Playing with Conway's Problem

Emmanuel Jeandel, Nicolas Ollinger

Playing with Conway's Problem

Emmanuel Jeandel[1] and Nicolas Ollinger[2]

[1] LIP, UMR CNRS, École Normale Supérieure de Lyon,
46 allée d'Italie, 69364 Lyon Cedex 07, France,
Emmanuel.Jeandel@ens-lyon.fr

[2] LIF, UMR CNRS, Université de Provence, CMI,
39 rue Joliot-Curie, 13453 Marseille Cedex 13, France,
Nicolas.Ollinger@lif.univ-mrs.fr

Abstract. The centralizer of a language is the maximal language commuting with it. The question, raised by Conway in 1971, whether the centralizer of a rational language is always rational, recently received a lot of attention. In Kunc 2005, a strong negative answer to this problem was given by showing that even complete co-recursively enumerable centralizers exist for finite languages. Using a combinatorial game approach, we give here an incremental construction of rational languages embedding any recursive computation in their centralizers.

1 Introduction

In 1999, Choffrut *et al.* [1] renewed an old problem raised by Conway [2] in 1971: given a rational language, does its centralizer — the maximal language commuting with it — have to be rational? The property is known to hold for some particular families of languages. In the case of codes, Ratoandramanana [13] showed in 1989 that it holds for biprefix codes, raising a restriction of Conway's problem to codes which recently recieved a positive answer by Karhumäki *et al.* [5]. In the general case, until recently, the best known result, by Karhumäki and Petre [6], was that the centralizer of a recursive language has to be co-recursively enumerable. This property may also be considered as a particular case of results of Okhotin [11] concerning the computational power of systems of equations on languages. In 2004, the community was thrilled by an announcement by Kunc [8] that Conway's problem deserved a strong negative answer. This announcement was followed by a conference communication [9] in 2005 showing that finite languages exist whose centralizers are complete for co-recursively enumerable languages. It includes a sketch of the proof for the special case of rational languages. While simpler than the proof for finite languages, this proof is still rather involved — mostly due to a direct construction of the language encoding a given Minsky machine.

In this paper we propose another proof of the existence of rational languages with non-recursive centralizers. The key arguments of the proof come from a careful study of the first example in Kunc [9] leading to the core constructions

of our proof: *checking* and *flooding*. Our approach significantly differs for two reasons. First, a combinatorial game point of view is taken through the whole proof. Apart from the fun of dealing with a problem raised by Conway using games, games are convenient tools to embed a dynamical process like a computation into a static object like a fix-point. Using this point of view, a computation can be transformed incrementally into a centralizer by transforming winning strategies from one game to another more specialized game. Secondly, the construction of the language embedding a particular computation is incremental — explicitly explaining how to compile a program into a language so that its centralizer corresponds to the computation. Whereas the final proof is by no way shorter than Kunc original proof, cutting the construction into locally independent propositions improves its readability. Our proof also uses Post tag systems instead of Minsky machines as Post tag systems are in a way closer to centralizers.

In this paper, the letters Σ and Γ denote *finite alphabets*. The set of finite words over an alphabet Σ is denoted by $\Sigma^\star$, the *empty word* by ε, the *catenation* of two words x and y by xy and the length of $x \in \Sigma^\star$ by $|x|$. A word x is a *prefix* (resp. *suffix*) of a word y, denoted by $x \prec y$ (resp. $y \succ x$), if there exists a word $z \in \Sigma^\star$ such that $xz = y$ (resp. $y = zx$); this word z is unique and is denoted as $x^{-1}y$ (resp. yx^{-1}). A word x is a *subword* of a word y if there exists two words $z, z' \in \Sigma^\star$ such that $zxz' = y$. A *language* over Σ is a subset of $\Sigma^\star$. The *product* XY of two languages X and Y is the language $\{xy : x \in X, y \in Y\}$. The language of prefixes (resp. suffixes) of a language X, denoted as $\mathrm{Pref}(X)$ (resp. $\mathrm{Suff}(X)$) is the set of every prefixes (resp. suffixes) of each word in X. The language of subwords of a language X, denoted as $\mathrm{Sub}(X)$ is the set of every subwords of each words in X. The language $X^{-1}Y$ is the language $\{z : \exists x \in X, \exists y \in Y, y = xz\}$. The language YX^{-1} is the language $\{z : \exists x \in X, \exists y \in Y, y = zx\}$.

Two languages X and Y *commute* if the equation $XY = YX$ is satisfied. The set of languages that commute with a given language X is closed by union. Thus it admits a unique maximal element for inclusion called the *centralizer* of X, denoted by $\mathcal{C}(X)$. The centralizer of X always contains $X^\star$. Moreover, if X contains the empty word then its centralizer is equal to $\Sigma^\star$. Otherwise, it is contained into $\mathrm{Pref}(X^\star) \cap \mathrm{Suff}(X^\star)$.

2 Cutenation games

In this section cutenation[3] games are introduced and their relations with centralizers are explained before sketching the proof of existence of rational languages with non-recursive centralizers.

2.1 Definition

A *cutenation game* is a tuple $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ where both $(\mathfrak{A}, \mathfrak{B}, L)$ and $(\mathfrak{A}, \mathfrak{B}, R)$ are bipartite graphs whose edges are tagged with words on Σ (*i.e.*

[3] *cutenation* is a free contraction of both words *cut* and *catenation*.

$L, R \subseteq \mathfrak{A} \times \mathfrak{B} \times \Sigma^\star$) and the mappings $V_A : \mathfrak{A} \to \Sigma^\star$ and $V_B : \mathfrak{B} \to \Sigma^\star$ constraint the positions. Given such a game, a *A-configuration* $(\mathfrak{a}, x) \in \mathfrak{A} \times \Sigma^\star$ verifies $x \in V_A(\mathfrak{a})$. Symmetrically a *B-configuration* $(\mathfrak{b}, y) \in \mathfrak{B} \times \Sigma^\star$ verifies $y \in V_B(\mathfrak{b})$.

