Theoretical Computer Science 10 (1980) 297–316

OPTIMAL EVALUATIONS OF GRAPH-LIKE EXPRESSIONS

John STAPLES

Department of Mathematics and Computer Science, Queensland Institute of Technology, Brisbane 4001, Australia

Communicated by Arto Salomaa
Received August 1978
Revised March 1979

Abstract. This paper is the second in a three-part study of efficient evaluations of expression. Here a general evaluation algorithm of the outermost type is given, and proved to be optimal for a wide class of systems of graph-like expressions. Several examples of the application of the algorithm are also given.

1. Introduction

1.1. In this paper we continue the study begun in [5]. Using the methods of (4, Section 5), we develop algorithms, and proofs of their optimality, for the evaluation of graph-like expressions in suitable systems. The algorithms are reasonably simple and practical, though the form in which they are presented here is chosen for ease of exposition rather than for efficiency of implementation.

A general algorithm is given and proved optimal in Section 4; particular applications of it to systems of McCarthy, Vuillemin and Pacini, and to systems of combinatory logic, are given in Section 5. Sections 2 and 3 collect the basic concepts and results which are used. Finally Section 6 discusses some limitations of the algorithm, its application to parallel processing and the scope for refining it.

1.2. Expression graphs, over a set F of function symbols with a distinguished subset $C \subseteq F$ of constants, were introduced and studied in [5]. They are finite, rooted, directed, ordered graphs, each of whose nonterminal nodes is labelled by an element of $F \backslash C$, and each of whose terminal nodes may be labelled by an element of C.

Since all graphs considered here are expression graphs, we may briefly call them 'graphs'.

1.3. In this paper we consider acyclic systems of graphs, in the following sense. A *system* $(B, \Rightarrow)$ of graphs over (F, C) is a set B of graphs over (F, C) together with a binary relation $\Rightarrow$ on B which satisfies the following conditions:

(i) Whenever $T \Rightarrow U$ there is some instance (as defined in [5]) $h : R_\rho \to T$ of some rule ρ such that $T \to U$ is the contraction defined by this instance,

(ii) Whenever $T \Rightarrow U$, $T \Rightarrow V$ are defined by instances $h : R_\rho \to T$, $k : R_\sigma \to T$ respectively, such that k induces an instance $k' : R_\sigma \to U$, and hence a contraction of U to say W, then $U \Rightarrow W$.

A system $(\mathcal{B}, \Rightarrow)$ of graphs is called *acyclic* if its graphs, and the rules which define its contractions, are acyclic. A system is called *disjoint* if each two distinct contractions of the system are defined by disjoint instances of rules, in the sense of [5].

2. Preliminaries

2.1. The definition of graph homomorphism $h : H \to G$ provided by [5, Section 2.8] permits an empty-labelled terminal node of H to have an image which has an arbitrary label; but a node n of H with a nonempty label must map to a node $h(n)$ with the same label and, if n is nonterminal, then the distinct out-edges $e_1, \ldots, e_m$ of n must have images $h(e_1), \ldots, h(e_m)$ which are all the distinct out-edges, in order, of $h(n)$. It is also required that when an edge e has end n' in H, $h(e)$ has end $h(n')$ in G.

2.2. A *subgraph* H of a graph G is a graph H together with a homomorphism which is one-one on nodes of H, and therefore on edges of H also. Note however that from Section 2.1 the image of the root of H need not be the root of G; and empty-labelled nodes of H may have images which are not empty-labelled. We shall generally discuss subgraphs as if h were the inclusion of H in G.

It will be convenient to call a node of a graph G an *assigned* node of G if it has a nonempty label.

2.3. Various sorts of subgraphs will be of interest. We define a *cofinal* subgraph H of G to be a subgraph such that empty-valued nodes of H are empty-valued nodes of G. The cofinal subgraph of a given node n of G is the subgraph H of G defined by all nodes of G below n. The cofinal subgraph of an edge e of G is defined to be the cofinal subgraph of its end.

An *initial* subgraph H of G is a subgraph such that the root of H is the root of G.

A *nontrivial* subgraph H of G is one whose root has nonempty label.

2.4. Given a contraction defined by an instance $h : R_\rho \to G$ of a rule ρ, we may call $h(r_\rho)$, the image of the root of R_ρ, the root of the contraction.

2.5. When U, V are subgraphs of a graph G, we may write $U \leqslant V$ if U is a subgraph of V. Note that if $U \leqslant V$ and U, V are not equal, then either the root of U is not the root of V, or else some empty-labelled node of U has a nonempty label in V. Note also that the inequality just introduced is reflexive, antisymmetric and transitive.

2.6. A path $m_1, e_1, \ldots, m_k, e_k, m_{k+1}$ in a graph G (where the m's denote nodes and e_i denotes an edge with start m_i and end m_{i+1}) is said to *match* a path $n_1, f_1, \ldots, n_l, f_l, n_{l+1}$ in a graph H if the following three conditions are satisfied:

(i) $k = l$,

(ii) $m_i = n_i$, $i = 1, \ldots, k$,

(iii) if m_i has label f and k out-edges, then n_i has label f and k outedges.

2.7. An *initial* path in a graph G is one whose start is the root of G. A *cofinal* path in a graph G is one whose end is a terminal node of G.

2.8. We say that a path $\pi' = (n_1, f_1, \ldots, n_l, f_l, n_{l+1})$ in a graph G *extends* a path $\pi = (m_1, e_1, \ldots, m_k, e_k, m_{k+1})$ of G, and that π is an *initial subpath* of π', if $k \leqslant l$ and $m_i = n_i$ and $e_i = f_i$, $i = 1, \ldots, k$. If $l > k$, we may call π' a proper extension of π.

2.9. We may also say that a path π in a graph G is *maximal* with respect to some property P of paths of G, if π satisfies P and there is no proper extension of π in G which satisfies P.

