A Logic for Fresh Labelled Transition Systems

Mohamed H. Bandukara
Queen Mary University of London, UK

Nikos Tzevelekos
Queen Mary University of London, UK

Abstract

We introduce a Hennessy-Milner logic with recursion for Fresh Labelled Transition Systems (FLTSs). These are nominal labelled transition systems which keep track of the history, i.e. of data values seen so far, and can model fresh data generation. In particular, FLTSs generalise the computations of Fresh-Register Automata, which in turn can be seen as a "regular" class of history-tracking automata operating on infinite input alphabets. The logic we introduce is a modal mu-calculus equipped with infinite disjunctions over arbitrary and fresh data values respectively, while its recursion is parameterised on vectors of data values. It can express a variety of properties, such as the existence of an infinite path of distinct data values, the absence of paths where values are repeated, or the existence of a finite path where some taint property is violated. We study the model-checking problem and its complexity via a reduction to parity games and, using nominal sets techniques, provide an exponential upper bound for it.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects; Theory of computation → Modal and temporal logics

Keywords and phrases Nominal Transition Systems, Hennessy-Milner Logic, Modal Mu-Calculus, Register Automata, Nominal Sets, Parity Games

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.23

Related Version *Full Version*: `https://arxiv.org/abs/2506.14538`

Funding *Mohamed H. Bandukara*: supported by EPSRC DTP EP/R513106/1.

1 Introduction

Nominal Labelled Transition Systems (LTSs) can capture computational scenarios which require unbounded sets of data values [5]. In the most elementary case, data values are just *names*, that is, atomic identifiers that can only be compared for equality. Nominal LTSs can be used, for example, for modelling mobile processes [38] and computation with variables [24, 21], and for verifying XML query languages [41]. A particularly popular class of nominal LTSs are the ones generated by computations of "regular" automata over infinite alphabets, such as register automata [24] and the (largely) equivalent nominal automata [5]. A register automaton is equipped with a finite set of states, and hence the nod to regularity, but also a finite set of registers where it can store data values and compare them with, and possibly update them with, data values from the input. This design allows register automata to capture languages over infinite alphabets even though their specification is finite.

Fresh-Register Automata (FRAs) [44] expand on this design by having the option of allowing the automaton to accept a name just if it is globally fresh, that is, it has not appeared so far in the input. FRAs can be used to capture computational models that use names and name generation, e.g. (finitary) π-calculus processes [44, 3] and programs with mutable references or objects [35, 32]. They have been applied to program verification and system modelling, leading to verification tools [32, 20] and model learning techniques [6, 1]. Verification of FRAs has mainly focussed on bisimulation equivalence [44, 33, 34]. In this

work we are interested in furthering formal verification for nominal computation, with an interest on FRAs and more generally history-dependent computation, in the direction of modal logics.

Hennessy-Milner Logic (HML) is a modal behavioural logic that captures branching-time properties of labelled transition systems. It was initially introduced in 1980 by Hennessy and Milner [22] as an alternative exposition of observational equivalence [8]. Observational equivalence has a focus on testing whether two systems can simulate each other in a step-by-step manner, whereas HML logic focuses on the expressiveness of a single system. In image-finite systems, two states are equivalent just if they satisfy the same HML formulas [23]. The extension of HML with recursion, called modal μ-calculus, was introduced by Kozen [29]. Here, we study a nominal extension thereof, where modalities are allowed to include names and where quantifications over some/all names and some/all fresh names are catered for. More specifically, our contributions include:

- We introduce the notion of *fresh LTS (FLTS)*, which extends nominal LTS with history dependence.
- We introduce *fresh HML with recursion*, in short μFHML, which allows us to express recursive branching-time properties involving names and name-freshness.
- We study the model-checking problem for μFHML formulas on FLTSs. This leads us to developing parity games for μFHML and, through a series of reductions and using nominal techniques, devising a decision procedure for them.
- We provide (exponential) upper bound calculations for μFHML parity games and μFHML model checking, in the latter case both on general FLTSs and on FRAs.

Although FRAs are our main target model, there are several advantages in defining and utilising FLTSs here. FLTS is a more general formalism, encompassing more scenarios besides FRAs (e.g. π-calculus processes). Actions performed in an FLTS can be more general than those given in FRAs, and allow e.g. for several fresh names to be generated at once or extra mechanisms for manipulating names between transitions (permutations, duplications, etc.). Additionally, FLTSs offer high-level reasoning and abstract away name/register manipulations.

Related work. HML for the π-calculus was first examined in [31]. For properties involving paths of unbounded length, it is natural to examine recursive extensions thereof such as the modal μ-calculus and variants thereof [8]. The logic we introduced is based on the π-μ-calculus of Dam [12] and is related to the recent work on nominal μ-calculi of Klin and collaborators [26, 27, 15]. The π-μ-calculus [12] specifically targets π-calculus processes, but the syntax used for recursion is the same as ours. The study of a history-dependent nominal μ-calculus in [15] is the work closest to ours. In a certain sense, our work continues that of *loc cit.* by operating on history-dependent LTSs (and FRAs) and accommodating a vectorial version of recursion. We show that this combination still yields a decidable model-checking problem, and also provide a more algorithmic view on model checking via parity games and produce complexity bounds.

In the area of nominal techniques, the modal logics for nominal LTSs by Parrow et al. [37] can subsume our logic but do not allow for decidable model checking. Logics for variants of register automata have been widely studied, e.g. [42, 14, 18]. The main focus of those works is on decidability, e.g. by restricting the number of registers allowed. No such restrictions are imposed on our logic, which is undecidable for satisfiability but our focus is instead on model checking. There have also been a series of works on logics related to the π-calculus, e.g. [36, 28, 4]. Those works mainly study expressiveness and characterisation of variants of behavioural equivalence for π-calculus processes. Other related works, such as [11, 7], expand on the modal μ-calculus and study model checking, and show it to be decidable for finitary π-calculus processes.

2 Nominal sets and fresh labelled transition systems (FLTSs)

We start by briefly introducing the nominal sets framework, which will be the basis of much of this paper. Let us fix $\mathbb{A}$ to be a countably infinite set of ***names*** (or ***data values***) upon which nominal sets are built. A permutation π on $\mathbb{A}$ (i.e. a bijection $\pi : \mathbb{A} \to \mathbb{A}$) is considered *finite* if $\mathrm{supp}(\pi) = \{a \in \mathbb{A} \mid \pi(a) \neq a\}$ is finite. For example, the identity permutation id is finite ($id(a) = a$ for all $a \in \mathbb{A}$) and, for any $a, b \in \mathbb{A}$, so is the permutation $(a\ b)$ given by: $(a\ b)(a) = b$, $(a\ b)(b) = a$, and $(a\ b)(x) = x$ for all $x \neq a, b$. We denote the set of all finite permutations in $\mathbb{A}$ as Perm. Given sequences of distinct names $\vec{a}, \vec{b} \in \mathbb{A}^k$, we write $(\vec{a} \mapsto \vec{b})$ for the finite permutation obtained by canonically extending the map $\{(a_i, b_i) \mid 1 \leq i \leq k\}$.

▶ **Definition 1** ([19, 39]). A ***nominal set*** $\mathcal{X}$ is a set X equipped with an action from Perm, i.e. a function $\_ \cdot \_ : \mathsf{Perm} \times X \to X$ such that for all $x \in X$ and $\pi, \pi' \in \mathsf{Perm}$:

- $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$, $id \cdot x = x$.

A set of names $N \subseteq \mathbb{A}$ *supports* an element $x \in X$ if, for all $\pi \in \mathsf{Perm}$, if π fixes all elements of N then $\pi \cdot x = x$. We stipulate that all elements of X be *finitely supported*, i.e. for all $x \in X$ there is some finite $N \subseteq \mathbb{A}$ that supports x. We write $\mathrm{supp}(x)$ for *the support of* x, i.e. the least set supporting x. We say that a *is fresh for* x, and write $a\#x$, if $a \notin \mathrm{supp}(x)$.

By abuse of notation, we may identify $\mathcal{X}$ with its carrier set X. The action on a nominal set extends to products and subsets by $\pi \cdot (x, y) = (\pi \cdot x, \pi \cdot y)$ and, for any $S \subseteq \mathcal{X}$, by $\pi \cdot S = \{\pi \cdot x \mid x \in S\}$. Additionally, if $\mathcal{X}, \mathcal{Y}$ are nominal sets then so is $\mathcal{X} \times \mathcal{Y}$, and the set of subsets of $\mathcal{X}$ with finite support. Accordingly, given a relation $R \subseteq \mathcal{X} \times \mathcal{Y}$ and $\pi \in \mathsf{Perm}$, we have $\pi \cdot R = \{(\pi \cdot x, \pi \cdot y) \mid (x, y) \in R\}$. We call R *equivariant* if $\mathrm{supp}(R) = \emptyset$, i.e. for all π and $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $(x, y) \in R \iff (\pi \cdot x, \pi \cdot y) \in R$.

A notion in nominal sets we will be making extensive use of is that of an *orbit*.

▶ **Definition 2.** Let $\mathcal{X}$ be a nominal set. Two elements $x, y \in \mathcal{X}$ are *nominally equivalent* (denoted $x \sim_{\mathsf{nom}} y$) if $\pi \cdot x = y$ for some $\pi \in \mathsf{Perm}$. Note $\sim_{\mathsf{nom}}$ is an equivalence relation. For each $x \in \mathcal{X}$, we define its ***orbit*** $\mathbb{O}(x)$ to be its $\sim_{\mathsf{nom}}$-equivalence class; whereas for each finite $N \subseteq \mathbb{A}$ we let $\mathbb{O}_N(x)$ be its ***N-orbit***, i.e. its closure under permutations fixing N:

$$\mathbb{O}(x) = \{\pi \cdot x \mid \pi \in \mathsf{Perm}\} \qquad \mathbb{O}_N(x) = \{\pi \cdot x \mid \forall a \in N.\, \pi(a) = a\}$$

Observe that $\mathbb{O}(x) = \mathbb{O}_\emptyset(x)$. Moreover, for any $S \subseteq \mathcal{X}$ with finite support we let:

- the *closure* of S be $\mathcal{C}\ell(S) = \{\pi \cdot x \mid x \in S \wedge \pi \in \mathsf{Perm}\}$;
- the *set of orbits* of S be $\mathcal{O}(S) = \{\mathbb{O}_{\mathrm{supp}(S)}(x) \mid x \in S\}$.

We say that a nominal set $\mathcal{X}$ is ***orbit-finite*** if $\mathcal{O}(\mathcal{X})$ is finite. If $\mathcal{X}$ is orbit-finite, we let its *register index* be $r_i(\mathcal{X}) = \max\{|\,\mathrm{supp}(x)\,| \mid x \in \mathcal{X}\}$.

Note in particular that $\mathbb{O}(x) = \mathcal{C}\ell(\{x\})$, for any $x \in \mathcal{X}$, and $\mathcal{O}(\mathcal{X}) = \{\mathbb{O}(x) \mid x \in \mathcal{X}\}$. Given $N \subseteq N'$ and any $x \in \mathcal{X}$, we have $\mathbb{O}_{N'}(x) \subseteq \mathbb{O}_N(x) \subseteq \mathcal{X}$ and $\mathrm{supp}(\mathbb{O}_N(x)) \subseteq N$. Moreover, for any finitely supported $S \subseteq \mathcal{X}$ we can see by inspection that:

$$\bigcup \mathcal{O}(S) = S \subseteq \mathcal{C}\ell(S) \subseteq \mathcal{X}$$

and $\mathrm{supp}(\mathcal{C}\ell(S)) = \emptyset$. Orbit-finiteness essentially captures the notion of a set being "finite up to permutation", and it is central in the development of automata over nominal sets [5]. Some useful albeit more technical properties follow.

▶ **Lemma 3.** *Let $\mathcal{X}, \mathcal{Y}$ be nominal sets and $X, Y \subseteq \mathcal{X}$. Then:*
1. $\{\mathbb{O}(x) \mid x \in X\} = \mathcal{O}(\mathcal{C}(X))$ *and* $|\mathcal{O}(\mathcal{C}(X))| \leq |\mathcal{O}(X)|$*;*
2. *if X has empty support then $\mathcal{C}(X \times Y) = X \times \mathcal{C}(Y)$;*
3. *if $\mathcal{X}, \mathcal{Y}$ are orbit-finite then $|\mathcal{O}(\mathcal{X} \times \mathcal{Y})| \leq |\mathcal{O}(\mathcal{X})| \cdot |\mathcal{O}(\mathcal{Y})| \cdot {n_1}^{n_2} \cdot (1+\epsilon)$ where $\{n_1, n_2\} = \{r_i(\mathcal{X}), r_i(\mathcal{Y})\} \neq \{0\}$ and $n_1 \geq n_2$, and $\epsilon \leq 1$ is a constant that vanishes for $n_2 \geq 4$.*

We next define nominal labelled transition systems, where states and labels form nominal sets, which retain a possibly unbounded history component but are otherwise orbit-finite. For that purpose, orbit-finiteness is relaxed to accommodate histories of unbounded size. One part of our solution is to use configurations comprising pairs of a *state* s and a *history* H, with $\mathrm{supp}(s) \subseteq H$, where only the set of states is orbit-finite. The other part is to stipulate that the names appearing only in the history component of a configuration *do not matter*, in the following way. If a transition can be taken from a given configuration then:
- *weakening* the configuration with some fresh name (i.e. fresh both for the source and target configurations), or
- *strengthening* the configuration by removing a name appearing only in its history (and not in its state),

should not affect whether the transition can be taken or not. Note here the asymmetry in the freshness requirements for a in weakening and strengthening: weakening requires that the name be completely fresh. This is in order to account for transitions performing fresh name generation. A transition freshly generating some name a would be rendered invalid if we added a to that transition's initial history.

▶ **Definition 4.** A *Fresh Nominal Labelled-Transition System (FLTS)* is a tuple $\mathcal{L} = \langle \mathcal{S}, L, \to \rangle$, where (writing $\mathcal{P}_{\mathrm{fin}}$ for the finite powerset operator):
- $\mathcal{S}$ is a nominal set of *states* and L is a nominal set of *actions*,
- $\to \subseteq \hat{\mathcal{S}} \times L \times \hat{\mathcal{S}}$ is an equivariant *transition relation*,
- where $\hat{\mathcal{S}} = \{(s, H) \in \mathcal{S} \times \mathcal{P}_{\mathrm{fin}}(\mathbb{A}) \mid \mathrm{supp}(s) \subseteq H\}$ is the nominal set of *configurations*;

and such that, for all $(s, H) \xrightarrow{\ell} (s', H')$ and $a \in \mathbb{A}$:
- $\mathrm{supp}(s') \subseteq \mathrm{supp}(s) \cup \mathrm{supp}(\ell)$ and $H' = H \cup \mathrm{supp}(\ell)$;
- if $a \notin H'$ then $(s, H \cup \{a\}) \xrightarrow{\ell} (s', H' \cup \{a\})$ (*weakening*);
- if $a \notin \mathrm{supp}(s)$ then $(s, H \setminus \{a\}) \xrightarrow{\ell} (s', H'')$ for some H'' (*strengthening*).

$\mathcal{L}$ is called *fresh-orbit-finite* if $\mathcal{S}$ and L are orbit-finite.

The notion of FLTS is fairly general and encompasses popular computational scenarios such as name-carrying processes (e.g. in the π-calculus) and name-manipulating programs (e.g. Java programs with objects). At the automata level, one such formalism is that of *fresh-register automata* [44], which are the baseline paradigm in our study. As we are interested in behavioural properties, rather than language acceptance, we disregard initial and final states. The automata we define next operate on inputs $(t, a) \in \Sigma \times \mathbb{A}$ where t comes from a finite set of *tags* and a is a name.

▶ **Definition 5.** An *r-**Fresh-Register Automaton** (r-**FRA**)* is a tuple $\mathcal{A} = \langle Q, \mu, \Sigma, \delta \rangle$, where $r \in \mathbb{N}$ is the number of *registers* in $\mathcal{A}$ and:
- Q is a finite set of *states*, and $\mu : Q \to \mathcal{P}([1, r])$ an *availability function*;
- Σ is a finite set of tags, and $\delta : Q \times \Sigma \times \{i, i^{\bullet}, i^{\circledast} \mid 1 \leq i \leq r\} \times Q$ is the transition relation;

subject to the following conditions:
- if $(q, t, i, q') \in \delta$ then $i \in \mu(q)$ and $\mu(q') \subseteq \mu(q)$;
- if $(q, t, i^{\bullet}, q') \in \delta$ or $(q, t, i^{\circledast}, q') \in \delta$, then $\mu(q') \subseteq \mu(q) \cup \{i\}$.

We shall write $q \xrightarrow{t,x} q'$ for $(q, t, x, q') \in \delta$.

We can see that the labels appearing in FRA transitions are of the form (t, x), where t comes from a finite alphabet while x is a variant of a register index i. Assuming the automaton is at state q and given $q \xrightarrow{t,x} q'$, depending on x the automaton will perform one of the following actions before moving to state q':

- $x = i$: read the input (t, a), where a is the name currently in register i – note here that i needs to be available at q (i.e. $i \in \mu(q)$);
- $x = i^\bullet$: read any input (t, a) so long a is not currently stored in any register (i.e. it is *locally fresh*), and store a in register i;
- $x = i^\circledast$: read any input (t, a) so long a has not be seen before (i.e. it is *globally fresh*), and store a in register i.

Formally, the semantics of an FRA can be given in terms of its derived FLTS. The latter features states of the form (q, ρ) where q is a state and ρ an assignment of values to available registers, i.e. the ones in $\mu(q)$. It is good to bear in mind that q has empty support.

▶ **Definition 6.** Given an r-FRA $\mathcal{A} = \langle Q, \mu, \Sigma, \delta\rangle$ we define its FLTS $\mathcal{G}_\mathcal{A} = \langle \mathcal{S}_\mathcal{A}, \Sigma \times \mathbb{A}, \rightarrow\rangle$ by setting (and writing configurations simply as (q, ρ, H) instead of $((q, \rho), H)$):

- $\mathcal{S}_\mathcal{A} = \{(q, \rho) \mid q \in Q \wedge \rho : \mu(q) \to \mathbb{A} \text{ injective}\}$;
- $(q, \rho, H) \xrightarrow{t,a} (q', \rho', H')$ whenever there is $q \xrightarrow{t,x} q'$ such that, for some register i:
  - $x = i$ and $a = \rho(i)$, while $\rho' = \rho$; or
  - $x = i^\bullet$ and $a \notin \mathrm{rng}(\rho)$, while $\rho' = \rho[i \mapsto a] \upharpoonright \mu(q')$; or
  - $x = i^\circledast$ and $a \notin H$, while $\rho' = \rho[i \mapsto a] \upharpoonright \mu(q')$;

  where $\rho[i \mapsto a]$ is the update of ρ mapping i to a, and the operator $\upharpoonright$ performs domain restriction. In all cases, $H' = H \cup \{a\}$.

It is not difficult to verify that $\mathcal{G}_\mathcal{A}$ is indeed an FLTS. First, the sets $\mathcal{S}_\mathcal{A}$ and $\Sigma \times \mathbb{A}$ are closed under permutation, hence they are nominal sets. The definition of the transition relation does not single out any specific names and is therefore equivariant. Moreover, the names appearing in the history but not in the registers of a configuration are indistinguishable to the automaton, and that makes the conditions of strengthening and weakening true. Finally, in every transition, names in the final register assignment are sourced from those of the initial assignment and the label, and the history is correctly updated.

▶ **Lemma 7.** *Given an r-FRA $\mathcal{A} = \langle Q, \mu, \Sigma, \delta\rangle$, its FLTS $\mathcal{G}_\mathcal{A} = \langle \mathcal{S}_\mathcal{A}, \Sigma \times \mathbb{A}, \rightarrow\rangle$ is well defined.*

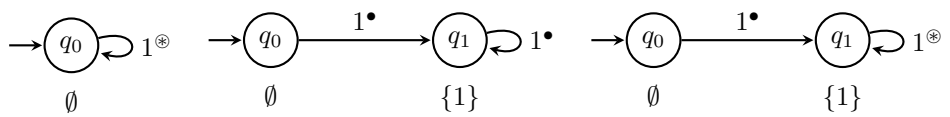
We conclude this section with some motivating examples. These include examples of properties used in the literature and simple properties one may want to check in a verification scenario.

▶ **Example 8.** Let us assume a unique tag, e.g. $\Sigma = \{\bullet\}$, which we suppress for brevity. Consider the following properties (cf. [15]):

All paths, finite and infinite, contain pairwise distinct names. (#All)

There is an infinite path where all names are distinct. (#Path)

Thus, both properties specify paths of the forms $a_1 a_2 \ldots a_n$ or $a_1 a_2 a_3 \ldots$ where, for each a_i, there is no $j < i$ such that $a_i = a_j$. Let us now examine the following basic FRAs.



The sets beneath each state indicate available registers. The paths (in the FLTS) of the first FRA will always have distinct names, so this FRA satisfies both #All and #Path. In fact, all three FRAs, from all possible configurations, satisfy #Path as the locally fresh transitions can be realised by names that are in fact globally fresh. Moreover:

- the second FRA fails #All from both states: for any H, ρ, a, b with $a\#\rho, b$ and $i = 0, 1$, $(q_i, \rho, H) \xrightarrow{a} (q_1, [1 \mapsto a], H \cup \{a\}) \xrightarrow{b} (q_1, [1 \mapsto b], H \cup \{a, b\}) \xrightarrow{a} (q_1, [1 \mapsto a], H \cup \{a, b\})$;
- the third FRA satisfies #All everywhere: even if going from q_0 to q_1 may involve an old name (already in history), the remaining names are all going to be globally fresh.

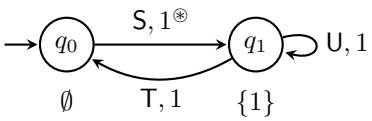
▶ **Example 9.** Let us assume a set of tags $\Sigma = \{\mathsf{S}, \mathsf{U}, \mathsf{T}\}$ which stand for start, use and terminate respectively. Consider an application in which a new session is created each time the application is launched. This session can be used an arbitrary number of times, and then the session is terminated once the application is closed. We declare the following property for this application:

There is an infinite path that repeatedly starts a fresh session, arbitrarily uses it and then terminates it. (#SUT)

Once the session is terminated, it cannot be resumed or reactivated, a new session must be created. A path for this will look as follows:

$$\mathsf{S}(s_1)\ \mathsf{U}(s_1)^*\ \mathsf{T}(s_1)\ \mathsf{S}(s_2)\ \mathsf{U}(s_2)^*\ \mathsf{T}(s_2)\ \mathsf{S}(s_3)\ \ldots$$

where, for each s_i, there is no $j < i$ such that $s_i = s_j$. Let us now consider this basic FRA below.



The FRA satisfies the property #SUT from q_0. For any H, s_1, s_2 with $s_1, s_2 \notin H$, $H' = H \cup \{s_1\}$ and $H'' = H' \cup \{s_2\}$:

$$(q_0, \emptyset, H) \xrightarrow{\mathsf{S}, s_1} (q_1, [1 \mapsto s_1], H') \xrightarrow{\mathsf{T}, s_1} (q_0, \emptyset, H') \xrightarrow{\mathsf{S}, s_2} (q_1, [1 \mapsto s_2], H'') \xrightarrow{\mathsf{T}, s_2} \cdots$$

with U, s_1 on a self-loop at $(q_1, [1 \mapsto s_1], H')$ and U, s_2 on a self-loop at $(q_1, [1 \mapsto s_2], H'')$.

▶ **Example 10.** In [44, 2] we showed that finitary π-calculus processes can be encoded as FRAs. We briefly discuss an example of a π-calculus process P that satisfies property #All:

$$P = \nu x.\bar{a}x.(P + 0)$$

The νx creates a new name (or *channel*) x, and then $\bar{a}x$ sends x on the given channel a. Following this, the process P is repeated, or the process terminates. This ensures that on each path, all names are pairwise distinct, as the ν operator ensures that each channel that is created is fresh, hence there will be no repetitions. See Appendix B for a short presentation of the π-calculus and its modelling in FLTSs.

3 Fresh HML with recursion (μFHML)

In this section we present our logic μFHML. We introduce the syntax and semantics of formulas, and show that the semantics is well defined. Recall the sets $\mathbb{A}, \Sigma$ of names and tags, where the former is ranged over by a, b, etc. It will be useful to consider tags as having arbitrary finite arities, and not just unary as in FRAs, determined by a map $\mathsf{ar} : \Sigma \to \mathbb{N}$. Let us also assume a set Var of *value variables* and a set VAR of *recursion variables*. These are ranged over respectively by x, y, etc. and X, Y, etc. Recursion variables have also arities, given by a map $\mathsf{ar} : \mathit{VAR} \to \mathbb{N}$ (note function overload). Here and in the sequel, when writing variable sequences $\vec{x}$ we shall tacitly assume they contain distinct elements.

▶ **Definition 11.** The formulas of fresh HML with recursion (μFHML) are:

$$\begin{aligned} \text{Formulas } \ni\ \phi &::= u = u \mid \phi \vee \phi \mid \neg\phi \mid \bigvee_x \phi \mid \text{И}x.\phi \mid \langle \ell \rangle \phi \mid (\mu X(\vec{x}).\phi)(\vec{u}) \mid X(\vec{u}) \\ \text{Values } \ni\ u &::= x \mid a \\ \text{Labels } \ni\ \ell &::= (t, \vec{u}) \qquad (\text{with } \mathsf{ar}(t) = |\vec{u}|) \end{aligned}$$

where in $X(\vec{u})$ and $(\mu X(\vec{x}).\phi)(\vec{u})$ we have $|\vec{u}| = |\vec{x}| = \mathsf{ar}(X)$.

A variable X can be free or bound, with binding performed with μX. As usual, we impose that each recursion variable X appears within an even number of negation operations from its binder, implying that there must be at least one appearance of every such X. Value variables x are bound by $\bigvee_x$, $\text{И}x.$ and $\mu X(\vec{x})$. We say that a formula is ***firm*** if it has no free value variables, and ***closed*** if it has no free recursion variables. Note also that formulas form a nominal set: for each formula, its support is simply the set of names featuring in it. Finally, we can express tautology e.g. by $\mathsf{true} \equiv \bigvee_x (x = x)$.

The size of a formula is calculated in such a way that it be no smaller than the size of its support, and moreover, be linearly related to a concise machine representation of the formula (e.g. using de Bruijn indices [13]).

▶ **Definition 12.** The size for each formula is inductively defined as:

$$\begin{aligned} |u = v| &= 2 & |\textstyle\bigvee_x \phi| = |\text{И}x.\phi| = |\neg\phi| &= 1 + |\phi| \\ |\langle t, \vec{u} \rangle \phi| &= 1 + |\vec{u}| + |\phi| & |X(\vec{u})| &= 1 + |\vec{u}| \\ |\phi \vee \psi| &= 1 + |\phi| + |\psi| & |(\mu X(\vec{x}).\phi)(\vec{u})| &= 1 + |\phi| + 2|\vec{u}| \end{aligned}$$

The semantics of formulas is given with respect to a given FLTS: a formula is mapped to a subset of its configurations. We shall use the following grammars for value and recursion variable substitutions respectively:

$$\gamma ::= \varepsilon \mid \gamma, \{\vec{a}/\vec{x}\} \qquad\qquad \theta ::= \varepsilon \mid \theta, \{\sigma X(\vec{x}).\phi / X\}$$

Thus, substitutions are sequences of mappings which can be on sequences of distinct value variables or single recursion variables. We may write $\mathrm{dom}(\gamma)$ for the union of the domains of all elements in γ, while in $\gamma, \{\vec{a}/\vec{x}\}$ we assume that $|\vec{a}| = |\vec{x}|$ and, for each i, $x_i \notin \mathrm{dom}(\gamma)$. A formula ϕ acted on by a substitution γ is written $\phi\{\gamma\}$ and defined recursively by:

$$\phi\{\varepsilon\} = \phi \qquad\qquad \phi\{\gamma, \{\vec{a}/\vec{x}\}\} = (\phi\{\gamma\})\{\vec{a}/\vec{x}\}$$

Similar conventions and definitions relate to recursion variable substitutions. In addition, $\phi\{\theta\}$ is only ever going to be considered in cases where no variable capture be possible.

▶ **Definition 13.** Given an FLTS $\mathcal{L} = \langle \mathcal{S}, L, \to \rangle$ with $L \subseteq \{(t, \vec{a}) \in \Sigma \times \mathbb{A}^* \mid \mathsf{ar}(t) = |\vec{a}|\}$, let us set $\mathcal{U} = \mathcal{P}(\hat{\mathcal{S}})$ and, for each $n \in \mathbb{N}$, $\mathcal{U}_n = \mathbb{A}^n \to \mathcal{U}$.
A *$\mathcal{U}$-variable assignment* is a finite partial map $\xi : \mathit{VAR} \rightharpoonup \bigcup_{n\in\mathbb{N}} \mathcal{U}_n$ such that, for each $X \in \mathrm{dom}(\xi)$, $\xi(X) \in \mathcal{U}_{\mathsf{ar}(X)}$ $(= \mathbb{A}^{\mathsf{ar}(X)} \to \mathcal{U})$. Given a $\mathcal{U}$-variable assignment ξ, the semantics $[\![\phi]\!]_\xi$ of a firm formula ϕ with respect to ξ is an element of $\mathcal{U}$ given inductively by:

$$[\![a = b]\!]_\xi = \emptyset \quad (\text{if } a \neq b) \qquad [\![\bigvee_x \phi]\!]_\xi = \bigcup_{a\in\mathbb{A}} [\![\phi\{a/x\}]\!]_\xi$$
$$[\![a = a]\!]_\xi = \hat{\mathcal{S}} \qquad [\![\mathsf{M}x.\phi]\!]_\xi = \bigcup_{a\#\phi,\xi} \{(s, H) \in [\![\phi\{a/x\}]\!]_\xi \mid a \notin H\}$$
$$[\![\phi_1 \vee \phi_2]\!]_\xi = [\![\phi_1]\!]_\xi \cup [\![\phi_2]\!]_\xi \qquad [\![X(\vec{a})]\!]_\xi = \xi(X)(\vec{a})$$
$$[\![\neg\phi]\!]_\xi = \hat{\mathcal{S}} \setminus [\![\phi]\!]_\xi \qquad [\![(\mu X(\vec{x}).\phi)(\vec{a})]\!]_\xi = (\mathsf{lfp}(\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}))(\vec{a})$$
$$[\![\langle\ell\rangle\phi]\!]_\xi = \{(s, H) \in \hat{\mathcal{S}} \mid \exists (s, H) \xrightarrow{\ell} (s', H').\, (s', H') \in [\![\phi]\!]_\xi\}$$

When ξ is empty, we may simply write $[\![\phi]\!]$.

Note that the notation $\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_\xi$ is shorthand for the function $\{(\vec{b}, [\![\phi\{\vec{b}/\vec{x}\}]\!]_\xi) \mid \vec{b} \in \mathbb{A}^n\}$. More generally, the semantics is only defined on firm formulas.

For the semantics to be well-defined, we need to ensure that the required least fixpoints exist. From the Knaster-Tarski theorem, this can be done by showing that $\mathcal{U}_n$ is a complete lattice and that $\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}$ is a monotone function. In particular, setting $F = \lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}$, we have $\mathsf{lfp}(F) = \bigsqcap\{g \in \mathcal{U}_n \mid F(g) \sqsubseteq g\}$, where order and least-upper-bounds in $\mathcal{U}_n$ are given as expected:

$$f \sqsubseteq g \iff \forall \vec{a} \in \mathbb{A}^n.\, f(\vec{a}) \subseteq g(\vec{a}) \qquad \bigsqcap S = \lambda\vec{a}.\, \bigcap\{f(\vec{a}) \mid f \in S\}$$

for any $f, g \in \mathcal{U}_n$ and $S \subseteq \mathcal{U}_n$.

▶ **Lemma 14.** *$\mathcal{U}_n$ is a complete lattice and, for any ϕ, ξ, the function $\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}$ is monotone.*

Finally, we show that for any μFHML formula ϕ and $\mathcal{U}$-variable assignment ξ, $\mathrm{supp}([\![\phi]\!]_\xi) \subseteq \mathrm{supp}(\phi) \cup \mathrm{supp}(\xi)$. To do this, we show that the function $\lambda\phi, \xi.[\![\phi]\!]_\xi$ is equivariant, and then use the equivariance principle.

▶ **Lemma 15.** *The function $\lambda\phi, \xi.[\![\phi]\!]_\xi$ is equivariant. Therefore, for any formula ϕ and $\mathcal{U}$-variable assignment ξ, $\mathrm{supp}([\![\phi]\!]_\xi) \subseteq \mathrm{supp}(\phi) \cup \mathrm{supp}(\xi)$.*

A direct consequence of the latter result is that $[\![\phi]\!]_\xi$ is finitely supported, and so is the function $\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}$.

▶ **Theorem 16.** *The semantics of Definition 13 is well defined. Its image is within the subset of $\mathcal{U}$ of finitely-supported sets of configurations.*

Proof. For the former, we must show that least fixpoints exist for any $\mathsf{lfp}(\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]})$. By definition, the function $F = \lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}$ is in $\mathcal{U}_{\mathsf{ar}(X)}$ and by Lemma 14, $\mathcal{U}_{\mathsf{ar}(X)}$ is a complete lattice and F is monotone. Thus, by the Knaster-Tarski theorem [43], the least fixpoint of any such F exists. The proof of the latter follows from Lemma 15. ◀

▶ Remark 17 (Derived connectives). We can define standard dual connectives for μFHML as:

$$\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi) \qquad [\ell]\phi \equiv \neg\langle\ell\rangle\neg\phi \qquad (\nu X(\vec{x}).\phi)(\vec{u}) \equiv \neg(\mu X(\vec{x}).\neg\phi\{\neg X/X\})(\vec{u})$$

and write $u \neq v$ for $\neg(u = v)$. The semantics of the above can be derived from Definition 13 as expected. For instance: $[\![(\nu X(\vec{x}).\phi)(\vec{a})]\!]_\xi = (\mathsf{gfp}(\lambda f.\lambda \vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}))(\vec{a})$.
In the same vein, we show below that fresh selection is its self-dual: $[\![\neg И x.\phi]\!]_\xi = [\![И x.\neg\phi]\!]_\xi$.

As in [19, 39], we can show that $И x.\phi$ can be taken as meaning "for any fresh x, ϕ holds" or "there is some fresh x such that ϕ holds".

▶ **Lemma 18.** *For any ϕ, ξ and configuration (s, H),*

$$(s,H) \in [\![И x.\phi]\!]_\xi \iff \exists a\#\phi,\xi,H.\,(s,H) \in [\![\phi\{a/x\}]\!]_\xi \quad (1)$$
$$\iff \forall a\#\phi,\xi,H.\,(s,H) \in [\![\phi\{a/x\}]\!]_\xi \quad (2)$$

Therefore, we have $[\![\neg И x.\phi]\!]_\xi = [\![И x.\neg\phi]\!]_\xi$.

Proof. We note that (1) is a restatement of the definition of the semantics for $И x.\phi$. For (2) it suffices to show the left-to-right inclusion, as the other one follows from there always being some $a\#\phi,\xi,H$. For any (s, H):

$$\begin{aligned}(s,H) \in [\![\neg И x.\phi]\!]_\xi &\iff \exists a\#\phi,\xi,H.\,(s,H) \in [\![\phi\{a/x\}]\!]_\xi \\ &\implies \exists a\#\phi,\xi,H.\forall b\#\phi,\xi,H.\,(s,H) = (a\ b)\cdot(s,H) \in [\![\phi\{b/x\}]\!]_\xi \\ &\implies \forall b\#\phi,\xi,H.\,(s,H) \in [\![\phi\{b/x\}]\!]_\xi\end{aligned}$$

The final claim then follows using (1) and (2). ◀

▶ Remark 19 (Comparison with logics of Dam [12] and Klin et al. [27, 15]). The π-μ-calculus is a recursive modal logic for π-calculus, accompanied with a proof system that is sound and complete for finitary processes [12]. Its connectives are very similar to ours and, in fact, our syntax for the recursive connectives is taken from *loc cit.* Being a logic specifically for the π-calculus, it employs connectives such as free/bound input and output, in addition to modal connectives that specify the communication channel. Bound I/O in particular makes fresh-name quantification redundant.
The μ-calculi with atoms of [27, 15] are probably the closest to our logic. While those logics only feature recursion variables of nullary arities, the vectorial version studied in [27] can capture scenarios where non-zero arities are needed. The history-dependent logic of [15] captures freshness with this connective and corresponding semantics (in our terminology):

$$\phi ::= \cdots \mid \#v \quad \text{(for value } v\text{)} \qquad\qquad [\![\#a]\!]_\xi = \{(s,H) \mid a \notin H\}$$

The analogy is not fully exact as our semantics is over history-dependent LTSs, while in *loc cit.* on nominal LTSs. In fact, the approaches are incomparable: fresh-orbit-finite LTSs (where model checking is decidable) are more general than the orbit-finite LTSs of [15], but our logic is less general than theirs. Using $\#u$, we can define fresh quantification:

$$И x.\phi \equiv \bigvee\nolimits_x \#x \wedge (x \neq v_1) \wedge \cdots \wedge (x \neq v_n) \wedge \phi$$

where $v_1, \ldots, v_n$ is an enumeration of all (free) values in ϕ. But one can be even more expressive. E.g. the following formula stipulates that there are exactly two names in the current history:

$$\bigvee\nolimits_{x,y} x \neq y \wedge \neg(\#x \vee \#y) \wedge \bigwedge\nolimits_z z \neq x, y \implies \#z$$

We find this added expressivity not necessary for specifying FLTSs built e.g. from FRAs, as the exact size of the history is in general not accessible. Another property expressible with the fresh-name connective is (we thank the anonymous reviewer for providing this):

Every next transition is on a fresh name. (#AllNext)

Assuming $\Sigma = \{\bullet\}$ and suppressing labels, this can be expressed by $\bigwedge_x [x] \# x$. One may wish to use #AllNext to specify e.g. a name generator that produces fresh names independently of its current history (for a given history H, #AllNext can be expressed by simply enumerating the names in H). Being able to express #AllNext, say by a formula $\phi_{[\#]}$, would again allow us to pry too invasively into the history. For example, the formula

$$\phi_{[\#]} \wedge \bigvee_{x,y} x \neq y \wedge \neg(\langle x \rangle \mathsf{true} \vee \langle y \rangle \mathsf{true}) \wedge \bigwedge_z z \neq x, y \implies \langle z \rangle \mathsf{true}$$

stipulates that #AllNext holds and there are exactly two names in the history. Such history-counting properties seem beyond the reach of μFHML.
Finally, the history-bounded parity games presented herein provide a simpler route to model checking than the history-bounding construction in [15]. It is interesting to see if our approach can also be applied in the presence of the $\#v$ connective. We conjecture that it can, and that would render the model-checking problem decidable in that case as well.

By reduction from satisfiability for the μ-calculus with atoms [27] (i.e. even without histories), we know that the satisfiability problem is undecidable for μFHML. In the next section we shall study instead the model-checking problem, show it is decidable and calculate an upper bound for its complexity. The following definition will be of use then.

▶ **Definition 20.** Given a μFHML formula ϕ, its binding depth $\|\phi\|$ is given recursively by:

$$\|u = v\| = \|X(\vec{u})\| = 0 \qquad \|\neg\phi\| = \|\langle \ell \rangle \phi\| = \|\phi\| \qquad \|(\mu X(\vec{x}).\phi)(\vec{u})\| = \|\phi\| + |\vec{x}|$$
$$\|\phi \vee \psi\| = \max(\|\phi\|, \|\psi\|) \quad \|\bigvee_x \phi\| = \|\mathcal{N}x.\phi\| = 1 + \|\phi\|$$

We conclude this section by revisiting Examples 8 and 9 and building formulas for them.

▶ **Example 21.** The property #All can be represented by the formula $\neg\phi$, where:

$$\phi = \mu X. \bigvee_x \langle x \rangle (X \vee \mu Y. \bigvee_y \langle y \rangle (Y \vee x = y))$$

Note we omit tags as they are not used here. The subformula $\mu Y. \bigvee_y \langle y \rangle (Y \vee x = y)$ represents all finite paths that eventually accept an x, that is, for each call of Y we accept any name y and either we repeat Y or, in case $x = y$, we stop. Similarly, ϕ represents all finite paths where eventually, some a is repeated, that is, for each call of X we accept any name x and either we repeat X, or x is eventually accepted again (using $\mu Y. \bigvee_y \langle y \rangle (Y \vee x = y)$). Thus, $\neg\phi$ represents all paths such that no name is ever repeated. Note that, by negation elimination, $\neg\phi$ can be written as: $\nu X. \bigwedge_x [x](X \wedge \nu Y. \bigwedge_y [y](Y \wedge x \neq y))$.

Additionally, the property #Path can be represented by the formula:

$$\phi' = \nu X. \mathcal{N}x. \langle x \rangle X$$

The formula states that the recursive property X always holds, where property X states that we select a fresh name x, and perform an x action where we can then repeat property X.

▶ **Example 22.** Let us consider Example 9, where we start, repeatedly use, and then terminate a session that cannot be resumed or reactivated. This can be represented by:

$$\psi = \nu X.И s.\langle \mathsf{S}, s\rangle.\mu Y.(\langle \mathsf{U}, s\rangle Y \vee \langle \mathsf{T}, s\rangle X)$$

where $\mathsf{S}, \mathsf{U}, \mathsf{T}$ stand for start, use and terminate respectively. We begin by entering the recursive ν-variable X, where each time we enter X, we select a fresh session s and perform the action $\mathsf{start}(s)$. From here, we enter the μ-variable Y where each time we enter, we repeatedly perform the action $\mathsf{use}(s)$ and eventually perform the action $\mathsf{terminate}(s)$. Once s has been terminated, we repeat the process by re-entering the ν-variable X, starting some new session s'. Note that the use of the μ-variable Y ensures that the session will eventually terminate, and the use of the ν-variable X ensures that the property is never violated.

4 Parity games and model checking

Given an FLTS L, a configuration (s_0, H_0) in L and a firm closed formula ϕ_0, the *model-checking* problem asks whether $(s_0, H_0) \in [\![\phi_0]\!]$. To solve this problem, in this section we introduce parity games that capture satisfiability of formulas in given configurations. To establish this connection it will be useful to consider formulas ϕ_0 which may not be closed. We start off with a general definition of parity games in nominal sets. These are essentially *atomic parity games* [27] though, in contrast to *loc. cit.*, here we use them for games with bounded histories.

▶ **Definition 23.** A ***(nominal) parity game*** is a tuple $\mathcal{G} = \langle V, V_A, V_D, E, \Omega\rangle$ where $\langle V, E\rangle$ is a directed graph, with vertices called *positions* and edges called *moves*, $V = V_A \uplus V_D$ is a partitioning of positions into *Attacker* and *Defender* ones respectively, and $\Omega : V \to \{0, \ldots, d\}$ is a *ranking function* for some maximum rank $d \in \mathbb{N}$.
In addition, V is a finitely supported subset of some nominal set $\hat{V}$ and, for all $P_1, P_2 \in V$ and $P_1', P_2' \in \hat{V}$, if $(P_1, P_1') \sim_{\mathsf{nom}} (P_2, P_2')$ then:
- if $P_1 \in V_\chi$ then $P_2 \in V_\chi$ (for $\chi \in \{A, D\}$), and $\Omega(P_1) = \Omega(P_2)$;
- if $(P_1, P_1') \in E$ then $(P_2, P_2') \in E$.

Given a game $\mathcal{G}$, we may write $\mathrm{Pos}(\mathcal{G})$ for its set of positions V. Intuitively, the nominal conditions in Definition 23 say that the game seen from two nominally equivalent positions is the same up to permutation. Note that $\mathcal{G}$ is not equivariant in general, and its support is the support of V.

Starting from a root position $P_0 \in V$, a *play* is a path within $\mathcal{G}$, i.e. a finite or infinite sequence $\Pi = P_0, P_1, \ldots$ such that $(P_i, P_{i+1}) \in E$ for each i. A play is *complete* if it ends in a leaf. Defender (Attacker) wins a complete play if the last position in it belongs to Attacker (resp. Defender). Defender (Attacker) wins an infinite play if the highest position rank in it is even (resp. odd). We say that P_0 is a *winning position* for Defender (or Defender *wins from* P_0) if there is a partial function $\Theta : V_D \rightharpoonup E$ (called a *strategy*) such that Defender wins every complete or infinite play $P_0, P_1, \ldots$ such that for each $P_i \in V_D$ we have $\Theta(P_i) = (P_i, P_{i+1})$.

The development of parity games for μFHML follows parity games for the μ-calculus, cf. [16]. We start by considering the negation-free fragment of our logic. In the sequel, we shall consider the derived connectives of Remark 17, along with inequality $u \neq v$, as primitive (and exclude negation).

▶ **Definition 24.** A formula is called *negation-free* if it contains no negations. For each formula ϕ, we write $!\phi$ for its negation-free variant. This can be constructed by induction using standard duality rules for pushing negation inside boolean, modal and recursion constructors, along with the rules: $!(\neg\neg\phi) = !\phi$, $!(\neg И x.\phi) = И x.!(\neg\phi)$.

▶ **Lemma 25.** *For any formula ϕ, $|!(\phi)|$ is bounded by $|\phi| - n$, where n is the number of negations in ϕ.*

For the remainder of this section, and unless specified otherwise, any μFHML formulas used are going to be assumed to be firm and negation-free.

We shall make use of the (recursion) *alternation depth* of formulas. At this point it is important that the formulas we examine have a unique specification for each of their bound variables. More specifically, for any formula ϕ we stipulate that:

- the bound and free recursion variables of ϕ be disjoint;
- if $\sigma_1 X(\vec{x}_1).\phi_1, \sigma_2 X(\vec{x}_2).\phi_2$ are subformulas of ϕ then $(\sigma_1, \vec{x}_1, \phi_1) = (\sigma_2, \vec{x}_2, \phi_2)$.

Note that these conditions are not unusual (cf. [26]). They are not restricting our logic either as any formula has an α-equivalent one in the form above. At the same time, they allow us to match bound recursion variables with their binding subformulas.

For value variables, on the other hand, we simply disallow a binder on a variable x to be within the scope of another binder on x.

▶ **Definition 26** ([9, 17])**.** Let ϕ be a μFHML formula. We define the *dependency order* on bound variables of ϕ as the smallest partial order $\leq_\phi$ such that $X \leq_\phi Y$ if $X(\vec{u})$ occurs free in some ϕ' such that $\sigma Y(\vec{x}).\phi'$ appears in ϕ (for some $\vec{u}$).
The ***alternation depth*** of a bound variable X in ϕ is defined as the maximal length of a chain $X = X_1 \leq_\phi \cdots \leq_\phi X_n$ such that:

- if X is a μ-variable, each X_i is a μ-variable if i is odd, and a ν-variable otherwise;
- if X is a ν-variable, each X_i is a ν-variables if i is odd, and a μ-variable otherwise.

The alternation depth of a formula ϕ (denoted *adepth(ϕ)*) is the maximum alternation depth of variables bound in ϕ, or zero if there are no fixpoints.

Parity games for μFHML have positions of the form (s, H, ξ, ψ), with:

- $(s, H) \in \hat{\mathcal{S}}$ being the current configuration that is examined;
- ξ being a $\mathcal{U}$-variable assignment;
- ψ being the current formula.

We define games with respect to a *root position* corresponding to the model-checking problem we aim to decide. The positions and moves are given inductively using *game rules*.

▶ **Definition 27.** Let ϕ_0 be a firm negation-free μFHML formula, ξ a $\mathcal{U}$-variable assignment containing the free recursion variables of ϕ_0, $\mathcal{L}$ an FLTS and (s_0, H_0) a configuration in $\mathcal{L}$.
The ***game*** $\mathcal{G}(\mathcal{L}, \phi_0, \xi, s_0, H_0)$ has root position (s_0, H_0, ξ, ϕ_0) and game rules:

- $(s, H, \xi, a = a)$ belongs to Attacker, whereas $(s, H, \xi, a = b)$ to Defender;
- $(s, H, a \neq a)$ belongs to Defender, whereas $(s, H, a \neq b)$ to Attacker;
- $(s, H, \xi, X(\vec{a}))$ belongs to Attacker if $(s, H) \in \xi(X)(\vec{a})$, and to Defender otherwise;
- $(s, H, \xi, \textcircled{\ell}\phi) \to (s', H', \xi, \phi)$, if $(s, H) \xrightarrow{\ell} (s', H')$, belongs to Defender if $\textcircled{\ell} = \langle\ell\rangle$, and to Attacker if $\textcircled{\ell} = [\ell]$;
- $(s, H, \xi, \phi_1 \odot \phi_2) \to (s, H, \xi, \phi_i)$, $i \in \{1, 2\}$, belongs to Defender if $\odot = \vee$, and to Attacker if $\odot = \wedge$;
- $(s, H, \xi, \bigodot_x \phi) \to (s, H, \xi, \phi\{a/x\})$, $a \in \mathbb{A}$, belongs to Defender if $\bigodot = \bigvee$, and to Attacker if $\bigodot = \bigwedge$;
- $(s, H, \xi, \text{Ⱶ}x.\phi) \to (s, H, \xi, \phi\{a/x\})$, $a\#\phi, \xi, H$, belongs to either;
- $(s, H, \xi, (\sigma X(\vec{x}).\phi)(\vec{a})) \to (s, H, \xi, (\phi\{\sigma X(\vec{x}).\phi/X\})\{\vec{a}/\vec{x}\})$, $\sigma \in \{\mu, \nu\}$, belongs to either.

The rank of a position with formula ϕ is given by $\Omega(\phi)$:

$$\Omega(\phi) = \begin{cases} 2 \cdot \lfloor adepth(X)/2 \rfloor & \text{if } \phi \text{ is of the form } (\nu X(\vec{x}).\phi')(\vec{a}) \\ 2 \cdot \lfloor adepth(X)/2 \rfloor + 1 & \text{if } \phi \text{ is of the form } (\mu X(\vec{x}).\phi')(\vec{a}) \\ 0 & \text{otherwise} \end{cases}$$

In the case when ϕ_0 is closed, we denote the game simply by $\mathcal{G}(\mathcal{L}, \phi_0, s_0, H_0)$.

Note that the formulas appearing inside positions are firm and negation-free (as is ϕ_0 by assumption).

Later in this section it will be important to count the orbits of a history-bounded version of the parity game of Definition 27. To that effect, it is useful to the formulas appearing in the game, in terms of subformulas of ϕ_0 and substitutions. This is achieved by the following notion of formula closure.

▶ **Definition 28.** Given a μFHML formula ϕ_0, we define its closure $\mathsf{clos}(\phi_0)$ to be the smallest set χ of triples (ϕ, γ, θ) such that $(\phi_0, \varepsilon, \varepsilon) \in \chi$ and:

- if $(\phi_1 \odot \phi_2, \gamma, \theta) \in \chi$, with $\odot \in \{\vee, \wedge\}$, then $(\phi_1, \gamma, \theta), (\phi_2, \gamma, \theta) \in \chi$;
- if $(\ell\!\!\rangle\phi, \gamma, \theta) \in \chi$, with $\ell\!\!\rangle \in \{\langle\ell\rangle, [\ell]\}$, then $(\phi, \gamma, \theta) \in \chi$;
- if $(\bigodot_x \phi, \gamma, \theta) \in \chi$, with $\bigodot \in \{\bigvee, \bigwedge\}$, then $(\phi, (\gamma, \{a/x\}), \theta) \in \chi$, for all $a \in \mathbb{A}$;
- if $(\mathit{N}x.\phi, \gamma, \theta) \in \chi$ then $(\phi, (\gamma, \{a/x\}), \theta) \in \chi$, for all $a \# \phi\{\theta\}\{\gamma\}$;
- if $((\sigma X(\vec{x}).\phi)(\vec{u}), \gamma, \theta) \in \chi$, with $\sigma \in \{\mu, \nu\}$, then $(\phi, (\gamma, \{\vec{c}/\vec{x}\}), (\theta, \{\sigma X(\vec{x}).\phi/X\})) \in \chi$, for all $\vec{c} \in \mathbb{A}^{\mathsf{ar}(X)}$.

We write $\mathsf{clos}^*(\phi_0) = \{\phi\{\theta\}\{\gamma\} \mid (\phi, \gamma, \theta) \in \mathsf{clos}(\phi_0)\}$.

The following lemma shows that the formulas appearing in the game $\mathcal{G}(\mathcal{L}, \phi_0, s_0, H_0)$ can be traced back to the closure. It moreover provides some bounds on their orbits and supports.

▶ **Lemma 29.** *Given a closed formula ϕ_0, an FLTS $\mathcal{L}$ and configuration (s_0, H_0):*

1. *for all positions (s, H, ϕ) in $\mathcal{G}(\mathcal{L}, \phi_0, s_0, H_0)$ we have that $\phi \in \mathsf{clos}^*(\phi_0)$;*
2. *$|\mathcal{O}(\mathsf{clos}^*(\phi_0))|$ is bounded by $|\phi_0| \cdot \frac{(|\operatorname{supp}(\phi_0)| + \|\phi_0\|)!}{|\operatorname{supp}(\phi_0)|!}$;*
3. *for all $(\phi, \gamma, \theta) \in \mathsf{clos}(\phi)$, $|\operatorname{supp}(\phi, \gamma, \theta)| \leq |\operatorname{supp}(\phi_0)| + \|\phi_0\|$.*

We proceed to showing the correspondence between the semantics of an μFHML formula and its corresponding parity game.

▶ **Theorem 30.** *Given a μFHML formula ϕ_0, $\mathcal{U}$-variable assignment ξ, FLTS $\mathcal{L}$ and (s_0, H_0) from $\mathcal{L}$, we have $(s_0, H_0) \in [\![\phi_0]\!]_\xi$ iff Defender wins from (s_0, H_0, ξ, ϕ_0) in $\mathcal{G}(\mathcal{L}, \phi_0, \xi, s_0, H_0)$.*

We have thus reduced model checking of formulas to winning in parity games. At this stage, we can restrict our attention to closed (firm negation-free) formulas ϕ_0. The next step is deciding winning regions of our parity games. The latter requires the games to be finite, whereas ours are not so. We therefore reduce our parity games to equivalent finite ones. For this to be possible, we restrict our attention to FLTSs that are fresh-orbit-finite.

For brevity, in the remainder of this section we shall say that $(\mathcal{L}, \phi_0, s_0, H_0)$ is a ***setup of grade*** N if:

- $\mathcal{L}$ is a fresh-orbit-finite FLTS with set of states $\mathcal{S}$, and $(s_0, H_0) \in \hat{\mathcal{S}}$;
- ϕ_0 is a closed (firm negation-free) μFHML formula;
- $N = |\operatorname{supp}(\phi_0)| + \|\phi_0\| + r_i(\mathcal{S})$.

Given a setup $(\mathcal{L}, \phi_0, s_0, H_0)$ of grade N, we first present a version of $\mathcal{G}(\mathcal{L}, \phi_0, s_0, H_0)$ with bounded histories. The idea is to restrict the size of the history to the maximum number of names that are available to the formula and the states of the FLTS, and one extra to represent a fresh name. This amounts to a bound of $N+1$ names. When a new name is added and the size of the history is greater than N, then names that are in the history but no longer present in the formula are forgotten. Each position in this game is a tuple (s, H, ϕ) with $|H| \leq N+1$.

▶ **Definition 31.** Let $(\mathcal{L}, \phi_0, s_0, H_0)$ be a setup of grade N. The ***history-bounded game*** $\mathcal{G}_N(\mathcal{L}, \phi_0, s_0, H_0)$ has root position (s_0, H_0', ϕ_0), for some $H_0' \lhd_N (s_0, \phi_0, H_0)$, and game rules as in Definition 27, apart from the rule for modal connectives:

- $(s, H, ⓛ\phi) \to (s', H', \phi)$, if $\ell = (t, \vec{a})$ and $(s, H) \xrightarrow{t,\vec{a}} (s', H \cup \{\vec{a}\})$ with $H' \lhd_N (s', \phi, H \cup \{\vec{a}\})$, played by Defender if $ⓛ = \langle \ell \rangle$, and by Attacker if $ⓛ = [\ell]$;

while the ranking function is the same as in Definition 27. The *well-bounding* relation $\lhd$ is given as follows. We let $H' \lhd_N (s, \phi, H)$ if $H' \subseteq H$ and:

- if $|H| \leq N$ then $H' = H$,
- else $H \cap \mathrm{supp}(s, \phi) \subseteq H' \subseteq H$ and $|H'| = N+1$.

We note that, though the positions in the unbounded game have their histories trimmed at modal operators, they remain FLTS configurations, i.e. for each (s, H, ϕ) we have $\mathrm{supp}(s) \subseteq H$. We next show that we can decide winning a parity game $\mathcal{G}$ by examining its equivalent history-bounded game $\mathcal{G}_N$. To do this, we are going to build a bisimulation-like relation $\mathcal{R}$ between $\mathrm{Pos}(\mathcal{G})$ and $\mathrm{Pos}(\mathcal{G}_N)$. Since the positions in the two games can have name-mismatches, but essentially do "the same" transitions, it is useful to introduce a notion of bisimilation up to renaming. Note we use relation composition: $R_1; R_2 = \{(x, z) \mid \exists y.\, (x, y) \in R_1 \wedge (y, z) \in R_2\}$.

▶ **Definition 32.** Given parity games $\mathcal{G}_1, \mathcal{G}_2$, we call a relation $R \subseteq \mathrm{Pos}(\mathcal{G}_1) \times \mathrm{Pos}(\mathcal{G}_2)$ a *bisimulation* if for all $(P_1, P_2) \in R$:

- P_1 has the same rank and belongs to the same player as P_2;
- for each $P_1 \to P_1'$, there exist some $P_2 \to P_2'$ such that $(P_1', P_2') \in R$;
- for each $P_2 \to P_2'$, there exist some $P_1 \to P_1'$ such that $(P_1', P_2') \in R$.

We say that P_1 and P_2 are *bisimilar* (denoted $P_1 \approx P_2$) if there is a bisimulation R such that $(P_1, P_2) \in R$.

A *bisimulation up to nominal equivalence (nom-bisimulation)* is defined to be a relation R with the same properties as above, apart from the requirement in the last two bullet points being instead that $(P_1', P_2') \in {\sim_{\mathsf{nom}}}; R; {\sim_{\mathsf{nom}}}$. We say that P_1 and P_2 are *nom-bisimilar* (denoted $P_1 \approx_{\mathsf{nom}} P_2$) if there is a nom-bisimulation R such that $(P_1, P_2) \in R$.

In the following two lemmas, we let P_1, P_2 be positions in games $\mathcal{G}_1, \mathcal{G}_2$ respectively.

▶ **Lemma 33.** *If $P_1 \approx_{\mathsf{nom}} P_2$ then $P_1 \approx P_2$.*

Proof. Let $R \subseteq \mathrm{Pos}(\mathcal{G}_1) \times \mathrm{Pos}(\mathcal{G}_2)$ be a nom-bisimulation. We show that $R' = ({\sim_{\mathsf{nom}}}; R; {\sim_{\mathsf{nom}}}) \cap (\mathrm{Pos}(\mathcal{G}_1) \times \mathrm{Pos}(\mathcal{G}_2))$ is a bisimulation. Suppose $(P_1, P_2) \in R'$, i.e. $\pi_i \cdot P_i = Q_i$ $(i = 1, 2)$ and $(Q_1, Q_2) \in R$, and let $P_1 \to P_1'$. We have $P_1, Q_1 \in \mathrm{Pos}(\mathcal{G}_1)$ and, taking $Q_1' = \pi_1 \cdot P_1'$, $(P_1, P_1') \sim_{\mathsf{nom}} (Q_1, Q_1')$, hence $Q_1 \to Q_1'$ (and $Q_1' \in \mathrm{Pos}(\mathcal{G}_1)$). Then, by nom-bisimulation, $Q_2 \to Q_2'$ (and $Q_2' \in \mathrm{Pos}(\mathcal{G}_2)$) and $(Q_1', Q_2') \in R'$. Like before, setting $P_2' = \pi_2^{-1} \cdot Q_2'$, we obtain $P_2 \to P_2'$ (and $P_2' \in \mathrm{Pos}(\mathcal{G}_2)$). We observe that $(P_1', P_2') \in ({\sim_{\mathsf{nom}}}; R'; {\sim_{\mathsf{nom}}}) \cap (\mathrm{Pos}(\mathcal{G}_1) \times \mathrm{Pos}(\mathcal{G}_2)) = R'$. ◀

▶ **Lemma 34.** *If $P_1 \approx P_2$ then Defender wins from P_1 iff they win from P_2.*

We can relate games with their history-bounded counterparts by nom-bisimulations.

▶ **Definition 35.** Given a setup $(\mathcal{L}, \phi_0, s_0, H_0)$ of grade N, consider $\mathcal{G}(\mathcal{L}, \phi_0, s_0, H_0)$ and $\mathcal{G}_N(\mathcal{L}, \phi_0, s_0, H_0)$. We define the *history-bounding relation* $\mathcal{R}_{(\mathcal{L},\phi_0,s_0,H_0)} \subseteq \mathrm{Pos}(\mathcal{G}) \times \mathrm{Pos}(\mathcal{G}_N)$, by setting $((s_1, H_1, \phi_1), (s_2, H_2, \phi_2)) \in \mathcal{R}$ if $s_1 = s_2$, $\phi_1 = \phi_2$ and $H_2 \lhd_N (s_1, \phi_1, H_1)$.

Thus, a position (s, H, ϕ) in the unbounded game is related to itself if its history H is small, i.e. it has no more than N names. If $|H|$ is larger, then we relate (s, H, ϕ) to each (s, H', ϕ) where $H' \subseteq H$ has $N + 1$ names and in particular contains $\mathrm{supp}(s)$ (as (s, H') is a configuration) and all names from $\mathrm{supp}(\phi)$ that appear in H.

▶ **Lemma 36.** *$\mathcal{R}_{(\mathcal{L},\phi_0,s_0,H_0)}$ is a nom-bisimulation.*

Combining Theorem 30 with Lemmas 33, 34, and 36, we can reduce μFHML model-checking to deciding winning a history-bounded parity game. However, although the size of histories, and therefore of positions, is now bounded, the number of positions can still be infinite due to the names appearing in them. We can resolve this by grouping together positions that are "equivalent" up to permuting their names. Formally, this is done by considering games in which positions are orbits of ordinary positions.

▶ **Definition 37.** Given a parity game $\mathcal{G} = \langle V, V_A, V_D, E, \Omega \rangle$, we construct the ***orbits game*** $\mathsf{Orbs}(\mathcal{G}) = \langle V', V'_A, V'_D, E', \Omega' \rangle$ as follows:

- $V' = \{\mathbb{O}(P) \mid P \in V\}$ and $V'_\chi = \{\mathbb{O}(P) \mid P \in V_\chi\}$ (for $\chi \in \{A, D\}$);
- $E' = \{(\mathbb{O}(P), \mathbb{O}(P')) \mid (P, P') \in E\}$;
- for each $P \in V$, $\Omega'(\mathbb{O}(P)) = \Omega(P)$.

We can see that $\mathsf{Orbs}(\mathcal{G})$ is well defined. In particular, Ω' is well defined due to the fact that $\Omega(P_1) = \Omega(P_2)$ whenever $P_1 \sim_{\mathsf{nom}} P_2$. Moreover, we can show that the relation $R = \{(P, \mathbb{O}(P)) \mid P \in \mathrm{Pos}(\mathcal{G})\}$ is a bisimulation.

▶ **Lemma 38.** *Given $\mathcal{G}$ as above and $P \in \mathrm{Pos}(\mathcal{G})$, $P \approx \mathbb{O}(P)$.*

Taking stock of the game transformations defined above and the bisimulation relations between them, we can decide winning in a setup of finite grade.

▶ **Corollary 39.** *Given a setup $(\mathcal{L}, \phi_0, s_0, H_0)$ of grade N, Defender wins in $\mathcal{G}(\mathcal{L}, \phi_0, s_0, H_0)$ (from (s_0, H_0, ϕ_0)) iff they win in $\mathsf{Orbs}(\mathcal{G}_N(\mathcal{L}, \phi_0, s_0, H_0))$ from $\mathbb{O}(s_0, H'_0, \phi_0)$. Hence, it is decidable whether $(s_0, H_0) \in [\![\phi_0]\!]$.*

Proof. The first part follows from Theorem 30 and Lemmas 33, 34, 36, and 38. The second part follows from Theorem 30 and the fact that $\mathsf{Orbs}(\mathcal{G}_N(\mathcal{L}, \phi_0, s_0, H_0))$ is a finite game. ◀

We next calculate upper bounds for the size of $\mathsf{Orbs}(\mathcal{G}_N(\mathcal{L}, \phi_0, s_0, H_0))$ and the time needed to construct it. This allows us to give a complexity bound for model checking. Recall that we set $N = |\mathrm{supp}(\phi_0)| + \|\phi_0\| + r_i(\mathcal{S})$. In the calculations below we prefer to distinguish between complexity due to ϕ_0 and $\mathcal{L}$, so instead of N we shall use the measure: $M(\phi_0) = |\mathrm{supp}(\phi_0)| + \|\phi_0\|$. This can be seen as the *nominal potential* of ϕ_0.

In constructing the orbits game, we need to be able to check if two positions of $\mathcal{G}_N$ are nominally equivalent. The way this is performed is the following: given $(s_1, H_1, \phi_1), (s_2, H_2, \phi_2)$, we try to build a permutation π such that $\pi \cdot s_1 = s_2$ and, if successful, we extend π to some π' so that, in addition, $\pi' \cdot \phi_1 = \phi_2$. The former part requires some oracle that is able to answer such questions for the given nominal set of states $\mathcal{S}$. We let a *permutation oracle* $\Psi_{\mathcal{L}}$ to be a function that, given $s, s' \in \mathcal{S}$, it returns a permutation $(\vec{a} \mapsto \vec{b})$ such that

$\{\vec{a}\} = \mathrm{supp}(s)$, $\{\vec{b}\} = \mathrm{supp}(s')$ and $(\vec{a} \mapsto \vec{b}) \cdot s = s'$, or NO if no such permutation exists (i.e. if $s \not\sim_{\mathsf{nom}} s'$). We stipulate that the time complexity of $\Psi_{\mathcal{L}}(s, s')$ is uniformly bounded by some $\Phi_{\mathcal{L}}$ for all $s, s' \in \mathcal{S}$.

Finally, given a configuration (s, H) and a label ℓ, we shall require the next configurations (s', H') under ℓ. While H' is completely determined from H', ℓ, the number of target states s' and the time complexity to retrieve them will depend on $\mathcal{L}$. We assume that both these quantities are bounded by the *non-deterministic branching factor* $\mathsf{ndb}_{\mathcal{L}}$ of $\mathcal{L}$.

▶ **Lemma 40.** *Given a setup $(\mathcal{L}, \phi_0, s_0, H_0)$ of grade N and $\mathcal{G}_N = \mathcal{G}_N(\mathcal{L}, \phi_0, s_0, H_0)$:*

1. *the size of* $\mathrm{Pos}(\mathsf{Orbs}(\mathcal{G}_N))$ *is bounded by* $|\mathcal{O}(\mathcal{S})| \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0) + r_i(\mathcal{S}) + 1)^{M(\phi_0)+1} \cdot (1 + o(1))$.
2. *the time complexity of constructing* $\mathsf{Orbs}(\mathcal{G}_N)$ *is in* $O((|\phi_0| + \Phi_{\mathcal{L}} + r_i(\mathcal{S})) \cdot \max(M(\phi_0) + r_i(\mathcal{S}), \mathsf{ndb}_{\mathcal{L}}) \cdot (|\mathcal{O}(\mathcal{S})| \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0) + r_i(\mathcal{S}) + 1)^{M(\phi_0)+1})^2)$.

Proof. Due to lack of space, we only show 1 here. By definition and Lemma 29(1), we know that $\mathrm{Pos}(\mathcal{G}_N) \subseteq \hat{\mathcal{S}} \times \mathsf{clos}^*(\phi_0)$. Then, by Lemma 3(1,2) we have that $\mathrm{Pos}(\mathsf{Orbs}(\mathcal{G}_N)) \subseteq \mathcal{O}(\mathcal{C}\ell(\hat{\mathcal{S}} \times \mathsf{clos}^*(\phi_0))) = \mathcal{O}(\hat{\mathcal{S}} \times \mathcal{C}\ell(\mathsf{clos}^*(\phi_0)))$. For every $(s, H, \phi) \in \mathcal{G}_N$, we have $\mathrm{supp}(s) \subseteq H$ and $|H| \leq N+1$, and so we have $|\mathcal{O}(\hat{\mathcal{S}})| \leq |\mathcal{O}(\mathcal{S})| \cdot (N+1)$ (what matters of H is the size of $|H \setminus \mathrm{supp}(\phi)|$). Using Lemma 3(3) and the fact that the largest support sizes in $\hat{\mathcal{S}}$ and $\mathsf{clos}^*(\phi_0)$ are bounded by $N+1$ and $|\mathrm{supp}(\phi_0)| + \|\phi_0\|$ (by Lemma 29(3)) respectively:

$$\begin{aligned}
|\mathcal{O}(\mathcal{C}\ell(\hat{\mathcal{S}} \times \mathsf{clos}^*(\phi_0))| &\leq |\mathcal{O}(\mathcal{S})| \cdot (N+1) \cdot |\mathcal{O}(\mathcal{C}\ell(\mathsf{clos}^*(\phi_0)))| \cdot (N+1)^{|\mathrm{supp}(\phi_0)| + \|\phi_0\|} \cdot (1+\epsilon) \\
&\leq |\mathcal{O}(\mathcal{S})| \cdot |\mathcal{O}(\mathsf{clos}^*(\phi_0))| \cdot (N+1)^{|\mathrm{supp}(\phi_0)| + \|\phi_0\| + 1} \cdot (1+\epsilon) \\
&\leq |\mathcal{O}(\mathcal{S})| \cdot |\phi_0| \cdot \frac{(|\mathrm{supp}(\phi_0)| + \|\phi_0\|)!}{|\mathrm{supp}(\phi_0)|!} \cdot (N+1)^{|\mathrm{supp}(\phi_0)| + \|\phi_0\| + 1} \cdot (1+\epsilon)
\end{aligned}$$

with the last two lines using Lemma 3(1, second part) and Lemma 29(2). ◀

▶ **Theorem 41.** *Let $\mathcal{L}$ be a fresh-orbit-finite FLTS with n orbits and set of states $\mathcal{S}$, with register index r, and let ϕ_0 be a μFHML closed formula with maximum alternation depth d. Then, the time complexity of model checking ϕ_0 on some configuration in $\mathcal{L}$ is in*

$$\begin{aligned}
O(\ &(n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0) + r + 1)^{M(\phi_0)+1})^{\log(d)+6} \\
&+ (|\phi_0| + \Phi_{\mathcal{L}} + r) \cdot \max(M(\phi_0) + r, \mathsf{ndb}_{\mathcal{L}}) \cdot (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0) + r + 1)^{M(\phi_0)+1})^2\ )
\end{aligned}$$

If $(|\phi_0| + \Phi_{\mathcal{L}} + r) \cdot \max(M(\phi_0) + r, \mathsf{ndb}_{\mathcal{L}}) \in O((n \cdot |\phi_0| \cdot \frac{(|\phi_0|)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0) + r + 1)^{|\phi_0|+1})^4)$ *then the bound can be simplified to* $O((n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0) + r + 1)^{M(\phi_0)+1})^{\log(d)+6})$.

Note above that if the formula ϕ_0 has empty support, we get $M(\phi_0) = \|\phi_0\|$ and $\frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} = \|\phi_0\|!$. Let us now adapt the above calculations to FRAs. Given ϕ_0 and an r-FRA $\mathcal{A} = \langle Q, \mu, \Sigma, \delta \rangle$, we have:

- $n = |\mathcal{O}(\mathcal{S})| = |Q|$, as two state-assignment pairs are equal up to permutation iff they share the same state;
- $\mathsf{ndb}_{\mathcal{L}} \in O(rn(r+N)) = O(rn(r+M(\phi_0)))$, as there can be at most rn next configurations out of a configuration on a given label ℓ, and constructing a new bounded configuration incurs cost $O(r+N)$;
- $r_i(\mathcal{S}) \leq r$, which is equality if there is a state using all r registers;
- $\Phi_{\mathcal{L}} \in O(r)$, as it suffices to check that the two configurations contain the same state and, if so, construct a matching between their register assignments.

Therefore, the time complexity is in $O((|Q| \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\mathrm{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^{\log(d)+6})$. Finally, the interested reader can find a similar calculation for model checking μFHML formulas on *finitary* π-calculus processes (under early semantics) in Appendix B.

5 Conclusion

We introduced a logic to capture properties of nominal transition systems with history dependence and fresh-name generation. We showed model checking is decidable and has exponential complexity. A next step would be to parameterise this complexity, e.g. by bounding the number of registers in the FLTS or the arities of recursion variables. We also feel that some of our calculations are overly generous, for example in using the bound on orbits of product nominal sets to calculate the number of orbits in the bounded parity game.

Implementation. We implemented our algorithm for model-checking μFHML on FRAs and tested it on simple examples, like the ones used in this paper for illustration, which are solved instantly.[1] We leave full benchmarking and analysis of results to a future publication.

References

1 Fides Aarts, Paul Fiterau-Brostean, Harco Kuppens, and Frits Vaandrager. Learning register automata with fresh value generation. In *ICTAC Proceedings*, pages 165–183, 2015. doi:10.1007/978-3-319-25150-9_11.

2 M. H. Bandukara and Nikos Tzevelekos. On-the-fly bisimilarity checking for fresh-register automata. In Wei Dong and Jean-Pierre Talpin, editors, *Dependable Software Engineering. Theories, Tools, and Applications - 8th International Symposium, SETTA 2022, Beijing, China, October 27-29, 2022, Proceedings*, volume 13649 of *Lecture Notes in Computer Science*, pages 187–204. Springer, 2022. doi:10.1007/978-3-031-21213-0_12.

3 M. H. Bandukara and Nikos Tzevelekos. On-the-fly bisimulation equivalence checking for fresh-register automata. *J. Syst. Archit.*, 145:103010, 2023. doi:10.1016/J.SYSARC.2023.103010.

4 Martin Berger, Kohei Honda, and Nobuko Yoshida. Completeness and logical full abstraction in modal logics for typed mobile processes. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 99–111. Springer, 2008. doi:10.1007/978-3-540-70583-3_9.

5 Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014. doi:10.2168/LMCS-10(3:4)2014.

6 Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. A Robust Class of Data Languages and an Application to Learning. *LMCS*, 10(4), 2014. doi:10.2168/LMCS-10(4:19)2014.

7 Julian C. Bradfield and Perdita Stevens. Observational μ-calculus. Technical Report RS-99-5, BRICS (Basic Research in Computer Science), University of Aarhus, Denmark, 1999. URL: https://tidsskrift.dk/brics/article/view/21701.

8 Julian C. Bradfield and Colin Stirling. Modal mu-calculi. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 721–756. North-Holland, 2007. doi:10.1016/S1570-2464(07)80015-2.

[1] https://github.com/HamzaBandukara/RAM-C.

9 Julian C. Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 871–919. Springer, 2018. doi:10.1007/978-3-319-10575-8_26.

10 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasi-polynomial time. *SIAM J. Comput.*, 51(2):17–152, 2022. doi:10.1137/17M1145288.

11 Mads Dam. Model checking mobile processes. *Inf. Comput.*, 129(1):35–51, 1996. doi:10.1006/INCO.1996.0072.

12 Mads Dam. Proof systems for π-calculus logics. In Ruy J. G. B. de Queiroz, editor, *Logic for Concurrency and Synchronisation*, volume 18 of *Trends in Logic*, pages 145–212. Kluwer, 2003. doi:10.1007/0-306-48088-3_4.

13 N.G de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, 1972. doi:10.1016/1385-7258(72)90034-0.

14 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.

15 Clovis Eberhart and Bartek Klin. History-dependent nominal μ-calculus. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. doi:10.1109/LICS.2019.8785736.

16 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185392.

17 E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.*, 258(1-2):491–522, 2001. doi:10.1016/S0304-3975(00)00034-7.

18 Diego Figueira and Luc Segoufin. Future-looking logics on data words and trees. In Rastislav Královic and Damian Niwinski, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 2009. doi:10.1007/978-3-642-03816-7_29.

19 Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects Comput.*, 13(3-5):341–363, 2002. doi:10.1007/S001650200016.

20 Radu Grigore, Dino Distefano, Rasmus Lerchedahl Petersen, and Nikos Tzevelekos. Runtime verification based on register automata. In *TACAS Proceedings*, pages 260–276, 2013. doi:10.1007/978-3-642-36742-7_19.

21 Orna Grumberg, Orna Kupferman, and Sarai Sheinvald. Variable automata over infinite alphabets. In *LATA Proceedings*, pages 561–572, 2010. doi:10.1007/978-3-642-13089-2_47.

22 Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In J. W. de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings*, volume 85 of *Lecture Notes in Computer Science*, pages 299–309. Springer, 1980. doi:10.1007/3-540-10003-2_79.

23 Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, January 1985. doi:10.1145/2455.2460.

24 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

25 Victor Khomenko and Roland Meyer. Checking pi-calculus structural congruence is graph isomorphism complete. In *Ninth International Conference on Application of Concurrency to System Design, ACSD 2009, Augsburg, Germany, 1-3 July 2009*, pages 70–79. IEEE Computer Society, 2009. doi:10.1109/ACSD.2009.8.

26 Bartek Klin and Mateusz Lelyk. Modal mu-calculus with atoms. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic, CSL 2017, August 20-24, 2017, Stockholm, Sweden*, volume 82 of *LIPIcs*, pages 30:1–30:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CSL.2017.30.

27 Bartek Klin and Mateusz Lelyk. Scalar and vectorial mu-calculus with atoms. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:5)2019.

28 Vasileios Koutavas and Matthew Hennessy. First-order reasoning for higher-order concurrency. *Comput. Lang. Syst. Struct.*, 38(3):242–277, 2012. doi:10.1016/J.CL.2012.04.003.

29 Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982. doi:10.1016/0304-3975(82)90125-6.

30 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I and II. *Inf. Comput.*, 100(1), 1992. doi:10.1016/0890-5401(92)90008-4.

31 Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theor. Comput. Sci.*, 114(1):149–171, 1993. doi:10.1016/0304-3975(93)90156-N.

32 Andrzej Murawski, Steven Ramsay, and Nikos Tzevelekos. A contextual equivalence checker for IMJ*. In *ATVA proceedings*, pages 234–240, 2015. doi:10.1007/978-3-319-24953-7_19.

33 Andrzej S. Murawski, Steven J. Ramsay, and Nikos Tzevelekos. Bisimilarity in fresh-register automata. In *LICS Proceedings*, pages 156–167, 2015. doi:10.1109/LICS.2015.24.

34 Andrzej S. Murawski, Steven J. Ramsay, and Nikos Tzevelekos. Polynomial-time equivalence testing for deterministic fresh-register automata. In *MFCS Proceedings*, 2018. doi:10.4230/LIPIcs.MFCS.2018.72.

35 Andrzej S. Murawski and Nikos Tzevelekos. Algorithmic nominal game semantics. In *ESOP Proceedings*, pages 419–438, 2011. doi:10.1007/978-3-642-19718-5_22.

36 Rocco De Nicola and Michele Loreti. Multiple-labelled transition systems for nominal calculi and their logics. *Math. Struct. Comput. Sci.*, 18(1):107–143, 2008. doi:10.1017/S0960129507006585.

37 Joachim Parrow, Johannes Borgström, Lars-Henrik Eriksson, Ramunas Gutkovas, and Tjark Weber. Modal logics for nominal transition systems. *Log. Methods Comput. Sci.*, 17(1), 2021. URL: https://lmcs.episciences.org/7137.

38 Marco Pistore. *History-Dependent Automata*. PhD thesis, Università di Pisa, 1999.

39 Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003. doi:10.1016/S0890-5401(03)00138-X.

40 Davide Sangiorgi and David Walker. *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press, 2001. doi:10.1017/9781316134924.

41 Thomas Schwentick. Automata for XML—a survey. *JCSS*, 73(3):289–315, 2007. doi:10.1016/j.jcss.2006.10.003.

42 Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, volume 4207 of *Lecture Notes in Computer Science*, pages 41–57. Springer, 2006. doi:10.1007/11874683_3.

43 Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955. doi:10.2140/pjm.1955.5.285.

44 Nikos Tzevelekos. Fresh-register automata. In *POPL Proceedings*. ACM, 2011. doi:10.1145/1926385.1926420.

A Appendix

▶ **Definition 42.** Given $\vec{a}, \vec{b} \in \mathbb{A}^k$ for any k, such that $\vec{a}$ and $\vec{b}$ are pointwise distinct, let us write $\vec{a} = a_1, \ldots, a_k$ and $\vec{b} = b_1, \ldots, b_k$. We let $(\vec{a} \mapsto \vec{b})$ be the permutation obtained by extending the mapping of $\vec{a}$ to $\vec{b}$ as follows:

$$f_{\vec{a},\vec{b}} = \{(a_i, b_i) \mid 1 \leq i \leq k\}$$

$$(\vec{a} \mapsto \vec{b})(x) = \begin{cases} b_i & \text{if } \exists 1 \leq i \leq k. x = a_i \\ a_j & \text{if } x = b_i \notin \{\vec{a}\}, a_j \notin \{\vec{b}\} \text{ and } \exists k \in \mathbb{N}.\ b_i = f^k_{\vec{a},\vec{b}}(a_j) \\ x & \text{otherwise} \end{cases}$$

▶ **Lemma 7.** *Given an r-FRA $\mathcal{A} = \langle Q, \mu, \Sigma, \delta \rangle$, its FLTS $\mathcal{G}_\mathcal{A} = \langle \mathcal{S}_\mathcal{A}, \Sigma \times \mathbb{A}, \rightarrow \rangle$ is well defined.*

Proof. First, we must show that for any $(q, \rho, H) \xrightarrow{t,a} (q', \rho', H')$, then $\mathrm{supp}(\rho') \subseteq \mathrm{supp}(\rho) \cup \{a\}$ and $H' = H \cup \{a\}$. For the latter, it follows from the previous definition that $H' = H \cup \{a\}$. For the former, suppose this transition appears due to some $(q, \rho) \xrightarrow{t,x} (q', \rho')$, for any $x \in \{i, i^\bullet, i^\circledast\}$. For this, $\rho' = \rho[i \mapsto a] \upharpoonright \mu(q')$, thus we have that:

$$\mathrm{supp}(\rho') = (\mathrm{supp}(\rho) \cup \{a\}) \setminus \{\rho(j) \mid j \in \mathrm{dom}(\rho) \wedge j \notin \mu(q')\} \subseteq \mathrm{supp}(\rho) \cup \{a\}.$$

We next examine weakening and strengthening. Suppose that $(q, \rho, H) \xrightarrow{t,a} (q', \rho', H')$ due to some $q \xrightarrow{t,x} q'$ for $x \in \{i, i^\bullet, i^\circledast\}$. For weakening, we show that for all $b \notin H'$, then $(q, \rho, H \cup \{b\}) \xrightarrow{t,a} (q', \rho', H' \cup \{b\})$ due to $q \xrightarrow{t,x} q'$ as well. We have that a and b must be distinct (as $a \in H'$ by the previous point). If x is of the form i or $i^\bullet$, then this clearly holds as the history is not relevant to the transition. If x is of the form $i^\circledast$ then we note that a is still fresh for $H \cup \{b\}$ and, therefore, the transition can be taken. Finally, for strengthening, we need to show that, for all $b \notin \mathrm{supp}(\rho)$, then $(q, \rho, H \setminus \{b\}) \xrightarrow{t,a} (q', \rho', H'')$ for some H'' due to $q \xrightarrow{t,x} q'$. Here, since we are shrinking the history, the only thing to verify is that $(q, \rho, H \setminus \{b\})$ is a valid configuration, which it is as we stipulate that $b \notin \mathrm{supp}(\rho)$. ◀

▶ **Lemma 36.** *$\mathcal{R}_{(\mathcal{L},\phi_0,s_0,H_0)}$ is a nom-bisimulation.*

Proof. Let us write $\mathcal{R}$ for $\mathcal{R}_{(\mathcal{L},\phi_0,s_0,H_0)}$. Let us choose some $((s, H_1, \phi), (s, H_2, \phi)) \in \mathcal{R}_{(\mathcal{L},\phi_0,s_0,H_0)}$. By definition, the rank and polarity of (s, H_1, ϕ) and (s, H_2, ϕ) are the same. Let us introduce the following notation for economy (where $H \subseteq \mathbb{A}$) : $H_{(-b+a)} = (H \setminus \{b\}) \cup \{a\}$. We next look at the possible moves. We perform a case analysis on ϕ:

- $u = v$: neither (s, H_1, ϕ) nor (s, H_2, ϕ) can make a move.
- $\phi_1 \vee \phi_2$: we have that $(s, H_1, \phi) \rightarrow (s, H_1, \phi_i)$ and $(s, H_2, \phi) \rightarrow (s, H_2, \phi_i)$ for $i \in \{1, 2\}$. For any such i, as the history does not change, if $|H_2| \leq N$ then $((s, H_1, \phi_i), (s, H_2, \phi_i)) \in \mathcal{R}$ for $i \in \{1, 2\}$. Now suppose $|H_2| = N + 1$. As $\mathrm{supp}(\phi_i) \subseteq \mathrm{supp}(\phi)$, it follows that $\mathrm{supp}(\phi_i, s) \cap H_1 \subseteq \mathrm{supp}(\phi, s) \cap H_1 \subseteq H_2 \subseteq H_1$, meaning $((s, H_1, \phi_i), (s, H_2, \phi_i)) \in \mathcal{R}$ for $i \in \{1, 2\}$ as required. Similar can be shown for $\phi_1 \wedge \phi_2$.
- $\mathit{M}x.\phi'$ the possible moves in $\mathcal{G}$ are $(s, H_1, \phi) \rightarrow (s, H_1, \phi'\{a/x\})$ for all $a \# \phi, H_1$, hence each such move can be matched by some $(s, H_2, \phi) \rightarrow (s, H_2, \phi'\{a/x\})$ and we have that $a \notin H_2 \subseteq H_1$ hence $((s, H_1, \phi'\{a/x\}), (s, H_2, \phi'\{a/x\})) \in \mathcal{R}$.
  Conversely, the possible moves in $\mathcal{G}_N$ are $(s, H_2, \phi) \rightarrow (s, H_2, \phi'\{a/x\})$ for $a \# \phi, H_2$. If $a \notin H_1$, then as before this can be matched by $(s, H_1, \phi) \rightarrow (s, H_1, \phi'\{a/x\})$ with $((s, H_1, \phi'\{a/x\}), (s, H_2, \phi'\{a/x\})) \in \mathcal{R}$. Otherwise, and this is the more interesting case, suppose $a \in H_1 \setminus H_2 \cup \mathrm{supp}(\phi)$. Then, we can select some $b \notin H_1$, meaning $b \notin H_2$ and $(a\ b) \cdot (s, H_2, \phi'\{a/x\}) = (s, H_2, \phi'\{b/x\})$. This can be matched by $(s, H_1, \phi) \rightarrow (s, H_1, \phi'\{b/x\})$ with $((s, H_1, \phi'\{b/x\}), (s, H_2, \phi'\{b/x\})) \in \mathcal{R}$ as required.
- $\bigvee_x \phi'$: if $|H_2| \leq N$, then $(s, H_1, \bigvee_x \phi') \rightarrow (s, H_1, \phi'\{a/x\})$ can be matched by the move $(s, H_2, \bigvee_x \phi') \rightarrow (s, H_2, \phi'\{a/x\})$ with $((s, H_1, \phi'\{a/x\}), (s, H_2, \phi'\{a/x\})) \in \mathcal{R}$ as required. And similarly for each $(s, H_2, \bigvee_x \phi') \rightarrow (s, H_2, \phi'\{a/x\})$.

Suppose $|H_2| = N+1$, and let $(s, H_1, \bigvee_x \phi') \to (s, H_1, \phi'\{a/x\})$ for some $a \in \mathbb{A}$. Note that $\mathrm{supp}(\phi', s) \cap H_1 \subseteq H_2 \subseteq H_1$. We do a case analysis on a.

- If $a \in H_2$ or $a \notin H_1$, then this means a is in both histories, or neither, so it suffices to again match with the move $(s, H_2, \bigvee_x \phi') \to (s, H_2, \phi'\{a/x\})$ with $((s, H_1, \phi'\{a/x\}), (s, H_2, \phi'\{a/x\})) \in \mathcal{R}$.
- If $a \in H_1 \setminus H_2$, then we no longer have $((s, H_1, \phi'\{a/x\}), (s, H_2, \phi'\{a/x\})) \in \mathcal{R}$. Note that then $a \notin \mathrm{supp}(s, \phi')$ by definition of $\mathcal{R}$. We pick $b \in H_2 \setminus \mathrm{supp}(s, \phi')$ and make the move $(s, H_2, \bigvee_x \phi') \to (s, H_2, \phi'\{b/x\})$. We have $(a\ b) \cdot (s, H_2, \phi'\{b/x\}) = (s, H_{2(-b+a)}, \phi'\{a/x\})$ and $((s, H_1, \phi'\{a/x\}), (s, H_{2(-b+a)}, \phi'\{a/x\})) \in \mathcal{R}$, as required.

Conversely, suppose $(s, H_2, \bigvee_x \phi') \to (s, H_2, \phi'\{a/x\})$. We do case analysis on a.

- If $a \in H_2$ or $a \notin H_1$, then $(s, H_1, \bigvee_x \phi') \to (s, H_1, \phi'\{a/x\}) \mathcal{R} (s, H_2, \phi'\{a/x\})$.
- If $a \in H_1 \setminus H_2$, then let us choose the transition $(s, H_1, \bigvee_x \phi') \to (s, H_1, \phi'\{b/x\})$ for some $b \notin H_1 \cup \mathrm{supp}(\phi', s)$. Then, we have that $(s, H_1, \phi'\{b/x\}) \sim_{\mathsf{nom}} (s, H_{1(-a+b)}, \phi'\{a/x\})$ and $((s, H_{1(-a+b)}, \phi'\{a/x\}), (s, H_2, \phi'\{a/x\})) \in \mathcal{R}$ as required.

- $\langle t, \vec{a}\rangle \phi'$: in $\mathcal{G}$, the possible moves are $(s, H_1, \phi) \to (s', H_1', \phi')$ for all (s', H_1') such that $(s, H_1) \xrightarrow{t,\vec{a}} (s', H_1')$. By definition, we know $\mathrm{supp}(s') \subseteq \mathrm{supp}(s) \cup \{\vec{a}\}$ and $H_1' = H_1 \cup \{\vec{a}\}$. If $|H_1'| \leq N$, then this is matched by some move $(s, H_2, \phi) \to (s', H_2', \phi')$ with $H_2' = H_2 \cup \{\vec{a}\}$ and $((s', H_1', \phi'), (s', H_2', \phi')) \in \mathcal{R}$. Otherwise, we can see that $|H_2 \cup \{\vec{a}\}| > N$. If $H_1 = H_2$, then this can be matched by some move $(s, H_2, \phi) \to (s', H_2', \phi')$ with $H_1' \cap \mathrm{supp}(\phi', s') \subseteq H_2' \subseteq H_1'$ and $|H_2'| = N+1$. As $|\mathrm{supp}(\phi', s')| \leq N$, we have that such a H_2' exists and $((s', H_1', \phi'), (s', H_2', \phi')) \in \mathcal{R}$ as required. If $H_2 \subsetneq H_1$, then by the conditions of the FLTS, $(s, H_1) \xrightarrow{\ell} (s', H_1')$ implies that $(s, H_2) \xrightarrow{\ell} (s', H_2')$ with $H_1' \cap \mathrm{supp}(\phi', s') \subseteq H_2' \subseteq H_1'$ and $|H_2'| = N+1$ thus completing the simulation in this step.

  Conversely, let $(s, H_2, \phi) \to (s', H_2', \phi')$ with $s \xrightarrow{t,\vec{a}} s'$. One of the following is the case:
  - $|H_2 \cup \{\vec{a}\}| \leq N$, then $H_2' = H_1 \cup \{\vec{a}\}$. This can be matched by the move $(s, H_1, \phi) \to (s', H_1', \phi')$ with $H_1' = H_1 \cup \{\vec{a}\}$ and $((s', H_1', \phi'), (s', H_2', \phi')) \in \mathcal{R}$.
  - $|H_2 \cup \{\vec{a}\}| > N$ and $(H_2 \cup \{\vec{a}\}) \cap \mathrm{supp}(s', \phi') \subseteq H_2' \subseteq H_2 \cup \{\vec{a}\}$ with $|H_2'| = N+1$. We note that $\mathrm{supp}(s') \subseteq \mathrm{supp}(s) \cup \{\vec{a}\}$ and $\mathrm{supp}(\phi) = \mathrm{supp}(\phi') \cup \{\vec{a}\}$, and examine the case where $H_2 \subseteq H_1$:

$$\begin{aligned}(H_1 \cup \{\vec{a}\}) \cap \mathrm{supp}(s', \phi') &= ((H_1 \setminus \{\vec{a}\}) \cap \mathrm{supp}(s', \phi')) \cup (\{\vec{a}\} \cap \mathrm{supp}(s', \phi')) \\ &\subseteq (H_2 \cap \mathrm{supp}(s', \phi')) \cup (\{\vec{a}\} \cap \mathrm{supp}(s', \phi')) \subseteq H_2'\end{aligned}$$

    If $H_2 = H_1$, the move can be matched by $(s, H_1, \phi) \to (s', H_1', \phi')$ with $H_1' = H_1 \cup \{\vec{a}\}$ and $((s', H_1', \phi'), (s', H_2', \phi')) \in \mathcal{R}$. Suppose that $H_2 \subsetneq H_1$. By the well-bounding relation, $\mathrm{supp}(\ell) \cap (H_1 \setminus H_2) = \emptyset$ as $\mathrm{supp}(\ell) \subseteq \mathrm{supp}(\phi)$. Therefore, this can also be matched by $(s, H_1, \phi) \to (s', H_1', \phi')$ with $H_1' = H_1 \cup \{\vec{a}\}$ and $((s', H_1', \phi'), (s', H_2', \phi')) \in \mathcal{R}$.
- $(\mu X(\vec{x}).\phi')(\vec{a})$: in $\mathcal{G}$, the only move is $(s, H_1, \phi) \to (s, H_1, (\phi'\{\mu X(\vec{x}).\phi'/X\})\{\vec{a}/\vec{x}\})$. As H_1 does not change and $\mathrm{supp}(\phi, s) = \mathrm{supp}((\phi'\{\mu X(\vec{x}).\phi'/X\})\{\vec{a}/\vec{x}\})$, then this can be matched by the only move in $\mathcal{G}_N$, i.e. $(s, H_2, \phi) \to (s, H_2, (\phi'\{\mu X(\vec{x}).\phi'/X\})\{\vec{a}/\vec{x}\})$ with $((s, H_1, (\phi'\{\mu X(\vec{x}).\phi'/X\})\{\vec{a}/\vec{x}\}), (s, H_2, (\phi'\{\mu X(\vec{x}).\phi'/X\})\{\vec{a}/\vec{x}\})) \in \mathcal{R}$.

The remaining cases can be shown similarly to the ones above. ◀

▶ **Theorem 41.** *Let $\mathcal{L}$ be a fresh-orbit-finite FLTS with n orbits and set of states $\mathcal{S}$, with register index r, and let ϕ_0 be a μFHML closed formula with maximum alternation depth d. Then, the time complexity of model checking ϕ_0 on some configuration in $\mathcal{L}$ is in*

$$O(\ (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^{\log(d)+6}$$
$$+ (|\phi_0| + \Phi_{\mathcal{L}} + r) \cdot \max(M(\phi_0)+r, \mathsf{ndb}_{\mathcal{L}}) \cdot (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^2\ )$$

If $(|\phi_0| + \Phi_{\mathcal{L}} + r) \cdot \max(M(\phi_0)+r, \mathsf{ndb}_{\mathcal{L}}) \in O((n \cdot |\phi_0| \cdot \frac{(|\phi_0|)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{|\phi_0|+1})^4)$ *then the bound can be simplified to* $O((n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^{\log(d)+6})$.

Proof. To verify whether $(s, H) \in \hat{\mathcal{S}}$ satisfies ϕ_0, we must check whether $(s, H) \in [\![\phi_0]\!]_{\xi_0}$ where ξ_0 is an empty $\mathcal{U}$-variable assignment. First, we set $\phi = !(\phi_0)$, which (by Lemma 25) will have size bounded by $|\phi_0|$. From this, we can construct the orbits parity game $\mathsf{Orbs}(\mathcal{G}_N)$ of $\mathcal{G}_N = \mathcal{G}_N(\mathcal{L}, \phi, s, H)$ where $N = |\phi_0| + \|\phi_0\| + r_i(\mathcal{S})$, By Theorem 30, we have that if $(s, H) \in [\![\phi_0]\!]_{\xi_0}$ then defender has a winning strategy from $\mathcal{G} = \mathcal{G}(\mathcal{L}, \phi, s, H)$. Using Lemma 36, the relation $\mathcal{R}_{(\mathcal{L},\phi,s,H)} \subseteq \operatorname{Pos}(\mathcal{G}) \times \operatorname{Pos}(\mathcal{G}_N)$ is a nom-bisimulation, and by Lemma 34 then defender wins in $\mathcal{G}$ from (s, H, ξ_0, ϕ) iff they win in $\mathcal{G}_N$ from (s, H, ξ_0, ϕ). Similarly, using Lemma 38 and Corollary 39 then defender wins from $\mathcal{G}_N$ from (s, H, ξ_0, ϕ) iff defender wins in $\mathsf{Orbs}(\mathcal{G}_N)$ from $\mathbb{O}(s, H, \xi_0, \phi)$. Then, by Lemma 40, $\mathsf{Orbs}(\mathcal{G}_N)$ can be constructed in:

$$O((|\phi_0| + \Phi_{\mathcal{L}} + N) \cdot \max(N, \mathsf{ndb}_{\mathcal{L}})$$
$$\cdot (|\mathcal{O}(\mathcal{S})| \cdot |\phi_0| \cdot \frac{(|\operatorname{supp}(\phi_0)| + \|\phi_0\|)!}{|\operatorname{supp}(\phi_0)|!} \cdot (N+1)^{|\operatorname{supp}(\phi_0)|+\|\phi_0\|+1} \cdot (1+o(1)))^2)$$

and has size:

$$n \cdot |\phi_0| \cdot \frac{(|\operatorname{supp}(\phi_0)| + \|\phi_0\|)!}{|\operatorname{supp}(\phi_0)|!} \cdot (N+1)^{|\operatorname{supp}(\phi_0)|+\|\phi_0\|+1} \cdot (1+o(1))$$

The winning regions of $\mathsf{Orbs}(\mathcal{G}_N)$ can be calculated in time bounded by (cf. [10]):

$$O((n \cdot |\phi_0| \cdot \frac{(|\operatorname{supp}(\phi_0)| + \|\phi_0\|)!}{|\operatorname{supp}(\phi_0)|!} \cdot (N+1)^{|\operatorname{supp}(\phi_0)|+\|\phi_0\|+1})^{\log(d)+6})$$

And so, the final complexity is in:

$$O((n \cdot |\phi_0| \cdot \frac{(|\operatorname{supp}(\phi_0)| + \|\phi_0\|)!}{|\operatorname{supp}(\phi_0)|!} \cdot (N+1)^{|\operatorname{supp}(\phi_0)|+\|\phi_0\|+1})^{\log(d)+6}$$
$$+ ((|\phi_0| + \Phi_{\mathcal{L}} + N) \cdot \max(N, \mathsf{ndb}_{\mathcal{L}}) \cdot (n \cdot |\phi_0| \cdot \frac{(|\operatorname{supp}(\phi_0)| + \|\phi_0\|)!}{|\operatorname{supp}(\phi_0)|!} \cdot (N+1)^{|\operatorname{supp}(\phi_0)|+\|\phi_0\|+1})^2))$$

Now, using $N = M(\phi_0) + r$ and $M(\phi_0) = |\operatorname{supp}(\phi_0)| + \|\phi_0\|$, we get:

$$O(\ (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^{\log(d)+6}$$
$$+ ((|\phi_0| + \Phi_{\mathcal{L}} + M(\phi_0) + r) \cdot \max(M(\phi_0)+r, \mathsf{ndb}_{\mathcal{L}}) \cdot (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^2)\ )$$

and, as $M(\phi_0) \leq |\phi_0|$:

$$O(\ (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^{\log(d)+6}$$
$$+ ((|\phi_0| + \Phi_{\mathcal{L}} + r) \cdot \max(M(\phi_0)+r, \mathsf{ndb}_{\mathcal{L}}) \cdot (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0)+r+1)^{M(\phi_0)+1})^2)\ )$$

Finally, if $(|\phi_0|+\Phi_{\mathcal{L}}+r)\cdot\max(M(\phi_0)+r, \mathsf{ndb}_{\mathcal{L}}) \in O((n\cdot|\phi_0|\cdot\frac{(|\phi_0|)!}{|\operatorname{supp}(\phi_0)|!}\cdot(M(\phi_0)+r+1)^{|\phi_0|+1})^4)$ then we obtain the required simplified bound. ◀

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad P \mid Q \equiv Q \mid P \qquad P \mid 0 \equiv P$$
$$P + (Q + R) \equiv (P + Q) + R \qquad P + Q \equiv Q + P \qquad P + 0 \equiv P$$
$$\nu x.(P \mid Q) \equiv P \mid \nu x.Q \ (*) \qquad \nu x.(P + Q) \equiv P + \nu x.Q \ (*) \qquad [a = a]P \equiv P$$
$$\nu x.\nu y.P \equiv \nu y.\nu x.P \qquad \nu x.P \equiv P \ (*) \qquad [a = b]P \equiv 0$$

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}\ \alpha = \tau, ab \qquad \frac{}{a(x).P \xrightarrow{ab} P\{b/x\}} \qquad \frac{P_1 \xrightarrow{\alpha} P'}{P_1 + P_2 \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{[a = a]P \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{[a \neq b]P \xrightarrow{\alpha} P'}$$

$$\frac{P\{b/x\} \xrightarrow{\bar{a}b} P'}{\nu x.P \xrightarrow{\bar{a}(b)} P'}\ b \notin \mathsf{n}(P) \qquad \frac{P_1 \xrightarrow{\bar{a}b} P_1' \qquad P_2 \xrightarrow{ab} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'} \qquad \frac{P_1 \xrightarrow{\alpha} P'}{P_1|P_2 \xrightarrow{\alpha} P'|P_2}\ \mathsf{bn}(\alpha) \cap \mathsf{n}(P_2) = \emptyset$$

$$\frac{P \xrightarrow{\alpha} P'}{\nu x.P \xrightarrow{\alpha} \nu x.P'} \qquad \frac{P_1 \xrightarrow{\bar{a}(b)} P_1' \qquad P_2 \xrightarrow{ab} P_2'}{P_1|P_2 \xrightarrow{\tau} \nu x.(P_1'|P_2')\{x/b\}}\ b \notin \mathsf{n}(P_2) \qquad \frac{P\{u_1/x_1, \ldots, u_n/x_n\} \xrightarrow{\alpha} P'}{A(\vec{u}) \xrightarrow{\alpha} P'}\ A(\vec{x}) = P$$

Figure 1 Structural congruence for π-calculus (top, where $(*)$ is the condition that x not free in P) and early reduction semantics (bottom). Symmetric versions of rules are omitted; where a, b appear in the same rule we assume $a \neq b$.

B Model-checking for the π-calculus

The π-calculus is a paradigmatic process language for concurrent interactions involving name passing [30, 40]. The syntax of π-calculus processes is defined by the following grammar:

$$\begin{aligned} P, Q &::= 0 \mid \pi.P \mid \nu x.P \mid P|Q \mid P + Q \mid [u = v]P \mid [u \neq v]P \mid A(\vec{u}) \\ \pi &::= \tau \mid \bar{u}v \mid u(x) \\ u, v &::= x \mid a \end{aligned}$$

where x, y, etc. range over a finite set of *variables*; A, B, etc. range over a countable set of *process variables*, while a, b, etc. are names. The constructs $\nu x.P$ and $u(x).P$ are binders (for x), and we use standard α-equivalence convention. Process variables are used to allow recursion by means of definitions of the form $A(\vec{x}) = P$ where P only contains free variables from $\vec{x}$. By abuse of notation, we may use P, Q for process variables as well as processes, and we define variable-capture-avoiding substitutions $P\{u/v\}$ in the usual way. Let us call a π-calculus process *finitary* if the processes it can reduce to are bounded in size up to *structural congruence*, where structural congruence is the least congruence relation satisfying the rules in Figure 1 (top). It is known that checking whether two processes are structurally congruent is GI-complete [25]. Let us write $\Pi_{SC}(n)$ for the complexity of checking this for two processes of sizes bounded by n.

We next show that the semantics of any given π-calculus process can be induced by an FLTS. We focus on early semantics using the labels given by the following grammar:

$$\alpha ::= \tau \mid ab \mid \bar{a}b \mid \bar{a}(b).$$

The label τ corresponds to silent transitions, whereas ab corresponds to inputting name b on the *channel* named a. The last two sorts of label are for outputs: $\bar{a}b$ outputs name b on channel a, whereas $\bar{a}(b)$ outputs a *fresh* name b on channel a. We write $\mathsf{n}(\alpha)$ for the set of names included in α, and similarly we write $\mathsf{n}(P)$ for the names appearing in a process P. It is convenient to have notation for bound/fresh names as well, setting $\mathsf{bn}(\bar{a}(b)) = \{b\}$ and $\mathsf{bn}(\alpha) = \emptyset$ for any other α. The LTS is produced by the rules in Figure 1 (bottom).

We proceed to the translation to FLTS, where each state will be of the form (P, H), where P represents the process, and H represents all names that have been seen by the process so far, and where $\mathrm{supp}(P) = \mathsf{n}(P) \subseteq H$.

▶ **Definition 43.** Given a π-calculus process P, we define its FLTS to initially contain the configuration $(P, n(P))$ and be generated according to the rule:

$$\frac{P \xrightarrow{\alpha} P'}{(P,H) \xrightarrow{(\alpha)^\circ} (P', H \cup \mathsf{n}(\alpha))} \qquad (\tau)^\circ = \tau \quad (ab)^\circ = (\mathsf{inp}, ab) \quad (\bar{a}b)^\circ = (\bar{a}(b))^\circ = (\mathsf{out}, \mathsf{ab})$$

For any process P, let us call P_{SC} the set of all processes that P can reduce to up to structural congruence. Intuitively, each element of P_{SC} is an equivalence classes $[Q]$ of all processes that are structurally congruent to Q. Next, we set P_R to be $\mathcal{O}(P_{SC})$, that is, the set of orbits of structurally congruent equivalence classes. With this, we are able to conclude with a time complexity bound for model checking π-calculus processes on μFHML.

▶ **Corollary 44.** *Let P be a finitary π-calculus process with P_R being the set of orbits of structurally congruent equivalence classes P can reduce to, and let ϕ_0 be a μFHML formula. Furthermore, let us set:*

- *B to be $\max\{|Q| \mid Q \in P_R\}$,*
- *n to be $|\mathcal{O}(P_R)|$,*
- *r to be $\max\{|\mathsf{n}(Q)| \mid Q \in P_R\}$,*
- *$\mathcal{L}$ to be the FLTS derived from P following Definition 43.*

Then, the time complexity of model checking ϕ_0 on a configuration in $\mathcal{L}$ is in:

$$\begin{aligned} O(\ & (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0) + r + 1)^{M(\phi_0)+1})^{\log(d)+6} \\ & + (|\phi_0| + \Pi_{SC}(B + cr) + r) \cdot \max(M(\phi_0) + r, rn(r + M(\phi_0))) \\ & \cdot (n \cdot |\phi_0| \cdot \frac{M(\phi_0)!}{|\operatorname{supp}(\phi_0)|!} \cdot (M(\phi_0) + r + 1)^{M(\phi_0)+1})^2\ ) \end{aligned}$$

with $\Phi_{\mathcal{L}}(M(\phi_0)) \in O(\Pi_{SC}(B + cr) + M(\phi_0))$ for some constant c.

Proof. The proof follows similarly to Theorem 41, here we justify our setting. By definition, for each $Q \in P_R$ there must exist some $Q' \in \mathcal{S}$ such that $Q \equiv Q'$, therefore we have that $|\mathcal{O}(P_R)| = |\mathcal{O}(\mathcal{S})|$. The value of r comes from the maximum number of names appearing in any formula in the reduction of P, thus representing the register index of $\mathcal{S}$. We have that $\mathsf{ndb}_{\mathcal{L}} \in rn(r + M(\phi_0))$ as there can be at most rn next configurations out of a configuration on a given label ℓ, and constructing a new bounded configuration incurs cost $O(r + M(\phi_0))$. It remains to discuss the bound for $\Phi_{\mathcal{L}}$. To determine if two processes P_1, P_2 with $|P_1|, |P_2| \leq B$ are structurally congruent up to permutation, it suffices to determine that

$$\nu x_1.\nu x_2.\ldots.\nu x_r.\tau.P_1' \equiv \nu x_1.\nu x_2.\ldots.\nu x_r.\tau.P_2'$$

where each P_i' is P_i with each of its fresh names a_j substituted by x_j. Thus, the new processes under examination will have size bounded by $B + cr$ for some constant c, and so can be computed in $\Pi_{SC}(B + cr)$. Thus, we obtain the time complexity for $\Phi_{\mathcal{L}}(M(\phi_0))$. ◀