On the μ-Calculus Augmented with Sabotage

Philipp Rohde

RWTH Aachen, Informatik VII
rohde@informatik.rwth-aachen.de

Abstract. We study logics and games over dynamically changing structures. Van Benthem's sabotage modal logic consists of modal logic with a cross-model modality referring to submodels from which a transition has been removed. We add constructors for forming least and greatest monadic fixed-points to that logic and obtain the sabotage μ-calculus. We introduce backup parity games as an extension of standard parity games where in addition, both players are able to delete edges of the arena and to store, resp., restore the current appearance of the arena by use of a fixed number of registers. We show that these games serve as model checking games for the sabotage μ-calculus, even if the access to registers follows a stack discipline. The problem of solving the games with restricted register access turns out to be PSPACE-complete while the more general games without a limited access become EXPTIME-complete (for at least three registers).

1 Introduction

In the classical framework of logics and corresponding model checking games, one considers changes of system states or movements of agents within a system, but the underlying structure is assumed to be static. This motivates the study of more general specification formalisms where we can directly address temporal changes of structures. In this contribution, we focus on the deletion of objects. Applications are, for example, (1) computer networks where connections may break down; (2) car navigation systems that cope with roadworks and traffic jams; (3) representations of knowledge where an increase in knowledge corresponds to a removal of uncertainty relations; and (4) Euler's famous problem of Seven Bridges of Königsberg (edges are removed after they were traversed for the first time). An algorithmic task for these systems is, for example, the reachability of designated states.

Van Benthem [2] proposed a modal logic with a transition-deleting modality, called sabotage modal logic SML. The main limitation of modal logics is the lack of a mechanism for unbounded iteration or recursion. To overcome this, we augment SML with constructors for forming least and greatest monadic fixed-points, which yields the sabotage μ-calculus SL_μ. This logic is capable of expressing iterative properties like reachability or recurrence as well as basic changes of the underlying structure, namely, the deletion of transitions.

In Section 2, we define the sabotage μ-calculus SL_μ and repeat some known results about the modal fragment SML. In Section 3, we introduce backup parity games as token-moving games between two players. Depending on the type of the current vertex, the owner can decide on the further direction, or he can delete edges, or the current appearance of the arena is stored, resp., restored by use of a fixed number of registers. As winning condition for infinite plays, we use the well-known parity condition. In order to keep the complexity of solving these games low, we additionally require that registers can only be accessed by following a stack discipline: New values stored in a higher register also overwrite the values of all lower registers and the restoring of edges out of a higher register also erases all values of lower registers. The restriction on the register access guarantees that these games can be solved in polynomial space with respect to the size of the arena (when the number of registers is fixed). We also show that the problem of solving the games without this limited access becomes EXPTIME-complete, even for games with three registers.

In Section 4, we show that the model checking problem for SL_μ can be reduced to the problem of solving a backup parity game with limited access (by a polynomial time reduction). In fact, the maximum number of nested fixed-points of the given formula gives the number of registers in the game, and the dependency order of inductive fixed-point constructions corresponds to the stack discipline of register access. We conclude with Section 5 by giving a summary of the presented results and stating some open questions.

Due to lack of space, proofs are omitted or only sketched. Full proofs can be found in [10–Chaps. 5–6].

2 Sabotage μ-Calculus

Recall that the μ-calculus L_μ is obtained by adding constructors for forming least and greatest monadic fixed-points to propositional modal logic [7]. We extend this logic to the sabotage μ-calculus SL_μ by adding a cross-model modality referring to submodels from which a transition has been removed. For convenience, we define the syntax of SL_μ in negation normal form. All results easily extend to the general case (with the restriction that bounded variables only occur positively).

In what follows, let Σ be a finite alphabet, Prop a finite set of unary predicate symbols, and $\mathrm{Var} = \{X, Y, \ldots\}$ a set of propositional variables.

Definition 1. *A* Kripke structure $\mathcal{K}$ *over* Prop *is a tuple* (S, Σ, R, L)*, where* S *is an (at most countable) set of states,* $R \subseteq S \times \Sigma \times S$ *is a transition relation, and* $L : S \to 2^{\mathrm{Prop}}$ *is a labeling function assigning sets of predicates to states. Its size is defined by* $|\mathcal{K}| := |S| + |R|$*. For a set* $E \subseteq R$*, we define the substructure* $\mathcal{K} \setminus E := (S, \Sigma, R \setminus E, L)$*.*

Definition 2. *Formulae of the* sabotage μ-calculus SL_μ *are inductively defined by the following grammar. For* $p \in \mathrm{Prop}$*,* $a \in \Sigma$*, and* $X \in \mathrm{Var}$*, let*

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \Diamond_a \varphi \mid \Box_a \varphi \mid \Diamonddot_a \varphi \mid \boxdot_a \varphi \mid \mu X.\varphi \mid \nu X.\varphi \ .$$

The fragment of formulae without fixed-point operators is called sabotage modal logic SML. *Let* $\mathrm{Cl}(\varphi)$ *be the set of subformulae,* $\mathrm{Var}(\varphi)$ *the set of variables and* $\mathrm{Bd}(\varphi)$ *the set of bounded variables of* φ*. Without loss of generality, we only deal with well-named formulae where every variable is bounded at most once and free variables are distinct from bounded variables. For each* $X \in \mathrm{Bd}(\varphi)$*, there is a unique binding definition* $\mathrm{Df}_\varphi(X) \in \mathrm{Cl}(\varphi)$ *equal to* $\mu X.\psi$ *or* $\nu X.\psi$*.*

Let $\mathcal{K} = (S, \Sigma, R, L)$ *be a Kripke structure and* φ *an* SL_μ*-formula. A* valuation *of* φ in $\mathcal{K}$ *is a function* $\mathcal{V} : \mathrm{Var}(\varphi) \to 2^S$*. The semantics of* SL_μ *is defined as the set* $\|\varphi\|^{\mathcal{K}}_{\mathcal{V}}$ *of states in which* φ *is true:*

$$\begin{aligned}
\|\top\|^{\mathcal{K}}_{\mathcal{V}} &:= S, \quad \|p\|^{\mathcal{K}}_{\mathcal{V}} := \{s \in S \mid p \in L(s)\}, \\
\|X\|^{\mathcal{K}}_{\mathcal{V}} &:= \mathcal{V}(X), \quad \|\varphi_1 \vee \varphi_2\|^{\mathcal{K}}_{\mathcal{V}} := \|\varphi_1\|^{\mathcal{K}}_{\mathcal{V}} \cup \|\varphi_2\|^{\mathcal{K}}_{\mathcal{V}}, \\
\|\Diamond_a \varphi\|^{\mathcal{K}}_{\mathcal{V}} &:= \{s \in S \mid \exists s' \in S : (s, a, s') \in R \wedge s' \in \|\varphi\|^{\mathcal{K}}_{\mathcal{V}}\}, \\
\|\Diamond\!\!\!\cdot_a \varphi\|^{\mathcal{K}}_{\mathcal{V}} &:= \{s \in S \mid \exists t, t' \in S : (t, a, t') \in R \wedge s \in \|\varphi\|^{\mathcal{K}\setminus\{(t,a,t')\}}_{\mathcal{V}}\}, \text{ and} \\
\|\mu X.\varphi\|^{\mathcal{K}}_{\mathcal{V}} &:= \bigcap\{A \subseteq S \mid \|\varphi\|^{\mathcal{K}}_{\mathcal{V}[X:=A]} \subseteq A\}.
\end{aligned}$$

The semantics for the other operators is defined dually. For convenience, we set $(\mathcal{K}, s, \mathcal{V}) \models \varphi$ *iff* $s \in \|\varphi\|^{\mathcal{K}}_{\mathcal{V}}$*.*

We need to justify the definition of fixed-point formulae. Let φ be an SL_μ-formula. Then the function $A \mapsto \|\varphi\|^{\mathcal{K}}_{\mathcal{V}[X:=A]}$ is monotone and thus, it has a unique least fixed-points by Knaster-Tarski, which is equal to $\|\mu X.\varphi\|^{\mathcal{K}}_{\mathcal{V}}$.

