Strong Extension Axioms and Shelah's Zero-One Law for Choiceless Polynomial Time

Andreas Blass*
Mathematics Department
University of Michigan
Ann Arbor, MI 48109–1109, U.S.A.
ablass@umich.edu

Yuri Gurevich
Microsoft Research
One Microsoft Way
Redmond, WA 98052, U.S.A.
gurevich@microsoft.com

September 11, 2000

Abstract

This paper developed from Shelah's proof of a zero-one law for the complexity class "choiceless polynomial time," defined by Shelah and the authors. We present a detailed proof of Shelah's result for graphs, and describe the extent of its generalizability to other sorts of structures. The extension axioms, which form the basis for earlier zero-one laws (for first-order logic, fixed-point logic, and finite-variable infinitary logic) are inadequate in the case of choiceless polynomial time; they must be replaced by what we call the strong extension axioms. We present an extensive discussion of these axioms and their role both in the zero-one law and in general.

*Partially supported by a grant from Microsoft Research.

1 Introduction

The BGS model of computation was defined in [4] with the intention of modeling computation with arbitrary finite relational structures as inputs, with essentially arbitrary data types, with parallelism, but without arbitrary choices. In the absence of any resource bounds, the lack of arbitrary choices makes no difference, because an algorithm could take advantage of parallelism to produce all possible linear orderings of its input and then use each of these orderings to make whatever choices are needed. But if we require the total computation time (summed over all parallel subprocesses) to be polynomially bounded, then there isn't time to construct all the linear orderings, and so the inability to make arbitrary choices really matters.

In fact, it was shown in [4] that choiceless polynomial time, $\tilde{\mathrm{C}}$PTime, the complexity class defined by BGS programs subject to a polynomial time bound, does not contain the parity problem: Given a set, determine whether its cardinality is even. Several similar results were proved, all depending on symmetry considerations, i.e., on automorphisms of the input structure.

Subsequently, Shelah [15] proved a zero-one law for $\tilde{\mathrm{C}}$PTime properties of graphs[1]. We shall state this law and present its proof later in this paper. For now, let us just mention a crucial difference from the earlier results in [4]: Almost all finite graphs have no non-trivial automorphisms, so symmetry considerations cannot be applied to them. Shelah's proof therefore depends on a more subtle concept of partial symmetry, which we explain in Section 12 below.

Much of the present paper is devoted to an exposition of Shelah's proof of (one version of) the zero-one law, but we also prove several other results concerning the so-called strong extension axioms, which play a crucial role in that proof. We also hope that, by concentrating on a single version of the zero-one law, by formulating the argument in terms of (nearly) the same computation model used in [4], and by including many details, we have made the presentation of Shelah's proof more accessible than the original source [15].

For simplicity, we shall generally deal only with input structures that are undirected, loopless graphs, i.e., sets equipped with a symmetric, irreflexive binary relation of adjacency. We also restrict our attention to the uniform probability model. That is, we define the *probability* of a class (assumed

[1]Actually, Shelah's proof covers more general structures; see Section 3.

closed under isomorphism) of n-vertex graphs by considering all graphs with vertex set $\{1, 2, \ldots, n\}$ to be equally probable. This probability measure can also be defined by saying that, for each potential edge, i.e., each set of two distinct vertices, we flip a fair coin to decide whether to include the edge in our graph. In such contexts, it is always to be understood that the coin flips are independent. The *asymptotic probability* of a class of graphs is defined as the limit, as $n \to \infty$, of the probability of its intersection with the class of n-vertex graphs. We sometimes refer to the probability of a property of graphs, meaning the probability of the class of graphs that have that property. In general, a zero-one law says that definable classes have asymptotic probability 0 or 1, but, as we shall see, some care is needed in formulating the zero-one law for $\tilde{\text{C}}$PTime.

All the results discussed in this paper can be routinely extended to other contexts, such as directed graphs, or sets with several relations, including relations of more than two arguments. It is also routine to replace the uniform probability measure by one where all potential edges have probability p, a constant other than $\frac{1}{2}$. We shall describe in Section 3 a general framework for such extensions, but, because these generalizations complicate the notation without contributing new ideas, we shall discuss only graphs in the rest of the paper.

An abridged version [1] of this paper sketches the proof of the zero-one law that is presented in detail here. Also, [2] is a popularization of part of the material in the present paper concerning the strong extension axioms.

2 The Zero-One Law

We start with a very brief description of the BGS model of computation, just adequate to formulate the zero-one law. In Section 4, we shall give more details about the model, in preparation for a description of the proof of the zero-one law.

The BGS model, introduced in [4], is a version of the abstract state machine (ASM) paradigm [11]. The input to a computation is a finite relational structure I. A state of the computation is a structure whose underlying set is $\text{HF}(I)$, which consists of the underlying set of I together with all hereditarily finite sets over it; the structure has the relations of I, some set-theoretical apparatus (for example the membership relation $\in$), and some dynamic functions. The computation proceeds in stages, always modifying

the dynamic functions in accordance with the program of the computation. The dynamic functions are initially constant with value $\emptyset$ and they change at only finitely many arguments at each step. So, although $\mathrm{HF}(I)$ is infinite, only a finite part of it is involved in the computation at any stage. The computation ends when and if a specific dynamic 0-ary function Halt acquires the value $\texttt{true} = \{0\}$, and the result of the computation is then the value of another dynamic 0-ary function Output. (We have adopted the convention that the truth values are identified with the first two von Neumann ordinals, $\texttt{false} = 0 = \emptyset$ and $\texttt{true} = 1 = \{0\}$. Recall that the finite von Neumann ordinals represent the natural numbers by identifying n with the set $\{0, 1, \ldots, n-1\}$.)

This model was used to define choiceless polynomial time $\tilde{\mathrm{C}}$PTime by requiring a computation to take only polynomially many (relative to the size of the input structure I) steps and to have only polynomially many active elements. (Roughly speaking, an element of $\mathrm{HF}(I)$ is active if it participates in the updating of some dynamic function at some stage.) Also, Output was restricted to have Boolean values, so the result of a computation could only be true, or false, or undecided. (The "undecided" situation arises if the computation exhausts the allowed number of steps or the allowed number of active elements without Halt becoming `true`.) We shall use the phrase *polynomial time BGS program* to refer to a BGS program, with Boolean Output, together with polynomial bounds on the number of steps and the number of active elements.

Two classes $\mathcal{K}_0$ and $\mathcal{K}_1$ of graphs are $\tilde{\mathrm{C}}$PTime-*separable* if there is a polynomial time BGS program Π such that, for all input structures from $\mathcal{K}_0$ (resp. $\mathcal{K}_1$), Π halts with output `false` (resp. `true`) without exceeding the polynomial bounds. It doesn't matter what Π does when the input is in neither $\mathcal{K}_0$ nor $\mathcal{K}_1$.

Theorem 2.1 (Shelah's Zero-One Law) *If $\mathcal{K}_0$ and $\mathcal{K}_1$ are $\tilde{\mathrm{C}}$PTime-separable classes of undirected graphs, then at least one of $\mathcal{K}_0$ and $\mathcal{K}_1$ has asymptotic probability zero.*

An equivalent formulation of this is that, for any given polynomial time BGS program, either almost all graphs produce output true or undecided or else almost all graphs produce output false or undecided. It is tempting to assert the stronger claim that either almost all graphs produce true, or almost all produce false, or almost all produce undecided. Unfortunately,

this stronger claim is false; a counterexample will be given after we review the definition of BGS programs in Section 4.

The theorem was, however, strengthened considerably in another direction in [15]. It turns out that the number of steps in a halting computation is almost independent of the input.

Theorem 2.2 *Let a BGS program* Π *with Boolean output and a polynomial bound for the number of active elements be given. There exist a number* m*, an output value* v*, and a class* $\mathcal{C}$ *of undirected graphs, such that* $\mathcal{C}$ *has asymptotic probability one and such that, for each* $\langle I, A\rangle \in \mathcal{C}$*, either*

- Π *on input* $\langle I, A\rangle$ *halts after exactly* m *steps with output value* v *and without exceeding the given bound on active elements, or*
- Π *on input* $\langle I, A\rangle$ *either never halts or exceeds the bound on active elements.*

The proof of the theorem gives a somewhat more precise result. If there is even one input $\langle I, A\rangle \in \mathcal{C}$ for which Π eventually halts, say at step m, without exceeding the bound on active elements, then in the second alternative in the theorem the computation will exceed the bound on active elements at or before step m.

Notice that this theorem does not assume a polynomial bound on the number of steps. It is part of the conclusion that the number of steps in a successful computation is not only polynomially bounded but constant as long as the input is in $\mathcal{C}$ and the number of active elements obeys its bound.

Intuitively, bounding the number of active elements, without bounding the number of computation steps, amounts to a restriction on space, rather than time. Thus, Theorem 2.2 can be viewed as a zero-one law for choiceless polynomial space computation.

The class $\mathcal{C}$ in the theorem actually has a fairly simple description; it consists of the graphs that have at least n_1 nodes and satisfy the strong extension axioms to be defined in Section 6 below for up to n_2 variables. The parameters n_1 and n_2 in this definition can be easily computed when the program Π and the polynomial bound on the number of active elements are specified.

3 Thesauri

In this section, we describe a general framework for dealing with random structures "similar" to graphs. The results in the rest of this paper can be generalized to this framework, but this section can be skipped without damage to the rest of the paper.

The uniform probability measure on graphs with vertex set $\{1, 2, \ldots, n\}$ has, as indicated above, two equivalent definitions, one "global" (all graphs have equal probability) and one "local" (each potential edge has probability $\frac{1}{2}$ of being present, and different potential edges are probabilistically independent). The global definition applies verbatim to structures of any sort, but it is considerably more difficult to study when it is not equivalent to a local definition, for example when the structures are partial orderings. Furthermore, the local definition admits variations, such as allowing edge probabilities other than $\frac{1}{2}$, that are relatively easy to analyze but do not admit a simple global definition. The framework developed in this section is intended to describe probability distributions of the local sort. Recall that local distributions make sense not only for graphs but also for general relations of arbitrary arity. Graphs are an example where the (adjacency) relation is subject to certain constraints, namely irreflexivity and symmetry. Thesauri are intended to capture general constraints of this sort. Notice that not all constraints are amenable to such treatment; for example, transitivity of a binary relation constrains it in a way not compatible with assigning probabilities by independent flips of a (possibly biased) coin.

We shall generalize the notion of a relation symbol in a first-order vocabulary in two ways. First, the symbol may come with an attached notion of symmetry. For binary relations, this means ordinary symmetry or antisymmetry, but for relations of higher arity more complicated notions of symmetry become possible. Second, our relations will not necessarily be two-valued (with values `true` and `false`) but will have some specified finite number of values.

We shall also restrict our generalized relations, which we call colorings, to apply only to tuples of *distinct* elements. Thus, for example, if R is a binary relation in the usual sense, then we would regard it as two (2-valued) colorings, a binary one assigning to each distinct a and b the value $R(a, b)$ and a unary one assigning to each a the value $R(a, a)$. Similarly, a ternary relation becomes five (2-valued) colorings — one ternary, three binary (from $R(a, a, b)$, $R(a, b, a)$, and $R(b, a, a)$), and one unary.

A thesaurus will consist of specifications — arity, number of values, and symmetry — for finitely many colorings. In addition, it will be convenient to include, for each coloring, a probability distribution on the colors. Formally, the definitions are as follows; we present the syntax first and then the semantics.

Definition 3.1 A *signum* is a 6-tuple $\langle R, j, v, G, h, p\rangle$ where

- R is an arbitrary symbol,
- j is a natural number, called the *arity*,
- v is a non-zero natural number called the *value number*,
- G is a group of permutations of $\{1, 2, \ldots, j\}$,
- h is a homomorphism from G into the group of permutations of $\{1, 2, \ldots, v\}$, and
- p is a probability distribution on $\{1, 2, \ldots, v\}$ that is invariant under the group $h(G)$ and that gives each element of $\{1, 2, \ldots, v\}$ non-zero probability.

A *thesaurus* is a finite set of signa with distinct first components.

Here R is analogous to relation symbols in traditional vocabularies for first-order logic. In keeping with this analogy, we sometimes write R when we really mean the whole signum. Thus, we may say that j is the arity of R or that R is a v-valued signum. We sometimes refer to a signum minus the symbol R, i.e., $\langle j, v, G, h, p\rangle$, as the *type* of the signum, and say that R is a signum of this type. The G and h in the definition serve to specify the symmetry, as will become clear from the following definition of the semantics.

Definition 3.2 Let X be a non-empty set. A *coloring* of X of type $\langle j, v, G, h, p\rangle$ is a function C from the set $X^{j\neq}$ of j-tuples of distinct elements of X into $\{1, 2, \ldots, v\}$ such that, for each permutation $\pi \in G$ and each $(x_1, \ldots, x_j) \in X^{j\neq}$,

$$C(x_1, \ldots, x_j) = h(\pi)(C(x_{\pi(1)}, \ldots, x_{\pi(j)})).$$

If Υ is a thesaurus, then an Υ-*structure* M consists of a nonempty base set $|M|$ with, for each $R \in \Upsilon$, a coloring R^M of $|M|$ of the same type as R.

This definition does not make use of the probability distribution, but the next one will. We remark that the requirement, in the definition of coloring, relating the colors to the permutations in G can be reformulated as follows. Since G acts on $\{1, 2, \ldots, j\}$, it also acts on the set of functions from this set into any set $|M|$ and, by restriction, on the subset $|M|^{j\neq}$. This induced action is given by

$$\pi(x_1, \ldots, x_j) = (x_{\pi^{-1}(1)}, \ldots, x_{\pi^{-1}(j)}).$$

Then the symmetry requirement in the definition says that $C : |M|^{j\neq} \to \{1, 2, \ldots, v\}$ commutes with the actions of G on its domain and (via h) on its range.

We make use of the action of G on $|M|^{j\neq}$ in the next definition by referring to its orbits. Recall that the orbit of an element (in a set acted on by a group) is the set of all elements obtainable from the given one by applying elements of the group. Recall also that distinct orbits are disjoint, so the orbits constitute a partition of the set.

Definition 3.3 A *random* Υ-structure M with base set $|M| = \{1, \ldots, n\}$ is obtained as follows. For each signum $R \in \Upsilon$, say of arity j, for each G-orbit in $|M|^{j\neq}$, pick one element $\vec{x}$ of the orbit, choose a value for $R^M(\vec{x})$ at random in $\{1, \ldots, v\}$ according to the probability distribution given in the signum R, and then define R^M on the rest of the orbit in the unique way that commutes with the action of G as required for a coloring.

This definition of random structure is independent of the choice of representatives in the orbits, because the probability distribution on $\{1, \ldots, v\}$ is $h(G)$-invariant.

Example 3.4 A j-ary relation can be regarded as a collection of 2-valued signa with various arities $\leq j$, one signum for each partition of the index set $\{1, \ldots, j\}$ (the arity of the signum being the number of blocks in the corresponding partition). Take the groups and (therefore) the homomorphism to be trivial, and take p to be the uniform distribution on the two values. Then a structure for this thesaurus amounts to just a j-ary relation, and a random structure is a random j-ary relation in the usual sense.

More generally, any finite first-order vocabulary can be converted in this way into a thesaurus, and structures of the thesaurus amount to structures of the original vocabulary. Furthermore, random structures for this thesaurus

are the random structures, with respect to the uniform distribution, for the original vocabulary.

By varying the probability distributions in the thesaurus, we get certain non-uniform probability distributions on random structures.

Example 3.5 Ordinary graphs (undirected, without loops or multiple edges), i.e., symmetric, irreflexive, binary relations, are the structures for the thesaurus with a single, 2-ary, 2-valued signum, the group G consisting of both permutations of $\{1, 2\}$ and the homomorphism h being trivial. With the uniform probability distribution on $\{1, 2\}$, random structures are random graphs in the sense defined earlier. With a different probability distribution on $\{1, 2\}$, we get random graphs with biased (but constant) edge probabilities.

Example 3.6 Modify the preceding example by taking h to be the identity homomorphism on the permutation group of $\{1, 2\}$. Now a structure amounts to a tournament. Only the uniform probability distribution on $\{1, 2\}$ satisfies the invariance requirement in the definition of signa, and it yields the usual notion of a random tournament. The fact that other probability distributions are not permitted corresponds to the fact that "a random tournament with biased edge directions" doesn't make sense — the two possible directions of any edge play symmetric roles so one can't say which one the bias should favor.

Example 3.7 Consider the thesaurus consisting of a single, 3-ary, 2-valued signum, where G is the group of all six permutations of $\{1, 2, 3\}$ and h is the non-trivial homomorphism from G to permutations of $\{1, 2\}$ (whose kernel is the subgroup of cyclic permutations in G). Then a structure for this thesaurus amounts to a set with a specified cyclic ordering on every 3-element subset.

By reading the rest of this paper with thesauri in mind, the reader will be able to prove:

Theorem 3.8 *Everything in this paper generalizes from the special case of random graphs to the case of random structures for any thesaurus.*

We omit all details of this but mention, to avoid possible confusion, that the generalized strong extension axioms (from Section 6) will involve the

probability distributions from the thesaurus's signa. The axioms will say that every quantifier-free type is realized at least half (or some other fraction < 1) as often as the expected number of realizers. That expected number can depend on the particular signa and on the particular values (replacing the truth values in types).

4 BGS Programs

In this section, we review the syntax and semantics of BGS programs, as well as the concept of active elements. These are the ingredients used in defining $\tilde{\text{C}}$PTime in [4].

The programs we consider take as inputs finite graphs, i.e., structures of the form $\langle I, A\rangle$, where I is the set of vertices (I stands for "input") and A the adjacency relation. Our computations take place in the universe $HF(I)$ of hereditarily finite sets over I. This universe is the smallest set containing all the members of I and all finite subsets of itself. In other words, it is the union of the sets $\mathcal{P}_n(I)$ defined inductively by

$$\begin{aligned} \mathcal{P}_0(I) &= I \\ \mathcal{P}_{n+1}(I) &= I \cup \mathcal{P}_{\text{fin}}(\mathcal{P}_n(I)), \end{aligned}$$

where $\mathcal{P}_{\text{fin}}(X)$ means the set of all finite subsets of X.

By the *rank* of an element x of $HF(I)$ we mean the smallest n such that $x \in \mathcal{P}_n(I)$. Thus, an atom has rank 0, and a set has the smallest rank that exceeds the ranks of all its members.

We identify the truth values `false` and `true` with the sets $0 = \emptyset$ and $1 = \{0\}$, respectively. Thus, relations can be regarded as functions taking values in $\{0, 1\}$.

We call a set x *transitive* if whenever $z \in y \in x$ then $z \in x$; equivalently, every set that is a member of x is also a subset of x. The *transitive closure* $TC(x)$ of a set x is the smallest transitive set having x as a member. Thus, it contains x, its members, their members, and so on. (Many authors define $TC(x)$ as the smallest transitive set having x as a subset; their $TC(x)$ is the same as ours except that it doesn't have x as a member.)

Definition 4.1 Our *function symbols* are

- the logical symbols, namely $=$ and the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, `true`, and `false`,

- the set-theoretic function symbols $\in$, $\emptyset$, Atoms, $\bigcup$, TheUnique, and Pair,
- the input predicate symbol A, and
- finitely many dynamic function symbols.

The intended interpretation of $\bigcup x$, where x is a family of sets and atoms, is the union of the sets in x (ignoring the atoms). If x is a set with exactly one member then TheUnique(x) is that member. Pair(x, y) means $\{x, y\}$. The input predicate A denotes the adjacency relation of the input graph. The intended meanings (and arities) of the other symbols should be clear; if they aren't then see the formal set-theoretic expressions in Section 11. We adopt the convention that if a function is applied to an inappropriate argument (like $\bigcup$ applied to an atom or A applied to sets) then the value is $\emptyset$.

Among the function symbols, $\in$, A, and the logical symbols are called *predicates* because their intended values are only `true` and `false`.

In addition to function symbols, we use a countably infinite supply of variables and certain symbols introduced in the following definitions of terms and rules.

Definition 4.2 *Terms* and *Boolean terms* are defined recursively as follows.

- Every variable is a term.
- If f is a j-ary function symbol and $t_1, \ldots, t_j$ are terms, then $f(t_1, \ldots, t_j)$ is a term. It is Boolean if f is a predicate.
- If v is a variable, $t(v)$ a term, r a term in which v is not free, and $\varphi(v)$ a Boolean term, then

  $$\{t(v) : v \in r : \varphi(v)\}$$

  is a term.

The construction $\{t(v) : v \in r : \varphi(v)\}$ binds the variable v.

In connection with $\{t(v) : v \in r : \varphi(v)\}$, we remark that, by exhibiting the variable v in $t(v)$ and $\varphi(v)$, we do not mean to imply that v must actually occur there, nor do we mean that other variables cannot occur there. We are merely indicating the places where v could occur free. The "two-colon"

notation $\{t(v) : v \in r : \varphi(v)\}$ is intended to be synonymous with the more familiar "one-colon" notation $\{t(v) : v \in r \wedge \varphi(v)\}$. By separating the $v \in r$ part from $\varphi(v)$, we indicate the computational intention that the set should be built by running through all members of r, testing for each one whether it satisfies φ, and collecting the appropriate values of t. Thus $\{t(v) : v \in r : v \in r'\}$ and $\{t(v) : v \in r' : v \in r\}$ are the same set, but produced in different ways. (Such "implementation details" have no bearing on our results but provide useful intuitive background for some of our definitions.)

Our official notation for terms, described above, contains propositional connectives but not quantifiers. We regard (bounded) quantifiers as abbreviations, namely

$$\begin{array}{lll} (\exists x \in r)\, \varphi(x) & \text{abbreviates} & 0 \in \{0 : x \in r : \varphi(x)\} \\ (\forall x \in r)\, \varphi(x) & \text{abbreviates} & \neg(\exists x \in r)\neg\varphi(x). \end{array}$$

Unbounded quantifiers are not available in BGS programs.

We shall also have occasion to use the set-forming construct $\{t(v) : v \in r : \varphi(v)\}$ with more than one variable in the role of v. This can be defined as

$$\{t(u,v) : u \in r, v \in s : \varphi(u,v)\} \text{ abbreviates } \bigcup\{\{t(u,v) : v \in s : \varphi(u,v)\} : u \in r : \mathtt{true}\},$$

and similarly with more than two variables.

Definition 4.3 *Rules* are defined recursively as follows.

- `Skip` is a rule.
- If f is a dynamic j-ary function symbol and $t_0, t_1, \ldots, t_j$ are terms, then

  $$f(t_1, \ldots, t_j) := t_0$$

  is a rule, called an *update rule*.
- If φ is a Boolean term and R_0 and R_1 are rules, then

  $$\mathtt{if}\ \varphi\ \mathtt{then}\ R_0\ \mathtt{else}\ R_1\ \mathtt{endif}$$

  is a rule, called a *conditional rule*.

- If v is a variable, r is a term in which v is not free, and $R(v)$ is a rule, then

  $$\texttt{do forall } v \in r,\ R(v) \texttt{ enddo}$$

  is a rule, called a *parallel combination.*

The construct `do forall` $v \in r,\ R$ `enddo` binds the variable v.

Convention 4.4 When the "else" part is `Skip`, we use `if` φ `then` R `endif` to abbreviate `if` φ `then` R `else Skip endif`.

We use `do in parallel` R_0, R_1 `enddo` as an abbreviation for

```
do forall v ∈ Pair(true, false)
   if v =true then R_0 else R_1
   endif
enddo
```

The `do in parallel` construct applied to more than two rules means an iteration of the binary `do in parallel` .

Definition 4.5 A *program* is a rule with no free variables.

Convention 4.6 By renaming bound variables if necessary, we assume that no variable occurs both bound and free, and no variable is bound twice, in any term or rule.

Throughout much of this paper, the context of our discussion will include a fixed program Π. In such situations, we adopt the following convention.

Convention 4.7 When we refer to a term or rule within Π, we mean a specific occurrence of the term or rule in Π.

Since a program Π has no free variables, every variable v occurring in it is bound exactly once, either by a term $\{t : v \in r : \varphi\}$ or by a rule `do forall` $v \in r,\ R$ `enddo`.

Definition 4.8 If v is bound by $\{t : v \in r : \varphi\}$, then the *scope* of v consists of the exhibited occurrence of v as well as t and φ. If v is bound by `do forall` $v \in r,\ R$ `enddo`, then the *scope* of v consists of its exhibited occurrence and R. In both cases, the *range* of v is r. Notice that the range of v is not in the scope of v.

This concludes the description of the syntax of BGS. The semantics will be defined in great detail in Section 11, but we give a brief informal explanation here. A *state* of a computation is a structure with base set $HF(I)$, with all the logical and set-theoretic function symbols and A interpreted as described immediately after Definition 4.1, and with the dynamic function symbols interpreted arbitrarily. A *pebbled state* consists of a state together with an assignment of values in $HF(I)$ to finitely many variables. When the relevant variables and their ordering are understood, we think of a pebbled state as a state plus a tuple of elements, $(H, a_1, \ldots, a_j)$, where a_i is the value assigned to the i^{th} variable. For brevity, we often use vector notation $\vec{a}$ for $(a_1, \ldots, a_j)$.

Terms and rules can be evaluated in any pebbled state that assigns values to all their free variables. The value of a term is a member of $HF(I)$ computed using the interpretations of the function symbols. The value of a rule is a set of *updates*, each changing the value of a dynamic function symbol at some tuple of arguments. An update rule $f(t_1, \ldots, t_j) := t_0$ produces a single update, changing the value of f at the tuple $(a_1, \ldots, a_j)$ to a_0, where each a_i is the value of the corresponding term t_i. A conditional rule `if` φ `then` R_0 `else` R_1 `endif` produces the same updates as R_0 or R_1 according to whether the value of φ is `true` or `false`. A parallel composition `do forall` $v \in r$, $R(v)$ `enddo` produces all the updates produced by $R(v)$ in all the pebbled states obtained from the current one by assigning arbitrary members of the value of r as values to v. And of course `Skip` produces no updates. If the update set of the program Π does not contain any conflicting updates — attempting to change the value of the same function at the same argument tuple to two different values — then the next state of the computation is obtained from the present state by applying all these updates. If there is a conflict, then the next state is the same as the present one. (Notice that, since a program has no free variables, it can be evaluated in an unpebbled state.)

The transition from one state of a computation to the next is often called a *macrostep*, where "macro" refers to the large amount of work that may be involved, especially because of parallelism. By contrast, a *microstep* would be a single action such as looking up the value of a function at some argument tuple or updating such a value. The number of microsteps in a computation provides a more honest measure of computation time than the number of macrosteps. As explained in [4], imposing a polynomial bound on the number of microsteps amounts to imposing polynomial bounds on both the number of macrosteps and the number of elements of $HF(I)$ that are active in the

sense of the following definition. We use the latter formulation as the official definition of choiceless polynomial time, $\tilde{\mathrm{C}}$PTime. Thus, we avoid the need to precisely define microsteps; we incur the need to precisely define active, but this is a concept that is directly needed in the proof of the zero-one law.

Definition 4.9 The *critical* elements of a pebbled state $(H, \vec{a})$ are

- all the atoms and the set I of atoms,
- the Boolean values `true` and `false`,
- all values of dynamic functions,
- all components of all tuples where dynamic functions have values other than $\emptyset$, and
- all components of $\vec{a}$.

An element is *active* if it is in the transitive closure of some critical element.

For ordinary (unpebbled) states, this definition differs from that in [4] only in that the set I of atoms is critical and therefore also active.

