USES OF C-GRAPHS IN A PROTOTYPE FOR AUTOMATIC TRANSLATION.

Marco A. CLEMENTE-SALAZAR

Centro de Graduados e Investigación,
Instituto Tecnológico de Chihuahua,
Av. Tecnológico No. 2909,
31310 Chihuahua, Chih., MEXICO.

ABSTRACT

This paper presents a prototype, not completely operational, that is intended to use c-graphs in the translation of assemblers. Firstly, the formalization of the structure and its principal notions (substructures, classes of substructures, order, etc.) are presented. Next section describes the prototype which is based on a Transformational System as well as on a rewriting system of c-graphs which constitutes the nodes of the Transformational System. The following part discusses a set of operations on the structure. Finally, the implementation in its present state is shown.

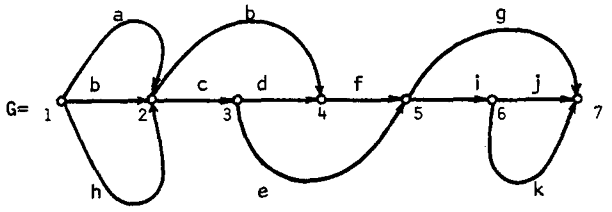
1. INTRODUCTION.

In the past [10,11], several kinds of representation have been used (strings, labelled trees, trees with "decorations", graphs of strings and (semantic) networks). C-graphs had its origin as an alternative in the representation and in the treatment of ambiguities in Automatic Translation. In earlier papers [4,5] this structure is named E-graph but c-graph is better suited since it is a generalized "grafo de cadenas" (graph of strings).

This structure combines some advantages of the Q-systems [7] and of the trees of ARIANE-78 [1,2,11], in particular, the use of only one structure for all the translation process (as in the former) and foreseeable decidability and parallelism (as in the latter). This paper presents a prototype, not completely operational, that uses c-graphs and is intended to translate assemblers to refine the adequacy of this kind of structure in the translation of natural languages.

2. DEFINITIONS

C-graph. A c-graph G is a cycle free, labelled graph [1,9] without isolated nodes and with exactly one entry node and one exit node. It is completely determined by a 7-tuple: G=(A,S,ρ,I,O,E,ε), where A is a set of arcs, S a set of nodes, ρ a mapping of A into SxS, I the input node, O the output node, E a set of labels (c-trees, c-graphs) and ε a mapping of A into E. For the sake of simplicity, arcs and labels will be merged in the representation of G (cf. Fig.1). Interesting c-graphs are sequential c-graphs (cf. Fig.2a) and bundles (cf. Fig.2b).



A={1,...,12} ; S={1,...,7} ; I={1} ; O={7}
ρ={(1,1,2),(2,2,4),(3,4,5),(4,5,7),(5,5,6),
(6,6,7),(7,6,7),(8,2,3),(9,3,4),(10,3,5),
(11,1,2),(12,1,2)}
E={a,b,c,d,e,f,g,h,i,j,k}
ε={(1,a),(2,b),(3,f),(4,g),(5,i),(6,j),
(7,k),(8,c),(9,d),(10,e),(11,b),(12,h)}

Fig.1. A c-graph.

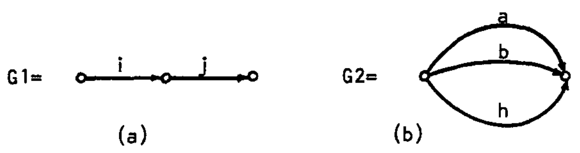


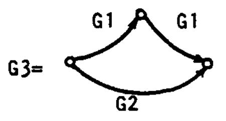
Fig.2. A seq. c-graph (a) and a bundle (b).

C-trees. A c-tree or a tree with decorations is an ordered tree, with nodes labelled by a label and a decoration that is itself a decorated tree, possibly empty.

Classes of c-graphs. There are three major classes: (1) recursive c-graphs (cf. Fig.3a) where each arc is labelled by a c-graph; (2) simple c-graphs (cf. Fig.1) where each arc is labelled by a c-tree and (3) regular c-graphs, a proper subclass of the second that is obtained by concatenation and alternation of simple arcs (cf. Fig.3b). By denoting concatenation by "." and alternation by "+", we have an evident linear representation. For example, G4=g+i.(j+k). Note that not every c-graph may be obtained by these operations, e.g.G.

Substructures. For the sake of homogeneity, the only substructures allowed are those that are themselves c-graphs. They will be called sub-

-c-graphs or seg's. For example, G1 and G2 are seg's of G.



a) A recursive c-graph.

b) A regular c-graph.

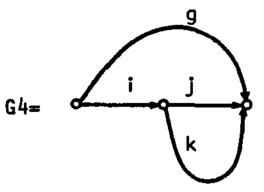


Fig.3. Two classes of c-graphs.

