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

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 is EXPTIME-complete.

Categories and Subject Descriptors

F.2.m [**Analysis of Algorithms and Problem Complexity**]: Miscellaneous; F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based Services*

General Terms

Algorithms, Theory

1. INTRODUCTION

In this paper, we study *Active Context-Free Games*, played by two players on finite strings. The motivation for these games comes from the study of *Active XML* documents [1, 3, 8]. In such documents, only some of the data is explicitly given, the rest of the data can be 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.

However, this approach raises some challenges when documents should be valid with respect to some schemas. The hosts not only need to ensure that their own documents conform to the schema, but also that this is the case for all possible documents resulting from web service calls.

This scenario was studied by Milo et al. [8], who formulated a polynomial time algorithm for a restricted setting, in which this *schema rewriting problem* on AXML trees can be solved by recursively solving a similar rewriting problem on strings.

In order to model and study this scenario, active context-free games, or context-free games for short, were introduced by Muscholl et al. [9]. Context-free games are two-player games played on strings, where the first player, Juliet, represents the host. By calling on letters she tries to rewrite the string into one that conforms to a schema, 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. Otherwise, Romeo wins.

In this particular paper we focus on so-called *left-to-right* (L2R) strategies. If Juliet follows such a strategy, she is not allowed to call a position that is to the left of a previously called position. L2R-strategies have been considered before, e.g. in [9, 2]. They are more feasible than unrestricted 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 EXPTIME whether she has a winning L2R-strategy [9]. The aforementioned efficient algorithm by Milo et al. [8] also requires a restriction to L2R rewritings.

It is thus useful to determine during the design phase of a system whether for Juliet, L2R-strategies are *universal.*

*We acknowledge the financial support of the Swedish Research Council through its grant 621-2011-6080.
†We acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599.

EDBT/ICDT '13 March 18 - 22 2013, Genoa, Italy.

[1] This is the L2RAll problem, studied here: given a game, does Juliet have a winning L2R-strategy for every string for which she has a winning strategy at all? The L2RAll problem was first considered in [2], where it was claimed to be undecidable.

We take the following approach to the problem. First we show that if L2R-strategies are not universal, there is a string for which Juliet has a winning strategy with one *left step* but no L2R winning strategy. Then we show how to construct automata for all strings with a winning L2R strategy and for all strings with a winning "1-left-step" 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. In a nutshell, the effect of a substring summarizes how the string that is obtained from it during the game can affect the automaton for the schema.

We show that the L2RAll problem can be solved in exponential space and that this is optimal. If the set of possible replacement strings from which Romeo can choose is finite and explicitly given, for every letter, the complexity drops to exponential time. Thus, we prove the following result.

Theorem 1 (Main Theorem).

(a) L2RAll *is* EXPSPACE-*complete.*

(b) If all replacement languages are finite and 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 "1-left-step" 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. Details for some omitted proofs in Section 7 can be found in the full version of this paper [4].

Related work. We already discussed the most important related papers [2, 9, 1, 8] above. That automata for the set of words with winning L2R strategies can be constructed in exponential time was already shown in [9]. 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 Kulbatzki, which was written under the supervision of the third author [7].

We thank the anonymous reviewers of ICDT 2013 for helpful suggestions and Ahmet Kara for proof reading.

2. PRELIMINARIES

In this section we define the fundamental notions.

2.1 Context-free games

A *context-free game* $G = (\Sigma, R, T)$ consists of a finite alphabet Σ, 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 $R_f =_{\text{def}} \{u \mid (f, u) \in R\}$ is regular. By Γ we denote the set $\Gamma =_{\text{def}} \{f \mid f \in \Sigma, R_f \neq \emptyset\}$ and we call the symbols from Γ *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 $w \in \Sigma^*$.

In its original form, as introduced in [9], the game proceeds in rounds, in each of which Juliet selects a position of the current string and Romeo chooses a rewriting rule to replace the current symbol f at that position by a string from R_f. For the purposes of this paper a different, but equivalent, definition of (the rules of) context-free games is more suitable,

In our definition, a 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 substitution for the symbol that is allowed by the rule set.

More formally, a *configuration* is a tuple $C = (p, u, v) \in \{1, 2\} \times \Sigma^* \times \Sigma^*$ where p is the player to move (1 for Juliet and 2 for Romeo), uv is the *current word*, and the first symbol of v is *the current position*. A *winning configuration* for Juliet is a configuration $C = (p, v, \varepsilon)$ with $v \in T$.

In each configuration $(1, u, v)$ with $v \neq \epsilon$, Juliet can either choose a Read move or, if the first symbol f of v is from Γ a Call move. If she selects Read, the play moves one step to the right. If she selects Call, then Romeo selects a string from the set R_f. In a configuration $(1, u, \epsilon)$ 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 $sv' = v$ 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) $p' = p = 1$, $u \notin T$, $v = \varepsilon$, $v' = u$, $u' = \varepsilon$, (Juliet plays LS).

If Juliet plays Stop in a configuration $C = (p, u, \epsilon)$ we write $C \to \top$ if $u \in T$ and $C \to \bot$ if $u \notin T$ 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 $\Pi \equiv p$ if player p wins Π.

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_f for R_f, for every $f \in \Gamma$.[2] In the sequel, let $A(T) = (Q, \Sigma, \delta, F, q_0)$ with

[1]The high complexity of lower bounds we prove for the L2RAll problem may seem to make this task forbiddingly difficult; however, since L2RAll is effectively a static analysis problem, the added complexity may be affordable as a pre-processing step.

[2]We note that whether R_f is represented by DFAs or NFAs

state set Q, transition function $\delta : Q \times \Sigma \to Q$, accepting states $F \subseteq Q$ and initial state $q_0 \in Q$.

We note that our definition of active context-free games is indeed equivalent to the one in [9]. JULIET can select an arbitrary position by playing a sequence of Read moves possibly followed by a LS move, another sequence of Read moves and, eventually, a Call move at the desired position.