2.10. A node n (respectively edge e) of a graph G is said to have a *descendant* in H after a contraction $G \Rightarrow H$, if n (respectively e) is a node (respectively edge) of H; in which case we may also say that n (respectively e) is the descendant in H of n (respectively e) in G.

2.11. It is then clear how to define, by induction on the length of the sequence, what it means for a node or edge of a graph G to have a descendant in a graph H after a sequence of contractions $G \Rightarrow^* H$. We omit details.

The following two lemmas will be found useful:

Lemma 1. *If n is a node of a graph G, if the cofinal subterm W of n in G does not include the root of the contraction $G \Rightarrow H$, and if n has a descendant in H under this contraction, then the cofinal subgraph of (the descendant of) n in H is the same graph W.*

Proof. From the definition of contraction, see in particular [5, Section 2.11], it is enough to prove that the cofinal subgraph of n in H has the same nodes as the cofinal subgraph of W in G. That is elementary and omitted.

Lemma 2. *If a node n of a graph G has a descendant in a graph H after a contraction $G \Rightarrow H$ defined by an instance $h : R_\rho \to G$, and if $\pi = (n_1, e_1, \ldots, n_k, e_k, n_{k+1})$ is a path of G with start n, no intermediate node of which is the root $h(r_\rho)$ of $G \Rightarrow H$, then $n_2, \ldots, n_{k+1}$ also have descendants in U and π is a path in U.*

Proof. By induction on k. The details are straightforward and are omitted.

2.12. Notice that Lemma 2 allows us to extend Lemma 1. We define the descendant of a suitable subgraph J of a graph G, after a contraction $G \Rightarrow H$. If the contraction $G \Rightarrow H$ has root r_ρ, and if r_ρ is not an assigned node of J, then from Lemma 2 all nodes of J have descendants in H, and those with labels in H have the same labels in G. That is, the descendants in H of the nodes of J define a copy of J in H, which we may call the descendant of J in H, and denote J also.

The descendant of a suitable subgraph J of G after a sequence of contractions $G \Rightarrow^* H$ is then defined by induction on the length of the sequence. Details are omitted.

3. Node-stability and soundness

3.1. For the rest of this paper we consider, unless otherwise stated, a fixed, disjoint, acyclic system $(B, \Rightarrow)$.

3.2. Given a node n_0 of a graph G it is convenient to define a non-n_0 contraction of G to be a contraction of G whose root is not n_0.

3.3. A node n of a graph G is defined to be *n_0-stable*, for some node n_0 of G, if there is no reduction $G \Rightarrow^* H$ such that n and n_0 have descendants in H with respect to this reduction, and n is the root of a non-n_0 contraction of H.

The following three useful properties of node-stability are immediate from the definition:

Property 1. *No n_0-stable node of a graph G is the root of a non-n_0 contraction of G.*

Property 2. *If n is n_0-stable in G and both n, n_0 have descendants in H after a reduction $G \Rightarrow^* H$, then n is n_0-stable in H.*

Property 3. *n_0 is n_0-stable.*

The next observation follows easily from Property 2.

Result 1. *If a graph G has a node n which is the root of a subterm V, if all assigned nodes of V except n are n_0-stable for some fixed node n_0 of G, and if $G \Rightarrow H$ is a non-n_0 contraction whose root is not n, then V has a descendant in H and all assigned nodes of V except n are n_0-stable in H.*

Proof. From Property 2, by induction on the length of the longest path of V. The details are straightforward and are omitted.

3.4. We next define a *sound* node n_0 of a graph G to be a node of G such that there is some initial path π in G which ends at n_0 and all of whose nodes are n_0-stable.

We also define a *sound* contraction of a graph G to be a contraction whose root is a sound node of G.

Note that if π is a path as in the above definition of sound node, then the only node of π which can be the root of a contraction of G is n_0.

The next two results show the point of the definition of sound contraction.

Result 2. *If n_0 is a sound node of a graph G which is the root of a contraction of G, then after a non-n_0 contraction $G \Rightarrow H$, n_0 has a descendant in H which is a sound node of H.*

Proof. Write π for an initial path of G of n_0-stable nodes with end n_0. As noted in Section 3.4, the root of $G \Rightarrow H$ is not any nonterminal node of π, so from Lemma 2 π is an initial path of H with end n_0. From Property 2, all the nodes of π are n_0-stable in H, as required.

Result 3. *If $(B, \Rightarrow)$ is a disjoint system in which every graph G which has a normal form but is not in normal form has a sound contraction, then sound contractions of such graphs G are optimal contractions.*

That is, for every such contraction $G \Rightarrow H$, H has a reduction to normal form whose length is one less than the length of the shortest reduction to normal form of G.

Proof. In view of the subcommutativity result for disjoint systems given in [5, Section 3.3], Result 2 shows that sound contractions satisfy the hypotheses of the abstract optimality result of [4, Section 5]. In particular, say $G \Rightarrow H$ is a sound contraction and that $G \Rightarrow J$ is another contraction. Now the subcommutativity result says, amongst other things, that one of the following three alternatives holds:

(i) $H \Rightarrow J$,

(ii) $J \Rightarrow H$,

(iii) there is a graph K such that $H \Rightarrow K$, $J \Rightarrow K$.

The subcommutativity result together with Result 2 rules out (i), since the root of $G \Rightarrow H$ has a descendant in J. Also Result 2 shows that in case (ii) $J \Rightarrow H$ (respectively in case (iii) $J \Rightarrow K$) is also a sound contraction, so that the hypothesis (ii) of the abstract optimality result is satisfied with $k = 0$ (respectively with $k = 1$).

3.5. Our task now is to give, in suitable cases, conditions under which graphs not in normal form always have sound contractions, and more particularly to give conditions under which reasonable algorithms can be given for finding sound contractions. That is the purpose of the following two sections of this paper. We conclude this section by giving some useful sufficient conditions for node-stability.

The first result we shall give is a general but abstract one. To support it we also give several more specialized conditions which are immediately useful in particular circumstances. The following definitions are required.

