The Baltic International Yearbook of
Cognition, Logic and Communication

November 2013 — Volume 8: *Games, Game Theory and Game Semantics*

pages 1-186 — DOI: 10.4148/1944-3676.1074

GIORGI JAPARIDZE
School of Computer Science and Technology, Shandong University
Department of Computing Sciences, Villanova University

PTARITHMETIC

ABSTRACT: The present article introduces *ptarithmetic* (short for "polynomial time arithmetic") — a formal number theory similar to the well known Peano arithmetic, but based on the recently born *computability logic* instead of classical logic. The formulas of ptarithmetic represent interactive computational problems rather than just true/false statements, and their "truth" is understood as existence of a polynomial time solution. The system of ptarithmetic elaborated in this article is shown to be sound and complete. Sound in the sense that every theorem T of the system represents an interactive number-theoretic computational problem with a polynomial time solution and, furthermore, such a solution can be effectively extracted from a proof of T. And complete in the sense that every interactive number-theoretic problem with a polynomial time solution is represented by some theorem T of the system.

The paper is self-contained, and can be read without any prior familiarity with computability logic.

This paper was completed in early 2009 (cf. http://arxiv.org/abs/0902.2969), and has remained unpublished since then. Among the reasons is that an alternative — in a sense more general and elegant — approach named "clarithmetic" was introduced

and developed (Japaridze 2011, unpublished, forth.). In retrospect, however, the author finds that "Ptarithmetic" contained a number of potentially useful ideas that have not been subsequently adopted by the "clarithmetics" line of research (at least not yet), and that, for this reason, it would be a pity to let this material remain unpublished. Probably the most important of such ideas is the "Polynomial Time Induction" (PTI) rule of Ptarithmetic, with no close or distant relatives elsewhere in the literature.

What follows is a copy of the original manuscript without any modifications, except some minor changes made in 2010 before shelving the paper. Some parts of the above manuscript were used in Japaridze (2011), which explains occasional textual overlaps between the two articles.

1. INTRODUCTION

Computability logic (CL), introduced in Japaridze (2003, 2006b, 2009a), is a semantical, mathematical and philosophical platform, and an ambitious program, for redeveloping logic as a formal theory of computability, as opposed to the formal theory of truth which logic has more traditionally been. Under the approach of CL, formulas represent computational problems, and their "truth" is seen as algorithmic solvability. In turn, computational problems — understood in their most general, *interactive* sense — are defined as games played by a machine against its environment, with "algorithmic solvability" meaning existence of a machine that wins the game against any possible behavior of the environment. And an open-ended collection of the most basic and natural operations on computational problems forms the logical vocabulary of the theory. With this semantics, CL provides a systematic answer to the fundamental question "*what can be computed?*", just as classical logic is a systematic tool for telling what is true. Furthermore, as it turns out, in positive cases "*what* can be computed" always allows itself to be replaced by "*how* can be computed", which makes CL of potential interest in not only theoretical computer science, but many more applied areas as well, including interactive knowledge base systems, resource oriented systems for planning and action, or declarative programming languages.

While potential applications have been repeatedly pointed out and outlined in introductory papers on CL, so far all technical efforts have been mainly focused on finding axiomatizations for various fragments of this semantically conceived and inordinately expressive logic. Considerable advances have already been made in this direction (Japaridze 2006d,a,c, 2009b, 2010; Mezhirov & Vereshchagin 2010), and more results in the same style are probably still to come. It should be, however, remembered that the main value of CL, or anything else claiming to be a "Logic" with a capital "L", will eventually be determined by whether and how it relates to the outside, to the extra-logical world. In this respect, unlike many other systems officially qualified as "logics", the merits of classical logic are obvious; these merits are most eloquently demonstrated by the fact that applied formal theories, a model example of which is *Peano arithmetic* **PA**, can be and have been successfully based on it. Unlike pure logics with their meaningless symbols, such theories are direct tools for studying and navigating the real world with its non-man-made, meaningful objects, such as natural numbers in the case of arithmetic. To make this point more clear to a computer scientist, one could compare a pure logic with a programming language, and applied theories based on it with application programs written in that language. A programming language created for its own sake, mathematically or aesthetically appealing but otherwise unusable as a general-purpose, comprehensive basis for application programs, would hardly be of much interest.

So, in parallel with studying possible axiomatizations and various metaproperties of pure computability logic, it would certainly be worthwhile to devote some efforts to justifying its right to existence through revealing its power and appeal as a basis for applied theories. The first and so far the only concrete steps in this direction have been made very recently in Japaridze (2010), where a CL-based system **CLA1** of (Peano) arithmetic was constructed.[1] Unlike its classical-logic-based counterpart **PA**, **CLA1** is not merely about which arithmetical facts are *true*, but about which arithmetical problems can be actually *computed* or effectively *solved*. More precisely, every formula of the language of **CLA1** expresses a number-theoretic computational *problem* (rather than just a true/false *fact*), every theorem expresses a problem that has an algorithmic solution, and every proof encodes such a solution. Does not this sound exactly like what the constructivists have been calling for?

Unlike the mathematical or philosophical constructivism, however, and even unlike the early-day theory of computation, modern computer science has long understood that, what really matters is not just *computability*, but rather *efficient computability*. So, the next natural step on the road to revealing the importance of CL for computer science would be showing that it can be used for studying efficient computability just as successfully as for studying computability-in-principle. Anyone familiar with the earlier work on CL could have found reasons for optimistic expectations here. Namely, every provable formula of any of the known sound axiomatizations of CL happens to be a scheme of not only "always computable" problems, but "always efficiently computable" problems as well, whatever efficiency exactly mens in the context of interactive computation that CL operates in. That is, at the level of pure logic, computability and efficient computability yield the same classes of valid principles. The study of logic abounds with phenomena in this style. One example would be the well known fact about classical logic, according to which validity with respect to all possible models is equivalent to validity with respect to just models with countable domains.

At the level of reasonably expressive applied theories, however, one should certainly expect significant differences depending on whether the underlying concept of interest is efficient computability or computability-in-principle. For instance, the earlier-mentioned system **CLA1** proves formulas expressing computable but not always efficiently computable arithmetical problems. The purpose of the present paper is to construct a CL-based system for arithmetic which, unlike **CLA1**, proves only efficiently — specifically, polynomial time — computable problems. The new applied formal theory **PTA** ("*ptarithmetic*", short for "polynomial time arithmetic") presented in Section 12 achieves this purpose.

Just like **CLA1**, our present system **PTA** is not only a cognitive, but also a problem-solving tool: in order to find a solution for a given problem, it would be sufficient to write the problem in the language of the system, and find a proof for it. An algorithmic solution for the problem would automatically come together with such a proof. However, unlike the solutions extracted from **CLA1**-proofs, which might be intractable, the solutions extracted from **PTA**-proofs would always be efficient.

Furthermore, **PTA** turns out to be not only sound, but also complete in a certain reasonable sense that we call *extensional completeness*. According to the latter, every number-theoretic computational problem that has a polynomial time solution is represented by some theorem of **PTA**. Taking into account that there are many ways to represent the same problem, extensional completeness is weaker than what can be called *intensional completeness*, according to which any formula representing an (efficiently) computable problem is provable. In these terms, Gödel's celebrated theorem, here with "truth"="computability", is about intensional rather than extensional incompleteness. In fact, extensional completeness is not at all interesting in the context of classical-logic-based theories such as **PA**. In such theories, unlike computability-logic-based theories, it is trivially achieved, as the provable formula $\top$ represents every true sentence.

Syntactically, our **PTA** is an extension of **PA**, and the semantics of the former is a conservative generalization of the semantics of the latter. Namely, the formulas of **PA**, which form only a proper subclass of the formulas of **PTA**, are seen as special, "moveless" sorts of problems/games, automatically solved/won when true and failed/lost when false. This makes the classical concept of truth just a special case of computability in our sense — it is nothing but computability restricted to (the problems represented by) the traditional sorts of formulas. And this means that Gödel's incompleteness theorems automatically extend from **PA** to **PTA**, so that, unlike extensional completeness, intensional completeness in **PTA** or any other sufficiently expressive CL-based applied theory is impossible to achieve in principle. As for **CLA1**, it turns out to be incomplete in both senses. Section 22 shows that any sufficiently expressive sound system would be (not only intensionally but also) extensionally incomplete, as long as the semantics of the system is based on unrestricted (as opposed to, say, efficient) computability.

Among the main moral merits of the present investigation and its contributions to the overall CL project is an illustration of the fact that, in constructing CL-based applied theories, successfully switching from computability to efficient computability is possible, and more than just

possible. As noted, efficient computability, in fact, turns out to be much better behaved than computability-in-principle: the former allows us to achieve completeness in a sense in which the latter yields inherent incompleteness.

An advanced reader will easily understand that the present paper, while focused on the system **PTA** of (pt)arithmetic, in fact is not only about arithmetic, but also just as much about CL-based applied theories or knowledge base systems in general, with **PTA** only serving as a model example of such systems and a starting point for what may be a separate (sub)line of research within the CL enterprise. Generally, the nonlogical axioms or the knowledge base of a CL-based applied system would be any collection of (formulas expressing) problems whose algorithmic or efficient solutions are known. Sometimes, together with nonlogical axioms, we may also have nonlogical rules of inference, preserving the property of computability or efficient computability. An example of such a rule is the *polynomial time induction* (PTI) rule of **PTA**. Then, the soundness of the corresponding underlying axiomatization of CL (in our present case, it is system **CL3** studied in Japaridze (2006c)) — which usually comes in the strong form called *uniform-constructive soundness* — guarantees that every theorem T of the theory also has an effective or efficient solution and that, furthermore, such a solution can be effectively extracted from a proof of T. It is this fact that, as mentioned, makes CL-based systems problem-solving tools.

Having said the above, motivationally (re)introducing and (re)justifying computability logic is not among the goals of the present paper. This job has been done in Japaridze (2003, 2006b, 2009a), and the reader would benefit from becoming familiar with any of those pieces of literature first, of which most recommended is the first 10 tutorial-style sections of Japaridze (2009a). While helpful in fully appreciating the import of the present results from the purely technical point of view, such familiarity is not necessary, as this paper provides all relevant definitions.

2. AN INFORMAL OVERVIEW OF THE MAIN OPERATIONS ON GAMES

As noted, formulas in CL represent computational problems. Such problems are understood as games between two players: $\top$, called

machine, and $\bot$, called **environment**. $\top$ is a mechanical device with a fully determined, algorithmic behavior. On the other hand, there are no restrictions on the behavior of $\bot$. A given machine is considered to be *solving* a given problem iff it wins the corresponding game no matter how the environment acts.

Standard atomic sentences, such as "$0=0$" or "Peggy is John's mother", are understood as special sorts of games, called **elementary**. There are no moves in elementary games, and they are automatically won or lost. Specifically, the elementary game represented by a true sentence is won (without making any moves) by the machine, and the elementary game represented by a false sentence is won by the environment.

Logical operators are understood as operations on games/problems. One of the important groups of such operations, called **choice operations**, comprises $\sqcap, \sqcup, \sqcap, \sqcup$. These are called **choice conjunction**, **choice disjunction**, **choice universal quantifier** and **choice existential quantifier**, respectively. $A_0 \sqcap A_1$ is a game where the first legal move ("choice"), which should be either 0 or 1, is by $\bot$. After such a move/choice i is made, the play continues and the winner is determined according to the rules of A_i; if a choice is never made, $\bot$ loses. $A_0 \sqcup A_1$ is defined in a symmetric way with the roles of $\bot$ and $\top$ interchanged: here it is $\top$ who makes an initial choice and who loses if such a choice is not made. With the universe of discourse being $\{0, 1, 10, 11, 100, \ldots\}$ (natural numbers identified with their binary representations), the meanings of the quantifiers $\sqcap$ and $\sqcup$ can now be explained by

$$\sqcap x A(x) = A(0) \sqcap A(1) \sqcap A(10) \sqcap A(11) \sqcap A(100) \sqcap \ldots$$

and

$$\sqcup x A(x) = A(0) \sqcup A(1) \sqcup A(10) \sqcup A(11) \sqcup A(100) \sqcup \ldots.$$

So, for example,

$$\sqcap x \big(Prime(x) \sqcup Composite(x)\big)$$

is a game where the first move is by the environment. Such a move should consist in selecting a particular number n for x, intuitively amounting to asking whether n is prime or composite. This move

brings the game down to (in the sense that the game continues as)

$$Prime(n) \sqcup Composite(n).$$

Now the machine has to move, or else it loses. The move should consist in choosing one of the two disjuncts. Let us say the left disjunct is chosen, which further brings the game down to $Prime(n)$. The latter is an elementary game, and here the interaction ends. The machine wins iff it has chosen a true disjunct. The choice of the left disjunct by the machine thus amounts to claiming/answering that n is prime. Overall, as we see, $\sqcap x\big(Prime(x) \sqcup Composite(x)\big)$ represents the problem of deciding the primality question.[2]

Similarly,

$$\sqcap x \sqcap y \sqcup z (z = x \times y)$$

is the problem of computing the product of any two numbers. Here the first two moves are by the environment, which selects some particular $m = x$ and $n = y$, thus asking the machine to tell what the product of m and n is. The machine wins if and only if, in response, it selects a (the) number k for z such that $k = m \times n$.

The present paper replaces the above-described choice quantifiers $\sqcap$ and $\sqcup$ with their *bounded* counterparts $\sqcap^{\mathfrak{b}}$ and $\sqcup^{\mathfrak{b}}$, where $\mathfrak{b}$ is a variable. These are the same as $\sqcap$ and $\sqcup$, with the difference that the choice here is limited only to the objects of the universe of discourse whose sizes do not exceed a certain bound, which is represented by the variable $\mathfrak{b}$. So, $\sqcap^{\mathfrak{b}} x A(x)$ is essentially the same as $\sqcap x\big(|x| \leq \mathfrak{b} \rightarrow A(x)\big)$ and $\sqcup^{\mathfrak{b}} x A(x)$ is essentially the same as $\sqcup x\big(|x| \leq \mathfrak{b} \wedge A(x)\big)$, where (the meanings of $\rightarrow, \wedge$ will be explained shortly and) $|x| \leq \mathfrak{b}$ means "the size of x does not exceed $\mathfrak{b}$". As we are going to see later, it is exactly the value of $\mathfrak{b}$ with respect to which the computational complexity of games will be measured.

Another group of game operations dealt with in this paper, two of which have already been used in the previous paragraph, comprises $\neg, \wedge, \vee, \rightarrow$. Employing the classical symbols for these operations is no accident, as they are conservative generalizations of the corresponding Boolean operations from elementary games to all games.

Negation $\neg$ is a role-switch operation: it turns $\top$'s moves and wins into $\bot$'s moves and wins, and vice versa. Since elementary games have no moves, only the winners are switched there, so that, as noted, $\neg$

acts just as the ordinary classical negation. For instance, as $\top$ is the winner in $0{+}1{=}1$, the winner in $\neg 0{+}1{=}1$ will be $\bot$. That is, $\top$ wins the negation $\neg A$ of an elementary game A iff it loses A, i.e., if A is false. As for the meaning of negation when applied to nonelementary games, at this point it may be useful to observe that $\neg$ interacts with choice operations in the kind old DeMorgan fashion. For example, it would not be hard to see that

$$\neg\sqcap x\sqcap y\sqcup z(z{=}x{\times}y) \;=\; \sqcup x\sqcup y\sqcap z(z{\neq}x{\times}y).$$

The operations $\wedge$ and $\vee$ are called **parallel conjunction** and **parallel disjunction**, respectively. Playing $A_0\wedge A_1$ (resp. $A_0\vee A_1$) means playing the two games in parallel where, in order to win, $\top$ needs to win in both (resp. at least one) of the components A_i. It is obvious that, just as in the case of negation, $\wedge$ and $\vee$ act as classical conjunction and disjunction when applied to elementary games. For instance, $0{+}1{=}1\vee 0{\times}1{=}1$ is a game automatically won by the machine. There are no moves in it as there are no moves in either disjunct, and the machine is an automatic winner because it is so in the left disjunct. To appreciate the difference between the two — choice and parallel — groups of connectives, compare

$$\sqcap x\big(Prime(x)\sqcup\neg Prime(x)\big)$$

and

$$\sqcap x\big(Prime(x)\vee\neg Prime(x)\big).$$

The former is a computationally nontrivial problem, existence of an easy (polynomial time) solution for which had remained an open question until a few years ago. As for the latter, it is trivial, as the machine has nothing to do in it: the first (and only) move is by the environment, consisting in choosing a number n for x. Whatever n is chosen, the machine wins, as $Prime(n)\vee\neg Prime(n)$ is a true sentence and hence an automatically $\top$-won elementary game.

The operation $\rightarrow$, called **reduction**, is defined by $A\rightarrow B=(\neg A)\vee B$. Intuitively, this is indeed the problem of *reducing* B to A: solving $A\rightarrow B$ means solving B while having A as an external *computational resource*. Resources are symmetric to problems: what is a problem to solve for one player is a resource that the other player can use, and vice versa.

Since A is negated in $(\neg A)\vee B$ and negation means switching the roles, A appears as a resource rather than problem for $\top$ in $A\rightarrow B$.

Consider $\sqcap x\sqcup y(y{=}x^2)$. Anyone who knows the definition of x^2 in terms of $\times$ (but perhaps does not know the meaning of multiplication, or is unable to compute this function for whatever reason) would be able to solve the problem

$$\sqcap z\sqcap u\sqcup v(v{=}z{\times}u)\;\rightarrow\;\sqcap x\sqcup y(y{=}x^2), \tag{1}$$

i.e., the problem

$$\sqcup z\sqcup u\sqcap v(v{\neq}z{\times}u)\;\vee\;\sqcap x\sqcup y(y{=}x^2),$$

as it is about reducing the consequent to the antecedent. A solution here goes like this. Wait till the environment specifies a value n for x, i.e. asks "what is the square of n?". Do not try to immediately answer this question, but rather specify the same value n for both z and u, thus asking the counterquestion: "what is n times n?". The environment will have to provide a correct answer m to this counterquestion (i.e., specify v as m where $m=n{\times}n$), or else it loses. Then, specify y as m, and rest your case. Note that, in this solution, the machine did not have to compute multiplication, doing which had become the environment's responsibility. The machine only correctly reduced the problem of computing square to the problem of computing product, which made it the winner.

Another group of operations that play an important role in CL comprises $\forall$ and its dual $\exists$ (with $\exists xA(x)=\neg\forall x\neg A(x)$), called **blind universal quantifier** and **blind existential quantifier**, respectively. $\forall xA(x)$ can be thought of as a "version" of $\sqcap xA(x)$ where the particular value of x that the environment selects is invisible to the machine, so that it has to play blindly in a way that guarantees success no matter what that value is.

Compare the problems

$$\sqcap x\big(Even(x)\sqcup Odd(x)\big)$$

and

$$\forall x\big(Even(x)\sqcup Odd(x)\big).$$

Both of them are about telling whether a given number is even or odd; the difference is only in whether that "given number" is known to the machine or not. The first problem is an easy-to-win, two-move-deep game of a structure that we have already seen. The second game, on the other hand, is one-move deep with only the machine to make a move — select the "true" disjunct, which is hardly possible to do as the value of x remains unspecified.

Just like all other operations for which we use classical symbols, the meanings of $\forall$ and $\exists$ are exactly classical when applied to elementary games. Having this full collection of classical operations makes computability logic a generalization and conservative extension of classical logic.

Going back to an earlier example, even though (1) expresses a "very easily solvable" problem, that formula is still not logically valid. Note that the successfulness of the reduction strategy of the consequent to the antecedent that we provided for it relies on the nonlogical fact that $x^2 = x \times x$. That strategy would fail in a general case where the meanings of x^2 and $x \times x$ may not necessarily be the same. On the other hand, the goal of CL as a general-purpose problem-solving tool should be to allow us find purely logical solutions, i.e., solutions that do not require any special, domain-specific knowledge and (thus) would be good no matter what the particular predicate or function symbols of the formulas mean. Any knowledge that might be relevant should be explicitly stated and included either in the antecedent of a given formula or in the set of axioms ("implicit antecedents" for every potential formula) of a CL-based theory. In our present case, formula (1) easily turns into a logically valid one by adding, to its antecedent, the definition of square in terms of multiplication:

$$\forall w(w^2 = w \times w) \wedge \sqcap z \sqcap u \sqcup v(v = z \times u) \ \rightarrow\ \sqcap x \sqcup y(y = x^2). \tag{2}$$

The strategy that we provided earlier for (1) is just as good for (2), with the difference that it is successful for (2) no matter what x^2 and $z \times u$ mean, whereas, in the case of (1), it was guaranteed to be successful only under the standard arithmetic interpretations of the square and product functions. Thus, our strategy for (2) is, in fact, a "purely logical" solution. Again, among the purposes of computability logic is to serve as a tool for finding such "purely logical" solutions, so that it can be applied to any domain of study rather than specific domains such as that of arithmetic, and to arbitrary meanings of nonlogical symbols rather than particular meanings such as that of the multiplication function for the symbol $\times$.

The above examples should not suggest that blind quantifiers are meaningful or useful only when applied to elementary problems. The following is an example of an effectively winnable nonelementary $\forall$-game:

$$\forall y \Big(Even(y) \sqcup Odd(y) \ \rightarrow\ \sqcap x \big(Even(x+y) \sqcup Odd(x+y) \big) \Big). \tag{3}$$

Solving this problem, which means reducing the consequent to the antecedent without knowing the value of y, is easy: $\top$ waits till $\bot$ selects a value n for x, and also tells — by selecting a disjunct in the antecedent — whether y is even or odd. Then, if n and y are both even or both odd, $\top$ chooses the first $\sqcup$-disjunct in the consequent, otherwise it chooses the second $\sqcup$-disjunct. Replacing the $\forall y$ prefix by $\sqcap y$ would significantly weaken the problem, obligating the environment to specify a value for y. Our strategy does not really need to know the exact value of y, as it only exploits the information about y's being even or odd, provided by the antecedent of the formula.

Many more — natural, meaningful and useful — operations beyond the ones discussed in this section have been introduced and studied in computability logic. Here we have only surveyed those that are relevant to our present investigation.

3. CONSTANT GAMES

Now we are getting down to formal definitions of the concepts informally explained in the previous section.

To define games formally, we need some technical terms and conventions. Let us agree that by a **move** we mean any finite string over the standard keyboard alphabet. A **labeled move** (**labmove**) is a move prefixed with $\top$ or $\bot$, with such a prefix (**label**) indicating which player has made the move. A **run** is a (finite or infinite) sequence of labmoves, and a **position** is a finite run.

Convention 3.1 We will be exclusively using the letters Γ, Δ, Φ for runs, and α, β for moves. The letter $\wp$ will always be a variable for players, and

$$\overline{\wp}$$

will mean "$\wp$'s adversary" ("the other player"). Runs will be often delimited by "$\langle$" and "$\rangle$", with $\langle\rangle$ thus denoting the **empty run**. The meaning of an expression such as $\langle\Phi,\wp\alpha,\Gamma\rangle$ must be clear: this is the result of appending to the position $\langle\Phi\rangle$ the labmove $\langle\wp\alpha\rangle$ and then the run $\langle\Gamma\rangle$.

The following is a formal definition of what we call constant games, combined with some less formal conventions regarding the usage of certain terminology.

Definition 3.2 A **constant game** is a pair $A=(\mathbf{Lr}^A,\mathbf{Wn}^A)$, where:

1. $\mathbf{Lr}^A$ is a set of runs satisfying the condition that a (finite or infinite) run is in $\mathbf{Lr}^A$ iff all of its nonempty finite initial segments are in $\mathbf{Lr}^A$ (notice that this implies $\langle\rangle\in\mathbf{Lr}^A$). The elements of $\mathbf{Lr}^A$ are said to be **legal runs** of A, and all other runs are said to be **illegal**. We say that α is a **legal move** for $\wp$ in a position Φ of A iff $\langle\Phi,\wp\alpha\rangle\in\mathbf{Lr}^A$; otherwise α is **illegal**. When the last move of the shortest illegal initial segment of Γ is $\wp$-labeled, we say that Γ is a **$\wp$-illegal** run of A.

2. $\mathbf{Wn}^A$ is a function that sends every run Γ to one of the players $\top$ or $\bot$, satisfying the condition that if Γ is a $\wp$-illegal run of A, then $\mathbf{Wn}^A\langle\Gamma\rangle=\overline{\wp}$. When $\mathbf{Wn}^A\langle\Gamma\rangle=\wp$, we say that Γ is a **$\wp$-won** (or **won by** $\wp$) run of A; otherwise Γ is **lost** by $\wp$. Thus, an illegal run is always lost by the player who has made the first illegal move in it.

An important operation not explicitly mentioned in Section 2 is what is called *prefixation*. This operation takes two arguments: a constant game A and a position Φ that must be a legal position of A (otherwise the operation is undefined), and returns the game $\langle\Phi\rangle A$. Intuitively, $\langle\Phi\rangle A$ is the game playing which means playing A starting (continuing) from position Φ. That is, $\langle\Phi\rangle A$ is the game to which A **evolves** (will be "**brought down**") after the moves of Φ have been made. We have already used this intuition when explaining the meaning of choice operations in Section 2: we said that after $\bot$ makes an initial move $i\in\{0,1\}$, the game $A_0\sqcap A_1$ continues as A_i. What this meant was nothing but that $\langle\bot i\rangle(A_0\sqcap A_1)=A_i$. Similarly, $\langle\top i\rangle(A_0\sqcup A_1)=A_i$. Here is a definition of prefixation:

Definition 3.3 Let A be a constant game and Φ a legal position of A. The game $\langle\Phi\rangle A$ is defined by:

- $\mathbf{Lr}^{\langle\breve{}\rangle A}=\{\Gamma\mid\langle\Phi,\Gamma\rangle\in\mathbf{Lr}^A\}$;
- $\mathbf{Wn}^{\langle\breve{}\rangle A}\langle\Gamma\rangle=\mathbf{Wn}^A\langle\Phi,\Gamma\rangle$.

Convention 3.4 A terminological convention important to remember is that we often identify a legal position Φ of a game A with the game $\langle\Phi\rangle A$. So, for instance, we may say that the move 1 by $\bot$ brings the game $B_0\sqcap B_1$ down to the position B_1. Strictly speaking, B_1 is not a position but a game, and what *is* a position is $\langle\bot 1\rangle$, which we here identified with the game $B_1=\langle\bot 1\rangle(B_0\sqcap B_1)$.

We say that a constant game A is **finite-depth** iff there is an integer d such that no legal run of A contains more than d labmoves. The smallest of such integers d is called the **depth** of A. An **elementary game** is a game of depth 0.

In this paper I will exclusively deal with finite-depth games. This restriction of focus makes many definitions and proofs simpler. Namely, in order to define a finite-depth-preserving game operation $O(A_1,\ldots,A_n)$ applied to such games, it suffices to specify the following:

(i) Who wins $O(A_1,\ldots,A_n)$ if no moves are made, i.e., the value of $\mathbf{Wn}^{O(A_1,\ldots,A_n)}\langle\rangle$.

(ii) What are the **initial legal (lab)moves**, i.e., the elements of $\{\wp\alpha\mid\langle\wp\alpha\rangle\in\mathbf{Lr}^{O(A_1,\ldots,A_n)}\}$, and to what game is $O(A_1,\ldots,A_n)$ brought down after such an initial legal labmove $\wp\alpha$ is made. Recall that, by saying that a given labmove $\wp\alpha$ brings a given game A down to B, we mean that $\langle\wp\alpha\rangle A=B$.

Then, the set of legal runs of $O(A_1,\ldots,A_n)$ will be uniquely defined, and so will be the winner in every legal (and hence finite) run of the game.

Below we define a number of operations for finite-depth games only. Each of these operations can be easily seen to preserve the finite-depth property. Of course, more general definitions of these operations — not restricted to finite-depth games — do exist (see, e.g., Japaridze (2009a)), but in this paper we are trying to keep things as simple as possible, and reintroduce only as much of computability logic as necessary.

Definition 3.5 Let A, B, $A_0, A_1, \ldots$ be finite-depth constant games, and n be a positive integer.

1. $\neg A$ is defined by:

   (i) $\mathbf{Wn}^{\neg A}\langle\rangle = \wp$ iff $\mathbf{Wn}^{A}\langle\rangle = \overline{\wp}$.

   (ii) $\langle\wp\alpha\rangle \in \mathbf{Lr}^{\neg A}$ iff $\langle\overline{\wp}\alpha\rangle \in \mathbf{Lr}^{A}$. Such an initial legal labmove $\wp\alpha$ brings the game down to $\neg\langle\overline{\wp}\alpha\rangle A$.

2. $A_0 \sqcap \ldots \sqcap A_n$ is defined by:

   (i) $\mathbf{Wn}^{A_0 \sqcap \ldots \sqcap A_n}\langle\rangle = \top$.

   (ii) $\langle\wp\alpha\rangle \in \mathbf{Lr}^{A_0 \sqcap \ldots \sqcap A_n}$ iff $\wp = \bot$ and $\alpha = i \in \{0, \ldots, n\}$.[3] Such an initial legal labmove $\bot i$ brings the game down to A_i.

3. $A_0 \wedge \ldots \wedge A_n$ is defined by:

   (i) $\mathbf{Wn}^{A_0 \wedge \ldots \wedge A_n}\langle\rangle = \top$ iff, for each $i \in \{0, \ldots, n\}$, $\mathbf{Wn}^{A_i}\langle\rangle = \top$.

   (ii) $\langle\wp\alpha\rangle \in \mathbf{Lr}^{A_0 \wedge \ldots \wedge A_n}$ iff $\alpha = i.\beta$, where $i \in \{0, \ldots, n\}$ and $\langle\wp\beta\rangle \in \mathbf{Lr}^{A_i}$. Such an initial legal labmove $\wp i.\beta$ brings the game down to

$$A_0 \wedge \ldots \wedge A_{i-1} \wedge \langle\wp\beta\rangle A_i \wedge A_{i+1} \wedge \ldots \wedge A_n.$$

4. $A_0 \sqcup \ldots \sqcup A_n$ and $A_0 \vee \ldots \vee A_n$ are defined exactly as $A_0 \sqcap \ldots \sqcap A_n$ and $A_0 \wedge \ldots \wedge A_n$, respectively, only with "$\top$" and "$\bot$" interchanged.

5. In addition to the earlier-established meanings, the symbols $\top$ and $\bot$ also denote two special — simplest — constant games, defined by $\mathbf{Wn}^{\top}\langle\rangle = \top$, $\mathbf{Wn}^{\bot}\langle\rangle = \bot$ and $\mathbf{Lr}^{\top} = \mathbf{Lr}^{\bot} = \{\langle\rangle\}$.

6. $A \rightarrow B$ is treated as an abbreviation of $(\neg A) \vee B$.

Example 3.6 The game $(0{=}0 \sqcap 0{=}1) \rightarrow (10{=}11 \sqcap 10{=}10)$, i.e.

$$\neg(0{=}0 \sqcap 0{=}1) \vee (10{=}11 \sqcap 10{=}10),$$

has thirteen legal runs, which are:

1 $\langle\rangle$. It is won by $\top$, because $\top$ is the winner in the right $\vee$-disjunct (consequent).

2 $\langle\top 0.0\rangle$. (The labmove of) this run brings the game down to $\neg 0{=}0 \vee (10{=}11 \sqcap 10{=}10)$, and $\top$ is the winner for the same reason as in the previous case.

3 $\langle\top 0.1\rangle$. It brings the game down to $\neg 0{=}1 \vee (10{=}11 \sqcap 10{=}10)$, and $\top$ is the winner because it wins in both $\vee$-disjuncts.

4 $\langle\bot 1.0\rangle$. It brings the game down to $\neg(0{=}0 \sqcap 0{=}1) \vee 10{=}11$. $\top$ loses as it loses in both $\vee$-disjuncts.

5 $\langle\bot 1.1\rangle$. It brings the game down to $\neg(0{=}0 \sqcap 0{=}1) \vee 10{=}10$. $\top$ wins as it wins in the right $\vee$-disjunct.

6-7 $\langle\top 0.0, \bot 1.0\rangle$ and $\langle\bot 1.0, \top 0.0\rangle$. Both bring the game down to the false $\neg 0{=}0 \vee 10{=}11$, and both are lost by $\top$.

8-9 $\langle\top 0.1, \bot 1.0\rangle$ and $\langle\bot 1.0, \top 0.1\rangle$. Both bring the game down to the true $\neg 0{=}1 \vee 10{=}11$, which makes $\top$ the winner.

10-11 $\langle\top 0.0, \bot 1.1\rangle$ and $\langle\bot 1.1, \top 0.0\rangle$. Both bring the game down to the true $\neg 0{=}0 \vee 10{=}10$, so $\top$ wins.

12-13 $\langle\top 0.1, \bot 1.1\rangle$ and $\langle\bot 1.1, \top 0.1\rangle$. Both bring the game down to the true $\neg 0{=}1 \vee 10{=}10$, so $\top$ wins.

4. GAMES AS GENERALIZED PREDICATES

Constant games can be seen as generalized propositions: while propositions in classical logic are just elements of $\{\top, \bot\}$, constant games are functions from runs to $\{\top, \bot\}$. As we know, however, propositions only offer a very limited expressive power, and classical logic needs to consider the more general concept of predicates, with propositions being nothing but special — constant — cases of predicates. The situation in computability logic is similar. Our concept of a (simple) game generalizes that of a constant game in the same sense as the classical concept of a predicate generalizes that of a proposition.

We fix an infinite set of expressions called **variables**:

$$\{\mathfrak{w}_0, \mathfrak{w}_1, \mathfrak{w}_2, \mathfrak{w}_3, \ldots\}.$$

The letters

$$x, y, z, s, r, t, u, v, w$$

will be used as metavariables for these variables. The Gothic letter

$$\mathfrak{b}$$

will be exclusively used as a metaname for the variable $\mathfrak{w}_0$, which is going to have a special status throughout our entire treatment.

We also fix another infinite set of expressions called **constants**:

$$\{0, 1, 10, 11, 100, 101, 110, 111, 1000, \ldots\}.$$

These are thus **binary numerals** — the strings matching the regular expression $0 \cup 1(0 \cup 1)^*$. We will be typically identifying such strings — by some rather innocent abuse of concepts — with the natural numbers represented by them in the standard binary notation, and vice versa. The above collection of constants is going to be exactly the *universe of discourse* — i.e., the set over which the variables range — in all cases that we consider. We will be mostly using a, b, c, d as metavariables for constants.

By a **valuation** we mean a function e that sends each variable x to a constant $e(x)$. In these terms, a classical predicate p can be understood as a function that sends each valuation e to a proposition, i.e., to a constant predicate. Similarly, what we call a game sends valuations to constant games:

Definition 4.1 A **game** is a function A from valuations to constant games. We write $e[A]$ (rather than $A(e)$) to denote the constant game returned by A for valuation e. Such a constant game $e[A]$ is said to be an **instance** of A. For readability, we usually write $\mathbf{Lr}^A_e$ and $\mathbf{Wn}^A_e$ instead of $\mathbf{Lr}^{e[A]}$ and $\mathbf{Wn}^{e[A]}$.

Just as this is the case with propositions versus predicates, constant games in the sense of Definition 3.2 will be thought of as special, constant cases of games in the sense of Definition 4.1. In particular, each constant game A' is the game A such that, for every valuation e, $e[A] = A'$. From now on we will no longer distinguish between such A and A', so that, if A is a constant game, it is its own instance, with $A = e[A]$ for every e.

Where n is a natural number, we say that a game A is n**-ary** iff there is are n variables such that, for any two valuations e_1 and e_2 that agree on all those variables, we have $e_1[A] = e_2[A]$. Generally, a game that is n-ary for some n, is said to be *finitary*. Our paper is going to exclusively deal with finitary games and, for this reason, we agree that, from now on, when we say "game", we usually mean "finitary game".

We say that a game A **depends** on a variable x iff there are two valuations e_1, e_2 that agree on all variables except x such that $e_1[A] \neq e_2[A]$. An n-ary game thus depends on at most n variables. And constant games are nothing but 0-ary games, i.e., games that do not depend on any variables.

We say that a (not necessarily constant) game A is **elementary** iff so are all of its instances $e[A]$. And we say that A is **finite-depth** iff there is a (smallest) integer d, called the **depth** of A, such that the depth of no instance of A exceeds d.

Just as constant games are generalized propositions, games can be treated as generalized predicates. Namely, we will see each predicate p of whatever arity as the same-arity elementary game such that, for every valuation e, $\mathbf{Wn}^p_e\langle\rangle = \top$ iff p is true at e. And vice versa: every elementary game p will be seen as the same-arity predicate which is true at a given valuation e iff $\mathbf{Wn}^p_e\langle\rangle = \top$. Thus, for us, "predicate" and "elementary game" are going to be synonyms. Accordingly, any standard terminological or notational conventions familiar from the literature for predicates also apply to them seen as elementary games.

Just as the Boolean operations straightforwardly extend from propositions to all predicates, our operations $\neg, \wedge, \vee, \rightarrow, \sqcap, \sqcup$ extend from constant games to all games. This is done by simply stipulating that $e[\ldots]$ commutes with all of those operations: $\neg A$ is the game such that, for every valuation e, $e[\neg A] = \neg e[A]$; $A \sqcap B$ is the game such that, for every e, $e[A \sqcap B] = e[A] \sqcap e[B]$; etc.

