic and Logic Programming*, volume 343 of *Lecture Notes in Computer Science*, 1988.

[Hue80] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980.

[Joh77] P. T. Johnstone. *Topos Theory.* Academic Press, 1977.

[Kaw73] Y. Kawahara. Relations in categories with pullbacks. *Mem. Fac. Sci. Kyushu University*, Ser.A 27:149–173, 1973.

[Kaw88] Y. Kawahara. Applications of relational calculus to computer mathematics. *Bull. of Informatics and Cybernetics*, 23:67–78, 1988.

[Kaw90] Y. Kawahara. Pushout-complements and basic concepts of grammars in toposes. *Theoretical Computer Science*, 77:267–289, 1990.

[KB70] D.E. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebras.* Pergamon Press, 1970.

[Ken87] R. Kennaway. On "On graph rewritings". *Theoretical Computer Science*, 52:37–58, 1987.

[Ken90] R. Kennaway. Graph rewriting in some categories of partial morphisms. In *Proc. 4th Intern. Workshop on graph grammars and their application to computer science*, volume 532 of *Lecture Notes in Computer Science*, pages 490–504, 1990.

[KKSV91] R. Kennaway, J.W. Klop, M.R. Sleep, and F.J. de Vries. Transfinite reductions in orthogonal term rewriting systems. In *Proc. 4th Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1991.

[KMa] Y. Kawahara and Y. Mizoguchi. A categorical rewritings of relational structures. manuscript.

[KMb] Y. Kawahara and Y. Mizoguchi. A relational calculus in topoi. manuscript.

[KM88a] Y. Kawahara and Y. Mizoguchi. Axiomatic semantics in elementary toposes. *Papers in RIMS, Kyoto Univ.*, 666:1–7, 1988.

[KM88b] Y. Kawahara and Y. Mizoguchi. Axiomatic semantics in elementary toposes. RMC 63–13E, Department of Mathematics, Kyushu University, 1988.

[KRB77] N.M. Khan, K. Rajamani, and S.K. Banerjee. A direct method to calculate the frequency and duration of failures from large networks. *IEEE Transactions on Reliability Analysis*, R-26(5):318–321, 1977.

[Löw89] M. Löwe. Algebraic approach to graph transformation based on single pushout derivations. Technical Report 89/26, Technical University of Berlin, 1989.

[Löw91] M. Löwe. *Extended algebraic graph transformation.* PhD thesis, Technical University of Berlin, 1991.

[LE90] M. Löwe and H. Ehrig. Algebraic approach to graph transformation based on single pushout derivations. In *Proc. 16th Intern. workshop on graph-theoretic concepts in computer science*, volume 484 of *Lecture Notes in Computer Science*, pages 338–353, 1990.

[LMS91] I. Litovsky, Y. Metivier, and E. SSopena. Definitions and comparisons of local computations on graphs. LaBRI 91-43, Université de Bordeaux I, 1991.

[LMZ91] I. Litovsky, Y. Metivier, and W Zielonka. The power and the limitations of local computations on graph and networks. LaBRI 91-31, Université de Bordeaux I, 1991.

[Möh90] R.H. Möhring, editor. *Graph-Theoretic Concepts in Computer Science*, volume 484 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[MA86] E.G. Manes and M.A. Arbib. *Algebraic approaches to program semantics.* Springer-Verlag, 1986.

[Man76] Ernest G. Manes. *Algebraic Theories.* Springer-Verlag, 1976.

[Man86] E.G. Manes. Weakest preconditions: categoral insights. In *Category Theory and Computer Programming*, volume 240 of *Lecture Notes in Computer Science*, pages 182–197, 1986.

[Miy78] S. Miyano. On an automaton which recognizes a family of automata. *Mem. Fac. Sci. Kyushu University*, Ser.A 32:37–51, 1978.

[Miz92a] Y. Mizoguchi. A graph reductions system using graph terms. *Bull. of Informatics and Cybernetics*, 25(1–2):27–40, 1992.

[Miz92b] Y Mizoguchi. A graph structure over the category of sets and partial functions. RIFIS TR-CS 53, Research Institute of Fundamental Information Science, Kyushu University, 1992.

[MK91] Y. Mizoguchi and Y. Kawahara. Graph rewritings without gluing conditions. RIFIS TR-CS 42, Research Institute of Fundamental Information Science, Kyushu University, 1991.

[ML72] S. Mac Lane. *Categories for the working mathematician.* Springer-Verlag, 1972.

[MOK87] Y. Mizoguchi, H. Ohtsuka, and Y. Kawahara. A symbolic calculus of regular expressions. *Bull. of Informatics and Cybernetics*, 22:165–170, 1987.

