

BUILD YOUR OWN CLARITHMETIC II: SOUNDNESS

GIORGI JAPARIDZE

Department of Computing Sciences, Villanova University, 800 Lancaster Avenue, Villanova, PA 19085, USA
URL: <http://www.csc.villanova.edu/~japaridz/>
e-mail address: giorgi.japaridze@villanova.edu

ABSTRACT. *Clarithmetics* are number theories based on *computability logic*. Formulas of these theories represent interactive computational problems, and their “truth” is understood as existence of an algorithmic solution. Various complexity constraints on such solutions induce various versions of clarithmetic. The present paper introduces a parameterized/schematic version $\text{CLA11}_{P_4}^{P_1, P_2, P_3}$. By tuning the three parameters P_1, P_2, P_3 in an essentially mechanical manner, one automatically obtains sound and complete theories with respect to a wide range of target *tricomplexity* classes, i.e., combinations of time (set by P_3), space (set by P_2) and so called amplitude (set by P_1) complexities. Sound in the sense that every theorem T of the system represents an interactive number-theoretic computational problem with a solution from the given tricomplexity class and, furthermore, such a solution can be automatically extracted from a proof of T . And complete in the sense that every interactive number-theoretic problem with a solution from the given tricomplexity class is represented by some theorem of the system. Furthermore, through tuning the 4th parameter P_4 , at the cost of sacrificing recursive axiomatizability but not simplicity or elegance, the above *extensional completeness* can be strengthened to *intensional completeness*, according to which every formula representing a problem with a solution from the given tricomplexity class is a theorem of the system. This article is published in two parts. The previous Part I has introduced the system and proved its completeness, while the present Part II is devoted to proving soundness.

CONTENTS

1. Getting started	2
2. Soundness of Logical Consequence	3
3. Soundness of Comprehension	4
4. Providence, prudence, quasilegality and unconditionality	5
5. Soundness of Induction	9
5.1. Soon enough or never	11
5.2. The procedure SIM	13

2012 ACM CCS: [Theory of computation]: Computational complexity and cryptography—Complexity theory and logic; Logic.

Key words and phrases: Computability logic; Interactive computation; Implicit computational complexity; Game semantics; Peano arithmetic; Bounded arithmetic.

5.3.	Aggregations	18
5.4.	The procedure MAIN	19
5.5.	\mathcal{M} is a solution of the target game	20
5.6.	\mathcal{M} runs in target amplitude	36
5.7.	\mathcal{M} runs in target space	36
5.8.	\mathcal{M} runs in target time	38
6.	Final remarks	41
Appendix A. Proof of Lemma 4.4		42
Appendix B. Proof of Lemma 4.2		46
B.1.	Getting started	47
B.2.	Procedure UPDATE SKETCH	50
B.3.	Procedure FETCH SYMBOL	51
B.4.	Procedure MAKE HISTORY	51
B.5.	The overall strategy and an example of its run	52
B.6.	\mathcal{K} is a provident and prudent solution of H	55
B.7.	\mathcal{K} plays in target tricomplexity	58
References		60
Index		61

1. GETTING STARTED

Being a continuation of [5], this article fully relies on the terminology and notation introduced in its predecessor, with which — or, at least, with the first two sections of which — the reader is assumed to be already familiar. Just like [5], this article further relies on [3], and familiarity with that self-contained, tutorial-style paper (with proofs omitted) is another prerequisite for reading the present piece of writing.

The sole purpose if the present article is to prove the soundness of the system **CLA11** introduced in [5]. Specifically, the goal is to prove clause 3 of Theorem 2.6 of [5], which, slightly paraphrased, reads:

If a theory $\mathbf{CLA11}_{\mathcal{A}}^{\mathcal{R}}$ is regular, then there is an effective procedure that takes an arbitrary extended $\mathbf{CLA11}_{\mathcal{A}!}^{\mathcal{R}}$ -proof of an arbitrary sentence X and constructs an \mathcal{R} tricomplexity solution for X .

Our soundness proof is written so that it can be read independently of the completeness proof given in [5].

Let us get started right now. Assuming that a theory $\mathbf{CLA11}_{\mathcal{A}}^{\mathcal{R}}$ is regular, the above-displayed statement can be verified by induction on the number of steps in an extended $\mathbf{CLA11}_{\mathcal{A}!}^{\mathcal{R}}$ -proof of X . The basis of this induction is a rather straightforward observation that all axioms have \mathcal{R} tricomplexity solutions. Namely, in the case of Peano axioms such a “solution” is simply a machine that does nothing. All axioms from \mathcal{A} have \mathcal{R} tricomplexity solutions by condition 1 of Definition 2.5 of [5]; furthermore, according to the same condition, such solutions can be effectively obtained even when \mathcal{A} is infinite. Finally, the Successor, Log and Bit axioms can be easily seen to have linear amplitude, logarithmic space and polynomial (in fact, linear) time solutions and, in view of conditions 2 and 3 of Definition 2.2 of [5], such solutions are automatically also \mathcal{R} tricomplexity solutions. As for

the inductive step, it will be taken care of by the later-proven Theorems 2.1, 3.1 and 5.1, according to which the rules of Logical Consequence, \mathcal{R} -Comprehension and \mathcal{R} -Induction preserve — in a constructive sense — the property of having an \mathcal{R} tricomplexity solution.

2. SOUNDNESS OF LOGICAL CONSEQUENCE

As we remember from [3], CoL understands algorithmic strategies as interactive Turing machines called HPMs (Hard-Play Machines).

Theorem 2.1. *Consider any regular boundclass triple \mathcal{R} . There is an (\mathcal{R} -independent) effective procedure¹ that takes an arbitrary **CL12**-proof \mathbb{P} of an arbitrary \mathbb{L} -sequent $E_1, \dots, E_n \circ-F$, arbitrary HPMs $\mathcal{N}_1, \dots, \mathcal{N}_n$ and constructs an HPM \mathcal{M} such that, if $\mathcal{N}_1, \dots, \mathcal{N}_n$ are \mathcal{R} tricomplexity solutions of E_1, \dots, E_n , respectively, then \mathcal{M} is an \mathcal{R} tricomplexity solution of F .*

Proof. Such an effective procedure is nothing but the one whose existence is stated in Theorem 11.1 of [3]. Consider an arbitrary **CL12**-proof \mathbb{P} of an arbitrary \mathbb{L} -sequent $E_1, \dots, E_n \circ-F$, and arbitrary HPMs $\mathcal{N}_1, \dots, \mathcal{N}_n$. Let \mathcal{M} be the HPM constructed for/from these parameters by the above procedure.

Assume \mathcal{R} is a regular boundclass triple, and $\mathcal{N}_1, \dots, \mathcal{N}_n$ are \mathcal{R} tricomplexity solutions of E_1, \dots, E_n , respectively. All three components of \mathcal{R} are linearly closed by condition 3 of Definition 2.2 of [5] and, being boundclasses, they are also closed under syntactic variation. This means that, for some common triple $(\mathfrak{a}(x), \mathfrak{s}(x), \mathfrak{t}(x)) \in \mathcal{R}_{\text{amplitude}} \times \mathcal{R}_{\text{space}} \times \mathcal{R}_{\text{time}}$ of unary bounds, all n machines run in tricomplexity $(\mathfrak{a}, \mathfrak{s}, \mathfrak{t})$. That is, we have:

- (i) For each $i \in \{1, \dots, n\}$, \mathcal{N}_i is an \mathfrak{a} amplitude, \mathfrak{s} space and \mathfrak{t} time solution of E_i .

In view of conditions 2 and 5 of Definition 2.2 of [5], we may further assume that:

- (ii) For any x , $\mathfrak{a}(x) \geq x$.
- (iii) For any x , $\mathfrak{s}(x) \geq \log(x)$.
- (iv) For any x , $\mathfrak{t}(x) \geq x$ and $\mathfrak{t}(x) \geq \mathfrak{s}(x)$.

Now, remembering that E_i stands for E_i^\dagger , our condition (i) is the same as condition (i) of Theorem 11.1 of [3] with † in the role of $*$. Next, taking into account that 0 is the only constant that may appear in the \mathbb{L} -sequent $E_1, \dots, E_n \circ-F$ and hence the native magnitude of the latter is 0, our condition (ii) is the same as condition (ii) of Theorem 11.1 of [3]. Finally, our conditions (iii) and (iv) are the same as conditions (iii) and (iv) of Theorem 11.1 of [3].

Then, according to that theorem, there are numbers \mathfrak{b} and \mathfrak{d} such that \mathcal{M} is an $\mathfrak{a}^{\mathfrak{b}}(\ell)$ amplitude, $O(\mathfrak{s}(\mathfrak{a}^{\mathfrak{b}}(\ell)))$ space and $O((\mathfrak{t}(\mathfrak{a}^{\mathfrak{b}}(\ell)))^{\mathfrak{d}})$ time solution of F . But, by condition 2 (if $\mathfrak{b} = 0$) or 4 (if $\mathfrak{b} > 0$) of Definition 2.2 of [5], we have $\mathfrak{a}^{\mathfrak{b}}(\ell) \preceq \mathcal{R}_{\text{amplitude}}$, meaning that \mathcal{M} runs in amplitude $\mathcal{R}_{\text{amplitude}}$. The fact $\mathfrak{a}^{\mathfrak{b}}(\ell) \preceq \mathcal{R}_{\text{amplitude}}$, again by condition 4 of Definition 2.2 of [3], further implies that $\mathfrak{s}(\mathfrak{a}^{\mathfrak{b}}(\ell)) \preceq \mathcal{R}_{\text{space}}$ and $\mathfrak{t}(\mathfrak{a}^{\mathfrak{b}}(\ell)) \preceq \mathcal{R}_{\text{time}}$. The fact $\mathfrak{t}(\mathfrak{a}^{\mathfrak{b}}(\ell)) \preceq \mathcal{R}_{\text{time}}$, in turn, by condition 3 of Definition 2.2 of [3], further implies that $(\mathfrak{t}(\mathfrak{a}^{\mathfrak{b}}(\ell)))^{\mathfrak{d}} \preceq \mathcal{R}_{\text{time}}$. Now, by Remark 2.4 of [3], the facts $\mathfrak{s}(\mathfrak{a}^{\mathfrak{b}}(\ell)) \preceq \mathcal{R}_{\text{space}}$ and $(\mathfrak{t}(\mathfrak{a}^{\mathfrak{b}}(\ell)))^{\mathfrak{d}} \preceq \mathcal{R}_{\text{time}}$, together with the earlier observation that \mathcal{M} runs in $O(\mathfrak{s}(\mathfrak{a}^{\mathfrak{b}}(\ell)))$ space and $O((\mathfrak{t}(\mathfrak{a}^{\mathfrak{b}}(\ell)))^{\mathfrak{d}})$ time, imply that \mathcal{M} runs in space $\mathcal{R}_{\text{space}}$ and time $\mathcal{R}_{\text{time}}$. To summarize, \mathcal{M} runs in tricomplexity \mathcal{R} , as desired. \square

¹Here and later in the similar Theorems 3.1 and 5.1, as one can easily guess, \mathcal{R} -independence of a procedure means that the procedure is the same regardless of what particular value \mathcal{R} assumes.

3. SOUNDNESS OF COMPREHENSION

Theorem 3.1. *Consider any regular boundclass triple \mathcal{R} . There is an (\mathcal{R} -independent) effective procedure that takes an arbitrary application² of \mathcal{R} -Comprehension, an arbitrary HPM \mathcal{N} and constructs an HPM \mathcal{M} such that, if \mathcal{N} is an \mathcal{R} tricomplexity solution of the premise, then \mathcal{M} is an \mathcal{R} tricomplexity solution of the conclusion.*

The rest of this section is devoted to a proof of the above theorem. Consider a regular boundclass triple \mathcal{R} . Further consider an HPM \mathcal{N} , and an application

$$\frac{p(y) \sqcup \neg p(y)}{\sqcup |x| \leq \mathfrak{b}|\vec{s}| \forall y < \mathfrak{b}|\vec{s}| (Bit(y, x) \leftrightarrow p(y))} \quad (3.1)$$

of \mathcal{R} -Comprehension. Let $\vec{v} = v_1, \dots, v_n$ be a list of all free variables of $p(y)$ other than y , and let us correspondingly rewrite (3.1) as

$$\frac{p(y, \vec{v}) \sqcup \neg p(y, \vec{v})}{\sqcup |x| \leq \mathfrak{b}|\vec{s}| \forall y < \mathfrak{b}|\vec{s}| (Bit(y, x) \leftrightarrow p(y, \vec{v}))}. \quad (3.2)$$

By condition 1 of Definition 2.2 of [5], from the bound $\mathfrak{b}(\vec{s})$ we can effectively extract an \mathcal{R} tricomplexity solution of $\sqcup \sqcup z(z = \mathfrak{b}|\vec{s}|)$. Fix such a solution/algorithm and call it ALGO.

Assume \mathcal{N} is an $(\mathfrak{a}, \mathfrak{s}, \mathfrak{t}) \in \mathcal{R}_{\text{amplitude}} \times \mathcal{R}_{\text{space}} \times \mathcal{R}_{\text{time}}$ tricomplexity solution of the premise of (3.2). We want to construct an \mathcal{R} tricomplexity solution \mathcal{M} for the conclusion of (3.2). It should be noted that, while our claim of \mathcal{M} 's being an \mathcal{R} tricomplexity solution of the conclusion of (3.2) relies on the assumption that we have just made regarding \mathcal{N} , our construction of \mathcal{M} itself does not depend on that assumption. It should also be noted that we construct \mathcal{M} as a single-work-tape machine.

This is how \mathcal{M} works. At the beginning, it puts the symbol $\#$ into its buffer. Then it waits till Environment specifies constants \vec{a} and \vec{b} for the free variables \vec{s} and \vec{v} of the conclusion of (3.2). If Environment never does so, then \mathcal{M} is an automatic winner. Otherwise, the game is brought down to $\sqcup |x| \leq \mathfrak{b}|\vec{a}| \forall y < \mathfrak{b}|\vec{a}| (Bit(y, x) \leftrightarrow p(y, \vec{b}))$. Now, using ALGO, \mathcal{M} computes and remembers the value c of $\mathfrak{b}|\vec{a}|$. Condition 5 of Definition 2.2 of [5] guarantees that c can be remembered with $\mathcal{R}_{\text{space}}$ space. Thus, recalling that ALGO runs in \mathcal{R} tricomplexity, the steps taken by \mathcal{M} so far do not take us beyond \mathcal{R} and hence, in view of Remark 2.4 of [5], can be ignored in our asymptotic analysis when arguing that \mathcal{M} runs in \mathcal{R} tricomplexity. After these initial steps, \mathcal{M} starts acting according to the following procedure:

Procedure ROUTINE:

Step 1. If $c = 0$, enter a move state and retire. Otherwise, if $c \geq 1$, simulate the play of the premise of (3.2) by \mathcal{N} in the scenario where, at the very beginning of the play, \mathcal{N} 's adversary chose the same constants \vec{b} for the variables \vec{v} as Environment did in the real play of the conclusion and, additionally, chose j for y , where $j = c - 1$. If (when) the simulation shows that, at some point, \mathcal{N} chose the \sqcup -disjunct $\neg p(j, \vec{b})$, decrement the value of c by 1 and repeat the present step. And if (when) the simulation shows that, at some point, \mathcal{N} chose the \sqcup -disjunct $p(j, \vec{b})$, decrement the value of c by 1, put the bit 1 into the buffer, and go to Step 2.

²Here and elsewhere in similar contexts, an “application” means an “instance”, i.e., a particular premise-conclusion pair. In the case of \mathcal{R} -Comprehension, it is fully determined by the comprehension formula and the comprehension bound.

Step 2. If $c = 0$, enter a move state and retire. Otherwise, if $c \geq 1$, simulate the play of the premise of (3.2) by \mathcal{N} in the scenario where, at the very beginning of the play, \mathcal{N} 's adversary chose the same constants \vec{b} for the variables \vec{v} as Environment did in the real play of the conclusion and, additionally, chose j for y , where $j = c - 1$. If (when) the simulation shows that, at some point, \mathcal{N} chose the \sqcup -disjunct $\neg p(j, \vec{b})$ (resp. $p(j, \vec{b})$), decrement the value of c by 1, put the bit 0 (resp. 1) into the buffer, and repeat the present step.

It is not hard to see that, what \mathcal{M} did while following the above routine was that it constructed, in its buffer, the constant d with $|d| \leq \mathfrak{b}|\vec{a}| \wedge \forall y < \mathfrak{b}|\vec{a}| (Bit(y, d) \leftrightarrow p(y, \vec{b}))$, and then made $\#d$ as its only move in the play. This means that \mathcal{M} is a solution of the conclusion of (3.2), as desired. And, of course, our construction of \mathcal{M} is effective. It thus remains to see that \mathcal{M} runs in \mathcal{R} tricomplexity. In what follows, we implicitly rely on Remark 2.4 of [5], the monotonicity of bounds and the obvious fact that the background of any cycle of the simulated \mathcal{N} does not exceed the background of (the cycles of) \mathcal{M} throughout its work within ROUTINE. The latter is the case because all moves that reside on \mathcal{N} 's imaginary run tape — namely, the moves (containing) \vec{b} — also reside on \mathcal{M} 's run tape.

Since $\#d$ is the only move that \mathcal{M} makes, our earlier observation $|d| \leq \mathfrak{b}|\vec{a}|$ immediately implies that \mathcal{M} runs in amplitude $\mathfrak{b} \in \mathcal{R}_{\text{amplitude}}$, as desired.

Next, observe that the space that \mathcal{M} consumes while performing ROUTINE is just the space needed to remember the value of the variable c , plus the space needed to simulate \mathcal{N} . The value of c never exceeds $\mathfrak{b}|\vec{a}|$, remembering which, as we have already observed, does not take us beyond the target $\mathcal{R}_{\text{space}}$. In order to simulate \mathcal{N} , on its work tape \mathcal{M} does not need to keep track of \mathcal{N} 's run tape, because information on that content is available on \mathcal{M} 's own run tape. So, \mathcal{M} (essentially) only needs to keep track of \mathcal{N} 's work-tape contents. By our assumption, \mathcal{N} runs in space \mathfrak{s} . Therefore, keeping track of its work-tape contents takes $O(\mathfrak{s})$ space, which is again within $\mathcal{R}_{\text{space}}$. To summarize, \mathcal{M} runs in space $\mathcal{R}_{\text{space}}$, as desired.

Finally, taking into account that \mathcal{N} runs in time \mathfrak{t} and space \mathfrak{s} , it is clear that the time needed for any given iteration of either step of ROUTINE is $O(\mathfrak{t} \times \mathfrak{s})$. This is so because simulating each step of \mathcal{N} takes $O(\mathfrak{s})$ time, and there are $O(\mathfrak{t})$ steps to simulate. Altogether, there are $O(\mathfrak{b})$ iterations of either Step 1 or Step 2 of ROUTINE. So, \mathcal{M} runs in time $O(\mathfrak{t} \times \mathfrak{s} \times \mathfrak{b})$. Then, in view of the fact that both $\mathfrak{s} \in \mathcal{R}_{\text{space}} \preceq \mathcal{R}_{\text{time}}$ and $\mathfrak{b} \in \mathcal{R}_{\text{amplitude}} \preceq \mathcal{R}_{\text{time}}$ (condition 5 of Definition 2.2 of [5]), we find that \mathcal{M} runs in time $O(\mathfrak{t} \times \mathfrak{t}_1 \times \mathfrak{t}_2)$ for some $\mathfrak{t}_1, \mathfrak{t}_2 \in \mathcal{R}_{\text{time}}$. But $\mathcal{R}_{\text{time}}$ is polynomially closed (condition 3 of Definition 2.2 of [5]), thus containing $\mathfrak{t} \times \mathfrak{t}_1 \times \mathfrak{t}_2$. So, \mathcal{M} runs in time $\mathcal{R}_{\text{time}}$, as desired.

4. PROVIDENCE, PRUDENCE, QUASILEGALITY AND UNCONDITIONALITY

In this section we establish certain terminology and facts necessary for our subsequent proof of the soundness of the induction rule.

A **numeric (lab)move** means a (lab)move ending in $\#b$ for some constant b . We shall refer to such a b as the **numer** of the (lab)move. To make the “numer” function total, we stipulate that the numer of a non-numeric move is 0 (is the empty string ϵ , that is).

Consider a bounded formula F . Let n be the number of occurrences of choice quantifiers in F , and $\mathfrak{b}_1(\vec{z}_1), \dots, \mathfrak{b}_n(\vec{z}_n)$ be the bounds used in those occurrences. Let $f(z)$ be the unarification (cf. [3], Section 12) of $\max(\mathfrak{b}_1(\vec{z}_1), \dots, \mathfrak{b}_n(\vec{z}_n))$. Here and elsewhere, as expected,

$\max(x_1, \dots, x_n)$ stands for the greatest of the numbers x_1, \dots, x_n , and is understood as 0 if $n = 0$. Finally, let \mathfrak{G} be the function defined by $\mathfrak{G}(z) = \max(f(z), f^2(z), \dots, f^n(z))$. Here, as in Section 2, $f^i(z)$ denotes the n -fold composition of f with itself, i.e., $f(f(\dots(f(z))\dots))$, with “ f ” repeated i times. Then we call the functions f and \mathfrak{G} the **subaggregate bound** and the **superaggregate bound** of F , respectively. As an aside, for our purposes, a “much smaller” function could have been taken in the role of superaggregate bound, but why try to economize.

Lemma 4.1. *Assume \mathcal{R} is a regular boundclass triple, F is an \mathcal{R}_{space} -bounded formula, and \mathfrak{G} is the superaggregate bound of F . Then $\mathfrak{G} \preceq \mathcal{R}_{space}$.*

Proof. Assume the conditions of the lemma. Further let $n, \mathfrak{b}_1(\vec{z}_1), \dots, \mathfrak{b}_n(\vec{z}_n), f$ be as in the paragraph preceding Lemma 4.1. Take a note of the fact that $\mathfrak{b}_1(\vec{z}_1), \dots, \mathfrak{b}_n(\vec{z}_n) \in \mathcal{R}_{space}$. If all tuples $\vec{z}_1, \dots, \vec{z}_n$ are empty, then (f and hence) \mathfrak{G} is a constant function and, by the linear closure of \mathcal{R}_{space} , $\mathfrak{G} \preceq \mathcal{R}_{space}$. Suppose now at least one of the tuples $\vec{z}_1, \dots, \vec{z}_n$ is nonempty. Pick one variable z among $\vec{z}_1, \dots, \vec{z}_n$, and consider the pterm $u(z)$ obtained from $\mathfrak{b}_1(\vec{z}_1) + \dots + \mathfrak{b}_n(\vec{z}_n)$ as a result of replacing all variables $\vec{z}_1, \dots, \vec{z}_n$ by z . Since \mathcal{R}_{space} is closed under syntactic variation as well as under $+$, we have $u(z) \in \mathcal{R}_{space}$. But obviously $f(z) \preceq u(z)$. Thus, $f(z) \preceq \mathcal{R}_{space}$. In view of condition 4 of Definition 2.2 of [5], $f(z) \preceq \mathcal{R}_{space}$ can be seen to imply $f^2(z) \preceq \mathcal{R}_{space}, f^3(z) \preceq \mathcal{R}_{space}, \dots$. Consequently, by the closure of \mathcal{R}_{space} under $+$, $f(z) + f^2(z) + \dots + f^n(z) \preceq \mathcal{R}_{space}$. But $\mathfrak{G}(z) \preceq f(z) + f^2(z) + \dots + f^n(z)$. Thus, $\mathfrak{G} \preceq \mathcal{R}_{space}$. \square

Recall from [3] that a **provident** computation branch of a given HPM \mathcal{M} is one containing infinitely many configurations with empty buffer contents (intuitively meaning that \mathcal{M} has actually made all moves that it has ever started to construct in its buffer). Then, given a constant game G , \mathcal{M} is said to play G **providently** iff every computation branch of \mathcal{M} that spells a \perp -legal run of G is provident. And \mathcal{M} is a **provident solution** of G iff \mathcal{M} is a solution of G and plays it providently.

Let $H(\vec{y}) = H(y_1, \dots, y_n)$ be a bounded formula with all free variables displayed, \mathfrak{G} be the superaggregate bound of $H(\vec{y})$, and $\vec{c} = c_1, \dots, c_n$ be an n -tuple of constants. We say that a move α is a **prudent** move of $H(\vec{c})$ iff the size of the numer of α does not exceed $\mathfrak{G}|\max(\vec{c})|$. The **$H(\vec{c})$ -prudentization** of α is defined as the following move α' . If α is a prudent move of $H(\vec{c})$, then $\alpha' = \alpha$. Suppose now α is not a prudent move of $H(\vec{c})$, meaning that α is a numeric move $\beta \# b$ with an “oversized” numer b . In this case we stipulate that $\alpha' = \beta \# a$, where a (as a bitstring) is the longest initial segment of b such that $\beta \# a$ is a prudent move of $H(\vec{c})$.

Further consider any run Γ and either player $\wp \in \{\top, \perp\}$. We say that Γ is a **\wp -prudent run** of $H(\vec{c})$ iff all \wp -labeled moves of Γ are prudent moves of $H(\vec{c})$. When we simply say “prudent” without indicating a player, it means both \top -prudent and \perp -prudent.

Further consider any machine \mathcal{M} . By saying that \mathcal{M} **plays $H(\vec{c})$ prudently**, we shall mean that, whenever $\langle \perp c_1, \dots, \perp c_n, \Gamma \rangle$ is a \perp -legal run of $\Box H(\vec{y})$ generated by \mathcal{M} , Γ is a \top -prudent run of $H(\vec{c})$. On the other hand, when we say that \mathcal{M} **plays $H(\vec{y})$ prudently**, we mean that, for any n -tuple \vec{c} of constants, \mathcal{M} plays $H(\vec{c})$ prudently. A **prudent solution** of $H(\vec{y})$ means an HPM that wins $H(\vec{y})$ — wins $\Box H(\vec{y})$, that is — and plays $H(\vec{y})$ prudently.

Lemma 4.2. *There is an effective procedure that takes an arbitrary bounded formula $H(\vec{y})$, an arbitrary HPM \mathcal{N} and constructs an HPM \mathcal{L} such that, for any regular boundclass triple*

\mathcal{R} , if $H(\vec{y})$ is $\mathcal{R}_{\text{space}}$ -bounded and \mathcal{N} is an \mathcal{R} tricomplexity solution of $H(\vec{y})$, then \mathcal{L} is a provident and prudent \mathcal{R} tricomplexity solution of $H(\vec{y})$.

Proof idea. \mathcal{L} is a machine that waits till $\Box H(\vec{y})$ is brought down to $H(\vec{c})$ for some constants \vec{c} and then, through simulating and mimicking \mathcal{N} within the specified complexity constraints, plays $H(\vec{c})$ just as \mathcal{N} would play it, with essentially the only difference that each (legal) move α made by \mathcal{N} is made by \mathcal{L} in the prudentized form α' . This does not decrease the chances of \mathcal{L} (compared with those of \mathcal{N}) to win: imprudent moves are at best inconsequential and at worst disadvantageous (resulting in a loss of the corresponding subgame) for a player, so, if the machine wins the game while it makes the imprudent move α , it would just as well (and “even more so”) win the game if it had made the prudent move α' instead. This is how prudence is achieved. As for providence, \mathcal{L} achieves it by never putting anything into its buffer unless it has already decided to make a move, after seeing that the simulated \mathcal{N} has moved.

Of course, the above strategy may yield some discrepancies between the contents of \mathcal{L} 's run tape and \mathcal{N} 's imaginary run tape: it is possible that the latter is showing a (\top -labeled) move α while the former is showing only a properly smaller part (prudentization) α' of α . To neutralize this problem, every time the simulated \mathcal{N} is trying to read some symbol b of α on its run tape, \mathcal{L} finds b through resimulating the corresponding portion of the work of \mathcal{N} . This, of course, results in \mathcal{L} 's being slower than \mathcal{N} ; yet, due to \mathcal{R} 's being regular, things can be arranged so that the running time of \mathcal{L} still remains within the admissible limits. ■

A detailed proof of Lemma 4.2, which materializes the above idea, is given in Appendix B. It can be omitted rather safely by a reader so inclined. The same applies to the forthcoming Lemma 4.4, whose proof idea is presented in this section and whose relatively detailed proof is given in Appendix A.

When Γ is a run, we let

$$\Gamma^\top \quad (\text{resp. } \Gamma^\perp)$$

denote the result of deleting in Γ all \perp -labeled (resp. \top -labeled) moves.

For a constant game A and run Γ , we say that Γ is a \top -quasilegal (resp. \perp -quasilegal) run of A iff there is a legal run Δ of A such that $\Delta^\top = \Gamma^\top$ (resp. $\Delta^\perp = \Gamma^\perp$). If we say “quasilegal” without the prefix “ \top -” or “ \perp -”, it is to be understood as “both \top -quasilegal and \perp -quasilegal”. We say that an HPM \mathcal{M} plays A quasilegally iff every run generated by \mathcal{M} is a \top -quasilegal run of A . A **quasilegal solution** of A is a solution of A that plays A quasilegally.

Our definitions of “ \mathcal{M} plays … providently” and “ \mathcal{M} plays … prudently”, just like our earlier [3] definitions of running within given complexity bounds, only look at (computation branches that spell) \perp -legal runs of a given game. Below we define stronger — “unconditional” — versions of such concepts, where the adversary’s having made an illegal move is no longer an excuse for the player to stop acting in the expected manner. Namely:

We say that an HPM \mathcal{M} plays **unconditionally providently**, or that \mathcal{M} is **unconditionally prudent**, iff all computation branches of \mathcal{M} are provident (note that the game that is being played is no longer relevant).

Consider an HPM \mathcal{M} , a bounded formula $H = H(\vec{y}) = H(y_1, \dots, y_n)$ with all free variables displayed, and an n -tuple $\vec{c} = c_1, \dots, c_n$ of constants. We say that \mathcal{M} **plays** $H(\vec{c})$ **unconditionally prudently** iff, whenever $\langle \perp c_1, \dots, \perp c_n, \Gamma \rangle$ is a run (whether it be \perp -legal or not) generated by \mathcal{M} , Γ is a \top -prudent run of $H(\vec{c})$. Next, when we say that \mathcal{M} **plays**

$H(\vec{y})$ **unconditionally prudently**, we mean that, for any n -tuple \vec{c} of constants, \mathcal{M} plays $H(\vec{c})$ unconditionally prudently.

The following definition of the unconditional versions of our complexity concepts is obtained from Definition 5.2 of [3] by simply dropping the condition “ \perp -legal” on the plays considered, and also removing any mention of a game A that is being played because the latter is no longer relevant.

Definition 4.3. Let \mathcal{M} be an HPM, and h a unary arithmetical function (if h is not unary, then it should be replaced by its unarification according to Convention 12.2 of [3]). We say that:

1. \mathcal{M} runs (plays) in **unconditional amplitude** h iff, in every computation branch of \mathcal{M} , whenever \mathcal{M} makes a move α , the magnitude of α does not exceed $h(\ell)$, where ℓ is the background of α ;
2. \mathcal{M} runs (plays) in **unconditional space** h iff, in every computation branch of \mathcal{M} , the spacecost of any given clock cycle c does not exceed $h(\ell)$, where ℓ is the background of c ;
3. \mathcal{M} runs (plays) in **unconditional time** h iff, in every computation branch of \mathcal{M} , whenever \mathcal{M} makes a move α , the timecost of α does not exceed $h(\ell)$, where ℓ is the background of α .

The above definition and the related concepts naturally — in the same way as in the old, “conditional” cases — extend from bounds (as functions) to boundclasses, as well as bound triples or boundclass triples. For instance, where \mathcal{C} is a boundclass, we say that \mathcal{M} runs (plays) in **unconditional time** \mathcal{C} iff it runs in unconditional time h for some $h \in \mathcal{C}$; where \mathcal{R} is a boundclass triple, we say that \mathcal{M} runs (plays) in **unconditional tricomplexity** \mathcal{R} iff it runs in unconditional amplitude $\mathcal{R}_{\text{amplitude}}$, unconditional space $\mathcal{R}_{\text{space}}$ and unconditional time $\mathcal{R}_{\text{time}}$; etc.

Lemma 4.4. *There is an effective procedure that takes an arbitrary bounded formula $H(\vec{y})$, an arbitrary HPM \mathcal{L} and constructs an HPM \mathcal{M} such that, as long as \mathcal{L} is a provident solution of $H(\vec{y})$, the following conditions are satisfied:*

1. \mathcal{M} is a quasilegal and unconditionally provident solution of $H(\vec{y})$.
2. If \mathcal{L} plays $H(\vec{y})$ prudently, then \mathcal{M} plays $H(\vec{y})$ unconditionally prudently.
3. For any arithmetical functions $\mathbf{a}, \mathbf{s}, \mathbf{t}$, if \mathcal{L} plays $H(\vec{y})$ in tricomplexity $(\mathbf{a}, \mathbf{s}, \mathbf{t})$, then \mathcal{M} plays in unconditional tricomplexity $(\mathbf{a}, \mathbf{s}, \mathbf{t})$.

Proof idea. In our preliminary attempt of constructing \mathcal{M} , we let it be a machine that works exactly like \mathcal{L} , except that \mathcal{M} retires as soon as it detects that the play has gone illegal. This way, unlike \mathcal{L} , \mathcal{M} is precluded from using Environment’s illegal actions as an excuse for some undesirable behavior of its own, such as making inherently illegal or oversized moves, or using excessive resources. That is, while \mathcal{L} “behaves well” only on the condition of Environment playing legally, \mathcal{M} is guaranteed to “behave well” unconditionally, because in legal cases \mathcal{M} ’s behavior coincides with that of \mathcal{L} , and in illegal cases \mathcal{M} simply does not “behave” at all. An unretired or not-yet-retired \mathcal{M} consumes exactly the same amount of time and space as \mathcal{L} does, because keeping track of whether the play has gone illegal only requires maintaining a certain bounded amount of information, which can be done through state (rather than work-tape) memory and hence done without any time or space overhead whatsoever. The only problem with the above solution is that \mathcal{M} ’s buffer

may not necessarily be empty at the time we want it to retire, and if so, then \mathcal{M} is not unconditionally provident. This minor complication is neutralized by letting \mathcal{M} , before retiring, extend (if necessary) the buffer content to a shortest possible move adding which to the already generated run does not destroy its \top -quasilegality, and then empty the buffer by making such a move in the play. ■

In what follows, we will be using the word “**reasonable**” (“**reasonably**”) as an abbreviation of “quasilegal(ly) and unconditionally prudent(ly)”. “**Unreasonable**” (“**unreasonably**”), as expected, means “not reasonable” (“not reasonably”). We can now strengthen Lemma 4.2 as follows:

Lemma 4.5. *There is an effective procedure that takes an arbitrary bounded formula $H(\vec{y})$, an arbitrary HPM \mathcal{N} and constructs an HPM \mathcal{M} such that, for any regular boundclass triple \mathcal{R} , if $H(\vec{y})$ is $\mathcal{R}_{\text{space}}$ -bounded and \mathcal{N} is an \mathcal{R} tricomplexity solution of $H(\vec{y})$, then \mathcal{M} is a reasonable, unconditionally provident and unconditionally \mathcal{R} tricomplexity solution of $H(\vec{y})$.*

Proof. Immediately from Lemmas 4.2 and 4.4. □

5. SOUNDNESS OF INDUCTION

Theorem 5.1. *Consider any regular boundclass triple \mathcal{R} . There is an (\mathcal{R} -independent) effective procedure that takes an arbitrary application of \mathcal{R} -Induction, arbitrary HPMs \mathcal{N}, \mathcal{K} and constructs an HPM \mathcal{M} such that, if \mathcal{N} and \mathcal{K} are \mathcal{R} tricomplexity solutions of the two premises, then \mathcal{M} is an \mathcal{R} tricomplexity solution of the conclusion.*

The rest of this long section is devoted to a proof of the above theorem. It should be noted that some ideas used in our proof are borrowed from [4].

