On optimum left-to-right strategies for active context-free games

(Preliminary Report - August 2012)

Henrik Björklund
Umeå University
henrikb@cs.umu.se

Martin Schuster
TU Dortmund
martin.schuster@udo.edu

Thomas Schwentick
TU Dortmund
thomas.schwentick@udo.edu

Joscha Kulbatzki
SCISYS Deutschland GmbH
Joscha.Kulbatzki@scisys.de

ABSTRACT

Active context-free games are two-player games on strings over finite alphabets with one player trying to rewrite the input string to match a target specification. These games have been investigated in the context of exchanging Active XML (AXML) data. While it was known that the rewriting problem is undecidable in general, it is shown here that it is EXPSPACE-complete to decide for a given context-free game, whether all safely rewritable strings can be safely rewritten in a left-to-right manner, a problem that was previously considered by Abiteboul et al. Furthermore, it is shown that the corresponding problem for games with finite replacement languages EXPSPACE-complete.

1. INTRODUCTION

In this paper, we study Active Context-Free Games, which are played by two players on finite strings. The motivation for these games comes from the study of Active XML documents [1, 3, 7]. These are documents in which only some of the data is explicitly given. The rest of the data is obtained by calls to Web services. An example could be a document that includes as a part the latest news headlines. Rather than storing these headlines on the host web server, a web service run by a news agency is called each time the document is requested by a user. The headlines retrieved by the call are then incorporated into the document before it is sent to the user. It can also be the case that the news agency returns another active document, i.e., one that contains further possibilities for calling web services.

This approach can, however, give rise to additional problems with conforming to document schemas. Not only must the hosts ensure that their own documents conform to certain schemas, but also that this is the case for all possible replacements resulting from their web service calls.

In order to model and study this situation, active context-free games, or context-free games for short, were introduced by Muscholl et al. [8]. Such games are played on strings rather than trees. The first player, JULIET, represents the host. She uses calls on letters to try to ensure that the string conforms to a schema, in this case represented by a regular language. Her opponent, ROMEO, gets to pick a string from a regular set to replace the letter JULIET called on. Starting from a given string, representing the active document, JULIET wins if the string is ever rewritten into a word in the schema language. If she doesn’t succeed, ROMEO wins.

In this particular paper we study a certain kind of strategies for JULIET, namely so-called left-to-right (L2R) strategies. Such a strategy, once it has played a call on a certain symbol $s$, never again plays a call on any symbol to the left of $s$. L2R-strategies have been considered before, e.g. in [8, 2]. The motivation for considering this restricted form of strategies is that they require less space when implemented, and that the complexity of dealing with them is much nicer than for general strategies. For instance, while it is in general impossible to determine, given a game and an input string, whether JULIET has a winning strategy, it can be decided in EXPSPACE whether she has a winning L2R-strategy [8].

The particular problem we study here is the L2RALL problem, that is, given a game, does JULIET have a winning L2R-strategy for every string for which she has a winning strategy at all? The practical importance of this problem is in schema and document design. It was also mentioned in [2] where it was claimed to be undecidable.

We take the following approach to the problem. First we show that we can concentrate on determining whether JULIET always has an winning L2R-strategy whenever she has a strategy that goes left only once. Then we show how to construct automata for all strings with a winning L2R-strategy and for all strings with a winning “almost L2R” strategy, respectively. The L2RALL problem then boils down to a containment test for these two automata. To show that the automata can be effectively (and optimally efficiently) computed, we use the concept of effects of a string. These are objects that summarize how substrings of the word the
players are playing on influence the outcome of the game. Even though this approach is pretty simple, in general, all steps require some care to yield optimal complexities.

Our Main Theorem has two parts. The first deals with arbitrary games and the second with games in which Romeo is restricted to finite sets of words when he chooses replacements for Call moves. More precisely, we show the following.

**Theorem 1** (Main Theorem). (a) For context-free games, $L2RAll$ is EXPSPACE-complete.

(b) For context-free games with finite replacement languages that are explicitly given in the input, $L2RAll$ is EXPTIME-complete.

The paper is organized as follows. After some preliminaries, we show in Section 3 that to decide the $L2RAll$ problem, general strategies can be replaced by "almost L2R" strategies. In Section 4 we define effects and state their basic properties. Section 5 shows how to define and compute automata for the set of words with winning L2R strategies. In Section 6 we give the decision algorithms and in Section 7 the matching lower bounds.

**Related work.** We already discussed the most important related papers [2, 8, 1, 7] above. That automata for the set of words with winning L2R strategies can be constructed in exponential time was already shown in [8]. However, the proof did not give an explicit construction but was by reduction to algorithmic problems for pushdown systems. That L2RAll is decidable was already claimed in the Diploma thesis of Joscha Kubatski, which was written under the supervision of the third author [6].

## 2. PRELIMINARIES

### 2.1 Context-free games

A context-free game $G = (\Sigma, R, T)$ consists of a finite alphabet $\Sigma$, a rule set $R \subseteq \Sigma \times \Sigma^*$ and a regular target language $T \subseteq \Sigma^*$. It is required that for each symbol $f \in \Sigma$, the set $R_f = \{ u \mid (f, u) \in R \}$ is regular. By $\Gamma$ we denote the set $\Gamma =\{ f \mid f \in \Sigma, R_f \neq \emptyset \}$ and we call the symbols from $\Gamma$ function symbols. We denote function symbols by $f, f_1, \ldots$ and terminal symbols from $\Sigma \setminus \Gamma$ by $a, b, a_1, \ldots$.

A play of the game $G$ is played by two players, JULIET and ROMEO, on a word $w \in \Sigma^*$. The play can have several passes in which the focus is moved along the current string, from left to right. In each round, JULIET selects whether the current symbol in the current word should be rewritten or passed over. If she chooses a rewrite, then ROMEO chooses a rewrite or, if the first symbol $f$ of $w$ is from $\Gamma$ a Call move. If she selects Call, then ROMEO selects a string from the set $R_f$. In a configuration $(1, u, v)$ JULIET can either do a left step or stop the game.

A move of JULIET is thus represented by Read, Call, LS or Stop and a move of ROMEO is represented by a string $x$. The configuration $C' = (p', u', v')$ is a possible successor configuration of $C = (p, u, v)$ (Notation: $C \to C'$) if

1. $p' = p = 1, u' = us, and svu' = v$ for some $s \in \Sigma$ (JULIET plays Read);
2. $p = 1, p' = 2, u' = u, and v' = v$ (JULIET plays Call);
3. $p = 2, p' = 1, u' = u, v = fx$ for some $f \in \Gamma, v' = yx$ for some $y \in R_f$ (ROMEO plays $y$);
4. $p' = p = 1, u \not\in T, v = \varepsilon, v' = u, u' = \varepsilon$ (JULIET plays LS).

If JULIET plays Stop in a configuration $C = (p, u, v)$ we write $C \to \top$ if $u \in T$ and $C \to \bot$ if $u \not\in T$ and we thus consider $\top$ and $\bot$ as configurations as well.

Since we will mostly consider configurations where JULIET is to move, we often omit the player when talking about them. Thus $(u, v)$ is a shorthand for $(1, u, v)$.

The initial configuration of game $G$ for string $u$ is defined as $C_0(u) =_{\text{def}} (1, \varepsilon, u)$.

A play of the game $G$ is either an infinite sequence $\Pi = C_0, C_1, \ldots$ or a finite sequence $\Pi = C_0, C_1, \ldots, C_k$ of configurations, where, for each $i > 0$, $C_{i-1} \to C_i$. If the sequence is finite, then $C_k$ must be either $\top$ or $\bot$. If $C_k = \top$, JULIET wins the play, in all other cases, ROMEO wins. We write $\Pi \not\in p$ if player $p$ wins $\Pi$.

We assume in this paper that a game $G = (\Sigma, R, T)$ is represented by a DFA $A(T)$ for $T$ and by a NFA $A_r$ for $R$, for every $f \in \Gamma$. In the sequel, let $A(T) = (Q, \Sigma, \delta, F, q_0)$.

We note that in order to facilitate our proofs, our definitions above differ slightly from the one used previously by, e.g., Abiteboul et al. [2] and Muscholl et al. [8]. It is easy to confirm that the definitions are equivalent, however. As an example, in [2] every round is played by JULIET selecting a function symbol $f$ in the string and ROMEO then chooses a string from $R_f$ to replace it. In our setting, JULIET can achieve the same thing by just playing Read moves (and possibly one LS move) until she reaches the position she wants to rewrite. She can then play a Call move.

### 2.2 Game trees

The game tree $\text{Tre}_{G,u}$ for $G$ on string $u$ is a tree labeled by configurations. Each branch of the tree represents one possible play of the game. The root of $\text{Tre}_{G,u}$ is labeled by the initial configuration $C_0(u)$. A node labeled $C$ has one child for every configuration $C'$ such that $C \rightarrow C'$. This means that the only leaves of $\text{Tre}_{G,u}$ are nodes labeled by final configurations of finite plays. In general, nodes labeled by configurations where JULIET is to move have one or two children, while nodes labeled by configurations where ROMEO is to move can have infinitely many children.

### 2.3 Strategies

A strategy for player $p \in \{1, 2\}$ maps prefixes $C_0, C_1, \ldots, C_k$ of plays, where $C_0$ is an initial configuration and $C_k$ is a $p$-configuration, to allowed moves. A strategy $\sigma$ is memoryless if, for every prefix $C_0, C_1, \ldots, C_k$ of a play, $\sigma(C_0, C_1, \ldots, C_k)$ only depends on $C_k$.

We note that whether $R_f$ is represented by DFAs or NFAs does not influence the complexity. However, we conjecture that allowing NFAs for $T$ may lead to an exponential blowup of the complexity.
We denote strategies for Juliet by $\sigma, \sigma', \sigma_1, \ldots$ and strategies for Romeo by $\tau, \tau', \tau_1, \ldots$.

For configurations $C, C'$ and strategies $\sigma, \tau$ we write $C \xrightarrow{\sigma} C'$ if $C'$ is the unique successor configuration of $C$ determined by the strategies $\sigma$ and $\tau$. Given an initial word $u$ and strategies $\sigma, \tau$ the play $^2 \Pi(\sigma, \tau, u) =_{def} C_0(u) \xrightarrow{\sigma} C_1 \xrightarrow{\tau} \cdots$ is uniquely determined.

A strategy $\sigma$ for Juliet is finite on string $u$ if the play $\Pi(\sigma, \tau, u)$ is finite for every strategy $\tau$ of Romeo. It is a winning strategy for $u$ if $\Pi(\sigma, \tau, u) \equiv 1$, for every $\tau$. A strategy $\tau$ for Romeo is a winning strategy for $u$ if $\Pi(\sigma, \tau, u) \equiv 2$, for every strategy $\sigma$ of Juliet.

We are particularly interested in restricted kinds of strategies of Juliet.

A left-to-right (L2R) strategy for Juliet is a strategy in which Juliet never does a LS move.

We denote the set of all unrestricted strategies for Juliet in the context-free game $G$ by $\text{STRAT}(G)$, and the set of all L2R-strategies by $\text{STRAT}^{\text{L2R}}(G)$. The set of all strategies for Romeo is denoted by $\text{STRAT}^{\text{Romeo}}(G)$.

By definition, $\text{STRAT}^{\text{L2R}}(G) \subseteq \text{STRAT}(G)$.

By $\text{safe}^{\text{L2R}}(G)$ we denote the set of all words for which Juliet has a winning strategy and by $\text{safe}^{\text{L2R}}(G)$ the set of all words for which she has a winning L2R-strategy.

In this paper we are mainly interested in the following algorithmic problem: given a context-free game $G$, decide whether $\text{STRAT}^{\text{L2R}}(G) = \text{STRAT}(G)$. By L2RALL we denote the set of all games $G$, for which $\text{STRAT}^{\text{L2R}}(G) = \text{STRAT}(G)$.

As context-free games are reachability games we can make use of the following classical result; see, e.g., [5].

