

Taking-and-merging games as rewrite games*

Eric Duchêne¹ Victor Marsault² Aline Parreau¹ Michel Rigo³

¹ LIRIS, Université Claude Bernard Lyon 1, CNRS, France

² LIGM, Université Gustave Eiffel, CNRS, France

³ Department of Mathematics, University of Liège, Belgique

received 21st Feb. 2019, revised 26th May 2020, accepted 3rd Aug. 2020.

This work is a contribution to the study of rewrite games. Positions are finite words, and the possible moves are defined by a finite number of local rewriting rules $\{u_i \rightarrow v_i\}_{i \in I}$: a move consists in the substitution of one occurrence of u_i by v_i , for some i . We introduce and investigate taking-and-merging games, that is, where each rule is of the form $a^k \rightarrow \varepsilon$. We give sufficient conditions for a game to be such that the losing positions (resp. the positions with a given Grundy value) form a regular language or a context-free language. We formulate several related open questions in parallel with the famous conjecture of Guy about the periodicity of the Grundy function of octal games.

Finally we show that more general rewrite games quickly lead to undecidable problems. Namely, it is undecidable whether there exists a winning position in a given regular language, even if we restrict to games where each move strictly reduces the length of the current position.

Keywords: Combinatorial game theory; rewrite games; Grundy values; regular languages; context-free languages; taking-and-merging games.

1 Introduction

Waldmann [9] introduces general *rewrite games* as follows. Let A be a finite alphabet, i.e., a finite set of symbols. We let A^* denote the set of finite words over A . The empty word is denoted by ε . A rewrite system is given by a (finite) set $R \subset A^* \times A^*$ of rules, called R -reductions, of the form $u \rightarrow v$. The latter rule can be applied to the word $w = xuy$, $x, y \in A^*$ where we replace one occurrence of u by v and we write $w \rightarrow_R xvy$. We consider only *terminating* rewrite systems, that is, such that there is no infinite chain of R -reductions starting from a given word. In the rewrite game associated with R , the positions are the words in A^* , and from a position w the possible moves are those that lead to each word w' such that $w \rightarrow_R w'$. Starting from a word, also called ground term, $t_1 \in A^*$, two players apply alternatively an R -reduction of their choice to get a sequence $t_1 \rightarrow_R t_2 \rightarrow_R t_3 \rightarrow_R \dots \rightarrow_R t_n$ until no R -reduction can be applied. The first player unable to apply an R -reduction, because t_n is in normal form (i.e., irreducible), loses the game (t_n is called a *final position* of the game).

Rewrite games belong to the family of impartial combinatorial games. In an impartial combinatorial game, two players move alternatively with perfect information, and the set of valid moves depends only

*Supported by the ANR-14-CE25-0006 project of the French National Research Agency and the CNRS PICS-07315 project.

on the position. The first player unable to move loses the game. A complete definition of combinatorial games can be found in [6]. Taking-and-breaking games are famous examples of combinatorial games. A position consists in several piles of tokens, and a move consists in removing some tokens from a pile, and then splitting that pile into smaller piles. A major issue when studying combinatorial games is the computation of the *outcome*. An impartial combinatorial game position has outcome \mathcal{N} if the player who starts has a winning strategy, and \mathcal{P} otherwise.

The notion of *Grundy value* (also called *Sprague–Grundy value*) is a refinement of the one of *outcome*: the position with outcome \mathcal{P} are exactly those whose Grundy value is 0. More precisely, the Grundy value of any position is recursively defined as the mex (minimum excluded value) of the set of Grundy values of the position reachable in one move. For example, $\text{mex}\{0, 1, 3\} = 2$, and by convention $\text{mex}\emptyset = 0$.

A background motivation for this work stems from octal games. They are a well-known family of combinatorial games that can be described as rewrite games. They are the taking-and-breaking games in which it is never allowed to split into more than two piles. An octal game is defined by its valid moves, which may be coded by a (finite or infinite) sequence of integers that are less than or equal to 7; see [6] for a formal definition. Octal games can be translated as rewrite games as follows. If we have r piles of token with respectively n_1, \dots, n_r tokens, then a position in such a game can be coded by the word over a two-letter alphabet

$$\mathbf{ba}^{n_1}\mathbf{ba}^{n_2}\mathbf{b}\dots\mathbf{ba}^{n_r}\mathbf{b}.$$

The \mathbf{b} 's play the role of separators between piles of \mathbf{a} 's and one has to carefully choose the convenient reductions to code the game of interest, see [9, Prop. 3].

Example 1. *Let us consider the game over the alphabet $A = \{\mathbf{a}, \mathbf{b}\}$, associated with the rewrite system $R = \{\mathbf{a} \rightarrow \varepsilon, \mathbf{aa} \rightarrow \varepsilon, \mathbf{aa} \rightarrow \mathbf{b}\}$. An example of sequence of play for this game, starting from the position $t_1 = \mathbf{baaabaab}$, is*

$$\mathbf{baaabaab} \rightarrow_R \mathbf{baaabab} \rightarrow_R \mathbf{babab} \rightarrow_R \mathbf{bbab} \rightarrow_R \mathbf{bbb}.$$

In this example, four moves have been played, hence the second player wins the game. Note that this game exactly corresponds to the octal game 0.37. Indeed the piles are the block of one or more consecutive \mathbf{a} 's. A player can remove one token of a pile, possibly emptying it, by using move $\mathbf{a} \rightarrow \varepsilon$. A player can also remove two tokens from a pile, possibly emptying it, by applying $\mathbf{aa} \rightarrow \varepsilon$. Finally, a player can remove two tokens from a pile and divide the remaining tokens into two piles, by applying $\mathbf{aa} \rightarrow \mathbf{b}$ in the middle of a block of \mathbf{a} 's.

The *Grundy sequence* of an octal game is defined as the integer sequence where the i -th element is the Grundy value of the position with one pile of i tokens. We may then reformulate a famous conjecture in combinatorial game theory:

Conjecture 2 (Guy's conjecture [2]). *All finite octal games have an eventually periodic Grundy sequence.*

In the context of a rewrite game G , positions are words over a finite alphabet A and we can associate a Grundy value $\mathcal{G}(w)$ with each word w in A^* . Thus, the family of languages $(\mathcal{L}_i)_{i \in \mathbb{N}}$, defined by $\mathcal{L}_i = \mathcal{G}^{-1}(i)$, is a partition of A^* ; they are called the *Grundy languages* of G . In his paper [9], Waldmann makes a correspondence between the regularity of the Grundy languages of octal games (seen as rewrite games) and the periodicity of the Grundy sequence.

Theorem 3 (Waldmann, 2002). *The Grundy sequence of an octal game is eventually periodic if and only if it has only finitely many non-empty Grundy languages \mathcal{L}_i , all of which are regular languages.*

This nice result translates the notion of periodicity of a taking-and-breaking game into the context of rewrite games. Therefore, the question of the regularity of rewrite games becomes paramount, and in particular would allow to make progress towards proving or disproving Conjecture 2. This leads to the general open question below, which we start to address in this article.

Question 4. *Which rewrite games have Grundy values bounded by a constant K and such that all the languages $\mathcal{L}_0, \dots, \mathcal{L}_K$ are regular?*

The following classical lemma (see [2]) characterizes the Grundy languages of a rewrite game; and will be heavily used throughout this article.

Lemma 5. *Given a rewrite game G over an alphabet A , the family $(\mathcal{L}_i)_{i \in \mathbb{N}}$ is the only family of languages $(\mathcal{M}_i)_{i \in \mathbb{N}}$ that satisfies:*

- *for all $i \in I$, every move from words in \mathcal{M}_i leads to a word outside of \mathcal{M}_i (stability property),*
- *for every $i \in \mathbb{N}$, every word $u \in \mathcal{M}_i$, and every $j < i$, there exists a move from u leading to a word in \mathcal{M}_j (absorption property).*

In addition to octal games, some other well-known games have also been considered in the context of rewrite games. It is for example the case of *Peg-solitaire* [4, 5], where R is of the form $\{aab \rightarrow bba, baa \rightarrow aab\}$. In *Peg-solitaire*, it has been proved that on one dimensional boards, the set of solvable configurations forms a regular language. In the 2-player version of the game, called *DUOTAIRE*, where series of hops can be done in a single move, neither the \mathcal{P} nor the \mathcal{N} positions form a regular nor even a context-free language. Another example of a combinatorial game seen as a rewrite game is the game *CLOBBER* [1], played over a 3-letter alphabet $\{a, b, \emptyset\}$ with $R = \{ab\emptyset \rightarrow \emptyset\emptyset a, ba\emptyset \rightarrow \emptyset\emptyset b\}$.