Isolatability. It is a feature that determines, for each c-graph G, several classes of seg's An isolated seg G' is intuitively a seg that has no arcs that "enter" or that "leave" G'. Depending on the relation that each isolated seg keeps with the rest of the c-graph, several classes of isolatability can be defined.

a) Weak isolatability. A seg G' of G is weakly isolatable (segif) if and only if for every node x of G' (except I' and O'), all of the arcs that leave or enter x are in G'. E.g.: G5=i is a segif of G.

b) Normal isolatability. A seg G' of G is normaly isolatable (segmi) if and only if it is a segif and there is a path, not in G', such that it leaves I' and enters O'. Example: G6=k is a segmi of G.

c) Strong isolatability. A seg G' of G is strongly isolatable (segfi) if and only if the only node that has entering arcs not in G' is I' and the only node that has leaving arcs not in G' is O'. When G' is not an arc and there is no segfi contained strictly in G', then G' is an "elementary segfi"; if G contains no segfi, then G is elementary. E.g. G4 is a segfi of G.

Order and roads. Two order relations are considered: (1) a "vertical" order or linear order of the arcs having the same initial node and (2) a "horizontal" order or partial order between two arcs on the same path. A road is a path from I to O Vertical order induces a linear order on roads.

3. DEFINITION OF THE PROTOTYPE.

The prototype consists of a model and a data structure. The model is essentially a generalization of a Transformational System (TS) analogous to ROBRA [2] and whose grammars are rewriting systems of c-graphs (RSC) [4,5,6]. Regarding data structure, we use c-graphs.

3.1 A Transformational System.

This TS is a c-graph→c-graph transducer. It is a "control" graph whose nodes are RSC and the arcs are labelled by conditions.

A TS is a cycle free oriented graph, with only one input and such that,

(1) Each node is labelled with a RSC or &nul.
(2) &nul has no successor.
(3) Each grammar of the RSC has a transition scheme S or ε (empty scheme).
(4) Arcs of the same initial node are ordered.

TS works heuristically. Given a c-graph g_0 as an input, it searches for the first path ending in &nul. This fact implies that all of the transition schemes on the path were satisfied. Any scheme not satisfied provokes a search of a new path. For example, if S1 is satisfied, TS produces $G1(g_0)=g_1$ and it proceeds to calculate $G2(G1(g_0))=g_2$. If S_4 is satisfied the system stops and produces g_2. Otherwise, it backtracks to G1 and tests S_2. If it is satisfied g_1 is produced. Otherwise, it tests S_3, etc.

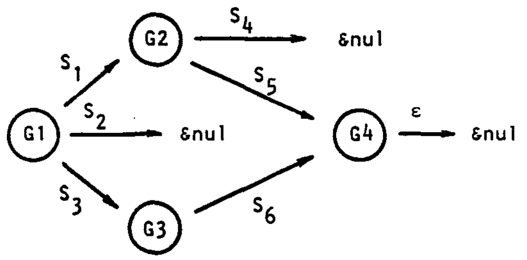


Fig.4. A Transformational System.

3.2 A REWRITING SYSTEM.

Let us consider a simple example: let GR be the following grammar for syntactic analysis (without intending an example of linguistic value).

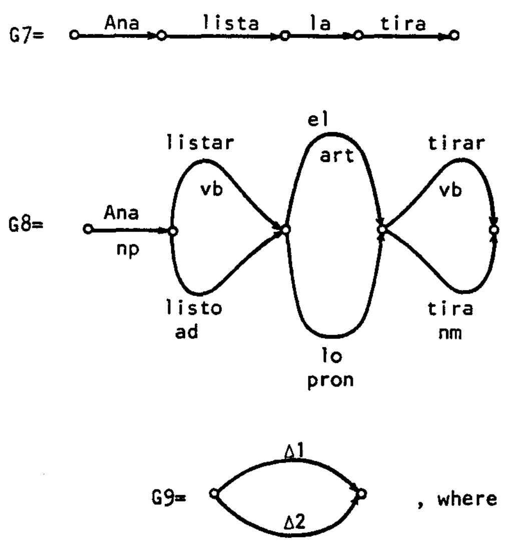
```
R1:(g1+α1+g2)(g3+α2+g4)* / α1=GN, α2=GV / ==
   (g1+g2)(g3+α2+g4)+β1  / β1:=PHRA(α1,α2) /.
R2:(g1+α1+g2)(g3+α2+g4)  / α1=VB, α2=GN / ==
   (g1+g2)(g3+α2+g4)+β1  / β1:=PRED(α1,α2) /.
R3:α1(g1+α2+g2)          / α1=NP, α2=AD / ==
   α1(g1+g2)+β1          / β1:=GN(α1,α2) /.
R4:α1(g1+α2+g2)          / α1=NP, α2=PRED / ==
   g1+g2+β1              / β1:=PHRA(α1,α2) /.
R5:(g1+α1+g2)(g3+α2+g4)  / α1=PRON, α2=VB / ==
   (g1+g2)(g3+α2+g4)+β1  / β1:=GV(α1,α2) /.
R6:(g1+α1+g2)(g3+α2+g4)  / α1=ART, α2=NM / ==
   (g1+g2)(g3+α2+g4)+β1  / β1:=GN(α1,α2) /.
```

As we can see, each rule has: a name (R1,R2, ...), a left side and a right side.