**Theorem 2.** Let $G$ be context-free game, and $u$ a string. Then the following statements holds for the game starting from $u$.

(a) Either Juliet or Romeo has a winning strategy. If Juliet or Romeo has a winning strategy then they also have a memoryless strategy.

(b) Either Juliet has a winning L2R strategy or Romeo has a winning strategy against all L2R strategies. If Juliet has a winning L2R strategy then she also has a memoryless winning L2R strategy. If Romeo has a winning strategy against all L2R strategies then he also has a memoryless such strategy.

Therefore, we will only consider memoryless strategies. Thus, in the following, strategies $\sigma$ for Juliet map configurations $C$ to moves $\sigma(C) \in \{\text{Call}, \text{Read}\}$ and strategies $\tau$ for Romeo map configurations $C$ to moves $\tau(C) \in \Sigma^*$.

We sometimes consider subgames on a certain part of a string and talk about strategies for subgames. From a configuration $(u, vw)$, Juliet can use a strategy $\sigma$ on the subgame on $v$. This means that she follows $\sigma$ until a configuration $(uv', w)$ is reached.

The strategy tree for a strategy $\sigma$ of Juliet is the restriction $\text{Tree}_{G,u}(\sigma)$ of $\text{Tree}_{G,u}$ to $\sigma$. In other words, for nodes labeled by configurations where Juliet is to move, we remove all subtrees rooted at children labeled by configurations that are not selected by $\sigma$. Strategy trees for Romeo are defined symmetrically. If we fix strategy $\sigma$ for Juliet and $\tau$ for Romeo, we get $\text{Tree}_{G,u}(\sigma, \tau)$, which only has one branch, labeled by the play $\Pi(\sigma, \tau, u)$. Notice that if a strategy $\sigma$ of Juliet is winning, then $\text{Tree}_{G,u}(\sigma)$ has no infinite branches.

If $\Pi(\sigma, \tau, u)$ is finite, then word $^3 \omega(u, \sigma, \tau)$ is the word in the final configuration of the play on $w$ following $\sigma$ and $\tau$ (and otherwise word $^3 \omega(w, \sigma, \tau) = \bot$). We let words $^3 \omega(u, \sigma, \tau) =_{def} \{\text{word}^3 \omega(w, \sigma, \tau) | \sigma \in \text{STRAT}^{\text{Romeo}}(G)\}$.

As usual, if the game $G$ is clear from the context, we shall omit $G$ from the notation. We may also restrict these definitions in a natural way to only include finite or L2R-strategies where mentioned.

To deal with “game effects” the following will be useful. We call a set of calls normal if it does not contain two sets $X$ and $Y$ with $X \subset Y$. A set $S$ of sets can be normalized by applying the $\text{Norm}$ operator, defined as follows.

$\text{Norm}(S) = \{ \forall \sigma \exists \chi (v, \sigma, \tau) \in S \mid \exists \chi \in S \}$

**Lemma 3.** Let $S_1, S_2$ be normal sets of calls. If for every $\delta \in S_1$ there is $\sigma \in S_2$ such that $\sigma \subseteq \delta$ and vice versa then $S_1 = S_2$.

**Proof.** We show that every $\delta \in S_1$ is also in $S_2$. The lemma then follows by symmetry.

Let $\delta \in S_1$. By our assumption there is $\sigma \in S_2$ such that $\sigma \subseteq \delta$ and there is a set $s_1' \in S_1$ such that $s_1' \subseteq \sigma$. However, as $S_1$ is normal, $s_1 = s_1'$ and we get $s_1 = s_1' \subseteq \sigma \subseteq \delta$ and thus $s_1 = s_2$. \hfill $\Box$

Where notation is dense, we sometimes just use $\mathcal{N}(S)$ for $\text{Norm}(S)$.

**3. FROM GENERAL TO L2R*-STRATEGIES**

**Definition 1.** A strategy $\sigma$ of Juliet is an extended L2R-strategy ($\text{L2R}^+$) if for every string $u$ and every strategy $\tau$ of Romeo, Juliet plays LS at most once and plays at most one Call before the LS-move.

**Lemma 4.** Let $G$ be a context-free game. Then $\text{safe}^{\text{L2R}}(G)$ if and only if $\text{safe}^{\text{L2R}^+}(G) = \text{safe}^{\text{L2R}}(G)$.

**Proof.** If $\text{safe}^{\text{L2R}}(G) = \text{safe}^{\text{L2R}^+}(G)$, then $\text{safe}^{\text{L2R}^+}(G)$ by definition.

Assume that $\text{safe}^{\text{L2R}}(G) \neq \text{safe}^{\text{L2R}^+}(G)$ and let $w$ be a string in $\text{safe}^{\text{L2R}}(G) \setminus \text{safe}^{\text{L2R}^+}(G)$. Let $\sigma$ be a winning strategy for Juliet on $w$, i.e., starting from the configuration $(1, w, e)$. Consider the strategy tree $\text{Tree}_{G,u}(\sigma)$. In addition to the configuration labels, we mark each node $n$ in this tree with a value $\text{LS}(n)$, where $\text{LS}(n)$ is the maximum number of LS moves, on any branch of the subtree rooted in $n$. Since the tree has infinite branching, the value $\text{LS}(n)$ can, in general, be unbounded, i.e., $\text{LS}(n) = \infty$. Since $\sigma$ is a winning strategy, however, the tree has no infinite branches.

Nodes $n$ with $\text{LS}(n) \neq \infty$ and $\text{LS}(n) > 0$ are also marked by $\text{Calls}(n)$, the maximum number of Call moves that occur before the first LS step, on any branch of the subtree rooted in $n$. We note that $\text{Calls}(n)$ might be $\infty$.

In the following, we call, for nodes $n$ with $\text{LS}(n) \neq \infty$, the pair $(\text{LS}(n), \text{Calls}(n))$ the marking of $n$ and we denote by $\prec$ the lexicographic order on markings.

Without loss of generality, we may assume that $\sigma$ is optimally efficient in the following sense. We assume that for

---

$^2$As the underlying game $G$ will always be clear from the context, our notation does not mention $G$ explicitly.

$^3$As the underlying game $G$ will always be clear from the context, our notation does not mention $G$ explicitly.
every node \( n \) of the strategy tree, labeled with a configuration \((p, u, v)\), such that \( \text{LS}(n) \neq \infty \), there is no other winning strategy \( \sigma' \) on \( w \), such that the strategy tree for \( \sigma' \) and \( w \) has a node \( n' \) labeled with the same configuration but having a lexicographically smaller marking. Such an optimally efficient strategy can be constructed inductively, by the length of \( w \).

As \( \text{safe}(G) \neq \text{safe}_{L2R}(G) \), there must be a node \( n \) in \( \text{Tree}_{G,w}(\sigma) \) with \( \text{LS}(n) = 0 \).

We first show that \( \text{Tree}_{G,w}(\sigma) \) must contain nodes \( n \) with \( \text{LS}(n) > 0 \) and with a marking different from \((1, 0, 0)\), i.e. configurations in which JULIET actually has to make at least one more Call before her last LS move.

If \( \text{Tree}_{G,w}(\sigma) \) has nodes with \( \text{LS-value} = \infty \), it also has a node \( n' \), where \( \text{LS}(n') = \infty \), but \( \text{LS}(n) \neq \infty \), for every child node \( n \) of \( n' \). Otherwise, \( \text{Tree}_{G,w}(\sigma) \) would have infinite branches, contradicting the fact that \( \sigma \) is a winning strategy. There must be arbitrarily large LS-values among the children of \( n' \) as otherwise LS\((n') \neq \infty \). In particular, \( n' \) must have a JULIET-grandchild \( n \) with \( \text{LS}(n) > 1 \) and therefore a marking differing from \((1, 0, 0)\). If \( \text{Tree}_{G,w}(\sigma) \) has no nodes with \( \text{LS-value} = \infty \), then for the root \( r \) of \( \text{Tree}_{G,w}(\sigma) \) it holds \( \text{LS}(r) \neq \infty \), and thus \( \text{LS}(r) > 1 \) (as otherwise \( w \in \text{safe}_{L2R}(G) \) and \( \text{Call}(r) > 0 \) (as otherwise one LS-step less would suffice — at the root the current position is 1)).

Thus, there must be a JULIET-node \( n_1 \) with \( \text{LS}(n_1) > 1 \), \( \text{LS}(n_1) \neq \infty \) and with a marking different from \((1, 0, 0)\).

Let \( n \) be any node with \( \text{LS}(n) > 0 \), \( \text{LS}(n) \neq \infty \) and with a marking \((i, j) \neq (1, 0)\). For the markings of the children and grandchildren of \( n \) there are the following possibilities:

(i) JULIET plays Read on \( n \) and for the unique child \( n' \) of \( n \) the marking is \((i, j)\).

(ii) JULIET plays Call on \( n, j = \infty \), and there is a grandchild \( n'' \) of \( n \) with marking \((i, \infty)\).

(iii) JULIET plays Call on \( n, j = \infty \), there are grandchildren \( n'' \) with \( \text{LS}(n'') = i \) and for all grandchildren markings of the form \((i, j') \), \( j' \neq \infty \). In particular, there is a grandchild \( n'' \) with marking \((i, j')\), for some \( j' > 0 \).

(iv) JULIET plays Call on \( n, j \neq \infty \), and all grandchildren have markings that are strictly smaller than \((i, j)\), including one child \( n'' \) with marking \((i, j-1)\).

(v) JULIET plays \text{LS} on \( n, j = 0 \) and the child \( n' \) of \( n \) has a configuration of the form \((1, u, \varepsilon)\) and marking \((i - 1, j')\) with \( j' > 0 \).

We can construct a sequence \( n_1, n_2, \ldots \) of nodes by choosing, in all cases (i)-(v), \( n_{i+1} = n'_i \), for \( i \geq 1 \). As this sequence follows a branch of the tree and \( n_1 \) is a winning node for \( \sigma \), the sequence can not be infinite. Furthermore, each leaf has marking \((1, 0)\). Therefore, the sequence must contain a JULIET-node \( n_t \) with marking \((1, 0)\). Let \((1, x, y)\) be the configuration of \( n_t \). We claim that \( xy \in \text{safe}_{L2R}(G) \setminus \text{safe}_{L2R}(G) \).

First, \( xy \notin \text{safe}_{L2R}(G) \), as otherwise the marking of \( n_t \) would be at most \((1, 0)\) (no Call move needed before the LS-step).

On the other hand, as the marking of \( n_t \) is \((1, 1)\), starting from \((1, x, y, \varepsilon)\), JULIET can play Read on \( x \) and can win with one Call before the one and only LS move, therefore \( xy \in \text{safe}_{L2R}(G) \).

Thus, \( \text{safe}_{L2R}(G) \neq \text{safe}_{L2R}(G) \), completing the proof.

### 4. EFFECTS FOR L2R STRATEGIES

**Effects** are a way to summarize the impact with respect to the automaton \( A(T) \) of the possible strings by which a (sub-)string can be rewritten in one pass of a play. In this section, we only consider L2R strategies for JULIET, that is, JULIET never makes an LS-move.

Suppose we have the game configuration \((1, v, u, w)\). As play goes on, it will eventually reach a configuration \((1, v', w', u')\), where \( u \) has been traversed and rewritten into \( u' \). If we fix a strategy for JULIET and ROMEO then \( u' \) is uniquely determined (unless the subgame on \( u \) does not terminate). If we only fix a strategy \( \sigma \) for JULIET, each strategy of ROMEO determines a string \( u' \) (or does not terminate) and we can associate the set \( \text{words}(u, \sigma) \) with \( \sigma \). The relative effect of \( u \) for a strategy \( \sigma \) of JULIET and a state \( q \) is just the set of states that \( A(T) \) can reach by reading strings in \( \text{words}(u, \sigma) \), starting from state \( q \). The effect of \( u \) is basically the set of all such sets \( \text{words}(u, \sigma) \), for all states \( q \).