3.6. A subgraph H of a graph G is said to be n_0-*allowable* in G if there is some non-n_0 reduction $G \Rightarrow^* G'$ and some contraction $G' \Rightarrow G''$ such that H has a descendant in G' and $G' \Rightarrow G''$ is defined by an instance $h : R_\rho \to G'$ such that the image in G' of the root of R_ρ is the root of the descendant of H.

If H is a subterm of G which is not n_0-allowable, then H is said to be n_0-*barred* in G.

Result 4. *If G is a graph with nodes n and n_0 and a subgraph J which*

(i) *J has root n,*

(ii) *all assigned nodes of J other than n are n_0-stable,*

(iii) *J is n_0-barred in G,*

then n is n_0-stable.

Proof. We show that n is not the root of a non-n_0 contraction of G, and that after a non-n_0 contraction $G \Rightarrow H$ such that n and n_0 have descendants in H, J has a descendant in H and the hypotheses are satisfied by the descendants of n, n_0 and J in H. The result then follows by induction, in view of the definition of node-stability. We consider only the nontrivial case, when $n \neq n_0$.

First, it is immediate from (i) and (iii) that n is not the root of any contraction of G.

Next, it follows from Result 1 that J has a descendant in H with root n, and that all assigned nodes of J, except perhaps n, are n_0-stable in H. It is immediate from the definition of n_0-barred that J is n_0-barred in H.

In order to apply Result 4 we need conditions under which a subgraph J of a graph G is n_0-barred in G. The following condition is suitable:

Proposition 1. *If G is a graph with node n_0, and if J is a subgraph of G which satisfies the following conditions, then J is n_0-barred in G:*

(i) *all assigned nodes of J other than its root are n_0-stable,*

(ii) *for every instance $h : R_\rho \to G$ which defines a contraction there is an initial path π_h of J which satisfies all the following conditions:*

(a) *the end of π_h is n_0-stable in G,*

(b) *if π_h matches an initial path of R_ρ, then its end is the root of a contraction of G; and so from* (a) *its end is n_0,*

(c) *Whenever an initial subpath of π_h matches any initial, cofinal path π^* of R_ρ, the end of π^* is an assigned node of R_ρ.*

Proof. From the definition of n_0-barred (Section 3.6) it is enough for a proof by induction on the length of reductions to show that the root n of J is not the root of a

non-n_0 contraction of G, and to show that after a non-n_0 contraction $G \Rightarrow H$ such that n and n_0 have descendants in H, J also has a descendant in H and the descendants of n, n_0 and J in H satisfy the hypotheses.

First we show that the root n of J is not the root of a non-n_0 contraction of G. Suppose it is; we find a contradiction as follows.

Write $h : R_\rho \to G$ for an instance of a non-n_0 contraction with root n, write π_h for a path of J as in hypothesis (ii), and consider the initial subpath π_1 of π_h which is maximal with respect to the condition that it matches an initial path of R_ρ. Note that π_1 cannot match any initial, cofinal path π^* of R_ρ, for then from (ii) (c) the end of π^* is an assigned node of R_ρ, whose value is therefore a constant, which is not the value of any intermediate address of π_h. Thus the hypothesis that π_1 matches π^* implies that $\pi_1 = \pi_h$, so that π_h matches an initial path of R_ρ, so that from (ii)(b) the end of π_h is the root of a contraction of G. That contradicts the disjointness of the system, since the end of π_h is an assigned address of $h(R_\rho)$.

Hence π_1 does not match an initial, cofinal path π^* of R_ρ; but that contradicts the assumption that there is a hom $h : R_\rho \to G$ which maps the root of R_ρ to n. Thus n is not equivalent to the root of a non-n_0 contraction of G.

As all assigned nodes of J other than n are n_0-stable, then J has a descendant in H, and from Result (1) (i) is satisfied in H. Also (ii) is satisfied when we choose in H the same paths π_h as were chosen in G, as we now show.

First, (ii) (a) is immediate from the definition of node-stability. To show (ii) (b), we show that if the end n_h of π_h is the root of a contraction of G, then it is the root of a contraction of H. Notice that n_h is n_0-stable in G, so $n_h = n_0$, so it is not the root of $G \Rightarrow H$. The result then follows from the disjointness of the system.

Finally, (ii) (c) is independent of whether π_h is a path of G or of H, since the end of π_h has the same label in both G and H.

We can combine Result 4 and Proposition 1 as follows:

Result 5. *Given a graph G with nodes n and n_0, if there is a subgraph J of G which satisfies conditions (i) and (ii) of Condition 1, then n is n_0-stable.*

The following more specialized results are more directly useful:

Result 6. *If a graph G has nodes n and n_0, and if the cofinal subterm J of n in G is in normal form, then n is n_0-stable in G.*

Proof. This result is a corollary of Result 5 and Lemma 1. It is however no easier to justify that assertion than to give an independent proof along the same lines as those above. That is, we observe that n is not the root of a non-n_0 contraction of G and that, from Lemma 1, after a non-n_0 contraction $G \Rightarrow H$ such that n and n_0 both have descendants in H, those descendants satisfy the hypotheses. The result then follows by induction.

Similar remarks suffice for the proof of the following:

Result 7. *If a node n of a graph G has label λ, where λ is a function symbol which is not the function symbol of the root of any rule redex, then n is n_0-stable for all nodes n_0 of G.*

Result 8. *If a graph G has cofinal subgraph K and if K has nodes n and n_0 such that n is n_0-stable in K, then n is n_0-stable in G.*

Proof. The method is again as for the proof of Result 4. That is, it is enough for a proof by induction to show that n is not the root of a non-n_0 contraction of G, and to show that after a non-n_0 contraction $G \Rightarrow H$ such that n and n_0 have descendants in H, these descendants also satisfy the hypotheses.

First, n is not a root of a non-n_0 contraction of G, else it would be the root of a non-n_0 contraction of K, contradiction.