Consider any regular boundclass triple \mathcal{R} and any application

$$\frac{F(0) \quad F(x) \rightarrow F(x')}{x \leq \mathfrak{b}|\vec{s}| \rightarrow F(x)} \tag{5.1}$$

of \mathcal{R} -Induction. Assume

$$\vec{v} = v_1, \dots, v_{\mathfrak{v}}$$

— fix this number \mathfrak{v} — are exactly the free variables of $F(x)$ other than x listed in the lexicographic order, and let us correspondingly rewrite (5.1) as

$$\frac{F(0, \vec{v}) \quad F(x, \vec{v}) \rightarrow F(x', \vec{v})}{x \leq \mathfrak{b}|\vec{s}| \rightarrow F(x, \vec{v})}. \tag{5.2}$$

Further, assume that \mathcal{N} and \mathcal{K} are \mathcal{R} tricomplexity solutions of the left and the right premise of (5.2), respectively. In view of Lemma 4.5, we may and will assume that \mathcal{N} and \mathcal{K} are reasonable, unconditionally provident and unconditionally \mathcal{R} tricomplexity solutions of the corresponding premises. In view of the closure of all three components of \mathcal{R} under syntactic variation, in combination with the other relevant closure conditions from Definition 2.2 of [5], there is one common triple

$$(\mathfrak{a}, \mathfrak{s}, \mathfrak{t}) \in \mathcal{R}_{\text{amplitude}} \times \mathcal{R}_{\text{space}} \times \mathcal{R}_{\text{time}}$$

of unary bounds such that both \mathcal{N} and \mathcal{K} run in unconditional $(\mathfrak{a}, \mathfrak{s}, \mathfrak{t})$ tricomplexity. Fix these $\mathfrak{a}, \mathfrak{s}, \mathfrak{t}$ for the rest of this section.

We want to construct an \mathcal{R} tricomplexity solution \mathcal{M} of the conclusion of (5.2). It is important to point out that, as in the case of Comprehension, our construction of \mathcal{M} does not rely on the assumptions on \mathcal{N} and \mathcal{K} that we have just made. Also, the pathological case of $F(x, \vec{v})$ having no free occurrences of x is trivial and, for the sake of simplicity, we exclude it from our considerations. \mathcal{M} will be designed as a machine with a single work tape.

As usual in such cases, we adopt the Clean Environment Assumption (cf. Section 8 of [3]), according to which \mathcal{M} 's adversary never makes illegal moves of the game under consideration.

At the beginning, our \mathcal{M} waits for Environment to choose constants for all free variables of the conclusion of (5.2). We rule out the possibility that the adversary never does so, because then \mathcal{M} is an automatic winner trivially running in zero amplitude, zero space and zero time unless it deliberately tries not to. For the rest of this section, assume k is the constant chosen for the variable x , $\vec{c} = \vec{c}_1, \dots, \vec{c}_b$ are the constants chosen for \vec{v} , and \vec{d} are the constants chosen for \vec{s} . Since the case of $k = 0$ is straightforward and not worth paying separate attention, for further simplicity considerations we will assume for the rest of this section that $k \geq 1$. From now on, we shall write $F'(x)$ as an abbreviation of $F(x, \vec{c})$.

The above event of Environment's initial choice of constants brings the conclusion of (5.2) down to $k \leq b|\vec{d}| \rightarrow F(k, \vec{c})$, i.e., to $k \leq b|\vec{d}| \rightarrow F'(k)$. \mathcal{M} computes $b|\vec{d}|$ and compares it with k . By condition 1 of Definition 2.2 of [5], this can be done in space $\mathcal{R}_{\text{space}}$ and time $\mathcal{R}_{\text{time}}$. If $k \leq b|\vec{d}|$ is false, \mathcal{M} retires, obviously being the winner and satisfying the expected complexity conditions. For the rest of this section, we rule out this straightforward case and, in the scenarios that we consider, assume that $k \leq b|\vec{d}|$ is true.

We shall write \mathcal{H}_0 as an abbreviation of the phrase “ \mathcal{N} in the scenario where the adversary, at the beginning of the play, has chosen the constants \vec{c} for the variables \vec{v} ”. So, for instance, when saying that \mathcal{H}_0 moves on cycle t , it is to be understood as that, in the above scenario, \mathcal{N} moves on cycle t . As we see, strictly speaking, \mathcal{H}_0 is not a separate “machine” but rather it is just \mathcal{N} in a certain partially fixed scenario.³ Yet, for convenience and with some abuse of language, in the sequel we may terminologically and even conceptually treat \mathcal{H}_0 as if it was a machine in its own right — namely, the machine that works just like \mathcal{N} does in the scenario where the adversary, at the beginning of the play, has chosen the constants \vec{c} for the variables \vec{v} . Similarly, for any $n \geq 1$, we will write \mathcal{H}_n for the “machine” that works just like \mathcal{K} does in the scenario where the adversary, at the beginning of the play, has chosen the constants \vec{c} for the variables \vec{v} and the constant $n - 1$ for the variable x . So, \mathcal{H}_0 (thought of as a machine) wins the constant game $F'(0)$ and, for each $n \geq 1$, \mathcal{H}_n wins the constant game $F'(n - 1) \rightarrow F'(n)$.

In the same style as the notation \mathcal{H}_n is used, we write \mathcal{M}_k for the “machine” that works just like \mathcal{M} does after the above event of Environment's having chosen k , \vec{c} and \vec{d} for x , \vec{v} and \vec{s} , respectively. So, in order to complete our description of \mathcal{M} , it will suffice to simply define \mathcal{M}_k and say that, after Environment has chosen constants for all free variables of the conclusion of (5.2), \mathcal{M} continues playing like (“turns itself into”) \mathcal{M}_k . Correspondingly, in showing that \mathcal{M} wins $\sqcap(x \leq b|\vec{s}| \rightarrow F(x, \vec{v}))$, it will be sufficient to show that \mathcal{M}_k wins $k \leq b|\vec{d}| \rightarrow F'(k)$.

³The beginning of that scenario is fixed but the continuations may vary.

Remark 5.2. It should be noted that our treating of $\mathcal{H}_0, \dots, \mathcal{H}_k$ and \mathcal{M}_k as “machines” may occasionally generate some ambiguity or terminological inconsistencies, for which the author wants to apologize in advance. For instance, when talking about the content of \mathcal{H}_0 ’s run tape or the run spelled by a given computation branch of \mathcal{H}_0 , \mathcal{N} ’s adversary’s initial moves $\perp c_1, \dots, \perp c_v$ may or may not be meant to be included. Such ambiguities or inconsistencies, however, can usually be easily resolved based on the context.

In the informal description below, we use the term “**synchronizing**” to mean applying copycat between two (sub)games of the form A and $\neg A$. This means mimicking one player’s moves in A as the other player’s moves in $\neg A$, and vice versa. The effect achieved this way is that the games to which A and $\neg A$ eventually evolve (the final positions hit by them, that is) will be of the form A' and $\neg A'$ — that is, one will remain the negation of the other, so that one will be won by a given player iff the other is lost by the same player.

The idea underlying the work of \mathcal{M}_k can be summarized by saying that what \mathcal{M}_k does is a synchronization between $k + 2$ games, real or imaginary (simulated). Namely:

- It synchronizes the imaginary play of $F'(0)$ by \mathcal{H}_0 with the antecedent of the imaginary play of $F'(0) \rightarrow F'(1)$ by \mathcal{H}_1 .
- For each n with $1 \leq n < k$, it synchronizes the consequent of the imaginary play of $F'(n-1) \rightarrow F'(n)$ by \mathcal{H}_n with the antecedent of the imaginary play of $F'(n) \rightarrow F'(n+1)$ by \mathcal{H}_{n+1} .
- It (essentially) synchronizes the consequent of the imaginary play of $F'(k-1) \rightarrow F'(k)$ by \mathcal{H}_k with the real play in the consequent of $k \leq \mathfrak{b}|\vec{d}| \rightarrow F'(k)$.

Therefore, since \mathcal{H}_0 wins $F'(0)$ and each \mathcal{H}_n with $1 \leq n \leq k$ wins $F'(n-1) \rightarrow F'(n)$, \mathcal{M}_k wins $k \leq \mathfrak{b}|\vec{d}| \rightarrow F'(k)$ and thus \mathcal{M} wins (the \sqcap -closure of) $x \leq \mathfrak{b}|\vec{s}| \rightarrow F(x, \vec{v})$, as desired.

If space complexity was of no concern, a synchronization in the above-outlined style could be achieved by simulating all imaginary plays in parallel. Our general case does not allow us doing so though, and synchronization should be conducted in a very careful way. Namely, a parallel simulation of all plays is not possible, because there are up to $\mathfrak{b}|\vec{s}|$ simulations to perform, and there is no guarantee that this does not take us beyond the \mathcal{R}_{space} space limits. So, instead, simulations should be performed in some sequential rather than parallel manner, with subsequent simulations recycling the space used by the previous ones, and with the overall procedure keeping forgetting the results of most previous simulations and recomputing the same information over and over many times. We postpone our description of how \mathcal{M}_k exactly works to Section 5.4, after having elaborated all necessary preliminaries in Sections 5.1–5.3.

5.1. Soon enough or never.

Notation 5.3. We agree that throughout the rest of Section 5:

1. \mathfrak{l} denotes the length $|a|$ of the greatest constant a among k, \vec{c}, \vec{d} .
2. \mathfrak{e}_\top (resp. \mathfrak{e}_\perp) is the maximum number of \top -labeled (resp. \perp -labeled) moves in any legal run of $F'(0)$, and $\mathfrak{e} = \mathfrak{e}_\top + \mathfrak{e}_\perp$.
3. \mathfrak{G} is the superaggregate bound of $F(x, \vec{v})$.
4. $\mathfrak{L}(w, u)$ abbreviates

$$\mathfrak{r} \times (u + 1)^{\mathfrak{g}} \times \left((\mathfrak{v} + 1) \times (w + 2) + 2\mathfrak{e}(\mathfrak{G}(w) + \mathfrak{h} + 2) + 1 \right) \times \mathfrak{q}^{\mathfrak{g}u} \times 2\mathfrak{e},$$

where \mathfrak{v} , as we remember, is the number of variables in \vec{v} , and:

- \mathfrak{r} is the maximum number of states of the two machines \mathcal{N} and \mathcal{K} ;
- \mathfrak{g} is the maximum number of work tapes of the two machines \mathcal{N} and \mathcal{K} ;
- \mathfrak{q} is the maximum number of symbols that may ever appear on any of the tapes of the two machines \mathcal{N} and \mathcal{K} ;
- \mathfrak{h} is the length of the longest string β containing no $\#$ such that β is a prefix of some move of some legal run of $F'(0)$.

In the sequel, we may say about a machine or its adversary that it plays so and so (reasonably, prudently, etc.) without mentioning the context-setting game that is played. As expected, it will be understood that, in such cases, the game is: $\sqcap(x \leq \mathfrak{b}|\vec{s}| \rightarrow F(x, \vec{v}))$ if the machine is \mathcal{M} ; $\sqcap F(0, \vec{v})$ if the machine is \mathcal{N} ; $\sqcap(F(x, \vec{v}) \rightarrow F(x', \vec{v}))$ if the machine is \mathcal{K} ; $F'(0)$ if the machine is \mathcal{H}_0 ; $F'(n-1) \rightarrow F'(n)$ if the machine is \mathcal{H}_n with $1 \leq n \leq k$; and $k \leq \mathfrak{b}|\vec{d}| \rightarrow F'(k)$ if the machine is \mathcal{M}_k .

Below, Υ_0 denotes the sequence of \mathfrak{v} \perp -labeled moves signifying the choice of the constants \vec{c} for the free variables \vec{v} of $F(0, \vec{v})$ — that is, $\Upsilon_0 = \langle \perp\#c_1, \dots, \perp\#c_{\mathfrak{v}} \rangle$. And Υ_n , for $n \in \{1, \dots, k\}$, denotes the sequence of $\mathfrak{v} + 1$ \perp -labeled moves signifying the choice of the constants $n - 1$ and \vec{c} for the free variables x and \vec{v} of $F(x, \vec{v}) \rightarrow F(x', \vec{v})$, respectively.

Whenever we say that \mathcal{H}_n 's **adversary plays quasilegally**, we shall mean that we are only considering the runs Γ generated by \mathcal{H}_n (i.e., runs $\langle \Upsilon_0, \Gamma \rangle$ generated by \mathcal{N} and runs $\langle \Upsilon_n, \Gamma \rangle$ generated by \mathcal{K}) such that Γ is a \perp -quasilegal run of $F'(0)$ (if $n = 0$) or $F'(n-1) \rightarrow F'(n)$ (if $n \geq 1$). Similarly for the adversary's playing **unconditionally prudently** or **reasonably**. By the **symbolwise length** of a position Φ we shall mean the number of cells that Φ takes when spelled on the run tape. Similarly for labmoves.

Lemma 5.4. *For any $n \in \{0, \dots, k\}$, at any time in any play by \mathcal{H}_n , as long as \mathcal{H}_n 's adversary plays reasonably, the symbolwise length of the position spelled on the run tape of \mathcal{H}_n does not exceed $(\mathfrak{v} + 1) \times (\mathfrak{l} + 2) + 2\epsilon(\mathfrak{G}(\mathfrak{l}) + \mathfrak{h} + 2)$.*

Proof. Any position spelled on the run tape of \mathcal{H}_n looks like $\langle \Upsilon_n, \Gamma \rangle$. The symbolwise length of the Υ_n part is at most $(\mathfrak{v} + 1) \times (\mathfrak{l} + 2)$, with $\mathfrak{v} + 1$ being the (maximum) number of labmoves in Υ_n and $\mathfrak{l} + 2$ being the maximum symbolwise length of each labmove, including the prefix $\perp\#$. By our assumption, \mathcal{H}_n ⁴ plays reasonably. The present lemma additionally assumes that so does \mathcal{H}_n 's adversary. If so, it is obvious that the symbolwise length of no labmove in the Γ part can exceed $\mathfrak{G}(\mathfrak{l}) + \mathfrak{h} + 2$; and there are at most 2ϵ such labmoves. The symbolwise length of the Γ part is thus at most $2\epsilon(\mathfrak{G}(\mathfrak{l}) + \mathfrak{h} + 2)$. \square

The following lemma states that the \mathcal{H}_n 's move soon enough or never, with \mathfrak{L} acting as a “statute of limitations” function:

Lemma 5.5. *Consider any machine $\mathcal{H}_n \in \{\mathcal{H}_0, \dots, \mathcal{H}_k\}$, and any cycle (step, time) c of any play by \mathcal{H}_n . Assume that u is the spacecost of cycle $c + \mathfrak{L}(\mathfrak{l}, u)$. Further assume that the adversary of \mathcal{H}_n plays reasonably, and it does not move at any time d with $d > c$. Then \mathcal{H}_n does not move at any time d with $d > c + \mathfrak{L}(\mathfrak{l}, u)$.*

Proof. Assume the conditions of the lemma and, remembering that (not only \mathcal{H}_n 's adversary but also) \mathcal{H}_n plays reasonably, answer the following question: How many different configurations of \mathcal{H}_n — ignoring the buffer content component — are there that may emerge in the play between (including) steps c and $c + \mathfrak{L}(\mathfrak{l}, u)$? We claim that this quantity cannot

⁴ \mathcal{N} (if $n \geq 1$) or \mathcal{K} (if $n = 0$), to be more precise.

exceed $\mathcal{L}(l, u)$. Indeed, there are at most r possibilities for the state component of such a configuration. These possibilities are accounted for by the 1st of the five factors of $\mathcal{L}(l, u)$. Next, clearly there are at most $(u + 1)^g$ possibilities for the locations of the work-tape heads,⁵ which is accounted for by the 2nd factor of $\mathcal{L}(l, u)$. Next, in view of Lemma 5.4, there are at most $(v + 1) \times (l + 2) + 2e(\mathfrak{G}(l) + h + 2) + 1$ possible locations of the run-tape head, and this number is accounted for by the 3rd factor of $\mathcal{L}(l, u)$. Next, there are at most q^{gu} possibilities for the contents of the g work tapes, and this number is accounted for by the 4th factor of $\mathcal{L}(l, u)$. Finally, the run-tape content can change (be extended) at most $2e$ times, and this number is accounted for by the 5th factor of $\mathcal{L}(l, u)$. Thus, there are at most $\mathcal{L}(l, u)$ possible configurations (ignoring the buffer content component), as promised. If so, some configuration repeats itself between steps c and $c + \mathcal{L}(l, u)$, meaning that \mathcal{H}_n is in a loop which will be repeated again and again forever. Within that loop \mathcal{H}_n makes no moves, for otherwise the run-tape-content component of the configurations would keep changing (expanding). \square

5.2. The procedure SIM. We define an **organ** to be a pair $O = (\vec{\alpha}, p)$, where $\vec{\alpha}$, called the **payload** of O , is a (possibly empty) finite sequence of moves, and p , called the **scale** of O , is a positive integer.

A **signed organ** S is $-O$ or $+O$, where O is an organ. In the first case we say that S is **negative**, and in the second case we say that it is **positive**. The **payload** and the **scale** of such an S mean those of O .

A **body** is a tuple $B = (O_1, \dots, O_s)$ of organs. The number s is said to be the **size** of such a body B .

A **SIM-appropriate triple** is (A, B, n) , where $n \in \{0, \dots, k\}$, B is a nonempty body, and A is a body required to be empty if $n = 0$.

Our \mathcal{M}_k simulates the work of the machines $\mathcal{H}_0, \dots, \mathcal{H}_k$ through running the procedure SIM defined below. This procedure takes a SIM-appropriate triple (A, B, n) as an argument, and returns a pair (S, u) , where S is a signed organ and u is a natural number. We indicate this relationship by writing $\text{SIM}_n(A, B) = (S, u)$. We usually understand SIM_n as the two-argument procedure — and/or the corresponding function — resulting from fixing the third argument of SIM to n . Similarly for the later-defined SIM_n^\bullet , SIM_n^\leftarrow , SIM_n^\rightarrow .

We first take a brief informal look at SIM_n with $1 \leq n \leq k$ (SIM_0 needs to be considered separately). Assume $A = ((\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_a, p_a))$ and $B = ((\vec{\beta}_1, q_1), \dots, (\vec{\beta}_b, q_b))$. The argument (A, B) determines the scenario of the work of \mathcal{H}_n that needs to be simulated. In this scenario, the moves made by \mathcal{H}_n 's adversary in the antecedent (resp. consequent) of $F'(n - 1) \rightarrow F'(n)$ come from $\vec{\alpha}_1, \dots, \vec{\alpha}_a$ (resp. $\vec{\beta}_1, \dots, \vec{\beta}_b$). The simulation starts by “fetching” the organ $(\vec{\beta}_1, q_1)$ from B and tracing the first q_1 steps of \mathcal{H}_n in the scenario where, at the very beginning of the play, i.e., on clock cycle 0, the adversary made the moves $\vec{\beta}_1$ in the consequent of $F'(n - 1) \rightarrow F'(n)$, all at once. Which organ is fetched next depends on how things have evolved so far, namely, on whether within the above q_1 steps \mathcal{H}_n has responded by a nonempty or empty sequence $\vec{\nu}$ of moves in the consequent of $F'(n - 1) \rightarrow F'(n)$. If $\vec{\nu} \neq \langle \rangle$, then the next organ to be fetched will be the first not-yet-fetched organ of B , i.e., $(\vec{\beta}_2, q_2)$; and if $\vec{\nu} = \langle \rangle$, then the next organ to be fetched will be the first not-yet-fetched organ of A , i.e., $(\vec{\alpha}_1, p_1)$. After fetching such an organ

⁵Remember that a scanning head of an HPM can never move beyond the leftmost blank cell.

$(\vec{\delta}, r) \in \{(\vec{\beta}_2, q_2), (\vec{\alpha}_1, p_1)\}$, the simulation of \mathcal{H}_n rolls back to the point w at which \mathcal{H}_n made its last move (if there are no such moves, then $w = 0$), and continues from there for additional r steps in the scenario where, at the very beginning of the episode, i.e., at step w , \mathcal{H}_n 's imaginary adversary responded by the moves $\vec{\delta}$, all at once, in the corresponding component (consequent if $\vec{\nu} \neq \langle \rangle$ and antecedent if $\vec{\nu} = \langle \rangle$) of $F'(n-1) \rightarrow F'(n)$. As in the preceding case, what to fetch next — the leftmost not-yet-fetched organ of B or that of A — depends on whether within the above r steps (i.e., steps w through $w+r$) \mathcal{H}_n responds by a nonempty or an empty sequence of moves in the consequent of $F'(n-1) \rightarrow F'(n)$. And similarly for the subsequent steps: whenever \mathcal{H}_n responds to the last series $\vec{\beta}_i$ (resp. $\vec{\alpha}_i$) of the imaginary adversary's moves with a nonempty sequence $\vec{\nu}$ of moves in the consequent of $F'(n-1) \rightarrow F'(n)$ within q_i (resp. p_i) steps, the next organ $(\vec{\delta}, r)$ to be fetched will be the first not-yet-fetched organ of B ; otherwise such a $(\vec{\delta}, r)$ will be the first not-yet-fetched organ of A . In either case, the simulation of \mathcal{H}_n rolls back to the point w at which \mathcal{H}_n made its last move, and continues from there for additional r steps in the scenario where, at step w , \mathcal{H}_n 's imaginary adversary responded by the moves $\vec{\delta}$ in the corresponding component (consequent if $\nu \neq \langle \rangle$ and antecedent if $\nu = \langle \rangle$) of the game. The overall procedure ends when it tries to fetch the next not-yet-fetched organ of A (resp. B) but finds that there are no such organs remaining. Then the S part of the output (S, u) of $\text{SIM}_n(A, B)$ is stipulated to be $-(\vec{\sigma}, r)$ (resp. $+(\vec{\sigma}, r)$), where $\vec{\sigma}$ is the sequence of moves made by \mathcal{H}_n in the antecedent (resp. consequent) of $F'(n-1) \rightarrow F'(n)$ since the last organ of A (resp. B) was fetched, and r is the scale of that organ. As for the u part of the output (S, u) , in either case it is simply the maximum number of non-blank cells on any (any *one*) work tape of \mathcal{H}_n at the end of the simulated episode.

The case of $\text{SIM}_0(\langle \rangle, B)$ is similar but simpler. In fact, $\text{SIM}_0(\langle \rangle, B)$ is a special case of $\text{SIM}_n(A, B)$ if we think of $F'(0)$ as the implication $F'(-1) \rightarrow F'(0)$ with the dummy antecedent $F'(-1) = \top$.

In order to be able to define SIM_0 or SIM_n ($1 \leq n \leq k$) more formally, we need a couple of notational conventions.

When $\vec{\alpha} = \langle \alpha_1, \dots, \alpha_s \rangle$ is a sequence of moves, ω is a string over the keyboard alphabet (such as, say, “0.”, “1.” or the empty string) and φ is one of the players \top or \perp , we shall write $\varphi\omega\vec{\alpha}$ for the run $\langle \varphi\omega\alpha_1, \dots, \varphi\omega\alpha_s \rangle$.

Next, when W is a configuration of \mathcal{H}_n ($0 \leq n \leq k$) and Θ is a finite sequence of lab-moves, we shall write $W \oplus \Theta$ to denote the configuration that results from W by appending Θ to the (description of the) run-tape content component of W .

In precise terms, this is how the **procedure** $\text{SIM}_0(\langle \rangle, B)$ works. It creates two integer-holding variables b and u , with b initialized to 1 and u to 0. It further creates a variable $\vec{\nu}$ to hold move sequences, initialized to the empty sequence $\langle \rangle$. It further creates a configuration-holding variable W , initialized to the start configuration of \mathcal{H}_0 where the run tape is empty (and, of course, so are the work tapes and the buffer). Finally, it creates two signed-organ-holding variables S and R , with S having no initial value and R initialized to $+O$, where O is the first organ of B (remember that B is required to be nonempty).⁶ After this initialization step, the procedure goes into the following loop LOOP_0 . Each iteration of

⁶The presence of the variable S may seem redundant at this point, as $\text{SIM}_0(\langle \rangle, B)$ (and likewise $\text{SIM}_n(A, B)$ with $n \geq 1$) could be defined in a simpler way without it. The reason why we want to have S will become clear in Section 5.5. Similarly, in the present case we could have done without the variable R as well — it merely serves the purpose of “synchronizing” the cases of $n = 0$ and $n \geq 1$.

the latter simulates a certain number of steps of H_0 starting from a certain configuration (namely, the then-current value of W) in the scenario where \mathcal{H}_0 's imaginary adversary makes no moves other than those already present in configuration W (i.e., already made by the time W was reached).

Procedure LOOP_0 : Let $+(\vec{\omega}, p)$ be the value of R (R never takes negative values when $n = 0$). Change the value of W to $W \oplus \perp \vec{\omega}$. Then simulate/trace p steps of \mathcal{H}_0 starting from configuration W . While performing this simulation, keep track of the maximum number of non-blank cells on any (any one) of the work tapes of \mathcal{H}_0 , and increment u to that number every time the latter exceeds u . Also, every time \mathcal{H}_0 makes a move μ , update $\vec{\nu}$ by adding μ at the end of it, and, additionally, update W to the configuration in which such a move μ was made. Once the simulation of p steps is complete, do the following. If $\vec{\nu}$ is empty, set the value of S to $-(\vec{\nu}, p)$ and return (S, u) . Suppose now $\vec{\nu}$ is nonempty. In this case set the value of S to $+(\vec{\nu}, p)$. Then, if b equals the size of B , return (S, u) . Otherwise, increment b to $b + 1$, set R to the b th organ of B prefixed with “+”, and repeat LOOP_0 .

Next, this is how the **procedure** $\text{SIM}_n(A, B)$ exactly works when $n \geq 1$. It creates three integer-holding variables a, b, u , with b initialized to 1 and a, u to 0.⁷ It further creates two move-sequence-holding variables $\vec{\psi}$ and $\vec{\nu}$, both initialized to the empty sequence $\langle \rangle$. It further creates a configuration-holding variable W , initialized to the start configuration of \mathcal{H}_n where the run tape is empty. Finally, it creates two signed-organ-holding variables S and R , with S having no initial value and R initialized to $+O$, where O is the first organ of B . After this initialization step, the procedure goes into the following loop LOOP_n . As before, each iteration of the latter simulates a certain number of steps of H_n starting from a certain configuration (namely, W) in the scenario where the imaginary adversary makes no new moves.

Procedure LOOP_n : Let $+(\vec{\omega}, p)$ (resp. $- (\vec{\omega}, p)$) be the value of R . Change the value of W to $W \oplus \perp 1. \vec{\omega}$ (resp. $W \oplus \perp 0. \vec{\omega}$). Then simulate/trace p steps of \mathcal{H}_n starting from configuration W . While performing this simulation, keep track of the maximum number of non-blank cells on any of the work tapes of \mathcal{H}_n , and increment u to that number every time the latter exceeds u . Also, every time \mathcal{H}_n makes a move μ in the antecedent (resp. consequent) of the game, update $\vec{\psi}$ (resp. $\vec{\nu}$) by adding μ at the end of it, and, additionally, update W to the configuration in which such a move μ was made. Once the simulation of p steps is complete, do the following.

- If $\vec{\nu}$ is nonempty, set the value of S to $+(\vec{\nu}, p)$. Then, if b equals the size of B , return (S, u) ; otherwise, increment b to $b + 1$, set R to the b th organ of B prefixed with “+”, reset $\vec{\nu}$ to $\langle \rangle$, and repeat LOOP_n .
- If $\vec{\nu}$ is empty, set the value of S to $-(\vec{\psi}, p)$. Then, if a equals the size of A , return (S, u) . Otherwise, increment a to $a + 1$, set R to the a th organ of A prefixed with “-”, reset $\vec{\psi}$ to $\langle \rangle$, and repeat LOOP_n .

For a SIM-appropriate triple (A, B, n) , we shall write

$$\text{SIM}_n^\bullet(A, B)$$

to refer to the signed organ S such that $\text{SIM}_n(A, B) = (S, u)$ for some (whatever) u .

⁷Intuitively, b keeps track of how many organs of B have been fetched so far, and a does the same for A .

Later, we may write $\text{SIM}_n(A, B)$ to refer to either the *procedure* SIM_n applied to arguments A and B , or to the *output* (S, u) of that procedure on the same arguments. It will be usually clear from the context which of these two is meant. The same applies to $\text{SIM}_n^\bullet(A, B)$ which, seen as a procedure, runs exactly like $\text{SIM}_n(A, B)$, and only differs from the latter in that it just outputs S rather than (S, u) .

Consider any two bodies $B = (O_1, \dots, O_t)$ and $B' = (O'_1, \dots, O'_{t'})$. We say that B' is an **extension** of B , and that B is a **restriction** of B' , iff $t \leq t'$ and $O_1 = O'_1, \dots, O_t = O'_{t'}$. As expected, “**proper extension**” means “extension but not restriction”. Similarly for “**proper restriction**”.

Lemma 5.6. *Consider any SIM-appropriate triple (A, B, n) .*

1. *If $\text{SIM}_n^\bullet(A, B)$ is negative, then, for every extension B' of B , $\text{SIM}_n(A, B') = \text{SIM}_n(A, B)$.*
2. *If $\text{SIM}_n^\bullet(A, B)$ is positive and $n \neq 0$, then, for every extension A' of A , $\text{SIM}_n(A', B) = \text{SIM}_n(A, B)$.*
3. *Whenever $\text{SIM}_n^\bullet(A, B)$ is positive, the size of B does not exceed ϵ_\top .*

Proof. Clauses 1-2 can be verified through a straightforward analysis of the work of SIM_n . For clause 3, assume $\text{SIM}_n(A, B) = +(\vec{\omega}, p)$, and let s be the size of B . Observe that, in the process of computing $\text{SIM}_n(A, B)$, the payloads of all positive values that the variable S ever takes, including its last value $+(\vec{\omega}, p)$, are nonempty. All such payloads consist of moves made by \mathcal{H}_n in the consequent of $F'(n-1) \rightarrow F'(n)$. From the work of SIM_n we can see that altogether there are s positive values taken by S . Now, remembering our assumption that \mathcal{H}_n plays quasilegally, implying that it does not make more than ϵ_\top moves in the consequent of $F'(n-1) \rightarrow F'(n)$, it is clear that s cannot exceed ϵ_\top . \square

By a **saturated** triple we shall mean a SIM-appropriate triple (A, B, n) such that:

1. If $\text{SIM}_n^\bullet(A, B)$ is negative, then, for every nonempty proper restriction B' of B , $\text{SIM}_n^\bullet(A, B')$ is positive.
2. If $\text{SIM}_n^\bullet(A, B)$ is positive, then, for every proper restriction A' of A , $\text{SIM}_n^\bullet(A', B)$ is negative.

For a body $B = (O_1, \dots, O_s)$, we will write B^{odd} (resp. B^{even}) to denote the body (O_1, O_3, \dots) (resp. (O_2, O_4, \dots)) obtained from B by deleting each O_i with an even (resp. odd) i .

Definition 5.7. Consider a saturated triple (A, B, n) . Let $A = (A_1, \dots, A_a)$ and $B = (B_1, \dots, B_b)$. Further let $-P_1, \dots, -P_p$ be the (sequence of the) negative values that the variable S of the procedure SIM_n goes through when computing $\text{SIM}_n(A, B)$, and let $+Q_1, \dots, +Q_q$ be the (sequence of the) positive values that S goes through. Observe that $a \leq p \leq a+1$ and $q \leq b \leq q+1$.

1. We define $\text{SIM}_n^\leftarrow(A, B)$ as the body $(P_1, A_1, P_2, A_2, \dots)$ — that is, the (unique) body C such that $C^{\text{odd}} = (P_1, \dots, P_p)$ and $C^{\text{even}} = (A_1, \dots, A_a)$.
2. We define $\text{SIM}_n^\rightarrow(A, B)$ as the body $(B_1, Q_1, B_2, Q_2, \dots)$ — that is, the (unique) body C such that $C^{\text{odd}} = (B_1, \dots, B_b)$ and $C^{\text{even}} = (Q_1, \dots, Q_q)$.

Let $B = ((\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_s, p_s))$ be a body. We define \overline{B} as the run $\langle \perp \vec{\alpha}_1, \top \vec{\alpha}_2, \dots \rangle$ obtained from $\langle \vec{\alpha}_1, \dots, \vec{\alpha}_s \rangle$ by replacing each $\vec{\alpha}_i$ ($1 \leq i \leq s$) with $\perp \vec{\alpha}_i$ if i is odd, and with $\top \vec{\alpha}_i$ if i is even.

Some more notation and terminology. When Γ and Δ are runs, we write $\Gamma \preceq \Delta$ to mean that Γ is a (not necessarily proper) initial segment of Δ . Next, as always in CoL, $\neg\Gamma$ means the result of changing in Γ each label \top to \perp and vice versa. $\Gamma^0.$ means the result of deleting from Γ all moves (together with their labels, of course) except those of the form $0.\alpha$, and then further deleting the prefix “0.” in the remaining moves. Similarly for $\Gamma^1.$ Intuitively, when Γ is a play of a parallel disjunction $G_0 \vee G_1$ or conjunction $G_0 \wedge G_1$ of games, $\Gamma^0.$ (resp. $\Gamma^1.$) is the play that has taken place — according to the scenario of Γ — in the G_0 (resp. G_1) component.