Thus \( E[u] \) is a mapping that assigns to every state of \( Q \) a set of sets of states and thus its type is \([Q] \to \mathcal{P}(\mathcal{P}(Q))\).

**Definition 2.** Let \( u \) be a string, \( q \in Q \) a state and \( \sigma \) a L2R-strategy of JULIET. The relative effect \( e(\sigma, u, q) \) is the set \( \{\delta(q, w) \mid w \in \text{words}(u, \sigma)\} \) or \( \bot \) if \( \bot \in \text{words}(u, \sigma) \).

The effect \( E[u] \) of \( u \) maps every state \( q \) to the normalized set of relative effects \( e(\sigma, u, q) \) for all \( \sigma \in \text{STRAT}_{L2R} \).

Stated less formally, \( e(\sigma, u, q) \) is the set of states for which there is a strategy \( \tau \) of ROMEO and a string \( w \in \Sigma^* \) such that \( w = \text{word}(u, \sigma, \tau) \) and \( \delta'(q, w) = p \), or \( \bot \) if \( \bot \in \text{words}(u, \sigma) \). The definition of the effect \( E[u] \) uses normalized sets of relative effects as JULIET can always restrict herself to strategies with minimal relative effects.

**Lemma 5.** Let \( u \) be a string and \( G \) a context-free game. Then, \( u \in \text{safe}_{L2R}(G) \) if and only if there is a relative effect \( e \in E[u] \) for which \( e \subseteq F \).

**Proof.** The latter condition is equivalent to the existence of a strategy for JULIET for which all states that can be reached by counter-strategies of ROMEO are in \( F \) and therefore is equivalent to \( u \in \text{safe}_{L2R}(G) \).

If we want to stress the game relative to which an effect is defined, we add a superscript to this notation as in \( E^G[s] \) or in \( e^G(\sigma, s, q) \).

It should be noted that strategies of JULIET for which ROMEO has a non-terminating counter strategy are not reflected in the effect of a word \( u \). We tacitly assume that JULIET will always follow a strategy that guarantees termination (and such strategies are always available as JULIET can simply stick to Read moves).

Henceforth, we will often consider relative effects and effects without having an underlying word \( u \) at hand. An (abstract) relative effect is just an element of \( \mathcal{P}(Q) \). An (abstract) effect is a mapping \( E \) of type \( Q \to \mathcal{P}(\mathcal{P}(Q)) \), such that every \( E[q] \) is normal. We denote the set of all abstract effects by \( \mathcal{E} \).

\(^3\)As always, we assume that the target automaton \( A(T) \) is fixed.
Composition.
We next define the composition operation \( \circ \) for effects. If \( E_1 = E[u] \) and \( E_2 = E[v] \) then \( E_1 \circ E_2 \) should just be \( E[u \cdot v] \). However, we need a definition of \( \circ \) for abstract effects, that is, a definition that is independent of the strings \( u \) and \( v \).

The definition uses the operation \( \text{Mix} \), which is defined on sets of sets of sets. Let \( D = \{ D_1, \ldots, D_n \} \) be a set of sets of sets. Then \( \text{Mix}(D) \) is the set

\[
\text{Norm}(\{ d_1 \cup \cdots \cup d_n \mid d_i \in D_1 \land \cdots \land d_n \in D_n \}).
\]

In other words, the \( \text{Mix} \) operation computes every way of taking the union of one element from each of \( D_1, \ldots, D_n \).

We define the composition \( E_1 \circ E_2 \) of two abstract effects \( E_1, E_2 : Q \to \mathcal{P}(\mathcal{P}(Q)) \) as follows.

\[
(E_1 \circ E_2)(q) = \text{Norm}(\bigcup_{X \in E_1(q)} \text{Mix}(\{E_2(p) \mid p \in X\})).
\]

Intuitively, for all sets \( X \) that \( \text{JULIET} \) can choose from \( E_1(q) \), \( \text{JULIET} \) can answer each choice of a state \( p \in X \) by ROMEO with a strategy from \( E_2(p) \). The resulting state sets, for each \( X \) have to be put together into one set of states that \( \text{JULIET} \) can enforce by some strategy.

**Lemma 6.** Let \( u, v \) be strings. Then \( E[u] \circ E[v] = E[u \cdot v] \).

**Proof.** We show that, for each \( q \), it holds that, for each relative effect \( e \in (E[u] \circ E[v])(q) \) there is a relative effect \( e' \in E[u \cdot v](q) \) with \( e' \subseteq e \) and vice versa. The statement of the lemma then follows by minimality of relative effects.

Let \( e \in (E[u] \circ E[v])(q) \) be a relative effect. We show that there is a relative effect \( e' \in E[u \cdot v](q) \) such that \( e' \subseteq e \).

By definition of \( \circ \) there is a relative effect \( X = \{ q_1, \ldots, q_k \} \in E[u](q) \) and relative effects \( e_i' \in E[v](q_i) \), for each \( i \), such that \( e = \bigcup_{i=1}^k e_i' \).

We denote the strategy of \( \text{JULIET} \) on \( u \) yielding \( X \) by \( \sigma_1 \) and the strategies on \( v \) yielding \( e_1', \ldots, e_k' \) (from \( q_1, \ldots, q_k \), respectively) by \( \sigma_2', \ldots, \sigma_k' \), respectively.

We define a strategy \( \sigma \) on \( uv \) for \( \text{JULIET} \). In the first phase, on \( u \), \( \text{JULIET} \) plays according to \( \sigma_1 \). If \( y \) is the word by which \( u \) is rewritten in the game on \( u \), then \( \delta'(q, y) = q_i \), for some \( i \in \{1, \ldots, k\} \). In the second phase, on \( v \), \( \text{JULIET} \) plays according to strategy \( \sigma_i \).

We claim that for \( e' = (\sigma, uv, \tau) \) holds \( e' \subseteq e \). Let \( p \in e' \) be arbitrarily chosen. Thus, there is a strategy \( \tau \) of ROMEO such that the word \( w = \text{word}(uv, \sigma, \tau) \) fulfills \( \delta'(q, w) = p \). We can write \( w \) as \( w_1 \cdot w_2 \), where \( w_1 \) is the rewriting of \( u \) and \( w_2 \) the rewriting of \( v \) in the game following \( \sigma \) and \( \tau \).

By definition of \( X = e(\sigma_1, u, q) \) and the definition of \( \sigma \) it follows that \( \delta'(q, w_1) \in X \) and thus \( \delta'(q, w_1) = q_i \), for some \( i \). Therefore, \( \text{JULIET} \) plays according to \( \sigma_i' \) in the game on \( v \) and consequently \( \delta'(q, w_2) \in e_i' \subseteq e \). Together,

\[
p = \delta'(q, w) = \delta'(q, w_1) = \delta'(q, w_2) \subseteq e_i' \subseteq e,
\]

as required.

Next we show that for each relative effect \( e' \in E[uv](q) \) there is a relative effect \( e \in (E[u] \circ E[v])(q) \) with \( e \subseteq e' \).

Let \( e' \in E[uv](q) \) be a relative effect and let \( \sigma \) be a strategy of \( \text{JULIET} \) such that \( e' = e(\sigma, uv, q) \). Let \( \sigma' \) denote the strategy of \( \text{JULIET} \) on \( u \) that is induced by \( \sigma \) and let \( X = e(\sigma', u, q) = \{ q_1, \ldots, q_k \} \).

Let \( w_1, \ldots, w_k \) be words from \( \text{words}(u, \sigma') \) such that, for every \( i \), \( \delta'(q_i, w_i) = q_i \) and let \( \tau_1, \ldots, \tau_k \) be corresponding strategies of ROMEO on \( u \). For every \( i \), let \( \sigma_i \) denote the strategy of \( \text{JULIET} \) on \( v \) induced by \( \sigma \) from configuration \((w_i, v) \) on and let \( X_i = e(\sigma_i, v, q_i) \). Finally, let

\[
e = \bigcup_{i=1}^k X_i \in \text{Mix}(\{ E[v](p) \mid p \in X \}).
\]

We claim that \( e \subseteq e' \). Let \( p \) be an arbitrary state in \( e \), thus \( p \in X_i \), for some \( i \). There exists a strategy \( \tau' \) of ROMEO on \( v \) such that for the word \( z = \text{word}(e, \sigma_i, \tau') \) holds \( \delta \ast (q, z) = p \). Combining \( \tau_i \) (on \( u \) and \( \tau' \) (on \( v \) yields a strategy \( \tau \) for ROMEO such that \( \text{word}(uv, \sigma, \tau) = w_i z \). Furthermore, \( \delta'(q, w_i z) = \delta'(q, w_i), z) = p \) and thus \( p \in e'(\sigma, uv, q) \).

5. AUTOMATA FOR L2R STRATEGIES

In this section, we define, for each context-free game \( G \), NFAs \( A_{L2R}(G) \) and \( A_{L2R}(G) \) for \( \text{safe}_{L2R}(G) \) and \( \text{safe}_{L2R}(G) \), respectively. One of them, \( A_{L2R}(G) \), is based on the computation of relative effects of the form \( e(\sigma, u, q) \) for strategies \( \sigma \) of \( \text{JULIET} \), whereas the other is based on the computation of dual effects (to be defined below) of the form \( e(\tau, u, q) \) for strategies \( \tau \) of ROMEO. Besides defining these automata and proving their correctness we also show how they can be computed in exponential time from \( G \).

5.1 Definition and Correctness of L2R automata

**Definition 3.** Let \( G = (\Sigma, R, T) \) be a context-free game with a DFA \( A(T) = (Q, \Sigma, \delta, q_0, F) \) for \( T \). The NFA \( A_{L2R}(G) = (Q_{L2R}, \Sigma, \delta_{L2R}, (q_0_1), F_{L2R}) \) is defined as follows:

- \( Q_{L2R} = \mathcal{P}(Q) \);
- \( \delta_{L2R}(X, s) = \text{Mix}\{E[s](q) \mid q \in X\} \), for each \( X \subseteq Q \) and \( s \in \Sigma \);
- \( F_{L2R} = \mathcal{P}(F) \).

**Proposition 7.** Let \( G = (\Sigma, R, T) \) be a context-free game with a DFA \( A(T) = (Q, \Sigma, \delta, q_0, F) \) for \( T \). Then \( L(A_{L2R}(G)) = \text{safe}_{L2R}(G) \).

**Proof.** We show by induction on \( |u| \) that for every string \( u \in \Sigma^* \) we have \( \text{Norm}(\delta_{L2R}(\{q_0\}, u)) = E[u](q_0) \):

For \( u = e \), \( \text{Norm}(\delta_{L2R}(\{q_0\}, e)) = \{q_0\} = E[e](q_0) \).

For \( u = uv \) we get

\[
N(\delta_{L2R}(\{q_0\}, uv)) = N(\bigcup_{X \in \delta_{L2R}(\{q_0\}, u)} \delta_{L2R}(X, s)) = N(\bigcup_{X \in \mathcal{P}(\mathcal{P}(Q))} \delta_{L2R}(X, s)) = N(\bigcup_{X \in \mathcal{P}(\mathcal{P}(Q))} \text{Mix}(\{E[s](q) \mid q \in X\})) = \text{Mix}(\{E[u](p) \mid p \in X\}) = E[u](q_0).
\]

We can conclude as follows that \( \text{JULIET} \) has a L2R winning strategy on \( u \) if and only if \( A_{L2R}(G) \) accepts \( u \).
5.2 Computing L2R automata

**Proposition 8.** There is an algorithm that computes in exponential time the NFA $A_{L2R}(G)$ for every context-free game $G = (\Sigma, R, T)$, provided that $T$ is represented by a DFA $A(T) = (Q, \Sigma, \delta, q_0, F)$ and the sets $R_j$ are represented by DFAs, NFAs or regular expressions.

**Proof.** The algorithm first computes in exponential time the effect $E[s]$, for every symbol $s \in \Sigma$. To this end, it uses Algorithm 1 below. By Proposition 9 below, this is possible in exponential time. The construction of $A_{L2R}(G)$ is then straightforward. It should be noted that $\text{Mix}(\{E[s](q) \mid q \in X\})$ can be computed in exponential time as $|\{E[s](q) \mid q \in X\}| \leq |Q|$ and each set $E[s](q)$ is of at most exponential size.

We next show how to compute the effect $E[s]$ for every symbol $s$ of a context-free game $G$ by a monotone fixed-point computation in exponential time. The pseudo-code of our algorithm is stated as Algorithm 1 below.

The algorithm uses a variable $P(s, q)$ for every symbol $s$ and every state $q \in Q$, intended to represent $E[s](q)$ and maintains the invariant $P(s, q) \subseteq E[s](q)$. In other words, for each $X$ in $P(s, q)$, there is an L2R-strategy of JULIET such that $X = o(\sigma, q, s)$. Slightly abusing notation, we write $P[s]$ for the function defined by $q \mapsto P(s, q)$. It should be noted that during the computation the functions $P[s]$ need not be “real effects” in the sense that there is some string $u$ with $P[s] = E[u]$. They are rather “partial effects”, that is, arbitrary functions of type $Q \to \mathcal{P}(\mathcal{P}(Q))$.

In the description of the algorithm, we use $P[w]$ as a shorthand for $P[a_1] \circ \cdots \circ P[a_r]$, where $a_1 \cdots a_r = w$ and the operation $\circ$ is defined just as for effects.

**Algorithm 1.** Compute the effects of symbols from $\Sigma$.

1. for all $s \in \Sigma, q \in Q$ do
2. \hspace{1em} $P[s](q) \leftarrow \{\delta(q, s)\}$
3. \hspace{1em} while some set $P[s](q)$ has changed in the previous iteration do
4. \hspace{2em} for all $f \in \Gamma, q \in Q$ do
5. \hspace{3em} $P[f](q) \leftarrow P[f](q) \cup \text{Mix}(\{P[w](q) \mid w \in R_j\})$
6. \hspace{3em} $P[f](q) \leftarrow \text{Norm}(P[f](q))$

**Proposition 9.** Algorithm 1 computes, for every context-free game $G = (\Sigma, R, T)$, the effect $E[s]$, for every symbol $s \in \Sigma$. Provided that $T$ is represented by a DFA $A(T) = (Q, \Sigma, \delta, q_0, F)$ and the sets $R_j$ are represented by DFAs, NFAs or regular expressions it can be carried out in exponential time.

**Proof.** We first show how the algorithm can be implemented such that it runs in exponential time. We assume without loss of generality that all sets $R_j$ are represented by NFAs.

As every set $P[s](q)$ can only contain sets of at most exponential size (in $|Q|$), the number of iterations of the while loop is at most exponential. The implementation of line 6 will make sure that $P[f](q)$ is always normal. It thus only remains to show how to implement a single execution of line 6, $P[f](q) \leftarrow P[f](q) \cup \text{Mix}(\{P[w](q) \mid w \in R_j\})$.

The idea is to cycle through all sets $U \subseteq Q$ that do not yet have a subset in $P[f](q)$, to test whether $U \in \text{Mix}(\{P[w](q) \mid w \in R_j\})$, and to add it to $P[f](q)$ if this is the case. We do this in a bottom up fashion, starting with the singleton subsets of $U$, then testing the subsets of size 2 and so on.

Given a set $U$ that does not yet have a subset in $P[f](q)$, we test whether, for each $w \in R_j$, there is a set $W \in P[w](q)$ with $W \subseteq U$. If this is not the case, then $U \in \text{Mix}(\{P[w](q) \mid w \in R_j\})$. If, on the other hand, this is the case, then there is a subset $U'$ of $U$ that belongs to $\text{Mix}(\{P[w](q) \mid w \in R_j\})$. In fact, we must have $U = U'$, since otherwise, we would have already added $U'$ to $P[f](q)$, and not considered $U$ for testing.

The test above can be implemented with the help of a suitable automaton. An NFA $B$ is constructed that accepts all strings $w \in \Sigma^*$ for which there is a set $W \in P[w](q)$ with $W \subseteq U$. This automaton is defined as $A_{L2R}(G)$ in Definition 3 below, but with the following modifications.

- The initial state is $\{q\}$;
- The set of accepting states is $\mathcal{P}(U)$;
- The transition function is defined with the sets $P[s](p)$ in place of $E[s](p)$, for symbols $s$ and states $p$.

That $L(B)$ is as stated above can be shown in analogy to the proof of Proposition 7. Whether, for each $w \in R_j$, there is a set $W \in P[w](q)$ with $W \subseteq U$, can then be tested by checking whether $R_j \subseteq L(B)$. This latter test asks whether the language of an NFA of polynomial size is contained in the language of an NFA of exponential size. It can be translated into a nonemptiness test for an automaton of exponential size (the intersection of $B$ with the complement of the automaton for $R_j$) and is thus doable in polynomial space.

It remains to show that the algorithm is also correct, that is, that after its termination it holds $P[s] = E[s]$, for every $s \in \Sigma$. We do this in two steps.

We make use of the following notation. Let $P[s]$ denote the value of $P[s]$ after the $j$-th iteration of the WHILE loop. For a strategy $\sigma$ of JULIET and a string $u \in \Sigma^*$, we write $\text{Depth}(\sigma, u)$ for the maximum nesting depth of Call moves in any play $\Pi(\sigma, u)$. If the nesting depth is unbounded, we let $\text{Depth}(\sigma, u) = \omega$.

We first show the following claim.

**Claim 1.** For every $j \geq 0$, for all symbols $\sigma \in \Sigma$ and for all $q \in Q$, $P[s](q)$ contains exactly the relative effects $e(\sigma, s, q)$, for all strategies $\sigma$ of JULIET with $\text{Depth}(\sigma, s) \leq j$.

We prove Claim 1 by induction on $j$. 

---

\footnote{We point out that the current proof is similar to the proof of Proposition 7 but not on the proof of Proposition 8. Rather the proof of Proposition 8 will be based on the current proof.}
For $j = 0$ this holds true as each $P^j[s(q) = q] = \{\emptyset(q, s)\}$ is just the set with the relative effect corresponding to the strategy of Juliet that reads $s$ in its very first step.

Now let $j > 0$ and let the induction hypothesis hold for all $m < j$. We need to prove the induction step only for symbols $f \in \Gamma$ (as opposed to $s \in \Sigma$), as symbols in $\Sigma \setminus \Gamma$ only have depth-0 strategies for Juliet.

As $\{P^{-i}[w(q) = w] \mid w \in R_j\}$ is a finite set, there are $n \in \mathbb{N}$, and strings $w_1, \ldots, w_n \in R_j$ such that $\{P^{-i-1}[w(q) = w] \mid w \in R_j\} = \{P^{-i}[w_i(q) \mid i \in \{1, \ldots, \ell\}\}$. For each $w$ we denote by $i(w)$ the number in $\{1, \ldots, \ell\}$ for reference, we denote the set $\{w_1, \ldots, w_n\}$ by $\{w_i, \ldots, w_n\}$. Let $e$ be a relative partial effect in $P^{-i}[f(q)]$. If $e \in P^{-i-1}[f(q)]$, then $e = e(\sigma, f, q)$ for some strategy $\sigma$ of Juliet with $\text{Depth}(\sigma, f, q) \leq j - 1$, by induction. Thus, we assume $e \in P^{-i}[f(q)] \setminus P^{-i-1}[f(q)]$ and thus $e$ has arrived in $P^{-i}[f(q)]$ in the $i$-th iteration of the WHILE loop. Therefore, $e \in \text{Min}(\{P^{-i-1}[w(q) \mid w \in R_j\})$. Furthermore, there are relative partial effects $e_{i, \ldots, e_{j}}$ such that

- $e_i \in P^{-i}[w_i(q)]$, for every $i$, and
- $e = e_1 \cup \cdots \cup e_\ell$.

By induction and the correctness of $\circ$ we can conclude that, for each $i$, there is a strategy $\sigma_i$ of Juliet on $w_i$, of depth $\leq j - 1$ such that $e_i = e(\sigma_i, w_i, q)$.

The strategy $\sigma$ of depth $j$ now can be obtained as follows. In the first round, Juliet does a Call Move. Then, we assume Romeo chooses a string $w \in R_j$ she follows the strategy $\sigma'$ such that $e(\sigma', w, q) = e_\ell$, for the $i \in \{1, \ldots, \ell\}$ with $P^{-1-\ell}[w(q) = P^{-i-1}[w_i(q)]$. Thus, $e(\sigma, w, q) = e_1 \cup \cdots \cup e_\ell = e$.

Conversely, let $e$ be a strategy of Juliet on $f$ of depth at least $1$ and at most $j$ and let $e = e(\sigma, f, q)$. The first step of Juliet, following $\sigma$, is a Call on $s$. For each $i \in \{1, \ldots, \ell\}$ let $\sigma_i$ be the strategy of Juliet that is induced by $\sigma$ on $w_i$ and let $e_i = e(\sigma_i, w_i, q)$. Now for each possible reply $w \in R_j$ of Romeo, let $\sigma_w$ be the strategy that yields $e(\sigma_i(w), w_i(w), q)$ and let $e_w = e(\sigma_w, w)$. Thus, $\sigma = e \cup e_w = e_1 \cup \cdots \cup e_\ell$.

Clearly, each strategy $\sigma_w$ and in particular every $\sigma_{e_i}$ has a Call depth $\leq j - 1$ on $w_i$. Thus, by induction we conclude that $e_i \in P^{-i}[w_i(q)]$, for every $i$ and therefore $e \in \text{Min}(\{P^{-i-1}[w(q) \mid w \in R_j\}$, as required.

This concludes the proof of the Claim 1.

So far we have not ruled out that there might be a strategy $\sigma$ of Juliet with unbounded Call depth such that $e(\sigma, f, q) \notin P^j[s(q)]$, at the end of the computation of Algorithm 1. To bridge the gap, we use an additional game $G'$ that is obtained from $G$ by a restriction to finite rule sets as follows. For each $f$ and each string $w \in R_j$ let $v(f, w)$ be a string of minimal length such that $v(f, w) \in R_j$ and $E[v(f, w)] = E[w]$.

Let $S$ be the union of the set of all strings of the form $v(f, w)$ with all sets of the form $S(j, q, f)$ that were defined in the proof of Claim 1. Let $G' = (\Sigma, R', T)$, where, for each $f, R'_f = R_f \cap S$.

As all sets $S(j, q, f)$ are finite and there are only finitely many effects with respect to $G$, $S$ is a finite set and thus all sets $R'_f$ are finite as well.

CLAIM 2. For every symbol $s$, $E^G[s] = E^{G'}[s]$.

Obviously, for each $s$ and $q$ and every finite $G'$-strategy $\sigma$ of Juliet, the $G'$-strategy $\sigma'$ that is induced by $\sigma$ fulfills $e^{G'}(\sigma', s, q) \subseteq e^{G'}(\sigma', s, q)$, simply because all plays in $G'$ are also plays in $G$.

To complete the proof of Claim 2 it thus suffices to prove the following claim.

CLAIM 3. For every string $u \in \Sigma^*$, state $q$ and finite $G'$-strategy $\sigma'$ of Juliet there is a finite $G'$-strategy $\sigma$ with $e^{G'}(\sigma', u, q) \not\subseteq e^{G'}(\sigma', u, q)$.

We first observe that $\text{Depth}^{G'}(\sigma', u) < \omega$ Otherwise, the strategy tree induced by $\sigma'$ on $u$ would be a finitely branching tree with arbitrarily long branches and thus would contain infinite branches by König's Lemma, contradicting finiteness.

Thus, we can show Claim 3 by induction on $\text{Depth}^{G'}(\sigma', u)$. The case $\text{Depth}^{G'}(\sigma', u) = 0$ is simple as $G'$ and $G$ coincide as long as no Calls are made (as in the play on $u$ following $\sigma'$).

