The Mathematics of Record Handling

Hartmut Ehrig

Fachbereich Informatik
Technische Universität Berlin
1000 Berlin 10
Federal Republic of Germany

Barry K. Rosen

Computer Sciences Department
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598
United States of America

Abstract: We propose a mathematical foundation for reasoning about the correctness and computational complexity of record handling algorithms, using algebraic methods recently introduced in graph theory. A class of pattern matching and replacement rules for graphs is specified, such that applications of rules in the class can readily be programmed as rapid transformations of record structures. When transformations of record structures are formalized as applications of rules to appropriate graphs, recent Church-Rosser type theorems of algebraic graph theory become available for proving that families of transformations are well behaved. In particular, we show that any Church-Rosser family of transformations can be combined with housekeeping operations involving indirect pointers and garbage collection without losing the Church-Rosser property, provided certain mild conditions on the rules defining the family are satisfied. This leads to suggestions for the design of record handling facilities in high level languages, especially when housekeeping chores are to be performed asynchronously by service processes that run in parallel with the main process. We also show how to express the net effect of two transformations as a single transformation. These results and the general theorems that support them can be used to analyze the behavior of a large record structure that can be updated asynchronously by several parallel processes or users.

1. INTRODUCTION

We use some well known concepts from record handling that are reviewed in Section 2. Slight familiarity with [Kn73, Sec. 2.1] or [WH66, Sec. 5.4] should be enough background for reading this introduction. One major difference between record handling and numerical computing is the lack of a standard mathematical foundation. Graph theory is the obvious place to look for appropriate material, since a record structure is naturally viewed as a directed graph wherein each record is a node and each pointer from one record to another is an arc. There is a voluminous literature on algorithms involving graphs, but the relevance of this literature to many record handling problems is doubtful. Running time linear in the size of an input graph is ordinarily and rightly considered fast in the literature. But processes requiring that an entire set of records be scanned are ordinarily and rightly considered slow in record handling. This is not to disparage the literature: we merely want to emphasize that different problems arise in record handling. In particular, record handling algorithms often deal with local properties and transformations on the graph, and with possibly unpleasant interactions between transformations when several users can update a data base asynchronously. An algebraic approach to graph theory has recently been introduced, partly to deal with these concerns. This paper's results are applicable to record handling, but the presentation of these results is also a convenient occasion to explain the algebraic approach in a new way. The earliest work [EPS73; Ro75] had unusual prerequisites and some technical complexities that were later eliminated [ER76, Sec. 2]. Enough results have now accumulated to support an exposition that is intelligible and perhaps even plausible without prerequisites. Those who wish to read the proofs carefully will need to consult [ER76] and occasionally [Ro75], but only as indicated by citations. A two pass reading is recommended, with proofs omitted on the first pass. In this version of the paper all proofs have been shortened or omitted. Phrases like "can be shown" are used to warn of shortening. The complete paper will be submitted to a journal.

Section 2 relates transformations of record structures to "productions" that can be applied to graphs. Unlike [EK75; EPS73; ER76; Ro75], this section should be accessible to those not already familiar with some basic aspects of category theory. Sections 3 and 4 present fundamental existence theorems that are applied here but are not limited to record handling. Suppose G can be transformed to $G^{\#}$ and G is a subgraph of Γ. Under certain conditions, the "same" transformations can be applied to Γ, yielding a graph $\Gamma^{\#}$ with $G^{\#}$ as a subgraph. Moreover,

$\Gamma^{\#}$ and the embedding of $G^{\#}$ into $\Gamma^{\#}$ depend only on G and the embedding of G into Γ, *not* on the specific derivation from G to $G^{\#}$. (Small liberties have been taken in this introductory paraphrase.)

The next two sections deal with housekeeping tasks in record structures: maintainence of indirect pointers in Section 5 and a simple form of garbage collection in Section 6. We establish conditions under which housekeeping operations do not interfere with whatever else is being done. For example, list processing should not be stymied because a little garbage has been collected by a second processor, operating asynchronously in parallel with the main processor. (The resulting suggestions for language design are collected in Section 7.) Noninterference is formalized by the "Church-Rosser property" (defined here essentially as in [Ro73, Def. 3.2]), involving an arbitrary set of objects and a single relation among the objects. Given a set B and a relation $>>$ on B, consider any $\mathsf{F} \subseteq \mathsf{B}$ such that F is *closed* under $>>$: if **G** is in F and **G** $>>$ **H** then **H** is in F. Then the system $(\mathsf{F}, >>)$ is *Church-Rosser* iff

$$(\forall\ \mathbf{G}, \mathbf{H}, \mathbf{H}^{\#} \text{ in } \mathsf{F})\ [(\mathbf{G} >>^{*} \mathbf{H} \ \&\ \mathbf{G} >>^{*} \mathbf{H}^{\#}) \text{ implies } (\exists\ \mathbf{X} \text{ in } \mathsf{B})(\mathbf{H} >>^{*} \mathbf{X} \ \&\ \mathbf{H}^{\#} >>^{*} \mathbf{X})], \tag{1.1}$$

where $>>^{*}$ is the reflexive transitive closure of $>>$. If **G** is in F and

$$\mathbf{G} >>^{*} \mathbf{H} \text{ and } \neg(\exists\ \mathbf{X} \text{ in } \mathsf{B})(\mathbf{H} >> \mathbf{X}) \tag{1.2}$$

then **H** is a *normal form* for **G** in $(\mathsf{F}, >>)$. The Church-Rosser property implies that normal forms are unique when they exist at all. Our (1.1) is a special case of the more general Church-Rosser property involving two relations as studied in [Se74; Ro76]. Some results from [Ro73, Sec. 3] are cited in proofs here. By distinguishing F from B we gain some notational convenience over [Ro73; Ro76; Se74] in applications.

2. RECORD STRUCTURES AND EXPRESSION GRAPHS

Complex information can be represented in a computing system's memory by distributing it over many "records", each of which has a relatively small number of "fields". A field may directly contain a little information, such as a short string of characters or an integer that can be represented with 15 bits, or it may contain a "pointer" to another record that must be consulted if more information is desired. (In some applications a pointer might loop back to the same record.) Records are often divided into "classes", so that all records in a class have the same number of fields, the same names for the respective fields, and the same kinds of direct or pointer data in their fields. (Different fields may contain different kinds of data.) These concepts are well known [Kn73, Sec. 2.1; WH66, Sec. 5.4] but appear in many places under many names. We will speak as above. A set of records such that each pointer is to a record in the set is a *record structure* and we associate with each record structure a *root* record accessible to the outside world. Other records can only be reached by following pointers. (For a practical situation where records RD1, RD2, ..., RDn are externally accessible, we assume there is a special root record with n fields, such that field k points to RDk.) We do *not* assume that all records in a structure are reachable from the root or that the record structure is acyclic, though both these properties hold and are important in many examples. We formalize the intuitive concept of record structures with the mathematical concept of *expression graphs*. Such a graph is a triple (G, m_G, e_G), where G is a finite directed graph. A *node* in G corresponds to a record. There is an *arc* from node x to node y whenever record x has a field that points to record y. (If there are two such fields then there are two such arcs; we do *not* assume that arcs are pairs of nodes.) Nodes and arcs are both called *items*, and m_G maps items to packets of information called *colors*. The color of a node tells how many fields it has and what their names are and what data is stored directly in fields that do not contain pointers. The color of an arc from x to y indicates which field of x is responsible for the arc's being in G. Mathematically, we just have nonempty sets of *node colors* and *arc colors* with m_G mapping nodes to node colors and arcs to arc colors. Finally, the root of the record structure corresponds to the node e_G of the graph. We will be casual about the distinction between the intuitive concepts like "record" and the mathematical concepts like "node" whenever there is no danger of confusion.