The operation of prefixation also extends to nonconstant games: $\langle\Phi\rangle A$ should be understood as the unique game such that, for every e, $e[\langle\Phi\rangle A] = \langle\Phi\rangle e[A]$. However, unlike the cases with all other operations, $\langle\Phi\rangle A$, as a function from valuations to constant games, may be partial even if A is total. Namely, it will be defined only for those valuations e for which we have $\Phi \in \mathbf{Lr}^A_e$. Let us call not-always-defined "games" **partial** (as opposed to the **total** games of Definition 4.1). In the rare cases when we write $\langle\Phi\rangle A$ for a non-constant game A (which always happens in just intermediate steps), it should be remembered that it is possible we are dealing with a partial rather than a total game. Otherwise, the default meaning of the word "game" is always a total game.

Definition 4.2 Let A be a game, $x_1,\ldots,x_n$ be pairwise distinct variables, and $c_1,\ldots,c_n$ be constants. The result of **substituting** $x_1,\ldots,x_n$ **by** $c_1,\ldots,c_n$ **in** A, denoted $A(x_1/c_1,\ldots,x_n/c_n)$, is defined by stipulating that, for every valuation e, $e[A(x_1/c_1,\ldots,x_n/c_n)] = e'[A]$, where e' is the valuation that sends each x_i to c_i and agrees with e on all other variables.

Following the standard readability-improving practice established in the literature for predicates, we will often fix pairwise distinct variables $x_1,\ldots,x_n$ for a game A and write A as $A(x_1,\ldots,x_n)$. Representing A in this form sets a context in which we can write $A(c_1,\ldots,c_n)$ to mean the same as the more clumsy expression $A(x_1/c_1,\ldots,x_n/c_n)$.

Definition 4.3 Below x is an arbitrary variable other than $\mathfrak{b}$, and $A(x)$ is an arbitrary finite-depth game.

1. We define $\sqcap^0 xA(x) = \sqcup^0 xA(x) = A(0)$ and, for any positive integer b, with 1^b standing for the binary numeral consisting of b "1"s, we define the games $\sqcap^b xA(x)$ and $\sqcup^b xA(x)$ as follows:

$$\sqcap^b xA(x) = A(0)\sqcap A(1)\sqcap A(10)\sqcap A(11)\sqcap A(100)\sqcap A(101)\sqcap \ldots \sqcap A(1^b);$$

$$\sqcup^b xA(x) = A(0)\sqcup A(1)\sqcup A(10)\sqcup A(11)\sqcup A(100)\sqcup A(101)\sqcup \ldots \sqcup A(1^b).$$

2. Using the above notation, we define

$$\sqcap^{\mathfrak{b}} xA(x)$$

as the unique game such that, for any valuation e, $e[\sqcap^{\mathfrak{b}} xA(x)] = e[\sqcap^b xA(x)]$, where $b = e(\mathfrak{b})$. Similarly,

$$\sqcup^{\mathfrak{b}} xA(x)$$

is the unique game such that, for any valuation e, $e[\sqcup^{\mathfrak{b}} xA(x)] = e[\sqcup^b xA(x)]$, where $b = e(\mathfrak{b})$. $\sqcap^{\mathfrak{b}}$ and $\sqcup^{\mathfrak{b}}$ are said to be **bounded choice universal quantifier** and **bounded choice existential quantifier**, respectively.

As we see, $\sqcap^{\mathfrak{b}}$ and $\sqcup^{\mathfrak{b}}$ are like the ordinary choice quantifiers $\sqcap,\sqcup$ of computability logic explained in Section 2, with the only difference that the size of a constant chosen for x in $\sqcap^{\mathfrak{b}}x$ or $\sqcup^{\mathfrak{b}}x$ should not exceed the value of $\mathfrak{b}$. (The case of that value being 0 is a minor technical exception which can be safely forgotten.)

Convention 4.4 Because throughout the rest of this paper we exclusively deal with the bounded choice quantifiers $\sqcap^{\mathfrak{b}},\sqcup^{\mathfrak{b}}$ (and never with the ordinary $\sqcap,\sqcup$ discussed in Section 2), and because the variable $\mathfrak{b}$ is fixed and is the same everywhere, we agree that, *from now on*, when we write $\sqcap$ or $\sqcup$, we *always* mean $\sqcap^{\mathfrak{b}}$ or $\sqcup^{\mathfrak{b}}$, respectively.

This is not a change of interpretation of $\sqcap,\sqcup$ but rather some, rather innocent, abuse of notation.

We will say that a game A is **unistructural** iff, for any two valuations e_1 and e_2 that agree on $\mathfrak{b}$, we have $\mathbf{Lr}^A_{e_1} = \mathbf{Lr}^A_{e_2}$. Of course, all constant or elementary games are unistructural. It can also be easily seen that all our game operations preserve the unistructural property of games. For the purposes of the present paper, considering only unistructural games would be sufficient.

We define the remaining operations $\forall$ and $\exists$ only for unistructural games:

Definition 4.5 Let x be a variable other than $\mathfrak{b}$, and $A(x)$ be a finite-depth unistructural game.

1. $\forall xA(x)$ is defined by stipulating that, for every valuation e, player $\wp$ and move α, we have:

   (i) $\mathbf{Wn}^{\forall xA(x)}_e\langle\rangle = \top$ iff, for every constant[4] c, $\mathbf{Wn}^{A(c)}_e\langle\rangle = \top$.

   (ii) $\langle\wp\alpha\rangle \in \mathbf{Lr}^{\forall xA(x)}_e$ iff $\langle\wp\alpha\rangle \in \mathbf{Lr}^{A(x)}_e$. Such an initial legal labmove $\wp\alpha$ brings the game $e[\forall xA(x)]$ down to $e[\forall x\langle\wp\alpha\rangle A(x)]$.

2. $\exists x A(x)$ is defined in exactly the same way, only with $\top$ and $\bot$ interchanged.

It is worth noting that $\forall x A(x)$ and $\exists x A(x)$ are total even if the game $\langle \wp \alpha \rangle A(x)$ used in their definition is only partial.

Example 4.6 Let G be the game (3) discussed earlier in Section 2 (only, now $\sqcap$ seen as $\sqcap^{\mathfrak{b}}$), and let e be a valuation with $e(\mathfrak{b}) = 10$. The sequence $\langle \bot 1.11,\ \bot 0.0,\ \top 1.1 \rangle$ is a legal run of $e[G]$, the effects of the moves of which are shown below:

$e[G]:$	$\forall y \Big(\mathit{Even}(y) \sqcup \mathit{Odd}(y) \rightarrow \sqcap^{10} x \big(\mathit{Even}(x+y) \sqcup \mathit{Odd}(x+y) \big) \Big)$
$\langle \bot 1.11 \rangle e[G]:$	$\forall y \big(\mathit{Even}(y) \sqcup \mathit{Odd}(y) \rightarrow \mathit{Even}(11+y) \sqcup \mathit{Odd}(11+y) \big)$
$\langle \bot 1.11, \bot 0.0 \rangle e[G]:$	$\forall y \big(\mathit{Even}(y) \rightarrow \mathit{Even}(11+y) \sqcup \mathit{Odd}(11+y) \big)$
$\langle \bot 1.11, \bot 0.0, \top 1.1 \rangle e[G]:$	$\forall y \big(\mathit{Even}(y) \rightarrow \mathit{Odd}(11+y) \big)$

The play hits (ends as) the true proposition $\forall y \big(\mathit{Even}(y) \rightarrow \mathit{Odd}(11+y) \big)$ and hence is won by $\top$.

Before closing this section, we want to make the rather straightforward observation that the DeMorgan dualities hold for all of our sorts of conjunctions, disjunctions and quantifiers, and so does the double negation principle. That is, we always have:

$$\neg\neg A = A;$$

$$\neg(A \wedge B) = \neg A \vee \neg B; \quad \neg(A \vee B) = \neg A \wedge \neg B;$$

$$\neg(A \sqcap B) = \neg A \sqcup \neg B; \quad \neg(A \sqcup B) = \neg A \sqcap \neg B;$$

$$\neg \forall x A(x) = \exists x \neg A(x); \quad \neg \exists x A(x) = \forall x \neg A(x);$$

$$\neg \sqcap x A(x) = \sqcup x \neg A(x); \quad \neg \sqcup x A(x) = \sqcap x \neg A(x).$$

5. ALGORITHMIC STRATEGIES THROUGH INTERACTIVE MACHINES

In traditional game-semantical approaches, including Blass's (1972; 1992) approach which is the closest precursor to ours, player's strategies are understood as *functions* — typically as functions from interaction histories (positions) to moves, or sometimes (Abramsky & Jagadeesan 1994) as functions that only look at the latest move of the history. This *strategies-as-functions* approach, however, is inapplicable in the context of computability logic, whose relaxed semantics, in striving to get rid of any "bureaucratic pollutants" and only deal with the remaining true essence of games, does not impose any regulations on which player can or should move in a given situation. Here, in many cases, either player may have (legal) moves, and then it is unclear whether the next move should be the one prescribed by $\top$'s strategy function or the one prescribed by the strategy function of $\bot$. In fact, for a game semantics whose ambition is to provide a comprehensive, natural and direct tool for modeling interaction, the strategies-as-functions approach would be simply less than adequate, even if technically possible. This is so for the simple reason that the strategies that real computers follow are not functions. If the strategy of your personal computer was a function from the history of interaction with you, then its performance would keep noticeably worsening due to the need to read the continuously lengthening — and, in fact, practically infinite — interaction history every time before responding. Fully ignoring that history and looking only at your latest keystroke in the spirit of Abramsky & Jagadeesan (1994) is also certainly not what your computer does, either.

In computability logic, ($\top$'s effective) strategies are defined in terms of interactive machines, where computation is one continuous process interspersed with — and influenced by — multiple "input" (environment's moves) and "output" (machine's moves) events. Of several, seemingly rather different yet equivalent, machine models of interactive computation studied in CL, here we will employ the most basic, **HPM** ("Hard-Play Machine") model.

An HPM is nothing but a Turing machine with the additional capability of making moves. The adversary can also move at any time, with such moves being the only nondeterministic events from the machine's perspective. Along with the ordinary work tape, the machine has two additional tapes called the valuation tape and the run tape. The valuation tape, serving as a static input, spells some (arbitrary but fixed) valuation applied to the game. And the run tape, serving as a dynamic input, at any time spells the "current position" of the play. The role of these two tapes is to make both the valuation and the run fully visible to the machine.

In these terms, an algorithmic solution ($\top$'s winning strategy) for a given game A is understood as an HPM $\mathcal{M}$ such that, no matter how the environment acts during its interaction with $\mathcal{M}$ (what moves it makes and when), and no matter what valuation e is spelled on the valuation tape, the run incrementally spelled on the run tape is a $\top$-won run of $e[A]$.

As for $\bot$'s strategies, there is no need to define them: all possible behaviors by $\bot$ are accounted for by the different possible nondeterministic updates of the run tape of an HPM.

In the above outline, we described HPMs in a relaxed fashion, without being specific about technical details such as, say, how, exactly, moves are made by the machine, how many moves either player can make at once, what happens if both players attempt to move "simultaneously", etc. As it turns out, all reasonable design choices yield the same class of winnable games as long as we consider a certain natural subclass of games called **static**. Such games are obtained by imposing a certain simple formal condition on games (see, e.g., Section 5 of Japaridze 2009a), which we do not reproduce here as nothing in this paper relies on it. We shall only point out that, intuitively, static games are interactive tasks where the relative speeds of the players are irrelevant, as it never hurts a player to postpone making moves. In other words, static games are games that are contests of intellect rather than contests of speed. And one of the theses that computability logic philosophically relies on is that static games present an adequate formal counterpart of our intuitive concept of "pure", speed-independent interactive computational problems. Correspondingly, computability logic restricts its attention (more specifically, possible interpretations of the atoms of its formal language) to static games. All elementary games turn out to be trivially static, and the class of static games turns out to be closed under all game operations studied in computability logic. More specifically, all games expressible in the language of the later-defined logic **CL3**, or theory **PTA**, are static. And, in this paper, we use the term "**computational problem**", or simply "**problem**", as a synonym of "static game".

6. THE HPM MODEL IN GREATER DETAIL

As noted, computability of static games is rather robust with respect to the technical details of the underlying model of interaction. And the loose description of HPMs that we gave in the previous section would be sufficient for most purposes, just as mankind had been rather comfortably studying and using algorithms long before the Church-Turing thesis in its precise form came around. Namely, relying on just the intuitive concept of algorithmic strategies (believed in CL to be adequately captured by the HPM model) would be sufficient if we only needed to show existence of such strategies for various games. As it happens, however, later sections of this paper need to arithmetize such strategies in order to prove the promised extensional completeness of ptarithmetic. The complexity-theoretic concepts defined in the next section also require certain more specific details about HPMs, and in this section we provide such details. It should be pointed out again that most — if not all — of such details are "negotiable", as different reasonable arrangements would yield equivalent models.

Just like an ordinary Turing machine, an HPM has a finite set of **states**, one of which has the special status of being the **start state**. There are no accept, reject, or halt states, but there are specially designated states called **move states**. It is assumed that the start state is not among the move states. As noted earlier, this is a three-tape machine, with a read-only **valuation tape**, read-write **work tape**, and read-only **run tape**. Each tape has a beginning but no end, and is divided into infinitely many **cells**, arranged in the left-to-right order. At any time, each cell will contain one symbol from a certain fixed finite set of **tape symbols**. The **blank** symbol, as well as $\top$ and $\bot$, are among the tape symbols. We also assume that these three symbols are not among the symbols that any (legal or illegal) move can ever contain. Each tape has its own **scanning head**, at any given time looking (located) at one of the cells of the tape. A transition from one **computation step** ("**clock cycle**") to another happens according to the fixed **transition function** of the machine. The latter, depending on the current state, and the symbols seen by the three heads on the corresponding tapes, deterministically prescribes the next state, the tape symbol by which the old symbol should be overwritten in the current cell (the cell currently scanned by the head) of the work tape, and, for each head, the direction — one cell left or one cell right — in which the head should move. A constraint here is that the blank symbol, $\top$ or $\bot$ can never be written by the machine on the work tape. An attempt to move left when the head of a given tape is looking at the first (leftmost) cell results in staying put. So does an attempt to move right when the head is looking at the blank symbol.

When the machine starts working, it is in its start state, all three scanning heads are looking at the first cells of the corresponding tapes, the valuation tape spells some valuation e by listing the values of the variables $\mathfrak{w}_0, \mathfrak{w}_1, \mathfrak{w}_2, \ldots$ (in this precise order) separated by commas, and (all cells of) the work and run tapes are blank (i.e., contain the blank symbol). Whenever the machine enters a move state, the string α spelled by (the contents of) its work tape cells, starting from the first cell and ending with the cell immediately left of the work-tape scanning head, will be automatically appended — at the beginning of the next clock cycle — to the contents of the run tape in the $\top$-prefixed form $\top\alpha$. And, on every transition, whether the machine is in a move state or not, any finite sequence $\bot\beta_1, \ldots, \bot\beta_m$ of $\bot$-labeled moves may be nondeterministically appended to the content of the run tape. If the above two events happen on the same clock cycle, then the moves will be appended to the contents of the run tape in the following order: $\top\alpha\bot\beta_1 \ldots \bot\beta_m$ (note the technicality that labmoves are listed on the run tape without blanks or commas between them).

With each labmove that emerges on the run tape we associate its **timestamp**, which is the number of the clock cycle immediately preceding the cycle on which the move first emerged on the run tape. Intuitively, the timestamp indicates on which cycle the move was *made* rather than *appeared* on the run tape; a move made during cycle $\#i$ appears on the run tape on cycle $\#i{+}1$ rather than $\#i$. Also, we agree that the count of clock cycles starts from 0, meaning that the very first clock cycle is cycle #0 rather than #1.

A **configuration** is a full description of (the "current") contents of the work and run tapes, the locations of the three scanning heads, and the state of the machine. An ***e*-computation branch** is an infinite sequence $C_0, C_1, C_2, \ldots$ of configurations, where C_0 is the initial configuration (as explained earlier), and every C_{i+1} is a configuration that could have legally followed (again, in the sense explained earlier)

C_i when the valuation e is spelled on the valuation tape. For an e-computation branch B, the **run spelled by** B is the run Γ incrementally spelled on the run tape in the corresponding scenario of interaction. We say that such a Γ is **a run generated by** the machine on valuation e.

We say that a given HPM $\mathcal{M}$ **wins** (**computes**, **solves**) a given game A on valuation e — and write $\mathcal{M} \models_e A$ — iff every run Γ generated by $\mathcal{M}$ on valuation e is a $\top$-won run of $e[A]$. We say that A is **computable** iff there is an HPM $\mathcal{M}$ such that, for every valuation e, $\mathcal{M} \models_e A$; such an HPM is said to be an (algorithmic) **solution**, or **winning strategy**, for A.

7. TOWARDS INTERACTIVE COMPLEXITY

At present, the theory of interactive computation is far from being well developed, and even less so is the theory of interactive complexity. The studies of interactive computation in the context of complexity, while going on for some time now, have been relatively scattered, and interaction has often been used for better understanding certain traditional, non-interactive complexity issues (examples would be alternating computation (Chandra et al. 1981), or interactive proof systems and Arthur-Merlin games (Goldwasser et al. 1989; Babai & Shlomo 1988)) rather than being treated as an object of systematic studies in its own rights. As if complexity theory was not "complex" enough already, taking it to the interactive level would most certainly generate, by an order of magnitude, greater diversity of species from the complexity zoo.

The present paper is the first modest attempt to bring complexity issues into computability logic and the corresponding part of the under-construction theory of interactive computation. Here we introduce one, perhaps the simplest, way of measuring interactive complexity out of the huge and interesting potential variety of complexity measures that are meaningful and useful in the interactive context.

Games happen to be so expressive that most, if not all, ways of measuring complexity will be meaningful and interesting only for certain (sub)classes of games and not quite so, or not so at all, for other classes. Our present approach is no exception. The time complexity concept that we are going to introduce is meaningfully applicable only to games that, in positive (winnable) cases, can be brought by $\top$ to a successful end within a finite number of moves. In addition, every instance of a game under consideration should be such that the length of any move in any legal run of it never exceeds a certain bound which only depends on the value of our special-status variable $\mathfrak{b}$. As mentioned earlier, it is exactly the value of this variable relative to which the computational complexity of games will be measured.

The above class of games includes all games obtained by closing elementary games (predicates) under the operations of Sections 3 and 4, which also happens to be the class of games expressible in the language of the later-defined logic **CL3**. Indeed, consider any such game A. Obviously the number of moves in any legal run — and hence any $\top$-won run — of any instance of A cannot exceed its $(\sqcap,\sqcup,\sqcap\!\!\!\sqcap,\sqcup\!\!\!\sqcup)$-depth; the sizes of moves associated with $\sqcap,\sqcup$ are constant; and the sizes of moves associated with $\sqcap\!\!\!\sqcap,\sqcup\!\!\!\sqcup$, in any given instance of the game, never exceed a certain constant plus the value of the variable $\mathfrak{b}$.

Games for which our present complexity concepts are meaningful also include the much wider class of games expressible in the language of logic **CL12** introduced in Japaridze (2010), if the quantifiers $\sqcap\!\!\!\sqcap,\sqcup\!\!\!\sqcup$ of the latter are understood (as they are in this paper) as their bounded counterparts $\sqcap\!\!\!\sqcap^{\mathfrak{b}},\sqcup\!\!\!\sqcup^{\mathfrak{b}}$. While those games may have arbitrarily long or even infinite legal runs, all runs won by $\top$ are still finite.

Bringing computability logic to a complexity-sensitive level also naturally calls for considering only **bounded valuations**. By a bounded valuation we mean a valuation e such that, for any variable x, the size of the binary numeral $e(x)$ does not exceed the value $e(\mathfrak{b})$ of $\mathfrak{b}$ (note: the *value* of $\mathfrak{b}$ rather than the *size* of that value). This condition makes it possible to treat free variables in the same way as if they were $\sqcap\!\!\!\sqcap$-bounded.

The starting philosophical-motivational point of our present approach to time complexity is that it should be an indicator of "how soon the game(s) can be won" in the worst case, with "how soon" referring to the number of computation steps (clock cycles) a given HPM $\mathcal{M}$ takes to reach a final and winning position. There is a little correction to be made in this characterization though. The point is that part of its time $\mathcal{M}$ may spend just waiting for its adversary to move,

and it would be unfair to bill $\mathcal{M}$ for the time for which it, probably, is not responsible. Our solution is to subtract from the overall time the moveless intervals preceding the adversary's moves, i.e. the intervals that intuitively correspond to the adversary's "thinking periods". These intuitions are accounted for by the following definitions.

Let $\mathcal{M}$ be an HPM, e a bounded valuation, B any e-computation branch of $\mathcal{M}$, and Γ the run spelled by B. For any labmove λ of Γ, we define the **thinking period** for λ as $m-n$, where m is the timestamp of λ and n is the timestamp of the labmove immediately preceding λ in Γ, or is 0 if there are no such labmoves. Next, we define **$\top$'s time** in B (or in Γ) as the sum of the thinking periods for all $\top$-labeled moves of Γ. **$\bot$'s time** is defined similarly. Note that, for either player $\wp$, $\wp$'s time will be finite iff there are only finitely many moves made by $\wp$; otherwise it will be infinite.

Definition 7.1 Let A be a game, h a function from natural numbers to natural numbers, and $\mathcal{M}$ an HPM.

1. We say that $\mathcal{M}$ **runs in time** h, or that $\mathcal{M}$ is an h **time machine**, iff, for any bounded valuation e and any e-computation branch B of $\mathcal{M}$, $\top$'s time in B is less than $h\big(e(\mathfrak{b})\big)$.

2. We say that $\mathcal{M}$ **wins** (**computes**, **solves**) A **in time** h, or that $\mathcal{M}$ **is an** h **time solution for** A, iff $\mathcal{M}$ is an h time machine and, for any bounded valuation e, $\mathcal{M}\models_e A$.

3. We say that A is **computable** (**winnable**, **solvable**) **in time** h iff it has an h time solution.

4. We say that $\mathcal{M}$ **runs in polynomial time**, or that $\mathcal{M}$ is a **polynomial time machine**, iff it runs in time h for some polynomial function h.

5. We say that $\mathcal{M}$ **wins** (**computes**, **solves**) A **in polynomial time**, or that $\mathcal{M}$ **is a polynomial time solution for** A, iff $\mathcal{M}$ is an h time solution for A for some polynomial function h. Symbolically, this will be written as
$$\mathcal{M}\models^{P} A.$$

6. We say that A is **computable** (**winnable**, **solvable**) **in polynomial time**, or **polynomial time computable** (**winnable**, **solvable**), iff it has a polynomial time solution.

Many concepts introduced within the framework of computability are generalizations — for the interactive context — of ordinary and well-studied concepts of the traditional theory of computation. The above-defined time complexity or polynomial time computability are among such concepts. Let us look at the traditional notion of polynomial time decidability of a predicate $p(x)$ for instance. With a moment's thought, it can be seen to be equivalent to polynomial time computability (in the sense of Definition 7.1) of the game $p(x) \sqcup \neg p(x)$, or — if you prefer — the game $\sqcap x\big(p(x) \sqcup \neg p(x)\big)$ (these two games are essentially the same, with the only difference being that, in one case, the value of x will have to be read from the valuation tape, while in the other case it must be read from the run tape).

8. THE LANGUAGE OF LOGIC CL3 AND ITS SEMANTICS

Logic **CL3** will be axiomatically constructed in Section 10. The present section is merely devoted to its *language*. The building blocks of this formal language are:

- **Nonlogical predicate letters**, for which we use p, q (possibly indexed) as metavariables. With each predicate letter is associated a nonnegative integer called its **arity**. We assume that, for any n, there are infinitely many n-ary predicate letters.
- **Function letters**, for which we use f, g as metavariables. Again, each function letter comes with a fixed **arity**, and we assume that, for any n, there are infinitely many n-ary function letters.
- The binary **logical predicate letter** $=$.
- Infinitely many **variables**. These are the same as the ones fixed in Section 4.
- Technical symbols: the left parenthesis, the right parenthesis, and the comma.

Terms, for which we use $\tau, \theta, \omega, \psi, \xi$ (possibly indexed) as metavariables, are defined as the elements of the smallest set of expressions such that:

- Variables are terms.

- If f is an n-ary function letter and $\tau_1, \ldots, \tau_n$ are terms, then $f(\tau_1, \ldots, \tau_n)$ is a term. When f is 0-ary, we write f instead of $f()$.

CL3-formulas, or, in most contexts simply **formulas**, are defined as the elements of the smallest set of expressions such that:

- If p is an n-ary predicate letter and $\tau_1, \ldots, \tau_n$ are terms, then $p(\tau_1, \ldots, \tau_n)$ is an (**atomic**) formula. We write $\tau_1 = \tau_2$ instead of $=(\tau_1, \tau_2)$. Also, when p is 0-ary, we write p instead of $p()$.
- If E is an atomic formula, $\neg(E)$ is a formula. We can write $\tau_1 \neq \tau_2$ instead of $\neg(\tau_1 = \tau_2)$.
- $\bot$ and $\top$ are formulas.
- If $E_1, \ldots, E_n$ $(n \geq 2)$ are formulas, then so are $(E_1) \wedge \ldots \wedge (E_n)$, $(E_1) \vee \ldots \vee (E_n)$, $(E_1) \sqcap \ldots \sqcap (E_n)$, $(E_1) \sqcup \ldots \sqcup (E_n)$.
- If E is a formula and x is a variable other than $\mathfrak{b}$, then $\forall x(E)$, $\exists x(E)$, $\sqcap x(E)$, $\sqcup x(E)$ are formulas.

Note that, terminologically, $\top$ and $\bot$ do not count as atoms. For us, atoms are formulas containing no logical operators. The formulas $\top$ and $\bot$ do not qualify because they *are* (0-ary) logical operators themselves.

Sometimes we can write $E_1 \wedge \ldots \wedge E_n$ for an unspecified $n \geq 1$ (rather than $n \geq 2$). Such a formula, in the case $n = 1$, should be understood as simply E_1. Similarly for $\vee, \sqcap, \sqcup$.

Also, where S is a set of formulas, we may write

$$\wedge S$$

for the $\wedge$-conjunction of the elements of S. Again, if S only has one element F, then $\wedge S$ is simply F. Similarly for $\vee, \sqcap, \sqcup$. Furthermore, we do not rule out the possibility of S being empty when using this notation. It is our convention that, when S is empty, both $\wedge S$ and $\sqcap S$ mean $\top$, and both $\vee S$ and $\sqcup S$ mean $\bot$.

$\neg E$, where E is not atomic, will be understood as a standard abbreviation: $\neg\top = \bot$, $\neg\neg E = E$, $\neg(A \wedge B) = \neg A \vee \neg B$, $\neg \sqcap x E = \sqcup x \neg E$, etc.

And $E \to F$ will be understood as an abbreviation of $\neg E \vee F$. Also, if we write

$$E_1 \to E_2 \to E_3 \to \ldots \to E_n,$$

this is to be understood as an abbreviation of $E_1 \to (E_2 \to (E_3 \to (\ldots(E_{n-1} \to E_n)\ldots)))$.

Parentheses will often be omitted — as we just did — if there is no danger of ambiguity. When omitting parentheses, we assume that $\neg$ and the quantifiers have the highest precedence, and $\to$ has the lowest precedence. So, for instance, $\neg \sqcap x E \to F \vee G$ means $(\neg(\sqcap x(E))) \to ((F) \vee (G))$.

The expressions $\vec{x}, \vec{y}, \ldots$ will usually stand for tuples of variables. Similarly for $\vec{\tau}, \vec{\theta}, \ldots$ (for tuples of terms) or $\vec{a}, \vec{b}, \ldots$ (for tuples of constants).

The definitions of *free* and *bound* occurrences of variables are standard, with $\sqcap, \sqcup$ acting as quantifiers along with $\forall, \exists$. We will try to use x, y, z for bound variables only, while using s, r, t, u, v, w for free variables only. There may be some occasional violations of this commitment though.

Convention 8.1 The present conventions apply not only to the language of **CL3** but also to the other formal languages that we deal with later, such as those of **CL4** and **PTA**.

1. For safety and simplicity, throughout the rest of this paper we assume that no formula that we ever consider — unless strictly implied otherwise by the context — may have both bound and free occurrences of the same variable. This restriction, of course, does not yield any loss of expressive power as variables can always be renamed so as to satisfy this condition.

2. Sometimes a formula F will be represented as $F(s_1, \ldots, s_n)$, where the s_i are variables. When doing so, we do not necessarily mean that each s_i has a free occurrence in F, or that every variable occurring free in F is among $s_1, \ldots, s_n$. However, it *will* always be assumed (usually only implicitly) that the s_i are pairwise distinct, and have no bound occurrences in F. In the context set by the above representation, $F(\tau_1, \ldots, \tau_n)$ will mean the result of replacing, in F, each occurrence of each s_i by term τ_i. When writing $F(\tau_1, \ldots, \tau_n)$, it will always be assumed (again, usually only implicitly) that the terms $\tau_1, \ldots, \tau_n$ contain no variables that have bound occurences in F, so that there are no unpleasant collisions of variables when doing replacements.

3. Similar — well established in the literature — notational conventions apply to terms.

An **interpretation**[5] is a function * that sends each n-ary predicate letter p to an n-ary predicate (elementary game) $p^*(s_1, \ldots, s_n)$ which does not depend on any variables other than $s_1, \ldots, s_n$; it also sends each n-ary function letter f to a function

$$f^*: \{0, 1, 10, 11, 100, \ldots\}^n \to \{0, 1, 10, 11, 100, \ldots\};$$

the additional condition required to be satisfied by * is that $=^*$ is an equivalence relation on $\{0, 1, 10, \ldots\}$ preserved by f^* for each function symbol f, and respected by p^* for each nonlogical predicate symbol p.[6]

The above uniquely extends to a mapping that sends each term τ to a function τ^*, and each formula F to a game F^*, by stipulating that:

(1) $s^* = s$ (any variable s).

(2) Where f is an n-ary function letter and $\tau_1, \ldots, \tau_n$ are terms, $\big(f(\tau_1, \ldots, \tau_n)\big)^* = f^*(\tau_1{}^*, \ldots, \tau_n{}^*)$.

(3) Where p is an n-ary predicate letter and $\tau_1, \ldots, \tau_n$ are terms, $\big(p(\tau_1, \ldots, \tau_n)\big)^* = p^*(\tau_1{}^*, \ldots, \tau_n{}^*)$.

(4) * commutes with all logical operators, seeing them as the corresponding game operations:

- $\top^* = \top$;
- $\bot^* = \bot$;
- $(\neg F)^* = \neg F^*$;
- $(E_1 \wedge \ldots \wedge E_n)^* = E^*{}_1 \wedge \ldots \wedge E^*{}_n$;
- $(E_1 \vee \ldots \vee E_n)^* = E^*{}_1 \vee \ldots \vee E^*{}_n$;
- $(E_1 \sqcap \ldots \sqcap E_n)^* = E^*{}_1 \sqcap \ldots \sqcap E^*{}_n$;
- $(E_1 \sqcup \ldots \sqcup E_n)^* = E^*{}_1 \sqcup \ldots \sqcup E^*{}_n$;
- $(\forall x E)^* = \forall x(E^*)$;
- $(\exists x E)^* = \exists x(E^*)$;

- $(\sqcap x E)^* = \sqcap x(E^*)$;
- $(\sqcup x E)^* = \sqcup x(E^*)$.[7]

When O is a function symbol, a predicate symbol, or a formula, and $O^* = W$, we say that * **interprets** O as W. We can also refer to such a W as "O **under interpretation** *".

When a given formula is represented as $F(x_1,\ldots,x_n)$, we will typically write $F^*(x_1,\ldots,x_n)$ instead of $\bigl(F(x_1,\ldots,x_n)\bigr)^*$. A similar practice will be used for terms as well.

Definition 8.2 We say that an HPM $\mathcal{M}$ is a **uniform polynomial time solution** for a formula F iff, for any interpretation *, $\mathcal{M}$ is a polynomial time solution for F^*.

Intuitively, a uniform polynomial time solution is a "purely logical" efficient solution. "Logical" in the sense that it does not depend on the meanings of the nonlogical symbols (predicate and function letters) — does not depend on a (the) interpretation *, that is. It is exactly these kinds of solutions that we are interested in when seeing CL as a logical basis for applied theories or knowledge base systems. As a universal-utility tool, CL (or a CL-based compiler) would have no knowledge of the meanings of those nonlogical symbols (the meanings that will be changing from application to application and from theory to theory), other than what is explicitly given by the target formula and the axioms or the knowledge base of the system.

9. SOME CLOSURE PROPERTIES OF POLYNOMIAL TIME COMPUTABILITY

In this section we establish certain important closure properties for polynomial time computability of games. For simplicity we restrict them to games expressible in the language of **CL3**, even though it should be pointed out that these results can be stated and proven in much more general forms than presented here.

By an (inference) **rule** we mean a binary relation $\mathcal{R}$ between sequences of formulas and formulas, instances of which are schematically written as

$$\frac{X_1 \quad \ldots \quad X_n}{X}, \qquad (4)$$

where $X_1,\ldots,X_n$ are (metavariables for) formulas called the **premises**, and X is (a metavariable for) a formula called the **conclusion**. Whenever $\mathcal{R}(\langle X_1,\ldots,X_n\rangle, X)$ holds, we say that X **follows from** $X_1,\ldots,X_n$ by $\mathcal{R}$.

We say that such a rule $\mathcal{R}$ is **uniform-constructively sound** iff there is an effective procedure that takes any instance $(\langle X_1,\ldots,X_n\rangle, X)$ of the rule, any HPMs $\mathcal{M}_1,\ldots,\mathcal{M}_n$ and returns an HPM $\mathcal{M}$ such that, for any interpretation *, whenever $\mathcal{M}_1 \models^P {X_1}^*,\ldots,\mathcal{M}_n \models^P {X_n}^*$, we have $\mathcal{M} \models^P X^*$.

Our formulations of rules, as well as our later treatment, rely on the following notational and terminological conventions.

(1) A **positive occurrence** of a subformula is an occurrence that is not in the scope of $\neg$. Since officially only atoms may come with a $\neg$, occurrences of non-atomic subformulas will always be positive.

(2) A **surface occurrence** of a subformula is an occurrence that is not in the scope of any choice operators ($\sqcap, \sqcup, \sqcap, \sqcup$).

(3) A formula not containing choice operators — i.e., a formula of the classical language — is said to be **elementary**.

(4) The **elementarization**

$$\|F\|$$

of a formula F is the result of replacing in F all surface occurrences of $\sqcup$- and $\sqcup$-subformulas by $\bot$, and all surface occurrences of $\sqcap$- and $\sqcap$-subformulas by $\top$. Note that $\|F\|$ is (indeed) an elementary formula.

(5) We will be using the notation

$$F[E_1,\ldots,E_n]$$

to mean a formula F together with some (single) fixed positive surface occurrences of each subformula E_i. Here the *formulas* E_i are not required to be pairwise distinct, but their *occurrences* are. Using this notation sets a context in which $F[H_1,\ldots,H_n]$ will mean the result of replacing in $F[E_1,\ldots,E_n]$ the (fixed) occurrence of each E_i by H_i. Note again that here we are talking about some *occurrences* of $E_1, \dots, E_n$. Only those occurrences get replaced when moving from $F[E_1, \dots, E_n]$ to $F[H_1, \dots, H_n]$, even if the formula also had some other occurrences of $E_1, \dots, E_n$.

(6) In any context where the notation of the previous clause is used (specifically, in the formulations of the rules of $\sqcup$-Choose, $\bigsqcup$-Choose and Wait below), all formulas are assumed to be in negation normal form, meaning that they contain no $\rightarrow$, and no $\neg$ applied to non-atomic subformulas.

Below we prove the uniform-constructive soundness of several rules. Our proofs will be limited to showing how to construct an HPM $\mathcal{M}$ from an arbitrary instance — in the form (4) — of the rule and arbitrary HPMs $\mathcal{M}_1, \dots, \mathcal{M}_n$ (purported solutions for the premises). In each case it will be immediately clear from our description of $\mathcal{M}$ that it can be constructed effectively, that it runs in polynomial time as long as so do $\mathcal{M}_1, \dots, \mathcal{M}_n$, and that its work in no way depends on an interpretation * applied to the games involved. Since an interpretation * is typically irrelevant in such proofs, we will often omit it and write simply F where, strictly speaking, F^* is meant. That is, we identify formulas with the games into which they turn once an interpretation is applied to them. Likewise, we may omit a valuation e and write F instead of $e[F]$ or $e[F^*]$.

9.1. $\sqcup$-Choose

$\sqcup$-Choose is the following rule:

$$\frac{F[H_i]}{F[H_0 \sqcup \dots \sqcup H_n]},$$

where $n \geq 1$ and $i \in \{0, \dots, n\}$.

Whenever a formula F follows from a formula E by $\sqcup$-Choose, we say that E is a **$\sqcup$-Choose-premise** of F.

Theorem 9.1 *$\sqcup$-Choose is uniform-constructively sound.*

Idea. This rule most directly encodes an action that $\mathcal{M}$ should perform in order to successfully solve the conclusion. Namely, $\mathcal{M}$ should choose H_i and then continue playing as the machine that (presumably) solves the premise. ■

Proof. Let $\mathcal{M}_1$ be an arbitrary HPM (a purported polynomial time solution for the premise). We let $\mathcal{M}$ (the will-be polynomial time solution for the conclusion) be the machine that works as follows.

At the beginning, without looking at its run tape or valuation tape, $\mathcal{M}$ makes the move α that brings $F[H_0 \sqcup \dots \sqcup H_n]$ down to $F[H_i]$. For instance, if $F[H_0 \sqcup \dots \sqcup H_n]$ is $X \wedge (Y \vee (Z \sqcup T))$ and $F[H_i]$ is $X \wedge (Y \vee Z)$, then 1.1.0 is such a move.

What $\mathcal{M}$ does after that can be characterized as "turning itself into $\mathcal{M}_1$" and playing the rest of the game as $\mathcal{M}_1$ would. In more detail, $\mathcal{M}$ starts simulating and mimicking $\mathcal{M}_1$. During this simulation, $\mathcal{M}$ "imagines" that $\mathcal{M}_1$ has the same valuation e on its valuation tape as $\mathcal{M}$ itself has, and that the run tape of $\mathcal{M}_1$ spells the same run as its own run tape does, with the difference that the move α made by $\mathcal{M}$ at the beginning is ignored (as if it was not there). To achieve the effect of consistency between the real and imaginary valuation and run tapes, what $\mathcal{M}$ does is that, every time the simulated $\mathcal{M}_1$ reads a cell of its valuation or run tape, $\mathcal{M}$ reads the content of the corresponding cell of its own valuation or run tape, and feeds that content to the simulation as the content of the cell that $\mathcal{M}_1$ was reading. And whenever, during the simulation, $\mathcal{M}_1$ makes a move, $\mathcal{M}$ makes the same move in the real play.

The run generated by $\mathcal{M}$ in the real play will look like $\langle \top\alpha, \Gamma \rangle$. It is not hard to see that then Γ will be a run generated by $\mathcal{M}_1$. So, if $\mathcal{M}_1$ wins $F[H_i]$, implying that $\mathbf{Wn}^{F[H_i]}{}_e\langle \Gamma \rangle = \top$, then $\mathcal{M}$ wins $F[H_0 \sqcup \dots \sqcup H_n]$, because $\mathbf{Wn}^{F[H_0 \sqcup \dots \sqcup H_n]}{}_e\langle \top\alpha, \Gamma \rangle = \mathbf{Wn}^{F[H_i]}{}_e\langle \Gamma \rangle$.

Simulation does impose a certain overhead, which makes $\mathcal{M}$ slower than $\mathcal{M}_1$. But, with some analysis, details of which are left to the reader, it can be seen that the slowdown would be at most polynomial, meaning that, if $\mathcal{M}_1$ runs in polynomial time, then so does $\mathcal{M}$. ■

9.2. $\sqcup$-*Choose*

$\sqcup$-Choose is the following rule:

$$\frac{F[H(s)]}{F[\sqcup xH(x)]},$$

where x is any non-$\mathfrak{b}$ variable, s is a variable with no bound occurrences in the premise, and $H(s)$ is the result of replacing by s all free occurrences of x in $H(x)$ (rather than vice versa).

Whenever a formula F follows from a formula E by $\sqcup$-Choose, we say that E is a **$\sqcup$-Choose-premise** of F.

Theorem 9.2 $\sqcup$-*Choose is uniform-constructively sound.*

Idea. Very similar to the previous case. $\mathcal{M}$ should specify x as (the value of) s, and then continue playing as the machine that solves the premise. ■

Proof. Let $\mathcal{M}_1$ be an arbitrary HPM (a purported polynomial time solution for the premise). We let $\mathcal{M}$ (the will-be polynomial time solution for the conclusion) be the machine that, with a valuation e spelled on its valuation tape, works as follows. At the beginning, $\mathcal{M}$ makes the move that brings $F[\sqcup xH(x)]$ down to $F[H(s)]$. For instance, if $F[\sqcup xH(x)]$ is $X \wedge (Y \vee \sqcup xZ(x))$ and $F[H(s)]$ is $X \wedge (Y \vee Z(s))$, then $1.1.c$ is such a move, where $c = e(s)$ (the machine will have to read c from its valuation tape). After this move, $\mathcal{M}$ starts simulating and mimicking $\mathcal{M}_1$ in the same fashion as in the proof of Theorem 9.1. And, again, as long as $\mathcal{M}_1$ wins $F[H(s)]$ in polynomial time, $\mathcal{M}$ wins $F[\sqcup xH(x)]$ in polynomial time. ■

9.3. *Wait*

Wait is the following rule:

$$\frac{\|F\| \qquad F_1 \qquad \ldots \qquad F_n}{F}$$

(remember that $\|F\|$ means the elementarization of F), where $n \geq 0$ and the following two conditions are satisfied:

(1) Whenever F has the form $X[Y_0 \sqcap \ldots \sqcap Y_m]$, each formula $X[Y_i]$ $(0 \leq i \leq m)$ is among $F_1, \ldots, F_n$.

(2) Whenever F has the form $X[\sqcap xY(x)]$, for some variable s different from $\mathfrak{b}$ and not occurring in F, the formula $X[Y(s)]$ is among $F_1, \ldots, F_n$. Here $Y(s)$ is the result of replacing by s all free occurrences of x in $Y(x)$ (rather than vice versa).

Whenever the above relation holds, we say that $\|F\|$ is the **special Wait-premise** of F, and that $F_1, \ldots, F_n$ are **ordinary Wait-premises** of F.

The following lemma, on which we are going to rely in this subsection, can be verified by a straightforward induction on the complexity of F, which we omit. Remember that $\langle\rangle$ stands for the empty run.

Lemma 9.3 *For any formula F, interpretation * and valuation e,* $\mathbf{Wn}^{F^*}{}_e\langle\rangle = \mathbf{Wn}^{\|F\|^*}{}_e\langle\rangle$.

Theorem 9.4 *Wait is uniform-constructively sound.*

Idea. $\mathcal{M}$ should wait (hence the name "Wait" for the rule) until the adversary makes a move. If this never happens, in view of the presence of the premise $\|F\|$, a win for $\mathcal{M}$ is guaranteed by Lemma 9.3. Otherwise, any (legal) move by the adversary essentially brings the conclusion down to one of the premises $F_1, \ldots, F_n$; then $\mathcal{M}$ continues playing as the machine that wins that premise. ■

Proof. Assume $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$ are polynomial time solutions for $\|F\|, F_1, \ldots, F_n$, respectively. We let $\mathcal{M}$, the will-be solution for F, whose construction does not depend on the just-made assumption, be a machine that, with a bounded valuation e spelled on its valuation tape, works as follows.

At the beginning, $\mathcal{M}$ keeps waiting until the environment makes a move. If such a move is never made, then the run that is generated is empty. Since $\|F\|$ is elementary and $\mathcal{M}_0$ wins it, it is classically true (a false elementary game would be automatically lost by any machine). But then, in view of Lemma 9.3, $\mathcal{M}$ wins (the empty run of) F. And, note, $\top$'s time in this case is 0.

Suppose now the environment makes a move. Note that the time during which the machine was waiting does not contribute anything to $\top$'s time. We may assume that the move made by the environment is legal, or else the machine immediately wins. With a little thought, one can see that any legal move α by the environment brings the game $e[F]$ down to $g[F_i]$ for a certain bounded valuation g — with $g(\mathfrak{b}) = e(\mathfrak{b})$ — and one of the premises F_i of the rule. For example, if F is $(X \sqcap Y) \vee \sqcap x Z(x)$, then a legal move α by the environment should be either 0.0 or 0.1 or $1.c$ for some constant c (of size $\leq e(\mathfrak{b})$). In the case $\alpha = 0.0$, the above-mentioned premise F_i will be $X \vee \sqcap x Z(x)$, and g will be the same as e. In the case $\alpha = 0.1$, F_i will be $Y \vee \sqcap x Z(x)$, and g, again, will be the same as e. Finally, in the case $\alpha = 1.c$, F_i will be $(X \sqcap Y) \vee Z(s)$ for a variable s different from $\mathfrak{b}$ and not occurring in F, and g will be the valuation that sends s to c and agrees with e on all other variables, so that $g[(X \sqcap Y) \vee Z(s)]$ is $e[(X \sqcap Y) \vee Z(c)]$, with the latter being the game to which $e[F]$ is brought down by the labmove $\bot 1.c$.

After the above event, $\mathcal{M}$ starts simulating and mimicking the machine $\mathcal{M}_i$ in the same fashion as in the proofs of Theorems 9.1 and 9.2, with the only being difference that, if $g \neq e$, the imaginary valuation tape of the simulated machine now spells g rather than e.

As in the earlier proofs, it can be seen that $\mathcal{M}$, constructed as above, is a polynomial time solution for F. ■

9.4. *Modus Ponens (MP)*

Modus Ponens is the following rule:

$$\frac{F_0 \qquad \ldots \qquad F_n \qquad F_0 \wedge \ldots \wedge F_n \rightarrow F}{F},$$

where $n \geq 0$.

Theorem 9.5 *Modus Ponens is uniform-constructively sound.*

Idea. Together with the real play of F, $\mathcal{M}$ plays an imaginary game for each of the premises, in which it mimics the machines that win those premises. In addition, it applies copycat between each premise F_i and the corresponding conjunct of the antecedent of the rightmost premise, as well as between (the real) F and the consequent of that premise. ■

Proof. Assume $\mathcal{M}_0, \ldots, \mathcal{M}_n$ and $\mathcal{N}$ are HPMs that win $F_0, \ldots, F_n$ and $F_0 \wedge \ldots \wedge F_n \rightarrow F$ in polynomial time, respectively (as in the previous proofs, our construction of $\mathcal{M}$ does not depend on this assumption; only the to-be-made conclusion $\mathcal{M} \models^P F$ does). For simplicity, below we reason under the assumption that $n \geq 1$. Extending our reasoning so as to also include the case $n = 0$ does not present a problem.

We let $\mathcal{M}$ be the following HPM. Its work on a valuation e consists in simulating, in parallel, the machines $\mathcal{M}_0, \ldots, \mathcal{M}_n, \mathcal{N}$ with the same e on their valuation tapes, and also continuously polling (in parallel with simulation) its own valuation tape to see if the environment has made a new move. These simulations proceed in the same fashion as in the proofs of the earlier theorems, with the only difference that now $\mathcal{M}$ actually maintains records of the contents of the imaginary run tapes of the simulated machines (in the proof of Theorem 9.1, $\mathcal{M}$ was simply using its own run tape in the role of such a "record").

As before, we may assume that the environment does not make illegal moves, for then $\mathcal{M}$ immediately wins. We can also safely assume that the simulated machines do not make illegal moves, or else our assumptions about their winning the corresponding games would be wrong.[8] If so, in the process of the above simulation-polling routine, now and then, one of the following four types of events will be happening (or rather detected):

Event 1. $\mathcal{M}_i$ $(0 \leq i \leq n)$ makes a move α. Then $\mathcal{M}$ appends the labmove $\bot 0.i.\alpha$ at the end of the position spelled on the imaginary run tape of $\mathcal{N}$ in its simulation.[9]

Event 2. $\mathcal{N}$ makes a move $0.i.\alpha$ $(0 \leq i \leq n)$. Then $\mathcal{M}$ appends the labmove $\bot\alpha$ at the end of the imaginary run tape of $\mathcal{M}_i$ in its simulation.

Event 3. $\mathcal{N}$ makes a move $1.\alpha$. Then $\mathcal{M}$ makes the move α in the real play.

Event 4. The environment makes a move α in the real play. Then $\mathcal{M}$ appends the labmove $\bot 1.\alpha$ at the end of the imaginary run tape of $\mathcal{N}$ in its simulation.

What is going on here is that $\mathcal{M}$ applies copycat between $n+2$ pairs of (sub)games, real or imaginary. Namely, it mimics, in (the real play of) F, $\mathcal{N}$'s moves made in the consequent of (the imaginary play of) $F_0 \wedge \ldots \wedge F_n \rightarrow F$, and vice versa: uses (the real) environment's moves made (in the real play of) F as (an imaginary) environment's moves in the consequent of $F_0 \wedge \ldots \wedge F_n \rightarrow F$. Also, for each $i \in \{0, \ldots, n\}$, $\mathcal{M}$ uses the moves made by $\mathcal{M}_i$ in F_i as environment's moves in the F_i component of $F_0 \wedge \ldots \wedge F_n \rightarrow F$, and vice versa: uses the moves made by $\mathcal{N}$ in that component as environment's moves in F_i. Therefore, the final positions[10] hit by the above imaginary and real plays will be

$$F'_0,\ \ldots,\ F'_n,\ F'_1 \wedge \ldots \wedge F'_n \rightarrow F' \text{ and } F'$$

for some $F'_0, \ldots, F'_n, F'$. Our assumption that the machines $\mathcal{M}_0, \ldots, \mathcal{M}_n$ and $\mathcal{N}$ win the games $F_0, \ldots, F_n$ and $F_1 \wedge \ldots \wedge F_n \rightarrow F$ implies that each $G \in \{F'_0,\ \ldots,\ F'_n,\ F'_1 \wedge \ldots \wedge F'_n \rightarrow F'\}$ is $\top$-won, in the sense that $\mathbf{Wn}^G{}_e\langle\rangle = \top$. It is then obvious that so should be F'. Thus, the (real) play of F brings it down to the $\top$-won F', meaning that $\mathcal{M}$ wins F.

With some thought, one can also see that $\mathcal{M}$ runs in polynomial time. The only reason why $\mathcal{M}$ may spend "too much" time thinking before making a move could be that it waited "too long" to see what move was made by one (or several) of the simulated machines. But this would not happen because, by our assumption, those machines run in polynomial time, so, whenever they make a move, it never takes them "too long" to do so. ■

10. LOGIC CL3

Before we get to our version of formal arithmetic, it would not hurt to identify the (pure) logic on which it is based — based in the same sense as the traditional Peano arithmetic is based on classical logic. This logic is **CL3**. With minor technical differences not worth our attention and not warranting a new name for the logic, our present version of **CL3** is the same as the same-name logic introduced and studied in Japaridze (2006c).[11]

The **language** of **CL3** has already been described in Section 8.

The **axioms** of this system are all classically valid elementary formulas. Here by classical validity, in view of Gödel's completeness theorem, we mean provability in classical first-order calculus. Specifically, in classical first-order calculus with function letters and $=$,

where $=$ is treated as the logical *identity* predicate (so that, say, $x = x$, $x = y \rightarrow (E(x) \rightarrow E(y))$, etc. are valid/provable).

As for the **rules of inference** of **CL3**, they are the $\sqcup$-Choose, $\bigsqcup$-Choose and Wait rules of Section 9. As will be easily seen from the forthcoming soundness and completeness theorem for **CL3** (in conjunction with Theorem 9.5), **CL3** is closed under Modus Ponens. So, there is no need for officially including it among the rules of inference, doing which would destroy the otherwise analytic property of the system.

A **CL3-proof** of a formula F is a sequence $E_1, \ldots, E_n$ of formulas, with $E_n = F$, such that each E_i is either an axiom or follows from some earlier formulas of the sequence by one of the rules of **CL3**. When a **CL3**-proof of F exists, we say that F is **provable** in **CL3**, and write **CL3** $\vdash F$. Otherwise we write **CL3** $\nvdash F$. Similarly for any other formal systems as well.

Example 10.1 The formula $\forall x p(x) \rightarrow \sqcap x p(x)$ is provable in **CL3**. It follows by Wait from the axioms $\forall x p(x) \rightarrow \top$ (special Wait-premise) and $\forall x p(x) \rightarrow p(s)$ (ordinary Wait-premise).

On the other hand, the formula $\sqcap x p(x) \rightarrow \forall x p(x)$, i.e. $\bigsqcup x \neg p(x) \vee \forall x p(x)$, in not provable. Indeed, this formula has no $\sqcup$-Choose-premises because it does not contain $\sqcup$. Its elementarization (special Wait-premise) is $\bot \vee \forall x p(x)$ which is not an axiom nor the conclusion of any rules. Hence $\bigsqcup x \neg p(x) \vee \forall x p(x)$ cannot be derived by Wait, either. This leaves us with $\bigsqcup$-Choose. But if $\bigsqcup x \neg p(x) \vee \forall x p(x)$ is derived by $\bigsqcup$-Choose, then the premise should be $\neg p(s) \vee \forall x p(x)$ for some variable s. The latter, however, is neither an axiom nor the conclusion of any of the three rules of **CL3**.

Example 10.2 The formula $\sqcap x \bigsqcup y \big(p(x) \rightarrow p(y)\big)$, whose elementarization is $\top$, is provable in **CL3** as follows:

1. $\top$ Axiom
2. $p(s) \rightarrow p(s)$ Axiom
3. $\bigsqcup y \big(p(s) \rightarrow p(y)\big)$ $\bigsqcup$-Choose: 2
4. $\sqcap x \bigsqcup y \big(p(x) \rightarrow p(y)\big)$ Wait: 1,3

On the other hand, the formula $\bigsqcup y \sqcap x \big(p(x) \rightarrow p(y)\big)$ can be seen to be unprovable, even though its classical counterpart $\exists y \forall x \big(p(x) \rightarrow p(y)\big)$ is an axiom and hence provable.

Example 10.3 While the formula $\exists x\bigl(x=f(s)\bigr)$ is classically valid and hence provable in **CL3**, its constructive counterpart $\sqcup x\bigl(x=f(s)\bigr)$ can be easily seen to be unprovable. This is no surprise. In view of the expected soundness of **CL3**, provability of $\sqcup x\bigl(x=f(s)\bigr)$ would imply that every function f is computable (worse yet: efficiently computable), which, of course, is not the case.

Exercise 10.4 To see the resource-consciousness of **CL3**, show that it does not prove $p\sqcap q\to(p\sqcap q)\wedge(p\sqcap q)$, even though this formula has the form $F\to F\wedge F$ of a classical tautology.

Theorem 10.5 **CL3** $\vdash X$ *iff* X *has a uniform polynomial time solution (any formula* X*). Furthermore:*

Uniform-constructive soundness: *There is an effective procedure that takes any* **CL3***-proof of any formula* X *and constructs a uniform polynomial time solution for* X.

Completeness: *If* **CL3** $\not\vdash X$*, then, for any HPM* $\mathscr{M}$*, there is an interpretation* * *such that* $\mathscr{M}$ *does not win* X^* *(let alone winning in polynomial time).*

Idea. The soundness of **CL3** was, in fact, already established in the preceding section. For completeness, assume **CL3** $\not\vdash X$ and consider any HPM $\mathscr{M}$. If $\|X\|$ is an axiom, a smart environment can always make a move that brings X down to an unprovable ordinary Wait-premise of X, or else X would be derivable by Wait; such a Wait-premise is less complex than X and, by the induction hypothesis, $\mathscr{M}$ loses. If $\|X\|$ is not an axiom, then it is false under a certain interpretation, and therefore $\mathscr{M}$ will have to make a move to avoid an automatic loss. But any (legal) move by $\mathscr{M}$ brings X down to an unprovable Choose-premise of it (or else X would be derivable by a Choose rule) and, by the induction hypothesis, $\mathscr{M}$ again loses. ∎

Proof. Modulo the results of Section 9, the soundness ("only if") part of this theorem, in the strong "uniform-constructive" form, is straightforward. Formally this fact can be proven by induction on the lengths of **CL3**-proofs. All axioms of **CL3** are obviously "solved" by a machine that does nothing at all. Next, as an induction hypothesis, assume $X_1,\ldots,X_n$ are **CL3**-provable formulas, $\mathscr{M}_1,\ldots,\mathscr{M}_n$ are uniform polynomial time solutions for them, and X follows from those formulas

by one of the rules of **CL3**. Then, as immediately implied by the results of Section 9, we can effectively construct a uniform polynomial time solution $\mathscr{M}$ for X.

The rest of this proof will be devoted to the completeness ("if") part of the theorem.

Consider an arbitrary formula X with **CL3** $\not\vdash X$, and an arbitrary HPM $\mathscr{M}$. Here we describe a scenario of the environment's behavior in interaction with $\mathscr{M}$ — call this "behavior" the *counterstrategy* — that makes $\mathscr{M}$ lose F^* on e for a certain appropriately selected interpretation * and a certain appropriately selected bounded valuation e even if the time of $\mathscr{M}$ is not limited at all.

For a formula Y and valuation g, we say that g is Y**-distinctive** iff g assigns different values to different free variables of Y. We select e to be an X-distinctive bounded valuation. Here we let $e(\mathfrak{b})$ be "sufficiently large" to allow certain flexibilities needed below.

How our counterstrategy acts depends on the current game (formula, "position") Y to which the original game X has been brought down by the present time in the play. Initially, Y is X.

As an induction hypothesis, we assume that **CL3** $\not\vdash Y$ and e is Y-distinctive. We separately consider the following two cases.

Case 1: $\|Y\|$ is classically valid. Then there should be a **CL3**-unprovable formula Z — an ordinary Wait-premise of Y — satisfying the conditions of one of the following two subcases, for otherwise Y would be derivable by Wait. Our counterstrategy selects one such Z (say, lexicographically the smallest one), and acts according to the corresponding prescription as given below.

Subcase 1.1: Y has the form $F[G_0\sqcap\ldots\sqcap G_m]$, and Z is $F[G_i]$ $(i\in\{0,\ldots,m\})$. In this case, the counterstrategy makes the move that brings Y down to Z, and calls itself on Z in the role of the "current" formula Y.

Subcase 1.2: Y has the form $F[\sqcap xG(x)]$, and Z is $F[G(s)]$, where s is a variable different from $\mathfrak{b}$ and not occurring in Y. We may assume here that $e(s)$ is different from any $e(r)$ where r is any other $(\neq s)$ free variable of Y. Thus, e remains a Z-distinctive valuation. In this case, our counterstrategy makes the move that brings Y down to Z (such a move is the one that specifies the value of x as $e(s)$ in the indicated occurrence of $\sqcap xG(x)$), and calls itself on Z in the role of Y.

Case 2: $\|Y\|$ is not classically valid. Then our counterstrategy inactively waits until $\mathcal{M}$ makes a move.

Subcase 2.1. If such a move is never made, then the run that is generated is empty. Since e is a Y-distinctive valuation, of course, it is also $\|Y\|$-distinctive. It is a common knowledge from classical logic that, whenever a formula F is invalid (as is $\|Y\|$ in our present case) and e is an F-distinctive valuation, $e[F]$ is false in some model. So, $e[\|Y\|]$ is false in/under some model/interpretation *. This, in view of Lemma 9.3, implies that $\mathbf{Wn}^{Y^*}{}_e\langle\rangle = \bot$ and hence $\mathcal{M}$ is the loser in the overall play of X^* on e.

Subcase 2.2. Now suppose $\mathcal{M}$ makes a move. We may assume that such a move is legal, or else $\mathcal{M}$ immediately loses. With a little thought, one can see that any legal move α by $\mathcal{M}$ will bring the game down to Z for a certain formula Z such that Y follows from Z by $\sqcup$-Choose or $\bigsqcup$-Choose, and e remains — or, at least, can be safely assumed to remain — Z-distinctive. But then, since $\mathbf{CL3} \not\vdash Y$, we also have $\mathbf{CL3} \not\vdash Z$. In this case, our counterstrategy calls itself on Z in the role of Y.

It is clear that, sooner or later, the interaction will end according to the scenario of Subcase 2.1, in which case, as we observed, $\mathcal{M}$ will be the loser in the overall play of X^* on e for a certain interpretation *. ■

11. CL4, THE METALOGIC OF CL3

In this section we present an auxiliary deductive system **CL4**. Syntactically, it is a conservative extension of **CL3**. Semantically, we treat **CL4** as a *metalogic* of **CL3**, in the sense that the formulas of **CL4** are seen as schemata of **CL3**-formulas, and the theorems of **CL4** as schemata of theorems of **CL3**. System **CL4** — in an unsubstantially different form — was initially introduced and studied in Japaridze (2007b) where, unlike our present treatment, it was seen as a logic (rather than metalogic) in its own rights, soundly and completely axiomatizing a more expressive fragment of computability logic than **CL3** does. Simplicity is the only reason why we prefer to see **CL4** as just a metalogic here.

The language of **CL4** is obtained from that of **CL3** by adding to it

nonlogical **general letters**, on top of the predicate letters of the language of **CL3** that in this new context, following the terminological tradition of computability logic, we rename as **elementary letters**. We continue using the lowercase p, q (possibly indexed) as metavariables for elementary letters, and will be using the uppercase P, Q (possibly indexed) as metavariables for general letters. Just as this is the case with the elementary letters, we have infinitely many n-ary general letters for each arity (natural number) n. In our present approach, the nonlogical elementary letters of the language of **CL4** will be seen as metavariables for elementary formulas of the language of **CL3**, the general letters of the language of **CL4** will be seen as metavariables for any, not-necessarily-elementary, formulas of the language of **CL3**, and the function letters of the language of **CL4** will be seen as metavariables for terms of the language of **CL3**.

Formulas of the language of **CL4**, to which we refer as **CL4-formulas**, are built from atoms, terms, variables and operators in exactly the same way as **CL3**-formulas are, with the only difference that now, along with the old **elementary atoms** — atoms of the form $p(\tau_1, \ldots, \tau_n)$ where p is an n-ary elementary letter and the τ_i are terms — we also have **general atoms**, which are of the form $P(\tau_1, \ldots, \tau_n)$, where P is an n-ary general letter and the τ_i are terms. An **elementary literal** is $\top$, $\bot$, or an elementary atom with or without negation $\neg$. And a **general literal** is a general atom with or without negation. As before, we always assume that negation can only occur in literals; $\neg$ applied to a non-atomic formula, as well as $\rightarrow$, are treated as abbreviations. The concepts of a surface occurrence, positive occurrence etc. straightforwardly extend from the language of **CL3** to the language of **CL4**.

We say that a **CL4**-formula is **elementary** iff it does not contain general atoms and choice operators. Thus, “elementary **CL4**-formula”, “elementary **CL3**-formula” and “formula of classical logic” mean the same. Note that we see the predicate letters of classical logic as elementary rather than general letters.

The **elementarization** $\|F\|$ of a **CL4**-formula F is the result of replacing in it all surface occurrences of $\sqcap$- and $\bigsqcap$-subformulas by $\top$, all surface occurrences of $\sqcup$- and $\bigsqcup$-subformulas by $\bot$, and all positive surface occurrences of general literals by $\bot$.

CL4 has exactly the same axioms as **CL3** does (all classically valid elementary formulas), and has four rules of inference. The first three rules are nothing but the rules of $\sqcup$-Choose, $\bigsqcup$-Choose and Wait of **CL3**, only now applied to any **CL4**-formulas rather than just **CL3**-formulas. The additional, fourth rule, which we call **Match**, is the following:

$$\frac{F[p(\vec{\tau}), \neg p(\vec{\theta})]}{F[P(\vec{\tau}), \neg P(\vec{\theta})]},$$

where P is any n-ary general letter, p is any n-ary nonlogical elementary letter not occurring in the conclusion, and $\vec{\tau}, \vec{\theta}$ are any n-tuples of terms; also, according to our earlier notational conventions, $F[P(\vec{\tau}), \neg P(\vec{\theta})]$ is a formula with two fixed positive occurrences of the literals $P(\vec{\tau})$ and $\neg P(\vec{\theta})$, and $F[p(\vec{\tau}), \neg p(\vec{\theta})]$ is the result of replacing in $F[P(\vec{\tau}), \neg P(\vec{\theta})]$ the above two occurrences by $p(\vec{\tau})$ and $\neg p(\vec{\theta})$, respectively.

It may help some readers to know that **CL4** is an extension of additive-multiplicative affine logic (classical linear logic with weakening), with the letters of the latter understood as our general letters. This fact is an immediate consequence of the earlier-known soundness of affine logic (proven in Japaridze 2009a) and completeness of **CL4** (proven in Japaridze 2007b) with respect to the semantics of computability logic. As seen from the following example, the extension, however, is not conservative.

Example 11.1 Below is a **CL4**-proof of the formula $(P \wedge P) \vee (P \wedge P) \to (P \vee P) \wedge (P \vee P)$. The latter was used by Blass Blass (1992) as an example of a game-semantically valid principle not provable in affine logic.

1. $(p_1 \wedge p_2) \vee (p_3 \wedge p_4) \to (p_1 \vee p_3) \wedge (p_2 \vee p_4)$ Axiom
2. $(p_1 \wedge p_2) \vee (p_3 \wedge P) \to (p_1 \vee p_3) \wedge (p_2 \vee P)$ Match: 1
3. $(p_1 \wedge p_2) \vee (P \wedge P) \to (p_1 \vee P) \wedge (p_2 \vee P)$ Match: 2
4. $(p_1 \wedge P) \vee (P \wedge P) \to (p_1 \vee P) \wedge (P \vee P)$ Match: 3
5. $(P \wedge P) \vee (P \wedge P) \to (P \vee P) \wedge (P \vee P)$ Match: 4

Example 11.2 In Example 10.2 we saw a **CL3**-proof of $\sqcap x \sqcup y\big(p(x) \to p(y)\big)$. The same proof, of course, is also a **CL4**-proof. Below is a **CL4**-proof of the stronger version of this formula where we have an uppercase rather than lowercase P:

1. $\top$ Axiom
2. $p(s) \to p(s)$ Axiom
3. $P(s) \to P(s)$ Match: 2
4. $\bigsqcup y\big(P(s) \to P(y)\big)$ $\bigsqcup$-Choose: 3
5. $\sqcap x \bigsqcup y\big(P(x) \to P(y)\big)$ Wait: 1,4

Example 11.3 While **CL4** proves the elementary formula $p \to p \wedge p$, it does not prove its general counterpart $P \to P \wedge P$. Indeed, $\|P \to P \wedge P\| = \top \to \bot \wedge \bot$ and hence, obviously, $P \to P \wedge P$ cannot be derived by Wait. This formula cannot be derived by Choose rules either, because it contains no choice operators. Finally, if it is derived by Match, the premise should be $p \to P \wedge p$ or $p \to p \wedge P$. In either case, such a premise cannot be proven, as it contains no choice operators and its elementarization is $p \to \bot \wedge p$ or $p \to p \wedge \bot$.

Let F be a **CL4**-formula. A **substitution** for F is a function $^\heartsuit$ that sends:

- each nonlogical n-ary elementary letter p of F to an elementary **CL3**-formula $p^\heartsuit(x_1, \ldots, x_n)$ — with here and below $x_1, \ldots, x_n$ being a context-setting fixed n-tuple of pairwise distinct variables — which does not contain any free variables that have bound occurrences in F;
- each n-ary general letter P of F to an (elementary or nonelementary) **CL3**- formula $P^\heartsuit(x_1, \ldots, x_n)$ which does not contain any free variables that have bound occurrences in F;
- each n-ary function symbol f of F to a term $f^\heartsuit(x_1, \ldots, x_n)$ which does not contain any variables that have bound occurrences in F.

The above uniquely extends to a mapping that sends each term τ of F to a term $\tau^\heartsuit$, and each subformula H of F to a **CL3**-formula $H^\heartsuit$ by stipulating that:

(1) $x^{\heartsuit} = x$ (any variable x).

(2) Where f is an n-ary function symbol and $\tau_1, \ldots, \tau_n$ are terms, $\big(f(\tau_1, \ldots, \tau_n)\big)^{\heartsuit} = f^{\heartsuit}(\tau_1{}^{\heartsuit}, \ldots, \tau_n{}^{\heartsuit})$.

(3) $(\tau_1 = \tau_2)^{\heartsuit}$ is $\tau_1{}^{\heartsuit} = \tau_2{}^{\heartsuit}$.

(4) Where $\mathfrak{L}$ is an n-ary nonlogical elementary or general letter and $\tau_1, \ldots, \tau_n$ are terms, $\big(\mathfrak{L}(\tau_1, \ldots, \tau_n)\big)^{\heartsuit} = \mathfrak{L}^{\heartsuit}(\tau_1{}^{\heartsuit}, \ldots, \tau_n{}^{\heartsuit})$.

(5) $^{\heartsuit}$ commutes with all logical operators:

- $\top^{\heartsuit} = \top$;
- $\bot^{\heartsuit} = \bot$;
- $(\neg E)^{\heartsuit} = \neg E^{\heartsuit}$;
- $(E_1 \wedge \ldots \wedge E_n)^{\heartsuit} = E^{\heartsuit}{}_1 \wedge \ldots \wedge E^{\heartsuit}{}_n$;
- $(E_1 \vee \ldots \vee E_n)^{\heartsuit} = E^{\heartsuit}{}_1 \vee \ldots \vee E^{\heartsuit}{}_n$;
- $(E_1 \sqcap \ldots \sqcap E_n)^{\heartsuit} = E^{\heartsuit}{}_1 \sqcap \ldots \sqcap E^{\heartsuit}{}_n$;
- $(E_1 \sqcup \ldots \sqcup E_n)^{\heartsuit} = E^{\heartsuit}{}_1 \sqcup \ldots \sqcup E^{\heartsuit}{}_n$;
- $(\forall x E)^{\heartsuit} = \forall x(E^{\heartsuit})$;
- $(\exists x E)^{\heartsuit} = \exists x(E^{\heartsuit})$;
- $(\bigsqcap x E)^{\heartsuit} = \bigsqcap x(E^{\heartsuit})$;
- $(\bigsqcup x E)^{\heartsuit} = \bigsqcup x(E^{\heartsuit})$.

We say that a **CL3**-formula E is an **instance** of a **CL4**-formula F, or that E **matches** F, iff $E = F^{\heartsuit}$ for some substitution $^{\heartsuit}$ for F.

Theorem 11.4 *A* **CL4***-formula is provable in* **CL4** *iff all of its instances are provable in* **CL3**.

Idea. The completeness part of this theorem is unnecessary for the purposes of the present paper, and its proof is omitted. For the soundness part, consider a **CL4**-provable formula F and an arbitrary instance $F^{\heartsuit}$ of it. We need to construct a **CL3**-proof of $F^{\heartsuit}$. The idea here is to let such a proof simulate the **CL4**-proof of F. Speaking very roughly, simulating steps associated with $\sqcup$-Choose, $\bigsqcup$-Choose and Wait is possible because these rules of **CL4** are also present in **CL3**. As for the

Match rule, it can be simulated by a certain "deductive counterpart" of the earlier seen copycat strategy. Namely, in the bottom-up view of the **CL3**-proof under construction, every application of Wait that modifies a subformula originating from a matched (in the **CL4**-proof) literal, should be followed by a symmetric application of $\sqcup$-Choose or $\bigsqcup$-Choose in the subformula originating from the other matched literal — an application that evens out the two subformulas so that one remains the negation of the other. ■

Proof. Our proof will be focused on the soundness ("only if") part of the theorem, as nothing in this paper relies on the completeness ("if") part. We only want to point out that, essentially, the latter has been proven in Section 5 of Japaridze (2007b). Specifically, the proof of Lemma 5.1 of Japaridze (2007b) proceeds by showing that, if **CL4** $\not\vdash F$, then there is a **CL3**-formula $\lceil F \rceil$ which is an instance of F such that **CL3** $\not\vdash \lceil F \rceil$. However, as noted earlier, the logics under the names "**CL3**" and "**CL4**" are not exactly the same in Japaridze (2006c, 2007b) as they are here. Namely, Japaridze (2006c, 2007b) allowed constants in formulas while now we do not allow them. On the other hand, now we have $=$ and function symbols in the language whereas the approach of Japaridze (2006c, 2007b) did not consider them, nor did it have the special-status variable $\mathfrak{b}$. Also, as we remember, in our present treatment $\bigsqcap, \bigsqcup$ mean $\bigsqcap^{\mathfrak{b}}, \bigsqcup^{\mathfrak{b}}$, whereas in Japaridze (2006c, 2007b) they meant properly $\bigsqcap, \bigsqcup$. Such technical differences, however, are minor, and have no impact on the relevant proofs. So, the above-mentioned proof from Japaridze (2007b), with just a few rather straightforward adjustments, goes through as a proof of the completeness part of the present theorem as well.

For the soundness part, we extend the language of **CL4** by adding to it a new sort of nonlogical letters called **hybrid**. Each n-ary hybrid letter is a pair P_q, where P — called its **general component** — is an n-ary general letter, and q — called its **elementary component** — is a nonlogical n-ary elementary letter. And vice versa: for every pair (P, q) of letters of the above sort, we have an n-ary hybrid letter P_q. Formulas of this extended language, to which we will be referring as **hyperformulas**, are built in the same way as **CL4**-formulas, with the difference that now atoms can be of any of the three — elementary, general or hybrid — sorts. **Surface occurrence**, (elementary, general, hybrid)

literal and similar concepts straightforwardly extend from **CL3**- and **CL4**-formulas to hyperformulas. Furthermore, concepts such as surface occurrence, positive occurrence, etc. extend from subformulas to parts of subformulas, such as letters occurring in them, in the obvious way.

We say that a hyperformula E is a **CL4°-formula** iff, for every hybrid letter P_q occurring in E, the following conditions are satisfied:

(1) E has exactly two occurrences of P_q, where one occurrence is positive and the other occurrence is negative, and both occurrences are surface occurrences. We say that the corresponding two literals — where one looks like $P_q(\vec{\tau})$ and the other like $\neg P_q(\vec{\theta})$ — are **matching**.

(2) The elementary letter q does not occur in E, nor is it the elementary component of any hybrid letter occurring in E other than P_q.

Of course, every **CL4**-formula is also a **CL4**°-formula — one with no hybrid letters.

The **elementarization** $\|E\|$ of a **CL4**°-formula E is the result of replacing, in E, each surface occurrence of the form $G_1 \sqcap \ldots \sqcap G_n$ or $\sqcap x G$ by $\top$, each surface occurrence of the form $G_1 \sqcup \ldots \sqcup G_n$ or $\sqcup x G$ by $\bot$, every positive surface occurrence of each general literal by $\bot$, and every surface occurrence of each hybrid letter by the elementary component of that letter.

We are going to employ a "version" of **CL4** called **CL4**°. Unlike **CL4** whose language consists only of **CL4**-formulas, the language of **CL4**° allows any **CL4**°-formulas. The axioms and rules of **CL4**° are the same as those of **CL4** — only, now applied to any **CL4**°-formulas rather than just **CL4**-formulas — with the difference that the rule of Match is replaced by the following rule that we call **Match**°:

$$\frac{F[P_q(\vec{\tau}), \neg P_q(\vec{\theta})]}{F[P(\vec{\tau}), \neg P(\vec{\theta})]},$$

where P is any n-ary general letter, q is any n-ary elementary letter not occurring in the conclusion (neither independently nor as the elementary component of some hybrid letter), and $\vec{\tau}$,$\vec{\theta}$ are any n-tuples of terms.

Claim 1. *For any* **CL4**-*formula* E, *if* **CL4** $\vdash E$, *then* **CL4**° $\vdash E$.

Proof. The idea that underlies our proof of this claim is very simple: every application of Match naturally turns into an application of Match°.

Indeed, consider any **CL4**-proof of E. It can be seen as a tree all of the leaves of which are labeled with axioms and every non-leaf node of which is labeled with a formula that follows by one of the rules of **CL4** from (the labels of) its children, with E being the label of the root. By abuse of terminology, here we identify the nodes of this tree with their labels, even though, of course, it may be the case that different nodes have the same label. For each node G of the tree that is derived from its child H by Match — in particular, where H is the result of replacing in G a positive and a negative surface occurrences of an n-ary general letter P by an n-ary nonlogical elementary letter q — do the following: replace q by the hybrid letter P_q in H as well as in all of its descendants in the tree. It is not hard to see that this way we will get a **CL4**°-proof of E. ■

The concept of a **substitution** $^\heartsuit$ for a **CL4**°-formula E, and the corresponding **CL3**-formula $E^\heartsuit$, are defined in the same ways as for **CL4**-formulas, treating each hybrid letter P_q as a separate (not related to P or any other P_p with $p \neq q$) general letter.

We say that a **CL3**-formula E is a **TROW-premise** of a **CL3**-formula F ("TROW"="Transitive Reflexive Ordinary Wait") iff E is F, or an ordinary Wait-premise of F, or an ordinary Wait-premise of an ordinary Wait-premise of F, or . . . .

Let E be a **CL4**°-formula with exactly n positive surface occurrences of general literals, with those occurences being (not necessarily pairwise distinct literals) $G_1, \ldots, G_n$. And let $^\heartsuit$ be a substitution for E. Then $E^\heartsuit$ can obviously be written as $H[G_1{}^\heartsuit, \ldots, G_n{}^\heartsuit]$, where $G_1{}^\heartsuit, \ldots, G_n{}^\heartsuit$ are surface occurrences originating from the occurrences of $G_1, \ldots, G_n$ in E. Under these conditions, by a $^\heartsuit$**-quasiinstance** of E we will mean any TROW-premise of $H[G_1{}^\heartsuit, \ldots, G_n{}^\heartsuit]$ that can be written as $H[J_1, \ldots, J_n]$. To summarize in more intuitive terms, a $^\heartsuit$-quasiinstance of E is a TROW-premise of $E^\heartsuit$ where all (if any) changes have taken place exclusively in subformulas ($G_1{}^\heartsuit, \ldots, G_n{}^\heartsuit$) that originate from positive occurrences of general literals ($G_1, \ldots, G_n$) in E. Of course, $E^{\heartsuit}$ is one of the $^{\heartsuit}$-quasiinstances of E.

By a (simply) **quasiinstance** of a **CL4**$^{\circ}$-formula E we mean a $^{\heartsuit}$-quasiinstance of E for some substitution $^{\heartsuit}$ for E. Note that every instance is a quasiinstance but not necessarily vice versa.

Claim 2. *For any* **CL4**$^{\circ}$*-formula E, if* **CL4**$^{\circ} \vdash E$*, then every quasiinstance of E is provable in* **CL3**.

Proof. Consider any **CL4**$^{\circ}$-provable formula E. We want to show that **CL3** proves any quasiinstance of E. This will be done by induction on the length of the **CL4**$^{\circ}$-proof of E; within the inductive step of this induction, we will use a second induction — induction on the complexity (the number of logical connectives) of the quasiinstance of E under consideration. Call the first induction *primary* and the second induction *secondary*. These adjectives will also be applied to the corresponding inductive hypotheses.

For the basis of the primary induction, assume E is an axiom of **CL4**$^{\circ}$ (and hence of **CL3** as well), i.e. E is a valid formula of classical logic. Consider any substitution $^{\heartsuit}$ for E. The formula $E^{\heartsuit}$ is an axiom of (**CL4**$^{\circ}$ and) **CL3**, because classical validity is closed under applying substitutions. And, since E is elementary, $E^{\heartsuit}$ is the only $^{\heartsuit}$-quasiinstance of it. So, we are done.

Below comes the inductive step of the primary induction, divided into three cases.

Case 1. Assume E is obtained from a premise G by $\sqcup$-Choose or $\bigsqcup$-Choose. Consider any substitution $^{\heartsuit}$ for E. Obviously, $E^{\heartsuit}$ follows from $G^{\heartsuit}$ by the same rule;[12] and, by the primary induction hypothesis, **CL3** $\vdash G^{\heartsuit}$, so we have **CL3** $\vdash E^{\heartsuit}$. Furthermore, what we just observed extends to any other (other than $E^{\heartsuit}$) $^{\heartsuit}$-quasiinstance H of E as well: with some thought, one can see that such an H follows from a certain (the corresponding) $^{\heartsuit}$-quasiinstance of G by the same rule $\sqcup$-Choose or $\bigsqcup$-Choose as E follows from G.

Case 2. Assume E is obtained from premises $\|E\|, G_1, \ldots, G_n$ by Wait. Consider any substitution $^{\heartsuit}$ for E and any $^{\heartsuit}$-quasiinstance H of E. We want to show that H can be derived in **CL3** by Wait.

The provability of the elementary formula $\|E\|$ obviously means that it is an axiom, i.e., a valid formula of classical logic. Let $J_1, \ldots, J_k$ be all

positive surface occurrences of general literals in E, and let E' be the formula obtained from E by replacing those occurrences by $q_1, \ldots, q_k$, where the q_i are pairwise distinct 0-ary elementary letters not occurring in E. Observe that then $\|E'\|$ differs from $\|E\|$ in that, where the former has k positive occurrences of $\bot$ (originating from $J_1, \ldots, J_k$ when elementarizing E), the latter has the k atoms $q_1, \ldots, q_k$. It is known from classical logic that replacing positive occurrences of $\bot$ by whatever formulas does not destroy validity. Hence, as $\|E\|$ is valid, so is $\|E'\|$. Now, with some analysis, details of which are left to the reader, one can see that the formula $\|H\|$ is a substitutional instance — in both our present sense as well as in the classical sense — of $\|E'\|$. So, as an instance of a classically valid formula, $\|H\|$ is classically valid, i.e. is an axiom of **CL3**, and we thus have

$$\mathbf{CL3} \vdash \|H\|. \tag{5}$$

We now want to show that:

$$\textit{Whenever } H = H[K_1 \sqcap \ldots \sqcap K_m] \textit{ and } 1 \leq i \leq m\textit{, we have } \mathbf{CL3} \vdash H[K_i]. \tag{6}$$

Indeed, assuming the conditions of (6), one of the following should be the case:

(1) The occurrence of $K_1 \sqcap \ldots \sqcap K_m$ in H originates from a (surface) occurrence of a subformula $L_1 \sqcap \ldots \sqcap L_m$ in E (so that $K_1 = L_1{}^{\heartsuit}$, ..., $K_m = L_m{}^{\heartsuit}$). Then, obviously, $H[K_i]$ is a $^{\heartsuit}$-quasiinstance of one of the ordinary Wait-premises G_j ($1 \leq j \leq n$) of E. But then, by the primary induction hypothesis, we have **CL3** $\vdash H[K_i]$.

(2) The occurrence of $K_1 \sqcap \ldots \sqcap K_m$ in H originates from a (positive surface) occurrence of some general literal L in E (so that $K_1 \sqcap \ldots \sqcap K_m$ has a surface occurrence in a TROW-premise of $L^{\heartsuit}$). Note that then $H[K_i]$, just like H, is a $^{\heartsuit}$-quasiinstance of E. By the secondary induction hypothesis, the formula $H[K_i]$, as a quasiinstance of E less complex than H itself, is provable in **CL3**.

(3) The occurrence of $K_1 \sqcap \ldots \sqcap K_m$ in H originates from a (positive surface) occurrence of some hybrid literal L in E (so that $K_1 \sqcap \ldots \sqcap K_m$ has a surface occurrence in $L^{\heartsuit}$). Then $H[K_i]$ contains a surface occurrence of the subformula $\neg K_1 \sqcup \ldots \sqcup \neg K_m$, originating from the occurrence of the matching hybrid literal L'

in E. Let H' be the result of replacing that $\neg K_1 \sqcup \ldots \sqcup \neg K_m$ by $\neg K_i$ in $H[K_i]$. Obviously H', just like H, is a quasiinstance of E, but it is less complex than H. Hence, by the secondary induction hypothesis, **CL3** $\vdash H'$. But $H[K_i]$ follows from H' by $\sqcup$-Choose. So, **CL3** $\vdash H[K_i]$.

In all cases we thus get **CL3** $\vdash H[K_i]$, as desired.

In a very similar way, we can further show that

Whenever $H = H[\sqcap x K(x)]$, *we have* **CL3** $\vdash H[K(s)]$
for some variable s not occurring in H. (7)

Now, from (5), (6) and (7), by Wait, we find the desired **CL3** $\vdash H$.

Case 3. Suppose P is a k-ary general letter, q is a k-ary nonlogical elementary letter, $\tau_1, \ldots, \tau_k$, $\theta_1, \ldots, \theta_k$ are terms,

$$E = E[P(\tau_1, \ldots, \tau_k), \neg P(\theta_1, \ldots, \theta_k)]$$

and it is obtained from the premise

$$E[P_q(\tau_1, \ldots, \tau_k), \neg P_q(\theta_1, \ldots, \theta_k)] \tag{8}$$

by Match°. Consider any substitution $^\heartsuit$ for E, and any $^\heartsuit$-quasiinstance of E. Obviously such a quasiinstance can be written in the form

$$H[K_1(\tau_1{}^\heartsuit, \ldots, \tau_k{}^\heartsuit), \neg K_2(\theta_1{}^\heartsuit, \ldots, \theta_k{}^\heartsuit)], \tag{9}$$

where H inherits the logical structure of E (but probably adds some extra complexity to it), $K_1(\tau_1{}^\heartsuit, \ldots, \tau_k{}^\heartsuit)$ is a TROW-premise of $P^\heartsuit(\tau_1{}^\heartsuit, \ldots, \tau_k{}^\heartsuit)$ and $\neg K_2(\theta_1{}^\heartsuit, \ldots, \theta_k{}^\heartsuit)$ is a TROW-premise of $\neg P^\heartsuit(\theta_1{}^\heartsuit, \ldots, \theta_k{}^\heartsuit)$. With a little thought, one can see that there is a series of $\sqcup$-Chooses and $\bigsqcup$-Chooses that we can apply — in the bottom-up sense — to (9) to "even out" the $K_1(\tau_1{}^\heartsuit, \ldots, \tau_k{}^\heartsuit)$ and $\neg K_2(\theta_1{}^\heartsuit, \ldots, \theta_k{}^\heartsuit)$ subformulas and bring (9) to

$$H[K(\tau_1{}^\heartsuit, \ldots, \tau_k{}^\heartsuit), \neg K(\theta_1{}^\heartsuit, \ldots, \theta_k{}^\heartsuit)] \tag{10}$$

for a certain formula $K(x_1, \ldots, x_n)$. Let $^\diamondsuit$ be the substitution for E which sends P_q to $K(x_1, \ldots, x_n)$ and agrees with $^\heartsuit$ on everything else.

With a little thought, we can see that (10) is a $^\diamondsuit$-quasiinstance of (8). Hence, by the primary induction hypothesis, **CL3** $\vdash$ (10). Now, as we already know, (9) is obtained from (10) using a series of $\sqcup$-Chooses and $\bigsqcup$-Chooses. Hence (9) — which, as we remember, was an arbitrary quasiinstance of E — is provable in **CL3**.

The above Cases 1,2,3 complete the inductive step of our primary induction, and we conclude that, whenever E is a **CL4°**-provable formula, every quasiinstance of it is provable in **CL3**. ■

To complete our proof of (the soundness part of) Theorem 11.4, assume **CL4** $\vdash F$. Then, by Claim 1, **CL4°** $\vdash F$. Consider any substitution $^\heartsuit$ for F. $F^\heartsuit$ is a (quasi)instance of F and hence, by Claim 2, **CL3** $\vdash F^\heartsuit$. Since both F and $^\heartsuit$ are arbitrary, we conclude that every instance of every **CL4**-provable formula is provable in **CL3**. ■

12. THE BASIC SYSTEM OF PTARITHMETIC INTRODUCED

There can be various interesting systems of arithmetic based on computability logic ("**clarithmetics**"), depending on what language we consider, what fragment of CL is taken as a logical basis, and what extra-logical rules and axioms are employed. Japaridze (2010) introduced three systems of clarithmetic, named **CLA1**, **CLA2** and **CLA3**, all based on the fragment **CL12** (also introduced in Japaridze 2010) of computability logic. The basic one of them is **CLA1**, with the other two systems being straightforward modifications of it through slightly extending (**CLA2**) or limiting (**CLA3**) the underlying nonlogical language. Unlike our present treatment, the underlying semantical concept for the systems of Japaridze (2010) was computability-in-principle rather than efficient computability.

The new system of clarithmetic introduced in this section, meant to axiomatize efficient computability of number-theoretic computational problems, is named **PTA**. The term "**ptarithmetic**" is meant to be a generic name for systems in this style, even though we often use it to refer to our present particular system **PTA** of ptarithmetic.

The language of **PTA**, whose formulas we refer to as **PTA-formulas**, is obtained from the language of **CL3** by removing all nonlogical predicate letters (thus only leaving the logical predicate letter $=$), and also removing all but four function letters, which are:

- $zero$, 0-ary. We will write $\mathbf{0}$ for $zero$.
- $successor$, unary. We will write τ' for $successor(\tau)$.
- sum, binary. We will write $\tau_1+\tau_2$ for $sum(\tau_1,\tau_2)$.
- $product$, binary. We will write $\tau_1\times\tau_2$ for $product(\tau_1,\tau_2)$.

From now on, when we just say "formula", we mean "**PTA**-formula", unless otherwise specified or suggested by the context.

Formulas that have no free occurrences of variables are said to be **sentences**.

The concept of an interpretation explained earlier can now be restricted to interpretations that are only defined on $\mathbf{0}$, $'$, $+$, $\times$ and $=$, as the present language has no other nonlogical function or predicate letters. Of such interpretations, the **standard interpretation** † is the one that interprets $\mathbf{0}$ as (the 0-ary function whose value is) 0, interprets $'$ as the standard successor ($x+1$) function, interprets $+$ as the sum function, interprets $\times$ as the product function, and interprets $=$ as the identity relation. Where F is a **PTA**-formula, the **standard interpretation of** F is the game $F^\dagger$, which we typically write simply as F unless doing so may cause ambiguity.

The **axioms** of **PTA** are grouped into logical and nonlogical.

The **logical axioms** of **PTA** are all elementary **PTA**-formulas provable in classical first-order logic. That is, all axioms of **CL3** that are **PTA**-formulas.

As for the **nonlogical axioms**, they are divided into what we call "Peano" and "extra-Peano" axioms.

The **Peano axioms** of **PTA** are all sentences matching the following seven schemes,[13] with $x,y,y_1,\dots,y_n$ being any pairwise distinct variables other than $\mathfrak{b}$:

Axiom 1: $\forall x(\mathbf{0}\neq x')$

Axiom 2: $\forall x\forall y(x'=y'\rightarrow x=y)$

Axiom 3: $\forall x(x+\mathbf{0}=x)$

Axiom 4: $\forall x\forall y\big(x+y'=(x+y)'\big)$

Axiom 5: $\forall x(x\times\mathbf{0}=\mathbf{0})$

Axiom 6: $\forall x\forall y\big(x\times y'=(x\times y)+x\big)$

Axiom 7: $\forall y_1\dots\forall y_n\Big(F(\mathbf{0})\wedge\forall x\big(F(x)\rightarrow F(x')\big)\rightarrow\forall xF(x)\Big)$, where $F(x)$ is any elementary formula and $y_1,\dots,y_n$ are all of the variables occurring free in it and different from $\mathfrak{b},x$.

Before we present the extra-Peano axioms of **PTA**, we need to agree on some notational matters. The language of **PTA** extends that of **Peano Arithmetic PA** (see, for example, Hajek & Pudlak 1993) through adding to it $\sqcap$, $\sqcup$, $\bigsqcup$, $\bigsqcap$. And the language of **PA** is known to be very expressive, despite its nonlogical vocabulary officially being limited to only $\mathbf{0}, ', +, \times$. Specifically, it allows us to express, in a certain reasonable and standard way, all recursive functions and relations, and beyond. Relying on the common knowledge of the power of the language of **PA**, we will be using standard expressions such as $x\leq y$, $y>x$, etc. in formulas as abbreviations of the corresponding proper expressions of the language. Namely, in our metalanguage, $|x|$ will refer to the length of (the binary numeral for the number represented by) x.[14] So, when we write, say, "$|x|\leq\mathfrak{b}$", it is to be understood as an abbreviation of a standard formula of **PA** saying that the size of x does not exceed $\mathfrak{b}$.

Where τ is a term, we will be using $\tau0$ and $\tau1$ as abbreviations for the terms $\mathbf{0}''\times\tau$ and $(\mathbf{0}''\times\tau)'$, respectively. The choice of this notation is related to the fact that, given any natural number a, the binary representation of $\mathbf{0}''\times a$ (i.e., of $2a$) is nothing but the binary representation of a with a "0" added on its right. Similarly, the binary representation of $(\mathbf{0}''\times a)'$ is nothing but the binary representation of a with a "1" added to it. Of course, here an exception is the case $a=0$. It can be made an ordinary case by assuming that adding any number of 0s at the beginning of a binary numeral b results in a legitimate numeral representing the same number as b.

The number $a0$ (i.e. $2a$) will be said to be the **binary** 0**-successor** of a, and $a1$ (i.e. $2a+1$) said to be the **binary** 1**-successor** of a; in turn, we can refer to a as the **binary predecessor** of $a0$ and $a1$. As for a', we can refer to it as the **unary successor** of a, and refer to a as the **unary predecessor** of a'. Every number has a binary predecessor, and every number except 0 has a unary predecessor. Note that the binary predecessor of a number is the result of deleting the last digit in its binary representation. Two exceptions are the numbers 0 and 1, both having 0 as their binary predecessor.

Below and elsewhere, by a **$\mathfrak{b}$-term** we mean a term of the official language of **PTA** containing no variables other than $\mathfrak{b}$. That is, a term exclusively built from $\mathfrak{b}, \mathbf{0}, {}', +, \times$.

Now, the **extra-Peano axioms** of **PTA** are all formulas matching the following six schemes, where s is any variable and x is any variable other than $\mathfrak{b}, s$:

Axiom 8: $\sqcup x(x=\mathbf{0})$

Axiom 9: $s=\mathbf{0} \sqcup s\neq\mathbf{0}$

Axiom 10: $|s'|\leq\mathfrak{b} \rightarrow \sqcup x(x=s')$

Axiom 11: $|s0|\leq\mathfrak{b} \rightarrow \sqcup x(x=s0)$

Axiom 12: $\sqcup x(s=x0 \sqcup s=x1)$

Axiom 13: $|s|\leq\mathfrak{b}$

The **rules of inference** are also divided into two groups: logical and nonlogical.

The **logical rules** of **PTA** are the rules $\sqcup$-Choose, $\sqcup$-Choose, Wait and Modus Ponens of Section 9.

And there is a single **nonlogical rule** of inference, that we call **Polynomial Time Induction** (**PTI**), in which τ is any $\mathfrak{b}$-term, s is any non-$\mathfrak{b}$ variable, and $E(s), F(s)$ are any formulas:

PTI

$$\frac{E(\mathbf{0})\wedge F(\mathbf{0}) \qquad\qquad E(s)\wedge F(s)\rightarrow E(s')\sqcap\big(F(s')\wedge E(s)\big)}{s\leq\tau\rightarrow E(s)\wedge F(s)}$$

Here the left premise is called the **basis** of induction, and the right premise called the **inductive step**.

A formula F is considered **provable** in **PTA** iff there is a sequence of formulas, called a **PTA-proof** of F, where each formula is either a (logical or nonlogical) axiom, or follows from some previous formulas by one of the (logical or nonlogical) rules of inference, and where the last formula is F. We write $\mathbf{PTA}\vdash F$ to say that F is provable (has a proof) in **PTA**, and $\mathbf{PTA}\nvdash F$ to say the opposite.

In view of the following fact, an alternative way to present **PTA** would be to delete Axioms 1-7 together with all logical axioms and, instead, declare all theorems of **PA** to be axioms of **PTA** along with Axioms 8-13:

Fact 12.1 *Every (elementary* **PTA**-*) formula provable in* **PA** *is also provable in* **PTA**.

Proof. Suppose (the classical-logic-based) **PA** proves F. By the deduction theorem for classical logic this means that, for some nonlogical axioms $H_1,\ldots,H_n$ of **PA**, the formula $H_1\wedge\ldots\wedge H_n\rightarrow F$ is provable in classical first order logic. Hence $H_1\wedge\ldots\wedge H_n\rightarrow F$ is a logical axiom of **PTA** and is thus provable in **PTA**. But the nonlogical axioms of **PA** are nothing but the Peano axioms of **PTA**. So, **PTA** proves each of the formulas $H_1,\ldots,H_n$. Now, in view of the presence of the rule of Modus Ponens in **PTA**, we find that $\mathbf{PTA}\vdash F$. ■

The above fact, on which we will be implicitly relying in the sequel, allows us to construct “lazy” **PTA**-proofs where some steps can be justified by simply indicating their provability in **PA**. That is, we will treat theorems of **PA** as if they were axioms of **PTA**. As **PA** is well known and studied, we safely assume that the reader has a good feel of what it can prove, so we do not usually further justify **PA**-provability claims that we make. A reader less familiar with **PA**, can take it as a rule of thumb that, despite Gödel’s incompleteness theorems, **PA** proves every true number-theoretic fact that a contemporary high school student can establish, or that mankind was or could be aware of before 1931.

Definition 12.2

1. By an **arithmetical problem** in this paper we mean a game A such that, for some formula F of the language of **PTA**, $A=F^{\dagger}$ (remember that † is the standard interpretation). Such a formula F is said to be a **representation** of A.

2. We say that an arithmetical problem A is **provable** in **PTA** iff it has a **PTA**-provable representation.

In these terms, the central result of the present paper sounds as follows:

Theorem 12.3 *An arithmetical problem has a polynomial time solution iff it is provable in* **PTA**.

Furthermore, there is an effective procedure that takes an arbitrary **PTA**-*proof of an arbitrary formula X and constructs a polynomial time solution for X (for $X^{\dagger}$, that is).*

Proof. The soundness ("if") part of this theorem will be proven in Section 15, and the completeness ("only if") part in Section 21. ■

13. ON THE EXTRA-PEANO AXIOMS OF PTA

While the well known Peano axioms hardly require any explanations as their traditional meanings are fully preserved in our treatment, the extra-Peano axioms of **PTA** may be worth briefly commenting on. Below we do so with the soundness of **PTA** (the "if" part of Theorem 12.3) in mind, according to which every **PTA**-provable formula expresses an efficiently (i.e. polynomial time) computable number-theoretic problem.

13.1. Axiom 8

$$\sqcup x(x=\mathbf{0})$$

This axiom expresses our ability to efficiently name the number (constant) 0. Nothing — even such a "trivial" thing — can be taken for granted when it comes to formal systems!

13.2. Axiom 9

$$s=\mathbf{0}\sqcup s\neq\mathbf{0}$$

This axiom expresses our ability to efficiently tell whether any given number is 0 or not. Yet another "trivial" thing that still has to be explicitly stated in the formal system.

13.3. Axiom 10

$$|s'|\leq\mathfrak{b}\rightarrow\sqcup x(x=s')$$

This axiom establishes the efficient computability of the unary successor function (as long as the size of the value of the function does not exceed the bound $\mathfrak{b}$). Note that its classical counterpart $|s'|\leq\mathfrak{b}\rightarrow\exists x(x=s')$ is simply a valid formula of classical first-order logic (because so is its consequent) and, as such, carries no information. Axiom 10, on the other hand, is not at all a logically valid formula, and does carry certain nontrivial information about the standard meaning of the successor function. A nonstandard meaning (interpretation) of s' could be an intractable or even incomputable function.

13.4. Axiom 11

$$|s0|\leq\mathfrak{b}\rightarrow\sqcup x(x=s0)$$

Likewise, Axiom 11 establishes the efficient computability of the binary 0-successor function. There is no need to state a similar axiom for the binary 1-successor function, as can be seen from the following lemma:

Lemma 13.1 **PTA** $\vdash|s1|\leq\mathfrak{b}\rightarrow\sqcup x(x=s1)$.

Proof. Informally, a proof of $|s1|\leq\mathfrak{b}\rightarrow\sqcup x(x=s1)$ would be based on the fact (known from **PA**) that the binary 1-successor of s is nothing but the unary successor of the binary 0-successor of s; the binary 0-successor r of s can be found using Axiom 11; and the unary successor u of that r can be further found using Axiom 10.

Here is a ("lazy" in the earlier-mentioned sense) **PTA**-proof formalizing the above argument:

1. $\top\wedge(|s0|\leq\mathfrak{b}\rightarrow\bot)\rightarrow(|s1|\leq\mathfrak{b}\rightarrow\bot)$ **PA**
2. $\top$ Logical axiom

3. $|s'| \leq \mathfrak{b} \to \sqcup x(x = s')$ Axiom 10
4. $\sqcap y\big(|y'| \leq \mathfrak{b} \to \sqcup x(x = y')\big)$ Wait: 2,3
5. $|s0| \leq \mathfrak{b} \to \sqcup x(x = s0)$ Axiom 11
6. $\big(|t'| \leq \mathfrak{b} \to \bot\big) \wedge \big(|s0| \leq \mathfrak{b} \to (t = s0)\big) \to \big(|s1| \leq \mathfrak{b} \to \bot\big)$ **PA**
7. $\big(|t'| \leq \mathfrak{b} \to r = t'\big) \wedge \big(|s0| \leq \mathfrak{b} \to (t = s0)\big) \to \big(|s1| \leq \mathfrak{b} \to r = s1\big)$ **PA**
8. $\big(|t'| \leq \mathfrak{b} \to r = t'\big) \wedge \big(|s0| \leq \mathfrak{b} \to (t = s0)\big) \to \big(|s1| \leq \mathfrak{b} \to \sqcup x(x = s1)\big)$ $\sqcup$-Choose: 7
9. $\big(|t'| \leq \mathfrak{b} \to \sqcup x(x = t')\big) \wedge \big(|s0| \leq \mathfrak{b} \to (t = s0)\big) \to \big(|s1| \leq \mathfrak{b} \to \sqcup x(x = s1)\big)$ Wait: 6,8
10. $\sqcap y\big(|y'| \leq \mathfrak{b} \to \sqcup x(x = y')\big) \wedge \big(|s0| \leq \mathfrak{b} \to (t = s0)\big) \to \big(|s1| \leq \mathfrak{b} \to \sqcup x(x = s1)\big)$ $\sqcup$-Choose: 9
11. $\sqcap y\big(|y'| \leq \mathfrak{b} \to \sqcup x(x = y')\big) \wedge \big(|s0| \leq \mathfrak{b} \to \sqcup x(x = s0)\big) \to \big(|s1| \leq \mathfrak{b} \to \sqcup x(x = s1)\big)$ Wait: 1,10
12. $|s1| \leq \mathfrak{b} \to \sqcup x(x = s1)$ MP: 4,5,11 ■

This was our first experience with generating a formal **PTA**-proof. We will do a good deal more exercising with **PTA**-proofs later in order to make it apparent that behind every informal argument in the style of the one given at the beginning of the proof of Lemma 13.1 is a "real", formal proof.

13.5. Axiom 12

$$\sqcup x(s = x0 \sqcup s = x1) \tag{11}$$

Let us compare the above with three other, "similar" formulas:

$$\exists x(s = x0 \sqcup s = x1) \tag{12}$$

$$\sqcup x(s = x0 \vee s = x1) \tag{13}$$

$$\exists x(s = x0 \vee s = x1) \tag{14}$$

All four formulas "say the same" about the arbitrary number represented by s, but in different ways. (14) is the weakest, least informative, of the four. It says that s has a binary predecessor x, and that

s is even (i.e., is the binary 0-successor of its binary predecessor) or odd (i.e., is the binary 1-successor of its binary predecessor). This is an almost trivial piece of information. (13) and (12) carry stronger information. According to (13), s not only *has* a binary predecessor x, but such a predecessor can be actually and efficiently *found*. (12) strengthens (14) in another way. It says that s can be efficiently determined to be even or odd. As for (11), which is Axiom 12 proper, it is the strongest. It carries two pieces of good news at once: we can efficiently find the binary predecessor x of s and, simultaneously, tell whether s is even or odd.

13.6. Axiom 13

$$|s| \leq \mathfrak{b}$$

Remember that our semantics considers only bounded valuations, meaning that the size of the number represented by a (free) variable s will never exceed the bound represented by the variable $\mathfrak{b}$. Axiom 13 simply states this fact. Note that this is the only elementary formula among the extra-Peano axioms.

In view of the above-said, whenever we say "an arbitrary s" in an informal argument, unless otherwise suggested by the context, it is always to be understood as an arbitrary s whose size does not exceed the bound $\mathfrak{b}$.

Due to Axiom 13, **PTA** proves that the bound is nonzero:

Lemma 13.2 **PTA** $\vdash \mathfrak{b} \neq \mathbf{0}$.

Proof. No binary numeral is of length 0 and, of course, **PA** knows this. Hence **PA** $\vdash |s| \leq \mathfrak{b} \to \mathfrak{b} \neq \mathbf{0}$. From here and Axiom 13, by Modus Ponens, **PTA** $\vdash \mathfrak{b} \neq \mathbf{0}$. ■

The formula of the following lemma is similar to Axiom 8, only it is about $\mathbf{0}'$ instead of $\mathbf{0}$.

Lemma 13.3 **PTA** $\vdash \sqcup x(x = \mathbf{0}')$.

Proof.

1. $\mathfrak{b} \neq \mathbf{0}$ Lemma 13.2
2. $\sqcup x(x = \mathbf{0})$ Axiom 8
3. $\top$ Logical axiom
4. $|s'| \leq \mathfrak{b} \rightarrow \sqcup x(x = s')$ Axiom 10
5. $\sqcap y\big(|y'| \leq \mathfrak{b} \rightarrow \sqcup x(x = y')\big)$ Wait: 3,4
6. $\mathfrak{b} \neq \mathbf{0} \wedge \bot \wedge \top \rightarrow \bot$ Logical axiom
7. $\mathfrak{b} \neq \mathbf{0} \wedge w = \mathbf{0} \wedge (|w'| \leq \mathfrak{b} \rightarrow \bot) \rightarrow \bot$ **PA**
8. $\mathfrak{b} \neq \mathbf{0} \wedge w = \mathbf{0} \wedge (|w'| \leq \mathfrak{b} \rightarrow v = w') \rightarrow v = \mathbf{0}'$ **PA**
9. $\mathfrak{b} \neq \mathbf{0} \wedge w = \mathbf{0} \wedge (|w'| \leq \mathfrak{b} \rightarrow v = w') \rightarrow \sqcup x(x = \mathbf{0}')$ $\sqcup$-Choose: 8
10. $\mathfrak{b} \neq \mathbf{0} \wedge w = \mathbf{0} \wedge \big(|w'| \leq \mathfrak{b} \rightarrow \sqcup x(x = w')\big) \rightarrow \sqcup x(x = \mathbf{0}')$ Wait: 7,9
11. $\mathfrak{b} \neq \mathbf{0} \wedge w = \mathbf{0} \wedge \sqcap y\big(|y'| \leq \mathfrak{b} \rightarrow \sqcup x(x = y')\big) \rightarrow \sqcup x(x = \mathbf{0}')$ $\sqcup$-Choose: 10
12. $\mathfrak{b} \neq \mathbf{0} \wedge \sqcup x(x = \mathbf{0}) \wedge \sqcap y\big(|y'| \leq \mathfrak{b} \rightarrow \sqcup x(x = y')\big) \rightarrow \sqcup x(x = \mathbf{0}')$ Wait: 6,11
13. $\sqcup x(x = \mathbf{0}')$ MP: 1,2,5,12 ■

14. ON THE POLYNOMIAL TIME INDUCTION RULE

$$\frac{E(\mathbf{0}) \wedge F(\mathbf{0}) \qquad\qquad E(s) \wedge F(s) \rightarrow E(s') \sqcap \big(F(s') \wedge E(s)\big)}{s \leq \tau \rightarrow E(s) \wedge F(s)}$$

Induction is the cornerstone of every system of arithmetic. The many versions of formal arithmetic studied in the literature (see Hajek & Pudlak 1993) mainly differ in varying — typically weakening — the unrestricted induction of the basic **PA**, which is nothing but our Axiom 7. In **PTA**, induction comes in two forms: Axiom 7, and the above-displayed PTI rule. Axiom 7, along with the other axioms of **PA**, is taken to preserve the full power of **PA**. But it is limited to elementary formulas and offers no inductive mechanism applicable to computational problems in general. The role of PTI is to provide such a missing mechanism.

A naive attempt to widen the induction of **PA** would be to remove, from Axiom 7, the condition requiring that $F(x)$ be an elementary formula. This would be a terribly wrong idea though. The resulting scheme would not even be a scheme of computable problems, let alone efficiently computable problems. Weakening the resulting scheme by additionally replacing the blind quantifiers with choice quantifiers, resulting in (a scheme equivalent to)

$$F(\mathbf{0}) \wedge \sqcap x\big(F(x) \rightarrow F(x')\big) \rightarrow \sqcap x F(x), \tag{15}$$

would not fix the problem, either. The intuitive reason why (15) is unsound with respect to the semantics of computability logic, even if the underlying concept of interest is computability-in-principle without any regard for efficiency, is the following. In order to solve $F(s)$ for an arbitrary s (i.e., solve the problem $\sqcap x F(x)$), one would need to "modus-ponens" $F(x) \rightarrow F(x')$ with $F(\mathbf{0})$ to compute $F(1)$, then further "modus-ponens" $F(x) \rightarrow F(x')$ with $F(1)$ to compute $F(2)$, etc. up to $F(s)$. This would thus require s "copies" of the resource $F(x) \rightarrow F(x')$. But the trouble is that only one copy of this resource is available in the antecedent of (15)!

The problem that we just pointed out can be neutralized by taking the following **rule** instead of the *formula* (scheme) (15):

$$\frac{F(\mathbf{0}) \qquad\qquad \sqcap x\big(F(x) \rightarrow F(x')\big)}{\sqcap x F(x)}.$$

Taking into account that both semantically and syntactically $\sqcap x Y(x)$ (in isolation) is equivalent to just $Y(s)$, we prefer to rewrite the above in the following form:

$$\frac{F(\mathbf{0}) \qquad\qquad F(s) \rightarrow F(s')}{F(s)}. \tag{16}$$

Unlike the situation with (15), the resource $F(s) \rightarrow F(s')$ comes in an unlimited supply in (16). As a rule, (16) assumes that the premise $F(s) \rightarrow F(s')$ has already been proven. If proven, we know how to solve it. And if we know how to solve it, we can solve it as many times as needed. In contrast, in the case of (15) we do not really know how to solve the corresponding problem of the antecedent, but rather we rely on the environment to demonstrate such a solution; and the environment is obligated to do so only once.

(16) can indeed be shown to be a computability-preserving rule. As we remember, however, we are concerned with efficient computability rather than computability-in-principle. And, in this respect, (16) is not sound. Roughly, the reason is the following: the way of computing $F(s)$ offered by (16) would require performing at least as many MP-style steps as the numeric value of s (rather than the dramatically smaller *size* of s). This would yield a computational complexity exponential in the size of s. (16) can be made sound by limiting s to "sufficiently small" numbers as done below, where τ is an arbitrary $\mathfrak{b}$-term:

$$\frac{F(\mathbf{0}) \qquad\qquad F(s)\rightarrow F(s')}{s\leq\tau\rightarrow F(s)}. \tag{17}$$

Here the value of τ, being a $(\mathfrak{b},\mathbf{0},{}',+,\times)$-combination, is guaranteed to be polynomial in (the value of) $\mathfrak{b}$. Hence, we are no longer getting an exponential complexity of computation. This, by the way, explains the presence of "$s\leq\tau$" in the conclusion of PTI. Unlike (15) and (16), (17) is indeed sound with respect to our present semantics of efficient computability.

A problem with (17), however, is that it is not strong enough — namely, not as strong as PTI, and with (17) instead of PTI, we cannot achieve the earlier promised extensional completeness of **PTA**. What makes PTI stronger than (17) is that its right premise is weaker. Specifically, while the right premise of (17) requires the ability to compute $F(s')$ only using $F(s)$ as a computational resource, the right premise of PTI allows using the additional resource $E(s)$ in such a computation.

Note that, in a classical context, identifying the two sorts of conjunction, there would be no difference between (17) and PTI. First of all, the (sub)conjunct $E(s)$ in the consequent of the right premise of PTI would be meaningless and hence could be deleted, as it is already present in the antecedent. Second, the conjunction of $E(s)$ and $F(s)$ could be thought of as one single formula of induction, and thus PTI would become simply (17).

