On the coinductive nature of centralizers

Charles Grellois*

June 2014

Abstract

The centralizer of a language of finite words L, denoted $\mathcal{C}(L)$, is the biggest solution of the language equation $X \cdot L = L \cdot X$. Conway conjectured that if L is regular its centralizer should be as well; Kunc proved however that such a language may have a non-recursively enumerable centralizer, by encoding a Minsky machine in its complementary. We show that clockwise Turing machines can be used in the proof and then remark that centralizers may be seen as greatest fixpoints of a function over a lattice of languages. Therefore centralizers are of *coinductive* nature, a key property which contributes to explain Kunc's result: they can compute the whole graph of configurations of a Turing machine, of which we may exclude some connected components. By removing every component containing an initial configuration, we obtain a centralizer which represents the "computations" of the complementary of a recursively enumerable language.

I would like to thank Prof. Karhumäki for introducing me to this problem and advising me during an internship in Turku in 2009.

1 Introduction

Given a finite word w, it is well-known (see e.g. [1] or [2]) that the commutation equation

$$x \cdot w \ = \ w \cdot x \tag{1}$$

has for solution the set of powers of w. It is tempting to generalize (1) to languages of finite words. Given such a language L, its *centralizer*, denoted $\mathcal{C}(L)$, is the biggest solution of the language equation

$$X \cdot L \ = \ L \cdot X \tag{2}$$

Such a solution always exists (see Section 4), and we obtain easily that

$$L^* \ \subseteq \ \mathcal{C}(L) \ \subseteq \ \mathrm{Pref}(L^*)$$

Conway formulated in [3] the problem of the regularity of the centralizer of a regular language. It remained open for many years, and progress was only made on restricted versions, leading to a series of results exposed in [4], until Kunc [5] gave a surprisingly negative result in 2005:

Theorem 1 (Kunc (2005) [5])**.** • *There exists a regular, star-free language whose centralizer is not recursively enumerable.*

*PPS & LIAFA, Université Paris Diderot

- *There exists a finite language whose centralizer is not recursively enumerable.*

In the sequel we expose some of the main ingredients of Kunc's proof, then use them to propose a similar, yet new proof of this result using clockwise Turing machines. We believe that they make the proof easier than the original one relying on Minsky machines, since they only have one possible type of transition, as detailed in Section 3. We then show that centralizers may be obtained as the greatest fixpoints of an order-preserving function over the lattice of languages of finite words, and therefore are of a *coinductive* nature. We finally explain how this helps understanding Kunc's result.

2 Elements of Kunc's proof

A game-theoretic intuition. Before sketching Kunc's proof, it is useful to recall the game-theoretic interpretation of the equation (2), which is closely connected to the following result:

Proposition 1. *Given u, $v \in L$, suppose that $x \cdot u = v \cdot y$. Then $x \in \mathcal{C}(L) \iff y \in \mathcal{C}(L)$.*

Fix an alphabet A and a language L, and consider a two-player game on A^*. On a position w, Adam chooses a word $u \in A^*$, and then a word $w' \in \{w \cdot u,\ u \cdot w\}$. Eve has to answer by choosing a word $v \in L$ which is a prefix of w'; and v becomes the new position of the game. Adam wins if Eve cannot answer at some point, meaning that the initial position (and all the ones on the path to the last position, due to Proposition 1) correspond to words which do not belong to $\mathcal{C}(L)$. If Eve can answer infinitely, then all the positions of the play are in $\mathcal{C}(L)$. This game is obviously determined.

Minksy machines. Recall that a Minsky machine [6] is a machine with registers (or counters) storing integers that one may increment, decrement or test if they are worth 0 or not. Minsky machines with two counters or more are equivalent to Turing machines. In his proof, Kunc fixes a Minsky machine with two counters and encodes the configuration in which the machine is in state q, with the integer n on its first counter and m on its second as the word

$$a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \tag{3}$$

He designs a language L such that its centralizer contains, among other, words corresponding to the configurations of the machine. The operations are simulated by commutation. For example, if the machine in state q increments its first counter and goes to state q', we have

$$\begin{array}{rr} & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q^2 \quad \in \mathcal{C}(L) \\ \iff & g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_q \quad \in \mathcal{C}(L) \\ \iff & e_q\ f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1} \quad \in \mathcal{C}(L) \\ \iff & f_q\ g_q\ a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'} \quad \in \mathcal{C}(L) \\ \iff & a^{n+2}\ b\ \widehat{a}^{m+1}\ \widehat{d}_{q'}^2 \quad \in \mathcal{C}(L) \end{array}$$

Kunc obtains this by enforcing that, if Adam catenates $g_q\ a$ to the left (notice that this increments the first counter), Eve has no choice but to remove $\widehat{d}_q$ on the right. Then if Adam plays $e_q\ f_q$ on the left, Eve has to remove $\widehat{d}_q$ again. L is constructed such that Adam can only add $\widehat{d}_{q'}$ to the right (or go back in the computation). Eve must then answer by removing e_q only; Adam plays the same move again, and Eve has to remove $f_q\ g_q$. Notice that this construction allows to add a new letter on the left of the word.