2.2 Game trees

The *game tree* $Tree_{G,u}$ 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 $Tree_{G,u}$ is labeled by the initial configuration $C_0(u)$. A node labeled C has one child for every configuration C' such that $C \to C'$. This means that the only leaves of $Tree_{G,u}$ are nodes labeled by final configurations of finite plays. In general, nodes labeled by configurations $C = (1, u, v)$ have one or two children: if $v = sv'$ for some $s \in \Sigma$, there is always one child corresponding to a Read move, and a second one corresponding to a Call move exists iff $s \in \Gamma$. If $v = \epsilon$, the two children correspond to a LS and Stop move respectively. 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 σ is *memoryless* if, for every prefix $C_0, C_1, \ldots, C_k$ of a play, the selected move $\sigma(C_0, C_1, \ldots, C_k)$ only depends on C_k.

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 σ, τ we write $C \xrightarrow{\sigma,\tau} C'$ if C' is the unique successor configuration of C determined by the strategies σ and τ. Given an initial word u and strategies σ, τ the play[3] $\Pi(\sigma, \tau, u) =_{\text{def}} C_0(u) \xrightarrow{\sigma,\tau} C_1 \xrightarrow{\sigma,\tau} C_2 \cdots$ is uniquely determined.

A strategy σ for JULIET is *finite* on string u if the play $\Pi(\sigma, \tau, u)$ is finite for every strategy τ of ROMEO. It is a *winning strategy* for u if $\Pi(\sigma, \tau, u) \equiv 1$, for every τ. A strategy τ for ROMEO is a *winning strategy* for u if $\Pi(\sigma, \tau, u) \equiv 2$, for every strategy σ 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 $\mathrm{STRAT}(G)$, and the set of all L2R-strategies by $\mathrm{STRAT}_{L2R}(G)$. The set of all strategies for ROMEO is denoted by $\mathrm{STRAT}_{\text{ROMEO}}(G)$.

By definition, $\mathrm{STRAT}_{L2R}(G) \subseteq \mathrm{STRAT}(G)$.

By $\mathrm{safe}(G)$ we denote the set of all words for which JULIET has a winning strategy and by $\mathrm{safe}_{L2R}(G)$ 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 $\mathrm{safe}_{L2R}(G) = \mathrm{safe}(G)$. By L2RALL we denote the set of all games G, for which $\mathrm{safe}_{L2R}(G) = \mathrm{safe}(G)$.

As context-free games are reachability games we can make use of the following classical result; see, e.g., [6].

THEOREM 2. *Let G be context-free game, and u a string. Then the following 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 σ for JULIET map configurations C to moves $\sigma(C) \in \{\text{Call}, \text{Read}\}$ and strategies τ 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 σ on the subgame on v. This means that she follows σ until a configuration (uv', w) is reached.

The *strategy tree* for a strategy σ of JULIET is the restriction $Tree_{G,u}(\sigma)$ of $Tree_{G,u}$ to σ. 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 σ. Strategy trees for ROMEO are defined symmetrically. If we fix strategy σ for JULIET and τ for ROMEO, we get $Tree_{G,u}(\sigma, \tau)$, which only has one branch, labeled by the play $\Pi(\sigma, \tau, u)$. Notice that if a strategy σ of JULIET is winning, then $Tree_{G,u}(\sigma)$ has no infinite branches.

If $\Pi(\sigma, \tau, w)$ is finite, then $\mathrm{word}^G(w, \sigma, \tau)$ is the word in the final configuration of the play on w following σ and τ. (and otherwise $\mathrm{word}^G(w, \sigma, \tau) = \bot$). We let

$$\mathrm{words}^G(w, \sigma) =_{\text{def}} \{\mathrm{word}^G(w, \sigma, \tau) | \tau \in \mathrm{STRAT}_{\text{ROMEO}}(G)\}.$$

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 sets *normal* if it does not contain two sets X and Y with $X \subset Y$. A finite set S of finite sets can be *normalized* by applying the following NORM operator.

$$\mathrm{NORM}(S) = \{Y \in S \mid \text{there is no } X \in S, \text{ such that } X \subset Y\}.$$

LEMMA 3. *Let S_1, S_2 be normal sets of sets. If for every $s_1 \in S_1$ there is $s_2 \in S_2$ such that $s_2 \subseteq s_1$ and vice versa then $S_1 = S_2$.*

PROOF. We show that every set $s_1 \in S_1$ is also in S_2. The lemma then follows by symmetry.

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

We sometimes just write $N(S)$ for $\mathrm{NORM}(S)$.

does not influence the complexity. However, we conjecture that allowing NFAs for T may lead to an unavoidable exponential blowup of the complexity. We chose DFAs for our setting as we are interested in cases with reasonable efficiency.

[3]As the underlying game G will always be clear from the context, our notation does not mention G explicitly.

3. FROM GENERAL TO L2R$^+$-STRATEGIES

Definition 1. A strategy σ of JULIET is an *extended L2R-strategy* (L2R$^+$) if for every string u and every strategy τ 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 $safe(G) = safe_{L2R}(G)$ if and only if $safe_{L2R^+}(G) = safe_{L2R}(G)$.*

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

Assume that $\text{safe}(G) \neq \text{safe}_{L2R}(G)$ and let w be a string in $\text{safe}(G) \setminus \text{safe}_{L2R}(G)$. Let σ be a winning strategy for JULIET on w, i.e., starting from the configuration $(1, w, \varepsilon)$. Consider the strategy tree $Tree_{G,w}(\sigma)$. 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 σ 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 ∞.

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 $\leq$ the lexicographic order on markings.

Without loss of generality, we may assume that σ is *optimally efficient* in the following sense. We assume that for 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 σ' on w, such that the strategy tree for σ' 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 for every configuration (p, u, v) by nested induction on the minimal value of $(\text{LS}(n), \text{Calls}(n))$ that nodes n representing (p, u, v) can assume in winning strategies for (p, u, v).

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

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

If $Tree_{G,w}(\sigma)$ has nodes with LS-value ∞, 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, $Tree_{G,w}(\sigma)$ would have infinite branches, contradicting the fact that σ is a winning strategy. There must be arbitrarily large LS-values among the children of n' as otherwise $\text{LS}(n') \neq \infty$. In particular, n' must have a JULIET-grandchild n with $\text{LS}(n) > 1$ and therefore a marking differing from $(1, 0)$.