Next we consider a contraction $G \Rightarrow H$ as above by means of the following two cases:

Case 1. The root of $G \Rightarrow H$ is not a node of K. Then from Lemma 1, K is a cofinal subterm in H also, and by hypothesis n is n_0-stable in K.

Case 2. The root of $G \Rightarrow H$ is a node in K. Then $G \Rightarrow H$ defines a contraction $K \Rightarrow K'$ say, where K' is a cofinal subterm of H which includes the descendants of n and n_0. Note that n is n_0-stable in K', from Property 2 applied to K.

4. An optimal evaluation algorithm

4.1. We begin with a simple example of the sort of algorithm to be studied. The system to which the example applies is the nonlinear weak combinatory logic defined in [5], and the example is a version of the familiar 'leftmost-outermost' evaluation algorithm. The general algorithm to be given later, however, considerably generalises the 'leftmost-outermost' concept.

The object of Algorithm 1 is to choose a root of a sound contraction of the graph G (not in normal form) to which it is applied. That it succeeds in doing so will be proved later, for the more general Algorithm 2. In Algorithm 1 the initial value of the variable m is chosen to be the root of G, and the final value of m is taken to be the output of the algorithm. The algorithm is given in a form which makes its properties easy to analyse, not in a form which is efficient to implement.

4.2. For nonlinear weak combinatory logic the algorithm is as follows (the notation is as in [5, Section 2.14]):

Algorithm 1

1. **if** m is not the root of a contraction of G and m has label with two out-edges e_1 and e_2
   then if the cofinal subterm of e_1 is not in normal form
   then $m :=$ the end of e_1
   else $m :=$ the end of e_2
   else stop
2. **go to** 1.

4.3. The generalization of Algorithm 1 which we now consider is, more precisely, a compromise between generality and economy. For, any system as in Section 1.2 undoubtedly has optimal evaluation algorithms of some sort. At worst, one obtains such an algorithm as follows. Examine all reductions of length zero, then all of length one, and so on until the first reduction to normal form is encountered. It is necessarily a reduction to normal form of minimal length.

It is clear then that practicability is an essential, if vaguely defined, requirement for an interesting optimal evaluation algorithm.

The idea of our generalization is to retain the basic structure of Algorithm 1, but to replace the notions of 'left out-edge' and 'right out-edge' by 'a preferred out-edge' and 'a non-preferred out-edge' respectively. Our generalized algorithm is nondeterministic in the sense that it may prescribe an arbitrary choice between certain edges. This nondeterminism is not an essential feature. It can be removed by, for example, always choosing the leftmost from the edges from which the choice is to be made. The nondeterministic specification is however helpful when considering implementation of the algorithm (for example by asynchronous parallel processing); see Section 6.2.

4.4. How then are the preferred out-edges of a node to be defined? Consideration of examples such as that of Section 5.4 shows that in general the choice of the preferred outedges of a node cannot always be the same for all nodes with out-edges, as occurred in Algorithm 1. Neither can it depend merely on the function symbol of the node. We proceed as follows.

4.5. We shall assume that certain *patterns* are available to assist our choice. Each pattern includes information which enables it to prescribe the out-edges which it prefers. We shall make a 'best possible' match of the available patterns and take as the preferred out-edges those which are prescribed by that pattern. We shall also assume that the patterns are consistent, in the sense that whenever two different patterns are both best possible matches, they prescribe the same preferred out-edges.

It will be demonstrated in the following section that this pattern-matching strategy is simple to apply in various cases of interest.

Each pattern will be a certain graph P, together with a distinguished, nonterminal node p of P, not necessarily the root of P, and also a distinguished, nonempty subset of the out-edges of p in P. Such a pattern will be applied to a node m in a graph G by a homomorphism $h : P \to G$ such that $h(p) = m$. Any such application of such a pattern P prescribes as preferred those out-edges of m which are the images under h of the preferred out-edges of p.

In the simplest cases it is enough to take as the patterns just the nontrivial, proper, initial subterms of rule redexes, and to take p to be the root of each pattern. We shall call a *preference* the collection of patterns chosen for a system, and we shall call a preference *elementary* if it is one of the simplest cases just mentioned. J.-J. Levy and G. Huet have pointed out in discussions that there are simple examples where elementary preferences are inadequate, so we allow here the more general notion of preference just indicated. See Section 6.1 for an example of a system for which elementary preferences are inadequate. Note however that we shall always assume that the preference chosen for a system includes at least all the nontrivial, proper, initial subterms of all rule redexes, with distinguished node as root, and that the preference is consistent in the sense mentioned at the beginning of Section 4.5.

4.6. To introduce the method we first treat the special case where for all P in a preference $\boldsymbol{P}$ the distinguished node p of P is the root of P. In this case, given a graph G and a node m of G, either there is at least one pattern $P \in \boldsymbol{P}$ such that there is a hom $P \to G$ which maps p to m, or not. If not, then all out-edges of m are preferred. If so, then write Q for a maximal such P; the preferred outedges of m are the homomorphic images of the distinguished out-edges of p in P. The consistency condition which is required to make sense of this definition is:

Whenever two distinct homs $Q_1 \to G$ and $Q_2 \to G$ are both maximal, they should prescribe the same out-edges of m as preferred out-edges.

Before discussing the general case we state the algorithm which we shall consider. As in Algorithm 1 the initial value of m is the root of the graph G under consideration, and the final value of m is the output of the algorithm.

Algorithm 2

1. **if** m is not the root of a contraction of G, and m is nonterminal
   then if some preferred out-edge e of m ends at a root of a cofinal subterm of G which is not in normal form
   then $m :=$ the end of any such e
   else if some out-edge e' of m ends at the root of a cofinal subterm of G not in normal form
   then $m :=$ the end of any such e'
   else stop
   else stop
2. **go to** 1.

