Model-Checking Games for Fixpoint Logics with Partial Order Models

Julian Gutierrez and Julian Bradfield

LFCS. School of Informatics. University of Edinburgh
Informatics Forum, 10 Crichton Street, Edinburgh, EH8 9AB, UK

Abstract. We introduce model-checking games that allow *local* second-order power on *sets* of independent transitions in the underlying partial order models where the games are played. Since the one-step interleaving semantics of such models is not considered, some problems that may arise when using interleaving semantics are avoided and new decidability results for partial orders are achieved. The games are shown to be *sound* and *complete*, and therefore determined. While in the interleaving case they coincide with the local model-checking games for the μ-calculus ($L\mu$), in a noninterleaving setting they verify properties of Separation Fixpoint Logic (SFL), a logic that can specify in partial orders properties not expressible with $L\mu$. The games underpin a novel decision procedure for model-checking all temporal properties of a class of infinite and regular event structures, thus improving previous results in the literature.

Keywords: Modal and temporal logics; Model-checking games; Hintikka game semantics; Partial order models of concurrency; Process algebras.

1 Introduction

Model-checking games, also called Hintikka evaluation games, are played by two players, a "Verifier" Eve ($\exists$) and a "Falsifier" Adam ($\forall$). These *logic games* are played in a formula ϕ and a mathematical model $\mathfrak{M}$. In a game $\mathcal{G}(\mathfrak{M}, \phi)$ the goal of Eve is to show that $\mathfrak{M} \models \phi$, while the goal of Adam is to refute such an assertion. Solving these games amounts to answering the question of whether or not Eve has a strategy to win all plays in the game $\mathcal{G}(\mathfrak{M}, \phi)$. These games have a long history in mathematical logic and in the last two decades have become an active area of research in computer science, both from theoretical and practical view points. Good introductions to the subject can be found in [2, 10].

In concurrency and program verification, most usually ϕ is a modal or a temporal formula and $\mathfrak{M}$ is a Kripke structure or a labelled transition system (LTS), i.e., a graph structure, and the two players play the game $\mathcal{G}(\mathfrak{M}, \phi)$ *globally* by picking single elements of $\mathfrak{M}$, according to the game rules defined by ϕ. This setting works well for concurrent systems with *interleaving* semantics since one always has a notion of global state enforced by the (nondeterministic) sequential computation of atomic actions, which in turn allows the players to choose only

single elements of the structure $\mathfrak{M}$. However, when considering concurrent systems with partial order models explicit notions of *locality* and *concurrency* have to be taken into account. A possible solution to this problem – the traditional approach – is to use the one-step interleaving semantics of such models in order to recover the *globality* and *sequentiality* of the semantics of formulae.

This solution is, however, problematic for at least five reasons. Firstly, interleaving models usually suffer from the state space explosion problem. Secondly, interleaving interpretations cannot be used to give completely satisfactory game semantics to logics with partial order models as all information on independence in the models is lost in the interleaving simplification. Thirdly, although temporal properties can still be verified with the interleaving simplification, properties involving concurrency, causality and conflict, natural to partial order models of concurrency, can no longer be verified. From a more practical standpoint, partial order reduction methods cannot be applied directly to interleaving models in order to build less complex model checkers based on these techniques. Finally, the usual techniques for verifying interleaving models cannot always be used to verify partial order ones since such problems may become undecidable.

For these reasons, we believe that the study of verification techniques for partial order models continues to deserve much attention since they can help alleviate some of the limitations related with the use of interleaving models. We, therefore, abandon the traditional approach to defining model-checking games for logics with partial order models and introduce a new class of games called 'trace local monadic second-order (LMSO) model-checking games', where *sets* of independent elements of the structure at hand can be *locally* recognised. These games avoid the need of using the one-step interleaving semantics of partial order models, and thus define a more natural framework for analysing fixpoint modal logics with noninterleaving semantics. As a matter of fact, their use in the temporal verification of a class of regular event structures [11] improves previous results in the literature [5, 7]. We do so by allowing a free interplay of fixpoint operators and local second-order power on *conflict-free* sets of transitions.

The logic we consider is Separation Fixpoint Logic (SFL) [3], a μ-calculus ($L\mu$) extension that can express causal properties in partial order models (e.g., transition systems with independence, Petri nets or event structures), and allows for doing *dynamic* local reasoning. The notion of locality in SFL, namely separation or disjointness of independent sets of resources, was inspired by the one defined *statically* for Separation Logic [8]. Since SFL is as expressive as $L\mu$ in an interleaving context, nothing is lost with respect to the main approaches to logics for concurrency with interleaving semantics. Instead, logics and techniques for interleaving concurrency are extended to a partial order setting with SFL.

Section 2 contains some background concepts and definitions. In Section 3, trace LMSO model-checking games are defined, and in Section 4 their soundness and completeness is proved. In Section 5, we show that they are decidable and their coincidence with the local model-checking games for $L\mu$ in the interleaving case. In Section 6 the game is used to effectively model-check a class of regular and infinite event structures. The paper concludes with Section 7.

2 Preliminaries

2.1 A Partial Order Model of Concurrency

A *transition system with independence* (TSI) [6] is an LTS where independent transitions can be recognised. Formally, a TSI $\mathfrak{T}$ is a structure (S, s_0, T, Σ, I), where S is a set of states with initial state s_0, $T \subseteq S \times \Sigma \times S$ is a transition relation, Σ is a set of labels, and $I \subseteq T \times T$ is an irreflexive and symmetric relation on independent transitions. The binary relation $\prec$ on transitions defined by

$$(s, a, s_1) \prec (s_2, a, q) \Leftrightarrow$$
$$\exists b.(s, a, s_1)I(s, b, s_2) \wedge (s, a, s_1)I(s_1, b, q) \wedge (s, b, s_2)I(s_2, a, q)$$

expresses that two transitions are instances of the same *action*, but in two different interleavings. We let $\sim$ be the least equivalence relation that includes $\prec$, i.e., the reflexive, symmetric and transitive closure of $\prec$. The equivalence relation $\sim$ is used to group all transitions that are instances of the same action in all its possible interleavings. Additionally, I is subject to the following axioms:

- **A1**. $(s, a, s_1) \sim (s, a, s_2) \Rightarrow s_1 = s_2$
- **A2**. $(s, a, s_1)I(s, b, s_2) \Rightarrow \exists q.(s, a, s_1)I(s_1, b, q) \wedge (s, b, s_2)I(s_2, a, q)$
- **A3**. $(s, a, s_1)I(s_1, b, q) \Rightarrow \exists s_2.(s, a, s_1)I(s, b, s_2) \wedge (s, b, s_2)I(s_2, a, q)$
- **A4**. $(s, a, s_1) \prec \cup \succ (s_2, a, q)I(w, b, w') \Rightarrow (s, a, s_1)I(w, b, w')$

Axiom **A1** states that from any state, the execution of an action leads always to a unique state. This is a determinacy condition. Axioms **A2** and **A3** ensure that independent transitions can be executed in either order. Finally, **A4** ensures that the relation I is well defined. More precisely, **A4** says that if two transitions t and t' are independent, then all other transitions in the equivalence class $[t]_\sim$ (i.e., all other transitions that are instances of the same action but in different interleavings) are independent of t' as well, and vice versa.