In the following, we assume that the reader is familiar with basic results about formal languages and combinatorial games. We refer the reader respectively to [3] and [6] for a general reference on these topics. For any given letter a and word w , we let $|w|_a$ denote the number of a occurring in the word w . We denote by ε the empty word.

Taking-and-merging games

In most of this article, we consider a family of rewrite games over a two-letter alphabet, say $\{a, b\}$, where any reduction rule of R is either of the form $a^k \rightarrow \varepsilon$ or $b^k \rightarrow \varepsilon$ for some k . In a certain way, this family allows us to model a new kind of pile games, where taking moves are combined with merging ones. For example, by following Waldmann's description of octal games with a rewrite system, playing $b \rightarrow \varepsilon$ from the word aba^5 leads to a^6 and can be seen as a merging of the piles a and a^5 .

From now on and for the sake of notation, we will omit the reduction to ε in the description of the rewrite system. In other words, the games considered here will be denoted by a set

$$\{a^{k_1}, a^{k_2}, \dots, a^{k_n}, b^{\ell_1}, b^{\ell_2}, \dots, b^{\ell_m}\}$$

where the k_i and ℓ_i are positive integers.

We now consider a first example of such a taking-and-merging game. Using a convenient invariant (denoted by S) is a strategy that will appear in several proofs encountered in this paper.

Example 6. Let us consider the game $G = \{a^2, b\}$. We claim that the DFA (deterministic finite automaton) depicted in Figure 1 computes the Grundy function of G : consider a word w and start reading it from the initial state marked with an incoming arrow. Follow transitions reading the word letter by letter from left to right and look at the state reached when reading the last letter of w . The states (0.0) and (0.1) correspond to the words of Grundy value 0, and the states (1.2) and (1.3) to those of Grundy value 1. First, note that this is true for the two final positions ε and a . To prove this result, we define the following quantity for a given word u .

$$S(u) = (|u|_a - 2|u|_b) \bmod 4$$

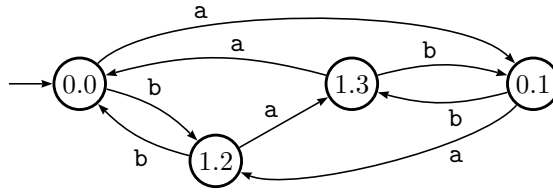


Fig. 1: A DFA computing the Grundy function of the game $\{a^2, b\}$.

One can first observe that for all $i = 0, \dots, 3$, every word u recognized by the state $(X.i)$ (for $X \in \{0, 1\}$) satisfies $S(u) = i$. To check this property, it suffices to consider each transition of the DFA and verify that $S(u)$ changes accordingly. For example, reading a letter a from the state (1.2) increases by 1 the value of $S(u)$, leading to the state (1.3) , while reading a letter b decreases by 2 the value and leads to the state (0.0) . Then, in order to prove that the DFA computes the Grundy values, by Lemma 5, it suffices to show that any move from a word recognized by a state $(0.X)$ (for $X \in \{0, 1\}$) leads to a word recognized by a state $(1.Y)$ (for some $Y \in \{2, 3\}$), and that any move from a word recognized by a state $(1.Y)$ leads to some $(0.X)$. These two properties can be easily checked by using the invariant $S(u)$:

- By definition of S , any move $a^2 \rightarrow \varepsilon$ from a word u such that $S(u) = 0, 1$ leads to a word u' having $S(u') = 2, 3$, and conversely.
- Any move $b \rightarrow \varepsilon$ satisfies the same property, as S is modified by $2 \bmod 4$.

In view of such an example and according to Guy's conjecture, it is natural to wonder whether the regularity of the languages \mathcal{L}_i would hold in the context of taking-and-merging games. In Section 2, we will give a negative answer to this question, for games where both reductions $a \rightarrow \varepsilon$ and $b \rightarrow \varepsilon$ are forbidden. In addition, a proof of context-freeness is given for simple instances of such games. In Section 3, we prove the regularity of several taking-and-merging games. In particular, we exhibit DFAs computing their Grundy functions. Section 4 deals with a discussion about a result of Waldmann about the correlation between the regularity of \mathcal{L}_0 , the other \mathcal{L}_i , and the number of Grundy values. The last section explains why we restricted our study to taking-and-merging games: in the slightly more general settings of strongly-terminating rewrite games (i.e., where each move strictly decreases the length of the

position), some problems become undecidable. Indeed, we show that then it is undecidable whether there exists a winning position in a given regular language L of starting positions.

2 Not all games lead to regular languages

Our first result shows that Guy's conjecture does not hold for taking-and-merging games. More precisely, it states that, considering any taking-and-merging game that excludes both reductions $\mathbf{a} \rightarrow \varepsilon$ and $\mathbf{b} \rightarrow \varepsilon$, the set of \mathcal{P} -positions is not a regular language.

2.1 Games $\{\mathbf{a}^{k_1}, \dots, \mathbf{a}^{k_n}, \mathbf{b}^{\ell_1}, \dots, \mathbf{b}^{\ell_m}\}$ with $k_1 > 1$ and $\ell_1 > 1$

Theorem 7. *Let G be the taking-and-merging game $\{\mathbf{a}^{k_1}, \dots, \mathbf{a}^{k_n}, \mathbf{b}^{\ell_1}, \dots, \mathbf{b}^{\ell_m}\}$, with $k_1 \leq k_2 \leq \dots \leq k_n$ and $\ell_1 \leq \ell_2 \leq \dots \leq \ell_m$. If $k_1 > 1$ and $\ell_1 > 1$, then the language of the \mathcal{P} -position of G is not regular.*

Proof: Let us show that the intersection of the set of \mathcal{P} -positions of G with the regular language L , defined below, is not a regular language.

$$L = \mathbf{b}^{\ell_1-1}(\mathbf{ab}^{\ell_1-1})^*(\mathbf{ba}^{k_1-1})^*$$

More precisely, we prove by induction that the word $u_{i,j} = \mathbf{b}^{\ell_1-1}(\mathbf{ab}^{\ell_1-1})^i(\mathbf{ba}^{k_1-1})^j$ is a \mathcal{P} -position if and only if $i \geq j$.

If $i = 0$ and $j > 0$, then there is only one valid move from position $u_{i,j}$ and it leads to position f , below.

$$u_{0,j} = \mathbf{b}^{\ell_1-1}(\mathbf{ba}^{k_1-1})^j \longrightarrow f = \mathbf{a}^{k_1-1}(\mathbf{ba}^{k_1-1})^{j-1}$$

It may be verified that f is a final position, hence that $u_{0,j}$ is a \mathcal{N} -position, for every $j > 0$. On the other hand, for every $i \geq 0$ then $u_{i,0}$ is a final position, hence a \mathcal{P} -position. In other words, the claim is true if $i = 0$ or $j = 0$.

Now, assume that $i > 0$ and $j > 0$. In that case, we denote by v the following word.

$$v = \mathbf{b}^{\ell_1-1}(\mathbf{ab}^{\ell_1-1})^{i-1}\mathbf{a}^{k_1}(\mathbf{ba}^{k_1-1})^{j-1}$$

It may be verified that only one move is valid from $u_{i,j}$, and that it leads to v . Similarly, the only move from v leads to $u_{i-1,j-1}$. Therefore, words $u_{i,j}$ and $u_{i-1,j-1}$ have the same outcome and by induction hypothesis $u_{i-1,j-1}$ is a \mathcal{P} -position if and only if $i-1 \geq j-1$, which concludes the induction. \square

2.2 Context-freeness for $\{\mathbf{a}^k, \mathbf{b}^\ell\}$

We have seen with Theorem 7 that the language made of \mathcal{P} -positions is, in general, not regular. Nevertheless, when limited to a rewrite game with two reductions, we get the following result.