4.7. If one is interested only in preferences $\boldsymbol{P}$ as discussed in Section 4.6, for which a definition of the preferred out-edges of nodes has already been given, one can now go direct to Section 4.8 and Proposition 2, where a condition on P is given which is sufficient for the above algorithm to be an optimal evaluation algorithm. However for preferences $\boldsymbol{P}$ in general, we still have to define the notion of preferred out-edge which is to be used in Algorithm 2.

The general definition takes into account the sequence of values, say $n_\nu, \ldots, n_0$ which have been taken by the variable m up to the present stage. Thus, n_ν is the root of G and n_0 is the current value of m, the preferred out-edges of which have to be defined.

Suppose first that there is at least one pattern $P \in \boldsymbol{P}$ such that there is a hom $P \to G$ which maps p to n_0 and the root of P to some n_i; and suppose that P is maximal in $\boldsymbol{P}$ in this respect. Then the preferred out-edges of n_0 are the homomorphic images of the distinguished out-edges in P of p, and the consistency condition as before is:

Whenever two distinct pattern homs $P \to G$ and $Q \to G$ as above are both maximal in the above respect, they should prescribe the same out-edges of n_0 as its preferred out-edges.

If there is no pattern $P \in \boldsymbol{P}$ as above, then all out-edges of n_0 are preferred out-edges.

4.8. In order to state a condition on a preference P which is sufficient (in disjoint systems) to ensure that Algorithm 2 is an optimal evaluation algorithm, we make the following definitions:

- A *generated* path in a graph G is one of those initial paths which is followed by the nondeterministic Algorithm 2.
- A *preferred* path in a graph G is a path

$$n_0, e_0, n_1, e_1, \ldots, n_{k-1}, e_{k-1}, n_k, \quad k \geq 0,$$

which is a terminal subpath of a generated path of G and is such that e_i is a preferred out-edge of n_i, $i = 0, \ldots, k-1$.

The condition is stated in Proposition 2. Since it is rather complex, it should be noted that the complexity is in the hypotheses, and that each additional clause of the hypothesis makes the hypotheses weaker. In fact in several elementary cases, as is noted in the following section, the hypotheses are never all satisfied, so that the condition is vacuously true.

Proposition 2. *Whenever a graph G has nodes n and n_0 and a preferred path π which satisfy the following conditions, then n is n_0-stable:*

(i) *n is the start of π and n is not the root of a contraction,*

(ii) *all nodes of π other than n are n_0-stable,*

(iii) *some initial subpath of π matches an initial, cofinal path of some rule redex,*

(iv) *there is a path π_0 in G with start n and end n_0 such that*

(a) *the maximal, preferred, initial subpath* π_1 *of* π_0 *is a subpath of* π,
(b) $\pi_1 \neq \pi$ *implies that the preferred out-edges in* G *of the end of* π_1 *head subterms of* G *which are in normal form,*

(v) n_0 *is the root of a contraction of* G.

4.9. We now prove that the condition of Proposition 2 is sufficient, in a disjoint system, to ensure that Algorithm 2 terminates at a node which is the root of a sound contraction, whenever it is applied to a graph G not in normal form. The proof is achieved by the following three steps.

4.10. Proof that the algorithm terminates

The successive choices of m and an out-edge of m define a path in G whose length is increased by one for each cycle of the algorithm. Since G is finite and acyclic, the lengths of paths of G are bounded, so the number of cycles of the algorithm is bounded.

4.11. Proof that the algorithm terminates at the first value of m which is the root of a contraction

The initial value of m has the property, and evidently each cycle of the algorithm preserves the property, that m is the start of at least one path of G which ends at the root of a contraction. Evidently the algorithm terminates at any value of m which is the root of a contraction of G. To see that the algorithm does not terminate for any other reason, suppose that the current value of m is not the root of a contraction, and show as follows that this current value is not the terminal value. Consider the two possible causes of termination.

Case 1: m is a terminal node of G. This case does not occur, since it would imply that there is no root of a contraction in the cofinal subterm of m.

Case 2: Every out-edge e of m has the cofinal subterm of its end in normal form. Clearly this case does not occur; there is the root of a contraction in the cofinal subterm of m, and m is not that root, so the root is in the cofinal subterm of the end of some out-edge of m.

4.12. Proof that the algorithm terminates at a sound node

Write $n_\nu, \ldots, n_0$ for the successive values taken by m during the execution of the algorithm, and write π_0^+ for the path from n_ν to n_0 likewise defined. We show by induction on i that each n_i is n_0-stable, $i = 0, \ldots, \nu$. The result then follows from the definition of soundness.

The case $i = 0$ is trivially true, so we suppose $i > 0$. Then by inductive hypothesis $n_{i-1}, \ldots, n_0$ are n_0-stable; write π_0 for the terminal subpath of π_0^+ whose start is n_i,

and write π_1 for the maximal, preferred, initial subpath of π_0. We consider the following cases:

Case 1: Some initial subpath of π_1 matches an initial, cofinal path of some rule redex. Then n_i is n_0-stable by the hypothesis of Proposition 2, taking π_1 for the π of Proposition 2.

Case 2: Otherwise. Consider the following two subcases:

Case 2.1: π_1 does not match any initial path of any rule redex. Then n_i is n_0-stable by Result 5, applied to the subterm J of G whose only assigned nodes are the nonterminal nodes of π_1.

Case 2.2: Otherwise. Consider two subcases.

Case 2.2.1: $\pi_1 = \pi_0$. As Case 1 is ruled out, then n_i is n_0-stable by Result 5, applied to the same subgraph J as in Case 2.1.

Case 2.2.2: $\pi_1 \neq \pi_0$. In this case there is a first node of π_0 which is not the end of a preferred out-edge of its predecessor. Thus every preferred out-edge of that predecessor is the root of a cofinal subgraph of G which is in normal form; so all nodes of those cofinal subterms are n_0-stable, by Result 6. Thus π_1 can be extended to a preferred path π_2 of G which is cofinal in G and all of whose nodes, other than n_i, are known to be n_0-stable. Consider two final subcases.