Our context is not classical though, and the difference between PTI and (17) is huge. First of all, we cannot think of "the conjunction" of $E(s)$ and $F(s)$ as a single formula of induction, for that

"conjunction" is $\sqcap$ in the consequent of the right premise while $\wedge$ elsewhere. For simplicity, consider the case $E(s)=F(s)$. Also, let us ignore the technicality imposed by the presence of "$E(s)$" in the consequent of the right premise of PTI. Then that premise would look like $F(s)\wedge F(s)\rightarrow F(s')\sqcap F(s')$ which, taking into account that $X\sqcap X$ is equivalent to X, would be essentially the same as simply $F(s)\wedge F(s)\rightarrow F(s')$. This is a much weaker premise than the premise $F(s)\rightarrow F(s')$ of (17). It signifies that computing a single copy of $F(s')$ requires computing two copies of $F(s)$. By back-propagating this effect, it would eventually mean that computing $F(s)$ requires computing an exponential number of copies of $F(\mathbf{0})$, even when s is "small enough" such as $s\leq\tau$.

The above sort of an explosion is avoided in PTI due to the presence of $E(s)$ in the consequent of the right premise — the "technical detail" that we have ignored so far. The reemergence of $E(s)$ in the consequent of that premise makes this resource "recyclable". Even though computing $F(s')$ still requires computing *both* $E(s)$ *and* $F(s)$, a new copy of $E(s)$ comes "for free" as a side-product of this computation, and hence can be directly passed to another, parallel computation of $F(s')$. Such and all other parallel computations would thus require a new copy of $F(s)$ but not a new copy of $E(s)$, as they get the required resource $E(s)$ from the neighboring computation. So, a group of n parallel computations of $F(s')$ would require n copies of $F(s)$ and only one copy of $E(s)$. This essentially cuts the demand on resources at each step (for each s) by half, and the eventual number of copies of $E(\mathbf{0})\wedge F(\mathbf{0})$ to be computed will be of the order of s rather than 2^s. How this effect is exactly achieved will be clear after reading the following section.

15. THE SOUNDNESS OF PTA

This section is devoted to proving the soundness part of Theorem 12.3. It means showing that any **PTA**-provable formula X (identified with its standard interpretation $X^{\dagger}$) has a polynomial time solution, and that, furthermore, such a solution for X can be effectively extracted from any **PTA**-proof of X.

We prove the above by induction on the lengths of **PTA**-proofs. Consider any **PTA**-provable formula X.

For the basis of induction, assume X is an axiom of **PTA**. Let us say that an elementary **PTA**-formula G is **true** iff, for any bounded valuation e, $e[G]$ is true in the standard arithmetical sense, i.e., $\mathbf{Wn}^{G^\dagger}{}_e\langle\rangle = \top$.

If X is a logical axiom or a Peano axiom, then it is a true elementary formula and therefore is "computed" by a machine that makes no moves at all. The same holds for the case when X is Axiom 13, remembering that, for any bounded valuation e, the size of $e(s)$ (whatever variable s) never exceeds $e(\mathfrak{b})$.

If X is $\bigsqcup x(x=\mathbf{0})$ (Axiom 8), then it is computed by a machine that makes the move 0 and never makes any moves after that.

If X is $s=\mathbf{0}\sqcup s\neq\mathbf{0}$ (Axiom 9), then it is computed by a machine that reads the value $e(s)$ of s from the valuation tape and, depending on whether that value is 0 or not, makes the move 0 or 1, respectively.

If X is $|s'|\leq\mathfrak{b}\to\bigsqcup x(x=s')$ (Axiom 10), it is computed by a machine that reads the value $e(s)$ of s from the valuation tape, then finds (the binary numeral) c with $c=e(s)+1$, compares its size with $e(\mathfrak{b})$ (the latter also read from the valuation tape) and, if $|c|\leq e(\mathfrak{b})$, makes $1.c$ as its only move in the game.

Similarly, if X is $|s0|\leq\mathfrak{b}\to\bigsqcup x(x=s0)$ (Axiom 11), it is computed by a machine that reads the value $e(s)$ of s from the valuation tape, then finds (the binary numeral) c with $c=e(s)0$, compares its size with $e(\mathfrak{b})$ and, if $|c|\leq e(\mathfrak{b})$, makes $1.c$ as its only move in the game.

Finally, if X is $\bigsqcup x(s=x0\sqcup s=x1)$ (Axiom 12), it is computed by a machine that reads the value $e(s)$ of s from the valuation tape, then finds the binary predecessor c of $e(s)$, and makes the two moves c and 0 or c and 1, depending whether the last digit of $e(s)$ is 0 or 1, respectively.

Needless to point out that, in all of the above cases, the machines that solve the axioms run in polynomial time. And, of course, such machines can be constructed effectively.

For the inductive step, suppose X is obtained from premises $X_1,\ldots,X_k$ by one of the four logical rules. By the induction hypothesis, we know how to (effectively) construct a polynomial time solution for each X_i. Then, by the results of Section 9 on the uniform-constructive soundness of the four logical rules, we also know how to construct a polynomial time solution for X.

Finally, suppose X is $s\leq\tau\to E(s)\wedge F(s)$, where τ is a $\mathfrak{b}$-term, and X is obtained by PTI as follows:

$$\frac{E(\mathbf{0})\wedge F(\mathbf{0}) \qquad E(s)\wedge F(s)\to E(s')\sqcap\big(F(s')\wedge E(s)\big)}{s\leq\tau\to E(s)\wedge F(s)}.$$

By the induction hypothesis, the following two problems have polynomial time solutions — and, furthermore, we know how to construct such solutions:

$$E(\mathbf{0})\wedge F(\mathbf{0}); \tag{18}$$

$$E(s)\wedge F(s)\to E(s')\sqcap\big(F(s')\wedge E(s)\big). \tag{19}$$

Then the same holds for the following four problems:

$$E(\mathbf{0}); \tag{20}$$

$$F(\mathbf{0}); \tag{21}$$

$$E(s)\wedge F(s)\to E(s'); \tag{22}$$

$$E(s)\wedge F(s)\to E(s)\wedge F(s'). \tag{23}$$

For (20) and (21), this is so because $\mathbf{CL4}\vdash P_1\wedge P_2\to P_i$ $(i=1,2)$, whence **CL3** proves both $E(\mathbf{0})\wedge F(\mathbf{0})\to E(\mathbf{0})$ and $E(\mathbf{0})\wedge F(\mathbf{0})\to F(\mathbf{0})$, whence — by the uniform-constructive soundness of **CL3** — we know how to construct polynomial time solutions for these two problems, whence — by the polynomial time solvability of (18) and the closure of this property (in the strong sense of Theorem 9.5) under Modus Ponens — we also know how to construct polynomial time solutions for $E(\mathbf{0})$ and $F(\mathbf{0})$. With (19) instead of (18), the arguments for (22) and (23) are similar, the first one relying on the fact that **CL4** proves $(P_1\to P_2\sqcap Q)\to(P_1\to P_2)$, and the second one relying on the fact that **CL4** proves $\big(P_1\to P_2\sqcap(Q_1\wedge Q_2)\big)\to(P_1\to Q_2\wedge Q_1)$.

Throughout the rest of this proof, assume some arbitrary bounded valuation e to be fixed. Correspondingly, when we write $\mathfrak{b}$ or τ, they are to be understood as $e(\mathfrak{b})$ or $e(\tau)$. As always, saying "polynomial" means "polynomial in $\mathfrak{b}$".

For a formula G and a positive integer n, we will be using the abbreviation

$$\overset{|}{\wedge}{}^n G$$

for the $\wedge$-conjunction $G \wedge \ldots \wedge G$ of n copies of G. If here $n = 1$, $\overset{|}{\wedge}^{n} G$ simply means G.

Claim 1.*For any integer $k \in \{1, \ldots, \tau\}$, the following problem has a polynomial time solution which, in turn, can be constructed in polynomial time:*

$$E(s) \wedge \overset{|}{\wedge}^{k+1} F(s) \rightarrow E(s') \wedge \overset{|}{\wedge}^{k} F(s'). \qquad (24)$$

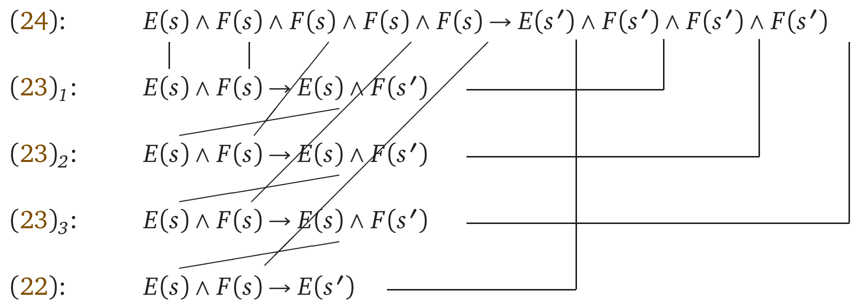
Proof. In this proof and later, we use the term "**synchronizing**" to mean applying copycat between two (sub)games of the form A and $\neg A$. This means copying one player's moves made in A as the other player's moves in $\neg A$, and vice versa. The effect achieved this way is that the games to which A and $\neg A$ eventually evolve (the final positions hit by them, that is) will be of the form A' and $\neg A'$, that is, one will remain the negation of the other, so that one will be won by a given player iff the other is lost by the same player. We already saw an application of this idea/technique in the proof of Theorem 9.5. Partly for this reason and partly because now we are dealing with a more complicated case, our present proof will be given in less detail than the proof of Theorem 9.5 was.

Here is a solution/strategy for (24). While playing the real play of (24) on valuation e, also play, in parallel, one imaginary copy of (22) and k imaginary copies of (23) on the same valuation e, using the strategies for (22) and (23) whose existence we already know. In this mixture of the real and imaginary plays, do the following:

- Synchronize the $F(s)$ of the antecedent of each ith copy of (23) with the ith conjunct of the $\overset{|}{\wedge}^{k+1} F(s)$ part of the antecedent of (24).
- Synchronize the $E(s)$ of the antecedent of the first copy of (23) with the $E(s)$ of the antecedent of (24).
- Synchronize the $E(s)$ of the antecedent of each copy $\#(i+1)$ of (23) with the $E(s)$ of the consequent of copy $\#i$ of (23).
- Synchronize the $E(s)$ of the antecedent of (the single copy of) (22) with the $E(s)$ of the consequent of copy $\#k$ of (23).
- Synchronize the $F(s)$ of the antecedent of (22) with the last conjunct of the $\overset{|}{\wedge}^{k+1} F(s)$ part of the antecedent of (24).

- Synchronize the $E(s')$ of the consequent of (22) with the $E(s')$ of the consequent of (24).
- Synchronize the $F(s')$ of the consequent of each copy $\#i$ of (23) with the ith conjunct of the $\overset{|}{\wedge}^{k} F(s')$ part of the consequent of (24).

Below is an illustration of such synchronization arrangements — indicated by arcs — for the case $k = 3$:


(24): $E(s) \wedge F(s) \wedge F(s) \wedge F(s) \wedge F(s) \rightarrow E(s') \wedge F(s') \wedge F(s') \wedge F(s')$
$(23)_1$: $E(s) \wedge F(s) \rightarrow E(s) \wedge F(s')$
$(23)_2$: $E(s) \wedge F(s) \rightarrow E(s) \wedge F(s')$
$(23)_3$: $E(s) \wedge F(s) \rightarrow E(s) \wedge F(s')$
(22): $E(s) \wedge F(s) \rightarrow E(s')$

Of course, the strategy that we have just described can be constructed effectively and, in fact, in polynomial time, from the strategies for (22) and (23). Furthermore, since the latter run in polynomial time, obviously so does our present one. It is left to the reader to verify that our strategy indeed wins (24). ■

Now, the sought polynomial time solution for

$$s \leq \tau \rightarrow E(s) \wedge F(s) \qquad (25)$$

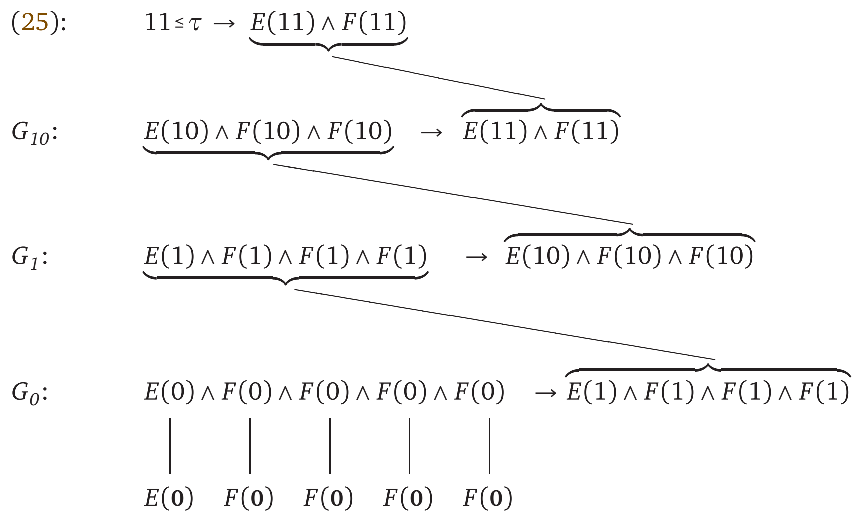
on valuation e will go like this. Read the value $d = e(s)$ of s from the valuation tape. Also read the value of $\mathfrak{b}$ and, using it, compute the value c of τ. Since τ is a $(\mathbf{0}, ', +, \times)$-combination of $\mathfrak{b}$, computing c only takes a polynomial amount of steps. If $d > c$, do nothing — you are the winner (again, comparing d with c, of course, takes only a polynomial amount of steps). Otherwise, using the strategy from Claim 1, for each $a \in \{0, \ldots, d-1\}$, play (a single copy of) the imaginary game G_a on valuation e, defined by

$$G_a \;=\; E(a)\wedge \overset{|}{\wedge}^{d-a+1}F(a)\rightarrow E(a')\wedge \overset{|}{\wedge}^{d-a}F(a').$$

Namely, the effect of playing G_a on valuation e is achieved by playing $E(s)\wedge \overset{|}{\wedge}^{d-a+1}F(s)\rightarrow E(s')\wedge \overset{|}{\wedge}^{d-a}F(s')$ on the valuation e' which sends s to a and agrees with e on all other variables. In addition, using the strategy for (20), play a single imaginary copy of $E(\mathbf{0})$ on e, and, using the strategy for (21), play $d+1$ imaginary copies of $F(\mathbf{0})$ on e. In this mixture of imaginary plays and the real play of (25), do the following:

- Synchronize the above $E(\mathbf{0})$ and $F(\mathbf{0})$s with the corresponding conjuncts of the antecedent of G_0.
- Synchronize the antecedent of each G_{i+1} with the consequent of G_i.
- Synchronize the consequent of G_{d-1} with the consequent of (25).

Below is an illustration of these synchronization arrangements for the case $d = 11$ (decimal 3):


(25):
$11\leq\tau \;\rightarrow\; \underbrace{E(11)\wedge F(11)}$
G_{10}:
$\underbrace{E(10)\wedge F(10)\wedge F(10)} \;\rightarrow\; \overbrace{E(11)\wedge F(11)}$
G_1:
$\underbrace{E(1)\wedge F(1)\wedge F(1)\wedge F(1)} \;\rightarrow\; \overbrace{E(10)\wedge F(10)\wedge F(10)}$
G_0:
$E(0)\wedge F(0)\wedge F(0)\wedge F(0)\wedge F(0) \;\rightarrow\; \overbrace{E(1)\wedge F(1)\wedge F(1)\wedge F(1)}$
$E(\mathbf{0})$ $F(\mathbf{0})$ $F(\mathbf{0})$ $F(\mathbf{0})$ $F(\mathbf{0})$

Again, with some thought, one can see that our strategy — which, of course, can be constructed effectively — runs in polynomial time, and it indeed wins (25), as desired.

16. SOME ADMISSIBLE LOGICAL RULES OF PTA

When we say that a given rule is **admissible** in **PTA**, we mean that, whenever all premises of any given instance of the rule are provable in **PTA**, so is the conclusion.

This section is devoted to observing the admissibility of a number of rules. From our admissibility proofs it can be seen that these rules are admissible not only in **PTA** but also in any **CL3**-based applied theory in general. This is the reason why these rules can be called "logical". Such rules can and will be used as shortcuts in **PTA**-proofs. Many of such rules can be further strengthened, but in this paper — for the sake of simplicity and at the expense of (here) unnecessary generality — we present them only in forms that (and as much as they) will be actually used in our further treatment.

In the formulations of some of the rules we use the expression

$$E^{\vee}[F].$$

It means the same as the earlier-used $E[F]$, i.e., a formula E with a fixed positive surface occurrence of a subformula F; only, in $E^{\vee}[F]$, the additional (to being a positive surface occurrence) condition on the occurrence of F is that this occurrence is not in the scope of any operator other than $\vee$.

16.1. CL4-*Instantiation*

$$\frac{}{F},$$

where F is any **PTA**-formula which is an instance of some **CL4**-provable formula E.

Unlike all other rules given in the present section, this one, as we see, takes no premises. It is a "rule" that simply allows us to jump to a formula F as long as it is an instance of a **CL4**-provable formula.

Fact 16.1 **CL4**-*Instantiation is admissible in* **PTA**.

Proof. Assume a **PTA**-formula F is an instance of some **CL4**-provable formula. Then, by Theorem 11.4, **CL3** $\vdash F$. **CL3** is an analytic system, in the sense that it never introduces into premises any function or predicate letters that are not present in the conclusion. So, all formulas involved in the **CL3**-proof of F will be **PTA**-formulas. This includes the axioms used in the proof. But such axioms are also axioms of **PTA**. And **PTA** has all inference rules that **CL3** does. Hence, the above **CL3**-proof of F will be a **PTA**-proof of F as well. ■

16.2. Transitivity (TR)

$$\frac{E_1 \to F \qquad F \to E_2}{E_1 \to E_2}$$

Fact 16.2 *Transitivity is admissible in* **PTA**.

Proof. Assume

$$\textbf{PTA} \vdash E_1 \to F \ \ \textit{and} \ \ \textbf{PTA} \vdash F \to E_2. \tag{26}$$

CL4 proves $(P_1 \to Q) \wedge (Q \to P_2) \to (P_1 \to P_2)$ (it is derived from the classical tautology $(p_1 \to q) \wedge (q \to p_2) \to \ (p_1 \to p_2)$ by Match applied three times). Hence, by **CL4**-Instantiation,

$$\textbf{PTA} \vdash (E_1 \to F) \wedge (F \to E_2) \to (E_1 \to E_2). \tag{27}$$

Now, from (26) and (27), by Modus Ponens, we get the desired **PTA** $\vdash E_1 \to E_2$. ■

16.3. ⊓-Elimination

$$\frac{\sqcap x F(x)}{F(s)},$$

where x is any variable, $F(x)$ is any formula, s is any variable not bound in the premise, and $F(s)$ is the result of replacing all free occurrences of x by s in $F(x)$.

Fact 16.3 ⊓-*Elimination is admissible in* **PTA**.

Proof. Assume **PTA** $\vdash \sqcap x F(x)$. $p(s) \to p(s)$ is classically valid and hence, by Match, **CL4** $\vdash P(s) \to P(s)$. From here, by ⊔-Choose, **CL4** $\vdash \sqcap x P(x) \to P(s)$. Then, by **CL4**-Instantiation, **PTA** $\vdash \sqcap x F(x) \to F(s)$. Now, by Modus Ponens, **PTA** $\vdash F(s)$. ■

16.4. ⊔-Elimination

$$\frac{F_1 \sqcup \ldots \sqcup F_n \qquad F_1 \to E \qquad \ldots \qquad F_n \to E}{E}$$

Fact 16.4 ⊔-*Elimination is admissible in* **PTA**.

Proof. Assume **PTA** proves all premises. For each $i \in \{1, \ldots, n\}$, the formula

$$p_i \wedge (\bot \to \top) \wedge \ldots \wedge (\bot \to \top) \wedge (p_i \to q) \wedge (\bot \to \top) \wedge \ldots \wedge (\bot \to \top) \to q$$

is a classical tautology and hence an axiom of **CL4**. By Wait from the above, we have

$$\textbf{CL4} \vdash p_i \wedge (P_1 \to Q) \wedge \ldots \wedge (P_{i-1} \to Q) \wedge (p_i \to q) \wedge (P_{i+1} \to Q) \wedge \ldots \wedge (P_n \to Q) \to q.$$

Now, by Match applied twice, we get

$$\textbf{CL4} \vdash P_i \wedge (P_1 \to Q) \wedge \ldots \wedge (P_n \to Q) \to Q.$$

We also have

$$\textbf{CL4} \vdash \bot \wedge (\bot \to \top) \wedge \ldots \wedge (\bot \to \top) \to \bot$$

because the above formula is a classical tautology. From the last two facts, by Wait, we find

$$\textbf{CL4} \vdash (P_1 \sqcup \ldots \sqcup P_n) \wedge (P_1 \to Q) \wedge \ldots \wedge (P_n \to Q) \to Q$$

and hence, by **CL4**-Instantiation,

$$\mathbf{PTA} \vdash (F_1 \sqcup \ldots \sqcup F_n) \wedge (F_1 \rightarrow E) \wedge \ldots \wedge (F_n \rightarrow E) \rightarrow E.$$

As all of the conjuncts of the antecedent of the above formula are **PTA**-provable by our original assumption, Modus Ponens yields $\mathbf{PTA} \vdash E$. ■

As an aside, one could show that the present rule with $\vee$ instead of $\sqcup$, while admissible in classical logic, is not admissible in **PTA** or **CL3**-based applied theories in general.

16.5. Weakening

$$\frac{E^{\vee}[G_1 \vee \ldots \vee G_m \vee H_1 \vee \ldots \vee H_n]}{E^{\vee}[G_1 \vee \ldots \vee G_m \vee F \vee H_1 \vee \ldots \vee H_n]},$$

where $m,n \geq 0$ and $m+n \neq 0$.

Fact 16.5 *Weakening is admissible in* **PTA**.

Proof. Assume **PTA** proves the premise. It is not hard to see that *Premise* $\rightarrow$ *Conclusion* can be obtained by **CL4**-Instantiation, so it is also provable in **PTA**. Hence, by Modus Ponens, **PTA** proves the conclusion. ■

16.6. $\sqcap$-Introduction

$$\frac{E^{\vee}[F_1] \qquad \ldots \qquad E^{\vee}[F_n]}{E^{\vee}[F_1 \sqcap \ldots \sqcap F_n]}$$

Fact 16.6 *$\sqcap$-Introduction is admissible in* **PTA**.

Proof. Assume **PTA** proves each of the n premises. Let G be the $\vee$-disjunction of all subformulas of $E^{\vee}[F_1 \sqcap \ldots \sqcap F_n]$, other than the indicated occurrence of $F_1 \sqcap \ldots \sqcap F_n$, that do not occur in the scope of any operators other than $\vee$ and whose main operator (if nonatomic) is not

$\vee$. We want to first verify the rather expected fact that $\mathbf{PTA} \vdash F_i \vee G$ for each i (expected, because, modulo the associativity of $\vee$, the formulas $E^{\vee}[F_i]$ and $F_i \vee G$ are the same). Indeed, $E^{\vee}[F_i] \rightarrow F_i \vee G$ can be easily seen to be obtainable by **CL4**-Instantiation. Then, $F_i \vee G$ follows by Modus Ponens. In a similar manner one can show that whenever $\mathbf{PTA} \vdash (F_1 \sqcap \ldots \sqcap F_n) \vee G$, we also have $\mathbf{PTA} \vdash E^{\vee}[F_1 \sqcap \ldots \sqcap F_n]$. So, in order to complete our proof of Fact 16.6, it would suffice to show that

$$\mathbf{PTA} \vdash (F_1 \sqcap \ldots \sqcap F_n) \vee G. \tag{28}$$

From $\mathbf{PTA} \vdash F_1 \vee G$, ..., $\mathbf{PTA} \vdash F_1 \vee G$ and the obvious fact that $\mathbf{PTA} \vdash \top$, by Wait, we get

$$\mathbf{PTA} \vdash (F_1 \vee G) \sqcap \ldots \sqcap (F_n \vee G). \tag{29}$$

Next, $p \vee q \rightarrow p \vee q$ is an axiom of **CL4**. From it, by Match applied twice, we get $\mathbf{CL4} \vdash P_i \vee Q \rightarrow P_i \vee Q$ (any $i \in \{1,\ldots,n\}$). Now, by $\sqcup$-Choose, we get

$$\mathbf{CL4} \vdash (P_1 \vee Q) \sqcap \ldots \sqcap (P_n \vee Q) \rightarrow P_i \vee Q.$$

From here and from (the obvious) $\mathbf{CL4} \vdash \top \rightarrow \top \vee \bot$, by Wait, we get

$$\mathbf{CL4} \vdash (P_1 \vee Q) \sqcap \ldots \sqcap (P_n \vee Q) \rightarrow (P_1 \sqcap \ldots \sqcap P_n) \vee Q.$$

The above, by **CL4**-Instantiation, yields

$$\mathbf{PTA} \vdash (F_1 \vee G) \sqcap \ldots \sqcap (F_n \vee G) \rightarrow (F_1 \sqcap \ldots \sqcap F_n) \vee G. \tag{30}$$

Now, the desired (28) follows from (29) and (30) by Modus Ponens. ■

It is worth pointing out that the present rule with $\wedge$ instead of $\sqcap$, while admissible in classical logic, is not admissible in **PTA** or **CL3**-based applied theories in general.

16.7. $\sqcap$-Introduction

$$\frac{E^{\vee}[F(s)]}{E^{\vee}[\sqcap x F(x)]},$$

where x is any (non-$\mathfrak{b}$) variable, $F(x)$ is any formula, s is any non-$\mathfrak{b}$ variable not occurring in the conclusion, and $F(s)$ is the result of replacing all free occurrences of x by s in $F(x)$.

Fact 16.7 *$\sqcap$-Introduction is admissible in* **PTA***.*

Proof. Assume **PTA** $\vdash E^{\vee}[F(s)]$. Let G be the $\vee$-disjunction of all subformulas of $E^{\vee}[\sqcap x F(x)]$, other than the indicated occurrence of $\sqcap x F(x)$, that do not occur in the scope of any operators other than $\vee$ and whose main operator (if nonatomic) is not $\vee$. As in the previous subsection, we can easily find that **PTA** $\vdash F(s) \vee G$, and that whenever **PTA** $\vdash \sqcap x F(x) \vee G$, we also have **PTA** $\vdash E^{\vee}[\sqcap x F(x)]$. So, in order to complete our proof of Fact 16.7, it would suffice to show that

$$\mathbf{PTA} \vdash \sqcap x F(x) \vee G. \tag{31}$$

From **PTA** $\vdash F(s) \vee G$ and the obvious fact that **PTA** $\vdash \top$, by Wait, we get

$$\mathbf{PTA} \vdash \sqcap y \big(F(y) \vee G\big), \tag{32}$$

where y is a “fresh” variable — a variable not occurring in $F(s) \vee G$.

Next, $p(t) \vee q \rightarrow p(t) \vee q$ is an axiom of **CL4**. From it, by Match applied twice, we find that **CL4** proves $P(t) \vee Q \rightarrow P(t) \vee Q$. Now, by $\sqcup$-Choose, we get **CL4** $\vdash \sqcap y \big(P(y) \vee Q\big) \rightarrow P(t) \vee Q$. From here and from (the obvious) **CL4** $\vdash \top \rightarrow \top \vee \bot$, by Wait, we get **CL4** $\vdash \sqcap y \big(P(y) \vee Q\big) \rightarrow \sqcap x P(x) \vee Q$. This, by **CL4**-Instantiation, yields

$$\mathbf{PTA} \vdash \sqcap y \big(F(y) \vee G\big) \rightarrow \sqcap x F(x) \vee G. \tag{33}$$

Now, the desired (31) follows from (32) and (33) by Modus Ponens. ■

We must again point out that the present rule with $\forall$ instead of $\sqcap$, while admissible in classical logic, is not admissible in **PTA** or **CL3**-based applied theories in general.

17. FORMAL VERSUS INFORMAL ARGUMENTS IN PTA

We have already seen a couple of nontrivial formal **PTA**-proofs, and will see more later. However, continuing forever in this style will be hardly

possible. Little by little, we will need to start trusting and relying on informal arguments in the style of the argument found at the beginning of the proof of Lemma 13.1, or the arguments that we employed when discussing the PTI rule in Section 14. Just as in **PA**, formal proofs in **PTA** tend to be long, and generating them in every case can be an arduous job. The practice of dealing with informal proofs or descriptions instead of detailed formal ones is familiar not only from the metatheory of **PA** or similar systems. The same practice is adopted, say, when dealing with Turing machines, where full transition diagrams are typically replaced by high-level informal descriptions, relying on the reader’s understanding that, if necessary, every such description can be turned into a real Turing machine.

In the nearest few sections we will continue generating formal proofs, often accompanied with underlying informal arguments to get used to such arguments and see that they are always translatable into formal ones. As we advance, however, our reliance on informal arguments and the degree of our “laziness” will gradually increase, and in later sections we may stop producing formal proofs altogether.

The informal language and methods of reasoning induced by computability logic and clarithmetic or ptarithmetic in particular, are in the painful initial process of forming and, at this point, can be characterized as “experimental”. They cannot be concisely or fully *explained*, but rather they should be *learned* through experience and practice, not unlike the way one learns a foreign language. A reader who initially does not find some of our informal **PTA**-arguments very clear or helpful, should not feel disappointed. Both the readers and the author should simply keep trying their best. Greater fluency and better understanding will come gradually and inevitably.

At this point we only want to make one general remark on the informal **PTA**-arguments that will be employed. Those arguments will often proceed in terms of game-playing and problem-solving instead of theorem-proving, or will be some kind of a mixture of these two. That is, a way to show how to prove a formula F will often be to show how to win/solve the game/problem F. The legitimacy of this approach is related to the fact that the logic **CL3** underlying **PTA** is a logic of problem-solving and, as such, is complete (Theorem 10.5). That is, whenever a problem F can be solved in a way that relies merely on the logical structure of F — and perhaps also those of some axioms of **PTA** — then we have a guarantee that F can also be proven. Basic problem-solving steps are very directly simulated (translated through) the rules of **CL3** or some derivative rules in the style of the rules of the previous section, with those rules seen bottom-up (in the "from conclusion to premises" direction). For instance, a step such as "choose the ith disjunct in the subformula/subgame $F_1 \sqcup \ldots \sqcup F_n$" translates as a bottom-up application of $\sqcup$-Choose which replaces $F_1 \sqcup \ldots \sqcup F_n$ by F_i; a step such as "specify x as s in $\bigsqcup x F(x)$" translates as a bottom-up application of $\bigsqcup$-Choose; a step such as "wait till the environment specifies a value s for x in $\bigsqcap x F(x)$" translates as a bottom-up application of $\bigsqcap$-Introduction; etc. Correspondingly, an informally described winning/solution strategy for F can usually be seen as a relaxed, bottom-up description of a formal proof of F.

18. SOME ADMISSIBLE INDUCTION RULES OF PTA

The present section introduces a few new admissible rules of induction. These rules are weaker than PTI, but are still useful in that, in many cases, they may offer greater convenience than PTI does.

18.1. WPTI

Here we reproduce rule (17) discussed in Section 14, and baptize it as "**WPTI**" ("W" for "Weak"):

$$\frac{F(\mathbf{0}) \qquad F(s) \to F(s')}{s \le \tau \to F(s)},$$

where s is any non-$\mathfrak{b}$ variable, $F(s)$ is any formula, and τ is any $\mathfrak{b}$-term.

Theorem 18.1 *WPTI is admissible in* **PTA**.

Idea. WPTI is essentially nothing but PTI with $\top$ in the role of $E(s)$. ■

Proof. Assume s, $F(s)$, τ are as stipulated in the rule, and **PTA** proves both $F(\mathbf{0})$ and $F(s) \to F(s')$. The following formula matches the

CL4-provable $(P \to Q) \to \big(\top \wedge P \to \top \sqcap (Q \wedge \top)\big)$ and hence, by **CL4**-Instantiation, is provable in **PTA**:

$$\big(F(s) \to F(s')\big) \to \Big(\top \wedge F(s) \to \top \sqcap \big(F(s') \wedge \top\big)\Big). \qquad (34)$$

By Modus Ponens from $F(s) \to F(s')$ and (34), we find that **PTA** proves

$$\top \wedge F(s) \to \top \sqcap \big(F(s') \wedge \top\big). \qquad (35)$$

Similarly, $F(\mathbf{0}) \to \top \wedge F(\mathbf{0})$ is obviously provable in **PTA** by **CL4**-Instantiation. Modus-ponensing this with our assumption **PTA** $\vdash F(\mathbf{0})$ yields **PTA** $\vdash \top \wedge F(\mathbf{0})$. From here and (35), by PTI with $\top$ in the role of $E(s)$, we find that **PTA** proves $s \le \tau \to \top \wedge F(s)$. But **PTA** also proves $\top \wedge F(s) \to F(s)$ because this is an instance of the **CL4**-provable $\top \wedge P \to P$. Hence, by Transitivity, **PTA** $\vdash s \le \tau \to F(s)$, as desired. ■

18.2. BSI

What we call **BSI** (**B**inary-**S**uccessor-based **I**nduction) is the following rule, where s is any non-$\mathfrak{b}$ variable and $F(s)$ is any formula:

$$\frac{F(\mathbf{0}) \qquad F(s) \to F(s0) \sqcap F(s1)}{F(s)}.$$

Theorem 18.2 *BSI is admissible in* **PTA**.

Idea. We manage to reduce BSI to WPTI with $\bigsqcap x\big(|x| \le s \to F(x)\big)$ in the role of $F(s)$ of the latter. ■

Proof. Assume s, $F(s)$ are as stipulated in the rule,

$$\mathbf{PTA} \vdash F(\mathbf{0}) \qquad (36)$$

and

$$\mathbf{PTA} \vdash F(s) \to F(s0) \sqcap F(s1). \qquad (37)$$

Let us observe right now that, by $\bigsqcap$-Introduction, (37) immediately implies

$$\mathbf{PTA} \vdash \bigsqcap x\big(F(x) \to F(x0) \sqcap F(x1)\big). \qquad (38)$$

The goal is to verify that **PTA** $\vdash F(s)$.

An outline of our strategy for achieving this goal is that we take the formula $\sqcap x\big(|x|\leq t \rightarrow F(x)\big)$ — let us denote it by $G(t)$ — and show that both $G(\mathbf{0})$ and $G(t)\rightarrow G(t')$ are provable. This, by (the already shown to be admissible) WPTI, allows us to immediately conclude that $t\leq \mathfrak{b}\rightarrow G(t)$ is also provable, which, in turn, implies that so is $\sqcap y\big(y\leq\mathfrak{b}\rightarrow G(y)\big)$, and hence $\mathfrak{b}\leq\mathfrak{b}\rightarrow G(\mathfrak{b})$, and hence $G(\mathfrak{b})$. $G(\mathfrak{b})$ asserts that, for any ($\sqcap$) given x whose length does not exceed $\mathfrak{b}$, we can solve $F(x)$. But the length of no x that we consider exceeds $\mathfrak{b}$, so that $G(\mathfrak{b})$, in fact, simply says that (we can solve) $F(x)$. Formalizing this argument in **PTA** and taking s for x yields the desired conclusion **PTA** $\vdash F(s)$.

In following the above outline, we first claim that **PTA** $\vdash G(\mathbf{0})$, i.e.,

$$\textbf{PTA}\vdash \sqcap x\big(|x|\leq \mathbf{0}\rightarrow F(x)\big). \tag{39}$$

An informal argument here is that, since no constant is of length 0, $|x|\leq\mathbf{0}$ is false, and hence the problem $|x|\leq\mathbf{0}\rightarrow F(x)$ is automatically "solved" (i.e., won without any moves by $\top$) no matter what $F(x)$ is. Formally, **PA** and hence **PTA** proves the true fact $\neg|v|\leq\mathbf{0}$. **PTA** also proves $\neg|v|\leq\mathbf{0}\rightarrow\big(|v|\leq\mathbf{0}\rightarrow F(v)\big)$, as this is an instance of the **CL4**-provable $\neg p\rightarrow(p\rightarrow Q)$. Then, by Modus Ponens, **PTA** $\vdash |v|\leq\mathbf{0}\rightarrow F(v)$, whence, by $\sqcap$-Introduction, **PTA** $\vdash \sqcap x\big(|x|\leq\mathbf{0}\rightarrow F(x)\big)$, as desired.

Our next goal is to show that **PTA** $\vdash G(t)\rightarrow G(t')$, i.e.,

$$\textbf{PTA}\vdash \sqcap x\big(|x|\leq t\rightarrow F(x)\big)\rightarrow\sqcap x\big(|x|\leq t'\rightarrow F(x)\big). \tag{40}$$

This can be done by showing the **PTA**-provability of

$$\sqcap x\big(|x|\leq t\rightarrow F(x)\big)\rightarrow |v|\leq t'\rightarrow F(v), \tag{41}$$

from which (40) follows by $\sqcap$-Introduction.