Theorem 8. *Let k, ℓ be positive integers. The taking-and-merging game $\{\mathbf{a}^k, \mathbf{b}^\ell\}$ has only two Grundy values and the corresponding languages \mathcal{L}_0 and \mathcal{L}_1 are context-free.*

Proof: The rewrite system $\{\mathbf{a}^k \rightarrow \varepsilon, \mathbf{b}^\ell \rightarrow \varepsilon\}$ is *weakly confluent*, that is, if $u \rightarrow v_1$ and $u \rightarrow v_2$, then there exists a w such that $v_1 \rightarrow^* w$ and $v_2 \rightarrow^* w$ (in our case, w can be reached in at most one step). Since moreover, this rewriting system is *terminating* (i.e., there is no infinite rewriting chain), Newman's

Lemma [8] yields that the rewriting system is *confluent* or, stated otherwise, from any position u can be reached a unique final position.

Let u be a word and w be the unique final position reachable from u . If $|u|_a = n$, $|u|_b = m$, there exists $\alpha, \beta \geq 0$ such that $|w|_a = n - \alpha k$ and $|w|_b = m - \beta \ell$. This means that the reduction $\mathbf{a}^k \rightarrow \varepsilon$ (resp. $\mathbf{b}^\ell \rightarrow \varepsilon$) has been applied α (resp. β) times in a sequence of $\alpha + \beta$ reductions. Hence, playing the game starting from u necessarily consists of $\alpha + \beta$ moves. Consequently u is a \mathcal{P} -position (resp. a \mathcal{N} -position) if and only if $\alpha + \beta$ is even (resp. odd)

To compute the Grundy value of a word, one just has to apply all the possible reductions in any order and count the parity of the number of applied reductions. This can be computed by a push-down automata: reading the word from left to right, each time there are k consecutive letters \mathbf{a} or ℓ consecutive letters \mathbf{b} , a reduction is simulated and the parity changed. Let us define more formally this push-down automata. It has with three states: $0, 1$ and an initial state q_0 . The stack alphabet is

$$\{(\mathbf{a}, 1), \dots, (\mathbf{a}, k-1), (\mathbf{b}, 1), \dots, (\mathbf{b}, \ell-1), \perp\}$$

where \perp is a special symbol to represent the bottom of the stack. Transitions are of the form

$$(i, x, y, z, j)$$

where $i, j \in \{0, 1\}$ are states, x is the symbol read by the automata, y is the symbol that is popped from the top of the stack, z is the word that is then pushed on the stack (with the usual convention that the leftmost symbol is on top of the stack).

First, there is a unique transition leaving the initial states; it initializes the stack with the bottom symbol \perp without reading any letter from the input:

$$(q_0, \varepsilon, \varepsilon, \perp, 0).$$

Second, the transition table for states $q \in \{0, 1\}$ is given in Table 1.

source state	input letter	popped symbol	pushed symbols	target state	
$(q,$	$\mathbf{a},$	$\perp,$	$(\mathbf{a}, 1) \perp,$	$q)$	
$(q,$	$\mathbf{b},$	$\perp,$	$(\mathbf{b}, 1) \perp,$	$q)$	
$(q,$	$\mathbf{a},$	$(\mathbf{b}, j),$	$(\mathbf{a}, 1)(\mathbf{b}, j),$	$q)$	for each $j < \ell$
$(q,$	$\mathbf{b},$	$(\mathbf{a}, j),$	$(\mathbf{b}, 1)(\mathbf{a}, j),$	$q)$	for each $j < k$
$(q,$	$\mathbf{a},$	$(\mathbf{a}, i),$	$(\mathbf{a}, i+1),$	$q)$	if $i < k-1$
$(q,$	$\mathbf{b},$	$(\mathbf{b}, i),$	$(\mathbf{b}, i+1),$	$q)$	if $i < \ell-1$
$(q,$	$\mathbf{a},$	$(\mathbf{a}, k-1),$	$\varepsilon,$	$1-q)$	
$(q,$	$\mathbf{b},$	$(\mathbf{b}, \ell-1),$	$\varepsilon,$	$1-q)$	

Tab. 1: Transition table for states $q \in \{0, 1\}$

For each of these transitions, observe that a symbol has to be popped from the stack. We store on the stack the blocks of letters that are were read but not consumed: note that symbol $(\mathbf{a}, 5)$ means a block of

five a's. If a block of k contiguous a's is found, that is if we read a from the input and that $(a, k - 1)$ is the symbol on top of the stack, we apply $a^k \rightarrow \varepsilon$, effectively popping $(a, k - 1)$ from the stack. Moreover, the automaton goes into the other state (from 0 to 1 or 1 to 0). A similar transition is taken when a block of ℓ contiguous b's is found. In all other cases, the stack is simply updated without changing the state. When the input is entirely read, the state of the automaton is the parity of the number of reductions that have been applied. We disregard the final content of the stack; it is the final position of the game. \square

Remark 9. *In the above result, when k or ℓ is equal to 1, the two languages \mathcal{L}_0 and \mathcal{L}_1 are regular. Indeed, the stack is not needed in that case.*

Assume that $k > 1$ and $\ell = 1$. Since the order of the moves does not matter, we may assume that all the moves $b \rightarrow \varepsilon$ are played first. Then the word contains only letters a and the rule $a^k \rightarrow \varepsilon$ is played until a position a^i with $i < k$ is reached. Thus, the number of moves from a starting position u is $|u|_b + \lfloor \frac{|u|_a}{k} \rfloor$ and the Grundy value is the parity of this number. This can easily be computed by a DFA. In Figure 2, we have represented the DFA for the game $\{a^3, b\}$. (The integers in the states are the Grundy values.)

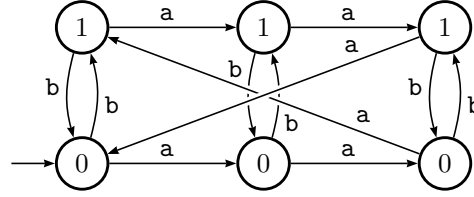


Fig. 2: The DFA computing the Grundy values of $\{a^3, b\}$.

Remark 10. *The proof of Theorem 8 generalizes to any n -letter game of the form $\{a_1^{k_1}, \dots, a_n^{k_n}\}$.*

3 Regularity of some games

In this section, we prove the regularity of some games of the form

$$G = \{a^{k_1}, a^{k_2}, \dots, a^{k_n}, b\}.$$

If the game has only two rules, $\{a^{k_1}, b\}$, the game is trivial: there are only two Grundy values and the two corresponding languages \mathcal{L}_0 and \mathcal{L}_1 are regular (see Remark 9). In the following, we consider games with at least three rules.

3.1 The game $\{a, a^{2k+1}, b\}$

In the game $\{a, a^{2k+1}, b\}$ the only irreducible word is ε (since $a \rightarrow \varepsilon$ and $b \rightarrow \varepsilon$ are moves), and all words $w \in A^*$ can be reduced to it. Let w be a word. We need $|w|_b$ reductions of the form $b \rightarrow \varepsilon$ to get rid of the b's. To get rid of all the a's, since the reduction rules all involve an odd number of a, the number of reductions to apply to eliminate the a's has the same parity as $|w|_a$. Hence, the number of reductions to apply to a word w to obtain ε is even if and only if $|w|_a + |w|_b$ is even. Let us partition A^* into two

sets \mathcal{M}_0 and \mathcal{M}_1 ; a word w belongs to \mathcal{M}_0 if $|w|_a + |w|_b$ is even and to \mathcal{M}_1 if it is odd. Lemma 5 then yields that \mathcal{M}_0 is the set of \mathcal{P} -positions and that \mathcal{M}_1 is the set of \mathcal{N} -position. It can be easily shown that these two languages are regular.