For the induction step, let $\text{Depth}^{G'}(\sigma', u) = k > 0$ and let us assume that the claim holds for smaller depths.

Let $e' = e^{G'}(\sigma', u, q)$. We consider two cases.

The first case is that $u = sw$, for some $s \in \Sigma$ and $w \in \Sigma^*$ and $\sigma'$ plays a Read on $s$.

In this case, we can conclude that $e^{G'}(\sigma', w, p) = e'$ and that $\text{Depth}^{G'}(\sigma'_w, w) < k$, where $p = \delta(q, s)$ and $\sigma'_w$ is the strategy of Juliet on $w$ induced by $\sigma'$. Thus, by induction, there is a $G'$-strategy $\sigma_w$ with $e^{G'}(\sigma_w, w, p) \not\subseteq e^{G'}(\sigma_w, w, p)$. Combining $\sigma_w$ with an initial Read on $s$ yields the desired strategy $\sigma$.

The second case is that $u = fw$, for some $f \in \Sigma$ and $w \in \Sigma^*$ and $\sigma'$ plays a Call on $f$.

We define $\sigma$ as follows. For each $z \in R_f$, $v(z, f)$ is a possible answer of Romeo in both games $G$ and $G'$. As $\text{Depth}^{G'}(\sigma', v(z, f)) < k$, induction yields a finite $G'$-strategy $\sigma_z$ with $e^{G'}(\sigma_z, v(z, f), w) \not\subseteq e^{G'}(\sigma_z, v(z, f), w) = e'$. As $E^{G'}[v(f, w)] = E^{G}[v]$, there is a strategy $s_z$ for Juliet with $e^{G'}(\sigma_z, v(z, f), q) \not\subseteq e^{G'}(\sigma_z, z, q)$.

We define strategy $\sigma$ for the case that Romeo replies by $z$ on $fw$ as follows. It plays according to $\sigma_z$ on $z$ and then follows the strategy induced by $\sigma_{z_1}$ on $w$. Altogether, we can conclude $e^{G'}(\sigma, f, w, q) \not\subseteq e'$, as required.

This completes the proof of Claim 3 and also the proof of Claim 2.

To complete the proof of the proposition it suffices to observe that, by the choice of the set $S$, the output of Algorithm 1 on input $G$ is the same as on input $G'$.

As all rule sets in $G'$ are finite, every finite strategy $\sigma'$ of Juliet contributing to $E^{G'}[s]$ are of bounded Call depth. Otherwise, the strategy tree induced by $\sigma'$ on a symbol $s$ would be a finitely branching tree with arbitrarily long branches and thus would again contain infinite branches by König's Lemma, contradicting finiteness.

Therefore, Algorithm 1 computes, on input $G$ or $G'$, all
effects $E^*|\sigma|$ correctly and thus, by Claim 2, also all effects of $G$. □

5.3 Automata for strategies of Romeo

For the proof of Theorem 17 below, we need an NFA of exponential size for $e^* \setminus \text{safe}_{L2R}(G)$. As the complementation of $A_{L2R}(G)$ might yield an automaton of doubly exponential size (in $|G|$), we follow a different approach by constructing an NFA for $e^* \setminus \text{safe}_{L2R}(G)$ that works analogous as $A_{L2R}(G)$ but is based on strategies of ROMEO. To this end, we define dual effects and the dual automaton $\hat{A}_{L2R}(G)$ next.

Definition 4. Let $G$ be a context-free game, $u$ a string, $q$ a state of the target automaton and $\tau$ a strategy of ROMEO. The dual relative effect $\hat{e}(\tau, u, q)$ is the set $\{\delta'(q, w) = \text{word}(u, \tau, \sigma), \sigma \in \text{STRAT}_{L2R}(G)\}$.

The dual effect $\hat{E}[u]$ of $u$ maps every state $q$ to the normalized set of dual relative effects $\hat{e}(\tau, u, q)$ of $u$ for all strategies $\tau \in \text{STRAT}_{\text{Romeo}}(G)$.

For the sake of clarity, we will sometimes refer to non-dual (relative) effects as primal (relative) effects.

The informal meaning of dual relative effects is dual to the informal meaning of primal relative effects: $\hat{e}(\tau, u, q)$ is the set of states, for which there is a strategy $\sigma$ of JULIET and a string $w \in \Sigma^*$ such that $w = \text{word}(u, \tau, \sigma)$ and $\delta'(q, w) = p$. We note that non-terminating plays do not contribute to dual effects, as for every strategy $\tau$ there is a strategy $\sigma$ of JULIET that yields a finite play (e.g., the strategy that always does Read), and thus reflecting non-terminating plays in $\hat{e}(\tau, u, q)$ would not have any consequences.

The dual effect of a string can be obtained from its primal effect via a simple operation, SMIX, very similar to the Mix operation used in previous sections. Let $D = \{D_1, \ldots, D_n\}$ be a set of sets. Then

$\text{SMIX}(D) = \text{NORM}\{\{d_1, \ldots, d_n\} \mid d_1 \in D_1 \land \cdots \land d_n \in D_n\}$.

In other words, SMIX contains all sets that can be formed by selecting one element from each of the elements of $D$. Notice that, while MIX takes a set of sets of sets and returns a set of sets, SMIX takes a set of sets and returns a set of sets.

Lemma 10. Let $u$ be a string and $q \in Q$ a state of $A(T)$. Then $\hat{E}[u](q) = \text{SMIX}(\hat{E}[u](q))$.

Proof. As both sets are normal it suffices, thanks to Lemma 3, to show that for every $\hat{e} \in \hat{E}[u](q)$ there is some $e \in \text{SMIX}(\hat{E}[u](q))$ such that $e \subseteq \hat{e}$ and vice versa.

Let $\hat{e} \in \hat{E}[u](q)$ and let $\tau$ be a strategy of ROMEO such that $\hat{e} = \hat{e}(\tau, u, q)$. By definition, $\hat{e} = \{\delta'(q, w) \mid w = \text{word}(u, \tau, \sigma), \sigma \in \text{STRAT}_{L2R}(G)\}$. This means that for every $\sigma \in \text{STRAT}_{L2R}(G)$ there is a state in $e(\sigma, u, q)$ that also belongs to $\hat{e}$. In particular, there is an element $e$ in $\text{SMIX}(\hat{E}[u](q))$ such that $e \subseteq \hat{e}$.

For the other direction, consider $e \in \text{SMIX}(\hat{E}[u](q))$. By definition of $\hat{E}[u](q)$ and $\text{SMIX}$, for every finite strategy $\sigma$ of JULIET there is a strategy $\tau$ of ROMEO such that $\delta'(q, w) \in e$, where $w = \text{word}(u, \sigma, \tau)$.

Let $t = \text{Tree}_{G,u}$ be the (full) game tree for $u$. Let $L_e$ be the set of leaves of $t$ that are labeled by configurations $(l, u, e)$ with $\delta'(q, w) \in e$. Let $S_e$ be the set of nodes of $t$ such that for every strategy $\sigma$ of JULIET, the subtree of the strategy tree $\text{Tree}_{G,u}(\sigma)$ rooted in $n$ either has an infinite branch or a leaf in $L_e$.

The root of $t$ must belong to $S_e$. Otherwise, JULIET would have a finite strategy $\sigma$ such that no strategy of ROMEO yields a state in $e$, contradicting the above statement about $e$. Furthermore, if a node in $t$ belongs to $S_e$ and is labeled by a configuration where JULIET is to move, then all its children belong to $S_e$. If a node in $t$ belongs to $S_e$ and ROMEO is to move, then at least one of its children belongs to $S_e$. We can define a strategy $\tau$ for ROMEO that from a node in $S_e$ always selects a child node in $S_e$. In the strategy tree $\text{Tree}_{G,u}(\tau)$, every node belongs to $S_e$. This immediately implies that $\hat{e}(\tau, u, q) \subseteq e$. □

From this connection it follows that composition of dual effects just works like composition of effects. In the following, the operation “$\circ$” for dual effects is defined exactly as for effects.

Lemma 11. Let $u, v$ be strings. Then $\hat{E}[u] \circ \hat{E}[v] = \hat{E}[uv]$.

Proof. This follows from definition 4 exactly like the corresponding statement for primal effects by reversing the roles of JULIET and ROMEO in the proof of lemma 6. □

Now we are ready to define the dual automaton $\hat{A}_{L2R}(G)$ for $e^* \setminus \text{safe}_{L2R}(G)$.

Definition 5. Let $G = (\Sigma, R, T)$ be a context-free game with a DFA $A(T) = (Q, \Sigma, \delta, q_0, F)$ for $T$. The NFA $\hat{A}_{L2R}(G) = (\hat{Q}_{L2R}(G), \Sigma, \delta_{L2R}, \{q_0\}, \hat{F}_{L2R})$ is defined as follows:

- $\hat{Q}_{L2R}(G) = \mathcal{P}(Q)$;
- $\delta_{L2R}(X, s) = \text{Mix}(\{\hat{E}[s](q) \mid q \in X\})$, for each $X \subseteq Q$ and $s \in \Sigma$;
- $\hat{F}_{L2R} = \mathcal{P}(Q \setminus F)$.

Proposition 12. Let $G = (\Sigma, R, T)$ be a context-free game with a DFA $A(T) = (Q, \Sigma, \delta, q_0, F)$ for $T$. Then $L(\hat{A}_{L2R}(G)) = (\hat{Q}_{L2R}(G), \Sigma, \delta_{L2R}, \{q_0\}, \hat{F}_{L2R})$ is a context-free game with $\text{LEXIC} \circ \text{LEXIC}$.

Proof. As for $A_{L2R}(G)$, we first show that $\text{NORM}(\delta_{L2R}^{s}(\{q_0\}, u)) = \hat{E}[u](q_0)$, by induction on $|u|$.

For $u = \epsilon$ we have

$\text{NORM}(\delta_{L2R}^{s}(\{q_0\}, \epsilon)) = \{\{q_0\}\} = \hat{E}[\epsilon](q_0)$.

For $u = vs$ we get

$\text{N}(\delta_{L2R}^{s}(\{q_0\}, vs)) = \{\{q_0\}\}$

We can conclude as follows that ROMEO has a L2R winning strategy on $u$ if and only if $\hat{A}_{L2R}(G)$ accepts $u$. 

□
\( u \in \Sigma^* \setminus \text{safe}_{L2R}(G) \iff \exists \epsilon \in \hat{E}(u)(q_0) : \epsilon \cap F = \emptyset \)
\( \hat{E}(u)(q_0) \cap \mathcal{P}(Q \setminus F) \neq \emptyset \)
\( N_{\hat{\delta}_{L2R}}(\{q_0\}, u) \cap \mathcal{P}(Q \setminus F) \neq \emptyset \)
\( \hat{\delta}_{L2R}(\{q_0\}, u) \cap \mathcal{P}(Q \setminus F) \neq \emptyset \)

\[ \square \]

**Proposition 13.** There is an algorithm that computes in exponential time the NFA \( A_{L2R}(G) \) for each context-free game \( G = (\Sigma, \delta, q_0, F) \) and the sets \( R_f \) are represented by DFAs, NFAs or regular expressions.

**Proof.** Similar to the algorithm computing \( A_{L2R}(G) \), this algorithm first computes in exponential time the effects \( E[s] \), for every symbol \( s \in \Sigma \). From these, it computes \( \hat{E}[s] \), for every \( s \in \Sigma \), via \( E[s](q) = \text{SMIX}(E[s](q)) \), for every \( q \in Q \). Each computation of a set \( \text{SMIX}(E[s](q)) \) can be done in exponential time, similarly as for line 5 of Algorithm 1. To this end, one can test, for every set \( X \subseteq Q \), whether it can be obtained by picking elements from the sets in \( E[s](q) \). The sets \( \text{SMIX}(E[s](q) : q \in X) \) can be computed in exponential time as well in a straightforward fashion.  