Remark. In this paper we will only consider connected cutenation games, that is cutenation games $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ for which the bipartite graph $(\mathfrak{A}, \mathfrak{B}, L \cup R)$ is connected and both sets $\mathfrak{A}$ and $\mathfrak{B}$ are finite.

Notation. We will depict L and R by a graph where $\mathfrak{A}$-vertices are represented by black points, $\mathfrak{B}$-vertices are represented by white points, L-edges are represented by plain edges and R-edges are represented by dashed edges (for clarity ε tags will be omitted).

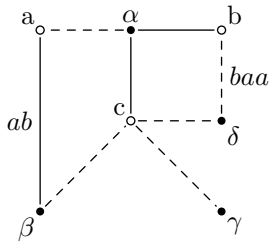

Fig. 1. graphical representation of a simple cutenation game

Example. A sample cutenation game, omitting V_A and V_B, is depicted on Fig. 1 where $\mathfrak{A} = \{\alpha, \beta, \gamma, \delta\}$, $\mathfrak{B} = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, $L = \{(\mathrm{a}, \beta, ab), (\mathrm{b}, \alpha, \varepsilon), (\mathrm{c}, \alpha, \varepsilon)\}$ and $R = \{(\mathrm{a}, \alpha, \varepsilon), (\mathrm{b}, \delta, baa), (\mathrm{c}, \beta, \varepsilon), (\mathrm{c}, \gamma, \varepsilon), (\mathrm{c}, \delta, \varepsilon)\}$.

A cutenation game is played as an iterated two-player combinatorial game where the set of A-configurations is the set of positions of the player A and the set of B-configurations is the set of positions of the player B. A move of the player A, from a A-configuration $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$, is a catenation:

- either a l-move $(\mathfrak{a}, x) \vdash_{A,l} (\mathfrak{b}, yx)$ such that $yx \in V_B(\mathfrak{b})$ and $(\mathfrak{a}, \mathfrak{b}, y) \in L$;
- or a r-move $(\mathfrak{a}, x) \vdash_{A,r} (\mathfrak{b}, xy)$ such that $xy \in V_B(\mathfrak{b})$ and $(\mathfrak{a}, \mathfrak{b}, y) \in R$.

Symmetrically, a move of the player B, from a B-configuration $(\mathfrak{b}, y)$ to a A-configuration $(\mathfrak{a}, x)$, is a cut:

- either a l-move $(\mathfrak{b}, yx) \vdash_{B,l} (\mathfrak{a}, x)$ such that $x \in V_A(\mathfrak{a})$ and $(\mathfrak{a}, \mathfrak{b}, y) \in L$;
- or a r-move $(\mathfrak{b}, xy) \vdash_{B,r} (\mathfrak{a}, x)$ such that $x \in V_A(\mathfrak{a})$ and $(\mathfrak{a}, \mathfrak{b}, y) \in R$.

A round of the game starts from a A-configuration $(\mathfrak{a}, x)$ and consists first of a move of the player A from $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$, then of a move of the player B from $(\mathfrak{b}, y)$ to a A-configuration $(\mathfrak{a}', x')$. Furthermore if A plays a

l-move then B must play a r-move and symmetrically if A plays a r-move then B must play a l-move. If a player cannot move then the player looses. The next round will start from $(\mathfrak{a}', x')$. If the game lasts forever then the player B wins.

Example. For the cutenation game of Fig. 1, this is a valid sequence of consecutive rounds:

$$\begin{array}{lllll} 1.\ (\beta, aa) & \vdash_{A,l} & (\mathrm{a}, abaa) & \vdash_{B,r} & (\alpha, abaa)\ ; \\ 2.\ (\alpha, abaa) & \vdash_{A,l} & (\mathrm{b}, abaa) & \vdash_{B,r} & (\delta, a)\ ; \\ 3.\ (\delta, a) & \vdash_{A,r} & (\mathrm{c}, a) & \vdash_{B,l} & (\alpha, a)\ . \end{array}$$

Lemma 1. *Starting from a A-configuration $(\mathfrak{a}, x)$ either the player A has a winning strategy or the player B has a winning strategy.*

Proof. Let $(\mathfrak{a}, x)$ be a A-configuration for which neither the player A nor the player B has a winning strategy. If every move from the player A starting from $(\mathfrak{a}, x)$ would lead to a B-configuration from which the player B could move to a A-configuration on which the player B has a winning strategy then the position $(\mathfrak{a}, x)$ would be winning for the player B. Thus, the player A has a valid move from $(\mathfrak{a}, x)$ to a B-configuration $(\mathfrak{b}, y)$ from which the player B can move either to a A-configurations on which the player A has a winning strategy or to A-configurations on which neither the player A nor the player B have a winning strategy. On such configurations the best moves from both the player A and the player B would lead to an infinite run. By the rules, the player B would win which implies that the player B has a winning strategy starting from $(\mathfrak{a}, x)$. ■

2.2 Languages and centralizers

Given a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ and an element $\mathfrak{a}$ of $\mathfrak{A}$, the language $\mathfrak{L}(\mathfrak{a})$ is the set of words $x \in \Sigma^\star$ such that the configuration $(\mathfrak{a}, x)$ admits a winning strategy for the player B.

In the special case where both $\mathfrak{A}$ and $\mathfrak{B}$ are finite and L, R, V_A and V_B are recursive, given an element $\mathfrak{a}$ of $\mathfrak{A}$, the language $\mathfrak{L}(\mathfrak{a})$ is co-recursively enumerable. It follows from the fact that one can exhaustively search a winning strategy for the player A as a finite one exists — the player B only has finitely many valid moves starting from a B-configuration.