Remark 11. *The same argument extends to each game whose set of rewriting rules contains $a \rightarrow \varepsilon$, $b \rightarrow \varepsilon$ and any number of rules of the form $a^{2k+1} \rightarrow \varepsilon$ and $b^{2\ell+1} \rightarrow \varepsilon$.*

3.2 The game $\{a, a^2, b\}$

In this section, we prove that for the game $\{a, a^2, b\}$, the language \mathcal{L}_i of words of Grundy value i is regular for any Grundy value i and we explicitly give a DFA that computes the Grundy values.

Every word in A^* can be uniquely written as

$$w = a^{i_0} b a^{i_1} b \cdots b a^{i_k}$$

where $k \geq 0$ and $i_0, \dots, i_k \geq 0$. With every word w is thus associated a tuple (i_0, \dots, i_k) , with $k = |w|_b$; this association is one-to-one. For $j \in \{0, \dots, k\}$, let $i'_j := i_j \bmod 3$. For $r \in \{1, 2\}$, let $\alpha_r = \#\{j \mid i'_j = r\}$ be the number of blocks of a 's of size r (modulo 3). Finally, we define for every word w the quantity

$$S(w) = 2k + 2\alpha_1 + \alpha_2 \bmod 4.$$

As an example, the word $w = a^5 b^2 a b a^2$ has $k = 3$, $(i_0, i_1, i_2, i_3) = (5, 0, 1, 2)$ thus $\alpha_1 = 1$, $\alpha_2 = 2$ and $S(w) = 2$.

Lemma 12. *Let $w \in A^*$, the Grundy value of w in the game $\{a, a^2, b\}$ is entirely determined by $S(w)$, i.e.,*

$$\mathcal{G}(w) = \begin{cases} 0, & \text{if } S(w) = 0; \\ 1, & \text{if } S(w) = 2; \\ 2, & \text{if } S(w) = 1; \\ 3, & \text{if } S(w) = 3. \end{cases}$$

Proof: The proofs consists in showing that the conditions of Lemma 5 are met by the following family of languages: $\mathcal{M}_0 = S^{-1}(0)$, $\mathcal{M}_1 = S^{-1}(2)$, $\mathcal{M}_2 = S^{-1}(1)$, $\mathcal{M}_3 = S^{-1}(3)$ and $\mathcal{M}_i = \emptyset$, for each $i > 3$.

First, let us show that playing any move changes the value of $S(w)$ modulo 4. Let $w \in A^*$ and consider each rule.

- If the rule $a \rightarrow \varepsilon$ is played on a block a^{i_r} , then $S(w)$ decreases by 2 if $i'_r = 1$ and increases by 1 if $i'_r \in \{0, 2\}$.
- If the rule $a^2 \rightarrow \varepsilon$ is played, on a block a^{i_r} , then $S(w)$ increases by 2 if $i'_r = 0$, by 1 if $i'_r = 1$ and decreases by 1 if $i'_r = 2$.
- Finally, assume that the rule $b \rightarrow \varepsilon$ is played. Let a^{i_m} and $a^{i_{m+1}}$ be the two blocks around the b that will be removed. Table 2 gives, for every value of i'_m and i'_{m+1} , the variation of $S(w)$ modulo 4.

As an example, consider the case $i'_m = i'_{m+1} = 1$. Then one b and two blocks of size 1 (modulo 3) are lost, decreasing the value of $S(w)$ by 6, but we obtain a new block of size 2. Thus the total value $S(w)$ decreases by 5 which is congruent to 1 modulo 4. Note that if $i_m = 0$ (respectively

$i'_m \backslash i'_{m+1}$	0	1	2
0	-2	-2	-2
1	-2	-1	-1
2	-2	-1	-2

Tab. 2: Variation of $S(w)$ when the rule $\mathbf{b} \rightarrow \varepsilon$ is applied to a \mathbf{b} between two blocks of \mathbf{a} 's respectively of length i'_m and i'_{m+1} modulo 4.

($i'_{m+1} = 0$), then the number of blocks of \mathbf{a} of size 1 and 2 do not change modulo 3 and only one \mathbf{b} is removed, decreasing by 2 the value $S(w)$.

Now, let us prove that $S(w) > 0$, there is a move to w' with $S(w') = 0$. If $S(w)$ is odd, α_2 is also odd and in particular, there must be a block $\mathbf{a}^{i'_m}$ with $i'_m = 2$. Then playing $\mathbf{a}^2 \rightarrow \varepsilon$ if $S(w) = 1$ or $\mathbf{a} \rightarrow \varepsilon$ if $S(w) = 3$ on this block leads to a word w' with $S(w') = 0$. Thus assume that $S(w) = 2$. If there is a block $\mathbf{a}^{i'_m}$ with $i'_m = 1$ then playing the rule $\mathbf{a} \rightarrow \varepsilon$ on this block decreases $S(w)$ by 2. Otherwise, there must be at least one \mathbf{b} . Then, using Table 2, removing any \mathbf{b} decreases $S(w)$ by 2 since the blocks around \mathbf{b} have size 0 or 2 modulo 3.

If $S(w) \in \{1, 3\}$, then there is a move to a word w' with $S(w') = 2$. Indeed, as before, there must be a block $\mathbf{a}^{i'_m}$ with $i'_m = 2$. Then playing $\mathbf{a} \rightarrow \varepsilon$ if $S(w) = 1$ or $\mathbf{a}^2 \rightarrow \varepsilon$ if $S(w) = 3$ on this block leads to a word w' with $S(w') = 2$.

Finally, if $S(w) = 3$, there is a move to a word w' with $S(w') = 1$. We do the same reasoning than before to find a move from $S(w) = 2$ to $S(w') = 0$. If there is a block of size 1 or a \mathbf{b} next to a block of size 0, we remove the block of size 1 or \mathbf{b} . If not, we remove any \mathbf{b} between two blocks of size 2.

Hence, the conditions of Lemma 5 are indeed met by the following family of languages: $\mathcal{M}_0 = S^{-1}(0)$, $\mathcal{M}_1 = S^{-1}(2)$, $\mathcal{M}_2 = S^{-1}(1)$, $\mathcal{M}_3 = S^{-1}(3)$ and $\mathcal{M}_i = \emptyset$, for each $i > 3$. \square

Theorem 13. *The Grundy values of the game $\{\mathbf{a}, \mathbf{a}^2, \mathbf{b}\}$ can be computed by a DFA.*

Proof: By Lemma 12, we just need to compute the value $S(w)$. This is done by the automaton depicted in Figure 3. There are 12 states. A state is denoted by $(s.i)$ where s is the value $S(w)$ and i is the size modulo 3 of the last block of \mathbf{a} of w . Reading \mathbf{b} from a state $(s.i)$ leads to state $(s-2.0)$ (values are taken modulo 4 for s and modulo 3 for i). Reading \mathbf{a} from a state $(s.i)$ leads to state $(s'.(i+1))$ with $s' = s+2$ if $i = 0$, $s' = s-1$ if $i \in \{1, 2\}$. \square

What could happen if we just replace the rule $\mathbf{a}^2 \rightarrow \varepsilon$ by $\mathbf{a}^4 \rightarrow \varepsilon$? Surprisingly, we did not find an automaton for the game $\{\mathbf{a}, \mathbf{a}^4, \mathbf{b}\}$ even though the Grundy values of this game seem to be bounded as suggested by computer experiments. For words of length at most 20, the Grundy function is bounded by 3. This leads to the following open question.