**6. UPPER BOUNDS**

In this section, we prove the lower bounds results of our Main Theorem 1. The problem \( L2RALL \) is decidable and can actually be decided in exponential space. If all rule subsets are finite and given in the input explicitly, then the problem can be decided in exponential time.

Before we describe the algorithm for \( L2RALL \), we state two auxiliary results that allow us to consider only finite subsets of replacement languages.

For any string \( w \), let \( F(w) = \{ q \in Q \mid E[w](q) \cap \mathcal{P}(F) \neq \emptyset \} \) be the set of states from which \( \text{JULIET} \) has a winning strategy on \( w \).

For a state \( q \) and a set \( S \) of states let \( A_{L2R}^q[S] \) denote the automaton that is obtained from \( A_{L2R}(G) \) by chosing \( q \) as initial state and \( \mathcal{P}(S) \) as set of accepting states.

**Lemma 14.** For every state \( q \) and \( w \in \Sigma^* \) the automaton \( A_{L2R}^q[F(w)] \) accepts exactly the strings \( v \) such that there is a winning strategy for \( \text{JULIET} \) on \( vw \) starting at state \( q \) in \( A(T) \).

The proof is similar to the proof of Proposition 7.

For a state \( q \in Q \) let \( G_q \) denote the game obtained from \( G \) by chosing the state \( q \) as initial state of the target automaton.

**Lemma 15.** Let \( q \in Q, w \in \Sigma^* \) and \( f \in \Gamma \). If there is a string \( v \in R_f \) such that \( vw \in \text{safe}_{L2R}(G_q) \) then there is a string \( v' \) of length at most \( |Q_f| \cdot 2^{|Q|} \) such that \( v'w \in \text{safe}_{L2R}(G_q) \).

**Proof.** This follows from Lemma 14 and a standard pumping argument for the product automaton \( B \) combining \( A_{L2R}(G) \) and \( A(R_f) \): For any two states \( (x_1, p_1), (x_2, p_2) \in Q_{L2R} \times Q_f \) there is a string \( v \) with \( \delta_B((x_1, p_1), v) = (x_2, p_2) \) and only if there is such a string \( v' \) of length at most \( |Q_{L2R} \times Q_f| = |Q_f| \cdot 2^{|Q|} \).

**Theorem 16.** For context-free games with regular replacement languages, \( L2RALL \in \text{EXPSPACE} \)

**Proof.** We give a nondeterministic exponential-space algorithm \( A \) deciding \( L2RALL \), the complement of \( L2RALL \). This yields the result since \( \text{EXPSPACE} \) is closed under complement and \( \text{NEXPSPACE} = \text{EXPSPACE} \) thanks to Savitch’s Theorem [9].

The idea is that \( A \) guesses a symbol \( f \) in \( \Gamma \) and strings \( u, w \) such that \( uw \in \text{safe}_{L2R}(G) \) \( \\& \text{safe}_{L2R}(G) \) is a witness string on which \( \text{JULIET} \) plays Call in the first pass on \( uw \).

Thanks to Lemma 15, \( A \) only needs to verify that, for all replacement strings \( v \in R_f \) of length at most \( |Q_f| \cdot 2^{|Q|} \), it holds that \( uwv \in \text{safe}_{L2R}(G) \). A short summary of \( A \) is given as Algorithm 2.

**Algorithm 2 Test for \( G \in L2RALL \)**

1: Guess \( f \in \Gamma \) and a dual relative effect \( \hat{e}_w \)
2: while Guessing a string \( u \) in a streaming fashion do
3: Use \( A_{L2R}(G) \) to compute the set \( U = E[u](q_0) \) nondeterministically
4: Use \( A_{L2R}(G) \) to nondeterministically verify \( \hat{e}_u \in E[u](q_0) \)
5: Guess a string \( w \) and compute \( F(w) \) by simulating \( A_{L2R}(G) \) backwards
6: if \( \hat{e}_w \cap F(w) = \emptyset \) then
7: \( \text{// } uw \notin \text{safe}_{L2R}(G) \)
8: for all \( v \in R_f \) with \( |v| \leq |Q_f| \cdot 2^{|Q|} \) do
9: Guess a set \( U_v \in U \)
10: for all \( q \in U_v \) do
11: Simulate \( A_{L2R}^{q,f} \) on input \( v \)
12: if \( A_{L2R}^{q,f} \) accepts \( v \) then
13: \( \text{// } uwv \notin \text{safe}_{L2R}(G) \)
14: else
15: Accept
16: Reject

The main challenge is that the string \( u \) may in general be of doubly exponential length and therefore cannot be stored.

Therefore, to compute the sets \( U = \{U_1, U_2, \ldots, U_n\} \), \( A \) guesses \( u \) in a streaming fashion, one symbol at a time. It simulates \( A_{L2R}(G) \) on \( u \) and computes \( E[w](q_0) \) online. This can be done in exponential space by storing the set \( E[w](q_0) \in \mathcal{P}(Q) \).

At the same time, having guessed a dual relative effect \( \hat{e}_w \), it guesses a run of \( A_{L2R}(G) \) on \( uf \) effectively verifying that there is a strategy corresponding to this relative effect.

Afterwards, to compute \( F(w) \in \mathcal{P}(Q) \), \( A \) guesses a string \( w \), and incrementally computes a set \( F(w) \subseteq Q \) of states from which \( \text{JULIET} \) can win the game as defined in Lemma 14. The set \( F(w) \) can be computed from the set \( F(w) \) by checking, for each \( q \in Q \), whether \( A_{L2R}^{q,f} \) accepts \( s \). As there are only exponentially many subsets of \( Q \) it is not hard to prove by a standard pumping argument that \( w \) can be chosen of exponential size and that its computation can be actually carried out in polynomial space. With \( F(e) = F \) the correctness of this incremental computation follows by a simple induction argument.

The algorithm then checks whether \( \hat{e}_w \) contains a state from \( F(w) \). If it does not, we know that \( uwv \notin \text{safe}_{L2R}(G) \). If it does, \( A \) immediately rejects.
Finally, \( A \) checks for all strings \( v \in R_f \) of length at most \( |Q_f| \cdot 2^{|Q_f|} \), if \( uwv \in \text{safe}_{L2R}(G) \). This can be done by (1) cycling through all strings \( v \) of this length, (2) checking if \( v \in R_f \) by simulating \( A(R_f) \) on \( v \) and (3) in case \( A(R_f) \) accepts \( v \), guessing a set \( U_v \in \mathcal{U} \) and testing whether for every \( q \in U_v \), there is a relative effect \( e \in E[v|q] \) such that \( e \subseteq F(w) \).

To perform test (3), \( A \) simulates, for each \( q \in U_v \), a run of \( A_{L2R}^{F(w)} \) on \( v \). This can be done in \( \text{PSPACE} \). If all runs succeed, \( A \) concludes that \( uwv \in \text{safe}_{L2R}(G) \), otherwise it rejects.

Altogether, \( A \) only requires exponential space; it remains to show that \( A \) accepts iff \( \text{safe}_{L2R}(G) \cap \text{safe}_{L2R}(G) \neq \emptyset \).

If \( A \) accepts, then there exists a string \( ufw \) such that (a) \( ufw \notin \text{safe}_{L2R}(G) \) (this follows directly from Lemma 14) and (b) for all \( v \in R_f \) of length at most \( |Q_f| \cdot 2^{|Q_f|} \) there exists a set \( U_v \in \mathcal{E}[u|q_0] \) such that \( v \) is accepted by \( A_{L2R}^{F(w)} \) for all \( q \in U_v \).

With Lemmas 14 and 15, it follows from (b) that for every \( v \in R_f \) there is a strategy \( \sigma_v \) of \( \text{JULIET} \) on \( u \) such that for all states \( \sigma \in \epsilon(u, \sigma, \sigma_0) \), \( \text{JULIET} \) has a winning strategy on \( uvw \) starting at \( q \).

This yields a winning \( L2R^+ \) strategy for \( \text{JULIET} \) on \( ufw \): In the first pass, \( \text{JULIET} \) calls \( f \). On the second pass, depending on \( \text{ROMEO} \)'s choice of \( v \), \( \text{JULIET} \) plays according to \( \sigma_v \) on \( u \) and is guaranteed to reach a state starting from which she has a winning strategy on \( uvw \).

For the "only if" part, assume \( \text{safe}_{L2R}(G) \cap \text{safe}_{L2R}(G) \neq \emptyset \). Then there exists a word on which \( \text{JULIET} \) has a winning \( L2R^+ \) strategy, but no winning \( L2R \) strategy. This word must be of the form \( ufw \) with \( f \) being the symbol \( \text{JULIET} \) calls on her first pass for some winning \( L2R^+ \) strategy \( \sigma \). In lines 1 through 4, \( A \) guesses this word.

Since \( \text{JULIET} \) has no winning \( L2R \) strategy on \( ufw \), \( \text{ROMEO} \) must have a strategy \( \tau \) on \( ufw \) such that \( \tau[ufw, \tau, \sigma_0] \cap \tau (w) = \emptyset \). Since this dual relative effect can be guessed by \( A \), the test on line 6 can be passed.

Let \( \sigma_v \in \text{JULIET} \)'s strategy on \( u \) in case \( \text{ROMEO} \) replaces \( f \) by \( v \in R_f \) and \( U_v = \epsilon(u, \sigma, \sigma_0) \in \mathcal{E}[u|q_0] \). Since \( \sigma \) is winning on \( uwv \), \( \text{JULIET} \) has a winning strategy on \( vuv \) starting at \( q \) for any \( q \in U_v \). Using Lemma 14, this means that for any \( v \in R_f \), \( A \) can guess a set \( U_v \in \mathcal{E}[u|q_0] = U \) on line 9 such that all \( A_{L2R}^{F(w)} \) accepts \( v \) for \( q \in U_v \). This condition is checked in lines 10 through 15, and since it is fulfilled for all \( v \in R_f \), \( A \) accepts.

For games \( G \) with finite replacement languages, this algorithm can be modified to run in exponential time in \(|G|\).

**Theorem 17.** \( L2RALL \in \text{EXPTIME} \) for games with finite replacement languages, given explicitly as part of the input.

**Proof.** We are going to modify Algorithm 2 such that it runs in exponential time. This works because the only NFAs of doubly exponential size that Algorithm 2 uses, can be replaced by NFAs of exponential size, if the replacement sets \( R_f \) are finite and explicitly given in the input.

Algorithm 2 uses nondeterminism for two kinds of purposes: for guessing effects and other sets and for guessing strings. The latter can be delegated to standard polynomial space non-emptiness tests for exponential size automata, while the former can be done by cycling through all possible candidates (as there are always only exponentially many).

To this end, the algorithm \( A' \) contains an outer loop over all \( f \in \Gamma \), sets \( W \subseteq Q \) and vectors of sets \( U_1, \ldots, U_{|R_f|} \in \mathcal{P}(Q) \). Inside this loop, similar to algorithm 2, \( A' \) checks if there are strings \( u \) and \( w \) such that \( U_1, \ldots, U_{|R_f|} \in \mathcal{E}[u|q_0] \) and \( W = F(w) \); then, all \( A' \)'s need to do is check for all \( i = 1, \ldots, |R_f| \) whether \( \delta_{L2R}(U_i, v_i) \cap \mathcal{P}(F(w)) \neq \emptyset \) (with \( R_f = \{v_1, \ldots, v_{|R_f|}\} \) and \( A_{L2R} \) accepts \( ufw \).

To verify the existence of a string \( u \) with \( U_1, \ldots, U_{|R_f|} \in \mathcal{E}[u|q_0] \), \( A' \) computes the product automaton of \( [R_f] \) copies of \( A_{L2R} \) and checks whether the product state \( (U_1, \ldots, U_{|R_f|}) \) is reachable in polynomial space (and thus exponential time).

To find a string \( w \) with \( W = F(w) \), \( A' \) computes the product automaton with one copy of \( A_{L2R}^{F(w)} \), for each \( q \) in \( W \); again, the verification of the existence of \( w \) is by a non-emptiness test.