Suppose RD1 is a record and we wish to replace all pointers to RD1 in the record structure with pointers to another record RD2. In many situations it is expensive to find all pointers to RD1: only pointers *from* RD1 are

stored as fields of RD1. It is also intuitively plausible that we only really need to change whatever pointers will actually be followed in the future. In this context it is natural to simply replace RD1's data without moving it. We change RD1 so that it has just one pointer field, pointing to RD2. The direct data at RD1 is changed to indicate that RD1 is now merely a dummy record, so that pointers to RD1 will be treated as indirect pointers to RD2. In the future, whenever we follow a pointer from RD3 to RD1, we will update the appropriate field of RD3 so as to point directly to RD2 if the pointer is followed again. One of the contexts where this natural use of *indirection* is especially appropriate is in the efficient evaluation of recursively defined functions, as is discussed in [OD76, App. A]. The discussion above and in [OD76, App. A] can be formalized easily. We assume there is a distinguished node color I such that any node colored I in an expression graph has exactly one outarc, and this outarc carries the distinguished arc color *ind*. For an arbitrary arc color c, suppose our expression graph (G, m_G, e_G) includes an arc z whose target (the node it points to) is colored I. This node tz has a unique outarc with a target w, and w is the record indirectly pointed to by pointers to tz. Thus z should be replaced by a new arc ζ with the same color c and the same source (the node it points from) that z has in G. In the new expression graph (H, m_H, e_H), the target of ζ is w. Otherwise the new expression graph is like the old one. For example, consider (G, m_G, e_G) as shown on the left in Figure 2.1. Then (H, m_H, e_H) is as shown on the right in Figure 2.1. The role of the colored graph (D, m_D) in the figure is explained below.

To specify the transformation precisely without committing ourselves to any one programming language, we use a "production" in the sense of algebraic graph theory. First, note that roots of expression graphs are just carried along in this transformation: e_H is the same record e_G. Simplifying notation by forgetting about roots as often as possible, we consider the *colored graphs* (G, m_G) and (H, m_H). We pass from (G, m_G) to (H, m_H) by applying a *production* $p = [(B_1, m_1) \leftarrow K \rightarrow (B_2, m_2)]$ consisting of a graph K, colored graphs (B_i, m_i), and maps b_i: $K \rightarrow B_i$. These maps are required to be *graph morphisms:* if x is an arc in K with source $s_K x$ and target $t_K x$, then the image arc $b_i x$ in B_i has sources and targets

$$s_i b_i x = b_i s_K x \text{ and } t_i b_i x = b_i t_K x. \tag{2.1}$$

The specific production $p_{ind}(c)$ that retargets an arc colored c whose target is colored I appears in Figure 2.2. In addition to the node color I and the arc colors c, *ind* already introduced, we use node colors u, v in (B_i, m_i) as variables to represent whatever color may actually appear on the nodes in G. The first step in applying p to (G, m_G) is to "recolor" the variables u, v to appropriate node colors ru, rv. The resulting production $rp_{ind}(c)$ has colored graphs (B_i, rm_i) wherein any node x in B_i has the color $rm_i x$. (As a map from colors to colors, r leaves all colors but u, v fixed.) For the example (G, m_G) in Figure 2.1, we use ru = RD3 and rv = RD2. The algebraic construction that applies $rp_{ind}(c)$ to (G, m_G) at the arc z with $m_G z = c$ is summarized in Figure 2.3.

The horizontal arrows at the top of Figure 2.3 come from the production $rp_{ind}(c)$, and we have added colorings $m_{K,i}$ of K such that b_1 and b_2 are *colored graph morphisms* now: they preserve colors as well as sources and targets. In this example $m_{K,1} = m_{K,2}$, but in general a production might change some colors: some x in K has $rm_1 b_1 x \neq rm_2 b_2 x$. The colored graph morphism g : $(B_1, rm_1) \rightarrow (G, m_G)$ picks out the "three" nodes and "two" arcs in G where the production is to be applied. (We use the quotation marks because there is no requirement that g or any other morphisms here be injective.) The image subgraph gB_1 in G may have connections with the rest of G: an arc in $G - gB_1$ may have a source or target in gB_1. We require that such a source or target be in gb_1K, so that the connections can be displayed by graph morphisms d : $K \rightarrow D$ and c_1: $D \rightarrow G$ such that the composition $c_1 d$ is equal to the composition gb_1. These connections are retained when (B_2, rm_2) replaces (B_1, rm_1) to form (H, m_H): there is a graph morphism c_2 with $c_2 d = hb_2$, where h displays how B_2 fits into H after the transformation. Thus Figure 2.3 is a commutative diagram in the category GRAPHS[C] of (finite directed) graphs together with colorings m_G: (items in G) $\rightarrow$ C, where C is the set of colors used. (Actually, C is a pair of sets, one for nodes and one for arcs.) The morphisms in this category are of course the colored graph morphisms defined above. Explanations of the common terms from category theory just used are in [HS73, Sec. 3] and other standard references. For any recolored production rp, not just our example $rp_{ind}(c)$, Figure 2.3 is called a *unit derivation* $(G, m_G) \Rightarrow (H, m_H)$ *via* rp *based on* g provided that two conditions hold. First, $m_{D,1}$ only differs from $m_{D,2}$ as required by color changes specified in p:

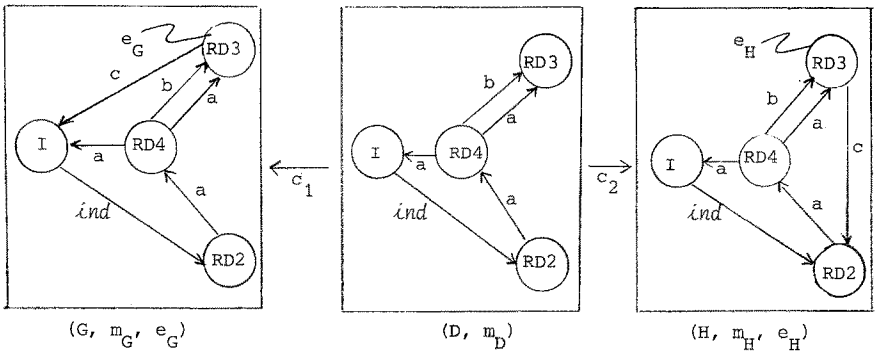

Figure 2.1. In (G, m_G, e_G) the field c of record RD3 points indirectly to RD2. In (H, m_H, e_H) the pointer data has been changed so as to point directly to RD2.

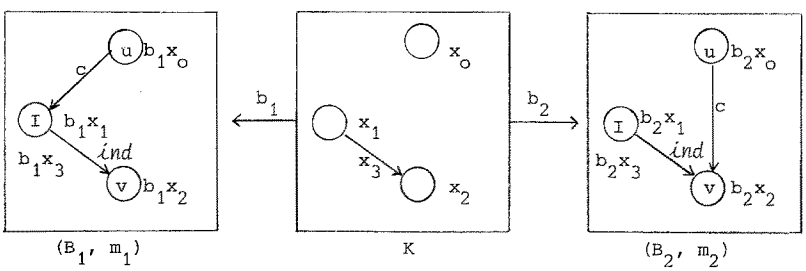

Figure 2.2. A production consists of an interface graph K, left and right colored graphs (B_i, m_i), and graph morphisms $b_i : K \to B_i$.