Lemma 5.8. *Consider any saturated SIM-appropriate triple (A, B, n) . Let $\text{SIM}_n(A, B) = (\pm(\vec{\omega}, v), u)$, where $\pm \in \{+, -\}$.*

1. *The case of $n = 0$ (and hence $A = ()$):*
 - (a) *There is a run Υ generated by \mathcal{H}_0 such that $\overline{\text{SIM}_0^{\rightarrow}(((), B))} \preceq \Upsilon$.*
 - (b) *Furthermore, if $\overline{\text{SIM}_0^{\rightarrow}(((), B))}$ is a reasonable run of $F'(0)$ and $v \geq \mathfrak{L}(l, u)$, then, for such an Υ , we simply have $\overline{\text{SIM}_0^{\rightarrow}(((), B))} = \Upsilon$.*
2. *The case of $1 \leq n \leq k$:*
 - (a) *There is a run Υ generated by \mathcal{H}_n such that $\overline{\text{SIM}_n^{\rightarrow}(A, B)} \preceq \Upsilon^1.$ and $\neg\overline{\text{SIM}_n^{\leftarrow}(A, B)} \preceq \Upsilon^0.$*
 - (b) *Furthermore, if $\text{SIM}_n^{\bullet}(A, B)$ is negative, $\overline{\text{SIM}_n^{\rightarrow}(A, B)}$ is a reasonable run of $F'(n)$, $\overline{\text{SIM}_n^{\leftarrow}(A, B)}$ is a reasonable run of $F'(n-1)$ and $v \geq \mathfrak{L}(l, u)$, then, for such an Υ , we simply have $\overline{\text{SIM}_n^{\rightarrow}(A, B)} = \Upsilon^1.$ and $\neg\overline{\text{SIM}_n^{\leftarrow}(A, B)} = \Upsilon^0.$*

Proof. Assume the conditions of the lemma. Let $A = ((\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_a, p_a))$ and $B = ((\vec{\beta}_1, q_1), \dots, (\vec{\beta}_b, q_b))$. Further let $-(\vec{\gamma}_1, r_1), \dots, -(\vec{\gamma}_c, r_c)$ be the negative values that the variable S of the procedure SIM_n goes through when computing $\text{SIM}_n(A, B)$, and let $+(\vec{\delta}_1, s_1), \dots, +(\vec{\delta}_d, s_d)$ be the positive values that S goes through.

1. Assume $n = 0$, and thus $A = ()$, i.e., $a = 0$. Analyzing the definitions of SIM_0 and $\text{SIM}_0^{\rightarrow}$ and taking into account that $(((), B, 0)$ is saturated, we see that, what the procedure $\text{SIM}_0(((), B))$ does is that it simulates the first t steps of a certain computation branch C of \mathcal{H}_0 for a certain t with $v = q_b \leq t \leq q_1 + \dots + q_b$, and the position spelled on \mathcal{H}_0 's imaginary run tape by the end of this episode (without counting the initial moves $\perp \vec{c}$ — see Remark 5.2) is nothing but $\overline{\text{SIM}_0^{\rightarrow}(((), B))}$. Let Υ be the run spelled by C . Then Υ satisfies the promise of clause 1(a) of the lemma. For clause 1(b), additionally assume that $\overline{\text{SIM}_0^{\rightarrow}(((), B))}$ is a reasonable run of $F'(0)$ and $v \geq \mathfrak{L}(l, u)$. We may assume that, in the above branch C , \mathcal{H}_0 's adversary makes no moves after (beginning from) time $t - v$. Then, by Lemma 5.5, \mathcal{H}_0 makes no moves after (beginning from) time t . Thus, the run Υ contains no labmoves in addition to those that are in $\overline{\text{SIM}_n^{\rightarrow}(A, B)}$, meaning that $\overline{\text{SIM}_n^{\rightarrow}(A, B)} = \Upsilon$, as desired.
2. Assume $1 \leq n \leq k$. Again, taking into account that (A, B, n) is saturated, we can see that, what the procedure $\text{SIM}_n(A, B)$ does is that it simulates the first t steps of a certain computation branch C of \mathcal{H}_n for a certain number t with $v \leq t \leq p_1 + \dots + p_a + q_1 + \dots + q_b$. Note that here v is either p_a or q_b . Let Φ be the position spelled on \mathcal{H}_n 's imaginary run tape by the end of this episode. It is not hard to see that $\Phi^1. = \overline{\text{SIM}_n^{\rightarrow}(A, B)}$. Further, if $\text{SIM}_n^{\bullet}(A, B)$ is negative, then we also have $\Phi^0. = \neg\overline{\text{SIM}_n^{\leftarrow}(A, B)}$. Otherwise, if $\text{SIM}_n^{\bullet}(A, B)$ is positive, $\Phi^0.$ is a (not necessarily proper) extension of $\neg\overline{\text{SIM}_n^{\leftarrow}(A, B)}$ through some

Υ -labeled moves. Let Υ be the run spelled by C . Then, in view of the observations that we have just made, Υ satisfies the promise of clause 2(a) of the lemma.

For clause 2(b), additionally assume that $\text{SIM}_n^\bullet(A, B)$ is negative, $\overline{\text{SIM}_n^\rightarrow(A, B)}$ is a reasonable run of $F'(n)$, $\neg\overline{\text{SIM}_n^\leftarrow(A, B)}$ is a reasonable run of $F'(n - 1)$, and $v \geq \mathfrak{L}(\mathfrak{l}, u)$. As observed in the preceding paragraph, on our present assumption of $\text{SIM}_n^\bullet(A, B)$'s being negative, we have $\Phi^0 = \neg\overline{\text{SIM}_n^\leftarrow(A, B)}$ and $\Phi^1 = \overline{\text{SIM}_n^\rightarrow(A, B)}$. We may assume that, in the above branch C , \mathcal{H}_n 's adversary makes no moves after (beginning from) time $t - v$. Then, by Lemma 5.5, \mathcal{H}_n makes no moves after (beginning from) time t . Thus, the run Υ contains no labmoves in addition to those that are (after removing the prefixes “0.” and “1.”) in $\neg\overline{\text{SIM}_n^\leftarrow(A, B)}$ and $\overline{\text{SIM}_n^\rightarrow(A, B)}$, meaning that $\neg\overline{\text{SIM}_n^\leftarrow(A, B)} = \Upsilon^0$ and $\overline{\text{SIM}_n^\rightarrow(A, B)} = \Upsilon^1$, as desired. \square

5.3. Aggregations. By an **entry** we shall mean a pair $E = [n, B]$, where n , called the **index** of E , is an element of $\{0, \dots, k\}$, and B , called the **body** of E , is a body. The **size** of an entry E should be understood as the size of its body. By saying that an entry is **n -indexed** we shall mean that n is the index of that entry.

An **aggregation** is a nonempty finite sequence \vec{E} of entries such that:

- (i) The last entry of \vec{E} is k -indexed, and its body is odd-size. We call it the **master entry** of \vec{E} , and call all other entries (if there are any) **common entries**.
- (ii) The indices of the entries of \vec{E} are strictly increasing. That is, the index of any given entry is strictly smaller than the index of any entries to the right of it.
- (iii) Each even-size entry (if there are such entries) is to the left of each odd-size entry.
- (iv) The sizes of the even-size entries are strictly decreasing. That is, the size of any even-size entry is strictly smaller than the size of any (even-size) entry to the left of it.
- (v) The sizes of the odd-size common entries are strictly increasing. That is, the size of any odd-size common entry is strictly smaller than the size of any (odd-size) common entry to the right of it.
- (vi) There are no entries of size 0.

The **central triple** of an aggregation \vec{E} is (L, R, n) , where:

- (1) n is the index of the leftmost odd-size entry of \vec{E} .
- (2) R is the body of the above n -indexed entry of \vec{E} .
- (3) If \vec{E} does not have an entry whose index is $n - 1$,⁸ then L is the empty body $()$. Otherwise, L is the body of the $(n - 1)$ -indexed entry of \vec{E} .

Consider any aggregation \vec{E} . The **master body** of \vec{E} is the body of the master entry of \vec{E} ; the **master organ** of \vec{E} is the last organ of the master body of \vec{E} ; and the **master payload** (resp. **master scale**) of \vec{E} is the payload (resp. scale) of the master organ of \vec{E} .

⁸This condition is always automatically satisfied when $n = 0$.

5.4. The procedure MAIN. We are now ready to finalize our description of the work of \mathcal{M}_k . This is a machine that creates an aggregation-holding variable \vec{E} and an integer-holding variable U , with \vec{E} initialized to the aggregation $\langle [k, ((\langle \rangle, 1))] \rangle^9$ and U initialized to 0. After this initialization step, \mathcal{M}_k goes into the below-described loop MAIN. As already noted, our description of \mathcal{M} and hence of MAIN and our subsequent analysis of its work relies on the Clean Environment Assumption.

Terminology: In our description of MAIN, whenever we say REPEAT, it is to be understood as repeating (going to) MAIN without changing the values of U and \vec{E} . On the other hand, whenever we say RESTART, it is to be understood as resetting U to 0, modifying \vec{E} by deleting all common entries in it (but leaving the master entry unchanged), and then repeating MAIN. Finally, when we say “*Environment has made a new move*”, we mean that the run tape of \mathcal{M}_k contains a $(q + 1)$ th \perp -labeled move (which we refer to as “*the new move*”), where q is the total number of moves in (all moves in the payloads of the organs of) B^{odd} , where B is the master body of \vec{E} .

Procedure MAIN. Let (L, R, n) be the central triple of \vec{E} . Start running the procedure SIM_n on (L^{even}, R^{odd}) while, in parallel, at some constant rate, polling the run tape to see if Environment has made a new move.¹⁰ Then act depending on which of the following two cases is “the case”:

Case 1: Before SIM_n terminates, one of the calls of the polling routine detects a new move $1.\theta$ (i.e., the move θ in the consequent of $k \leq \mathbf{b}|\vec{d}| \rightarrow F'(k)$) by Environment. Let θ' be the $F'(k)$ -prudentization of θ . Modify \vec{E} by adding θ' to its master payload, and resetting the master scale to 1. Then RESTART.

Case 2: SIM_n terminates without any of the calls of the polling routine meanwhile detecting a new move by Environment. Let (S, u) be the value computed/returned by $\text{SIM}_n(L^{even}, R^{odd})$. Update U to $\max(u, U)$. Then act depending on whether S is positive or negative.

Subcase 2.1: S is positive, namely, $S = +(\vec{\omega}, s)$. Let B be the body of the n -indexed entry of \vec{E} . Act depending on whether $n < k$ or not.

Subsubcase 2.1.1: $n < k$. Update \vec{E} by adding $(\vec{\omega}, s)$ as a new organ to B . Further modify \vec{E} by deleting all $(< n)$ -indexed entries whose size does not exceed that of the n -indexed entry, if such entries exist. Then REPEAT.

Subsubcase 2.1.2: $n = k$. Update \vec{E} by adding $(\vec{\omega}, s)$ and $((\langle \rangle, s)$ as two new organs to B . Then make the moves $\vec{\omega}$ in the consequent of (the real play of) $k \leq \mathbf{b}|\vec{d}| \rightarrow F'(k)$. Finally, REPEAT.

Subcase 2.2: S is negative, namely, $S = -(\vec{\omega}, s)$. Act depending on whether $n > 0$ or not.

Subsubcase 2.2.1: $n > 0$. Then, if \vec{E} has an $(n - 1)$ -indexed entry E , modify \vec{E} by adding $(\vec{\omega}, s)$ as a new organ to the body of E ; otherwise modify \vec{E} by inserting into it the entry $E = [n - 1, ((\vec{\omega}, s))]$ immediately on the left of the n -indexed entry. In either case, further modify \vec{E} by deleting all $\geq n$ -indexed common entries whose size does not exceed that of the $(n - 1)$ -indexed entry, if such entries exist. After that REPEAT.

⁹I.e., the single-entry aggregation where the master body is of size 1, the master payload is empty and the master scale is 1.

¹⁰Clarifying: the polling routine is called, say, after every 1000 steps of performing SIM_n ; such a call — which, itself, may take more than a constant amount of time — interrupts SIM_n , saves its state, checks the run tape to see if a new move is made and, if not, returns control back to the caller.

Subsubcase 2.2.2: $n = 0$. Let v be the master scale of \vec{E} . Act depending on whether $v < \mathcal{L}(\mathfrak{l}, U)$ or not.¹¹

Subsubsubcase 2.2.2.1: $v < \mathcal{L}(\mathfrak{l}, U)$. Then modify \vec{E} by doubling its master scale v , and RESTART.

Subsubsubcase 2.2.2.2: $v \geq \mathcal{L}(\mathfrak{l}, U)$. Keep polling the run tape of \mathcal{M}_k to see if Environment has made a new move $1.\theta$. If and when such a move is detected, modify \vec{E} by adding the $F'(k)$ -prudentization θ' of θ to the master payload of \vec{E} , and resetting the master scale to 1. Then RESTART.

5.5. \mathcal{M} is a solution of the target game. In this subsection we want to verify that \mathcal{M}_k indeed wins $k \leq \mathfrak{b}|\vec{d}| \rightarrow F'(k)$ and hence \mathcal{M} wins $x \leq \mathfrak{b}|\vec{s}| \rightarrow F(x, \vec{v})$. For this purpose, when analyzing the work and behavior of \mathcal{M}_k , we will implicitly have some arbitrary but fixed computation branch (“play”) of \mathcal{M}_k in mind. So, for instance, when we say “the i th iteration of MAIN”, it should be understood in the context of that branch.

Notation 5.9. *In what follows, \mathbb{I} will stand for the set of positive integers i such that MAIN is iterated at least i times. Also, for each $i \in \mathbb{I}$, \vec{E}_i will stand for the value of the aggregation/variable \vec{E} at the beginning of the i th iteration of MAIN.*

Lemma 5.10. *For any $i \in \mathbb{I}$ and any entry E of \vec{E}_i , the size of E does not exceed $2\epsilon_{\top} + 1$.*

Proof. For a contradiction, assume $i \in \mathbb{I}$, and \vec{E}_i has an entry of size greater than $2\epsilon_{\top} + 1$. Let n be the index of such an entry.

First, consider the case $n < k$. Let $j \leq i$ be the smallest number in \mathbb{I} such that \vec{E}_j has a $(2\epsilon_{\top} + 2)$ -size, n -indexed entry $[n, (O_1, \dots, O_{2\epsilon_{\top}+2})]$ — it is not hard to see that such a j exists, and $j > 1$ because \vec{E}_1 has no common entries. The only way the above entry could have emerged in \vec{E}_j is that \vec{E}_{j-1} contained the entry $[n, (O_1, \dots, O_{2\epsilon_{\top}+1})]$, and its body “grew” into $(O_1, \dots, O_{2\epsilon_{\top}+2})$ on the transition from \vec{E}_{j-1} to \vec{E}_j according to the prescriptions of Subsubcase 2.1.1 of the description of MAIN. This in turn means that the central triple of \vec{E}_{j-1} was $(A, (O_1, \dots, O_{2\epsilon_{\top}+1}), n)$ for a certain body A , and $\text{SIM}_n^{\bullet}(A^{\text{even}}, (O_1, \dots, O_{2\epsilon_{\top}+1})^{\text{odd}}) = +O_{2\epsilon_{\top}+2}$. This, however, is impossible by clause 3 of Lemma 5.6, because the size of $(O_1, \dots, O_{2\epsilon_{\top}+1})^{\text{odd}}$ is $\epsilon_{\top} + 1$, exceeding ϵ_{\top} .

The case $n = k$ is similar, only with “ k ” instead of “ n ”, and “ $2\epsilon_{\top} + 3$ ” instead of “ $2\epsilon_{\top} + 2$ ”. \square

Lemma 5.11. *There is a bound $\mathfrak{z}(w) \in \mathcal{R}_{\text{time}}$ such that the cardinality of \mathbb{I} does not exceed $\mathfrak{z}(\mathfrak{l})$.*

Proof. In this proof we will be using \mathfrak{d} as an abbreviation of $2\epsilon_{\top} + 1$. Whenever we say “ \vec{E} always (never, etc.) so and so”, it is to be understood as that, throughout the work of MAIN, the value of the variable \vec{E} always (never, etc.) so and so. Similarly for U . “Case”, “Subcase”, etc. mean those of the description of MAIN.

According to Lemma 5.10, we have:

$$\text{The size of no entry of } \vec{E} \text{ ever exceeds } \mathfrak{d}. \quad (5.3)$$

¹¹For \mathcal{L} , remember clause 4 of Notation 5.3.

Our next claim is the following:

$$\begin{aligned} & \text{The number of moves in the payload of no organ} \\ & \text{of the master body of } \vec{E} \text{ ever exceeds } \max(\epsilon_T, \epsilon_\perp). \end{aligned} \quad (5.4)$$

Indeed, let (O_1, \dots, O_a) be the master body of \vec{E} at a given stage of the work of MAIN, and consider any organ $O_i = (\vec{\alpha}, s)$ ($1 \leq i \leq a$) of this body. From an analysis of the work of MAIN we can see that, if i is odd, then $\vec{\alpha}$ are moves made by Environment within the $F'(k)$ component in the real play. Therefore, in view of the Clean Environment Assumption, the number of such moves is at most ϵ_\perp . If i is even, then $\vec{\alpha}$ are moves made by \mathcal{H}_k in a certain play simulated through SIM_k . As in the preceding case, the number of such moves cannot exceed ϵ_T because, as we have agreed, \mathcal{H}_k plays quasilegally.

Taking into account that each \mathcal{H}_n (\mathcal{N} and \mathcal{K} , that is) plays unconditionally prudently and that Environment's moves in $F'(k)$ are also prudentized when copied by MAIN according to the prescriptions of Case 1 or Subsubsubcase 2.2.2.2 (and that every move that emerges in \vec{E} originates either from Environment or from one of \mathcal{H}_i), one can see that the run tape of any simulated machine does not contain moves whose magnitude is greater than $\mathfrak{G}(l)$ where, as we remember, \mathfrak{G} is the superaggregate bound of $F(x, \vec{v})$. Since the \mathcal{H}_n s (\mathcal{N} and \mathcal{K} , to be more precise) play in unconditional space \mathfrak{s} , we then find that the value of the variable U of MAIN never exceeds $\mathfrak{s}(\mathfrak{G}(l))$. Thus, the maximum value of $\mathfrak{L}(l, U)$ is bounded by $\mathfrak{L}(l, \mathfrak{s}(\mathfrak{G}(l)))$. The master scale v of \vec{E} increases — namely, doubles — only according to the prescriptions of Subsubsubcase 2.2.2.1, and such an increase happens only when v is smaller than $\mathfrak{L}(l, U)$. For this reason, we have:

$$\text{The master scale of } \vec{E} \text{ is always smaller than } 2\mathfrak{L}(l, \mathfrak{s}(\mathfrak{G}(l))). \quad (5.5)$$

Let f be the unarification of the bound $b \in \mathcal{R}_{time}$ from (5.2). Note that, since $k \leq b|\vec{d}|$, we have $k \leq f(l)$.

Let $\mathfrak{K}(w)$ be the unary function defined by

$$\mathfrak{K}(w) = \max(|\mathfrak{L}(w, \mathfrak{s}(\mathfrak{G}(w)))|, f(w), \mathfrak{d}, \epsilon_\perp) + 1, \quad (5.6)$$

and let \mathfrak{k} be an abbreviation of $\mathfrak{K}(l)$.

With each element i of \mathbb{I} we now associate an integer $\text{Rank}(i)$ defined as follows:

$$\text{Rank}(i) = c_0 \times \mathfrak{k}^0 + c_1 \times \mathfrak{k}^1 + c_2 \times \mathfrak{k}^2 + \dots + c_{\mathfrak{d}} \times \mathfrak{k}^{\mathfrak{d}} + c_{\mathfrak{d}+1} \times \mathfrak{k}^{\mathfrak{d}+1} + c_{\mathfrak{d}+2} \times \mathfrak{k}^{\mathfrak{d}+2} + c_{\mathfrak{d}+3} \times \mathfrak{k}^{\mathfrak{d}+3},$$

where:

- $c_0 = 0$. Take a note of the fact that $c_0 < \mathfrak{k}$.
- For each even $j \in \{1, \dots, \mathfrak{d}\}$: If \vec{E}_i contains a common entry of size j , then c_j is $n + 1$, where n is the index of that entry; otherwise $c_j = 0$. Thus, c_j cannot exceed k and, since $k \leq f(l)$, from (5.6) we can see that $c_j < \mathfrak{k}$.
- For each odd $j \in \{1, \dots, \mathfrak{d}\}$: If \vec{E}_i contains a common entry of size j , then c_j is $k - n$, where n is the index of that entry; otherwise $c_j = 0$. Again, we have $c_j < \mathfrak{k}$.
- $c_{\mathfrak{d}+1}$ is $|v|$, where v is the master scale of \vec{E}_i . In view of (5.5), we find $c_{\mathfrak{d}+1} < \mathfrak{k}$.
- $c_{\mathfrak{d}+2}$ is the number of moves in the master payload of \vec{E}_i . From (5.4), we see that $c_{\mathfrak{d}+2} < \mathfrak{k}$.
- $c_{\mathfrak{d}+3}$ is the size of the master body of \vec{E}_i . The fact (5.3) guarantees that $c_{\mathfrak{d}+3} < \mathfrak{k}$.

As we have observed in each case above, all of the factors $c_0, c_1, \dots, c_{\mathfrak{d}+3}$ from $\text{Rank}(i)$ are smaller than \mathfrak{k} . This allows us to think of $\text{Rank}(i)$ as a \mathfrak{k} -ary numeral of length $\mathfrak{d} + 4$, with the least significant digit being c_0 and the most significant digit being $c_{\mathfrak{d}+3}$.

With some analysis of the work of MAIN, which we here leave to the reader, one can see that

$$\text{For any } i \text{ with } (i+1) \in \mathbb{I}, \quad \text{Rank}(i) < \text{Rank}(i+1). \quad (5.7)$$

But, by our observation $c_0, c_1, \dots, c_{\mathfrak{d}+3} < \mathfrak{k}$, no rank can exceed the (generously taken) number

$$(\mathfrak{k}-1) \times \mathfrak{k}^0 + (\mathfrak{k}-1) \times \mathfrak{k}^1 + (\mathfrak{k}-1) \times \mathfrak{k}^2 + \dots + (\mathfrak{k}-1) \times \mathfrak{k}^{\mathfrak{d}+3},$$

i.e., the number $\mathfrak{M}(\mathfrak{l})$, where $\mathfrak{M}(w)$ is the unary function

$$(\mathfrak{K}(w)-1) \times (\mathfrak{K}(w))^0 + (\mathfrak{K}(w)-1) \times (\mathfrak{K}(w))^1 + (\mathfrak{K}(w)-1) \times (\mathfrak{K}(w))^2 + \dots + (\mathfrak{K}(w)-1) \times (\mathfrak{K}(w))^{\mathfrak{d}+3}.$$

Thus:

$$\text{For any } i \in \mathbb{I}, \quad \text{Rank}(i) \leq \mathfrak{M}(\mathfrak{l}). \quad (5.8)$$

By the conditions of \mathcal{R} -Induction, $F(x, \vec{v})$ is \mathcal{R}_{space} -bounded. Hence, by Lemma 4.1, $\mathfrak{G}(w) \preceq \mathcal{R}_{space}$. This, by condition 4 of Definition 2.2 of [5], implies $\mathfrak{s}(\mathfrak{G}(w)) \preceq \mathcal{R}_{space}$. The two facts $\mathfrak{G}(w) \preceq \mathcal{R}_{space}$ and $\mathfrak{s}(\mathfrak{G}(w)) \preceq \mathcal{R}_{space}$, by condition 5 of Definition 2.2 of [5], further yield $\mathfrak{G}(w) \preceq \mathcal{R}_{time}$ and $\mathfrak{s}(\mathfrak{G}(w)) \preceq \mathcal{R}_{time}$. Looking back at our definition of \mathfrak{L} in Notation 5.3(4), we see that

$$|\mathfrak{L}(w, u)| = O(|w| + |\mathfrak{G}(w)| + u) \quad (5.9)$$

and thus $|\mathfrak{L}(w, \mathfrak{s}(\mathfrak{G}(w)))| = O(|w| + |\mathfrak{G}(w)| + \mathfrak{s}(\mathfrak{G}(w)))$. This, together with $\mathfrak{G}(w) \preceq \mathcal{R}_{time}$ and $\mathfrak{s}(\mathfrak{G}(w)) \preceq \mathcal{R}_{time}$, by the linear closure of \mathcal{R}_{time} , implies

$$|\mathfrak{L}(w, \mathfrak{s}(\mathfrak{G}(w)))| \preceq \mathcal{R}_{time}. \quad (5.10)$$

Since f is the unarification of $\mathfrak{b} \in \mathcal{R}_{time}$, we obviously have $f \preceq \mathcal{R}_{time}$. This, together with (5.10), (5.6) and the fact of \mathcal{R}_{time} 's being linearly closed, implies that $\mathfrak{K} \preceq \mathcal{R}_{time}$. The latter, in turn, in view of \mathcal{R}_{time} 's being polynomially closed, implies that $\mathfrak{M} \preceq \mathcal{R}_{time}$. So, there is a bound $\mathfrak{z}(w)$ in \mathcal{R}_{time} with $\mathfrak{M}(w) \preceq \mathfrak{z}(w)$ and hence $\mathfrak{M}(\mathfrak{l}) \leq \mathfrak{z}(\mathfrak{l})$. In view of (5.8), no rank can thus ever exceed $\mathfrak{z}(\mathfrak{l})$. But, by (5.7), different elements of \mathbb{I} have different ranks. Hence, the cardinality of \mathbb{I} does not exceed $\mathfrak{z}(\mathfrak{l})$ either, as desired. \square

For a number $h \in \mathbb{I}$, we define the set \mathbb{I}^h by

$$\mathbb{I}^h = \{i \mid i \in \mathbb{I} \text{ and } i \leq h\} = \{1, \dots, h\}.$$

We say that a given iteration of MAIN is **restarting** (resp. **repeating**) iff it terminates and calls the next iteration through RESTART (resp. REPEAT). The repeating iterations are exactly those that proceed according to Subcase 2.1 or Subsubcase 2.2.1 of MAIN; and the restarting iterations are those that proceed according to Case 1 or Subsubsubcase 2.2.2.1, as well as the terminating iterations that proceed according to Subsubsubcase 2.2.2.2. Next, we say that a given iteration of MAIN is **locking** iff it proceeds according to Subsubcase 2.1.2 of MAIN.

Consider any $h \in \mathbb{I}$ and any $i \in \mathbb{I}^h$. We say that the i th iteration of MAIN is \mathbb{I}^h -**transient** iff there is a $j \in \mathbb{I}^h$ such that the following three conditions are satisfied:

- $i \leq j < h$.
- The j th iteration of MAIN restarting.
- There is no e with $i \leq e < j$ such that the e th iteration of MAIN is locking.

For a number $h \in \mathbb{I}$, we define

$$\mathbb{I}_!^h = \{i \mid i \in \mathbb{I}^h \text{ and the } i\text{'th iteration of MAIN is not } \mathbb{I}^h\text{-transient}\}.$$

We say that two bodies are **consistent** with each other iff one is an extension of the other. This, of course, includes the case of their being simply equal.

Lemma 5.12. *Consider any $n \in \{0, \dots, k\}$, $h \in \mathbb{I}$ and $i, j \in \mathbb{I}_!^h$. Suppose \vec{E}_i has an entry $[n, B_i]$, and \vec{E}_j has an entry $[n, B_j]$. Then B_i and B_j are consistent with each other.*

Proof. Assume the conditions of the lemma. The case $i = j$ is trivial, so we shall assume that $i < j$.

First, consider the case $n = k$. We thus want to show that the master bodies of \vec{E}_i and \vec{E}_j are consistent with each other. Notice that only those iterations of MAIN affect the master body of (the evolving) \vec{E} that are either restarting or locking. So, if no iteration between¹² i and j is either restarting or locking, then the master entry of \vec{E}_j is the same as that of \vec{E}_i , and we are done. Now suppose there is an e with $i \leq e \leq j$ such that the e th iteration is restarting or locking. We may assume that e is the smallest such number. Then the e th iteration cannot be restarting, because this would make the i th iteration \mathbb{I}^h -transient. Thus, the e th iteration is locking. Such an iteration “locks” the master body of \vec{E}_i , in the sense that no later iterations can destroy what is already there — such iterations will only extend the master body. So, the master body of \vec{E}_j is an extension of that of \vec{E}_i , implying that, as desired, the two bodies are consistent with each other.

Now, for the rest of this proof, assume $n < k$. Note that $i, j > 1$, because \vec{E}_1 has no common (n -indexed) entries. Further note that the $(i - 1)$ th and $(j - 1)$ th iterations are not restarting ones, because RESTART erases all common entries. Hence, obviously, both $i - 1$ and $j - 1$ are in $\mathbb{I}_!^h$.

The case of either B_i or B_j being empty is trivial, because the empty body is consistent with every body. Thus, we shall assume that

B_i looks like $((\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_a, p_a), (\vec{\alpha}, p))$ and B_j looks like $((\vec{\beta}_1, q_1), \dots, (\vec{\beta}_b, q_b), (\vec{\beta}, q))$

for some $a, b \geq 0$. In what follows, we will be using \vec{P} and \vec{Q} as abbreviations of “ $(\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_a, p_a)$ ” and “ $(\vec{\beta}_1, q_1), \dots, (\vec{\beta}_b, q_b)$ ”, respectively. Thus, $B_i = (\vec{P}, (\vec{\alpha}, p))$ and $B_j = (\vec{Q}, (\vec{\beta}, q))$.

We prove the lemma by complete induction on $i + j$. Assume the aggregation \vec{E}_{i-1} contains the entry $[n, B_i]$. Since $(i - 1) + j < i + j$ and (as we established just a while ago) $(i - 1) \in \mathbb{I}_!^h$, the induction hypothesis applies, according to which B_i is consistent with B_j , as desired. The case of \vec{E}_{j-1} containing the entry $[n, B_j]$ is similar. Now, for the rest of the present proof, we assume that

\vec{E}_{i-1} does not have the entry $[n, B_i]$, and \vec{E}_{j-1} does not have the entry $[n, B_j]$. (5.11)

Assume $a < b$. Then $b \geq 1$. In view of this fact and (5.11), it is easy to see that \vec{E}_{j-1} contains an n -indexed entry whose body is (\vec{Q}) . By the induction hypothesis, $(\vec{P}, (\vec{\alpha}, p))$ is consistent with (\vec{Q}) , meaning (as $a + 1 \leq b$) that the latter is an extension of the former. Hence, $(\vec{P}, (\vec{\alpha}, p))$ is just as well consistent with $(\vec{Q}, (\vec{\beta}, q))$, as desired. The case of $b < a$ will be handled in a similar way.

¹²Here and later, we may terminologically identify iterations with their ordinal numbers.

Now, for the rest of this proof, we further assume that $a = b$. We claim that

$$\vec{P} = \vec{Q}, \text{ i.e., } ((\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_a, p_a)) = ((\vec{\beta}_1, q_1), \dots, (\vec{\beta}_b, q_b)). \quad (5.12)$$

Indeed, the case of $a, b = 0$ is trivial. Otherwise, if $a, b \neq 0$, in view of (5.11), obviously \vec{E}_{i-1} contains the entry $[n, (\vec{P})]$ and \vec{E}_{j-1} contains the entry $[n, (\vec{Q})]$. Hence, by the induction hypothesis, the two bodies (\vec{P}) and (\vec{Q}) are consistent, which, as $a = b$, simply means that they are identical. (5.12) is thus verified. In view of (5.12), all that now remains to show is that $(\vec{\alpha}, p) = (\vec{\beta}, q)$.

Assume a is odd. Analyzing the work of MAIN and keeping (5.11) in mind, we see that the $(i-1)$ th iteration of MAIN proceeds according to Subsubcase 2.1.1, where the central triple of \vec{E}_{i-1} is $((C, (\vec{P}), n)$ for a certain even-size body C , and $\text{SIM}_n^\bullet(C^{\text{even}}, (\vec{P})^{\text{odd}}) = +(\vec{\alpha}, p)$. Similarly, the $(j-1)$ th iteration of MAIN proceeds according to Subsubcase 2.1.1, where the central triple of \vec{E}_{j-1} is $((D, (\vec{Q}), n)$ — which, by (5.12), is the same as $((D, (\vec{P}), n)$ — for a certain even-size body D , and $\text{SIM}_n^\bullet(D^{\text{even}}, (\vec{P})^{\text{odd}}) = +(\vec{\beta}, q)$. Here, if one of the bodies C, D is empty, the two bodies are consistent with each other. Otherwise obviously $n > 0$, \vec{E}_{i-1} contains the entry $[n-1, C]$, and \vec{E}_{j-1} contains the entry $[n-1, D]$. Then, by the induction hypothesis, again, C is consistent with D . Thus, in either case, C and D are consistent. Then clause 2 of Lemma 5.6 implies that $\text{SIM}_n(C^{\text{even}}, (\vec{P})^{\text{odd}}) = \text{SIM}_n(D^{\text{even}}, (\vec{P})^{\text{odd}})$. Consequently, $(\vec{\alpha}, p) = (\vec{\beta}, q)$, as desired.