The left side defines the geometrical form and the condition that an actual seg must meet in order to be transformed. It is a c-graph scheme composed of two parts: the structural descriptor that defines the geometrical form and the condition (between slashes) that tests label information. The first part use "*" as an "element of structural description" in the first rule. It denotes the fact that no seg must be right-concatenated to $\dot{g}3+\alpha2+g4$.

The right side defines the transformation to be done. It consists of a structural descriptor, similar to the one on the left side and a list of label assignments (also between slashes) where for each new label we precise the values it takes; and for each old one, its possible modifications. A point ends the rule. Note the properties of an empty g: if g' is any c-graph, then g.g'=g and g+g'=g'.

Let us analyze the phrase: "Ana lista la tira". The representation in our formalism is G7. Morphological analysis produces G8. Note that all ambiguities are kept in the same structure in the form of parallel arcs. The application of GR to G8 results in G9, where each arc will be labelled with a c-tree with a possible interpretation of G8 in grammar GR. The sequence of applications is R3, R6, R5, R1, R2, R4. The system stops when no more rules are applicable.



, where

Δ1=PHRA(GN(NP(Ana), AD(listo)), GV(PRON(lo), VB(tirar)))

Δ2=PHRA(NP(Ana), PRED(VB(listar, GN(ART(el), NM(tira))))

Fig.5. Example of sentence analysis.

3.3 Operations.

Operations are divided in two classes: (1) those where the structure is taken as a whole (global) and (2) those that transform substructures (local).

1. Global Operations.

Concatenation and alternation have been defined above. These operations produce sequential c-graphs and bundles respectively, as well as the polynomial writing of regular c-graphs.

Expansion. This operation produces a bundle exp(G) from all the roads of a c-graph G. For example, expansion of G10 produces exp(G10)=(b.f)+(c.d.f)+(c.e).

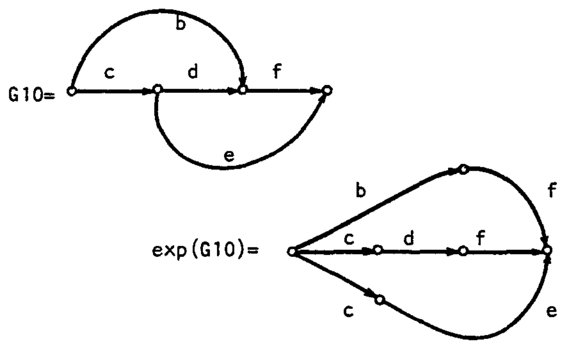


Fig.6. Expansion of a c-graph.

Factorization. There are two kinds and their results may differ. Consider G11=a.b+a.c+d.e+d.f+g.f+h.e. Left factorization produces G12=a.(b+c)+d.(e+f)+g.f+h.e, and right factorization G13=a.b+a.c+(d+h).e+(d+g).f.

Arborization. This operation constructs a c-tree from a c-graph. There may be several kinds of c-trees that can be constructed but we search for a tree that keeps vertical and horizontal orders, i.e. one that codes the structure of the c-graph. An "and-or" (y-o) tree is well suited for this purpose. The result of the operation will be a c-graph with one and only one arc labelled by the and-or tree. For example, arb(G)=G14 (cf. Fig. 7). Note that the non-regular seg has $\underline{a}$ as a root. Regular seg's have $\underline{o}$.

, where

Δ= y(o(y(a),y(b),y(h)),a(y(b,f),y(c,d,f), y(c,e)),o(g,y(i,o(j,k)))

Fig.7. Arborization of G.

2. Local Operations.

Replacement. Given two c-graphs G and G",this operation substitutes a seg G' in G for G", e.g. if G=G4, G"=m+n and G'=i, then the result will be

G15=g+(m+n).(j+k).

Addition. This operation inserts a c-graph G' into another, G, by merging two distinct nodes (x, y) of G with the input and output of G'. Addition requires only that insertion does not produce cycles. Note that if (I,0) are taken as a couple of nodes, we have alternation. Example, let (2,3) be a couple of nodes of G16 and take G'=G17=s+u. The resulting c-graph is G18.