Remark 1. There is a fundamental difference between $\Diamond_a$ and $\Diamond\!\!\!\cdot_a$ with respect to fixed-points. When least and greatest fixed-points are constructed inductively, then movements are passed to the next stage, while the deletion of transitions is 'encapsulated' within a stage: The deletion is always restored when we proceed to the next step. This is due to the fact that, if we have determined $F_i = \|\psi\|^{\mathcal{K}}_{\mathcal{V}[X:=F_{i-1}]}$ for some formula ψ with deletion modalities and proceed to F_{i+1}, then we calculate $\|\psi\|^{\mathcal{K}}_{\mathcal{V}[X:=F_i]}$ over $\mathcal{K}$ and not over the substructure that results from the deletion of transitions.

Before we turn to the investigation of SL_μ, we repeat some results about the fragment SML. We start with an example:

Example 1. Consider the SML-formula $\varphi := \Diamond_a \Diamond_a \top \wedge \boxdot_a \Box_a \bot$. It is easy to see that every model of φ has exactly one a-transition and that it is a loop at the origin. In particular, SML is not bisimulation-invariant.

In [8, 9], it was shown that the sabotage modality already strengthens modal logic in such a way that all nice model-theoretic properties and algorithmic complexities get lost. In fact, from the viewpoint of complexities, SML much more resembles first-order logic than modal logic (with the exception that the formula complexity remains in PTIME):

Theorem 1 ([8, 9]). *For every* SML*-formula φ, there is an effectively constructible equivalent* FO*-formula with a size polynomial in $|\varphi|$. The model checking problem for* SML *is* PSPACE*-complete. Further,* SML *lacks the finite model property and the satisfiability problem becomes undecidable.* □

We proceed with two examples dealing with fixed-points and sabotage.

Example 2. For a given Kripke structure $\mathcal{K} = (S, \Sigma, R, L)$ and state $s \in S$ let $\hat{\mathcal{K}}_s$ be the unraveling of $\mathcal{K}$ at s. For convenience, we assume a unary alphabet Σ. We say that $\hat{\mathcal{K}}_s$ *contains a perfect subtree* if there is a non-empty subtree of $\hat{\mathcal{K}}_s$ such that each path of this subtree contains infinitely many splitting points. Let $\varphi := \nu X.\mu Y.(\boxdot\Diamond X \vee \Diamond Y)$ and suppose that $R \neq \emptyset$. We claim that $(\mathcal{K}, s) \models \varphi$ iff $\hat{\mathcal{K}}_s$ contains a perfect subtree. The L_μ-formula $\mu Y.(\psi \vee \Diamond Y)$ expresses that there is a finite path to a state where ψ holds. The subformula $\boxdot\Diamond X$ guarantees that there are at least two successors of the current state that belong to (the interpretation of) X. Let $G \subseteq S$ be the outer, greatest fixed-point according to X. If $s \in G$, then G is a perfect subtree of $\hat{\mathcal{K}}_s$. Conversely, if $P \subseteq S$ witnesses a perfect subtree of $\hat{\mathcal{K}}_s$, then $P \subseteq G$ by construction. Since P is prefix-closed, it follows that $s \in P$ and hence also $s \in G$.

The next example shows that we can ensure an infinite 'depth' of models:

Example 3. Let $\psi := \Diamond_b\top \wedge \Diamonddot_b(\Box_b\bot \wedge \nu Y.(\Box_a(\Diamond_b\top \wedge Y)))$ and $\varphi := \nu X.(\Diamond_a X \wedge \psi)$. Let $\mathcal{K} = (S, \Sigma, R, L)$ be a Kripke structure. Suppose that $(\mathcal{K}, t) \models \psi$ for some $t \in S$. Then t has exactly one outgoing b-transition and if this b-transition is removed, then every state that is reachable from t by a non-empty path along a-transitions still has a b-successor (due to the greatest fixed-point according to Y). In particular, every state that is reachable from t by a non-empty a-path is distinct from t. Suppose now that $(\mathcal{K}, s) \models \varphi$. Due to the greatest fixed-point according to X, there is an infinite path $\pi = s_0 \xrightarrow{a} s_1 \xrightarrow{a} s_2 \xrightarrow{a} \ldots$ with $s_0 = s$ and $(\mathcal{K}, s_i) \models \psi$ for every $i \in \mathbb{N}$. It follows that $s_i \neq s_j$ for every $i < j$ and thus, π consists of infinitely many pairwise distinct elements.

The complexity of model checking SL_μ can be readily determined.

Lemma 1. *The model checking problem for SL_μ is* PSPACE*-complete. The model checking problem for SL_μ with a fixed formula is* PTIME*-complete (program complexity).*

Proof. The hardness of combined model checking follows from the fact that SML is a fragment of SL_μ. Further, we can extend the embedding of SML into FO (cf. Theorem 1) to an embedding of SL_μ into $\mathrm{LFP}_{\mathrm{mon}}$, the first-order logic with least fixed-points over monadic relations. The model checking problem of the latter logic is known to be PSPACE-complete [13].

Further, the program complexity of L_μ is PTIME-complete [3] and L_μ is a fragment of SL_μ. Again, we can embed SL_μ into (full) LFP as above and we obtain equivalent LFP-formulae that are polynomial with respect to the sizes of the SL_μ-formulae. Since the program complexity of LFP is known to be PTIME-complete [12], the statement follows. □

3 Backup Parity Games

Recall that parity games are closely related to L_μ: They serve as model checking games for L_μ [5] and conversely, the winning condition of a parity game can be expressed by an L_μ-formula [14]. Despite the aforementioned solution of the model checking problem for SL_μ via LFP, we want to define a model checking game for SL_μ in the style of parity games. The games are defined as token-moving games between two players (called 0 and 1). Depending on the type of the current vertex, the owner of the vertex can decide on the further direction, or he can delete edges, or the current arena is stored, resp., restored. We use the parity condition as winning condition for infinite plays. To obtain a lower complexity, we require that the storing and restoring operations follow a stack discipline: New values stored in a higher register also overwrite the values of all lower registers and the restoring of edges out of a higher register erases all values of lower registers.

Definition 3. *Let (V, E) be a graph. For $v \in V$, let vE be the set of E-successors of v. If vE is a singleton set, then $\mathrm{scc}(v)$ denotes its unique element. For a set $A \subseteq V$, let $AE := \bigcup_{v \in A} vE$ be the set of E-successors of elements in A. Finally, let $\mathrm{Out}(A) := \{(v, v') \in E \mid v \in A, v' \in V\}$ be the set of edges with sources in A. A* backup parity game *of index n with m registers, (n, m)-backup game for short, is given by $\mathcal{G} = (\mathcal{A}, v_{\mathrm{in}})$, where $\mathcal{A}$ is an arena and v_{in} is an initial vertex of $\mathcal{A}$. An* arena *is a labeled graph $\mathcal{A} = (V, E, \varDelta, \varOmega)$ where V is a non-empty, finite set of vertices that can be partitioned into the following sets: (1) movement vertices M_i of player i; (2) deletion vertices D_i of player i; (3) storing vertices S_j for $j \in [1, m]$; and (4) restoring vertices R_j for $j \in [1, m]$. In this case, we write $V = (M_i, D_i, S_j, R_j)$. Let $M := M_0 \cup M_1$ be the set of movement vertices and $D := D_0 \cup D_1$ the set of deletion vertices. For the edge relation $E \subseteq V \times V$, we require that $|vE| = 1$ for each $v \in V \setminus M$. Finally, $\varDelta \subseteq D \times 2^{\mathrm{Out}(M)}$ is a deletion relation and $\varOmega : V \to \{0 \dots n\}$ is a priority function.*