We are now in a position to give the example, promised in Section 2, of a BGS program Π together with polynomial bounds on the number of steps and the number of active elements, such that not all of the following three classes of graphs have asymptotic probability 0 or 1: the graphs on which Π halts with output `true` (within the prescribed bounds on steps and active elements), the analogous class for `false`, and the class of graphs on which Π fails to halt within the prescribed bounds. The required Π can be taken to be

```
do forall x ∈ Atoms
do forall y ∈ Atoms
   do in parallel
      if A(x, y) then f(Pair(x, y)) := true endif,
      Output := true,
      Halt := true
   enddo
enddo enddo
```

This program Π only executes once before halting, so we can take the polynomial bound on the number of steps to be 2 and ignore this bound. The

number of active elements is $n+3+e$ where n and e are the numbers of vertices and edges in the input graph. (The active elements are the n atoms, the e two-element sets corresponding to edges, the two boolean values, and the set I of atoms.) In a large random graph, the expected value of e is $n(n-1)/4$, i.e., half the number of possible edges, but small fluctuations about this value are probable. The following easy lemma gives the information we need about these fluctuations.

Lemma 4.10 *Consider N independent random trials, each succeeding with probability $1/2$. Then the probability that the number of successes in these N trials is $\leq N/2$ approaches $1/2$ as $N \to \infty$.*

Proof For odd values of N, the probability in question is exactly $1/2$ by symmetry; k successes are exactly as likely as $N-k$. For even values of N, the probability exceeds $1/2$ because the event "number of successes $\leq N/2$" includes all, rather than just half, of the cases where the number of successes is exactly $N/2$. So the probability exceeds $1/2$ by

$$\frac{1}{2} \cdot \frac{\binom{N}{N/2}}{2^N} = \frac{N!}{2^{N+1}((N/2)!)^2}.$$

By Stirling's asymptotic formula for factorials, $n! \sim n^n\sqrt{2\pi n}/e^n$, we find that this excess probability is asymptotically

$$\frac{N^N\sqrt{2\pi N}}{e^N 2^{N+1}} \cdot \frac{e^N}{(N/2)^N 2\pi(N/2)} = \frac{1}{\sqrt{2\pi N}},$$

and this approaches 0 as $N \to \infty$. □

According to the lemma, the asymptotic probability that $e \leq n(n-1)/4$ is $1/2$. So, if we impose a bound of $n+3+n(n-1)/4$ on the number of active elements, then with asymptotic probability $1/2$ our program will halt with output `true`, and with asymptotic probability $1/2$ it will fail to halt because it cannot execute its single computation step without activating too many elements.

5 Outline of Proof of Zero-One Law

The proof of the zero-one law for $\tilde{\text{C}}$PTime involves several ingredients. The first is to show that whether a BGS program halts at a particular step with a

particular output can be defined by a first-order sentence over the structure $H = \langle HF(I), \in, I, A\rangle$, with a number of variables that does not depend on the number of steps. A natural approach to proving Theorem 2.2 would then be to use Ehrenfeucht-Fraïssé games and produce winning strategies for the duplicator, to show that such sentences have the same truth value for almost all input graphs $\langle I, A\rangle$. Unfortunately, that isn't true; for example, the parity of $|I|$ can be defined by a first-order statement over H, saying that there is a bijection between I and an even von Neumann ordinal.

The second ingredient in the proof is to realize that bijections as in this counterexample will not be relevant to polynomial time BGS computations. Like the first ingredient, this idea was already present in [4], but it was considerably easier to implement there. In [4], the role of the input graph $\langle I, A\rangle$ was played by a set with no additional structure, or at most with a partition into a fixed number of pieces. In such a situation, symmetry considerations ensure that, if some bijection b between the atoms and an ordinal were used in the computation, then many others would be used as well, namely all those obtainable from b by automorphisms of the input. But then the polynomial bound on the number of active elements would be violated. In our present situation, though, these symmetry considerations are unavailable, because almost all finite graphs have no non-trivial automorphisms.

Therefore, a subtler approach to symmetry is needed. Shelah introduces a suitable class of partial automorphisms (for any given program Π and any given polynomial bound on the number of active elements) and shows that it leads to an appropriate notion of symmetry. Here "appropriate" means that the symmetry requirements are restrictive enough to provide winning strategies for the duplicator yet are lenient enough to include all the sets actually involved in a computation of Π, limited by the given bound on active elements.

To show that the symmetry requirements are restrictive enough depends on showing that, just as a permutation of the atoms extends to an automorphism of the set-theoretic structure $HF(I)$, a partial automorphism of the graph of atoms extends to a partial automorphism of $HF(I)$. The proof of the existence and good behavior of such extensions is considerably longer than the corresponding proof for total permutations.

The hardest part of the proof is showing that this notion of partial symmetry is lenient enough: The computation involves only symmetric sets. This will be proved by a double induction, first on the stages of the computation and second, within each stage, on the subterms and subrules of Π. The in-

ner induction proceeds along a rather unusual ordering of the subterms and subrules, which we call the computational ordering.

In this double induction, it is necessary to strengthen the induction hypothesis, to say not only that every set x involved in the computation is symmetric but also that all sets x' obtained from x by applying suitable partial automorphisms are also involved in the computation. The assumed bound on the number of active elements will imply a polynomial bound on the number of involved elements. (Not all involved elements are active, but there is a close conection between the two.) That means that the number of x''s is limited, which in turn implies, via a highly non-trivial combinatorial lemma, that x is symmetric.

The traditional extension axioms, as in [8], are satisfied by almost all graphs and are adequate to produce the duplicator's strategies that we need, but they are not adequate to imply the combinatorial lemma needed in the symmetry proof. For this purpose, we need what we call strong extension axioms, saying that every possible type over a finite set is not only realized but realized by a large number of points.

We devote the next few sections to a discussion of the strong extension axioms. Thereafter, we shall give the set-theoretic definitions of the behavior of BGS computations. These provide the first of the ingredients in the proof outline above; they also make precise our informal explanations of the meaning of terms and rules in Section 4. Next, we shall describe in detail the other concepts mentioned in the outline: computational order, objects involved in a computation, extension of partial automorphisms of the graph of atoms to partial automorphisms of $HF(I)$. Then we shall prove the highly non-trivial combinatorial lemma that ensures that all involved objects are symmetric. Finally, we assemble all these ingredients to complete the proof.

6 Strong Extension Axioms

Extension axioms (for graphs) assert the existence of vertices in any possible "configuration" relative to finitely many given vertices; strong extension axioms assert not only existence but plentitude.

More precisely, a k-parameter *type* is a formula $\tau(y, x_1, \ldots, x_k)$ of the form

$$\bigwedge_{1 \le i \le k} (y \neq x_i \wedge \pm(yAx_i)).$$

Here $\pm$ before a formula means that the formula may or may not be negated. So τ specifies the adjacency and non-adjacency relationships betwen y and the k parameters x_i; in addition, it says that y is distinct from the x_i's (which is redundant when yAx_i is not negated, since the adjacency relation A is irreflexive). The *extension axiom* $\mathrm{EA}(\tau)$ associated to a type τ is

$$\forall x_1, \ldots, x_k \left(\left(\bigwedge_{1 \le i < j \le k} x_i \neq x_j \right) \rightarrow \exists y \, \tau(y, x_1, \ldots, x_k) \right).$$

For a fixed k, there are 2^k of these extension axioms, because of the k choices for the $\pm$ signs in τ. We write EA_k for their conjunction together with the statement that there are at least k vertices (so that the $\mathrm{EA}(\tau)$'s aren't vacuous). Thus, EA_k says that every possible configuration for a vertex y, relative to k distinct, given vertices, is realized at least once. (We have deviated a bit from the standard terminology of model theory. There, one would replace the variables x_i by names for specific, distinct vertices, and one would call the resulting modified τ a generator for a k-parameter, quantifier-free 1-type. Since we won't need types of any other kind, we shortened the terminology to fit our needs.)

We say that a graph satisfies the *strong extension axiom* $\mathrm{SEA}(\tau)$ if, for every k distinct vertices $x_1, \ldots, x_k$, there are at least $\frac{1}{2}n/2^k$ vertices y satisfying $\tau(y, x_1, \ldots, x_k)$. (Unlike the extension axioms, strong extension axioms are not first-order formulas.) We write SEA_k for the conjunction of all 2^k of the strong extension axioms $\mathrm{SEA}(\tau)$ as τ ranges over all the k-parameter types, together with the statement that there are at least k vertices. Thus, SEA_k says that each possible configuration of y relative to k distinct x_i's is realized not just once (as EA_k says) but fairly often, $\frac{1}{2}n/2^k$ times.

Why $\frac{1}{2}n/2^k$? In a random graph with n vertices, the probability that an arbitrary vertex b, different from $a_1, \ldots, a_k$, satisfies $\tau(b, a_1, \ldots, a_k)$ is $1/2^k$, so the expected number of vertices b satisfying $\tau(b, a_1, \ldots, a_k)$ is $(n - k)/2^k$. So it is reasonable to expect, and we shall prove below, that with high probability there are at least half the expected number (even with the factor $n-k$ increased slightly to n), i.e., $\frac{1}{2}n/2^k$, vertices b satisfying $\tau(b, a_1, \ldots, a_k)$. The factor $\frac{1}{2}$ could be replaced here with any positive constant $\alpha < 1$; that gives a strong extension axiom SEA_k^α.

As indicated in Section 2.1, the class $\mathcal{C}$ of graphs in Theorem 2.2 consists of those graphs that have sufficiently many vertices and satisfy SEA_k for a sufficiently large k. (In fact, the requirement that the number of vertices

be large can be subsumed by the SEA requirement, perhaps at the cost of increasing k.) Thus, it is crucial for our purposes that each strong extension axiom, like each (ordinary) extension axiom, is satisfied by almost all graphs.

Proposition 6.1 *For each k, the asymptotic probability of SEA_k is 1.*

The proof of this proposition depends on a "large deviation" inequality of the sort given in Chernoff's paper [7] and Loève's book [14, Section 18]. The references we have found prove stronger results than we need and therefore give more complicated proofs than we need. We present here a simple proof of an inequality strong enough for our purposes.

Lemma 6.2 *Fix numbers α, r in the open interval $(0,1)$. There is a constant c, also in $(0,1)$, such that the following is true for every positive integer m. Let X be the number of successes in m independent trials, each trial having probability r of success. Then Prob $[X \leq \alpha mr] \leq c^m$.*

That is, the probability that the number of successes (X) is smaller than the expected number (mr) by at least the (fixed) factor α decreases exponentially as a function of the number m of trials.

Proof We begin with the well-known observation that, if Z is a nonnegative random variable and q is a positive real number, then

$$\text{Prob}\,[Z \geq q] \leq \frac{E(Z)}{q},$$

where E means "expectation." Indeed,

$$\begin{aligned} E(Z) &= E(Z|Z \geq q)\text{Prob}\,[Z \geq q] + E(Z|Z < q)\text{Prob}\,[Z < q] \\ &\geq E(Z|Z \geq q)\text{Prob}\,[Z \geq q] \\ &\geq q \cdot \text{Prob}\,[Z \geq q]\,. \end{aligned}$$

We apply this with $Z = \exp[t(mr - X)]$, where t is a positive parameter to be chosen later. Thus, we have

$$\begin{aligned} \text{Prob}\,[X \leq \alpha mr] &= \text{Prob}\,[Z \geq \exp[t(mr - \alpha mr)]] \\ &= \text{Prob}\,[Z \geq \exp[tmr(1-\alpha)]] \\ &\leq \frac{E(Z)}{\exp[tmr(1-\alpha)]}. \end{aligned}$$

We continue by computing $E(Z)$. The random variable $mr - X$ can be viewed as the sum, over all m trials, of $r - S$, where S is 1 if the trial is a success and 0 if not. Thus, Z is the product over all trials of $\exp[t(r-S)]$. But the trials are independent, so the expectation of this product is the product of the individual expectations. For each individual trial, we have

$$\begin{aligned} E(\exp[t(r-S)]) &= r \cdot \exp[t(r-1)] + (1-r) \cdot \exp[t(r-0)] \\ &= \exp[tr](r\exp[-t] + 1 - r). \end{aligned}$$

Therefore,

$$E(Z) = \exp[tmr](r\exp[-t] + 1 - r)^m.$$

Substituting this into the inequality for $\text{Prob}\,[X \leq \alpha mr]$, we find

$$\begin{aligned} \text{Prob}\,[X \leq \alpha mr] &\leq \left(\frac{\exp[tr](r\exp[-t] + 1 - r)}{\exp[tr(1-\alpha)]}\right)^m \\ &= [\exp[t\alpha r] \cdot (r\exp[-t] + 1 - r)]^m. \end{aligned}$$

So the lemma will be proved if we can find a positive t for which the value of

$$f(t) = \exp[t\alpha r] \cdot (r\exp[-t] + 1 - r)$$

is in the open interval $(0, 1)$, for then this value can serve as the required c.

Notice that $f(0) = 1$ and that

$$f'(t) = \alpha r \exp[t\alpha r] \cdot (r\exp[-t] + 1 - r) + \exp[t\alpha r] \cdot (-r)\exp[-t].$$

Thus, $f'(0) = \alpha r - r < 0$ (because $\alpha < 1$ and $r > 0$). Therefore, any sufficiently small positive t will give $0 < f(t) < 1$ as required. □

Remark 6.3 The best, i.e., smallest value of c obtainable by the preceding argument is the minimum value of $f(t)$. A routine calculation, setting $f'(t) = 0$, shows that this minimum is

$$\left(\frac{1-r}{1-\alpha r}\right)^{1-\alpha r} \cdot \left(\frac{1}{\alpha}\right)^{\alpha r}.$$

Notice that this is the weighted geometric mean of two quantities whose correspondingly weighted arithmetic mean is

$$\left(\frac{1-r}{1-\alpha r}\right)\cdot(1-\alpha r)+\left(\frac{1}{\alpha}\right)\cdot(\alpha r)=1.$$

Since the two quantities are not equal to 1 (as $\alpha < 1$), the arithmetic-geometric mean inequality shows again that the optimal c is smaller than 1.

Proof of Proposition 6.1 We shall show that, for each fixed k-parameter type τ, the probability that SEA(τ) fails, in a random graph on vertex set $\{1, 2, \ldots, n\}$, approaches 0 as $n \to \infty$. Then, as SEA_k is the conjunction of a fixed number 2^k (independent of n) of SEA(τ)'s, its probability of failure also approaches 0, as required.

So we concentrate henceforth on a single τ. Temporarily, also concentrate on k specific, distinct vertices $a_1, \ldots, a_k \in \{1, 2, \ldots, n\}$. Let X be the number of vertices b satisfying $\tau(b, a_1, \ldots, a_k)$. In a random graph, each of the $n-k$ vertices other than $a_1, \ldots, a_k$ has probability $1/2^k$ of satisfying τ, and these $n-k$ trials are independent. So, applying the lemma with $m = n-k$, with $r = 1/2^k$, and with some α in the interval $(\frac{1}{2}, 1)$, and noting that, as $\alpha > \frac{1}{2}$, we have $\alpha \cdot (n-k) \geq \frac{1}{2}n$ for large n, we obtain some $c \in (0,1)$ such that

$$\begin{aligned}\mathrm{Prob}\left[X < \frac{1}{2}n/2^k\right] &\leq \mathrm{Prob}\left[X < \alpha(n-k)/2^k\right] \\ &\leq c^{n-k}.\end{aligned}$$

This bounds the probability that our specific choice of $a_1, \ldots, a_k$ is a counterexample to SEA(τ).

Now un-fix $a_1, \ldots, a_k$. Since the number of choices for this k-tuple is $\leq n^k$, the probability that at least one choice gives a counterexample, i.e., the probability that SEA(τ) fails, is at most

$$n^k c^{n-k}.$$

Since $0 < c < 1$, this bound approaches 0 as $n \to \infty$. □

The same proof can be used to show that, for each k and each $\alpha \in (0,1)$, the axiom SEA_k^α has asymptotic probability 1.

Remark 6.4 Kolaitis and Vardi introduced in [13] a notion of "richness" intermediate between the ordinary and the strong extension axioms. It requires that each k-parameter type be realized at least $\sqrt{n}$ times, where n is the total number of vertices. It appears that this notion could be used in place of our strong extension axioms in proving the zero-one law. The main change needed would be a doubling of the exponent $q+1$ in Theorem 15.2 below, to compensate for the square root in the definition of richness.

7 Inadequacy of Extension Axioms

Do we really need strong extension axioms? The zero-one law for first-order logic [10, 9] is based on the (ordinary) extension axioms: for every first-order sentence φ, there exists k such that EA_k implies φ or EA_k implies $\neg\varphi$. The same holds for fixed-point logic FO+LFP [3, 16] and for the infinitary logic $L^{\omega}_{\infty,\omega}$ [12] (see also [8]). Might extension axioms suffice to define the class $\mathcal{C}$ in Theorem 2.2? Then the use of strong extension axioms would be merely an artifact of the proof. We show in this section that this is not the case. Extension axioms are too weak to support the zero-one law for $\tilde{\mathrm{C}}$PTime. We give an example of a single polynomial time BGS program that separates structures satisfying arbitrarily many extension axioms. So strong extension axioms are really needed for the $\tilde{\mathrm{C}}$PTime zero-one law. (See, however, Section 18 for a restricted situation where extension axioms suffice.)

Though our general policy has been, for expository purposes, to concentrate on algorithms whose inputs are graphs, this example will use as input a graph together with a single distinguished vertex, i.e., a rooted graph. That is, we add a constant symbol d to the vocabulary $\{A\}$ of graphs. We expect that a similar example could be given without introducing the constant symbol, but we have not seen how to do this.

It should be noted that the zero-one laws for the usual logics, like $L^{\omega}_{\infty,\omega}$, continue to hold, and to follow from the extension axioms, in the presence of a distinguished vertex. (They fail when there are two distinguished vertices, simply because these two are adjacent with probability $\frac{1}{2}$.) It should also be noted that, instead of using a distinguished vertex, we could add a unary relation R to the vocabulary and modify the extension axioms to specify, in addition to adjacency information, whether y should satisfy R. In that version of the construction, R would play the role played in our proof by the set of neighbors of d.

Proposition 7.1 *There is a polynomial time BGS program* Π *such that, for any given* k*, there are two rooted graphs, both satisfying* EA_k*, such that* Π *produces output* `true` *on one of them and* `false` *on the other.*

Proof We begin by exhibiting the BGS program Π; the polynomial bounds on the number of macrosteps and the number of active elements will be n and $2n+3$, respectively. The program Π computes the parity of the maximum size of a clique containing the distinguished vertex d. It does this by building up the collection of all i-element subsets of $\{x : xAd\}$ for $i = 0, 1, \ldots$, checking at each step whether any cliques remain. One essential ingredient of the proof will be that d has so few neighbors in our graphs that the time used by this computation is polynomial relative to the sizes of these graphs.

Π uses four dynamic 0-ary function symbols: Halt, Output, Mode, and C. Recall that in the initial state of a computation these have the value $\emptyset = \texttt{false} = 0$. The program Π is

```
do in parallel
   if Mode = 0 then
      do in parallel
         C := {∅}, Mode := 1
      enddo
   endif
   if Mode = 1 then
      do in parallel
         C := {x ∪ {y} : x ∈ C, y ∈ Atoms : yAd ∧ y ∉ x},
         Output := ¬Output, Mode := 2
      enddo
   endif
   if Mode = 2 then
      if (∃x ∈ C)(∀u, v ∈ x) uAv
      then Mode := 1
      else Halt := true
      endif
   endif
enddo.
```

After the first part of Π, with Mode $= 0$, has been executed, C is initialized to $\{\emptyset\}$, the family of 0-element subsets of the set R of neighbors of

d. After i executions of the part with Mode = 1, C has become the family of i-element subsets of R. The part with Mode = 2 checks whether there are any cliques in C. If so, we return to Mode 1 to enlarge the sets in C; if not, then the common size i of the sets in C is one more than the maximum size of a clique included in R. Since Output reverses its truth value at each Mode 1 step and since it is initially `false`, we see that, the final value of Output is `true` if and only if the maximum clique size in R is even, if and only if the maximum size of a clique containing d is odd.

For future reference, we estimate the amount of work done by this program. Writing n for the number of vertices in the input graph, r for the number of neighbors of d, and s for the maximum size of a clique among these neighbors, we find that the Mode 0 part of Π is executed once and the Mode 1 and Mode 2 parts are executed $s+1$ times each. So the whole computation takes $2s+3$ macrosteps. The elements that the computation activates are the subsets of R of cardinality at most $s+1$, the $s+2$ values taken by C, and the number 2. (The numbers 0 and 1 were identified with the truth values and were therefore already active in the initial state. Since 0 is also among the subsets of R and $1 = \{0\}$ is also among the values of C, we have a slight overcount of activated elements.) For non-trivial values of r and s, the number of activated elements is easily seen to be majorized by $(r+1)^{s+1}$. The initially active elements are the n atoms, the set of atoms, and the two truth values. Thus, the total number of active elements in this computation is at most $n+3+(r+1)^{s+1}$. We shall design our graphs to have quite small r and s (relative to n), so that $n+3+(r+1)^{s+1}$ is below the bound $2n+3$ that we imposed on the number of active elements, and $2s+3$ is below the bound n on the number of macrosteps.

The graphs required in the proposition will be described in three steps. First, we give a general description depending on two parameters: the (large) number n of vertices and the (much smaller but still rather large) number r of neighbors of d. Second, leaving n arbitrary (but large), we prescribe two values for r, to produce the two graphs we want. Finally, we fix n so large that these graphs have all the required properties. Actually, the description in the first step involves randomization, and rather than fixing n in the last step we simply show that, for all sufficiently large n, the graphs have the required properties with high probability. This clearly suffices for the existence claim in the proposition.

Given n and r, we build a (random) graph $G(n,r)$ as follows. The vertex set consists of the distinguished vertex d and $n-1$ others which we denote by

$1, 2, \ldots, n-1$. (This introduces an ambiguity, since these vertices are atoms in $HF(I)$ and the same symbols denote natural numbers (finite von Neumann ordinals) which are sets. Fortunately, the ambiguity never leads to any possibility of confusion.) d is adjacent to the vertices $1, 2, \ldots, r$ and no others. The rest of the adjacency relation is chosen at random; flip independent, fair coins for all potential edges to decide whether to include them in the graph. This completes the description of $G(n, r)$. We record for future reference that the subgraph induced by the set $R = \{1, 2, \ldots, r\}$ of neighbors of d is a random graph (in the usual sense) on r vertices.

The next part of the proof, choosing r as a function of n, is the most delicate. We need r small enough so that Π stays within the bound on active elements, but if we take r too small then $G(n, r)$ will violate the extension axioms. We need the following result from [6, Section XI.1].

Lemma 7.2 *There is a function ρ from natural numbers to natural numbers with the following two properties. First,*

$$C_1 s 2^{s/2} \leq \rho(s) \leq C_2 s 2^{s/2}$$

for certain positive constants C_1 and C_2. Second, if $p(s)$ denotes the probability that a random graph on $\rho(s)$ vertices has maximum clique size exactly s, then $p(s) \to 1$ as $s \to \infty$.

Actually, Bollobás proves a far more precise result. The constants in the lemma can be taken to be any constants satisfying $C_1 < 1/(e\sqrt{2}) < C_2$ provided s is sufficiently large. All we shall need, however, is the lemma as stated.

Using this lemma, we associate to each (large) n two values of r as follows. Let s and s' be the two largest integers below $3 \log\log n$. (We use log to mean base 2 logarithm and ln to mean base e logarithm.) Let $r = \rho(s)$, $r' = \rho(s')$, $G = G(n, r)$, and $G' = G(n, r')$. According to the lemma, when n is large enough there is a very high probability that, among the neighbors of d, the largest clique in G has size s and similarly for G' and s'. In particular, since s and s' are consecutive integers, the program Π will (unless it runs out of time) produce output `true` for one of G and G' and `false` for the other.

We next address the question whether Π with these inputs G and G' succeeds in carrying out its computation within the bounds on the number of macrosteps (n) and active elements ($2n + 3$). We already computed the

number of macrosteps and an upper bound for the number of active elements. In the present context, these are, with high probability for large n,

$$2s + 3 < 3 \log\log n + 3 < n$$

and

$$n + 3 + (r+1)^{s+1} < n + 3 + \left(C_3 s 2^{s/2}\right)^{s+1}$$

for G (where C_3 is slightly larger than C_2 to compensate for changing from $r+1$ to r), and similarly for G' with s' and r' in place of s and r. So the bound on macrosteps is satisfied with high probability for sufficiently large n. As for the bound on active elements, we must show that

$$\left(C_3 s 2^{s/2}\right)^{s+1} \leq n.$$

To this end, we first compute that, since $s < 3 \log\log n$,

$$C_3 s 2^{s/2} < C_3 \cdot 3 \log\log n \cdot (\log n)^{3/2} < (\log n)^2$$

for large n. The desired inequality follows because the logarithm of its left side is at most

$$(s+1) \log\left((\log n)^2\right) < (3 \log\log n + 1) \cdot 2 \log\log n < \log n.$$

To complete the proof of the proposition, we must still verify that (with high probability, when n is large) G and G' satisfy EA_k. We give the argument for G; it applies equally well to G'. Recall that G was defined as $G(n, r)$ with n sufficiently large and $r = \rho(s) \geq C_1 s 2^{s/2}$, where s is one of the largest two integers below $3 \log\log n$. In particular, $s > 2 \log\log n$ (for large n), and so

$$r \geq C_1 \cdot 2 \log\log n \cdot \log n = \beta(n) \log n,$$

where all we need to know about $\beta(n) = C_1 \cdot 2 \log\log n$ is that it tends to infinity with n. We can therefore complete the proof by showing that, for each fixed k-parameter type τ, the probability that $G(n, r)$ satisfies $\mathrm{EA}(\tau)$ is close to 1 when n is large and $r \geq \beta(n) \log n$.

Fix, therefore, an arbitrary k-parameter type τ, and temporarily fix values $a_1, \ldots, a_k$ for its parameters. There are three cases to consider, according to whether d is among the parameters and, if it is, whether τ says y should be adjacent or non-adjacent to d. Since d has so few neighbors (only r,

compared with approximately $n/2$ for other vertices), the probability that $\tau(y, a_1, \ldots, a_k)$ holds is smallest in the case where some a_i is d and τ says y is adjacent to d. We calculate this worst case first and then indicate the changes for the other cases.

So suppose d is one of the parameters and τ requires y to be adjacent to d. Then the only candidates for values of y satisfying τ are the r neighbors $1, 2, \ldots, r$ of d. For any one of these neighbors b, distinct from the other parameters, the probability that it satisfies τ, i.e., that it satisfies $k-1$ additional adjacency or non-adjacency requirements each of which has probability $\frac{1}{2}$, is $1/2^{k-1}$. Since these probabilities are independent for different b's and since there are at least $r-k+1$ available b's (the r neighbors of d minus at most $k-1$ that are among the other parameters),

$$\begin{aligned} \text{Prob}\left[\text{no } b \text{ satisfies } \tau(y, a_1, \ldots, a_k)\right] &\leq \left(1 - \frac{1}{2^{k-1}}\right)^{r-k+1} \\ &\leq e^{-(r-k+1)/2^{k-1}}, \end{aligned}$$

where we used the fact that $1 - t \leq e^{-t}$.