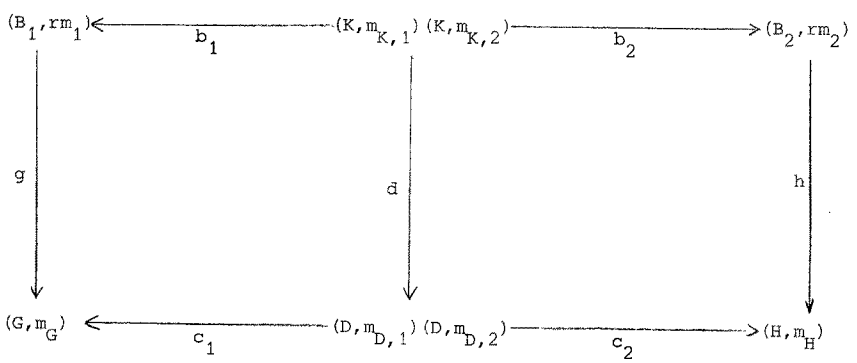

Figure 2.3. A derivation consists of two pushouts in the category of colored graphs that share a graph morphism $d : K \to D$.

$$m_{D,1}y = m_{D,2}y \text{ for all } y \text{ in } D - dK. \tag{2.2.1}$$

Second, the two squares in Figure 2.3 completely describe (G, m_G) and (H, m_H) in the following precise sense:

$$\text{both squares are pushouts in GRAPHS[C].} \tag{2.2.2}$$

See [HS73, Sec. 21] for the pushout concept, which is fundamental for the developments in [EPS73; ER76; Ro75]. Most of this paper can be read with only the knowledge that pushouts are commutative squares. For the example (G, m_G) in Figure 2.1, there is a unit derivation $(G, m_G) \Rightarrow (H, m_H)$ via $rp_{ind}(c)$ based on $g : B_1 \to G$, where B_1 is from Figure 2.2. The morphism g maps the arc colored c in (B_1, rm_1) to the arc colored c in (G, m_G) and it maps the arc colored *ind* in (B_1, rm_1) to the arc colored *ind* in (G, m_G). The horizontal arrows at the bottom of Figure 2.3 are shown in detail in Figure 2.1, with $m_{D,1} = m_{D,2}$ in this example. In general, derivations via $rp_{ind}(c)$ will have $m_{D,1} = m_{D,2}$ but derivations via many other productions will not. In this example the morphisms are all injective, but this is not required in general. We could easily have $g^{\#}b_1x_0 = g^{\#}b_1x_2$ in an application of $rp_{ind}(c)$ to some other graph $(G^{\#}, m_G{}^{\#})$.

Of course a derivation is a sequence of unit derivations, and it is via the sequence of productions used. The qualifier "unit" is often omitted when confusion is unlikely. Of special interest are the *natural* derivations: those with c_1 injective. In our example b_1 is injective, and this implies (by an elementary property of pushouts) that any derivation via $rp_{ind}(c)$ is natural. We have now reviewed all but the last paragraph of [ER76, Sec. 2], but in a different order and with stress on the record handling application. Reading that portion of [ER76] now would be a good way to solidify one's understanding of the basic concepts and prevent any possible confusion between the general requirements of the theory and the specifics of the initial example we have used.

The finiteness of graphs is not required in [ER76] but will sometimes be helpful here, so we restrict attention to finite graphs at the start. Because of our interest in manipulating large record structures, however, we want to avoid productions such that the work required to apply rp to (G, m_G) depends on the size of G. In particular, suppose there is a node y in $B_1 - b_1K$. Then we cannot apply rp with $g : B_1 \to G$ unless the node gy in G has no inarcs or outarcs beyond those in gB_1. This is a property of pushouts in GRAPHS[C] reflecting the intuition that such arcs would be left "dangling" when B_2 replaced B_1. But to list all the inarcs of gy in G requires either an exhaustive scan of G or maintainence of housekeeping information such as backpointers or reference counts. For garbage collection we cannot escape such costs, but for other transformations we would like each production to be amenable to rapid implementation. Given rp and $g : B_1 \to G$, it should be possible to determine whether a natural derivation via rp based on g can be constructed (and then construct one if the answer is yes) in such a way that the costs depend on the sizes of graphs in rp but not on the size of G. Mathematically, p is said to be *fast* iff

$$b_1 \text{ is surjective on nodes;} \tag{2.3.1}$$

$$b_1 \text{ and } b_2 \text{ are injective.} \tag{2.3.2}$$

All productions in [ER76, Secs. 7, 8] are fast. Before showing that fast productions can be applied rapidly, we state a useful general lemma that can be derived from [Ro75, Lemmas 4.1, 4.2].

LEMMA 2.4. Given an injective graph morphism $b_1 : K \to B_1$ and a colored graph morphism $g : (B_1, m_1) \to (G, m_G)$, consider the pair (nodes, arcs) of sets

$$(N_0, A_0) = G - gB_1. \tag{1}$$

There is a pushout of the form

$$\begin{array}{ccc} (B_1, m_1) & \xleftarrow{\;b_1\;} & (K, m_K) \\ \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle d} \\ (G, m_G) & \xleftarrow{\;c_1\;} & (D, m_D) \end{array} \tag{2}$$

if and only if

$$s_G A_O \cup t_G A_O \subseteq N_O \cup gb_1 K; \tag{3}$$

$$(\forall\, y, y' \text{ in } B_1)[\, gy = gy' \text{ implies } (y = y' \text{ or } y, y' \text{ in } b_1 K)\,]. \tag{4}$$

In that case (2) is unique up to isomorphism.

COROLLARY 2.5. There is an algorithm that, given a colored graph (G, m_G), a recoloring $r : C \to C$, a fast production p, and a graph morphism $g : B_1 \to G$, determines whether there is a derivation $(G, m_G) \Rightarrow (H, m_H)$ via rp based on g and constructs one if possible. The derivation is unique up to isomorphism and the number of steps required by the procedure is independent of the size of G provided that evaluations of relevant maps (g, m_G, s_G, t_G) can be done in single steps.

Proof. It is easy to reduce the problem of constructing derivations to that of constructing "analyses" (the left square in Figure 2.3). More precisely, an *analysis* of (G, m_G) for rp based on g is any pushout as in Lemma 2.4(2) such that

$$(\forall\, x, x' \text{ in } K)(\, gb_1 x = gb_1 x' \text{ implies } rm_2 b_2 x = rm_2 b_2 x'\,). \tag{1}$$

Given the pushout, we can test whether (1) holds in time independent of the size of G. By [Ro75, Thm. 3.6], an analysis determines a unique derivation. For fast productions the number of steps required to pass from an analysis to a derivation is independent of the size of G because b_2 is injective. Thus the whole derivation problem reduces to the problem addressed in Lemma 2.4. For fast productions Lemma 2.4(3) always holds because b_1 is surjective on nodes. Whether Lemma 2.4(4) holds can be checked in time independent of the size of G. ■

Colored graph concepts can be extended to expression graphs by carrying roots along. In particular, suppose we have a natural derivation

$$(G, m_G) \Rightarrow (H, m_H) \text{ via rp based on g} \tag{2.6.1}$$

and a node e_G in G such that

$$e_G \text{ is in } c_1 D. \tag{2.6.2}$$

Then there is a unique expression graph (H, m_H, e_H) with

$$e_H = c_2 c_1^{-1} e_G, \tag{2.7.1}$$

so we say there is a natural derivation

$$(G, m_G, e_G) \Rightarrow (H, m_H, e_H) \text{ via rp based on g.} \tag{2.7.2}$$

Similarly for other concepts. In particular, (2.6.2) always holds if p is fast.

Graph morphisms $g : B_1 \to G$ provide a flexible way to say *where* we are applying a production, but in most examples we only need the values of g on a few items in B_1 to determine the rest. In particular, let y be the arc colored c in Figure 2.2. Because nodes colored I have unique outarcs in the expression graphs we consider, g is determined by gy. The pair $(p_{ind}(c), y)$ is a *rule*, as is any pair consisting of a production p and an item y in B_1. In a derivation based on g the rule is applied *at* gy. This wording has no theoretical significance, but it enhances brevity and clarity whenever the rule is such that g can be recovered from knowledge of gy alone.

We restrict the set R of recolorings $r : C \to C$ a little more than necessary rather than burden the statements of theorems with assumptions about R that are weak but difficult to remember. Specializing (5.5) and (5.7) from [ER76] for technical convenience here, we consider the *fixed* colors C_{fix} and the *variable* colors C_{var}:

$$C_{fix} = \{\, c \text{ in } C \mid (\forall\, r \text{ in } R)(rc = c)\,\} \text{ and } C_{var} = C - C_{fix}. \tag{2.8.1}$$

Now R is assumed to contain everything it might conceivably contain, with no correlation between the recolorings of different colors:

$$R = \{\, r : C \to C \mid (\forall\, c \text{ in } C_{fix})(rc = c)\,\}. \tag{2.8.2}$$

Derivations via $Rp_{ind}(c)$ (which is to say, derivations via $rp_{ind}(c)$ for some r in R) now have the intended effect.