The centralizer $\mathcal{C}(X)$ of a given language X can be expressed as the language $\mathfrak{L}$ associated to the unique element of $\mathfrak{A}$ of the cutenation game where: both $\mathfrak{A}$ and $\mathfrak{B}$ are singletons, L and R are both equal to the language X and both $V_A(\mathfrak{a}) = \Sigma^\star$ and $V_B(\mathfrak{b}) = \Sigma^\star$. In the following, we call such a game a commutation game. For the sake of readability, when manipulating cutenation game where $\mathfrak{A}$ and $\mathfrak{B}$ are singletons, we will manipulate L, R, V_A and V_B as subsets of $\Sigma^\star$ and denote the language associated with the game as $\mathfrak{L}$. For the same reasons A-configurations and B-configurations will be considered as elements of $\Sigma^\star$.

In order to prove the main result of this paper, we will proceed through the following steps. First, we restrict ourselves to a specific subset of special cutenation games. Then, we show how to recursively encode co-recursively enumerable languages into the language of such a game. After that we proceed to the core of the proof and explain how to transform such special cutenation game into a commutation game in which the language associated with any element of $\mathfrak{A}$ is recursively encoded into the language associated to the commutation game of a rational language.

3 Encoding Post Tag Systems

In order to encode every co-recursively enumerable language into the language associated to a commutation game, the family of cutenation games is first restricted to games with special properties that will allow further reductions; then Post tag systems are encoded into games verifying these particular properties.

3.1 Restraining Cutenation Games

The following special kinds of cutenation games will be used in the proof. The main reason to enforce these properties is to enable the latter encoding of both $\mathfrak{A}$ and $\mathfrak{B}$ into L, R, V_A and V_B.

Unfairness. A cutenation game is *unfair* if the player A has no constraint. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ over the alphabet Σ is *unfair* if $V_B = \Sigma^\star$.

Rootedness. A cutenation game is *rooted* if the player A can catenate non-empty words on the left (respectively on the right) from at most one position called the left root (respectively the right root). More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *rooted* if there exists a left root $\mathfrak{a}_L \in \mathfrak{A}$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in L$ if $x \neq \varepsilon$ then $\mathfrak{a} = \mathfrak{a}_L$ and there exists a right root $\mathfrak{a}_R \in \mathfrak{A}$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in R$ if $x \neq \varepsilon$ then $\mathfrak{a} = \mathfrak{a}_R$.

Oscillation. A cutenation game is *oscillating* if the player A is enforced to oscillate at each round between l-moves and r-moves. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *oscillating* if the set $\mathfrak{A}$ can be split into two disjoint sets $\mathfrak{A}_L$ and $\mathfrak{A}_R$ such that for all $(\mathfrak{a}, \mathfrak{b}, x) \in L$ necessarily $\mathfrak{a} \in \mathfrak{A}_L$ and for all $(\mathfrak{a}, \mathfrak{b}, x) \in R$ necessarily $\mathfrak{a} \in \mathfrak{A}_R$.

Separation. A cutenation game is *separated* if positions can be viewed as a product of a left and right position modified independently by l-moves and r-moves. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *separated* if there exists two sets S_L and S_R such that $\mathfrak{A} \cup \mathfrak{B} \subseteq S_L \times S_R$ and both L and R satisfies the following requirements. For all move $((s,t),(s',t'),x) \in L$ only the left part is modified so $t = t'$. Moreover, for all $t'' \in S_R$ such that $(s,t'') \in \mathfrak{A}$ necessarily

$(s', t'') \in \mathfrak{B}$ and the move $((s, t''), (s', t''), x)$ must be in L. Symmetrically, For all move $((s, t), (s', t'), x) \in R$ only the right part is modified so $s = s'$. Moreover, for all $s'' \in S_L$ such that $(s'', t) \in \mathfrak{A}$ necessarily $(s'', t') \in \mathfrak{B}$ and the move $((s'', t), (s'', t'), x)$ must be in R.

Orientation. A cutenation game is *oriented* if it is both separated and oscillating and if its left and right positions can be ordered into minimal and maximal positions, a move changing the corresponding position from minimal to maximal. More formally, a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ is *oriented* if it is both separated and oscillating and if the set S_L, respectively S_R, can be split into two disjoint sets S_L^- and S_L^+, respectively S_R^- and S_R^+, such that the set $\mathfrak{A}_L$ subsets $S_L^- \times S_R^+$, the set $\mathfrak{A}_R$ subsets $S_L^+ \times S_R^-$, and the set $\mathfrak{B}$ subsets $S_L^+ \times S_R^+$.

Lemma 2. *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an oscillating cutenation game. Let ν_L, respectively ν_R, be the application mapping an element of $\mathfrak{A} \cup \mathfrak{B}$ to its associated connected component in the bipartite graph $(\mathfrak{A}, \mathfrak{B}, L)$, respectively $(\mathfrak{A}, \mathfrak{B}, R)$. If the mapping $\nu : x \mapsto (\nu_R(x), \nu_L(x))$ is injective then the given oscillating cutenation game can be considered, up to the isomorphism ν, as a separated oscillating cutenation game where $S_L = \nu_R(\mathfrak{A} \cup \mathfrak{B})$ and $S_R = \nu_L(\mathfrak{A} \cup \mathfrak{B})$.*

Proof. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an oscillating cutenation game satisfying the hypothesis. Let $(\mathfrak{a}, \mathfrak{b}, x)$ be in L and let both $(s, t) = \nu(\mathfrak{a})$ and $(s', t') = \nu(\mathfrak{b})$. By definition of ν_L, as $\mathfrak{a}$ and $\mathfrak{b}$ are connected by L then $\nu_L(\mathfrak{a}) = \nu_L(\mathfrak{b})$ thus $t = t'$. Moreover, as the game is oscillating $\mathfrak{a} \in \mathfrak{A}_L$. Let $t'' \in S_R$ be such that $(s, t'') = \nu(\mathfrak{a}')$ for some $\mathfrak{a}' \in \mathfrak{A}$. By definition of ν_R this means that $\mathfrak{a}$ and $\mathfrak{a}'$ are connected by R. As $\mathfrak{a} \in \mathfrak{A}_L$ it implies that $\mathfrak{a} = \mathfrak{a}'$. A symmetrical reasoning applies to R. Therefore, the game is, up to isomorphism ν, separated. ■

