Zeitschrift: L'Enseignement Mathématique

Herausgeber: Commission Internationale de l'Enseignement Mathématique

Band: 26 (1980)

Heft: 1-2: L'ENSEIGNEMENT MATHÉMATIQUE

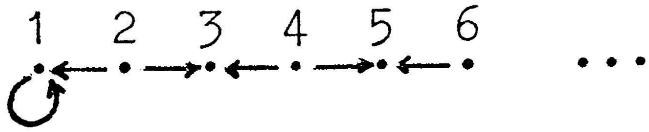
Artikel: THE FAST SKEW-CLOSURE ALGORITHM

Autor: Fischer, M. J. / Paterson, M. S.

Kapitel: 4. Generalized algorithm for power-group algebras

DOI: https://doi.org/10.5169/seals-51079

$Z_{XXXX \vee XXXX} \notin M_{\{J,K\}}$ and so $M_{\{J,K\}}$ is infinite, since for the infinite graph shown below:



$(JK)^m$ adds all edges $< i, j >$ with $i, j \leqslant 2m$

and $(JK)^m J$ adds all edges $< i, j >$ with $i, j \leqslant 2m + 1$

Therefore J, JK, JKJ, ... are all distinct.

4. Generalized algorithm for power-group algebras

To elucidate the correctness of the algorithm and to encompass some more general applications we need to generalize from the $\{0, 1\}$ Boolean algebra to a slightly richer structure. The *power-group algebra* $P(G)$ is a structure defined from an arbitrary group G. The elements of $P(G)$ are the subsets of G; the operations we require are *union* ($\cup$), complex *product*:

$$ab = \{ gh \mid g \in a,\ h \in b \} \quad \text{for} \quad a, b \subseteq G$$

and *converse*:

$$\bar{a} = \{ g^{-1} \mid g \in a \}$$

$P(G)$ is a monoid with respect to product with identity $\lambda = \{\text{identity}_G\}$. As before we shall be considering matrices over the structure, with matrix product and union defined in the obvious way from product and union in $P(G)$, and matrix *converse* defined by

$$(\bar{A})_{ij} = \overline{A_{ji}}$$

The key properties of power-group algebras which are needed are given below

Lemma. *Let* a, b *be elements and* A, B *matrices*

(i) $\bar{\bar{a}} = a$; $\bar{\bar{A}} = A$

(ii) $\overline{ab} = \bar{b}\bar{a}$; $\overline{AB} = \bar{B}\bar{A}$

(*iii*) *if* $a \neq \varnothing$ *then* $a\bar{a} \supseteq \lambda$; $A\bar{A}A \supseteq A$

Proof. We prove only (iii). The first part is immediate and has the consequence that $a\bar{a}a \supseteq a$ for *all* a. For the second part

$$(A\bar{A}A)_{ij} \supseteq A_{ij}\,\bar{A}_{ji}\,A_{ij} = A_{ij}\,\overline{A_{ij}}\,A_{ij} \supseteq A_{ij}\,. \qquad \square$$

We observe that the $\{0, 1\}$ Boolean algebra is the power-group algebra corresppnding to the trivial one-element group. Other groups we shall use are $< \mathbf{Z}_k, + >$ and $< \mathbf{R}, + >$, the integers modulo k and the reals.

The operators $*$, $+$, V and W are defined just as before for matrices and elements. In the Boolean case we had the trivial results

$$\bar{a} = a^{+} = a^{W} = a$$

and

$$a^{*} = a^{V} = 1$$

In the general case we must augment the algorithm a little. Suppose for example there are edges labelled a, b from i to j and k respectively, and a self-loop at i labelled c. Then the label of the edge from j to k must eventually receive a term corresponding to the indirect paths from j to k i.e.

$$\bar{a}(a\bar{a} \vee c \vee \bar{c} \vee b\bar{b})^{*}\, b$$

The generalized form of ψ_i is:

$A_{i*} := A_{ii}{}^{V} A_{i*}$

$k := i + 1$ step 1 until $(k = n$ or $A_{ik} \neq \varnothing)$

if $A_{ik} \neq \varnothing$ then $A_{k*} := A_{k*} \cup \overline{A_{ik}}\, A_{i*}$

The programs for ψ_i', Ψ and Ψ' are analogous. They simplify to the programs of section 2 in the Boolean case.

The question of generalization could have been tackled axiomatically. Suppose $i < j < k$ and A_{ij}, $A_{ik} \neq \varnothing$. If row i were sent directly to row k we would have $\overline{A_{ik}}\, A_{i*}$, whereas via j we get

$$\overline{\overline{A_{ij}}\, A_{ik}}\, .\, \overline{A_{ij}}\, A_{i*} = \overline{A_{ik}}\, A_{ij}\, \overline{A_{ij}}\, A_{i*}$$

and we seem to require $A_{ij}\,\overline{A_{ij}} \geqslant \lambda$ for correctness. A power-group algebra seems the only structure of possible interest with this property.

Suppose that A is a matrix over $P(G)$ and we compute $A^{RQ} = A^{V}$. $A^{V}{}_{ij}$ is the set of elements of G which are the product of labels from a (weak) path from i to j in the graph corresponding to A. Each diagonal element

$A^V{}_{ii}$ is a subgroup of G and two diagonal elements in the same (weak) component are conjugate since we may show:

$$A^V{}_{ii} \cdot g = g \,.\, A^V{}_{jj} = A^V{}_{ij} \quad \text{for any} \quad g \in A^V{}_{ij}$$

In particular if all circuits on non-empty edges correspond to the group identity then in A^V each entry has at most one element. For example given a matrix A over $< \mathbf{R}, + >$ we determine from A^V for each component of the graph whether there are (weak) circuits with non-zero sums. If all circuits sum to zero then the graph is a "potential" graph, i.e. there is a function pot : $\mathbf{R} \rightarrow \{\text{vertices}\}$ such that each edge $< u, v >$ has the value (pot (v) $-$ pot (u)). Similarly over $< \mathbf{Z}_k, + >$ we may determine whether each weak circuit of a directed graph has zero sum where forward and backward edges are accounted $+1$ and -1 respectively. Naturally we may find it convenient in some cases to hold only a homomorphic image of $P(G)$ for the computations e.g.

$$\begin{aligned} h(\varnothing) &= \varnothing \\ h(\{g\}) &= g \\ h(a) &= \omega \quad \text{when} \quad |a| > 1 \,. \end{aligned}$$

5. Proofs of correctness

An operator ϕ is *monotonic* if $A \subseteq B$ implies $A^\phi \subseteq B^\phi$. ψ_i is *not* a monotonic operator but rather surprisingly Ψ *is*. To simplify our proofs we introduce several monotonic operators. Define ϕ_i by the program

$$A_{i*} := A_{ii}{}^V A_{i*}$$

$$\text{for } k := i + 1 \text{ step } 1 \text{ until } n \text{ do } A_{k*} := A_{k*} \cup \overline{A_{ik}}\, A_{i*}$$

and ϕ_i' analogously using "$i - 1$ step -1 until 1". Both are obviously monotonic. Φ and Φ' are defined from ϕ_i and ϕ_i' in a similar way to Ψ and Ψ'. Although $\Psi \subseteq \Phi$ is evident, the following result is not.

THEOREM 5. $\Psi = \Phi$ *(and* $\Psi' = \Phi'$*).*

Proof. Consider ψ_i applied to an arbitrary matrix A and suppose it selects the index k with $A_{ik} \neq \varnothing$. (If no index is selected then of course $A^{\psi_i} = A^{\phi_i}$). We verify that: