Synthesis for prefix first-order logic on data words *

Julien Grange[1][0009−0005−0470−1781] and Mathieu Lehaut[2][0000−0002−6205−0682]

[1] Univ Paris Est Creteil, LACL, F-94010 Creteil, France julien.grange@lacl.fr
[2] University of Gothenburg, Sweden lehaut@chalmers.se

Abstract. We study the reactive synthesis problem for distributed systems with an unbounded number of participants interacting with an uncontrollable environment. Executions of those systems are modeled by data words, and specifications are given as first-order logic formulas from a fragment we call prefix first-order logic that implements a limited kind of order. We show that this logic has nice properties that enable us to prove decidability of the synthesis problem.

1 Introduction

Distributed algorithms have been increasingly more common in recent years, and can be found in a wide range of domains such as distributed computing, swarm robotics, multi-agent systems, and communication protocols, among others. Those algorithms are often more complex than single-process algorithms due to the interplay between the different processes involved in the computations. Another complication arises in the fact that some algorithms must be designed for distributed systems where the number of participants is not known in advance, which is often the case in applications where agents can come and leave at a moment's notice as is the case, for instance, in ad-hoc networks. Those properties make it hard for programmers to design such algorithms without any mistake, thus justifying the development of formal methods for their verification.

In this paper, we focus on the *reactive synthesis* problem. This problem involves systems that interact with an uncontrollable environment, with the system outputting some values depending on the inputs that are given by the environment. Given a specification stating what are the allowed behaviors of the whole system, the goal is to automatically build a program that would satisfy the specification. This problem dates all the way back to Church [4], whose original statement focused on sequential systems, and was solved in this context by Büchi and Landweber [3]. They reformulated this problem as a two-player synthesis game between the System and an adversarial Environment alternatively choosing actions from a finite alphabet. The goal of System is for the resulting sequence of actions to satisfy the specification, while Environment wants to falsify it. A winning strategy for System, if it exists, can then be seen as a program ensuring that the specification is always met.

* Supported by the ERC Consolidator grant D-SynMA (No. 772459).

At the cost of having one copy of the alphabet for each participant, one can adapt this setting to distributed systems where the number of participants is fixed. However, a finite alphabet is too restrictive to handle systems with an unbounded number of participants such as those we described earlier. It is therefore worth extending this problem to infinite action alphabets. To that end, we turn to *data words*, as introduced by Bojanczyk et al. [2]. A data word is a sequence of pairs consisting of an action from a finite alphabet and a data from an infinite alphabet. In our context the data represents the identity of the process doing the action, meaning that a data word is seen as an execution of the system describing sequentially which actions have been taken by each process. In the corresponding synthesis game, the two players alternatively choose both an action and a process, and the resulting play is a data word.

The last ingredient needed to properly define the synthesis problem is the choice of a formalism in which specifications are written. Unfortunately, there is no strong candidate for a "standard" way of representing sets of data words, in contrast to finite automata in the case of simple words. Many formalisms have been proposed so far, but all of them lack either good closure properties (union, complementation, etc.), have bad complexity for some basic decision procedures (membership, ...), are lacking in expressivity power or do not have a good equivalent automata $\Leftrightarrow$ logic characterization. Let us cite nonetheless register automata who were considered first by Kaminsky and Francez [10] but have seen different extensions over time, pebble automata by Neven et al. [12] and data automata [2]. On the logical side, several formalisms have been proposed, such as a variant of first-order logic [2], Freeze LTL [6] and the logic of repeating values [5]. We refer the reader to Ahmet Kara's dissertation for a more comprehensive survey [11]. Most of the previous works study the membership problem (in the case of automata) and the satisfiability problem (for logics), which are useful for model-checking applications but not enough in the synthesis context, as they lack the adversarial environment factor that is central to the problem. A few attempts have been made in this direction, notably for the logic of repeating values by Figueira and Praveen [8] and for register automata by Exibard et al. [7].

We follow previous work [1,9] and focus on synthesis for first-order logic extended to data words. In this extension, we add a predicate $\sim$ to the logic such that $x \sim y$ is true when two positions x and y of a data word share the same data value, i.e. when both actions have been made by the same process. Moreover, we partition the set of processes into System processes and Environment processes, and restrict each player to their own processes. The reason for this is two-fold: first, when processes are shared, the synthesis problem has been shown to be undecidable even for the simplest logic possible $\mathrm{FO}^2[\sim]$. Second, inputs and outputs are usually located in different components of the system; it thus makes sense to see them as different processes.

As always, there is a trade-off between expressivity of the logic and decidability of its synthesis problem. The well-known LTL undecidability result from Pnueli and Rosner [13] occurs because the logic allows specifying properties that the system is too weak to satisfy. We must therefore find a balance between

making the logic expressive enough to be useful, while limiting its power so that it cannot specify properties the system is not expected to be able to satisfy in the first place. Our previous results have shown that adding any kind of order on the positions of the data word, such as either the immediate successor predicate or the happens-before predicate, makes the synthesis problem undecidable. While those two predicates are fine from a centralized point of view that sees the whole system as a sequential machine, they are not that well suited for real-life systems. Indeed, it is ambitious to expect each process to know everything that happened on other processes in the exact order those actions happened; this would require every process to be informed instantly after every action happening in the system. What is more reasonable is simply to expect a process to know the order of occurrence of its own actions only, without knowing whether those actions happened before or after actions made by other processes.

This leads us to introduce a new operator $\lesssim$, for which $x \lesssim y$ if $x \sim y$ and x occurs before y in the data word. We call FO[$\lesssim$] the extension of FO that includes only this new predicate. It is strictly more expressive than FO[$\sim$], as the $\sim$ predicate can easily be simulated by $\lesssim$. Similar to FO[$\sim$], we can study separately the class of each process. Whereas FO[$\sim$] could only count how many times each action happened (up to some threshold), we can now express anything that FO[$<$] can express on (simple) words. Unfortunately, the decidability of the synthesis problem for FO[$\lesssim$] remains open. In this paper, we show a positive result for a restriction of this logic that we call *prefix first-order logic*, denoted by $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$. In this restriction, the first variable quantified for each process is called a bounding variable, and every subsequently defined variable belonging to the same process must occur before the bounding variable. In other words, the first variable for each class pins down a finite prefix of the class and throws away the rest of the class; the rest of the variables can only talk about positions that fall inside this prefix. This restriction allows us to obtain good properties that we leverage to show decidability of the synthesis problem.

This paper is organized as follows. We first define prefix first-order logic $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ in Section 2, and the synthesis problem and its equivalent games in Section 3. We then show the synthesis problem for $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ to be decidable of in Section 4, before concluding in Section 5. Omitted proofs can be found in the appendix.

2 Prefix first-order logic

2.1 Data words and preliminaries

Fix two disjoint alphabets Σ_S and Σ_E, which are respectively the System and Environment *actions*. Let $\Sigma = \Sigma_S \uplus \Sigma_E$ denote their union. Moreover, let $\mathbb{P}_{\mathrm{S}}$ and $\mathbb{P}_{\mathrm{E}}$ be two disjoint sets of System and Environment *processes*, respectively.

A *data word* is a (finite or infinite) sequence $\mathfrak{w} = (a_0, p_0)(a_1, p_1)\ldots$ of pairs $(a_i, p_i) \in (\Sigma_S \times \mathbb{P}_{\mathrm{S}}) \cup (\Sigma_E \times \mathbb{P}_{\mathrm{E}})$. A pair (a, p) indicates that action a has been taken by process p. The *class* of a process p is the word $w_p = a_{k_0} a_{k_1} \cdots \in \Sigma^\star$

where $(k_i)_i$ is exactly the sequence of positions in $\mathfrak{w}$ where actions are made by p. We see data words as logical structures (and will conveniently identify a data word with its associated structure) over the vocabulary consisting of two binary predicates $\sim$, $<$, and two unary predicates $\mathbb{P}_{\mathrm{S}}$, $\mathbb{P}_{\mathrm{E}}$ as well as one additional unary predicate for each letter of Σ. The universe of a data word has one element for each process (called *process elements* – these are needed in order to quantify over processes that have not played any action, and will drastically increase the expressive power of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$), and one element for each position in the data word. Predicate $\mathbb{P}_{\mathrm{S}}$ (resp. $\mathbb{P}_{\mathrm{E}}$) is interpreted as the set of all System (resp. Environment) process elements. For $a \in \Sigma$, the predicate a holds on every position which correspond to action a. Predicate $<$ is interpreted as the linear order on the set of positions corresponding to their order in the data word (and is thus not defined on process elements), and $\sim$ is interpreted as an equivalence relation which has one equivalence class for every process, encompassing both its process element and all positions of its class. It will be convenient to use $\mathbb{P}(x)$ as an alias for "$\mathbb{P}_{\mathrm{S}}(x) \vee \mathbb{P}_{\mathrm{E}}(x)$" and $x \lesssim y$ as an alias for "$x \sim y \wedge x < y$".

Let $\mathrm{DW}_{\Sigma}^{\mathbb{P}}$ denote the set of all data words over actions Σ and processes $\mathbb{P}$. We write $\mathfrak{w}[i \dots j]$ for the factor of $\mathfrak{w}$ occurring between positions i and j (both included), and $\mathfrak{w}[i \dots]$ for the suffix starting at position i; we extend both notations to regular words as well.

2.2 Prefix first-order logic on data words

We define *prefix first-order logic on data words* $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ by induction on its formulas. We write $\varphi(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}})$ to mean that the free variables of φ belong to the pairwise disjoint union X of the three sets X^{proc} (the *process variables*), X^{bnd} (the *bounding variables*) and X^{pref} (the *prefix variables*).

$$
\begin{aligned}
&\varphi(X^{\mathrm{proc}};\ X^{\mathrm{bnd}}; X^{\mathrm{pref}}) ::= \\
&\quad x = y && (x, y \in X) \\
&\mid \mathbb{P}_{\mathrm{S}}(x) \quad \mid \mathbb{P}_{\mathrm{E}}(x) && (x \in X^{\mathrm{proc}}) \\
&\mid a(x) && (x \in X^{\mathrm{bnd}} \cup X^{\mathrm{pref}}, a \in \Sigma) \\
&\mid x \lesssim y && (x, y \in X^{\mathrm{pref}}) \\
&\mid x \sim y && ((x, y) \in X^{\mathrm{proc}} \times X) \\
&\mid \varphi_1(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) \wedge \varphi_2(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) \\
&\mid \neg\psi(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) \\
&\mid \exists x, \mathbb{P}(x)\ \wedge\ \psi(X^{\mathrm{proc}} \cup \{x\}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) && (x \notin X) \\
&\mid \exists x, \neg\mathbb{P}(x)\ \wedge\ \Big(\bigwedge_{y \in X^{\mathrm{bnd}}} x \not\sim y \Big) \wedge \psi(X^{\mathrm{proc}}; X^{\mathrm{bnd}} \cup \{x\}; X^{\mathrm{pref}}) && (x \notin X) \\
&\mid \exists x,\ x \lesssim y \wedge \psi(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}} \cup \{x\}) && (x \notin X, y \in X^{\mathrm{bnd}})
\end{aligned}
$$

with $X = X^{\mathrm{proc}} \uplus X^{\mathrm{bnd}} \uplus X^{\mathrm{pref}}$. The intuition is as follows.

We allow one to quantify over the process elements with process variables. Bounding and prefix variables are used to quantify over the elements of the process classes; when quantifying (existentially or universally) over an element of a process class (that is, an actual position of the data word), one must first use a bounding variable. From then on, only prefix variables can be used on this process class, which can only quantify earlier positions in the class (i.e. positions which are $\lesssim$ to the bounding position). Note that one can still quantify over other classes, using new bounding variables.

The semantics is defined as usual. As always, the *quantifier depth* of a formula is the maximal number of nested quantifiers (without regard to whether they quantify process, bounding or prefix variables).

By construction, $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ is a fragment of $\mathrm{FO}[\sim, <]$ (first-order logic with $\sim$ and $<$). Example 1 below illustrates that $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ encompasses $\mathrm{FO}[\sim]$.

Example 1. Let us fix the alphabets $\Sigma_E := \{a_E\}$ and $\Sigma_S := \{a_S\}$. There exists an $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formula $\varphi_{1a_E \leftrightarrow 1a_S}$ of quantifier depth 3 stating that there exists a process with exactly one a_E if and only if there exists a process with exactly one a_S. Indeed, the existence of a process with exactly one a_E (and similarly for a_S) can be stated as

$$\begin{aligned} \exists x,\ \mathbb{P}(x) \quad &\wedge \quad \big(\exists y,\ \neg\mathbb{P}(y)\ \wedge\ y \sim x\ \wedge\ a_E(y)\big) \\ &\wedge \quad \neg\big(\exists y,\ \neg\mathbb{P}(y)\ \wedge\ y \sim x\ \wedge\ a_E(y)\ \wedge\ \exists z,\ z \lesssim y \wedge a_E(z)\big)\,. \end{aligned}$$

Here, x is a process variable, both occurrences of y are bounding variables and z is a prefix variable.

If a (finite or infinite) word is seen as a data word with exactly one data class, then $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ can in particular be seen a logic on words: it is equivalent to the restriction of first-order logic on words where

- the formula must start with a universal or existential quantification on the variable $\overline{x}$,
- after that, each new quantification must be of the form $\exists x < \overline{x}$ or $\forall x < \overline{x}$.

We will refer to this logic on words as $\mathrm{FO}^{\mathrm{PREF}}$.

For any word w, we let $\langle w \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$ denote its $\mathrm{FO}^{\mathrm{PREF}}$-type of depth k, i.e. the set of all sentences of $\mathrm{FO}^{\mathrm{PREF}}$ with quantifier depth at most k satisfied in w. Having fixed an alphabet, we denote by $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ the set of $\mathrm{FO}^{\mathrm{PREF}}$-types of depth k on words.

Lemma 2. *Let $k \in \mathbb{N}$ and let $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ be two finite or infinite data-words on the same alphabet Σ, such that for every $\tau \in \mathrm{Types}^k_{FO^{PREF}}$, the number of classes of type τ in $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ are either the same or both at least k. Then $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ agree on all $FO^{PREF}[\lesssim]$-sentences of quantifier depth at most k.*

As a direct corollary of Lemma 2, in order to decide whether a data word $\mathfrak{w}$ satisfies an $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formula of quantifier depth k, it is enough to know, for each k-type τ for $\mathrm{FO}^{\mathrm{PREF}}$, how many (up to k) classes of $\mathfrak{w}$ have type τ. We shall use this fact later in proofs, and refer to this abstraction of a data word as its *collection* of types.

2.3 Properties of $\mathrm{FO}^{\mathrm{PREF}}$ types

Let us now try to understand the behavior of $\mathrm{FO}^{\mathrm{PREF}}$ on words. In the following, we fix an alphabet Σ and an integer k.

First, note that the equivalence relation "have the same $\mathrm{FO}^{\mathrm{PREF}}$-type" is not a congruence in the monoid of finite words. Indeed, one can convince themselves (or prove formally, using the Ehrenfeucht-Fraïssé games introduced in the appendix) that for any $k \in \mathbb{N}$, the two words

$$u = ababa \cdots aba$$

and

$$v = ababa \cdots abab$$

have the same $\mathrm{FO}^{\mathrm{PREF}}$ k-type (as long as they are long enough with respect to k). However, $u \cdot a$ and $v \cdot a$ can be separated by the $\mathrm{FO}^{\mathrm{PREF}}$-sentence of quantifier depth 3 stating the existence of two consecutive a's.

Lemma 3. *Let u, v be words such that $\langle u \rangle^k_{FO^{PREF}} = \langle u \cdot v \rangle^k_{FO^{PREF}} = \tau$. Then for every prefix w of v, $\langle u \cdot w \rangle^k_{FO^{PREF}} = \tau$.*

This lemma has a straightforward consequence in the case of infinite words: for every infinite word u, there exists some $\tau \in \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ and an index $n \in \mathbb{N}$ such that for every $m \geq n$, $u[0 \dots m]$ has type τ. Indeed, $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ is finite, thus there must be some type appearing infinitely often in the prefixes of u. Lemma 3 ensures that such a type is unique. We refer to this type as the *stationary type* of u.

Next, we prove that the stationary type of an infinite word is none other than its own type:

Lemma 4. *Let u be an infinite word with stationary type $\tau \in \mathrm{Types}^k_{FO^{PREF}}$. Then $\langle u \rangle^k_{FO^{PREF}} = \tau$.*

Combining those results we get the following:

Corollary 5. *Let $k \in \mathbb{N}$, u be an infinite word, and $\tau = \langle u \rangle^k_{FO^{PREF}}$. Then there exists $n \in \mathbb{N}$ such that for all $m > n$, $u[0 \dots m]$ also has type τ.*

Let us now try to understand the structure of $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$. We consider the binary relation $\rightharpoonup_k$ defined on $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ as follows: $\tau \rightharpoonup_k \tau'$ if and only if there exists two finite words u and v such that $\langle u \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau$ and $\langle uv \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau'$. We refer to $(\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}, \rightharpoonup_k)$ as the *graph of FO^{PREF}-types* of depth k.

The following lemmas break down its properties. First, the choice of u and v above does not really matter:

Lemma 6. *Let $k \geq 1$, let τ and τ' be such that $\tau \rightharpoonup_k \tau'$ and let w be a finite word such that $\langle w \rangle^k_{FO^{PREF}} = \tau$. There exists some finite word w' such that $\langle ww' \rangle^k_{FO^{PREF}} = \tau'$.*

Second, $\rightharpoonup_k$ is an order:

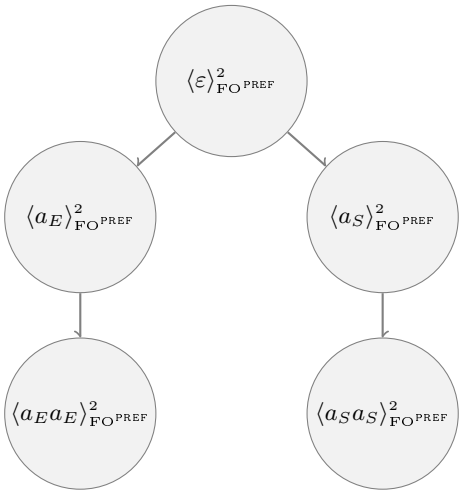

Fig. 1: A partial representation of $\mathrm{Types}^2_{\mathrm{FO}^{\mathrm{PREF}}}$ for $\Sigma_E = \{a_E\}$ and $\Sigma_S = \{a_S\}$. Types of words containing both a_E and a_S have been omitted, as data classes cannot have such a type.

Lemma 7. *The binary relation $\rightharpoonup_k$ is an order on $\mathrm{Types}^k_{FO^{\mathrm{PREF}}}$, with minimum $\langle\varepsilon\rangle^k_{FO^{\mathrm{PREF}}}$.*

From those two lemmas, we conclude that $(\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}, \rightharpoonup_k)$ can be seen as a finite directed tree rooted in $\langle\varepsilon\rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$. This is illustrated in Figure 1.

3 Synthesis and token games

In this section we define the standard synthesis game, and then give equivalent games that are more suitable for our purpose.

3.1 Standard synthesis game

Given a formula φ, the *standard synthesis game* is a game played between two players, System and Environment, who collaborate to create a data word. System's goal is to make the created data word satisfy φ, while Environment wants to falsify it. Formally, a *strategy* for System is a function $\mathcal{S} : \mathrm{DW}^{\mathbb{P}}_{\Sigma} \to (\Sigma_S \times \mathbb{P}_{\mathrm{S}}) \cup \{\varepsilon\}$ which given a data word created so far (the *history*) returns either an action and process to play on, or passes its turn on output ε. A data word $\mathfrak{w} = (a_0, p_0)(a_1, p_1)\ldots$ is *compatible* with $\mathcal{S}$ if for all i, $a_i \in \Sigma_S$ implies $\mathcal{S}(\mathfrak{w}[0\ldots i-1]) = (a_i, p_i)$. Furthermore, $\mathfrak{w}$ is *fair* with $\mathcal{S}$ if either $\mathfrak{w}$ is finite and $\mathcal{S}(\mathfrak{w}) = \varepsilon$, or $\mathcal{S}(\mathfrak{w}[0\ldots i]) \neq \varepsilon$ for infinitely many $i \in \mathbb{N}$ implies $a_i \in \Sigma_S$ for infinitely many $i \in \mathbb{N}$. Intuitively, a fair data word prevents the pathological case where System wants to do some action but Environment forever prevents it

by continually playing its own actions instead. Strategy $\mathcal{S}$ is said to be *winning* if all compatible and fair data words satisfy φ. System wins a synthesis game if there exists a winning strategy for System.

The *existential synthesis problem* asks, for a given alphabet Σ and formula φ, whether there exists a set of processes $\mathbb{P} = \mathbb{P}_{\mathrm{S}} \uplus \mathbb{P}_{\mathrm{E}}$ such that System wins the corresponding synthesis game. In the case of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$, since the logic can only compare process identities with respect to equality, it is easy to see that the actual sets $\mathbb{P}_{\mathrm{S}}$ and $\mathbb{P}_{\mathrm{E}}$ do not matter: only their cardinality does. With that in mind, we slightly reformulate the existential synthesis problem to ask whether there exists a pair $(n_S, n_E) \in \mathbb{N}^2$ such that System wins the synthesis game for all sets of processes $\mathbb{P}_{\mathrm{S}}$ and $\mathbb{P}_{\mathrm{E}}$ of respective size n_S and n_E. If (n_S, n_E) is such a pair, we say that System has a (n_S, n_E)-winning strategy for φ.

3.2 Symmetric game

Notice that the standard synthesis game is asymmetric: while Environment can interrupt System at any point (provided that System gets the possibility to play infinitely often if they want to), System does not choose exactly the timing of their moves. In particular, System is never guaranteed to be able to play successive moves in the game. Let us define a variation of the game, which turns out to be equivalent with respect to the logic $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$, in which System can play arbitrarily many successive moves. This makes the game symmetric, and will make the following proofs simpler.

The *symmetric game* is defined in the same way as the standard game, with the following exceptions. Instead of a function from $\mathrm{DW}_{\Sigma}^{\mathbb{P}}$ to $(\Sigma_S \times \mathbb{P}_{\mathrm{S}}) \cup \{\varepsilon\}$, a strategy for System is now a function from $\mathrm{DW}_{\Sigma}^{\mathbb{P}}$ to $(\Sigma_S \times \mathbb{P}_{\mathrm{S}})^\star \cup \{\varepsilon\}$, i.e. System is allowed to play an arbitrary (but finite) amount of actions at once. Moreover, players strictly alternate, starting with Environment. The rest is defined as in the standard game. It is obvious that System has an easier time winning the symmetric game than the standard game. It turns out the symmetric game offers System no advantage when the relative positions of the classes are incomparable:

Lemma 8. *Let φ be a $FO^{\mathrm{PREF}}[\lesssim]$-sentence, and $n_S, n_E \in \mathbb{N}$. System has an (n_S, n_E)-winning strategy for φ in the symmetric game if and only if System has an (n_S, n_E)-winning strategy for φ in the standard game.*

Note that when positions between classes can be compared, the standard and the symmetric game do not necessarily agree on who wins for a given formula: consider for instance the formula making System the winner if they manage to play twice in a row: the symmetric game is easily won by System, but is won by Environment in the standard setting.

3.3 Token game on words

Our goal is to give an alternative game played on a finite arena that would still be equivalent to the symmetric game. As an intermediate step, consider the

(infinite) graph of all finite words over Σ: $\mathcal{A}_\Sigma = (\Sigma^\star, \varepsilon, \Delta)$ where $\Sigma^\star$ is the set of nodes, ε is the initial node and $\Delta \subseteq \Sigma^\star \times \Sigma \to \Sigma^\star$ is the transition function such that $\Delta(w, a) = w \cdot a$. We use this graph as an arena over which a number of tokens are located, each token representing one process and its location being the history of what has been played on this process. We have two sets of System and Environment tokens, numbering n_S for System tokens and n_E for Environment tokens, all of which are initially placed in the ε node. Alternatively and starting from Environment, each player picks one of their token, move it along one edge, and repeat those two operations a finite amount of times. A player can also opt not to move any of its tokens. Then the other player does the same, and this goes on forever. The winning condition is defined by the formula φ: given a play, we take the limit word reached by each token, and see if the collection of those words satisfy φ. It is easy to see that this game is simply a different view of the symmetric game: playing a word $w = (a_0, p_0) \dots (a_n, p_n)$ is equivalent to picking the token representing p_0, moving it along the a_0-labeled edge, and so on. Thus System wins in the symmetric game if and only if System wins the token game on words, for any choice of token sets of correct sizes.

3.4 Token game on types

We already established as a consequence of Lemma 2 that we do not actually need to keep track of the full history of each token to know whether φ (which has depth k) is satisfied; counting how many tokens (up to threshold k) there are for each k-type of $\mathrm{FO}^{\mathrm{PREF}}$ is enough to decide. Therefore, the final step is to use the finite graph of $\mathrm{FO}^{\mathrm{PREF}}$-types as an arena instead of using the infinite graph of all words. As for the acceptance condition, the formula φ is abstracted by a set α_φ of k-counting functions of the form $\kappa : \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}} \to \{0, 1, \dots, k-1, k+\}$. Each such function gives a count of how many tokens (up to k) can be found in each type, and the acceptance condition α_φ is exactly the set of those functions that satisfy φ.

Let the $\mathrm{FO}^{\mathrm{PREF}}$-arena of depth k be $\mathcal{A}_k = (\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}, \langle\varepsilon\rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}, \rightharpoonup_k)$ where $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ is the set of nodes, $\langle\varepsilon\rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$ is the initial node, and $\rightharpoonup_k$ is the transition function as defined in Section 2.3. Given a pair $(n_S, n_E) \in \mathbb{N}^2$, let us fix two arbitrary disjoint sets of System and Environment *tokens* $\mathbb{T}_\mathrm{S}$ and $\mathbb{T}_\mathrm{E}$ of sizes n_S and n_E respectively, and let $\mathbb{T}$ denote their union. The *token game* over $\mathbb{T}$ for a $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formula φ of depth k is given by the tuple $\mathfrak{G}^{\mathbb{T}}_\varphi = (\mathbb{T}, \mathcal{A}_k, \alpha_\varphi)$.

A *configuration* of this game is a mapping $C : \mathbb{T} \to \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ indicating where each token lies in the arena. The initial configuration C_0 maps every token to the initial type $\tau_0 = \langle\varepsilon\rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$. Starting from Environment, players alternatively pick a number of their respective tokens and move them in the arena following transitions from $\rightharpoonup_k$. A *move* for System (resp. Environment) is a mapping $m_\mathrm{S} : \mathbb{T}_\mathrm{S} \to \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ (resp. $m_\mathrm{E} : \mathbb{T}_\mathrm{E} \to \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$) indicating where to move each token such that for all $t \in \mathbb{T}_\mathrm{S}$, $C(t) \rightharpoonup_k m_\mathrm{S}(t)$ (and similarly for Environment). In particular, in a given configuration C, an empty System move is simply a move equal to C restricted to $\mathbb{T}_\mathrm{S}$, indicating that all System tokens should stay where they are.

Then a *play* π is a sequence of configuration and moves starting from an Environment move and alternating players: $\pi = C_0 \xrightarrow{m^0_{\mathrm{E}}} C_1 \xrightarrow{m^1_{\mathrm{S}}} C_2 \xrightarrow{m^2_{\mathrm{E}}} \cdots$ such that each new configuration is the result of applying the previous move to the previous configuration. Let $\mathrm{Plays}^k_{\mathbb{T}}$ denote the set of plays. A play is *maximal* if it is infinite.

For a given token $t \in \mathbb{T}$, a maximal play π generates a sequence of types $\langle \varepsilon \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau_0 \rightharpoonup_k \tau_1 \rightharpoonup_k \dots$ such that $\tau_i = C_{2i}(t)$. Note that it is fine to skip every other configuration, as a token can only be moved during either a System or Environment move but not both. This infinite sequence eventually loops in some type τ_t forever due to the graph of types being a finite tree. The *limit configuration* for π, denoted by C^π_∞, is the configuration that returns τ_t for every token t. Note that there must exists some $i \geq 0$ such that for all $j > i$, $C_j = C^\pi_\infty$. We slightly abuse notations and denote by $\pi(t)$ the type of token t in either the last configuration of π if it is finite or its limit configuration if it is infinite. A play is *winning* if its limit configuration satisfies the acceptance condition α_φ, that is if there is a function $\kappa \in \alpha_\varphi$ such that for all $\tau \in \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$, $|\{t \in \mathbb{T} \mid \pi(t) = \tau\}| = \kappa(\tau)$ if $\kappa(\tau) < k$, or $|\{t \in \mathbb{T} \mid \pi(t) = \tau\}| \geq k$ if $\kappa(\tau) = k+$.

A *strategy* for System is a function $\mathcal{S}$ that given a play returns a System move. A play is *compatible* with $\mathcal{S}$ if all System moves in that play are those given by $\mathcal{S}$. A strategy for System is winning if all maximal plays compatible with it are winning. Finally, we say that a pair $(n_S, n_E) \in \mathbb{N}^2$ is winning for System if System has a winning strategy in the token game (for any choice of token sets $\mathbb{T}_{\mathrm{S}}$ and $\mathbb{T}_{\mathrm{E}}$ of corresponding sizes).

Lemma 9. *A pair (n_S, n_E) is winning for System in the token game for φ if and only if System has a (n_S, n_E)-winning strategy for φ in the standard synthesis game.*

For a fixed pair (n_S, n_E), the token game is a finite, albeit very large, game. Remember that our goal is to find whether there exists such a pair that is winning. We show in the next section how to reduce the search to a (large but) finite space.

4 Double cutoff for solving the synthesis problem

Recall that in terms of expressive power, $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ is located somewhere between $\mathrm{FO}[\sim]$ (whose existential synthesis problem is known to be decidable [9]) and $\mathrm{FO}[\sim, <]$ (for which is it undecidable already when restricting to two variables, i.e. for $\mathrm{FO}^2[\sim, <]$ [9]).

In this section, we make a step towards closing the gap by proving our main result:

Theorem 10. *The existential synthesis problem for $FO^{PREF}[\lesssim]$ is decidable.*

To prove Theorem 10 we follow a double cutoff strategy. We first show in Section 4.1 that there is no point in considering too many tokens for Environment, where the bound depends on the quantifier depth k of the formula φ but,

importantly, not on the number of System tokens. As a second step, we prove in Section 4.2 that given a fixed number of Environment tokens (which is a reasonable assumption in view of the previous point), one can restrict one's study to a space where the number of System tokens is bounded by a function of k and the number of Environment tokens. The reasoning is detailed in Section 4.3.

4.1 Having more tokens makes things easier for Environment

First, we prove that beyond some threshold, Environment can only benefit from having more tokens. Let us stress that this is not true when the number of tokens is small, as witnessed by the formula φ stating the existence of at least two Environment tokens: in that case, System can benefit from Environment having more tokens.

Lemma 11. *For every $k \in \mathbb{N}$, there exists some $f_E(k)$ such that for any $n_S \in \mathbb{N}$, any $n_E \geq f_E(k)$, and any $FO^{PREF}[\lesssim]$-sentence φ of depth k, if $(n_S, n_E + 1)$ is winning for System then (n_S, n_E) is winning for System.*

Proof. Let $k \in \mathbb{N}$ and let φ be a $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$-sentence of depth k. Let us fix three token sets $\mathbb{T}_{\mathrm{S}}^{n_S}$, $\mathbb{T}_{\mathrm{E}}^{n_E}$, and $\mathbb{T}_{\mathrm{E}}^{n_E+1}$ of sizes n_S, n_E, n_E +1 respectively. We note $\mathbb{T} = \mathbb{T}_{\mathrm{S}}^{n_S} \uplus \mathbb{T}_{\mathrm{E}}^{n_E}$ and $\mathbb{T}_+ = \mathbb{T}_{\mathrm{S}}^{n_S} \uplus \mathbb{T}_{\mathrm{E}}^{n_E+1}$. Let $\mathfrak{G} = (\mathbb{T}, \mathcal{A}_k, \alpha_\varphi)$ be the token game over $\mathbb{T}$ for φ and $\mathfrak{G}_+ = (\mathbb{T}_+, \mathcal{A}_k, \alpha_\varphi)$ the same over $\mathbb{T}_+$. We show how to build a winning strategy for System in $\mathfrak{G}$ from a winning strategy in $\mathfrak{G}_+$. But first, let us define some useful properties.

For all types $\tau \in \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ we define the height of τ, denoted by $h(\tau)$, as its height in the tree of types, e.g. $h(\tau) = 0$ for any leaf in the tree. Let $h_{\max}$ denote the height of the root $\tau_0 = \langle\varepsilon\rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$.

In any given configuration obtained by following a play in $\mathfrak{G}$, we say that a type τ is *large* in that configuration if there are at least k Environment tokens in τ. Intuitively, this means that the acceptance condition α_φ cannot differentiate between a configuration with a large type τ and the same configuration with even more tokens in τ. Therefore, if we can ensure that System has a strategy in $\mathfrak{G}$ that simulates the $\mathfrak{G}_+$ winning strategy while always keeping the missing Environment token in a large type, then that strategy would also be winning as α_φ (which is the same in both $\mathfrak{G}$ and $\mathfrak{G}_+$) has no way of distinguishing them.

To that end, we define what it means to have a *huge* number of Environment tokens in one type τ. This is given by a lower bound $F(\tau) = k \cdot |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}|^{h(\tau)}$ that depends only on k and the height of τ. It guarantees the following properties:

1. A type that is huge is *a fortiori* large.
2. If τ is a huge type with $h(\tau) = 0$, it contains at least $F(\tau) = k$ Environment tokens. And since it is a leaf, all those tokens will stay in this type forever. Thus τ will remain large from this point on.
3. If τ is a huge type with height greater than 0, after any Environment move either τ still remains large, or by the pigeonhole principle there exists another type τ' whose height is strictly lower than $h(\tau)$, that can be reached from τ (i.e. such that $\tau \rightharpoonup_k \tau'$), and such that the number of Environment tokens in τ' is greater than $F(\tau')$, ensuring τ' is also huge.

We then define $f_E(k) = F(\tau_0) = k \cdot |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}|^{h_{\max}}$. Note that it depends only on k.

By these definitions, and since we assume that $n_E \geq f_E(k)$, τ_0 is huge in the initial configuration because all Environment tokens start in type τ_0. By the third property, this means that after any move by Environment, either τ_0 is still large, or there is (at least) one huge type τ_1 reachable from τ_0. This can be repeated until either a type of height 0 is reached, which will be large forever according to the second property, or we stay in the same large type forever. Formally, we define inductively a function $lt : \mathrm{Plays}^k_{\mathbb{T}} \to \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ (lt for "large type") such that $lt(C_0) = \tau_0$, $lt(\pi \xrightarrow{m_{\mathrm{S}}} C) = lt(\pi)$, and $lt(\pi \xrightarrow{m_{\mathrm{E}}} C) = \tau$ where τ is either a minimal (in terms of height) type such that $lt(\pi) \rightharpoonup_k \tau$ and τ is huge in C if such a type exists, or $\tau = lt(\pi)$ otherwise. This is well-defined due to the above-mentioned third property, and we easily obtain that $lt(\pi)$ is large in the last configuration of π for any play π.

Now assume $\mathcal{S}_+$ is a winning strategy for System in $\mathfrak{G}_+$. We define a strategy $\mathcal{S}$ for System in $\mathfrak{G}$ using lt and additionally maintaining a play π_+ of $\mathfrak{G}_+$ with the following invariant: for all plays π of $\mathfrak{G}$ that are $\mathcal{S}$-compatible, π_+ is a $\mathcal{S}_+$-compatible play such that the configuration reached after π_+ has the same number of tokens in each type as the configuration reached after π, plus one extra Environment token in $lt(\pi)$. The strategy $\mathcal{S}$ is simply defined as $\mathcal{S}(\pi) = \mathcal{S}_+(\pi_+)$, i.e. it mimics the actions of $\mathcal{S}_+$ on play π_+. Assuming that π_+ is properly defined and that the previously mentioned invariant holds, it is then easy to prove that $\mathcal{S}$ is winning. Indeed, for every configuration that can be reached from a $\mathcal{S}$-compatible play π, there is an almost similar configuration that can be reached by following π_+, which is $\mathcal{S}_+$-compatible, with the only difference being one extra Environment token in the type designated by lt. Since this type is large by definition of lt, α_φ cannot distinguish between the two configurations. Thus, for any maximal play π compatible with $\mathcal{S}$, its limit configuration is indistinguishable from the limit configuration of π_+, which satisfies α_φ by assumption of $\mathcal{S}_+$ being winning. This proves that $\mathcal{S}$ is indeed a winning strategy for System in $\mathfrak{G}$.

It only remains to explain how π_+ is defined and show it satisfies the invariant. Without loss of generality, assume that $\mathbb{T}^{n_E+1}_{\mathrm{E}} = \mathbb{T}^{n_E}_{\mathrm{E}} \uplus \{\gamma\}$. We strengthen the second part of the invariant so that the configuration reached after π_+ is such that every token in $\mathbb{T}^{n_E}_{\mathrm{E}}$ is in the same type as in the configuration reached after π, and with γ being the extra token in type $lt(\pi)$. Initially this play π_+ is the empty play C_0 of $\mathfrak{G}_+$, which trivially satisfies both conditions. Suppose now that π is a $\mathcal{S}$-compatible play and π_+ is a $\mathcal{S}_+$-compatible play that also satisfies the (strengthened) second part of the invariant.

- On any System move m_{S} in $\mathfrak{G}$ leading to play $\pi \xrightarrow{m_{\mathrm{S}}} C$, we simply update π_+ to $\pi_+ \xrightarrow{m_{\mathrm{S}}} C_+$ where C_+ is the result of applying m_{S} to the last configuration of π_+. Since $\mathcal{S}$ mimics $\mathcal{S}_+$, if $\pi \xrightarrow{m_{\mathrm{S}}} C$ is $\mathcal{S}$-compatible, then $\pi_+ \xrightarrow{m_{\mathrm{S}}} C_+$ is $\mathcal{S}_+$-compatible. Moreover, by induction hypothesis, the last configuration of π_+ is the last configuration of π with the extra token γ in $lt(\pi)$. By applying the same System move m_{S} to both, we easily obtain that C_+ is the same as C plus γ in $lt(\pi \xrightarrow{m_{\mathrm{S}}} C) = lt(\pi)$.

– On an Environment move m_{E} in $\mathfrak{G}$ resulting in play $\pi \xrightarrow{m_{\mathrm{E}}} C$, let $\tau = lt(\pi)$ and $\tau' = lt(\pi \xrightarrow{m_{\mathrm{E}}} C)$. Let m_{E}^{γ} be the Environment move that only affects γ by moving it from τ to τ' (the move can be empty if $\tau = \tau'$). We know such a move is possible because by definition of lt we have that $\tau \rightharpoonup_k \tau'$. Then with m_{E}^{+} being the Environment move combining m_{E} and m_{E}^{γ}, we update π_+ to $\pi_+ \xrightarrow{m_{\mathrm{E}}^{+}} C_+$. It is trivially still $\mathcal{S}_+$-compatible since no System move has been made. The extra token γ was moved to the new large type given by lt, and all other Environment tokens made the same moves as in $\pi \xrightarrow{m_{\mathrm{E}}} C$, so the invariant still holds.

Thus π_+ is properly defined and always satisfy the required invariant, which concludes the proof. □

4.2 Too many tokens are useless to System

Dually, we prove in the appendix the following lemma, stating that only so many tokens can help the System win; beyond that point, no amount of additional tokens can turn the table and change a losing game into a winning one.

Lemma 12. *For every $k, n_E \in \mathbb{N}$, there exists some $f_S(k, n_E)$ such that for any $n_S \geq f_S(k, n_E)$, if System has an $(n_S + 1, n_E)$-winning strategy in a token game of depth k, then System has an (n_S, n_E)-winning strategy in that game.*

Note that contrary to Lemma 11, where the bound depends only on k, here $f_S(k, n_E)$ depends both on k and the number of tokens of Environment. This cannot be avoided, as showcased by the following example.

Example 13. Remember formula $\varphi_{1a_E \leftrightarrow 1a_S}$ from Example 1. We can show that the (n_S, n_E)-game for $\varphi_{1a_E \leftrightarrow 1a_S}$ cannot be won by System when $n_S < n_E$, as exemplified in Figure 2. Note that we use for simplicity's sake the representation of $\mathrm{Types}^2_{\mathrm{FO}^{\mathrm{PREF}}}$ from Figure 1 when we should consider $\mathrm{Types}^3_{\mathrm{FO}^{\mathrm{PREF}}}$, as $\varphi_{1a_E \leftrightarrow 1a_S}$ has depth 3 – this does not matter as $\varphi_{1a_E \leftrightarrow 1a_S}$ cannot distinguish $\langle a_E a_E \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ from $\langle a_E a_E a_E \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$.

Starting from the initial configuration depicted in Figure 2a, Environment can move one of their tokens from $\langle \varepsilon \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ to $\langle a_E \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ (Figure 2b), forcing System to answer by moving on of their tokens from $\langle \varepsilon \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ to $\langle a_S \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ (Figure 2c) in order to satisfy the win condition of $\varphi_{1a_E \leftrightarrow 1a_S}$. System could also directly move down other tokens from $\langle \varepsilon \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ to $\langle a_S a_S \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$, but this would only make things worse.

Then by moving the same token to $\langle a_E a_E \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ as in Figure 2d, Environment would force System to move as well their first token to $\langle a_S a_S \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ (cf. Figure 2e). Repeating this sequence a total of n_S times on different tokens would end up "using" all of System's tokens, which would all end up in $\langle a_S a_S \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$, as illustrated in Figure 2g. Environment then only needs to move one last token to $\langle a_E \rangle^2_{\mathrm{FO}^{\mathrm{PREF}}}$ to falsify the win conditions for $\varphi_{1a_E \leftrightarrow 1a_S}$.

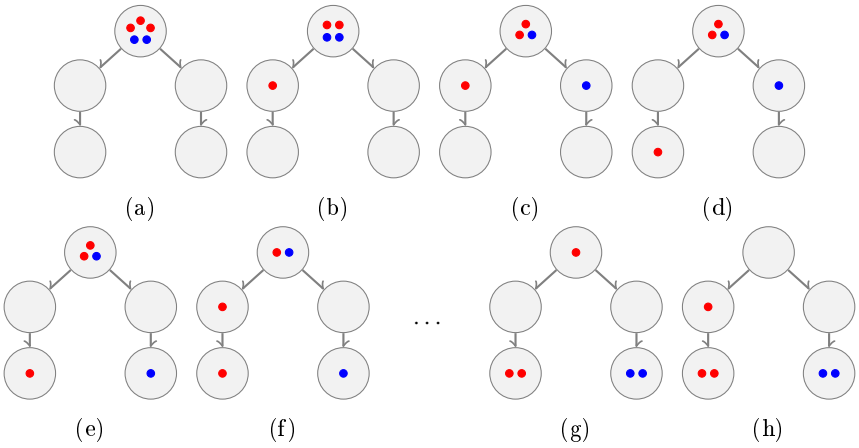

Fig. 2: System may need at least as many tokens as Environment to win.

Although the insight gained in the proof of Lemma 11 is useful when considering the proof of Lemma 12, the latter is much more involved. Let us nevertheless give the key ideas of the proof. This time, the goal is to convert an $(n_S + 1, n_E)$-winning strategy $\mathcal{S}_+$ for System in a token game of depth k to an (n_S, n_E)-winning strategy $\mathcal{S}$ for the same game.

In order to adapt $\mathcal{S}_+$ to the situation where System has one less token, we will track a *ghost* token. The central idea of the proof is to guarantee at all time that the ghost shares its type with many other tokens, so that its presence or absence is of no import to the winning conditions of the game. Up to that point, the proof is very similar to that of Lemma 11. The main difference in this case is that the ghost is not a fixed token: its identity among the $n_S + 1$ System tokens may vary depending on the way the play unfolds. But once again, if one starts with enough System tokens and is careful in the tracking of this ghost token, it is possible to "hide it" among many others throughout the entire play.

4.3 Solving the synthesis problem

Remember that our goal is to decide whether there exists a pair $(n_S, n_E) \in \mathbb{N}^2$ that is winning for a given formula φ of depth k. Using Lemmas 11 and 12, we have shown that we can restrict the search space to the set $N = \{(n_S, n_E) \in \mathbb{N}^2 \mid n_E \leq f_E(k) \wedge n_S \leq f_S(k, n_E)\}$ (which is finite and computable) in the sense that if there is no winning pair in N, then there will be no winning pair in $\mathbb{N}^2$.

Recall that for a fixed pair (n_S, n_E), the corresponding token game is a finite game with finite configurations. As such, it can be solved by seeing it as a game on the graph of configurations, with a Büchi winning condition where accepting configurations are exactly those that satisfy α_φ. Therefore, the synthesis problem can be solved by iterating on all pairs in N and solving the token game for that

pair, and then accepting the first winning pair found if there is one, and rejecting if no such pair is found.

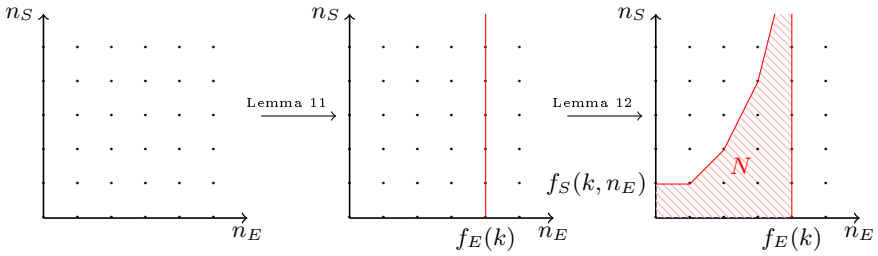

Fig. 3: Bounding the search space for winning pairs.

5 Conclusion

We have shown that the synthesis problem is decidable for $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$, the prefix first-order logic on data words. Our proof is based on successively bounding the number of Environment and System processes that are relevant for solving the problem, thus restricting the search space to a finite set.

The correctness of those bounds heavily relies on the nice properties exhibited by $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$. The most important of them is that $\mathrm{FO}^{\mathrm{PREF}}$-types on regular words form a tree, which would not be the case with the unrestricted $\mathrm{FO}[\lesssim]$. We show in the appendix that $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ is actually the finest restriction enjoying this property, in the sense that $\mathrm{FO}^{\mathrm{PREF}}$-types correspond exactly to connected components in the graph of types of $\mathrm{FO}[\lesssim]$. To the best of our knowledge, this natural restriction and the properties it enjoys have not been studied anywhere else.

On the synthesis side, our result narrows the gap between decidability and undecidability of first-order logic fragments on data words. Previous results had shown that adding unrestricted order to the allowed predicates immediately lead to undecidability, greatly reducing the kind of specifications that could be written. It turns out that the limited kind of order added by the predicate $\lesssim$ does not share the same outcome, so we obtain a fragment with more expressivity than before and for which synthesis is still decidable. The decidability for the full $\mathrm{FO}[\lesssim]$ remains open; our technique for tracking a missing ghost token cannot be easily adapted in that setting as the type structure for that logic is not as nice as in the $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ setting. We conjecture that it remains decidable, and leave this as potential future works.

References

1. Bérard, B., Bollig, B., Lehaut, M., Sznajder, N.: Parameterized synthesis for fragments of first-order logic over data words. In: Foundations of Software Science and Computation Structures FOSSACS. Springer (2020)
2. Bojanczyk, M., Muscholl, A., Schwentick, T., Segoufin, L., David, C.: Two-variable logic on words with data. In: 21th IEEE Symposium on Logic in Computer Science LICS (2006)
3. Büchi, J.R., Landweber, L.H.: Solving sequential conditions by finite-state strategies. Transactions of the American Mathematical Society (1969)
4. Church, A.: Applications of recursive arithmetic to the problem of circuit synthesis. In: Summaries of the Summer Institute of Symbolic Logic – Volume 1 (1957)
5. Demri, S., d'Souza, D., Gascon, R.: Temporal logics of repeating values. Journal of Logic and Computation **22**(5), 1059–1096 (2012)
6. Demri, S., Lazić, R.: Ltl with the freeze quantifier and register automata. ACM Transactions on Computational Logic (TOCL) **10**(3), 1–30 (2009)
7. Exibard, L., Filiot, E., Khalimov, A.: A generic solution to register-bounded synthesis with an application to discrete orders. arXiv preprint arXiv:2205.01952 (2022)
8. Figueira, D., Praveen, M.: Playing with repetitions in data words using energy games. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS (2018)
9. Grange, J., Lehaut, M.: First order synthesis for data words revisited. arXiv preprint arXiv:2307.04499 (2023)
10. Kaminski, M., Francez, N.: Finite-memory automata. Theoretical Computer Science **134**(2), 329–363 (1994)
11. Kara, A.: Logics on data words (2016)
12. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Transactions on Computational Logic (TOCL) **5**(3), 403–435 (2004)
13. Pneuli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science. pp. 746–757. IEEE (1990)

A Proofs for Section 2 (Prefix first-order logic)

When we defined $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ in Section 2.2, we did not considered constant symbols, as the game from Section 3.1 does not deal with constants – note that considering constants in these games would be cumbersome without any increase in expressivity, as one can already implement them by adding new letters to the alphabets Σ_S and Σ_E and ensure their uniqueness in the formula φ.

It will however be convenient to allow the use of constant symbols in $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$, in order to make the proof of Proposition 15 smoother. Let us thus extend the definition of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ in the following way.

We add to the vocabulary of data words a finite set of constant symbols C, which we partition into C^{proc} (the *process constants*), C^{bnd} (the *bounding constants*) and C^{pref} (the *prefix constants*). We restrict our attention to data words in which the interpretation of these constant symbols is such that

- each process constant is interpreted as a process,
- each bounding and prefix are interpreted as positions of the data word,
- no two bounding constants are in the same data class, and
- each prefix constant is in the same data class as some bounding constant, which appear at a later position (with respect to $\leq$).

We define as before the formulas of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ by induction, taking now into account the constant symbols. Again, $\varphi(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}})$ means that the free variables of φ belong to the pairwise disjoint union X of the three sets X^{proc} (the *process variables*), X^{bnd} (the *bounding variables*) and X^{pref} (the *prefix variables*). The additions to the original definition are highlighted.

$$
\begin{aligned}
& x = y && (x, y \in X \cup C) \\
\mid\ & \mathbb{P}_{\mathrm{S}}(x) \quad \mid \mathbb{P}_{\mathrm{E}}(x) && (x \in X^{\mathrm{proc}} \cup C^{\mathrm{proc}}) \\
\mid\ & a(x) && (x \in X^{\mathrm{bnd}} \cup X^{\mathrm{pref}} \cup C^{\mathrm{bnd}} \cup C^{\mathrm{pref}}, a \in \Sigma) \\
\mid\ & x \lesssim y && (x, y \in X^{\mathrm{pref}} \cup C^{\mathrm{pref}}) \\
\mid\ & x \sim y && ((x, y) \in (X^{\mathrm{proc}} \cup C^{\mathrm{proc}}) \times (X \cup C)) \\
\mid\ & \varphi_1(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) \wedge \varphi_2(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) \\
\mid\ & \neg\psi(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) \\
\mid\ & \exists x, \mathbb{P}(x) \ \wedge\ \psi(X^{\mathrm{proc}} \cup \{x\}; X^{\mathrm{bnd}}; X^{\mathrm{pref}}) && (x \notin X) \\
\mid\ & \exists x, \neg\mathbb{P}(x) \ \wedge \Big(\bigwedge_{y \in X^{\mathrm{bnd}} \cup C^{\mathrm{bnd}}} x \not\sim y \Big) \wedge \psi(X^{\mathrm{proc}}; X^{\mathrm{bnd}} \cup \{x\}; X^{\mathrm{pref}}) && (x \notin X) \\
\mid\ & \exists x,\ x \lesssim y \wedge \psi(X^{\mathrm{proc}}; X^{\mathrm{bnd}}; X^{\mathrm{pref}} \cup \{x\}) && (x \notin X, y \in X^{\mathrm{bnd}} \cup C^{\mathrm{bnd}})
\end{aligned}
$$

As intended, the logic $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ we defined in Section 2.2 is just the restriction of the above definition when the set C of constant symbols is empty.

We write $\mathfrak{w} \equiv_k^{\mathrm{FO}^{\mathrm{PREF}}[\lesssim]} \widehat{\mathfrak{w}}$ when $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ agree on every $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$-sentence of quantifier depth at most k.

As per usual, when introducing a new logic, it is useful to find a characterization via an Ehrenfeucht-Fraïssé game. We introduce the suiting Ehrenfeucht-Fraïssé game below, and show its equivalence to $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ in Proposition 15.

Definition 14. *Let k be an integer, and $\mathfrak{w}, \widehat{\mathfrak{w}}$ be two (finite or infinite) data words on the same alphabet with constant set C. The* k-round prefix Ehrenfeucht-Fraïssé game *on data words is played by two players: the Spoiler, who tries to highlight the differences between $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$, and the Duplicator, whose goal is to show that $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ look alike. There are three kinds of pebbles available to place on each data word, to which both players have access:* process pebbles p_i^{proc} *(in $\mathfrak{w}$) and $\widehat{p}_i^{proc}$ (in $\widehat{\mathfrak{w}}$),* bounding pebbles p_i^{bnd} *(in $\mathfrak{w}$) and $\widehat{p}_i^{bnd}$ (in $\widehat{\mathfrak{w}}$) and* prefix pebbles p_i^{pref} *(in $\mathfrak{w}$) and $\widehat{p}_i^{pref}$ (in $\widehat{\mathfrak{w}}$), for $i \in \{1, \dots, k\}$. These pebbles will be placed on elements of $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ by the players. With a slight abuse of notation, we will sometimes identify a pebble with the element on which it has been placed.*

In round i, the Spoiler starts by choosing one of the two data words. Let us assume first that they choose to play in $\mathfrak{w}$. Then the Spoiler has three possibilities: they can make a process move*, a* bounding move *or a* prefix move *by respectively placing p_i^{proc}, p_i^{bnd} or p_i^{pref} on some element of $\mathfrak{w}$, with the following restriction.*

1. *In a process move, p_i^{proc} must be placed on a process element of $\mathfrak{w}$.*
2. *In a bounding move, pebble p_i^{bnd} must be placed on a position of $\mathfrak{w}$ whose class does not already possess a bounding pebble, nor a bounding constant.*
3. *In a prefix move, the Spoiler must place p_i^{pref} on an position e of $\mathfrak{w}$ such that either some bounding pebble p_j^{bnd} (for $j < i$) or some bounding constant is in the same class as e and larger than e with respect to $\leq$.*

Then the Duplicator responds by placing the corresponding pebble in $\widehat{\mathfrak{w}}$ (i.e. $\widehat{p}_i^{proc}$ if the Spoiler placed p_i^{proc}, $\widehat{p}_i^{bnd}$ if they placed p_i^{bnd}, or $\widehat{p}_i^{pref}$ if they placed p_i^{pref}) with the same constraints. Conversely, if the Spoiler decided to play in $\widehat{\mathfrak{w}}$, then the Duplicator would respond by placing either p_i^{proc}, p_i^{bnd} or p_i^{pref} (depending on whether the Spoiler played $\widehat{p}_i^{proc}$, $\widehat{p}_i^{bnd}$ or $\widehat{p}_i^{pref}$) in $\mathfrak{w}$.

At the end of round i, let us assume that pebbles $p_1, \dots, p_i$ have been played on $\mathfrak{w}$ (where p_j is one of p_j^{proc}, p_j^{bnd} or p_j^{pref}, depending on whether round j saw a process move, a bounding move or a prefix move), and $\widehat{p}_1, \dots, \widehat{p}_i$ on $\widehat{\mathfrak{w}}$. Then the Spoiler immediately wins if any of the following equivalences fail, where $\alpha, \widehat{\alpha}$ are either the interpretations of the same constant symbol in $\mathfrak{w}$, $\widehat{\mathfrak{w}}$, or the elements on which $p_j, \widehat{p}_j$ (for some $j \in \{1, \dots, i\}$) have been placed (and similarly for $\beta, \widehat{\beta}$):

- $\mathfrak{w} \models \alpha = \beta$ *iff* $\widehat{\mathfrak{w}} \models \widehat{\alpha} = \widehat{\beta}$
- $\mathfrak{w} \models \mathbb{P}_{\mathrm{S}}(\alpha)$ *(resp.* $\mathbb{P}_{\mathrm{E}}(\alpha)$*) iff* $\widehat{\mathfrak{w}} \models \mathbb{P}_{\mathrm{S}}(\widehat{\alpha})$ *(resp.* $\mathbb{P}_{\mathrm{E}}(\widehat{\alpha})$*)*
- $\mathfrak{w} \models a(\alpha)$ *iff* $\widehat{\mathfrak{w}} \models a(\widehat{\alpha})$*, for* $a \in \Sigma$
- $\mathfrak{w} \models \alpha \lesssim \beta$ *iff* $\widehat{\mathfrak{w}} \models \widehat{\alpha} \lesssim \widehat{\beta}$
- $\mathfrak{w} \models \alpha \sim \beta$ *iff* $\widehat{\mathfrak{w}} \models \widehat{\alpha} \sim \widehat{\beta}$

The Duplicator wins if the game reaches the end of round k and if the Spoiler still doesn't win. In that case, we write $\mathfrak{w} \equiv_k^{\textit{pref-EF}} \widehat{\mathfrak{w}}$.

The prefix Ehrenfeucht-Fraïssé game has been tailored to capture the expressive power of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$:

Proposition 15. *Let k be an integer, and let $\mathfrak{w}$, $\widehat{\mathfrak{w}}$ be two data words on the same alphabet (with constants). Then $\mathfrak{w} \equiv_k^{\textit{pref-EF}} \widehat{\mathfrak{w}}$ if and only if $\mathfrak{w} \equiv_k^{FO^{\textit{PREF}}[\lesssim]} \widehat{\mathfrak{w}}$.*

Proof. We prove this result by induction on k. For $k = 0$, notice that both $\mathfrak{w} \equiv_0^{\text{pref-EF}} \widehat{\mathfrak{w}}$ and $\mathfrak{w} \equiv_0^{\mathrm{FO}^{\mathrm{PREF}}[\lesssim]} \widehat{\mathfrak{w}}$ hold exactly when $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ agree quantifier-free formulas.

Suppose that this equivalence hold for some integer k, and let us consider two data words $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ with constant set C.

Let us first assume that $\mathfrak{w} \equiv_{k+1}^{\text{pref-EF}} \widehat{\mathfrak{w}}$ and show that in that case, $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ agree on all $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ of quantifier depth at most $k+1$. Note that every sentence of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ of quantifier depth at most $k+1$ is a boolean combination of sentences quantifier-free sentences, and of sentences of the form

$$\exists x,\ \mathbb{P}(x)\ \wedge\ \varphi(\{x\};\emptyset;\emptyset), \tag{1}$$

$$\exists x,\ \neg\mathbb{P}(x)\ \wedge\ \Big(\bigwedge_{c\in C^{\mathrm{bnd}}} x \not\sim c\Big)\ \wedge\ \varphi(\emptyset;\{x\};\emptyset)\,, \tag{2}$$

or

$$\exists x,\ x \lesssim c\ \wedge\ \varphi(\emptyset;\emptyset;\{x\}) \qquad \text{where } c \in C^{\mathrm{bnd}}\,, \tag{3}$$

where φ is a $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formula of quantifier depth at most k.
By assumption the Spoiler does not immediately win, hence $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ must agree on all quantifier-free sentences. It is thus enough to show that they agree on all sentences of the form (1), (2) and (3). Assume that $\mathfrak{w}$ satisfies such an existential sentence, witnessed by some $\alpha \in \mathfrak{w}$, and let us prove that $\widehat{\mathfrak{w}}$ also satisfies this formula – the reverse is symmetric. We distinguish between the three cases.
Let us consider the case of a sentence of form (1). By assumption, the Duplicator is able to win the $(k+1)$-round game. In particular, if the Spoiler makes a process move by playing p_1^{proc} on α, then the Duplicator can respond by placing $\widehat{p}_1^{\mathrm{proc}}$ on $\widehat{\alpha}$ (which must be a process of $\widehat{\mathfrak{w}}$ by definition of the game) and still win r rounds. This means, if we add a new process constant symbol to C and interpret it as α in $\mathfrak{w}$ and $\widehat{\alpha}$ in $\widehat{\mathfrak{w}}$, yielding respectively $\mathfrak{w}'$ and $\widehat{\mathfrak{w}}'$, that $\mathfrak{w}' \equiv_k^{\text{pref-EF}} \widehat{\mathfrak{w}}'$. By induction hypothesis, we thus get $\mathfrak{w} \equiv_k^{\mathrm{FO}^{\mathrm{PREF}}[\lesssim]} \widehat{\mathfrak{w}}$. Then $\widehat{\alpha}$ is a process element witnessing that $\widehat{\mathfrak{w}}$ also satisfies the sentence $\exists x, \mathbb{P}(x) \wedge \varphi(\{x\};\emptyset;\emptyset)$.
We deal similarly with formulas of type (2) and (3), the only distinction being that in the former the Spoiler starts by playing a bounding move, while they play a prefix move in the latter.

Conversely, let us prove that if $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ satisfy the same $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$-sentences of quantifier depth at most $k+1$, then the Duplicator wins the $(k+1)$-round game. Let us assume that the Spoiler's first move is a process move by placing p_1^{proc} on α in $\mathfrak{w}$ (the case where they play in $\widehat{\mathfrak{w}}$ is fully symmetric). An easy

induction on the quantifier depth shows that up to equivalence, there are only a finite number of formulas of a given quantifier depth. Let $\varphi(\{x\},\emptyset,\emptyset)$ be the conjunction of all the formulas of quantifier depth at most k satisfied by α in $\mathfrak{w}$. By construction, $\mathfrak{w}$ then satifies the sentence $\exists x, \mathbb{P}(x) \wedge \varphi(\{x\};\emptyset;\emptyset)$, which is also true in $\widehat{\mathfrak{w}}$ as it has quantifier depth $k+1$. Let $\widehat{\alpha} \in \widehat{\mathfrak{w}}$ be a witness to that fact. The Duplicator will place $\widehat{p}_1^{\text{proc}}$ on $\widehat{\alpha}$ (note that this is a valid process move, as $\widehat{\alpha}$ must be a process in $\widehat{\mathfrak{w}}$). Once again, let us enrich the constant set with a new process constant interpreted in $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ respectively as α and $\widehat{\alpha}$, which yield two data words $\mathfrak{w}'$ and $\widehat{\mathfrak{w}}'$. By choice of φ we have $\mathfrak{w}' \equiv_k^{\mathrm{FO}^{\text{PREF}}[\lesssim]} \widehat{\mathfrak{w}}'$ and thus, by induction hypothesis, $\mathfrak{w}' \equiv_k^{\text{pref-EF}} \widehat{\mathfrak{w}}'$, which precisely mean that the Duplicator can win k rounds after the first, and thus can win the $(k+1)$-round games starting with a process move.
When the Spoiler starts with a bounding move on an element α of $\mathfrak{w}$, we consider a formula $\varphi(\emptyset,\{x\},\emptyset)$ characterizing the set of formulas of quantifier depth at most k satisfied by α in $\mathfrak{w}$. Then $\exists x, \neg\mathbb{P}(x) \wedge \Big(\bigwedge_{c\in C^{\text{bnd}}} x \not\sim c\Big) \wedge \varphi(\emptyset;\{x\};\emptyset)$ has quantifier depth $k+1$ and is valid both in $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$. Any existential witness $\widehat{\alpha}$ in $\widehat{\mathfrak{w}}$ is necessarily a valid option for a bounding move by the Duplicator, who responds by placing $\widehat{p}_1^{\text{bnd}}$ on $\widehat{\alpha}$. For the same reason as in the previous case, the Duplicator can the proceed to win k more rounds of the game.
If the Spoiler starts the game with a prefix move in which they place p_1^{pref} on some element α, then the rules of the game ensure the existence of a bounding constant $c \in C$ such that $x \lesssim c$. In this case, we consider the sentence $\exists x, x \lesssim c \wedge \varphi(\emptyset;\emptyset;\{x\})$ (where once again φ subsumes all formulas of quantifier depth at most k satisfied by α in $\mathfrak{w}$) and conclude as before.
All in all, the Duplicator wins the $(k+1)$-round game whenever $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ are such that $\mathfrak{w} \equiv_k^{\mathrm{FO}^{\text{PREF}}[\lesssim]} \widehat{\mathfrak{w}}$, which concludes the proof. □

On (finite or infinite) words, a straightforward adaptation of this game (where there is no process pebble, and only one bounding pebble) characterizes $\mathrm{FO}^{\text{PREF}}$.

Lemma 2. *Let $k \in \mathbb{N}$ and let $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ be two finite or infinite data-words on the same alphabet Σ, such that for every $\tau \in \mathrm{Types}^k_{FO^{\text{PREF}}}$, the number of classes of type τ in $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ are either the same or both at least k. Then $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ agree on all $FO^{\text{PREF}}[\lesssim]$-sentences of quantifier depth at most k.*

Proof. To prove this result, we use the previously introduced Ehrenfeucht-Fraïssé games on data-words. In view of Proposition 15, it is enough to show that given $k \in \mathbb{N}$, if for each k-type τ for $\mathrm{FO}^{\text{PREF}}$ $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ have the same number (up to threshold k) of classes of type τ, then $\mathfrak{w} \equiv_k^{\text{pref-EF}} \widehat{\mathfrak{w}}$.

Under this assumption, we are going to show that the Duplicator can play in such a way as to preserve the invariant described in the remainder of this paragraph. Let us assume that pebbles $p_1,\dots,p_i$ and $\widehat{p}_1,\dots,\widehat{p}_i$ have been placed respectively on $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$. Define the equivalence relation R on $\{p_1,\dots,p_i\}$ (resp. $\widehat{R}$ on $\{\widehat{p}_1,\dots,\widehat{p}_i\}$) where $\alpha R\beta$ iff $\mathfrak{w} \models \alpha \sim \beta$ (resp. $\widehat{\alpha}\widehat{R}\widehat{\beta}$ iff $\widehat{\mathfrak{w}} \models \widehat{\alpha} \sim \widehat{\beta}$). The first part of the invariant is that the bijection sending p_j to $\widehat{p}_j$ maps R to $\widehat{R}$.

For every equivalence class $c = \{p_{j_1}, \ldots, p_{j_l}\}$ of R (and the corresponding class $\{\widehat{p}_{j_1}, \ldots, \widehat{p}_{j_l}\}$ of $\widehat{R}$), let w_c be the class (seen as a word) of $\mathfrak{w}$ on which $p_{j_1}, \ldots, p_{j_l}$ have been placed, together with one constant for each bounding or prefix pebble among $p_{j_1}, \ldots, p_{j_l}$ (there is no need to remember process pebbles) placed at the same position as the corresponding pebble in $\mathfrak{w}$, and let $\widehat{w}_c$ be defined similarly and on the same vocabulary, but for $\widehat{\mathfrak{w}}$. The second part of the invariant is that $w_c \equiv^{\text{pref-EF}}_{k-i} \widehat{w}_c$.

Let us describe how the Duplicator can enforce this invariant during the play, by induction on the number i of rounds. For $i = 0$, there is nothing to show. Assuming the invariant holds after i round, let us consider (without loss of generality) the case where the Spoiler places a pebble p_{i+1} among $\{p^{\text{proc}}_{i+1}, p^{\text{bnd}}_{i+1}, p^{\text{pref}}_{i+1}\}$ in $\mathfrak{w}$. We distinguish between two situations:

- Let us first assume p_{i+1} is placed on a class where some previous pebble has already been placed, and let c be the corresponding equivalence class of R. Our inductive assumption ensures $w_c \equiv^{\text{pref-EF}}_{k-i} \widehat{w}_c$. If $p_{i+1} = p^{\text{proc}}_{i+1}$, the Duplicator simply places $\widehat{p}^{\text{proc}}_{i+1}$ on the process element of $\widehat{\mathfrak{w}}$ corresponding to $\widehat{w}_c$, and the invariant for $i+1$ obviously holds. Assume now $p_{i+1} = p^{\text{bnd}}_{i+1}$ (resp. p^{pref}_{i+1}) has been placed on element α. Remember that $w_c \equiv^{\text{pref-EF}}_{k-i} \widehat{w}_c$: then the Duplicator in the game between $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ answers by placing $\widehat{p}^{\text{bnd}}_{i+1}$ (resp. $\widehat{p}^{\text{pref}}_{i+1}$) on $\widehat{\alpha}$, where $\widehat{\alpha}$ is what the winning strategy for the Duplicator in the game between w_c and $\widehat{w}_c$ answers when the Spoiler makes their move on α. The invariant after round $i+1$ follows easily, since $w'_c \equiv^{\text{pref-EF}}_{k-i-1} \widehat{w}'_c$, where w'_c (resp. $\widehat{w}'_c$) is the extension of the word w_c with a new constant, interpreted as α (resp. $\widehat{\alpha}$).
- Suppose now that p_{i+1} is placed on a class w of $\mathfrak{w}$ on which no previous pebble has been placed. By assumption about $\mathfrak{w}$ and $\widehat{\mathfrak{w}}$ sharing the same number of classes of every type up to threshold k, we can find a class $\widehat{w}$ of $\widehat{\mathfrak{w}}$ which has the same k-type for FO^{PREF} as w (i.e. such that $w \equiv^{\text{pref-EF}}_{k} \widehat{w}$), and on which no previous pebble has been placed either. If $p_{i+1} = p^{\text{proc}}_{i+1}$, then the Duplicator places $\widehat{p}^{\text{proc}}_{i+1}$ on the process element of $\widehat{w}$, and $w \equiv^{\text{pref-EF}}_{k} \widehat{w}$ in particular entails $w \equiv^{\text{pref-EF}}_{k-i-1} \widehat{w}$. On the other hand, if $p_{i+1} = p^{\text{bnd}}_{i+1}$ has been placed on element α of w, then $w \equiv^{\text{pref-EF}}_{k} \widehat{w}$ means that the Duplicator can place $\widehat{p}^{\text{bnd}}_{i+1}$ on some element $\widehat{\alpha}$ of $\widehat{w}$ such that $w' \equiv^{\text{FO}^{\text{PREF}}}_{k-1} \widehat{w}'$ (and, a fortiori, $w' \equiv^{\text{FO}^{\text{PREF}}}_{k-i-1} \widehat{w}'$) where w' (resp. $\widehat{w}'$) is the same as w (resp. $\widehat{w}$) where a new constant is interpreted as α (resp. $\widehat{\alpha}$). The invariant after round $i+1$ follows.

It is straightforward to notice that enforcing this invariant after the k rounds of the game guarantees a win for the Duplicator. □

This easy technical lemma will prove most useful in the following:

Lemma 16. *Let u and u' be two words on Σ, and $n, n' \in \mathbb{N}$ be such that such that $\langle u[0 \ldots n] \rangle^k_{FO^{\text{PREF}}} = \langle u'[0 \ldots n'] \rangle^k_{FO^{\text{PREF}}}$.*

In the k-round prefix Ehrenfeucht-Fraïssé game between u and u', if the Spoiler's first move is in $u[0 \ldots n]$ or in $u'[0 \ldots n']$, then the Duplicator wins the game.

Proof. By assumption, the Spoiler plays their first move (which must be a bounding move) in $u[0 \dots n]$ or $u'[0 \dots n']$. The Duplicator responds as they would in the k-round prefix Ehrenfeucht-Fraïssé game between $u[0 \dots n]$ and $u'[0 \dots n']$.

The bounding pebbles are thus in the prefixes of u and u' for which the Duplicator has a winning strategy, and have been placed according to this strategy. By definition of the game, no pebble will ever be placed outside of these prefixes. The Duplicator can thus follow the same strategy to win the game between u and u' where the Spoiler has played their first move in $u[0 \dots n]$ or $u'[0 \dots n']$. □

When one word is the prefix of the other, we immediately get the following consequence:

Corollary 17. *Let u and v be two words such that u is a prefix of v, and let us consider the prefix Ehrenfeucht-Fraïssé game between u and v.*

If the Spoiler's first move is in u, then the Duplicator wins the game, no matter the number of rounds.

Lemma 3. *Let u, v be words such that $\langle u \rangle^k_{FO^{\text{PREF}}} = \langle u \cdot v \rangle^k_{FO^{\text{PREF}}} = \tau$. Then for every prefix w of v, $\langle u \cdot w \rangle^k_{FO^{\text{PREF}}} = \tau$.*

Proof. In order to prove this result, we fix such a w and show that the Duplicator has a winning strategy in the k-round prefix Ehrenfeucht-Fraïssé game between u and $u \cdot w$.

If the Spoiler first plays in u, then Lemma 16 guarantees the Duplicator wins the game. Thus, we can assume that the Spoiler's first move is a bounding move in $u \cdot w$. By Lemma 16, the Duplicator wins the k-round game between $u \cdot w$ and $u \cdot v$ where the Spoiler starts by playing in $u \cdot w$. To win the game between u and $u \cdot w$ when Spoiler starts in $u \cdot w$, the Duplicator combines their strategy in the game between u and $u \cdot v$ and their strategy in the game between $u \cdot v$ and $u \cdot w$ where Spoiler starts in $u \cdot w$. □

Lemma 4. *Let u be an infinite word with stationary type $\tau \in \text{Types}^k_{FO^{\text{PREF}}}$. Then $\langle u \rangle^k_{FO^{\text{PREF}}} = \tau$.*

Proof. Let n be such that $u[0 \dots n]$ has type τ. We show that the Duplicator has a winning strategy in the k-round prefix Ehrenfeucht-Fraïssé game between u and $u[0 \dots n]$.

Due to Corollary 17, the Spoiler cannot win by playing their first move in $u[0 \dots n]$. Let us therefore assume that the Spoiler plays first on position m of u. By assumption, there exists a position $m' > m$ such that $u[0 \dots m']$ has type τ, i.e. $u[0 \dots n] \equiv^{\text{FO}^{\text{PREF}}}_k u[0 \dots m']$. Combining the winning strategies for the Duplicator in the game between $u[0 \dots n]$ and $u[0 \dots m']$, and in the game between $u[0 \dots m']$ and $u[0 \dots m]$ where the Spoiler plays first in $u[0 \dots m]$ (following from Corollary 17), the Duplicator can win the k-round prefix Ehrenfeucht-Fraïssé game between $u[0 \dots n]$ and $u[0 \dots m]$ when the Spoiler plays first in $u[0 \dots m]$.

All in all, $u \equiv^{\text{FO}}_k u[0 \dots n]$, i.e. u has type τ. □

Lemma 6. *Let $k \geq 1$, let τ and τ' be such that $\tau \rightharpoonup_k \tau'$ and let w be a finite word such that $\langle w \rangle^k_{FO^{\text{PREF}}} = \tau$. There exists some finite word w' such that $\langle ww' \rangle^k_{FO^{\text{PREF}}} = \tau'$.*

Proof. By definition, there exists some finite words u, v such that $\langle u \rangle^k_{\text{FO}^{\text{PREF}}} = \tau$ and $\langle uv \rangle^k_{\text{FO}^{\text{PREF}}} = \tau'$.

We can assume that $u \neq \varepsilon$, for otherwise $w = u$ and there is nothing to show.

The Duplicator has a winning strategy in the k-round prefix Ehrenfeucht-Fraïssé game between u and w. Let n be the position in u of their response when the Spoiler plays a bounding move on the last position of w (which we assumed exists). Let us call $\mathcal{S}$ the rest of Duplicator's strategy when these first bounding moves have been played: $\mathcal{S}$ thus allows the Duplicator to win for $k-1$ more prefix moves made in w and $u[1 \dots n]$.

With this strategy fixed, let us show that

$$w' := u[n+1 \dots] \cdot v$$

is a good candidate for our purpose, i.e. that $\langle w \cdot w' \rangle^k_{\text{FO}^{\text{PREF}}} = \tau'$ for such a choice of w'. To that end, we describe a winning strategy for the Duplicator in the k-round prefix Ehrenfeucht-Fraïssé game between $u \cdot v$ and $w \cdot u[n+1 \dots] \cdot v$.

If the Spoiler makes their bounding move in the prefix u of $u \cdot v$ or in the prefix w of $w \cdot u[n+1 \dots] \cdot v$, Lemma 16 ensures the Duplicator wins the game.

Let us now assume that the Spoiler plays their bounding move in the suffix $u[n+1 \dots] \cdot v$ of $w \cdot u[n+1 \dots] \cdot v$ (resp. in the suffix v of uv). In this case, the Duplicator answers by playing on the corresponding element of the suffix $u[n+1 \dots] \cdot v$ of uv (resp. the corresponding element of the suffix v of $w \cdot u[n+1 \dots] \cdot v$). Following this, the Duplicator's strategy is

- to play tit-for-tat whenever the Spoiler plays in either of the suffixes $u[n+1 \dots] \cdot v$, and
- to follow $\mathcal{S}$ whenever the Spoiler plays in the prefix w of $w \cdot u[n+1 \dots] \cdot v$ or in the prefix $u[1 \dots n]$ of uv.

By definition of $\mathcal{S}$, this strategy allows the Duplicator to win for $k-1$ additional rounds.

We have covered every cases, which all lead to the Duplicator winning the k-round prefix game between $u \cdot v$ and $w \cdot u[n+1 \dots] \cdot v$, thus showing that $\langle ww' \rangle^k_{\text{FO}^{\text{PREF}}} = \tau'$ for $w' := u[n+1 \dots] \cdot v$. □

Lemma 7. *The binary relation $\rightharpoonup_k$ is an order on $\text{Types}^k_{FO^{\text{PREF}}}$, with minimum $\langle \varepsilon \rangle^k_{FO^{\text{PREF}}}$.*

Proof. Reflexivity is straightforward, and so is the fact that for every $\tau \in \text{Types}^k_{\text{FO}^{\text{PREF}}}$, $\langle \varepsilon \rangle^k_{\text{FO}^{\text{PREF}}} \rightharpoonup_k \tau$.

To check that $\rightharpoonup_k$ is transitive, let us consider three types such that $\tau \rightharpoonup_k \tau' \rightharpoonup_k \tau''$. By definition, there exists two finite words u and u' such that $\langle u \rangle^k_{\text{FO}^{\text{PREF}}} = \tau$ and $\langle uu' \rangle^k_{\text{FO}^{\text{PREF}}} = \tau'$. Lemma 6 entails the existence of some finite word u'' such that $\langle uu'u'' \rangle^k_{\text{FO}^{\text{PREF}}} = \tau''$, which concludes the proof.

It only remains to check the antisymmetry of $\rightharpoonup_k$. Let $\tau, \tau' \in \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$ be such that $\tau \rightharpoonup_k \tau'$ and $\tau' \rightharpoonup_k \tau$. By Lemma 6, there must exist three finite words u, u', u'' such that $\langle u \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \langle uu'u'' \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau$ and $\langle uu' \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau'$. From there, Lemma 3 derives the equality of τ and τ'. □

B Proofs for Section 3 (Synthesis and token games)

Lemma 8. *Let φ be a $FO^{\mathrm{PREF}}[\lesssim]$-sentence, and $n_S, n_E \in \mathbb{N}$. System has an (n_S, n_E)-winning strategy for φ in the symmetric game if and only if System has an (n_S, n_E)-winning strategy for φ in the standard game.*

Proof. It is straightforward to notice that if System has a winning strategy for the standard game, then the same strategy is winning for the symmetric game.

For the reverse direction, we need an additional observation regarding the logic $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$. In general, given a data word $\mathfrak{w}$, in any reordering of the positions of $\mathfrak{w}$ such that the class of each process is left unchanged, then the satisfaction value of any formula in $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ is the same as for the original word. Indeed, since the satisfaction of $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ only depends on the collection of types of processes and not their relative ordering, a $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formula cannot distinguish between the two data words. In other terms, we can freely swap any two consecutive positions in a data words as long as they do not share the same process, while keeping the same satisfaction value for any $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formula. Formally, we say that two data words $\mathfrak{w}_1$ and $\mathfrak{w}_2$ are *equivalent*, denoted by $\mathfrak{w}_1 \equiv \mathfrak{w}_2$, if $\mathfrak{w}_2$ is a reordering of $\mathfrak{w}_1$ and for all processes p, the class of p in $\mathfrak{w}_1$ and $\mathfrak{w}_2$ are the same. As explained above, if two data words are equivalent, then they satisfy exactly the same $\mathrm{FO}^{\mathrm{PREF}}[\lesssim]$ formulas; formally, one shows that the natural strategy for the Duplicator in the prefix Ehrenfeucht-Fraïssé game at any depth (which consists of responding to a Spoiler's move by playing on the same position of the same class) is winning.

Now, let us assume that System has a (n_S, n_E)-winning strategy $\mathcal{S}$ in the symmetric game. We build a strategy $\mathcal{S}'$ in the standard game using additional memory $m(\mathfrak{w}) = (\mathfrak{w}_{\mathrm{S}}, \mathfrak{w}_{\mathrm{E}}, \mathfrak{w}_{\equiv}) \in \left(\mathrm{DW}^{\mathbb{P}}_{\Sigma}\right)^3$ that depends on the history $\mathfrak{w}$. Intuitively, $\mathcal{S}'$ tries to mimic $\mathcal{S}$, and to this end $\mathfrak{w}_{\mathrm{S}}$ is a word played by $\mathcal{S}$ that $\mathcal{S}'$ tries to play bit by bit. However Environment actions can happen in between System actions. Those are then stored in $\mathfrak{w}_{\mathrm{E}}$, a queue of actions made by Environment to be acknowledged later by $\mathcal{S}'$. That is, when $\mathcal{S}'$ finishes playing the word that was stored in $\mathfrak{w}_{\mathrm{S}}$ and needs to consult $\mathcal{S}$ to see what is the next word to be played, then all of $\mathfrak{w}_{\mathrm{E}}$ is added to the history as if they had just happened at this point. Finally, $\mathfrak{w}_{\equiv}$ records a history in the symmetric game such that $\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}$ is equivalent to the current history $\mathfrak{w}$ in the standard game and such that $\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{S}}$ is a $\mathcal{S}$-compatible history.

First let us define $\mathcal{S}'$. On a given history $\mathfrak{w} \in \mathrm{DW}^{\mathbb{P}}_{\Sigma}$ and with memory $m(\mathfrak{w}) = (\mathfrak{w}_{\mathrm{S}}, \mathfrak{w}_{\mathrm{E}}, \mathfrak{w}_{\equiv})$, $\mathcal{S}'$ is defined as follows. If $\mathfrak{w}_{\mathrm{S}} = (a_0, p_0) \dots$ is not empty, then $\mathcal{S}'(\mathfrak{w}) = (a_0, p_0)$. Otherwise $\mathfrak{w}_{\mathrm{S}} = \varepsilon$, and with $\mathcal{S}(\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}) = (a_0, p_0) \dots$ we let $\mathcal{S}'(\mathfrak{w}) = (a_0, p_0)$. If $\mathcal{S}(\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}) = \varepsilon$ instead, then $\mathcal{S}'(\mathfrak{w}) = \varepsilon$ as well. Let

us now show how the memory is updated after each action. The initial memory is $m(\varepsilon) = (\varepsilon, \varepsilon, \varepsilon)$. Suppose now that the current history is $\mathfrak{w}$ and the current memory is $m(\mathfrak{w}) = (\mathfrak{w}_{\mathrm{S}}, \mathfrak{w}_{\mathrm{E}}, \mathfrak{w}_{\equiv})$. On an Environment action (a, p), we update the memory to $m(\mathfrak{w} \cdot (a, p)) = (\mathfrak{w}_{\mathrm{S}}, \mathfrak{w}_{\mathrm{E}} \cdot (a, p), \mathfrak{w}_{\equiv})$. On a System action (a, p), if $\mathfrak{w}_{\mathrm{S}} = (a, p) \cdot \mathfrak{w}_{\mathrm{S}}^-$ then $m(\mathfrak{w} \cdot (a, p)) = (\mathfrak{w}_{\mathrm{S}}^-, \mathfrak{w}_{\mathrm{E}}, \mathfrak{w}_{\equiv} \cdot (a, p))$. If $\mathfrak{w}_{\mathrm{S}} = \varepsilon$, and $\mathcal{S}(\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}) = (a, p) \cdot \mathfrak{w}^{\mathcal{S}}$, then we let $m(\mathfrak{w} \cdot (a, p)) = (\mathfrak{w}^{\mathcal{S}}, \varepsilon, \mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}} \cdot (a, p))$. Otherwise $m(\mathfrak{w} \cdot (a, p))$ is undefined.

We need a few properties regarding m before showing that $\mathcal{S}'$ is winning. Let $\mathfrak{w}$ be a history in the standard game and let $m(\mathfrak{w}) = (\mathfrak{w}_{\mathrm{S}}, \mathfrak{w}_{\mathrm{E}}, \mathfrak{w}_{\equiv})$. First we show by induction that $\mathfrak{w} \equiv \mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}$. The initial case is trivial. Suppose now that $\mathfrak{w} \equiv \mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}$, and let (a, p) be the next action with $m(\mathfrak{w} \cdot (a, p)) = (\mathfrak{w}_{\mathrm{S}}^2, \mathfrak{w}_{\mathrm{E}}^2, \mathfrak{w}_{\equiv}^2)$. If a is an Environment action or if $\mathfrak{w}_{\mathrm{S}} = \varepsilon$, then we instantly get that $\mathfrak{w} \cdot (a, p) \equiv \mathfrak{w}_{\equiv}^2 \cdot \mathfrak{w}_{\mathrm{E}}^2$ by the definition of m and the induction hypothesis. If a is a System action and $\mathfrak{w}_{\mathrm{S}} \neq \varepsilon$, then notice that since $\mathfrak{w}_{\mathrm{E}}$ entirely consists of Environment actions, none of those involve process p which is a System process. Thus $\mathfrak{w}_{\equiv} \cdot (a, p) \cdot \mathfrak{w}_{\mathrm{E}} \equiv \mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}} \cdot (a, p)$, and the rest follows. The facts that $\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{S}}$ is always $\mathcal{S}$-compatible and that $m(\mathfrak{w})$ is always defined on any $\mathcal{S}'$-compatible history are also straightforward to prove inductively.

Finally, it remains to be shown that $\mathcal{S}'$ is winning in the standard game. Let $\mathfrak{w}$ be a $\mathcal{S}'$-compatible $\mathcal{S}'$-fair history in the standard game.

If $\mathfrak{w}$ is finite, then necessarily $\mathcal{S}'(\mathfrak{w}) = \varepsilon$. The only possible way for this to occur is that, with $m(\mathfrak{w}) = (\mathfrak{w}_{\mathrm{S}}, \mathfrak{w}_{\mathrm{E}}, \mathfrak{w}_{\equiv})$, we have that $\mathfrak{w}_{\mathrm{S}} = \varepsilon$ and $\mathcal{S}(\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}) = \varepsilon$. In that case, since $\mathcal{S}$ is winning then $\mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}$ must satisfy φ. Since $\mathfrak{w} \equiv \mathfrak{w}_{\equiv} \cdot \mathfrak{w}_{\mathrm{E}}$, then so does $\mathfrak{w}$.

If $\mathfrak{w}$ is infinite, then either $\mathcal{S}'$ only outputs ε from some point on, in which case the previous argument also shows that $\mathfrak{w}$ satisfies φ, or $\mathcal{S}'$ outputs some action infinitely often. By the fairness constraint, it means that System actions have been made infinitely often in $\mathfrak{w}$. Let $m(\mathfrak{w}[0 \dots i]) = (\mathfrak{w}_{\mathrm{S}}^i, \mathfrak{w}_{\mathrm{E}}^i, \mathfrak{w}_{\equiv}^i)$ denote the memory after the i-th first actions in $\mathfrak{w}$. Let $\mathfrak{w}_{\equiv}$ be the limit of the sequence of $\mathfrak{w}_{\equiv}^i \cdot \mathfrak{w}_{\mathrm{E}}^i$, which is well-defined by definition of the memory. Since for all i the "buffer" $\mathfrak{w}_{\mathrm{S}}^i$ only contains a finite amount of System actions to be played by $\mathcal{S}'$, it follows that $\mathfrak{w}_{\mathrm{S}}^i = \varepsilon$ for infinitely many i. At those points, we have that $\mathfrak{w}[0 \dots i] \equiv \mathfrak{w}_{\equiv}^i \cdot \mathfrak{w}_{\mathrm{E}}^i$ and that $\mathfrak{w}_{\equiv}^i \cdot \mathfrak{w}_{\mathrm{E}}^i$ is $\mathcal{S}$-compatible, as $\mathfrak{w}_{\mathrm{S}}^i$ is empty and adding Environment actions to $\mathfrak{w}_{\equiv}^i$ does not change its $\mathcal{S}$-compatibility. Thus, $\mathfrak{w}_{\equiv}$ is an infinite $\mathcal{S}$-compatible data word, and for infinitely many i we have that $\mathfrak{w}[0 \dots i] \equiv \mathfrak{w}_{\equiv}[0 \dots i]$. From this, we deduce that $\mathfrak{w} \equiv \mathfrak{w}_{\equiv}$. Indeed if that was not the case then either $\mathfrak{w}$ is not a reordering of $\mathfrak{w}_{\equiv}$, which means that there is some finite position i in $\mathfrak{w}$ or $\mathfrak{w}_{\equiv}$ that does not have a counterpart in the other data word, contradicting the equivalence between the next two prefixes. Or, there is a class p in $\mathfrak{w}$ that differs from $\mathfrak{w}_{\equiv}$, which again means that they differ on some finite position in that class that corresponds to some position i in $\mathfrak{w}$ and $\mathfrak{w}_{\equiv}$, again contradicting the infinitely many prefixes that are equivalent. To sum up, $\mathfrak{w}$ is equivalent to $\mathfrak{w}_{\equiv}$, and $\mathfrak{w}_{\equiv}$ satisfies φ because $\mathcal{S}$ is winning in the symmetric game, therefore $\mathfrak{w}$ satisfies φ and thus $\mathcal{S}'$ is winning. □

Lemma 9. *A pair (n_S, n_E) is winning for System in the token game for φ if and only if System has a (n_S, n_E)-winning strategy for φ in the standard synthesis game.*

Proof. Winning in the token game on types is the same as winning in the token game on words: a strategy in the word arena moving a token from word w to $w \cdot w'$ is mimicked in the type arena by the strategy moving the same token from $\langle w \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$ to $\langle w \cdot w' \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}}$ (by definition of $\rightharpoonup_k$, this is a valid move). Conversely, if a strategy in the type arena wants to move a token from τ to τ', then by Lemma 6 we know that whatever word w such that $\langle w \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau$ the token is in at the time, there is an extension w' such that $\langle ww' \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}} = \tau'$ that can be played by the strategy in the arena of words. Furthermore, the acceptance conditions are equivalent: by Lemma 5 the type of some token t in the limit configuration of a play is exactly the type of the data word of the corresponding process. Thus, the acceptance condition of the token game in the type arena is equivalent to winning in the word arena. Finally, winning in the token game on words is equivalent to winning in the symmetric game, which itself is equivalent to winning in the standard synthesis game, thus System wins the token game on types if and only if System wins the standard synthesis game. □

C Proofs for Section 4 (Double cutoff for solving the synthesis problem)

Lemma 12. *For every $k, n_E \in \mathbb{N}$, there exists some $f_S(k, n_E)$ such that for any $n_S \geq f_S(k, n_E)$, if System has an $(n_S + 1, n_E)$-winning strategy in a token game of depth k, then System has an (n_S, n_E)-winning strategy in that game.*

Idea of the proof and definitions. Let us start by giving a value to $f_S(k, n_E)$. For that, we define the following function F by induction on n:

$$\begin{cases} F(-1) := k \\ F(n) := |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}|^{n_E+2} \cdot \big(F(n-1)+1\big)^2 + k \quad \text{for } n \geq 0 \end{cases} \tag{4}$$

Let us note already that F is non-decreasing, and that $F(n) \geq k$ for all $n > 0$. Let $h(\tau)$ denote the height of type τ in $(\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}, \rightharpoonup_k)$, as defined in Section 4.1. In a given configuration, we say that a type τ is *large* if it has more than k System tokens and *huge* if it has more than $F(h(\tau))$ System tokens. We then set

$$f_S(k, n_E) := F(h(\langle \varepsilon \rangle^k_{\mathrm{FO}^{\mathrm{PREF}}})) \tag{5}$$

The relevance of this choice of $f_S(k, n_E)$ will become apparent throughout the proof.

In order to prove Lemma 12, we explain how an (n_S+1, n_E)-winning strategy $\mathcal{S}_+$ for System in a token game of depth k can be converted to an (n_S, n_E)-winning strategy $\mathcal{S}$ for the same game.

Let $\mathbb{T}_{\mathrm{S}}^{n_S+1}$ (resp. $\mathbb{T}_{\mathrm{S}}^{n_S}$) be the set of System's tokens in the original (n_S+1, n_E) (resp. new (n_S, n_E)) game, and $\mathbb{T}_{\mathrm{E}}$ be the set of Environment's tokens in both games. Note that $|\mathbb{T}_{\mathrm{S}}^{n_S+1}| = n_S + 1$, $|\mathbb{T}_{\mathrm{S}}^{n_S}| = n_S$ and $|\mathbb{T}_{\mathrm{E}}| = n_E$. Let $\mathfrak{G}_+ = (\mathbb{T}_{\mathrm{S}}^{n_S+1} \uplus \mathbb{T}_{\mathrm{E}}, \mathcal{A}_k, \alpha_\varphi)$ and $\mathfrak{G} = (\mathbb{T}_{\mathrm{S}}^{n_S} \uplus \mathbb{T}_{\mathrm{E}}, \mathcal{A}_k, \alpha_\varphi)$ denote the corresponding games.

In order to adapt $\mathcal{S}_+$ to a setting where System has one fewer token, we will need to track a *ghost* token γ_π; i.e. a System token which appears in $\mathcal{S}_+$ but not in $\mathcal{S}$. The key idea is to make sure that at any point during the unfolding of $\mathcal{S}_+$, γ_π is always in a large type, so that the winning conditions of the game cannot notice its absence. Contrary to the previous section, there is no hope for one token to be the ghost throughout the whole play: depending on the way $\mathcal{S}_+$ unfolds, we may have to change candidates for being the ghost.

We will need additional information to decide when to switch to another token for our choice of ghost. First, we not only need to track the token that has disappeared when going from $\mathcal{S}_+$ to $\mathcal{S}$, but is it also crucial that we remember which token in $\mathfrak{G}$ plays the role of which token in $\mathfrak{G}_+$. We store this information in a one-to-one mapping $\sigma_\pi : \mathbb{T}_{\mathrm{S}}^{n_S} \to \mathbb{T}_{\mathrm{S}}^{n_S+1}$, where γ_π is precisely the token from $\mathbb{T}_{\mathrm{S}}^{n_S+1}$ that has no preimage. Second, the strategy $\mathcal{S}$ we are defining mimics the winning strategy $\mathcal{S}_+$ in $\mathfrak{G}_+$, and we need a way to remember how both connect. This information is stored via π_+, such that each play π in $\mathfrak{G}$ compatible with $\mathcal{S}$ has a corresponding play π_+ in $\mathfrak{G}_+$ that is compatible with $\mathcal{S}_+$. Note that π_+ can be longer than π, as we will sometimes need to move faster in $\mathfrak{G}_+$ than in $\mathfrak{G}$, for technical reasons.

It will be convenient to write τ_π^γ for $\pi_+(\gamma_\pi)$, i.e. the type of the ghost after the play π_+ in $\mathfrak{G}_+$.

We work by induction on the length of the play π, and define as we go along the strategy $\mathcal{S}$ on those π which end with an Environment move. If

$$\pi = C_0 \xrightarrow{m_{\mathrm{E}}^0} C_1 \xrightarrow{m_{\mathrm{S}}^1} C_2 \xrightarrow{m_{\mathrm{E}}^2} \cdots \xrightarrow{m_?^{n-1}} C_n \tag{6}$$

(where the last move is an Environment move if and only if n is odd), we denote $\pi_{|i}$ the prefix $\pi = C_0 \xrightarrow{m_{\mathrm{E}}^0} C_1 \xrightarrow{m_{\mathrm{S}}^1} \cdots \xrightarrow{m_?^{i-1}} C_i$ of π.

Let us consider a play π as in (6) compatible with (what has been defined so far of) $\mathcal{S}$, which satisfies the invariants defined below. Then

- if $n = 2l$ is even, we show that for any valid Environment move m_{E}^{2l} compatible with π and with $\pi' = \pi \xrightarrow{m_{\mathrm{E}}^{2l}} C_{n+1}$, we can define π'_+ and $\sigma_{\pi'}$ such that the invariants also hold for π'
- if $n = 2l + 1$ is odd, we define the next System move $\mathcal{S}(\pi)$ of our new strategy, and with $\pi' = \pi \xrightarrow{\mathcal{S}(\pi)} C_{n+1}$ we also define π'_+ and $\sigma_{\pi'}$ such that the invariants also hold for π'.

To conclude the proof of Lemma 12 we show that the new strategy $\mathcal{S}$ is indeed winning in $\mathfrak{G}$.

One of the invariant states that the type of the ghost should go down as much as possible when choosing the next System move. To that end, for any

play π ending in an Environment move, we define a *τ-descending sequence* for π as a pair $(\xi, \widehat{m})$ such that ξ is a permutation of $\mathbb{T}_{\mathrm{S}}^{n_S+1}$, $\widehat{m}$ is a finite sequence of moves in $\mathfrak{G}_+$ starting from π_+ that starts and end with a System move, that satisfies the following conditions:

$$\begin{cases} \pi'_+ = \pi_+ \xrightarrow{\widehat{m}} C'_+ \text{ is compatible with } \mathcal{S}_+ \\ \forall e \in \mathbb{T}_{\mathrm{E}},\ \pi_+(e) = \pi'_+(e) \\ \forall s \in \mathbb{T}_{\mathrm{S}}^{n_S+1},\ \pi_+(s) \rightharpoonup_k \pi'_+(\xi(s)) \\ \pi'_+(\xi(\gamma_\pi)) = \tau \\ |\{s \in \mathbb{T}_{\mathrm{S}}^{n_S+1} : \pi'_+(s) = \tau\}| \geq F(h(\tau)) + 1 \end{cases} \tag{7}$$

A System move m_{S} is said to be *max-descending* if it results in a play $\pi' = \pi \xrightarrow{m_{\mathrm{S}}} C'$ such that there are no τ-descending sequence for π for any type $\tau \neq \tau^{\gamma}_{\pi'}$ such that $\tau^{\gamma}_{\pi'} \rightharpoonup_k \tau$.

Statement of the invariants. We are now ready to list the various invariants, which are indexed with the play to which they refer. We give below an intuition about their meaning and usefulness.

- $(\mathtt{I}_+[\pi])$: π_+ is compatible with $\mathcal{S}_+$ and π_+ has the same length parity as π furthermore, for every prefix π' of π, π'_+ is a prefix of π_+
- $(\mathtt{I}_{\mathrm{E}}[\pi])$: $\forall e \in \mathbb{T}_{\mathrm{E}},\ \pi(e) = \pi_+(e)$
- $(\mathtt{I}_{\mathrm{S}}[\pi])$: $\forall s \in \mathbb{T}_{\mathrm{S}}^{n_S},\ \pi(s) = \pi_+(\sigma_\pi(s))$
- $(\mathtt{I}_{\tau^\gamma}[\pi])$: If π' is a prefix of π, then $\tau^{\gamma}_{\pi'} \rightharpoonup_k \tau^{\gamma}_{\pi}$
- $(\mathtt{I}_{\mathrm{F}}[\pi])$: If there is no strict prefix π' of π such that $\tau^{\gamma}_{\pi'} = \tau^{\gamma}_{\pi}$, then τ^{γ}_{π} is huge in π; otherwise τ^{γ}_{π} is still large in π.
- $(\mathtt{I}_{\max}[\pi])$: All System moves are max-descending.

Let us briefly give some intuition about these invariants. First of all, $(\mathtt{I}_+[\pi])$, $(\mathtt{I}_{\mathrm{E}}[\pi])$ and $(\mathtt{I}_{\mathrm{S}}[\pi])$ ensure that π mimics π_+. In particular, then all tokens (if we identify $\sigma_\pi(s)$ with s) except the ghost have the same type after π and π_+. The invariant $(\mathtt{I}_{\tau^\gamma}[\pi])$ ensures that, although the ghost can change during the play, the type containing it can only go down in $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$. Another crucial requirement, if we are to show that the limit configuration of π in $\mathfrak{G}$ is winning if and only if the limit configuration of $\nu(\pi)$ is winning in $\mathfrak{G}_+$, is that the ghost (which is an extra token in $\mathfrak{G}_+$) cannot make a difference for the acceptance condition of the game. This is ensured by $(\mathtt{I}_{\mathrm{F}}[\pi])$, which has as consequence the fact that the ghost is always in a large type. The first part of $(\mathtt{I}_{\mathrm{F}}[\pi])$ is a technical requirement needed in the proofs, and ensures that whenever the type of the ghost moves down in $\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$, its new type is huge. As for $(\mathtt{I}_{\max}[\pi])$, it ensures that the System brings the ghost down as much as possible: if a play by System would have permitted the ghost to be lower in the tree $(\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}, \rightharpoonup_k)$, then the System would have played it. This is needed to guarantee $(\mathtt{I}_{\mathrm{F}}[\pi])$.

Proof by induction. We are now ready to move to the induction proper.

Empty play. In order to start the induction, let us consider the empty play π_0 in $\mathfrak{G}$. We set σ_{π_0} to be any one-to-one mapping $\mathbb{T}_{\mathrm{S}}^{n_S} \to \mathbb{T}_{\mathrm{S}}^{n_S+1}$, and $(\pi_0)_+$ to be the empty play in $\mathfrak{G}_+$. One easily checks that all the invariants are satisfied for π_0; in particular, $(\mathtt{I}_{\mathrm{F}}[\pi_0])$ follows from the choice of $f_S(k, n_E)$ in (5) and from the assumption that $n_S \geq f_S(k, n_E)$.

When Environment makes a move. Let π be a play ending with a System move, and consider the next move m_{E}^{2l} by Environment resulting in play π'. We set $\sigma_{\pi'} := \sigma_\pi$ and $\pi'_+ := \pi_+ \xrightarrow{m_{\mathrm{E}}^{2l}} C_+$. The latter is well defined according to $(\mathtt{I}_{\mathrm{E}}[\pi])$. All the invariants are trivially satisfied with these definitions.

When System makes a move. Let us now consider a play π ending with an Environment move. By induction hypothesis, we can assume that all the $(\pi_{|2i})_+$ and $\sigma_{\pi_{|2i}}$, for $i \leq l$, are defined and satisfy all the invariants. We will now explain how to construct the next System move $\mathcal{S}(\pi)$ resulting in play $\pi' = \pi \xrightarrow{\mathcal{S}(\pi)} C_{n+1}$, as well as π'_+ and $\sigma_{\pi'}$ so that these invariants still hold for π'. We distinguish two cases: when the ghost token moves down in the tree $(\mathrm{Types}^k_{\mathrm{F}\,\mathrm{O}^{\mathrm{PREF}}}, \rightharpoonup_k)$ and when it stays in the same type.

When the ghost goes down in the tree. Let τ be a maximal type such that $\tau \neq \tau_\pi^\gamma$, $\tau_\pi^\gamma \rightharpoonup_k \tau$, and there is a τ-descending sequence exists in π with associated ξ and $\widehat{m}$, assuming such a type exists. Let $\pi'_+ = \pi_+ \xrightarrow{\widehat{m}} C'_+$. In this case, we define the move $\mathcal{S}(\pi)$ as follows:

$$\forall s \in \mathbb{T}_{\mathrm{S}}^{n_S},\ \mathcal{S}(\pi)(s) := \pi'_+(\xi(\sigma_\pi(s)))\,. \tag{8}$$

This is well defined, insomuch as for every $s \in \mathbb{T}_{\mathrm{S}}^{n_S}$ we have, according to $(\mathtt{I}_{\mathrm{S}}[\pi])$ and (7),

$$\pi(s) = \pi_+(\sigma_\pi(s)) \rightharpoonup_k \pi'_+(\xi(\sigma_\pi(s)))\,.$$

Let us write $\bar{\pi}$ for the play π followed by our newly defined response:

$$\bar{\pi} := \pi \xrightarrow{\mathcal{S}(\pi)} C'\,.$$

Let us now define $\bar{\pi}_+$ and $\sigma_{\bar{\pi}}$ as follows:

$$\bar{\pi}_+ := \pi'_+ \tag{9}$$

and

$$\forall s \in \mathbb{T}_{\mathrm{S}}^{n_S},\ \sigma_{\bar{\pi}}(s) := \xi(\sigma_\pi(s))\,. \tag{10}$$

Notice that these definitions entail

$$\gamma_{\bar{\pi}} = \xi(\gamma_\pi) \quad \text{and} \quad \tau_{\bar{\pi}}^\gamma = \tau\,. \tag{11}$$

Let us show that with these definitions, the invariants hold on $\bar{\pi}$:

Proof of $(\mathtt{I}_+[\bar{\pi}])$**:** Because $\widehat{m}$ satisfies (7), $\bar{\pi}_+ = \pi_+ \xrightarrow{\widehat{m}} C'_+$ is compatible with $\mathcal{S}_+$. It ends with a System move, as does $\bar{\pi}$. The second part of the invariant comes trivially as $\bar{\pi}_+$ is built by extending π_+.

Proof of $(\mathtt{I_E}[\bar{\pi}])$**:** We need to show that for every $e \in \mathbb{T}_{\mathrm{E}}$, $\bar{\pi}(e) = \bar{\pi}_+(e)$. Because $\widehat{m}$ satisifes (7), we have $\pi_+(e) = \pi'_+(e)$, from which $(\mathtt{I}_{\mathrm{E}}[\bar{\pi}])$ follows, since for every $e \in \mathbb{T}_{\mathrm{E}}$, $\pi_+(e) = \bar{\pi}(e)$ (as a System move doesn't change the type of an Environment token) and $\bar{\pi}_+ = \pi'_+$ (9).

Proof of $(\mathtt{I_S}[\bar{\pi}])$**:** Let us show that for every $s \in \mathbb{T}_{\mathrm{S}}^{n_S}$, $\bar{\pi}(s) = \bar{\pi}_+(\sigma_{\bar{\pi}}(s))$. By definition, $\bar{\pi}(s) = \pi'_+(\xi(\sigma_\pi(s)))$ which, by definition of $\bar{\pi}_+$ (9) and $\sigma_{\bar{\pi}}$ (10), amounts to $\bar{\pi}_+(\sigma_{\bar{\pi}}(s))$.

Proof of $(\mathtt{I}_{\tau^\gamma}[\bar{\pi}])$**:** This follows directly from $\tau_\pi^\gamma \rightharpoonup_k \tau = \tau_{\bar{\pi}}^\gamma$ and $(\mathtt{I}_{\tau^\gamma}[\pi])$.

Proof of $(\mathtt{I_F}[\bar{\pi}])$**:** Let us show that $\tau_{\bar{\pi}}^\gamma$ is huge in $\bar{\pi}$. By (7) and (11), we have $|\{s \in \mathbb{T}_{\mathrm{S}}^{n_S+1} : \bar{\pi}_+(s) = \tau_{\bar{\pi}}^\gamma\}| \geq F(h(\tau_{\bar{\pi}}^\gamma)) + 1$. It follows from $(\mathtt{I}_{\mathrm{S}}[\bar{\pi}])$ that $|\{s \in \mathbb{T}_{\mathrm{S}}^{n_S} : \bar{\pi}(s) = \tau_{\bar{\pi}}^\gamma\}| \geq F(h(\tau_{\bar{\pi}}^\gamma))$, as every token of $\mathbb{T}_{\mathrm{S}}^{n_S+1}$ but one (the ghost) has a preimage by $\sigma_{\bar{\pi}}$.

Proof of $(\mathtt{I_{max}}[\bar{\pi}])$**:** The maximality in our choice of τ ensures $(\mathtt{I}_{\max}[\bar{\pi}])$ holds.

When the ghost stays put. Assume now that there is no type τ such that $\tau \neq \tau_\pi^\gamma$, $\tau_\pi^\gamma \rightharpoonup_k \tau$, and there is a τ-descending sequence in π. We then prove the following:

Claim. There exists at least $k+1$ tokens $s \in \mathbb{T}_{\mathrm{S}}^{n_S+1}$ such that $\mathcal{S}_+(\pi_+)(s) = \tau_\pi^\gamma$.

Proof. Let us decompose the play π as

$$\pi = C_0 \xrightarrow{m_{\mathrm{E}}^0} C_1 \xrightarrow{m_{\mathrm{S}}^1} C_2 \xrightarrow{m_{\mathrm{E}}^2} \cdots \xrightarrow{m_{\mathrm{S}}^{2l-1}} C_{2l} \xrightarrow{m_{\mathrm{E}}^{2l}} C_{2l+1}$$

It will be convenient to extend the notation of partial plays and to write $(\pi_{|2l+2})_+$ to denote the play $\pi_+ \xrightarrow{\mathcal{S}_+(\pi_+)} C'_+$.

According to $(\mathtt{I}_{\tau^\gamma}[\pi])$, the place of the ghost can only go down during the play π_+ in $\mathfrak{G}_+$. Let us consider the sequence $i_1 < \cdots < i_n$ of all indices $i \leq l$ such that

- $\tau^\gamma_{\pi_{|2i+1}} = \tau_\pi^\gamma$, meaning that the ghost at that point was already in the same place as it is after the play π, and
- if we denote by N_i the number of tokens $s \in \mathbb{T}_{\mathrm{S}}^{n_S+1}$ such that

  $$(\pi_{|2i+1})_+(s) = \tau_\pi^\gamma \text{ and } (\pi_{|2i+2})_+(s) \neq \tau_\pi^\gamma\,,$$

  we have $N_i \geq 1$. In other words, we consider only partial plays such that in the next move, at least one System token moved down from τ_π^γ.

Because of $(\mathtt{I_F}[\pi])$ and $(\mathtt{I_S}[\pi])$, we know that after $(\pi_{|2i_1+1})_+$, the number of System tokens in τ_π^γ is at least $F(h(\tau_\pi^\gamma)) + 1$. Thus

$$\sum_{i \in \{i_1,\ldots,i_n\}} N_i + |\{s \in \mathbb{T}_{\mathrm{S}}^{n_S+1} : \mathcal{S}_+(\pi_+)(s) = \tau_\pi^\gamma\}| \geq F(h(\tau_\pi^\gamma)) + 1\,. \tag{12}$$

We will show that $\sum_{i\in\{i_1,\dots,i_n\}} N_i$ cannot be too large, by proving that for every i,

$$N_i \leq |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}| \cdot \big(F(h(\tau^\gamma_\pi) - 1) + 1\big)\,, \tag{13}$$

and that n also must remain rather small:

$$n \leq |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}|^{n_E+1} \cdot (F(h(\tau^\gamma_\pi) - 1) + 1)\,. \tag{14}$$

Combining the definition of F in (4) with (12), (13) and (14) we get

$$|\{s \in \mathbb{T}_{\mathrm{S}}^{n_S+1} : \pi_+(s) = \tau^\gamma_\pi\}| \geq k+1$$

as stated by the claim. Note that one can prove tighter bounds for (13) and (14), and thus pick a smaller $f_S(k, n_E)$ in (5). We refrain from doing so for the sake of readability.

Let us first show that (13) holds: if there exists some $i \in \{i_1, \dots, i_n\}$ such that

$$N_i > |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}| \cdot \big(F(h(\tau^\gamma_\pi) - 1) + 1\big)$$

then by the pigeonhole principle, at least one type $\tau \neq \tau^\gamma_\pi$ with $\tau^\gamma_\pi \rightharpoonup_k \tau$ is such that there exist at least $F(h(\tau^\gamma_\pi) - 1) + 1$ (which is at least $F(h(\tau)) + 1$ because F is non-decreasing) tokens $s \in \mathbb{T}_{\mathrm{S}}^{n_S+1}$ such that

$$(\pi_{|2i+1})_+(s) = \tau^\gamma_\pi = \tau^\gamma_{\pi_{|2i+1}} \text{ and } (\pi_{|2i+2})_+(s) = \tau\,.$$

Let us consider one such token $s^{n_S+1} \in \mathbb{T}_{\mathrm{S}}^{n_S+1}$. With $\xi := (s^{n_S+1}\, \gamma_{2i+1})$ (the permutation swapping s^{n_S+1} and γ_{2i+1}) and $\widehat{m} := \mathcal{S}_+((\pi_{|2i+1})_+)$, this τ contradicts $(\mathtt{I}_{\max}[\pi_{|2i+2}])$ – if $i = l$, it contradicts the non-existence of a τ with a τ-descending sequence.

To conclude the proof, let us prove that (14) holds. Assume toward a contradiction that

$$n \geq |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}|^{n_E+1} \cdot (F(h(\tau^\gamma_\pi) - 1) + 1)\,.$$

By the pigeonhole principle, there must exist a subsequence $j_1 < \cdots < j_{n'}$ of $i_1 < \cdots < i_n$ of length $n' \geq |\mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}| \cdot (F(h(\tau^\gamma_\pi) - 1) + 1)$ such that for $j, j' \in \{j_1, \dots, j_{n'}\}$,

$$\forall e \in \mathbb{T}_{\mathrm{E}},\ (\pi_{|2j+1})_+(e) = (\pi_{|2j'+1})_+(e)\,.$$

Given that $N_j \geq 1$ for each such j, the pigeonhole principle guarantees the existence of some $\tau \neq \tau^\gamma_\pi = \tau^\gamma_{2j_1+2}$ such that $\tau^\gamma_{2j_1+1} \rightharpoonup_k \tau$ and

$$\exists^{\geq F(h(\tau^\gamma_\pi)+1)+1} s \in \mathbb{T}_{\mathrm{S}}^{n_S+1},\ (\pi_{|2j_1+1})_+(s) = \tau^\gamma_\pi \text{ and } (\pi_{|2j_{n'}+2})_+(s) = \tau\,.$$

Let us consider one such token $s^{n_S+1} \in \mathbb{T}_{\mathrm{S}}^{n_S+1}$. With $\xi := (s^{n_S+1}\, \gamma_{2j_1+1})$ (the permutation swapping s^{n_S+1} and γ_{2j_1+1}) and

$$\widehat{m} := m_{\mathrm{S}}^{2j_1+1} \cdot m_{\mathrm{E}}^{2j_1+2} \cdots m_{\mathrm{S}}^{2j_{n'}+2}\,,$$

this τ-descending sequence contradicts $(\mathtt{I}_{\max}[\pi_{|2j_1+2}])$, which ends the proof of the claim. □

Let us consider a token $s^{n_S+1} \in \mathbb{T}_S^{n_S+1} \cap \mathrm{Im}(\sigma_\pi)$ such that $\mathcal{S}_+(\pi_+)(s^{n_S+1}) = \tau_\pi^\gamma$ and the token $s^{n_S} \in \mathbb{T}_S^{n_S}$ such that $\sigma_\pi(s^{n_S}) = s^{n_S+1}$.

We define $\mathcal{S}(\pi)$ as follows:

$$\begin{cases} \forall s \in \mathbb{T}_S^{n_S} \setminus \{s^{n_S}\},\ \mathcal{S}(\pi)(s) := \mathcal{S}_+(\pi_+)(\sigma_\pi(s)) \\ \mathcal{S}(\pi)(s^{n_S}) := \mathcal{S}(\pi_+)(\gamma_\pi) \end{cases} \tag{15}$$

This is well defined, as for every $s \in \mathbb{T}_S^{n_S}$ different from s^{n_S} we have

$$\pi(s) = \pi_+(\sigma_\pi(s)) \rightharpoonup_k \mathcal{S}_+(\pi_+)(\sigma_\pi(s))$$

as well as $\pi(s^{n_S}) = \pi_+(s^{n_S+1}) \rightharpoonup_k \tau_\pi^\gamma \rightharpoonup_k \mathcal{S}(\pi_+)(\gamma_\pi)$.

Once again, we write $\bar{\pi}$ for the play π followed by the response we just defined

$$\bar{\pi} := \pi \xrightarrow{\mathcal{S}(\pi)} C' .$$

We then define

$$\bar{\pi}_+ := \pi_+ \xrightarrow{\mathcal{S}_+(\pi_+)} C'_+ \tag{16}$$

and

$$\forall s \in \mathbb{T}_S^{n_S},\ \sigma_{\bar{\pi}}(s) := \begin{cases} \gamma_\pi & \text{if } s = s^{n_S} \\ \sigma_\pi(s) & \text{otherwise} \end{cases} \tag{17}$$

which immediately gives

$$\gamma_{\bar{\pi}} = s^{n_S+1} \quad \text{and} \quad \tau_{\bar{\pi}}^\gamma = \tau_\pi^\gamma . \tag{18}$$

Invariant $(\mathtt{I}_F[\bar{\pi}])$ is satisfied due to the previous claim, $(\mathtt{I}_{\max}[\bar{\pi}])$ due to the assumption that there is no descending sequence, and all other invariants are straightforward from the definition of $\bar{\pi}$.

***$\mathcal{S}$* is a winning strategy for System.** In order to conclude the proof of Lemma 12, it remains to prove that $\mathcal{S}$ is a winning strategy for System in $\mathfrak{G}$. To that end, let us consider a maximal play $\widehat{\pi}$ compatible with $\mathcal{S}$. Let us define the play $\widehat{\pi}_+$ defined as the limit of π_+ for increasing prefixes π of $\widehat{\pi}$, which is well defined due to $(\mathtt{I}_+[\pi])$.

There exists a finite prefix π of $\widehat{\pi}$ such that the configurations reached after π and π_+ are the limit configurations of $\widehat{\pi}$ and $\widehat{\pi}_+$ respectively. Since $\widehat{\pi}_+$ is a winning play for System, the configuration reached after π_+ must be winning for System. Let us argue that the last configuration of π is winning, which entails that $\widehat{\pi}$ is a winning play for System.

For every $\tau \in \mathrm{Types}^k_{\mathrm{FO}^{\mathrm{PREF}}}$,

- the number of Environment tokens in τ after π and the number of Environment tokens in τ after π_+ are equal, in view of $(\mathtt{I}_E[\pi])$, and
- following $(\mathtt{I}_S[\pi])$ and $(\mathtt{I}_F[\pi])$, the number of System tokens in τ after π and the number of System tokens in τ after π_+ are either equal or both larger than k, k being the depth of the game.

Since the configuration reached after π_+ is winning for System, the configuration reached after π is winning as well for System.

All in all, we have shown that any winning strategy $\mathcal{S}_+$ for System in $\mathfrak{G}_+$ ensures the existence of a winning strategy $\mathcal{S}$ for System in $\mathfrak{G}$. This concludes the proof of Lemma 12. □

D Relation between FO and $\mathrm{FO}^{\mathrm{PREF}}$ types

In this section, we investigate the link between FO and $\mathrm{FO}^{\mathrm{PREF}}$ types on regular words. Note first that every $\mathrm{FO}^{\mathrm{PREF}}$-sentence of quantifier depth k is equivalent to some FO-sentence of quantifier depth k. What this means is that FO k-types are more fine-grained than $\mathrm{FO}^{\mathrm{PREF}}$ k-types – in particular, it makes sense to speak of the $\mathrm{FO}^{\mathrm{PREF}}$ k-type of an FO k-type.

Loosely speaking, two finite words u and v agree on $\mathrm{FO}^{\mathrm{PREF}}$ sentences if and only if there exists words u' and v' such that $u \cdot u'$ and v agree on FO sentences, as well as u and $v \cdot v'$. This is stated more precisely in the two following lemmas.

Lemma 18. *Let u, v be two finite words for which there exists u', v' such that $u \cdot u' \equiv_k^{FO} v$ and $u \equiv_k^{FO} v \cdot v'$. Then $u \equiv_k^{FO^{PREF}} v$.*

Proof. Since $\equiv_k^{\mathrm{FO}}$ is a congruence for the concatenation in the monoid of finite words, the assumptions combine to give $u \cdot u' \cdot v' \equiv_k^{\mathrm{FO}} u$, which implies $u \cdot u' \cdot v' \equiv_k^{\mathrm{FO}^{\mathrm{PREF}}} u$.

Lemma 3 then ensures that $u \cdot u' \equiv_k^{\mathrm{FO}^{\mathrm{PREF}}} u$. This equivalence, combined with $u \cdot u' \equiv_k^{\mathrm{FO}^{\mathrm{PREF}}} v$ (which follows directly from the assumptions), concludes the proof. □

Conversely, we have:

Lemma 19. *Let u, v be two finite words such that $u \equiv_{k+1}^{FO^{PREF}} v$. Then there exists u', v' such that $u \cdot u' \equiv_k^{FO} v$ and $u \equiv_k^{FO} v \cdot v'$.*

Before turning to the proof of this lemma, let us note that the requirement that u and v be $\mathrm{FO}^{\mathrm{PREF}}$ similar at depth $k+1$ (and not just k) is necessary. Consider indeed the words

$$u := abaac$$

and

$$v := abaaac$$

in the case $k := 3$.

Let us consider the following strategy for the Duplicator in the 3-round prefix Ehrenfeucht-Fraïssé game between u and v: if the Spoiler places their bounding pebble on any of the first four positions of u or v, then the Duplicator responds by playing on the same position in the other word, and wins since both prefixes are then isomorphic. Otherwise, if the Spoiler starts by playing on the last a of v, then the Duplicator plays on the last a of u, and can easily win two more

rounds from there. Finally, if the Spoiler makes their first move on a c, then the Duplicator responds on the other c, and can once again win two more rounds from that configuration. We thus have $u \equiv_3^{\mathrm{FO}^{\mathrm{PREF}}} v$.

However, for any word u', $u \cdot u'$ and v will disagree on the FO-sentence of quantifier depth 3 stating the existence of three consecutives a's before the first occurrence of c.

Proof. Let u and v be finite words such that $u \equiv_{k+1}^{\mathrm{FO}^{\mathrm{PREF}}} v$. We can assume that both of them have at least one letter, for otherwise we already have $u \equiv_k^{\mathrm{FO}} v$.

We show the existence of some word u' such that $u \cdot u' \equiv_k^{\mathrm{FO}} v$, and we conclude by symmetry. By assumption, the Duplicator has a winning strategy in the $(k+1)$-round prefix Ehrenfeucht-Fraïssé game between u and v. Let n be the position in v of the Duplicator's response when the Spoiler first plays on the last letter of u; from that configuration, the Duplicator can win for k for rounds. We claim that $u \equiv_k^{\mathrm{FO}} v[0 \dots n]$: indeed, in the k-round (regular) Ehrenfeucht-Fraïssé game between u and $v[0 \dots n]$, the Duplicator can follow the above-mentioned strategy (in which every play in v is left of the bounding pebble, i.e. in $v[0 \dots n]$) and win.

Let us now define $u' := v[n+1 \dots]$: by virtue of $\equiv_k^{\mathrm{FO}}$ being a congruence for the concatenation of finite words, $u \equiv_k^{\mathrm{FO}} v[0 \dots n]$ entails $u \cdot u' \equiv_k^{\mathrm{FO}} v$, which concludes the proof. □