The case of a being even is rather similar. In this case, the $(i-1)$ th iteration of MAIN deals with Subsubcase 2.2.1, where the central triple of \vec{E}_{i-1} is $((\vec{P}), C, n+1)$ for a certain odd-size body C , with $\text{SIM}_{n+1}^\bullet((\vec{P})^{\text{even}}, C^{\text{odd}}) = -(\vec{\alpha}, p)$. And the $(j-1)$ th iteration of MAIN also deals with Subsubcase 2.2.1, where the central triple of \vec{E}_{j-1} is $((\vec{P}), D, n+1)$ for a certain odd-size body D , with $\text{SIM}_{n+1}^\bullet((\vec{P})^{\text{even}}, D^{\text{odd}}) = -(\vec{\beta}, q)$. So, \vec{E}_{i-1} contains the entry $[n+1, C]$ and \vec{E}_{j-1} contains the entry $[n+1, D]$. Therefore, by the induction hypothesis, C and D are again consistent. Then clause 1 of Lemma 5.6 implies that $\text{SIM}_{n+1}((\vec{P})^{\text{even}}, C^{\text{odd}}) = \text{SIM}_{n+1}((\vec{P})^{\text{even}}, D^{\text{odd}})$, meaning that, as desired, $(\vec{\alpha}, p) = (\vec{\beta}, q)$. \square

Consider any $n \in \{0, \dots, k\}$ and $h \in \mathbb{I}$. We define

$$\mathbb{B}_n^h \quad (5.13)$$

as the smallest-size body such that, for every $i \in \mathbb{I}_1^h$, whenever \vec{E}_i has an n -indexed entry, \mathbb{B}_n^h is a (not necessarily proper) extension of that entry's body. In view of Lemma 5.12, such a \mathbb{B}_n^h always exists. We further define the bodies $\mathbb{B}_n^h \uparrow$ and $\mathbb{B}_n^h \downarrow$ as follows. Let $\mathbb{B}_n^h = (O_1, \dots, O_s)$. We agree that below and later, where t is 0 or a negative integer, the denotation of an expression like (P_1, \dots, P_t) should be understood as the empty tuple $()$. Then:

$$\begin{aligned} \mathbb{B}_n^h \uparrow &= \begin{cases} (O_1, \dots, O_s) & \text{if } s \text{ is even;} \\ (O_1, \dots, O_{s-1}) & \text{if } s \text{ is odd.} \end{cases} \\ \mathbb{B}_n^h \downarrow &= \begin{cases} (O_1, \dots, O_s) & \text{if } s \text{ is odd;} \\ (O_1, \dots, O_{s-1}) & \text{if } s \text{ is even.} \end{cases} \end{aligned}$$

Assume $h \in \mathbb{I}$, $n \in \{0, \dots, k\}$, and (P_1, \dots, P_p) is a nonempty, not necessarily proper, restriction of the body \mathbb{B}_n^h . By the (h, n) -birthtime of (P_1, \dots, P_p) we shall mean the smallest number $i \in \mathbb{I}_1^h$ such that, for some (not necessarily proper) extension B of (P_1, \dots, P_p) , \vec{E}_i

has the entry $[n, B]$. We extend this concept to the case $p = 0$ by stipulating that the (h, n) -birthtime of the empty body () is always 0. In informal discourses we may say “ (O_1, \dots, O_p) was (h, n) -born at time i ” to mean that i is the (h, n) -birthtime of (O_1, \dots, O_p) . When h and n are fixed or clear from the context, we may omit a reference to (h, n) and simply say “birthtime” or “born”.

Lemma 5.13. *Consider any $h \in \mathbb{I}$ and $n \in \{1, \dots, k\}$. Let $\mathbb{B}_{n-1}^h \downarrow = (P_1, \dots, P_p)$ and $\mathbb{B}_n^h \uparrow = (Q_1, \dots, Q_q)$, where $q > 0$. Further let i_P be the $(h, n - 1)$ -birthtime of (P_1, \dots, P_p) and i_Q be the (h, n) -birthtime of (Q_1, \dots, Q_q) .*

1. If $i_Q > i_P$, then we have:

$$\text{SIM}_n^\bullet((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = +Q_q; \quad (5.14)$$

The triple $((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}, n)$ is saturated; (5.15)

$$\text{SIM}_n^\rightarrow((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = \mathbb{B}_n^h \uparrow; \quad (5.16)$$

$$\text{SIM}_n^\leftarrow((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = \mathbb{B}_{n-1}^h. \quad (5.17)$$

2. If $i_P > i_Q$, then we have:

$$\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, (\mathbb{B}_n^h)^{\text{odd}}) = -P_p; \quad (5.18)$$

The triple $((P_1, \dots, P_{p-1})^{\text{even}}, (\mathbb{B}_n^h)^{\text{odd}}, n)$ is saturated; (5.19)

$$\text{SIM}_n^\rightarrow((P_1, \dots, P_{p-1})^{\text{even}}, (\mathbb{B}_n^h)^{\text{odd}}) = \mathbb{B}_n^h; \quad (5.20)$$

$$\text{SIM}_n^\leftarrow((P_1, \dots, P_{p-1})^{\text{even}}, (\mathbb{B}_n^h)^{\text{odd}}) = \mathbb{B}_{n-1}^h \downarrow. \quad (5.21)$$

Proof. Assume the conditions of the lemma. Take a note of the fact that $i_P, i_Q \in \mathbb{I}_!^h$.

1. Assume $i_Q > i_P$. Note that, by the definition of $\mathbb{B}_n^h \uparrow$, q is even.

Since ($q > 0$ and) q is even, at time i_Q the body (Q_1, \dots, Q_q) obviously must have been “born” — i.e., the transition from the $(i_Q - 1)$ th iteration to the i_Q th iteration must have happened — according to the scenario of Subcase 2.1 of MAIN. Namely, in that scenario, the central triple of \vec{E}_{i_Q-1} was $(C, (Q_1, \dots, Q_{q-1}), n)$ for a certain even-size body C , and $\text{SIM}_n^\bullet(C^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = +Q_q$. Since the $(i_Q - 1)$ th iteration of MAIN was not a restarting one, $i_Q - 1$ is in $\mathbb{I}_!^h$ just like i_Q is. Therefore, by the definition (5.13) of \mathbb{B}_{n-1}^h , \mathbb{B}_{n-1}^h is an extension of C . Now, (5.14) holds by clause 2 of Lemma 5.6.

To verify claim (5.15), deny it for a contradiction. That is, assume there is a proper restriction D of $(\mathbb{B}_{n-1}^h)^{\text{even}}$ such that $\text{SIM}_n^\bullet(D, (Q_1, \dots, Q_{q-1})^{\text{odd}})$ is positive. Since $(\mathbb{B}_{n-1}^h)^{\text{even}}$ has a proper restriction, the size of \mathbb{B}_{n-1}^h is at least 2, and therefore, by the definition of $\mathbb{B}_{n-1}^h \downarrow$, p is an odd positive integer. Since D is a proper restriction of $(\mathbb{B}_{n-1}^h)^{\text{even}}$, it is also a (not necessarily proper) restriction of $(P_1, \dots, P_p)^{\text{even}}$. Furthermore, since p is odd, $(P_1, \dots, P_p)^{\text{even}} = (P_1, \dots, P_{p-1})^{\text{even}}$. Consequently, $D = (P_1, \dots, P_r)^{\text{even}}$ for some r strictly smaller than p . We may assume that r is even, for otherwise $(P_1, \dots, P_r)^{\text{even}} = (P_1, \dots, P_{r-1})^{\text{even}}$ and we could have taken $r - 1$ instead of r . Thus, for the nonnegative even integer r with $r < p$,

$$\text{SIM}_n^\bullet((P_1, \dots, P_r)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) \text{ is positive.} \quad (5.22)$$

Let j be the $(h, n - 1)$ -birthtime of (P_1, \dots, P_{r+1}) . Note that $j \leq i_P$, and hence $j < i_Q$. Since $r + 1$ is odd, (P_1, \dots, P_{r+1}) must have been born according to the scenario of Subsubcase 2.2.1 of MAIN. Namely, in that scenario, $(j > 1 \text{ and} j - 1 \in$

$\mathbb{I}_!^h$, the central triple of \vec{E}_{j-1} is $((P_1, \dots, P_r), A, n)$ for some odd-size body A , and $\text{SIM}_n^\bullet((P_1, \dots, P_r)^{\text{even}}, A^{\text{odd}}) = -P_{r+1}$. By definition (5.13), the body \mathbb{B}_n^h is an extension of A . But, since $j < i_Q$, (Q_1, \dots, Q_q) was not yet (h, n) -born at time j . So, we must have $A = (Q_1, \dots, Q_s)$ for some $s \leq q-1$. Therefore, by clause 1 of Lemma 5.6,

$$\text{SIM}_n^\bullet((P_1, \dots, P_r)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = -P_{r+1}.$$

The above, however, contradicts (5.22). Claim (5.15) is thus proven.

To justify (5.16), assume $\text{SIM}_n^\rightarrow((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = (U_1, \dots, U_u)$. We want to show that $(U_1, \dots, U_u) = (Q_1, \dots, Q_q)$. With (5.14) and the evenness of q in mind, we can see directly from the definition of SIM_n^\rightarrow that $u = q$, and that $(U_1, \dots, U_q)^{\text{odd}} = (Q_1, \dots, Q_{q-1})^{\text{odd}}$. q 's being even further implies that $(Q_1, \dots, Q_{q-1})^{\text{odd}} = (Q_1, \dots, Q_q)^{\text{odd}}$. So, $(U_1, \dots, U_u)^{\text{odd}} = (Q_1, \dots, Q_q)^{\text{odd}}$. It remains to show that we have $(U_1, \dots, U_u)^{\text{even}} = (Q_1, \dots, Q_q)^{\text{even}}$ as well, i.e., $(U_1, \dots, U_q)^{\text{even}} = (Q_1, \dots, Q_q)^{\text{even}}$. Consider any even $r \in \{1, \dots, q\}$. Let j be the (h, n) -birthtime of (Q_1, \dots, Q_r) . Obviously this body must have been born according to the scenario of Subcase 2.1 of MAIN in which $j-1 \in \mathbb{I}_!^h$, \vec{E}_{j-1} has the entry $[n, (Q_1, \dots, Q_{r-1})]$ and, with $(C, (Q_1, \dots, Q_{r-1}), n)$ being the central triple of \vec{E}_{j-1} for some even-size body C , we have $\text{SIM}_n^\bullet(C^{\text{even}}, (Q_1, \dots, Q_{r-1})^{\text{odd}}) = +Q_r$. By definition (5.13), \mathbb{B}_{n-1}^h is an extension of C . So, by clause 2 of Lemma 5.6,

$$\text{SIM}_n^\bullet((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{r-1})^{\text{odd}}) = +Q_r. \quad (5.23)$$

But how does the computation of (5.23) differ from the computation of (5.14)? The two computations proceed in exactly the same ways, with the variable S of SIM_n^\bullet going through exactly the same values in both cases, with the only difference that, while the computation of (5.23) stops after S takes its $(r/2)$ th positive value $+U_r$ and returns that value as $+Q_r$, the computation of (5.14) continues further (if $r \neq q$) until the value of S becomes $+U_q$. As we see, we indeed have $U_r = Q_r$ as desired. Claim (5.16) is now verified.

Claim (5.17) can be verified in a rather similar way. Assume

$$\text{SIM}_n^\leftarrow((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = (V_1, \dots, V_v).$$

We want to show that $(V_1, \dots, V_v) = \mathbb{B}_{n-1}^h$. By the definition of SIM_n^\leftarrow , $(V_1, \dots, V_v)^{\text{even}} = (\mathbb{B}_{n-1}^h)^{\text{even}}$. It remains to show that we also have $(V_1, \dots, V_v)^{\text{odd}} = (\mathbb{B}_{n-1}^h)^{\text{odd}}$. Notice that $(\mathbb{B}_{n-1}^h)^{\text{odd}} = (\mathbb{B}_{n-1}^h \downarrow)^{\text{odd}} = (P_1, \dots, P_p)^{\text{odd}}$, and that $(p \leq v \text{ and}) (V_1, \dots, V_v)^{\text{odd}} = (V_1, \dots, V_p)^{\text{odd}}$. So, what we want to show is $(V_1, \dots, V_p)^{\text{odd}} = (P_1, \dots, P_p)^{\text{odd}}$. Consider any odd $r \in \{1, \dots, p\}$. Let j be the $(h, n-1)$ -birthtime of (P_1, \dots, P_r) . Note that $j \leq i_P$ and hence $j < i_Q$. The birth of (P_1, \dots, P_r) should have occurred according to Subsubcase 2.2.1 of MAIN, in a situation where $1 \leq j-1 \in \mathbb{I}_!^h$, the central triple of \vec{E}_{j-1} is $((P_1, \dots, P_{r-1}), C, n)$ for some odd-size body C , and $\text{SIM}_n^\bullet((P_1, \dots, P_{r-1})^{\text{even}}, C^{\text{odd}}) = -P_r$. But (Q_1, \dots, Q_q) is an extension of C because so is \mathbb{B}_n^h . In fact, it is a proper extension, because (Q_1, \dots, Q_q) was not yet (h, n) -born at time j . So, $(Q_1, \dots, Q_{q-1})^{\text{odd}}$ is a (not necessarily proper) extension of C^{odd} . Hence, by clause 1 of Lemma 5.6,

$$\text{SIM}_n^\bullet((P_1, \dots, P_{r-1})^{\text{even}}, (Q_1, \dots, Q_{q-1})^{\text{odd}}) = -P_r. \quad (5.24)$$

But how does the computation of (5.24) differ from the computation of (5.14)? The two computations proceed in exactly the same ways, with the variable S of SIM_n^\bullet going through exactly the same values in both cases, with the only difference that, while

the computation of (5.24) stops after S takes its $((r+1)/2)$ th negative value $-V_r$ and returns that value as $-P_r$, the computation of (5.14) continues further until the value of S becomes $+Q_q$. As we see, we indeed have $V_r = P_r$ as desired. This completes our proof of clause 1 of the lemma.

2. Assume $i_P > i_Q$. Note that p is odd, q is even and $q \neq 0$.

The way (P_1, \dots, P_p) was born is that the central triple of \vec{E}_{i_P-1} had the form $((P_1, \dots, P_{p-1}), C, n)$ for a certain odd-size body C , and $\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, C^{\text{odd}}) = -P_p$. But \mathbb{B}_n^h is an extension of C . Therefore (5.18) holds by Lemma 5.6.

To verify (5.19), deny it for a contradiction: assume there is a proper restriction D of $(\mathbb{B}_n^h)^{\text{odd}}$ such that $\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, D)$ is negative. D 's being a proper restriction of $(\mathbb{B}_n^h)^{\text{odd}}$ implies that $D = (Q_1, \dots, Q_r)^{\text{odd}}$ for some odd r — fix it — strictly smaller than q . Thus,

$$\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, (Q_1, \dots, Q_r)^{\text{odd}}) \text{ is negative.} \quad (5.25)$$

Let j be the birthtime of (Q_1, \dots, Q_{r+1}) . Note that $j \leq i_Q$, and hence $j < i_P$. (Q_1, \dots, Q_{r+1}) 's birth must have happened in a situation where $1 \leq j-1 \in \mathbb{I}_!^h$, the central triple of \vec{E}_{j-1} happens to be $(A, (Q_1, \dots, Q_r), n)$ for some even-size body A , and $\text{SIM}_n^\bullet(A^{\text{even}}, (Q_1, \dots, Q_r)^{\text{odd}}) = +Q_{r+1}$. \mathbb{B}_{n-1}^h is an extension of A . But since $j < i_P$, (P_1, \dots, P_p) was not yet born at time j . So, $A = (P_1, \dots, P_s)$ for some $s \leq p-1$. Therefore, by Lemma 5.6,

$$\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, (Q_1, \dots, Q_r)^{\text{odd}}) = +Q_{r+1}.$$

The above, however, contradicts (5.25). Claim (5.19) is thus proven.

For (5.20), assume $\text{SIM}_n^\rightarrow((P_1, \dots, P_{p-1})^{\text{even}}, (\mathbb{B}_n^h)^{\text{odd}}) = (U_1, \dots, U_u)$. We want to show $(U_1, \dots, U_u) = \mathbb{B}_n^h$. Directly from the definition of SIM_n^\rightarrow , $(U_1, \dots, U_u)^{\text{odd}} = (\mathbb{B}_n^h)^{\text{odd}}$. It remains to show that $(U_1, \dots, U_u)^{\text{even}} = (\mathbb{B}_n^h)^{\text{even}}$. Note that $(\mathbb{B}_n^h)^{\text{even}} = (\mathbb{B}_n^h \uparrow)^{\text{even}} = (Q_1, \dots, Q_q)^{\text{even}}$, and that ($q \leq u$ and) $(U_1, \dots, U_u)^{\text{even}} = (U_1, \dots, U_q)^{\text{even}}$. So, what we want to show is $(U_1, \dots, U_q)^{\text{even}} = (Q_1, \dots, Q_q)^{\text{even}}$. For this purpose, consider any even $r \in \{1, \dots, q\}$. Let j be the (h, n) -birthtime of (Q_1, \dots, Q_r) . Obviously (Q_1, \dots, Q_r) must have been born according to the scenario of Subcase 2.1 of MAIN in which $j-1 \in \mathbb{I}_!^h$, \vec{E}_{j-1} has the entry $[n, (Q_1, \dots, Q_{r-1})]$ and, with $(C, (Q_1, \dots, Q_{r-1}), n)$ being the central triple of \vec{E}_{j-1} for some even-size body C , $\text{SIM}_n^\bullet(C^{\text{even}}, (Q_1, \dots, Q_{r-1})^{\text{odd}}) = +Q_r$. By definition (5.13), \mathbb{B}_{n-1}^h is an extension of C . Therefore, by Lemma 5.6, $\text{SIM}_n^\bullet((\mathbb{B}_{n-1}^h)^{\text{even}}, (Q_1, \dots, Q_{r-1})^{\text{odd}}) = +Q_r$. However, $(\mathbb{B}_{n-1}^h)^{\text{even}} = (\mathbb{B}_{n-1}^h \downarrow)^{\text{even}} = (P_1, \dots, P_p)^{\text{even}}$; further, since p is odd, $(P_1, \dots, P_p)^{\text{even}} = (P_1, \dots, P_{p-1})^{\text{even}}$, and hence $(\mathbb{B}_{n-1}^h)^{\text{even}} = (P_1, \dots, P_{p-1})^{\text{even}}$. Thus we have:

$$\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, (Q_1, \dots, Q_{r-1})^{\text{odd}}) = +Q_r. \quad (5.26)$$

Comparing the computations of (5.18) and (5.26), we see that the two computations proceed in exactly the same ways, with the only difference that, while the computation of (5.26) stops after variable S of SIM_n^\bullet takes its $(r/2)$ th positive value $+U_r$ and returns that value as $+Q_r$, the computation of (5.18) continues further until the value of S becomes $-P_p$. As we see, we indeed have $U_r = Q_r$ as desired. Claim (5.20) is verified.

For (5.21), assume $\text{SIM}_n^\leftarrow((P_1, \dots, P_{p-1})^{\text{even}}, (\mathbb{B}_n^h)^{\text{odd}}) = (V_1, \dots, V_v)$. We want to show that $(V_1, \dots, V_v) = \mathbb{B}_{n-1}^h \downarrow$. With (5.18) and the oddness of p in mind, we see from the definition of SIM_n^\leftarrow that $v = p$, and that $(V_1, \dots, V_p)^{\text{even}} = (P_1, \dots, P_{p-1})^{\text{even}}$. The fact

that p is odd additionally implies $(P_1, \dots, P_{p-1})^{even} = (P_1, \dots, P_p)^{even}$. Consequently, $(V_1, \dots, V_v)^{even} = (P_1, \dots, P_p)^{even} = (\mathbb{B}_{n-1}^h \downarrow)^{even}$. So, it remains to show that we also have $(V_1, \dots, V_v)^{odd} = (\mathbb{B}_{n-1}^h \downarrow)^{odd}$, i.e., $(V_1, \dots, V_p)^{odd} = (P_1, \dots, P_p)^{odd}$. Consider any odd $r \in \{1, \dots, p\}$. Let j be the $(h, n-1)$ -birthtime of (P_1, \dots, P_r) . Note that $j \leq i_P$. The birth of (P_1, \dots, P_r) should have occurred according to Subsubcase 2.2.1 of MAIN, in a situation where $1 \leq j-1 \in \mathbb{I}_l^h$, the central triple of \vec{E}_{j-1} is $((P_1, \dots, P_{r-1}), C, n)$ for some odd-size body C , and $\text{SIM}_n^\bullet((P_1, \dots, P_{r-1})^{even}, C^{odd}) = -P_r$. But \mathbb{B}_n^h is an extension of C . Hence, by Lemma 5.6,

$$\text{SIM}_n^\bullet((P_1, \dots, P_{r-1})^{even}, (\mathbb{B}_n^h)^{odd}) = -P_r. \quad (5.27)$$

Compare the computations of (5.27) and (5.18). The two computations proceed in exactly the same ways, with the only difference that, while the computation of (5.27) stops after S takes its $((r+1)/2)$ th negative value $-V_r$ and returns that value as $-P_r$, the computation of (5.18) continues further (if $r \neq p$) until the value of S becomes $-P_p$. Thus $V_r = P_r$, as desired. \square

We agree for the rest of Section 5 that \hbar is the greatest element of \mathbb{I} . The existence of such an element is guaranteed by Lemma 5.11.

Lemma 5.14. *The following statements are true (with \hbar as above):*

1. *For every $n \in \{0, \dots, k\}$, the size of \mathbb{B}_n^h is odd.*
2. *For every $n \in \{1, \dots, k\}$, the $(\hbar, n-1)$ -birthtime of \mathbb{B}_{n-1}^h is greater than the (\hbar, n) -birthtime of \mathbb{B}_n^h .*
3. *For every $n \in \{0, \dots, k\}$, the scale of the last organ of \mathbb{B}_n^h is the same as the master scale of \vec{E}_\hbar .*

Proof.

1. Assume $n \in \{0, \dots, k\}$, $\mathbb{B}_n^h = (Q_1, \dots, Q_q)$, and i_Q is the (\hbar, n) -birthtime of \mathbb{B}_n^h . If $n \geq 1$, further assume that $\mathbb{B}_{n-1}^h = (P_1, \dots, P_p)$, and i_P is the $(\hbar, n-1)$ -birthtime of \mathbb{B}_{n-1}^h .

We first verify that

$$\text{If } n = 0, \text{ then } q \text{ is odd.} \quad (5.28)$$

Indeed, assume $n = 0$. Consider the last, i.e., \hbar th, iteration of MAIN. This must be an iteration that proceeds according to Subsubsubcase 2.2.2.2, because all other sorts of iterations always either REPEAT or RESTART. Namely, the central triple of \vec{E}_\hbar is $(((), B, 0))$ for some odd-size body B , and $\text{SIM}_0^\bullet(((), B)^{odd})$ is negative. Of course the \hbar th iteration is not \hbar -transient, so $\hbar \in \mathbb{I}_l^h$. By definition (5.13), \mathbb{B}_0^h is an extension of B . So, $B = (Q_1, \dots, Q_a)$ for some odd a with $a \leq q$. Suppose $a < q$. Let i be the $(\hbar, 0)$ -birthtime of (Q_1, \dots, Q_{a+1}) . Obviously the birth of (Q_1, \dots, Q_{a+1}) must have occurred according to the scenario of Subcase 2.1 of MAIN in which $i > 1$, \vec{E}_{i-1} contains the entry $[0, (Q_1, \dots, Q_a)]$, i.e., $[0, B]$, and $\text{SIM}_0^\bullet(((), B)^{odd}) = +Q_{a+1}$. This, however, contradicts with our earlier observation that $\text{SIM}_0^\bullet(((), B)^{odd})$ is negative. From this contradiction we conclude that $a = q$. If so, (5.28) holds, because, as already noted, a is odd.

We next verify that

$$\text{If } n \in \{1, \dots, k\}, \text{ then } q \text{ is odd.} \quad (5.29)$$

Our proof of (5.29) is, in fact, by induction on $n \geq 1$. Assume $n \in \{1, \dots, k\}$. By (5.28) if $n = 1$ (i.e., if we are dealing with the basis of induction), and by the induction

hypothesis if $n > 1$ (i.e., if we are dealing with the inductive step), we have:

$$p \text{ (the size of } \mathbb{B}_{n-1}^{\hbar} \text{) is odd.} \quad (5.30)$$

Obviously, (5.30) implies that (P_1, \dots, P_p) was born according to the scenario of Subsubcase 2.2.1 of MAIN in which $i_P - 1 \in \mathbb{I}_!^{\hbar}$, the central triple of \vec{E}_{i_P-1} is $((P_1, \dots, P_{p-1}), C, n)$ for a certain odd-size body C , and

$$\text{SIM}_n^{\bullet}((P_1, \dots, P_{p-1})^{even}, C^{odd}) = -P_p.$$

By definition (5.13), \mathbb{B}_n^{\hbar} is an extension of C . Hence, by clause 1 of Lemma 5.6,

$$\text{SIM}_n^{\bullet}((P_1, \dots, P_{p-1})^{even}, (Q_1, \dots, Q_q)^{odd}) = -P_p. \quad (5.31)$$

For a contradiction suppose (5.29) is false, i.e., assume q is even. Then $q \geq 2$, because $\mathbb{B}_n^{\hbar} = (Q_1, \dots, Q_q)$ is an extension of the odd-size C . Remember that i_Q is the (\hbar, n) -birthtime of (Q_1, \dots, Q_q) . Since q is even, (Q_1, \dots, Q_q) must have been born according to the scenario of Subcase 2.1 of MAIN in which $i_Q - 1 \in \mathbb{I}_!^{\hbar}$, \vec{E}_{i_Q-1} contains the entry $[n, (Q_1, \dots, Q_{q-1})]$ and, with $(D, (Q_1, \dots, Q_{q-1}), n)$ being the central triple of \vec{E}_{i_Q-1} for some even-size restriction D of (P_1, \dots, P_p) ,

$$\text{SIM}_n^{\bullet}(D^{even}, (Q_1, \dots, Q_{q-1})^{odd}) = +Q_q.$$

But since — by (5.30) — p is odd, (P_1, \dots, P_{p-1}) is an extension of D . Hence, by clause 2 of Lemma 5.6,

$$\text{SIM}_n^{\bullet}((P_1, \dots, P_{p-1})^{even}, (Q_1, \dots, Q_{q-1})^{odd}) = +Q_q,$$

which, as q is even and hence $(Q_1, \dots, Q_q)^{odd} = (Q_1, \dots, Q_{q-1})^{odd}$, is the same as to say that

$$\text{SIM}_n^{\bullet}((P_1, \dots, P_{p-1})^{even}, (Q_1, \dots, Q_q)^{odd}) = +Q_q. \quad (5.32)$$

Comparing (5.31) with (5.32), we see a desired contradiction. This completes our proof of (5.29) and hence of clause 1 of the lemma, because the latter is nothing but (5.28) and (5.29) put together.

2. Assume $n \in \{1, \dots, k\}$, $\mathbb{B}_{n-1}^{\hbar} = (P_1, \dots, P_p)$, i_P is the $(\hbar, n - 1)$ -birthtime of \mathbb{B}_{n-1}^{\hbar} , $\mathbb{B}_n^{\hbar} = (Q_1, \dots, Q_q)$, and i_Q is the (\hbar, n) -birthtime of \mathbb{B}_n^{\hbar} . For a contradiction, further assume $i_P \leq i_Q$. From the already verified clause 1 of the present lemma, we know that both p and q are odd. The oddness of p implies that, at time i_P , (P_1, \dots, P_p) was born according to the scenario of Subsubcase 2.2.1 of MAIN in which $i_P - 1 \in \mathbb{I}_!^{\hbar}$, the central triple of \vec{E}_{i_P-1} is $((P_1, \dots, P_{p-1}), C, n)$ for some odd-size body C , and $\text{SIM}_n^{\bullet}((P_1, \dots, P_{p-1})^{even}, C^{odd}) = -P_p$. By definition (5.13), (Q_1, \dots, Q_q) is an extension of C . Further, since q is odd and the body (Q_1, \dots, Q_q) was not yet born at time $i_P - 1$, we have $q \geq 3$, with (Q_1, \dots, Q_{q-2}) being an extension of C . Then, by clause 1 of Lemma 5.6,

$$\text{SIM}_n^{\bullet}((P_1, \dots, P_{p-1})^{even}, (Q_1, \dots, Q_{q-2})^{odd}) = -P_p. \quad (5.33)$$

Let j be the (\hbar, n) -birthtime of (Q_1, \dots, Q_{q-1}) . The birth of (Q_1, \dots, Q_{q-1}) should have occurred according to the scenario Subcase 2.1 of MAIN in which $j - 1 \in \mathbb{I}_!^{\hbar}$, the central triple of \vec{E}_{j-1} is $(D^{even}, (Q_1, \dots, Q_{q-2})^{odd}, n)$ for some even-size body D , and

$$\text{SIM}_n^{\bullet}(D^{even}, (Q_1, \dots, Q_{q-2})^{odd}) = +Q_{q-1}.$$

By definition (5.13), (P_1, \dots, P_p) is an extension of D . So, by clause 2 of Lemma 5.6, $\text{SIM}_n^\bullet((P_1, \dots, P_p)^{\text{even}}, (Q_1, \dots, Q_{q-2})^{\text{odd}}) = +Q_{q-1}$. But, since p is odd, we have

$$(P_1, \dots, P_p)^{\text{even}} = (P_1, \dots, P_{p-1})^{\text{even}}.$$

Thus,

$$\text{SIM}_n^\bullet((P_1, \dots, P_{p-1})^{\text{even}}, (Q_1, \dots, Q_{q-2})^{\text{odd}}) = +Q_{q-1}.$$

The above is in contradiction with (5.33).

3. We start with the following claim:

Claim 1. Consider any $n \in \{0, \dots, k\}$. Assume $\mathbb{B}_n^\hbar = (Q_1, \dots, Q_{q+1})$, and t is an even number with $2 \leq t \leq q$. Then the scale of Q_t is the same as that of Q_{t-1} .

To verify this claim, assume its conditions. We proceed by induction on $n = 0, 1, \dots, k$.

For the basis of induction, consider the case of $n = 0$. Let i be the $(\hbar, 0)$ -birthtime of (Q_1, \dots, Q_t) . Obviously the $(i - 1)$ th iteration of MAIN follows the scenario of Subcase 2.1 where $i - 1 \in \mathbb{I}_i^\hbar$, the central triple of \vec{E}_{i-1} is $(((), (Q_1, \dots, Q_{t-1}), 0))$, and

$$\text{SIM}_0^\bullet(((), (Q_1, \dots, Q_{t-1}))^{\text{odd}}) = +Q_t. \quad (5.34)$$

Looking back at the description of the procedure SIM_0^\bullet , we see that, in computing (5.34), the procedure simply lets the scale of the output $+Q_t$ be a copy of the scale of the “last-fetched” organ Q_{t-1} . Done.

For the inductive step, assume $n \geq 1$. Let $\mathbb{B}_{n-1}^\hbar = (P_1, \dots, P_p)$. From clause 1 of the present lemma we know that both p and $q + 1$ are odd. Note that, for this reason, $\mathbb{B}_{n-1}^\hbar \downarrow = (P_1, \dots, P_p)$ and $\mathbb{B}_n^\hbar \uparrow = (Q_1, \dots, Q_q)$. Let i_P be the $(\hbar, n - 1)$ -birthtime of $\mathbb{B}_{n-1}^\hbar \downarrow$, and i_Q be the (\hbar, n) -birthtime of $\mathbb{B}_n^\hbar \uparrow$. Clause 2 of the present lemma implies that $i_P > i_Q$. Hence the statements (5.18)-(5.21) of Lemma 5.13, with \hbar in the role of h , are true. Let us again remember the work of SIM^\bullet and imagine the computation of (5.18) (with $h = \hbar$). With some thought and with (5.18)-(5.21) in mind, we can see the following scenario. At some point — by the end of one of the iterations of LOOP_n , to be more specific — the variable R of SIM_n^\bullet takes the value $+Q_{t-1}$. Let g be the scale of Q_{t-1} . By the end of the next iteration of LOOP_n , the variable S of SIM_n^\bullet becomes either $+Q_t$, or $-P_{j-1}$ for some even $j \in \{1, \dots, p\}$, with the scale of S in either case being the same as the scale g of the latest (by that time) value of R . Thus, if S becomes $+Q_t$, the scale of Q_t is the same as that of Q_{t-1} , and we are done. If S becomes $-P_{j-1}$, then, immediately after that (on the same iteration of LOOP_n), R takes the value $-P_j$. By the induction hypothesis, the scale of P_j is the same as the scale g of P_{j-1} . On the iterations of LOOP_n that follow, S and R may take several (possibly zero) consecutive values from the series $-P_{j+1}, -P_{j+3}, \dots$ and $-P_{j+2}, -P_{j+4}, \dots$, respectively, and the scales of all these values will remain to be g for the same reasons as above. Sooner or later, after this series of negative values, S becomes $+Q_t$. The scale of this signed organ, as before, will be the same as the scale g of the latest value of R . The scale of Q_t is thus the same as that of Q_{t-1} , which ends our proof of Claim 1.

Now, we prove clause 3 of the lemma by induction on $k - n$. Let m be the master scale of \vec{E}_\hbar . The basis case of $k - n = 0$, i.e., $n = k$, is straightforward. Next, consider any $n \in \{1, \dots, k\}$. By the induction hypothesis, the scale of the last organ of \mathbb{B}_n^\hbar is m . Let, as in the inductive step of the above proof of Claim 1, $\mathbb{B}_{n-1}^\hbar = (P_1, \dots, P_p)$ and $\mathbb{B}_n^\hbar = (Q_1, \dots, Q_{q+1})$. Arguing as in that proof — with $q + 1$ in the role of $t - 1$, m in the

role of g and relying on Claim 1 itself where the proof of the inductive step of the proof of Claim 1 relied on its induction hypothesis — we find that, in the process of computing (5.18) (with $h = \hbar$), at some point, the variable R of the procedure SIM_n^\bullet takes the value $+Q_{q+1}$ (its last positive value) and that, beginning from that point, the scale m will be inherited by all subsequent negative values that the variables S and R assume, which (in the present case) include the final value $-P_p$ assumed by S . Thus, as desired, the scale of the last organ P_p of \mathbb{B}_{n-1}^h is the same as the master scale m of \vec{E}_\hbar . \square

Lemma 5.15. *Consider an arbitrary member h of \mathbb{I} .*

1. (a) *There is a run Γ_0^h generated by \mathcal{H}_0 such that $\overline{\mathbb{B}_0^h} \preceq \Gamma_0^h$.*
 (b) *Furthermore, if h is the greatest element of \mathbb{I} and $\overline{\mathbb{B}_0^h}$ is a reasonable run of $F'(0)$, then, for such a Γ_0^h , we simply have $\overline{\mathbb{B}_0^h} = \Gamma_0^h$.*
2. *Consider any $n \in \{1, \dots, k\}$.*
 - (a) *There is a run Γ_n^h generated by \mathcal{H}_n such that $\overline{\mathbb{B}_n^h \uparrow} \preceq (\Gamma_n^h)^1$ and $\neg \overline{\mathbb{B}_{n-1}^h \downarrow} \preceq (\Gamma_n^h)^0$.*
 - (b) *Furthermore, if h is the greatest element of \mathbb{I} , $\overline{\mathbb{B}_n^h}$ is a reasonable run of $F'(n)$ and $\overline{\mathbb{B}_{n-1}^h}$ is a reasonable run of $F'(n-1)$, then, for such a Γ_n^h , we simply have $\overline{\mathbb{B}_n^h} = (\Gamma_n^h)^1$ and $\neg \overline{\mathbb{B}_{n-1}^h} = (\Gamma_n^h)^0$.*

Proof. Fix an arbitrary $h \in \mathbb{I}$.