In the case where d is one of the parameters but τ says that y is not adjacent to d, the computation works the same way but with $n-r-k$ in place of $r-k+1$. In the case where d is not one of the parameters, the result is again similar but with $(n-k-1)/2$ in place of $r-k+1$. In either of these cases, $r-k+1$ has been replaced with something larger (when n is large), so the upper bound for the probability of failure is even smaller than in the first case. Summarizing, we have, for every choice of k distinct parameters, an upper bound of $e^{-(r-k+1)/2^{k-1}}$ for the probability that τ has no solution. Therefore, the probability that $\text{EA}(\tau)$ fails is at most

$$\binom{n}{k} e^{-(r-k+1)/2^{k-1}} \leq n^k e^{-(r-k+1)/2^{k-1}}.$$

To estimate this, we consider its natural logarithm, which is at most

$$k \ln n - \frac{r-k+1}{2^{k-1}} \leq -\frac{\beta(n) \log n}{2^{k-1}} + k \ln n + \text{constant}.$$

Since $\beta(n) \to \infty$ as $n \to \infty$, the right side of this formula tends to $-\infty$. So our upper bound for the probability that $\text{EA}(\tau)$ fails tends to 0 as $n \to \infty$. $\square$

It should perhaps be pointed out explicitly that the graphs constructed in the preceding proof violate SEA_1, for the number of neighbors of the special vertex d is far smaller than the $n/4$ that SEA_1 would require.

8 Rigidity and Hamiltonicity

In this section, we discuss two familiar properties of random (finite) graphs.

Definition 8.1 A graph is *rigid* if its only automorphism is the identity.

Definition 8.2 A graph is *hamiltonian* if it includes a cycle containing all its vertices.

It is known (see [6, Chapters VIII and IX]) that both of these properties have asymptotic probability 1. It is also known (see [5]) that neither of them follows from any extension axiom. Since we know, from the preceding section, that the strong extension axioms are genuinely stronger than the (ordinary) extension axioms, it is reasonable to ask whether some SEA_k implies rigidity or hamiltonicity. We can give a complete (negative) answer for rigidity but only a partial answer for hamiltonicity.

Proposition 8.3 *For any k, there exists a non-rigid graph satisfying SEA_k.*

Proof We use the same randomizing construction as in [5]. Let l be a large natural number, and let $G(l)$ have the integers from $-l$ through l as vertices. We require that, if a is adjacent to b, then $-a$ is adjacent to $-b$; this ensures that $G(l)$ is not rigid, for $a \mapsto -a$ is a non-trivial automorphism. Except for this symmetry requirement, $G(l)$ is random. That is, for each pair of corresponding potential edges $\{a, b\}$ and $\{-a, -b\}$ (which may be just a single potential edge if $a = -b$), we decide whether to include both or neither in $G(l)$ by flipping a fair coin. We shall show that, for any fixed k, the probability that the graph $G(l)$ satisfies SEA_k tends to infinity with l.

As usual, it suffices to check this for $\mathrm{SEA}(\tau)$ for every single k-parameter type τ. So let τ be given, and temporarily fix values $a_1, \ldots, a_k$ for the parameters. Let b range over positive vertices different from all the $\pm a_i$'s. For each such b, the probability that it satisfies τ with our fixed parameters is $1/2^k$, and for different b's these probabilities are independent. By Lemma 6.2, for any $\alpha \in (0, 1)$, the probability that fewer than $\alpha \cdot \frac{l-k}{2} \cdot 2^{-k}$ of these

b's satisfy τ decreases exponentially with l. Of course the same goes for negative b's. Therefore, the same goes for the probability that fewer than $\alpha \cdot (l-k) \cdot 2^{-k}$ vertices altogether satisfy τ. (We had to consider positive and negative b's separately, because their behavior is not independent as required for application of Lemma 6.2.) Taking α slightly larger than $\frac{1}{2}$, to get $\alpha \cdot (l-k) > \frac{1}{2} \cdot l$ for large l, we find that the probability that our fixed $a_1, \ldots, a_k$ constitute a counterexample to SEA(τ) decreases exponentially with l.

Now un-fix the parameters a_i. Notice that the number of choices of the parameters is bounded by a polynomial in l, namely $(2l+1)^k$. Therefore, the probability that SEA(τ) fails is small for large l. □

We remark that the proof shows that even the axioms SEA_k^α for arbitrary $\alpha \in (0,1)$ do not imply rigidity.

Proposition 8.4 *For any k and any $\alpha \in (0, \frac{1}{2})$, there is a graph that satisfies SEA_k^α but is not hamiltonian.*

Proof We simplify a randomizing construction from [5]. Let l be a large natural number, and let $G(l)$ be the graph produced as follows. The vertices are the natural numbers from 0 to $2l$; we call the first l of these vertices *friendly* and the remaining $l+1$ *unfriendly*. No two unfriendly vertices will be adjacent. For each potential edge subject to this constraint, i.e., for each two vertices of which at least one is friendly, flip a fair coin to decide whether to include that edge in $G(l)$.

No matter what happens in the randomization, this graph cannot be hamiltonian. Indeed, in any cycle, at most half the vertices can be unfriendly, since unfriendly vertices are not adjacent. But in the whole graph, more than half of the vertices are unfriendly.

To complete the proof, we show that, with high probability for large l, $G(l)$ satisfies SEA_k^α. As usual, it suffices to show, for each fixed k-parameter type τ and each choice of values $a_1, \ldots, a_n$, that the probability that fewer than $\alpha \cdot (2l+1) \cdot 2^{-k}$ vertices b satisfy τ (with the chosen parameters) decreases exponentially as a function of l. So let τ and the parameters be fixed, and let b range over friendly vertices distinct from all the parameters. So there are at least $l-k$ values for b, and each satisfies τ with probability 2^{-k}, these events being independent for different b's. We apply Lemma 6.2 to this situation, but with the α of the lemma replaced by some $\beta > 2\alpha$ (where this α is the one in the present proposition). Since $\alpha < \frac{1}{2}$, we can find such a $\beta < 1$, so

Lemma 6.2 is applicable. It gives us exponentially decreasing probability for the event that fewer than $\beta \cdot (l-k) \cdot 2^{-k}$ friendly vertices satisfy τ with the chosen parameters. But since $\beta > 2\alpha$, we have $\beta \cdot (l-k) > \alpha \cdot (2l+1)$ once l is large enough. Thus, we also have exponentially decreasing probability for the event that fewer than $\alpha \cdot (2l+1)$ vertices satisfy τ with the chosen parameters. □

We do not know whether the preceding proposition can be extended to $\alpha \geq \frac{1}{2}$. Its proof made crucial use of the assumption that $\alpha < \frac{1}{2}$. The non-hamiltonicity of the graph depended on the presence of an independent set containing more than half the vertices (the unfriendly ones), and no such set can exist when $\alpha > \frac{1}{2}$. More precisely, we have the following proposition, which prevents any construction like the preceding one from working when $\alpha > \frac{1}{2}$. (The specific randomizing construction in the preceding proof doesn't work for $\alpha = \frac{1}{2}$ either.)

Proposition 8.5 *Let $\alpha > \frac{1}{2}$. There exists a number k (depending on α) such that, for every sufficiently large n and every n-vertex graph satisfying SEA_k^α, no independent set contains more than half of the vertices.*

Proof Given α, choose k so large that

$$\left(1 - \frac{1}{2^k}\right)\alpha > \frac{1}{2},$$

and let n be much larger yet. Consider an n-vertex graph satisfying SEA_k^α, and suppose it had an independent set U of cardinality at least $n/2$. Fix k distinct elements $a_1, \ldots, a_k \in U$, and consider the axioms $\mathrm{SEA}(\tau)$ applied to these k parameters, where τ ranges over all k-parameter types *except* the one that says y is adjacent to none of the parameters. Thus, we are considering $2^k - 1$ types, and each can be satisfied only by elements outside U. Thus, these types altogether have at most $n/2$ elements satisfying them. But, by SEA_k^α, each of them is realized by at least $\alpha \cdot n \cdot 2^{-k}$, so altogether they are realized by at least $\alpha \cdot n \cdot 2^{-k} \cdot (2^k - 1)$ vertices. But this number exceeds $n/2$ by our choice of k. □

9 The Almost Sure Theory is Undecidable

In the case of first-order logic, the almost sure theory (that is the set of sentences with asymptotic probability 1) is decidable. The same holds if we

add the least fixed point operator to first-order logic [3]. But it fails for $\tilde{\mathrm{C}}$PTime.

Proposition 9.1 *The class of almost surely accepting polynomially bounded BGS programs and the class of almost surely rejecting polynomially bounded BGS programs are recursively inseparable.*

Proof Consider Turing machines with two halting states h_1 and h_2. For $i = 1, 2$, let H_i be the collection of Turing machines that eventually halt in state h_i if started on the empty input tape. It is well-known that H_1 and H_2 are recursively inseparable. Associate to each Turing machine T a polynomial time BGS program as follows. The program Π ignores its input graph and simulates T on empty input tape (working exclusively with pure sets). Π outputs `true` (resp. `false`) if T halts in state h_1 (resp. h_2). The polynomial bounds on steps and active elements are both twice the number of atoms. Then if $T \in H_1$ (resp. $T \in H_2$) our polynomial time BGS program will accept (resp. reject) all sufficiently large inputs. □

10 Existential Second-Order Logic

We remarked earlier that the strong extension axioms cannot be expressed in first-order logic. To prove this remark, it suffices to notice that the proof of Proposition 7.1 exhibited graphs satisfying EA_k for any prescribed k but not satisfying even SEA_1. Since the extension axioms provide a complete axiomatization of the first-order theory of random graphs, it follows that this theory does not contain SEA_1 (and therefore does not contain any SEA_k).

On the other hand, each SEA_k can be expressed in existential second-order logic. The following lemma implies this and a bit more.

Lemma 10.1 *Let $\varphi(x)$ and $\psi(x)$ be first-order formulas, and let α be a positive rational number. There is an existential second-order formula ν expressing "the number of x's satisfying $\varphi(x)$ is at least α times the number of x's satisfying $\psi(x)$.*

The formulas $\varphi(x)$ and $\psi(x)$ may have additional free variables; those variables will then be free in ν also.

Proof Write α as the quotient of two positive integers, p/q. We want to express that q times the number of solutions of $\varphi(x)$ is at least p times the

number of solutions of $\psi(x)$. The required ν says that there are p partial functions $f_1, \ldots, f_p$ such that

- each f_i has domain exactly the solutions of $\psi(x)$,
- each f_i has range included in the solutions of $\varphi(x)$, and
- no point has $q+1$ pre-images under these functions.

Here a pre-image of x is a y such that $f_i(y) = x$ for some i; if several i's, say m of them, work for the same x and y, then y counts as m pre-images of x. So formalizing the third requirement on the f_i's will involve a conjunction over all ways of expressing $q+1$ as a sum of p non-negative numbers. □

Remark 10.2 The lemma could be strengthened by allowing $\varphi(x)$ to be any existential second-order formula and $\psi(x)$ to be any universal second-order formula. To see this, modify the ν in the proof to say that the f_i are relations (rather than partial functions) with domains including (rather than exactly) the solutions of $\psi(x)$. Then $\varphi(x)$ is used only positively and $\psi(x)$ only negatively in ν, so ν is still existential second-order under the weakened assumptions on $\varphi(x)$ and $\psi(x)$.

Proposition 10.3 *For every k and every rational α, SEA^{α}_{k} is expressible as an existential second-order sentence.*

Proof Use the conjunction, over all k-parameter types τ, of existential second-order sentences expressing

$$(\forall x_1, \ldots, x_k)\left(\left(\bigwedge_{1 \le i < j \le k} x_i \neq x_j\right) \rightarrow \right.$$
$$\text{the number of solutions of } \tau(y, x_1, \ldots, x_k) \text{ is at least}$$
$$\left. \alpha \cdot \frac{1}{2^k} \text{ times the number of solutions of } y = y\right),$$

where an existential second-order form of the right side of the implication is provided by the lemma. □

One could hope that, just as the extension axioms imply the whole first-order theory of random graphs, so the strong extension axioms might imply the whole existential second-order theory of random graphs. Unfortunately, that is not the case.

Proposition 10.4 *There is an existential second-order sentence that has asymptotic probability 1 but is not a consequence of any SEA$_k^\alpha$.*

Proof In a graph with an odd number of vertices, call a vertex *balanced* if it is adjacent to exactly half of the other vertices. By Lemma 10.1, there is an existential second-order formula expressing "x is balanced." Use this to form an existential second-order sentence θ saying "either the number of vertices is even or there are at least two balanced vertices." We shall show that this sentence has asymptotic probability 1 but is not a consequence of any strong extension axiom.

Notice first that, in a random graph with $n = 2m + 1$ vertices, the probability that a specific vertex is balanced is

$$\binom{2m}{m} \cdot \frac{1}{2^{2m}} \sim \frac{1}{\sqrt{\pi m}} \sim \sqrt{\frac{2}{\pi}} \cdot \frac{1}{\sqrt{n}},$$

by Stirling's formula (see the proof of Lemma 4.10). Defining the random variable B as the number of balanced vertices, we infer that the expectation of B is asymptotically $\sqrt{2/\pi}\sqrt{n}$. So it is certainly reasonable that a large random graph should have at least 2 balanced vertices. To prove it, however, is not trivial, since the events "v is balanced" for different vertices v are not independent. Nevertheless, it is proved in [6, Theorem III.1] that, because the expected number of balanced vertices (there called $\lambda_{(n-1)/2}(n)$) tends to infinity with n, the asymptotic probability of "there are at least t balanced vertices" is 1 for every fixed t. In particular, this holds for $t = 2$, so θ has asymptotic probability 1.

To show that θ is not a consequence of any SEA$_k^\alpha$, we shall need to know that a random graph does not have too many balanced vertices. For this purpose, we use the well known estimate

$$\mathrm{Prob}\,[B \geq q] \leq \frac{E(B)}{q},$$

which is true for every non-negative random variable (see the beginning of the proof of Lemma 6.2), in particular for the number B of balanced vertices. So we have, for odd n,

$$\mathrm{Prob}\left[B \geq n^{2/3}\right] \leq \frac{Cn^{1/2}}{n^{2/3}} = Cn^{-1/6},$$

for a suitable constant C. So this probability tends to 0 as $n \to \infty$. Thus, a large random graph almost certainly has fewer than $n^{2/3}$ balanced vertices.

Using this, we shall produce, for any prescribed strong extension axiom SEA_k^α, a graph satisfying this axiom yet making θ false. Fix a number β with $\alpha < \beta < 1$ (recall that the definition of SEA_k^α requires $\alpha < 1$). Let G be a graph that satisfies SEA_k^β, that has fewer than $n^{2/3}$ balanced vertices, and that satisfies $n^{2/3} < (\beta - \alpha)n \cdot 2^{-k}$, where n is the number of vertices of G. By the preceding paragraph and by Proposition 6.1 (and the remark immediately following its proof), such a G is easy to find; any sufficiently large random graph will work with high probability. We shall complete the proof by showing how to modify G so as to violate θ while still satisfying SEA_k^α.

If the number of balanced vertices in G is even, then group them arbitrarily into pairs; if the number is odd, then pair off all but one of them. For each of these pairs, if the two (balanced) vertices were adjacent in G then make them non-adjacent, but if they were non-adjacent then make them adjacent. This changes by 1 the number of neighbors of each of these vertices, so they are no longer balanced. Thus, the resulting graph G' has at most one balanced vertex, i.e., it violates θ.

The modifications leading from G to G' affected the adjacency relationships of fewer than $n^{2/3}$ vertices. Thus, for any k-parameter type and any choice of the k parameters, since there were at least $\beta n 2^{-k}$ vertices satisfying the type in G, there remain at least $\beta n 2^{-k} - n^{2/3}$ vertices satisfying it in G'. And we chose n large enough to ensure that this is more than $\alpha n 2^{-k}$. So G' satisfies SEA_k^α. □

11 Set-Theoretic Definitions

We now begin working toward the proof of Theorem 2.2. The first part of the proof is to express, in set-theoretic terms, the behavior of BGS programs. By explicitly exhibiting the relevant set-theoretic formulas, we also formalize the interpretation of terms and rules, described informally in Section 4.

Consider any particular state of one of our ASM's. It is a structure H^+ with underlying set $HF(I)$ and with interpretations for all the function symbols listed in Definition 4.1. Let H be the structure

$$H = \langle HF(I), \in, I, A \rangle$$

that is like H^+ except that among the non-logical symbols only $\in$, Atoms, and A are interpreted. (In the usual terminology of mathematical logic, H is a reduct of H^+.) Let H^D be the intermediate structure in which the dynamic function symbols are also interpreted (by the same functions as in H^+). We shall need to know that all essential aspects of the execution of Π in the state H^+, i.e., the computation leading from H^+ to its sequel state, can be defined in the structure H^D. (It will turn out that, for every state H^+ that actually arises during the computation of a BGS machine, the dynamic functions will be definable in H, and so we could use H instead of H^D here. See the proof of Proposition 11.1 below.)

To avoid having to introduce new symbols for a multitude of set-theoretic formulas, we adopt the notational convention that $\lceil\varphi\rceil$ means the set-theoretic formalization of the (informal) statement φ.

We begin by defining $\lceil y \in t\rceil$ and $\lceil y = t\rceil$ for all terms t; here y is a variable not free in t. In most cases one of $\lceil y \in t\rceil$ and $\lceil y = t\rceil$ is a bit easier to define than the other, and once one is defined it is usually easy to obtain the other. Specifically, if we have $\lceil y = t\rceil$ then we can define $\lceil y \in t\rceil$ as $\exists z\,(\lceil z = t\rceil \wedge y \in z)$. Conversely, if we have $\lceil y \in t\rceil$ and we know that the value of t is not an atom, then we can define $\lceil y = t\rceil$ as

$$y \notin \text{Atoms} \wedge \forall z(z \in y \leftrightarrow \lceil z \in t\rceil).$$

In the definitions that follow, whenever only one of $\lceil y \in t\rceil$ and $\lceil y = t\rceil$ is specified, the other is to be obtained by these standard definitions. We proceed by recursion on t.

- If v is a variable then $\lceil y \in v\rceil$ is $y \in v$ and $\lceil y = v\rceil$ is $y = v$.
- If f is a dynamic function symbol, then $\lceil y = f(t_1, \ldots, t_j)\rceil$ is

$$\exists z_1 \ldots \exists z_j \left(\bigwedge_{i=1}^{j} \lceil z_i = t_i\rceil \wedge y = f(z_1, \ldots, z_j) \right).$$

- $\lceil y \in \texttt{false}\rceil$ is $\texttt{false}$.
- $\lceil y \in \texttt{true}\rceil$ is $\lceil y = \texttt{false}\rceil$.
- $\lceil y \in (t_1 = t_2)\rceil$ is

$$\lceil y = \texttt{false}\rceil \wedge \exists z\,(\lceil z = t_1\rceil \wedge \lceil z = t_2\rceil).$$

- $\lceil y = \neg t \rceil$ is $(\lceil \mathtt{false} = t \rceil \wedge \lceil y = \mathtt{true} \rceil) \vee (\neg \lceil \mathtt{false} = t \rceil \wedge \lceil y = \mathtt{false} \rceil)$, and similarly for the other connectives. (Note that the definition is arranged so as to give the value `false` when the argument is not a truth value.)

- $\lceil y \in (t_1 \in t_2) \rceil$ is

$$\lceil y = \mathtt{false} \rceil \wedge \exists z_1 \exists z_2 \, (\lceil z_1 = t_1 \rceil \wedge \lceil z_2 = t_2 \rceil \wedge z_1 \in z_2).$$

- $\lceil y = \emptyset \rceil$ is $\lceil y = \mathtt{false} \rceil$.

- $\lceil y = \mathrm{Atoms} \rceil$ is $y = \mathrm{Atoms}$

- $\lceil y \in \bigcup t \rceil$ is $\exists z \, (y \in z \wedge \lceil z \in t \rceil)$.

- $\lceil y = \mathrm{TheUnique}(t) \rceil$ is

$$[\forall z \, (\lceil z \in t \rceil \leftrightarrow z = y)] \vee$$
$$[\lceil y = \emptyset \rceil \wedge \neg \exists v \forall z \, (\lceil z \in t \rceil \leftrightarrow z = v)]$$

- $\lceil y \in \mathrm{Pair}(t_1, t_2) \rceil$ is $\lceil y = t_1 \rceil \vee \lceil y = t_2 \rceil$.

- $\lceil y \in A(t_1, t_2) \rceil$ is

$$\lceil y = \mathtt{false} \rceil \wedge \exists z_1 \exists z_2 \, (\lceil z_1 = t_1 \rceil \wedge \lceil z_2 = t_2 \rceil \wedge A(z_1, z_2))$$

- $\lceil y \in \{t(v) : v \in r : \varphi(v)\} \rceil$ is

$$\exists v \, (\lceil v \in r \rceil \wedge \lceil \mathtt{true} = \varphi(v) \rceil \wedge \lceil y = t(v) \rceil).$$

In the last clause here, y should be different from v; otherwise rename v.

Next, we define in H^D the semantics of rules. For each rule R and each dynamic function symbol f, say of arity j, we first define a preliminary formula, $\lceil R$ wants to set f at $x_1, \ldots, x_j$ to $y \rceil$, which ignores possible conflicts between updates. This is a formula with free variables $x_1, \ldots, x_j, y$ and any variables $z_1, \ldots, z_k$ free in the rule R. It holds in state H^D of elements $a_1, \ldots, a_j, b, c_1 \ldots, c_k$ if and only if $((f, a_1, \ldots, a_j), b)$ is in the update set of rule $R(c_1, \ldots, c_k)$ in state H^+, as defined in [11]. (We use the symbol f in our name for the formula $\lceil R$ wants to set f at $x_1, \ldots, x_j$ to $y \rceil$, but f need not occur in the formula itself.)

- $\lceil$`Skip` wants to set f at $x_1, \ldots, x_j$ to $y\rceil$ is `false`.
- $\lceil f(t_1, \ldots, t_j) := t_0$ wants to set f at $x_1, \ldots, x_j$ to $y\rceil$ is

$$\bigwedge_{i=1}^{j} \lceil x_i = t_i \rceil \wedge \lceil y = t_0 \rceil.$$

- $\lceil g(t_1, \ldots, t_k) := t_0$ wants to set f at $x_1, \ldots, x_j$ to $y\rceil$ is `false` when g is distinct from f.
- $\lceil$`if` φ `then` R_0 `else` R_1 `endif` wants to set f at $x_1, \ldots, x_j$ to $y\rceil$ is

$$\begin{aligned}&(\lceil \texttt{true} = \varphi \rceil \wedge \lceil R_0 \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y \rceil) \vee \\ &(\lceil \texttt{false} = \varphi \rceil \wedge \lceil R_1 \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y \rceil).\end{aligned}$$

- $\lceil$`do forall` $v \in r,\ R$ `enddo` wants to set f at $x_1, \ldots, x_j$ to $y\rceil$ is

$$\exists v \, (\lceil v \in r \rceil \wedge \lceil R \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y \rceil).$$

A rule may want to set f at $x_1, \ldots, x_j$ to several values. We adopt the standard ASM convention that if the program Π contains such a conflict, then all the dynamic functions remain unchanged. The formal definitions are as follows.

- $\lceil \Pi$ clashes$\rceil$ is

$$\begin{aligned}\bigvee_f \exists x_1 \quad \ldots \quad \exists x_j \exists y \exists z \\ &(\Pi \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y) \wedge \\ &(\Pi \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } z) \wedge y \neq z.\end{aligned}$$

  Of course, the arity j depends on f.
- $\lceil \Pi$ sets f at $x_1, \ldots, x_j$ to $y\rceil$ is

$$\lceil \Pi \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y \rceil \wedge \neg \lceil \Pi \text{ clashes} \rceil.$$

Finally, we define the dynamic functions for the sequel state, that is, for the state obtained from H^+ by executing Π once.

For a j-ary dynamic function f, we define

$$\lceil y = f(x_1, \ldots, x_j) \text{ in the sequel} \rceil$$

to be

$$\lceil \Pi \text{ sets } f \text{ at } x_1, \ldots, x_j \text{ to } y \rceil \vee$$
$$(\lceil y = f(x_1, \ldots, x_j) \rceil \wedge \neg \exists y' \lceil \Pi \text{ sets } f \text{ at } x_1, \ldots, x_j \text{ to } y' \rceil.$$

The preceding definitions provide most of the proof of the following result.

Proposition 11.1 *For each BGS program Π, for each natural number m, and for each dynamic function symbol f, there is a first-order formula $\lceil y = f(x_1, \ldots, x_j)$ at step $m \rceil$ in the vocabulary $\{\in, Atoms, A\}$ such that, for any input structure $\langle I, A \rangle$, the tuples that satisfy $\lceil y = f(x_1, \ldots, x_j)$ at step $m \rceil$ in $H = \langle HF(I), \in, I, A \rangle$ constitute the graph of f in the m^{th} state of the run of Π on $\langle I, A \rangle$. Furthermore, there exists a number B such that, for each m, the number of variables occurring in $\lceil y = f(x_1, \ldots, x_j)$ at step $m \rceil$ is at most B.*

It will be important that the bound B depends only on Π, not on m. To avoid possible confusion, we emphasize that variables can be re-used in these formulas; thus the same variable may be bound many times and may occur free as well.

Proof We construct the required formulas by induction on m, starting with $\lceil y = f(x_1 \ldots, x_j)$ at step $0 \rceil$, which can be taken to be $\lceil y = \emptyset \rceil$ because all dynamic functions are initialized to be constant with value $\emptyset$.

For the induction step from m to $m + 1$, we begin with the formula $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$ as constructed above. Then, for each dynamic function symbol g, we replace each occurrence of a subformula $\lceil t_0 = g(t_1, \ldots, t_k) \rceil$ with $\lceil t_0 = g(t_1, \ldots, t_k)$ at step $m \rceil$.

As for the bound B, the argument will be slightly easier if we do not allow the same variable to be both free and bound, though we do allow variables to be bound many times. Any bound B that works under this convention clearly continues to work under the standard convention that allows free variables to also occur bound.

We shall need the fact that, in the formulas $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$, whenever a dynamic function symbol g occurs in an atomic subformula, that subformula has the form $t_0 = g(t_1, \ldots, t_k)$ where all the terms t_i are variables or constants. To prove this fact, just go through all the clauses in the definitions leading up to $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$ and notice that all atomic formulas that ever get introduced have the required form. In particular, such an atomic subformula never has more than $k + 1$ free variables, where k is the arity of g.

Let B be the maximum number of variables in any of the formulas $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$ as f ranges over all the dynamic function symbols. This exceeds the number of variables in $\lceil y = \emptyset \rceil$, so the bound is obeyed when $m = 0$.

It continues to be obeyed for larger m. To see this, consider any one of the substitutions involved in converting $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$ into $\lceil y = f(x_1 \ldots, x_j)$ at step $m+1\rceil$, say a replacement of $t_0 = g(t_1, \ldots, t_k)$ with $\lceil t_0 = g(t_1, \ldots, t_k)$ at step $m\rceil$. The subformula being replaced is an atomic formula with at most $k+1$ free variables, by virtue of our observations above; the formula replacing it has, by induction hypothesis, at most $B - (k+1)$ bound variables (because $\lceil y = g(x_1, \ldots, x_k)$ at step $m\rceil$ has $k+1$ free variables and at most B variables altogether). So by using recycled variables as the bound ones in $\lceil t_0 = g(t_1, \ldots, t_k)$ at step $m\rceil$, we can avoid exceeding the overall bound of B variables. □

Remark 11.2 For the purpose of proving Theorem 2.2, it is important that the number of variables in the formulas $\lceil y = f(x_1, \ldots, x_j)$ at step $m\rceil$ be bounded independently of m, but it is not crucial that the formulas be finite. Formulas of the infinitary language $L_{\infty,\omega}$ would serve as well. An alternate approach to obtaining such formulas is to express $\lceil y = f(x_1, \ldots, x_j)$ at step $z\rceil$ (with a variable z for the number of steps, numbers being viewed as von Neumann ordinals) in first-order logic with the least-fixed-point operator, and then to use the known translation from this logic into the infinitary, finite-variable logic $L^{\omega}_{\infty,\omega}$ (see [8]). This is the approach used in [4]. Then, to get the corresponding formulas for specific m in place of z, one only has to check that each natural number m can be defined with a fixed number of variables, independent of m. In fact, each natural number can be defined with just three variables.

12 Supports

In this section, we describe the notion of symmetry with respect to partial automorphisms that will, as explained in Section 5, apply to all objects involved in a computation and lead to winning strategies for the duplicator in certain Ehrenfeucht-Fraïssé games. Two numerical parameters will be involved in this notion of symmetry, namely the size of the partial automorphisms and the number of atoms a symmetric object can depend on. The appropriate values for these parameters will depend on the BGS program and the polynomial bound on the number of active elements.

For the purposes of this section, let $q \geq 1$ and $k \geq 4$ be fixed integers. When this material is applied in the proof of Theorem 2.2, q will be the degree of a polynomial bounding the number of involved elements (obtainable from Π and the bound on active elements, via Proposition 14.10) and k will be $B+2V+4$, where B is as in Proposition 11.1 and V is the number of variables in Π.

We assume that the input graph $\langle I, A\rangle$ satisfies the extension axioms EA_{3kq} for up to $3kq$ parameters. (We don't need the strong extension axioms yet.)

For brevity we adopt the conventions that in this section

- w, x, y, z (possibly with subscripts, superscripts, or accents) stand for members of $HF(I)$,
- a, b, c (possibly with subscripts, superscripts, or accents) stand for sets of $\leq q$ atoms, and we call such sets *possible supports*, and
- $\alpha, \beta, \gamma, \delta, \zeta$ (possibly with subscripts or superscripts) stand for partial automorphisms of the graph $\langle I, A\rangle$ whose domains have size $\leq kq$, and we call such maps *motions*.

The inverse α^{-1} of a motion and the composite $\alpha\circ\beta$ of two motions are defined in the obvious way. In particular, the domain of $\alpha \circ \beta$ is $\beta^{-1}(\mathrm{Dom}(\alpha))$.

The extension axioms imply that, if α is a motion and s is a set of atoms with $|\mathrm{Dom}(\alpha)| + |s| \leq kq$, then α can be extended to a motion whose domain includes s. (In fact, the extension axioms imply considerably more, as they go up to $3kq$ parameters, not just the $kq-1$ needed for the preceding statement.)

We next define, by simultaneous recursion on the rank of x, the three concepts "a supports x," "x is supported," and "$\hat{\alpha}(x)$," where the last of these is defined if and only if $\mathrm{Dom}(\alpha)$ includes some a that supports x.

Definition 12.1 If x is an atom, then

- a supports x if and only if $x \in a$,
- x is supported (always), and
- $\hat{\alpha}(x) = \alpha(x)$.

If, on the other hand, x is a set, then

- a supports x if and only if every $y \in x$ is supported and, for every y of lower rank than x and every motion α, if α pointwise fixes a and if $\mathrm{Dom}(\alpha)$ includes some set supporting y, then

$$y \in x \iff \hat{\alpha}(y) \in x,$$

- x is supported if and only if some a supports x, and
- if a supports x and $a \subseteq \mathrm{Dom}(\alpha)$, then $\hat{\alpha}(x)$ is the set of all $\hat{\beta}(y)$ where $y \in x$, $\beta \restriction a = \alpha \restriction a$, and $\mathrm{Dom}(\beta)$ includes some support of y.

The definition of $\hat{\alpha}(x)$ when x is a set seems to depend on the choice of a particular support a of x. The first part of the following lemma gets rid of that apparent dependence; the rest of the lemma gives useful technical information about supports and about the application of motions to supported sets.

Lemma 12.2 *1. $\hat{\alpha}$ is well-defined. Specifically, if a_1 and a_2 both support x and are both included in $Dom(\alpha)$, then $\hat{\alpha}^1(x)$ and $\hat{\alpha}^2(x)$, defined as above using a_1 and a_2 respectively, are equal.*

2. If a supports x and $a \subseteq Dom(\alpha) \cap Dom(\beta)$ and $\alpha \restriction a = \beta \restriction a$, then $\hat{\alpha}(x) = \hat{\beta}(x)$.

3. If $\hat{\alpha}(x)$ is defined then it has the same rank as x.

4. If α is an identity map, then so is $\hat{\alpha}$, i.e., $\hat{\alpha}(x) = x$ whenever $\hat{\alpha}(x)$ is defined.

5. *$\hat{\alpha}$ is a partial automorphism of the structure $\langle HF(I), \in, I, A\rangle$. In other words, $\hat{\alpha}(I) = I$ and, whenever x and x' have supports included in $Dom(\alpha)$,*

$$\begin{aligned} x' = x &\iff \hat{\alpha}(x') = \hat{\alpha}(x) \\ x' \in x &\iff \hat{\alpha}(x') \in \hat{\alpha}(x) \\ x' A x &\iff \hat{\alpha}(x') A \hat{\alpha}(x). \end{aligned}$$

6. *If a supports x and $a \subseteq Dom(\alpha)$ then $\alpha[a]$ supports $\hat{\alpha}(x)$.*

7. *If $\hat{\alpha}(x)$ is defined and $a \subseteq Dom(\alpha)$ and $\alpha[a]$ supports $\hat{\alpha}(x)$, then a supports x.*

8. *$\widehat{\beta \circ \alpha} = \hat{\beta} \circ \hat{\alpha}$ in the following sense: If $\widehat{\beta \circ \alpha}(x)$ is defined then so is $\hat{\beta}(\hat{\alpha}(x))$ and they are equal.*

Proof We prove all parts of the lemma by simultaneous induction on the maximum of the ranks of x and x'. Observe first that $\hat{\alpha}$ always sends atoms to atoms and sets to sets. Given this observation, the lemma is trivial whenever x is an atom. So we assume from now on that x is a set.

Proof of (1) Let $x, a_1, a_2, \alpha, \hat{\alpha}^1$, and $\hat{\alpha}^2$ be as in (1). It suffices to show that every $z \in \hat{\alpha}^1(x)$ is also in $\hat{\alpha}^2(x)$. Such a z has the form $\hat{\beta}_1(y_1)$ for some $y_1 \in x$ and some β_1 that extends $\alpha \restriction a_1$ and has some support b_1 of y_1 included in its domain. (By induction hypothesis, $\hat{\beta}_1(y_1)$ is well-defined.)

Let β_2 be any motion extending $\alpha \restriction (a_1 \cup a_2)$ with $\mathrm{Range}(\beta_2) \supseteq \beta_1[b_1]$. Such a β_2 exists by the extension axioms. Set $\gamma = {\beta_2}^{-1} \circ \beta_1$.

Since both β_1 and β_2 agree with α on a_1, we know that γ fixes a_1 pointwise. Also, since $\beta_1[b_1] \subseteq \mathrm{Range}(\beta_2)$, we have that $\mathrm{Dom}(\gamma)$ includes a support b_1 of y. Thus, by definition of "a_1 supports x," we have

$$y_1 \in x \iff \hat{\gamma}(y_1) \in x.$$

But $y_1 \in x$, and therefore $\hat{\gamma}(y_1) \in x$. Set $y_2 = \hat{\gamma}(y_1)$ and $b_2 = \gamma[b_1]$. Thus, $y_2 \in x$ and b_2 supports y_2 by induction hypothesis (6). Furthermore, β_2 was chosen to extend $\alpha \restriction a_2$ and its domain includes b_2 (because, by definition of γ, $\mathrm{Range}(\gamma) \subseteq \mathrm{Dom}(\beta_2)$). Therefore, by definition of $\hat{\alpha}^2(x)$, we have $\hat{\beta}_2(y_2) \in \hat{\alpha}^2(x)$. But

$$\begin{aligned} \hat{\beta}_2(y_2) &= \hat{\beta}_2(\hat{\gamma}(y_1)) = \widehat{\beta_2 \circ \gamma}(y_1) \\ &= \hat{\beta}_1(y_1) = z, \end{aligned}$$

where the first line uses induction hypothesis (8) and the second uses induction hypothesis (2) as $\beta_2 \circ \gamma$ agrees with β_1 on b_1 which supports y_1. To justify the use of induction hypothesis (8) here, we need to know that $\widehat{\beta_2 \circ \gamma}(y_1)$ is defined; it is, because y_1 has a support b_1 included in the domain of $\beta_2 \circ \gamma$. In the future, such justifications for (8) will be given in the abbreviated form "use support b_1 for y_1." The displayed equations above and the comment immediately preceding them give $z \in \hat{\alpha}^2(x)$, as required. □

Having proved (1), we can choose any support of x that is included in $\mathrm{Dom}(\alpha)$ to serve as the a in the definition of $\hat{\alpha}(x)$. We shall make use of this freedom without mentioning it explicitly.

Proof of (2) The definition of $\hat{\alpha}(x)$ refers to α only in the context $\alpha \restriction a$. □

Proof of (3) All elements of $\hat{\alpha}(x)$ have the form $\hat{\beta}(y)$ with $y \in x$. Here y has lower rank than x and, by induction hypothesis, so does $\hat{\beta}(y)$. Therefore, $\mathrm{rank}(\hat{\alpha}(x)) \leq \mathrm{rank}(x)$.

To complete the proof, we assume strict inequality, $\mathrm{rank}(\hat{\alpha}(x)) < \mathrm{rank}(x)$, and deduce a contradiction. The strict inequality implies that x has an element y of rank $\geq \mathrm{rank}(\hat{\alpha}(x))$. By the extension axioms, there is a motion β that agrees with α on a and also has a support of y included in its domain. Then $\hat{\beta}(y) \in \hat{\alpha}(x)$. This is absurd since, by induction hypothesis, $\mathrm{rank}(\hat{\beta}(y)) = \mathrm{rank}(y) \geq \mathrm{rank}(\hat{\alpha}(x))$. □

Proof of (4) Suppose α is an identity map and $\hat{\alpha}(x)$ is defined, so some support a of x is included in $\mathrm{Dom}(\alpha)$. Given any $y \in x$, say with support b, let β be the identity map on $a \cup b$. By definition of $\hat{\alpha}(x)$, we have $\hat{\beta}(y) \in \hat{\alpha}(x)$. By induction hypothesis, we have $\hat{\beta}(y) = y$. Thus, all elements y of x are also in $\hat{\alpha}(x)$.

For the converse, consider any element of $\hat{\alpha}(x)$, say $\hat{\beta}(y)$ where $y \in x$ and β agrees with the identity map on a. Then, by definition of "a supports x," we have $\hat{\beta}(y) \in x$, as required. □

Proof of (5) Because $\hat{\alpha}$ sends atoms to atoms and sets to sets, because it agrees with the partial automorphism α on atoms, and because the adjacency relation A holds only between atoms, the third equivalence in (5) is trivial. It also follows easily that I is supported by $\emptyset$ and $\hat{\alpha}(I) = I$ for all motions α.

For the rest of the proof, we fix supports a and a' for x and x', respectively, both included in $\mathrm{Dom}(\alpha)$.

We begin by proving the second equivalence,

$$x' \in x \iff \hat{\alpha}(x') \in \hat{\alpha}(x).$$

We may assume that x' has lower rank than x, for otherwise the left side of the equivalence is clearly false and, by (3), so is the right side. If $x' \in x$ then, in the definition of $\hat{\alpha}(x)$, we can take $y = x'$ and $\beta = \alpha$ to conclude $\hat{\alpha}(x') \in \hat{\alpha}(x)$. So we have one direction of the desired equivalence.

For the converse, suppose $\hat{\alpha}(x') \in \hat{\alpha}(x)$. So there exist $y \in x$, a support b of y, and a motion β such that

- $\beta \restriction a = \alpha \restriction a$,
- $b \subseteq \mathrm{Dom}(\beta)$, and
- $\hat{\beta}(y) = \hat{\alpha}(x')$.

Let γ be an extension of $(\alpha \restriction (a \cup a'))^{-1}$ whose domain includes $\beta[b]$. From $\hat{\beta}(y) = \hat{\alpha}(x')$ we infer, using the induction hypothesis, that $\widehat{\gamma \circ \beta}(y) = \widehat{\gamma \circ \alpha}(x') = x'$. (In more detail: $\mathrm{Dom}(\gamma)$ includes $\beta[b]$ and $\alpha[a']$ which, by induction hypothesis (6), support $\hat{\beta}(y)$ and $\hat{\alpha}(x')$. So we can say $\hat{\gamma}(\hat{\beta}(y)) = \hat{\gamma}(\hat{\alpha}(x'))$. Apply induction hypothesis (8) to both sides of this, using supports b for y and a' for x'. Finally, observe that $\gamma \circ \alpha$ agrees with the identity map on a' which supports x'. So by induction hypotheses (2) and (4), $\widehat{\gamma \circ \alpha}(x') = x'$.) Because β agrees with α on a and γ extends $(\alpha \restriction a)^{-1}$, $\gamma \circ \beta$ pointwise fixes a. Referring to the definition of "a supports x" and using that $\mathrm{Dom}(\gamma \circ \beta)$ includes b which supports y, we find that

$$y \in x \iff \widehat{\gamma \circ \beta}(y) \in x.$$

But $y \in x$ and $\widehat{\gamma \circ \beta}(y) = x'$. So $x' \in x$ as required.

It remains to prove the first equivalence in (5), and here the left-to-right implication is trivial (since we have proved (1)). So assume that $\hat{\alpha}(x') = \hat{\alpha}(x)$. Observe that, thanks to (3), x and x' have the same rank. We must show that $x = x'$; it suffices to consider an arbitrary element y of x and prove that $y \in x'$. Extend $\alpha \restriction (a \cup a')$ to a β whose domain includes a support of y. Then, by definition of $\hat{\alpha}(x)$, we have $\hat{\beta}(y) \in \hat{\alpha}(x)$. Also, by (2), $\hat{\alpha}(x') = \hat{\beta}(x')$. Thus, we have

$$\hat{\beta}(y) \in \hat{\alpha}(x) = \hat{\alpha}(x') = \hat{\beta}(x').$$

By the second equivalence in (5), already proved above, we infer $y \in x'$, as required. $\square$

Proof of (6) Let x, a, and α be as in (6). To show that $\alpha[a]$ supports $\hat{\alpha}(x)$, we must first check that all elements of $\hat{\alpha}(x)$ are supported. But these elements have the form $\hat{\beta}(y)$ with $y \in x$. Here y is supported (as a supports x) and therefore so is $\hat{\beta}(y)$, by induction hypothesis.

Now we verify the main clause in the definition of "$\alpha[a]$ supports $\hat{\alpha}(x)$." Let y have lower rank than $\hat{\alpha}(x)$, i.e., lower rank than x since we've proved (3). Let β be a motion pointwise fixing $\alpha[a]$ and with domain including some support b of y. We must prove

$$y \in \hat{\alpha}(x) \iff \hat{\beta}(y) \in \hat{\alpha}(x).$$

Replace α by some extension of $\alpha \restriction a$ whose range includes $b \cup \beta[b]$. This affects neither $\alpha[a]$ nor (thanks to (2)) $\hat{\alpha}(x)$, so there is no loss of generality in assuming $b \cup \beta[b] \subseteq \mathrm{Range}(\alpha)$.

Now $\widehat{\alpha^{-1}}(y)$ is defined and is, by induction hypothesis (6), supported by $\alpha^{-1}[b]$. Furthermore, $\alpha^{-1}[b] \subseteq \mathrm{Dom}(\alpha^{-1} \circ \beta \circ \alpha)$. (Indeed, a point in $\alpha^{-1}[b]$ would be mapped first into b by α; it then gets mapped into $\beta[b]$; and we arranged for this to be in the domain of α^{-1}.) Since $\alpha^{-1} \circ \beta \circ \alpha$ pointwise fixes a support a of x (because β pointwise fixes $\alpha[a]$), and $\widehat{\alpha^{-1}}(y)$ has lower rank than x (by (3)) and has a support included in $\mathrm{Dom}(\alpha^{-1} \circ \beta \circ \alpha)$ (namely $\alpha^{-1}[b]$), we conclude, by definition of "a supports x," that

$$\widehat{\alpha^{-1}}(y) \in x \iff (\widehat{\alpha^{-1} \circ \beta \circ \alpha})(\widehat{\alpha^{-1}}(y)) \in x.$$

Because $\alpha^{-1} \circ \beta \circ \alpha \circ \alpha^{-1}$ and $\alpha^{-1} \circ \beta$ agree on a support of y, namely b, we obtain by induction hypotheses (8) (using support b for y) and (2) that $(\widehat{\alpha^{-1} \circ \beta \circ \alpha})(\widehat{\alpha^{-1}}(y)) = \widehat{\alpha^{-1}}(\hat{\beta}(y))$. Therefore,

$$\widehat{\alpha^{-1}}(y) \in x \iff \widehat{\alpha^{-1}}(\hat{\beta}(y)) \in x.$$

By (5), we can apply $\hat{\alpha}$ to obtain

$$\hat{\alpha}(\widehat{\alpha^{-1}}(y)) \in \hat{\alpha}(x) \iff \hat{\alpha}(\widehat{\alpha^{-1}}(\hat{\beta}(y))) \in \hat{\alpha}(x).$$

Finally, induction hypotheses (8) (again using support b for y) and (4) allow us to simplify this to

$$y \in \hat{\alpha}(x) \iff \hat{\beta}(y) \in \hat{\alpha}(x),$$

as required. □

Proof of (7) As $\hat{\alpha}(x)$ is defined, x has a support included in $\mathrm{Dom}(\alpha)$; in particular, all elements of x are supported. So we need only check the main clause in the definition of "a supports x."

Assume y has lower rank than x, β pointwise fixes a, and $\mathrm{Dom}(\beta)$ includes a support b of y. We must show that $y \in x$ if and only if $\hat{\beta}(y) \in x$.

Replace α by its restriction to the union of a and some support of x, and then extend the resulting motion to have both b and $\beta[b]$ included in its domain. None of this affects $\alpha[a]$ or $\hat{\alpha}(x)$. So we may assume without loss of generality that $b \cup \beta[b] \subseteq \mathrm{Dom}(\alpha)$. (Here we use that $k \geq 4$, since we need a motion with four supports in its domain.)

Now $\hat{\alpha}(y)$ is defined and is supported by $\alpha[b]$, which is included in $\mathrm{Dom}(\alpha \circ \beta \circ \alpha^{-1})$. Also, $\alpha \circ \beta \circ \alpha^{-1}$ pointwise fixes $\alpha[a]$ (because β pointwise fixes a), and $\hat{\alpha}(y)$ has lower rank than $\hat{\alpha}(x)$ by (3). So, from the definition of "$\alpha[a]$ supports $\hat{\alpha}(x)$," we conclude that

$$\hat{\alpha}(y) \in \hat{\alpha}(x) \iff (\widehat{\alpha \circ \beta \circ \alpha}^{-1})(\hat{\alpha}(y)) \in \hat{\alpha}(x).$$

By induction hypothesis (including (8) justified by using support b for y), $(\widehat{\alpha \circ \beta \circ \alpha}^{-1})(\hat{\alpha}(y)) = \hat{\alpha}(\hat{\beta}(y))$. So we have

$$\hat{\alpha}(y) \in \hat{\alpha}(x) \iff \hat{\alpha}(\hat{\beta}(y)) \in \hat{\alpha}(x).$$

Finally, by (5),

$$y \in x \iff \hat{\beta}(y) \in x,$$

as required. □

Proof of (8) Assume that $\widehat{\beta \circ \alpha}(x)$ is defined. So x has a support $a \subseteq \mathrm{Dom}(\beta \circ \alpha)$. Then $a \subseteq \mathrm{Dom}(\alpha)$, so $\hat{\alpha}(x)$ is defined. Furthermore, by (6), $\hat{\alpha}(x)$ is supported by $\alpha[a] \subseteq \mathrm{Dom}(\beta)$, so $\hat{\beta}(\hat{\alpha}(x))$ is defined.

To show that $\widehat{\beta \circ \alpha}(x) = \hat{\beta}(\hat{\alpha}(x))$, consider first an arbitrary $z \in \hat{\beta}(\hat{\alpha}(x))$. Then, by definition of the "hat" operation on motions, we have some $w \in \hat{\alpha}(x)$ and some motion δ agreeing with β on $\alpha[a]$ such that $z = \hat{\delta}(w)$ (so in particular some support of w is included in $\mathrm{Dom}(\delta)$). Furthermore, we have some $y \in x$ and some motion γ agreeing with α on a such that $w = \hat{\gamma}(y)$ and, in particular, some support b of y is included in $\mathrm{Dom}(\gamma)$. Extending δ if necessary, we can arrange that its domain includes $\gamma[b]$. Then we can apply induction hypothesis (8) using the support b of y to obtain $z = \hat{\delta}(\hat{\gamma}(y)) =$

$\widehat{\delta \circ \gamma}(y)$. But $\delta \circ \gamma$ agrees with $\beta \circ \alpha$ on the support a of x. Therefore, $z = \widehat{\delta \circ \gamma}(y) \in \widehat{\beta \circ \alpha}(x)$. This proves the inclusion from right to left in the desired equation.

For the converse inclusion, suppose $z \in \widehat{\beta \circ \alpha}(x)$. So $z = \hat{\zeta}(y)$ for some $y \in x$ and some motion ζ that agrees with $\beta \circ \alpha$ on a and has a support b of y included in its domain.

Extend $\alpha \restriction a$ to a motion γ whose domain includes b. Then set $\delta = \zeta \circ \gamma^{-1}$. So $\mathrm{Dom}(\delta)$ includes both $\alpha[a] = \gamma[a]$ and $\gamma[b]$. Also $\delta \circ \gamma$ agrees with ζ on $a \cup b$ (as γ is defined there). By induction hypotheses (2) and (8) (using support b for y), $z = \widehat{\delta \circ \gamma}(y) = \hat{\delta}(\hat{\gamma}(y))$. Now as γ extends α and $y \in x$, we have $\hat{\gamma}(y) \in \hat{\alpha}(x)$. And then, as δ extends β, we have $\hat{\delta}(\hat{\gamma}(y)) \in \hat{\beta}(\hat{\alpha}(x))$. That is, $z \in \hat{\beta}(\hat{\alpha}(x))$ as required. □

This completes the proof of Lemma 12.2. □

We record for future reference a fairly easy consequence of the lemma.

Corollary 12.3 *If a supports x then a also supports $\bigcup x$.*

Proof We first observe that every member of $\bigcup x$ is a member of a member of x and is therefore supported since x is. Now to verify the main clause in the definition of "a supports x" let y be any object of lower rank than x and let α be a motion that is the identity on a and has some support b of y included in its domain. We must show that

$$y \in \bigcup x \iff \hat{\alpha}(y) \in \bigcup x.$$

Suppose first that $y \in \bigcup x$, so $y \in z \in x$ for some z. As a member of a supported set x, z is also supported. Without loss of generality, restrict α to $a \cup b$ and then extend it so that its domain includes some support of z. Then by part 5 of the lemma, $\hat{\alpha}(y) \in \hat{\alpha}(z) \in \hat{\alpha}(x)$. But $\hat{\alpha}(x) = x$ by parts 2 and 4 of the lemma. So $\hat{\alpha}(y) \in \bigcup x$.

Conversely, suppose $\hat{\alpha}(y) \in \bigcup x$, say $\hat{\alpha}(y) \in z \in x$. Again, z is supported. Without loss of generality, restrict α to $a \cup b$ and then extend it so that its range includes a support of z. Thus, $\hat{\alpha}^{-1}(z)$ is defined. Furthermore, by parts 2, 4, and 8 of the lemma, we have $\hat{\alpha}^{-1}(x) = x$ and $\hat{\alpha}^{-1}(\hat{\alpha}(y)) = y$. Therefore, by part 5 of the lemma applied to α^{-1}, we have $y \in \hat{\alpha}^{-1}(z) \in x$ and so $y \in \bigcup x$. □

Definition 12.4 S is the collection of supported objects in $HF(I)$. We also write S for the structure $\langle S, I, \in, A\rangle$.

By the definition of supports, S is a transitive set containing all the atoms; by (5) of Lemma 12.2, it also contains I. Furthermore, by (5) and (6) of that lemma, each $\hat{\alpha}$ is a partial automorphism of S. In fact, as the following lemma shows, the $\hat{\alpha}$'s are much better than just partial automorphisms, because they fit together well. Recall that $L^k_{\infty,\omega}$ is the part of the infinitary first-order language $L_{\infty,\omega}$ (allowing infinite conjunctions and disjunctions) consisting of formulas with at most k variables (free or bound, but the same variable can be re-used). See [8] for more information about $L^k_{\infty,\omega}$, including the Ehrenfeucht-Fraïssé criterion for $L^k_{\infty,\omega}$-equivalence. Recall also that $k \geq 4$ is fixed throughout this section.

Lemma 12.5 *Let φ be a formula of $L^k_{\infty,\omega}$ with $j \leq k$ free variables. Let α be a motion, and let $x_1, \ldots, x_j$ be elements of S with supports included in $Dom(\alpha)$. Then*

$$S \models \varphi(x_1, \ldots, x_j) \iff S \models \varphi(\hat{\alpha}(x_1), \ldots, \hat{\alpha}(x_j)).$$

Proof We give a strategy for the duplicator in the Ehrenfeucht-Fraïssé game for $L^k_{\infty,\omega}$. At any stage of the game, let $\vec{y}$ and $\vec{z}$ be the positions of the pebbles on the two boards; so initially, $y_i = x_i$ and $z_i = \hat{\alpha}(x_i)$. The duplicator's strategy is to arrange that there is, after each of his moves, a motion β whose $\hat{\beta}$ sends each y_i to the corresponding z_i. There is such a β initially, namely α, and as long as he maintains such a β the duplicator cannot lose, by (5) of Lemma 12.2. So we need only check that, if such a β exists and then the spoiler moves, the duplicator can move so that again a (possibly new) β does the job. Without loss of generality, suppose the spoiler moves the first pebble on the left board from its position y_1 to a new $y'_1 \in S$. Restrict the old β to the union of supports of the y_i for $i \neq 1$. There are strictly fewer than k of these supports, hence at most $(k-1)q$ points in the domain of the restricted β. So we can extend this motion to a new β having in its domain some support of the new y'_1. The resulting $\hat{\beta}(y'_1)$ is where the duplicator should move the pebble from z_1. The resulting board position and the new β satisfy the specification of the duplicator's strategy (thanks to (2) of Lemma 12.2). So we have shown that the duplicator can carry out the indicated strategy. □

We shall need variants of these results, dealing with two graphs and the universes of hereditarily finite sets built over them. Specifically, suppose $\langle I_1, A\rangle$ and $\langle I_2, A\rangle$ are graphs satisfying the extension axioms for up to $3kq$ parameters. (We've simplified notation slightly by using the same name A for the adjacency relations in both graphs I_i.) For $i, j \in \{1, 2\}$, we define an i, j-*motion* to be a partial isomorphism of size at most kq from I_i to I_j. Thus 1,1-motions and 2,2-motions are motions in the earlier sense for I_1 and I_2, respectively. Note that the inverse of an i, j-motion is a j, i-motion and that, if α is a j, k-motion and β is an i, j-motion, then $\alpha \circ \beta$ is an i, k-motion.

If α is a 1,2-motion, then we define $\hat{\alpha}(x)$ for all $x \in HF(I_1)$ having supports included in $\mathrm{Dom}(\alpha)$. We do this in exact analogy with the earlier definition: If x is an atom then $\hat{\alpha}(x) = \alpha(x)$. If x is a set with a support $a \subseteq \mathrm{Dom}(\alpha)$, then $\hat{\alpha}(x)$ is the set of all $\hat{\beta}(y)$ where $y \in x$, β is a 1,2-motion extending $\alpha \restriction a$, and $\mathrm{Dom}(\beta)$ includes some support of y.

Similarly, we define $\hat{\alpha}(x)$ when α is a 2,1-motion and $x \in HF(I_2)$ has a support $\subseteq \mathrm{Dom}(\alpha)$. These definitions together with the definitions already available for 1,1- and 2,2-motions allow us to refer to $\hat{\alpha}(x) \in HF(I_j)$ whenever α is an i, j-motion and $x \in HF(I_i)$ has a support included in $\mathrm{Dom}(\alpha)$.

We can now repeat the earlier arguments in this slightly more general context. No conceptual changes or additions are needed, only a little bookkeeping to keep track of the four different sorts of motions. For example, the analog of Part 5 of Lemma 12.2 for 1,2-motions α would say that $\hat{\alpha}$ is a partial isomorphism from $\langle HF(I_1), \in, A, I_1\rangle$ to $\langle HF(I_2), \in, A, I_2\rangle$. It implies, in particular, that $\hat{\alpha}(I_1) = I_2$.

We exhibit for future reference the 1,2-analog of Lemma 12.5. Let S_i be the collection of supported objects in $HF(I_i)$; as before, we also write S_i for the structure $\langle S_i, \in, I_i, A\rangle$.

Lemma 12.6 *Let φ be a formula of $L^k_{\infty,\omega}$ with $j \leq k$ free variables. Let α be a 1,2-motion, and let $x_1, \ldots, x_j$ be elements of S_1 with supports included in $\mathrm{Dom}(\alpha)$. Then*

$$S_1 \models \varphi(x_1, \ldots, x_j) \iff S_2 \models \varphi(\hat{\alpha}(x_1), \ldots, \hat{\alpha}(x_j)).$$

We omit the proof because it is the same as for Lemma 12.5.

Appendix

Item (8) in Lemma 12.2 treats the two sides of the equation $\widehat{\beta \circ \alpha} = \hat{\beta} \circ \hat{\alpha}$ asymmetrically, raising the obvious question: If $\hat{\beta}(\hat{\alpha}(x))$ is defined, must $\widehat{\beta \circ \alpha}(x)$ also be defined? In this appendix, we provide an affirmative answer. This result will not be used elsewhere in the paper.

Most of the work in obtaining the affirmative answer is in establishing the special case where α and β are the identity functions on two different supports of x. Then $\hat{\beta}(\hat{\alpha}(x))$ is defined and equal to x (by (4) of Lemma 12.2) but for $\widehat{\beta \circ \alpha}(x)$ to be defined we need to know that x is also supported by the intersection of the two supports, for this is the domain of $\beta \circ \alpha$.

Lemma 12.7 *The intersection of any two supports of x is again a support of x.*

Proof The lemma is obvious if x is an atom, so we assume that x is a set.

Let the two supports of x be $a \cup a_1$ and $a \cup a_2$, where a is their intersection and a_1 and a_2 are disjoint from a (and thus from each other). We must show that, if y has lower rank than x, if α is a motion fixing a pointwise, and if y has a support $b \subseteq \mathrm{Dom}(\alpha)$, then

$$y \in x \iff \hat{\alpha}(y) \in x.$$

We intend to accomplish this by producing a finite sequence of motions $\beta_1, \ldots, \beta_j$ such that

- $\mathrm{Dom}(\beta_1 \circ \cdots \circ \beta_j) \supseteq b$,
- $\beta_1 \circ \cdots \circ \beta_j \restriction b = \alpha \restriction b$, and
- each β_i pointwise fixes either $a \cup a_1$ or $a \cup a_2$.

If we can do this then, using the assumption that both $a \cup a_1$ and $a \cup a_2$ support x, we find that

$$\begin{aligned} y \in x &\iff \hat{\beta}_j(y) \in x \\ &\iff \hat{\beta}_{j-1}(\hat{\beta}_j(y)) \in x \\ &\iff \ldots \\ &\iff \hat{\beta}_1(\ldots(\hat{\beta}_j(y))\ldots) \in x \\ &\iff (\widehat{\beta_1 \circ \cdots \circ \beta_j})(y) \in x \\ &\iff \hat{\alpha}(y) \in x, \end{aligned}$$

where at the last two steps we used (8) and (2) of Lemma 12.2. Therefore, it suffices to produce β_i's with the properties listed above.

We shall do this first in an easy case, where both b and $\alpha[b]$ are disjoint from $a_1 \cup a_2$. Enumerate b as $\{w_1, \ldots, w_r\}$. By the extension axioms, there are atoms $w'_1, \ldots, w'_r \notin a_1 \cup a_2$ such that

- $w_m \in a \rightarrow w'_m = w_m \, (= \alpha(w_m))$,
- $w_m \notin a \rightarrow w'_m \notin a \cup a_1 \cup a_2$,
- $w'_l A w'_m \iff w_l A w_m \; (\iff \alpha(w_l) A \alpha(w_m))$,
- for all $z \in a \cup a_1$, $w'_m A z \iff w_m A z$, and
- for all $z \in a \cup a_2$, $w'_m A z \iff \alpha(w_m) A z$.

The consistency of these requirements follows from the facts that α is a partial automorphism fixing a pointwise and that a_1, a_2, and a are pairwise disjoint. Let β_2 be the identity on $a \cup a_1$ and send each w_m to w'_m. Let β_1 be the identity on $a \cup a_2$ and send each w'_m to $\alpha(w_m)$. Then our choice of the w''s ensures that these are motions, and it is clear that they satisfy our requirements above. So we have achieved our goal in the easy case that $b \cup \alpha[b]$ is disjoint from $a_1 \cup a_2$.

To reduce the general case to this special case, we first observe that, by composing a motion fixing $a \cup a_1$ pointwise with one fixing $a \cup a_2$ pointwise, we can move b off $a_1 \cup a_2$. The first motion moves the elements of $b - (a \cup a_1)$ away from a_2 and the second leaves these in place while moving the elements of $b \cap a_1$ away from a_1. The existence of such motions follows from the extension axioms. Choose such motions and call their composite γ. Similarly, let δ be a composite of a motion fixing $a \cup a_1$ pointwise and one fixing $a \cup a_2$ pointwise, such that $\delta(\alpha[b])$ is disjoint from $a_1 \cup a_2$. Apply the special case to the motion $\delta \circ \alpha \circ \gamma^{-1}$, obtaining a composite of two motions, one fixing $a \cup a_1$ pointwise and one fixing $a \cup a_2$ pointwise, and agreeing with $\delta \circ \alpha \circ \gamma^{-1}$ on $\gamma[b]$. Then composing this with δ^{-1} on the left and γ on the right, we get the required composite of six morphisms — some fixing $a \cup a_1$ pointwise, others fixing $a \cup a_2$ pointwise, and the composite extending $\alpha \restriction b$. (By being careful about which motions fix which supports, one can reduce the six motions to four.) $\square$

Corollary 12.8 *If $\hat{\beta}(\hat{\alpha}(x))$ is defined, then so is $\widehat{\beta \circ \alpha}(x)$.*

Proof Since $\hat{\alpha}(x)$ is defined, x has a support $a \subseteq \mathrm{Dom}(\alpha)$. By (6) of Lemma 12.2, $\alpha[a]$ supports $\hat{\alpha}(x)$. Also, since $\hat{\beta}(\hat{\alpha}(x))$ is defined, $\hat{\alpha}(x)$ has a support $b \subseteq \mathrm{Dom}(\beta)$. Applying the lemma just proved, we find that $\alpha[a] \cap b$ also supports $\hat{\alpha}(x)$. Since this support is included in $\mathrm{Range}(\alpha)$, we can write it as $\alpha[c]$ for some $c \subseteq \mathrm{Dom}(\alpha)$. By (7) of Lemma 12.2, c supports a. Furthermore, not only is $c \subseteq \mathrm{Dom}(\alpha)$ but $\alpha[c] \subseteq b \subseteq \mathrm{Dom}(\beta)$. So $c \subseteq \mathrm{Dom}(\beta \circ \alpha)$ and therefore $\widehat{\beta \circ \alpha}(x)$ is defined. □

The preceding corollary is all that is needed to improve (8) of Lemma 12.2 to the following symmetric version.

Corollary 12.9 $\widehat{\beta \circ \alpha} = \hat{\beta} \circ \hat{\alpha}$ *in the sense that, if either of* $\widehat{\beta \circ \alpha}(x)$ *and* $\hat{\beta}(\hat{\alpha}(x))$ *is defined, then so is the other and they are equal.*

Finally, we note that Lemma 12.7 provides canonical supports for all supported objects.

Corollary 12.10 *Every supported x has a smallest support, i.e., a support that is included in every support.*

Proof Take a support a with the smallest possible cardinality. If there were another support b not including a, then $a \cap b$ would be a support of smaller cardinality than a, a contradiction. □

13 Tasks, Order, and Rank

To compute the value of a term or the update set of a rule, one needs a structure (the state of the computation) and values for the variables free in the term or rule. In most of our discussion, there will be a fixed structure under consideration but many choices of values for variables. It will therefore be convenient to push the structures into the background, often neglecting even to mention them, and to work with pairs whose first component is a term or rule and whose second component is an assignment of values to some variables (including all the variables free in the first component). We shall call such pairs "tasks" because we regard them as things to be evaluated in the course of a computation.

The official definition of tasks will differ in two respects from this informal description. First, we shall always have a particular program Π under

consideration, and we deal only with terms and rules occurring in Π. Recall in this connection our Convention 4.7, by which we are really dealing with occurrences of terms and rules. Second, it will be convenient to include in our tasks not only the obviously necessary values for free variables but also values for certain additional variables, defined as follows.

Definition 13.1 A variable v is *pseudo-free* in a term t or rule R if t or R lies in the scope of v.

Because Π, being a program, has no free variables, all the free variables in a term or rule must also be pseudo-free in it. But there may be additional pseudo-free variables. From a syntactic point of view, v is pseudo-free in t or R if one could introduce v as a free variable in t or R without violating the requirement that Π have no free variables. From a computational point of view, the pseudo-free variables of t or R are those variables to which specific values have been assigned whenever one is about to evaluate t or R in the natural algorithm for executing Π.

Definition 13.2 A *term-task* (relative to a program Π and a state H^+) is a pair $(t, \vec{a})$ where t is a term in Π and $\vec{a}$ is an assignment of values in H^+ to all the pseudo-free variables of t. *Rule-tasks* are defined similarly except that the first component is a rule in Π. Term-tasks and rule-tasks are collectively called *tasks*.

Although the second component $\vec{a}$ of a task is officially a function assigning values to the pseudo-free variables of the first component, we sometimes view it as simply a tuple of values. This makes good sense provided we imagine a fixed order for the pseudo-free variables. If, for example, the pseudo-free variables of t' are those of t plus one additional variable v, then we may write tasks $(t, \vec{a})$ and $(t', \vec{a}, b)$ with the understanding that the additional value b corresponds to the new variable v. We also sometimes write $\vec{a}\restriction$ for the restriction of the function $\vec{a}$ to an appropriate subset of its domain; it will always be clear from the context what the appropriate subset is.

When the first component of a task has no pseudo-free variables, we often omit the empty second component and regard the first component by itself as a task. In particular, the whole program Π can be regarded as a task.

We write $\mathrm{Val}(t, \vec{a})$ for the value of the term t in the structure H^+ (assumed fixed throughout the discussion) when the free variables are assigned values according to $\vec{a}$.

We shall need, for several purposes, a somewhat unusual ordering on the set of all tasks. We call it the "computational order" because it seeks to capture the intuition of one task being evaluated before another in the execution of Π.

Definition 13.3 The *computational order* $\prec$ is the smallest transitive relation on tasks satisfying the following conditions.

- $(t_i, \vec{a}) \prec (f(t_1, \ldots, t_j), \vec{a})$ for $1 \leq i \leq j$.
- $(r, \vec{a}) \prec (\{s : v \in r : \varphi\}, \vec{a})$ and, for each $b \in \mathrm{Val}(r, \vec{a})$, all three of $(v, \vec{a}, b)$, $(s, \vec{a}, b)$ and $(\varphi, \vec{a}, b)$ are $\prec (\{s : v \in r : \varphi\}, \vec{a})$.
- $(t_i, \vec{a}) \prec (f(t_1, \ldots, t_j) := t_0, \vec{a})$ for $0 \leq i \leq j$.
- All three of $(\varphi, \vec{a})$, $(R_0, \vec{a})$, and $(R_1, \vec{a})$ are $\prec$ $(\texttt{if } \varphi \texttt{ then } R_0 \texttt{ else } R_1 \texttt{ endif}, \vec{a})$.
- $(r, \vec{a}) \prec (\texttt{do forall } v \in r,\ R \texttt{ enddo}, \vec{a})$ and, for each $b \in \mathrm{Val}(r, \vec{a})$, both $(v, \vec{a}, b)$ and $(R, \vec{a}, b)$ are $\prec$ $(\texttt{do forall } v \in r,\ R \texttt{ enddo}, \vec{a})$.
- If r is the range of a variable v pseudo-free in t or R, then $(r, \vec{a}\restriction) \prec (t, \vec{a})$ or $(R, \vec{a})$.

Except for the last clause, the definition assigns as the immediate predecessors of a task simply the subterms or subrules of its first component, equipped with suitable values for the variables. We observe that, in these clauses, the lower task has values for either the same variables as the upper or one additional variable if in passing to the subterm or subrule we moved into the scope of an additional variable.

The last clause, however, is quite different. The lower task involves fewer variables than the upper, since every variable pseudo-free in the range r of v is also pseudo-free in the whole term or rule binding v and therefore in all terms or rules within the scope of v. Note also that in this last clause v is among the pseudo-free variables of t or R but not among those of r. So in this last clause, which we refer to as the *range clause*, the second component of the lower task is a proper restriction of the second component of the upper task. On the other hand, the first component r of the lower task may well be much larger than the first component t or R of the upper task.

Intuitively, if one task precedes another in this computational ordering then, in the natural calculation involved in the execution of Π, the former

task would be carried out before the latter. In the range clause, the idea is that, before attempting to evaluate t or R, we would have assigned a value to v, and before that we would have evaluated the range in order to know what the possible values for v are.

This intuition strongly suggests that the computational order should be well-founded. In fact it is, but the proof is not trivial and will require defining some rather strange-looking height functions. (Without the range-clause, the proof would be trivial, since in all the other clauses, as one goes downward in the ordering, the first component of the task gets strictly smaller.) To treat term-tasks and rule-tasks simultaneously, we use $X, Y, \ldots$ to stand for either terms t or rules R.

Proposition 13.4 *There is a height function ρ, mapping terms and rules to natural numbers, such that if $(X, \vec{a}) \prec (Y, \vec{b})$ then $\rho(X) < \rho(Y)$.*

Proof Let V be the number of variables occurring in Π, and let D be the maximum depth of nesting of terms and rules occurring in Π.

We begin by fixing a numerical function τ on the variables occurring in Π, such that if x is bound in the range of y then $\tau(x) < \tau(y)$. (Recall Convention 4.6 which ensures that every variable of Π is bound exactly once and has a well-defined range.) Such a τ exists because, if x is bound in the range of y then the term binding x is a proper subterm of the term or rule binding y. So we can define $\tau(v)$ to be the size (i.e., number of symbols in) the term or rule that binds v. (The depth of nesting of subterms could also be used instead of the size.)

Next, we define a preliminary version σ of the desired height function. Call a variable *relevant* to a term or rule X if it is either pseudo-free in X or bound in X. In other words, the scope of v either includes or is included in X. Define $\sigma(X)$ to be the sum of $V^{\tau(v)}$ over all variables relevant to X.

We analyze what can happen to σ of the first component of a task as we go down in the computational order, treating each clause in the definition of $\prec$ in turn.

When we go from $f(t_1, \ldots, t_j)$ to any t_i, the same variables remain pseudo-free, and we may lose but cannot gain bound variables, so σ cannot increase.

When we go from $\{s : v \in r : \varphi\}$ to r, σ decreases because v loses its relevance (as might some additional variables) and no new variables become relevant.

When we go from $\{s : v \in r : \varphi\}$ to any of s, v, and φ, the pseudo-free variables remain pseudo-free, the bound variable v becomes pseudo-free, and we may lose but cannot gain other bound variables, so σ cannot increase.

Similarly, in any of the three non-range clauses governing rules, the only change in relevant variables is that in general some bound variables might lose their relevance, a bound variable might become pseudo-free, and in going from `do forall` $v \in r,\ R$ `enddo` to r, v definitely loses its relevance. So in this one case σ decreases while in all cases σ does not increase.

Finally, we come to the range clause, where we go from a term or rule X to the range r of one of its pseudo-free variables. What variables are relevant to r? First, there are the pseudo-free variables of r. Their scopes include r, therefore include the whole term or rule binding v, and therefore include X. So these variables are also pseudo-free in X and therefore relevant to X. Furthermore, they are distinct from v, as the scope of v does not include its range r. Second, there are the variables bound in r, which have nothing to do with X. Let W be the number of these variables, and notice that $W < V$. Thus, as we go from X to r, the set of relevant variables has lost at least one member, namely v, but it has gained W others. These other members x, being bound in the range of v, have $\tau(x) < \tau(v)$. Therefore, the effect on σ is a decrease of at least $V^{\tau(v)}$ (from the loss of v) and an increase of at most $W \cdot V^{\tau(v)-1}$ (from the gain of the other W variables). Because $W < V$, the net effect is a decrease in σ.

The preceding discussion shows that, at any downward step in the computational ordering, either σ of the first component of the task decreases, or else it remains the same while the first component is replaced by a proper subterm or subrule and therefore the depth of nesting in this term or rule decreases.

Therefore, let us define

$$\rho(X) = (D+1)\sigma(X) + \mathrm{depth}(X)$$

where $\mathrm{depth}(X)$ means the depth of nesting of terms and rules in X. (So $\mathrm{depth}(\Pi) = D$.) Then this $\rho(X)$ decreases whenever $\sigma(X)$ does, because, although $\mathrm{depth}(X)$ might increase, it cannot increase by more than the depth D of the whole program. Also, $\rho(X)$ decreases when $\sigma(X)$ stays the same and $\mathrm{depth}(X)$ decreases. Therefore, $\rho(X)$ decreases at every downward step in the computational ordering. □

This proposition immediately implies that the computational order is a

strict partial order; it has no cycles. Since finite partial orders are well-founded, it is legitimate to do proofs by induction with respect to $\prec$. Such proofs can also be regarded as ordinary mathematical induction on the height ρ. Furthermore, induction on ρ is somewhat more powerful than induction with respect to $\prec$ because ρ-induction allows us, when proving a statement for some task, to assume the same statement not only for all computationally earlier tasks $(X, \vec{a})$ but also for all tasks $(X, \vec{b})$ having the same first components as these. This is because ρ depends only on the first component.

The following lemma will be useful technically, and it also serves to clarify the role of the range clause in the definition of the computational ordering. The definition of this ordering $\prec$ ensures that, if one task T precedes another T' then there is a chain

$$T' = T_0 \succ T_1 \succ \cdots \succ T_n = T$$

joining them, in which each two consecutive terms are related as in one of the clauses in Definition 13.3.

Lemma 13.5 *If $T' \succ T$ then there is a chain as above, in which the range clause is used, if at all, only at the first step, from T_0 to T_1.*

Proof We show, by induction on the length of a chain

$$T' = T_0 \succ T_1 \succ \cdots \succ T_n = T$$

as above, that there is another chain, of the same length n or shorter, satisfying the conclusion of the lemma. If the range clause isn't used in the given chain or is used only at the first step, we're done. There remain two cases to consider.

Case 1: The range clause is used at the first step and again later.

Applying the induction hypothesis to the sub-chain from T_1 to T_n, we may assume that the only later use of the range clause is at the second step, from T_1 to T_2. Then these two consecutive steps can be replaced by a single step from T_0 to T_2 by the range clause. To see this, consider the first components X_i of these T_i and notice that X_1 is the range of a variable v pseudo-free in X_0 and X_2 is the range of a variable w pseudo-free in X_1. But then w is pseudo-free in the whole term or rule Y binding v and is therefore also pseudo-free in X_0 which is part of Y. Thus, the range clause applied to the variable w justifies a step from T_0 to T_2. Now that the chain has

been shortened, we apply the induction hypothesis to arrange that the range clause is used, if at all, only at the first step.

Case 2: The first use of the range clause occurs later than the first step.

Let this first use be at the step from T_k to T_{k+1} (where $k \geq 1$). As before, we use the notation X_i for the first component of T_i. So X_{k+1} is the range of a variable v pseudo-free in X_k. If this variable v is pseudo-free in some earlier X_j, with $j < k$, then we can shorten our chain by going directly from X_j to X_{k+1}. The induction hypothesis then completes the proof.

So we may assume that X_k is the first X_i in which v is pseudo-free. It follows (since the step from X_{k-1} to X_k didn't use the range clause) that X_{k-1} is the term or rule binding v. But then we can omit T_k from our chain, going directly from X_{k-1} to its sub-term X_{k+1} (the range of v) by a non-range clause. Again, we've shortened the chain, and the induction hypothesis completes the proof. □

Corollary 13.6 *If, in the situation of the lemma, the first component of T' has no pseudo-free variables (in particular if T' is Π), then there is a chain as above, in which the range clause is not used.*

Proof By the lemma, we can arrange that the range clause is used, if at all, only in the first step of the chain. But since there are no pseudo-free variables in the first task, the range clause is not applicable at the first step. □

14 Involved and Active Elements

Throughout this section, we assume that we are dealing with a fixed program Π and a fixed state arising during its execution on some input $\langle I, A\rangle$. As before, we write H for the structure $\langle HF(I), \in, I, A\rangle$. The state under consideration is an expansion of H, interpreting the dynamic function symbols as well as the other static function symbols. We write H^+ for this state. Notice that, by Proposition 11.1, the interpretations of the additional function symbols of H^+ are all definable in H. The definitions of the dynamic function symbols depend on the stage of the computation; the others do not.

The following definition is intended to capture the intuitive notion of an element of $HF(I)$ being "looked at" during the execution of a task T. It

doesn't quite fulfill this intention, for it includes too many elements, pretending for example that execution of an `if ... then ... else ...` includes execution of both the consitituent rules regardless of the truth value of the guard. Nevertheless, it serves our purposes because (1) it includes all the elements that are really needed (see Proposition 14.9 below) and (2) it doesn't include too many additional elements (see Proposition 14.10 below).

Definition 14.1 An element $c \in HF(I)$ is *involved* in a task $(X, \vec{a})$ (with respect to a program Π and a state H^+) if either it is active in $(H^+, \vec{a})$ or it is the value in H^+ of some term-task $\preceq (X, \vec{a})$.

The rest of this section is devoted to establishing the properties of involved elements that will be needed in the proof of the zero-one law.

We begin with the trivial observation that any element involved in a task is also involved in any task that is higher in the computational ordering.

Proposition 14.2 *The set of elements involved in a task $(X, \vec{a})$ is transitive.*

Proof We proceed by induction on $(X, \vec{a})$ with respect to the computational order $\prec$. The definition of "involved" and the fact that $\preceq$ is a transitive relation imply that the set of elements involved in $(X, \vec{a})$ is the union of three parts:

- the set of active elements,
- the set of elements involved in term-tasks $\prec (X, \vec{a})$, and
- $\{\mathrm{Val}(X, \vec{a})\}$ if X is a term.

The first of these sets is transitive by definition of "active" and the second by induction hypothesis (since the union of a collection of transitive sets is again transitive). So it remains to show that, when X is a term, every element d of $\mathrm{Val}(X, \vec{a})$ is in one of the first two sets. We consider the various possibilities for the term X.

If X is a variable v_i, then $d \in a_i$. Since a_i is critical, d is active.

If X is $\{t(v) : v \in r : \varphi(v)\}$, then $d = \mathrm{Val}(t, \vec{a}, b)$ for some $b \in \mathrm{Val}(r, \vec{a})$. Then d is involved in $(t, \vec{a}, b) \prec (X, \vec{a})$.

There remains the possibility that X has the form $f(t_1, \ldots, t_j)$. If f is a predicate or Atoms or $\emptyset$ or a dynamic function symbol, then $\mathrm{Val}(X, \vec{a})$ is critical, and so d is active. There remain three possibilities for f, namely $\bigcup$,

TheUnique, and Pair. We can ignore TheUnique, for whenever it produces a set (as it must since d is to be a member) it coincides with $\bigcup$.

If X is $\bigcup t$, then $d \in e$ for some $e \in \mathrm{Val}(t, \vec{a})$. Then, by induction hypothesis, d is involved in $(t, \vec{a}) \prec (X, \vec{a})$.

Finally, if X is $\mathrm{Pair}(t_1, t_2)$, then $d = \mathrm{Val}(t_i, \vec{a})$ is involved in $(t_i, \vec{a}) \prec (X, \vec{a})$ for $i = 1$ or 2. □

Proposition 14.3 *Suppose a rule R, when executed in pebbled state $(H^+, \vec{a})$, updates a dynamic function f at arguments $\vec{b}$, giving $f(\vec{b})$ the value c. Then c and all components of $\vec{b}$ are involved in the task $(R, \vec{a})$ (with respect to H^+).*

Proof We proceed by induction on $(R, \vec{a})$ with respect to the computational ordering. The hypothesis implies that R has one of the forms

- $f(t_1, \ldots, t_j) := t_0$,
- `if` φ `then` R_0 `else` R_1 `endif`,
- `do forall` $v \in r$, $R_0(v)$ `enddo`.

In the first case, c and the components of $\vec{b}$ are $\mathrm{Val}(t_i, \vec{a})$; they are involved in $(R, \vec{a})$ because $(t_i, \vec{a}) \prec (R, \vec{a})$.

In the second case, the update in question comes from one of the R_i, and so c and all components of $\vec{b}$ are involved in $(R_i, \vec{a})$ by induction hypothesis. But then they are also involved in $(R, \vec{a})$ because $(R_i, \vec{a}) \prec (R, \vec{a})$.

Finally, in the third case, the update in question comes from $R_0(d)$ for some $d \in \mathrm{Val}(r, \vec{a})$, and so c and all components of $\vec{b}$ are involved in $(R_0, \vec{a}, d)$ by induction hypothesis. But then they are also involved in $(R, \vec{a})$ because $(R_0, \vec{a}, d) \prec (R, \vec{a})$. □

We remark for future reference that the preceding proof applies also if the rule "wants to" update $f(\vec{b})$ to c, in the sense of Section 11's "wants to set" formulas, even if a clash prevents the update from actually taking place.

Corollary 14.4 *Any object that is not active in a state H^+ but is active in its sequel with respect to Π must be involved in the task Π with respect to H^+.*

Proof In view of the definition of "active" and of Proposition 14.2, it suffices to consider the critical elements of the sequel that were not critical in H^+. These elements are among the values c and components of tuples $\vec{b}$

such that Π, executed in state H^+, updates some dynamic f at $\vec{b}$ to have value c. By the proposition, they are involved in Π with respect to H^+. $\square$

Our next goal is to show that the set-theoretic definitions in Section 11 retain their meaning when all quantifiers are interpreted as ranging only over elements involved in the terms or rules. In particular, the dynamic functions at any stage of the computation can be defined, as in Proposition 11.1, in the substructure of H consisting of only the elements involved in Π at some stage of the computation.