Question 14. *Are the Grundy values of the game $\{\mathbf{a}, \mathbf{a}^4, \mathbf{b}\}$ bounded? Are the corresponding sets regular?*

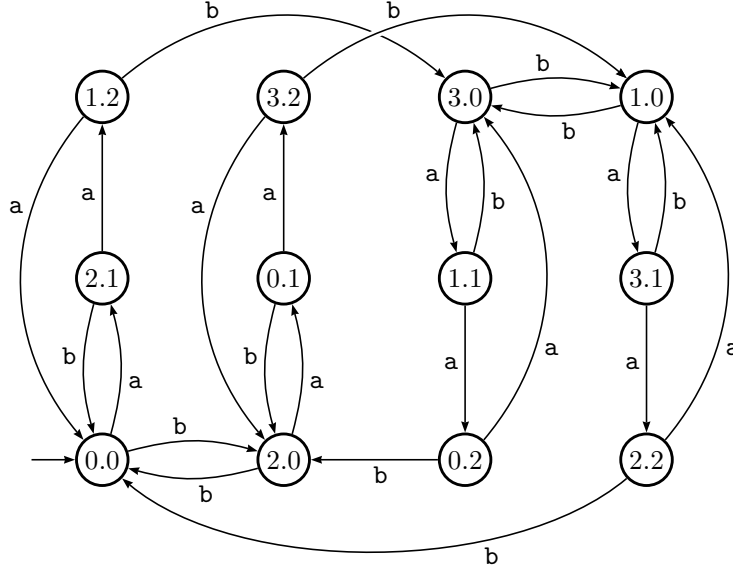


Fig. 3: A DFA for the game $\{a, a^2, b\}$.

3.3 The game $\{a, a^2, a^3, b\}$

We now prove that for the game $\{a, a^2, a^3, b\}$, the corresponding sets \mathcal{L}_i are again regular and give a DFA that computes the Grundy values. As before, every word in A^* can be written as

$$w = a^{i_0} b a^{i_1} b \cdots b a^{i_k}$$

where $k = |w|_b \geq 0$ and $i_0, \dots, i_k \geq 0$. We now write $i'_j := i_j \bmod 4$. For $r \in \{1, 2, 3\}$, let $\alpha_r = \#\{j \mid i'_j = r\}$ be the number of blocks of size r modulo 4. Finally, we define for any word the triplet of $\{0, 1\}^3$.

$$S(w) = (k + \alpha_1 \bmod 2, \alpha_2 \bmod 2, \alpha_3 \bmod 2).$$

For convenience reasons, we will denote the triplet $S(w) = (x, y, z)$ by the word xyz . As an example, the word $w = a^5 b a^3 b b a^2 b a$ has $k = 4$, $\alpha_1 = 2$, $\alpha_2 = \alpha_3 = 1$ and thus $S(w) = 011$. As before, the value $S(w)$ is enough to compute the Grundy values.

Lemma 15. *Let $w \in A^*$, the Grundy value of w in the game $\{a, a^2, a^3, b\}$ is determined by $S(w)$, i.e.,*

$$\mathcal{G}(w) = \begin{cases} 0, & \text{if } S(w) \in \{000, 111\}; \\ 1, & \text{if } S(w) \in \{011, 100\}; \\ 2, & \text{if } S(w) \in \{010, 101\}; \\ 3, & \text{if } S(w) \in \{001, 110\}. \end{cases}$$

Note that the values $S(w)$ are paired with their complement

Proof: We denote by \mathcal{M}_i , $i \in \{0, 1, 2, 3\}$ the potential candidates for \mathcal{L}_i , that are:

- $\mathcal{M}_0 = \{w \in A^* | S(w) \in \{000, 111\}\}$;
- $\mathcal{M}_1 = \{w \in A^* | S(w) \in \{011, 100\}\}$;
- $\mathcal{M}_2 = \{w \in A^* | S(w) \in \{010, 101\}\}$;
- $\mathcal{M}_3 = \{w \in A^* | S(w) \in \{001, 110\}\}$.

We aim to prove that $\mathcal{L}_i = \mathcal{M}_i$. We first list the evolution of $S(w)$ depending on the rule that is played.

- The rule $a^k \rightarrow \varepsilon$ is played on a block a^{i_r} . Table 3 gives the vector (in a compact form) that is applied to $S(w)$ in function of the values of i'_r and the rule a^k . As an example, consider the case $i'_r = 2$ and the rule $a^3 \rightarrow \varepsilon$. One block of size 2 is replaced by a block of size $2 - 3 = 3 \pmod 4$. Thus the vector applied to $S(w)$ is $(0, 1, 1)$ (values are taken modulo 2)..

$k \backslash i'_r$	0	1	2	3
1	001	100	110	011
2	010	101	010	101
3	100	110	011	001

Tab. 3: Variation of $S(w)$ with the rules $a^i \rightarrow \varepsilon$ on a block a^{i_r} .

- The rule $b \rightarrow \varepsilon$ is played. Let a^{i_m} and $a^{i_{m+1}}$ be the two blocks around the b that is removed. Table 4 gives the vector applied to $S(w)$ depending on the values of i'_m and i'_{m+1} .

$i'_m \backslash i'_{m+1}$	0	1	2	3
0	100	100	100	100
1	100	110	011	001
2	100	011	100	011
3	100	001	011	110

Tab. 4: Variation of $S(w)$ when the rule $b \rightarrow \varepsilon$ is applied to a b between two blocks of a 's respectively of length i'_m and i'_{m+1} modulo 4.

In both cases, there is no variation with vector 000 or 111 which proves that all the sets \mathcal{M}_i are stable. We now prove that for any word in \mathcal{M}_i there is a move to a word in \mathcal{M}_j if $i > j$.

First note that, except if w contains only a 's and an even number of them, it is always possible to change $S(w)$ by either vector 100 or 011. Indeed, consider such a word w . If there is a block of a 's of size 1 or 3, then playing $a \rightarrow \varepsilon$ to any a of this block changes the value of $S(w)$ by 100 or 011. Otherwise, there are only blocks of size 0 or 2 (modulo 4), and necessarily one b . Then playing $b \rightarrow \varepsilon$ to any b changes the value of $S(w)$ by 100 or 011 (according to Table 4). This remark implies that there is always a move from a word in \mathcal{M}_1 to \mathcal{M}_0 and from a word in \mathcal{M}_3 to \mathcal{M}_2 .

Second, we prove that if w belongs to $\mathcal{M}_2 \cup \mathcal{M}_3$, it is always possible to change $S(w)$ by either vector 001 or vector 110. By definition of \mathcal{M}_2 and \mathcal{M}_3 , there is always in w either a block of size 2 or a block of size 3 modulo 4. Then, using Table 3, playing $\mathbf{a} \rightarrow \varepsilon$ (in the first case) or $\mathbf{a}^3 \rightarrow \varepsilon$ (in the second case) changes the value of $S(w)$ with vector 110 (in the first case) or 001 (in the second case). This implies that there is always a move from a word in \mathcal{M}_2 to \mathcal{M}_1 and from a word in \mathcal{M}_3 to \mathcal{M}_0 .

Third we prove that if w belong to $\mathcal{M}_2 \cup \mathcal{M}_3$, it is always possible to change $S(w)$ by either vector 010 or vector 101. As before, $w \in \mathcal{M}_2 \cup \mathcal{M}_3$ must contain either a block of size 2 or a block of size 3 modulo 4. Then playing $\mathbf{a}^2 \rightarrow \varepsilon$ changes the value of $S(w)$ with vector 010 (in the first case) or 101 (in the second case). This implies that there is always a move from a word in \mathcal{M}_2 to \mathcal{M}_0 and from a word in \mathcal{M}_3 to \mathcal{M}_1 .

Lemma 5 yields that $\mathcal{M}_i = \mathcal{L}_i$ for all $i \in \{0, 1, 2, 3\}$. \square

Theorem 16. *The Grundy values of the game $\{\mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \mathbf{b}\}$ can be computed by a DFA.*

Proof: We construct a DFA that computes the Grundy values of the game $\{\mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \mathbf{b}\}$. By Lemma 15, one just needs to compute the value of $S(w)$, which, by definition of $S(w)$, can be done by an automaton that stores the value of k , α_i , $i \in \{1, 2, 3\}$ modulo 2 and the number modulo 4 of \mathbf{a} in the last block. \square

Remark 17. *In the proof of Theorem 16, we don't need to maintain $S(w)$ entirely, if all we want is the Grundy value. Indeed, it is enough to store the Grundy value and the parity of the last block of \mathbf{a} .*

This is due to the fact that, with a given parity for the last block of \mathbf{a} 's, in order to obtain $S(wx)$ from $S(w)$ for some word w and letter x , we apply some vector or its complement to $S(w)$. For instance, if w ends with an odd number of \mathbf{a} 's and that \mathbf{a} is read, then the vector applied to $S(w)$ is 110 if w ends with one \mathbf{a} and 001 if it ends with \mathbf{a}^3 . The automaton computing the Grundy values in this way is depicted in Figure 4. It has eight states and the label $(g.i)$ in a state indicates that g is the Grundy value and i the parity of the number of \mathbf{a} in the last block. As an example, from state (3.1) , when reading \mathbf{a} , $S(w)$ changes by vector 110 or 001 and thus the Grundy value that was 3 is now 0 and there are now an even number of \mathbf{a} . Hence the new state is 0.0 . When reading \mathbf{b} , $S(w)$ changes by vector 100 and thus the Grundy value is now 2 and the final letter is \mathbf{b} , thus the new state is (2.0) .