If $Tree_{G,w}(\sigma)$ has no nodes with LS-value ∞, then for the root r of $Tree_{G,w}(\sigma)$ it holds $\text{LS}(r) \neq \infty$, and thus $\text{LS}(r) \geq 1$ (as otherwise $w \in \text{safe}_{L2R}(G)$) and $\text{Calls}(r) > 0$ (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 $(1, 0)$.

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, ∞).

(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 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 $n_1, n_2, \dots$ of nodes by chosing, 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 σ, the sequence can not be infinite. Furthermore, each leaf has marking $(1, 0)$. Therefore, the sequence must contain a JULIET-node n_ℓ with marking $(1, 1)$. Let $(1, x, y)$ be the configuration of n_ℓ. We claim that $xy \in \text{safe}_{\text{L2R}^+}(G) \setminus \text{safe}_{L2R}(G)$.

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

On the other hand, as the marking of n_ℓ is $(1, 1)$, starting from $(1, xy, \varepsilon)$, JULIET can play Read on x and can win with one Call before the one and only LS move, therefore $xy \in \text{safe}_{\text{L2R}^+}(G)$. Thus, $\text{safe}_{\text{L2R}^+}(G) \neq \text{safe}_{L2R}(G)$, completing the proof. $\Box$

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. (Recall that $A(T) = (Q, \Sigma, \delta, F, q_0)$ is the DFA accepting the regular language T.) In this section, we only consider L2R strategies for JULIET, that is, JULIET never makes an LS-move.

Suppose we have the game configuration $(1, v, uw)$. As play goes on, it will eventually reach some configuration $(1, vu', w)$, 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 σ 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 σ. The *relative effect* $e(\sigma, u, q)$ of u for a strategy σ 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 $e(\sigma, u, q)$, for all states q and strategies σ.

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 σ a L2R-strategy of JULIET. The *relative* effect $e(\sigma, u, q)$ is the set $\{\delta^*(q, w) | w \in \text{words}(u, \sigma)\}$ or $\bot$ if $\bot \in \text{words}^G(w, \sigma)$.

The effect $E[u]$ of u maps every state q to the normalized set of relative effects $e(\sigma, u, q)$ of u 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 τ of ROMEO and a string $w \in \Sigma^*$ such that $w = \text{word}(u, \sigma, \tau)$ and $\delta^*(q, w) = p$, or $\perp$ if $\perp \in \text{words}^G(w, \sigma)$. 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, $u \in safe_{L2R}(G)$ if and only if there is a relative effect $e \in E[u](q_0)$ 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[4] abstract effects by $\mathcal{E}$.

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 $E[uv]$. 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 MIX, which is defined on sets of sets of sets. Let $\mathcal{D} = \{D_1, \dots, D_n\}$ be a set of sets of sets. Then $\text{MIX}(\mathcal{D})$ is the set

$$\text{NORM}(\{d_1 \cup \dots \cup d_n \mid d_1 \in D_1 \wedge \dots \wedge d_n \in D_n\}).$$

In other words, the MIX operation computes every way of taking the union of one element from each of $D_1, \dots, D_n$.

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 JULIET can choose from $E_1(q)$, 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 JULIET can enforce by some strategy.

LEMMA 6. *Let u, v be strings. Then $E[u] \circ E[v] = E[uv]$.*

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 $e' \in E[uv](q)$ 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 $e' \in E[uv](q)$ such that $e' \subseteq e$.

By the definition of $\circ$ there exists a relative effect $X = \{q_1, \dots, q_k\} \in E[u](q)$ and relative effects $e_2^i \in E[v](q_i)$, for each i, such that $e = \bigcup_{i=1}^{k} e_2^i$.

We denote the strategy of JULIET on u yielding X by σ_1 and the strategies on v yielding $e_2^1, \dots, e_2^k$ (from $q_1, \dots, q_k$, respectively) by $\sigma_2^1, \dots, \sigma_2^k$, respectively.

We define a strategy σ on uv for JULIET as follows. In the first phase, on u, JULIET plays according to σ_1. If y is the word by which u is rewritten in the game on u, then $\delta^*(q, y) = q_i$, for some $i \in \{1, \dots, k\}$. In the second phase, on v, JULIET plays according to strategy σ_2^i.

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

By definition of $X = e(\sigma_1, u, q)$ and the definition of σ it follows that $\delta^*(q, w_1) \in X$ and thus $\delta^*(q, w_1) = q_i$, for some i. Therefore, JULIET plays according to σ_2^i in the game on v and consequently $\delta^*(q_i, w_2) \in e(\sigma_2^i, v, q_i) = e_2^i$. Altogether,

$$p = \delta^*(q, w) = \delta^*(\delta^*(q, w_1), w_2) = \delta^*(q_i, w_2) \in e_2^i \subseteq e,$$

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 σ be a strategy of JULIET such that $e' = e(\sigma, uv, q)$. Let σ' denote the strategy of JULIET on u that is induced by σ and let $X = e(\sigma', u, q) = \{q_1, \dots, q_k\}$.

Let $w_1, \dots, w_k$ be words from $\text{words}(u, \sigma')$ such that, for every i, $\delta^*(q, w_i) = q_i$ and let $\tau_1, \dots, \tau_k$ be corresponding strategies of ROMEO on u. For every i, let σ_i denote the strategy of JULIET on v induced by σ 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 τ' of ROMEO on v such that for the word $z = \text{word}(v, \sigma_i, \tau')$ it holds $\delta * (q_i, z) = p$. Combining τ_i (on u) and τ' (on v) yields a strategy τ for ROMEO such that $\text{word}(uv, \sigma, \tau) = w_i z$. Furthermore, $\delta^*(q, w_i z) = \delta^*(\delta^*(q, w_i), z) = p$ and thus $p \in e' = e(\sigma, uv, q)$. □

5. AUTOMATA FOR L2R STRATEGIES

In this section, we define, for each context-free game G, the NFAs $A_{\text{L2R}}(G)$ and $\hat{A}_{\text{L2R}}(G)$, recognizing $\text{safe}_{L2R}(G)$ and $\Sigma^* \setminus \text{safe}_{L2R}(G)$, respectively. One of them, $A_{\text{L2R}}(G)$, is based on the computation of relative effects of the form $e(\sigma, u, q_0)$ for strategies σ of JULIET, whereas the other is based on the computation of dual effects (to be defined below) of the form $\hat{e}(\tau, u, q_0)$ for strategies τ of ROMEO. Furthermore, we show correctness of these automata and how to compute them in exponential time from G.