We shall use here the notation of Section 11: H^+ denotes a state; $H = \langle HF(I), \in, I, A\rangle$ is its reduct to the vocabulary containing only $\in$, Atoms, and the adjacency symbol A; and H^D is the intermediate structure in which, in addition to $\in$, Atoms, and A, the dynamic function symbols are interpreted as in H^+. We also use the notations $\lceil\varphi\rceil$ for various statements φ, as introduced in Section 11.

If C is a subset of $HF(I)$ that contains I, `true`, and `false` (as members) and is closed under the dynamic functions of H^D, then we write C^D for the substructure of H^D determined by C; its base set is C and its functions are the restrictions to C of the functions of H^D. We shall be concerned exclusively with the case where C is a transitive set; in this case we refer to C^D as a *transitive substructure* of H^D.

Definition 14.5 Let H^+ be a state and C^D a transitive substructure of H^D. Let $\varphi(\vec{y})$ be a first-order formula in the vocabulary of H^D, having free variables among $\vec{y}$ and possibly also containing (names for) some elements of C as parameters. Then $\varphi(\vec{y})$ is *absolute* for C if, for all choices of values $\vec{c} \in C$ for the variables $\vec{y}$,

$$C^D \models \varphi(\vec{c}) \iff H^D \models \varphi(\vec{c}).$$

Lemma 14.6 *Let $t = t(\vec{v})$ be any term with pseudo-free variables $\vec{v}$, let $\vec{a}$ be an assignment of values to those variables, and let C^D be a transitive substructure of H^D containing all elements involved in the task $(t, \vec{a})$ with respect to H^+. Then the formulas $\lceil y \in t(\vec{a})\rceil$ and $\lceil y = t(\vec{a})\rceil$ are absolute for C.*

Proof Notice first that C contains I, the truth values, the components of $\vec{a}$, and all the values of dynamic function symbols in H^D, for all of these are active in $(H^+, \vec{a})$ and therefore involved in $(t, \vec{a})$ and therefore is in C. So C^D

makes sense. Notice also that $\mathrm{Val}(t, \vec{a})$ is also involved in $(t, \vec{a})$ and therefore is in C.

To prove the lemma, we inspect all the clauses in the definitions of $\lceil y \in t(\vec{a}) \rceil$ and $\lceil y = t(\vec{a}) \rceil$ in Section 11 and check that the meanings of these formulas are the same whether quantifiers are interpreted as ranging over all of $HF(I)$ or only over C. That means, in the case of existential quantifiers, if a witness exists in $HF(I)$, then one should exist in C; in the case of universal quantifiers it means that, if a counterexample exists in $HF(I)$, then one should exist in C. The verification of this is very similar for many of the clauses, so we present it explicitly for only a few clauses (including the most difficult ones) and leave the rest to the reader.

First, consider the transitions connecting $\lceil y = t \rceil$ and $\lceil y \in t \rceil$. Given the former, we produced the latter as $\exists z \, (\lceil z = t \rceil \wedge y \in z)$. The existential quantifier here has (when t is replaced with $t(\vec{a})$ as in the statement of the lemma) $\mathrm{Val}(t, \vec{a})$ as its only witness, and this lies in C because it is involved in $(t, \vec{a})$. Conversely, given $\lceil y \in t \rceil$, we produced $\lceil y = z \rceil$ as

$$y \notin \mathrm{Atoms} \wedge \forall z (z \in y \leftrightarrow \lceil z \in t \rceil).$$

The only possible counterexamples for the universal quantifier here are the members of $\mathrm{Val}(t, \vec{a})$ and the members of y. The former are in C because they are involved in $(t, \vec{a})$ by Proposition 14.2, and the latter will, when we apply the definition of absoluteness, be in C also because y will be instantiated as an element $c \in C$ and C is transitive.

Now we proceed by induction on terms, looking at some typical clauses in the recursive definitions of $\lceil y = t \rceil$ and $\lceil y \in t \rceil$.

If t is $f(t_1, \ldots, t_j)$ for some dynamic function symbol f, then $\lceil y = t \rceil$ was defined as

$$\exists z_1 \ldots \exists z_j \left(\bigwedge_{i=1}^{j} \lceil z_i = t_i \rceil \wedge y = f(z_1, \ldots, z_j) \right).$$

The only witnesses for the existential quantifiers $\exists z_i$ are the corresponding $\mathrm{Val}(t_i, \vec{a})$, which are involved in $(t, \vec{a})$ because $(t_i, \vec{a}) \prec (t, \vec{a})$. In addition, we need to know that each of the subformulas $\lceil z_i = t_i \rceil$ means the same in C^D as in H^D (when the pseudo-free variables $\vec{v}$ of t are instantiated by $\vec{a}$). This follows from the induction hypothesis, because all elements involved in $(t_i, \vec{a})$ are also involved in the higher (in $\prec$) task $(t, \vec{a})$ and because the only relevant value of z_i was already seen to be in C.

If t is $\bigcup t_1$ then $\lceil y \in t \rceil$ was defined as

$$\exists z\,(y \in z \wedge \lceil z \in t_1 \rceil).$$

The witnesses for $\exists z$ here are the members of $\mathrm{Val}(t_1, \vec{a})$, and these are in C because $(t_1, \vec{a}) \prec (t, \vec{a})$ and C is transitive. In addition, $\lceil z \in t_1 \rceil$ is absolute for C by induction hypothesis.

If t is $\{t_1(v) : v \in r : \varphi(v)\}$ then $\lceil y \in t \rceil$ was defined as

$$\exists v\,(\lceil v \in r \rceil \wedge \lceil \mathtt{true} = \varphi(v) \rceil \wedge \lceil y = t_1(v) \rceil).$$

The possible witnesses for $\exists v$ here are the members b of $\mathrm{Val}(r, \vec{a})$, and these are in C because $(r, \vec{a}) \prec (t, \vec{a})$ and C is transitive. In addition, the subformulas of the form $\lceil \ldots \rceil$ are absolute by induction hypothesis because, for each $b \in \mathrm{Val}(r, \vec{a})$, both $(\varphi, \vec{a}, b)$ and $(t_1, \vec{a}, b)$ are $\prec (t, \vec{a})$.

The three clauses we have treated here are typical; the remaining eleven require no new ideas, so we leave them to the reader. □

Lemma 14.7 *Let $R = R(\vec{v})$ be any rule with pseudo-free variables $\vec{v}$, let $\vec{a}$ be an assignment of values to those variables, and let C^D be a transitive substructure of H^D containing all elements involved in the task $(R, \vec{a})$ with respect to H^+. Then, for each dynamic function symbol f, the formula $\lceil R(\vec{a})$ wants to set f at $x_1, \ldots, x_j$ to $y \rceil$ is absolute for C.*

Proof Consider first an update rule involving the relevant function symbol; say $R(\vec{v})$ is $f(t_1, \ldots, t_j) := t_0$. Then $\lceil R(\vec{a})$ wants to set f at $x_1, \ldots, x_j$ to $y \rceil$ was defined to be

$$\bigwedge_{i=1}^{j} \lceil x_i = t_i \rceil \wedge \lceil y = t_0 \rceil.$$

This is absolute because, by the preceding lemma,the subformulas $\lceil x_i = t_i \rceil$ and $\lceil y = t_0 \rceil$ are absolute. This application of the preceding lemma requires that C contain all elements involved in any of the tasks $(t_i, \vec{a})$, but this requirement is satisfied since all these tasks are $\prec (R, \vec{a})$.

If R is `do forall` $v \in r$, R_1 `enddo` then $\lceil R(\vec{a})$ wants to set f at $x_1 \ldots, x_j$ to $y \rceil$ was defined to be

$$\exists v\,(\lceil v \in r \rceil \wedge \lceil R_1 \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y \rceil).$$

The possible witnesses for $\exists v$ here are the elements of $\mathrm{Val}(r, \vec{a})$, and these are in C just as in the $\{t_1(v) : v \in r : \varphi(v)\}$ case of the preceding lemma. And the subformula $\lceil R_1$ wants to set f at $x_1, \ldots, x_j$ to $y\rceil$ is absolute by induction hypothesis, since, for each $b \in \mathrm{Val}(r, \vec{a})$ we have $(R_1, \vec{a}, b) \prec (R, \vec{a})$.

If R is a conditional rule `if` φ `then` R_0 `else` R_1 `endif`, then the induction hypothesis applied to R_0 and R_1 immediately gives the result. The remaining two cases, `Skip` and an update rule for a dynamic function other than f are trivial, since `false` is always absolute. □

Lemma 14.8 *Let C^D be a transitive substructure of H^D containing all elements involved in the task Π, our whole program. Then $\lceil\Pi$ clashes$\rceil$ is absolute for C, and so are $\lceil\Pi$ sets f at $x_1, \ldots, x_j$ to $y\rceil$ and $\lceil y = f(x_1, \ldots, x_n)$ in the sequel$\rceil$ for each dynamic function symbol f.*

Proof Again, we inspect the definitions in Section 11. $\lceil\Pi$ clashes$\rceil$ was defined as

$$\bigvee_f \exists x_1 \quad \ldots \quad \exists x_j \exists y \exists z$$
$$(\Pi \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } y) \wedge$$
$$(\Pi \text{ wants to set } f \text{ at } x_1, \ldots, x_j \text{ to } z) \wedge y \neq z.$$

The witnesses for the existential quantifiers here are either components of locations where Π wants to update f or the new values that Π wants to give f at these locations. By Proposition 14.3 and the observation immediately following its proof, all these elements are involved in Π and are therefore in C. Furthermore, for such values of the variables, the two "wants to set" subformulas in $\lceil\Pi$ clashes$\rceil$ are absolute by the preceding lemma; therefore so is $\lceil\Pi$ clashes$\rceil$.

The formula $\lceil\Pi$ sets f at $x_1, \ldots, x_j$ to $y\rceil$ is absolute because it was defined as a propositional combination of subformulas which we've already proved absolute.

Finally, $\lceil y = f(x_1, \ldots, x_n)$in the sequel$\rceil$ was defined as

$$\lceil\Pi \text{ sets } f \text{ at } x_1, \ldots, x_j \text{ to } y\rceil \vee$$
$$(\lceil y = f(x_1, \ldots, x_j)\rceil \wedge \neg\exists y' \lceil\Pi \text{ sets } f \text{ at } x_1, \ldots, x_j \text{ to } y'\rceil.$$

Here the possible witnesses for $\exists y'$ are in C thanks to Proposition 14.3, and with such a value for y' all the subformulas are absolute by the preceding two lemmas. □

Proposition 14.9 *Let C be a transitive substructure of H. Consider a run of Π, and assume that C contains all elements that are involved in Π with respect to states H^+ occurring in this run. Then, for every dynamic function symbol f and every natural number m, the formula $\lceil y = f(x_1, \ldots, x_j)$ at step $m\rceil$ is absolute for C.*

Recall that the formulas $\lceil t_0 = g(t_1, \ldots, t_k)$ at step $m\rceil$ are in the vocabulary of H, not H^D, i.e., they do not involve the dynamic function symbols. That is why the proposition refers to absoluteness for C rather than C^D.

Proof We proceed by induction on m, following the construction of $\lceil y = f(x_1, \ldots, x_j)$ at step $m\rceil$ in the proof of Proposition 11.1. The basis of the induction is trivial, as $\lceil y = \emptyset\rceil$ is certainly absolute. At the induction step, $\lceil y = f(x_1, \ldots, x_j)$ at step $m{+}1\rceil$ was formed from $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$ by replacing atomic subformulas $\lceil t_0 = g(t_1, \ldots, t_k)\rceil$ involving dynamic function symbols with $\lceil t_0 = g(t_1, \ldots, t_k)$ at step $m\rceil$. This last formula, $\lceil t_0 = g(t_1, \ldots, t_k)$ at step $m\rceil$, is absolute for C by induction hypothesis. That means that, if we use this formula, in C, to define interpretations of the dynamic function symbols, the result is a substructure C^D of H^D. Then the preceding lemma can be applied to these structures, since C contains the elements involved in any state H^+ of the run, to infer that, with these interpretations, $\lceil y = f(x_1, \ldots, x_j)$ in the sequel$\rceil$ has the same meaning in C as in H. But that means that $\lceil y = f(x_1, \ldots, x_j)$ at step $m + 1\rceil$ is absolute. □

Observe for future reference that the proof of the proposition would remain correct if we assumed only that C contains all elements involved in Π with respect to states occurring at or before step m. Later states were not involved in the argument for absoluteness at step m.

The final property of involved elements that we shall need is that there are not too many of them.

Proposition 14.10 *For any state H^+ and any task $(X, \vec{a})$, the number of elements involved in $(X, \vec{a})$ is bounded by a polynomial function of the number of active elements in $(H^+, \vec{a})$. The polynomial depends only on the term or rule X.*

Proof We proceed by induction on the task $(X, \vec{a})$ with respect to the computational order. Notice that the elements involved in $(X, \vec{a})$ are of three sorts:

1. active elements,

2. elements involved in tasks immediately preceding $(X, \vec{a})$ in the computational order, and

3. $\mathrm{Val}(X, \vec{a})$ if X is a term.

Here "immediately preceding $(X, \vec{a})$" means explicitly listed as preceding it in Definition 13.3, without using the transitivity of $\prec$. We need not concern ourselves with tasks strictly but not immediately preceding $(X, \vec{a})$, since anything involved in them is also involved in a task immediately preceding $(X, \vec{a})$.

We must associate to every term or rule X a polynomial $p_X(n)$ such that, when there are n active elements then at most $p_X(n)$ elements can be involved in $(X, \vec{a})$ for any $\vec{a}$. Referring to the list above, we see that there are n elements of sort 1 and at most one element of sort 3, so it suffices to bound the number of elements of sort 2.

Notice that the range clause in Definition 13.3 produces only a fixed finite number of immediate predecessors (independent of H^+) for any $(X, \vec{a})$, since X has only finitely many pseudo-free variables and each of these has only one range.

When X is a variable or the rule `Skip`, there are no other immediate predecessors. When X is a term of the form $f(t_1, \ldots, t_j)$ or an update rule or a conditional rule, then there are only a fixed finite number of other immediate predecessors. In any of these cases, the required bound on the number of elements of sort 2 can be obtained simply by summing the polynomials associated to the fixed family of immediate predecessors.

There remain two cases, a term of the form $\{t(v) : v \in r : \varphi(v)\}$ and a parallel combination rule. Consider first the case that X is $\{t(v) : v \in r : \varphi(v)\}$. The immediate predecessors of $(X, \vec{a})$ are those given by the range clause, $(r, \vec{a})$, and for each $b \in \mathrm{Val}(r, \vec{a})$ the three tasks $(v, \vec{a}, b)$, $(t(v), \vec{a}, b)$, and $(\varphi(v), \vec{a}, b)$. Since all elements of $\mathrm{Val}(r, \vec{a})$ are involved in $(r, \vec{a})$, we can bound the number of elements of sort 2 produced by a rule other than the range rule by

$$p_r(n) + p_r(n) \cdot (p_v(n) + p_t(n) + p_\varphi(n)).$$

Adding finitely many polynomials to account for elements of sort 2 produced by the range rule, and adding $n + 1$ to account for elements of sorts 1 and 3, we obtain the required polynomial for $\{t(v) : v \in r : \varphi(v)\}$.

The case of a parallel combination is very similar. Let X be `do forall` $v \in r,\ R(v)$ `enddo`. The immediate predecessors of $(X, \vec{a})$ are those given by the range rule, $(r, \vec{a})$, and for each $b \in \mathrm{Val}(r, \vec{a})$ the two tasks $(v, \vec{a}, b)$ and $(R(v), \vec{a}, b)$. Since all elements of $\mathrm{Val}(r, \vec{a})$ are involved in $(r, \vec{a})$, we can bound the number of elements of sort 2 produced by a rule other than the range rule by

$$p_r(n) + p_r(n) \cdot (p_v(n) + p_R(n)).$$

Adding finitely many polynomials to account for elements of sort 2 produced by the range rule, and adding $n + 1$ to account for elements of sorts 1 and 3, we obtain the required polynomial for `do forall` $v \in r,\ R(v)$ `enddo`. □

15 Combinatorics

In this section, we prove the main combinatorial lemma needed for the proof of the main theorem. As in the preceding sections, the parameters q and k will be fixed here. The graph $\langle I, A \rangle$ of atoms will be assumed to satisfy the strong extension axioms for up to $3kq$ parameters.

By a *polymer* we mean a sequence of at most kq atoms. (The reason for the terminology is that, in [4], we used "molecule" for a one-to-one listing of a support; here that would be a sequence of length $\leq q$. A polymer is essentially the concatenation of up to k molecules. It gives the supports for up to k objects.) The *configuration* of a polymer consists of the following information: which components are equal and which are adjacent in the graph $\langle I, A \rangle$ of atoms. If the polymer has length l, then its configuration could be viewed as the combination of an equivalence relation on $\{1, 2, \ldots, l\}$ (telling which components are equal) and an irreflexive symmetric relation on the quotient set (telling which components are adjacent). By the *joint configuration* of two (or more) polymers, we mean the equality and adjacency information about all components of both (or all) of the polymers. It could be viewed as the configuration of the concatenation of the polymers, except for the technicality that the concatenation may be too long to count as a polymer.

We shall be concerned with equivalence relations of the following sort.

Definition 15.1 A *configuration-determined equivalence* (*cde* for short) is an equivalence relation E with the following two properties.

- Its domain consists of all polymers of one specified configuration.
- Whether two polymers ξ and η of this configuration are related by E depends only on their joint configuration.

When dealing with polymers ξ of a specified configuration (e.g., those in the domain of a cde), we can simplify their presentation by omitting any repetitions of components in ξ. Notice that the configuration of ξ tells us where the repetitions must occur, so, knowing the configuration, we can recover ξ from its compressed version. Notice also that the configuration of the compressed version is entirely determined by that of ξ. Also, if ξ and η both have the specified configuration, then their joint configuration determines and is determined by the joint configuration of their compressed versions. Because of this tight correspondence between the polymers ξ and their compressed versions, we can confine our attention to the compressed versions; that is, we can assume that we deal only with polymers that are one-to-one sequences.

Now we are ready to state the theorem that is the combinatorial heart of the zero-one law.

Theorem 15.2 *Assume that SEA_{3kq} holds and that $|Atoms| \geq kq2^{3kq+1}$. Let E be a configuration-determined equivalence with fewer than*

$$\frac{1}{(q+1)!}\left(\frac{|Atoms|}{2^{3kq+1}}\right)^{q+1}$$

equivalence classes. Let l be the common length of the polymers in the domain of E. There exists a set $u \subseteq \{1, 2, \ldots, l\}$ of size at most q, and there exists a group G of permutations of u such that, for any ξ and η in the domain of E,

$$\xi E \eta \iff (\exists \sigma \in G)\,(\forall i \in u)\,\xi_i = \eta_{\sigma(i)}.$$

Notice that, although l can be as large as kq, the theorem requires u to be relatively small, of size at most q.

The conclusion of the theorem completely describes E in terms of u and G. It says that the E-equivalence class of ξ consists of those polymers (of the right configuration) obtainable from ξ by

- permuting the components indexed by u, using a permutation from G, and

- changing the other components completely arbitrarily.

In particular, the equivalence class of ξ depends only on the ξ_i for $i \in u$.

The hypothesis of the theorem involves a complicated bound on the number of equivalence classes. Most of the complication disappears if one remembers that q and k are fixed, so the bound is, up to a constant factor, just $|\text{Atoms}|^{q+1}$. When we apply the theorem, the cde's of interest will be such that the number of equivalence classes is bounded by a polynomial in $|\text{Atoms}|$ of degree at most q. So the bound will automatically be satisfied once the number of atoms is large enough.

Proof Let E be a cde satisfying the hypotheses of the theorem. All polymers mentioned below are assumed to be of the right configuration so that they are in the domain of E.

Since the theorem requires u to be small and since the components ξ_i of a polymer for $i \notin u$ must be irrelevant to its E-equivalence class, we shall have to obtain some strong "irrelevance of components" results. The following is a first step in that direction, giving, under certain circumstances, one irrelevant component.

Lemma 15.3 *Suppose $\xi E \eta$ and suppose a certain component ξ_r of ξ does not occur in η. Then r is irrelevant in the sense that, if ζ and ζ' agree except in position r, then $\zeta E \zeta'$.*

Proof Fix ξ and η as in the hypothesis of the lemma, and build a sequence of polymers, $\xi^0, \xi^1, \ldots, \xi^L$ such that:

- $L = l!$, where l is the length of the polymers.
- $\xi^0 = \xi$ and $\xi^1 = \eta$.
- Every two consecutive polymers ξ^m and ξ^{m+1} in the sequence have the same joint configuration as ξ and η.
- Any atom in ξ^{m+1} that isn't in ξ^m isn't in ξ either.

Such a sequence exists because the extension axioms allow us to choose the ξ^m by induction on m. Notice in this connection that the constraints on ξ^{m+1} involve only the immediately previous ξ^m and the first ξ, so extension axioms up to $3kq$ parameters suffice.

We remark that each two consecutive polymers in our list are E-equivalent, since they have the same joint configuration as the E-equivalent pair ξ, η and E is configuration-determined. Of course it follows that the entire list lies in a single E-equivalence class.

Because of our convention of using compressed polymers (i.e., tuples without repetitions), there is a one-to-one partial function σ from $\{1, 2, \ldots, l\}$ to itself telling us where to find components of ξ in η. That is,

$$\begin{aligned} \xi_i = \eta_{\sigma(i)} \quad & \text{if } i \in \mathrm{Dom}(\sigma) \\ \xi_i \text{ doesn't occur in } \eta \quad & \text{if } i \notin \mathrm{Dom}(\sigma). \end{aligned}$$

We claim that the m^{th} iterate σ^m similarly tells us where to find components of ξ in ξ^m. Indeed, if $i \in \mathrm{Dom}(\sigma^m)$ then, because all pairs of consecutive polymers in our list have the same joint configuration as ξ, η,

$$\xi_i = \eta_{\sigma(i)} = \xi^1_{\sigma(i)} = \xi^2_{\sigma^2(i)} = \cdots = \xi^m_{\sigma^m(i)}.$$

What if $i \notin \mathrm{Dom}(\sigma^m)$? Then there is a unique $j < m$ such that $i \in \mathrm{Dom}(\sigma^j) - \mathrm{Dom}(\sigma^{j+1})$. Then, as in the computation above, $\xi_i = \xi^j_{\sigma^j(i)}$ and, since $\sigma^j(i) \notin \mathrm{Dom}(\sigma)$, $\xi_{\sigma^j(i)}$ doesn't occur in η. But ξ^j and ξ^{j+1} have the same joint configuration as ξ and η. So $\xi^j_{\sigma^j(i)} = \xi_i$ doesn't occur in ξ^{j+1}. By the final requirement in our choice of the sequence of polymers, ξ_i cannot occur in any polymer after ξ^{j+1} in our sequence. In particular, it doesn't appear in ξ^m. This completes the verification of our claim about σ^m:

$$\begin{aligned} \xi_i = \xi^m_{\sigma^m(i)} \quad & \text{if } i \in \mathrm{Dom}(\sigma^m) \\ \xi_i \text{ doesn't occur in } \xi^m \quad & \text{if } i \notin \mathrm{Dom}(\sigma^m). \end{aligned}$$

We apply the preceding information with $m = L = l!$. Then σ^L is the identity function on its domain. To see this, recall that σ is a one-to-one partial function from $\{1, 2, \ldots, l\}$ to itself, and visualize it as a directed graph, with vertex set $\{1, 2, \ldots, l\}$ and with arcs $(i, \sigma(i))$. The connected components of this graph are some directed paths leading to points $\notin \mathrm{Dom}(\sigma)$ and some directed cycles (because σ is one-to-one). On any of the paths, since there are at most l vertices, if we iterate σ at least l times we get a totally undefined function; in particular, σ^L is undefined at all points in the paths. As for the cycles, their lengths are also $\leq l$ and therefore divide L. So σ^L is the identity function on the cycles.

Summarizing the preceding discussion, we have that $\xi E \xi^L$ and that any any atom occurring as a component in both ξ and ξ^L must occur in the same position in both and must also occur (though not necessarily in the same position) in η. Recall that ξ and η were chosen to satisfy, with a certain r, the hypothesis of the lemma. So ξ_r does not occur in η and therefore does not occur in ξ^L either.

Now suppose ζ and ζ' agree except at position r. Since we deal with one-to-one polymers, ζ_r does not occur in ζ' and ζ'_r does not occur in ζ. To complete the proof of the lemma, we must show that $\zeta E \zeta'$.

The extension axioms provide a polymer θ (of the right configuration to be in the domain of E) such that both pairs (ζ, θ) and (ζ', θ) have the same joint configuration as (ξ, ξ^L). To see this, consider first those positions i where $\xi_i = \xi^L_i$. As noted above, r is not such a position, so $\zeta_i = \zeta'_i$ and we can (indeed we must) take $\theta_i = \zeta_i = \zeta'_i$. For all other positions j, ξ^L_j doesn't occur in ξ, so the θ_j we seek should not occur in ζ or in ζ', but it should satisfy certain adjacencies and non-adjacencies with the components of ζ and ζ' and the other components of θ. The extension axioms allow us to choose the desired θ_j's one after the other; at each step, the constraints on the desired θ_j are consistent because any common component of ζ and ζ' (which might threaten to give two constraints) occurs in the same position in both of these polymers (because this is the case for ξ and ξ^L), so it gives merely two occurrences of the same constraint.

Because $\xi E \xi^L$ and because E is configuration-determined, we have both $\zeta E \theta$ and $\zeta' E \theta$. Therefore $\zeta E \zeta'$, and the proof of the lemma is complete. □

Returning to the proof of the theorem, we proceed to define the required u and G. Let

$$u = \{i \in \{1, 2, \ldots, l\} : \text{Whenever } \xi E \eta \text{ then } \xi_i \text{ occurs in } \eta\}.$$

We record for future reference that the preceding lemma gives us part of the information about u required in the theorem.

Corollary 15.4 *If ζ and θ are polymers in the domain of E and if $\zeta_i = \theta_i$ for all $i \in u$, then $\zeta E \theta$.*

Proof We first reduce the problem to the case where ζ and θ are disjoint except on u, i.e., $\zeta_i \neq \theta_{i'}$ for all $i, i' \notin u$. To do this, we use a polymer η in the domain of E that agrees on u with ζ (and therefore with θ) but is disjoint

from both of them except on u. Such an η can easily be found by using the extension axioms to produce it one component at a time. If the corollary holds for polymers disjoint except on u, then we have $\zeta E \eta$ and $\eta E \theta$, and so $\zeta E \theta$, as required.

So from now on we assume that ζ and θ are disjoint except on u (and of course agree on u). The idea for the proof in this situation is to connect ζ and θ by a sequence of polymers, each differing from the next in just one position, that position being outside u. Then the lemma will tell us that consecutive polymers in this sequence are E-equivalent; therefore so are the ends ζ and θ of the sequence. In order to apply the lemma, however, we must make sure that all the polymers in our sequence are in the domain of E, i.e., that they have the right configuration. This means that we cannot, in general, form our sequence by simply replacing the unwanted components of ζ one at a time by the corresponding components of θ. Instead, we go from ζ to θ via an intermediate polymer μ that agrees on u with ζ and θ, is disjoint from them except on u, and is chosen so that the domain of E contains all the polymers obtained by replacing the unwanted components of ζ one by one (say in left to right order) with those of μ and then replacing these (in the opposite order) with the corresponding components of θ. These requirements on μ trivially determine its components on u. Its remaining components are subject to certain constraints to ensure disjointness and especially to ensure that all the polymers in our sequence have the same configuration and are therefore in the domain of E. These constraints require, for each non-u component of μ, certain adjacency or non-adjacency relationships with the u components of ζ, with the earlier (i.e. farther left) components of μ, and with the later components of ζ and θ. These required relationships cannot contradict each other because ζ and θ are disjoint except on u. So the extension axioms allow us to find the desired components of μ one at a time. □

For each pair ξ, η of E-equivalent polymers, let $\sigma = \sigma_{\xi,\eta}$ be the one-to-one partial function as in the proof of the lemma, i.e.,

$$\begin{aligned} \xi_i = \eta_{\sigma(i)} \quad & \text{if } i \in \mathrm{Dom}(\sigma) \\ \xi_i \text{ doesn't occur in } \eta \quad & \text{if } i \notin \mathrm{Dom}(\sigma). \end{aligned}$$

We define

$$G = \{\sigma_{\xi,\eta} \restriction u : \xi E \eta\}.$$

It is clear from the definition of u that the domain of each $\sigma_{\xi,\eta}$ includes u.