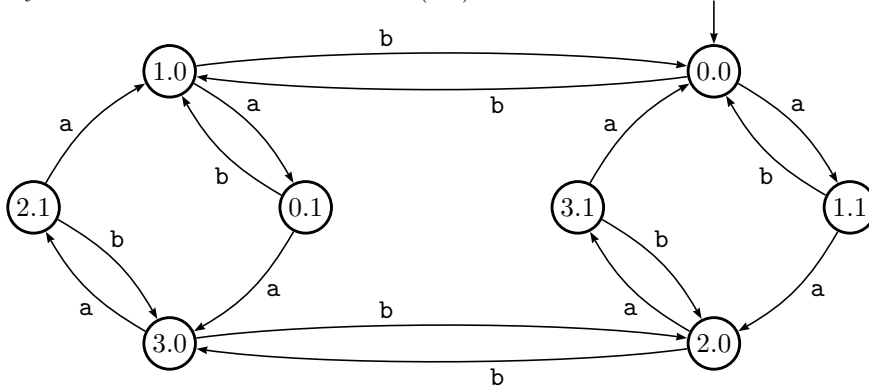


Fig. 4: DFA for the game $\{\mathbf{a}, \mathbf{a}^2, \mathbf{a}^3, \mathbf{b}\}$.

One could hope to show a similar result for each game of the form $\{\mathbf{a}, \mathbf{a}^2, \dots, \mathbf{a}^k, \mathbf{b}\}$, by computing the number of \mathbf{b} and blocks of \mathbf{a} of size 1, 2, 3, ..., $k - 1$ modulo 2 and finding some invariant for the Grundy

values. However, this method already fails for $k = 4$ since the word a^2ba^2 is a \mathcal{P} -position for this game whereas b is not. We have computed the Grundy values for all words of length up to 23 for this game and already found 14 Grundy values:

$$(\max\{\mathcal{G}(u)\})_{|u|=0,1,2,\dots} = 0, 1, 2, 3, 4, 5, 5, 6, 7, 7, 7, 7, 8, 9, 9, 10, 11, 11, 12, 13, 13, 13, 14$$

This suggests that the automaton, if it exists for this game, is not as simple as it was for $k = 2$ or $k = 3$.

4 Does a regular set of \mathcal{P} -positions imply regular sets of Grundy values?

Given a rewrite game, deciding whether each set \mathcal{L}_i forms a regular language remains an open problem in a certain number of cases. Therefore, it seems natural to know whether a positive or a negative answer can be given without considering all the sets. A first step towards this direction has been given by Waldmann, who obtained the following result [9, Thm. 6].

Theorem 18 (Waldmann, 2002). *For all taking-and-breaking games, if the language \mathcal{L}_0 is regular, then the Grundy function is bounded, and all the Grundy languages \mathcal{L}_i are regular.*

Hence in the case of taking-and-breaking games, the regularity of \mathcal{L}_0 implies the regularity of all the languages \mathcal{L}_i . In our different setting of taking-and-merging games, Waldmann's proof cannot be transposed easily. In addition, the situation does not seem that clear. Let us consider a particular game.

Proposition 19. *For the game $\{a, a^2, b, b^2\}$, the set \mathcal{L}_0 of \mathcal{P} -positions is regular.*

Proof: Consider the partition of A^* into two sets

$$P = \{w \in A^* : |w|_a - |w|_b = 0 \pmod{3}\} \text{ and } N = A^* \setminus P.$$

The set P satisfies the stability property of Lemma 5: take any word $w \neq \varepsilon$ in P and apply one of the reductions. The resulting word u is such that

$$|u|_a - |u|_b \in (|w|_a - |w|_b + \{-2, -1, 1, 2\}).$$

Hence there is no move between two words in P .

The set P is absorbing: take a word w such that $|w|_a - |w|_b = 1 \pmod{3}$. If $|w|_a > 0$, then using the reduction $a \rightarrow \varepsilon$ leads to the set P . Otherwise, w contains only b 's. Notice that it contains at least two b 's and using the reduction $b^2 \rightarrow \varepsilon$ leads again to the set P . Now take a word w such that $|w|_a - |w|_b = 2 \pmod{3}$. The argument is similar. If $|w|_b > 0$, then using the reduction $b \rightarrow \varepsilon$ leads to the set P . Otherwise, w contains only a 's. Notice that it contains at least two a 's and using the reduction $a^2 \rightarrow \varepsilon$ leads again to the set P .

Hence according to Lemma 5, the set P is the set of the \mathcal{P} -positions of the game and is exactly \mathcal{L}_0 . It is a straightforward exercise to see that P is a regular language recognized by a DFA with three states. \square

In parallel with this result, we have computed the first few elements from the sets \mathcal{L}_i of $\{a, a^2, b, b^2\}$ for all words of length less than 24:

$$(\max\{\mathcal{G}(u)\})_{|u|=0,1,2,\dots} = 0, 1, 2, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 8, 8, \dots$$

Our program that iteratively builds the DFA for the Grundy function did not found any reasonable candidate up to this length. Hence a natural question arises.

Question 20. For the game $\{a, a^2, b, b^2\}$, is there a set \mathcal{L}_i that is not regular?

In addition, our program found that new Grundy values regularly appear when the length of the words grows. For example, there are words of length 22 with a Grundy value of 8. This correlation between the regularity of \mathcal{L}_0 and a finite number of Grundy values has already been established for some rewrite games. Indeed, in the game DUOTAIRE, as well as for taking-and-breaking games (see Theorem 18), an argument to ensure that \mathcal{L}_0 is not regular consists in showing that the Grundy values are not bounded. We wonder whether this property remains true for taking-and-merging games:

Question 21. Are there taking-and-merging games for which the set \mathcal{L}_0 is regular but the Grundy function is not bounded?

Note that in Question 21, the converse property does not hold. Indeed, a game for which the Grundy values are bounded does not necessarily has a regular language for \mathcal{L}_0 . Consider the example of the game $\{a^2, b^2\}$ detailed in Section 2.2, for which \mathcal{L}_0 is proved to be not regular (and where the Grundy values do not exceed 1).

5 Winning positions and regular languages

Here we consider slightly more general rewriting rules and show that a very simple problem then become undecidable. More precisely we consider strongly terminating rewriting games, as defined below.

Definition 22. A rewriting game G is called strongly terminating if every reduction $u \rightarrow v$ is such that $|u| > |v|$.

As the name suggests, a strongly terminating game is such that, from any given starting position, the game is terminating. For such a game, there is a trivial algorithm computing the Grundy value of a given position, although in the worst case, this algorithm runs in exponential time with respect to the length of the starting position. We consider here the following more general problem.

Problem 23. Given a strongly terminating game G , and a language L of “starting positions”, decide whether there is a N -position for G belonging to L .

The main result of this section is the following.

Theorem 24. Problem 23 is undecidable, even though the language L is a star-free regular language.

We will prove Theorem 24 by a reduction from the halting problem of a deterministic Turing machine on the empty word. It takes indeed the rest of Section 5.