[4]As always, we assume that the target automaton $A(T)$ is fixed.

5.1 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 $A_{\mathrm{L2R}}(G) = (Q_{L2R}, \Sigma, \delta_{L2R}, \{q_0\}, F_{L2R})$ is defined as follows:

- $Q_{L2R} = \mathcal{P}(Q)$;
- $\delta_{\mathrm{L2R}}(X, s) = \textsc{Mix}(\{E[s](q) \mid q \in X\})$, 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 $L(A_{L2R}(G)) = safe_{L2R}(G)$.*

Proof. We show by induction on $|u|$ that for every string $u \in \Sigma^*$ we have $\textsc{Norm}(\delta^*_{\mathrm{L2R}}(\{q_0\}, u)) = E[u](q_0)$:
For $u = \epsilon$, $\textsc{Norm}(\delta^*_{\mathrm{L2R}}(\{q_0\}, \epsilon)) = \{\{q_0\}\} = E[\epsilon](q_0)$.
For $u = vs$ we get

$$\begin{aligned}
N(\delta^*_{\mathrm{L2R}}(\{q_0\}, vs)) &= N(\bigcup_{X \in \delta^*_{\mathrm{L2R}}(\{q_0\}, v)} \delta_{\mathrm{L2R}}(X, s)) \\
&= N(\bigcup_{X \in E[v](q_0)} \delta_{\mathrm{L2R}}(X, s)) \\
&= N(\bigcup_{X \in E[v](q_0)} \textsc{Mix}(\{E[s](q) \mid q \in X\})) \\
&= (E[v] \circ E[s])(q_0) \\
&= E[u](q_0).
\end{aligned}$$

We can conclude as follows that Juliet has a L2R winning strategy on u if and only if $A_{\mathrm{L2R}}(G)$ accepts u.

$$\begin{aligned}
u \in \mathrm{safe}_{L2R}(G) &\Leftrightarrow \exists e \in E[u](q_0) : e \subseteq F \\
&\Leftrightarrow E[u](q_0) \cap \mathcal{P}(F) \neq \emptyset \\
&\Leftrightarrow N(\delta^*_{\mathrm{L2R}}(\{q_0\}, u)) \cap \mathcal{P}(F) \neq \emptyset \\
&\Leftrightarrow \delta^*_{\mathrm{L2R}}(\{q_0\}, u) \cap \mathcal{P}(F) \neq \emptyset
\end{aligned}$$

□

5.2 Computing L2R automata

Proposition 8. *There is an algorithm that computes in exponential time the NFA $A_{L2R}(G)$ for each 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_f are represented by DFAs, NFAs or regular expressions.*

Proof. The algorithm first computes in exponential time the effects $E[s]$, for every symbol $s \in \Sigma$, using Algorithm 1 below. By Proposition 9 below, this is possible in exponential time. The construction of $A_{\mathrm{L2R}}(G)$ is then straightforward. It should be noted that $\textsc{Mix}(\{E[s](q) \mid q \in X\})$ 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 each 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 each 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 set X in $P(s, q)$, there is an L2R-strategy σ of Juliet such that $X = e(\sigma, s, q)$.

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 $P[a_1] \circ \cdots \circ P[a_\ell]$, where $a_1 \cdots a_\ell = w$ and the operation $\circ$ is defined just as for effects.