Let us first try to justify (41) informally. Consider any t, v with $|v|\leq t'$, and also assume that (a single copy of) the resource $\sqcap x\big(|x|\leq t\rightarrow F(x)\big)$ is at our disposal. The goal is to establish $F(v)$. $F(0)$ is immediate by (36). In turn, by (37), $F(0)$ easily implies $F(1)$. Thus, we are done for the case $v\leq 1$. Suppose now $v>1$. Then (unlike the case $v\leq 1$), remembering that $|v|\leq t'$, v must have a binary predecessor r with $|r|\leq t$. By Axiom 12, we can actually find such an r and, furthermore, tell whether $v=r0$ or $v=r1$. Specifying x as r in the antecedent of (41), we can bring it down to the resource $|r|\leq t\rightarrow F(r)$ and — as we already know that $|r|\leq t$ — essentially to the resource $F(r)$. By (38), the resource $\sqcap x\big(F(x)\rightarrow F(x0)\sqcap F(x1)\big)$ and hence $F(r)\rightarrow F(r0)\sqcap F(r1)$ is also available. This is a resource that consumes $F(r)$ and generates $F(r0)\sqcap F(r1)$. Feeding to its consumption needs[15] our earlier-obtained $F(r)$, we thus get the resource $F(r0)\sqcap F(r1)$. As noted earlier, we know precisely whether $v=r0$ or $v=r1$. So, by choosing the corresponding $\sqcap$-conjunct, we can further turn $F(r0)\sqcap F(r1)$ into the sought $F(v)$.

Strictly verifying (41) is quite some task, and we break it into several subtasks/subgoals.

Our first subgoal is to show that **PTA** proves the following:

$$v=\mathbf{0}\sqcup v=\mathbf{0}'\sqcup v>\mathbf{0}', \tag{42}$$

implying our ability to (efficiently) tell whether v is 0, 1, or greater than 1. For simplicity considerations, in our earlier informal justification of (41), we, in a sense, cheated by taking this ability for granted — or, rather, by not really mentioning the need for it at all. Some additional evidence of such "cheating" can be discovered after reading the later parts of the present proof as well.

Informally, an argument for (42) goes like this. Due to Axiom 12, we can find the binary predecessor r of v. Moreover, due to the same axiom, we can tell whether $v=r0$ or $v=r1$. Using Axiom 9, we can further tell whether $r=0$ or $r\neq 0$. So, we will know precisely which of the four combinations $v=r0\wedge r=0$, $v=r1\wedge r=0$, $v=r0\wedge r\neq 0$, $v=r1\wedge r\neq 0$ is the case. From **PA**, we also know that in the first case we have $v=0$, in the second case we have $v=1$, and in the third and the fourth cases we have $v>1$. So, one of $v=0$, $v=1$, $v>1$ will be true and, moreover, we will be able to actually tell which one is true.

Below is a full formalization of this argument:

1. $s=\mathbf{0}\sqcup s\neq\mathbf{0}$ Axiom 9
2. $\sqcap x(x=\mathbf{0}\sqcup x\neq\mathbf{0})$ $\sqcap$-Introduction: 1
3. $\sqcup x(v=x0\sqcup v=x1)$ Axiom 12
4. $r=\mathbf{0}\wedge v=r0 \rightarrow v=\mathbf{0}$ **PA**
5. $r=\mathbf{0}\wedge v=r0 \rightarrow v=\mathbf{0}\sqcup v=\mathbf{0}'\sqcup v>\mathbf{0}'$ $\sqcup$-Choose: 4

6. $r \neq \mathbf{0} \wedge v = r0 \ \rightarrow\ v > \mathbf{0}'$ **PA**
7. $r \neq \mathbf{0} \wedge v = r0 \ \rightarrow\ v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\sqcup$-Choose: 6
8. $(r = \mathbf{0} \sqcup r \neq \mathbf{0}) \wedge v = r0 \ \rightarrow\ v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\sqcap$-Introduction: 5,7
9. $\bigsqcap x(x = \mathbf{0} \sqcup x \neq \mathbf{0}) \wedge v = r0 \ \rightarrow\ v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\bigsqcup$-Choose: 8
10. $r = \mathbf{0} \wedge v = r1 \ \rightarrow\ v = \mathbf{0}'$ **PA**
11. $r = \mathbf{0} \wedge v = r1 \ \rightarrow\ v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\sqcup$-Choose: 10
12. $r \neq \mathbf{0} \wedge v = r1 \ \rightarrow\ v > \mathbf{0}'$ **PA**
13. $r \neq \mathbf{0} \wedge v = r1 \ \rightarrow\ v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\sqcup$-Choose: 12
14. $(r = \mathbf{0} \sqcup r \neq \mathbf{0}) \wedge v = r1 \ \rightarrow$ $v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\sqcap$-Introduction: 11,13
15. $\bigsqcap x(x = \mathbf{0} \sqcup x \neq \mathbf{0}) \wedge v = r1 \ \rightarrow\ v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\bigsqcup$-Choose: 14
16. $\bigsqcap x(x = \mathbf{0} \sqcup x \neq \mathbf{0}) \wedge (v = r0 \sqcup v = r1) \ \rightarrow$ $v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\sqcap$-Introduction: 9,15
17. $\bigsqcap x(x = \mathbf{0} \sqcup x \neq \mathbf{0}) \wedge \bigsqcup x(v = x0 \sqcup v = x1) \ \rightarrow$ $v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ $\bigsqcap$-Introduction: 16
18. $v = \mathbf{0} \sqcup v = \mathbf{0}' \sqcup v > \mathbf{0}'$ MP: 2,3,17

The theoremhood of (42) thus has been verified.

Our next subgoal is to show that each disjunct of (42) implies (41), that is, that each of the following formulas is provable in **PTA**:

$$v = \mathbf{0} \rightarrow \bigsqcap x\big(|x| \leq t \rightarrow F(x)\big) \rightarrow |v| \leq t' \rightarrow F(v) \tag{43}$$

$$v = \mathbf{0}' \rightarrow \bigsqcap x\big(|x| \leq t \rightarrow F(x)\big) \rightarrow |v| \leq t' \rightarrow F(v) \tag{44}$$

$$v > \mathbf{0}' \rightarrow \bigsqcap x\big(|x| \leq t \rightarrow F(x)\big) \rightarrow |v| \leq t' \rightarrow F(v) \tag{45}$$

To see the provability of (43), observe that **CL4** proves the formula

$$P(f) \rightarrow g = f \rightarrow P(g). \tag{46}$$

The formula $F(\mathbf{0}) \rightarrow v = \mathbf{0} \rightarrow F(v)$ is an instance of (46) and therefore is provable in **PTA**. By (36), $F(\mathbf{0})$ is also provable. Hence, by Modus Ponens, **PTA** $\vdash v = \mathbf{0} \rightarrow F(v)$. From here, by Weakening applied twice, we find the desired **PTA** $\vdash$ (43).

The **PTA**-provability of (44) is established as follows:

1. $\bigsqcup x(x = \mathbf{0})$ Axiom 8
2. $F(\mathbf{0}) \rightarrow s = \mathbf{0} \rightarrow F(s)$ **CL4**-Instantiation, matches (46)
3. $s = \mathbf{0} \rightarrow F(s)$ MP: (36),2
4. $s = \mathbf{0} \rightarrow F(s0) \sqcap F(s1)$ TR: 3, (37)
5. $F(s0) \sqcap F(s1) \rightarrow F(s1)$ **CL4**-Instantiation, matches $P \sqcap Q \rightarrow Q$
6. $s = \mathbf{0} \rightarrow F(s1)$ TR: 4,5
7. $\big(s = \mathbf{0} \rightarrow F(s1)\big) \rightarrow \big(s = \mathbf{0} \rightarrow F(\mathbf{0}1)\big)$ **CL4**-Instantiation, matches $(s = f \rightarrow P(g(s))) \rightarrow (s = f \rightarrow P(g(f)))$
8. $s = \mathbf{0} \rightarrow F(\mathbf{0}1)$ MP: 6,7
9. $\bigsqcup x(x = \mathbf{0}) \rightarrow F(\mathbf{0}1)$ $\bigsqcap$-Introduction: 8
10. $F(\mathbf{0}1)$ MP: 1,9
11. $\mathbf{0}1 = \mathbf{0}'$ **PA**
12. $F(\mathbf{0}1) \wedge \mathbf{0}1 = \mathbf{0}' \rightarrow F(\mathbf{0}')$ **CL4**-Instantiation, matches $P(f) \wedge f = g \rightarrow P(g)$
13. $F(\mathbf{0}')$ MP: 10,11,12
14. $F(\mathbf{0}') \rightarrow v = \mathbf{0}' \rightarrow F(v)$ **CL4**-Instantiation, matches (46)
15. $v = \mathbf{0}' \rightarrow F(v)$ MP: 13,14
16. $v = \mathbf{0}' \rightarrow \bigsqcap x\big(|x| \leq t \rightarrow F(x)\big) \rightarrow |v| \leq t' \rightarrow F(v)$ Weakening (twice): 15

Finally, to construct a proof of (45), observe that the following formula is valid in classical logic:

$$\big(p_1(f) \wedge p_2(f) \rightarrow p_3\big) \ \rightarrow\ \big(p_4 \rightarrow p_5(f)\big) \wedge v = f \ \rightarrow\ p_2(v)$$
$$\rightarrow\ \big(p_3 \rightarrow p_4\big) \ \rightarrow\ p_1(v) \rightarrow\ p_5(v).$$

Hence, by Match applied twice, **CL4** proves

$$\big(p_1(f) \wedge p_2(f) \rightarrow p_3\big) \ \rightarrow\ \big(P \rightarrow Q(f)\big) \wedge v = f \ \rightarrow\ p_2(v) \ \rightarrow \\ \big(p_3 \rightarrow P\big) \ \rightarrow\ p_1(v) \rightarrow\ Q(v). \tag{47}$$

The following two formulas are instances of (47), and are therefore provable in **PTA**:

$$\big(|r0| \leq t' \wedge r0 > \mathbf{0}' \rightarrow |r| \leq t\big) \rightarrow \big(F(r) \rightarrow F(r0)\big) \wedge v = r0 \rightarrow v > \mathbf{0}' \\ \rightarrow \big(|r| \leq t \rightarrow F(r)\big) \rightarrow |v| \leq t' \rightarrow F(v). \tag{48}$$

$$\big(|r1|\le t' \wedge r1>\mathbf{0}' \to |r|\le t\big) \to \big(F(r)\to F(r1)\big) \wedge v=r1 \to v>\mathbf{0}' \\ \to \big(|r|\le t \to F(r)\big) \to |v|\le t' \to F(v). \qquad (49)$$

Now, the following sequence is a **PTA**-proof of (45):

1. $\bigsqcup x(v=x0 \sqcup v=x1)$ Axiom 12
2. $|r0|\le t' \wedge r0>\mathbf{0}' \to |r|\le t$ **PA**
3. $\big(F(r)\to F(r0)\big) \wedge v=r0 \to v>\mathbf{0}' \to \big(|r|\le t \to F(r)\big) \to |v|\le t' \to F(v)$ MP: (48),2
4. $\big(F(r)\to F(r0)\sqcap F(r1)\big) \wedge v=r0 \to v>\mathbf{0}' \to \big(|r|\le t \to F(r)\big) \to |v|\le t' \to F(v)$ $\sqcup$-Choose: 3
5. $|r1|\le t' \wedge r1>\mathbf{0}' \to |r|\le t$ **PA**
6. $\big(F(r)\to F(r1)\big) \wedge v=r1 \to v>\mathbf{0}' \to \big(|r|\le t \to F(r)\big) \to |v|\le t' \to F(v)$ MP: (49),5
7. $\big(F(r)\to F(r0)\sqcap F(r1)\big) \wedge v=r1 \to v>\mathbf{0}' \to \big(|r|\le t \to F(r)\big) \to |v|\le t' \to F(v)$ $\sqcup$-Choose: 8
8. $\big(F(r)\to F(r0)\sqcap F(r1)\big) \wedge (v=r0 \sqcup v=r1) \to v>\mathbf{0}' \to \big(|r|\le t \to F(r)\big) \to |v|\le t' \to F(v)$ $\sqcap$-Intro: 4,7
9. $\bigsqcap x\big(F(x)\to F(x0)\sqcap F(x1)\big) \wedge (v=r0 \sqcup v=r1) \to v>\mathbf{0}' \to \bigsqcap x\big(|x|\le t \to F(x)\big) \to |v|\le t' \to F(v)$ $\bigsqcup$-Chooses: 8
10. $\bigsqcap x\big(F(x)\to F(x0)\sqcap F(x1)\big) \wedge \bigsqcup x(v=x0 \sqcup v=x1) \to v>\mathbf{0}' \to \bigsqcap x\big(|x|\le t \to F(x)\big) \to |v|\le t' \to F(v)$ $\bigsqcap$-Intro: 9
11. $v>\mathbf{0}' \to \bigsqcap x\big(|x|\le t \to F(x)\big) \to |v|\le t' \to F(v)$ MP: (38),1,10

The provability of each of the three formulas (43), (44) and (43) has now been verified. From these three facts and the provability of (42), by $\sqcup$-Elimination, we find that **PTA** proves (41). This, in turn, as noted earlier, implies (40). Now, from (39) and (40), by WPTI, we find that

$$\mathbf{PTA} \vdash t\le \mathfrak{b} \to \bigsqcap x\big(|x|\le t \to F(x)\big).$$

The above, by $\bigsqcap$-Introduction, yields $\mathbf{PTA} \vdash \bigsqcap y\Big(y\le \mathfrak{b} \to \bigsqcap x\big(|x|\le y \to F(x)\big)\Big)$, from which, by $\bigsqcap$-Elimination, $\mathbf{PTA} \vdash \mathfrak{b}\le \mathfrak{b} \to \bigsqcap x\big(|x|\le \mathfrak{b} \to F(x)\big)$. But $\mathbf{PA} \vdash \mathfrak{b}\le\mathfrak{b}$. So, by Modus Ponens, $\mathbf{PTA} \vdash \bigsqcap x\big(|x|\le \mathfrak{b} \to F(x)\big)$, from which, by $\bigsqcap$-Elimination,

$\mathbf{PTA} \vdash |s|\le \mathfrak{b} \to F(s)$. This, together with Axiom 13, by Modus ponens, yields the desired conclusion $\mathbf{PTA} \vdash F(s)$.
■

18.3. *An illustration of BSI in work*

In this section we prove one **PTA**-provability fact which, with the soundness of **PTA** in mind, formally establishes the efficient decidability of the equality predicate. The proof of this fact presents a good exercise on using BSI, and may help the reader appreciate the convenience offered by this rule, which is often a more direct and intuitive tool for efficiency-preserving inductive reasoning than PTI is.

Lemma 18.3 $\mathbf{PTA} \vdash \bigsqcap x \bigsqcap y(y=x \sqcup y\neq x)$.

Idea. Using BSI, prove $\bigsqcap y(y=s \sqcup y\neq s)$, from which the target formula follows by $\bigsqcap$-Introduction. ■

Proof.
Let us first give an informal justification for $\bigsqcap x \bigsqcap y(y=x \sqcup y\neq x)$. We proceed by BSI-induction on s, where the formula $F(s)$ of induction is $\bigsqcap y(y=s \sqcup y\neq s)$. By Axiom 9, for an arbitrary y, we can tell whether $y=\mathbf{0}$ or $y\neq\mathbf{0}$. This takes care of the basis (left premise)

$$\bigsqcap y(y=\mathbf{0} \sqcup y\neq\mathbf{0}) \qquad (50)$$

of induction. For the inductive step (right premise)

$$\bigsqcap y(y=s \sqcup y\neq s) \to \bigsqcap y(y=s0 \sqcup y\neq s0) \sqcap \bigsqcap y(y=s1 \sqcup y\neq s1), \qquad (51)$$

assume the resource $\bigsqcap y(y=s \sqcup y\neq s)$ is at our disposal. We need to show that we can solve

$$\bigsqcap y(y=s0 \sqcup y\neq s0) \sqcap \bigsqcap y(y=s1 \sqcup y\neq s1),$$

i.e., either *one* of the problems $\bigsqcap y(y=s0 \sqcup y\neq s0)$ and $\bigsqcap y(y=s1 \sqcup y\neq s1)$. Let us for now look at the first problem. Consider an arbitrary y. Axiom 12 allows us to find the binary predecessor r of y and also tell whether $y=r0$ or $y=r1$. If $y=r1$, then we already know that $y\neq s0$ (because $s0$ is even while $r1$ is odd). And if $y=r0$, then $y=s0$ — i.e. $r0=s0$ — iff $r=s$. But whether $r=s$ we can figure out using (the available single copy of) the resource $\sqcap y(y=s\sqcup y\neq s)$. To summarize, in any case we can tell whether $y=s0$ or $y\neq s0$, meaning that we can solve $\sqcap y(y=s0\sqcup y\neq s0)$. The case of $\sqcap y(y=s1\sqcup y\neq s1)$ is handled in a similar way. Then, by BSI, (50) and (51) imply $\sqcap y(s=y\sqcup s\neq y)$, which, in turn (by $\sqcap$-Introduction), implies $\sqcap x\sqcap y(x=y\sqcup x\neq y)$.

The above informal argument can be formalized as follows:

1. $s=\mathbf{0}\sqcup s\neq\mathbf{0}$ Axiom 9
2. $\sqcap y(y=\mathbf{0}\sqcup y\neq\mathbf{0})$ $\sqcap$-Introduction: 1
3. $\sqcup x(t=x0\sqcup t=x1)$ Axiom 12
4. $t=r0\rightarrow r=s\rightarrow t=s0$ Logical axiom
5. $t=r0\rightarrow r=s\rightarrow t=s0\sqcup t\neq s0$ $\sqcup$-Choose: 4
6. $t=r0\rightarrow r\neq s\rightarrow t\neq s0$ **PA**
7. $t=r0\rightarrow r\neq s\rightarrow t=s0\sqcup t\neq s0$ $\sqcup$-Choose: 6
8. $t=r0\rightarrow r=s\sqcup r\neq s\rightarrow t=s0\sqcup t\neq s0$ $\sqcap$-Introduction: 5,7
9. $t=r0\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s0\sqcup t\neq s0$ $\sqcup$-Choose: 8
10. $t=r1\rightarrow t\neq s0$ **PA**
11. $t=r1\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t\neq s0$ Wakening: 10
12. $t=r1\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s0\sqcup t\neq s0$ $\sqcup$-Choose: 11
13. $t=r0\sqcup t=r1\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s0\sqcup t\neq s0$ $\sqcap$-Introduction: 9,12
14. $\sqcup x(t=x0\sqcup t=x1)\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s0\sqcup t\neq s0$ $\sqcap$-Introduction: 13
15. $\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s0\sqcup t\neq s0$ MP: 3,14
16. $\sqcap y(y=s\sqcup y\neq s)\rightarrow\sqcap y(y=s0\sqcup y\neq s0)$ $\sqcap$-Introduction: 15
17. $t=r1\rightarrow r=s\rightarrow t=s1$ Logical axiom
18. $t=r1\rightarrow r=s\rightarrow t=s1\sqcup t\neq s1$ $\sqcup$-Choose: 17
19. $t=r1\rightarrow r\neq s\rightarrow t\neq s1$ **PA**
20. $t=r1\rightarrow r\neq s\rightarrow t=s1\sqcup t\neq s1$ $\sqcup$-Choose: 19
21. $t=r1\rightarrow r=s\sqcup r\neq s\rightarrow t=s1\sqcup t\neq s1$ $\sqcap$-Introduction: 18,20

22. $t=r1\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s1\sqcup t\neq s1$ $\sqcup$-Choose: 21
23. $t=r0\rightarrow t\neq s1$ **PA**
24. $t=r0\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t\neq s1$ Weakening: 23
25. $t=r0\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s1\sqcup t\neq s1$ $\sqcup$-Choose: 24
26. $t=r0\sqcup t=r1\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s1\sqcup t\neq s1$ $\sqcap$-Introduction: 25,22
27. $\sqcup x(t=x0\sqcup t=x1)\rightarrow\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s1\sqcup t\neq s1$ $\sqcap$-Introduction: 26
28. $\sqcap y(y=s\sqcup y\neq s)\rightarrow t=s1\sqcup t\neq s1$ MP: 3,27
29. $\sqcap y(y=s\sqcup y\neq s)\rightarrow\sqcap y(y=s1\sqcup y\neq s1)$ $\sqcap$-Introduction: 28
30. $\sqcap y(y=s\sqcup y\neq s)\rightarrow\sqcap y(y=s0\sqcup y\neq s0)\sqcap\sqcap y(y=s1\sqcup y\neq s1)$ $\sqcap$-Introduction: 16,29
31. $\sqcap y(y=s\sqcup y\neq s)$ BSI: 2,30
32. $\sqcap x\sqcap y(y=x\sqcup y\neq x)$ $\sqcap$-Introduction: 31 ■

18.4. PTI+, WPTI+ and BSI+

The conclusion of PTI limits s to "very small" values — those that do not exceed (the value of) some $\mathfrak{b}$-term τ. On the other hand, the right premise of the rule does not impose the corresponding restriction $s<\tau$ on s, and appears to be stronger than necessary. Imposing the additional condition $|s'|\leq\mathfrak{b}$ on s in that premise also seems reasonable, because the size of s in the conclusion cannot exceed $\mathfrak{b}$ anyway, and hence there is no need to prove the induction hypothesis for the cases with $|s'|>\mathfrak{b}$.[16] So, one might ask why we did not state PTI in the following, seemingly stronger, form — call it "**PTI+**":

$$\frac{E(\mathbf{0})\wedge F(\mathbf{0})\qquad\qquad s<\tau\wedge|s'|\leq\mathfrak{b}\wedge E(s)\wedge F(s)\rightarrow E(s')\sqcap\big(F(s')\wedge E(s)\big)}{s\leq\tau\rightarrow E(s)\wedge F(s)}$$

(with the same additional conditions as in PTI.)

The answer is very simple: PTI+, while being aesthetically (or from the point of view of simplicity) inferior to PTI, does not really offer any greater deductive power, as implied by the forthcoming Theorem 18.6.

The following two rules — call them **WPTI+** (left) and **BSI+** (right) — are pseudostrengthenings of WPTI and BSI in the same sense as PTI+ is a pseudostrengthening of PTI:

$$\frac{F(\mathbf{0}) \qquad s<\tau \wedge |s'|\leq\mathfrak{b} \wedge F(s)\rightarrow F(s')}{s\leq\tau\rightarrow F(s)} \qquad \frac{F(\mathbf{0}) \qquad |s0|\leq\mathfrak{b}\wedge F(s)\rightarrow F(s0)\sqcap F(s1)}{F(s)}$$

where s is any variable different from $\mathfrak{b}$, $F(s)$ is any formula, and τ is any $\mathfrak{b}$-term.

Theorem 18.4 *WPTI+ is admissible in* **PTA**.

Idea. WPTI+ is essentially a special case of WPTI with $|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s)$ in the role of $F(s)$. ■

Proof. Assume s, $F(s)$, τ are as stipulated in the rule,

$$\mathbf{PTA}\vdash F(\mathbf{0}) \tag{52}$$

and

$$\mathbf{PTA}\vdash s<\tau\wedge|s'|\leq\mathfrak{b}\wedge F(s)\rightarrow F(s'). \tag{53}$$

Our goal is to verify that $\mathbf{PTA}\vdash s\leq\tau\rightarrow F(s)$.
From (52), by Weakening applied twice, we get

$$\mathbf{PTA}\vdash |\mathbf{0}|\leq\mathfrak{b}\rightarrow\mathbf{0}\leq\tau\rightarrow F(\mathbf{0}). \tag{54}$$

Next, observe that

$$\mathbf{CL4}\vdash (q_1\wedge q_2\rightarrow p_1\wedge p_2\wedge p_3)\wedge(p_1\wedge q_2\wedge P\rightarrow Q) \\ \rightarrow(p_3\rightarrow p_2\rightarrow P)\rightarrow(q_2\rightarrow q_1\rightarrow Q).$$

Hence, by **CL4**-Instantiation, we have

$$\mathbf{PTA}\vdash (s'\leq\tau\wedge|s'|\leq\mathfrak{b}\rightarrow s<\tau\wedge s\leq\tau\wedge|s|\leq\mathfrak{b})\wedge \\ (s<\tau\wedge|s'|\leq\mathfrak{b}\wedge F(s)\rightarrow F(s'))\rightarrow \\ (|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s))\rightarrow(|s'|\leq\mathfrak{b}\rightarrow s'\leq\tau\rightarrow F(s')). \tag{55}$$

We also have $\mathbf{PA}\vdash s'\leq\tau\wedge|s'|\leq\mathfrak{b}\rightarrow s<\tau\wedge s\leq\tau\wedge|s|\leq\mathfrak{b}$. This, together with (53) and (55), by Modus Ponens, yields

$$\mathbf{PTA}\vdash(|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s))\rightarrow(|s'|\leq\mathfrak{b}\rightarrow s'\leq\tau\rightarrow F(s')). \tag{56}$$

From (54) and (56), by WPTI, we get

$$\mathbf{PTA}\vdash s\leq\tau\rightarrow(|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s)). \tag{57}$$

But $\mathbf{CL4}\vdash (p\rightarrow(q\rightarrow p\rightarrow Q))\rightarrow(q\rightarrow p\rightarrow Q)$ and hence, by **CL4**-Instantiation,

$$\mathbf{PTA}\vdash\Big(s\leq\tau\rightarrow(|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s))\Big)\rightarrow(|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s)).$$

Modus-ponensing the above with (57) yields $\mathbf{PTA}\vdash|s|\leq\mathfrak{b}\rightarrow s\leq\tau\rightarrow F(s)$. Now, remembering Axiom 13, by Modus Ponens, we get the desired $\mathbf{PTA}\vdash s\leq\tau\rightarrow F(s)$. ■

Note that the above proof established something stronger than what Theorem 18.4 states. Namely, our proof of the admissibility of WPTI+ relied on WPTI without appealing to PTI. This means that WPTI+ would remain admissible even if **PTA** had WPTI instead of PTI. It is exactly this fact that justifies the qualification "pseudostrengthening of WPTI" that we gave to WPTI+. The same applies to the other two pseudostrengthening rules PTI+ and BSI+ discussed in this subsection.

Theorem 18.5 *BSI+ is admissible in* **PTA**.

Idea. BSI+ reduces to BSI with $|s|\leq\mathfrak{b}\rightarrow F(s)$ in the role of $F(s)$. ■

Proof. Assume s, $F(s)$ are as stipulated in the rule,

$$\mathbf{PTA}\vdash F(\mathbf{0}) \tag{58}$$

and

$$\mathbf{PTA}\vdash|s0|\leq\mathfrak{b}\wedge F(s)\rightarrow F(s0)\sqcap F(s1). \tag{59}$$

Our goal is to verify that $\mathbf{PTA}\vdash F(s)$.
From (58), by Weakening, we have

$$\mathbf{PTA}\vdash|\mathbf{0}|\leq\mathfrak{b}\rightarrow F(\mathbf{0}). \tag{60}$$

Next, in a routine (analytic) syntactic exercise, one can show that

$$\mathbf{CL4}\vdash(p_0\vee p_1\rightarrow p_0\wedge q)\wedge(p_0\wedge Q\rightarrow P_0\sqcap P_1)\rightarrow(q\rightarrow Q)\rightarrow \\ (p_0\rightarrow P_0)\sqcap(p_1\rightarrow P_1).$$

Hence, by **CL4**-Instantiation,

$$\textbf{PTA} \vdash \big(|s0| \leq \mathfrak{b} \vee |s1| \leq \mathfrak{b} \rightarrow |s0| \leq \mathfrak{b} \wedge |s| \leq \mathfrak{b}\big) \wedge \big(|s0| \leq \mathfrak{b} \wedge F(s) \rightarrow F(s0) \sqcap F(s1)\big) \rightarrow \big(|s| \leq \mathfrak{b} \rightarrow F(s)\big) \rightarrow \big(|s0| \leq \mathfrak{b} \rightarrow F(s0)\big) \sqcap \big(|s1| \leq \mathfrak{b} \rightarrow F(s1)\big). \tag{61}$$

But $\textbf{PA} \vdash |s0| \leq \mathfrak{b} \vee |s1| \leq \mathfrak{b} \rightarrow |s0| \leq \mathfrak{b} \wedge |s| \leq \mathfrak{b}$. This, together with (59) and (61), by Modus Ponens, yields

$$\textbf{PTA} \vdash \big(|s| \leq \mathfrak{b} \rightarrow F(s)\big) \rightarrow \big(|s0| \leq \mathfrak{b} \rightarrow F(s0)\big) \sqcap \big(|s1| \leq \mathfrak{b} \rightarrow F(s1)\big).$$

The above and (60), by BSI, yield $\textbf{PTA} \vdash |s| \leq \mathfrak{b} \rightarrow F(s)$. Finally, modus-ponensing the latter with Axiom 13, we get the desired $\textbf{PTA} \vdash F(s)$. ■

Theorem 18.6 *PTI+ is admissible in* **PTA**.

Idea. PTI+ reduces to PTI with $|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)$ and $|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow F(s)$ in the roles of $E(s)$ and $F(s)$, respectively.

The present theorem will not be relied upon later, so, a reader satisfied with this explanation can safely omit the technical proof given below. ■

Proof. Assume s, $E(s)$, $F(s)$, τ are as stipulated in the rule,

$$\textbf{PTA} \vdash E(\mathbf{0}) \wedge F(\mathbf{0}) \tag{62}$$

and

$$\textbf{PTA} \vdash s < \tau \wedge |s'| \leq \mathfrak{b} \wedge E(s) \wedge F(s) \rightarrow E(s') \sqcap \big(F(s') \wedge E(s)\big). \tag{63}$$

Our goal is to show that $\textbf{PTA} \vdash s \leq \tau \rightarrow E(s) \wedge F(s)$.

CL4 proves $P \wedge Q \rightarrow (p \rightarrow q \rightarrow P) \wedge (p \rightarrow q \rightarrow Q)$ and hence, by **CL4**-Instantiation,

$$\textbf{PTA} \vdash E(\mathbf{0}) \wedge F(\mathbf{0}) \rightarrow \big(|\mathbf{0}| \leq \mathfrak{b} \rightarrow \mathbf{0} \leq \tau \rightarrow E(\mathbf{0})\big) \wedge \big(|\mathbf{0}| \leq \mathfrak{b} \rightarrow \mathbf{0} \leq \tau \rightarrow F(\mathbf{0})\big).$$

Modus-ponensing the above with (62) yields

$$\textbf{PTA} \vdash \big(|\mathbf{0}| \leq \mathfrak{b} \rightarrow \mathbf{0} \leq \tau \rightarrow E(\mathbf{0})\big) \wedge \big(|\mathbf{0}| \leq \mathfrak{b} \rightarrow \mathbf{0} \leq \tau \rightarrow F(\mathbf{0})\big). \tag{64}$$

Next, in a routine syntactic exercise we observe that

$$\textbf{CL4} \vdash \neg(p \wedge q) \rightarrow Q \wedge P_1 \rightarrow (q \rightarrow p \rightarrow P_2) \sqcap \big((q \rightarrow p \rightarrow P_3) \wedge Q\big).$$

Hence, by **CL4**-Instantiation,

$$\textbf{PTA} \vdash \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \rightarrow \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)\big) \wedge \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow F(s)\big) \rightarrow \big(|s'| \leq \mathfrak{b} \rightarrow s' \leq \tau \rightarrow E(s')\big) \sqcap \Big(\big(|s'| \leq \mathfrak{b} \rightarrow s' \leq \tau \rightarrow F(s')\big) \wedge \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)\big)\Big). \tag{65}$$

In another syntactic exercise we find that

$$\textbf{CL4} \vdash (p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge \big(p_0 \wedge q_2 \wedge P_1 \wedge Q_1 \rightarrow P_2 \sqcap (Q_2 \sqcap P_1)\big) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow P_1) \wedge (q_1 \rightarrow p_1 \rightarrow Q_1) \rightarrow (q_2 \rightarrow p_2 \rightarrow P_2) \sqcap \big((q_2 \rightarrow p_2 \rightarrow Q_2) \sqcap (q_1 \rightarrow p_1 \rightarrow P_1)\big). \tag{66}$$

Since this “exercise” is longer than the previous one, below we provide a full proof of (66):

1.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge (p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow \top) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow \top$	Tautology
2.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge (p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow p_4) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow (q_2 \rightarrow p_2 \rightarrow p_4)$	Tautology
3.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge (p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow P_2) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow (q_2 \rightarrow p_2 \rightarrow P_2)$	Match: 2
4.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge \big(p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow P_2 \sqcap (Q_2 \wedge P_1)\big) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow (q_2 \rightarrow p_2 \rightarrow P_2)$	$\sqcup$-Choose: 3
5.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge (p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow q_4 \wedge p_4) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow (q_2 \rightarrow p_2 \rightarrow q_4) \wedge (q_1 \rightarrow p_1 \rightarrow p_4)$	Tautology
6.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge (p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow Q_2 \wedge P_1) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow (q_2 \rightarrow p_2 \rightarrow Q_2) \wedge (q_1 \rightarrow p_1 \rightarrow P_1)$	Match (twice): 5
7.	$(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge \big(p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow P_2 \sqcap (Q_2 \wedge P_1)\big) \rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow (q_2 \rightarrow p_2 \rightarrow Q_2) \wedge (q_1 \rightarrow p_1 \rightarrow P_1)$	$\sqcup$-Choose:6

8. $(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge \big(p_0 \wedge q_2 \wedge p_3 \wedge q_3 \rightarrow P_2 \sqcap (Q_2 \wedge P_1)\big)$
$\rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow p_3) \wedge (q_1 \rightarrow p_1 \rightarrow q_3) \rightarrow$
$(q_2 \rightarrow p_2 \rightarrow P_2) \sqcap \big((q_2 \rightarrow p_2 \rightarrow Q_2) \wedge (q_1 \rightarrow p_1 \rightarrow P_1)\big)$ Wait: 1,4,7

9. $(p_2 \wedge q_2 \rightarrow p_1 \wedge q_1 \wedge p_0) \wedge \big(p_0 \wedge q_2 \wedge P_1 \wedge Q_1 \rightarrow P_2 \sqcap (Q_2 \wedge P_1)\big)$
$\rightarrow p_2 \wedge q_2 \rightarrow (q_1 \rightarrow p_1 \rightarrow P_1) \wedge (q_1 \rightarrow p_1 \rightarrow Q_1) \rightarrow$
$(q_2 \rightarrow p_2 \rightarrow P_2) \sqcap \big((q_2 \rightarrow p_2 \rightarrow Q_2) \wedge (q_1 \rightarrow p_1 \rightarrow P_1)\big)$ Match (twice): 8

The formula below matches the formula of (66) and therefore, by **CL4**-Instantiation,

$$\begin{aligned}&\mathbf{PTA} \vdash (s' \leq \tau \wedge |s'| \leq \mathfrak{b} \rightarrow s \leq \tau \wedge |s| \leq \mathfrak{b} \wedge s < \tau) \wedge \\ &\Big(s < \tau \wedge |s'| \leq \mathfrak{b} \wedge E(s) \wedge F(s) \rightarrow E(s') \sqcap \big(F(s') \wedge E(s)\big)\Big) \rightarrow \\ &s' \leq \tau \wedge |s'| \leq \mathfrak{b} \rightarrow \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)\big) \wedge \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow F(s)\big) \rightarrow \\ &\big(|s'| \leq \mathfrak{b} \rightarrow s' \leq \tau \rightarrow E(s')\big) \sqcap \Big(\big(|s'| \leq \mathfrak{b} \rightarrow s' \leq \tau \rightarrow F(s')\big) \\ &\wedge \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)\big)\Big).\end{aligned} \tag{67}$$

Obviously we have **PA** $\vdash s' \leq \tau \wedge |s'| \leq \mathfrak{b} \rightarrow s \leq \tau \wedge |s| \leq \mathfrak{b} \wedge s < \tau$. This fact, together with (63) and (67), by Modus Ponens, implies

$$\begin{aligned}&\mathbf{PTA} \vdash s' \leq \tau \wedge |s'| \leq \mathfrak{b} \rightarrow \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)\big) \wedge \big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow F(s)\big) \rightarrow \\ &\big(|s'| \leq \mathfrak{b} \rightarrow s' \leq \tau \rightarrow E(s')\big) \sqcap \Big(\big(|s'| \leq \mathfrak{b} \rightarrow s' \leq \tau \rightarrow F(s')\big) \wedge \\ &\big(|s| \leq \mathfrak{b} \rightarrow s \leq \tau \rightarrow E(s)\big)\Big).\end{aligned} \tag{68}$$

According to the forthcoming Lemmas 19.8 and 19.9, whose proofs (as any other proofs in this paper) do not rely on PTI+, we have:

$$\textit{For any term } \theta,\ \mathbf{PTA} \vdash \neg|\theta| \leq \mathfrak{b} \sqcup \bigsqcup z(z = \theta); \tag{69}$$

$$\mathbf{PTA} \vdash \bigsqcap x \bigsqcap y \bigsqcup z(x = y + z \sqcup y = x + z). \tag{70}$$

Below is a proof of the fact that

$$\mathbf{PTA} \vdash \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b}): \tag{71}$$

1. $\neg|s'| \leq \mathfrak{b} \sqcup \bigsqcup z(z = s')$ (69) with $\theta = s'$
2. $\neg|s'| \leq \mathfrak{b} \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ Logical axiom
3. $\neg|s'| \leq \mathfrak{b} \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Choose: 2

4. $\neg|\tau| \leq \mathfrak{b} \sqcup \bigsqcup z(z = \tau)$ (69) with $\theta = \tau$
5. $|r| \leq \mathfrak{b}$ Axiom 13
6. $|r| \leq \mathfrak{b} \rightarrow \neg|\tau| \leq \mathfrak{b} \rightarrow r = s' \rightarrow s' \leq \tau \wedge |s'| \leq \mathfrak{b}$ **PA**
7. $\neg|\tau| \leq \mathfrak{b} \rightarrow r = s' \rightarrow s' \leq \tau \wedge |s'| \leq \mathfrak{b}$ MP: 5,6
8. $\neg|\tau| \leq \mathfrak{b} \rightarrow r = s' \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Choose: 7
9. $\bigsqcup z(t = r + z \sqcup r = t + z)$ $\sqcap$-Elimination (twice): (70)
10. $|r| \leq \mathfrak{b} \wedge t = r + v \rightarrow t = \tau \rightarrow r = s' \rightarrow s' \leq \tau \wedge |s'| \leq \mathfrak{b}$ **PA**
11. $|r| \leq \mathfrak{b} \wedge t = r + v \rightarrow t = \tau \rightarrow r = s' \rightarrow$
$\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Choose: 10
12. $v = \mathbf{0} \sqcup v \neq \mathbf{0}$ Axiom 8
13. $v = \mathbf{0} \rightarrow |r| \leq \mathfrak{b} \wedge r = t + v \rightarrow t = \tau \rightarrow r = s' \rightarrow s' \leq \tau \wedge |s'| \leq \mathfrak{b}$ **PA**
14. $v = \mathbf{0} \rightarrow |r| \leq \mathfrak{b} \wedge r = t + v \rightarrow t = \tau \rightarrow r = s' \rightarrow$
$\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Choose: 13
15. $v \neq \mathbf{0} \rightarrow |r| \leq \mathfrak{b} \wedge r = t + v \rightarrow t = \tau \rightarrow r = s' \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ **PA**
16. $v \neq \mathbf{0} \rightarrow |r| \leq \mathfrak{b} \wedge r = t + v \rightarrow t = \tau \rightarrow r = s' \rightarrow$
$\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Choose: 15
17. $|r| \leq \mathfrak{b} \wedge r = t + v \rightarrow t = \tau \rightarrow r = s' \rightarrow$
$\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Elimination: 12,14,16
18. $|r| \leq \mathfrak{b} \wedge (t = r + v \sqcup r = t + v) \rightarrow t = \tau \rightarrow r = s' \rightarrow$
$\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcap$-Introduction: 11,17
19. $|r| \leq \mathfrak{b} \wedge \bigsqcup z(t = r + z \sqcup r = t + z) \rightarrow t = \tau \rightarrow r = s' \rightarrow$
$\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\bigsqcap$-Introduction: 18
20. $t = \tau \rightarrow r = s' \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ MP: 5,9,19
21. $\bigsqcup z(z = \tau) \rightarrow r = s' \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\bigsqcap$-Introduction: 20
22. $r = s' \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Elimination: 4,8,21
23. $\bigsqcup z(z = s') \rightarrow \neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\bigsqcap$-Introduction: 22
24. $\neg(s' \leq \tau \wedge |s'| \leq \mathfrak{b}) \sqcup (s' \leq \tau \wedge |s'| \leq \mathfrak{b})$ $\sqcup$-Elimination: 1,3,23

Now, from (71), (65) and (68), by $\sqcup$-Elimination, we get

$$\mathbf{PTA}\vdash \big(|s|\leq\mathfrak{b}\to s\leq\tau\to E(s)\big)\wedge\big(|s|\leq\mathfrak{b}\to s\leq\tau\to F(s)\big)\to \big(|s'|\leq\mathfrak{b}\to s'\leq\tau\to E(s')\big)\sqcap\Big(\big(|s'|\leq\mathfrak{b}\to s'\leq\tau\to F(s')\big)\wedge \big(|s|\leq\mathfrak{b}\to s\leq\tau\to E(s)\big)\Big). \quad (72)$$

From (64) and (72), by PTI, we get

$$\mathbf{PTA}\vdash s\leq\tau\to\big(|s|\leq\mathfrak{b}\to s\leq\tau\to E(s)\big)\wedge\big(|s|\leq\mathfrak{b}\to s\leq\tau\to F(s)\big). \quad (73)$$

But **CL4** obviously proves $\big(p\to(q\to p\to P)\wedge(q\to p\to Q)\big)\to(q\to p\to P\wedge Q)$ and hence, by **CL4**-Instantiation,

$$\mathbf{PTA}\vdash\Big(s\leq\tau\to\big(|s|\leq\mathfrak{b}\to s\leq\tau\to E(s)\big)\wedge\big(|s|\leq\mathfrak{b}\to s\leq\tau\to F(s)\big)\Big)\to \big(|s|\leq\mathfrak{b}\to s\leq\tau\to E(s)\wedge F(s)\big).$$

Modus-ponensing the above with (73) yields $\mathbf{PTA}\vdash |s|\leq\mathfrak{b}\to s\leq\tau\to E(s)\wedge F(s)$, further modus-ponensing which with Axiom 13 yields the desired $\mathbf{PTA}\vdash s\leq\tau\to E(s)\wedge F(s)$.
■

18.5. BPI

For any formula $E(s)$, we let $E(\lfloor s/2\rfloor)$ stand for the formula $\forall z\big(s=z0\vee s=z1\to E(z)\big)$, asserting that E holds for the binary predecessor of s.