5.1 Instantiation of Problem 23

In the following, we consider a deterministic Turing machine T defined by

- Q , the finite set of states;
- $q_0 \in Q$, the initial state;
- $q_{\text{accept}}, q_{\text{reject}} \in Q$, the accept and reject state;
- Γ , the finite alphabet of tape symbols;
- $\$ \in \Gamma$, the left marker of the tape;
- $\beta \in \Gamma$, the blank symbol;

- $\Sigma \subseteq \Gamma$, the set of input symbols⁽ⁱ⁾; and
- $\delta : ((Q \setminus F) \times \Gamma) \rightarrow (Q \times \Gamma \times \{\triangleleft, \triangleright\})$, the partial transition function.

We denote by F the set of halting states, that is: $F = \{q_{\text{accept}}, q_{\text{reject}}\}$. As usual, we assume that the head of T is on the symbol $\$$ at the beginning of a computation. We also assume for every state q in $(Q \setminus F)$ that $\delta(q, \$)$, if it is defined, is always equal to $(r, \$, \triangleright)$ for some state r . For more details on Turing machine or the Halting Problem, see for instance [3, 7].

Now, let us define the instance (G, L) of Problem 23 to which we reduce the halting of T on the empty word.

In the following, the first player is called A(lice) and the second one is called B(ob).

First, the alphabet of the game G is $Q \uplus \Gamma \uplus \{\#\} \uplus M$, where Q, Γ are defined above, where $\#$ is the ‘erasable’ symbol that will make G strongly terminating, and where

$$M = \{\triangleright_A, \triangleright_B, \triangleleft_A, \triangleleft_B\} \quad (1)$$

is the set of ‘head symbols’, which indicate the position and direction of the head, as well as the current player (A or B).

Second, the reductions are defined by equations (2) to (11), below. For every state q in Q , the *left-shift reductions* are as follows.

$$\#\#\#\#\triangleleft_A q \longrightarrow_G \#\triangleleft_B q\#\#\# \quad (2)$$

$$\#\triangleleft_B q \longrightarrow_G \triangleleft_A q \quad (3)$$

Symmetrically, the *right-shift reductions* are as follows, for every state q in Q .

$$q\triangleright_A \#\#\#\# \longrightarrow_G \#\#\#q\triangleright_B \# \quad (4)$$

$$q\triangleright_B \# \longrightarrow_G q\triangleright_A \quad (5)$$

The *right-transition reductions* are as follows, for every states p, q in $(Q \setminus F)$ and for every tape symbol α, γ in Γ such that $\delta_T(p, \alpha) = (q, \gamma, \triangleleft)$.

$$\#\#\#\#\alpha\triangleleft_A p \longrightarrow_G \#\triangleleft_B q\#\#\#\gamma \quad (6)$$

$$\#\#\#\#p\triangleright_A \alpha \longrightarrow_G \#\triangleleft_B q\#\#\#\gamma \quad (7)$$

Similarly, if $\delta_T(p, \alpha) = (q, \gamma, \triangleright)$, the *left-transition reductions* are as follows.

$$\alpha\triangleleft_A p\#\#\#\# \longrightarrow_G \gamma\#\#\#q\triangleright_B \# \quad (8)$$

$$p\triangleright_A \alpha\#\#\#\# \longrightarrow_G \gamma\#\#\#q\triangleright_B \# \quad (9)$$

Finally, the *halting reductions* are as follows, for every states p in $(Q \setminus F)$ and for every tape symbol c in Γ such that $\delta_T(p, c) = (q, d, x)$, for some $q \in F, d \in \Gamma$ and $x \in \{\triangleleft, \triangleright\}$.

$$c\triangleleft_A p \longrightarrow_G q \quad (10)$$

$$p\triangleright_A c \longrightarrow_G q \quad (11)$$

⁽ⁱ⁾ Since we only consider the empty word as input, the set of input symbols is irrelevant.

Third, the language L of starting positions is

$$L = \$ \triangleleft_A q_0 (\# + \beta)^* . \quad (12)$$

It may be verified that, thus defined, G is indeed strongly terminating and L is a star-free regular language.

5.2 Game G is a zero-player game

First, easy inductions show the following.

Lemma 25. *Let us consider the game G starting from a starting position $w_0 \in L$. Let w be a later position in a run of the game.*

- *Position w contains exactly one occurrence of a symbol from Q .*
 - *If it is player A's turn, w contains no occurrence of \triangleright_B or \triangleleft_B , and contains exactly one occurrence of either \triangleright_A or \triangleleft_A .*
 - *If it is player B's turn, then w contains no occurrence of \triangleright_A or \triangleleft_A and w contains at most one occurrence of a symbol in $\{\triangleright_B, \triangleleft_B\}$.*
- Moreover, if w contains no head symbol, then the last reduction applied was either (10) or (11).*

It follows immediately that the game is in fact asymmetrical. The only reduction that player B can ever apply are (3) and (5); while the only reduction that player A can ever apply are the other ones (i.e., rules (2), (4), (6), (7), (8), (9), (10) and (11)). Next lemma states the condition for the game to end.

Lemma 26. *Let us consider the game G starting from a position in L . If player A makes a halting move, then she wins the game. Otherwise, player B always has a move to make afterwards.*

Proof: After applying (10) or (11), then no symbol in $\{\triangleright_A, \triangleright_B, \triangleleft_A, \triangleleft_B\}$ appear in the position; hence no reduction can be applied any longer. After applying (2), (6) or (7), then player B can always apply (3). Similarly, after applying (4), (8) or (9), then player B can always apply (5). \square

Finally, let us show that G is a zero-player game, i.e., each move of the game is forced.

Proposition 27. *Let us consider the game G starting from a position w_0 in L . There is a unique sequence of words w_1, \dots, w_n and a unique sequence of reductions r_0, \dots, r_{n-1} such that for every integer i , $0 \leq i < n$, it holds $w_i \xrightarrow{r_i} w_{i+1}$. Moreover, player A wins if and only if $n > 0$ and r_{n-1} is an instance of (10) or (11).*

Proof: From Lemma 25, one may see that no position coming from w_0 may ever have more than one head symbol. Let u be a word with only one head symbol h and let us show that at most one reduction may be applied to u . Indeed, if h is \triangleleft_B or \triangleright_B , only reduction (5) or (3) may be applied, respectively. If $h = \triangleleft_A$, the only reductions that may be applied are (2), (6), (8), or (10), and it is easy to see that if one may be applied the other ones cannot (sometimes because T was assumed to be deterministic). A similar reasoning yields for the case $h = \triangleright_A$. Applying Lemma 26 concludes the proof. \square

5.3 Game G simulates part of the run of T

In this section, we define how the current position in G bears the state and tape of a step of the run of the Turing machine T .

Definition 28. Let $u = a_0 a_1 \cdots a_n$ be a word in $\Gamma^*((Q \triangleright_A \Gamma) + (\Gamma \triangleleft_A Q))\Gamma^*$.

- We denote by $\text{TAPE}(u)$ the infinite sequence $v\beta^\omega$, where v is the word in Γ^* resulting from erasing from u each letter that belongs to $(Q \cup \{\triangleright_A, \triangleleft_A\})$.
- We denote by $\text{STATE}(u)$ the unique symbol in Q that appears in u .
- We denote by $\text{HEAD}(u)$ the integer $j - 1$, where j is such that $a_j \in \{\triangleright_A, \triangleleft_A\}$ in u .

Note that the letter at index $\text{HEAD}(u)$ in $\text{TAPE}(u)$ is exactly the letter pointed at by \triangleright_A or \triangleleft_A in u . For instance if $u = \$\alpha_1\alpha_3\alpha_3p\triangleright_A\alpha_2\beta\beta$, then $\text{TAPE}(u) = \$\alpha_1\alpha_3\alpha_3\alpha_2\beta^\omega$, $\text{STATE}(u) = p$ and $\text{HEAD}(u) = 4$, that is, the index of α_2 in $\text{TAPE}(u)$.

Proposition 29. We again take notation of Proposition 27. Let φ be the word morphism erasing the symbols $\#$. Let i be an even integer, $0 \leq i < n$ (that is, a position where it is player A's turn). Then, the run of the Turing machine T on the empty word eventually reaches state $\text{STATE}(\varphi(w_i))$ with tape $\text{TAPE}(\varphi(w_i))$ and head at position $\text{HEAD}(\varphi(w_i))$.