Algorithm 1 Compute the effects of symbols from Σ

```
    for all s ∈ Σ, q ∈ Q do
2:    P[s](q) ← {{δ_s(q)}}
    while some set P[s](q) has changed in the previous iteration do
4:    for all f ∈ Γ, q ∈ Q do
        P[f](q) ← P[f](q) ∪ Mix({P[w](q) | w ∈ R_f})
6:      P[f](q) ← 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 $s \in \Sigma$. If T is represented by a DFA $A(T) = (Q, \Sigma, \delta, q_0, F)$ and the sets R_f 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 to run in exponential time. We assume without loss of generality that all sets R_f 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. 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 5,

$$P[f](q) \leftarrow P[f](q) \cup \textsc{Mix}(\{P[w](q) \mid w \in R_f\}).$$

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 $U \in \textsc{Mix}(\{P[w](q) \mid w \in R_f\})$, and to add it to $P[f](q)$ if this is the case. We do this in a bottom up fashion, starting with the singelton 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 $w \in R_f$, there is a set $W \in P[w](q)$ with $W \subseteq U$. If this is not the case, then $U \notin \textsc{Mix}(\{P[w](q) \mid w \in R_f\})$. If, on the other hand, this is the case, then there is a subset U' of U that belongs to $\textsc{Mix}(\{P[w](q) \mid w \in R_f\})$. 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 $A_{\mathrm{L2R}}(G)$ in Definition 3 below, but with the following modifications. First, the initial state is $\{q\}$. Second, the set of accepting states is $\mathcal{P}(U)$. Third, 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[5] of Proposition 7. Whether, for each $w \in R_f$, there

[5] We point out that the current proof is similar to the proof of

is a set $W \in P[w](q)$ with $W \subseteq U$, can then be tested by checking whether $R_f \subseteq L(B)$. 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_f) 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 $P^j[s]$ denote the value of $P[s]$ after the j-th iteration of the WHILE loop. For a strategy σ 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 $\Pi(\sigma, \tau, u)$. 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 $q \in Q$, $P^j[s](q)$ contains exactly the relative effects $e(\sigma, s, q)$, for all strategies σ of JULIET with $\text{Depth}(\sigma, s) \leq j$.

We prove Claim 1 by induction on j.

For $j = 0$ this holds true as each $P^0[s](q) = \{\{\delta(q, s)\}\}$ 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 $\{P^{j-1}[w](q) \mid w \in R_f\}$ is a finite set, there are $\ell \in \mathbb{N}$, and strings $w_1, \ldots, w_\ell \in R_f$ such that $\{P^{j-1}[w](q) \mid w \in R_f\} = \{P^{j-1}[w_i](q) \mid i \in \{1, \ldots, \ell\}\}$. For each w we denote by $i(w)$ the number in $\{1, \ldots, \ell\}$ with $P^{j-1}[w](q) = P^{j-1}[w_{i(w)}](q)$. For reference, we denote the set $\{w_1, \ldots, w_\ell\}$ by $S(j, f, q)$.

Let now e be a relative partial effect in $P^j[f](q)$. If $e \in P^{j-1}[f](q)$, then $e = e(\sigma, f, q)$ for some strategy σ of JULIET with $\text{Depth}(\sigma, f) \leq j - 1$, by induction. Thus, we assume $e \in P^j[f](q) \setminus P^{j-1}[f](q)$ and thus e "arrived" in $P[f](q)$ in the j-th iteration of the WHILE loop. Therefore, $e \in \text{MIX}(\{P^{j-1}[w](q) \mid w \in R_f\})$. Furthermore, there are relative partial effects $e_1, \ldots, e_\ell$ such that

- $e_i \in P^{j-1}[w_i](q)$, 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 σ_i of JULIET on w_i of depth $\leq j - 1$ such that $e_i = e(\sigma_i, w_i, q)$.

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

Conversely, let σ 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 σ, is a Call on s. For each $i \in \{1, \ldots, \ell\}$ let σ_i be the strategy of JULIET that is induced by σ on w_i and let $e_i = e(\sigma_i, w_i, q)$. Now for each possible reply $w \in R_f$ of ROMEO, let σ_w be the strategy that yields $e(\sigma_{i(w)}, w_{i(w)}, q)$ and let $e_w = e(\sigma_w, w)$. Thus,

$$e = \bigcup_{w \in R_f} e_w = e_1 \cup \cdots \cup e_\ell.$$

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

This concludes the proof of the Claim 1.

So far we have not ruled out that there might be a strategy σ of JULIET with unbounded Call depth such that $e(\sigma, f, q) \notin P[s](q)$, 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 $w \in R_f$ let $v(f, w)$ be a string of minimal length such that $v(f, w) \in R_f$ 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[6] 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 σ of JULIET, the G'-strategy σ' that is is induced by σ 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 σ' of JULIET there is a finite G-strategy σ with $e^G(\sigma, u, q) \subseteq e^{G'}(\sigma', u, q)$.

We first observe that $\text{Depth}^{G'}(\sigma', u) < \omega$ Otherwise, the strategy tree induced by σ' 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 σ').

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 σ' 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 σ'_w is the strategy of JULIET on w induced by σ'. Thus, by induction, there is a G-strategy σ_w with $e^G(\sigma_w, w, p) \subseteq e^{G'}(\sigma_w, w, p)$. Combing σ_w with an initial Read on s yields the desired strategy σ.

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

Proposition 7 but not on the proof of Proposition 8. Rather the proof of Proposition 8 will be based on the current proof.

[6]The sets $S(j, q, f)$ will become important after the proof of Claim 2.

We define σ 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 $\sigma_{z,1}$ with $e^G(\sigma_{z.1}, v(z,f)w, q) \subseteq e^{G'}(\sigma', v(z,f)w, q) = e'$. As $E^G[v(f,w)] = E^G[v]$, there is a strategy σ_z for JULIET with $e^G(\sigma_{z.1}, v(z,f), q) = e^G(\sigma_z, z, q)$.

We define strategy σ for the case that ROMEO replies by z on fw as follows. It plays according to σ_z on z and then follows the strategy induced by $\sigma_{z.1}$ on w. Altogether, we can conclude $e^G(\sigma, fw, q) \subseteq e'$, as required.

This completes the proofs of Claim 3 and 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 σ' of JULIET contributing to $E^{G'}[s]$ are of bounded Call depth. Otherwise, the strategy tree induced by σ' 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 $E^{G'}[s]$ 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 $\Sigma^* \setminus \text{safe}_{L2R}(G)$. As complementation of $A_{\text{L2R}}(G)$ might yield an automaton of doubly exponential size (in $|G|$), we follow a different approach by constructing an NFA for $\Sigma^* \setminus \text{safe}_{L2R}(G)$ that works analogous as $A_{\text{L2R}}(G)$ but is based on strategies of ROMEO. To this end, we define dual effects and the dual automaton $\hat{A}_{\text{L2R}}(G)$ next.

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

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 σ of JULIET and a string $w \in \Sigma^*$ such that $w = \text{word}(u, \sigma, \tau)$ and $\delta^*(q, w) = p$. We note that non-terminating plays do not contribute to dual effects, as for every strategy τ there is a strategy σ 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 $\mathcal{D} = \{D_1, \ldots, D_n\}$ be a set of sets. Then

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

In other words, SMIX contains all sets that can be formed by selecting one element from each of the elements of $\mathcal{D}$. 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 $\hat{E}[u](q) = \text{SMIX}(E[u](q))$.*

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 $e \in \text{SMIX}(E[u](q))$ such that $e \subseteq \hat{e}$, and vice versa.

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

For the other direction, consider $e \in \text{SMIX}(E[u](q))$. By definition of $E[u](q)$ and SMIX, for every finite strategy σ of JULIET there is a strategy τ of ROMEO such that $\delta^*(q,w) \in e$, where $w = \text{word}(u,\sigma,\tau)$. Let $t = \mathit{Tree}_{G,u}$ be the (full) game tree for u. Let L_e be the set of leaves of t that are labeled by configurations $(1, w, \varepsilon)$ with $\delta^*(q,w) \in e$. Let S_e be the set of nodes n of t such that for every strategy σ of JULIET, the subtree of the strategy tree $\mathit{Tree}_{G,u}(\sigma)$ 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 σ 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 τ for ROMEO that from a node in S_e always selects a child node in S_e. In the strategy tree $\mathit{Tree}_{G,u}(\tau)$, every node belongs to S_e. This immediately implies that $\hat{e}(\tau, u, q) \subseteq e$. □

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 $\hat{A}_{\text{L2R}}(G)$ for $\Sigma^* \setminus \text{safe}_{L2R}(G)$.

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. Then the NFA $\hat{A}_{\text{L2R}}(G) = (\hat{Q}_{L2R}, \Sigma, \hat{\delta}_{L2R}, \{q_0\}, \hat{F}_{L2R})$ is defined as follows:

- $\hat{Q}_{L2R} = \mathcal{P}(Q)$;
- $\hat{\delta}_{\text{L2R}}(X, s) = \text{MIX}(\{\hat{E}[s](q) \mid q \in X\})$, 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 the target language. Then $L(\hat{A}_{L2R}(G)) = \Sigma^* \setminus \mathit{safe}_{L2R}(G)$.*

PROOF. As for $A_{\text{L2R}}(G)$, the first step is to show that $\text{NORM}(\hat{\delta}^*_{\text{L2R}}(q_0, u)) = \hat{E}[u](q_0)$, by induction on $|u|$.

For $u = \epsilon$ we have

$$\text{NORM}(\hat{\delta}^*_{\text{L2R}}(\{q_0\}, \epsilon)) = \{\{q_0\}\} = \hat{E}[\epsilon](q_0).$$

For $u = vs$ we get

$$\begin{aligned}
N(\hat{\delta}^*_{\text{L2R}}(\{q_0\}, vs)) &= N(\bigcup_{X \in \hat{\delta}^*_{\text{L2R}}(\{q_0\}, v)} \hat{\delta}_{\text{L2R}}(X, s)) \\
&= N(\bigcup_{X \in \hat{E}[v](q_0)} \hat{\delta}_{\text{L2R}}(X, s)) \\
&= N(\bigcup_{X \in \hat{E}[v](q_0)} \textsc{Mix}(\{\hat{E}[s](q) \mid q \in X\})) \\
&= (\hat{E}[v] \circ \hat{E}[s])(q_0) \\
&= \hat{E}[v](q_0).
\end{aligned}$$

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

$$\begin{aligned}
u \in \Sigma^* \setminus \text{safe}_{L2R}(G) &\Leftrightarrow \exists e \in \hat{E}[u](q_0) : e \cap F = \emptyset \\
&\Leftrightarrow \hat{E}[u](q_0) \cap \mathcal{P}(Q \setminus F) \neq \emptyset \\
&\Leftrightarrow N(\hat{\delta}^*_{\text{L2R}}(\{q_0\}, u)) \cap \mathcal{P}(Q \setminus F) \neq \emptyset \\
&\Leftrightarrow \hat{\delta}^*_{\text{L2R}}(\{q_0\}, u) \cap \mathcal{P}(Q \setminus F) \neq \emptyset
\end{aligned}$$

□

PROPOSITION 13. *There is an algorithm that computes in exponential time the NFA $\hat{A}_{L2R}(G)$ for each 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_f are represented by DFAs, NFAs or regular expressions.*

PROOF. Similar to the algorithm computing $A_{\text{L2R}}(G)$, 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 $\hat{E}[s](q) = \textsc{SMix}(E[s](q))$, for every $q \in Q$. Each computation of a set $\textsc{SMix}(E[s](q))$ 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 $\textsc{Mix}(\{\hat{E}[s](q) \mid q \in X\})$ can be computed in exponential time as well in a straightforward fashion. □

6. UPPER BOUNDS

In this section, we prove the upper bounds results of our Main Theorem 1. The problem L2RALL is decidable and can actually be decided in exponential space. If all rule sets 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 auxilliary results that allow us to consider only finite subsets of each 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 JULIET has a winning strategy on w.

For a state q and a set S of states let $A^{q,S}_{\text{L2R}}$ denote the automaton that is obtained from $A_{\text{L2R}}(G)$ 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 $A^{q,F(w)}_{L2R}$ accepts exactly the strings v such that there is a winning strategy for* 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 $vw \in safe_{L2R}(G_q)$ then there is a string v' of length at most $|Q_f| \cdot 2^{|Q|}$ such that $v'w \in safe_{L2R}(G_q)$.*

PROOF. This follows from Lemma 14 and a pumping argument for the product automaton B combining $\hat{A}_{\text{L2R}}$ and $A(R_f)$: For any two states $(X_1, p_1), (X_2, p_2) \in \hat{Q}_{L2R} \times Q_f$ there is a string v with $\delta_B((X_1, p_1), v) = (X_2, p_2)$ if and only if there is such a string v' of length at most $|\hat{Q}_{L2R} \times Q_f| = |Q_f| \cdot 2^{|Q|}$. □

THEOREM 16. L2RALL ∈ EXPSPACE

PROOF. We give a nondeterministic exponential-space algorithm $\mathcal{A}$ deciding $\overline{\text{L2RALL}}$, the complement of L2RALL. This yields the result since EXPSPACE is closed under complement and NEXPSPACE = EXPSPACE thanks to Savitch's Theorem [10].

The idea is that $\mathcal{A}$ guesses a symbol $f \in \Gamma$ and strings u, w such that $ufw \in \text{safe}_{\text{L2R}^+}(G) \setminus \text{safe}_{L2R}(G)$ is a witness string on which JULIET plays Call in the first pass on ufw. Thanks to Lemma 15, $\mathcal{A}$ 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 $uvw \in \text{safe}_{L2R}(G)$. A short summary of $\mathcal{A}$ is given as Algorithm 2.

Algorithm 2 Test for $G \in$ L2RALL

