On the computation of union-extensions of finite semigroups

By R. Bröck and H. Jürgensen

In his dissertation of 1968 [3] Verbeek proposed a generalization of the theory of semigroup extensions, which until that date consisted of the two nearly disjoint parts of Schreier- and ideal-extensions. According to Verbeek we define a semigroup extension as follows:

Definition 1. Let A, S, E be semigroups and δ a congruence on E. The pair (E, δ) is a semigroup extension of A by S, iff $E/\delta \cong S$ and there is a subsemigroup A' of E, isomorphic to A, which is a δ-class.

In the rest of this paper we shall often say that some semigroup E is an extension of A by S in the sense that there is a congruence δ, such that (E, δ) is a semigroup extension of A by S.

Schreier- and ideal-extensions are semigroup extensions according to this definition. Verbeek proved that there is an extension of A by S, iff S contains an idempotent element. Thus for finite S there is always an extension of arbitrary A by S. The idempotent concerned is the image of A' in S and is called the extension idempotent.

For ideal-extensions the homomorphism δ_{nat} induced by δ is a very special one: it is a bijection of $E \setminus A'$. Generalization of this idea led Verbeek to the concept of union-extensions:

Definition 2. Let A and S be semigroups, (E, δ) a semigroup extension of A by S. (E, δ) is a union-extension of A by S, iff the restriction of δ to $E \setminus A'$ is the identity relation, where A' is as in definition 1.

As for ideal-extensions for finite A and S the set of all union-extensions (up to isomorphism) may be obtained in a rather simple way.

Theorem 1. (Verbeek). Let A, S be disjoint semigroups, $i \in S$ an idempotent element. For $E = A \cup S^-$, where $S^- = S \setminus \{i\}$, define an associative multiplication $*$ such that the following conditions hold for all $a, b \in A$, $s, t \in S^-$

$$a * b = ab, \tag{1}$$

$$a * s \begin{cases} = is & \text{if } is \neq i, \\ \in A & \text{if } is = i, \end{cases} \tag{2}$$

$$s*a\begin{cases}=si & \text{if } si\neq i,\\ \in A & \text{if } si=i,\end{cases} \tag{3}$$

$$s*t\begin{cases}=st & \text{if } st\neq i,\\ \in A & \text{if } st=i.\end{cases} \tag{4}$$

Then $((E, *), \delta)$ is a union-extension of A by S for

$$\delta = A\times A \cup \{(x, x) | x\in S^-\}.$$

Moreover, any union-extension (E', δ') of A by S is isomorphic to one constructed in this way, where i is the extension idempotent.

Theorem 1 indicates a combinatorial method of computing the set of all union-extensions of A by S (disjoint) with extension idempotent i as follows. For A and S both finite, given by their Cayley-tables T^A and T^S, consider column c_i and row r_i of i in S; the entry t^S_{ii} belonging to ii will be replaced by A; the rest of c_i and r_i will be copied $|A|$ times to obtain a full table again; then, wherever it appears, i will be replaced by a cross indicating that the corresponding position is unknown; call the resulting partial table $T^{A,S}$.

Example

T^A	a	b
a	a	b
b	b	b

T^S	s	t	i	u	v
s	t	i	s	s	s
t	i	t	t	t	t
i	s	t	i	i	i
u	s	t	i	u	i
v	s	t	i	i	v

$T^{A,S}$	s	t	a	b	u	v
s	t	+	s	s	s	s
t	+	t	t	t	t	t
a	s	t	a	b	+	+
b	s	t	b	b	+	+
u	s	t	+	+	u	+
v	s	t	+	+	+	v

One obtains all union-extensions of A by S with extension idempotent i by replacing the crosses in $T^{A,S}$ by elements of A in all possible ways, such that the resulting table will be associative. Of course, this purely combinatorial method would soon lead to enormous computing time.

A solution to this problem is indicated by Verbeek's discussion of the composition of S with respect to i and by his theorems on the existence of union-extensions of A by S, when S has some special composition. The set of all possible compositions of semigroups has been described in parts by Verbeek [3, 4] and fully by van Leeuwen; unfortunately, he published his results in an abstract [2] only up to now.

We took a quite different and a rather naive way for computing the set of all union-extensions of A by S with extension idempotent i; all the same the computing time needed is very well below the time for the purely combinatorial method, at least when the number of extensions is small compared to the number of tables to be checked.