Case 2.2.2.1: Some initial subpath of π_2 matches an initial, cofinal path of some rule redex. Then n_i is n_0-stable by Proposition 2.

Case 2.2.2.2: Otherwise. Then π_2 does not match any initial path π_ρ of any rule redex R_ρ—otherwise, as π_2 is cofinal in G, π_ρ would be cofinal in R_ρ, contrary to the hypothesis that Case 2.2.2.1 is ruled out. Thus n_i is n_0-stable, by Result 5 applied to the subgraph J of G whose only assigned nodes are the nonterminal nodes of π_2.

5. Applications

5.1. The discussion of Rosen [3, Section 7] explains how recursive definitions generate replacement systems. Rosen actually generates subtree replacement systems; modifications which give disjoint systems of graph-like expressions in suitable cases are often evident and are systematically discussed in [6].

The question then arises: to which of the systems generated by recursive definitions do our results apply? It is easy to see that our results are at least able to deal with all of the original recursive definitions of McCarthy [1]. Those definitions comprise

(i) equations of the form

$$f(x_0, \ldots, x_k) := E, \quad (x_0, \ldots, x_k \text{ variables})$$

where E is some compound expression. Each such equation defines, from our point

of view, a replacement rule whose redex has (in term notation) the form

$$\{p_0 := f(x_0, \ldots, x_k)$$
$$x_0 := e$$
$$\vdots$$
$$x_k := e\};$$

(ii) an **if** ... **then** ... **else** ... construction which defines, from our point of view, two rules $\rho(T)$ and $\rho(F)$ say, whose redexes have the respective forms

$$\{p_0 := C(p_1, p_2, p_3) \qquad \{p_0 := C(p_1, p_2, p_3)$$
$$p_1 := T \qquad\qquad p_1 := F$$
$$\text{and}$$
$$p_2 := e \qquad\qquad p_2 := e$$
$$p_3 := e\} \qquad\qquad p_3 := e\},$$

where T, F and C are function symbols which are not used in the rule redexes indicated in (i) above.

It is evident that such systems are disjoint in the sense of [5], and they satisfy the restrictions of Section 1.2. We prescribe as follows an elementary preference for such a system, and then check that Proposition 2 is satisfied, so that Algorithm 2 gives an optimal evaluation algorithm.

Notice that redexes as in (i) above have no nontrivial proper initial subterms. For nodes corresponding to addresses p_0 as in (ii) above, we define the leftmost of its three out-edges to be preferred.

Evidently an elementary preference is thereby defined. To see that Proposition 2 is true, observe that no preferred paths of any graphs of the system match initial, cofinal paths of any rule redex.

5.2. In the remainder of this section we give three further examples of systems of graph-like expressions, for two of which the establishment of an optimal evaluation algorithm has been an unsolved problem, and we show that the method of this paper deals with all of them.

5.3. Our first example is nonlinear weak combinatory logic, as defined in [5]. We recall that the two rule redexes of this system have (in term notation) the following forms, where α denotes a binary function letter.

$$\{r_S := \alpha(p_5, p_2) \qquad \{r_K := \alpha(p_4, p_2)$$
$$p_5 := \alpha(p_6, p_3) \qquad\qquad p_4 := \alpha(p_5, p_3)$$
$$p_6 := \alpha(p_7, p_4) \qquad\qquad p_5 := K$$
$$p_7 := S \qquad \text{and} \qquad p_2 := e$$

$p_2 := e$ $\qquad$ $p_3 := e\}$.

$p_3 := e$

$p_4 := e\}$

Nonlinear weak combinatory logic is evidently disjoint. As foreshadowed in Algorithm 1, we define an elementary preference by defining the preferred out-edge of the root of each proper initial subgraph of each rule redex to be the left out-edge. Evidently an elementary preference is thereby defined.

Again Proposition 2 is vacuously true; there are no preferred paths of any graph in the system which match initial, cofinal paths of any rule redex. For, any path which matches an initial, cofinal path of any rule redex is not a preferred path, because its start is the root of a contraction.

5.4. Our next example is the nonlinear version of the extended system of weak combinatory logic which is obtained from the basic system by adding three new function symbols, R, **suc** and 0, and by adding an infinite family of new rules as follows.

In classical notation the additional rules can be expressed, writing $\bar{0}$ for 0, and $\overline{n+1}$ for (**suc** $\bar{n}$), thus:

$$RXY\bar{0} \Rightarrow X \qquad RXY\overline{n+1} \Rightarrow Y\bar{n}(RXY\bar{n}), \qquad n \geq 0,$$

where X and Y denote arbitrary classical terms.

The additional rules of the corresponding graph-like system, which we do not describe completely, have (in term notation) redexes of the form

$\{r := \alpha(p_1, p_6)$

$p_1 := \alpha(p_2, p_5)$

$p_2 := \alpha(p_3, p_4)$

$p_3 := R$

$p_4 := e$

$p_5 := e$

'$p_6 := \bar{n}$'$\}$,

where the last entry is an abbreviation for a continuation of the list whose length depends on n and whose form is as follows:

$n = 0$. The continuation is just $p_6 := 0$.

$n > 0$. The continuation is

$$\begin{array}{ll} p_6 := (p_7, p_8) & p_7 := \mathbf{suc} \\ \vdots & \vdots \\ p_{2n+4} := (p_{2n+5}, p_{2n+6}) & p_{2n+5} := \mathbf{suc}. \\ p_{2n+6} := 0 & \end{array}$$

This extended combinatory logic is evidently disjoint. A suitable elementary preference is defined as follows:

(i) if the root r of the proper initial redex subgraph in question can be extended to a path of that subgraph of the form

$$r, e_1, n_2, e_2, n_3, e_3, n_4$$

where n_4 has label R in the subgraph, then the right out-edge of the root r is the preferred out-edge,

(ii) otherwise the left out-edge of r is the preferred out-edge.

This system is a case where the condition of Proposition 2 is substantial. There are graphs with π, n and n_0 satisfying the hypotheses of Proposition 2, and we have to use the theory of node-stability to show that n is n_0-stable.