A position *of the game is an element of $V \times (2^E)^{m+1}$. The initial position is $(v_{\mathrm{in}}, E \dots E)$. Let $(v, Y, X_1 \dots X_m)$ be the current position. Depending on v, a legal successor position is defined as follows. Assume that $v \in M_i$ for $i \in \{0, 1\}$. If $vY = \emptyset$, then Player i has lost the play. Otherwise, Player i chooses $v' \in vY$ and the new position becomes $(v', Y, X_1 \dots X_m)$. Assume that $v \in D_i$ for $i \in \{0, 1\}$. If there is no $\varXi$ with $(v, \varXi) \in \varDelta$ and $\emptyset \neq \varXi \subseteq Y$, then Player i has lost the play. Otherwise, Player i chooses such a set $\varXi$ and the new position becomes $(\mathrm{scc}(v), Y \setminus \varXi, X_1 \dots X_m)$. If $v \in S_j$ for $j \in [1, m]$, then the new position becomes $(\mathrm{scc}(v), Y, Y \dots Y, X_{j+1} \dots X_m)$. Finally, if $v \in R_j$ for $j \in [1, m]$, then the new position becomes $(\mathrm{scc}(v), X_j, X_j \dots X_j, X_{j+1} \dots X_m)$.*

If the game goes on infinitely and the greatest number appearing infinitely often in the sequence $\varOmega(v_0)\varOmega(v_1)\varOmega(v_2)\dots$ is even, then Player 0 wins the play; otherwise Player 1 wins. Finally, the size of a game is defined as $|\mathcal{A}| := |V| + |E| + \sum_{v \in D} \sum_{\varXi : (v, \varXi) \in \varDelta} |\varXi|$.

Remark 2. By definition, a player gets stuck if either the current vertex is a moving vertex, but it has no successors with respect to the current set of edges.

Or it is a deletion vertex, but an appropriate deletion is not possible. Further, one has $Y \subseteq X_1 \subseteq \ldots \subseteq X_m \subseteq E$ for any position $(v, Y, X_1 \ldots X_m)$ that is reachable from the initial position.

We say that a game is in *normal form*, if Δ is a function $\Delta : D \to \mathrm{Out}(M)$ (in which case we also write δ) and $\Omega(v) = 0$ for each $v \in V \backslash M$. It is straightforward to show that for every (n, m)-backup game there is an equivalent (n, m)-backup game in normal form with a size linear in the size of the original game. For backup games in normal form, player-based choices are only made at movement vertices while for all other vertices, the successor position is uniquely determined (provided that the play does not end). Note that it suffices to consider only n and m that are bounded by some term in $\mathcal{O}(|V|)$.

There is a straightforward transformation of backup parity games into standard parity games, but at the cost of an exponential blow-up of the arena. To this end, the current appearance of the arena and the content of registers are encoded within the vertices of the new game.

Lemma 2. *For any (n, m)-backup game $\mathcal{G} = (\mathcal{A}, v_{\mathrm{in}})$, there is an equivalent parity game $\mathcal{G}' = (\mathcal{A}', v'_{\mathrm{in}})$ of same index such that $|\mathcal{A}'| \in \mathcal{O}(|\mathcal{A}| \cdot 2^{(m+1)|E|})$, where E is the set of edges in $\mathcal{A}$.* □

In fact, due to the restricted access to registers, the size of an equivalent parity game can be improved to $\mathcal{O}(|\mathcal{A}| \cdot (m+2)^{|E|})$. Since parity games are determined [4], we immediately obtain that for any (n, m)-backup game over the arena (V, E, Δ, Ω), the set of positions $V \times (2^E)^{m+1}$ can be partitioned into the winning regions W_0 and W_1 such that Player τ has a positional winning strategy on W_τ. Note that positional strategies for backup games can be easily transformed into automaton strategies over the arena using deterministic Mealy automata.

We turn to the algorithmic complexity of the problem of solving backup games, that is, the problem of deciding whether Player 0 can win the game $\mathcal{G} = (\mathcal{A}, v_{\mathrm{in}})$ starting from the initial position $(v_{\mathrm{in}}, E \ldots E)$, no matter how Player 1 moves (E is the set of edges in $\mathcal{A}$).

Theorem 2. *For a fixed number of registers, the problem of solving backup games is* PSPACE*-complete.*

Proof (Sketch). In [8], it was shown that the so-called *sabotage game*, where one player moves along edges of a finite graph and the other player removes an arbitrary edge in each round, is PSPACE-hard when the reachability of designated vertices is considered as game objective. Since the sabotage game is a special backup game, it follows that the problem of solving backup games is PSPACE-hard, even when restricted to games without priorities and without registers.

Let $\mathcal{G} = (\mathcal{A}, v_{\mathrm{in}})$ be an (n, m)-backup game with $\mathcal{A} = (V, E, \delta, \Omega)$ and $V = (M_i, D_i, S_j, R_j)$. Without loss of generality, we assume that $\mathcal{G}$ is in normal form. In what follows, we show that it can be decided whether Player 0 wins the game from $(v_{\mathrm{in}}, E \ldots E)$ in a space polynomial with respect to $|\mathcal{A}|$. To this end, we sketch a recursive alternating procedure with a running time polynomial in $|\mathcal{A}|$

(for m fixed). By APTIME = PSPACE (cf. [1]), it follows that for a fixed number of registers, the problem of solving the games belongs to PSPACE.

The algorithm is called with the current position $(v, Y, X_1 \dots X_m)$ as parameter (among some other data that is explained below). If v is a movement vertex, but a sink, or v is a deletion vertex, but the demanded edge is not present, then the algorithm immediately stops. In this case, it accepts or rejects subject to the player to which vertex v belongs. In all other cases, a successor vertex v' is chosen and the procedure is recursively called with parameter $(v', Y', X'_1 \dots X'_m)$ depending on the type of v. If $v \in M_0$, then the successor v' is non-deterministically guessed. If $v \in M_1$, then v' is chosen universally. If v is a deletion, storing, or restoring vertex, then its successor v' is uniquely determined. The parameters $Y', X'_1 \dots X'_m$ are then chosen according to the update rules of positions.

Beside the current position, the algorithm remembers for each vertex from $V \setminus D$ whether it was already visited and if so, which was the highest priority since then. This information is partly reset whensoever vertices are visited that alter the set of edges or the value of registers. The memory is realized by the functions $\tau : M \to [-1, n]$ as well as $\sigma_j : S_j \to [-1, n]$ and $\rho_j : R_j \to [-1, n]$ for $j \in [1, m]$. A function value of -1 means that this vertex was not seen yet or that this information was reset in the meantime. A function value greater or equal 0 gives the highest priority since the last visit of the respective vertex. The functions are updated depending on the type of the current vertex v:

- $v \in M$: If $\tau(v) \geq 0$, then the algorithm terminates. Otherwise, $\tau(v)$ is set to be 0. The value of $\tau(w)$ for each w in the domain of τ is updated to $\Omega(v)$ if $0 \leq \tau(w) < \Omega(v)$. The functions $\sigma_1 \dots \sigma_m$ and $\rho_1 \dots \rho_m$ are updated analogously.
- $v \in D$: The entire function τ is reset.
- $v \in S_j$ for $j \in [1, m]$: If $Y = X_j$ and $\sigma_j(v) \geq 0$, then the algorithm terminates. Otherwise, the entire functions τ, $\sigma_1 \dots \sigma_{j-1}$ and $\rho_1 \dots \rho_{j-1}$ are reset. Additionally, if $Y \subsetneq X_j$, then the functions σ_j and ρ_j are also reset. Finally, the value $\sigma_j(v)$ is set to be 0.
- $v \in R_j$ for $j \in [1, m]$: If $\rho_j(v) \geq 0$, then the algorithm terminates. Otherwise, the entire functions τ, $\sigma_1 \dots \sigma_j$ and $\rho_1 \dots \rho_{j-1}$ are reset and the value $\rho_j(v)$ is set to be 0.