For $x, y\in A\cup S^-$ let $x*y$ be undefined in $T^{A,S}$. This entry of $T^{A,S}$ is considered as an unknown $u_{x,y}$ over A. Then by associativity one has a set G of equations over $A\cup S^-$ with unknowns $u_{x,y}$ over A such that exactly the solutions of G are the allowable ways of replacing the crosses in $T^{A,S}$. We classify the equations according to their forms as follows:

$$\begin{array}{lll} G_1 = \{x*u_{y,z} = u_{x,y}*z\} & x,z\in A, & G_6 = \{u_{x,u_{y,z}} = u_{u_{x,y},z}\},\\ G_2 = \{x*u_{y,z} = u_{xy,z}\} & x\in A, & G_7 = \{u_{x,u_{y,z}} = u_{xy,z}\},\\ G_3 = \{u_{x,yz} = u_{x,y}*z\} & z\in A, & G_8 = \{u_{x,yz} = u_{u_{x,y},z}\},\\ G_4 = \{x*u_{y,z} = u_{u_{x,y},z}\} & x\in A, & G_9 = \{u_{x,yz} = u_{xy,z}\}.\\ G_5 = \{u_{x,u_{y,z}} = u_{x,y}*z\} & z\in A, & \end{array}$$

It is the aim of the following method for solving G to successively narrow the domains of the unknowns and thus to avoid unnecessary trials.

We denote the domain of the unknown u by $D(u)$. In the computer programme the set of the $D(u)$ is realized by an $n\times|A|$-integer-array DOM, where n is the number of unknowns, such that

$$DOM_{u,a} = \begin{cases} 0 & \text{if } a\notin D(u),\\ 1 & \text{if } a\in D(u).\end{cases}$$

To enable an easy test, whether G has been solved, we put $|D(u)| = \sum_{a\in A} DOM_{u,a}$ in another array, which of course will be changed whenever DOM is changed. In the beginning all the $D(u)$ are A, i.e. $DOM_{u,a}=1$ for all u and all $a\in A$.

Step 1 consists of evaluating each of the equations in $K_1=G_1\cup G_2\cup G_3$. An equation $x*u_{y,z}=u_{x,y}*z$ in G_1 leads to $xD(u_{y,z})=D(u_{x,y})z$, which, however, will not be valid in most cases. Clearly there is a solution $u_{y,z}=w_1\in D(u_{y,z})$, $u_{x,y}= =w_2\in D(u_{x,y})$ to the equation only, if

$$xw_1\in D=D(u_{x,y})z\cap xD(u_{y,z})\ni w_2z.$$

Hence we can cancel all those $w_1\in D(u_{y,z})$ $\big(w_2\in D(u_{x,y})\big)$ in DOM, for which $xw_1\notin D$ $(w_2z\notin D)$ and thus narrow the domains $D(u_{x,y})$ and $D(u_{y,z})$. Furthermore, all equations from G_9 in which $u_{x,y}$ $(u_{y,z})$ appears lead to restrictions; let $u_{x,y}=u$ be such an equation; then $D(u)$ will be narrowed to $D(u_{x,y})$. For the equations in G_2 or G_3 one proceeds analogously. Some special cases arise when x and (or) z are (one-sided) identity- or zero-elements of A; they may result in transferring the corresponding equation to another type G_j (e.g. to G_9 if $x=z$ is the identity-element of A).

Since a change of $D(u)$ for an unknown u might lead to consequences from equations which have already been evaluated, step 1 is repeated until there is no $D(u)$ that can be narrowed any more.

Performing step 1 might result in one of the following three situations; otherwise we continue with step 2.

(1) For each u, $|D(u)|=1$. Then DOM represents the only solution of G.

(2) For some u, $|D(u)|=0$. Then G has no solution.

(3) For some u, $|D(u)|=1$. Wherever u appears in equation $e\in K_2=G_4\cup G_5\cup \cup G_6\cup G_7\cup G_8$ as a subscript of an unknown, it is replaced by its unique value. As a consequence in most cases e must be transferred to another class G_j. If by this procedure K_1 or G_9 is extended, e is evaluated and if this results in a restriction of some $D(u)$ execution of step 1 is resumed; otherwise step 2 is started.

In *step* 2 combinatorics comes in. G, DOM and all other information relevant to the situation are saved. Then for one unknown u we assume $u=a$ for arbitrary $a \in D(u)$, i.e. restrict $D(u)$ to be $\{a\}$ in DOM, and try to solve G applying step 1 again. With G, DOM etc. restored this is repeated until $D(u)$ is exhausted. Evidently in this way we compute exactly the set of solutions of G.