One last rule of induction that we are going to look at is what we call **BPI** (**B**inary-**P**redecessor-based **I**nduction):

$$\frac{F(\mathbf{0})\qquad F(\lfloor s/2\rfloor)\to F(s)}{F(s)},$$

where s is any non-$\mathfrak{b}$ variable and $F(s)$ is any formula.[17]

This rule could be characterized as an "alternative formulation of BSI+", and is apparently equivalent to the latter in the sense that replacing PTI with BSI+ in ptarithmetic yields the same class of provable formulas as replacing PTI with BPI. One direction of this equivalence is immediately implied by our proof of the following theorem.

Theorem 18.7 *BPI is admissible in* **PTA**.

Idea. As noted, BPI is essentially the same as BSI+. ■

Proof. Assume s, $F(s)$ are as stipulated in the rule,

$$\mathbf{PTA}\vdash F(\mathbf{0}) \quad (74)$$

and

$$\mathbf{PTA}\vdash F(\lfloor s/2\rfloor)\to F(s). \quad (75)$$

Our goal is to verify that $\mathbf{PTA}\vdash F(s)$.
We observe that

$$\mathbf{CL4}\vdash \big(p\to t=f(s)\big)\wedge\Big(\forall z\big(t=f(z)\vee q\to P(z)\big)\to Q(t)\Big) \to p\wedge P(s)\to Q\big((f(s)\big)$$

(bottom-up, apply Match twice and you will hit a classically valid formula). By $\sqcup$-Choose, this yields

$$\mathbf{CL4}\vdash \big(p\to t=f(s)\big)\wedge\sqcap x\Big(\forall z\big(x=f(z)\vee q\to P(z)\big)\to Q(x)\Big) \to p\wedge P(s)\to Q\big((f(s)\big).$$

The above, together with the obvious fact $\mathbf{CL4}\vdash(p\to\bot)\wedge\top\to p\wedge\top\to\bot$, by Wait, yields

$$\mathbf{CL4}\vdash\Big(p\to\sqcup x\big(x=f(s)\big)\Big)\wedge\sqcap x\Big(\forall z\big(x=f(z)\vee q\to P(z)\big)\to Q(x)\Big) \to p\wedge P(s)\to Q\big((f(s)\big).$$

Hence, by **CL4**-Instantiation, we have

$$\mathbf{PTA}\vdash\big(|s0|\leq\mathfrak{b}\to\sqcup x(x=s0)\big)\wedge\sqcap x\Big(\forall z\big(x=z0\vee x=z1\to F(z)\big) \to F(x)\Big)\to|s0|\leq\mathfrak{b}\wedge F(s)\to F(s0),$$

which we abbreviate as

$$\mathbf{PTA}\vdash\big(|s0|\leq\mathfrak{b}\to\sqcup x(x=s0)\big)\wedge\sqcap x\big(F(\lfloor x/2\rfloor)\to F(x)\big) \to|s0|\leq\mathfrak{b}\wedge F(s)\to F(s0). \quad (76)$$

In a similar way we find that

$$\mathbf{PTA} \vdash \big(|s0| \le \mathfrak{b} \to \sqcup x(x = s1)\big) \wedge \sqcap x\big(F(\lfloor x/2 \rfloor) \to F(x)\big) \to |s0| \le \mathfrak{b} \wedge F(s) \to F(s1). \quad (77)$$

Now, we construct a sought **PTA**-proof of $F(s)$ as follows:

1. $\sqcap x\big(F(\lfloor x/2 \rfloor) \to F(x)\big)$ $\sqcap$-Introduction: (75)
2. $|s0| \le \mathfrak{b} \to \sqcup x(x = s0)$ Axiom 11
3. $|s0| \le \mathfrak{b} \wedge F(s) \to F(s0)$ MP: 2,1,(76)
4. $|s0| \le \mathfrak{b} \to |s1| \le \mathfrak{b}$ **PA**
5. $|s1| \le \mathfrak{b} \to \sqcup x(x = s1)$ Lemma 13.1
6. $|s0| \le \mathfrak{b} \to \sqcup x(x = s1)$ TR: 4,5
7. $|s0| \le \mathfrak{b} \wedge F(s) \to F(s1)$ MP: 6,1,(77)
8. $|s0| \le \mathfrak{b} \wedge F(s) \to F(s0) \sqcap F(s1)$ $\sqcap$-Introduction: 3,7
9. $F(s)$ BSI+: (74),8 ■

19. EFFICIENT COMPUTABILITY THROUGH PTA-PROVABILITY

In this section we establish several **PTA**-provability facts. In view of the soundness of **PTA**, each such fact tells us about the efficient solvability of the associated number-theoretic computational problem.

19.1. The efficient computability of logarithm

The term “logarithm” in the title of this subsection refers to the size of the binary numeral for a given number, which happens to be an integer approximation of the (real) base-2 logarithm of that number.

Lemma 19.1 **PTA** $\vdash \sqcap x \sqcup y(y = |x|)$.

Proof. An outline of our proof is that $\sqcap x \sqcup y(y = |x|)$ follows by $\sqcap$-Introduction from $\sqcup y(y = |s|)$, and the latter will be proven by BPI. Let us first try to justify the two premises of BPI informally.

From **PA**, we know that the size of $\mathbf{0}$ is $\mathbf{0}'$, and the value of $\mathbf{0}'$ can be found using Lemma 13.3. This allows us to resolve $\sqcup y(y = |\mathbf{0}|)$, which is the basis of our BPI-induction.

The inductive step looks like

$$\sqcup y(y = |\lfloor s/2 \rfloor|) \to \sqcup y(y = |s|).$$

Resolving it means telling the size of s (in the consequent) while knowing (from the antecedental resource) the size r of the binary predecessor $\lfloor s/2 \rfloor$ of s. As was established earlier in the proof of Theorem 18.2, we can tell whether s equals $\mathbf{0}$, $\mathbf{0}'$, or neither. If $s = \mathbf{0}$ or $s = \mathbf{0}'$, then its size is the value of $\mathbf{0}'$ which, as pointed out in the previous paragraph, we know how to compute. Otherwise, the size of s is r', which we can compute using Axiom 10. In all cases we thus can tell the size of s, and thus we can resolve the consequent of the above-displayed inductive step.

Below is a formal counterpart of the above argument.

1. $\sqcup x(x = \mathbf{0}')$ Lemma 13.3
2. $v = \mathbf{0}' \to v = |\mathbf{0}|$ **PA**
3. $v = \mathbf{0}' \to \sqcup y(y = |\mathbf{0}|)$ $\sqcup$-Choose: 2
4. $\sqcup x(x = \mathbf{0}') \to \sqcup y(y = |\mathbf{0}|)$ $\sqcap$-Introduction: 3
5. $\sqcup y(y = |\mathbf{0}|)$ MP: 1,4
6. $s = \mathbf{0} \sqcup s = \mathbf{0}' \sqcup s > \mathbf{0}'$ (42), established in the proof of Theorem 18.2
7. $v = \mathbf{0}' \to s = \mathbf{0} \to v = |s|$ **PA**
8. $v = \mathbf{0}' \to s = \mathbf{0} \to \sqcup y(y = |s|)$ $\sqcup$-Choose: 7
9. $\sqcup x(x = \mathbf{0}') \to s = \mathbf{0} \to \sqcup y(y = |s|)$ $\sqcap$-Introduction: 8
10. $s = \mathbf{0} \to \sqcup y(y = |s|)$ MP: 1,9
11. $s = \mathbf{0} \to \sqcup y(y = |\lfloor s/2 \rfloor|) \to \sqcup y(y = |s|)$ Weakening: 10
12. $v = \mathbf{0}' \to s = \mathbf{0}' \to v = |s|$ **PA**
13. $v = \mathbf{0}' \to s = \mathbf{0}' \to \sqcup y(y = |s|)$ $\sqcup$-Choose: 12
14. $\sqcup x(x = \mathbf{0}') \to s = \mathbf{0}' \to \sqcup y(y = |s|)$ $\sqcap$-Introduction: 13
15. $s = \mathbf{0}' \to \sqcup y(y = |s|)$ MP: 1,14
16. $s = \mathbf{0}' \to \sqcup y(y = |\lfloor s/2 \rfloor|) \to \sqcup y(y = |s|)$ Weakening: 15
17. $|s| \le \mathfrak{b}$ Axiom 13

18. $|s'| \le \mathfrak{b} \to \sqcup x(x = s')$ Axiom 10
19. $\sqcap y\big(|y'| \le \mathfrak{b} \to \sqcup x(x = y')\big)$ $\sqcap$-Introduction: 18
20. $|s| \le \mathfrak{b} \wedge \big(|r'| \le \mathfrak{b} \to \bot\big) \to s > \mathbf{0}' \to r = |\lfloor s/2 \rfloor| \to \bot$ **PA**
21. $|s| \le \mathfrak{b} \wedge (|r'| \le \mathfrak{b} \to w = r') \to s > \mathbf{0}' \to r = |\lfloor s/2 \rfloor| \to w = |s|$ **PA**
22. $|s| \le \mathfrak{b} \wedge (|r'| \le \mathfrak{b} \to w = r') \to s > \mathbf{0}' \to r = |\lfloor s/2 \rfloor| \to \sqcup y(y = |s|)$ $\sqcup$-Choose: 21
23. $|s| \le \mathfrak{b} \wedge \big(|r'| \le \mathfrak{b} \to \sqcup x(x = r')\big) \to s > \mathbf{0}' \to r = |\lfloor s/2 \rfloor| \to \sqcup y(y = |s|)$ Wait: 20,22
24. $|s| \le \mathfrak{b} \wedge \sqcap y\big(|y'| \le \mathfrak{b} \to \sqcup x(x = y')\big) \to s > \mathbf{0}' \to r = |\lfloor s/2 \rfloor| \to \sqcup y(y = |s|)$ $\sqcup$-Choose: 23
25. $|s| \le \mathfrak{b} \wedge \sqcap y\big(|y'| \le \mathfrak{b} \to \sqcup x(x = y')\big) \to s > \mathbf{0}' \to \sqcup y(y = |\lfloor s/2 \rfloor|) \to \sqcup y(y = |s|)$ $\sqcap$-Introduction: 24
26. $s > \mathbf{0}' \to \sqcup y(y = |\lfloor s/2 \rfloor|) \to \sqcup y(y = |s|)$ MP: 17,19,25
27. $\sqcup y(y = |\lfloor s/2 \rfloor|) \to \sqcup y(y = |s|)$ $\sqcup$-Elimination: 6,11,16,26
28. $\sqcup y(y = |s|)$ BPI: 5,27 ■

Lemma 19.2 **PTA** $\vdash \sqcap x(|x| = \mathfrak{b} \sqcup |x| < \mathfrak{b})$.

Proof. An informal argument for $\sqcap x(|x| = \mathfrak{b} \sqcup |x| < \mathfrak{b})$ is the following. Given an arbitrary x, we can find a t with $t = |x|$ using Lemma 19.1. Lemma 18.3 allows us to tell whether $t = \mathfrak{b}$ or $t \ne \mathfrak{b}$. In the second case, in view of Axiom 13, we have $t < \mathfrak{b}$. Thus, we can tell whether $t = \mathfrak{b}$ or $t < \mathfrak{b}$, i.e., whether $|x| = \mathfrak{b}$ or $|x| < \mathfrak{b}$. This means that we can resolve $|x| = \mathfrak{b} \sqcup |x| < \mathfrak{b}$. Formally, we have:

1. $\sqcap x \sqcup y(y = |x|)$ Lemma 19.1
2. $\sqcap x \sqcap y(y = x \sqcup y \ne x)$ Lemma 18.3
3. $t = |s| \wedge t = \mathfrak{b} \to |s| = \mathfrak{b}$ Logical axiom
4. $t = |s| \wedge t = \mathfrak{b} \to |s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ $\sqcup$-Choose: 3
5. $|s| \le \mathfrak{b}$ Axiom 13
6. $|s| \le \mathfrak{b} \to t = |s| \wedge t \ne \mathfrak{b} \to |s| < \mathfrak{b}$ **PA**
7. $t = |s| \wedge t \ne \mathfrak{b} \to |s| < \mathfrak{b}$ MP: 5,6
8. $t = |s| \wedge t \ne \mathfrak{b} \to |s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ $\sqcap$-Choose: 7

9. $t = |s| \wedge (t = \mathfrak{b} \sqcup t \ne \mathfrak{b}) \to |s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ $\sqcap$-Introduction: 4,8
10. $t = |s| \wedge \sqcap x \sqcap y(y = x \sqcup y \ne x) \to |s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ $\sqcup$-Choose (twice): 9
11. $\sqcup y(y = |s|) \wedge \sqcap x \sqcap y(y = x \sqcup y \ne x) \to |s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ $\sqcap$-Introduction: 10
12. $\sqcap x \sqcup y(y = |x|) \wedge \sqcap x \sqcap y(y = x \sqcup y \ne x) \to |s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ $\sqcup$-Choose: 11
13. $|s| = \mathfrak{b} \sqcup |s| < \mathfrak{b}$ MP: 1,2,12
14. $\sqcap x(|x| = \mathfrak{b} \sqcup |x| < \mathfrak{b})$ $\sqcap$-Introduction: 13 ■

19.2. *The efficient computability of unary successor*

In our subsequent treatment we will be using the abbreviation

$$E \sqsupset F$$

for the expression $\neg E \sqcup F$. The operator $\sqsupset$ thus can be called **choice implication**.

When omitting parentheses, $\sqsupset$ will have the same precedence level as $\sqcup$, so that, say, $E \sqsupset F \to G$ should be understood as $(E \sqsupset F) \to G$ rather than $E \sqsupset (F \to G)$.

The following lemma strengthens (the $\sqcap$-closure of) Axiom 10 by replacing $\to$ with $\sqsupset$.

Lemma 19.3 **PTA** $\vdash \sqcap x\big(|x'| \le \mathfrak{b} \sqsupset \sqcup y(y = x')\big)$.

Proof. An informal argument for $\sqcap x\big(|x'| \le \mathfrak{b} \sqsupset \sqcup y(y = x')\big)$ goes like this. Given an arbitrary x, using Lemma 19.2, we can figure out whether $|x| = \mathfrak{b}$ or $|x| < \mathfrak{b}$.

If $|x| < \mathfrak{b}$, then (by **PA**) $|x'| \le \mathfrak{b}$. Then, using Axiom 10, we can find a t with $t = x'$. In this case, $|x'| \le \mathfrak{b} \sqsupset \sqcup y(y = x')$ will be resolved by choosing its right component $\sqcup y(y = x')$ and then specifying y as t in it.

Suppose now $|x| = \mathfrak{b}$. Then, by **PA**, $|x'| \le \mathfrak{b}$ if and only if x is even. And Axiom 12 allows us to tell whether x is even or odd. If x is even, we resolve $|x'| \le \mathfrak{b} \sqsupset \sqcup y(y = x')$ as in the previous case. And if x is odd, then $|x'| \le \mathfrak{b} \sqsupset \sqcup y(y = x')$, i.e. $\neg|x'| \le \mathfrak{b} \sqcup \sqcup y(y = x')$, is resolved by choosing its left component $\neg|x'| \le \mathfrak{b}$.

The following is a formalization of the above argument.

1. $\sqcap x(|x|=\mathfrak{b}\sqcup|x|<\mathfrak{b})$ Lemma 19.2
2. $|s|=\mathfrak{b}\sqcup|s|<\mathfrak{b}$ $\sqcap$-Elimination: 1
3. $\sqcup x(s=x0\sqcup s=x1)$ Axiom 12
4. $|s'|\leq\mathfrak{b}\rightarrow\sqcup x(x=s')$ Axiom 10
5. $(|s'|\leq\mathfrak{b}\rightarrow\bot)\rightarrow s=r0\rightarrow|s|=\mathfrak{b}\rightarrow\bot$ **PA**
6. $(|s'|\leq\mathfrak{b}\rightarrow t=s')\rightarrow s=r0\rightarrow|s|=\mathfrak{b}\rightarrow t=s'$ **PA**
7. $(|s'|\leq\mathfrak{b}\rightarrow t=s')\rightarrow s=r0\rightarrow|s|=\mathfrak{b}\rightarrow\sqcup y(y=s')$ $\sqcup$-Choose: 6
8. $(|s'|\leq\mathfrak{b}\rightarrow\sqcup x(x=s'))\rightarrow s=r0\rightarrow|s|=\mathfrak{b}\rightarrow\sqcup y(y=s')$ Wait: 5,7
9. $s=r0\rightarrow|s|=\mathfrak{b}\rightarrow\sqcup y(y=s')$ MP: 4,8
10. $s=r0\rightarrow|s|=\mathfrak{b}\rightarrow|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ $\sqcup$ -Choose: 9
11. $s=r1\rightarrow|s|=\mathfrak{b}\rightarrow\neg|s'|\leq\mathfrak{b}$ **PA**
12. $s=r1\rightarrow|s|=\mathfrak{b}\rightarrow|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ $\sqcup$ -Choose: 11
13. $s=r0\sqcup s=r1\rightarrow|s|=\mathfrak{b}\rightarrow|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ $\sqcap$ -Introduction: 10,12
14. $\sqcup x(s=x0\sqcup s=x1)\rightarrow|s|=\mathfrak{b}\rightarrow|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ $\sqcap$-Introduction: 13
15. $|s|=\mathfrak{b}\rightarrow|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ MP: 3,14
16. $(|s'|\leq\mathfrak{b}\rightarrow\bot)\rightarrow|s|<\mathfrak{b}\rightarrow\bot$ **PA**
17. $(|s'|\leq\mathfrak{b}\rightarrow t=s')\rightarrow|s|<\mathfrak{b}\rightarrow t=s'$ **PA**
18. $(|s'|\leq\mathfrak{b}\rightarrow t=s')\rightarrow|s|<\mathfrak{b}\rightarrow\sqcup y(y=s')$ $\sqcup$-Choose: 17
19. $(|s'|\leq\mathfrak{b}\rightarrow\sqcup x(x=s'))\rightarrow|s|<\mathfrak{b}\rightarrow\sqcup y(y=s')$ Wait: 16,18
20. $|s|<\mathfrak{b}\rightarrow\sqcup y(y=s')$ MP: 4,19
21. $|s|<\mathfrak{b}\rightarrow|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ $\sqcup$ -Choose: 20
22. $|s'|\leq\mathfrak{b}\sqsupset\sqcup y(y=s')$ $\sqcup$ -Elimination: 2,15,21
23. $\sqcap x(|x'|\leq\mathfrak{b}\sqsupset\sqcup y(y=x'))$ $\sqcap$-Introduction: 22 ■

19.3. The efficient computability of binary 0*-successor*

The following lemma strengthens Axiom 11 in the same way as Lemma 19.3 strengthens Axiom 10.

Lemma 19.4 **PTA** $\vdash\sqcap x(|x0|\leq\mathfrak{b}\sqsupset\sqcup y(y=x0))$.

Proof. Informally, the argument underlying our formal proof of $\sqcap x(|x0|\leq\mathfrak{b}\sqsupset\sqcup y(y=x0))$ is the following. Consider an arbitrary x. Using Lemma 19.2, we can tell whether $|x|=\mathfrak{b}$ or $|x|<\mathfrak{b}$. If $|x|<\mathfrak{b}$, then $|x0|\leq\mathfrak{b}$ and, using Axiom 11, we can find a t with $t=x0$. We then resolve $|x0|\leq\mathfrak{b}\sqsupset\sqcup y(y=x0)$ by choosing its right $\sqsupset$-component and specifying y as t in it. Suppose now $|x|=\mathfrak{b}$. Using Axiom 9, we can tell whether x is 0 or not. If x is 0, then $|x0|\leq\mathfrak{b}\sqsupset\sqcup y(y=x0)$ is resolved by choosing its right $\sqsupset$-component and specifying y as x in it. Otherwise, if $x\neq\mathbf{0}$, then the size of $x0$ exceeds $\mathfrak{b}$. So, $|x0|\leq\mathfrak{b}\sqsupset\sqcup y(y=x0)$ is resolved by choosing its left $\sqsupset$-component $\neg|x0|\leq\mathfrak{b}$. Formally, we have:

1. $\sqcap x(|x|=\mathfrak{b}\sqcup|x|<\mathfrak{b})$ Lemma 19.2
2. $|s|=\mathfrak{b}\sqcup|s|<\mathfrak{b}$ $\sqcap$-Elimination: 1
3. $s=\mathbf{0}\sqcup s\neq\mathbf{0}$ Axiom 9
4. $s=\mathbf{0}\rightarrow|s|=\mathfrak{b}\rightarrow s=s0$ **PA**
5. $s=\mathbf{0}\rightarrow|s|=\mathfrak{b}\rightarrow\sqcup y(y=s0)$ $\sqcup$-Choose: 4
6. $s=\mathbf{0}\rightarrow|s|=\mathfrak{b}\rightarrow|s0|\leq\mathfrak{b}\sqsupset\sqcup y(y=s0)$ $\sqcup$ -Choose: 5
7. $s\neq\mathbf{0}\rightarrow|s|=\mathfrak{b}\rightarrow\neg|s0|\leq\mathfrak{b}$ **PA**
8. $s\neq\mathbf{0}\rightarrow|s|=\mathfrak{b}\rightarrow|s0|\leq\mathfrak{b}\sqsupset\sqcup y(y=s0)$ $\sqcup$ -Choose: 7
9. $|s|=\mathfrak{b}\rightarrow|s0|\leq\mathfrak{b}\sqsupset\sqcup y(y=s0)$ $\sqcup$ -Elimination: 3,6,8
10. $|s0|\leq\mathfrak{b}\rightarrow\sqcup x(x=s0)$ Axiom 11
11. $(|s0|\leq\mathfrak{b}\rightarrow\bot)\rightarrow|s|<\mathfrak{b}\rightarrow\bot$ **PA**
12. $(|s0|\leq\mathfrak{b}\rightarrow t=s0)\rightarrow|s|<\mathfrak{b}\rightarrow t=s0$ **PA**
13. $(|s0|\leq\mathfrak{b}\rightarrow t=s0)\rightarrow|s|<\mathfrak{b}\rightarrow\sqcup y(y=s0)$ $\sqcup$-Choose: 12
14. $(|s0|\leq\mathfrak{b}\rightarrow\sqcup x(x=s0))\rightarrow|s|<\mathfrak{b}\rightarrow\sqcup y(y=s0)$ Wait: 11,13
15. $|s|<\mathfrak{b}\rightarrow\sqcup y(y=s0)$ MP: 10,14
16. $|s|<\mathfrak{b}\rightarrow|s0|\leq\mathfrak{b}\sqsupset\sqcup y(y=s0)$ $\sqcup$ -Choose: 15

17. $|s0| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=s0)$ $\sqcup$-Elimination: 2,9,16
18. $\bigsqcap x(|x0| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x0))$ $\bigsqcap$-Introduction: 17 ■

19.4. The efficient computability of binary 1*-successor*

The following lemma is the same to Lemma 13.1 as Lemma 19.4 is to Axiom 11.

Lemma 19.5 **PTA** $\vdash \bigsqcap x(|x1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x1))$.

Proof. The argument underlying our formal proof of $\bigsqcap x(|x1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x1))$ is the following. Consider an arbitrary x. Using Lemma 19.4, we can tell whether the size of $x0$ exceeds $\mathfrak{b}$, or else find a t with $t=x0$. In the first case we resolve $|x1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x1)$ by choosing its left component $\neg|x1| \leq \mathfrak{b}$, because (we know from **PA** that) $|x0| = |x1|$. In the second case, using Axiom 10, we find an r with $r=t'$. This axiom is applicable here because $|t'| \leq \mathfrak{b}$; $|t'| \leq \mathfrak{b}$, in turn, is true because $|t| \leq \mathfrak{b}$ (by Axiom 13) and t is even, so the unary successor of t is of the same size as t itself. Note that (as **PA** can help us to figure out), in the present case, $r=x1$. So, we can resolve $|x1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x1)$ by choosing its right component and then specifying y as r in it. Formally, we have:

1. $\bigsqcap x(\neg|x0| \leq \mathfrak{b} \sqcup \bigsqcup y(y=x0))$ Lemma 19.4
2. $\neg|s0| \leq \mathfrak{b} \sqcup \bigsqcup y(y=s0)$ $\bigsqcap$-Elimination: 1
3. $\neg|s0| \leq \mathfrak{b} \rightarrow \neg|s1| \leq \mathfrak{b}$ **PA**
4. $\neg|s0| \leq \mathfrak{b} \rightarrow |s1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=s1)$ $\sqcup$-Choose: 3
5. $|t| \leq \mathfrak{b}$ Axiom 13
6. $|t'| \leq \mathfrak{b} \rightarrow \bigsqcup x(x=t')$ Axiom 10
7. $|t| \leq \mathfrak{b} \wedge (|t'| \leq \mathfrak{b} \rightarrow \bot) \rightarrow t=s0 \rightarrow \bot$ **PA**
8. $|t| \leq \mathfrak{b} \wedge (|t'| \leq \mathfrak{b} \rightarrow r=t') \rightarrow t=s0 \rightarrow r=s1$ **PA**
9. $|t| \leq \mathfrak{b} \wedge (|t'| \leq \mathfrak{b} \rightarrow r=t') \rightarrow t=s0 \rightarrow \bigsqcup y(y=s1)$ $\bigsqcup$-Choose: 8
10. $|t| \leq \mathfrak{b} \wedge (|t'| \leq \mathfrak{b} \rightarrow \bigsqcup x(x=t')) \rightarrow t=s0 \rightarrow \bigsqcup y(y=s1)$ Wait: 7,9
11. $t=s0 \rightarrow \bigsqcup y(y=s1)$ MP: 5,6,10

12. $\bigsqcup y(y=s0) \rightarrow \bigsqcup y(y=s1)$ $\bigsqcap$-Introduction: 11
13. $\bigsqcup y(y=s0) \rightarrow |s1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=s1)$ $\sqcup$-Choose: 12
14. $|s1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=s1)$ $\sqcup$-Elimination: 2,4,1
15. $\bigsqcap x(|x1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x1))$ $\bigsqcap$-Introduction: 14 ■

19.5. The efficient computability of addition

Lemma 19.6 **PTA** $\vdash \bigsqcap x \bigsqcap y\big(|x+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=x+y)\big)$.

Proof. The main idea behind our proof of $\bigsqcap x \bigsqcap y\big(|x+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=x+y)\big)$, which proceeds by BSI induction, is the fact — known from **PA** — that the sum of two numbers can be "easily" found from the sum of the binary predecessors of those numbers. Specifically, observe that we have:

(i) $s0+r0=(s+r)0$, because $2s+2r=2(s+r)$;

(ii) $s0+r1=(s+r)1$, because $2s+(2r+1)=2(s+r)+1$;

(iii) $s1+r0=(s+r)1$, because $(2s+1)+2r=2(s+r)+1$;

(iv) $s1+r1=\big((s+r)1\big)'$, because $(2s+1)+(2r+1)=\big(2(s+r)+1\big)+1$.

The formula of induction is $\bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big)$ (from which the target formula immediately follows by $\bigsqcap$-Introduction).

The basis $\bigsqcap y\big(|\mathbf{0}+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=\mathbf{0}+y)\big)$ of induction can be established/resolved rather easily, by choosing the right component of the $\sqsupset$ combination and selecting the value of z to be the same as the value of y.

In resolving the inductive step

$$\bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big) \rightarrow \bigsqcap y\big(|s0+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+y)\big)$$
$$\sqcap \bigsqcap y\big(|s1+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+y)\big),$$

we wait for the environment to select a $\sqcap$-conjunct in the consequent (bottom-up $\sqcap$-Introduction) and then select a value t for y in it (bottom-up $\bigsqcap$-Introduction). Let us say the left conjunct is selected, meaning that the inductive step will be brought down to (i.e.

the premise we are talking about will be)

$$\sqcap y\big(|s+y|\leq\mathfrak{b}\sqsupset\sqcup z(z=s+y)\big)\rightarrow\big(|s0+t|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+t)\big).$$

Using Axiom 12, we can find the binary predecessor r of t, and also figure out whether t is $r0$ or $r1$. Let us say $t=r0$. Then we specify y as r in the antecedent of the above formula, after which the problem we need to resolve is, in fact,

$$\big(|s+r|\leq\mathfrak{b}\sqsupset\sqcup z(z=s+r)\big)\rightarrow\big(|s0+r0|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+r0)\big).$$

Here we can wait till the environment selects one of the $\sqsupset$-components in the antecedent. If the left component is selected, we can resolve the problem by selecting the left $\sqsupset$-component in the consequent, because, if $|s+r|$ exceeds $\mathfrak{b}$, then "even more so" does $|s0+r0|$. Otherwise, if the right component is selected, then we further wait till the environment also selects a value u for z there, after which the problem will be brought down to

$$u=s+r\rightarrow\big(|s0+r0|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+r0)\big).$$

But from the earlier observation (i) we know that $s0+r0=(s+r)0$. So, the above problem is, in fact, nothing but

$$u=s+r\rightarrow\big(|u0|\leq\mathfrak{b}\sqsupset\sqcup z(z=u0)\big),$$

which — whose consequent, that is — we can resolve using Lemma 19.4.

The remaining three possibilities of the above scenario are similar, but will rely on observation (ii), (iii) or (iv) instead of (i), and Lemma 19.5 instead of 19.4. The case corresponding to (iv), in addition, will also use Lemma 19.3.