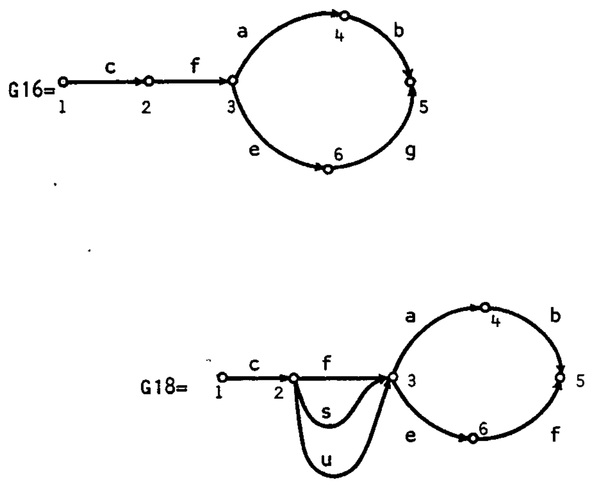


Fig.8. Addition of a c-graph.

Erasing. This eliminates a substructure G' of a c-graph G. Erasing may destroy the structure even if we work with isolated seg's. Consequently, it is only defined on particular classes of seg's, namely segfi's and segmi's. For any other substructure, we eliminate the smaller segmi that contains it. A special case constitutes a segfi G' such that I and O do not belong to G'. Eliminating G' in such a case produces two non-connecting nodes in the c-graph that we have chosen to merge to preserve homogeneity. Example: let us take G and G'= G10, then the result of erasing G10 from G is G19= G2.G4.

4. IMPLEMENTATION.

A small system has been programmed in PROLOG [4] (mainly operations) and in PASCAL (TS and RSC). For the first approach, we chose regular c-graphs to work with, since there is always a string to represent a c-graph of this class.

In its present state, the system has two parts: (1) the Transformational System including the rewriting system and (2) the set of local and global operations.

The TS is interactive. It consists of an analyzer that verifies the structure of the TS given as a console input and of the TS proper. As data we have the console input and a segment composed of transition schemes. There are no finer controls for different modes of grammar execution.

Regarding operations and from a methodological point of view, algorithms for c-graph treatment can be divided in two classes: (1) the one where we search for substructures and (2) the one where this search is not needed. Obviously, local operations belong to the first class, but among global operations, only concatenation, alternation and expansion belong to the second one. Detailed description of algorithms of this part of the system can be found in [4].

5. CONCLUSION.

Once we have an operational version of the prototype, it is intended as a first approach to proceed to the translation of assemblers of the microprocessors available in our laboratory such as INTEL's 8085 or 8080 and MOTOROLA's 6800.

6. REFERENCES.

1.[1] Boitet, Ch. UN ESSAI DE REPONSE A QUELQUES QUESTIONS THEORIQUES ET PRATIQUES LIEES A LA TRADUCTION AUTOMATIQUE. DEFINITION D'UN SYSTEME PROTOTYPE. Thèse d'Etat. Grenoble. Avril. 1976.

2.[2] Boitet, Ch. AUTOMATIC PRODUCTION OF CF AND CS ANALYSERS USING A GENERAL TREE TRANSDUCER. Rapport de recherche de l'Institut de Mathématiques Appliquées N°218. Grenoble. Novembre. 1979.

3.[4] Clemente-Salazar, M. ETUDES ET ALGORITHMES LIES A UNE NOUVELLE STRUCTURE DE DONNEES EN T.A.: LES E-GRAPHES. Thèse Dr-Ing. Grenoble. Mai. 1982.

4.[5] Clemente-Salazar, M. E-GRAPHS: AN INTERESTING DATA STRUCTURE FOR M.T. Paper presented in COLING-82. Prague. July. 1982.

5.[6] Clemente-Salazar, M. C-GRAPHS: A DATA STRUCTURE FOR AUTOMATED TRANSLATION. Paper presented in the 26th International Midwest Symposium on Circuits and Systems. Puebla. Mexico. August. 1983.

6.[7] Colmerauer, A. LES SYSTEMES-Q. Université de Montréal.Publication Interne N°43. Septembre. 1970.

7.[9] Kuntzmann, J. THEORIE DES RESEAUX (GRAPHES). Dunod. Paris. 1972.

8.[10] Vauquois, B. LA TRADUCTION AUTOMATIQUE A GRENOBLE. Document de Linguistique Quantitative N°24. Dunod. Paris. 1975.

9.[11] Vauquois, B. ASPECTS OF MECHANICAL TRANSLATION IN 1979. Conference for Japan IBM Scientific Program. Document du Groupe d'Etudes pour la Traduction Automatique. Grenoble. July. 1979.

****