1. Let $\mathbb{B}_0^h = (T_1, \dots, T_t)$.

If $t = 0$, then the position $\overline{\mathbb{B}_0^h}$ is empty, and is thus an initial segment of any run. So, an arbitrarily selected run Γ_0^h generated by \mathcal{H}_0 — such as, for instance, the run in which Environment made no moves at all — satisfies subclause (a). As for subclause (b), it is trivially satisfied because, by clause 1 of Lemma 5.14, h is not the greatest element of \mathbb{I} , for otherwise t would have to be odd.

Now, for the rest of our proof of clause 1, assume $t \geq 1$. This automatically makes $(((), \mathbb{B}_0^h)^{odd}, 0)$ a SIM-appropriate triple. We first claim that

$$\text{For any nonempty proper restriction } C \text{ of } (\mathbb{B}_0^h)^{odd}, \text{SIM}_0^\bullet(((), C)) \text{ is positive.} \quad (5.35)$$

For a contradiction suppose (5.35) if false, and assume that, for some nonempty proper restriction C of $(\mathbb{B}_0^h)^{odd}$, $\text{SIM}_0^\bullet(((), C))$ is negative. Obviously $C = (T_1, \dots, T_s)^{odd}$ for some odd s with $s < t$. Fix such an s . Thus,

$$\text{SIM}_0^\bullet(((), (T_1, \dots, T_s)^{odd})) \text{ is negative.} \quad (5.36)$$

Let i be the $(h, 0)$ -birthtime of (T_1, \dots, T_{s+1}) . This means that $i - 1 \in \mathbb{I}_i^h$, the $(i - 1)$ th iteration of MAIN proceeds according to Subcase 2.1, \vec{E}_{i-1} contains the entry $[0, (T_1, \dots, T_s)]$ and, with $(((), (T_1, \dots, T_s), 0)$ being the central triple of \vec{E}_{i-1} , we have $\text{SIM}_0^\bullet(((), (T_1, \dots, T_s)^{odd})) = +T_{s+1}$. This, however, contradicts (5.36). Claim (5.35) is thus verified.

Now we observe that

$$\text{The triple } (((), \mathbb{B}_0^h)^{odd}, 0) \text{ is saturated.} \quad (5.37)$$

Indeed, if $\text{SIM}_0^\bullet(((), \mathbb{B}_0^h)^{odd})$ is positive, then (5.37) automatically holds because the empty body $()$ has no proper restrictions; and if $\text{SIM}_0^\bullet(((), \mathbb{B}_0^h)^{odd})$ is negative, then (5.37) is an immediate consequence of (5.35).

Our next claim is that

$$\text{SIM}_0^{\rightarrow}(((), (\mathbb{B}_0^h)^{odd})) \text{ is an extension of } \mathbb{B}_0^h. \quad (5.38)$$

To justify this claim, assume $\text{SIM}_0^{\rightarrow}(((), (\mathbb{B}_0^h)^{odd})) = (W_1, \dots, W_w)$. From the definition of $\text{SIM}_0^{\rightarrow}$, we have $(W_1, \dots, W_w)^{odd} = (\mathbb{B}_0^h)^{odd}$. So, we only need to show that $(W_1, \dots, W_w)^{even}$ is an extension of $(\mathbb{B}_0^h)^{even}$, i.e., of $(T_1, \dots, T_t)^{even}$. But indeed, consider any even $r \in \{1, \dots, t\}$. Let i be the $(h, 0)$ -birthtime of (T_1, \dots, T_r) . This means that $i-1 \in \mathbb{I}_1^h$, the $(i-1)$ th iteration of MAIN proceeds according to the scenario of Subcase 2.1 where \vec{E}_{i-1} has the entry $[0, (T_1, \dots, T_{r-1})]$ and, with $(((), (T_1, \dots, T_{r-1}), 0)$ being the central triple of \vec{E}_{i-1} , $\text{SIM}_0^{\bullet}(((), (T_1, \dots, T_{r-1}), 0)) = +T_r$. But how does the computation of $\text{SIM}_0^{\bullet}(((), (T_1, \dots, T_{r-1}), 0))$ differ from the computation of $\text{SIM}_0^{\bullet}(((), (T_1, \dots, T_t), 0))$ (from which the value (W_1, \dots, W_w) of $\text{SIM}_0^{\rightarrow}(((), (\mathbb{B}_0^h)^{odd}))$ is extracted)? Both computations proceed in exactly the same way, with the variable S of SIM_0^{\bullet} going through exactly the same values, with the only difference that, while the computation of $\text{SIM}_0^{\bullet}(((), (T_1, \dots, T_{r-1}), 0))$ stops after S takes its $(r/2)$ th value $+W_r$ and returns that value as $+T_r$, the computation of $\text{SIM}_0^{\bullet}(((), (T_1, \dots, T_t), 0))$ continues further until that value becomes $+W_w$ (if the output is positive) or $-(((), s))$ for some s (if the output is negative). Thus $W_r = T_r$, which completes our proof of claim (5.38).

Since, by (5.37), the triple $(((), (\mathbb{B}_0^h)^{odd}, 0)$ is saturated, clause 1(a) of Lemma 5.8 guarantees that there is a run Υ — let us rename it into Γ_0^h — generated by \mathcal{H}_0 such that $\overline{\text{SIM}_0^{\rightarrow}(((), (\mathbb{B}_0^h)^{odd}))} \preceq \Gamma_0^h$. This, by (5.38), implies that $\overline{\mathbb{B}_0^h} \preceq \Gamma_0^h$, as promised in clause 1(a) of the present lemma.

For clause 1(b) of the present lemma, let us additionally assume that h is the greatest element of \mathbb{I} and $\overline{\mathbb{B}_0^h}$ is a reasonable run of $F'(0)$. Note that the last, h th iteration of MAIN deals with Subsubsubcase 2.2.2.2, for any other case causes a next iteration to occur. Let $(((), B, 0)$ be the central triple of \vec{E}_h . So,

$$\text{SIM}_0(((), B^{odd}) = (-(((), v), u) \quad (5.39)$$

for some numbers v, u . Fix these numbers. By definition (5.13), B is a restriction of \mathbb{B}_0^h . And, by clause 1 of Lemma 5.14, the size of \mathbb{B}_0^h is odd. Consequently, B is not a proper restriction of \mathbb{B}_0^h , because otherwise B^{odd} would be a proper restriction of $(\mathbb{B}_0^h)^{odd}$, making the statements (5.35) and (5.39) contradictory. We thus find that $B = \mathbb{B}_0^h$, which allows us to re-write (5.39) as

$$\text{SIM}_0(((), (\mathbb{B}_0^h)^{odd}) = (-(((), v), u). \quad (5.40)$$

In view of $\text{SIM}_0^{\bullet}(((), (\mathbb{B}_0^h)^{odd})$'s being negative, one can see immediately from the definition of $\text{SIM}_0^{\rightarrow}$ that the size of $\text{SIM}_0^{\rightarrow}(((), (\mathbb{B}_0^h)^{odd})$ does not exceed the size of \mathbb{B}_0^h . This, in combination with (5.38), means that

$$\text{SIM}_0^{\rightarrow}(((), (\mathbb{B}_0^h)^{odd}) = \mathbb{B}_0^h. \quad (5.41)$$

Imagine the work of SIM_0 when computing (5.40). Taking (5.37) into account, we can see that v is just a copy of the scale of the last organ of $(\mathbb{B}_0^h)^{odd}$ and hence, by clause 1 of Lemma 5.14, of the last organ of (\mathbb{B}_0^h) . Consequently, by clause 3 of Lemma 5.14, v is the master scale of \vec{E}_h . Then, since the h th iteration of MAIN proceeds according to Subsubsubcase 2.2.2.2, we have $v \geq \mathcal{L}(\mathfrak{l}, U_h)$, where U_h is the value that the variable U of MAIN assumes on the h th iteration as a result of updating the old value to $\max(u, U)$.

We thus have $u \leq U_h$. And the function \mathfrak{L} is, of course, monotone. Consequently, from the fact $v \geq \mathfrak{L}(\mathfrak{l}, U_h)$, we find that $v \geq \mathfrak{L}(\mathfrak{l}, u)$. But then, by (5.41) and clause 1(b) of Lemma 5.8, there is a run Υ generated by \mathcal{H}_0 — let us rename it into Γ_0^h — such that $\overline{\mathbb{B}_0^h} = \Gamma_0^h$. Done.

2. Fix any $n \in \{1, \dots, k\}$, and assume

$$\begin{aligned}\mathbb{B}_{n-1}^h \downarrow &= (P_1, \dots, P_p); & \mathbb{B}_n^h \uparrow &= (Q_1, \dots, Q_q); \\ \mathbb{B}_{n-1}^h &= (P_1, \dots, P_{p'}); & \mathbb{B}_n^h &= (Q_1, \dots, Q_{q'}).\end{aligned}$$

For clause 2(a), we want to show the existence of a run Γ_n^h generated by \mathcal{H}_n such that

$$\overline{(Q_1, \dots, Q_q)} \preceq (\Gamma_n^h)^1 \quad \text{and} \quad \overline{(P_1, \dots, P_p)} \preceq (\Gamma_n^h)^0. \quad (5.42)$$

It is not hard to see that, if q is 0, then so is p , because there is no way for (P_1) to be ever $(h, n-1)$ -born. Then the runs $\overline{(P_1, \dots, P_p)}$ and $\overline{(Q_1, \dots, Q_q)}$ are empty and, therefore, any run Γ_n^h generated by \mathcal{H}_n satisfies (5.42). Now, for the rest of this proof, assume q is non-zero, which, in view of q 's being even, means that $q \geq 2$. In what follows, we use i_P to denote the $(h, n-1)$ -birthtime of (P_1, \dots, P_p) and i_Q to denote the (h, n) -birthtime of (Q_1, \dots, Q_q) . We claim that

$$i_P \neq i_Q. \quad (5.43)$$

Indeed, it is easy to see that two bodies have identical birthtimes only if they are both empty (and hence their birthtimes are both 0). However, as we have already agreed, (Q_1, \dots, Q_q) is nonempty. In view of (5.43), it is now sufficient to consider the two cases $i_Q > i_P$ and $i_P > i_Q$.

Case of $i_Q > i_P$: In this case, according to clause 1 of Lemma 5.13, the triple

$$((P_1, \dots, P_{p'}))^{even}, (Q_1, \dots, Q_{q-1})^{odd}, n$$

is saturated, and we have:

$$\begin{aligned}\text{SIM}_n^\rightarrow((P_1, \dots, P_{p'}))^{even}, (Q_1, \dots, Q_{q-1})^{odd} &= (Q_1, \dots, Q_q); \\ \text{SIM}_n^\leftarrow((P_1, \dots, P_{p'}))^{even}, (Q_1, \dots, Q_{q-1})^{odd} &= (P_1, \dots, P_{p'}).\end{aligned}$$

Therefore, by clause 2(a) of Lemma 5.8, there is a run Υ — let us rename it into Γ_n^h — generated by \mathcal{H}_n such that $\overline{(Q_1, \dots, Q_q)} \preceq (\Gamma_n^h)^1$ and $\overline{(P_1, \dots, P_{p'})} \preceq (\Gamma_n^h)^0$. Of course, $\overline{(P_1, \dots, P_{p'})} \preceq (\Gamma_n^h)^0$ implies $\overline{(P_1, \dots, P_p)} \preceq (\Gamma_n^h)^0$. So, (5.42) holds, which takes care of clause 2(a) of the present lemma. As for clause 2(b), it is satisfied vacuously because h is not the greatest element of \mathbb{I} . To see why h is not the greatest element of \mathbb{I} , assume the opposite. Let $i_{P'}$ be the $(h, n-1)$ -birthtime of $(P_1, \dots, P_{p'})$ and $i_{Q'}$ be the (h, n) -birthtime of $(Q_1, \dots, Q_{q'})$. By clause 1 of Lemma 5.14, p is odd, implying that $p' = p$ and hence $i_{P'} = i_P$. Next, the fact $q' \geq q$ obviously implies that $i_{Q'} \geq i_Q$. So, the condition $i_Q > i_P$ of the present case implies $i_{Q'} > i_{P'}$. But this is in contradiction with clause 2 of Lemma 5.14.

Case of $i_P > i_Q$: In this case, according to clause 2 of Lemma 5.13, we have:

$$\text{SIM}_n^\rightarrow((P_1, \dots, P_{p-1}))^{even}, (Q_1, \dots, Q_{q'})^{odd} = -P_p; \quad (5.44)$$

$$\text{The triple } ((P_1, \dots, P_{p-1}))^{even}, (Q_1, \dots, Q_{q'})^{odd}, n \text{ is saturated}; \quad (5.45)$$

$$\text{SIM}_n^\rightarrow((P_1, \dots, P_{p-1}))^{even}, (Q_1, \dots, Q_{q'})^{odd} = (Q_1, \dots, Q_{q'}); \quad (5.46)$$

$$\text{SIM}_n^\leftarrow((P_1, \dots, P_{p-1}))^{even}, (Q_1, \dots, Q_{q'})^{odd} = (P_1, \dots, P_p). \quad (5.47)$$

From (5.45)-(5.47), by clause 2(a) of Lemma 5.8 with $(P_1, \dots, P_{p-1})^{even}$ in the role of A and $(Q_1, \dots, Q_{q'})^{odd}$ in the role of B , there is a run Υ — let us rename it into Γ_n^h — generated by \mathcal{H}_n such that $\overline{(Q_1, \dots, Q_{q'})} \preceq (\Gamma_n^h)^{1\cdot}$ and $\neg\overline{(P_1, \dots, P_p)} \preceq (\Gamma_n^h)^{0\cdot}$. But $\overline{(Q_1, \dots, Q_{q'})} \preceq (\Gamma_n^h)^{1\cdot}$ implies $\overline{(Q_1, \dots, Q_q)} \preceq (\Gamma_n^h)^{1\cdot}$. So, (5.42) holds, which takes care of clause 2(a) of the present lemma.

For clause 2(b), let us assume moreover that h is the greatest element of \mathbb{I} , $\overline{(Q_1, \dots, Q_{q'})}$ is a reasonable run of $F'(n)$, and $\neg\overline{(P_1, \dots, P_{p'})}$ is a reasonable run of $F'(n-1)$. By clause 1 of Lemma 5.14, p' is odd, implying that $p = p'$. So, (5.44)-(5.47) can be re-written as

$$\text{SIM}_n^\bullet((P_1, \dots, P_{p'-1})^{even}, (Q_1, \dots, Q_{q'})^{odd}) = -P_{p'}; \quad (5.48)$$

$$\text{The triple } ((P_1, \dots, P_{p'-1})^{even}, (Q_1, \dots, Q_{q'})^{odd}, n) \text{ is saturated}; \quad (5.49)$$

$$\text{SIM}_n^\rightarrow((P_1, \dots, P_{p'-1})^{even}, (Q_1, \dots, Q_{q'})^{odd}) = (Q_1, \dots, Q_{q'}); \quad (5.50)$$

$$\text{SIM}_n^\leftarrow((P_1, \dots, P_{p'-1})^{even}, (Q_1, \dots, Q_{q'})^{odd}) = (P_1, \dots, P_{p'}). \quad (5.51)$$

Let $P_{p'} = (\vec{\omega}, v)$. In view of (5.48), there is a number u (fix it) such that

$$\text{SIM}_n((P_1, \dots, P_{p'-1})^{even}, (Q_1, \dots, Q_{q'})^{odd}) = (-(\vec{\omega}, v), u). \quad (5.52)$$

As observed earlier when verifying clause 2(b) of the lemma in the case of $i_Q > i_P$, we have $p = p'$, meaning that i_P is the $(h, n-1)$ -birthtime of $\mathbb{B}_{n-1}^h = (P_1, \dots, P_{p'})$. In addition, let i_L be the (h, k) -birthtime of \mathbb{B}_k^h . By clause 2 of Lemma 5.14, $i_P > i_L$. This means that, for any $j \in \{i_P, \dots, h\}$, the j th iteration of MAIN is not locking, because a locking iteration always gives birth to a new, “bigger” master body. But the absence of locking iterations between i_P and h implies the following, because otherwise i_P would be h -transient:

$$\text{For any } j \in \{i_P, \dots, h\}, \text{ the } j\text{'th iteration of MAIN is not restarting}. \quad (5.53)$$

Since h is the greatest element of \mathbb{I} , according to clause 3 of Lemma 5.14, v is the master scale of \vec{E}_h . Also, as observed earlier in the proof of clause 1(b), the h th iteration of MAIN deals with Subsubsubcase 2.2.2.2, implying that $v \geq \mathfrak{L}(\mathfrak{l}, U_h)$, where U_h is the final value of the variable U of MAIN (assumed on the h th iteration). But note that U_{i_P} — the value of U assumed on the i_P th iteration of MAIN — does not exceed U_h . That is because only restarting iterations of MAIN can decrease the value of U , but, by (5.53), there are no such iterations between i_P and h . Also, it is clear that, on the i_P th iteration, $(P_1, \dots, P_{p'})$ was born according to the scenario of Subsubcase 2.2.1 due to (5.52), implying that $U_{i_P} \geq u$, because, at the beginning of that iteration, the variable U was updated to $U_{i_P} = \max(u, U)$. Thus, $U_h \geq u$ and hence, due to the monotonicity of \mathfrak{L} and the earlier-established fact $v \geq \mathfrak{L}(\mathfrak{l}, U_h)$, we have

$$v \geq \mathfrak{L}(\mathfrak{l}, u). \quad (5.54)$$

From (5.49), (5.52), (5.50), (5.51) and (5.54), using clause 2(b) of Lemma 5.8, with $(P_1, \dots, P_{p'-1})^{even}$ in the role of A and $(Q_1, \dots, Q_{q'})^{odd}$ in the role of B , there is a run Υ — let us rename it into Γ_n^h — such that $\overline{(Q_1, \dots, Q_{q'})} = (\Gamma_n^h)^{1\cdot}$ and $\neg\overline{(P_1, \dots, P_{p'})} = (\Gamma_n^h)^{0\cdot}$, as desired. \square

Lemma 5.16. *For every $n \in \{0, \dots, k\}$ and every $h \in \mathbb{I}$, $\overline{\mathbb{B}_n^h}$ is a reasonable run of $F'(n)$.*

Proof. Fix an $n \in \{0, \dots, k\}$ and an $h \in \mathbb{I}$. Below, whenever we say that a player \wp has made — or is responsible for making — a given run unreasonable, it is to be (or, at least, can be) understood as that the last move of the shortest unreasonable initial segment of the run is \wp -labeled.

First, consider the case $n = 0$. For a contradiction, assume $\overline{\mathbb{B}_0^h}$ is not a reasonable run of $F'(0)$. By clause 1(a) of Lemma 5.15, $\overline{\mathbb{B}_0^h}$ is an initial segment of a certain run Γ_0^h generated by \mathcal{H}_0 . Therefore, in view of our assumption that \mathcal{H}_0 plays $F'(0)$ reasonably, the only way $\overline{\mathbb{B}_0^h}$ could be unreasonable is if \perp (\mathcal{H}_0 's adversary) made it so. But, according to clause 2(a) of Lemma 5.15, a certain extension $(\Gamma_1^h)^0$ of $\overline{\mathbb{B}_0^h}$ is a run generated by \mathcal{H}_1 (with \mathcal{H}_1 playing as \top) in the component $\neg F'(0)$ of $\neg F'(0) \vee F'(1)$. Therefore, as — by our assumption — $\mathcal{H}_1 = \top$ plays reasonably, \top cannot be responsible for making $\overline{\mathbb{B}_0^h}$ unreasonable. Then \top cannot be responsible for $\overline{\mathbb{B}_0^h}$'s being unreasonable either, because $\overline{\mathbb{B}_0^h}$ differs from $\overline{\mathbb{B}_0^h} \downarrow$ only in that the former perhaps has some additional \perp -labeled moves at the end. Contradiction.

Next, consider the case $0 < n < k$. It is rather similar to the preceding one. For a contradiction, assume $\overline{\mathbb{B}_n^h}$ is not a reasonable run of $F'(n)$. By clause 2(a) of Lemma 5.15, there is a run Γ_n^h generated by \mathcal{H}_n such that $\overline{\mathbb{B}_n^h} \uparrow$ is an initial segment of $(\Gamma_n^h)^1$. $\overline{\mathbb{B}_n^h}$ only differs from $\overline{\mathbb{B}_n^h} \uparrow$ in that the former perhaps has some additional \perp -labeled moves at the end. For this reason, as \mathcal{H}_n plays $F'(n-1) \rightarrow F'(n)$ reasonably, the only way $\overline{\mathbb{B}_n^h}$ could be unreasonable is if \perp (\mathcal{H}_n 's adversary) made it so. Then $\overline{\mathbb{B}_n^h}$ is an unreasonable run of $\neg F'(n)$, with player \top being responsible for making it so. But, (again) according to clause 2(a) of Lemma 5.15, a certain extension $(\Gamma_{n+1}^h)^0$ of $\overline{\mathbb{B}_n^h} \downarrow$ is a run generated by \mathcal{H}_{n+1} in the component $\neg F'(n)$ of $\neg F'(n) \vee F'(n+1)$. As $\mathcal{H}_{n+1} = \top$ plays this game reasonably, it cannot be responsible for making $\overline{\mathbb{B}_n^h} \downarrow$ an unreasonable run of $\neg F'(n)$. Then \top cannot be responsible for making $\overline{\mathbb{B}_n^h}$ unreasonable either, because $\overline{\mathbb{B}_n^h}$ only differs from $\overline{\mathbb{B}_n^h} \downarrow$ in that the former perhaps has some additional \perp -labeled moves at the end. Contradiction.

Finally, consider the case $n = k$. Just as in the preceding cases, \mathcal{H}_k cannot be responsible for making $\overline{\mathbb{B}_k^h}$ an unreasonable run of $F'(k)$. Looking at Case 1, Subsubcase 2.1.2 and Subsubsubcase 2.2.2.2 of the description of MAIN, it is clear that \mathcal{H}_k 's imaginary adversary does not make $\overline{\mathbb{B}_k^h}$ unreasonable either. This is so because, in $F'(k)$, MAIN lets \mathcal{H}_k 's adversary mimic \mathcal{M}_k 's real environment's play. The latter, by the Clean Environment Assumption, plays (legally and hence) quasilegally. And even if it does not play prudently, MAIN prudentizes \mathcal{M}_k 's environment's moves before copying and adding them to $\overline{\mathbb{B}_k^h}$ as \mathcal{H}_k 's imaginary adversary's moves. \square

Recall that \hbar is the greatest element of \mathbb{I} .

Lemma 5.17. *For every $n \in \{0, \dots, k\}$, $\overline{\mathbb{B}_n^h}$ is a \top -won run of $F'(n)$.*

Proof. Induction on n . According to clause 1(b) of Lemma 5.15, in conjunction with Lemma 5.16, $\overline{\mathbb{B}_0^h}$ is a run generated by \mathcal{H}_0 . So, since \mathcal{H}_0 wins $F'(0)$, $\overline{\mathbb{B}_0^h}$ is a \top -won run of $F'(0)$.

Next, consider any n with $0 < n \leq k$. According to clause 2(b) of Lemma 5.15, in conjunction with Lemma 5.16, there is a run Γ_n^h generated by \mathcal{H}_n such that $(\Gamma_n^h)^0 = \overline{\mathbb{B}_{n-1}^h}$ and $(\Gamma_n^h)^1 = \overline{\mathbb{B}_n^h}$. Note that, since \mathcal{H}_n plays quasilegally, every move of Γ_n^h has one of the

two prefixes “0.” or “1.”. But we know that \mathcal{H}_n wins $\neg F'(n-1) \vee F'(n)$. So, Γ_n^h has to be a \top -won run of $\neg F'(n-1) \vee F'(n)$, meaning that either $(\Gamma_n^h)^0$, i.e. $\neg \overline{\mathbb{B}_{n-1}^h}$, is a \top -won run of $\neg F'(n-1)$, or $(\Gamma_n^h)^1$, i.e. $\overline{\mathbb{B}_n^h}$, is a \top -won run of $F'(n)$. But, by the induction hypothesis, $\overline{\mathbb{B}_{n-1}^h}$ is a \top -won run of $F'(n-1)$. This obviously means that $\neg \overline{\mathbb{B}_{n-1}^h}$ is a \perp -won (and thus not \top -won) run of $\neg F'(n-1)$. Therefore, $\overline{\mathbb{B}_n^h}$ is a \top -won run of $F'(n)$. \square

According to Lemma 5.17, $\overline{\mathbb{B}_k^h}$ is a \top -won run of $F'(k)$. Therefore, by the known property of static games and delays (see the end of Section 3 of [3]) we have:

$$\text{Whenever a run } \Pi \text{ is a } \top\text{-delay of } \overline{\mathbb{B}_k^h}, \Pi \text{ is a } \top\text{-won run of } F'(k). \quad (5.55)$$

Let Θ be the run generated by \mathcal{M}_k that took place in the real play of $k \leq \mathbf{b}|\vec{d}| \rightarrow F'(k)$. How does Θ^1 relate to $\overline{\mathbb{B}_k^h}$? As promised earlier, the real play in the consequent of $k \leq \mathbf{b}|\vec{d}| \rightarrow F'(k)$ — that is, the run Θ^1 — would be “essentially synchronized” with the play $\overline{\mathbb{B}_k^h}$ by \mathcal{H}_k in the consequent of $F'(k-1) \rightarrow F'(k)$, meaning that Θ^1 is “essentially the same” as $\overline{\mathbb{B}_k^h}$. The qualification “essentially” implies that the two runs, while being similar, may not necessarily be strictly identical.

One reason why $\overline{\mathbb{B}_k^h}$ may differ from Θ^1 is that, as seen from Case 1 and Subsubsubcase 2.2.2.2 of the description of MAIN, if Θ^1 contains a (legal but) imprudent (with respect to $F'(k)$) move by \perp , then such a move appears in $\overline{\mathbb{B}_k^h}$ in the prudentized form. Namely, if \mathcal{H}_k ’s adversary chose some “oversized” constant a for z in a subcomponent $\Box z G$ of $F'(k)$, then the same move will appear in $\overline{\mathbb{B}_k^h}$ as if a' was chosen instead of a , where a' is a certain “small” constant. Note, however, that having made the above imprudent choice makes \perp lose in the $\Box z G$ component. So, prudentizing \perp ’s imprudent moves can only increase rather than decrease \perp ’s chances to win the overall game. That is, if \top (i.e., \mathcal{M}_k) wins the game even after such moderation of the adversary’s imprudent moves, it would also win (“even more so”) without moderation. For this reason, we can and will safely assume that \mathcal{M}_k ’s environment plays not only legally, but also prudently.

But even if \mathcal{M}_k ’s adversary has played Θ^1 prudently, there is another reason that could make $\overline{\mathbb{B}_k^h}$ “somewhat” different from Θ^1 . Namely, with some thought, one can see that Θ^1 may be a proper \top -delay of (rather than equal to) $\overline{\mathbb{B}_k^h}$. Luckily, however, by (5.55), Θ^1 is still a \top -won run of $F'(k)$.

Thus, as desired, \mathcal{M}_k wins $k \leq \mathbf{b}|\vec{d}| \rightarrow F'(k)$, and hence \mathcal{M} wins the conclusion of (5.2).

5.6. \mathcal{M} runs in target amplitude. \mathcal{M} plays $x \leq \mathbf{b}|\vec{s}| \rightarrow F(x, \vec{v})$ prudently, and the latter is an \mathcal{R}_{space} -bounded formula. By condition 5 of Definition 2.2 of [5], $\mathcal{R}_{space} \preceq \mathcal{R}_{amplitude}$. This, of course, implies that \mathcal{M} runs in amplitude $\mathcal{R}_{amplitude}$, as desired.

5.7. \mathcal{M} runs in target space. As we agreed earlier, $(\mathbf{a}, \mathbf{s}, \mathbf{t}) \in \mathcal{R}_{amplitude} \times \mathcal{R}_{space} \times \mathcal{R}_{time}$ is a common tricomplexity in which the machines \mathcal{N} and \mathcal{K} — and hence the \mathcal{H}_n s — run. All three bounds are unary.

Remember from Section 5.1 that \mathfrak{l} is the size of the greatest of the constants chosen by \mathcal{M} ’s environment for the free variables of $x \leq \mathbf{b}|\vec{s}| \rightarrow F(x, \vec{v})$. This, of course, means that the background of any clock cycle of \mathcal{M}_k in any scenario of its work is at least \mathfrak{l} . For this reason and with Remark 2.4 of [5] in mind, in order to show that \mathcal{M} runs in space \mathcal{R}_{space} ,

it is sufficient to show that the spacecost of any clock cycle of \mathcal{M}_k is bounded by $O(\mathfrak{p}(\mathfrak{l}))$ for some $\mathfrak{p}(z) \in \mathcal{R}_{space}$. In what follows, we shall write $\mathcal{R}_{space}(\mathfrak{l})$ as an abbreviation of the phrase “ $O(\mathfrak{p}(\mathfrak{l}))$ for some $\mathfrak{p}(z) \in \mathcal{R}_{space}$ ”. Similarly for $\mathcal{R}_{time}(\mathfrak{l})$.

In asymptotic terms, the space consumed by \mathcal{M}_k — namely, by any given h th ($h \in \mathbb{I}$) iteration of MAIN — is the sum of the following two quantities:

$$\text{the space needed to hold the value of the aggregation } \vec{E}; \quad (5.56)$$

$$\text{the space needed to update } \vec{E} = \vec{E}_h \text{ to } \vec{E} = \vec{E}_{h+1} \text{ (if } (h+1) \in \mathbb{I}). \quad (5.57)$$

Here we did not mention the space needed to hold the value of the variable U , and to update it to its next value. That is because, as it is easy to see, the space taken by U or its updates does not exceed the maximum possible value of the quantity (5.57) (in fact, the logarithm of the latter). So, this component of \mathcal{M}_k 's space consumption, being superseded by another component, can be safely ignored in an asymptotic analysis. Consider any $h \in \mathbb{I}$.

In verifying that (5.56) is bounded by $\mathcal{R}_{space}(\mathfrak{l})$, we observe that, by conditions (iv) and (v) of Section 5.3, an aggregation cannot contain two same-size entries. Next, by Lemma 5.10, the size of an entry never exceeds $2\mathfrak{e}_T + 1$. Thus, the number of entries in \vec{E}_h is bounded by the constant $2\mathfrak{e}_T + 1$. For this reason, it is sufficient for us to just show that any given entry $[n, C]$ of \vec{E}_h can be held with $\mathcal{R}_{space}(\mathfrak{l})$ space. This is done in the following two paragraphs.

The component n of an entry $[n, C]$ never exceeds k . As observed in the proof of Lemma 5.11, we have $k \leq f(\mathfrak{l})$, where $f(z)$ is the unarification of \mathfrak{b} . As further observed near the end of the same proof, $f(z) \preceq \mathcal{R}_{time}$. This, by condition 5 of Definition 2.2 of [5], implies that $|f(z)| \preceq \mathcal{R}_{space}$. So, $|n|$, which asymptotically is the amount of space needed to hold n , is bounded by $\mathcal{R}_{space}(\mathfrak{l})$.

As for the component C of an entry $[n, C]$, it is a restriction of (and hence not “bigger” than) $\underline{\mathbb{B}}_n^h$, so let us talk about $\underline{\mathbb{B}}_n^h$ instead. Let $\underline{\mathbb{B}}_n^h = ((\vec{\alpha}_1, p_1), \dots, (\vec{\alpha}_m, p_m))$. By Lemma 5.16, $\underline{\mathbb{B}}_n^h$ is a reasonable run of $F'(n)$. Consequently, the overall number of moves in it cannot exceed the constant bound \mathfrak{e} . Remembering the work of SIM_n^\bullet , we see that only negative values of this procedure's output may have empty payloads. With this fact in mind, a look back at the work of MAIN — its Subcase 2.1 in particular — easily reveals that, for each even $i \in \{2, \dots, m\}$, $\vec{\alpha}_i$ is nonempty. Therefore $m \leq 2\mathfrak{e} + 1$. That is, the number of organs in $\underline{\mathbb{B}}_n^h$ is bounded by a constant. So, asymptotically, $\underline{\mathbb{B}}_n^h$ does not take more space than any organ $(\vec{\alpha}_i, p_i)$ of it, which allows us now to just focus on $(\vec{\alpha}_i, p_i)$. Due to $\underline{\mathbb{B}}_n^h$'s being reasonable, there is only a constant ($\leq \mathfrak{e}$) number of moves in the payload $\vec{\alpha}_i$ of $(\vec{\alpha}_i, p_i)$, and the size of no such move exceeds $O(\mathfrak{G}(\mathfrak{l}))$, where \mathfrak{G} , as we remember, is the superaggregate bound of the formula $F(x, \vec{v})$ and hence, by Lemma 4.1, $\mathfrak{G} \preceq \mathcal{R}_{space}$. Thus, $\mathcal{R}_{space}(\mathfrak{l})$ space is sufficient to record $\vec{\alpha}_i$. It now remains to show that the same holds for p_i . An analysis of MAIN reveals that, during its work, the only case when a new scale (as opposed to an old, inherited scale) greater than 1 of whatever organ of whatever entry is ever created is Subsubsubcase 2.2.2.1, and when such a creation takes place, the new scale is smaller than $2\mathfrak{L}(\mathfrak{l}, U)$. As observed earlier in this proof when we agreed to ignore U , the value of U is bounded by $\mathfrak{s}'(\mathfrak{l})$ for some $\mathfrak{s}' \in \mathcal{R}_{space}$. So, $p_i < 2\mathfrak{L}(\mathfrak{l}, \mathfrak{s}'(\mathfrak{l}))$ and thus $|p_i| \leq |2\mathfrak{L}(\mathfrak{l}, \mathfrak{s}'(\mathfrak{l}))|$. In view of our earlier observation (5.9), $|2\mathfrak{L}(\mathfrak{l}, \mathfrak{s}'(\mathfrak{l}))| = O(|\mathfrak{l}| + |\mathfrak{G}(\mathfrak{l})| + \mathfrak{s}'(\mathfrak{l}))$. This fact, in conjunction with $\mathfrak{G} \in \mathcal{R}_{space}$ and condition 2 of Definition 2.2 of [5], implies that $|p_i|$, which asymptotically is the amount of memory needed to hold p_i , does not exceed $\mathcal{R}_{space}(\mathfrak{l})$.