Local Dualities. Based on the previous axiomatization one can define two ways of observing *concurrency* such that in each case they are dual to *causality* and *conflict*, respectively [3]. The semantics of SFL is based on these dualities. Given a state s and a transition $t = (s, a, s')$, s is called the source node, $src(t) = s$, and s' the target node, $trg(t) = s'$. Thus, define the following relations on transitions:

$$\otimes \stackrel{\text{def}}{=} \{(t_1, t_2) \in T \times T \mid src(t_1) = src(t_2) \wedge t_1 \; I \; t_2\}$$
$$\# \stackrel{\text{def}}{=} \{(t_1, t_2) \in T \times T \mid src(t_1) = src(t_2) \wedge \neg(t_1 \; I \; t_2)\}$$
$$\ominus \stackrel{\text{def}}{=} \{(t_1, t_2) \in T \times T \mid trg(t_1) = src(t_2) \wedge t_1 \; I \; t_2\}$$
$$\leq \stackrel{\text{def}}{=} \{(t_1, t_2) \in T \times T \mid trg(t_1) = src(t_2) \wedge \neg(t_1 \; I \; t_2)\}$$

2.2 Sets in a Local Context

The relation $\otimes$ defined on pairs of transitions, can be used to recognize *sets* where every transition is independent of each other and hence can all be executed concurrently. Such sets are said to be *conflict-free* and belong to the same *trace*.

Definition 1 A *conflict-free set of transitions* P is a set of transitions with the same source node, where $t \otimes t'$ for each two elements t and t' in P.

Given a TSI $\mathfrak{T} = (S, s_0, T, \Sigma, I)$, all conflict-free sets of transitions at a state $s \in S$ can be defined locally from the *maximal set* of transitions $R_{\max}(s)$, where $R_{\max}(s)$ is the set of all transitions $t \in T$ such that $src(t) = s$. Hence all maximal sets and conflict-free sets of transitions are fixed given a particular TSI. Now we define the notion of locality used to give the semantics of SFL.

Definition 2 Given a TSI $\mathfrak{T}$, a *support set* R in $\mathfrak{T}$ is either a maximal set of transitions in $\mathfrak{T}$ or a non-empty conflict-free set of transitions in $\mathfrak{T}$.

Given a TSI, the set of all its support sets is denoted by $\mathfrak{P}$. The next definition is useful when defining the semantics of SFL: $R_1 \uplus R_2 \sqsubseteq R \stackrel{\text{def}}{=} R_1 \cup R_2 \subseteq R$, and R_1 and R_2 are two disjoint non-empty conflict-free support sets, and $\neg\exists t \in R \setminus (R_1 \cup R_2).\ \forall t' \in R_1 \cup R_2.\ t \otimes t'$.

Notice that the last definition characterizes support sets R_1 and R_2 that contain only concurrent transitions and cannot be made any bigger with respect to the support set R. Therefore, the sets $R_1 \uplus R_2$ with the properties above are the biggest *traces* that can be recognized from a support set R.

2.3 Separation Fixpoint Logic

Definition 3 *Separation Fixpoint Logic* (SFL) [3] has formulae ϕ built from a set Var of variables $Y, Z, ...$ and a set $\mathcal{L}$ of labels $a, b, ...$ by the following grammar:

$$\phi ::= Z \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \langle K\rangle_c\phi_1 \mid \langle K\rangle_{nc}\phi_1 \mid \phi_1 * \phi_2 \mid \mu Z.\phi_1$$

where $Z \in \text{Var}$, $K \subseteq \mathcal{L}$, and $\mu Z.\phi_1$ has the restriction that any free occurrence of Z in ϕ_1 must be within the scope of an even number of negations. Dual operators are defined as expected: $\phi_1 \vee \phi_2 \stackrel{\text{def}}{=} \neg(\neg\phi_1 \wedge \neg\phi_2)$, $\phi_1 \bowtie \phi_2 \stackrel{\text{def}}{=} \neg(\neg\phi_1 * \neg\phi_2)$, $[K]_c\,\phi_1 \stackrel{\text{def}}{=} \neg\langle K\rangle_c\neg\phi_1$, $[K]_{nc}\,\phi_1 \stackrel{\text{def}}{=} \neg\langle K\rangle_{nc}\neg\phi_1$, $\nu Z.\phi_1 \stackrel{\text{def}}{=} \neg\mu Z.\neg\phi_1\,[\neg Z/Z]$.

Definition 4 A *TSI-based SFL model* $\mathfrak{M}$ is a TSI $\mathfrak{T} = (S, s_0, T, \Sigma, I)$ together with a valuation $\mathcal{V} : \text{Var} \to 2^{\mathfrak{S}}$, where $\mathfrak{S} = S \times \mathfrak{P} \times \mathfrak{A}$ is the set of tuples (s, R, t) of states $s \in S$, support sets $R \in \mathfrak{P}$ in the TSI $\mathfrak{T}$, and transitions $t \in \mathfrak{A} = T \cup \{t_\epsilon\}$. The denotation $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ of an SFL formula ϕ in the model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ is a subset of $\mathfrak{S}$, given by the following rules (omitting the superscript $\mathfrak{T}$):

$$\begin{aligned}
\|Z\|_{\mathcal{V}} &= \mathcal{V}(Z)\\
\|\neg\phi\|_{\mathcal{V}} &= \mathfrak{S} - \|\phi\|_{\mathcal{V}}\\
\|\phi_1 \wedge \phi_2\|_{\mathcal{V}} &= \|\phi_1\|_{\mathcal{V}} \cap \|\phi_2\|_{\mathcal{V}}\\
\|\langle K\rangle_c\phi\|_{\mathcal{V}} &= \{(s,R,t) \mid \exists b \in K.\ \exists s' \in S.\\
&\qquad t \leq (t' = s \xrightarrow{b} s') \wedge t' \in R \wedge (s', R'_{\max}(s'), t') \in \|\phi\|_{\mathcal{V}}\}\\
\|\langle K\rangle_{nc}\phi\|_{\mathcal{V}} &= \{(s,R,t) \mid \exists b \in K.\ \exists s' \in S.\\
&\qquad t \ominus (t' = s \xrightarrow{b} s') \wedge t' \in R \wedge (s', R'_{\max}(s'), t') \in \|\phi\|_{\mathcal{V}}\}\\
\|\phi_1 * \phi_2\|_{\mathcal{V}} &= \{(s,R,t) \mid \exists R_1, R_2.\\
&\qquad R_1 \uplus R_2 \sqsubseteq R \wedge (s, R_1, t) \in \|\phi_1\|_{\mathcal{V}} \wedge (s, R_2, t) \in \|\phi_2\|_{\mathcal{V}}\}\\
\|\mu Z.\phi(Z)\|_{\mathcal{V}} &= \bigcap\{Q \subseteq \mathfrak{S} \mid \|\phi\|_{\mathcal{V}[Z:=Q]} \subseteq Q\}
\end{aligned}$$

where $\mathcal{V}[Z := Q]$ is the valuation $\mathcal{V}'$ which agrees with $\mathcal{V}$ save that $\mathcal{V}'(Z) = Q$. A tuple (s, R, t) of a model $\mathfrak{M}$ is called a *process*. The initial process is the tuple $(s_0, R_{\max}(s_0), t_\epsilon)$, where s_0 is the initial state of $\mathfrak{T}$ and t_ϵ is the empty transition, such that for all $t \in T$ if $s_0 = src(t)$, then $t_\epsilon \leq t$.

Remark 1 *The models we consider here are infinite state systems of finite branching, i.e., image-finite. Also, henceforth, w.l.o.g., we consider only formulae in positive normal form for the definition of the games in the next section.*

3 Trace LMSO Model-Checking Games

Trace LMSO model-checking games $\mathcal{G}(\mathfrak{M}, \phi)$ are played on a model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$, where $\mathfrak{T} = (S, s_0, T, \Sigma, I)$ is a TSI, and on an SFL formula ϕ. The game can also be presented as $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$, or even as $\mathcal{G}_{\mathfrak{M}}(s_0, \phi)$, where $H_0 = (s_0, R_{\max}(s_0), t_\epsilon)$ is the *initial process* of $\mathfrak{S}$. The board in which the game is played has the form $\mathfrak{B} = \mathfrak{S} \times Sub(\phi)$, for a process space $\mathfrak{S} = S \times \mathfrak{P} \times \mathfrak{A}$ of states s in S, support sets R in $\mathfrak{P}$ and transitions t in $\mathfrak{A}$ in the TSI $\mathfrak{T}$. The subformula set $Sub(\phi)$ of an SFL formula ϕ is defined by the Fischer–Ladner closure of SFL formulae in the standard way.

A play is a possibly infinite sequence of configurations $C_0, C_1, ...$, written as $(s, R, t) \vdash \phi$ or $H \vdash \phi$ whenever possible; each C_i is an element of the board $\mathfrak{B}$. Every play starts in the configuration $C_0 = H_0 \vdash \phi$, and proceeds according to the rules of the game given in Fig. 1. As usual for model-checking games, player $\exists$ tries to prove that $H_0 \models \phi$ whereas player $\forall$ tries to show that $H_0 \not\models \phi$.

The rules (FP) and (VAR) control the unfolding of fixpoint operators. Their correctness is based on the fact that $\sigma Z.\phi \equiv \phi[\sigma Z.\phi/Z]$ according to the semantics of the logic. Rules $(\vee)$ and $(\wedge)$ have the same meaning as the disjunction and conjunction rules, respectively, in a Hintikka game for propositional logic. Rules $(\langle\ \rangle_c)$, $(\langle\ \rangle_{nc})$, $([\ ]_c)$ and $([\ ]_{nc})$ are like the rules for quantifiers in a standard Hintikka game semantics for first-order (FO) logic, provided that the box and diamond operators behave, respectively, as restricted universal and existential quantifiers sensitive to the causal information in the partial order model.

Finally, the most interesting rules are $(*)$ and $(\bowtie)$. Local monadic second-order moves are used to recognize conflict-free sets of transitions in $\mathfrak{M}$, i.e., those in the same *trace*. Such moves, which restrict the second-order power (locally) to traces, give the name to this game. The use of $(*)$ and $(\bowtie)$ requires both players to make a choice, but at different levels and with different amount of knowledge. The first player must look for two non-empty conflict-free sets of transitions, with no information on which formula ϕ_i the other player will choose afterwards.

Guided by the semantics of $*$ (resp. $\bowtie$), it is defined that player $\exists$ (resp. $\forall$) must look for a pair of non-empty conflict-free sets of transitions R_0 and R_1 to be assigned to each formula ϕ_i as their support sets. This situation is equivalent to playing a trace for each subformula in the configuration. Then player $\forall$ (resp. $\exists$) must choose one of the two subformulae, with full knowledge of the sets that have been given by player $\exists$ (resp. $\forall$). It is easy to see that $*$ should be regarded as

(FP)	$\dfrac{H \vdash \sigma Z.\phi}{H \vdash Z}$	$\sigma \in \{\mu, \nu\}$	(VAR)	$\dfrac{H \vdash Z}{H \vdash \phi}$	$fp(Z) = \sigma Z.\phi$
$(\vee)$	$\dfrac{H \vdash \phi_0 \vee \phi_1}{H \vdash \phi_i}$	$[\exists]\, i: \quad i \in \{0,1\}$	$(\wedge)$	$\dfrac{H \vdash \phi_0 \wedge \phi_1}{H \vdash \phi_i}$	$[\forall]\, i: \quad i \in \{0,1\}$
$(\langle\ \rangle_c)$	$\dfrac{(s,R,t) \vdash \langle K \rangle_c \phi}{(s', R'_{\max}(s'), t') \vdash \phi}$	$[\exists]\, b: \quad b \in K, s \xrightarrow{b} s' = t' \in R, t \leq t'$			
$(\langle\ \rangle_{nc})$	$\dfrac{(s,R,t) \vdash \langle K \rangle_{nc} \phi}{(s', R'_{\max}(s'), t') \vdash \phi}$	$[\exists]\, b: \quad b \in K, s \xrightarrow{b} s' = t' \in R, t \ominus t'$			
$([\ ]_c)$	$\dfrac{(s,R,t) \vdash [K]_c\, \phi}{(s', R'_{\max}(s'), t') \vdash \phi}$	$[\forall]\, b: \quad b \in K, s \xrightarrow{b} s' = t' \in R, t \leq t'$			
$([\ ]_{nc})$	$\dfrac{(s,R,t) \vdash [K]_{nc}\, \phi}{(s', R'_{\max}(s'), t') \vdash \phi}$	$[\forall]\, b: \quad b \in K, s \xrightarrow{b} s' = t' \in R, t \ominus t'$			
$(*)$	$\dfrac{(s,R,t) \vdash \phi_0 * \phi_1}{(s, R_i, t) \vdash \phi_i}$	$[\exists]\, R_0, R_1; [\forall]\, i: \quad R_0 \uplus R_1 \sqsubseteq R, i \in \{0,1\}$			
$(\bowtie)$	$\dfrac{(s,R,t) \vdash \phi_0 \bowtie \phi_1}{(s, R_i, t) \vdash \phi_i}$	$[\forall]\, R_0, R_1; [\exists]\, i: \quad R_0 \uplus R_1 \sqsubseteq R, i \in \{0,1\}$			

Fig. 1. Trace LMSO Model-Checking Game Rules of SFL. Whereas the notation $[\forall]$ denotes a choice made by Player $\forall$, the notation $[\exists]$ denotes a choice by Player $\exists$.

a special kind of conjunction and $\bowtie$ of disjunction. Indeed, they are a structural conjunction and disjunction, respectively.

Definition 5 The following rules are the *winning conditions* that determine a unique winner for every finite or infinite play $C_0, C_1, ...$ in a game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$.

Player $\forall$ wins a finite play $C_0, C_1, ..., C_n$ or an infinite play $C_0, C_1, ...$ iff:

1. $C_n = H \vdash Z$ and $H \notin \mathcal{V}(Z)$.
2. $C_n = (s,R,t) \vdash \langle K \rangle_c \psi$ and $\{(s', R', t') : b \in K \wedge t \leq t' = s \xrightarrow{b} s' \in R\} = \emptyset$.
3. $C_n = (s,R,t) \vdash \langle K \rangle_{nc} \psi$ and $\{(s', R', t) : b \in K \wedge t \ominus t' = s \xrightarrow{b} s' \in R\} = \emptyset$.
4. $C_n = (s,R,t) \vdash \phi_0 * \phi_1$ and $\{R_0 \cup R_1 : R_0 \uplus R_1 \sqsubseteq R\} = \emptyset$.
5. The play is infinite and there are infinitely many configurations where Z appears, such that $fp(Z) = \mu Z.\psi$ for some formula ψ and Z is the syntactically outermost variable in ϕ that occurs infinitely often.

Player $\exists$ wins a finite play $C_0, C_1, ..., C_n$ or an infinite play $C_0, C_1, ...$ iff:

1. $C_n = H \vdash Z$ and $H \in \mathcal{V}(Z)$.
2. $C_n = (s,R,t) \vdash [K]_c\, \psi$ and $\{(s', R', t') : b \in K \wedge t \leq t' = s \xrightarrow{b} s' \in R\} = \emptyset$.
3. $C_n = (s,R,t) \vdash [K]_{nc}\, \psi$ and $\{(s', R', t) : b \in K \wedge t \ominus t' = s \xrightarrow{b} s' \in R\} = \emptyset$.
4. $C_n = (s,R,t) \vdash \phi_0 \bowtie \phi_1$ and $\{R_0 \cup R_1 : R_0 \uplus R_1 \sqsubseteq R\} = \emptyset$.
5. The play is infinite and there are infinitely many configurations where Z appears, such that $fp(Z) = \nu Z.\psi$ for some formula ψ and Z is the syntactically outermost variable in ϕ that occurs infinitely often.

4 Soundness and Completeness.

Let us first give some intermediate results. Due to lack of space some proofs are omitted or sketched. Let $\mathfrak{T}$ be a TSI and $C = (s, R, t) \vdash \psi$ a configuration in the game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$, as defined before. As usual, the denotation $\|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ of an SFL formula ϕ in the model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ is a subset of $\mathfrak{S}$. We say that a configuration C of $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$ is *true* iff $(s, R, t) \in \|\psi\|^{\mathfrak{T}}_{\mathcal{V}}$ and *false* otherwise.

Fact 1 *SFL is closed under negation.*

Lemma 1 *A game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$, where player $\exists$ has a winning strategy, has a dual game $\mathcal{G}_{\mathfrak{M}}(H_0, \neg\phi)$ where player $\forall$ has a winning strategy, and conversely.*

Proof. We use Fact 1 and duality and completeness of winning conditions. □

Lemma 2 *Player $\exists$ preserves falsity and can preserve truth with her choices. Player $\forall$ preserves truth and can preserve falsity with his choices.*

Proof. The cases for the rules $(\wedge)$ and $(\vee)$ are just as for the Hintikka evaluation games for FO logic. Thus, let us go on to check the rules for the other operators. Firstly, consider the rule $(\langle\ \rangle_c)$ and a configuration $C = (s, R, t) \vdash \langle K\rangle_c\psi$, and suppose that C is false. In this case there is no $b \in K$ such that $t \leq t' = s \xrightarrow{b} s' \in R$ and $(s', R'_{\max}(s'), t') \in \|\psi\|^{\mathfrak{T}}_{\mathcal{V}}$. Hence, the following configurations will be false as well. Contrarily, if C is true, then player $\exists$ can make the next configuration $(s', R'_{\max}(s'), t') \vdash \psi$ true by choosing a transition $t' = s \xrightarrow{b} s' \in R$ such that $t \leq t'$. The case for $(\langle\ \rangle_{nc})$ is similar (simply change $\leq$ for $\ominus$), and the cases for $([\ ]_c)$ and $([\ ]_{nc})$ are dual. Now, consider the rule $(*)$ and a configuration $C = (s, R, t) \vdash \psi_0 * \psi_1$, and suppose that C is false. In this case there is no pair of sets R_0 and R_1 such that $R_0 \uplus R_1 \sqsubseteq R$ and both $(s, R_0, t) \in \|\psi_0\|^{\mathfrak{T}}_{\mathcal{V}}$ and $(s, R_1, t) \in \|\psi_1\|^{\mathfrak{T}}_{\mathcal{V}}$ to be chosen by player $\exists$. Hence, player $\forall$ can preserve falsity by choosing the $i \in \{0, 1\}$ where $(s, R_i, t) \notin \|\psi_i\|^{\mathfrak{T}}_{\mathcal{V}}$, and the next configuration $(s, R_i, t) \vdash \psi_i$ will be false as well. On the other hand, suppose that C is true. In this case, regardless of which i player $\forall$ chooses, player $\exists$ has previously fixed two support sets R_0 and R_1 such that for every $i \in \{0, 1\}$, $(s, R_i, t) \in \|\psi_i\|^{\mathfrak{T}}_{\mathcal{V}}$. Therefore, the next configuration $(s, R_i, t) \vdash \psi_i$ will be true as well. Finally, the deterministic rules (FP) and (VAR) preserve both truth and falsity because of the semantics of fixpoint operators. Recall that for any process H, if $H \in \|\sigma Z.\psi\|$ then $H \in \|\psi\|_{Z:=\|\sigma Z.\psi\|}$ for all free variables Z in ψ. □

Lemma 3 *In any infinite play of a game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$ there is a unique syntactically outermost variable that occurs infinitely often.*

Proof. By a contradiction argument and following an analysis of the structure of those formulae that appear in the configurations of infinite plays. □

Fact 2 *Only rule* (VAR) *can increase the size of a formula in a configuration. All other rules decrease the size of formulae in configurations.*

Lemma 4 *Every play of a game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$ has a uniquely determined winner.*

Proof. For finite plays follows from winning conditions one to four (Definition 5). For plays of infinite length, by analysing the unfolding of fixpoints and winning conditions five of both players. We use Fact 2 and Lemma 3 in this case. □

Definition 6 ***(Approximants)*** *Let $fp(Z) = \mu Z.\phi$ for some formula ϕ and let $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$ be two ordinals, where λ is a limit ordinal. Then:*

$$Z^0 := \mathrm{ff}, \qquad Z^{\alpha+1} = \phi\,[Z^\alpha/Z], \qquad Z^\lambda = \bigvee_{\alpha<\lambda} Z^\alpha$$

For greatest fixpoints the approximants are defined dually. We can now show that the analysis for fixpoint modal logics [1] can be extended to this scenario.

Theorem 1 ***(Soundness)*** *Let $\mathfrak{T}$ be the TSI in the model $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ of a formula ϕ in the game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$. If $H_0 \notin \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ then player $\forall$ wins $H_0 \vdash \phi$.*

Proof. Suppose $H_0 \notin \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. We construct a possibly infinite game tree that starts in $H_0 \vdash \phi$, for player $\forall$. We do so by preserving falsity according to Lemma 2, i.e., whenever a rule requires player $\forall$ to make a choice then the tree will contain the successor configuration that preserves falsity. All other choices that are available for player $\exists$ are included in the game tree.

First, consider only finite plays. Since player $\exists$ only wins finite plays that end in true configurations, then she cannot win any finite play by using her winning conditions one to four. Hence, player $\forall$ wins each finite play in this game tree.

Now, consider infinite plays. The only chance for player $\exists$ to win is to use her winning condition five. So, let the configuration $H \vdash \nu Z.\phi$ be reached such that Z is the syntactically outermost variable that appears infinitely often in the play according to Lemma 3. In the next configuration $H \vdash Z$, variable Z is interpreted as the least approximant Z^α such that $H \notin \|Z^\alpha\|^{\mathfrak{T}}_{\mathcal{V}}$ and $H \in \|Z^{\alpha-1}\|^{\mathfrak{T}}_{\mathcal{V}}$, by the principle of fixpoint induction. As a matter of fact, by monotonicity and due to the definition of fixpoint approximants it must also be true that $H \in \|Z^\beta\|^{\mathfrak{T}}_{\mathcal{V}}$ for all ordinals β such that $\beta < \alpha$. Note that, also due to the definition of fixpoint approximants, α cannot be a limit ordinal λ because this would mean that $H \notin \|Z^\lambda = \bigwedge_{\beta<\lambda} Z^\beta\|^{\mathfrak{T}}_{\mathcal{V}}$ and $H \in \|Z^\beta\|^{\mathfrak{T}}_{\mathcal{V}}$ for all $\beta < \lambda$, which is impossible.

Since Z is the outermost variable that occurs infinitely often and the game rules follow the syntactic structure of formulae, the next time that a configuration $C' = H' \vdash Z$ is reached, Z can be interpreted as $Z^{\alpha-1}$ in order to make C' false as well. And again, if $\alpha - 1$ is a limit ordinal λ, there must be a $\gamma < \lambda$ such that $H' \notin \|Z^\gamma\|^{\mathfrak{T}}_{\mathcal{V}}$ and $H' \in \|Z^{\gamma-1}\|^{\mathfrak{T}}_{\mathcal{V}}$. One can repeat this process even until $\lambda = \omega$.

But, since ordinals are well-founded the play must eventually reach a false configuration $C'' = H'' \vdash Z$ where Z is interpreted as Z^0. And, according to Definition 6, $Z^0 := \mathrm{tt}$, which leads to a contradiction since the configuration $C'' = H'' \vdash \mathrm{tt}$ should be false, i.e., $H'' \in \|\mathrm{tt}\|^{\mathfrak{T}}_{\mathcal{V}}$ should be false, which is impossible. In other words, if H had failed a maximal fixpoint, then there must have been a descending chain of failures, but, as can be seen, there is not.

As a consequence, there is no such least α that makes the configuration $H \vdash Z^\alpha$ false, and hence, the configuration $H \vdash \nu Z.\phi$ could not have been

false either. Therefore, player $\exists$ cannot win any infinite play with her winning condition 5 either. Since player $\exists$ can win neither finite plays nor infinite ones whenever $H_0 \notin \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$, then player $\forall$ must win all plays of $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$. $\square$

Remark 2 *If only finite state systems are considered* $\mathbb{O}\mathrm{rd}$*, the set of ordinals, can be replaced by* $\mathbb{N}$*, the set of natural numbers.*

Theorem 2 ***(Completeness)*** *Let* $\mathfrak{T}$ *be the TSI in the model* $\mathfrak{M} = (\mathfrak{T}, \mathcal{V})$ *of a formula* ϕ *in the game* $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$ *. If* $H_0 \in \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$ *then player* $\exists$ *wins* $H_0 \vdash \phi$*.*

Proof. Suppose that $H_0 \in \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. Due to Fact 1 it is also true that $H_0 \notin \|\neg\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. According to Theorem 1, player $\forall$ wins $H_0 \vdash \neg\phi$, i.e., has a winning strategy in the game $\mathcal{G}_{\mathfrak{M}}(H_0, \neg\phi)$. And, due to Lemma 1, player $\exists$ has a winning strategy in the dual game $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$. Therefore, player $\exists$ wins $H_0 \vdash \phi$ if $H_0 \in \|\phi\|^{\mathfrak{T}}_{\mathcal{V}}$. $\square$

Theorems 1 and 2 imply that the game is determined. Determinacy and perfect information make the notion of truth defined by this Hintikka game semantics coincide with its Tarskian counterpart.

Corollary 1 ***(Determinacy)*** *Player* $\forall$ *wins the game* $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$ *iff player* $\exists$ *does not win the game* $\mathcal{G}_{\mathfrak{M}}(H_0, \phi)$*.*

5 Local Properties and Decidability

We have shown that trace LMSO model-checking games are still sound and complete even when players are allowed to manipulate sets of independent transitions. Importantly, the power of these games, and also of SFL, is that such a second-order quantification is kept both *local* and restricted to transitions in the same *trace*. We now show that trace LMSO model-checking games enjoy several local properties that in turn make them *decidable* in the finite case. Such a decidable result is used in the forthcoming sections to extend the decidability border of model-checking a category of partial order models of concurrency.

Proposition 1 ***(Winning strategies)*** *The winning strategies for the trace LMSO model-checking games of Separation Fixpoint Logic are history-free.*

Proof. Consider a winning strategy π for player $\exists$. According to Lemma 2 and Theorem 2 such a strategy consists of preserving truth with her choices and annotating variables with their approximant indices. But neither of these two tasks depends on the history of a play. Instead they only depend on the current configuration of the game. In particular notice that, of course, this is also the case for the structural operators since the second-order quantification has only a local scope. Similar arguments apply for the winning strategies of player $\forall$. $\square$

This result is key to achieve *decidability* of these games in the presence of the local second-order quantification on the traces of the partial order models we consider. Also, from a more practical standpoint, memoryless strategies are desirable as they are easier to synthesize. However, synthesis is not studied here.

Theorem 3 *The model-checking game for finite Transition systems with independence against Separation Fixpoint Logic specifications is decidable.*

Proof. Since the game is determined, finite plays are decided by winning conditions one to four of either player. Now consider the case of plays of infinite length; since the winning strategies of both players are history-free, we only need to look at the set of different configurations in the game, which is finite even for plays of infinite length. Now, in a finite system an infinite play can only be possible if the model is cyclic. But, since the model has a finite number of states, there is an upper bound on the number of fixpoint approximants that must be calculated (as well as on the number of configurations of the game board that must be checked) in order to ensure that either a greatest fixpoint is satisfied or a least fixpoint has failed. As a consequence, all possible history-free winning strategies for a play of infinite length can be computed, so that the game can be decided using winning condition five of one of the players. □

Remark 3 *A naive local tableau algorithm is at least doubly exponential in the system size, but applying global model-checking techniques, a formula of length k and alternation depth d on a system of size n can be decided in time $k.2^{\mathcal{O}(nd)}$.*

The Interleaving Case. Local properties of trace LMSO model-checking games can also be found in the interleaving case, namely, they coincide with the local model-checking games for the modal μ-calculus as defined by Stirling [9]. As shown in [3] interleaving systems can be cast using SFL by both syntactic and semantic means. The importance of this feature of SFL is that even having constructs for independence and a partial order model, nothing is lost with respect to the main approaches to interleaving concurrency. Recall that $L\mu$ can be obtained from SFL by considering the $*$-free language and using only the following derived operators: $\langle K\rangle\phi = \langle K\rangle_c\phi \vee \langle K\rangle_{nc}\phi$ and $[K]\,\phi = [K]_c\,\phi \wedge [K]_{nc}\,\phi$.

Proposition 2 *If either a model with an empty independence relation or the syntactic $L\mu$ fragment of SFL is considered, then the trace LMSO model-checking games for SFL degenerate to the local model-checking games for the μ-calculus.*

6 Model-Checking Partial Order Models of Concurrency

In this section we use trace LMSO model-checking games to push forward the decidability border of the model-checking problem of a particular class of partial order models, namely, of a class of event structures [6]. More precisely, we improve previous results [5, 7] in terms of logical expressive power.

6.1 SFL on Event Structures

Definition 7 A *labelled event structure* $\mathfrak{E}$ is a tuple $(E, \preccurlyeq, \sharp, \eta, \Sigma)$, where E is a set of events that are partially ordered by $\preccurlyeq$, the causal dependency relation on events, $\sharp \subseteq E \times E$ is an irreflexive and symmetric conflict relation, and $\eta : E \to \Sigma$ is a labelling function such that the following holds:

If $e_1, e_2, e_3 \in E$ and $e_1 \sharp e_2 \preccurlyeq e_3$, then $e_1 \sharp e_3$.
$\forall e \in E$ the set $\{e' \in E \mid e' \preccurlyeq e\}$ is finite.

The independence relation on events is defined with respect to the causal and conflict relations. Two events e_1 and e_2 are *concurrent*, denoted by $e_1 \mathsf{co}\, e_2$, iff $e_1 \not\preccurlyeq e_2$ and $e_2 \not\preccurlyeq e_1$ and $\neg(e_1 \sharp e_2)$. The notion of computation state for event structures is that of a *configuration*. A configuration C is a conflict-free set of events (i.e., if $e_1, e_2 \in C$, then $\neg(e_1 \sharp e_2)$) such that if $e \in C$ and $e' \preccurlyeq e$, then $e' \in C$. The restriction to image-finite models implies that the partial order $\preccurlyeq$ of $\mathfrak{E}$ is of *finite branching*, and hence for all C, the set of immediately next configurations is bounded. If one further requires that for all $e \in C$, the set of future non-isomorphic configurations rooted at e defines an equivalence relation of *finite index*, then $\mathfrak{E}$ is also *regular* [11].

An event structure $\mathfrak{E} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ determines a TSI $\mathfrak{T} = (S, T, \Sigma, I)$ by means of an inclusion functor from the category $\mathcal{ES}$ of event structures to the category $\mathcal{TSI}$ of TSI. Here we give such a mapping in a set-theoretic way since this presentation is more convenient for us. A categorical presentation can be found in [4]. The construction $\lambda : \mathcal{ES} \to \mathcal{TSI}$ is as follows:

$$
\begin{aligned}
S &= \{C \subseteq E \mid \forall e_1, e_2 \in C.\ \neg(e_1 \sharp e_2), (e \in C \wedge e' \preccurlyeq e \Rightarrow e' \in C)\}\ . \\
T &= \{(C, a, C') \in S \times \Sigma \times S \mid \exists e \in E.\ \eta(e) = a, e \notin C, C' = C \cup \{e\}\} \\
I &= \{((C_1, a, C_1'), (C_2, b, C_2')) \in T \times T \mid \exists (e_1, e_2) \in \mathsf{co}. \\
&\qquad \eta(e_1) = a, \eta(e_2) = b, C_1' = C_1 \cup \{e_1\}, C_2' = C_2 \cup \{e_2\}\}
\end{aligned}
$$

where the set of states S of the TSI $\mathfrak{T}$ is isomorphic to the set *Conf* of *configurations* $C \subseteq E$ of the event structure $\mathfrak{E}$, and the set of labels Σ remains the same. Since the semantics of SFL is given only by defining the relations $\leq$, $\ominus$, $\#$ and $\otimes$ on pairs of transitions at every state, i.e., on pairs of events at every configuration, such a semantics can also be given directly from the elements of an event structure using the construction above. Moreover, support sets and all elements needed to build a lattice $\mathfrak{S}$ and hence a model for SFL in the category of event structures are defined using the same definitions as for the TSI case.

6.2 A Computable Folding Functor from Event Structures to TSI

Although we have defined satisfiability of SFL formulae in event structure models, model-checking these structures is rather difficult since very simple concurrent systems can have infinite event structures as models, in particular, all those with recursive behaviour. In order to overcome this problem we define a morphism (a functor) that folds a possibly infinite event structures into a TSI. Such a morphism and the procedure to effectively compute it is described below.

The Quotient Set Method. Let $Q = (\mathit{Conf}/\sim)$ be the *quotient set representation* of *Conf* by $\sim$ in a finite or infinite event structure $\mathfrak{E}$, where *Conf* is the set of configurations in $\mathfrak{E}$ and $\sim$ is an equivalence relation on such configurations. The equivalence class $[X]_\sim$ of a configuration $X \in \mathit{Conf}$ is the set

$\{C \in \mathit{Conf} \mid C \sim X\}$. A quotient set Q where $\sim$ is decidable is said to have a decidable characteristic function, and will be called a *computable quotient set.*

Definition 8 A *regular quotient set* $(\mathit{Conf}/\sim)$ of an event structure $\mathfrak{E}$ is a computable quotient set representation of $\mathfrak{E}$ with a finite number of equivalence classes.

Having defined a regular quotient set representation of $\mathfrak{E}$, the morphism $\lambda : \mathcal{ES} \to \mathcal{TSI}$ above can be modified to defined a new map $\lambda_f : \mathcal{ES} \to \mathcal{TSI}$ which folds a (possibly infinite) event structure into a TSI:

$$
\begin{aligned}
S &= \{[C]_\sim \subseteq \mathit{Conf} \mid \exists [X]_\sim \in Q = (\mathit{Conf}/\sim).\ C \sim X\} \\
T &= \{([C]_\sim, a, [C']_\sim) \in S \times \Sigma \times S \mid \exists e \in E.\ \eta(e) = a, e \notin C, C' = C \cup \{e\}\} \\
I &= \{(([C_1]_\sim, a, [C_1']_\sim), ([C_2]_\sim, b, [C_2']_\sim)) \in T \times T \mid \exists (e_1, e_2) \in \mathsf{co}. \\
&\qquad \eta(e_1) = a, \eta(e_2) = b, C_1' = C_1 \cup \{e_1\}, C_2' = C_2 \cup \{e_2\}\}
\end{aligned}
$$

Lemma 5 *Let $\mathfrak{T}$ be a TSI and $\mathfrak{E}$ an event structure. If $\mathfrak{T} = \lambda_f(\mathfrak{E})$, then the models $(\mathfrak{T}, \mathcal{V})$ and $(\mathfrak{E}, \mathcal{V})$ satisfy the same set of SFL formulae.*

Proof. The morphism $\lambda_f : \mathcal{ES} \to \mathcal{TSI}$ from the category of event structures to the category of TSI has a unique right adjoint $\varepsilon : \mathcal{TSI} \to \mathcal{ES}$, the unfolding functor that preserves labelling and the independence relation between events, such that for any $\mathfrak{E}$ we have that $\mathfrak{E}' = (\varepsilon \circ \lambda_f)\ (\mathfrak{E})$, where $\mathfrak{E}'$ is isomorphic to $\mathfrak{E}$. But SFL formulae do not distinguish between models and their unfoldings, and hence cannot distinguish between $(\mathfrak{T}, \mathcal{V})$ and $(\mathfrak{E}', \mathcal{V})$. Moreover, SFL formulae do not distinguish between isomorphic models equally labelled, and therefore cannot distinguish between $(\mathfrak{E}', \mathcal{V})$ and $(\mathfrak{E}, \mathcal{V})$ either. □

Having defined a morphism λ_f that preserves SFL properties, one can now define a procedure that constructs a TSI model from a given event structure.

Definition 9 Let $\mathfrak{E} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ be an event structure and $(\mathit{Conf}/\sim)$ a regular quotient set representation of $\mathfrak{E}$. A *representative set* E_r of $\mathfrak{E}$ is a subset of E such that $\forall C \in \mathit{Conf}.\ \exists X \subseteq E_r.\ C \sim X$.

Lemma 6 *Let $\mathfrak{E}$ be an event structure. If $\mathfrak{E}$ is represented as a regular quotient set $(\mathit{Conf}/\sim)$, then a finite representative set E_r of $\mathfrak{E}$ is effectively computable.*

Proof. Construct a finite representative set E_r as follows. Start with $E_r = \emptyset$ and $C_j = C_0 = \emptyset$, the initial configuration or root of the event structure. Check $C_j \sim X_i$ for every equivalence class $[X_i]_\sim$ in $Q = (\mathit{Conf}/\sim)$ and whenever $C_j \sim X_i$ holds define both a new quotient set $Q' = Q \setminus [X_i]_\sim$ and a new $E_r = E_r \cup C_j$. This subprocedure terminates because there are only finitely many equivalence classes to check and the characteristic function of the quotient set is decidable. Now, do this recursively in a breadth-first search fashion in the partial order defined on E by $\preccurlyeq$, and stop when the quotient set is empty. Since $\preccurlyeq$ is of finite branching and all equivalence classes must have finite configurations, the procedure is bounded both in depth and breath and the quotient set will always eventually get smaller. Hence, such a procedure always terminates. It is easy to see that this procedure only terminates when E_r is a representative set of $\mathfrak{E}$. □

A finite representative set E_r is big enough to define all states in the TSI representation of $\mathfrak{E}$ when using λ_f. However, such a set may not be enough to recognize all transitions in the TSI. In particular, cycles cannot be recognized using E_r. Therefore, it is necessary to compute a set E_f where cycles in the TSI can be recognized. We call E_f a *complete representative set* of $\mathfrak{E}$. The procedure to construct E_f is similar to the previous one.

Lemma 7 *Let $\mathfrak{E} = (E, \preccurlyeq, \sharp, \eta, \Sigma)$ be an event structure and E_r a finite representative set of $\mathfrak{E}$. If $\mathfrak{E}$ is represented as a regular quotient set $(Conf/\sim)$, then a finite complete representative set E_f of $\mathfrak{E}$ is effectively computable.*

Proof. Start with $E_f = E_r$, and set $\mathfrak{C} = Conf(E_r)$, the set of configurations generated by E_r. For each C_j in E_r check in $\preccurlyeq$ the set $Next(C_j)$ of next configurations to C_j, i.e., those configurations C'_j such that $C'_j = C_j \cup \{e\}$ for some event e in $E \setminus C_j$. Having computed $Next(C_j)$, set $E_f = E_f \cup (\bigcup Next(C_j))$ and $\mathfrak{C} = \mathfrak{C} \setminus \{C_j\}$, and stop when $\mathfrak{C}$ is empty. This procedure behaves as the one described previously. Notice that at the end of this procedure E_f is complete since it contains the next configurations of all elements in E_r. □

Proposition 3 *The TSI $\mathfrak{T}$ generated from an event structure $\mathfrak{E}$ using λ_f and a finite complete representative E_f of $\mathfrak{E}$ is the smallest TSI that represents $\mathfrak{E}$.*

Proof. From Lemmas 6 and 7. There is only one state in $\mathfrak{T}$ for each equivalence class in the quotient set representation of $\mathfrak{E}$. Similarly there can be only one transition in $\mathfrak{T}$ for each relation on the equivalence classes of configurations in $\mathfrak{E}$ since, due to **A1** of TSI (determinacy), λ_f forgets repeated transitions in T. □