3. EXISTENCE THEOREMS FOR DERIVATIONS

This section presents fundamental existence theorems for derivations among colored graphs and among expression graphs. We begin by paraphrasing the early material in order to save space. Displays (3.1) and (3.2) pose the problem of showing that natural derivations among expression graphs *commute*: given $\mathbf{G} \Rightarrow \mathbf{H}$ via Rp and $\mathbf{G} \Rightarrow \mathbf{H}^{\#}$ via $\mathrm{Rp}^{\#}$, can we obtain $\mathbf{H} \Rightarrow \mathbf{X}$ via $\mathrm{Rp}^{\#}$ and $\mathbf{H}^{\#} \Rightarrow \mathbf{X}$ via Rp for a single expression graph **X**? For colored graphs there are several commutativity results in [ER76]. It should be possible to carry along roots of expression graphs in these results. Lemma 3.3 (and slight variants that even the complete paper need not state explicitly) allows us to do this, so that conclusions about expression graphs can be derived from reasoning about colored graphs. For example, Lemma 3.3 and [ER76, Thm. 4.7] yield

COROLLARY 3.4. Independent natural derivations among expression graphs commute.

The corollaries [ER76, Sec. 5] of the basic commutativity theorem also apply to expression graphs. In Section 6 the distinction between colored graphs and expression graphs will be important, since collecting the root of an expression graph as garbage would be wrong even if there were no pointers to the root in the graph. Until then, Lemma 3.3 and its variants will allow us to ignore the distinction. Considering only colored graphs simplifies the following theorem, which is the main result of this section. Given a derivation from (G_0, m_0) to (G_n, m_n) and a morphism $\gamma_0 : (G_0, m_0) \to (\Gamma_0, \mu_0)$, we would like to construct a "corresponding" derivation from (Γ_0, μ_0) to a colored graph (Γ_n, μ_n), with an "embedding" $\gamma_n : (G_n, m_n) \to (\Gamma_n, \mu_n)$. This should be possible if γ_0 is sufficiently close to being injective. Moreover, details of the intermediate steps between (G_0, m_0) and (G_n, m_n) should not affect (Γ_n, μ_n) and γ_n unless they also affect (G_n, m_n). To formulate this intuition precisely requires some ingenuity, but the effort buys the ability to draw conclusions about infinite sets of large graphs from calculations with specific small graphs. A typical application is in Section 5.

THEOREM 3.5. Let there be a colored graph (G_0, m_0) and a derivation

$$(G_0, m_0) \Rightarrow (G_1, m_1) \Rightarrow \ldots \Rightarrow (G_n, m_n) \text{ via } (p_1, \ldots, p_n) \tag{1}$$

where all productions p_j are fast. The *residue* map r_i from nodes of G_0 to nodes of G_i is defined inductively by

$$r_0 z = z \text{ and } r_j z = c_{2j} c_{1j}^{-1} r_{j-1} z \text{ for } 1 \le j \le n, \tag{2}$$

where c_{1j} and c_{2j} are from the step via p_j in (1). Suppose there is a colored graph (Γ_0, μ_0) and a morphism $\gamma_0 : (G_0, m_0) \to (\Gamma_0, \mu_0)$ such that

$$\gamma_0 \text{ is injective on arcs;} \tag{3}$$

$$(\forall\, z, z' \text{ nodes in } G_0)\ [\ \gamma_0 z = \gamma_0 z' \text{ implies } (\forall\, j)(m_j r_j z = m_j r_j z')\ ]. \tag{4}$$

Then there are colored graphs (Γ_j, μ_j) and a derivation

$$(\Gamma_0, \mu_0) \Rightarrow (\Gamma_1, \mu_1) \Rightarrow \ldots \Rightarrow (\Gamma_n, \mu_n) \text{ via } (p_1, \ldots, p_n). \tag{5}$$

Moreover, there are graphs S and Σ and graph morphisms $f_0 : S \to G_0$ and $\sigma : S \to \Sigma$, depending only on γ_0, such that for every i with $0 \le i \le n$ there is a pushout

$$\begin{array}{ccc} (G_i, m_i) & \xleftarrow{\ f_i\ } & (S, n_i) \\ \Big\downarrow{\scriptstyle \gamma_i} & & \Big\downarrow{\scriptstyle \sigma} \\ (\Gamma_i, \mu_i) & \xleftarrow{\ \phi_i\ } & (\Sigma, \nu_i) \end{array} \tag{6}$$

with colorings n_i and ν_i that depend only on γ_0, μ_0 and on $m_i r_i$. There are no arcs in S and f_i is just $r_i f_0$ on nodes.

Proof. We begin by obtaining (6) for i = 0. Let the nodes of S be all nodes z of G_0 such that $\gamma_0 z$ is a source or target of an arc in $\Gamma_0 - \gamma_0 G_0$, together with all nodes z of G_0 such that $\gamma_0 z = \gamma_0 z'$ for some $z' \ne z$. Let S have

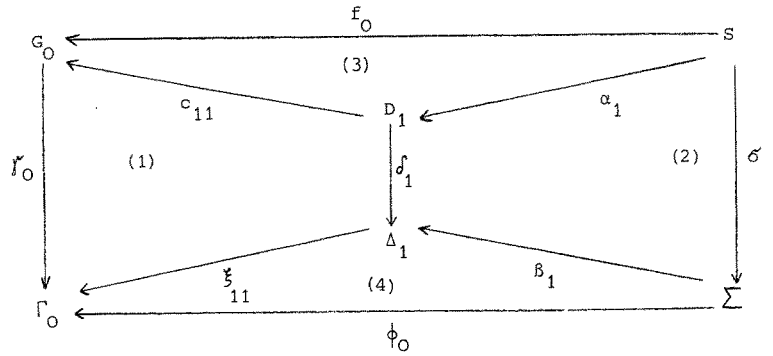

Figure 3.1. Colorings are suppressed for readability. There are colorings $m_{1,1}$ and $m_{1,2}$ of D_1 in the step via p_1. We use $m_{1,1}$ here because it is preserved by $c_{11} : (D_1, m_{1,1}) \to (G_0, m_0)$.

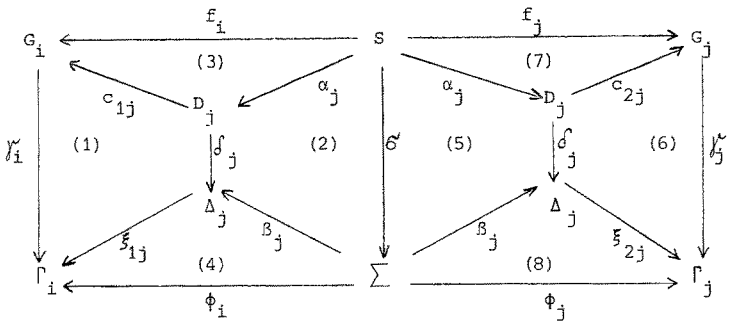

Figure 3.2. In (1)–(4), coloring $m_{j,1}$ of D_j in the step via p_j is preserved by c_{1j}. In (5)–(8), the other coloring $m_{j,2}$ is preserved by c_{2j}. As in (2.2.1), $m_{j,1}$ agrees with $m_{j,2}$ on $D_j - d_jK_j$.