We check next that each $\sigma = \sigma_{\xi,\eta}$ maps u into itself. Indeed, suppose $i \in u$ but $\sigma(i) \notin u$. So there exist polymers $\zeta E \theta$ such that $\zeta_{\sigma(i)}$ does not occur in θ. Since all polymers under consideration have the same configuration, the extension axioms provide a polymer λ such that the pair (λ, ζ) has the same joint configuration as (ξ, η). Therefore $\lambda E \zeta$ because E is configuration-determined, and $\lambda_i = \zeta_{\sigma(i)}$. But that means that λ_i doesn't occur in θ, yet λ and θ are E-equivalent as both are equivalent to ζ. This contradicts $i \in u$ and thus completes the proof that each member of G maps u into itself.

As each σ is one-to-one and as u is finite, we now know that G is a set of permutations of u. It contains the identity permutation (as $\sigma_{\xi,\xi}$) and inverses of all its members (for $\sigma_{\xi,\eta}^{-1} = \sigma_{\eta,\xi}$). To see that G is closed under composition, consider any two of its members, say $\sigma_{\xi,\eta}$ and $\sigma_{\xi',\eta'}$ where $\xi E \eta$ and $\xi' E \eta'$. Since all four polymers here have the same configuration, extension axioms provide ζ such that (ζ, ξ) has the same configuration as (ξ', η') and no components of ζ are in η except for ones also in ξ. Then $\zeta E \xi E \eta$ and $\sigma_{\zeta,\xi} = \sigma_{\xi',\eta'}$. Therefore (omitting for brevity the restriction to u that should accompany each σ)

$$\sigma_{\xi,\eta} \circ \sigma_{\xi',\eta'} = \sigma_{\xi,\eta} \circ \sigma_{\zeta,\xi} = \sigma_{\zeta,\eta} \in G.$$

Thus G is a group of permutations of u. We check next that E can be defined in terms of u and G as in the statement of the theorem. One direction of that equivalence is trivial, for if $\xi E \eta$ then $\sigma_{\xi,\eta} \restriction u$ serves as the required $\sigma \in G$. For the other direction, suppose $(\forall i \in u)\, \xi_i = \eta_{\sigma(i)}$, where $\sigma = \sigma_{\zeta,\theta}$ for some $\zeta E \theta$. The extension axioms provide a polymer λ (with the same configuration as all the others) such that the joint configuration of (λ, η) matches that of (ζ, θ). So for all $i \in u$ we have $\lambda_i = \eta_{\sigma(i)} = \xi_i$, so the corollary of the lemma tells us that $\lambda E \xi$. Furthermore, as $\zeta E \theta$ and E is configuration-determined, we have $\lambda E \eta$. Therefore $\xi E \eta$, as desired.

All that remains in the proof of the theorem is to show that $|u| \leq q$. (A priori, u could be as large as l, which in turn could be as large as kq, the maximum length for a polymer.) Suppose, toward a contradiction, that $|u| \geq q + 1$. Let us abbreviate

$$\frac{|\text{Atoms}|}{2^{3kq+1}} \quad \text{as} \quad z.$$

Then the hypothesis of the theorem ensures that $z \geq kq$.

Also, the strong extension axioms ensure that every type with at most kq parameters is realized at least z times. This allows us to estimate the number

of E-equivalence classes. To build a polymer ξ (of the right configuration to be in the domain of E), we first choose, in succession, its components at positions in u; at each step there are at least z choices satisfying all the constraints imposed by the configuration, so there are at least $z^{|u|}$ possibilities for $\xi \restriction u$. Extend each of these to a polymer ξ of the right configuration. Note that, among the $z^{|u|}$ polymers so obtained, two are E-equivalent if and only if one is transformed to the other by a permutation (of component positions) from G. Now G, being a group of permutations of u, has order at most $|u|!$, and so each E-equivalence class of our chosen polymers has size at most $|u|!$. Therefore, there are at least $z^{|u|}/|u|!$ equivalence classes.

As a function of m, $z^m/m!$ is non-decreasing as long as m remains $\leq z$. Indeed,

$$\frac{z^{m-1}}{(m-1)!} \leq \frac{z^m}{m!} \iff m \leq z.$$

Since we know $|u| \leq kq \leq z$ and since we are assuming $q+1 \leq |u|$, it follows that $z^m/m!$ is a non-decreasing function of m in the interval from $q+1$ up to $|u|$. In particular, $z^{q+1}/(q+1)! \leq z^{|u|}/|u|!$. Therefore, the number of E-equivalence classes, as estimated above, is at least $z^{q+1}/(q+1)!$. That contradicts the hypothesis of the theorem. □

16 Involved Objects Are Supported

In this section, we assemble most of the preceding work to show that, for a run of a BGS program with a polynomial bound on the number of active elements, all elements involved in the program, with respect to any state of the run, are supported. Once this is established, it will be easy to deduce Theorem 2.2, from which the zero-one law, Theorem 2.1, is an immediate consequence.

Fix a BGS program Π and a polynomial bound $\sigma(n)$ on the number of active elements (in terms of the cardinality n of the input graph). Consider any run of Π, on input graph $\langle I, A\rangle$, or any initial segment of such a run, in which the number of active elements never exceeds $\sigma(n)$ where $n = |I|$. By Proposition 14.10, the number of elements involved in Π at any state in the run is bounded by a polynomial function p_Π of the number of active elements and therefore by a polynomial function $p_\Pi(\sigma(n))$ of the input size n. Increasing this last polynomial if necessary, we may assume that the number of involved elements is bounded by Cn^q for certain positive integers C and

q. Notice that the proof of Proposition 14.10 provides an explicit calculation of a suitable polynomial p_Π. Since σ is also given explicitly, we can compute C and q effectively from Π and σ.

In all our previous discussions of supports, starting in Section 12, we assumed that the parameters $q \geq 1$ and $k \geq 4$ had been fixed arbitrarily. We now give them the following specific values: q gets the value described at the end of the last paragraph, the exponent in the bound Cn^q. The value of k is to be the sum of

- twice the number of variables in Π,
- the bound B from Proposition 11.1, and
- 4.

Henceforth, all references to supports, motions, and related concepts are to be understood with these values of q and k. Also, we assume the strong extension axioms for up to $3kq$ parameters in our input graph.

The main result of this section is that, under the assumptions above, involved elements in sufficiently large graphs are supported.

Proposition 16.1 *Let Π, σ, C, q, and k be as above. Let $\langle I, A\rangle$ be a graph satisfying the strong extension axiom SEA_{3kq}. Consider a run or an initial segment of a run of Π on input $\langle I, A\rangle$ in which each state has at most $\sigma(|I|)$ active elements. If $|I|$ is large enough, then, in every state of the run, every element involved in Π is supported.*

After giving the proof, we shall extract an explicit estimate of how large $|I|$ should be.

Proof We prove the proposition by induction on the steps of the run (or partial run). More precisely, we prove the following two properties for each state H^+ occurring in the run.

- Every active element is supported.
- Every element involved in Π is supported.

(Of course the second of these, the one we are really interested in, subsumes the first, but it is convenient to consider the first separately.) In the initial state, all dynamic functions are constant with value $\emptyset$, and so the only active

elements are the atoms, the set I of atoms, and the two truth values. All these are supported (see Lemma 12.2 for the atoms and the set of atoms; the truth values $\emptyset$ and $\{\emptyset\}$ are trivial to check directly). Furthermore, if both our claims are true at some state in the run, then the first claim, that all active elements are supported, will be true in the next state, by Corollary 14.4. So to complete the induction, it suffices to show, for each state H^+ in the run, that if every active element is supported then so is every element involved in Π.

For the rest of the proof, we work with a fixed state H^+ that occurs, in a (possibly partial) run of Π, in which the input is a sufficiently large graph satisfying SEA_{3kq}, and in which the number of active elements never exceeds $\sigma(|I|)$. We assume that all active elements of this state H^+ are supported, and we must prove that all elements involved in Π with respect to H^+ are also supported. In view of the definition of "involved," it suffices to prove that $\mathrm{Val}(t, \vec{a})$ is supported for every term-task $(t, \vec{a}) \prec \Pi$.

This proof will be carried out by induction on the height $\rho(t)$ of t, where ρ is defined as in Proposition 13.4. By that proposition, we know that, when we are proving that $\mathrm{Val}(t, \vec{a})$ is supported, we may assume the same property for all predecessors of $(t, \vec{a})$ in the computational order and also for all tasks having the same first components as such predecessors.

To facilitate carrying out the induction, we strengthen the statement being proved, as follows. We shall show, by induction on $\rho(t)$, that all term-tasks $(t, \vec{a}) \prec \Pi$ have the following three properties.

1. All the a_i are supported.

2. If α is a motion whose domain includes supports of all the a_i, then $(t, \hat{\alpha}(\vec{a})) \prec \Pi$.

3. Given any supports for the components a_i of $\vec{a}$, their union includes a set that supports $\mathrm{Val}(t, \vec{a})$.

In Assertion 2, $\hat{\alpha}(\vec{a})$ means the result of applying $\hat{\alpha}$ to each component a_i of $\vec{a}$. Assertions 1 and 3 together imply that $\mathrm{Val}(t, \vec{a})$ is supported, which was our goal.

For the proofs of Assertions 1 and 2, it will be useful to first invoke Corollary 13.6 to obtain a chain $\mathcal{S}$ of tasks

$$(t, \vec{a}) \prec \cdots \prec \Pi$$

in which each two consecutive tasks are related as in one of the first five clauses in Definition 13.3, i.e., not using the range clause. This means in particular that, of any two tasks in $\mathcal{S}$,

- the first component of the earlier task is a subterm or subrule of the first component of the later task, and
- the second component of the later task is a restriction of the second component of the earlier task

(We've displayed the chain here in the opposite order from Lemma 13.5, so that words like "predecessor" will have the same meaning in the display as in the ordering $\prec$.)

Proof of Assertion 1 Consider any a_i, and let $(s, \vec{b}, a_i)$ be the last task in $\mathcal{S}$ in which a_i appears. Let r be the range of the variable v_i (that has the value a_i in $\vec{a}$), and notice, by inspection of the non-range clauses in Definition 13.3, that $a_i \in \mathrm{Val}(r, \vec{b})$. Because $(r, \vec{b}) \prec (t, \vec{a})$ by the range clause, our induction hypothesis ensures that $\mathrm{Val}(r, \vec{b})$ is supported. Therefore, so is its element a_i. This establishes Assertion 1 . □

Proof of Assertion 2 Apply $\hat{\alpha}$ to the second component of every task in $\mathcal{S}$; this makes sense because, as noted above, these second components are restrictions of $\vec{a}$, all of whose components have supports included in the domain of α. We claim that, in the resulting sequence of tasks, each is still an immediate predecessor of the next in the ordering $\prec$ (and still without using the range clause). This claim will clearly suffice to establish Assertion 2. Inspection of the relevant (i.e., non-range) clauses in Definition 13.3 shows that the claim is trivial except when a pseudo-free variable disappears, i.e., when we have $(s, \vec{c}, b)$ or $(v, \vec{c}, b)$ or $(\varphi, \vec{c}, b)$ as an immediate predecessor of $\{s : v \in r : \varphi\}$ or when we have $(v, \vec{c}, b)$ or $(R, \vec{c}, b)$ as an immediate predecessor of `do forall` $v \in r$, R `enddo`. In these cases, we have $b \in \mathrm{Val}(r, \vec{c})$ and we need to know that $\hat{\alpha}(b) \in \mathrm{Val}(r, \hat{\alpha}(\vec{c}))$ in order to know that we still have an immediate predecessor relationship after applying $\hat{\alpha}$.

Fortunately, $(r, \vec{c}) \prec (t, \vec{a})$ by the range clause. So we know, by induction hypothesis, that the values of $(r, \vec{c})$ and of all its $\prec$-predecessors are supported. Since all active elements are supported, we know that every element involved in the task $(r, \vec{c})$ is supported. Therefore, the class S of supported objects satisfies the hypotheses of Lemma 14.6 for the task $(r, \vec{c})$. This allows us to argue as follows.

Since $b \in \mathrm{Val}(r, \vec{c})$, the formula $\lceil b \in r(\vec{c}) \rceil$ is true in H^D. (Recall from Section 11 that H^D is the reduct of H^+ in which only $\in$, Atoms, A, and the dynamic function symbols are interpreted. Recall also that H is the further reduct, omitting the interpretations of the dynamic function symbols.) By Lemma 14.6, $\lceil b \in r(\vec{c}) \rceil$ is also true in S^D, the substructure of H^D whose base set is the set S of supported elements. (The requirement in Lemma 14.6 that S^D be a substructure, i.e., that S be closed under the dynamic functions, is satisfied because values of dynamic functions are active and are therefore in S.)

Let H^+_* be the state immediately preceding H^+ in our run of Π and recall that, since the proposition is being proved by an induction along the run, all elements involved in Π with respect to H^+_* are supported. Therefore, Lemma 14.8 provides definitions in H, absolute for S, of the dynamic functions in the sequel of H^+_*, i.e., in the state H^+. Inserting these definitions for the dynamic functions into $\lceil b \in r(\vec{c}) \rceil$, we obtain a formula, true in H and absolute for S, expressing that $b \in \mathrm{Val}(r, \vec{c})$ in H^+. Since this formula is true in S, it remains true in S if we apply $\hat{\alpha}$ to b and to all components of $\vec{c}$, by Lemma 12.5.

Now we reverse the preceding steps to obtain $\hat{\alpha}(b) \in \mathrm{Val}(r, \hat{\alpha}(\vec{c}))$. In order to do this, we need to know that the task $(r, \hat{\alpha}(\vec{c}))$ has, like $(r, \vec{c})$, the property that all elements involved in it (with respect to H^+) are supported. Although $(r, \hat{\alpha}(\vec{c}))$ need not be $\prec (t, \vec{a})$ (as $(r, \vec{c})$ is), it has the same first component, and this suffices since (1) we are proceeding by induction on $\rho(t)$, which depends only on the first component and (2) $(r, \hat{\alpha}(\vec{c})) \prec \Pi$ by Assertion 2 for $(r, \vec{c})$.

This completes the proof that $\hat{\alpha}(b) \in \mathrm{Val}(r, \hat{\alpha}(\vec{c}))$, which suffices to complete the proof of Assertion 2. □

Before proceeding to the proof of Assertion 3, it will be useful to give a short name to the main part of the preceding proof, because it will be needed again. We are referring to the argument leading from $b \in \mathrm{Val}(r, \vec{c})$ to $\hat{\alpha}(b) \in \mathrm{Val}(r, \hat{\alpha}(\vec{c}))$. We shall refer to this argument as "applying the motion α via S." Notice that this argument required that the domain of α includes supports of b and all components of $\vec{c}$ and that everything involved in the tasks $(r, \vec{c})$ and $(r, \hat{\alpha}(\vec{c}))$ is supported. Notice also that the same argument would apply to statements of the form $b = \mathrm{Val}(r, \vec{c})$.

Proof of Assertion 3 This proof splits into cases depending on the nature of the term t. If t is a variable v_i, then $\mathrm{Val}(t, \vec{a}) = a_i$ and the assertion is

trivially true.

Consider next the hardest case, namely where t is $\{s : v \in r : \varphi\}$. Choose supports for all the a_i and let ξ_0 be a polymer in which all those supports are listed. (Note that, by our choice of k, the number of variables in Π is less than $\frac{1}{2}(k-4)$; since supports have size at most q, the length of ξ_0 is less than $\frac{1}{2}(k-4)q$, well below the length bound kq in the definition of "polymer.") For any other polymer ξ of the same configuration as ξ_0, let α be the motion sending ξ_0 to ξ. Thus, $\hat{\alpha}(a_i)$ is defined for all i. Write $t(\xi)$ for $\mathrm{Val}(t, \hat{\alpha}(\vec{a}))$. In particular, $t(\xi_0)$ is the object $\mathrm{Val}(t, \vec{a})$ that we hope to prove to be supported by a subset of the range of ξ_0.

Define an equivalence relation E on the set of polymers of the same configuration as ξ_0 by

$$\xi E \xi' \iff t(\xi) = t(\xi').$$

We claim that this E is configuration-determined. To prove this claim, suppose $\xi E \xi'$ and suppose another pair η, η' has the same joint configuration as ξ, ξ'; we must show $\eta E \eta'$. Let α be (as above) the motion $\xi_0 \mapsto \xi$, and similarly, let $\alpha' : \xi_0 \mapsto \xi'$. Also, let θ be the motion sending $\xi \mapsto \eta$ and $\xi' \mapsto \eta'$. (Such a θ exists because the joint configurations agree and because ξ and ξ' together involve fewer than $(k-4)q$ elements, well below the bound kq for the size of the domain of a motion.) Of course then the composite motions $\theta \circ \alpha$ and $\theta \circ \alpha'$ send ξ_0 to η and η', respectively. In view of the definition of E, we know $\mathrm{Val}(t, \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\alpha}'(\vec{a}))$ and we must deduce $\mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}'(\vec{a}))$ (where we've also used part (8) of Lemma 12.2).

To prove the desired equation, it suffices, by symmetry, to establish the inclusion in one direction. So we consider an arbitrary element $z \in \mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}(\vec{a}))$ and we try to prove $z \in \mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}'(\vec{a}))$. Since t is $\{s : w \in r : \varphi\}$, we have $z = \mathrm{Val}(s, \hat{\theta}\hat{\alpha}(\vec{a}), b)$ for some $b \in \mathrm{Val}(r, \hat{\theta}\hat{\alpha}(\vec{a} \restriction))$ with $\mathtt{true} = \mathrm{Val}(\varphi, \hat{\theta}\hat{\alpha}(\vec{a}), b)$. (As usual, $\vec{a} \restriction$ here denotes the restriction of $\vec{a}$ to the variables pseudo-free in r.)

By induction hypothesis, b is supported. Extend θ, if necessary, so that its range includes a support of b. (The extension axioms provide a partial isomorphism, and it is a motion because its domain still has cardinality less than $(k-3)q$.) Since the range of θ includes supports of b and of all components of $\hat{\theta} \circ \hat{\alpha}(\vec{a})$ (see Lemma 12, part 6), another application of the induction hypothesis gives that the range of θ supports $z = \mathrm{Val}(s, \hat{\theta}\hat{\alpha}(\vec{a}), b)$. Applying motion θ^{-1} via S to the formulas at the end of the preceding

paragraph, we obtain

- $\widehat{\theta^{-1}}(z) = \mathrm{Val}(s, \hat{\alpha}(\vec{a}), \widehat{\theta^{-1}}(b))$,
- $\widehat{\theta^{-1}}(b) \in \mathrm{Val}(r, \hat{\alpha}(\vec{a}\restriction))$, and
- $\mathtt{true} = \mathrm{Val}(\varphi, \hat{\alpha}(\vec{a}), \widehat{\theta^{-1}}(b))$.

The support information needed for this application of a motion via S follows from the induction hypothesis, because s, r, and φ all have lower height than t. The three displayed formulas imply that $\widehat{\theta^{-1}}(z) \in \mathrm{Val}(t, \hat{\alpha}(a)) = \mathrm{Val}(t, \hat{\alpha}'(\vec{a}))$.

So we can fix $b' \in \mathrm{Val}(r, \hat{\alpha}'(\vec{a}\restriction))$ such that $\widehat{\theta^{-1}}(z) = \mathrm{Val}(s, \hat{\alpha}'(\vec{a}), b')$ and $\mathtt{true} = \mathrm{Val}(\varphi, \hat{\alpha}'(\vec{a}), b')$. Extend θ again, if necessary, so that its domain includes a support of b'. (Its domain still has size smaller than $(k-2)q$, so it is a motion.) Using the induction hypothesis again to get the necessary support information, apply the motion θ via S to the three formulas at the beginning of this paragraph. The results are

- $\hat{\theta}(b') \in \mathrm{Val}(r, \hat{\theta} \circ \hat{\alpha}'(\vec{a}\restriction))$,
- $z = \mathrm{Val}(s, \hat{\theta} \circ \hat{\alpha}'(\vec{a}), \hat{\theta}(b'))$, and
- $\mathtt{true} = \mathrm{Val}(\varphi, \hat{\theta} \circ \hat{\alpha}'(\vec{a}), \hat{\theta}(b'))$.

These facts say that $z \in \mathrm{Val}(t, \hat{\theta}\hat{\alpha}'(\vec{a}))$, as desired. This completes the proof that $\eta E \eta'$ and thus that E is configuration determined.

We wish to apply the combinatorial Theorem 15.2 to the configuration determined equivalence relation E. For this purpose, we must bound the number of atoms from below and bound the number of equivalence classes from above. The lower bound is no problem, since the proposition we are proving includes the hypothesis that I is "large enough." As for the upper bound, notice that the number of equivalence classes is, by definition of E, just the number of different elements of the form $t(\xi) = \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$ as ξ ranges over polymers of the same configuration as ξ_0. By Assertion 2, already proved, all the tasks $(t, \hat{\alpha}(\vec{a}))$ are $\prec \Pi$, so their values are involved in Π (with respect to the current state H^+). Then, by our choice of C and q, the number of these elements is bounded above by $C|I|^q$. For sufficiently large $|I|$, this polynomial of degree q is smaller than the polynomial of of degree $q+1$

$$\frac{1}{(q+1)!}\left(\frac{|I|}{2^{3kq+1}}\right)^{q+1}$$

occurring in Theorem 15.2. So this theorem can be applied to E. We thus see that $t(\xi)$ depends only on the restriction of ξ to a certain set u of size at most q. We shall show that the range of $\xi_0 \restriction u$ supports $\mathrm{Val}(t, \vec{a})$; in view of the definition of ξ_0, this will complete the proof of Assertion 3 for terms of the form $\{s : v \in r : \varphi\}$.

The first part of the definition of support requires that all elements of $\mathrm{Val}(t, \vec{a})$ are supported. But these elements are of the form $\mathrm{Val}(s, \vec{a}, b)$ for suitable b, so they are supported, thanks to the induction hypothesis applied to s.

To verify the main clause in the definition of support, suppose that z has lower rank than $\mathrm{Val}(t, \vec{a})$ and that α is a motion, fixing the alleged support $\xi_0(u)$ pointwise, and having a support of z included in its domain. We must prove that

$$z \in \mathrm{Val}(t, \vec{a}) \iff \hat{\alpha}(z) \in \mathrm{Val}(t, \vec{a}).$$

Since α fixes $\xi_0(u)$ pointwise, $\xi_0 E (\alpha \circ \xi_0)$, which means, by definition of E, that $\mathrm{Val}(t, \vec{a}) = \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$. So what we must prove is that

$$z \in \mathrm{Val}(t, \vec{a}) \iff \hat{\alpha}(z) \in \mathrm{Val}(t, \hat{\alpha}(\vec{a})).$$

Notice that we cannot simply apply motion α via S to obtain this equivalence, for one of the support conditions that would be required is that everything involved in $(t, \vec{a})$ is supported; in particular, we would need to know that $\mathrm{Val}(t, \vec{a})$ is supported, which is part of what we are trying to prove. Therefore, we take an indirect approach, very similar to the argument showing that E is configuration determined.

We assume that $z \in \mathrm{Val}(t, \vec{a})$ and prove that $\hat{\alpha}(z) \in \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$; the converse is proved the same way, using α^{-1} in place of α. The assumption $z \in \mathrm{Val}(t, \vec{a})$ means, in view of the form of t, that

- $z = \mathrm{Val}(s, \vec{a}, b)$,
- $b \in \mathrm{Val}(r, \vec{a}\restriction)$, and
- $\mathtt{true} = \mathrm{Val}(\varphi, \vec{a}, b)$

for some b. Now the induction hypothesis (applied to s, to r, and to φ) provides the support conditions needed to apply α via S. We thus obtain

- $\hat{\alpha}(z) = \mathrm{Val}(s, \hat{\alpha}(\vec{a}), \hat{\alpha}(b))$,

- $\hat{\alpha}(b) \in \mathrm{Val}(r, \hat{\alpha}(\vec{a})\restriction)$, and
- $\mathtt{true} = \mathrm{Val}(\varphi, \hat{\alpha}(\vec{a}), \hat{\alpha}(b))$.

But these facts mean that $\hat{\alpha}(z) \in \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$, as required.

This completes the proof of that $\xi_0(u)$ supports $\mathrm{Val}(t, \vec{a})$, and so it completes the proof of Assertion 3 for terms of the form $\{s : v \in r : \varphi\}$.

It remains to prove Assertion 3 when t has the form $f(t_1, \ldots, t_k)$. The proof splits into subcases depending on the function symbol f. If f is a predicate symbol or $\emptyset$ or Atoms, then $\mathrm{Val}(t, \vec{a})$ is $\mathtt{true}$ or $\mathtt{false}$ or I, all of which are supported by the empty set.

If f is $\bigcup$, so $t = \bigcup t_1$, then Corollary 12.3 tells us that $\mathrm{Val}(t, \vec{a}) = \bigcup \mathrm{Val}(t_1, \vec{a})$ is supported by any set that supports $\mathrm{Val}(t_1, \vec{a})$. Since t_1 has lower height than t, the induction hypothesis completes the proof.

Suppose next that f is TheUnique, so t is TheUnique(t_1). If the value of t is a set, then TheUnique could be replaced with $\bigcup$, and we would be back in the situation of the preceding paragraph. So assume that $\mathrm{Val}(t, \vec{a})$ is an atom c. Therefore $\mathrm{Val}(t_1, \vec{a}) = \{c\}$. By induction hypothesis, $\{c\}$ has a support s included in the union of any given supports of the a_i. Unraveling the definition of "s supports $\{c\}$," (specializing the y in the definition to be c), we find that, for every motion α fixing s pointwise and defined at c, $\alpha(c) \in \{c\}$, i.e., every such α fixes c. This requires that $c \in s$, for otherwise α could interchange c with some other atom outside s. But from $c \in s$, we obtain that s supports c, as required.