Lemma 3. *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be a separated oscillating cutenation game obtained by lemma 2. Such a game is oriented.*

Proof. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be a spearated oscillating cutenation game obtained by lemma 2. Let S_L^+ be defined has $\{s : \exists t \in S_R, (s, t) \in \mathfrak{B}\}$ and $S_L^- = S_L \setminus S_L^+$. Symmetrically, let S_R^+ be defined has $\{t : \exists s \in S_L, (s, t) \in \mathfrak{B}\}$ and $S_R^- = S_R \setminus S_R^+$. By construction $\mathfrak{B}$ subsets $S_L^+ \times S_R^+$. Let (s, t) be in $\mathfrak{A}_L$. As the cutenation game is connected there is at least one move in L involving (s, t) thus $t \in S_R^+$. Assume that $s \in S_L^+$. This means that there exists some $t' \in S_R^+$ such that $(s, t') \in \mathfrak{B}$. By definition of ν_R necessarily (s, t) and (s, t') are connected by R. As the game is oscillating it implies that $t = t'$ but $\mathfrak{A}$ and $\mathfrak{B}$ are disjoint. Therefore s must be in S_L^-. Symmetrically, the same holds for $\mathfrak{A}_R$. ■

3.2 Post Tag Systems

A *Post tag system* $\mathcal{P}$ is a triple (Σ, k, φ) where Σ is a finite alphabet, k the step of the system and φ is a mapping from Σ^k to $\Sigma^\star$. A configuration of the system is a word u from $\Sigma^\star$. For all u in $\Sigma^\star$ and i in Σ^k, the configuration iu evolves into the configuration $u\varphi(i)$. The computation stops when no further evolution

is possible. The language $L_{\mathcal{P}}$ associated to the Post tag system is the set of words for which the evolution eventually stops. Post tag systems recursively encode any recursively enumerable language into their languages. For more details about tag systems and their computational power, the reader might consult Minsky [10].

Proposition 1. *Let $\mathcal{P}$ be a Post tag system over the alphabet Σ. An unfair rooted oriented cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ over the same alphabet exists such that, for some distinguished element $\mathfrak{a} \in \mathfrak{A}$, both language $\mathfrak{L}(\mathfrak{a})$ and $\Sigma^\star \setminus L_{\mathcal{P}}$ are equal.*

Proof. Let $\mathcal{P}$ be a Post tag system (Σ, k, φ) where Σ is an alphabet of size n. The tag system will be encoded as a cutenation game $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ where

$$\begin{aligned}\mathfrak{A} &= \{\alpha, \eta\} \cup \textstyle\bigcup_{i\in\Sigma^k} \{\beta_i, \gamma_i, \delta_i, \zeta_i\}\,,\\ \mathfrak{B} &= \{\mathrm{a}\} \cup \textstyle\bigcup_{i\in\Sigma^k} \{\mathrm{b}_i, \mathrm{c}_i, \mathrm{d}_i\}\end{aligned}$$

and the relations L and R are depicted on Fig. 2.

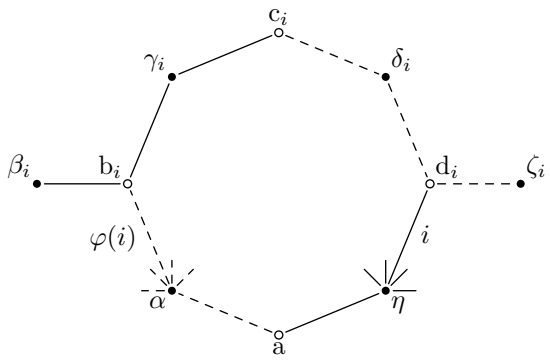

Fig. 2. the rooted oriented cutenation game of a Post System

The constraints V_A are defined as follows: $V_A(\alpha) = \Sigma^\star$, $V_A(\eta) = \Sigma^\star$, and for all i in Σ^k:

$$\begin{aligned}V_A(\gamma_i) &= i\Sigma^\star\varphi(i),\\ V_A(\beta_i) &= \left(\Sigma^k \setminus \{i\}\right)\Sigma^\star\varphi(i),\\ V_A(\delta_i) &= i\Sigma^\star\varphi(i),\\ V_A(\zeta_i) &= \Sigma^\star \setminus i\Sigma^\star\varphi(i).\end{aligned}$$

This game is unfair and rooted, the roots being α and η. Moreover it is oscillating and fulfills the requirements of lemma 2 thus by lemma 3 it is oriented. It remains to prove that $\mathfrak{L}(\alpha)$ equals $\Sigma^\star \setminus L_{\mathcal{P}}$.

Let x be a word in $L_{\mathcal{P}}$. A winning strategy for the player A starting from the A-configuration (α, x) is to follow the computation steps of the Post tag system. If a transition of the tag system exists starting from x then x can be rewritten as iy with $i \in \Sigma^k$. Going through the states $\mathrm{b_i}$, $\mathrm{c_i}$, $\mathrm{d_i}$ and a, the player A will

force the player B to go to the A-configuation $(\alpha, y\varphi(i))$. If no transition of the tag system exists starting from x this means that $|x|$ is less than k, the player A moves to $\mathrm{b_i}$ for any $i \in \Sigma^k$. The player B has no valid move. The player A wins. Therefore, the player A has a winning strategy starting from (α, x) with $x \in L_{\mathcal{P}}$.