[OH91] Y. Okada and M. Hayashi. Graph rewriting systems and their application to network reliability analysis. In *Proc. 17th Intern. workshop on graph-theoretic concepts in computer science*, volume 570 of *Lecture Notes in Computer Science*, pages 36–47, 1991.

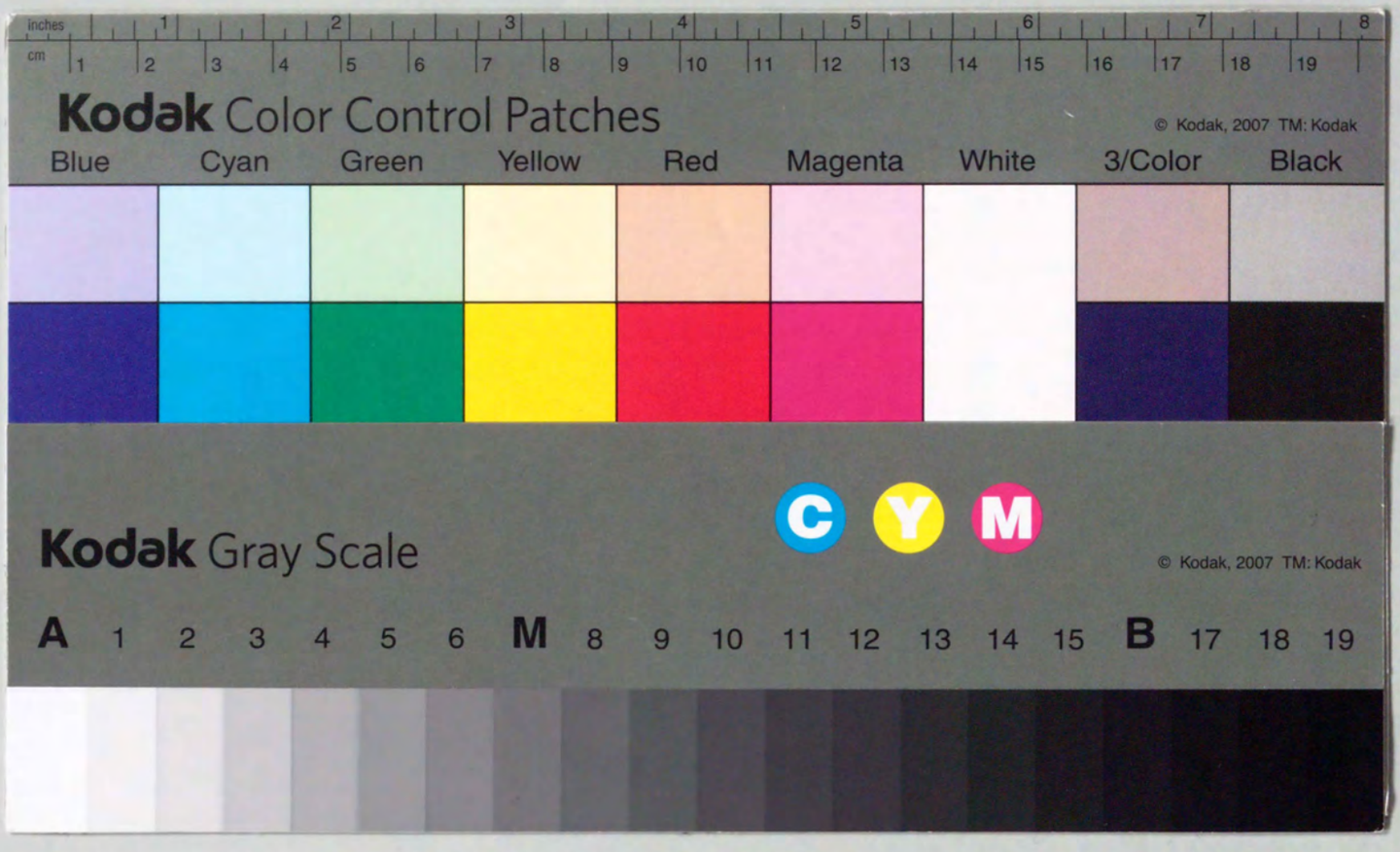
[Rao84] J.C. Raoult. On graph rewritings. *Theoretical Computer Science*, 32:1–24, 1984.

[RR88] E. Robinson and G Rosolini. Categories of partial maps. *Information and Computation*, 79:95–130, 1988.

[SE76] H.J. Schneider and H. Ehrig. Grammars on partial graphs. *Acta Informatica*, 6:297–316, 1976.

[SW85] A. Satayanarayana and R.K. Wood. A linear-time algorithm from computing K-terminal reliability of normal graphs in series-parallel network. *SIAM J. Computing*, 14, 1985.

[Tur79] D.A. Turner. A new implementation technique applicative languages. *Software-practice and experience*, 9:31–49, 1979.


inches
cm
Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19