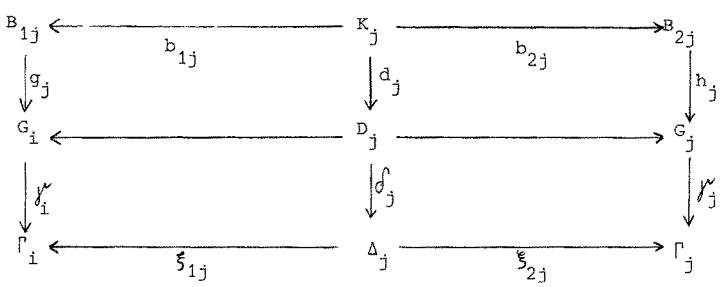

Figure 3.3. Two pushouts, one from b_{1j} and $\delta_j d_j$ and the other from b_{2j} and $\delta_j d_j$, will be shown to form a derivation from (Γ_i, μ_i) to (Γ_j, μ_j).

no arcs. Let f_0 be the inclusion of S in G_0. For each j with $1 \le j \le n$ let f_j be $r_j f_0$ on nodes and empty on arcs. Because S has no arcs and c_{1j} is bijective on nodes, there is a unique

$$\alpha_j : S \to D_j \text{ with } c_{1j}\alpha_j = f_{j-1} \text{ and } c_{2j}\alpha_j = f_j. \tag{7}$$

Lemma 2.4 is applicable to the morphisms c_{11} and γ_0, yielding the pushout in Figure 3.1(1). Lemma 2.4 is applicable to the morphisms α_1 and δ_1, yielding Figure 3.1(2). Then Figure 3.1(3) comes from (7) and Figure 3.1(4) defines $\phi_0 : \Sigma \to \Gamma_0$. When two pushouts are combined in this way the outer square is a pushout [ER76, Lemma 4.9], so we have obtained (6) for i = 0. The colorings n_0 and ν_0 depend only on γ_0 and μ_0. By construction, there are no arcs in S and f_0 is $r_0 f_0$ on nodes.

In seeking (6) for each j with $1 \le j \le n$ we use the morphism $f_j : S \to G_j$ already specified. We proceed by induction on j. Suppose j is such that for i = j−1 we have the situation shown in Figure 3.2(1−4), as is the case if j = 1 by Figure 3.1. The colorings of S, D_j, Σ in Figure 3.2(2) are n_i, $m_{j,1}$, ν_i. There is a unique coloring n_j of S such that α_j preserves colors when $m_{j,2}$ is used to color D_j. We claim there is a unique coloring ν_j of Σ such that σ preserves n_j but ν_j agrees with ν_i on $\Sigma - \sigma S$. It will suffice to show that any z, z' in S with $z \ne z'$ but $\sigma z = \sigma z'$ have $m_{j,2}\alpha_j z = m_{j,2}\alpha_j z'$. But $\sigma z = \sigma z'$ implies $\gamma_0 f_0 z = \gamma_0 f_0 z'$ and hence by (7) and (4) that $m_{j,2}\alpha_j z = m_j c_{2j}\alpha_j z = m_j f_j z = m_j r_j f_0 z = m_j r_j f_0 z' = m_{j,2}\alpha_j z'$. With colorings n_j, $m_{j,2}$, ν_j for S, D_j, Σ in Figure 3.2(2) we get Figure 3.2(5). Pushing out from c_{2j} and δ_j yields Figure 3.2(6), with a coloring μ_j of Γ_j as well as the graph itself. Figure 3.2(7) comes from (7) and Figure 3.2(8) defines a morphism ϕ_j. By [ER76, Lemma 4.9] again, the outer square in Figure 3.2(5−8) is a pushout. We have obtained (6) for j, with the desired limited dependencies of n_j and ν_j. To continue the induction we must prove a little more: (6) for j can be factored in the manner of Figure 3.2(1−4) but with j+1 in place of j and with j in place of i. This follows readily from the uniqueness of the pushout from f_j and σ and from $c_{2j}\alpha_j = f_j = c_{1j+1}\alpha_{j+1}$.

All that remains is to verify the existence of a derivation (5). For each j we want a derivation

$$(\Gamma_i, \mu_i) \Rightarrow (\Gamma_j, \mu_j) \text{ via } p_j \text{ based on } \gamma_i g_j \tag{8}$$

with i = j−1. By [ER76, Lemma 4.9] again, the four pushouts in Figure 3.3 combine to form two pushouts like Figure 2.3 but with lettering appropriate for (8). We must show that the colorings $\mu_{j,1}$ and $\mu_{j,2}$ of Δ_j agree on $\Delta_j - \delta_j d_j K_j$. But it can be shown that $\Delta_j - \delta_j d_j K_j \subseteq \beta_j(\Sigma - \sigma S) \cup \delta_j(D_j - d_j K_j)$. On $\beta_j(\Sigma - \sigma S)$ the agreement between ν_i and ν_j on $\Sigma - \sigma S$ implies agreement between $\mu_{j,1}$ and $\mu_{j,2}$. On $\delta_j(D_j - d_j K_j)$ the agreement between $m_{j,1}$ and $m_{j,2}$ on $D_j - d_j K_j$ implies agreement between $\mu_{j,1}$ and $\mu_{j,2}$. ■

COROLLARY 3.6. As in Theorem 3.5, suppose there is also another derivation

$$(G_0, m_0) = (G_0^{\#}, m_0^{\#}) \Rightarrow (G_1^{\#}, m_1^{\#}) \Rightarrow \ldots \Rightarrow (G_{n^{\#}}^{\#}, m_{n^{\#}}^{\#}) \text{ via } (p_1^{\#}, \ldots, p_{n^{\#}}^{\#})$$

satisfying the same hypotheses. Suppose $r_{n^{\#}}^{\#} = r_n$ and $(G_{n^{\#}}^{\#}, m_{n^{\#}}^{\#}) = (G_n, m_n)$. Then a single colored graph (Γ_n, μ_n) has derivations

$$(\Gamma_0, \mu_0) \Rightarrow (\Gamma_n, \mu_n) \text{ via } (p_1, \ldots, p_n) \text{ and } (\Gamma_0, \mu_0) \Rightarrow (\Gamma_n, \mu_n) \text{ via } (p_1^{\#}, \ldots, p_{n^{\#}}^{\#}).$$

Proof. Apply Theorem 3.5(6) for i = n and for $i = n^{\#}$, then uniqueness of the pushout from f_i and σ. ■

4. AN EXISTENCE THEOREM FOR PRODUCTIONS

The *proper* productions defined by four conditions (5.6.1)—(5.6.4) in [ER76] will be helpful in the next section. The last of the conditions is made trivially true by (2.8) here, so we restate only the first three conditions here for ease of reference:

$$(\forall\, x,y \text{ in } B_1)(m_1 x = m_1 y \text{ in } C_{var} \text{ implies } x = y); \tag{4.1.1}$$

$$(\forall\, y \text{ in } B_2)(m_2 y \text{ in } C_{var} \text{ implies } y \text{ in } b_2 K); \tag{4.1.2}$$

$$(\forall\, x \text{ in } K)(m_1 b_1 x \text{ or } m_2 b_2 x \text{ in } C_{var} \text{ implies } m_1 b_1 x = m_2 b_2 x). \tag{4.1.3}$$

A *biproper* production satisfies these three conditions and their mirror images, with 1 and 2 subscripts reversed. Indirect productions $p_{ind}(c)$ are biproper, as are all the productions in [ER76, Sec. 7]. Productions with only fixed colors are trivially biproper. The proof of the following theorem is too long to be given here.

THEOREM 4.2. Let p and $p^\#$ be biproper fast productions. There is a set $[p \bullet p^\#]$ of biproper fast productions (using only colors that appear in p or in $p^\#$) such that, for every derivation

$$\mathbf{G} \Rightarrow \mathbf{X} \text{ via } (rp, r^\# p^\#) \tag{1}$$

there is a derivation

$$\mathbf{G} \Rightarrow \mathbf{X} \text{ via } \rho q \text{ for some q in } [p \bullet p^\#]. \tag{2}$$

Conversely, for every derivation (2) there is a derivation (1).

We have some additional results of a similar nature that simulate the net effect of applying two productions independently in parallel by applying a single production $[p + p^\#]$. These lead to a generalization of canonical derivation sequences in formal language theory.