```
Guess f ∈ Γ and a dual relative effect ê_uf
while Guessing a string u in a streaming fashion do
    Use A_L2R(G) to compute the set U = E[u](q_0) non-
    deterministically
    Use Â_L2R(G) to nondeterministically verify ê_uf ∈
    Ê[uf](q_0)
Guess a string w and compute F(w) by simulating
A_L2R(G) backwards
if ê_uf ∩ F(w) = ∅ then
    // ufw ∉ safe_L2R(G)
    for all v ∈ R_f with |v| ≤ |Q_f| · 2^|Q| do
        Guess a set U_v ∈ U
        for all q ∈ U_v do
            Simulate A_L2R^{q,F(w)} on input v
            if A_L2R^{q,F(w)} accepts v then
                // uvf ∈ safe_L2R(G)
            else
                Reject
    Accept
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 $U = \{U_1, U_2, \ldots, U_n\}$, $\mathcal{A}$ guesses u in a streaming fashion, one symbol at a time. It simulates A_{L2R} on u and computes $E[u](q_0)$ online. This can be done in exponential space by storing the set $E[u](q_0) \in \mathcal{P}(\mathcal{P}(Q))$. At the same time, having guessed a dual relative effect $\hat{e}_{uf}$, it guesses a run of $\hat{A}_{\text{L2R}}$ on uf, effectively verifying that there is a strategy corresponding to this relative effect.