Say $\pi = (n_k, e_k, n_{k-1}, \ldots, e_1, n_0)$. As in Section 5.3 it is not the case that e_i is the leftmost out-edge of n_i, $i = 1, \ldots, k$; for that would imply either

(i) $k = 2$ and n_0 has label K—contradiction since then n_k is the root of a contraction, or

(ii) $k = 3$ and n_0 has label S—contradiction similarly, or

(iii) $k = 3$ and n_0 has value R—contradiction since then e_3 is not a preferred out-edge of n_3.

To cover the remaining possibilities we consider three cases.

Case 1: $k = 3$, e_3 and e_2 are left out-edges, e_1 is not. Then n_1 starts a left path

$$n_1, f_1, m_1, f_2, m_2, f_3, m_3$$

where m_2 has label R and f_1, f_2 and f_3 are left out-edges.

Now m_3 is n_0-stable from Result 6; m_2 and m_1 are then successively seen to be n_0-stable from Result 5. In the case of m_2, the subgraph J required for Result 5 is the subgraph whose assigned nodes are m_2 and m_3; in the case of m_1 the subgraph J is the subgraph whose assigned nodes are m_1, m_2 and m_3. Now n_1 and n_2 are given by hypothesis to be n_0-stable, so we finally see that n_3 is n_0-stable from Result 5, applied to the subterm J whose assigned nodes are n_3, n_2, n_1, m_1, m_2 and m_3.

Case 2: The argument is almost identical to that of Case 1, and is therefore omitted.

Case 3: $k = 1$, and e_1 is not a left out-edge. In this case, n_1 begins a path

$$\pi_0 = (n_1, f_1, m_1, f_2, m_2, f_3, m_3)$$

where m_3 has value R in G. It is easy, as in the previous cases, to verify successively that m_3, m_2 and m_1 are n_0-stable in G. Thus n_1 is n_0-stable in G, by Result 5 applied to the subterm J of G whose assigned nodes are the assigned nodes of the paths π_0 and π. For, π_0 is a path π_h as required by Proposition 1 (ii), when h is an instance of the S-rule or the K-rule. Also, π is a path π_h as required by Proposition 1 (ii) when h is an instance of an R-rule.

5.5. The system of Pacini et al. [2] is treated as follows. Graph-like systems corresponding to those considered by Vuillemin [8] can be considered in the same way. There is a function symbol which we denote α. There are also constants as prescribed in [2].

All rule redexes of the system have, in term notation, one of the forms

$$\begin{array}{lcl} \{r := \alpha(p_0, \ldots, p_m) & & \{r := \alpha(p_0, \ldots, p_m) \\ p_0 := f & & p_0 := P \\ p_1 := a_1 & \text{or} & p_1 := e \\ \vdots & & \vdots \\ p_m := a_m\} & & p_m := e\}, \end{array}$$

where $f, a_1, \ldots, a_m, P$ denote constants which satisfy various further conditions, of which we mention only one: no constant occurs as the value of p_0 in rule redexes of both of the above sorts.

The restrictions on the constants make it easy to see that the system is disjoint. Preferred out-edges are defined to be left out-edges, in order to define an elementary preference. To show that Proposition 2 is satisfied consider a graph G with nodes n and n_0 and a preferred path π which satisfy the hypotheses of Proposition 2; write π_0 for a path as assumed by Proposition 2. To see that n is n_0-stable we apply Result 5, taking as J the subterm of G whose assigned nodes are the assigned nodes of π and π_0. The argument is similar to the corresponding argument of Section 5.4, but is simpler. Details are therefore omitted.

6. Discussion

First we give an example of a system which can be treated by the above methods, but for which no elementary preference is sufficient. Then we discuss briefly the implications of the paper for parallel processing. Finally we note the scope for improving the practicability of the algorithm of Proposition 2.

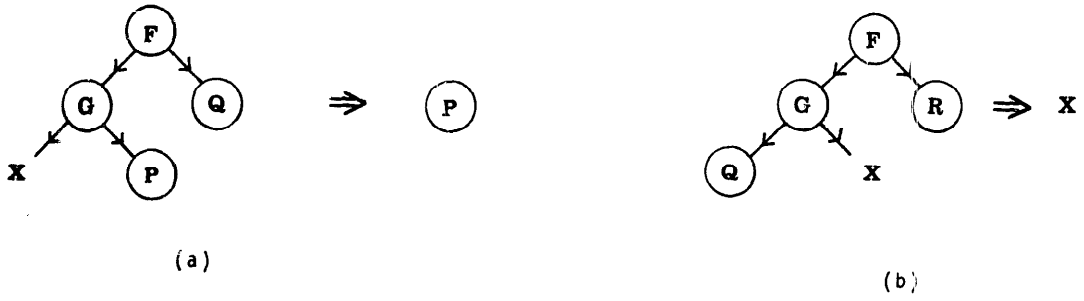

Fig. 1.

6.1. The example now to be given is adapted from an example due to G. Huet and J.-J. Levy. It concerns a system of acyclic graphs defined by the binary function symbols F and G, and the constants P, Q and R. It has two rules, as sketched in Fig. 1(a) and (b).

In this example no elementary preference can define an optimal evaluation algorithm, because of graphs such as those sketched in Fig. 2. Observe that the left out-edge of the root will be on every generated path. The question then arises: which out-edges of its end should be preferred? Using an elementary preference, the decision as to which out-edge should be preferred can take into account only the structure of the cofinal subterm of n. That is clearly inadequate to discriminate between the graphs of Fig. 2.

However a preference in the wider sense in which we have defined it is adequate to deal with this example. A suitable preference is sketched in Fig. 3, where the preferred out-edges have solid black arrowheads, and where the distinguished node p is of course the one which has the preferred out-edges. Empty-labelled nodes have been omitted. It is straightforward to check that with such a preference the condition of Proposition 2 is vacuously true.