Now about the quantity (5.57). Let us only consider the case $n > 0$, with the case $n = 0$ being similar but simpler. Updating \vec{E}_h to \vec{E}_{h+1} happens through running $\text{SIM}_n(A^{\text{even}}, B^{\text{odd}})$, where (A, B, n) is the central triple of \vec{E}_h . So, we just need to show that the space consumed by $\text{SIM}_n(A^{\text{even}}, B^{\text{odd}})$ is bounded by $\mathcal{R}_{\text{space}}(\mathfrak{l})$. This quantity, with asymptotically irrelevant technicalities suppressed, is the sum of (I) the space needed for simulating \mathcal{H}_n , and (II) the space needed for maintaining (the contents of) the variables $a, b, u, \vec{\psi}, \vec{\nu}, W, S, R$ of SIM_n , as well as the space needed to keep track of how many steps of \mathcal{H}_n have been simulated so far within the present iteration of LOOP_n .

(I): There are two groups of moves on the simulated \mathcal{H}_n 's run tape. The first group, that we here shall refer to as the **early moves**, comprises the \perp -labeled moves signifying the initial choices of the constants $n-1$ and \vec{c} for the free variables x and \vec{v} of $F(x, \vec{v}) \rightarrow F(x', \vec{v})$. All other moves constitute the second group, which we shall refer to as the **late moves**. The information that \mathcal{M}_k needs to keep track of (and whose size is asymptotically relevant) in order to simulate \mathcal{H}_n consists of the contents (here also including the scanning head locations) of \mathcal{H}_n 's run and work tapes, and the content of \mathcal{H}_n 's buffer. Since (A, B, n) is the central triple of \vec{E}_h , A is a restriction of \mathbb{B}_{n-1}^h and B is a restriction of \mathbb{B}_n^h . This, in view of Lemma 5.16 and in view of \mathcal{H}_n 's playing reasonably, obviously implies that the run spelled by the simulated \mathcal{H}_n 's run tape is reasonable. As a result, there is only a constant number of late moves, and the magnitude of each such move is obviously bounded by $\mathfrak{G}(\mathfrak{l})$. In view of Lemma 4.1, this means that the combined size of all late moves is bounded by $\mathcal{R}_{\text{space}}(\mathfrak{l})$. Since \mathcal{H}_n is unconditionally provident, everything written in its buffer will sooner or later mature into a late move, so, whatever we said about the sizes of the late moves, also applies to the maximum possible size of \mathcal{H}_n 's buffer content. As for the early moves, they reside on \mathcal{M}_k 's own run tape, and hence \mathcal{M}_k does not need to utilize any of its work-tape space to keep track of them. Thus, keeping track of the contents of \mathcal{H}_n 's imaginary run tape and buffer does not take \mathcal{M}_k beyond the target $\mathcal{R}_{\text{space}}(\mathfrak{l})$ space limits. It remains to see that the same holds for the contents of \mathcal{H}_n 's work tapes. But indeed, the magnitude of no (early or late) move on \mathcal{H}_n 's imaginary run tape exceeds $\max(\mathfrak{l}, \mathfrak{G}(\mathfrak{l}))$ and hence (as $\mathcal{R}_{\text{amplitude}}$ is linearly closed and $\mathfrak{G} \in \mathcal{R}_{\text{space}} \preceq \mathcal{R}_{\text{amplitude}}$) $\mathfrak{a}'(\mathfrak{l})$ for some $\mathfrak{a}' \in \mathcal{R}_{\text{amplitude}}$. But then, since \mathcal{H}_n runs in unconditional space $\mathfrak{s} \in \mathcal{R}_{\text{space}}$, it consumes at most $\mathfrak{s}(\mathfrak{a}'(\mathfrak{l}))$ space of its work tapes. \mathcal{M}_k can keep track of the contents of those tapes with asymptotically the same amount $\mathfrak{s}(\mathfrak{a}'(\mathfrak{l}))$ of its own work-tape space. And the latter, by condition 4 of Definition 2.2 of [5], does not exceed $\mathcal{R}_{\text{space}}(\mathfrak{l})$.

(II): The sizes of the variables a and b of SIM are bounded by a constant (namely, $|2\mathfrak{e} + 1|$). As for the sizes of the remaining variables $u, \vec{\psi}, \vec{\nu}, W, S, R$, as well as the space needed to keep track of how many steps of \mathcal{H}_n have been simulated so far within the present iteration of LOOP_n , can be easily seen to be superseded by (5.56) or (I).

5.8. \mathcal{M} runs in target time. We agree that, for an $h \in \mathbb{I}$, \mathbb{I}_{\bullet}^h denotes the set of all numbers $i \in \mathbb{I}^h$ satisfying the condition that there is no j with $i \leq j < h$ such that the j th iteration of MAIN proceeds according to the scenario of Case 1 or Subsubsubcase 2.2.2.2. Next, $\mathbb{I}_{\bullet\bullet}^h$ denotes the set of all numbers $i \in \mathbb{I}_{\bullet}^h$ (additionally) satisfying the condition that there is no j with $i \leq j < h$ such that the j th iteration of MAIN proceeds according to the scenario of Subsubsubcase 2.2.2.1. Finally, $\mathbb{I}_{\bullet\bullet\bullet}^h$ denotes the set of all numbers $i \in \mathbb{I}_{\bullet\bullet}^h$ (additionally) satisfying the condition that there is no j with $i \leq j < h$ such that the j th iteration of MAIN proceeds according to the scenario of Subsubsubcase 2.1.2.

Lemma 5.18. *Consider any $h \in \mathbb{I}$ such that the h 'th iteration of MAIN is locking. Then the master scale of \vec{E}_h is bounded by $\mathcal{R}_{\text{time}}(\mathfrak{l})$.*

Proof. Throughout this proof, \mathfrak{w} will be an abbreviation of the constant $\mathfrak{e}_{\top} + 1$. Consider any $h \in \mathbb{I}$ such that the h th iteration of MAIN is locking. Let m be the master scale of \vec{E}_h . We claim that m is smaller than $2^{\mathfrak{w}-1}\mathfrak{t}(\max(\mathfrak{l}, \mathfrak{G}(\mathfrak{l})))$ and hence, by Lemma 4.1 and conditions 2, 3 and 4 of Definition 2.2 of [5], m is bounded by $\mathcal{R}_{\text{time}}(\mathfrak{l})$. Indeed, for a contradiction, assume $m \geq 2^{\mathfrak{w}-1}\mathfrak{t}(\max(\mathfrak{l}, \mathfrak{G}(\mathfrak{l})))$. We (may) additionally assume that $\mathfrak{t}(\max(\mathfrak{l}, \mathfrak{G}(\mathfrak{l}))) \neq 0$. Let b_1 be the smallest element of $\mathbb{I}_{\bullet\bullet}^h$. So, there are no restarting iterations between b_1 (including) and h (not including). But only restarting iterations of MAIN modify the master scale of \vec{E} . Thus, the master scale of \vec{E}_{b_1} is the same m as that of \vec{E}_h . Since $m > 1$ and b_1 is the smallest element of $\mathbb{I}_{\bullet\bullet}^h$, the $(b_1 - 1)$ th iteration of MAIN (exists and) is restarting. Besides, that iteration does not proceed by the scenario of Case 1 or Subsubsubcase 2.2.2.2 of MAIN, because in either case the master scale of the resulting aggregation \vec{E}_{b_1} would be reset to 1. Hence, the $(b_1 - 1)$ th iteration of MAIN proceeds according to the scenario of (the master-scale-doubling) Subsubsubcase 2.2.2.1. This means that the master scale of \vec{E}_{b_1-1} is $m/2$. Let b_2 be the smallest element of $\mathbb{I}_{\bullet\bullet}^{b_1-1}$. Reasoning as above but this time with b_2 and $b_1 - 1$ instead of b_1 and h , respectively, we find that the master scale of \vec{E}_{b_2} is $m/2$ and the master scale of \vec{E}_{b_2-1} is $m/4$. Continuing this pattern, we further define $b_3 > b_4 > \dots > b_{\mathfrak{w}}$ in the same style as we defined b_1, b_2 and find that the master scales of $\vec{E}_{b_3}, \vec{E}_{b_3-1}, \vec{E}_{b_4}, \vec{E}_{b_4-1}, \dots, \vec{E}_{b_{\mathfrak{w}}}, \vec{E}_{b_{\mathfrak{w}}-1}$ are $m/4, m/8, m/8, m/16, \dots, m/2^{\mathfrak{w}-1}, m/2^{\mathfrak{w}}$, respectively. Each iteration of MAIN that proceeds according to Subsubcase 2.1.2 results in \mathcal{M}_k making a move in the real play of $h \leq \mathfrak{b}|\vec{d}| \rightarrow F'(k)$. Since \mathcal{M}_k plays (quasi)legally, altogether it makes fewer than \mathfrak{w} moves. This means that, altogether, there are fewer than \mathfrak{w} iterations of MAIN that proceed according to Subsubcase 2.1.2. Besides, one of such iterations is the h th iteration. Therefore there is at least one i with $1 \leq i < \mathfrak{w}$ such that $\mathbb{I}_{\bullet\bullet}^{b_i-1} = \mathbb{I}_{\bullet\bullet}^{b_{i+1}-1}$ and hence $b_{i+1} \in \mathbb{I}_{\bullet\bullet}^{b_i-1}$. Pick the smallest such i (fix it!), and let us rename b_i into c and b_{i+1} into a . Further, let d be the smallest element of \mathbb{I}^h such that $c \leq d$ and the d th iteration of MAIN is locking. It is not hard to see that such a d exists. Namely, $d \in \{b_1, \dots, h\}$ if $i = 1$ and $d \in \{b_i, \dots, b_{i-1} - 1\}$ if $1 < i < \mathfrak{w}$.

In what follows, we shall say that two organs $(\vec{\alpha}, p)$ and $(\vec{\beta}, q)$ are **essentially the same** iff $\vec{\alpha} = \vec{\beta}$ and either $p = q$ or $p, q \in \{m/2^i, m/2^{i-1}\}$, where i is as above. This extends to all pairs X, Y of organ-containing objects/structures (such as signed organs, bodies, aggregations, etc.) by stipulating that X and Y are essentially the same iff they only differ from each other in that where X has an organ P , Y may have an essentially the same organ Q instead. For instance, two signed organs are essentially the same iff they are both in $\{+P, +Q\}$ or both in $\{-P, -Q\}$ for some essentially the same organs P and Q ; two bodies (P_1, \dots, P_s) and (Q_1, \dots, Q_t) are essentially the same iff $s = t$ and, for each $r \in \{1, \dots, s\}$, the organs P_r and Q_r are essentially the same; etc.

Claim 2. For any $j \in \{0, \dots, d - c + 1\}$, the aggregations \vec{E}_{a+j} and \vec{E}_{c+j} are essentially the same.

This claim can be proven by induction on j . We give an outline of such a proof, leaving more elaborate details to the reader. For the basis of induction, we want to show that the aggregations \vec{E}_a and \vec{E}_c are essentially the same. To see that this is so, observe that the master entry is the only entry of both aggregations. Also, the only iteration of MAIN between a (including) and c that modifies the master entry of \vec{E} is the $(c - 1)$ th iteration,

which proceeds according to Subsubsubcase 2.2.2.1 and the only change that it makes in the master body of \vec{E} is that it doubles \vec{E} 's master scale $m/2^i$, turning it into $m/2^{i-1}$.

For the inductive step, consider any $j \in \{0, \dots, d - c\}$ and make the following observations. Updating \vec{E}_{c+j} to \vec{E}_{c+j+1} happens through running SIM_n^\bullet ¹³ (for a certain n) on certain arguments A, B . The same is the case with updating \vec{E}_{a+j} to \vec{E}_{a+j+1} , where, by the induction hypothesis, the arguments A' and B' on which \mathcal{H}_n is run are essentially the same as A and B , respectively. So, the only difference between the two computations $\text{SIM}_n^\bullet(A, B)$ and $\text{SIM}_n^\bullet(A', B')$ is that, occasionally, one traces $m/2^{i-1}$ steps of \mathcal{H}_n 's work beginning from a certain configuration W while the other only traces $m/2^i$ steps in otherwise virtually the same scenario. This guarantees that the outcomes of the two computations — and hence the ways \vec{E}_{c+j} and \vec{E}_{a+j} are updated to their next values — are essentially the same. The point is that, since \mathcal{H}_n runs in time t and since — as observed near the end of the preceding subsection — the magnitude of no move on the simulated \mathcal{H}_n 's run tape exceeds $\max(\mathbf{l}, \mathbf{G}(\mathbf{l}))$, all moves that \mathcal{H}_n makes within $m/2^{i-1} \geq 2t(\max(\mathbf{l}, \mathbf{G}(\mathbf{l})))$ Steps are in fact made within the first $m/2^i \geq t(\max(\mathbf{l}, \mathbf{G}(\mathbf{l})))$ steps of the simulated interval, so the computations of $\text{SIM}_n^\bullet(A, B)$ and $\text{SIM}_n^\bullet(A', B')$ proceed in “essentially the same” ways, yielding essentially the same outcomes.

Taking $j = d - c + 1$, Claim 2 tells us that the master body of $\vec{E}_{c+(d-c+1)}$ — i.e., of \vec{E}_{d+1} — and the master body of $\vec{E}_{a+(d-c+1)}$ are essentially the same. This is however a contradiction, because the size of the former, as a result of the d th iterations' being locking, is greater than the size of the master body of any earlier aggregations $\vec{E}_1, \dots, \vec{E}_d$. \square

Lemma 5.19. *Consider any $h \in \mathbb{I}$ such that the h 'th iteration of MAIN is locking. Assume $e \in \mathbb{I}_\bullet^h$, and (A, B, n) is the central triple of \vec{E}_e . Then the scales of all organs of A and B are bounded by $\mathcal{R}_{\text{time}}(\mathbf{l})$.*

Proof. Assume h is an element of \mathbb{I} such that the h th iteration of MAIN is locking. Let C be the master body of \vec{E}_h . It is not hard to see (by induction on $e - e_0$, where e_0 is the smallest element of \mathbb{I}_\bullet^h) that, for any $e \in \mathbb{I}_\bullet^h$, the scale of any organ of the body of any entry of \vec{E}_e is either the same as the scale of one of the organs of C , or one half, or one quarter, or... of such a scale. Thus, the scales of the organs of C (at least, the greatest of such scales) are not smaller than the scales of the organs of the entries of \vec{E}_e , including the scales of the organs of A and B . For this reason, it is sufficient to prove that the scales of all organs of C are bounded by $\mathcal{R}_{\text{time}}(\mathbf{l})$.

Let $C = (O_1, \dots, O_{2m}, O_{2m+1})$, and let $p_1, \dots, p_{2m}, p_{2m+1}$ be the corresponding scales. Note that, since the h th iteration of MAIN is locking, we have $h \in \mathbb{I}_!^h$ and, consequently, C is a restriction of \mathbb{B}_k^h . Therefore, according to Claim 1 from the proof of Lemma 5.14, we have $p_1 = p_2, p_3 = p_4, \dots, p_{2m-1} = p_{2m}$. So, it is sufficient to consider p_i where i is an odd member of $\{1, \dots, 2m+1\}$. The case of $i = 2m+1$ is immediately taken care of by Lemma 5.18. Now consider any odd member i of $\{1, \dots, 2m-1\}$. Let j be the (h, k) -birthtime of (O_1, \dots, O_{i+1}) . Thus, the $(j-1)$ th iteration of MAIN is locking. But note that p_i is the master scale of \vec{E}_{j-1} . Then, according to Lemma 5.18, p_i is bounded by $\mathcal{R}_{\text{time}}(\mathbf{l})$. \square

¹³Of course, MAIN runs SIM_n rather than SIM_n^\bullet , but this is only relevant to the value of the variable U of MAIN. The latter may only become relevant to the way the variable \vec{E} is updated when a given iteration of MAIN proceeds according to Subsubcase 2.2.2. However, no iterations between (including) c and d proceed according to that Subsubcase. So, it is safe to talk about SIM_n^\bullet instead of SIM_n here.

Now we are ready to argue that \mathcal{M} runs in target time. We already know from Lemma 5.11 that, for a certain bound $\mathfrak{z} \in \mathcal{R}_{\text{time}}$, MAIN is iterated only $\mathfrak{z}(l)$ times. In view of $\mathcal{R}_{\text{time}}$'s being at least polynomial as well as polynomially closed, it remains to see that each relevant iteration takes a polynomial (in l) amount of time. Here “relevant” means an iteration that is followed (either within the same iteration or in some later iteration) by an \mathcal{M}_k -made move without meanwhile being interrupted by Environment's moves. In other words, this is an e th iteration with $e \in \mathbb{I}_{\bullet}^h$ for some $h \in \mathbb{I}$ such that the h th iteration of MAIN is locking. Consider any such e .

There are two asymptotically relevant sources/causes of the time consumption of the e th (as well as any other) iteration of MAIN: running $\text{SIM}_n(A^{\text{even}}, B^{\text{odd}})$, where (A, B, n) is the central triple of \vec{E}_e , and periodically polling \mathcal{M}_k 's run tape to see if Environment has made any new moves.

Running $\text{SIM}_n(A^{\text{even}}, B^{\text{odd}})$ requires simulating the corresponding machine \mathcal{H}_n in the scenario determined by A^{even} and B^{odd} . With asymptotically irrelevant or superseded details suppressed, simulating a single step of \mathcal{H}_n requires going, a constant number of times, through \mathcal{M}_k 's work and run tapes to collect the information necessary for updating \mathcal{H}_n 's “current” configuration to the next one, and to actually make such an update. As we already know from Section 5.7, the size of (the non-blank, to-be-scanned portion of) \mathcal{M}_k 's work tape is bounded by $\mathcal{R}_{\text{space}}$.¹⁴ And the size of \mathcal{M}_k 's run tape is $O(l)$ (the early moves) plus $O(\mathfrak{G}(l))$ (the late moves). Everything together, in view of the linear closure of $\mathcal{R}_{\text{time}}$ (condition 3 of Definition 2.2 of [5]) and the facts $\mathfrak{G} \in \mathcal{R}_{\text{space}}$ (Lemma 4.1) and $\mathcal{R}_{\text{space}} \preceq \mathcal{R}_{\text{time}}$ (condition 5 of Definition 2.2 of [5]), is well within the target $\mathcal{R}_{\text{time}}(l)$.

The amount of steps of \mathcal{H}_n to be simulated when running $\text{SIM}_n(A^{\text{even}}, B^{\text{odd}})$ is obviously at most a constant times the greatest of the scales of the organs of A and B , which, in view of Lemma 5.19, is $\mathcal{R}_{\text{time}}(l)$.

Thus, the time T needed for running $\text{SIM}_n(A^{\text{even}}, B^{\text{odd}})$ is the product of the two $\mathcal{R}_{\text{time}}(l)$ quantities established in the preceding two paragraphs. By the polynomial closure of $\mathcal{R}_{\text{time}}$, such a product remains $\mathcal{R}_{\text{time}}(l)$. How much time is added to this by the polling routine? Obviously the latter is repeated at most T times. Any given repetition does not require more time than it takes to go from one end of the run tape of \mathcal{M}_k to the other end. And this quantity, as we found just a while ago, is $\mathcal{R}_{\text{time}}(l)$. Thus, the e th iteration of MAIN takes $\mathcal{R}_{\text{time}}(l) + \mathcal{R}_{\text{time}}(l) \times \mathcal{R}_{\text{time}}(l)$ time, which, by $\mathcal{R}_{\text{time}}$'s being polynomially closed, remains $\mathcal{R}_{\text{time}}(l)$ as promised.

6. FINAL REMARKS

In writing this paper and its predecessor, the author has tried to keep balance between generality and simplicity, often sacrificing the former for the sake of the latter. Among the ways that the present results could be strengthened is relaxing the concept of a regular theory $\mathbf{CLA11}_{\mathcal{A}}^{\mathcal{R}}$. Specifically, the condition of $\mathcal{R}_{\text{amplitude}}$'s being linearly closed can be removed as long as Definition 2.2 of [5] is correspondingly refined/readjusted. This condition, in fact, amounts to adopting an asymptotic view of amplitude complexity, which significantly simplifies the completeness proofs, allowing us to avoid numerous annoying exceptions and details one would need to otherwise deal with. As noted in [3], however, unlike time and

¹⁴Of course, a tape is infinite in the rightward direction, but in contexts like the present one we treat the leftmost blank cell of a tape as its “end”.

space complexities, we may not always be willing to — and it is almost never really necessary to — settle for merely asymptotic analysis when it comes to amplitude complexity. A non-asymptotic approach to amplitude complexity would make it possible to consider much finer amplitude complexities, such as “strictly ℓ ” (“non-size-increasing”, as studied in [1]), “ ℓ plus a constant”, etc.

APPENDIX A. PROOF OF LEMMA 4.4

Lemma 4.4 states:

There is an effective procedure that takes an arbitrary bounded formula $H(\vec{y})$, an arbitrary HPM \mathcal{L} and constructs an HPM \mathcal{M} such that, as long as \mathcal{L} is a provident solution of $H(\vec{y})$, the following conditions are satisfied:

1. \mathcal{M} is a quasilegal and unconditionally provident solution of $H(\vec{y})$.
2. If \mathcal{L} plays $H(\vec{y})$ prudently, then \mathcal{M} plays $H(\vec{y})$ unconditionally prudently.
3. For any arithmetical functions $\mathfrak{a}, \mathfrak{s}, \mathfrak{t}$, if \mathcal{L} plays $H(\vec{y})$ in tricomplexity $(\mathfrak{a}, \mathfrak{s}, \mathfrak{t})$, then \mathcal{M} plays in unconditional tricomplexity $(\mathfrak{a}, \mathfrak{s}, \mathfrak{t})$.

Consider an arbitrary HPM \mathcal{L} and an arbitrary bounded formula $H(\vec{y})$ with all free variables displayed. We want to (show how to) construct an HPM \mathcal{M} — with the same number of work tapes as \mathcal{L} — satisfying the above conditions 1-3. From our construction of \mathcal{M} it will be immediately clear that \mathcal{M} is built effectively from $H(\vec{y})$ and \mathcal{L} . As usual, we may not always be very careful about the distinction between $H(\vec{y})$ and $\Box H(\vec{y})$, but which of these two is really meant can always easily be seen from the context.

We agree on the following terminology. A **semiposition** is a string S of the form $\wp_1\alpha_1\dots\wp_n\alpha_n\omega$, where each \wp_i is a label \top or \perp , each α_i is a string over the keyboard alphabet, and $\omega \in \{\epsilon, \text{BLANK}\}$ (remember that ϵ stands for the empty string). When ω is **BLANK**, we say that S is **complete**; otherwise S is **incomplete**. We say that a semiposition S' is a **completion** of S iff (1) either S is complete and $S' = S$, or (2) S is incomplete and $S' = S\beta_{\text{BLANK}}$ for some (possibly empty) string β over the keyboard alphabet. When S is complete — namely, is $\wp_1\alpha_1\dots\wp_n\alpha_n\text{BLANK}$ — then the **position spelled by** S , as expected, is the position $\langle \wp_1\alpha_1, \dots, \wp_n\alpha_n \rangle$. We say that a semiposition S is **legitimate** (resp. **quasilegitimate**) iff there is a completion S' of S such that the position spelled by S' is a legal (resp. quasilegal) position of $\Box H(\vec{y})$. The **compression** of a legitimate or quasilegitimate semiposition S is the expression \overline{S} resulting from S through replacing the numer of every numeric move by the symbol \star . Note that, while generally there are infinitely many possible legitimate or quasilegitimate semipositions, the number of their compressions is finite. The reason is that an infinite variety of legal runs of $\Box H(\vec{y})$ exists only due to numeric variations within numeric moves; in compressions, however, all numer degenerate into \star .

In the context of a given step i of a given computation branch of a given HPM, by the **so-far-seen semiposition** we shall mean the semiposition W written at time i on the initial section of the run tape that has ever been visited (at steps $\leq i$) by the run-tape scanning head, except that the last symbol of W should be **BLANK** if the corresponding cell contained a **BLANK** at the time when it was last seen by the scanning head, even if the content of that cell changed (namely, became \top or \perp) later. Intuitively, such a W is exactly what the machine knows at time i about its run-tape content based on what it has seen there so far. Next, let Z be the semiposition $\top\delta_1\dots\top\delta_m$, where $\delta_1, \dots, \delta_m$ are the moves

made by the machine so far (at steps $\leq i$). And let π be the string residing in the buffer at time i . Then by the **so-far-authored semiposition** we shall mean the (complete) semiposition Z_{BLANK} if π is empty, and the (incomplete) semiposition $Z \top \pi$ if π is nonempty. The **windup** of a quasilegitimate yet incomplete semiposition V of the form $\top \delta_1 \dots \top \delta_m \top \pi$ is the lexicographically smallest string ω such that $\langle \top \delta_1, \dots, \top \delta_m, \top \pi \omega \rangle$ is a \top -quasilegal position of $\sqcap H(\vec{y})$. Note that there is only a constant number of strings that are windups of some incomplete quasilegitimate semipositions. Also note that knowing the compression \overline{V} of an (incomplete quasilegitimate) semiposition V is sufficient to determine V 's windup.

We let \mathcal{M} keep partial track of the so-far-authored quasilegitimate semiposition V through remembering its compression \overline{V} . Similarly, \mathcal{M} keeps track of the so-far-seen legitimate semiposition W through remembering its compression \overline{W} ; besides, one of the symbols of \overline{W} is marked to indicate (keep track of) the current location of \mathcal{M} 's run-tape scanning head.¹⁵ With appropriately arranged details that are not worth discussing here, it is possible for \mathcal{M} , this way, to be able to immediately detect if and when W becomes illegitimate. If and when this happens, we let \mathcal{M} retire; besides, if V is quasilegitimate yet incomplete at the time of this event, then \mathcal{M} puts V 's windup into the buffer and, simultaneously, enters a move state before retiring. We shall refer to a move made this way as a **retirement move**. Maintaining the above \overline{W} (together with its mark) and \overline{V} only requires a constant amount of memory, so this can be fully done through \mathcal{M} 's state rather than tape memory. This means that, as long as W remains legitimate, \mathcal{M} can follow the work of \mathcal{L} step-by-step without having any time or space overhead, and act (reposition heads, put things into the buffer, move, etc.) exactly like \mathcal{L} , with the only difference between the two machines being that \mathcal{M} has a greater number of states than \mathcal{L} does, with any given state of \mathcal{L} being imitated by one of many “counterpart” states of \mathcal{M} , depending on the present values of \overline{V} and the marked \overline{W} that each such state “remembers” (e.g., is labeled with).

For the rest of this appendix, assume \mathcal{L} is a provident solution of $H(\vec{y})$. Fix an arbitrary computation branch B of \mathcal{M} , and let Γ_∞^B be the run spelled by B . From now on, whenever a context requires a reference to a computation branch but such a reference is missing, it should be understood as that we are talking about B . For simplicity, we shall assume that, in B , Environment made (legal) initial moves that brought $\sqcap H(\vec{y})$ down to $H(\vec{c})$ for some constants \vec{c} . Fix these \vec{c} . The case of B violating this assumption is not worth our attention for the reason of being trivial or, at least, much simpler than the present case.

We also fix arbitrary arithmetical functions $\mathbf{a}, \mathbf{s}, \mathbf{t}$. We may assume that all three functions are unary, or else replace them with their unarifications. Since the parameters $B, \Gamma_\infty^B, \vec{c}, \mathbf{a}, \mathbf{s}, \mathbf{t}$ are arbitrary, it is sufficient for us to focus on them and just show that the three conditions of the lemma are satisfied in the context of these particular parameters. For instance, to show that \mathcal{M} plays $\sqcap H(\vec{y})$ quasilegally, it is sufficient to show that Γ_∞^B is a \top -quasilegal run of $\sqcap H(\vec{y})$.

We extend the notation Γ_∞^B from B to any computation branch C of either \mathcal{M} or \mathcal{L} , stipulating that Γ_∞^C is the run spelled by C . We further agree that, for any $i \geq 0$, Γ_i^C stands for the position spelled on the run tape of the corresponding machine at step i of branch C , and ℓ_i^C stands for the background of that step. We also agree that W_i^C denotes the so-far-seen semiposition at step i of branch C , and V_i^C denotes the so-far-authored semiposition at step i of C . Finally, since $\sqcap H(\vec{y})$ and $H(\vec{c})$ are the only formulas/games we deal with in

¹⁵Namely, a marked symbol of \overline{W} other than \star indicates that the head is looking at the corresponding symbol of W , and a marked \star indicates that the head is looking at one of the bits of the corresponding numer.

this appendix, without risk of ambiguity we will often omit references to them when saying “legal”, “quasilegal”, “prudent” etc.

Consider any i such that W_i^B is legitimate. The legitimacy of W_i^B means it has a completion $U = \varphi_1\alpha_1 \dots \varphi_n\alpha_n\beta_{\text{BLANK}}$ such that the position $\Omega = \langle \varphi_1\alpha_1, \dots, \varphi_n\alpha_n\beta \rangle$ spelled by U is legal. Let k be the number of \perp -labeled moves in Ω . And let C be the computation branch of \mathcal{M} in which Environment acts exactly as it does in B , with only the following two differences: (1) Environment stops making any moves after it makes its k th move (meaning that, if $k = 0$, Environment simply never moves); (2) If $\varphi_n = \perp$, Environment’s k th move (i.e., the n th move of the play) is $\alpha_n\beta$. Of course, C spells a legal run. For this reason, in this branch \mathcal{M} behaves just like \mathcal{L} in the branch D where the environment makes exactly the same moves, in exactly the same order and at exactly the same times, as in C . We call such a D the W_i^B -induced branch of \mathcal{L} . The following two lemmas are simple observations, hardly requiring any proofs:

Lemma A.1. *Consider any $j \geq 0$ such that Γ_j^B is legitimate, and let D be the W_j^B -induced branch of \mathcal{L} . We have:*

1. *In D , \mathcal{L} ’s environment makes no moves at any step e with $e > j$.*
2. *Γ_∞^D is a legal run of $\Box H(\vec{y})$.*
3. *The initial segment of Γ_∞^B that brings $\Box H(\vec{y})$ down to $H(\vec{c})$ is also an initial segment of Γ_∞^D .*
4. *$V_{j+1}^D = V_{j+1}^B$, and hence also $(\Gamma_{j+1}^D)^\top = (\Gamma_{j+1}^B)^\top$.*
5. *For any $e \geq 0$, $\ell_e^D \leq \ell_e^B$.*

Lemma A.2. *There is a number s such that, for every $j \geq s$, $W_j = W_s$. The smallest of such numbers s we call the **W-stabilization point**.*

Having set up the above preliminaries, we prove the lemma clause by clause.

CLAUSE 1: Our goal is to show that:

$$\Gamma_\infty^B \text{ is } \top\text{-won (so, } \mathcal{M} \text{ is a solution of } \Box H(\vec{y})) \text{; } \quad (\text{A.1})$$

$$\Gamma_\infty^B \text{ is } \top\text{-quasilegal (so, } \mathcal{M} \text{ plays quasilegally)} \text{; } \quad (\text{A.2})$$

$$B \text{ is provident (so, } \mathcal{M} \text{ plays unconditionally providently). } \quad (\text{A.3})$$

(A.1): From the description of \mathcal{M} we can see that, if Γ_∞^B is \perp -legal and thus the so-far-seen semiposition always remains legitimate, \mathcal{M} interacts with its environment exactly like \mathcal{L} interacts with its environment in the corresponding scenario¹⁶ and, since \mathcal{L} is a solution of $\Box H(\vec{y})$, Γ_∞^B is \top -won. And if Γ_∞^B is \perp -illegal, then Γ_∞^B is automatically \top -won.

(A.2): For a contradiction, suppose Γ_∞^B is not \top -quasilegal. Let i be the smallest number such that the position Γ_i^B is not \top -quasilegal. Let ϕ be the (“offending”) move that \mathcal{M} made at step i of B .

Assume W_{i-1} is legitimate. Let D be the W_{i-1} -induced branch of \mathcal{L} . According to clause 4 of Lemma A.1, $(\Gamma_i^D)^\top = (\Gamma_i^B)^\top$. So, Γ_i^D is not \top -quasilegal, and then the same holds for the extension Γ_∞^D of Γ_i^D . Of course, Γ_∞^D ’s not being \top -quasilegal implies that it is simply illegal. But this contradicts clause 2 of Lemma A.1, according to which Γ_∞^D is legal.

¹⁶Namely, in the computation branch where \mathcal{L} ’s environment makes exactly the same moves at exactly the same times and in exactly the same order as in B .

Now assume W_{i-1} is not legitimate. Note that $i \geq 2$, because, at the initial step 0, \mathcal{M} would not be able to see an illegitimate semiposition (at best, \mathcal{M} would only see the label \perp in the leftmost cell, nothing else). Further note that the semiposition W_{i-2} is legitimate, because otherwise \mathcal{M} would have retired right after seeing it and thus would not have moved at step i . As soon as \mathcal{M} sees the illegitimate W_{i-1} , it retires. Thus, the move ϕ made at step i is a retirement move. Looking back at the conditions under which \mathcal{M} makes a retirement move, we see that the so-far-authored semiposition V_i^B should be complete and quasilegal. Let Σ_i^B be the position spelled by V_i^B . So, Σ_i^B is \top -quasilegal. But note that $(\Sigma_i^B)^\top = (\Gamma_i^B)^\top$, and thus we are facing a contradiction because, as we remember, Γ_i^B is not \top -quasilegal.

(A.3): As already noted in the proof of (A.1), if the run Γ_∞^B is \perp -legal, \mathcal{M} and its environment behave exactly like \mathcal{L} and its environment in the corresponding scenario. Then, since \mathcal{L} plays providently, B is a provident branch. Suppose now Γ_∞^B is \perp -illegal.

First, assume the so-far-seen semiposition in B becomes illegitimate at some step i . Note that ($i > 0$ and) W_{i-1}^B is legitimate. Let D be the W_{i-1}^B -induced branch of \mathcal{L} . By clauses 2 and 4 of Lemma A.1, Γ_∞^D is \perp -legal and $V_i^D = V_i^B$. The semiposition V_i^D must be quasilegal because otherwise, as can be seen with a little thought, (the provident) \mathcal{L} will have to make an illegal move in branch D at some point. But, in branch B , \mathcal{M} retires immediately after seeing the non-legitimate W_i . The only possibility for the buffer content of \mathcal{M} to remain nonempty after retirement would be if V_i^B was not quasilegal. However, as just observed, this is not the case.