Finally, \( A \) runs one copy of \( A_{L2R} \) with starting state \( U \) and final state set \( \mathcal{P}(W) \) on \( v_i \) for each \( i = 1, \ldots, |R_f| \) and \( A_{L2R} \) on \( ufw \); if all copies of \( A_{L2R} \) accept, \( A \) accepts, since a separating string \( ufw \) has been found.

The correctness of this algorithm follows similar to the proof of theorem 16, and since it loops an exponential number of times and takes no more than exponential time in each iteration, \( A' \) is an \( \text{EXPTIME} \) algorithm deciding \( L2RALL \) for games with finite replacement languages, given explicitly as part of the input.

7. **LOWER BOUNDS**

In this section, we prove the hardness results of the Main Theorem 1. More precisely, we show that \( L2RALL \) is \text{EXPSPACE}-hard in general and \text{EXPTIME}-hard for games with finite replacement sets.

**Proposition 18.** \( L2RALL \) is hard for \text{EXPSPACE}.

**Proof.** We give a reduction from the **Exponential Width Corridor Tiling** problem. In this problem, we are given a set \( U = \{u_1, \ldots, u_s\} \) of tiles, where \( u_1 \) is the bottom tile and \( u_s \) is the top tile. There are also two relations \( H, V \subseteq U \times U \). These are the *horizontal* and *vertical* constraints, respectively. A tile \( u_i \) is only allowed to the right of a tile \( u_j \), if \( u_i, u_j \in H \) and only allowed on top of \( u_i \), if \( u_i, u_j \in V \). We are also given a number \( n \) in unary notation.

Formally, a corridor tiling of width \( 2^n \) is a mapping \( c : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, m\} \to U \), for some \( m \). A tiling \( c \) is **valid** if

- \( c(0,0) = u_1 \),
- \( c(2^n - 1, m) = u_s \),
- for every \( i \in \{0, \ldots, 2^n - 2\} \) and \( j \in \{1, \ldots, m\} \), \( (c(i,j), c(i+1,j)) \in H \), and
- for every \( i \in \{0, \ldots, 2^n - 1\} \) and \( j \in \{1, \ldots, m - 1\} \), \( (c(i,j), c(i,j+1)) \in V \).

**Exponential Width Corridor Tiling** asks whether an instance \( I = (U, V, H, n) \) has a valid corridor tiling of width \( 2^n \). This problem is well known to be \text{EXPSPACE}-complete; see, e.g., [4, 10].

Let \( I = (U, V, H, n) \) be an instance. We construct a game \( G = (\Sigma, R, T) \) such that the following statements are equivalent.

(a) \( I \) has a solution.
(b) \( \text{safe}(G) \setminus \text{safe}_{\text{L2R}}(G) \neq \emptyset \).

The basic idea is to encode tilings as strings of the form

\[
(U\ 0\ c(0)\ U\ c(1)\ \cdots\ U\ c(2^n - 1)\ #).\
\]

Here, each string \( c(i) \) represents the column number \( i \) as a binary string of length \( n \) over \( \{0, 1\} \).

We will construct \( G \) in such a way that the strings in \( \text{safe}(G) \setminus \text{safe}_{\text{L2R}}(G) \) are of the form \( u_1Vguf \), where \( u \) is the encoding of a correct tiling. Furthermore, these strings are then even in \( \text{safe}_{\text{L2R}}(G) \).

The task of \( \text{Juliet} \) in the game is to show that the input string indeed represents a correct tiling, while \( \text{Romeo} \) tries to disprove her.

For the purpose of the game we use additional symbols beyond \( U \cup \{0, 1, \#\} \). First, we use two disjoint copies of \( U \), called \( \bar{U} \) and \( U' \) with symbols \( \hat{a}_1, \ldots, \hat{u}_k, u_1, u_1', \ldots, u_k \) and \( u_1', \ldots, u_k' \), respectively. Furthermore, we use the additional symbols \( 0, 1, u_1, 1. N, \bar{a}, \bar{f}, g, g', h, h', \@.N \).

Before we give the rules of \( G \), we will highlight the main ideas by explaining some typical scenarios that can occur.

1. The first type of play is on input strings \( guf \), where \( u \) is not a correct encoding of a valid tiling for simple reasons. All strings of these forms are not in \( T \) and therefore, \( \text{Juliet} \) cannot win on them. We note that the moves of \( \text{Juliet} \) and \( \text{Romeo} \) cannot transform any string of such an incorrect form into a correct one. Examples of strings on which \( \text{Juliet} \) cannot win are strings

   - in which some bit string \( c \) is not of length \( 2n \),
   - where the first bit string of a row does not represent \( 0 \),
   - the last bit string does not represent \( 2^n - 1 \),
   - a horizontal tile constraint is violated.

2. The second type is on incorrect input strings, where \( \text{Romeo} \) can show the incorrectness in a L2R game. This is the case for strings that have a substring of the form \( c(i)U(c(j)) \) where \( i \neq j \). The play on such a string should proceed as follows. \( \text{Juliet} \) calls all \( U \)-positions successively from left to right (if she does not do that, \( \text{Romeo} \) has a way to win, as will be explained below) and \( \text{Romeo} \) rewrites each symbol \( u_1 \) by \( \hat{u}_1 \). As soon as \( \text{Juliet} \) selects the \( U \)-position before \( c(i) \), \( \text{Romeo} \) rewrites it by \( \@.N \). Afterwards, \( \text{Juliet} \) calls all positions of \( c(i) \) and \( \text{Romeo} \) rewrites each bit by \( \bar{b} \) until he rewrites one bit by \( b. c \).

   The resulting string is in \( T \) if the rewritten bit and the corresponding bit in \( c(j) \) (a) have the same value and it is not the case that in \( c(i) \) the tail after the selected position is a 1-string and in \( c(j) \) it is a 0-string, or (b) have different values and it is not the case that the tail from the selected position in \( c(i) \) on 0 is 1 and the respective tail in \( c(j) \) is 10. If \( i + 1 \neq j \), then \( \text{Romeo} \) can find a critical position such that \( A(T) \) does not accept.

3. The third type is on input strings that are almost valid tilings but violate a vertical constraint in some column \( i \) in successive rows. A game on such strings starts by a \( \text{Call} \) of the final symbol \( f \) which is replaced by the bit string \( c(i) \) of length \( n \), followed by a symbol, \( h \) or \( h' \).

   (This \( \text{Call} \) can also happen in the other scenarios but there it is not relevant.) Afterwards, \( \text{Juliet} \) calls \( g \) if the final symbol is \( h' \) (so that it is rewritten by \( g' \) otherwise she skips it. Then, as in the previous scenario, \( \text{Juliet} \) calls all \( U \)-positions successively from left to right until column \( i \) of the lower violating row. There, \( \text{Romeo} \) rewrites the tile \( u_1 \) by \( u_1' \). Afterwards, \( \text{Juliet} \) selects a bit position in the following bit string which is rewritten by \( b. c \) or \( \bar{b} \). \( \text{Juliet} \) then continues calling all \( U \)-symbols until \( \text{Romeo} \) replaces the tile \( u_1 \) of the \( i \)-column of the next row by \( u_1' \). Again, \( \text{Juliet} \) selects a bit position in the following bit string which is rewritten by \( b. c \) or \( \bar{b} \). The resulting string can only be in \( T \) if the claims of \( \text{Romeo} \) are wrong. If the two positions do not violate \( V \), \( A(T) \) accepts.

   Another possibility is that \( \text{Romeo} \) picks positions in no-successive rows — this can be detected by \( A(T) \), as well. Finally, it could be that \( \text{Romeo} \) choose positions in two different columns. However, this would be detected by \( A(T) \) as well, as in this case one of the selected bit positions would be different from the respective bit position in the string \( c(i) \) that rewrite \( f \).

4. If \( \text{Juliet} \) deviates from the above “protocols” then \( \text{Romeo} \) can use the \( \@. \) symbol to protest. E.g., if \( \text{Juliet} \) does not call a \( U \)-position and jumps to a later one, \( \text{Romeo} \) can rewrite the latter by \( \@. \). Unless he did that without justification (that is, \( \text{Juliet} \) did not leave out a position, in which case the string is in \( T \) and \( \text{Juliet} \) wins directly), he can henceforth always answer with \( \@. \) and \( \text{Juliet} \) cannot win as strings with more than one symbol \( \@. \) are not in \( T \). \( \text{Romeo} \) can protest similarly, if \( \text{Juliet} \) leaves out a bit position in the respective part of Scenario 2.

The rules of \( G \) are as follows.

\[
\begin{align*}
g & \rightarrow \ g' \\
0 \rightarrow 0 & \mid 1. N & \@. \\
1 \rightarrow 1 & \mid 1. N & \@. \\
u_1 & \rightarrow 0 & \mid u_1' & \@.N \\
\vdots \\
u_k & \rightarrow 0 & \mid u_k' & \@.N \\
f & \rightarrow \{0, 1\}^n \mid \{h, h'\} & \@.
\end{align*}
\]

The possibility to rewrite \( f \) by \( \@. \) is crucial, here. \( T \) accepts a string with final symbol \( \@. \) only if it is of the form \( guf \) where \( u \) is a syntactically correct encoding of a tiling (all bit strings have the correct length, only symbols from \( U \cup \{0, 1, \#\} \), each row starts with \( U0^* \) and ends with \( U'1^* \) and there are no violations of \( H \), the first row starts with \( u_1 \), the last ends with \( u_k \) etc.). That is, whenever \( \text{Juliet} \) calls \( f \) in a string \( v \) where \( v \) is not of the form \( guf \) as above, \( \text{Juliet} \) loses the game.

\( T \) is defined such that it only accepts strings as indicated above. We sketch the correctness proof for the reduction, a full proof will be given in the full version of the paper.

\( (a) \Rightarrow (b) \)

Assume there is a valid tiling. We argue that \( \text{Juliet} \) has a winning L2R*-strategy, but no winning L2R-strategy on the word \( vuf \) that encodes this tiling. The winning L2R*-strategy simply first plays \( \text{Call} \) on the \( f \) on the end, after which \( \text{Romeo} \) rewrites it into some binary number \( c(j) \) followed by \( h \) or \( h' \). Then \( \text{Juliet} \) calls the initial symbol \( g \) to
rewrite it with \( g' \), in case ROMEO chose \( h' \). The she proceeds to play Call on every \( U \)-symbol as long as ROMEO doesn’t protest, i.e., as long as he answers with symbols from \( \bar{U} \). If this goes on until play reaches the final \( \# \), JULIET plays Read on the \( n \) digits at the end. Since the input word represents a correct encoding, ROMEO cannot make any correct protests about the column number or the vertical constraints and thus JULIET wins.

Indeed, assume ROMEO tries to make a protest about a vertical constraint by playing a symbol from \( U^V \). The first time he does this, say at column \( i \), JULIET checks whether \( i = c_f \). If this is not the case, she calls a bit position at which \( i \) and \( c_f \) differ, and lets ROMEO mark this position with \( 0 \), \( 1 \) or \( 2 \). She can then just read the rest of the word and win. If, on the other hand, \( i = c_f \), then JULIET continues as if nothing had happened. If ROMEO marks no second tile, then she wins. If ROMEO does play another symbol from \( U^V \), say at column \( j \) in the next row, then JULIET again checks whether \( j = c_f \). If this is not the case, she counter-protests by marking a differing bit in \( j \) and wins as before. If, on the other hand, \( j = c_f \) she wins because the tiles in question do not violate \( V \). If ROMEO did not play his protest in the next row, she wins as well.

Next, we argue that JULIET has no winning L2R-strategy on \( v \). This is simply because against an L2R strategy, ROMEO can choose \( h \), in case the first symbol is \( g' \) or \( h' \), otherwise. As only strings of the form \( guh \) and \( g'wh' \) are in \( T \), JULIET can not win by a L2R strategy.