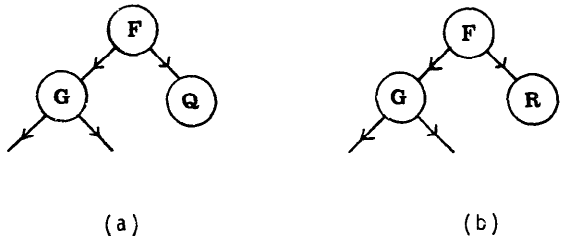

Fig. 2.

6.2. Application to parallel processing

The algorithm of Section 4.9 can be regarded as defining a set of sound contractions: one for each of the equivalence classes of paths defined by the algorithm, where equivalence of two paths means that they have the same ends and therefore define the same contraction.

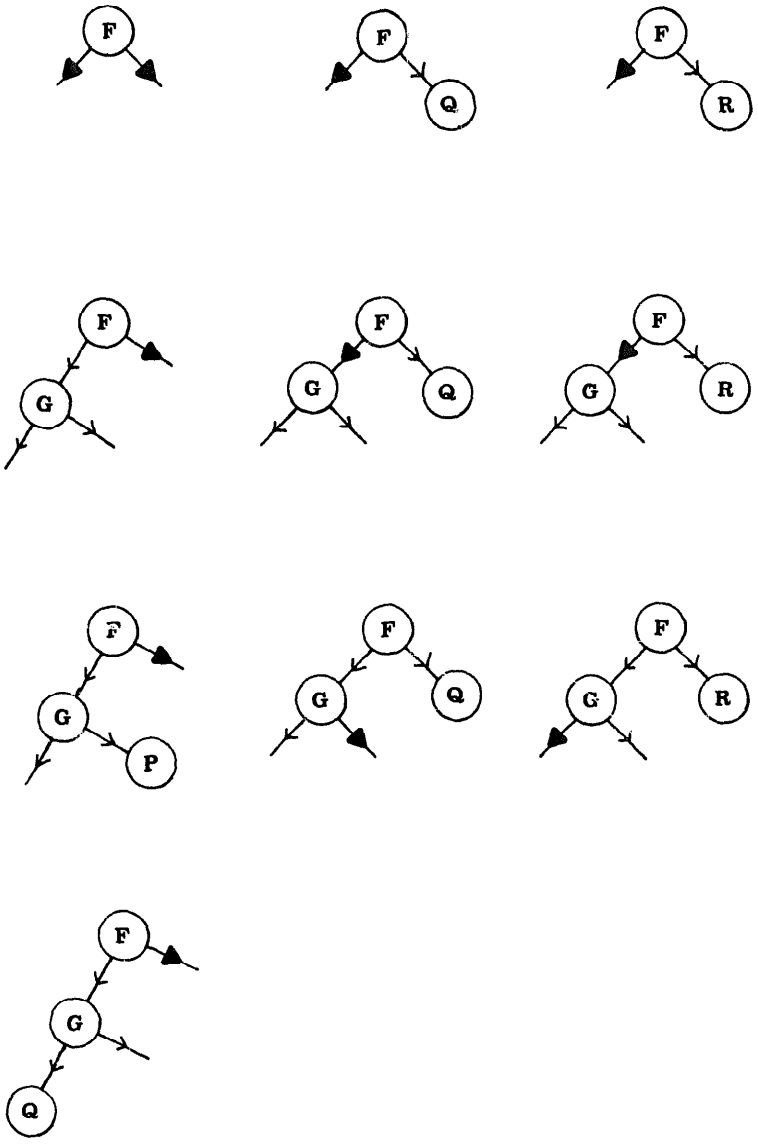

Fig. 3.

When this set of (disjoint) sound contractions comprises $n > 1$ elements, it is evident from Property 2 and [5] that a reduction of length n is defined by carrying out these contractions in arbitrary order, that the graph so obtained is independent of the order chosen and, in the case of graphs with normal form, that all contractions of the reductions are optimal.

That the reduction is independent of the order in which the steps are carried out is what is often meant by saying that the steps can be carried out by asynchronous parallel processing, since strict simultaneity is usually prohibited, explicitly or implicitly, by 'tie-breaking' conventions.

6.3. The algorithm of Proposition 2 obviously should be refined prior to any implementation. But one can say more. It is for example unnecessary in many cases

to apply the algorithm after each contraction, provided that information which has been generated by a previous application of the algorithm can be utilised. There seems to be considerable scope for developing control structures for the purpose of using information as it is generated by a modified algorithm, to speed up the search for subsequent sound contractions.

References

[1] J A. McCarthy, Basis for a mathematical theory of computation, in: P. Braffort and D. Hirschberg, Eds., *Computer Programming and Formal Systems* (North-Holland, Amsterdam, 1963) 33–70.

[2] G. Pacini, C. Montangero and F. Turini, Graph representation and computation rules for typeless recursive languages, in: J. Loeckx, Ed., *Automata, Languages and Programming*, Lecture Notes in Computer Science **14** (Springer, Berlin, 1974) 157–169.

[3] B.K. Rosen, Tree-manipulating systems and Church–Rosser theorems, *J. A.C.M.* **20** (1973) 160–187.

[4] J. Staples, A class of replacement systems with simple optimality theory, *Bull. Austral. Math. Soc.* **17** (1977) 335–350.

[5] J. Staples, Computation on graph-like expressions, *Theoret. Comput. Sci.* **10** (1980) 171–185.

[6] J. Staples, Speeding up subtree replacement systems, *Theoret. Comput. Sci.* **11** (1980) to appear.

[7] J. Staples, A graph-like lambda calculus for which leftmost-outermost reduction is optimal, in: *Proc. 1978 Graph Grammars Workshop*, Bad Honnef, F.R. Germany.

[8] J. Vuillemin, Correct and optimal implementations of recursion in a simple programming language, in: *Proc. 5th Annual A.C.M. Symposium on the Theory of Computing*, Austin, TX (1973).