Now assume the so-far-seen semiposition in B never becomes illegitimate. Let i be the W -stabilization point (which exists according to Lemma A.2). And let D be the W_i^B -induced branch of \mathcal{L} . It is not hard to see that, throughout the entire play, \mathcal{M} behaves — makes moves, puts strings into the buffer, repositions scanning heads — the same way in B as \mathcal{L} behaves in D . From clause 2 of lemma A.1, we also know that D spells a \perp -legal run and hence, due to \mathcal{L} 's playing providently, D contains infinitely many steps with empty buffer contents. Then so does B . That is, B is provident.

CLAUSE 2: Assume \mathcal{L} is a prudent solution of $H(\vec{y})$. We want to show that the run Γ_∞^B is \top -prudent. For a contradiction, deny this. Let i be the smallest number such that Γ_i^B is not \top -prudent. Note that $i > 0$. It is obvious that a move is made in B at step i . Let us call that move ϕ .

Assume W_{i-1}^B is legitimate. Let D be the W_{i-1}^B -induced branch of \mathcal{L} . Clauses 3 and 4 of Lemma A.1 imply that Γ_i^D is not \top -prudent, and then the same holds for the extension Γ_∞^D of Γ_i^D . At the same time, by clause 2 of the same lemma, Γ_∞^D is legal. This is a contradiction, because \mathcal{L} is a prudent solution of $H(\vec{y})$ and, as such, it could not have generated a (\perp -)legal run (Γ_∞^D) that is not \top -prudent.

Now assume W_{i-1}^B is not legitimate. Then, just as in the last paragraph of our proof of (A.2), $i \geq 2$, W_{i-2}^B is legitimate, and ϕ is a retirement move. Let D be the W_{i-2}^B -induced branch of \mathcal{L} . Analyzing the conditions under which \mathcal{M} makes a retirement move, we see that ϕ (rather than some proper prefix of it) was the content of \mathcal{M} 's buffer at step $i-1$ of B . Then, by clause 4 of Lemma A.1, the same is the case for \mathcal{L} 's buffer in branch D . But, since \mathcal{L} plays providently and (by clause 2 of Lemma A.1) Γ_∞^D is legal, in D , sooner or later \mathcal{L} will have to make a move ϕ' such that ϕ is a prefix of ϕ' . Obviously

such a move ϕ' , if legal, will inherit the imprudence of ϕ . This, together with clause 2 of Lemma A.1, contradicts our assumption that \mathcal{L} is a prudent solution of $H(\vec{y})$.

CLAUSE 3: : Assume \mathcal{L} is a $(\mathfrak{a}, \mathfrak{s}, \mathfrak{t})$ tricomplexity solution of $H(\vec{y})$.

Amplitude: Assume \mathcal{M} makes a move ϕ at a step i . Let m_ϕ be the magnitude of ϕ . We want to show that $m_\phi \leq \mathfrak{a}(\ell_i^B)$.

First, suppose W_{i-1}^B is legitimate. Let D be the W_{i-1}^B -induced branch of \mathcal{L} . In view of clause 4 of Lemma A.1, in D , \mathcal{L} makes the same move ϕ at the same time i . Since \mathcal{L} plays in amplitude \mathfrak{a} and since, by clause 2 of Lemma A.1, the run Γ_∞^D is legal, we have $m_\phi \leq \mathfrak{a}(\ell_i^D)$. The desired $m_\phi \leq \mathfrak{a}(\ell_i^B)$ follows from here by clause 5 of Lemma A.1.

Now suppose W_{i-1}^B is not legitimate. Then, as in the last paragraph of our proof of (A.2) $i \geq 2$, W_{i-2}^B is legitimate, and ϕ is a retirement move. Let D be the W_{i-2}^B -induced branch of \mathcal{L} . And let β be the content of \mathcal{M} 's buffer at step $i-1$ of B . By clause 4 of Lemma A.1, the same β is in the buffer of \mathcal{L} at step $i-1$ of D . At some step $s \geq i$ of D , the provident \mathcal{L} should make a move γ such that β is a prefix of γ . Let m_γ be the magnitude of that move. Since the run spelled by D is legal (clause 2 of Lemma A.1) and \mathcal{L} plays in amplitude \mathfrak{a} , we have $m_\gamma \leq \mathfrak{a}(\ell_s^D)$. But, in view of clause 1 of Lemma A.1, $\ell_s^D = \ell_i^D$. Thus, $m_\gamma \leq \mathfrak{a}(\ell_i^D)$. This, in view of clause 5 of Lemma A.1, implies $m_\gamma \leq \mathfrak{a}(\ell_i^B)$. From the way we measure magnitudes and from the way the windup operation is defined, it is clear that $m_\phi \leq m_\gamma$. Consequently, $m_\phi \leq \mathfrak{a}(\ell_i^B)$.

Space: Let i be the W -stabilization point, and let D be the W_i^B -induced branch of \mathcal{L} . If W_i^B is legitimate, then, as observed in the last paragraph of our proof of (A.3), \mathcal{M} 's behavior throughout B is indistinguishable from that of \mathcal{L} in D ; this, in view of clause 5 of Lemma A.1, means that B , just like D , does not violate the \mathfrak{s} space limits. Now suppose W_i^B is not legitimate. Whatever we said above still applies to the behavior of \mathcal{M} up to (including) step $i-1$. After that it makes a transition to step i and retires without consuming any additional space. So, the space consumption again remains within the limits of \mathfrak{s} .

Time: Again, let i be the stabilization point, and let D be the W_i^B -induced branch of \mathcal{L} . If W_i^B is legitimate, then, for the same reasons as in the case of space, B does not violate the \mathfrak{t} time limits. Now suppose W_i^B is not legitimate. Whatever we said in the preceding sentence still applies to the behavior of \mathcal{M} in B up to (including) step $i-1$. Then \mathcal{M} makes a transition to step i and retires. If no move is made upon this transition, all is fine. And if a move is made, then, in view of the relevant clauses of Lemma A.1, it can be seen that the timecost of that move does not exceed the timecost of the move that the provident \mathcal{L} would have to make in D sooner or later after time $i-1$. So, the time bound \mathfrak{t} is not violated.

APPENDIX B. PROOF OF LEMMA 4.2

Lemma 4.2, to a proof of which this appendix is exclusively devoted, reads:

There is an effective procedure that takes an arbitrary bounded formula $H(\vec{y})$, an arbitrary HPM \mathcal{N} and constructs an HPM \mathcal{K} such that, for any regular boundclass triple \mathcal{R} , if $H(\vec{y})$ is $\mathcal{R}_{\text{space}}$ -bounded and \mathcal{N} is an \mathcal{R} tricomplexity solution of $H(\vec{y})$, then \mathcal{K} is a provident and prudent \mathcal{R} tricomplexity solution of $H(\vec{y})$.

B.1. Getting started. Pick and fix an HPM \mathcal{N} and a bounded formula $H = H(\vec{y}) = H(y_1, \dots, y_u)$ with all free variables displayed. The case of $H(\vec{y})$ being elementary is trivial, so we assume that $H(\vec{y})$ contains at least one choice operator. Fix \mathfrak{D} as the maximum number of labmoves in any legal run of $\sqcap H$. Further fix \mathfrak{S} as the superaggregate bound of H .

Assume \mathcal{R} is a regular boundclass triple such that the formula $H(\vec{y})$ is \mathcal{R}_{space} -bounded and \mathcal{N} is an \mathcal{R} tricomplexity solution of $H(\vec{y})$. Note that, by Lemma 4.1, $\mathfrak{S} \in \mathcal{R}_{space}$. It is important to point out that our construction of \mathcal{K} below does not depend on \mathcal{R} or any assumptions on it.

In view of Lemma 10.1 of [3] and with Remark 2.4 of [5] in mind, we may and will assume that \mathcal{N} plays H providently. Then Lemma 4.4 (whose proof does not rely on the present lemma) allows us to further assume that \mathcal{N} is a quasilegal, unconditionally provident and unconditionally \mathcal{R} tricomplexity solution of H .

Following the notational practice of Section 5.7, we shall write $\mathcal{R}_{space}(\ell)$ as an abbreviation of the phrase “ $O(\mathfrak{p}(\ell))$ for some $\mathfrak{p}(z) \in \mathcal{R}_{space}$ ”. Similarly for $\mathcal{R}_{time}(\ell)$ and $\mathcal{R}_{amplitude}(\ell)$.

The technique that we employ below is very similar to the one used in Section 11 of [3]. Our goal is to construct a machine \mathcal{K} such that \mathcal{K} is a provident and prudent \mathcal{R} -tricomplexity solution of $H(\vec{y})$. From our construction it will be immediately clear that the construction is effective as required.

In both our description of the work of \mathcal{K} and our subsequent analysis of it, we shall rely — usually only implicitly — on the Clean Environment Assumption. Making this assumption is safe because the desired properties of \mathcal{K} are (1) being a solution of $H(\vec{y})$, (2) playing $H(\vec{y})$ providently, (3) playing $H(\vec{y})$ prudently and (4) playing $H(\vec{y})$ in \mathcal{R} tricomplexity. The definitions of all four of these properties, unlike, for instance, the definitions of the *unconditional* versions of the last three (cf. Section 4), only look at the \perp -legal plays of $\sqcap H$ by \mathcal{K} . This means that it virtually does not matter what happens if \mathcal{K} 's adversary starts playing illegally.

We design \mathcal{K} as a single-work-tape HPM. At the beginning of the play, as usual, it waits — without consuming any space — till Environment chooses constants \vec{c} for all u free variables \vec{y} of H . If this never happens, \mathcal{K} is an automatic winner trivially complying with the providence, prudence and \mathcal{R} tricomplexity conditions. Having said that, for the rest of this construction and our subsequent analysis of it, we shall assume that, in the scenario that we are considering, Environment indeed chose the constants \vec{c} (fix them!) for \vec{y} during an initial episode of the play.

Let us agree that a **quasilegal move** (of $H(\vec{c})$) means a move that may appear, with either label, in some quasilegal run of $H(\vec{c})$. And the **truncation** of a move α is the $H(\vec{c})$ -prudentization of the longest prefix α' of α such that α' is also a prefix of some quasilegal move. Note that, in view of our earlier assumption that H is not elementary, every move has a (possibly empty) truncation.

Once all constants \vec{c} are chosen by Environment, \mathcal{K} computes the value of $\mathfrak{S}|\max(\vec{c})|$ and remembers it for possible use in the future. It is not hard to see that, in view of the basic closure properties of boundclasses and the relevant conditions of Definition 2.2 of [5], $\mathfrak{S}|\max(\vec{c})|$ can be computed and recorded in space $\mathcal{R}_{space}|\max(\vec{c})|$ and time $\mathcal{R}_{time}|\max(\vec{c})|$. For this reason, when trying to show that \mathcal{K} runs in tricomplexity \mathcal{R} , the present episode of computing and remembering $\mathfrak{S}|\max(\vec{c})|$ can (and will) be safely ignored.

Upon the completion of the above step, \mathcal{K} starts simulating \mathcal{N} in the scenario where, at the very beginning of the play — on cycle 0, that is — the imaginary adversary of the latter chose the same constants \vec{c} for the free variables of H as (\mathcal{K} 's real) Environment did. A simulation would generally require maintaining and continuously updating configurations of \mathcal{N} . However, the challenge is that \mathcal{K} cannot afford to fully represent such configurations on its work tape. For instance, if all bounds in \mathcal{R}_{space} are sublinear, representing the run tape content of \mathcal{N} would require more than \mathcal{R}_{space} space. Similarly, the size of the content of the buffer of \mathcal{N} could occasionally go beyond the \mathcal{R}_{space} bound. For the above reasons, when dealing with a j th computation step of the simulated \mathcal{N} , we let \mathcal{K} , on its work tape, only keep representations of the other (and some additional, previously redundant) components of the corresponding configuration of \mathcal{N} . Namely, with “current” below referring to an arbitrary given j th computation step of \mathcal{N} , on its work tape \mathcal{K} maintains the following pieces of information¹⁷ — call them together the **sketch** of the j th configuration (computation step) of \mathcal{N} :

- 1st component:** The current state of \mathcal{N} .
- 2nd component:** The current contents of the work tapes of \mathcal{N} .
- 3rd component:** The current locations of the work-tape heads of \mathcal{N} .
- 4th component:** The current location of the run-tape head of \mathcal{N} .
- 5th component:** The number of moves that \mathcal{N} has made so far (at steps $\leq j$) in the play.
- 6th component:** The current number of symbols in the buffer of \mathcal{N} .
- 7th component:** The (possibly empty) string α that has been added to the buffer of \mathcal{N} when it made a transition to the j th step from the preceding, $(j - 1)$ th, step; here we stipulate that, if $j = 0$, i.e., if there is no preceding step, then such a string α is empty.
- 8th component:** The truncation α' of the move α currently written in the buffer.

Lemma B.1. *For any j , with ℓ standing for the background of the j 'th step of the simulated \mathcal{N} , maintaining the sketch for that step takes $\mathcal{R}_{space}(\ell)$ space.*

Proof. It is sufficient to verify that each of the eight components of the sketch, individually, can be maintained/recorded with $\mathcal{R}_{space}(\ell)$ space. Below we shall implicitly rely on Remark 2.4 of [5].

- 1st component:** Recording this component, of course, takes a constant and hence $\mathcal{R}_{space}(\ell)$ amount of space.
- 2nd component:** Since \mathcal{N} runs in unconditional space \mathcal{R}_{space} , this component can be represented with $\mathcal{R}_{space}(\ell)$ space.
- 3rd component:** The amount of space needed for recording this component obviously does not exceed the preceding amount — in fact, it is logarithmic in $\mathcal{R}_{space}(\ell)$.
- 4th component:** By our definition of HPMs from [3], the run-tape head can never go beyond the leftmost blank cell. So, how many non-blank cells may be on the imaginary run tape of \mathcal{N} ? Since \mathcal{N} plays in unconditional amplitude $\mathcal{R}_{amplitude}$, and since it plays H quasilegally and hence makes at most \mathfrak{D} moves, the \top -labeled moves residing on \mathcal{N} 's run tape only take $\mathcal{R}_{amplitude}(\ell)$ space. Next, as we are going to see later, all \perp -labeled moves residing on \mathcal{N} 's run tape are copies (made by \mathcal{K}) of \perp -labeled moves residing on \mathcal{K} 's run tape, by the Clean Environment Assumption meaning that their quantity is

¹⁷ Together with the never-changing representation of the transition function of \mathcal{N} , as well as the earlier computed $\mathfrak{S}|\max(\vec{c})|$. Whenever possible, we prefer not to mention explicitly these or similar, asymptotically irrelevant/superseded, pieces of information or events.

bounded by \mathfrak{D} , and also implying that those moves are quasilegal, due to which (not only their magnitudes but also) their sizes do not exceed $O(\ell)$. For this reason, the \perp -labeled moves of \mathcal{N} 's run tape, just like the \top -labeled moves, only take $\mathcal{R}_{\text{amplitude}}(\ell)$ of total space. Thus, there are at most $\mathcal{R}_{\text{amplitude}}(\ell)$ different possible locations of \mathcal{N} 's run-tape head. Representing any of such locations takes $|\mathcal{R}_{\text{amplitude}}(\ell)|$ and hence — by clause 5 of Definition 2.2 of [5] — $\mathcal{R}_{\text{space}}(\ell)$ space.

5th component: Since \mathcal{N} plays H quasilegally, the number of moves that \mathcal{N} has made so far can never exceed \mathfrak{D} , so holding the 5th component in memory only takes a constant amount of space.

6th component: Let m be the number of symbols currently in \mathcal{N} 's buffer. Assume $m > 0$, for otherwise holding it takes no space. Consider the scenario where \mathcal{N} 's adversary does not make any moves beginning from the current point. Since \mathcal{N} is unconditionally provident, sooner or later it should make a move α that is an extension of the move currently in the buffer, so the number of symbols in α is at least m . But, since \mathcal{N} plays H quasilegally and runs in unconditional $\mathcal{R}_{\text{amplitude}}$ amplitude, the number of symbols in α cannot exceed $\mathcal{R}_{\text{amplitude}}(\ell)$. That is, m does not exceed $\mathcal{R}_{\text{amplitude}}(\ell)$. Holding such an m therefore requires at most $|\mathcal{R}_{\text{amplitude}}(\ell)|$ space, and hence — again by clause 5 of Definition 2.2 of [5] — $\mathcal{R}_{\text{space}}$ space.

7th component: Recording this component, of course, only takes a constant amount of space.

8th component: With a moment's thought and with Lemma 4.1 in mind, it can be seen that, since α' is a truncation, the number of symbols in it does not exceed $\mathcal{R}_{\text{space}}(\ell)$. \square

Unfortunately, the sketch of a given computation step j of \mathcal{N} alone is not sufficient to fully trace the subsequent steps of \mathcal{N} and thus successfully conduct simulation. The reason is that, in order to compute (the sketch of) the $(j + 1)$ th step of \mathcal{N} , one needs to know the content of the cell scanned by the run-tape head of \mathcal{N} . However, sketches do not keep track of what is on \mathcal{N} 's run tape, and that information — unless residing on the run tape of \mathcal{K} itself by good luck — is generally forgotten. We handle this difficulty by letting the simulation routine recompute the missing information every time such information is needed. This is done through recursive calls to the routine itself. Properly materializing this general idea requires quite some care though. Among the crucial conditions for our recursive procedure to work within the required space limits is to make sure that the depth of the recursion stack never exceeds a certain constant bound.

To achieve the above goal, we let \mathcal{K} , in addition to the sketches for the simulated steps of \mathcal{N} , maintain what we call the **global history**. The latter is a list of all moves made by \mathcal{N} and its adversary throughout the imaginary play of H “so far”. More precisely, this is not a list of moves themselves, but rather entries with certain partial information on those moves. Namely, the entry for each move α does not indicate the actual content of α (which could require more than $\mathcal{R}_{\text{space}}$ space), but rather only the label of α (\top or \perp , depending on whether α was made by \mathcal{N} or its adversary) and the size of α , i.e., the number of symbols in α . Recording this information only takes $|\mathcal{R}_{\text{amplitude}}(\ell)|$ and hence $\mathcal{R}_{\text{space}}(\ell)$ space. Further, according to the forthcoming observation (B.3), the number of entries in the global history never exceeds $2\mathfrak{D}$ (in fact \mathfrak{D} , but why bother). Since \mathfrak{D} is a constant, we find that \mathcal{K} only consumes an $\mathcal{R}_{\text{space}}(\ell)$ amount of space for maintaining the overall global history. While a move α is not the same as the entry for it in the global history, in the sequel we may terminologically identify these two.

What do we need the global history for? As noted earlier, during its work, \mathcal{K} will often have to resimulate some already simulated portions of the work of \mathcal{N} . To make such a resimulation possible, it is necessary to have information on the times at which the adversary of \mathcal{N} has made its moves in the overall scenario that we are considering and re-constructing. Recording the actual move *times* as they were detected during the initial simulation, however, could take us beyond our target space limits. After all, think of a situation where \mathcal{N} waits “very long” before its environment makes a move. So, instead, we only keep track — via the global history — of the *order* of moves. Then we neutralize the problem of not remembering the “actual” times of \mathcal{N} ’s adversary’s moves by simply assuming that \mathcal{N} ’s adversary always makes its moves instantaneously in response to \mathcal{N} ’s moves. The point is that, if \mathcal{N} wins H , it does so in all scenarios, including the above scenario of instantaneously responding adversary.

It is important to note that, as will be immediately seen from our description of the work of \mathcal{K} , the moves recorded in the global history at any step of the work of \mathcal{K} are the same as the moves on the run tape of \mathcal{N} . And the latter, in turn, are copies of moves on the run tape of \mathcal{K} , with the only difference that, on \mathcal{K} ’s run tape, the \top -labeled moves appear in truncated forms. The orders of moves in the global history and on the run tape of \mathcal{N} are exactly the same. As for the run spelled on the run tape of \mathcal{K} , even if truncation did not really modify \mathcal{N} ’s moves, it may not necessarily be the same as the run spelled on the run tape of \mathcal{N} . Instead, the former is only guaranteed to be a \top -delay of the latter (see Section 3 of [3]). However, this kind of a difference, just like having the \top -labeled moves truncated, for our purposes (for \mathcal{K} ’s chances to win) is just as good as — or “even better than” — if the two runs were exactly the same.

The work of \mathcal{K} relies on the three subprocedures called UPDATE SKETCH, FETCH SYMBOL and MAKE HISTORY. We start with UPDATE SKETCH.

B.2. Procedure UPDATE SKETCH. In the context of a given global history \mathbb{H} , this procedure takes the sketch \mathbb{S}_j of a given computation step j of \mathcal{N} , and returns the sketch \mathbb{S}_{j+1} of the next computation step $j + 1$ of \mathcal{N} .

Let m be the 5th component of \mathbb{S}_j . The number m tells us how many moves \mathcal{N} had made by time j . In most cases, UPDATE SKETCH will be used while re-constructing some past episode of \mathcal{N} ’s work. It is then possible that the global history contains an $(m + 1)$ th move by \mathcal{N} (i.e., with label \top). If so, then such a move, as well as all subsequent moves of \mathbb{H} , are “future moves” from the perspective of the j th step of \mathcal{N} that UPDATE SKETCH is currently dealing with. This means that, when “imagining” the situation at the j th step of \mathcal{N} , those moves should be discarded. So, let \mathbb{H}' be the result of deleting from \mathbb{H} the $(m + 1)$ th \top -labeled move and all subsequent, whatever-labeled moves (if there are no such moves, then simply $\mathbb{H}' = \mathbb{H}$). Thus, \mathbb{H}' is exactly a record of the moves that \mathcal{N} would see — in the same order as they appear in \mathbb{H}' — on its run tape at step j .

The information contained in \mathbb{S}_j is “almost” sufficient for UPDATE SKETCH to calculate the sought value of \mathbb{S}_{j+1} . The only missing piece of information is the symbol s scanned by the run-tape head of \mathcal{N} on step j . UPDATE SKETCH thus needs, first of all, to figure out what that symbol s is. To do this, UPDATE SKETCH computes the sum p of the sizes of all moves (including their labels) of \mathbb{H}' . Next, let q (found in the 4th component of \mathbb{S}_j) be the number indicating the location of the run-tape head of \mathcal{N} on step j . Note that, in the scenario that UPDATE SKETCH is dealing with, the length of the “active” content of \mathcal{N} ’s run tape is p , with cell $\#(p + 1)$ and all subsequent cells being blank. So, UPDATE

`SKETCH` compares q with p . If $q > p$, it concludes that s is `BLANK`. Otherwise, if $q \leq p$, s should be one of the symbols of one of the moves α recorded in \mathbb{H}' . From \mathbb{H} , using some easy logarithmic-space arithmetic, `UPDATE SKETCH` figures out the author/label \wp of α , and also finds two integers k and n . Here k is the number of moves made by \wp before it made the move α . And n is the number such that the sought symbol s is the n th symbol of α . If $\wp = \perp$, using k and n , `UPDATE SKETCH` finds the sought symbol s on the run tape of \mathcal{K} . Otherwise, if $\wp = \top$, `UPDATE SKETCH` calls the below-described procedure `FETCH SYMBOL` on (k, n) . As will be seen later, `FETCH SYMBOL` then returns the sought symbol s . Thus, in any case, `UPDATE SKETCH` now knows the symbol s read by the run-tape head of \mathcal{N} on step j .

Keeping the above s as well as the earlier computed value $\mathfrak{S}|\max(\vec{c})|$ in mind,¹⁸ `UPDATE SKETCH` now additionally consults \mathbb{S}_j and finds (all 8 components of) the sought sketch \mathbb{S}_{j+1} using certain rather obvious logarithmic space calculations, details of which we omit.

B.3. Procedure `FETCH SYMBOL`. In the context of a given global history \mathbb{H} , this procedure takes two numbers k, n , where k is smaller than the number of \top -labeled moves in \mathbb{H} , and n is a positive integer not exceeding the length of the $(k + 1)$ th \top -labeled move there. The goal of `FETCH SYMBOL` is to return, through rerunning \mathcal{N} , the n th symbol of the $(k + 1)$ th \top -labeled move of \mathbb{H} .

To achieve the above goal, `FETCH SYMBOL` creates a sketch-holding variable \mathbb{S} , and sets the initial value of \mathbb{S} to the **initial sketch**. By the latter we mean the sketch of the initial configuration of \mathcal{N} , i.e., the configuration where \mathcal{N} is in its start state, the buffer and the work tapes are empty,¹⁹ and all scanning heads are looking at the leftmost cells of their tapes.

After the above initialization step, `FETCH SYMBOL` performs the following subprocedure:

- (1) Perform `UPDATE SKETCH` on \mathbb{S} . Let \mathbb{S}' be the resulting sketch, and let σ be the 7th component of \mathbb{S}' . Below, as always, $|\sigma|$ means the length of (number of symbols in) σ .
- (2) Let a and b be the 5th and 6th components of \mathbb{S} , respectively. If $a = k$ and $b < n \leq b + |\sigma|$, then return the $(n - b)$ th symbol of σ . Otherwise, update the value of \mathbb{S} to \mathbb{S}' , and go back to step 1.

Before proceeding, the reader may want to convince himself or herself that, as promised, `FETCH SYMBOL` indeed returns the n th symbol of the $(k + 1)$ th \top -labeled move of \mathbb{H} .

B.4. Procedure `MAKE HISTORY`. This procedure takes a global history \mathbb{H} as an argument and, treating \mathbb{H} as a variable that may undergo updates, acts according to the following prescriptions:

Stage 1: Create a variable \mathbb{S} and initialize its value to the initial sketch of \mathcal{N} . Proceed to Stage 2.

Stage 2: Check out \mathcal{K} 's run tape to see if Environment has made a new move (this can be done, say, by counting the \perp -labeled moves on the run tape, and comparing their number with the number of \perp -labeled moves recorded in the global history). If yes, update \mathbb{H} by adding to it a record for that move, and repeat `MAKE HISTORY`. If not, go to Stage 3.

¹⁸This value is (could be) needed for determining the 8th component of \mathbb{S}_{j+1} .

¹⁹As for the run tape, what is on it is irrelevant because a sketch has no record of the run-tape content anyway.

Stage 3:

- (a) Perform UPDATE SKETCH on \mathbb{S} . Let \mathbb{T} be the resulting sketch.
- (b) If \mathcal{N} did not make a **globally new** move on its transition from \mathbb{S} to \mathbb{T} ,²⁰ change the value of the variable \mathbb{S} to \mathbb{T} , and go back to Stage 2. Here and later in similar contexts, by a “globally new” move we mean a move not recorded in the global history \mathbb{H} . Figuring out whether \mathcal{N} made a globally new move is easy. Technically, \mathcal{N} made a globally new move if and only if, firstly, it *did* make a move, i.e., the 1st component of \mathbb{T} is a move state; and secondly, such a move is not recorded in \mathbb{H} , meaning that the 5th component of \mathbb{T} exceeds the total number of \top -labeled moves recorded in \mathbb{H} .
- (c) Suppose now \mathcal{N} made a globally new move α . Let α' be the 8th component of \mathbb{S} . Thus, α' is the truncation of α . Copy α' to the buffer (of \mathcal{K}) symbol by symbol, after which go to a move state. This results in \mathcal{K} making the move α' in the real play. Now update the global history \mathbb{H} by adding to it a record for the move α , and repeat MAKE HISTORY.

B.5. The overall strategy and an example of its run. We continue our description of the overall work \mathcal{K} , started on page 47 but interrupted shortly thereafter. As we remember, at the very beginning of the play, \mathcal{K} waited till Environment specified the u constants $\vec{c} = c_1, \dots, c_u$ for all free variables of H . What \mathcal{K} does after that is that it creates the variable \mathbb{H} , initializes its value to record the sequence $\langle \perp c_1, \dots, \perp c_u \rangle$, and then switches to running MAKE HISTORY forever. This completes our description of \mathcal{K} .

Here we look at an example scenario to make sure we understand the work of \mathcal{K} . Let

$$H = \Box y \left(|y| \leq |x| \rightarrow \Box z \left(|z| \leq |x| \wedge p(z, y) \right) \right) \vee \Box u \left(|u| \leq |x| \rightarrow \Box v \left(|v| \leq |x| \wedge q(u, v) \right) \right).$$

Note that the superaggregate bound of this formula is the identity function $\mathfrak{S}(w) = w$.

At the beginning of its work, \mathcal{K} waits till Environment specifies a value for x . Let us say 1001 is that value. After calculating $\mathfrak{S}[1001]$, which in the present case is 4, \mathcal{K} creates the variable \mathbb{H} and sets its value to contain a record for the (single) labmove $\perp\#1001$. The rest of the work of \mathcal{K} just consists in running MAKE HISTORY. So, in what follows, we can use “ \mathcal{K} ” and “MAKE HISTORY” as synonyms.

During its initialization Stage 1, MAKE HISTORY creates the variable \mathbb{S} and sets its value to the initial sketch of \mathcal{N} . The result of this step reflects the start situation, where “nothing has yet happened” in the mixture of the real play of H by \mathcal{K} and the simulated play of H by \mathcal{N} , except for Environment’s initial move #1001.

Now MAKE HISTORY starts performing, over and over, Stages 2 and 3. The work in those two stages can be characterized as “global simulation”. This is a routine that keeps updating, one step at a time, the sketch \mathbb{S} (Stage 3) to the sketch of the “next configurations” of \mathcal{N} in the scenario where the imaginary adversary of \mathcal{N} has made the move #1001 at the very beginning of the play; every time the simulated \mathcal{N} is trying to read some symbol of this move, \mathcal{K} finds that symbol on its own run tape and feeds it back to the simulation. Simultaneously, MAKE HISTORY keeps checking (Stage 2) the run tape of \mathcal{K} to see if Environment has made a new move. This will continue until either Environment or the simulated \mathcal{N} is detected to make a new move. In our example, let us imagine that

²⁰Here and later in similar contexts, we terminologically identify sketches with the corresponding steps of \mathcal{N} .

Environment makes the move 0.#10, signifying choosing the constant 10 for y in H . What happens in this case?

MAKE HISTORY simply restarts the global simulation by resetting the sketch \mathbb{S} to the initial sketch of \mathcal{N} . The earlier-described “Stage 2 and Stage 3 over and over” routine will be repeated, with the only difference that the global history \mathbb{H} is now showing the presence of both $\perp\#1001$ and $\perp0.\#10$. This means that the simulation of \mathcal{N} will now proceed in the scenario where, at the very beginning of the play, \mathcal{N} ’s adversary had made the two moves #1001 and 0.#10. So, every time the simulated \mathcal{N} tries to read one of the symbols of either move on its imaginary run tape, MAKE HISTORY— \mathcal{K} , that is — looks that symbol up on its own run tape. By switching to this new scenario, MAKE HISTORY, in fact, deems the previous scenario invalid, and simply forgets about it. This routine will continue until either Environment or \mathcal{N} , again, is detected to make a move.

Let us say it is now \mathcal{N} , which makes the imprudent move 0.1.#1111111, signifying choosing the “oversized” (of size > 4) constant 1111111 for z in H . In this event, MAKE HISTORY — \mathcal{K} , that is — assembles the truncation 0.1.#1111 of 0.1.#1111111 in its buffer copying it from the 8th component of \mathbb{S} , and then makes the move 0.1.#1111 in the real play. After that, as always when a new (lab)move is detected, the global simulation restarts. Now the global history \mathbb{H} is showing records for the sequence $\langle \perp\#1001, \perp0.\#10, \top0.1.\#1111111 \rangle$ of three moves. In the present, 3rd attempt of global simulation, just like in the 2nd attempt, \mathcal{N} is resimulated in the scenario where, at the beginning of the play, its adversary had made the moves #1001 and 0.#10. The only difference between the present attempt of global simulation and the previous one is that, once \mathcal{N} is detected to make the expected move 0.1.#1111111, nothing special happens. Namely, the global history is not updated (as 0.1.#1111111 is already recorded there); the move 0.1.#1111 is not made in the real play (as it already has been made); and the global simulation continues in the ordinary fashion rather than restarts. The present attempt of global simulation, again, will be interrupted if and when either Environment or the simulated \mathcal{N} is detected to make a globally new move, i.e., a move not recorded in the global history.

Let us say it is again Environment, which makes the move 1.#1, signifying choosing the constant 1 for u in H . As always, a record for the new move is added to \mathbb{H} , and the global simulation restarts. The resimulation of \mathcal{N} will start in the scenario where, at the beginning of the play, its adversary had made the moves #1001 and 0.#10. We already know that, in this scenario, sooner or later, \mathcal{N} will make its previously detected move 0.1.#1111111. Once this event is detected, \mathcal{N} ’s simulation continues for the scenario where its adversary responded by the move 1.#1 *immediately* after \mathcal{N} made the move 0.1.#1111111.

Imagine that the final globally new move detected is one by \mathcal{N} , and such a move is 1.1.#0, signifying choosing the constant 0 for v in H . MAKE HISTORY copies this move in the truncated form — which remains the same 1.1.#0 because this move is quasilegal and prudent — in the real play. Then, as always, \mathbb{H} is correspondingly updated, and the global simulation is restarted with that updated \mathbb{H} .

The last attempt of global simulation (the one that never got discarded/reconsidered) corresponds to the “ultimate” scenario that determined \mathcal{K} ’s real play. Namely, in our present example, the “ultimate” scenario in which \mathcal{N} was simulated is that, at the very beginning of the play, \mathcal{N} ’s adversary had made the moves #1001 and 0.#10, to which \mathcal{N} later responded with 0.1.#1111111, to which \mathcal{N} ’s adversary immediately responded with 1.#1, to which, some time later, \mathcal{N} responded with 1.1.#0, and no moves were made ever after. While the

imaginary run generated by \mathcal{N} in this scenario is

$$\langle \perp\#1001, \perp 0.\#10, \top 0.1.\#1111111, \perp 1.\#1, \top 1.1.\#0 \rangle, \quad (\text{B.1})$$

the real run generated by \mathcal{K} is

$$\langle \perp\#1001, \perp 0.\#10, \top 0.1.\#1111, \perp 1.\#1, \top 1.1.\#0 \rangle, \quad (\text{B.2})$$

with (B.2) being nothing but the result of replacing in (B.1) all \top -labeled moves by their truncations. Since it is our assumption that \mathcal{N} wins H , (B.1) is a \top -won run of H . But then so is (B.2) because, as noted earlier, truncating a given player's moves can (increase but) never decrease that player's chances to win.