Let x be a word in $\Sigma^\star \setminus L_{\mathcal{P}}$. A winning strategy for the player B starting from the A-configuration (α, x) works as follows. In this game the player B has no choice so his strategy is to play when he can. The only possibility for the player B to have no valid move is to play from some position $\mathrm{b_i}$ obtained from a position α with a word of size less than k. Observe that the only possible sequences of moves going from a configuration (α, y) to a configuration (α, z) imply that in the tag system there is a valid sequence of transitions either from z to y or from y to z. Thus as in the tag system x has an infinite sequence of valid transitions the position (α, x) is winning for the player B. Therefore, the player B has a winning strategy starting from (α, x) with $x \in \Sigma^\star \setminus L_{\mathcal{P}}$. ■

4 Removing states

In order to transform unfair rooted oriented cutenation games into commutation game, the first step is to transform the state sets $\mathfrak{A}$ and $\mathfrak{B}$ into singletons and to ensure that $L = R$. This is done by choosing a proper encoding of every configuration $((s,t),x)$ into a proper word $\langle s, x, t\rangle$.

4.1 Encoding states

Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, V_B)$ be an unfair rooted oriented cutenation game. We encode each configuration $((s,t),x)$ into a proper word $\langle s, x, t\rangle$ using the following encoding.

Let m be the size of S_L^- and Γ_L^- be an alphabet of $m-1$ ordered new letters $\{\alpha_1, \ldots, \alpha_{m-1}\}$. Let ρ map S_L^- into $\{0, 1, \ldots, m-1\}$ so that the left root is mapped into 0. Let φ_L^- map $s \in S_L^-$ into the word $\alpha_{\rho(s)} \cdots \alpha_2\alpha_1$ of size $\rho(s)$. The encoding $\varphi_L(s)$ of a state $s \in S_L$ is equal to $\varphi_L^-(s)$ when $s \in S_L^-$. Let n be the size of S_L^+ and Γ_L^+ be an alphabet of n ordered new letters $\{\beta_1, \ldots, \beta_n\}$. Let σ map S_L^+ into $\{1, \ldots, n\}$. Let φ_L^+ map $s \in S_L^+$ into the word $\beta_{\sigma(s)}\alpha_{m-1} \cdots \alpha_1$ of size m. The encoding $\varphi_L(s)$ of a state $s \in S_L$ is equal to $\varphi_L^+(s)$ when $s \in S_L^+$. For each pair of states $(s, s') \in S_L^- \times S_L^+$ define $\phi_L(s, s')$ as $\varphi_L^+(s')\varphi_L^-(s)^{-1}$, which is $\beta_{\sigma(s')}\alpha_{m-1} \cdots \alpha_{\rho(s)+1}$. Notice that $\phi_L(s, s') \in \Gamma_L^+ \left(\Gamma_L^-\right)^\star$.

Symmetrically, let m' be the size of S_R^- and Γ_R^- be an alphabet of $m' - 1$ ordered new letters $\{\gamma_1, \ldots, \gamma_{m'-1}\}$. Let ρ' map S_R^- into $\{0, 1, \ldots, m'-1\}$ so that the right root is mapped into 0. Let φ_R^- map $t \in S_R^-$ into the word $\gamma_1\gamma_2 \cdots \gamma_{\rho'(t)}$ of size $\rho'(t)$. The encoding $\varphi_R(t)$ of a state $t \in S_R$ is equal to $\varphi_R^-(t)$ when $t \in S_R^-$. Let n' be the size of S_R^+ and Γ_R^+ be an alphabet of n' ordered new letters $\{\delta_1, \ldots, \delta_{n'}\}$. Let σ' map S_R^+ into $\{1, \ldots, n'\}$. Let φ_R^+ map

$t \in S_R^+$ into the word $\gamma_1 \cdots \gamma_{m'-1}\delta_{\sigma'(t)}$ of size m'. The encoding $\varphi_R(t)$ of a state $t \in S_R$ is equal to $\varphi_R^+(t)$ when $t \in S_R^+$. For each pair of states $(t, t') \in S_R^- \times S_R^+$ define $\phi_R(t, t')$ as $\varphi_R^-(t)^{-1}\varphi_R^+(t')$, which is $\gamma_{\rho'(t)+1} \cdots \gamma_{m'-1}\delta_{\sigma'(t')}$. Notice that $\phi_R(t, t') \in \left(\Gamma_R^-\right)^\star \Gamma_R^+$.

Let τ_L and τ_R be the two morphisms from $\Sigma^\star$ to $(\Sigma \cup \{o\})^\star$, where o is a new letter, defined for each letter $a \in \Sigma$ by $\tau_L(a) = oa$ and $\tau_R(a) = ao$. For each word $x \in \Sigma^\star$ define $\tau(x)$ as $\tau_L(x)o$, which is equal to $o\tau_R(x)$.

A configuration $((s, t), x) \in (S_L \times S_R) \times \Sigma^\star$ of the game will be encoded by the word $\varphi_L(s)\tau(x)\varphi_R(t)$ denoted as $\langle s, x, t\rangle$. The set L will be encoded using the mapping ψ_L defined by $\psi_L((s, t), (s', t), x) = \phi_L(s, s')\tau_L(x)$. Symmetrically, The set R will be encoded using the mapping ψ_R defined by $\psi_R((s, t), (s, t'), y) = \tau_R(y)\phi_R(t, t')$.

Proposition 2. *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be an unfair rooted oriented cutenation game. Let V'_A and V'_B be respectively the sets $\{\langle s, x, t\rangle : x \in V_A((s, t))\}$ and $\mathrm{Sub}(\{\langle s, x, t\rangle : x \in \Sigma^\star, (s, t) \in S_L^+ \times S_R^+\})$. Let $((s, t), x)$ be a configuration of the game. There exists a valid move from the configuration $\langle s, x, t\rangle$ to a configuration w in the cutenation game $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, \psi_L(L), \psi_R(R), V'_A, V'_B)$ if and only if $w = \langle s', y, t'\rangle$ for some s', y, t' and the move from $((s, t), x)$ to $((s', t'), y)$ is valid in the first game.*

Proof. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be an unfair rooted oriented cutenation game. Let $((s, t), x)$ be a configuration of the game.