Proof: By induction on i . Case $i = 0$ yields $\text{STATE}(w_0) = q_0$, $\text{TAPE}(w_0) = \$\beta^\omega$, $\text{HEAD}(w_0) = 0$, that is, the initial setup of the Turing machine T .

Let $(i + 2)$ be an even integer, $0 \leq i < (n - 2)$. If r_i is (2) (resp. (4)), then r_{i+1} is (3) (resp. (5)) and $\varphi(w_{i+2}) = \varphi(w_i)$, and induction hypothesis concludes the case. Reduction r_i cannot be (10) or (11), because then player B would have no rule to apply and it would hold $(i + 2) = n$. Reduction r_i is either (6), (7), (8) or (9). We will assume that r_i is (8); other cases are treated similarly.

Since we may apply reduction (8), w_i is equal to $u(\alpha \triangleleft_A p \# \# \# \#)v$ with $u, v \in (\Gamma + \{\#\})^*$ and such that $\delta_T(p, \alpha)$ is defined and equal to $(q, \gamma, \triangleright)$ for some $q \in (Q \setminus F)$ and $\gamma \in \Gamma$. It follows that $w_{i+2} = u(\gamma \# \# \# q \triangleright_A)v$, since player A uses reduction (8) and then player B necessarily uses reduction (5). We write

$$\text{STATE}(\varphi(w_i)) = p, \text{ HEAD}(\varphi(w_i)) = j \text{ and } \text{TAPE}(\varphi(w_i)) = u'\alpha v',$$

where u' is such that $|u'| = j$. Hence, the following equalities hold.

$$\text{STATE}(\varphi(w_{i+2})) = q \quad \text{HEAD}(\varphi(w_{i+2})) = j + 1 \quad \text{TAPE}(\varphi(w_{i+2})) = u'\gamma v'$$

It is exactly the state, tape and head position one obtains by applying the transition defined by $\delta_T(p, \alpha) = (q, \gamma, \triangleright)$ from the state p , tape $u'\alpha v'$ and head position j . \square

Corollary 30. We again take notation of Proposition 29. If w_0 is a winning position for player A, then T halts on the empty word.

Proof: Let $j \in \mathbb{N}$, $\alpha \in \Gamma$, $p \in Q$ and $u, v \in \Gamma^*$ be such that $|u| = j$,

$$\text{STATE}(\varphi(w_{n-1})) = p, \text{ HEAD}(\varphi(w_{n-1})) = j \text{ and } \text{TAPE}(\varphi(w_{n-1})) = u\alpha v.$$

Since w_0 is a winning position for player A, n is odd and r_{n-1} is an instance of a halting reduction. It follows that $\delta_T(p, \alpha)$ is defined and that its first component is an accepting state. Moreover, $n - 1$ is even and we apply Proposition 29: the Turing machine T eventually reaches state p with its head on α , hence accepts. \square

5.4 From the proper starting position, G may simulate each finite run of T

Application of left-transition, right-transition or halting reductions (i.e., reductions from (6) to (11)), corresponds to the actual simulation of transitions of T . Other reductions simply allow to shift the head symbol to the next tape symbol. Next lemma states a sufficient condition for a ‘complete’ simulation of one (non-halting) transition of T , that is 1) applying the transition and 2) shifting entirely the head to the next tape symbol.

Lemma 31. *Let $\alpha, \gamma, \theta \in \Gamma$ be three tape symbols, $p, q \in Q$ be two states, u, v be two positions of G and n be a positive integer. Then u reduces to v in $2n$ moves in the following cases.*

- (i) $\delta_T(p, \alpha) = (q, \gamma, \triangleright)$, $u = \alpha \triangleleft_A p \#^{4n} \theta$, $v = \gamma \#^{2n} q \triangleright_A \theta$
- (ii) $\delta_T(p, \alpha) = (q, \gamma, \triangleright)$, $u = p \triangleright_A \alpha \#^{4n} \theta$, $v = \gamma \#^{2n} q \triangleright_A \theta$
- (iii) $\delta_T(p, \alpha) = (q, \gamma, \triangleleft)$, $u = \theta \#^{4n} \alpha \triangleleft_A p$, $v = \theta \triangleleft_A q \#^{2n} \gamma$
- (iv) $\delta_T(p, \alpha) = (q, \gamma, \triangleleft)$, $u = \theta \#^{4n} p \triangleright_A \alpha$, $v = \theta \triangleleft_A q \#^{2n} \gamma$

Proof: For item (i), apply reduction (8), then alternatively n times reduction (5) and $(n - 1)$ times reduction (4). Proofs of items (ii), (iii) and (iv) are similar. \square

In other words, if a transition of T makes the head shift right (resp. left) then it will be executed ‘completely’ in G if there are $4n$ consecutive occurrences of symbol $\#$, for some positive n , right of (resp. left of) the factor of u that belongs to $((\Gamma \triangleright_A Q) + (Q \triangleleft_A \Gamma))$. Then, an induction yields the following.

Proposition 32. *Let us assume that T halts on the empty word after m transitions. Then, the following word w is a winning position for G .*

$$w = \$ \triangleleft_A q_0 \left(\#^{2^{(m+1)}} \beta \right)^m$$

Corollary 33. *If T halts on the empty word, then L contains a winning position for G .*

Finally, Corollaries 30 and 33 directly yield Theorem 24. Indeed, assume to the contrary that Problem 23 is decidable, then one would conclude that the halting problem on the empty word is decidable. Nevertheless the latter problem is well-known to be undecidable [7]. Let us conclude Section 5 with a conjecture.

Problem 34. *Given a strongly terminating game \mathcal{G} , decide whether the winning positions of \mathcal{G} form a regular language.*

Conjecture 35. *Problem 34 is undecidable.*

6 Perspectives

Rewrite games open the door to a large field of new interesting questions, as it generalizes a large set of combinatorial games. In the previous sections, we have given a couple of open problems that we found the most relevant ones in the context of taking-and-merging games. Could they be adapted with an alphabet of a larger size?

Moreover, there are other instances of rewrite games that would make sense to be investigated as their rules can also be expressed with piles of tokens. Consider for example taking-and-merging games where

rules of the form $a^k \rightarrow b^\ell$ are adjoined. Such games can be seen as taking games where tokens have two colors, say black (for a) and white (for b). Moves consist in either removing tokens or flipping black tokens (that become white). In such games, what would the \mathcal{L}_i languages look like? For example, in the game $\{a \rightarrow b, a \rightarrow \varepsilon, b \rightarrow \varepsilon\}$, each Grundy language is regular.

Acknowledgements

We would like to thank Idris Ayouaz for his useful computations made on several instances of this game and the reviewer for his useful comments.

References

- [1] L. Beaudou, E. Duchêne and S. Gravier: A survey about Solitaire Clobber, in *Games of No Chance 4*, MSRI Publ. (R.J. Nowakowski, ed.), Vol. 63, Cambridge University Press, Cambridge, 2015.
- [2] E. R. Berlekamp, J. H. Conway, R. K. Guy, *Winning ways for your mathematical plays*, Academic Press, 1983.
- [3] J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Co., Reading, Mass., 2006.
- [4] C. Moore, D. Eppstein, One-Dimensional Peg Solitaire, and Duotaire, in *More games of no chance* (Berkeley, CA, 2000), 341–350, Math. Sci. Res. Inst. Publ., 42, Cambridge Univ. Press, Cambridge, 2002.
- [5] B. Ravikumar, Peg-solitaire, string rewriting systems and finite automata, *Theoret. Comput. Sci.* **321** (2004), 383–394.
- [6] A. N. Siegel, *Combinatorial Game Theory*, San Francisco, CA, (2013).
- [7] M. Sipser, *Introduction to the Theory of Computation*, Third Int. Ed. Cengage Learning, 2013.
- [8] Terese, *Term Rewriting Systems*, Cambridge Tracts in Theoret. Comput. Sci., Vol. 55, Cambridge University Press, 2003
- [9] J. Waldmann, Rewrite games, in *Rewriting techniques and applications*, 144–158, *Lecture Notes in Comput. Sci.* **2378**, Springer, Berlin, 2002.