For the correctness of the algorithm, one shows that the following statements hold. First, each computation branch corresponds to an admissible prefix of a play. In fact, by the choice of parameters for recursive calls, the computation tree forms a complete prefix of a game tree according to a strategy of Player 0. Second, if the algorithm terminates, then one of two cases have occurred: Either the current position is a sink (with respect to movement or deletion) or the corresponding prefix of a play can be extended to a loop, where the highest priority of this loop is known to the algorithm. Note that in this case, by the update of the functions τ, $\sigma_1 \dots \sigma_m$ and $\rho_1 \dots \rho_m$, exactly the same position from $V \times (2^E)^{m+1}$ is repeated. Hence, the algorithm can decide the winner of the play that is constituted by infinitely many repetitions of the loop. By positional determinacy, a player wins the game iff he wins by moving always identical at same positions. Therefore, the validation of loops suffices to determine the winner.

We use a binary encoding of the parameters. Then each call of the procedure takes a time polynomial in $\mathcal{A}$. Regarding the running time, one shows that each computation branch of the alternating algorithm terminates after at most $\mathcal{O}(|V|^{4m+2})$ calls. There are four properties of backup games that are responsible for termination and that play a key role for estimating the running time. (1) Without deletion, storing, or restoring, a position is repeated after at most $|V|$ steps. (2) The deletion of edges is a one-way process: Without restoring, deletion vertices may occur at most $|D|$ times. If some deletion vertex is visited for the second time, then the related edge is no longer available and the corresponding player loses. (3) The storing of data by overwriting a register value with a *properly* smaller set is also bounded: Without storing or restoring by accessing higher registers, proper storing cannot be carried out more often than the number of edges. (4) Due to the dependency order, the algorithm is allowed to forget all information regarding lower registers when data is stored or restored. It follows that the alternating algorithm accepts its initial input iff Player 0 has a winning strategy in the game starting from $(v_{\text{in}}, E \dots E)$. This concludes the proof. $\square$

Next, we settle the complexity of backup parity games when we skip the restriction that storing and restoring has to a follow stack discipline.

Definition 4. *An* (n, m)-RAM game *is defined analogously to an* (n, m)*-backup game, but the update of game positions for storing and restoring vertices is modified as follows. Let* $(v, Y, X_1 \dots X_m)$ *be the current position in the game. If* $v \in S_j$ *for* $j \in [1, m]$*, then the new position becomes* $(\text{scc}(v), Y, X'_1 \dots X'_m)$*, where* $X'_i := Y$ *if* $i = j$ *and* $X'_i := X_i$ *otherwise. If* $v \in R_j$ *for* $j \in [1, m]$*, then the new position becomes* $(\text{scc}(v), X_j, X_1 \dots X_m)$*. Thus, registers are accessed independently of each other. The updates for the other vertices remain unchanged.*

Theorem 3. *For* $n \geq 1$ *and* $m \geq 3$*, the problem of solving* (n, m)*-RAM games is* EXPTIME*-complete.*

Proof (Sketch). Let $\mathcal{G} = (\mathcal{A}, v_{\text{in}})$ be an (n, m)-RAM game. We can use the same transformation of Lemma 2 to obtain an equivalent parity game $\mathcal{G}' = (\mathcal{A}', v'_{\text{in}})$ of index n such that $|\mathcal{A}'| \in \mathcal{O}(|\mathcal{A}| \cdot 2^{(m+1)|E|})$, where E is the set of edges in $\mathcal{A}$. By a result of Jurdziński [6], parity games can be solved in a time polynomial with respect to the size of the arena and exponential with respect to the index of the game. It follows that $\mathcal{G}$ can be solved in time exponential with respect to the size of the arena, the index of the game, and the number of registers. Since we can assume that $n, m \in \mathcal{O}(|\mathcal{A}|)$, it follows that the problem of solving (n, m)-RAM games belongs to EXPTIME.

To establish the EXPTIME-hardness, we give a reduction from a two-player game introduced by Stockmeyer and Chandra [11], which is called *block game* and which is known to be EXPTIME-hard. It consists of an undirected graph $\mathcal{A}$, where each edge is labeled by a, b, or c, together with two sets F_0 and

F_1 of winning vertices. The players move alternatingly. A position is a tuple (τ, N_0, N_1) where $\tau \in \{0, 1\}$ signifies whose turn it is, and N_0, N_1 are disjoint sets of markers (i.e., sets of vertices) that belong to Player 0 and Player 1. Assume that $(0, N_0, N_1)$ is the current position. Player 0 chooses one of his markers from N_0 and an edge label $x \in \{a, b, c\}$. Then he moves the chosen marker to a new vertex along a finite, non-empty path subject to the following conditions: (1) all traversed edges are labeled by x, and (2) no passed vertex (including the last one) carries a marker of either player. Player 0 immediately wins if he places his chosen marker on a vertex in F_0. The moves of Player 1 are defined analogously. The players are not permitted to pass. In order to cover plays where never any marker of Player τ is placed on a vertex in F_τ, we agree that Player 1 wins every infinite play.

We present an equivalent $(1, 3)$-RAM game $\mathcal{G}' = (\mathcal{A}', v_{\text{in}})$ for a given block game $\mathcal{G} = (\mathcal{A}, p_{\text{in}})$ with initial position p_{in} that can be computed in polynomial time with respect to $|\mathcal{A}|$. Let V be the set of vertices of $\mathcal{A}$. The arena $\mathcal{A}'$ consists of several copies of $\mathcal{A}$ and special components in between. The first register of $\mathcal{G}'$ always contains the edge set of the original arena $\mathcal{A}'$ and is used to guarantee a 'clean board' at the beginning of each round. Positions of $\mathcal{G}$ are encoded in the second register of $\mathcal{G}'$ (see below). The arena $\mathcal{A}'$ contains an initial part that encodes the position p_{in}. It follows a loop that simulates two successive moves in $\mathcal{G}$. Let p be the encoded position of $\mathcal{G}$ at the beginning of the loop. Without loss of generality, we assume that a turn of Player 0 is simulated first. By deletion of edges, Player 0 chooses a candidate p' for a successor position of p and its encoding is stored into the third register. Then it is verified whether p' is indeed a legal successor of p by alternatingly restoring the second and the third register and checking all conditions separately. Note that for RAM games, the restoring does not affect the value of the other registers. If the check was successful and p' is a winning position for Player 0 in the block game, then Player 0 also wins the RAM game. If p' is not winning, then the value of the third register is shifted to the second register (by restoring out of the third register and immediately storing into the second register). Afterwards, the same procedure is repeated, but now Player 1 is the one who chooses the candidate for a successor position. At the end of the loop, the second register contains the encoding of a position and it is again Player 0's turn.

Markers are simulated by sets of edges. Assume that the current position of $\mathcal{G}$ is $p = (0, N_0, N_1)$. Player 0's choice of a successor position p' is simulated as follows. First, the original set of edges is restored out of the first register. Second, both players propose the sets N_0' and N_1' by deleting $|V| - |N_i|$ edge sets each of which corresponds to a marker. The result is stored into register 3. Third, it is checked whether $N_1' = N_1$: If they differ, then Player 0 can choose some edge that is present in register 2, but not in register 3 and lead the play towards a sink of Player 1. And last, if $N_1' = N_1$, then it is verified that exactly one marker in N_0 was moved according to the rules of the block game. This is done by the following steps: (1) By entering a special component, Player 0 asserts that a marker at vertex $v \in V$ was moved; (2) it is checked whether v

carried a marker in N_0, but not in N_0'; (3) by entering a special component, Player 0 asserts that the marker at v was moved along a x-labeled path for some $x \in \{a, b, c\}$; (4) it is checked whether there is a non-empty x-labeled path from v to a vertex $w \in V$ such that no intermediate vertex carries a marker of either player: Player 0 chooses the edges that are traversed; if there is a marker from $N_0 \cup N_1$ at intermediate vertices, then Player 1 can lead the play to a sink of Player 0; (5) finally, it is checked whether no other marker than the one at v was moved (otherwise, Player 1 can lead the play to a sink of Player 0).