Why do we need to restart the global simulation every time a globally new move is detected? The reason is that otherwise we generally would not be able to rely on calls of **FETCH SYMBOL** for obtaining required symbols. Going back to our example, imagine we did not restart the global simulation (**MAKE HISTORY**) after the moves $\#1001$ and $0.\#10$ were made by Environment. Perhaps (but not necessarily), as before, \mathcal{N} would still make its move $0.1.\#1111111$ sometime after $0.\#10$. Fine so far. But the trouble starts when, after that event, \mathcal{N} tries to read some symbol of $0.1.\#1111111$ from its imaginary run tape. A way to provide such a symbol is to invoke **FETCH SYMBOL**, which will resimulate \mathcal{N} to find that symbol. However, in order to properly resimulate \mathcal{N} up to the moment when it made the move $0.1.\#1111111$ (or, at least, put the sought symbol of the latter into its buffer), we need to know when (on which computation steps of \mathcal{N}), exactly, the labmoves $\perp\#1001$ and $\perp 0.\#10$ emerged on \mathcal{N} 's run tape. Unfortunately, we do not remember this piece of information, because, as noted earlier, remembering the exact times (as opposed to merely remembering the order) of moves may require more space than we possess. So, instead, we assume that the moves $\#1001$ and $0.\#10$ were made right at the beginning of \mathcal{N} 's play. This assumption, however, disagrees with the scenario of the original simulation, where $\#1001$ was perhaps only made at step 888, and $0.\#10$ perhaps at step 77777. Therefore, there is no guarantee that \mathcal{N} will still generate the same move $0.1.\#1111111$ in response to those two moves. Restarting the global simulation — as we did — right after $\#1001$ was made, and then restarting it again after $0.\#10$ was detected, neutralizes this problem. If \mathcal{N} made its move $0.1.\#1111111$ after $0.\#10$ in this new scenario (the scenario where its imaginary adversary always acted instantaneously), then every later resimulation, no matter how many times **MAKE HISTORY** is restarted, will again take us to the same move $0.1.\#1111111$ made after $0.\#10$, because the global history, which “guides” resimulations, will always be showing the first three labmoves in the order $\perp\#1001, \perp 0.\#10, \top 0.1.\#1111111$. To see this, note that all updates of the global history only add some moves to it, and otherwise do not affect the already recorded moves or their order.

We also want to understand one remaining issue. As we should have noticed, **FETCH SYMBOL** always calls **UPDATE SKETCH**, and the latter, in turn, may again call **FETCH SYMBOL**. Where is a guarantee that infinitely many or “too many” nested calls will not occur? Let us again appeal to our present example, and imagine we (**UPDATE SKETCH**, that is) are currently simulating a step of \mathcal{N} sometime after it already has made the move $1.1.\#0$. Whenever \mathcal{N} tries to read a symbol of $\top 1.1.\#0$, **FETCH SYMBOL** is called to resimulate \mathcal{N} and find that symbol. While resimulating \mathcal{N} , however, we may find that, at some point, its run-tape head is trying to read a symbol of the earlier labmove $\top 0.1.\#1111111$. To get that symbol, **FETCH SYMBOL** will be again called to resimulate \mathcal{N} and find that symbol. Can this process of mutual calls go on forever? Not really. Notice that, when **FETCH SYMBOL** is

called to find the sought symbol of $\top 0.1.\#1111111$, **FETCH SYMBOL**, guided by the global history, will resimulate \mathcal{N} only up to the moment when it made the move $0.1.\#1111111$. But during that episode of \mathcal{N} 's work, the labmove $\top 1.1.\#0$ was not yet on its run tape. So, **FETCH SYMBOL** will not have to be called further. Generally, as we are going to see in Section B.7.2, there can be at most a constant number of nested invocations of **FETCH SYMBOL** or **UPDATE SKETCH**.

B.6. \mathcal{K} is a provident and prudent solution of H . Consider an arbitrary play by (computation branch of) \mathcal{K} , and fix it for the rest of this appendix.

As seen from the description of **MAKE HISTORY**, the \perp -labeled moves recorded in \mathbb{H} are the moves made by (\mathcal{K} 's real) Environment. Since the latter is assumed to play legally, the number of \perp -labeled moves in \mathbb{H} cannot exceed \mathfrak{D} . Similarly, the \top -labeled moves of \mathbb{H} are the moves made by \mathcal{N} in a certain play. Therefore, as \mathcal{N} is a quasilegal solution of $\Box H$, the number of such moves cannot exceed \mathfrak{D} , either. Thus, with “never” below meaning “at no stage of the work of \mathcal{K} ”, we have:

$$\text{The number of labmoves in } \mathbb{H} \text{ never exceeds } 2\mathfrak{D}.^{21} \quad (\text{B.3})$$

Since every iteration of **MAKE HISTORY** increases the number of labmoves in \mathbb{H} , an immediate corollary of (B.3) is that

$$\text{MAKE HISTORY is iterated at most } 2\mathfrak{D} \text{ times.} \quad (\text{B.4})$$

Since **MAKE HISTORY** is restarted only finitely many times, the last iteration of it never terminates. Let Γ be the sequence of labmoves recorded in the final value of \mathbb{H} (i.e., the value of \mathbb{H} throughout the last iteration of **MAKE HISTORY**). This is the run generated by the simulated \mathcal{N} in what we referred to as the “ultimate scenario” in the preceding subsection (scenario = computation branch). Next, let Δ be the run generated by \mathcal{K} in the real play that we are considering. Since \mathcal{N} is a solution of $\Box H$, Γ is a \top -won run of $\Box H$. We want to verify that then Δ is also a \top -won run of $\Box H$, meaning that \mathcal{K} , too, is a solution of $\Box H$.

How do Γ and Δ differ from each other? As noted at the end of Section B.1, an analysis of the work of \mathcal{K} , details of which are left to the reader, reveals that there are only two differences.

The first difference is that the \top -labeled moves of Γ appear in Δ in truncated forms. This is so because, whenever \mathcal{K} makes a move (according to the prescriptions of Stage 3(c) of **MAKE HISTORY**), it copies that move from the 8th component of the sketch of the step of \mathcal{N} on which the latter made a move α ; but the 8th component of a sketch always holds the truncation of the move residing in \mathcal{N} 's buffer; thus, the move α' made by \mathcal{K} in the real play/run Δ is the truncation of the move α made by \mathcal{N} in the imaginary play/run Γ .

Let us use Ω to denote the result of changing in Γ all \top -labeled moves to their truncations.

The second difference between Γ and Δ is that, even if we ignore the first difference — that is, even if we consider Ω instead of Γ — the run is still not guaranteed to be exactly the same as Δ ; rather, we only know that the latter is a \top -delay of the former. The reason for this discrepancy is that, while performing **MAKE HISTORY**, \mathcal{K} may notice a move δ by Environment with some delay, only after it has first noticed a move γ by \mathcal{N} and made the truncation γ' of γ as a move in the real play; if this happens, $\top\gamma$ will appear before $\perp\delta$ in

²¹In fact, with some additional analysis, $2\mathfrak{D}$ can be lowered to \mathfrak{D} here, but why bother.

Γ but after $\perp\delta$ in Δ . But the game $\Box H$ is static, as are all games studied in CoL. And, by the very definition of static games (cf. Section 3 of [3]), Δ 's being a \top -delay of Ω implies that, if Ω is a \top -won run of $\Box H$, then so is Δ . This means that, in order to achieve our goal of proving that Δ is a \top -won run of $\Box H$, it is sufficient to simply show that Ω is a \top -won of $\Box H$. This is what the rest of this subsection is devoted to, for the exception of the last paragraph of it.

We may and will assume that different occurrences of quantifiers in $\Box H$ bind different variables. This is a legitimate assumption, because, if it is false, we can rename variables in $\Box H$ so as to make it true, with the new sentence, as a game, being virtually the same as the old sentence.

By a **unit** we shall mean a subformula U of H of the form $\Box r(|r| \leq \mathbf{b}|\vec{s}| \rightarrow E)$ (a \Box -**unit**) or $\Box r(|r| \leq \mathbf{b}|\vec{s}| \wedge E)$ (a \Box -**unit**). Here r is said to be the **master variable** of U , and $|r| \leq \mathbf{b}|\vec{s}|$ is said to be the **master condition** of U . “Subunit” and “superunit”, applied to units, mean the same as “subformula” and “superformula”. The **depth** of a unit U is the number of its superunits (including U itself). A unit U is **resolved** iff Γ contains a move signifying choosing a constant for U 's master variable. For instance, if H is $\Box x(|x| \leq |y| \wedge x = 0) \wedge \Box z(|z| \leq |y| \rightarrow \Box t(|t| \leq |z|' \wedge t = z + z))$ and Γ is $\langle \perp\#1000, \perp 1.\#11, \top 1.1.\#110 \rangle$, then the units $\Box z(|z| \leq |y| \rightarrow \Box t(|t| \leq |z|' \wedge t = z + z))$ and $\Box t(|t| \leq |z|' \wedge t = z + z)$ are resolved while $\Box x(|x| \leq |y| \wedge x = 0)$ is not. When w is a free variable of H or the master variable of some resolved unit, then the **resolvent** of w is the constant chosen for w in (according to) Γ . For instance, if H and Γ are as above, 1000 is the resolvent of y , 11 is the resolvent of z and 110 is the resolvent of t . A unit U is **well-resolved** iff U is resolved and the result of replacing all free variables by their resolvents in U 's master condition is true. A unit is **ill-resolved** iff it is resolved but not well-resolved. A **critical** unit is an ill-resolved unit all of whose proper superunits are well-resolved.

Let f be the subaggregate bound of H . For $i \in \{1, 2, \dots\}$, we define

$$\mathfrak{S}_i(z) = \max(f(z), f^2(z), \dots, f^i(z)).$$

Note that the superaggregate bound \mathfrak{S} of H is nothing but $\mathfrak{S}_{\mathfrak{H}}$, where \mathfrak{H} is the total number of all units. For this reason, taking into account that the depth of no unit can exceed \mathfrak{H} , we have:

$$\text{Whenever } i \text{ is the depth of some unit, } \mathfrak{S}_i \preceq \mathfrak{S}. \quad (\text{B.5})$$

Lemma B.2. *Consider an arbitrary resolved unit U . Let i be its depth, and a be the resolvent of its master variable. If all superunits of U (including U itself) are well-resolved, then $|a| \leq \mathfrak{S}_i |\max(\vec{c})|$.*

Proof. Induction on i . Assume the conditions of the lemma. Let w be the master variable of U , and let $|w| \leq \mathbf{b}(|x_1|, \dots, |x_k|, |z_1|, \dots, |z_m|)$ be the master condition of U , with all free variables displayed, where x_1, \dots, x_k are from among the free variables of H , and z_1, \dots, z_m are from among the master variables of the proper superunits of U . Let $d_1, \dots, d_k, e_1, \dots, e_m$ be the resolvents of $x_1, \dots, x_k, z_1, \dots, z_m$, respectively. Below we shall use c, d and e as abbreviations of $\max(\vec{c})$, $\max(d_1, \dots, d_k)$ and $\max(e_1, \dots, e_m)$, respectively. Let \mathbf{b}' be the unarification of \mathbf{b} .

U 's being well-resolved means that $|a|$ does not exceed $\mathbf{b}(|d_1|, \dots, |d_k|, |e_1|, \dots, |e_m|)$. Hence, by the monotonicity of \mathbf{b} , we have $|a| \leq \mathbf{b}' |\max(d, e)|$. But, of course, $\mathbf{b}' \preceq f$ (recall that f is the subaggregate bound of H). Thus, $|a| \leq f |\max(d, e)|$. This means that, in order to verify our target $|a| \leq \mathfrak{S}_i |c|$, it is sufficient to show that both $f |d| \leq \mathfrak{S}_i |c|$ and $f |e| \leq \mathfrak{S}_i |c|$.

That $f|d| \leq \mathfrak{S}_i|c|$ follows from the straightforward observations that $d \leq c$ and $f \preceq \mathfrak{S}_i$.

As for $f|e| \leq \mathfrak{S}_i|c|$, first assume $i = 1$. Then $m = 0$ and hence $e = 0$; also, G_i is f . Thus, we want to show that $f|0| \leq f|c|$. But this is immediate from the monotonicity of f . Now assume $i > 1$. By the induction hypothesis, $|e| \leq \mathfrak{S}_{i-1}|c|$. So, $f|e| \leq f(\mathfrak{S}_{i-1}|c|)$. But, of course, $f(\mathfrak{S}_{i-1}|c|) \leq \mathfrak{S}_i|c|$. Thus, $f|e| \leq \mathfrak{S}_i|c|$. \square

We are now in the position to see that Ω inherits Γ 's being a \top -won run of $\Box H$. Let

$$\Box u_1(|u_1| \leq \mathfrak{p}_1|\vec{r}_1| \rightarrow A_1), \dots, \Box u_a(|u_a| \leq \mathfrak{p}_a|\vec{r}_a| \rightarrow A_a)$$

be all the critical \Box -units, and let $u_1^\circ, \vec{r}_1^\circ, \dots, u_a^\circ, \vec{r}_a^\circ$ be the resolvents of $u_1, \vec{r}_1, \dots, u_a, \vec{r}_a$, respectively. Similarly, let

$$\sqcup v_1(|v_1| \leq \mathfrak{q}_1|\vec{s}_1| \wedge B_1), \dots, \sqcup v_b(|v_b| \leq \mathfrak{q}_b|\vec{s}_b| \wedge B_b)$$

be all the critical \sqcup -units, and let $v_1^\circ, \vec{s}_1^\circ, \dots, v_b^\circ, \vec{s}_b^\circ$ be the resolvents of $v_1, \vec{s}_1, \dots, v_b, \vec{s}_b$, respectively.

It is not hard to see that, following the notational conventions of Section 5.3 of [5] and Section 7 of [3], the paraformula $\langle \Gamma \rangle !\Box H$ can be written as

$$X \left[|u_1^\circ| \leq \mathfrak{p}_1|\vec{r}_1^\circ| \rightarrow A_1^\circ, \dots, |u_a^\circ| \leq \mathfrak{p}_a|\vec{r}_a^\circ| \rightarrow A_a^\circ, \right. \\ \left. |v_1^\circ| \leq \mathfrak{q}_1|\vec{s}_1^\circ| \wedge B_1^\circ, \dots, |v_b^\circ| \leq \mathfrak{q}_b|\vec{s}_b^\circ| \wedge B_b^\circ \right] \quad (\text{B.6})$$

for some $X, A_1^\circ, \dots, A_a^\circ, B_1^\circ, \dots, B_b^\circ$. With some additional analysis of the situation and with (B.5) and Lemma B.2 in mind, one can see that the paraformula $\langle \Omega \rangle !\Box H$ can then be written as

$$X \left[|u_1^\circ| \leq \mathfrak{p}_1|\vec{r}_1^\circ| \rightarrow A_1^\bullet, \dots, |u_a^\circ| \leq \mathfrak{p}_a|\vec{r}_a^\circ| \rightarrow A_a^\bullet, \right. \\ \left. |v_1^\bullet| \leq \mathfrak{q}_1|\vec{s}_1^\circ| \wedge B_1^\bullet, \dots, |v_b^\bullet| \leq \mathfrak{q}_b|\vec{s}_b^\circ| \wedge B_b^\bullet \right] \quad (\text{B.7})$$

for some $v_1^\bullet, \dots, v_b^\bullet, A_1^\bullet, \dots, A_a^\bullet, B_1^\bullet, \dots, B_b^\bullet$ (and with all other parameters the same as in (B.6)).

By the definition of the prefixation operation (Definition 2.2 of [3]), the fact that Γ is a \top -won run of $\Box H$ — written as $\langle \Gamma \rangle \Box H = \top$ — implies (in fact, means the same as) that the empty run $\langle \rangle$ is a \top -won run of $\langle \Gamma \rangle \Box H$, which, since $\langle \Gamma \rangle \Box H = (B.6)$, can be written as

$$\langle \rangle X \left[|u_1^\circ| \leq \mathfrak{p}_1|\vec{r}_1^\circ| \rightarrow A_1^\circ, \dots, |u_a^\circ| \leq \mathfrak{p}_a|\vec{r}_a^\circ| \rightarrow A_a^\circ, \right. \\ \left. |v_1^\circ| \leq \mathfrak{q}_1|\vec{s}_1^\circ| \wedge B_1^\circ, \dots, |v_b^\circ| \leq \mathfrak{q}_b|\vec{s}_b^\circ| \wedge B_b^\circ \right] = \top. \quad (\text{B.8})$$

Consider any $i \in \{1, \dots, b\}$. Since the unit $\sqcup v_i(|v_i| \leq \mathfrak{q}_i|\vec{s}_i| \wedge B_i)$ is critical and hence ill-resolved, $|v_i^\circ|$ exceeds $\mathfrak{q}_i|\vec{s}_i^\circ|$. Hence $\langle \rangle (|v_i^\circ| \leq \mathfrak{q}_i|\vec{s}_i^\circ| \wedge B_i^\circ) = \perp$. This clearly allows us to rewrite (B.8) as

$$\langle \rangle X \left[|u_1^\circ| \leq \mathfrak{p}_1|\vec{r}_1^\circ| \rightarrow A_1^\circ, \dots, |u_a^\circ| \leq \mathfrak{p}_a|\vec{r}_a^\circ| \rightarrow A_a^\circ, \perp, \dots, \perp \right] = \top.$$

The monotonicity of the operators $(\wedge, \vee, \forall, \exists)$ of X , just as in classical logic, allows us to replace the \perp s by whatever games in the above equation, so the latter can be further rewritten as

$$\langle \rangle X \left[|u_1^\circ| \leq \mathfrak{p}_1|\vec{r}_1^\circ| \rightarrow A_1^\circ, \dots, |u_a^\circ| \leq \mathfrak{p}_a|\vec{r}_a^\circ| \rightarrow A_a^\circ, \right. \\ \left. |v_1^\bullet| \leq \mathfrak{q}_1|\vec{s}_1^\circ| \wedge B_1^\bullet, \dots, |v_b^\bullet| \leq \mathfrak{q}_b|\vec{s}_b^\circ| \wedge B_b^\bullet \right] = \top. \quad (\text{B.9})$$

Next, for similar reasons, for every $i \in \{1, \dots, a\}$ we have $|u_i^\circ| > \mathfrak{p}_i |\vec{r}_i^\circ|$ and hence $\langle \rangle(|u_i^\circ| \leq \mathfrak{p}_i |\vec{r}_i^\circ| \rightarrow A_i^\bullet) = \top$, which allows us to rewrite (B.9) as

$$\begin{aligned} \langle \rangle X \left[|u_1^\circ| \leq \mathfrak{p}_1 |\vec{r}_1^\circ| \rightarrow A_1^\bullet, \dots, |u_a^\circ| \leq \mathfrak{p}_a |\vec{r}_a^\circ| \rightarrow A_a^\bullet, \right. \\ \left. |v_1^\bullet| \leq \mathfrak{q}_1 |\vec{s}_1^\circ| \wedge B_1^\bullet, \dots, |v_b^\bullet| \leq \mathfrak{q}_b |\vec{s}_b^\circ| \wedge B_b^\bullet \right] = \top. \end{aligned} \quad (\text{B.10})$$

However, the $X[\dots]$ part of (B.10) is identical to (B.7), which, in turn, is nothing but $\langle \Omega \rangle \sqcap H$. If so, the target $\langle \Omega \rangle \sqcap H = \top$ is an immediate consequence of (B.10).

Thus, \mathcal{K} is a solution of H , as desired. As such, it is both provident and prudent. \mathcal{K} is provident because, as a simple examination shows, it only puts something into its buffer while acting according to clause 3(c) of the description of MAKE HISTORY; however, at the end of the same clause, we see a prescription for \mathcal{K} to move, and thus empty the buffer. As for prudence, it is automatically achieved because \mathcal{K} only makes truncated moves, and such moves are always prudent.

B.7. \mathcal{K} plays in target tricomplexity. It remains to show that \mathcal{K} plays H in tricomplexity \mathcal{R} . Our analysis is going to be asymptotic, implicitly relying on Remark 2.4 of [5].

B.7.1. Amplitude. Since \mathcal{K} merely mimics — perhaps in the truncated form and perhaps with some delay — \mathcal{N} 's moves, it is obvious that the amplitude complexity of the former does not exceed that of the latter.

In fact, \mathcal{K} 's running in the target amplitude is also guaranteed by the facts that $\mathcal{R}_{space} \preceq \mathcal{R}_{amplitude}$ (clause 5 of Definition 2.2 of [5]), H is \mathcal{R}_{space} -bounded and \mathcal{K} plays H prudently.

B.7.2. Space. Let \mathbb{H} be a global history, and m a natural number. We define the **\mathbb{H} -index** of m as the number of moves in \mathbb{H}' , where \mathbb{H}' is the result of deleting from \mathbb{H} the $(m+1)$ th \top -labeled move and all subsequent whatever-labeled moves; if here \mathbb{H} does not contain more than m \top -labeled moves, then \mathbb{H}' is simply \mathbb{H} . Next, where \mathbb{S} is a sketch, we define the **\mathbb{H} -index** of \mathbb{S} as the \mathbb{H} -index of m , where m is the value of the 5th component of \mathbb{S} . We extend the concept of \mathbb{H} -index to particular runs/iterations of UPDATE SKETCH and FETCH SYMBOL in the process of performing MAKE HISTORY. Namely, UPDATE SKETCH is always run on a sketch \mathbb{S} , and we define the **\mathbb{H} -index** of that run of UPDATE SKETCH to be the \mathbb{H} -index of \mathbb{S} . Similarly, FETCH SYMBOL is always called on a pair (k, n) for some numbers k and n , and we define the **\mathbb{H} -index** of such a call/run of UPDATE SKETCH as the \mathbb{H} -index of k (n is thus irrelevant here). If \mathbb{H} is fixed or clear from the context, we may omit “ \mathbb{H} -” and simply say “**index**”.

Lemma B.3. *In the process of any given iteration of MAKE HISTORY and in the context of the then-current value of the global history variable \mathbb{H} , we have:*

1. *The index of any run of UPDATE SKETCH does not exceed $2\mathfrak{D}$.*
2. *Whenever a given run of UPDATE SKETCH calls FETCH SYMBOL, the index of the callee is strictly smaller than that of the caller.*
3. *Whenever a given run of FETCH SYMBOL calls UPDATE SKETCH, the index of the callee does not exceed that of the caller.*

Proof. Clause 1 is immediate from the obvious fact that an index can never exceed the number of moves in the global history, and the latter, according to (B.3), is bounded by $2\mathfrak{D}$. Clauses 2 and 3 can be verified through a rather straightforward (albeit perhaps somewhat long) analysis of the two procedures UPDATE SKETCH and FETCH SYMBOL; details of such an analysis are left to the reader. \square

We are now ready to examine the space complexity of \mathcal{K} . The space consumption of \mathcal{K} comes from the need to simultaneously maintain the global history and various sketches. As observed earlier, maintaining the global history consumes \mathcal{R}_{space} space, and, by Lemma B.1, each sketch also consumes \mathcal{R}_{space} space. At any given time, the global history is kept in memory in a single copy. So, to show that the overall space consumption is \mathcal{R}_{space} , we need to show that, at any given time, the number of sketches simultaneously kept in the memory of \mathcal{K} does not exceed a certain constant. But this is indeed so. Looking back at the work of MAKE HISTORY, we see that, at any time, its top level maintains a single sketch. It also keeps going through this sketch and updating it through UPDATE SKETCH, one step at a time. Since updates are done sequentially, space used for them can be recycled, so space consumptions for updating different sketches (this includes not only the top-level sketch of MAKE HISTORY, but also many additional sketches that will emerge during calls to FETCH SYMBOL when updating each individual sketch) do not add together unless those sketches happen to be on a same branch of nested recursive calls that UPDATE SKETCH and FETCH SYMBOL make to each other. In view of Lemma B.3, however, the depth of recursion (the height of the recursion stack at any time) is bounded, because the index of UPDATE SKETCH in the topmost level of recursion does not exceed $2\mathfrak{D}$, and every pair of successor levels of recursion strictly decreases the index of the corresponding call of UPDATE SKETCH.

B.7.3. *Time.* As we observed in (B.4), during the entire work of \mathcal{K} , MAKE HISTORY is iterated at most $2\mathfrak{D}$ times. The last iteration runs forever, but \mathcal{K} is not billed for that time because it makes no moves during that period. Likewise, \mathcal{K} will not be billed for the time spent on an iteration of MAKE HISTORY that was terminated at Stage 2, because a move by Environment resets \mathcal{K} 's time counter to 0. Call the remaining sorts of iterations of MAKE HISTORY — namely, the iterations that terminate according to the scenario of case (c) of Stage 3 — **time-billable**. So, it is sufficient for us to understand how much time a single time-billable iteration of MAKE HISTORY takes. Pick any such iteration and fix it throughout the context of the rest of this section, including the forthcoming Lemma B.4. We will use ℓ to denote the background of the last clock cycle of that iteration.

Lemma B.4. *The time consumed by any single run of UPDATE SKETCH or FETCH SYMBOL is $\mathcal{R}_{time}(\ell)$.*

Proof. We verify this lemma by induction on the index $i \in \{0, \dots, 2\mathfrak{D}\}$ of the corresponding call/run of UPDATE SKETCH or FETCH SYMBOL. Assume $i \geq 0$ is the index of a given run of UPDATE SKETCH. Looking back at our description of UPDATE SKETCH, we see that this routine makes at most one call of FETCH SYMBOL. First, assume no such call is made. Due to \mathcal{K} 's playing prudently, $\max(\ell, \mathfrak{S}(\ell))$ is the maximum magnitude of any move that may appear on \mathcal{K} 's run tape at any given time of the iteration. We also know from Lemma 4.1 that $\mathfrak{S} \preceq \mathcal{R}_{space}$. So, \mathcal{K} 's run-tape size (by which, as usual, we mean the size of the non-blank segment of the tape) is $O(\ell) + \mathcal{R}_{space}(\ell)$ and hence, by the relevant clauses of Definition 2.2 of [5], is $\mathcal{R}_{time}(\ell)$. We also know that the sketch and the global history are both of size $\mathcal{R}_{space}(\ell)$ and hence $\mathcal{R}_{time}(\ell)$. Keeping these facts in mind, with some analysis it

is obvious that, in this case, UPDATE SKETCH spends $\mathcal{R}_{time}(\ell)$ time. Now assume UPDATE SKETCH *does* call FETCH SYMBOL. By clause 2 of Lemma B.3, the index j of such a call is less than i . Hence, by the induction hypothesis, the time taken by the latter is $\mathcal{R}_{time}(\ell)$. In addition to this, UPDATE SKETCH only spends the same amount $\mathcal{R}_{time}(\ell)$ of time as in the preceding case to complete its work. Thus, in either case, the time consumption of UPDATE SKETCH is $\mathcal{R}_{time}(\ell)$.

Now consider a run of FETCH SYMBOL, and let $i \geq 0$ be its index. By clause 3 of Lemma B.3, the index of any call of UPDATE SKETCH that the given run of FETCH SYMBOL makes is at most i . By the induction hypothesis, each such call of UPDATE SKETCH consumes at most $\mathcal{R}_{time}(\ell)$ time. Processing any such call (doing additional work related to it), in turn, obviously takes at most $\mathcal{R}_{time}(\ell)$ time. So, each call of UPDATE SKETCH costs our run of FETCH SYMBOL at most $\mathcal{R}_{time}(\ell)$ time. How many such calls of UPDATE SKETCH will FETCH SYMBOL make? Since \mathcal{N} runs in time \mathcal{R}_{time} , with a little thought one can see that the number of calls of UPDATE SKETCH is at most $\mathcal{R}_{time}(\ell)$. So, the overall time cost of the run of FETCH SYMBOL is $\mathcal{R}_{time}(\ell) \times \mathcal{R}_{time}(\ell)$, which, in view of the closure of \mathcal{R}_{time} under \times , remains $\mathcal{R}_{time}(\ell)$. \square

We are now ready to look at the time consumption of the single time-billable iteration of MAKE HISTORY fixed earlier.

Stage 1 of MAKE HISTORY obviously takes a constant amount of time, and this stage is iterated only once. So, asymptotically, it contributes nothing to the overall time consumption of the procedure.

Stage 2 checks out the run tape, which may require moving the run-tape head of \mathcal{K} from one end of the (non-blank segment of the) run tape to the other end. Additionally, the global history needs to be updated, but this can be done even faster. So, this stage obviously takes as much time as the size of \mathcal{K} 's run tape, which, as observed in the proof of Lemma B.4, is $\mathcal{R}_{time}(\ell)$.

Stage 3 starts with performing UPDATE SKETCH (Substage 1), and this, by Lemma B.4, takes $\mathcal{R}_{time}(\ell)$ time. With a little thought, the time taken by Substages (b) and (c) of Stage 3 can be seen to be at most quadratic in the size of \mathcal{K} 's run tape. We know that the latter is $\mathcal{R}_{space}(\ell)$. Hence so is $(\mathcal{R}_{space}(\ell))^2$, because \mathcal{R}_{time} is closed under \times .

To summarize, none of the 3 stages of MAKE HISTORY takes more than $\mathcal{R}_{time}(\ell)$ time. Stage 1 is repeated only once, and the remaining two stages are repeated at most $\mathcal{R}_{time}(\ell)$ times as can be seen with a little thought, keeping in mind that the iteration of MAKE HISTORY that we are dealing with is a time-billable one. If so, due to \mathcal{R}_{time} 's closure under \times , the overall time consumption is $\mathcal{R}_{time}(\ell)$, which obviously implies that \mathcal{K} plays $\Box H$ in time \mathcal{R}_{time} , as desired.

REFERENCES

- [1] K. Aehlig, U. Berger, M. Hoffmann and H. Schwichtenberg. *An arithmetic for non-size-increasing polynomial-time computation*. **Theoretical Computer Science** 318 (2004), pp. 3–27.
- [2] G. Japaridze. *Introduction to clarithmetic I*. **Information and Computation** 209 (2011), pp. 1312–1354.
- [3] G. Japaridze. *On the system CL12 of computability logic*. **Logical Methods in Computer Science** 11 (2015), Issue 1, Paper 1, pp. 1–71.
- [4] G. Japaridze. *Introduction to clarithmetic II*. **Information and Computation** 247 (2016), pp. 290–312.
- [5] G. Japaridze. *Build your own clarithmetic I: Setup and completeness*. arXiv:1510.08564 (2015).

INDEX

- \mathbb{B}_n^h 24
- $\mathbb{B}_n^h \uparrow, \mathbb{B}_n^h \downarrow$ 24
- birthtime (of an entry) 24
- body 13
- body (of an entry) 18
- born 25
- central triple 18
- common entry 18
- complete semiposition 42
- critical unit 56
- completion (of a semiposition) 42
- compression (of a semiposition) 42
- consistent (bodies) 23
- \mathfrak{D} 47
- depth (of a unit) 56
- $\epsilon, \epsilon_{\top}, \epsilon_{\perp}$ 11
- \vec{E} 19
- \vec{E}_i 20
- early moves 38
- entry (of an aggregation) 18
- essentially the same (aggregations, bodies etc.) 39
- even* (superscript) 16
- extension (of a body) 16
- F 9
- F' 10
- FETCH SYMBOL 51
- \mathfrak{g} 12
- \mathfrak{G} 11
- $\Gamma_{\infty}^C, \Gamma_i^C$ 43
- global history 49
- globally new move 52
- \mathfrak{h} 12
- \hbar 28
- \mathcal{H}_0 10
- \mathcal{H}_n 10
- HPM 3
- \mathbb{I} 20
- \mathbb{I}^h 22
- $\mathbb{I}^h_!$ 23
- \mathbb{I}^h_{\bullet} 38
- $\mathbb{I}^h_{\bullet\bullet}$ 38
- $\mathbb{I}^h_{\bullet\bullet\bullet}$ 38
- ill-resolved unit 56
- incomplete semiposition 42
- index (of an entry) 18
- index (\mathbb{H} -index) 58
- initial sketch 51
- k 10
- \mathcal{K} 47
- \mathfrak{l} 11
- \mathfrak{L} 11
- ℓ_i^C 43
- late moves 38
- legitimate semiposition 42
- locking (iteration of MAIN) 22
- \mathcal{M} 3,4,10,42
- \mathcal{M}_k 10
- MAIN 19
- MAKE HISTORY 51
- master condition 56
- master entry 18
- master: organ, payload, scale 18
- master variable (of a unit) 56
- $\max(\vec{c})$ 6
- negative (signed organ) 13
- numer 5
- numeric (lab)move 5
- odd* (as a superscript) 16
- organ 13
- payload 13
- positive (signed organ) 13
- provident(ly) (branch, solution, play) 6
- prudent move 6
- prudent(ly) play 6
- prudent run 6
- prudent solution 6
- prudentization 6
- \mathfrak{q} 12
- quasilegal move 47
- quasilegal(ly) play 7,12
- quasilegal run 7
- quasilegal solution 7
- quasilegitimate semiposition 42
- \mathfrak{r} 12
- reasonable (play, solution) 9,12
- REPEAT 19
- repeating (iteration of MAIN) 22
- resolved unit 56
- resolvent 56
- RESTART 19
- restarting (iteration of MAIN) 22

- restriction (of a body) 16
- retirement move 43
- \mathfrak{S} 47
- \mathfrak{S}_i 56
- saturated 16
- scale 13
- semiposition 42
- signed organ 13
- SIM 13,15
- SIM^\bullet 15
- SIM^\leftarrow 16
- SIM^\rightarrow 16
- SIM-appropriate triple 13
- size (of a body) 13
- sketch 48
- so-far-authored semiposition 43
- so-far-seen semiposition 42
- subaggregate bound 6
- superaggregate bound 6
- symbolwise length 12
- synchronizing 11
- time-billable 59
- transient (iteration of MAIN) 22
- truncation 47
- UPDATE SKETCH 50
- U 19
- unconditional (amplitude, space, time) 8
- unconditionally provident(ly) 7
- unconditionally prudent(ly) 7,12
- unit 56
- unreasonable (play, solution) 9
- V_i^C 43
- W_i^C 43
- W_i^B -induced branch of \mathcal{L} 44
- W -stabilization point 44
- well-resolved unit 56
- windup 43
- \mathfrak{v} 9
- $\wp\omega\vec{\alpha}$, $\top\omega\vec{\alpha}$, $\perp\omega\vec{\alpha}$ 14
- \overline{B} (where B is a body) 16
- Γ^\top , Γ^\perp 7
- Γ^0 , Γ^1 17
- \oplus 14
- \preceq (as a relation between runs) 17
- \neg (as an operation on runs) 17
- \uparrow , \downarrow 24