Afterwards, to compute $F(w) \in \mathcal{P}(Q)$, $\mathcal{A}$ guesses a string w, and incrementally computes a set $F(w) \subseteq Q$ of states from which JULIET can win the game as defined in Lemma 14. The set $F(sw')$ can be computed from the set $F(w')$ by checking, for each $q \in Q$, whether $A_{\text{L2R}}^{q,F(w')}$ 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 $F(\epsilon) = F$ the correctness of this incremental computation follows by a simple induction argument.

The algorithm then checks whether $\hat{e}_{uf}$ contains a state from $F(w)$. If it does not, we know that $ufw \notin \text{safe}_{L2R}(G)$. If it does, $\mathcal{A}$ immediately rejects.

Finally, $\mathcal{A}$ checks for all strings $v \in R_f$ of length at most $|Q_f| \cdot 2^{|Q|}$, if $uvw \in \text{safe}_{L2R}(G)$. 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 $U_i \in U$ and testing whether for every $q \in U_i$ there is a relative effect $e \in E[v](q)$ such that $e \subseteq F(w)$.

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

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

If $\mathcal{A}$ 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 $|Q_f| \cdot 2^{|Q|}$ there exists a set $U_v \in E[u](q_0)$ such that v is accepted by $A_{\text{L2R}}^{q,F(w)}$ 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 σ_v of JULIET on u such that for all states $q \in e(u, \sigma_v, q_0)$, JULIET has a winning strategy on vw starting at q.

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

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

Since JULIET has no winning L2R strategy on ufw, ROMEO must have a strategy τ on uf such that $\hat{e}(uf, \tau, q_0) \cap F(w) = \emptyset$. Since this dual relative effect can be guessed by $\mathcal{A}$, the test on line 6 can be passed.

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

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

THEOREM 17. L2RALL $\in$ 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 $\mathcal{A}'$ contains an outer loop over all $f \in \Gamma$, sets $W \subseteq Q$ and vectors of sets $U_1, \dots, U_{|R_f|} \in \mathcal{P}(Q)$. Inside this loop, similar to algorithm 2, $\mathcal{A}'$ checks if there are strings u and w such that $U_1, \dots, U_{|R_f|} \in E[u](q_0)$ and $W = F(w)$; then, all $\mathcal{A}'$ needs to do is check for all $i = 1, \dots, |R_f|$ whether $\delta_{\text{L2R}}{}^*(U_i, v_i) \cap \mathcal{P}(F(w) \neq \emptyset$ (with $R_f = \{v_1, \dots, v_{|R_f|}\}$) and $\hat{A}_{\text{L2R}}$ accepts ufw.

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

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

Finally, $\mathcal{A}$ runs one copy of A_{L2R} with starting state U_i and final state set $\mathcal{P}(W)$ on v_i for each $i = 1, \dots, |R_f|$ and runs $\hat{A}_{\text{L2R}}$ on ufw; if all copies of A_{L2R} and $\hat{A}_{\text{L2R}}$ accept, $\mathcal{A}$ 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, $\mathcal{A}'$ is an EXPTIME algorithm deciding L2RALL for games with finite replacement languages, given explicitly as part of the input. $\square$

7. LOWER BOUNDS

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

PROPOSITION 18. L2RALL *is hard for* EXPSPACE.

PROOF (SKETCH). The proof is by a reduction from the EXPONENTIAL WIDTH CORRIDOR TILING problem. In this problem, we are given a set $U = \{u_1, \dots, u_k\}$ of tiles with a designated *initial tile* $u_I \in U$ and *final tile* $u_F \in U$. There are also two relations $H, V \subseteq U \times U$. These are the *horizontal* and *vertical* constraints, respectively. A tile u_j is only allowed to the right of a tile u_i 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 ℓ is a mapping $t : \{0, \dots, \ell - 1\} \times \{0, \dots, m\} \to U$, for some m. A tiling t is *valid* if

- $t(0, 0) = u_I$,
- $t(\ell - 1, m) = u_F$,

- for every $i \in \{0, \ldots, \ell - 2\}$ and $j \in \{0, \ldots, m\}$, $(t(i, j), t(i + 1, j)) \in H$, and
- for every $i \in \{0, \ldots, \ell - 1\}$ and $j \in \{0, \ldots, m - 1\}$, $(t(i, j), t(i, j + 1)) \in V$.

Exponential Width Corridor Tiling asks whether an instance $\mathcal{I} = (U, u_I, u_F, V, H, n)$ has a valid corridor tiling of width 2^n. This problem is well known to be EXPSPACE-complete; see, e.g., [5, 11].

Given an input instance $\mathcal{I} = (U, u_I, u_F, V, H, n)$ for Exponential Width Corridor Tiling, we construct from $\mathcal{I}$ a context-free game $G = (\Sigma, R, T)$ such that there exists a valid corridor tiling for $\mathcal{I}$ if and only if there is a string for which Juliet has a winning L2R$^+$strategy but no L2R strategy in G. The claim then follows from this reduction by Lemma 4.

The rough idea of the construction of G is as follows. Let 2^n be the target width for a tiling. Tilings are encoded by strings of the form $v = ((uc)^*)\#)^*$, where u is a tile and c a 0-1-string of length n that should encode the column number of the position of u. A sequence $(uc)^*$ encodes a row of a tiling and rows are separated by $\#$. For each column number $i \in \{0, 1, \ldots, 2^n - 1\}$, we denote by $c(i)$ the encoding of i as a binary string of length n over $\{0, 1\}$.

We construct G in such a way that all the strings in $\text{safe}_{\text{L2R}^+}(G) \setminus \text{safe}_{L2R}(G)$ are of the form gvf, where v is the encoding of a correct tiling.