The priorities 0 and 1 are used to ensure that players do not move ad infinitum when they simulate the movement of markers. If the game does not end, then Player 1 wins, because priority 1 is visited infinitely often. $\square$

Note that a RAM game with only one register necessarily follows a stack discipline and thus, it is already a backup game that can be solved in polynomial space. It remains open whether this is true for RAM games with two registers.

4 A Model Checking Game for SL$_\mu$

In this section, we show that the model checking problem for SL$_\mu$ can be reduced to the problem of solving a backup game. We present a backup game $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ for a finite Kripke structure $\mathcal{K}$, an SL$_\mu$-formula φ, and a valuation $\mathcal{V}$ such that for every state s: Player 0 wins the game from some designated vertex iff $(\mathcal{K}, s, \mathcal{V}) \models \varphi$. The construction is an adaptation of the one for L$_\mu$, which is based on a transformation of the model checking problem for L$_\mu$ into the emptiness problem for parity tree automata [5].

In what follows, we fix a finite Kripke structure $\mathcal{K} = (S, \Sigma, R, L)$ and an SL$_\mu$-formula φ over Σ, both over the set Prop of predicates symbols. Further, we assume that $\lambda \notin \Sigma$. Let $\mathcal{V} : \mathrm{Var}(\varphi) \to 2^S$ be a valuation for φ.

The structure $\mathcal{K}_\varphi = (S_\varphi, \Sigma \cup \{\lambda\}, R_\varphi, \emptyset)$ is induced by the structure of φ: First, there is a state for each subformula in $\mathrm{Cl}(\varphi)$. For simplicity, we name the states after subformulae. Second, there are two additional states s_X and r_X for each $X \in \mathrm{Bd}(\varphi)$. Third, we add an extra state q_a for each $a \in \Sigma$. The latter states are needed for deletion purposes and are independent of the structure of φ. Let $\mathrm{init} : \mathrm{Cl}(\varphi) \to S_\varphi$ be defined by $\mathrm{init}(\psi) := r_X$ if $\psi = X$ and $X \in \mathrm{Bd}(\varphi)$, $\mathrm{init}(\psi) := s_X$ if $\psi = \mu X.\psi'$ or $\psi = \nu X.\psi'$, and $\mathrm{init}(\psi) := \psi$ otherwise.

We define R_φ by giving a list of transitions for each type of subformula. Let $\psi \in \mathrm{Cl}(\varphi)$. (1) The states $\top$, $\bot$, p, $\neg p$, and X for $X \in \mathrm{Var}(\varphi) \setminus \mathrm{Bd}(\varphi)$ are sinks; (2) if $\psi = \psi_1 \vee \psi_2$ or $\psi = \psi_1 \wedge \psi_2$, then there are the transitions $\psi \xrightarrow{\lambda} \mathrm{init}(\psi_1)$ and $\psi \xrightarrow{\lambda} \mathrm{init}(\psi_2)$; (3) if $\psi = \Diamond_a\psi'$ or $\psi = \Box_a\psi'$ for $a \in \Sigma$, then there is the transition $\psi \xrightarrow{a} \mathrm{init}(\psi')$; (4) if $\psi = \diamond\!\!\!\cdot_a\psi'$ or $\psi = \boxdot_a\psi'$ for $a \in \Sigma$, then there is the transition $\psi \xrightarrow{\lambda} \mathrm{init}(\psi')$; (5) if $\psi = \mu X.\psi'$ or $\psi = \nu X.\psi'$, then there are the transitions $s_X \xrightarrow{\lambda} \psi$, $\psi \xrightarrow{\lambda} \mathrm{init}(\psi')$, $r_X \xrightarrow{\lambda} X$, and $X \xrightarrow{\lambda} \psi$; (6) there is a transition $q_a \xrightarrow{a} q_a$ for every $a \in \Sigma$.

Let $\mathcal{K} \otimes \mathcal{K}_\varphi$ be the synchronized product of $\mathcal{K}$ and $\mathcal{K}_\varphi$ where predicates are ignored: for $a \in \Sigma$, we have $(s, t) \xrightarrow{a} (s', t')$ iff $s \xrightarrow{a} s'$ in $\mathcal{K}$ and $t \xrightarrow{a} t'$ in $\mathcal{K}_\varphi$ as well as $(s,t) \xrightarrow{\lambda} (s',t')$ iff $t \xrightarrow{\lambda} t'$ in $\mathcal{K}_\varphi$. Let $G = (V,E)$ be the transition graph of $\mathcal{K} \otimes \mathcal{K}_\varphi$ without transition labels. Let m be the fixed-point depth of φ. The game $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ has then m registers. We start with the declaration of the vertices in V as movement, deletion, storing, or restoring vertices. Each (s,q_a) for $s \in S$ and $a \in \Sigma$ belongs to M_0. Let $s \in S$, $\psi \in \mathrm{Cl}(\varphi)$, and $a \in \Sigma$. Then $(s,\psi) \in M_0$ if (1) $\psi = \bot$, $\psi = \psi_1 \vee \psi_2$, $\psi = \Diamond_a\psi'$, or $\psi = \nu X.\psi'$, or (2) $\psi = p$ and $p \notin L(s)$, or $\psi = \neg p$ and $p \in L(s)$, or (3) $\psi = X$ and $s \notin \mathcal{V}(X)$. The movement vertices M_1 of Player 1 are defined dually. We set $(s,\psi) \in D_0$ if $\psi = \Diamonddot_a\psi'$ and $(s,\psi) \in D_1$ if $\psi = \boxdot_a\psi'$. Finally, let $\mathrm{fh}_\varphi(X)$ to be the maximum number of nested fixed-point operators in $\mathrm{Df}_\varphi(X)$ for $X \in \mathrm{Bd}(\varphi)$. Then we set $(s,s_X) \in S_{\mathrm{fh}_\varphi(X)}$ and $(s,r_X) \in R_{\mathrm{fh}_\varphi(X)}$ for $X \in \mathrm{Bd}(\varphi)$.

Next, we define the deletion relation Δ. For $t,t' \in S$ and $a \in \Sigma$, let $\Xi^\varphi_{t,a,t'}$ be the following set:

$$\{((t,\psi),(t',\mathrm{init}(\psi'))) \mid \psi \in \mathrm{Cl}(\varphi) \wedge (\psi = \Diamond_a\psi' \text{ or } \Box_a\psi')\} \cup \{((t,q_a),(t',q_a))\}.$$

Note that, if $(t,a,t') \in R$, then we have $\Xi^\varphi_{t,a,t'} \subseteq \mathrm{Out}(M)$. By the synchronized product, we have $((t,q_a),(t',q_a)) \in E$ iff $(t,a,t') \in R$. Thus, $\Xi^\varphi_{t,a,t'}$ is non-empty for $(t,a,t') \in R$, regardless of the structure of φ. For $a \in \Sigma$ and $\psi \in \mathrm{Cl}(\varphi)$ with $\psi = \Diamonddot_a\psi'$ or $\psi = \boxdot_a\psi'$, we set for every $s \in S$: $((s,\psi),\Xi^\varphi_{t,a,t'}) \in \Delta$ iff $t,t' \in S \wedge (t,a,t') \in R$.