5. INDIRECTION

For each fixed arc color c the production $p_{ind}(c)$ shown in Figure 2.2 uses variable node colors u, v and fixed node color I. As in Section 2, we can simplify notation by requiring that any node colored I in an expression graph have a unique outarc, and that this arc be colored *ind*. Let y be the arc colored c in Figure 2.2, so that applying the rule $(p_{ind}(c), y)$ at $z = gy$ in (G, m_G, e_G) has the effect of following the indirect c-pointer from sz and changing the pointer field at sz so as to point directly to the node in G corresponding to the node colored v in Figure 2.2. Given any expression graphs $\mathbf{G} = (G, m_G, e_G)$ and $\mathbf{H} = (H, m_H, e_H)$, let $\mathbf{G} >>_{ind} \mathbf{H}$ iff there is a fixed arc color c, a recoloring r in R, and an arc z in G such that

$$\mathbf{G} \Rightarrow \mathbf{H} \text{ via } rp_{ind}(c) \text{ at z.} \tag{5.1}$$

Given any family F of expression graphs for which indirection makes sense, we can try to establish the Church-Rosser property (1.1) for the system $(\mathsf{F}, >>_{ind})$, no matter what other properties F may have. The family F should be closed under $>>_{ind}$:

$$(\mathbf{G} \text{ is in } \mathsf{F} \text{ and } \mathbf{G} >>_{ind} \mathbf{H}) \text{ implies } \mathbf{H} \text{ is in } \mathsf{F}. \tag{5.2.1}$$

There should be nothing analogous to the tight loop (LABEL : **goto** LABEL) in programming: for all G in F,

$$m_G s_G x = I \text{ implies } t_G x \neq s_G x \text{ for all arcs x in G.} \tag{5.2.2}$$

In particular, (5.2.2) holds if all graphs in F are acyclic. Families that satisfy (5.2) are said to *allow indirection.*

LEMMA 5.3. If F allows indirection then every member of F has a unique normal form in $(\mathsf{F}, >>_{ind})$.

Proof. With the aid of (5.2) it can be shown that the Church-Rosser property here will follow from

$$(\forall\, \mathbf{G}, \mathbf{H}, \mathbf{H}^\# \text{ in } \mathsf{F})\ [(\mathbf{G} >>_{ind} \mathbf{H} \ \&\ \mathbf{G} >>_{ind} \mathbf{H}^\#) \text{ implies } (\exists\, \mathbf{X})(\mathbf{H} >>_{ind}{}^* \mathbf{X} \ \&\ \mathbf{H}^\# >>_{ind}{}^* \mathbf{X})]. \tag{1}$$

We prove (1) by assuming that $\mathbf{G}, \mathbf{H}, \mathbf{H}^\#$ are as above and deriving the existence of an appropriate $\mathbf{X}$. Consider the derivations (5.1) for $\mathbf{G} >>_{ind} \mathbf{H}$ and (5.1#) for $\mathbf{G} >>_{ind} \mathbf{H}^\#$. Disposing of the easy cases with the aid of [ER76, Cor. 5.4], we may assume $z^\# = gb_1x_3$ where x_3 is the unique arc in K, so that gB_1 and $g^\#B_1{}^\#$ overlap as shown in Figure 5.1(top). Note that $c^\#$ here is *ind*. The nodes colored ru and $r^\#v^\#$ in the picture might actually be one node in G, but (5.2.2) requires that all other seemingly distinct nodes in the picture be truly distinct in the graph G. The relevant parts of $\mathbf{H}$ and $\mathbf{H}^\#$ are pictured in Figure 5.1(left) and Figure 5.1(right). As suggested by Figure 5.1(bottom), there is an expression graph $\mathbf{X}$ with $\mathbf{H} >>_{ind} >>_{ind} \mathbf{X}$ by applying $p_{ind}(ind)$ and $p_{ind}(c)$ and with $\mathbf{H}^\# >>_{ind} \mathbf{X}$ by applying $p_{ind}(c)$. To *show* this we apply Corollary 3.6 to embed Figure 5.1 into derivations $\mathbf{G} >>_{ind} \mathbf{H} >>_{ind} >>_{ind} \mathbf{X}$ and $\mathbf{G} >>_{ind} \mathbf{H}^\# >>_{ind} \mathbf{X}$. Hypothesis (4) in Theorem 3.5 follows from the fact that $m_{1i}b_{1i} = m_{2i}b_{2i}$ for all the productions p_i. (The transition from colored graphs to expression graphs is routine, as in Lemma 3.3.) The proof of (1) is complete. ■

The good behavior of $>>_{ind}$ is not in itself very interesting. Given some other well behaved relation $>>_1$ on F, perhaps induced by productions that add I nodes to expression graphs, we would like to show that the union $>>$, with $G >> H$ iff ($G >>_1 H$ or $G >>_{ind} H$), still behaves well. Some mild restrictions on $>>_1$ will be needed. We are concerned with situations where $>>_1$ is induced by a set P of productions. Productions that delete (or change the colors of) I nodes or *ind* arcs could destroy opportunities to apply $p_{ind}(c)$. On the other hand, applying indirect productions could destroy opportunities to apply productions in P such that (B_1, m_1) includes an I node with an inarc. Appropriate restrictions are imposed on P in the next theorem. Before proving the theorem we state a general Church-Rosser lemma that will be used. The lemma can be proved by a straightforward induction.

LEMMA 5.4. Let F be closed under a relation $>>$ on $\mathsf{B} \supseteq \mathsf{F}$. Suppose $>>_4$ is a relation on B such that

$$(>>_4{}^*) \subseteq (>>^*); \tag{1}$$

$$(\forall\, G, H \text{ in } \mathsf{F})\ [G >>^* H \text{ implies } (\exists L)(G >>_4{}^* L \,\&\, H >>^* L)]; \tag{2}$$

$$(\forall\, G, H, H^{\#} \text{ in } \mathsf{F})\ [(G >>_4 H \,\&\, G >>_4 H^{\#}) \text{ implies } (\exists\, X)(H >>_4 X \,\&\, H^{\#} >>_4 X)]. \tag{3}$$

Then $(\mathsf{F}, >>)$ is Church-Rosser.

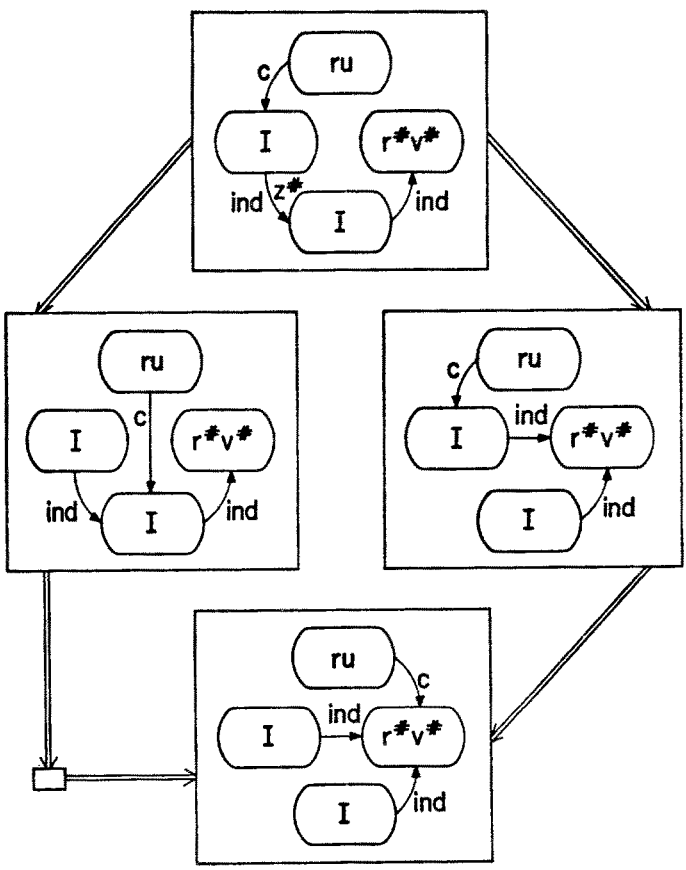

Figure 5.1. Difficult case in the proof of Lemma 5.3.