Let $((s', t'), y)$ be a configuration of the game such that a move from $((s, t), x)$ to $((s', t'), y)$ is valid. Let $w = \langle s', y, t'\rangle$. If the move is a l-move for the player A then $t = t'$ and $((s, t), (s', t), z) \in L$ where $y = zx$ and thus $\phi_L(s, s')\tau_L(z) \in \psi_L(L)$. To prove that this move is a valid l-move in the new game, it is sufficient to show that $\phi_L(s, s')\tau_L(z)\langle s, x, t\rangle = \langle s', y, t'\rangle$. If (s, t) is the left root then $\varphi_L(s) = \varepsilon$ and $\phi(s, s') = \varphi_L(s')$ thus $\phi_L(s, s')\tau_L(z)\langle s, x, t\rangle = \varphi_L(s')\tau(zx)\varphi_R(t)$. If (s, t) is not the left root then $z = \varepsilon$ and $\phi_L(s, s')\tau_L(z)\langle s, x, t\rangle = \varphi_L(s')\tau(x)\varphi_R(t)$ as $\phi_L(s, s')\varphi_L(s) = \varphi_L(s')$. Therefore, if the move is valid l-move for the player A in the original game then it is a valid l-move for the player A in the new game. The three other cases works on the same principle (don't forget to check with V_A in the case of a move for the player B).

Let w be a word such that there is a valid move for the player A in the new game from $\langle s, x, t\rangle$ to w where $(s, t) \in \mathfrak{A}$. If it is a l-move there exists some s', s'' and z such that $\phi_L(s', s'')\tau_L(z) \in \psi_L(L)$ and $w = \phi_L(s', s'')\tau_L(z)\langle s, x, t\rangle$. As $w \in V'_B$ and both $s'' \in S_L^+$ and $t \in S_R^+$ then $w = \langle s'', y, t\rangle$ for some $y \in \Sigma^\star$. This implies that $s = s'$ and $y = zx$. To prove that there is a valid l-move in the original game for the player A from the configuration $((s, t), x)$ to the configuration $((s'', t), zx)$ it is sufficient to show that $(s'', t) \in \mathfrak{B}$ and $((s, t), (s'', t), z) \in L$. As $\phi_L(s, s'')\tau_L(z) \in \psi_L(L)$ there exists some t' such that $((s, t'), (s'', t'), z) \in L$. As the original game is separated and both $(s, t) \in \mathfrak{A}$ and $(s, t') \in \mathfrak{A}$ then $(s'', t) \in \mathfrak{B}$ and $((s, t), (s'', t), z) \in L$. The case of a r-move for the player A works symmetrically.

Let w be a word such that there is a valid move for the player B in the new game from $\langle s, x, t\rangle$ to w where $(s,t) \in \mathfrak{B}$. If it is a l-move then there exists some s', s'' and z such that $\phi_L(s', s'')\tau_L(z) \in \psi_L(L)$ and $\phi_L(s', s'')\tau_L(z)w = \langle s, x, t\rangle$. As $w \in V'_A$ and both $s \in S_L^+$ and $t \in S_R^+$ then $s'' = s$ and $w = \langle s', y, t\rangle$ for some $(s', t) \in \mathfrak{A}$ and $y \in V_A((s', t))$ such that $zy = x$. To prove that there is a valid l-move in the original game for the player B from the configuration $((s,t), zy)$ to the configuration $((s', t), y)$ it is sufficient to show that $(s', t) \in \mathfrak{A}_L$ and $((s', t), (s, t), z) \in L$. As $\phi_L(s', s)\tau_L(z) \in \psi_L(L)$ there exists some t' such that $((s', t'), (s, t'), z) \in L$. As the original game is separated and both $(s', t) \in \mathfrak{A}$ and $(s', t') \in \mathfrak{A}$ then $((s', t), (s, t), z) \in L$. The case of a r-move for the player B works symmetrically. ■

4.2 Enforcing symmetry

In a commutation game both sets of left moves L and right moves R are equal. If the sets V_A and V_B bring enough constraints to the game both L and R can be replaced by $L \cup R$ to enforce this symmetry.

Proposition 3. *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, L, R, V_A, V_B)$ be a cutenation game. Let X be the language $L \cup R$. If the four sets $RV_A \cap V_B$, $V_A L \cap V_B$, $R^{-1}V_B \cap V_A$, and $V_B L^{-1} \cap V_A$ are empty then the valid moves, both the player A and the player B, are the same in the given game and in the cutenation game $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$.*

Proof. Every move in the original game is allowed in the new game. Conversely, let $x \vdash_{A,l} y$ be a valid left move for the player A in the new game. There exists $z \in X$ such that $y = zx$ so $y \in XV_A \cap V_B$. As $RV_A \cap V_B$ is empty, then $z \in L$ and the move is also valid in the original game. The three remaining cases are similar using the three other empty sets (use $V_A L \cap V_B$ for $\vdash_{A,r}$, use $R^{-1}V_B \cap V_A$ for $\vdash_{B,l}$, and use $V_B L^{-1} \cap V_A$ for $\vdash_{B,r}$). ■

The construction to enforce symmetry can be applied directly after encoding the states as the new encoding verifies the required hypothesis.

Lemma 4. *Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be some unfair rooted oriented cutenation game. Let V'_A and V'_B be defined has in proposition 2. Let X be the set $\psi_L(L) \cup \psi_R(R)$. The valid moves for both the player A and the player B are the same in both games $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, \psi_L(L), \psi_R(R), V'_A, V'_B)$ and $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V'_A, V'_B)$.*

Proof. By proposition 3 it is sufficient to show that the four sets $\psi_R(R)V'_A \cap V'_B$, $V'_A\psi_L(L) \cap V'_B$, $\psi_R(R)^{-1}V'_B \cap V'_A$, and $V'_B\psi_L(L)^{-1} \cap V'_A$ are empty.

$\psi_R(R)V'_A$ does not intersect V'_B because every word of $\psi_R(R)V'_A$ contains an occurence of a letter in Γ_R^+ before a letter o and this is never the case in V'_B. A symmetrical proof works for $V'_A\psi_L(L) \cap V'_B$.