6.3 Temporal Verification of Regular Infinite Event Structures

Based on Lemmas 5 and 7 and on Theorem 3, we can give a decidability result for the class of event structures studied in [5, 11] against SFL specifications. Such a result, which is obtained by representing a regular event structure as a regular quotient set, is a corollary of the following theorem:

Theorem 4 *The model-checking problem for an event structure $\mathfrak{E}$ represented as a regular quotient set $(Conf/\sim)$ against SFL specifications is decidable.*

Regular Event Structures as Finite CCS Processes. A regular event structure [5, 11] can be generated by a finite concurrent system represented by a finite number of (possibly recursive) CCS processes [12]. Syntactic restrictions on CCS that generate only finite systems have been studied. Finiteness of CCS processes and restriction to image-finite models give both requirements for regularity on the event structures that are generated. Now, w.l.o.g., consider only deterministic CCS processes without auto-concurrency. A CCS process is deterministic if whenever $a.M + b.N$, then $a \neq b$, and similarly has no auto-concurrency if whenever $a.M \parallel b.N$, then $a \neq b$. Notice that any CCS process P

that either is nondeterministic or has auto-concurrency can be converted into an equivalent process Q which generates an event structure that is isomorphic, up to relabelling of events, to the one generated by P. Eliminating nondeterminism and auto-concurrency can be done by relabelling events in $\mathcal{P}(P)$, the powerset of CCS processes of P, with an injective map $\theta : \Sigma \to \Sigma^*$ (where Σ^* is a set of labels and $\Sigma \subseteq \Sigma^*$), and by extending the Synchronization Algebra according to the new labelling of events so as to preserve pairs of (labels of) events that can synchronize. Also notice that the original labelling can always be recovered from the new one, i.e., the one associated with the event structure generated by Q, since θ is injective and hence has inverse $\theta^{-1} : \Sigma^* \to \Sigma$. In [5, 11], deterministic regular event structures are called *trace* event structures.

Finite CCS Processes as Regular Quotient Sets. Call $ESProc(P)$ to the set of configurations of the event structure generated by a CCS process P of the kind described above. The set $ESProc(P)$ together with an equivalence relation between CCS processes $\equiv_{CCS}$ given simply by syntactic equality between them is a regular quotient set representation ($ESProc(P)$ / $\equiv_{CCS}$) of the event structure generated by P. Notice that since there are finitely many CCS processes, i.e., $\mathcal{P}(P)$ is finite, then the event structure generated by P is of finite-branching and the number of equivalence classes is also bounded. Finally, $\equiv_{CCS}$ is clearly decidable because process P is always associated with configuration $\emptyset$ and any other configuration in $ESProc(P)$ can be associated with only one CCS process in $\mathcal{P}(P)$ as they are deterministic and have no auto-concurrency after relabelling.

Corollary 2 *Model-checking regular trace event structures against Separation Fixpoint Logic specifications is decidable.*

7 Concluding Remarks and Related Work

In this paper we introduced a new kind of model-checking games where both players are allowed to choose *sets* of independent elements in the underlying model. These games, which we call trace LMSO model-checking games, are proved to be *sound* and *complete*, and therefore determined. They can be played on partial order models of concurrency since the one-step interleaving semantics of such models need not be considered. We showed that, similar to [3], by defining infinite games where both players have a *local* second-order power on *conflict-free* sets of transitions, i.e., those in the same *trace*, one can obtain new positive decidability results on the study of partial order models of concurrency. Indeed, we have pushed forward the borderline of the decidability of model-checking event structures. To the best of our knowledge the technique we presented here is the only game-based procedure defined so far that can be used to verify all usual temporal properties of the kind of event structures we studied. We wonder how much further one can go in terms of logical expressive power before reaching the MSO undecidability barrier when model-checking event structures.

Related Work. Model-checking games have been an active area of research in the last decades (cf. [2, 10]). Most approaches based on games have considered either only interleaving models or the one-step interleaving semantics of partial order models. Our work differs from these approaches in that we deal with games played on partial order models without considering interleaving simplifications. However, verification procedures in finite partial order models can be undecidable. Nevertheless, the game presented here is *decidable* in the finite case.

Regarding the temporal verification of event structures, previous studies have been done on restricted classes. Closer to our work is [5, 7]. Indeed, model-checking regular event structures [11] has turned out to be rather difficult and previous work has shown that verifying MSO properties on these structures is already *undecidable*. For this reason weaker logics have been studied. Unfortunately, although very interesting results have been achieved, especially in [5] where CTL* properties can be verified, previous approaches have not managed to define decidable theories for a logic with enough power to express all usual temporal properties as can be done with $L\mu$ in the interleaving case, and hence with SFL in a noninterleaving setting. The difference between [5] and the approach we presented is that in [5] a *global* second-order quantification on conflict-free sets in the partial order is permitted, whereas only a *local* second-order quantification in the same kind of sets is defined here, but such a second-order power can be embedded into fixpoint specifications, which in turn allows one to express more temporal properties. Therefore, we have improved in terms of temporal expressive power previous results on model-checking regular event structures against a branching-time logic. Our work is the first (local) game approach in doing so.

References

1. J. Bradfield and C. Stirling. Modal mu-calculi. In *Handbook of Modal Logic*, volume 3, pages 721–756. Elsevier, 2006.
2. E. Grädel. Model checking games. *Electr. Notes Theor. Comput. Sci.*, 67, 2002.
3. J. Gutierrez. Logics and bisimulation games for concurrency, causality and conflict. In L. de Alfaro, editor, *FoSSaCS*, LNCS **5504**, pages 48–62. Springer, 2009.
4. A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Inf. Comput.*, 127(2):164–185, 1996.
5. P. Madhusudan. Model-checking trace event structures. In *LICS*, pages 371–380. IEEE Computer Society, 2003.
6. M. Nielsen and G. Winskel. Models for concurrency. In *Handbook of Logic in Computer Science*, volume 4, pages 1–148. Oxford University Press, 1995.
7. W. Penczek. Model-checking for a subclass of event structures. In E. Brinksma, editor, *TACAS*, LNCS **1217**, pages 145–164. Springer, 1997.
8. J. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, pages 55–74. IEEE Computer Society, 2002.
9. C. Stirling. Local model checking games. In I. Lee and S. A. Smolka, editors, *CONCUR*, LNCS **962**, pages 1–11. Springer, 1995.
10. C. Stirling. *Modal and Temporal Properties of Processes*. LNCS. Springer, 2001.
11. P. S. Thiagarajan. Regular trace event structures. Technical report, BRICS, 1996.
12. G. Winskel. Event structure semantics for ccs and related languages. In M. Nielsen and E. Schmidt, editors, *ICALP*, LNCS **140**, pages 561–576. Springer, 1982.