Similar equivalences hold for relating configurations in a way which simulates every of the operations of the Minsky machine, and we refer to [5] for the ones corresponding to decrementation, nullity test and incrementation of the second counter (obtained by symmetry). For our encoding using clockwise Turing machines, it is useful to recall that in Kunc's construction the following holds:

$$\begin{array}{rcl} & a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d_q^2} & \in \mathcal{C}(L) \\ \iff & d_q\ a^{n+1}\ b\ \widehat{a}^{m+1}\ \widehat{d_q} & \in \mathcal{C}(L) \\ \iff & d_q^2\ a^{n+1}\ b\ \widehat{a}^{m+1} & \in \mathcal{C}(L) \end{array}$$

A last key feature of L is that it is designed to exclude from $\mathcal{C}(L)$ the words encoding the initial configurations of the machine – but only them and, by previous considerations, all the reachable configurations. As a consequence, in $\mathcal{C}(L)$ every word corresponding to the encoding of a final configuration describes an unreachable configuration, and thus an element of the complementary of the language of the machine. Since there exists Minsky machines whose complementary is not recursively enumerable, Theorem 1 holds.

3 Clockwise Turing machines

An inconvenient of Kunc's proof is that Minsky machines have several distinct kind of transitions, and he has to give a specific encoding for every of these. We present in this Section an alternative formalism due to Neary and Woods [7], which leads in our opinion to a more economical proof since it only has one behaviour.

Informally, a clockwise Turing machine is a variant of Turing machine with only one tape, which is circular, and whose tape head only moves in a clockwise direction.

Definition 1. A clockwise Turing machine is a tuple $\mathcal{M} = (Q, A, \delta, q_i, q_f)$, where

- Q is a finite set of states,
- A is a finite set of tape symbols,
- $\delta : Q \times A \to \{A \cup A \cdot A\} \times Q$ is the transition function,
- q_i is the initial state,
- and q_f is the final state.

Note that the tape is finite, but not limited since δ may output two symbols at once. In a configuration whose state is q and in which the circular tape, starting from its head and reading clockwise, contains the word $u_1 \cdots u_n$, applying the transition $\delta(q, u_1) = (v, q')$ results in a new configuration with state q' and tape $u_2 \cdots u_n \cdot v$ (note that v may consist of two letters). Neary and Woods proved that

Proposition 2. *Any Turing machine can be simulated by a clockwise Turing machine.*

It is worth noticing that their encoding does not require an additional counter for simulating the transitions where the tape head should move counter-clockwise. We may then adapt Kunc's construction, encoding a configuration where the current state is q and the circular tape, starting from its head and reading it clockwise, contains the word $u_1 \cdots u_n$, by the word

$$d_{q,u_1}^2\ u_1\ \cdots\ u_n \tag{4}$$

We can obtain, as previously, the following useful property:

$$d^2_{q,u_1}\ u_1\ \cdots\ u_n \quad \in \mathcal{C}(L) \qquad \Longleftrightarrow \qquad u_1\ \cdots\ u_n\ \widehat{d}^2_{q,u_1} \quad \in \mathcal{C}(L)$$

The simulation of a transition $\delta(q, u_1) = (v, q')$ is performed by adapting Kunc's ideas to obtain:

$$\begin{array}{rrl}
 & u_1\ u_2\ \cdots\ u_n \widehat{d}^2_{q,u_1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ \widehat{d}_{q,u_1} & \in \mathcal{C}(L) \\
\Longleftrightarrow & e_{q,u_1}\ f_{q,u_1}\ g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n & \in \mathcal{C}(L) \\
\Longleftrightarrow & g_{q,u_1}\ u_1\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v}\ \widehat{e}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d_{q',v}\ u_2\ \cdots\ u_n\ v\ \widehat{g}_{q',v}\ \widehat{f}_{q',v} & \in \mathcal{C}(L) \\
\Longleftrightarrow & d^2_{q',v}\ u_2\ \cdots\ u_n\ v & \in \mathcal{C}(L)
\end{array}$$

To finish, we exclude any initial configuration from $\mathcal{C}(L)$ using Kunc's techniques. This gives a similar, yet shorter proof of Kunc's theorem.

4 Centralizers as fixpoints

In the proofs of Theorem 1, the idea is to build a language whose centralizer represents the configuration graph of the machine, and then to exclude every component connected to an initial configuration. In this way one obtains a graph whose final configurations precisely represent the complementary of the machine's language. In this Section, we explain why these centralizers can represent the whole configurations graph, and not only its computable parts, by unravelling their *coinductive* nature.

