Computing the translational hull of an arbitrary semigroup

Finn Smith

27th July 2016

We can compute the translational hull of an arbitrary finite semigroup S by backtrack search. Let $R = \{r_i : 1 \leq i \leq n\}$ be a minimal set of representatives of the $\mathcal{L}$- and $\mathcal{R}$-classes of S. We write left [right] translations on the left [right]. Let λ and ρ be the linked left and right translations to be constructed, respectively. Then all pairs λ and ρ are determined by the values $\lambda(r_i)$, $(r_i)\rho$ for $1 \leq i \leq n$.
To determine allowed values for λ and ρ we can backtrack search:

Algorithm 1 Backtrack search for linked pairs

```
Compute the multiplication table of S for faster checks
while there are more possibilities to check do
    for i = 1 to n do
        Choose λ(r_i) from the available values
        for s ∈ S^1 do
            Set λ(r_i s) = λ(r_i) s
            if a value has been overwritten then
                Backtrack
            end if
            for i < j ≤ n do
                if r_i s = r_j t for some t ∈ S^1 then
                    Restrict possible values for λ(r_j) using the translation condition λ(r_j) t = λ(r_i) s
                    if there are no possible values for λ(r_j) then
                        Backtrack
                    end if
                end if
                Restrict possible values for (r_j)ρ using the linked pairs condition r_j λ(r_i s) = (r_j)ρ r_i s
                if there are no possible values for (r_j)ρ then
                    Backtrack
                end if
            end for
        end for
        Choose (r_i)ρ such that r_i λ(r_i) = (r_i)ρ r_i
        Propagate ρ and restrict, dually to λ.
    end for
    Store λ and ρ, backtrack.
end while
```

To see that the produced λ, ρ are linked translations, let $a, b \in S$. Then $a = r_i x$ for some $1 \leq i \leq n$, $x \in S$. Since we backtracked if we found $r_k y = r_i x$ with $\lambda(r_k) y \neq \lambda(r_i) x$, we can choose

r_i freely. Then

$$\begin{aligned}\lambda(a)\,b &= \lambda(r_i x)\,b \\ &= \lambda(r_i)\,xb \\ &= \lambda(r_i x b) \\ &= \lambda(ab)\end{aligned}$$

Similarly, $a(b)\,\rho = (ab)\,\rho$.
Now $a = ur_i$, and $b = r_j x$ for some $1 \leq i, j \leq n$ and $u, s \in S^1$. By construction, $r_i \lambda(r_j) = (r_i)\,\rho r_j$ and hence

$$\begin{aligned}a\lambda(b) &= ur_i \lambda(r_j)\,x \\ &= u(r_i)\,\rho r_j x \\ &= (a)\,\rho b\end{aligned}$$

Note: instead of R, we could modify the algorithm to work with a set A which is minimal such that $AS = S$, $SA = S$, and then would have that $|A| \leq n$.