THEOREM 5.5. Let F allow indirection and let $>>_1$ be a relation on expression graphs such that, for some set P of biproper fast productions wherein no node in B_1 is colored I and no arc in B_1 is colored *ind* or in C_{var},

$$\mathbf{G} >>_1 \mathbf{H} \text{ iff } (\exists\, p \text{ in } \mathsf{P})(\exists\, r \text{ in } R)(\mathbf{G} \Rightarrow \mathbf{H} \text{ via } rp); \tag{1}$$

$$\mathsf{F} \text{ is closed under } >>_1. \tag{2}$$

Let $>>$ be the union of $>>_1$ and $>>_{ind}$. Then $(\mathsf{F}, >>)$ is Church-Rosser if $(\mathsf{F}, >>_1)$ is Church-Rosser.

Proof. Let

$$\mathbf{G} >>_2 \mathbf{H} \text{ iff } \mathbf{G} >>_1{}^* \mathbf{H};$$

$$\mathbf{G} >>_3 \mathbf{H} \text{ iff } [\mathbf{G} \text{ has normal form } \mathbf{H} \text{ in } (\mathsf{F}, >>_{ind})];$$

$$\mathbf{G} >>_4 \mathbf{H} \text{ iff } (\exists\, \mathbf{X})(\mathbf{G} >>_2 \mathbf{X} >>_3 \mathbf{H}).$$

We will apply Lemma 5.4. Assume for the moment that

$$(\mathbf{G} >>_2 \mathbf{H} \,\&\, \mathbf{G} >>_3 \mathbf{H}^{\#}) \text{ implies } (\exists\, \mathbf{W}, \mathbf{X})(\mathbf{H} >>_3 \mathbf{X} \,\&\, \mathbf{H}^{\#} >>_2 \mathbf{W} >>_3 \mathbf{X}). \tag{3}$$

By filling circles in diagrams as in [Ro73, Sec. 3], (3) can be shown to imply

$$(\mathbf{G} >> \mathbf{X} >>_4{}^* \mathbf{Y} >>_4 \mathbf{L}) \text{ implies } (\mathbf{G} >>_4{}^* \mathbf{L}). \tag{4}$$

We can now verify the three hypotheses of Lemma 5.4. Lemma 5.4(1) is trivial. Lemma 5.4(2) can now be verified by induction on n in $\mathbf{G} >>^n \mathbf{H}$. For Lemma 5.4(3) we apply the hypothesis that $(\mathsf{F}, >>_1)$ is Church-Rosser and then apply (3) and uniqueness of normal forms in $(\mathsf{F}, >>_{ind})$. By Lemma 5.4, $(\mathsf{F}, >>)$ is Church-Rosser.

All that remains is to prove (3). Suppose $\mathbf{G} >>_2 \mathbf{H}$ and $\mathbf{G} >>_3 \mathbf{H}^{\#}$. If $\mathbf{G} = \mathbf{H}$ then we may let $\mathbf{W}$ and $\mathbf{X}$ be $\mathbf{H}^{\#}$, so we may assume $\mathbf{G} \neq \mathbf{H}$. Therefore there is a derivation from $\mathbf{G}$ to $\mathbf{H}$ with one or more steps. Theorem 4.2 will allow us to deal only with the case of one step. Let Q be the smallest set of productions that includes P and all productions $[p \bullet p^{\#}]$ whenever p and $p^{\#}$ are in Q. In Theorem 4.2 we observe that members of Q satisfy all the restrictions on members of P. From $\mathbf{G} >>_2 \mathbf{H}$ and $\mathbf{G} \neq \mathbf{H}$ it follows that there is a unit derivation

$$\mathbf{G} \Rightarrow \mathbf{H} \text{ via } rq \text{ based on } g \tag{5}$$

for some q in Q, r in R, and $g : B_1 \to G$. We want to construct a similar derivation

$$\mathbf{H}^{\#} \Rightarrow \mathbf{W} \text{ via } r^{\#}q \text{ based on } g^{\#} \tag{6}$$

such that $\mathbf{W}$ and $\mathbf{H}$ have the same normal form $\mathbf{X}$ in $(\mathsf{F}, >>_{ind})$. A closer look at such normal forms will be helpful. From $\mathbf{G} >>_3 \mathbf{H}^{\#}$ it follows that $\mathbf{H}^{\#}$ has the same (under the obvious bijection between pairs of sets) nodes and arcs as does $\mathbf{G}$. The roots, the colorings, and the source maps are the same. Targets of arcs, however, are different:

$$t_H{}^{\#}x = \textbf{if } m_G t_G x \neq I \textbf{ then } t_G x \textbf{ else } t_H{}^{\#}y, \tag{7}$$

where y is the outarc from $t_G x$ in case $m_G t_G x = I$. The rest of the proof is a somewhat delicate verification that (5) can be systematically transformed to (6) in light of (7). ■

6. GARBAGE COLLECTION

Given a string $\lambda = (\lambda_1 ... \lambda_N)$ of fixed arc colors we consider a production $p_{gar}(\lambda)$ as shown in Figure 6.1. This is not a fast production. The rule $(p_{gar}(\lambda), y)$, where y is the node colored u in Figure 6.1, may be used to delete a node with outarcs colored $\lambda_1, ..., \lambda_N$ and with no inarcs. Much as in (5.1) but with roots of expression graphs involved now, let $\mathbf{G} >>_{gar} \mathbf{H}$ iff there is a string λ of fixed arc colors, a recoloring r in R, and a node $z \neq e_G$ in G such that

$$\mathbf{G} => \mathbf{H} \text{ via } rp_{gar}(\lambda) \text{ at } z. \tag{6.1}$$

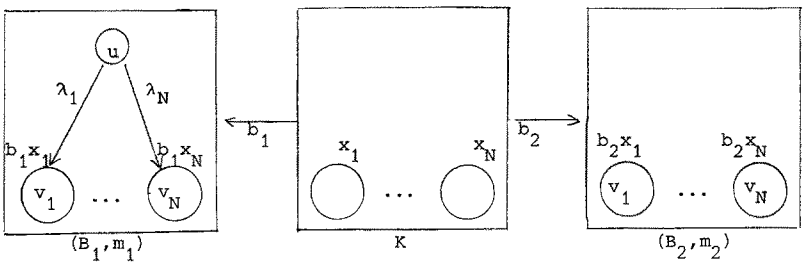

Figure 6.1. The node colored u in the production $p_{gar}(\lambda)$ is collected as garbage if it has no inarcs and has no other outarcs beside those colored $\lambda_1, ..., \lambda_N$.

LEMMA 6.2. If F is closed under $>>_{gar}$ then each member of F has a unique normal form in $(\mathsf{F}, >>_{gar})$.

Proof. If $\mathbf{G} >>_{gar} \mathbf{H}$ and $\mathbf{G} >>_{gar} \mathbf{H}^{\#}$ and $\mathbf{H} \neq \mathbf{H}^{\#}$, then [ER76, Cor. 5.4] and Lemma 3.3 provide $\mathbf{X}$ with $\mathbf{H} >>_{gar} \mathbf{X}$ and $\mathbf{H}^{\#} >>_{gar} \mathbf{X}$. This clearly implies the Church-Rosser property. Existence of normal forms follows from the fact that $>>_{gar}$ decreases the size of an expression graph. ■

It is easy to define a family F and relation $>>_1$ on F such that $(\mathsf{F}, >>_1)$ is Church-Rosser and F is closed under garbage collection, but the union system $(\mathsf{F}, >>)$ is not Church-Rosser. Premature garbage collection may destroy opportunities to add arcs between nongarbage nodes. Considering the recent interest in collecting garbage in parallel with list processing [Do77; Gr77], we seek simple conditions such that the union will inherit the Church-Rosser property from $(\mathsf{F}, >>_1)$. Let s_i and t_i be the source and target maps of B_i in a production p. Then p is *treelike* iff

$$B_2 \text{ is acyclic} \tag{6.3.1}$$

and there is a node r in K such that

$$\text{every node in } B_1 \text{ is reachable from } b_1 r; \tag{6.3.2}$$

$$m_1 b_1 x = m_2 b_2 x \text{ for all items } x \neq r \text{ in } K \tag{6.3.3}$$

and, for $1 = 1,2$ and for every arc z in B_i and node $x \neq r$ in K,

$$b_i x = s_i z \text{ implies } (\exists \text{ y arc in K})(b_i y = z \text{ and } t_K y \neq r). \tag{6.3.4}$$

All productions $p_{ind}(c)$ are treelike. So are all productions considered in [ER76, Sec. 7]. Suppose p is a treelike fast production. The nodes in $b_i(K - \{r\})$ and their outarcs in B_i generate a subgraph of B_i. By (6.3.4), all nodes in this subgraph are in $b_i(K - \{r\})$ and all arcs in this subgraph are in $b_i K$. By injectivity of b_1 and b_2, the subgraphs of B_1 and B_2 are isomorphic. By (6.3.3), the colored subgraphs of (B_i, m_i) are also isomorphic.