Centralizer as fixpoints. Consider the set $Fin(A)$ of languages of finite words over A. It is a complete lattice ordered by inclusion, on which the map

$$\phi\ :\ X\ \mapsto\ (L^{-1}\,X)\cdot L \tag{5}$$

is order-preserving. Indeed, if $X \subseteq Y$ then $L^{-1}\,X \subseteq L^{-1}\,Y$ and thus $\phi(X) \subseteq \phi(Y)$. Recall the following Theorem:

Theorem 2 (Knaster–Tarski)**.** *Let $\mathcal{L}$ be a complete lattice and $f : \mathcal{L} \to \mathcal{L}$ be an order-preserving function. Then the set of fixed points of f in L is also a complete lattice.*

This implies in particular that ϕ has a fixpoint. In the case of centralizers, it has in general many fixpoints, since any solution to the commuting equation (2) corresponds to a fixpoint. The centralizer of L is actually its greatest fixpoint, and can thus be defined as their supremum. This corresponds to the well-known property that the centralizer of a language L may be described as the union of all the languages commuting with L.

Recall that, in a complete lattice, one can compute the least fixpoint of an order-preserving function f as

$$\mathrm{lfp}(f) \ =\ \bigvee_{i\in\mathbb{N}} f^i(\bot)$$

and dually its greatest fixpoint is

$$\mathrm{gfp}(f) \;=\; \bigwedge_{i\in\mathbb{N}} f^i(\top)$$

In $Fin(A)$, $\bot$ corresponds to $\emptyset$ and $\top$ to A^*. We thus see that these two dual fixpoints are very different in nature: the first is built *inductively*, starting from the least possible input, and iteratively computing a value. Machines are inductive: they start from a configuration, and compute step by step the result of their computation.

The latter is *coinductive*: it is constructed starting from every possible input, and iterately removing data. In the case of ϕ, this can be understood as computing the set of positions where Eve wins, that is, the elements of $\mathcal{C}(L)$: starting from all the positions, we remove the ones where Eve looses immediatly, then the ones on which she looses after a step, and so on for every finitely reachable position. The result is thus the set of positions from which Eve can play forever.

Calculus is inductive, but centralizers are coinductive.

This explains why centralizers can compute the whole graph of configurations of a machine, and obtain non-computable configurations, from the moment one gives them the ability to simulate their transitions. From this perspective, we may summarize Kunc's proof as follows:

1. Create a language L whose centralizer $\mathcal{C}(L)$ can simulate the transitions of a machine.
2. $\mathcal{C}(L)$ contains the encoding of all configurations of the machine, and a word describing a configuration may be obtained by commuting from another such word if and only if the configurations they describe belong to the same connected component of the configurations graph of the machine.
3. Modify L so that the encoding of every initial configuration of the machine is excluded from $\mathcal{C}(L)$. As a result, $\mathcal{C}(L)$ now only contains encodings of unreachable configurations, and thus represents the complementary of the language.
4. Take an universal machine, and obtain the Theorem.

5 Conclusion and perspectives

In this paper we gave the main ingredients of Kunc's proof of the existence of regular languages whose centralizer is not recursively enumerable. We proposed an alternate encoding, which allows to simplify a bit the proof, and emphasized on the fact that the coinductive nature of centralizers is their key to the non-recursively enumerable world.

Note that Kunc's original article [5] adapts the proof we sketched to finite languages. Using similar techniques, our approach also gives a proof of the existence of a non-recursively enumerable centralizer in the case of a finite language.

An advantage of this approach, which was the first reason for us to consider Turing machines instead of Minsky ones, is that much more research on small universal Turing machines has been carried on than on such Minsky machines, and it is thus more relevant to consider the former in order to estimate the cardinality of a finite language with a non-recursively enumerable centralizer – especially due to the fact that Kunc's proof is for two counter machines, and does not seem to adapt to three or more counters, while "small" universal Minsky machines would require eight. Using small universal Turing machines from [7], I could estimate at around 10^{22} words the size of a language with a non-recursively enumerable centralizer.

References

[1] *Combinatorics on words*. Encyclopedia of mathematics and its applications. Addison-Wesley, Advanced Book Program, World Science Division, 1983.

[2] Christian Choffrut and Juhani Karhumäki. Combinatorics of words, 1997.

[3] J. H. Conway. *Regular algebra and finite machines*. Chapman and Hall, 1971.

[4] Juhani Karhumäki and Ion Petre. Two problems on commutation of languages, 2003.

[5] Michal Kunc. The power of commuting with finite sets of words. *Theory Comput. Syst.*, 40(4):521–551, 2007.

[6] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

[7] Turlough Neary and Damien Woods. Four small universal turing machines, 2009.