$\psi_R(R)^{-1}V'_B$ does not intersect V'_A because $\psi_R(R)^{-1}V'_B$ only contains the empty word which is not in V'_A. The same holds for $V'_B\psi_L(L)^{-1}$. ■

5 Removing constraints

In order to conclude the construction the constraints sets V_A and V_B must be removed. This part is the core of the proof. The construction proceeds in two steps : first we remove V_A through *checking*, then we remove V_B through *flooding*.

5.1 Checking

To remove V_A means to remove constraints on the positions at the end of a move from the player B. To ensure that the set of winning strategies of the player B does not grow, the idea is to allow the player A to challenge the player B if he plays outside of V_A by *checking* the validity of the move.

Proposition 4. *Let* $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$ *be a cutenation game over the alphabet* Σ *with associated language* $\mathfrak{L}$*. Let* X' *and* V'_B *be respectively the languages* $X \cup cV_A^\star \cup V_A^\star c$ *and* $V_B \cup cV_B \cup V_B c$*. Let* $\left(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X', X', (\Sigma \cup \{c\})^\star, V'_B\right)$ *be the cutenation game over the alphabet* $\Sigma \cup \{c\}$ *with associated language* $\mathfrak{L}'$ *where* c *is a new letter not in* Σ*. If the four sets* $X^{-1}X^{-1}V_B$*,* $V_B X^{-1}X^{-1}$*,* $\left(\left(X^{-1}\left(V_A X \cap V_B\right)\right) \setminus V_A\right) \cap V_A^\star$*, and* $\left(\left(\left(XV_A \cap V_B\right)X\right)^{-1} \setminus V_A\right) \cap V_A^\star$ *are empty and both inclusions* $X^{-1}V_B \subseteq V_B$ *and* $V_B X^{-1} \subseteq V_B$ *hold then* $\mathfrak{L}$ *is equal to* $\mathfrak{L}'$*.*

Proof. We prove that the player B has a winning strategy in the original game if and only if the player B has a winning strategies in the new game.

If the player B had a winning strategy in the original game starting from a given position then he keeps playing according to its original strategy as long as the player A keeps using moves that were valid in the original game. If the player A uses a new move from a position $x \in V_A$ then there are two possibilities:

- either he catenates a word of X leading to a new position in $V'_B \setminus V_B$; this is impossible as all the new valid positions must contain the new letter c which does not appear in X;
- or he catenates a word of $X' \setminus X$ containing the new letter c leading to a new valid position y which must be either in cV_B or in $V_B c$; as $x \in V_A$ and as $X' \setminus X = cV_A^\star \cup V_A^\star c$, necessarily $y \in cV_A^\star \cup V_A^\star c$ and thus $y \in X'$.

A winning strategy for the player B starting from a position in X' is simply to cut x completely thus accessing to the empty word position. The empty word position is winning for the player B: when the player A catenates a word x the player B just cuts x coming back to the empty word. Therefore, the player B still has a winning strategy in the new game.

If the player A had a winning strategy in the original game starting from a given position then he keeps playing according to its original strategy as long as the player B keeps using moves that were valid in the original game. If the player B uses a new move from a position $x \in V_B$ then there are two possibilities:

- either he cuts a word of $X' \setminus X$ containing the new letter c; this is impossible as the new letter c does not appear in V_B;

– or he cuts a word of X leading to a new valid position in $\Sigma^\star \setminus V_A$, more precisely in $\left(\left(X^{-1}\left(V_A X \cap V_B\right)\right) \cup \left(\left(X V_A \cap V_B\right) X^{-1}\right)\right) \setminus V_A$.

A winning strategy for the player A starting from a position y in the language $\left(X^{-1}\left(V_A X \cap V_B\right)\right) \setminus V_A$ is simply to catenates the word c on the right leading to the valid position yc in $V_B c$. As $y \notin V_A^\star$ and $X^{-1}X^{-1}V_B = \emptyset$ the player B has not valid move starting from yc, thus the player A wins. By a symmetrical argument, the player A has a winning strategy starting from a position in $\left(\left(X V_A \cap V_B\right) X^{-1}\right) \setminus V_A$. Therefore, the player A still has a winning strategy in the game. ■

5.2 Flooding

To remove V_B means to remove constraints on the positions at the end of a move from the player A. To ensure that the set of winning strategies of the player A does not grow, the idea is to ensure that every position outside of V_B admits a winning strategy for the player B by *flooding* the language with every word outside of V_B.

Proposition 5. *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, \Sigma^\star, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathfrak{L}$. If V_B is closed by subword then the centralizer $\mathcal{C}(X \cup \Sigma^\star \setminus V_B)$ is equal to $\mathfrak{L}$.*

Proof. We prove that the player B has a winning strategy in the game if and only if the player B has a winning strategies in the commutation game of $X \cup \Sigma^\star \setminus V_B$.

If the player A had a winning strategy in the original game starting from a given position then he keeps playing according to its original strategy. As V_B is closed by subword, starting from a word in V_B the player B cannot use a transition in $\Sigma^\star \setminus V_B$ to cut: the player B has exactly the same possible moves as in the original game. Therefore, the player A still has a winning strategy in the new game.

If the player B had a winning strategy in the original game starting from a given position then he keeps playing according to its original strategy as long as the player A keeps using moves that were valid in the original game. If the player A use a new move then, as V_B is closed by subword, just after this move the new position is a word in $\Sigma^\star \setminus V_B$. A winning strategy for the player B starting from a word x in $\Sigma^\star \setminus V_B$ is simply to cut x completely thus accessing to the empty word position. The empty word position is winning for the player B: when the player A catenates a word x the player B just cuts x coming back to the empty word. Therefore, the player B still has a winning strategy in the new game. ■

To combine both checking and flooding to remove the constraints on a game it is sufficent to ensure that the original constraints V_B are closed by subword.