Some care has to be taken with the choice of u in step 2. It is chosen in such a way that changing $D(u)$ is likely to induce changes of the domains of as many other unknowns as possible; hence, with priority as stated, the following criteria are applied:

(1) The number of unknowns u is equal to by equations in G_9 (using transitivity, too) is maximal.

(2) The number of equations in K_2, in which u appears as a subscript, is maximal.

(3) $|D(u)|$ is maximal.

The algorithm has been realized as an ALGOL 60 programme [1] and is run on an ELECTROLOGICA X8 computer (cycle time 2.5 μsec).

Whereas it is evident that for the combinatorial method the time is $\geqq O(n^{|A|})$, where n is the number of unknowns, it seems to be impossible to give a rather correct estimate for our method; it is bad, of course, when the number of union extensions is approximately $n^{|A|}$; but in this case any method should be bad. The

Table 1

Example No.	1	2	3	4	5	6	7	8
|A|	3	3	4	4	5	5	6	6
|S|	2	5	2	5	2	5	2	5
with ideal-extensions	yes	no	yes	no	yes	no	yes	no
unknowns	6	16	8	20	10	24	12	28
combinations	729	$>4\cdot 10^7$	65 536	$>10^{13}$	$>9\cdot 10^6$	$>5\cdot 10^{16}$	$>2\cdot 10^9$	$>6\cdot 10^{21}$
union-extensions	26	163	4	15	8	3	16	0
our time	20 s	5.5 m	11 s	80 s	25 s	17 s	67 s	11 s
time for combinatorial method	14 s	$\approx$140 h	$\approx$10 m	$\approx$870 years	$\approx$30 h	$\approx 5\cdot 10^7$ years	$\approx$300 days	$\approx 3\cdot 10^{12}$ years

following table 1 allows a comparison of actual computing times; of course the figures in the last line can be considered just as hints to the approximate size, since they were calculated from the state of the pogramme after a short run only. The corresponding semigroups are listed in table 2.

Table 2

Example No.	1	2	3	4	5	6	7	8
Semigroups	A_1+S_1	A_1+S_2	A_2+S_1	A_2+S_2	A_3+S_1	A_3+S_2	A_4+S_1	A_4+S_2

Multiplication tables

S_1	*ab*
a	*aa*
b	*ab*

S_2	*abcde*
a	*abcaa*
b	*bcabb*
c	*cabcc*
d	*abcda*
e	*abcae*

A_1	*xyz*
x	*xxx*
y	*xxx*
z	*xxx*

A_2	*wxyz*
w	*wwww*
x	*wxww*
y	*wwyz*
z	*wwzy*

A_3	*vwxyz*
v	*vvvvv*
w	*vvvww*
x	*xxxxx*
y	*vvvyy*
z	*vvxyz*

A_4	*uvwxyz*
u	*uuwwyz*
v	*uuwwyz*
w	*wwuuzy*
x	*wwuuzy*
y	*yyzzwu*
z	*zzyyuw*

The element a is the extension idempotent.

О получении с помощью вычислительной машины обьединённого расширения конечных полугрупп

В 1968. году Verbeek дал определение для понятия обьединённого расширения полугрупп —, как обобщение этого понятия для идеальных расширений.

Как и для идеального расширения, мы имеем простой алгоритм для получения на вычислительной машине семейства обьединённого расширений двух конечных полугрупп, но этот алгоритм требует большого количества машинного времени. Эта статья описывает один такой алгоритм, который в общем требует значительно меньшего времени, он реализован как программа на языке ALGOL—60.

INSTITUT FÜR INFORMATIK
UND PRAKTISCHE MATHEMATIK
DER UNIVERSITÄT KIEL

MATHEMATISCHES SEMINAR
DER UNIVERSITÄT KIEL
D—23 KIEL 1, GERMANY

References

[1] Bröck, R., *Ein Programm zur Berechnung aller Vereinigungserweiterungen zweier Semigruppen*, Diplomarbeit, Kiel 1972.
[2] van Leeuwen, J., On compositions of semigroups, *Notices Amer. Math. Soc.*, v. 71T—A56, 1971, p. 405.
[3] Verbeek, L. A. M., *Semigroup extensions*, Proefschrift, Delft, 1968.
[4] Verbeek, L. A. M., Union extensions of semigroups, *Trans. Amer. Math. Soc.*, v. 150, 1970, pp. 409—423.

(Received December 15, 1972)