AG: A Set of Maple Packages for Symbolic Computing of Automata and Semigroups

Pascal Caron

Laboratoire d'Informatique de Rouen
Université de Rouen
76821 Mont-Saint-Aignan, Cedex
email : caron@dir.univ-rouen.fr

1 Introduction

AG is a set of packages for manipulating finite state automata and finite semigroups. This software includes on the one hand the AUTOMAP package which affords the usual operations on automata (union, concatenation, Kleene closure, ...) as well as the usual transformations (trimming, minimization, ...) and on the other hand the GREEN package which yields the (regular or not) $\mathcal{D}$-classes structure of a finite monoid. The AG software has been implemented using the Maple symbolic computation system. It is the first software manipulating automata and semigroups and developed inside a symbolic computation system. As we show below in a session example, this feature enables users to implement their own procedures in order to test conjectures. So AG should be of a particular interest in the numerous domains of computer science where automata have applications, such as protocol verification or modelisation of distributed systems,...

2 Functionalities

2.1 The AUTOMAP package

Representation of an automaton
We have implemented automata as Maple lists of three data structures. The first datum is an integer coding the initial state, the second datum is a set of integers representing the final states and the third datum is a Maple table corresponding to the states transition table.
Operations on automata
The AUTOMAP package supplies a set of operations on automata. These operations can be classified into two categories. The first one contains operations on languages (concatenation, union, quotient, ...), the second one contains operations on automata that have no effect on the language they represent (determinization, minimization, ...).

Here is a fairly complete table of operations used in the AUTOMAP package.

operator	Maple function	operation	Maple function	operation
&U	`Union`	union	`aut`	automaton of a letter
&C	`Concat`	concatenation	`deter`	determinization
&E	`Etoile`	Kleene closure	`mini`	minimization
&P	`Puiss`	power	`reduc`	trimming
&W	`Melange`	shuffle product	`dform`	deterministic format
&I	`Inter`	intersection	`Comp`	complement
&G	`QuotientG`	left quotient	`Miroir`	mirror
&D	`QuotientD`	right quotient		
&M	`Moins`	difference		

2.2 The GREEN package

Representation of mappings
Mappings are represented by Maple lists. For example the list $[3,2,1,5,3]$ represents the $[1,5] \to [1,5]$ mapping that maps 1 to 3, 2 to 2, 3 to 1 ... A set of generators is a set of mappings from which all the elements of the semigroup are generated by successive compositions. Several words correspond to the same mapping. The shortest word in the hierarchic order will be called the associated word of a mapping. If a mapping is a generator the associated word has only one letter.
Operations on semigroups
The GREEN package supplies a toolkit for both computing the $\mathcal{D}$-classes of a semigroup and for manipulating mappings.

Here is a list of functions of the GREEN package.

`_X`	from word to mapping
`ker`	kernel of a mapping
`mul`	composition of two mappings
`genere_Sgroupe`	generation of the semigroup
`structure`	splitting in $\mathcal{D}$-classes

3 A session example through AG

Here is an illustration of AG being used to test conjectures. Let us consider the "star removal" conjecture [2] that can be expressed as follows:

F is a star expression if and only if $F^* = F$. G is a maximal left star subexpression of E if G is a star expression such that $E = G \cdot K$ and if it does not exist H, a left subexpression of E, such that $H = (G \cdot G')^*$. Any expression E can be factorized into a finite number of maximal left star subexpressions and in

a tail C such that its maximal left star subexpression is the empty word.

According to this conjecture, E can be decomposed in the following way: $E = B_1 \cdot B_2 \dots B_n \cdot C$, where B_i is a star expression and C is a tail.

Example: $E = a\cdot(a+b)^*$ can be written $(a\cdot(a+b)^* + \varepsilon)\cdot a\cdot b^*$. In this example $B_1 = a\cdot(a+b)^* + \varepsilon$ and $C = a\cdot b^*$.