Lemma 5. *Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V_A, V_B)$ be a cutenation game over the alphabet Σ with associated language $\mathfrak{L}$ satisfying the hypothesis of the checking proposition. If the language V_B is closed by subword then $\mathfrak{L}$ is equal to the centralizer $\mathcal{C}\left(X \cup cV_A^\star \cup V_A^\star c \cup \left(\Sigma^\star \cup \{c\}\right) \setminus \left(V_B \cup cV_B \cup V_B c\right)\right)$.*

Proof. If V_B is closed by subword, so is $V'_B = V_B \cup cV_B \cup V_Bc$. ■

6 Gluing all together

We can now prove the main result of this article by combining the three parts of the construction together.

Theorem 1. *There exists a rational language X the centralizer $\mathcal{C}(X)$ of which is complete for co-recursively enumerable languages.*

Proof. Let $\mathcal{P}$ be a Post tag sytem with a language complete for recursively enumerable languages. Let $(\mathfrak{A}, \mathfrak{B}, L, R, V_A, \Sigma^\star)$ be the unfair rooted oriented cutenation game obtained by proposition 1 and such that for some $\mathfrak{a} \in \mathfrak{A}$ the equation $\mathfrak{L}(\mathfrak{a}) = \Sigma^\star \setminus L_{\mathcal{P}}$ holds. Let $(\{\mathfrak{a}\}, \{\mathfrak{b}\}, X, X, V'_A, V'_B)$ be the cutenation game over the alphabet Σ' obtained by proposition 2 and lemma 4 such that the equation $\mathfrak{L} \cap \langle s, \Sigma^\star, t\rangle = \langle s, \Sigma^\star \setminus L_{\mathcal{P}}, t\rangle$ holds for some $(s,t) \in \mathfrak{A}$. To combine this cutenation game with lemma 5, as V'_B is closed by subword it is sufficient to show that the hypothesis of proposition 4 are satisfied. More precisely it is sufficient to show that the four sets $X^{-1}X^{-1}V'_B$, $V'_BX^{-1}X^{-1}$, $\left(\left(X^{-1}\left(V'_AX \cap V'_B\right)\right) \setminus V'_A\right) \cap V'^\star_A$, and $\left(\left(\left(XV'_A \cap V'_B\right)X^{-1}\right) \setminus V'_A\right) \cap V'^\star_A$ are empty and both inclusions $X^{-1}V'_B \subseteq V'_B$ and $V'_BX^{-1} \subseteq V'_B$ hold. Both inclusions hold because V'_B is closed by subword.

The set $X^{-1}X^{-1}V'_B$ is empty because first $R^{-1}V'_B$ only contains the empty word and X does not contain the empty word, secondly because $L^{-1}V'_B$ contains the empty word and words which begins with a letter in Γ^-_L and words in X never begins with such a letter. Symmetrically, the set $V'_BX^{-1}X^{-1}$ is empty.

The set $\left(\left(X^{-1}\left(V'_AX \cap V'_B\right)\right) \setminus V'_A\right) \cap V'^\star_A$ is empty because the only words in the language $X^{-1}\left(V'_AX \cap V'_B\right)$ are of the kind $\langle s, x, t\rangle$ for some $s \in S^-_L$, $x \in \Sigma^\star$ and $t \in S^+_R$ while the only words of $V'^\star_A$ which are of the shape $\langle s', y, t'\rangle$ for some $s' \in S_L$, $t' \in S_R$ and $y \in \Sigma^\star$ are either in V'_A either in $V'_AV'_A$ but in $V'_AV'_A$ necessarily $s' \in S^+_L$ and $t' \in S^+_R$. Therefore the set is empty. Symmetrically, the set $\left(\left(\left(XV'_A \cap V'_B\right)X^{-1}\right) \setminus V'_A\right) \cap V'^\star_A$ is empty.

Therefore, the lemma 5 can be applied and $\Sigma^\star \setminus L_{\mathcal{P}}$ can be recursively computed from the centralizer of ther rational set $X \cup cV'^\star_A \cup V'^\star_Ac \cup (\Sigma'^\star \cup \{c\}) \setminus (V'_B \cup cV'_B \cup V'_Bc)$. As a consequence, the centralizer of this rational language is complete for co-recursively enumerable languages. ■

References

[1] C. Choffrut, J. Karhumäki, and N. Ollinger, The commutation of finite sets: a challenging problem, *Theoret. Comput. Sci.*, **273**(2002), 69–79.

[2] J. H. Conway, *Regular Algebra and Finite Machines*, Chapman Hall, 1971.

[3] T. Harju, O. Ibarra, J. Karhumäki, and A. Salomaa, Decision questions in semilinearity and commutation, *J. Comput. Syst. Sci.*, **65**(2002), 278–294.

[4] J. Karhumäki, Challenges of commutation: an advertisement, in *Proc. of FCT 2001*, volume 2138 of *LNCS*, pages 15–23, Springer, 2001.
[5] J. Karhumäki, M. Latteux, and I. Petre, The commutation with codes and ternary sets of words, in *Proc. of STACS 2003*, volume 2607 of *LNCS*, pages 74–84, Springer, 2003.
[6] J. Karhumäki and I. Petre, Conway's problem for three-word sets, *Theoret. Comput. Sci.*, **289**(2002), 705–725.
[7] J. Karhumäki and I. Petre, Two problems on commutation of languages, in G. Rozenberg and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, World Scientific, 2004.
[8] M. Kunc, Regular solutions of language inequalities and well quasi-orders, in *Proc. of ICALP 2004*, LNCS, Springer, 2004.
[9] M. Kunc, The power of commuting with finite sets of words, in *Proc. of STACS 2005*, Springer, 2005.
[10] M. Minsky, *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.
[11] A. Okhotin, Decision problems for language equations with boolean operations, in *Proc. of ICALP 2003*, LNCS, Springer, 2003.
[12] I. Petre, *Commutation Problems on Sets of Words and Formal Power Series*, Ph.D. thesis, University of Turku, 2002.
[13] B. Ratoandramanana, Codes et motifs, *RAIRO Theor. Informat.*, **23**(1989), 425–444.
[14] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.