THEOREM 6.4. Let F be closed under $>>_{gar}$ and let $>>_1$ be a relation on expression graphs such that, for some set P of treelike fast productions,

$$\mathbf{G} >>_1 \mathbf{H} \text{ iff } (\exists \text{ p in } \mathsf{P})(\exists \text{ r in R})(\mathbf{G} \Rightarrow \mathbf{H} \text{ via rp}); \tag{1}$$

$$\mathsf{F} \text{ is closed under } >>_1. \tag{2}$$

Let $>>$ be the union of $>>_1$ and $>>_{gar}$. Then $(\mathsf{F}, >>)$ is Church-Rosser if $(\mathsf{F}, >>_1)$ is Church-Rosser.

Proof. Suppose $(\mathsf{F}, >>_1)$ is Church-Rosser, so that $(\mathsf{F}, >>)$ is a union of Church-Rosser systems by Lemma 6.2. By [Ro73, Thm. 3.5, Lemma 3.6], it will suffice to show

$$(\exists \mathbf{X})[\mathbf{H} >>_{gar}{}^* \mathbf{X} \text{ and } (\mathbf{H}^{\#} >>_1 \mathbf{X} \text{ or } \mathbf{H}^{\#} = \mathbf{X})] \tag{3}$$

under the assumption $\mathbf{G} >>_1 \mathbf{H}$ and $\mathbf{G} >>_{gar} \mathbf{H}^{\#}$. Let $\mathbf{G} \Rightarrow \mathbf{H}$ via rp in (1) and let $\mathbf{G} \Rightarrow \mathbf{H}^{\#}$ via $r^{\#}p^{\#}$ at $z^{\#}$ in (6.1), with $z^{\#} \neq e_G$. There are two cases to consider. If $z^{\#}$ is not in gB_1 then the two derivations can be shown to be independent in the sense of [ER76, Sec. 4]. Cor. 3.4 yields $\mathbf{X}$ with $\mathbf{H} >>_{gar} \mathbf{X}$ and $\mathbf{H}^{\#} >>_1 \mathbf{X}$, so (3) holds. Otherwise it can be shown that $\mathbf{H} >>_{gar}{}^* \mathbf{H}^{\#}$, so (3) holds. ■

7. LANGUAGE DESIGN SUGGESTIONS

Results like Theorem 5.5 and Theorem 6.4 can be interpreted as suggestions for the low level language programmer or for the high level language designer. Their significance is clearest in a multiprocessing context. One or more main processes under user control manipulate a record structure while service processes operate asynchronously in parallel with the main processes. The service processes follow paths of *ind* pointers or collect garbage. For the sake of definiteness, consider garbage collection and suppose that a list of backpointers is maintained for each record. Writing at low level, the user can inspect and manipulate backpointer information. Without some synchronization between the user and the garbage collector, a very unpleasant interaction can occur. The user looks at a backpointer and decides to follow it to a record RD. Then the garbage collector deletes RD. Then the user follows the invalid pointer. To prevent this interaction the user should lock out the garbage collector during some user computations, but locked out periods should be kept brief to obtain the benefits of parallelism. Theorem 6.4 suggests that garbage collection between applications of treelike fast productions can do no harm. The user could apply a production, give the garbage collector free rein while he does a long private computation not involving the record structure, and then apply another production. If other user's activities during the private computation do not cause trouble [the Church-Rosser property for $(\mathsf{F}, >>_1)$], then no combination of other user's activities and garbage collection during in the private computation can cause trouble [the Church-Rosser property for $(\mathsf{F}, >>)$]. This is a slight overstatement, in that "trouble" is an intuitive concept and does not perfectly coincide with any precise mathematical concept like lack of the Church-Rosser property. As in our backpointer example, many particular troubles do correspond to failures of the Church-Rosser property.

Of course it is difficult to correlate low level code with the definition of treelike fast productions. Rather than ask programmers to write at low level and keep Theorem 6.4 in mind, we would ask language designers to insure that the record handling facilities of high level languages have the same net effect when high level programs are compiled. Garbage collection can take place "during" execution of a single high level operation, provided that the compiled code corresponds to a sequence of applications of treelike fast productions interspersed with private computations. The garbage collector is only locked out during each application of a production. Similar but more complex recommendations can be made for user communities with several record structures, for synchronization mechanisms that can lock the garbage collector out of a region rather than the whole structure, and so on. In practice it may be necessary to give programmers more flexibility, so that some high level programs will be such that the compiler does not guarantee good interaction with the garbage collector. In that case the language facilities that put the burden of assuring good interaction on the user should be very clearly visible whenever they occur in a program. The facilities for which good interaction is guaranteed should be rich enough that only expert programmers with stringent performance goals will ever feel a need to use the dangerous facilities. This is like the

consensus that is beginning to emerge regarding control structures: **goto** should be provided, but a varied collection of more disciplined control structures should be provided to serve the needs of most programmers most of the time. The main limitation on the significance of Theorem 6.4 for language design is the simplicity of the garbage collection considered here. Unreachable cycles are not recognized as garbage, so we needed to assume that B_2 is acyclic in a treelike production. The basic theory has no such limitation, but its role in the more complex situation remains to be worked out.

REFERENCES

[Do77] T.W. Doeppner, *Parallel program correctness through refinement,* Proc. 4th ACM Symp. on Principles of Programming Languages, Santa Monica, January 1977, pp. 155-169.

[EK75] H. Ehrig & H.J. Kreowski, *Categorical theory of graph grammars,* Rept. 75-08, Tech. U. Berlin, February 1975.

[EPS73] H. Ehrig, M. Pfender, & H.J. Schneider, *Graph grammars: an algebraic approach,* Proc. 14th Ann. IEEE Symp. on Switching and Automata Theory, Iowa City, October 1973, pp. 167-180.

[ER76] H. Ehrig & B.K. Rosen, *Commutativity of independent transformations on complex objects,* IBM Research Report RC 6251, October 1976.

[Gr77] D. Gries, *On believing programs to be correct,* CACM **20** (1977), 49-50.

[HS73] H. Herrlich & G. Strecker, Category Theory, Allyn and Bacon, Rockleigh, 1973.

[Kn73] D.E. Knuth, The Art of Computer Programming, Vol. 1 (2nd ed.), Addison-Wesley, Reading, 1973.

[Ro73] B.K. Rosen, *Tree-manipulating systems and Church-Rosser theorems,* JACM **20** (1973), 160-187.

[Ro75] B.K. Rosen, *Deriving graphs from graphs by applying a production,* Acta Inf. **4** (1975), 337-357.

[Ro76] B.K. Rosen, *Correctness of parallel programs: the Church-Rosser approach,* Th. Comp. Sci. **2** (1976), 183-207.

[Se74] R. Sethi, *Testing for the Church-Rosser property,* JACM **21** (1974), 671-679.

[WH66] N. Wirth & C.A.R. Hoare, *A contribution to the development of ALGOL,* CACM **9** (1966), 413-431.