We conclude the definition of the arena $\mathcal{A}_{\mathcal{K},\varphi,\mathcal{V}}$ by specifying the priority function $\Omega : V \to \mathbb{N}$. We first define a function Ω' for $\mathcal{K}_\varphi$ and then we extend Ω' to V. For $X \in \mathrm{Bd}(\varphi)$, let $B_\varphi(X) := \mathrm{Bd}(\mathrm{Df}_\varphi(X)) \setminus \{X\} \subseteq \mathrm{Var}(\varphi)$. Then we define $\Omega' : S_\varphi \to \mathbb{N}$ by $\Omega'(s) := 0$ for every $s \in S_\varphi \setminus \mathrm{Bd}(\varphi)$, and $\Omega'(X) := \min\{c \in \mathbb{N} \mid c \text{ odd} \wedge c > \max\{0, \{\Omega'(Y) \mid Y \in B_\varphi(X)\}\}\}$ if X is a μ-variable of φ, and $\Omega'(X) := \min\{c \in \mathbb{N} \mid c \text{ even} \wedge c > \max\{0, \{\Omega'(Y) \mid Y \in B_\varphi(X)\}\}\}$ if X is a ν-variable of φ.

We set $\Omega((s,\psi)) = \Omega'(\psi)$ for every $s \in S$ and $\psi \in S_\varphi$. Finally, let $n := \max\{\Omega'(X) \mid X \in \mathrm{Bd}(\varphi)\}$. This concludes the definition of the arena $\mathcal{A}_{\mathcal{K},\varphi,\mathcal{V}} = (V,E,\Delta,\Omega)$ and the (n,m)-backup game $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$. Note that n and m depend on φ only. Regarding the size of the game, it is straightforward to check that $|\mathcal{A}_{\mathcal{K},\varphi,\mathcal{V}}|$ is quadratic with respect to $|\mathcal{K}| \cdot |\varphi|$.

Before we show that $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ indeed serves as a model checking game for SL_μ, we observe that the initial content of registers has no effect on plays starting 'at the top' of the game.

Lemma 3. *With the same notation as before, one has for each state $s \in S$ and for each $X_1 \dots X_m \in 2^E$ that Player τ wins $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ from position $((s,\mathrm{init}(\varphi)),E,X_1 \dots X_m)$ iff he wins $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ from position $((s,\mathrm{init}(\varphi)),E,E \dots E)$.* $\Box$

We need two auxiliary results concerning backup games. The first one deals with winning regions of subgames. The second one provides an unfolding of the parity condition, which we need for the fixed-point operators. The unfolding is a classical construction based on Knaster-Tarski.

Lemma 4. *Let $\mathcal{A} = (V,E,\Delta,\Omega)$ be the arena of an (n,m)-backup game $\mathcal{G}$ with $V = (M_\tau,D_\tau,S_j,R_j)$. Suppose that $V' \subseteq V$ such that $V'E \subseteq V'$ and that* *for each $v \in V'$, if $(v, \Xi) \in \Delta$ and $\Xi \neq \emptyset$, then $\Xi \cap (V' \times V') \neq \emptyset$. Let $u \in V'$ and $X_0 \dots X_m \subseteq E$ be fixed. We define $V' = (M_\tau \cap V', D_\tau \cap V', S_j \cap V', R_j \cap V')$, $E' := E \cap (V' \times V')$, $X'_i := X_i \cap E'$ for each $i \in [0, m]$, and $\Delta' := \{(v, \Xi \cap E') \mid v \in V' \wedge (v, \Xi) \in \Delta\}$. Let $\mathcal{G}'$ be the (n, m)-backup game with arena $\mathcal{A}' = (V', E', \Delta', \Omega|_{V'})$. Then Player τ wins $\mathcal{G}$ from $(u, X_0 \dots X_m)$ iff he wins $\mathcal{G}'$ from $(u, X'_0 \dots X'_m)$.* □

We turn to the unfolding of the parity condition. Let $\mathcal{A} = (V, E, \Delta, \Omega)$ be the arena of an (n, m)-backup game $\mathcal{G}$ with $V = (M_\tau, D_\tau, S_j, R_j)$. Let $\Omega_{\max} := \max_{v \in V} \Omega(v)$, $T := \Omega^{-1}(\Omega_{\max})$, $U := TE$ and $\kappa := |T| + 1$. We make the following assumptions: (1) $\Omega_{\max}$ is even; (2) $T \subseteq M$; (3) every $v \in T$ has a unique successor $\mathrm{scc}(v)$; (4) for every $u \in U$ and every play π that starts from $(u, E \dots E)$, if $\pi_i = (v, X_0 \dots X_m)$ for some $v \in T$, then $X_j = E$ for every $j \in [0, m]$; (5) if $(v, \Xi) \in \Delta$ for some $v \in V$, then $\Xi \cap (T \times U) = \emptyset$.

Under this assumptions, the *unfolding* of $\mathcal{G}$ is a sequence of (n, m)-backup games $\mathcal{G}^i$ for $i \in [0, \kappa]$. Let $E^- := E \setminus (T \times U)$ and $\mathcal{A}^- := (V, E^-, \Delta, \Omega)$. Note that the vertices in T become terminal in $\mathcal{A}^-$. The arena of $\mathcal{G}^i$ coincides with $\mathcal{A}^-$ up to the winning condition for T. For every $i \in [0, \kappa]$, we define a decomposition $T = T^i_0 \cup T^i_1$ and declare Player τ to be the winner of the game $\mathcal{G}^i$ when a play reaches $v \in T^i_\tau$. Let $W^i_\tau \subseteq V \times (2^{E^-})^{m+1}$ be the winning region of Player τ in the game $\mathcal{G}^i$. Clearly, W^i_τ depends on the decomposition $T = T^i_0 \cup T^i_1$. In turn, the decomposition of T for $i + 1$ depends on W^i_τ: We define $T^0_0 := T$ and $T^{i+1}_0 := \{v \in T \mid (\mathrm{scc}(v), E^- \dots E^-) \in W^i_0\}$.

It is easy to check that $T^0_1 \subseteq T^1_1 \subseteq \dots \subseteq T^\kappa_1$ and $W^0_1 \subseteq W^1_1 \subseteq \dots \subseteq W^\kappa_1$. By determinacy, we also have $T^0_0 \supseteq T^1_0 \supseteq \dots \supseteq T^\kappa_0$ and $W^0_0 \supseteq W^1_0 \supseteq \dots \supseteq W^\kappa_0$. Since $\kappa = |T| + 1$ and $T^i_1 \subseteq T$ for each $i \in [0, \kappa]$, there exists $\alpha < \kappa$ such that $T^\alpha_1 = T^{\alpha+1}_1$ (and then also $T^\alpha_0 = T^{\alpha+1}_0$, $W^\alpha_\tau = W^{\alpha+1}_\tau$). We claim that we can determine the winner of a play in the original game $\mathcal{G}$ that starts from a vertex $u \in U$ by considering this fixed-point of winning regions for the unfolding of $\mathcal{G}$:

Lemma 5. *Let $\mathcal{G}$, U, and κ be as before and let $\mathcal{G}^0 \dots \mathcal{G}^\kappa$ be the unfolding of $\mathcal{G}$. Then for every $u \in U$, Player τ wins $\mathcal{G}$ from $(u, E \dots E)$ iff he wins $\mathcal{G}^\kappa$ from $(u, E^- \dots E^-)$.* □

If $\mathcal{G}$ is as before, but $\Omega_{\max}$ is odd, then we can proceed to the dual game where the roles of the players are swapped and the priority function is increased by one. We are now prepared to prove the main result of this section:

Theorem 4. *Suppose that $\mathcal{K} = (S, \Sigma, R, L)$ is a finite Kripke structure with state $s \in S$, φ is an SL_μ-formula over Σ, and $\mathcal{V} : \mathrm{Var}(\varphi) \to 2^S$ is a valuation. Then $(\mathcal{K}, s, \mathcal{V}) \models \varphi$ iff Player 0 wins $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ from $((s, \mathrm{init}(\varphi)), E \dots E)$, where E is the edge relation of the arena of $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$.*