"(b) \implies (a)"

For the other direction, assume that there is a string \( w \in \text{safe}(G) \setminus \text{safe}_{L2R}(G) \). We first make a couple of observations about what \( w \) must be like.

- If \( w \) is of the form \( vf \), then it must be of the form \( guf \) with \( u \) as described above (after the presentation of the rules of \( G \)). Otherwise, JULIET will not be able to yield a string of the form \( w'h \) or \( w'h' \), as ROMEO can choose \( c \) as soon as JULIET plays Call on \( f \).

- \( w \) cannot be of the form \( vh \) or \( vh' \) as on such strings she has an L2R strategy if and only if she has a strategy at all. Here, it is important that the protesting capabilities of ROMEO basically enforce that JULIET follows a L2R strategy (besides an initial call of \( f \)).

- \( w \) can not be of the form \( vs \) for any \( s \notin \{ f, h, h' \} \) as JULIET can not win on such strings.

Thus, we safely assume that \( w \) is of the form \( guf \) where \( u \) is syntactically correct. If JULIET tries to play Call moves before the Call on \( f \) then ROMEO can win by choosing \( c \) as soon as JULIET plays the (necessary) Call move on \( f \). Thus, we can assume that the strategy of JULIET starts by calling \( f \), thus yielding a string of the form \( guch \). Then the rules of the game ensure that JULIET has a L2R winning strategy from \( (1, guch, \epsilon) \) or from \( (1, guch', \epsilon) \) if on and only if \( u \) encodes a valid tiling. The missing details will be given in the full version of the paper.

In conclusion, the assumption that \( w \in \text{safe}(G) \setminus \text{safe}_{L2R}(G) \) implies that \( w \) contains the encoding of a correct tiling with the correct width, such that neither the horizontal nor the vertical constraints are violated, and with the correct top and bottom tile. Thus (b) \implies (a).

The proof for our lower bound result for games with finite replacement languages is given in the appendix.

**Proposition 19.** \( \text{L2RALL} \) is hard for \( \text{EXPTIME} \), even for games with finite replacement language.

8. CONCLUSION

We investigated a practically relevant restriction of strategies for context-free games and their relation to general strategies. That \( \text{L2RALL} \) is \( \text{EXPTIME} \)-complete in general but \( \text{EXPTIME} \)-complete in the restricted case where the replacement languages in \( G \) are finite, is somewhat surprising, since the word problem of checking whether a given string is safely rewritable in a left-to-right fashion is \( \text{EXPTIME} \)-complete in both cases[8].

The automaton construction for \( \text{safe}_{L2R} \) we give here can be generalised to yield automata for strings which can be safely rewritten using up to \( k \) left steps (with a full L2R pass being played before each left step). This is done by generalising our definition of effects to \( k \)-effects, each of which is a set of sets of \( (k-1) \)-effects representing games on later passes. In this framework, effects as defined in this paper would correspond to \( 1 \)-effects.

It can also be shown that for every game \( G \) there is a game \( G' \) with finite replacement languages whose safely rewritable strings are exactly those of \( G \).

A further open frontier remains in the form of One-Pass \( (1P) \) strategies[2], which restrict L2R strategies by forcing JULIET to make her decisions in a streaming manner, i.e. without knowing the entire input string. While Abiteboul, Milo and Benjelloun[2] have shown a number of interesting properties of such strategies, the general problem of testing whether every safely L2R-rewritable string of a given game can also safely rewritten in a 1P fashion is not even known to be decidable.

9. REFERENCES

[1] Serge Abiteboul, Angela Bonifati, Gregory Cobena, Ioana Manolescu, and Tova Milo. Dynamic XML documents with distribution and replication. In Alon Y. Halevy, Zachary G. Ives, and AnHai Doan, editors, *SIGMOD Conference*, pages 527–538. ACM, 2003.

[2] Serge Abiteboul, Tova Milo, and Omar Benjelloun. Regular rewriting of active XML and unambiguity. In *PODS*, pages 295–303, 2005.

[3] ActiveXML. http://www-rocx.inria.fr/verso/Geno/Projects/axml.

[4] B.S. Chlebus. Domino-tiling games. *Journal of Computer and System Sci.*. 32(3):374–392, 1986.

[5] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games. A Guide to Current Research*. Springer, 2002.

[6] J. Kulbatzki. Active XML und Untersuchungen perfekter Rewriting-Strategien für kontextfreie Spiele auf Strings. Master’s thesis, Technische Universität Dortmund, 2010.

[7] Tova Milo, Serge Abiteboul, Bernd Amann, Omar Benjelloun, and Frederic Dang Ngoc. Exchanging intensive XML data. *ACM Trans. Database Syst.*, 30(1):1–40, 2005.
[8] Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Active context-free games. *Theory Comput. Syst.*, 39(1):237–276, 2006.

[9] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[10] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic and Recursion*, volume 187 of *Notes in Pure and App. Math.*, pages 331–363, Routledge, 1997.

10. APPENDIX

**Proof for Proposition 19.** The proof is by a polynomial time reduction from the L2R word problem, i.e., given a game $G = (\Sigma, R, T)$ and a string $u$, decide whether $u \in \text{safe}_{L2R}(G)$. This problem is shown to be EXPTIME-complete in [8].

To this end, we show how to construct in polynomial time a game $G' = (\Sigma', R', T')$ from $G$ and $u$ such that the following statements are equivalent.

(a) $u \in \text{safe}_{L2R}(G)$.

(b) $\text{safe}(G') \setminus \text{safe}_{L2R}(G') \neq \emptyset$.

The construction of $G'$ will ensure that JULIET can deduce a winning strategy on a string $g_0u_0h_0$ wrt $G'$ with a single Call move in a first phase followed by an L2R phase if and only if she has an L2R winning strategy on $u$ in $G$. In $G'$ we use additional symbols $g_0, g_1, g_2, h_0, h_1, h_2, \#, @$, where

- $g_0, g_1, g_2, h_0, h_1, h_2$ are used to rule out L2R strategies for many strings,
- $@$ can be used by ROMEO to “protest” if JULIET deviates from the intended flow of the game, and
- $#$ is used to force JULIET to follow an L2R strategy on $u$ (or otherwise ROMEO can “protest”).

The alphabet $\Sigma'$ is $\Sigma \cup \{g_0, g_1, g_2, h_0, h_1, h_2, \#, @\}$ and we assume that the latter eight symbols do not belong to $\Sigma$.

For each rule $f \to w_1 | \cdots | w_l$ of $R$, there is a rule $f \to \#w_1 | \cdots | \#w_l | @$ in $R'$. Furthermore, $R'$ contains the following rules.

- $g_0 \to g_1 | @$
- $g_1 \to g_2 | @$
- $h_0 \to h_1 | h_2 | @$

For a string $w \in (\Sigma \cup \{\#\})^*$, we write $\text{cl}(w)$ for the string that results from $w$ by eliminating all occurrences of $\#$.

The target language $T'$ of $G'$ contains

- all strings $g_1w_1$ with $\text{cl}(w) \in T$;
- all strings $g_2w_2$ with $\text{cl}(w) \in T$;
- the string $g_0@$;
- all strings of the form $gw_1$ where $g \in \{g_0, g_1, g_2\}$, $h \in \{h_0, h_1, h_2\}$, and in $w$ there is at least one occurrence of $@$ but no occurrence of $\#$ to the right of an occurrence of $@$;
- all strings $@wh_1$ and $@wh_2$, where $w$ only contains symbols from $\Sigma$.

Clearly, $G'$ can be constructed in polynomial time from $G$ and $u$, in particular a DFA for $T'$ (assuming a DFA for $T$).

It remains to show that (a) and (b) are indeed equivalent.

“(a) $\Rightarrow$ (b)”:

We show that if $u \in \text{safe}_{L2R}(G)$ it follows that $g_0u_0h_0 \in \text{safe}(G') \setminus \text{safe}_{L2R}(G')$.

First $g_0u_0h_0 \in \text{safe}(G')$ as JULIET can choose the last position (carrying $h_0$) first. If ROMEO answers with $@$ she immediately wins as $g_0u@ \in T'$. Otherwise, she enforces $g_1$ as first symbol if ROMEO chose $h_1$ and $g_2$ if ROMEO chose $h_2$. If ROMEO chooses @ for the first symbol, JULIET wins directly as strings of the forms $@wh_1$ and $@wh_2$ with $w \in \Sigma^*$ are in $T'$. Then JULIET can basically follow her L2R winning
strategy on \( u \). It is easy to see that she wins the game in this fashion.

We show next that \( g_0u h_0 \notin \text{safe}_{L2R}(G') \). Clearly, JULIET needs to play a Call on the last position as she cannot enforce a win otherwise (ROMEO simply never protests). However, ROMEO can reply by \( h_1 \) just if the first position of the string is not \( g_1 \), enforcing a win for ROMEO.

Thus, (a) \( \Rightarrow \) (b).

(b) \( \Rightarrow \) (a):
Assume that there is a word \( v \in \text{safe}(G') \setminus \text{safe}_{L2R}(G') \).

We start with some observations on what \( v \) can look like.

1. The word \( v \) must begin with some \( g_i \in \{g_0, g_1, g_2\} \). Indeed, no letter not in \( \{g_0, g_1, g_2\} \) can ever be rewritten into a letter in \( \{g_0, g_1, g_2\} \). Indeed, strings that are accepted that do not begin with such a letter are strings on the form \( \#wh_1 \) or \( \#wh_2 \). If JULIET wins on a string that begins with \( \# \), it must be by never playing Call, since otherwise ROMEO could protest with a second \( @ \) and win. Thus, if JULIET wins, she wins with an L2R-strategy.

2. The word \( v \) must end with some \( h_j \in \{h_0, h_1, h_2\} \). There is only a single accepted string that is accepted that does not end with such a letter and no other letter can be rewritten into them. If the string \( v \) ends with \( @ \), it is either \( g_0u0 \), in which case JULIET wins with an L2R-strategy by just reading it. If the string \( v \) begins with \( \# \), JULIET can answer with a second \( @ \) and win, in which case she cannot win, since any Call move can be answered by ROMEO with a second \( @ \) symbol.

3. If JULIET has a winning strategy on \( g_iwh_j \), then the strategy must play left to right on \( gw \). If JULIET plays Call on a symbol \( f \) in \( w \), ROMEO can answer with a string \( \#u_j \in R_f \), introducing a \( \# \) symbol into the word. If JULIET ever plays a Call on a position to the left of this symbol, ROMEO can protest with an \( @ \) symbol, creating a word with an \( @ \) to the left of a \( \# \). After this, JULIET cannot win.

4. That JULIET can win on \( v = g_iwh_j \), but not with an L2R-strategy means that she needs to call on the last position before completing play on \( g_iw \). This implies \( h_j = h_0 \).

5. On strings of the form \( g_1wh_0 \) or \( g_2wh_0 \), JULIET has no winning strategy. Indeed, since no accepted string ends with \( h_0 \), JULIET will sooner or later have to play Call on the last position. When she does this, ROMEO can answer with \( @ \). The only string ending with \( @ \) that is accepted is \( g_0u0 \), but \( g_1 \) or \( g_2 \) can never be rewritten into \( g_0 \) and thus JULIET loses.

From (1)–(5) above, we can conclude that if \( v \in \text{safe}(G') \setminus \text{safe}_{L2R}(G') \), then \( v \) has the form \( g_0wh \). When JULIET starts play on a word \( g_0wh \) by calling on the last position, ROMEO can answer with \( @ \). The only accepted string that ends with \( @ \) is \( g_0u0 \). This means that the string \( v \) must be \( g_0uwh \). ROMEO can, however, also answer the call on \( h_0 \) with \( h_1 \) or \( h_2 \). In this case, JULIET must play an L2R-strategy that transforms \( g_0u \) into some \( g_iw' \) with \( \text{cl}(w') \in T \). This same strategy, restricted to \( u \) and ignoring the \( \# \)-symbols, is a winning L2R-strategy on \( u \). \( \square \)