Suppose next that f is Pair, so t is Pair(t_1, t_2). The argument in this case is similar to (but a little simpler than) the one for $\{s : v \in r : \varphi\}$. Again, let ξ_0 be a polymer in which supports of all the a_i are listed, and define an equivalence relation $\xi E \xi'$ on polymers of the same configuration as ξ_0 by $t(\xi) = t(\xi')$, where $t(\xi)$ is defined as $\mathrm{Val}(t, \hat{\alpha}(\vec{a}))$ for any motion α sending ξ_0 to ξ. As before, we claim that E is configuration determined. To prove this, consider ξ, ξ', η, η', α, α', and θ as in the previous argument. That is, $\alpha : \xi_0 \mapsto \xi$, $\alpha : \xi_0 \mapsto \xi'$, $\theta : \xi, \xi' \mapsto \eta, \eta'$, and $\xi E \xi'$. We must prove $\eta E \eta'$, i.e.,

$$\mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}'(\vec{a})).$$

By symmetry, it suffices to prove one inclusion, so consider any element $z \in \mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}(\vec{a}))$. So $z = \mathrm{Val}(t_j, \hat{\theta} \circ \hat{\alpha}(\vec{a}))$ for $j = 1$ or 2. By induction hypothesis, we have the necessary support information to apply θ^{-1} via S; for example, since (the range of) ξ_0 supports $\vec{a}$, Lemma 12.2 implies that ξ

supports $\hat{\alpha}(\vec{a})$ and then that the range of θ supports $\hat{\theta} \circ \hat{\alpha}(\vec{a})$ and therefore supports z. The result of applying θ^{-1} via S is that $\hat{\theta}^{-1}(z) = \mathrm{Val}(t_j, \hat{\alpha}(\vec{a}))$. So $\hat{\theta}^{-1}(z)$ is a member of $\mathrm{Val}(t, \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\alpha}'(\vec{a}))$, where this equation comes from the assumption that $\xi E \xi'$. So $\hat{\theta}^{-1}(z) = \mathrm{Val}(t_j, \hat{\alpha}(\vec{a}))$ for some $j = 1$ or 2 (not necessarily the same j as above). Again, the induction hypothesis provides the support information needed to apply θ via S and conclude that $z = \mathrm{Val}(t_j, \hat{\theta} \circ \hat{\alpha}'(\vec{a})) \in \mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}'(\vec{a}))$, as desired. This completes the proof that E is configuration determined.

The same argument as in the case of $t = \{s : v \in r : \varphi\}$ shows that the combinatorial Theorem 15.2 applies to E. We thus find that the equivalence class of ξ, and thus the value of $t(\xi)$, depend only on $\xi \restriction u$ for a certain u of size at most q. We shall show that $\xi_0(u)$ supports $\mathrm{Val}(t, \vec{a})$. The elements of $\mathrm{Val}(t, \vec{a})$ are the $\mathrm{Val}(t_j, \vec{a})$ for $j = 1$ and 2; these are supported by induction hypothesis.

It remains to check the main clause in the definition of "support." So consider any z of lower rank than $\mathrm{Val}(t, \vec{a})$ and any motion that fixes $\xi(u)$ pointwise and is defined on a support of z. As in the earlier argument, we must prove

$$z \in \mathrm{Val}(t, \vec{a}) \iff \hat{\alpha}(z) \in \mathrm{Val}(t, \hat{\alpha}(\vec{a})).$$

(The second $\hat{\alpha}$ on the right is justified by $\xi_0 E(\alpha \circ \xi_0)$ as before.) We prove the left-to-right implication, the converse being similar with α^{-1} in place of α. Any z satisfying $z \in \mathrm{Val}(t, \vec{a})$ must satisfy $z = \mathrm{Val}(t_j, \vec{a})$ for $j = 1$ or 2. But, after checking that the induction hypothesis gives the necessary support information, we can apply α via S to get $\hat{\alpha}(z) = \mathrm{Val}(t_j, \hat{\alpha}(\vec{a}))$, which immediately gives $\hat{\alpha}(z) \in \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$, as desired. This completes the proof for the subcase where f is Pair.

There remains only the subcase that f is a dynamic function symbol. Then $\mathrm{Val}(t, \vec{a})$ is critical, hence active, and hence supported. But we need to make sure it has a support included in the union of any supports of the a_i. For this purpose, we apply once more the combinatorial Theorem 15.2; the argument follows the same general lines as in the cases where t is $\{s : v \in r : \varphi\}$ or $\mathrm{Pair}(t_1, t_2)$, but it is considerably simpler because we already know that $\mathrm{Val}(t, \vec{a})$ and indeed $\mathrm{Val}(t, \vec{b})$ for any $\vec{b}$ are supported.

As in the two earlier arguments, introduce a polymer ξ_0 whose range includes supports for all the a_i; we'll complete the proof by finding a support of the form $\xi_0(u)$ for $\mathrm{Val}(t, \vec{a})$. For this purpose define, as before, an equivalence relation on polymers of the same configuration as ξ_0 by letting $\xi E \xi'$

mean $t(\xi) = t(\xi')$, where $t(\xi) = \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$ for any motion sending ξ_0 to ξ. We claim that E is configuration determined. To prove this, suppose $\xi E \xi'$ and suppose the pair of polymers η, η' have the same joint configuration as ξ, ξ'; we must show $\eta E \eta'$. Introduce the motions $\alpha : \xi_0 \mapsto \xi$, $\alpha' : \xi_0 \mapsto \xi'$, and $\theta : \xi, \xi' \mapsto \eta, \eta'$ as in the previous arguments. Let $z = \mathrm{Val}(t, \hat{\alpha}(\vec{a}))$. Since $\xi E \xi'$, we also have $z = \mathrm{Val}(t, \hat{\alpha}'(\vec{a}))$. Extend θ if necessary so that its domain includes a support of z. The domain of θ also includes supports for $\hat{\alpha}(\vec{a})$ and $\hat{\alpha}'(\vec{a})$, namely the ranges of ξ and ξ' (by Lemma 12.2). Also, everything involved in the tasks $(t, \hat{\alpha}(\vec{a}))$ and $(t, \hat{\alpha}'(\vec{a}))$ is supported. Indeed, we observed earlier that the values of these tasks are supported (being critical and therefore active); the rest of the involved elements are, by definition, either active (and thus supported) or values of tasks $\prec$ the ones under consideration (and thus supported by induction hypothesis). These observations about supports provide the information needed to apply θ via S to the two equations about z. We obtain that $\hat{\theta}(z)$ equals both $\mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}(\vec{a}))$ and $\mathrm{Val}(t, \hat{\theta} \circ \hat{\alpha}(\vec{a}))$. Therefore $\eta E \eta'$, as desired. This completes the proof that E is configuration determined.

Estimating the number of equivalence classes exactly as in the earlier arguments, we obtain from Theorem 15.2 a set u of cardinality at most q, such that $t(\xi)$ depends only on $\xi \restriction u$. We shall show that $\xi_0(u)$ supports $\mathrm{Val}(t, \vec{a})$. Since we already know that this value is supported, all we need to check is the main clause in the definition of "support." Consider, therefore, any z of lower rank than $\mathrm{Val}(t, \vec{a})$ and any motion α that pointwise fixes $\xi_0(u)$ and that has a support of z included in its domain. We must show

$$z \in \mathrm{Val}(t, \vec{a}) \iff \hat{\alpha}(z) \in \mathrm{Val}(t, \vec{a}).$$

By our choice of u, $\xi_0 E (\alpha \circ \xi_0)$, and so what we must show is equivalent to

$$z \in \mathrm{Val}(t, \vec{a}) \iff \hat{\alpha}(z) \in \mathrm{Val}(t, \hat{\alpha}(\vec{a})).$$

Extend α if necessary, so that its domain includes supports for all the a_i. This domain also includes a support for z by hypothesis. And, as we saw above, everything involved in the tasks $(t, \hat{\alpha}(\vec{a}))$ and $(t, \hat{\alpha}'(\vec{a}))$ is supported. Thus, we have the support information needed to apply α via S to the formula $z \in \mathrm{Val}(t, \vec{a})$, and this gives the left-to-right half of the desired equivalence. The other half is proved analogously, applying α^{-1}. This completes the last subcase of the last case of Assertion 3. □

This completes the induction on tasks and thus completes the proof of Proposition 16.1. □

Remark 16.2 The proof of the preceding proposition allows us to be quite explicit about the "sufficiently large" in the statement of the proposition. Specifically, there were four places where we needed $n = |I|$ to be sufficiently large. First, SEA_{3kq} includes the requirement that $n \geq 3kq$. The use of Theorem 15.2 presupposes the second requirement

$$n \geq kq \cdot 2^{3kq+1}$$

and the third

$$\frac{1}{(q+1)!}\left(\frac{n}{2^{3kq+1}}\right)^{q+1} \geq \text{number of } E\text{-classes.}$$

Finally, in the subcase where $t = \mathrm{TheUnique}(t_1)$, we said that a motion "could interchange c with some other atom outside s," which presupposes that there is such an atom, i.e., that $n \geq q+2$ (since s, being a support, has size at most q). This fourth requirement and the first are obviously subsumed by the second, so we ignore them. In the third requirement, the number of equivalence classes with respect to E was at most the number of involved elements, which was bounded above by Cn^q. Thus, the third requirement will be satisfied provided

$$\frac{1}{(q+1)!} \cdot \frac{n^{q+1}}{2^{(3kq+1)(q+1)}} \geq Cn^q,$$

which simplifies to

$$n \geq C \cdot 2^{(3kq+1)(q+1)}(q+1)!.$$

An easy calculation shows that this version of the third requirement subsumes the second. Therefore, we can take "sufficiently large" in the proposition to mean at least $C \cdot 2^{(3kq+1)(q+1)}(q+1)!$.

It should also be pointed out that the two requirements, SEA_{3kq} and "sufficiently large" could be combined into one, since SEA_m was defined as including the statement that the graph has at least m vertices. Thus, the requirements follow from SEA_m where m is chosen to be at least the bound calculated above

At last, we are in a position to prove the main result, Theorem 2.2, from which the zero-one law follows immediately.

Proof of Theorem 2.2 Given a BGS program Π with Boolean output and given a polynomial σ, define C, q, and k as at the beginning of this section.

Let $\mathcal{C}$ be the class of graphs that are sufficiently large (in the sense of Proposition 16.1 and Remark 16.2) and satisfy SEA_{3kq}. Then $\mathcal{C}$ has asymptotic probability one, by Proposition 6.1. We shall produce m and v such that the remaining requirement of Theorem 2.2 is also satisfied: For each input $\langle I, A\rangle \in \mathcal{C}$ either Π on input $\langle I, A\rangle$ halts after exactly m steps with output value v and without exceeding the bound $\sigma(|I|)$ on the number of active elements, or else Π on input $\langle I, A\rangle$ either exceeds the bound on active elements by step m or fails to ever halt.

We may assume that there is at least one input $\langle I, A\rangle \in \mathcal{C}$ on which Π halts without exceeding the bound on active elements, for otherwise we can take arbitrary m and v and obtain the last option in the conclusion of the theorem — Π on any input $\langle I, A\rangle \in \mathcal{C}$ fails to ever halt.

So fix an input $\langle I_1, A_1\rangle \in \mathcal{C}$ on which Π halts, say after m steps and with output v. We shall complete the proof by showing that, on any other input $\langle I_2, A_2\rangle \in \mathcal{C}$, Π either halts after the same number m of steps with the same output v or else exceeds the bound on active elements by step m. We consider an arbitrary input $\langle I_2, A_2\rangle \in \mathcal{C}$ for which Π has not exceeded the bound on active elements by step m, and we compare the run of Π on input $\langle I_1, A_1\rangle$ with the possibly partial run of m steps on input $\langle I_2, A_2\rangle$. The fact that the former run halts at step m with output v means that the three formulas

$$\lceil \texttt{true} = \text{ Halt at step } m\rceil,$$
$$\lceil \texttt{false} = \text{ Halt at step } m-1\rceil, \text{ and}$$
$$\lceil v = \text{ Output at step } m\rceil$$

are satisfied in $HF(I_1)$. By Propositions 14.9 and 16.1, the same formulas are satisfied in S_1, the sub-universe of supported sets over $\langle I_1, A_1\rangle$. But then, by Lemma 12.6, these formulas also hold in S_2, the universe of supported sets over $\langle I_2, A_2\rangle$. Since the m-step partial run of Π on this input was assumed to obey the bound on active elements, we can again apply Propositions 14.9 and 16.1, obtaining that the same formulas are true in $HF(I_2)$. But that means that this partial run has reached a halting state at step m (and not yet at step $m-1$), with output v, as claimed. □

17 Counting

It was proved in [4] (and it also follows from the zero-one law) that, when the input is a set with no structure, the parity of the number of atoms is not $\tilde{\text{C}}$PTime computable. Since this parity is easy to compute — for example by a PTime algorithm that uses an ordering (or, equivalently, arbitrary choices) but produces a result independent of the ordering — it is reasonable to enlarge the complexity class $\tilde{\text{C}}$PTime to include such problems. We briefly discuss two such extensions in this section.

The weaker of the two extensions is $\tilde{\text{C}}$PTime + InputSize. It is defined exactly like $\tilde{\text{C}}$PTime except that there is one additional, nullary, static function symbol, InputSize, whose value in any initial state (and therefore in any state occurring in a run) is the von Neumann ordinal for the number of atoms. Since it is easy to compute in $\tilde{\text{C}}$PTime the parity of this ordinal, using the program (with one dynamic nullary symbol c in addition to the always present Halt and Output)

```
if c = InputSize
   then Halt := true
   else do in parallel
      c := S(c)
      Output := ¬Output
   enddo
endif
```

(where S is the successor function on von Neumann ordinals, $S(x) = x \cup \{x\}$ or in primitive notation $\bigcup(\text{Pair}(x, \text{Pair}(x, x)))$) there cannot be a zero-one law for $\tilde{\text{C}}$PTime + InputSize. Nevertheless, we have the following theorem analogous to Theorem 2.2, which seems to be as close as one could hope to get to a zero-one law in this situation.

Theorem 17.1 *Let Π be a BGS program in which InputSize is allowed to occur, and let a polynomial bound for the number of active elements be given. There exists a class $\mathcal{C}$ of undirected graphs, and there exist, for each positive integer n, a number $m(n)$ and an output value $v(n)$ such that $\mathcal{C}$ has asymptotic probability one and such that, for each $\langle I, A\rangle \in \mathcal{C}$, either*

- *Π on input $\langle I, A\rangle$ halts after exactly $m(|I|)$ steps with output value $v(|I|)$ and without exceeding the given bound on active elements, or*

- Π *on input* $\langle I, A\rangle$ *either never halts or exceeds the bound on active elements.*

This differs from Theorem 2.2 only in that m and v depend on the input size, as the parity example shows they must.

Proof The only change needed in the proof of Theorem 2.2 is that part 5 of Lemma 12.2 will need to include that $\hat{\alpha}$ maps InputSize to itself. Although this is true ($\hat{\alpha}$ fixes all von Neumann ordinals), it does not generalize to 1,2 motions unless InputSize has the same value in both structures. Thus we need to assume, when comparing BGS + InputSize computations on two graphs, that they have equal numbers of vertices. This accounts for the dependence on n in the present theorem. □

A stronger and more natural extension of $\tilde{\text{C}}$PTime is $\tilde{\text{C}}$PTime + Card, obtained by adding to the vocabulary a static, unary function symbol Card, to be interpreted as sending any set in $HF(I)$ to its cardinality (represented as a von Neumann ordinal). In this context, we lose even the weak "one size at a time" version of the zero-one law that we had for $\tilde{\text{C}}$PTime + InputSize. To see this, just apply the following lemma and notice that Card allows us to compute the parity of the number of edges in a graph.

Lemma 17.2 *For any $n \geq 2$, the probability that a random n-vertex graph has an even number of edges is $\frac{1}{2}$.*

Proof This amounts to a well known binomial coefficient identity. For a simple, direct proof, fix one potential edge e and define a bijection from graphs with an even number of edges to graphs with an odd number of edges as follows. Remove e from any graph that contains it, and add e to any graph that doesn't contain it. □

18 Total Functions and Extension Axioms

We saw in Proposition 7.1 that the ordinary extension axioms EA_k do not suffice to decide the almost sure behavior of polynomially bounded BGS programs. The program exhibited in the proof of this proposition, while producing output `true` on some inputs and `false` on others satisfying any specified extension axioms, produces no output at all on almost all graphs.

Indeed, for almost all n-vertex graphs with distinguished vertex d, the number of neighbors of d will be approximately $n/2$ and the maximum size of a clique containing d will be larger than $\log n$. So the program in question will activate all sets of neighbors of d of size $\leq \log n$, a non-polynomial number of sets.

It is reasonable to ask whether strong extension axioms are still needed if one considers only those polynomially bounded BGS programs that always produce an output, without exceeding their bounds. The purpose of this section is to present a proof, due to Shelah (unpublished), that under this restriction the ordinary extension axioms suffice.

Theorem 18.1 *Let a BGS program Π with Boolean output and a polynomial bound for the number of active elements be given. Assume that, on every input graph $\langle I, A\rangle$, Π halts without exceeding the bound on active elements. Then there exist numbers m and s and an output value v such that Π halts after exactly m steps with output v on all sufficiently large graphs satisfying EA_s.*

Proof Let k and q be as in Section 16. By the results of that section, including the proof of Theorem 2.2, we obtain a number m and an output v such that Π halts in m steps with output v and without exceeding its bound on active elements whenever the input is a sufficiently large graph $\langle I_1, A_1\rangle$ satisfying SEA_{3kq}. These values of m and v will be the ones required in the present theorem; the value of s will be specified later, but it will be at least $3kq$.

Fix a graph $\langle I_1, A_1\rangle$ as in the preceding paragraph and also satisfying EA_s (for the yet to be chosen s). Also, consider any sufficiently large graph $\langle I_2, A_2\rangle$ satisfying EA_s (but not necessarily satisfying any strong extension axioms). Thus we know that Π on input $\langle I_1, A_1\rangle$ produces output v in exactly m steps, and we wish to show that the same occurs when the input is $\langle I_2, A_2\rangle$. For this purpose, we use the argument in the last paragraph of the proof of Theorem 2.2, at the end of Section 16. The only part of this argument that might fail, because $\langle I_2, A_2\rangle$ doesn't satisfy strong extension axioms, is the application of Proposition 16.1 to $\langle I_2, A_2\rangle$. Specifically, the only thing that can go wrong is that, at step m or earlier, some non-supported element has become involved in Π.

We consider the first step where this happens, and we look back into the proof of Proposition 16.1 to see how such a situation could arise. That

proof was an induction on the steps of the computation, and since we are looking at the first troublesome step we know that all active elements are supported (since they were involved at a previous step, just as in the proof of Proposition 16.1). So something must go wrong with the argument for Assertions 1–3, which led to the result that all involved elements are supported. Looking for uses of the strong extension axioms here, we find that they occur only in the proof of Assertion 3. (More precisely: Ordinary extension axioms suffice everywhere else in the proof.) Three of the cases in the proof of Assertion 3, namely the cases where t has the form $\{s : v \in r : \varphi\}$ or $\mathrm{Pair}(t_1, t_2)$ or $f(t_1, \ldots, t_j)$ with f dynamic, required the use of Theorem 15.2, whose proof used the strong extension axiom. So our task is now to show that, even though $\langle I_2, A_2 \rangle$ doesn't satisfy the strong extension axiom, these three cases in the proof of Assertion 3 still work. Suppose not, and consider a failure involving a task $(t, \vec{a})$ for which the height $\rho(t)$ is as small as possible. Thus, we are in the same situation as in the proof of Assertion 3 insofar as all three assertions are true for all tasks of lower height; the only difference now is that we cannot invoke Theorem 15.2.

Actually, a significant part of Theorem 15.2 is still available even in our present situation without SEA_{3kq}. Inspection of the proof of Theorem 15.2 shows that only ordinary extension axioms EA_{3kq} (which are available for $\langle I_2, A_2 \rangle$ since $s \geq 3kq$) were needed to define the set u and the group G and to show that the equivalence relation E is defined in terms of u and G as in the statement of Theorem 15.2. The only thing missing is that we don't know that $|u| \leq q$. All we can say is $|u| \leq kq$, since polymers have length at most kq.

Bringing this information back into the three potentially troublesome cases in the proof of Assertion 3 in Proposition 16.1, we find that we have sets u as needed there except that $|u|$ may be bigger than q. We suppose that this actually happens, and we seek to deduce a contradiction.

Looking at the definition of u in terms of E (in the proof of Theorem 15.2) and the definition of E in terms of $(t, \vec{a})$ (in the proof of Assertion 3), we can formulate our supposition as follows, using the same notation as in the proof of Assertion 3. There are more than q values of i such that, whenever $\mathrm{Val}(t, \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\beta}(\vec{a}))$ then there is j such that $\alpha(\xi_0(i)) = \beta(\xi_0(j))$. Here ξ_0 is a polymer listing supports of all components of $\vec{a}$, and α and β range over motions defined on (at least) all components of ξ_0. (We've written i and j as argument s rather than subscripts of ξ_0 in the hope of improving readability by avoiding double subscritps.)

At this point, the argument proceeds differently for the three possible forms of t. We give the proof for the case that t is $\{s : v \in r : \varphi\}$. The other two cases are similar but easier.

The equation $\mathrm{Val}(t, \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\beta}(\vec{a}))$ occurring above can be reformulated, in view of the form of t, as: For every $b \in \mathrm{Val}(r, \hat{\alpha}(\vec{a}))$ with $\mathrm{Val}(\varphi, \hat{\alpha}(\vec{a}), b) = \mathtt{true}$, there is $b' \in \mathrm{Val}(r, \hat{\beta}(\vec{a}))$ with $\mathrm{Val}(\varphi, \hat{\beta}(\vec{a}), b') = \mathtt{true}$ and $\mathrm{Val}(s, \hat{\alpha}(\vec{a}), b) = \mathrm{Val}(s, \hat{\beta}(\mathrm{Val}(a)), b')$. The advantage of this rather awkward reformulation is that the terms whose values occur here are of lower height than t, so these values are known to be supported. (A similar reformulation works when t is $\mathrm{Pair}(t_1, t_2)$. No reformulation is needed when t begins with a dynamic function symbol, as in this case the value of t is already known to be supported; the only issue in this case of Assertion 3 was finding a support inside the range of ξ_0.)

Let us take our supposition, as formulated three paragraphs ago, and replace the equation $\mathrm{Val}(t, \hat{\alpha}(\vec{a})) = \mathrm{Val}(t, \hat{\beta}(\vec{a}))$ by the reformulation from the last paragraph. The result, which we refrain from writing out in full, but which we shall continue to call "our supposition," is not entirely about supported objects; the polymer ξ_0 can involve as many as kq atoms and the motions α and β can involve as many as $2kq$ atoms each (the members of their domains and ranges).

By virtue of Proposition 11.1, our supposition can be written as a set-theoretic statement about the structure $\langle HF(I), \in, I, A\rangle$. Let k^+ be larger than k and larger than the number of variables occurring in any such statement (for any term t occurring in Π). Notice that, just as in Proposition 11.1, the number of variables needed here does not depend on the step of the computation of Π. So k^+ is well-defined; it depends only on the program Π. Also define $q^+ = 2kq$. We shall need to use concepts like "motion" and "support," previously defined using k and q, but now with the larger numbers k^+ and q^+. To avoid confusion with the previous concepts, we shall systematically use superscripts + when referring to the new concepts. Thus, for example, a motion$^+$ is a partial automorphism with domain of size $\leq k^+q^+$. Supports$^+$ are defined exactly as in Section 12 only with k^+ in place of k and q^+ in place of q; they are sets of size $\leq q^+$.

We shall need to know that the old and new concepts are coherent. Specifically, if a supports x then a also supports$^+$ x. Also, if α is a motion defined on a support of x (and thus is also a motion$^+$ defined on a support$^+$ of x), then $\hat{\alpha}(x)$ is the same in both senses. (So we need not attach + superscripts to the hats over motions.) These facts are easily proved by induction on the

rank of x.

Define s to be $3k^+q^+$. Then the extension axiom EA_s is adequate for all the work in Section 12 as applied to the new concepts of support$^+$ and motion$^+$.

Returning to our supposition, we see that, although the objects mentioned in it are not all supported, they are all supported$^+$; indeed, q^+ was chosen precisely to be big enough for this purpose. The supposition itself is a set-theoretic statement about $\langle HF(I), \in, I, A\rangle$ with at most k^+ variables (k^+ having been chosen precisely to achieve this). So, by Lemma 12.6$^+$, it is preserved by any 2,1 motion$^+$. Such a motion exists because both $\langle I_1, A_1\rangle$ and $\langle I_2, A_2\rangle$ satisfy EA_s. (Recall, from the beginning of this proof, that $\langle I_1, A_1\rangle$ was a large graph satisfying both SEA_{3kq} and EA_s.) Thus, our supposition holds also for the computation of Π on input $\langle I_1, A_1\rangle$. That is, at a certain stage (m or earlier) in this computation, the argument for Assertion 3 in the proof of Proposition 16.1 encounters a set u of size $> q$. But this is absurd; since $\langle I_1, A_1\rangle$ satisfies SEA_{3kq}, the argument for Assertion 3 in Section 16 applies, and it shows that the relevant u's are never bigger than q. This contradiction completes the proof that our situation cannot actually occur. □

Remark 18.2 The hypothesis, in Theorem 18.1, that the computation of Π succeeds on all input graphs can be weakened. The same proof would work if we assumed only that the asymptotic probability of success is 1. In fact, all that we really needed in the proof was that, for every natural number k, Π succeeds on at least one graph satisfying SEA_k.

References

[1] Andreas Blass and Yuri Gurevich, *Choiceless Polynomial Time Computation and the Zero-One Law*, in Computer Science Logic 2000, editors Peter Clote and Helmut Schwichtenberg, Springer Lecture Notes in Computer Science 1862 (2000) 18–40.

[2] Andreas Blass and Yuri Gurevich, *A new zero-one law and strong extension axioms*, to appear in Bull. European Assoc. Theoret. Comp. Sci., October, 2000.

[3] Andreas Blass, Yuri Gurevich, and Dexter Kozen, *A zero-one law for logic with a fixed-point operator*, Information and Control 67 (1985) 70–90.

[4] Andreas Blass, Yuri Gurevich, and Saharon Shelah, *Choiceless polynomial time*, Ann. Pure Applied Logic 100 (1999) 141–187.

[5] Andreas Blass and Frank Harary, *Properties of almost all graphs and complexes*, J. Graph Theory 3 (1979) 225–240.

[6] Béla Bollobás, Random graphs, Academic Press, 1985.

[7] H. Chernoff, *A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations*, Ann. Math. Statist. 23 (1952) 493–507.

[8] Heinz-Dieter Ebbinghaus and Jörg Flum, Finite Model Theory, Springer-Verlag (1995).

[9] Ron Fagin, *Probabilities on finite models*, J. Symbolic Logic 41 (1976) 50–58.

[10] Y. V. Glebskii, D. I. Kogan, M. I. Liogonkii, and V. A. Talanov, *Range and degree of realizability of formulas in the restricted predicate calculus*, Cybernetics 5 (1969) 142–154.

[11] Yuri Gurevich, *Evolving algebras 1993: Lipari guide*, in Specification and Validation Methods, ed. E. Börger, Oxford University Press (1995) pp. 9–36. See also the *May 1997 draft of the ASM guide*, Tech Report CSE-TR-336-97, EECS Dept., Univ. of Michigan, 1997. Found at `http://www.eecs.umich.edu/gasm/`.

[12] Phokion Kolaitis and Moshe Vardi, *0-1 laws for infinitary logics*, Proc. 5th IEEE Symposium on Logic in Computer Science (1990) 156–167.

[13] Phokion Kolaitis and Moshe Vardi, *0–1 laws and decision problems for fragments of second-order logic*, Information and Computation 87 (special issue for the 3rd IEEE Symp. on Logic in Computer Science, 1988) (1990) 302–339.

[14] M. Loève, Probability Theory, Van Nostrand 1955.

[15] Saharon Shelah, *Choiceless polynomial time logic: inability to express* [paper number 634], in Computer Science Logic 2000, editors Peter Clote and Helmut Schwichtenberg, Springer Lecture Notes in Computer Science 1862 (2000) 72–125.

[16] V. A. Talanov and V. V. Knyazev, *The asymptotic truth of infinite formulas*, Proceedings of All-Union Seminar on Discrete Mathematics and Its Applications (Moscow 1984) (1986) 56–61.