Proof (Sketch). The proof is by induction on the structure of φ. We only present some interesting cases. Case $\varphi = \Diamond_a \psi$. Let $\mathcal{A} = (V, E, \Delta, \Omega)$ be the arena of $\mathcal{G} := \mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ and $\mathcal{A}' = (V', E', \Delta', \Omega')$ be the arena of $\mathcal{G}' := \mathcal{G}_{\mathcal{K},\psi,\mathcal{V}}$. It is easy to see that $\mathcal{G}'$ is a subgame of $\mathcal{G}$ that meets the requirements of Lemma 4. It follows that for each $t \in S$, Player 0 wins $\mathcal{G}$ from $((t, \mathrm{init}(\psi)), E \dots E)$ iff he wins $\mathcal{G}'$ from $((t, \mathrm{init}(\psi)), E' \dots E')$. We have $\mathrm{init}(\varphi) = \varphi$ and $(s, \varphi) \in M_0$. By definition of $\mathcal{A}$, there is an edge from (s, φ) to $(t, \mathrm{init}(\psi))$ iff $(s, a, t) \in R$. Thus

$$\begin{aligned}
& \text{Player 0 wins } \mathcal{G} \text{ from } ((s, \varphi), E \dots E) \\
\iff & \exists t \in S : (s, a, t) \in R \text{ and Player 0 wins } \mathcal{G} \text{ from } ((t, \mathrm{init}(\psi)), E \dots E) \\
\iff & \exists t \in S : (s, a, t) \in R \text{ and Player 0 wins } \mathcal{G}' \text{ from } ((t, \mathrm{init}(\psi)), E' \dots E') \\
\iff & \exists t \in S : (s, a, t) \in R \text{ and } (\mathcal{K}, t, \mathcal{V}) \models \psi \quad \text{[by induction]} \\
\iff & (\mathcal{K}, s, \mathcal{V}) \models \varphi.
\end{aligned}$$

Case $\varphi = \diamondsuit_a \psi$. Let $\mathcal{A} = (V, E, \Delta, \Omega)$ be the arena of $\mathcal{G} := \mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$. We have $\mathrm{init}(\varphi) = \varphi$, $(s, \varphi) \in D_0$, and (s, φ) has the unique E-successor $(s, \mathrm{init}(\psi))$. By definition, there is $((s, \varphi), \Xi) \in \Delta$ with $\emptyset \neq \Xi \subseteq E$ iff there are $t, t' \in S$ with $(t, a, t') \in R$ and $\Xi = \Xi^{\varphi}_{t,a,t'}$. Thus, it follows that Player 0 wins $\mathcal{G}$ from $((s, \varphi), E \dots E)$ iff there are $t, t' \in S$ with $(t, a, t') \in R$ such that Player 0 wins $\mathcal{G}$ from $((s, \mathrm{init}(\psi)), E \setminus \Xi^{\varphi}_{t,a,t'}, E \dots E)$. Let $E^- := E \setminus \Xi^{\varphi}_{t,a,t'}$. By Lemma 3, we have that Player 0 wins $\mathcal{G}$ from $((s, \mathrm{init}(\psi)), E^-, E \dots E)$ iff he wins $\mathcal{G}$ from $((s, \mathrm{init}(\psi)), E^-, E^- \dots E^-)$.

The arena (V, E^-, Δ, Ω) is identical with the arena $\mathcal{A}' := (V', E', \Delta', \Omega')$ of the game $\mathcal{G}' := \mathcal{G}_{\mathcal{K} \setminus \{(t,a,t')\},\varphi,\mathcal{V}}$ up to the deletion relation Δ'. In particular, we have $E^- = E'$. But for any play in $\mathcal{G}$ starting at $((s, \mathrm{init}(\psi)), E^- \dots E^-)$, neither player can choose $\Xi^{\varphi}_{t,a,t'}$ at deletion vertices without losing immediately. It follows that Player 0 wins $\mathcal{G}$ from $((s, \mathrm{init}(\psi)), E^- \dots E^-)$ iff he wins $\mathcal{G}'$ from $((s, \mathrm{init}(\psi)), E' \dots E')$. Finally, let $\mathcal{G}'' := \mathcal{G}_{\mathcal{K} \setminus \{(t,a,t')\},\psi,\mathcal{V}}$ with arena $\mathcal{A}'' = (V'', E'', \Delta'', \Omega'')$. Again, $\mathcal{G}''$ is a subgame of $\mathcal{G}'$ that contains the state $(s, \mathrm{init}(\psi))$ and that meets the requirements of Lemma 4. It follows that Player 0 wins $\mathcal{G}'$ from $((s, \mathrm{init}(\psi)), E' \dots E')$ iff he wins $\mathcal{G}''$ from $((s, \mathrm{init}(\psi)), E'' \dots E'')$. By induction, this is equivalent to $(\mathcal{K} \setminus \{(t, a, t')\}, s, \mathcal{V}) \models \psi$. Together, we get that Player 0 wins $\mathcal{G}$ from $((s, \varphi), E \dots E)$ iff there exists $t, t' \in S$ with $(t, a, t') \in R$ and $(\mathcal{K} \setminus \{(t, a, t')\}, s, \mathcal{V}) \models \psi$. The latter is equivalent to $(\mathcal{K}, s, \mathcal{V}) \models \varphi$.

Case $\varphi = \nu X.\psi$. Let $\mathcal{A} = (V, E, \Delta, \Omega)$ be the arena of the game $\mathcal{G} := \mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ with $V = (M_\tau, D_\tau, S_j, R_j)$. By definition, we have that $\Omega_{\max} = \Omega'(X)$ is even. Let $T := \Omega^{-1}(\Omega_{\max})$. By definition of $\mathcal{A}$, we have $T = \{(t, X) \mid t \in S\} \subseteq M$, $|(t, X)E| = 1$ for each $t \in S$, and $U := TE = \{(t, \varphi) \mid t \in S\}$. Let $\kappa := |T| + 1 = |S| + 1$. We have $\mathrm{fh}_\varphi(X) = m$ and thus, register $\mathrm{fh}_\varphi(X)$ is the highest register of $\mathcal{G}$. Further, we have $S_m = \{(t, \mathrm{init}(\varphi)) \mid t \in S\}$ and each of these vertices has a single E-successor, namely (t, φ), and no incoming E-edge. In particular, Player 0 wins $\mathcal{G}$ from $((s, \mathrm{init}(\varphi)), E \dots E)$ iff he wins $\mathcal{G}$ from $((s, \varphi), E \dots E)$. Every vertex (t, X) occurs in combination with the restoring vertex (t, r_X). Let $t, t' \in S$ and π be a play that starts from $((t, \varphi), E \dots E)$ and that reaches some $\pi_i = ((t', X), X_0 \dots X_m)$. Note that no storing vertex from S_m can occur in π. Thus, we have $X_m = E$ and π_{i-1} visits vertex (t', r_X). Be definition of the restoring process, it follows that $X_j = E$ for every $j \in [0, m]$. Finally, the edges $(t, X) \to (t, \varphi)$ for $t \in S$ do not occur in Δ. It follows that $\mathcal{G}$ meets the requirements of Lemma 5.