Let $\mathcal{A} = (\Sigma, Q, i, F, T)$ be the automaton that recognizes the language denoted by the regular expression E. Let $\mathcal{B} = (\Sigma, Q, i, F', T)$, where $F' = \{q \mid q\cdot L(\mathcal{A}) \subset F\}$. $\mathcal{B}$ is the automaton which recognizes B_1. So we have:

$$L(\mathcal{B}) = \{u \mid \forall v \in L(\mathcal{A}),\ u\cdot v \in L(\mathcal{A})\}$$

Let us remark that:

$$\begin{aligned} L(\mathcal{A}\cdot\mathcal{A}^{-1}) &= \{u \mid \exists v \in L(\mathcal{A}),\ u\cdot v \in L(\mathcal{A})\}, \\ L(\mathcal{A}^c\cdot\mathcal{A}^{-1}) &= \{u \mid \exists v \in L(\mathcal{A}),\ u\cdot v \in L(\mathcal{A}^c)\} \\ L((\mathcal{A}^c\cdot\mathcal{A}^{-1})^c) &= \{u \mid \forall v \in L(\mathcal{A}),\ u\cdot v \in L(\mathcal{A})\} \end{aligned}$$

where $\mathcal{A}^c$ denotes the complement of $\mathcal{A}$ with respect to Σ^*. So we have $L(\mathcal{B}) = L((\mathcal{A}^c\cdot\mathcal{A}^{-1})^c)$, and we can compute B_1 using functions `Comp` and `QuotientD` (`&D` operator). Then we compute K such that $E = B_1\cdot K$ and we iterate the process on K. K is computed in the following way: $K = E \setminus ((B_1 \setminus \varepsilon)\cdot E)$.

Let us test the conjecture on the language $L = ((ab)^*a)\sqcup\!\sqcup(ba^*b)\sqcup\!\sqcup(ab^*)$.

```
> E:=(&E (A &C B) &C A) &W (B &C &E A &C B) &W (A &C &E B):
> B1:=Comp({a,b},Comp({a,b},E) &D E):
> B1:=mini(deter(B1)):
> dform(B1);
```

```
                    [1, {1}, table([
                    b = [3, 1, 3],
                    a = [2, 3, 3]
                 ])]
```

```
> K:=E &M ((B1 &M _1A) &C E):
> K:=mini(deter(K)):
```

So we have: $E = B_1\cdot K$, where B_1 is a maximal left star subexpression. Let us remark that $B_1 = (a\cdot b)^*$. We now decompose K.

```
> B2:=Comp({a,b},Comp({a,b},C) &D C):
> B2:=mini(deter(B2)):
> dform(B2);
```

$$[1, \{1,4\}, \mathtt{table}([\quad b = [2,7,5,2,7,7,7],\quad a = [3,4,7,4,6,2,7] \quad])]$$

```
> K1:=K &M ((B2 &M _1A) &C K):
> K1:=mini(deter(K1)):
```

So we have $K = B_2 \cdot K_1$ where B_2 is a maximal left star subexpression. Let us remark that $B_2 = ((ba + abaaa) \cdot ba^*)$. We now decompose K_1.

```
> B3:=Comp({a,b},Comp({a,b},K1) &D K1);
```

$$[1, \{1\ \}, \mathtt{table}()]$$

We can conclude that the "star removal" conjecture is verified for L. We have $L = (ab)^* \cdot ((ba + abaaa)ba^*) \cdot C$, where C has ε for maximal left star subexpression. Furthermore we can write this test as a procedure.

```
starrem:=proc(E)
local B,K,Er,Bm,halt;

halt:=false;
Er:=E;
while (halt<>true) do
        B:=Comp({},Comp({},Er) &D Er);
        if op(2,B &M _1A)={} then
                halt:=true
        else
                Bm:=mini(deter(B));
                lprint(Bm);
                K:=Er &M ((B &M _1A) &C Er);
                Er:=mini(deter(K));
        fi;
od;
end;
```

References

1. P.Caron, AG: A set of Maple packages for manipulating automata and semigroups, *Rapport LIR* **96-10**, 1996, submitted to Software Practice and Experience.
2. J. A. Brzozowski, Open problems about regular languages, in *Formal language theory, perspectives and open problems*, R. V. Book editor, Academic Press, New York, 1980, 23-47.