The main task of Juliet in the game is to show that the middle part v of the input string indeed represents a correct tiling, while Romeo tries to disprove her. For this, we utilize a *protest technique* [9], in which we force Juliet to call potentially inconsistent symbols in the input, allowing Romeo to flag constraint violations. The additional symbols f and g are primarily meant to ensure that Juliet needs a L2R$^+$strategy to win; f is also needed to identify violations of vertical constraints, as we shall describe later.

We next sketch the different ways in which a string v of the form $(U0^n(U\{0,1\}^n)^*U1^n\#)^*$ may fail to encode a valid tiling. After that, we examine how to deal with these types of violations.

- *Horizontal error:* v violates the horizontal constraints, i.e. v contains a substring of the form $u\{0,1\}^n u'$ with $(u, u') \notin H$;
- *Constant error:* The first (last) symbol from U in v is not u_I (u_F);
- *Increment error:* Two subsequent column number encodings are inconsistent, i.e. v contains a substring of the form $c(i)Uc(j)$ with $j \neq i + 1$;
- *Vertical error:* v violates the vertical constraints, i.e. v contains a substring of the form $uc(i)(U \cup \{0,1\})^*\#(U \cup \{0,1\})^*u'c(i)$ with $(u, u') \notin V$ for some $i \leq 2^n - 1$.

We construct G such that Romeo can win without any effort on inputs with horizontal or constant errors and by pinpointing positions with increment or vertical errors otherwise. Horizontal and constant errors can be basically tested by the target DFA, so we merely need to make certain that strings with these kinds of errors can never be rewritten into the target language.

In the main part of the game, during the second pass, Juliet calls all positions of tiling symbols and gives Romeo the possibility to mark a violating position. If v contains an increment error at some position, Romeo can prove this with a simple subgame. Verifying vertical errors is slightly more complicated. To this end, Juliet has to allow Romeo to add an n-digit number c_f to the end of v in the single move of the first pass. Romeo should pick c_f as the encoding of the number of a column in which a vertical error occurs. In the main part, Romeo can then indicate the positions of the two tiles of that error and in a subgame it is verifed that that are actually in the same column (with number c_f) on consecutive rows.

We force Juliet to call all positions of tiling symbols by introducing into the alphabet a disjoint copy $\hat{U}$ of U, the set of *marked tiles*, as well as a *protest symbol* @. The idea is that for as long as Juliet keeps calling tile positions in order, Romeo replaces those tiles with their corresponding marked tiles, but as soon as Juliet skips a tile, Romeo protests by returning @ the next time Juliet plays a call move. By including only appropriate strings in the target language, we make sure that Romeo wins on strings on which Juliet has tried to "cheat" by skipping a tile and Romeo has rightfully protested, and that Romeo loses on strings on which he protests without just cause.

Increment errors are dealt with in a similar manner by use of a *number protest symbol* $@_N$. As soon as Juliet calls the tile position immediately before the violating substring $c(i)Uc(j)$, Romeo returns $@_N$, signifying that Juliet now has to call each of the n bits to the right of $@_N$ in turn until Romeo returns a *flag bit* 0_N or 1_N to pinpoint a position in $c(i)$ that witnesses $j \neq i + 1$. (The correctness of this flagging procedure follow from Lemma 19 below.) Similarly as for tiles, we use additional *marked bits* $\hat{0}, \hat{1}$ and the protest symbol @ to force Juliet to call each position of $c(i)$.

Finally, to handle vertical errors, we add another disjoint copy U^V of U, called *flagged tiles* to the alphabet. As described above, after giving the encoding c_f of a column where vertical constraints are violated, Romeo replaces two tiles involved in this violation by their corresponding flagged tiles. Again, we need to make sure via the target language that Romeo always wins on rewritten strings with correctly flagged vertical errors and loses on strings with incorrect claims of vertical errors.

More details of the proof can be found in the full version of this paper. □

Lemma 19. *For a number $i \in \{0, 1, \ldots, 2^n - 1\}$ let $c(i)$ be the n-bit binary encoding of i, and for an n-bit string c let c_k denote the k-th position of c. For any two numbers $i, j \in \{0, 1, \ldots, 2^n - 1\}$, it holds that $j \neq i + 1$ if and only if there exists a number $k \leq n$ such that one of the following conditions holds:*

(a) $c(i)_k \neq c(j)_k$, $c(i) \neq 1^n$ and for some $k' > k$, it holds that $c(i)_{k'} = 0$ or $c(j)_{k'} = 1$;

(b) $c(i)_k = c(j)_k$ and it holds that either $k = n$ or $c(i)_{k+1} = 1$ and $c(j)_{k+1} = 0$

(c) $c(i) = 1^n$

The proof of the following result is given in the full version of this paper.

Proposition 20. L2RAll *is hard for* EXPTIME*, even for games with finite replacement language.*

PROOF (SKETCH). 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 was shown to be EXPTIME-complete in [9].

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' ensures that JULIET can deduce a winning strategy on a string $g_0 u h_0$ 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 Σ' 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 Σ.

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

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

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 $g_1 w h_1$ with $\text{cl}(w) \in T$;
- all strings $g_2 w h_2$ with $\text{cl}(w) \in T$;
- the string $g_0 u @$;
- all strings of the form gwh 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 Σ.

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

That (a) and (b) are indeed equivalent is shown in the full paper. □

8. CONCLUSION

We investigated a practically relevant restriction of strategies for context-free games and their relation to general strategies. That L2RALL is EXPSPACE-complete in general but EXPTIME-complete in the restricted case where the replacement languages in G are finite, is somewhat surprising, since the L2R word problem, i.e., checking whether a given string is safely rewritable in a left-to-right fashion, is EXPTIME-complete in both cases[9].

The automaton construction for 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-rocq.inria.fr/verso/Gemo/Projects/axml.

[4] H. Björklund, M. Schuster, T. Schwentick, and J. Kulbatzki. On optimum left-to-right strategies for active context-free games, December 2012. Available online at http://arxiv.org/abs/1212.3501.

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

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

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

[8] Tova Milo, Serge Abiteboul, Bernd Amann, Omar Benjelloun, and Frederic Dang Ngoc. Exchanging intensional XML data. *ACM Trans. Database Syst.*, 30(1):1–40, 2005.

[9] Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Active context-free games. *Theory Comput. Syst.*, 39(1):237–276, 2006.

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

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