Before we proceed, we fix some notation. Let $\mathcal{W} : \mathrm{Var}(\varphi) \to 2^S$ be a valuation of φ. Note that the definitions of the edge relation, the deletion relation, and the priority function of a game $\mathcal{G}_{\mathcal{K},\chi,\mathcal{W}}$ do not depend on the valuation $\mathcal{W}$. Let $\mathcal{A}_\mathcal{W} = (V_\mathcal{W}, E, \Delta, \Omega)$ be the arena of $\mathcal{G}_\mathcal{W} := \mathcal{G}_{\mathcal{K},\varphi,\mathcal{W}}$. The game $\mathcal{G}_\mathcal{W}$ is identical to $\mathcal{G}$ up to the assignment of vertices (t, Y) for $t \in S, Y \in \mathrm{Var}(\varphi)$ to one of the players. Let $E^- := E \setminus (T \times U)$ and $\mathcal{G}_\mathcal{W}^-$ be the game with arena $(V_\mathcal{W}, E^-, \Delta, \Omega)$. Finally, we define the game $\mathcal{G}_\mathcal{W}^* := \mathcal{G}_{\mathcal{K},\psi,\mathcal{W}}$ with arena $\mathcal{A}_\mathcal{W}^* = (V_\mathcal{W}^*, E^*, \Delta^*, \Omega^*)$. The variable X occurs free in ψ. Thus, the vertices (t, X) for $t \in S$ have no outgoing edges in $\mathcal{A}_\mathcal{W}^*$.

The game $\mathcal{G}_\mathcal{W}^*$ is a subgame of $\mathcal{G}_\mathcal{W}^-$ that contains the vertices $(t, \mathrm{init}(\psi))$ for every $t \in S$ and that meets the requirements of Lemma 4. In $\mathcal{A}_\mathcal{W}^-$, each vertex (t, φ) for $t \in S$ has the unique successor $(t, \mathrm{init}(\psi))$. Together with the induction hypothesis, it follows for every $t \in S$ that Player 0 wins $\mathcal{G}_\mathcal{W}^-$ from $((t, \varphi), E^- \dots E^-)$ iff he wins $\mathcal{G}_\mathcal{W}^-$ from $((t, \mathrm{init}(\psi)), E^- \dots E^-)$ iff he wins $\mathcal{G}_\mathcal{W}^*$ from $((t, \mathrm{init}(\psi)), E^* \dots E^*)$ iff $(\mathcal{K}, t, \mathcal{W}) \models \psi$.

For $i \in [0, \kappa + 1]$, let $F_0 := S$ and $F_{i+1} := \|\psi\|^\mathcal{K}_{\mathcal{V}[X:=F_i]}$. Since $\kappa = |S| + 1$, it follows by Knaster-Tarski that $\|\varphi\|^\mathcal{K}_\mathcal{V} = F_{\kappa+1}$. In particular, $(\mathcal{K}, s, \mathcal{V}) \models \varphi$ iff $(\mathcal{K}, s, \mathcal{V}[X := F_\kappa]) \models \psi$. Let $\mathcal{G}^0 \dots \mathcal{G}^\kappa$ be the unfolding of $\mathcal{G}$. It is straightforward to check by induction on i that $\mathcal{G}^i$ is identical to $\mathcal{G}^-_{\mathcal{V}[X:=F_i]}$ for each $i \in [0, \kappa]$. By Lemma 5, we therefore obtain that Player 0 wins $\mathcal{G}$ from $((s, \varphi), E \dots E)$ iff he wins $\mathcal{G}^\kappa$ from $((s, \varphi), E^- \dots E^-)$. We can now conclude the case of greatest fixed-points: Player 0 wins $\mathcal{G}$ from $((s, \mathrm{init}(\varphi)), E \dots E)$ iff he wins $\mathcal{G}$ from $((s, \varphi), E \dots E)$ iff he wins $\mathcal{G}^\kappa$ from $((s, \varphi), E^- \dots E^-)$ iff he wins $\mathcal{G}^-_{\mathcal{V}[X:=F_\kappa]}$ from $((s, \varphi), E^- \dots E^-)$ iff $(\mathcal{K}, s, \mathcal{V}[X := F_\kappa]) \models \psi$ iff $(\mathcal{K}, s, \mathcal{V}) \models \varphi$.

Note that the cases $\varphi = \varphi_1 \wedge \varphi_2$, $\varphi = \Box_a \psi$, $\varphi = \boxdot_a \psi$, and $\varphi = \mu X.\psi$ are dual to the cases $\varphi = \varphi_1 \vee \varphi_2$, $\varphi = \Diamond_a \psi$, $\varphi = \diamonddot_a \psi$, and $\varphi = \nu X.\psi$. □

5 Conclusion

We augmented the μ-calculus with a transition-deleting modality, which yields the fixed-point logic SL_μ over dynamically changing structures. We have seen that model checking is not algorithmically harder than model checking the sabotage modal logic without fixed-points. We introduced backup games as extended parity games with the feature of edge deletion and of storing and restoring the current arena in a fixed number of registers. Even when the access to registers has to follow a stack discipline, these games serve as model checking games for SL_μ. The problem of solving these games is PSPACE-complete. The games without limited access become EXPTIME-complete.

Model checking for SL_μ via backup games is not optimal yet: We could only show that the problem of solving backup games belongs to PSPACE for a fixed number of registers. But the number of registers of the game $\mathcal{G}_{\mathcal{K},\varphi,\mathcal{V}}$ is equal to the fixed-point depth of φ. Our result yields, so far, only a PSPACE-procedure for formulae with bounded fixed-point depth. Note that we already know that the model checking with a fixed formula can be done in polynomial time with respect to the size of the structure, cf. Lemma 1. It remains an open question whether backup games can be solved in polynomial space, regardless of the number of registers. If we can answer this question positively, then we would obtain an optimal model checking procedure for SL_μ. Otherwise, there may be a subclass of games that serve as model checking games, but which can be solved in polynomial space regardless of the number of registers. Finally, it remains open whether we can express the winning condition for backup games as SL_μ-formulae. Recall that this is the case for standard parity games and L_μ [14]. If such a translation yields SL_μ-formulae with a size polynomial in the size of the arena, then this would also give a positive answer to the above question.

Note that SL_μ does not provide a way to express, for example, a general reachability *while* transitions are deleted (due to the restoration in inductive fixed-point constructions). It is worth to study logics that allow such an overlap of fixed-points and deletion. Finally, sabotage logics that allow to address the deletion of objects are only a first step towards a general theory of logics over dynamically changing structures.

References

1. Balcázar, J.L., Díaz, J., Gabarró, J.: Structural Complexity II. Springer 1990
2. van Benthem, J.: An essay on sabotage and obstruction. In: Mechanizing Mathematical Reasoning. LNAI 2605 (2005), 268–276
3. Bernholtz, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. In: CAV '94. LNCS 818 (1994), 142–155
4. Emerson, E.A., Jutla, C.S.: Tree automata, mu-calculus and determinacy. In: FOCS '91 (1991), 368–377
5. Emerson, E.A., Jutla, C.S., Sistla, A.P.: On model-checking for fragments of μ-calculus. In: CAV '93. LNCS 697 (1993), 385–396
6. Jurdziński, M.: Small progress measures for solving parity games. In: STACS '00. LNCS 1770 (2000), 290–301
7. Kozen, D.: Results on the propositional μ-calculus. TCS **27** (1983), 333–354
8. Löding, Ch., Rohde, Ph.: Solving the sabotage game is PSPACE-hard. In: MFCS '03. LNCS 2747 (2003), 531–540
9. Löding, Ch., Rohde, Ph.: Model checking and satisfiability for sabotage modal logic. In: FSTTCS '03. LNCS 2914 (2003), 302–313
10. Rohde, Ph.: On Games and Logics over Dynamically Changing Structures. Technical report submitted as dissertation thesis at RWTH Aachen (2005). Available under `www-i7.informatik.rwth-aachen.de/~rohde/thesis.pdf`
11. Stockmeyer, L.J., Chandra, A.K.: Provably difficult combinatorial games. SIAM Journal on Computing **8** (1979), 151–174
12. Vardi, M.Y.: The complexity of relational query languages. In: STOC '83 (1982), 137–146
13. Vardi, M.Y.: On the complexity of bounded-variable queries. In: PODS '95 (1995), 266–276
14. Walukiewicz, I.: Monadic second-order logic on tree-like structures. TCS **275** (2002), 311–346