Below is a formal counterpart of the above argument in full detail:

1. $s=\mathbf{0}+s$ **PA**
2. $\sqcup z(z=\mathbf{0}+s)$ $\sqcup$-Choose: 1
3. $|\mathbf{0}+s|\leq\mathfrak{b}\sqsupset\sqcup z(z=\mathbf{0}+s)$ $\sqcup$-Choose: 2
4. $\sqcap y\big(|\mathbf{0}+y|\leq\mathfrak{b}\sqsupset\sqcup z(z=\mathbf{0}+y)\big)$ $\sqcap$-Introduction: 3
5. $t=r0\rightarrow\neg|s+r|\leq\mathfrak{b}\rightarrow\neg|s0+t|\leq\mathfrak{b}$ **PA**

6. $t=r1\rightarrow\neg|s+r|\leq\mathfrak{b}\rightarrow\neg|s0+t|\leq\mathfrak{b}$ **PA**
7. $t=r0\sqcup t=r1\rightarrow\neg|s+r|\leq\mathfrak{b}\rightarrow\neg|s0+t|\leq\mathfrak{b}$ $\sqcap$-Introduction: 5,6
8. $t=r0\sqcup t=r1\rightarrow\neg|s+r|\leq\mathfrak{b}\rightarrow|s0+t|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+t)$ $\sqcup$-Choose: 7
9. $\sqcap x(|x0|\leq\mathfrak{b}\sqsupset\sqcup y(y=x0))$ Lemma 19.4
10. $|u0|\leq\mathfrak{b}\sqsupset\sqcup y(y=u0)$ $\sqcap$-Elimination: 9
11. $\neg|u0|\leq\mathfrak{b}\rightarrow t=r0\rightarrow u=s+r\rightarrow\neg|s0+t|\leq\mathfrak{b}$ **PA** (observation (i))
12. $\neg|u0|\leq\mathfrak{b}\rightarrow t=r0\rightarrow u=s+r\rightarrow|s0+t|\leq\mathfrak{b}$ $\sqsupset\sqcup z(z=s0+t)$ $\sqcup$-Choose: 11
13. $w=u0\rightarrow t=r0\rightarrow u=s+r\rightarrow w=s0+t$ **PA** (observation (i))
14. $w=u0\rightarrow t=r0\rightarrow u=s+r\rightarrow\sqcup z(z=s0+t)$ $\sqcup$-Choose: 13
15. $\sqcup y(y=u0)\rightarrow t=r0\rightarrow u=s+r$ $\rightarrow\sqcup z(z=s0+t)$ $\sqcap$-Introduction: 14
16. $\sqcup y(y=u0)\rightarrow t=r0\rightarrow u=s+r\rightarrow|s0+t|\leq\mathfrak{b}$ $\sqsupset\sqcup z(z=s0+t)$ $\sqcup$-Choose: 15
17. $|u0|\leq\mathfrak{b}\sqsupset\sqcup y(y=u0)\rightarrow t=r0\rightarrow u=s+r$ $\rightarrow|s0+t|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+t)$ $\sqcap$-Introduction: 12,16
18. $t=r0\rightarrow u=s+r\rightarrow|s0+t|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+t)$ MP: 10,17
19. $\sqcap x(|x1|\leq\mathfrak{b}\sqsupset\sqcup y(y=x1))$ Lemma 19.5
20. $|u1|\leq\mathfrak{b}\sqsupset\sqcup y(y=u1)$ $\sqcap$-Elimination: 19
21. $\neg|u1|\leq\mathfrak{b}\rightarrow t=r1\rightarrow u=s+r\rightarrow\neg|s0+t|\leq\mathfrak{b}$ **PA** (observation (ii))
22. $\neg|u1|\leq\mathfrak{b}\rightarrow t=r1\rightarrow u=s+r\rightarrow|s0+t|\leq\mathfrak{b}$ $\sqsupset\sqcup z(z=s0+t)$ $\sqcup$-Choose: 21
23. $w=u1\rightarrow t=r1\rightarrow u=s+r\rightarrow w=s0+t$ **PA** (observation (ii))
24. $w=u1\rightarrow t=r1\rightarrow u=s+r\rightarrow\sqcup z(z=s0+t)$ $\sqcup$-Choose: 23
25. $\sqcup y(y=u1)\rightarrow t=r1\rightarrow u=s+r\rightarrow$ $\sqcup z(z=s0+t)$ $\sqcap$-Introduction: 24
26. $\sqcup y(y=u1)\rightarrow t=r1\rightarrow u=s+r\rightarrow|s0+t|\leq\mathfrak{b}$ $\sqsupset\sqcup z(z=s0+t)$ $\sqcup$-Choose: 25
27. $|u1|\leq\mathfrak{b}\sqsupset\sqcup y(y=u1)\rightarrow t=r1\rightarrow u=s+r$ $\rightarrow|s0+t|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+t)$ $\sqcap$-Introduction: 22,26
28. $t=r1\rightarrow u=s+r\rightarrow|s0+t|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0+t)$ MP: 20,27

29. $t=r0 \sqcup t=r1 \to u=s+r \to |s0+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+t)$ $\sqcap$-Introduction: 18,28
30. $t=r0 \sqcup t=r1 \to \bigsqcup z(z=s+r) \to |s0+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+t)$ $\bigsqcap$-Introduction: 29
31. $t=r0 \sqcup t=r1 \to |s+r| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+r) \to |s0+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+t)$ $\sqcap$-Introduction: 8,30
32. $t=r0 \sqcup t=r1 \to \bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big) \to |s0+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+t)$ $\bigsqcup$-Choose: 31
33. $\bigsqcup x(t=x0 \sqcup t=x1) \to \bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big) \to |s0+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+t)$ $\bigsqcap$-Introduction: 32
34. $\bigsqcup x(t=x0 \sqcup t=x1)$ Axiom 12
35. $\bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big) \to |s0+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+t)$ MP: 34,33
36. $\bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big) \to \bigsqcap y\big(|s0+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s0+y)\big)$ $\bigsqcap$-Introduction: 35
37. $t=r0 \to \neg|s+r| \leq \mathfrak{b} \to \neg|s1+t| \leq \mathfrak{b}$ **PA**
38. $t=r1 \to \neg|s+r| \leq \mathfrak{b} \to \neg|s1+t| \leq \mathfrak{b}$ **PA**
39. $t=r0 \sqcup t=r1 \to \neg|s+r| \leq \mathfrak{b} \to \neg|s1+t| \leq \mathfrak{b}$ $\sqcap$-Introduction: 37,38
40. $t=r0 \sqcup t=r1 \to \neg|s+r| \leq \mathfrak{b} \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcup$-Choose: 39
41. $\neg|u1| \leq \mathfrak{b} \to t=r0 \to u=s+r \to \neg|s1+t| \leq \mathfrak{b}$ **PA** (observation (iii))
42. $\neg|u1| \leq \mathfrak{b} \to t=r0 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcup$-Choose: 41
43. $w=u1 \to t=r0 \to u=s+r \to w=s1+t$ **PA** (observation (iii))
44. $w=u1 \to t=r0 \to u=s+r \to \bigsqcup z(z=s1+t)$ $\bigsqcup$-Choose: 43
45. $\bigsqcup y(y=u1) \to t=r0 \to u=s+r \to \bigsqcup z(z=s1+t)$ $\bigsqcap$-Introduction: 44
46. $\bigsqcup y(y=u1) \to t=r0 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcup$-Choose: 45
47. $|u1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=u1) \to t=r0 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcap$-Introduction: 42,46
48. $t=r0 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ MP: 20,47

49. $\neg|u1| \leq \mathfrak{b} \to t=r1 \to u=s+r \to \neg|s1+t| \leq \mathfrak{b}$ **PA** (observation (iv))
50. $\neg|u1| \leq \mathfrak{b} \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcup$-Choose: 49
51. $\bigsqcap x\big(|x'| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=x')\big)$ Lemma 19.3
52. $|w'| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=w')$ $\bigsqcap$-Elimination: 51
53. $\neg|w'| \leq \mathfrak{b} \to w=u1 \to t=r1 \to u=s+r \to \neg|s1+t| \leq \mathfrak{b}$ **PA** (observation (iv))
54. $\neg|w'| \leq \mathfrak{b} \to w=u1 \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcup$-Choose: 53
55. $v=w' \to w=u1 \to t=r1 \to u=s+r \to v=s1+t$ **PA** (observation (iv))
56. $v=w' \to w=u1 \to t=r1 \to u=s+r \to \bigsqcup z(z=s1+t)$ $\bigsqcup$-Choose: 55
57. $\bigsqcup y(y=w') \to w=u1 \to t=r1 \to u=s+r \to \bigsqcup z(z=s1+t)$ $\bigsqcap$-Introduction: 56
58. $\bigsqcup y(y=w') \to w=u1 \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcup$-Choose: 57
59. $|w'| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=w') \to w=u1 \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcap$-Introduction: 54,58
60. $w=u1 \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ MP: 52,59
61. $\bigsqcup y(y=u1) \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\bigsqcap$-Introduction: 60
62. $|u1| \leq \mathfrak{b} \sqsupset \bigsqcup y(y=u1) \to t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcap$-Introduction: 50,61
63. $t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ MP: 20,62
64. $t=r0 \sqcup t=r1 \to u=s+r \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcap$-Introduction: 48,63
65. $t=r0 \sqcup t=r1 \to \bigsqcup z(z=s+r) \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\bigsqcap$-Introduction: 64
66. $t=r0 \sqcup t=r1 \to |s+r| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+r) \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\sqcap$-Introduction: 40,65
67. $t=r0 \sqcup t=r1 \to \bigsqcap y\big(|s+y| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s+y)\big) \to |s1+t| \leq \mathfrak{b} \sqsupset \bigsqcup z(z=s1+t)$ $\bigsqcup$-Choose: 66

68. $\bigsqcup x(t=x0\sqcup t=x1)\to\bigsqcap y\bigl(|s+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s+y)\bigr)\to|s1+t|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s1+t)$ $\bigsqcap$-Introduction: 67

69. $\bigsqcap y\bigl(|s+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s+y)\bigr)\to|s1+t|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s1+t)$ MP: 34,68

70. $\bigsqcap y\bigl(|s+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s+y)\bigr)\to\bigsqcap y\bigl(|s1+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s1+y)\bigr)$ $\bigsqcap$-Introduction: 69

71. $\bigsqcap y\bigl(|s+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s+y)\bigr)\to\bigsqcap y\bigl(|s0+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s0+y)\bigr)\sqcap\bigsqcap y\bigl(|s1+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s1+y)\bigr)$
$\sqcap$-Introduction: 36,70

72. $\bigsqcap y\bigl(|s+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s+y)\bigr)$ BSI: 4,71

73. $\bigsqcap x\bigsqcap y\bigl(|x+y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=x+y)\bigr)$ $\bigsqcap$-Introduction: 72 ■

19.6. The efficient computability of multiplication

The following lemma is fully analogous to the lemma of the previous subsection, with the difference that this one is about multiplication instead of addition. Morally, the proof of this lemma is also very similar to the proof of its counterpart. But, as multiplication is somewhat more complex than addition, technically a formal proof here would be considerably longer than the 73-step proof of Lemma 19.6, and producing it would be no fun. For this reason, we limit ourselves to only an informal proof. As noted earlier, sooner or later it would be necessary to abandon the luxury of generating formal proofs, anyway.

Lemma 19.7 **PTA** $\vdash\bigsqcap x\bigsqcap y\bigl(|x\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=x\times y)\bigr)$.

Proof. By BSI induction on s, we want to prove $\bigsqcap y\bigl(|s\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s\times y)\bigr)$, from which the target formula follows by $\bigsqcap$-Introduction.

The *basis*

$$\bigsqcap y\bigl(|\mathbf{0}\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=\mathbf{0}\times y)\bigr)\tag{78}$$

of induction is simple: for whatever y, since $\mathbf{0}=\mathbf{0}\times y$, the problem $|\mathbf{0}\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=\mathbf{0}\times y)$ is resolved by choosing the right $\sqsupset$-component and specifying z as the value of $\mathbf{0}$. Our ability to produce such a value is guaranteed by Axiom 8.

The *inductive step* is

$$\bigsqcap y\bigl(|s\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s\times y)\bigr)\to\bigsqcap y\bigl(|s0\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s0\times y)\bigr)\sqcap\bigsqcap y\bigl(|s1\times y|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s1\times y)\bigr).\tag{79}$$

In justifying it, we rely on the following facts — call them "*observations*" for subsequent references — provable in **PA**:

(i) $s0\times r0=(s\times r)00$, because $2s\times 2r=4(s\times r)$;

(ii) $s0\times r1=(s\times r)00+s0$, because $2s\times(2r+1)=4(s\times r)+2s$;

(iii) $s1\times r0=(s\times r)00+r0$, because $(2s+1)\times 2r=4(s\times r)+2r$;

(iv) $s1\times r1=(s\times r)00+(s+r)1$, because $(2s+1)\times(2r+1)=4(s\times r)+\bigl(2(s+r)+1\bigr)$.

In resolving (79), at the beginning we wait till the environment selects one of the two $\sqcap$-conjuncts in the consequent, and also a value t for y there. What we see as a "beginning" here is, in fact, the end of the proof of (79) for, as pointed out in Section 17, such proofs correspond to winning strategies only when they are read bottom-up. And, as we know, the steps corresponding to selecting a $\sqcap$-conjunct and selecting t for y are (bottom-up) $\sqcap$-Introduction and $\bigsqcap$-Introduction. Then, using Axiom 12, we find the binary predecessor r of t. Furthermore, the same axiom will simultaneously allow us to tell whether $t=r0$ or $t=r1$. We immediately specify (bottom-up $\bigsqcup$-Choose) y as r in the antecedent of (79). We thus have the following four possibilities to consider now, depending on whether the left or the right $\sqcap$-conjunct was selected in the consequent of (79), and whether $t=r0$ or $t=r1$. In each case we will have a different problem to resolve.

Case 1: The problem to resolve (essentially) is

$$|s\times r|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s\times r)\to|s0\times r0|\le\mathfrak{b}\sqsupset\bigsqcup z(z=s0\times r0).\tag{80}$$

Pretending for a while — for simplicity — that no values that we are going to deal with have sizes exceeding $\mathfrak{b}$, here is our strategy. Using the resource provided by the antecedent of (80), we find the product w of s and r. Then, using the resource provided by Lemma 19.4 (which, unlike the resource provided by the antecedent of (80), comes in an unlimited supply) twice, we find the value v of $w00$, i.e. of $(s\times r)00$. In view of observation (i), that very v will be (equal to) $s0\times r0$, so (80) can be resolved by choosing the right $\sqsupset$-component in its consequent and specifying z as v.

The above, however, was a simplified scenario. In a complete scenario without "cheating", what may happen is that, while using the antecedent of (80) in computing $s\times r$, or while — after that — using Lemma 19.4 in (first) computing $(s\times r)0$ and (then) $(s\times r)00$, we discover that the size of the to-be-computed value exceeds $\mathfrak{b}$ and hence the corresponding resource (the antecedent of (80), or Lemma 19.4) does not really allow us to compute that value. Such a corresponding resource, however, does allow us to tell that the size of the value sought has exceeded $\mathfrak{b}$. And, in that case, (80) is resolved by choosing the left component $\neg|s0\times r0|\leq\mathfrak{b}$ of its consequent.

Case 2: The problem to resolve is

$$|s\times r|\leq\mathfrak{b}\sqsupset\sqcup z(z=s\times r)\to|s0\times r1|\leq\mathfrak{b}\sqsupset\sqcup z(z=s0\times r1). \tag{81}$$

Here and in the remaining cases, as was done in the first paragraph of Case 1, we will continue pretending that no values that we deal with have sizes exceeding $\mathfrak{b}$. Violations of this simplifying assumption will be handled in the way explained in the second paragraph of Case 1.

Here, we fist compute (the value of) $(s\times r)00$ exactly as we did in Case 1. Exploiting Lemma 19.4 one more time, we also compute $s0$. Using these values, we then employ Lemma 19.6 to compute $(s\times r)00+s0$, and use the computed value to specify z in the consequent of (81) (after first choosing the right $\sqsupset$-component there, of course). Observation (ii) guarantees success.

Case 3: The problem to resolve is

$$|s\times r|\leq\mathfrak{b}\sqsupset\sqcup z(z=s\times r)\to|s1\times r0|\leq\mathfrak{b}\sqsupset\sqcup z(z=s1\times r0). \tag{82}$$

This case is very similar to the previous one, with the only difference that Lemma 19.4 will be used to compute $r0$ rather than $s0$, and the success of the strategy will be guaranteed by observation (iii) rather than (ii).

Case 4: The problem to resolve is

$$|s\times r|\leq\mathfrak{b}\sqsupset\sqcup z(z=s\times r)\to|s1\times r1|\leq\mathfrak{b}\sqsupset\sqcup z(z=s1\times r1). \tag{83}$$

First, we compute $(s\times r)00$ exactly as in Case 1. Using Lemma 19.6, we also compute $s+r$ and then, using Lemma 19.5, compute $(s+r)1$. With the values of $(s\times r)00$ and $(s+r)1$ now known, Lemma 19.6 allows us to compute the value of $(s\times r)00+(s+r)1$. Finally, using the resulting value to specify z in the consequent of (83), we achieve success. It is guaranteed by observation (iv). ■

19.7. *The efficient computability of all explicitly polynomial functions*

By "explicitly polynomial functions" in the title of this subsection we mean functions represented by terms of the language of **PTA**. Such functions are "explicitly polynomial" because they, along with variables, are only allowed to use $\mathbf{0}$, $'$, $+$ and $\times$.

Lemma 19.8 *For any*[18] *term* τ, $\mathbf{PTA}\vdash|\tau|\leq\mathfrak{b}\sqsupset\sqcup z(z=\tau)$.

Proof. We prove this lemma by (meta)induction on the complexity of τ. The following Cases 1 and 2 comprise the basis of this induction, and Cases 3-5 the inductive step.

Case 1: τ is a variable t. In this case the formula $|\tau|\leq\mathfrak{b}\sqsupset\sqcup z(z=\tau)$, i.e. $|t|\leq\mathfrak{b}\sqsupset\sqcup z(z=t)$, immediately follows from the logical axiom $t=t$ by $\sqcup$-Choose and then $\sqcup$-Choose.

Case 2: τ is $\mathbf{0}$. Then $|\mathbf{0}|\leq\mathfrak{b}\sqsupset\sqcup z(z=\mathbf{0})$ follows in a single step from Axiom 8 by $\sqcup$-Choose.

Case 3: τ is θ' for some term θ. By the induction hypothesis, **PTA** proves

$$|\theta|\leq\mathfrak{b}\sqsupset\sqcup z(z=\theta). \tag{84}$$

Our goal is to establish the **PTA**-provability of $|\theta'|\leq\mathfrak{b}\sqsupset\sqcup z(z=\theta')$, which is done as follows:

1. $\sqcap x\big(|x'|\leq\mathfrak{b}\sqsupset\sqcup y(y=x')\big)$ Lemma 19.3

2. $\neg|\theta|\leq\mathfrak{b}\rightarrow\neg|\theta'|\leq\mathfrak{b}$ **PA**
3. $\neg|\theta|\leq\mathfrak{b}\wedge\bigsqcap x\big(|x'|\leq\mathfrak{b}\sqsupset\bigsqcup y(y=x')\big)\rightarrow\neg|\theta'|\leq\mathfrak{b}$ Weakening: 2
4. $\neg|\theta|\leq\mathfrak{b}\wedge\bigsqcap x\big(|x'|\leq\mathfrak{b}\sqsupset\bigsqcup y(y=x')\big)\rightarrow |\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\sqcup$-Choose: 3
5. $s=\theta\wedge\neg|s'|\leq\mathfrak{b}\rightarrow\neg|\theta'|\leq\mathfrak{b}$ Logical axiom
6. $s=\theta\wedge\neg|s'|\leq\mathfrak{b}\rightarrow|\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\sqcup$-Choose: 5
7. $s=\theta\wedge t=s'\rightarrow t=\theta'$ Logical axiom
8. $s=\theta\wedge t=s'\rightarrow\bigsqcup z(z=\theta')$ $\bigsqcup$-Choose: 7
9. $s=\theta\wedge\bigsqcup y(y=s')\rightarrow\bigsqcup z(z=\theta')$ $\bigsqcap$-Introduction: 8
10. $s=\theta\wedge\bigsqcup y(y=s')\rightarrow|\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\sqcup$-Choose: 9
11. $s=\theta\wedge\big(|s'|\leq\mathfrak{b}\sqsupset\bigsqcup y(y=s')\big)\rightarrow |\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\sqcap$-Introduction: 6,10
12. $s=\theta\wedge\bigsqcap x\big(|x'|\leq\mathfrak{b}\sqsupset\bigsqcup y(y=x')\big)\rightarrow |\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\bigsqcup$-Choose: 11
13. $\bigsqcup z(z=\theta)\wedge\bigsqcap x\big(|x'|\leq\mathfrak{b}\sqsupset\bigsqcup y(y=x')\big)\rightarrow |\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\bigsqcap$-Introduction: 12
14. $\big(|\theta|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta)\big)\wedge\bigsqcap x\big(|x'|\leq \mathfrak{b}\sqsupset\bigsqcup y(y=x')\big)\rightarrow|\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ $\sqcap$-Introduction: 4,13
15. $|\theta'|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta')$ MP: (84),1,14

Case 4: τ is $\theta_1+\theta_2$ for some terms θ_1 and θ_2. By the induction hypothesis, **PTA** proves both of the following formulas:

$$|\theta_1|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1); \tag{85}$$

$$|\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_2). \tag{86}$$

Our goal is to establish the **PTA**-provability of $|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$, which is done as follows:

1. $\bigsqcap x\bigsqcap y\big(|x+y|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=x+y)\big)$ Lemma 19.6
2. $\neg|\theta_1|\leq\mathfrak{b}\rightarrow\neg|\theta_1+\theta_2|\leq\mathfrak{b}$ **PA**
3. $\neg|\theta_1|\leq\mathfrak{b}\rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcup$-Choose: 2

4. $\neg|\theta_1|\leq\mathfrak{b}\wedge\big(|\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_2)\big)\wedge\bigsqcap x\bigsqcap y \big(|x+y|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=x+y)\big)\rightarrow |\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ Weakenings: 3
5. $\neg|\theta_2|\leq\mathfrak{b}\rightarrow\neg|\theta_1+\theta_2|\leq\mathfrak{b}$ **PA**
6. $\neg|\theta_2|\leq\mathfrak{b}\rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcup$-Choose: 5
7. $\bigsqcup z(z=\theta_1)\wedge\neg|\theta_2|\leq\mathfrak{b}\wedge\bigsqcap x\bigsqcap y\big(|x+y| \leq\mathfrak{b}\sqsupset\bigsqcup z(z=x+y)\big)\rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ Weakenings: 6
8. $t_1=\theta_1\wedge t_2=\theta_2\wedge\neg|t_1+t_2|\leq\mathfrak{b}\rightarrow\neg|\theta_1+\theta_2|\leq\mathfrak{b}$ Logical axiom
9. $t_1=\theta_1\wedge t_2=\theta_2\wedge\neg|t_1+t_2|\leq\mathfrak{b} \rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcup$-Choose: 8
10. $t_1=\theta_1\wedge t_2=\theta_2\wedge t=t_1+t_2\rightarrow t=\theta_1+\theta_2$ Logical axiom
11. $t_1=\theta_1\wedge t_2=\theta_2\wedge t=t_1+t_2\rightarrow\bigsqcup z(z=\theta_1+\theta_2)$ $\bigsqcup$-Choose: 10
12. $t_1=\theta_1\wedge t_2=\theta_2\wedge\bigsqcup z(z=t_1+t_2)\rightarrow \bigsqcup z(z=\theta_1+\theta_2)$ $\bigsqcap$-Introduction: 11
13. $t_1=\theta_1\wedge t_2=\theta_2\wedge\bigsqcup z(z=t_1+t_2)\rightarrow |\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcup$-Choose: 12
14. $t_1=\theta_1\wedge t_2=\theta_2\wedge\big(|t_1+t_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=t_1+t_2)\big) \rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcap$-Introduction: 9,13
15. $t_1=\theta_1\wedge t_2=\theta_2\wedge\bigsqcap x\bigsqcap y\big(|x+y|\leq\mathfrak{b} \sqsupset\bigsqcup z(z=x+y)\big)\rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\bigsqcup$-Chooses: 14
16. $\bigsqcup z(z=\theta_1)\wedge\bigsqcup z(z=\theta_2)\wedge\bigsqcap x\bigsqcap y\big(|x+y|\leq\mathfrak{b} \sqsupset\bigsqcup z(z=x+y)\big)\rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\bigsqcap$-Introductions: 15
17. $\bigsqcup z(z=\theta_1)\wedge\big(|\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_2)\big) \wedge\bigsqcap x\bigsqcap y\big(|x+y|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=x+y)\big) \rightarrow|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcap$-Introduction: 7,16
18. $\big(|\theta_1|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1)\big)\wedge\big(|\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_2)\big)\wedge \bigsqcap x\bigsqcap y\big(|x+y|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=x+y)\big)\rightarrow |\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ $\sqcap$-Introduction: 4,17
19. $|\theta_1+\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1+\theta_2)$ MP: (85),(86),1,18

Case 5: τ is $\theta_1\times\theta_2$ for some terms θ_1 and θ_2. Here we only outline a proof/solution for the target $|\theta_1\times\theta_2|\leq\mathfrak{b}\sqsupset\bigsqcup z(z=\theta_1\times\theta_2)$. Using the induction hypothesis (85) and Axiom 9,[19] we figure out whether $\theta_1=\mathbf{0}$

or not. If $\theta_1=\mathbf{0}$, then $\theta_1\times\theta_2$ is also $\mathbf{0}$, and we solve the target by choosing its right component $\sqcup z(z=\theta_1\times\theta_2)$ and then naming the value of $\mathbf{0}$ (which is found using Axiom 8) for z. Suppose now $\theta_1\neq\mathbf{0}$. Then we do for θ_2 the same as what we did for θ_1, and figure out whether $\theta_2=\mathbf{0}$ or $\theta_2\neq\mathbf{0}$. If $\theta_2=\mathbf{0}$, we solve the target as we did in the case $\theta_1=\mathbf{0}$. Suppose now θ_2, just like θ_1, does not equal to $\mathbf{0}$. Note that then the proof given in Case 4 goes through for our present case virtually without any changes, only with "$\times$" instead of "$+$" and "Lemma 19.7" instead of "Lemma 19.6". Indeed, the only steps of that proof that would be generally incorrect for $\times$ instead of $+$ are steps 2 and 5. Namely, the formula of step 2 is false when $\theta_2=\mathbf{0}$, and the formula of step 5 is false when $\theta_1=\mathbf{0}$. But, in the case that we are considering, these possibilities have been handled separately and by now are already ruled out. ■

19.8. The efficient computability of subtraction

The formula of the following lemma, as a computational problem, is about finding the difference z between any two numbers x and y and then telling whether this difference is $x-y$ or $y-x$.

Lemma 19.9 **PTA** $\vdash \sqcap x\sqcap y\sqcup z(x=y+z\sqcup y=x+z)$.

Proof. As we did in the case of Lemma 19.7, showing a proof idea or sketch instead of a detailed formal proof would be sufficient here. By BSI+ induction on s, we want to prove $\sqcap y\sqcup z(s=y+z\sqcup y=s+z)$, from which the target formula follows by $\sqcap$-Introduction.

The *basis*

$$\sqcap y\sqcup z(\mathbf{0}=y+z\sqcup y=\mathbf{0}+z) \qquad (87)$$

of induction is proven as follows:

1. $t=\mathbf{0}+t$ **PA**
2. $\mathbf{0}=t+t\sqcup t=\mathbf{0}+t$ $\sqcup$-Choose: 1
3. $\sqcup z(\mathbf{0}=t+z\sqcup t=\mathbf{0}+z)$ $\sqcup$-Choose: 2
4. $\sqcap y\sqcup z(\mathbf{0}=y+z\sqcup y=\mathbf{0}+z)$ $\sqcap$-Introduction: 3

The *inductive step* is

$$|s0|\leq \mathfrak{b}\wedge\sqcap y\sqcup z(s=y+z\sqcup y=s+z)\rightarrow\sqcap y\sqcup z(s0=y+z\sqcup y=s0+z)$$
$$\sqcap\sqcap y\sqcup z(s1=y+z\sqcup y=s1+z). \qquad (88)$$

To prove (88), it would be sufficient to prove the following two formulas, from which (88) follows by $\sqcap$-Introduction:

$$|s0|\leq \mathfrak{b}\wedge\sqcap y\sqcup z(s=y+z\sqcup y=s+z)\rightarrow\sqcap y\sqcup z(s0=y+z\sqcup y=s0+z); \qquad (89)$$

$$|s0|\leq \mathfrak{b}\wedge\sqcap y\sqcup z(s=y+z\sqcup y=s+z)\rightarrow\sqcap y\sqcup z(s1=y+z\sqcup y=s1+z). \qquad (90)$$

Let us focus on (89) only, as the case with (90) is similar. (89) follows from the following formula by $\sqcap$-Introduction:

$$|s0|\leq \mathfrak{b}\wedge\sqcap y\sqcup z(s=y+z\sqcup y=s+z)\rightarrow\sqcup z(s0=t+z\sqcup t=s0+z). \qquad (91)$$

A strategy for the above, which can eventually be translated into a bottom-up **PTA**-proof, is the following. Using Axiom 12, we find the binary predecessor r of t, and also determine whether $t=r0$ or $t=r1$.

Consider the case of $t=r0$. Solving (91) in this case essentially means solving

$$|s0|\leq \mathfrak{b}\wedge\sqcap y\sqcup z(s=y+z\sqcup y=s+z)\rightarrow\sqcup z(s0=r0+z\sqcup r0=s0+z). \qquad (92)$$

We can solve the above by using the second conjunct of the antecedent (specifying y as r in it) to find a w such that $s=r+w$ or $r=s+w$, with "or" here being a choice one, meaning that we will actually know which of the two alternatives is the case. Let us say the case is $s=r+w$ (with the other case being similar). From **PA** we know that, if $s=r+w$, then $s0=r0+w0$. So, in order to solve the consequent of (92), it would be sufficient to specify z as the value u of $w0$, and then choose the left $\sqcup$-disjunct $s0=r0+u$ of the resulting formula. Such a u can be computed using Axiom 11: $|w0|\leq\mathfrak{b}\rightarrow\sqcup x(x=w0)$, whose antecedent is true because, according to the first conjunct of the antecedent of (92), the size of $s0$ — and hence of $w0$ — does not exceed $\mathfrak{b}$.

The remaining case of $t=r1$ is similar, but it additionally requires proving $\sqcap x\big(x\neq\mathbf{0}\sqsupset\sqcup y(x=y')\big)$ (the efficient computability of unary predecessor), doing which is left as an exercise for the reader. ■

19.9. The efficient computability of "x's yth bit"

For any natural numbers n and i — as always identified with the corresponding binary numerals — we will write $(n)_i\,{=}\,0$ for a formula saying that $|n|{>}i$ and bit $\#i$ of n is 0. Similarly for $(n)_i\,{=}\,1$. In either case the count of the bits of n starts from 0 rather than 1, and proceeds from left to right rather than (as more common in the literature) from right to left. So, for instance, if $n = 100$, then 1 is its bit #0, and the 0s are its bits #1 and #2.

Lemma 19.10 **PTA** $\vdash \sqcap x \sqcap y\big(|x|{>}y \sqsupset (x)_y{=}0 \sqcup (x)_y{=}1\big)$.

Proof. We limit ourselves to providing an informal argument within **PTA**. The target formula follows by $\sqcap$-Introduction from $\sqcap y\big(|s|{>}y \sqsupset (s)_y{=}0 \sqcup (s)_y{=}1\big)$, and the latter we prove by BSI.

The basis of induction is

$$\sqcap y\big(|\mathbf{0}|{>}y \sqsupset (\mathbf{0})_y{=}0 \sqcup (\mathbf{0})_y{=}1\big). \tag{93}$$

Solving it is easy. Given any y, using Axiom 9, figure out whether $y{=}\mathbf{0}$ or $y{\neq}\mathbf{0}$. If $y{=}\mathbf{0}$, then resolve (93) by choosing $(\mathbf{0})_y{=}0$ in it. Otherwise, choose $\neg|\mathbf{0}|{>}y$.

The inductive step is

$$\sqcap y\big(|s|{>}y \sqsupset (s)_y{=}0 \sqcup (s)_y{=}1\big) \rightarrow$$
$$\sqcap y\big(|s0|{>}y \sqsupset (s0)_y{=}0 \sqcup (s0)_y{=}1\big) \sqcap \sqcap y\big(|s1|{>}y \sqsupset (s1)_y{=}0 \sqcup (s1)_y{=}1\big). \tag{94}$$

Solving it is not hard, either. It means solving the following two problems, from which (94) follows by first applying $\sqcup$-Choose, then $\sqcap$-Introduction and then $\sqcap$-Introduction:

$$|s|{>}r \sqsupset (s)_r{=}0 \sqcup (s)_r{=}1 \rightarrow |s0|{>}r \sqsupset (s0)_r{=}0 \sqcup (s0)_r{=}1; \tag{95}$$

$$|s|{>}r \sqsupset (s)_r{=}0 \sqcup (s)_r{=}1 \rightarrow |s1|{>}r \sqsupset (s1)_r{=}0 \sqcup (s1)_r{=}1. \tag{96}$$

To solve (95), wait till the environment selects one of the three $\sqcup$-disjuncts in the antecedent. If $(s)_r{=}0$ is selected, then select $(s0)_r{=}0$ in the consequent and you are done. Similarly, if $(s)_r{=}1$ is selected, then select $(s0)_r{=}1$ in the consequent. Suppose now $\neg|s|{>}r$ is selected. In this case, using Lemma 19.1, find the value of $|s|$ and then, using

Lemma 18.3, figure out whether $|s|{=}r$ or $|s|{\neq}r$. If $|s|{=}r$, then select $(s0)_r{=}0$ in the consequent of (95); otherwise, if $|s|{\neq}r$, select $\neg|s0|{>}r$ there.

The problem (96) is solved in a similar way, with the difference that, where in the previous case we selected $(s0)_r{=}0$, now $(s1)_r{=}1$ should be selected. ■

20. TWO MORE INDUCTION RULES

This section establishes the closure of **PTA** under two additional variations of the PTI and WPTI rules. These variations are not optimal as they could be made stronger,[20] nor are they natural enough to deserve special names. But these two rules, exactly in their present forms, will be relied upon later in Section 21. Thus, the present section is a purely technical one, and a less technically-minded reader may want to omit the proofs of its results.

Lemma 20.1 *The following rule is admissible in* **PTA***:*

$$\frac{R \rightarrow E(w) \wedge F(w) \qquad |t'|{\leq}\mathfrak{b} \wedge E(t) \wedge F(t) \rightarrow E(t) \wedge \big(E(t') \sqcap F(t')\big)}{R \wedge w{\leq}t{\leq}\tau \rightarrow E(t) \wedge F(t)},$$

where R is any elementary formula, w is any variable, t is any variable other than $\mathfrak{b}$, τ is any $\mathfrak{b}$-term, $E(t), F(t)$ are any formulas, $E(w)$ (resp. $E(t')$) is the result of replacing in $E(t)$ all free occurrences of t by w (resp. t'), and similarly for $F(w), F(t')$.

Idea. We manage reduce this rule to PTI by taking $R \wedge |w{+}s|{\leq}\mathfrak{b} \rightarrow E(w{+}s)$ and $R \wedge |w{+}s|{\leq}\mathfrak{b} \rightarrow F(w{+}s)$ in the roles of the formulas $E(s)$ and $F(s)$ of the latter. ■

Proof. Assume all conditions of the rule, and assume its premises are provable, i.e.,

$$\mathbf{PTA} \vdash R \rightarrow E(w) \wedge F(w); \tag{97}$$

$$\mathbf{PTA} \vdash |t'|{\leq}\mathfrak{b} \wedge E(t) \wedge F(t) \rightarrow E(t) \wedge \big(E(t') \sqcap F(t')\big). \tag{98}$$

Our goal is to show that **PTA** $\vdash R \wedge w{\leq}t{\leq}\tau \rightarrow E(t) \wedge F(t)$.

Let us agree on the following abbreviations:

$$\tilde{E}(s) \;=\; R \wedge |w+s| \leq \mathfrak{b} \to E(w+s); \qquad \tilde{F}(s) \;=\; R \wedge |w+s| \leq \mathfrak{b} \to F(w+s).$$

As easily seen, we have

$$\mathbf{CL4} \vdash f = w \wedge \big(p \to P(w) \wedge Q(w)\big) \to \big(p \wedge q \to P(f)\big) \wedge \big(p \wedge q \to Q(f)\big)$$

and hence, by **CL4**-Instantiation,

$$\mathbf{PTA} \vdash w + \mathbf{0} = w \wedge \big(R \to E(w) \wedge F(w)\big) \to \big(R \wedge |w+\mathbf{0}| \leq \mathfrak{b} \to E(w+\mathbf{0})\big) \wedge \big(R \wedge |w+\mathbf{0}| \leq \mathfrak{b} \to F(w+\mathbf{0})\big).$$

The above, together with (97) and the obvious fact **PA** $\vdash w+\mathbf{0}=w$, by Modus Ponens, yields

$$\mathbf{PTA} \vdash \big(R \wedge |w+\mathbf{0}| \leq \mathfrak{b} \to E(w+\mathbf{0})\big) \wedge \big(R \wedge |w+\mathbf{0}| \leq \mathfrak{b} \to F(w+\mathbf{0})\big),$$

i.e., using our abbreviations,

$$\mathbf{PTA} \vdash \tilde{E}(\mathbf{0}) \wedge \tilde{F}(\mathbf{0}). \tag{99}$$

With a little effort, the following can be seen to be a valid formula of classical logic:

$$\begin{array}{l}\big(|t'| \leq \mathfrak{b} \wedge p_1(t) \wedge q_1(t) \to p_2(t) \wedge q_2(t')\big) \to \\ t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to \\ \big(R \wedge |w+s| \leq \mathfrak{b} \to p_1(w+s)\big) \wedge \big(R \wedge |w+s| \leq \mathfrak{b} \to q_1(w+s)\big) \to \\ \big(R \wedge |w+s| \leq \mathfrak{b} \to p_2(w+s)\big) \wedge \big(R \wedge |w+s'| \leq \mathfrak{b} \to q_2(w+s')\big).\end{array}$$

Applying Match four times to the above formula, we find that **CL4** proves

$$\begin{array}{l}\big(|t'| \leq \mathfrak{b} \wedge P_1(t) \wedge Q_1(t) \to P_2(t) \wedge Q_2(t')\big) \to \\ t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to \\ \big(R \wedge |w+s| \leq \mathfrak{b} \to P_1(w+s)\big) \wedge \big(R \wedge |w+s| \leq \mathfrak{b} \to Q_1(w+s)\big) \to \\ \big(R \wedge |w+s| \leq \mathfrak{b} \to P_2(w+s)\big) \wedge \big(R \wedge |w+s'| \leq \mathfrak{b} \to Q_2(w+s')\big).\end{array} \tag{100}$$

Now we claim that

$$\mathbf{PTA} \vdash s \leq \tau \to \tilde{E}(s) \wedge \tilde{F}(s). \tag{101}$$

Below comes a justification of this claim:

1. $\neg|w+s'| \leq \mathfrak{b} \sqcup \bigsqcup z(z = w+s')$ Lemma 19.8

2. $\neg|w+s'| \leq \mathfrak{b} \to \tilde{E}(s) \wedge \tilde{F}(s) \to \tilde{E}(s) \wedge$
   $\Big(\big(R \wedge |w+s'| \leq \mathfrak{b} \to E(w+s')\big) \sqcap \big(R \wedge |w+s'| \leq \mathfrak{b} \to F(w+s')\big)\Big)$
   CL4-Instantiation, instance of $\neg p \to P \wedge Q_1 \to P \wedge \big((q \wedge p \to Q_2) \sqcap (q \wedge p \to Q_3)\big)$

3. $\neg|w+s'| \leq \mathfrak{b} \to \tilde{E}(s) \wedge \tilde{F}(s) \to \tilde{E}(s) \wedge \big(\tilde{E}(s') \sqcap \tilde{F}(s')\big)$ abbreviating 2

4. $|v| \leq \mathfrak{b}$ Axiom 13

5. $|v| \leq \mathfrak{b} \to v = w+s' \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s'$ **PA**

6. $v = w+s' \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s'$ MP: 4,5

7. $\neg|w+s| \leq \mathfrak{b} \sqcup \bigsqcup z(z = w+s)$ Lemma 19.8

8. $\neg|w+s| \leq \mathfrak{b} \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to$
   $\tilde{E}(s) \wedge \tilde{F}(s) \to \tilde{E}(s) \wedge \big(\tilde{E}(s') \sqcap \tilde{F}(s')\big)$
   CL4-Instantiation, instance of $\neg p \to q_1 \wedge p \wedge q_2 \to Q$

9. $\big(|t'| \leq \mathfrak{b} \wedge E(t) \wedge F(t) \to E(t) \wedge E(t')\big) \to t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b}$
   $\wedge (w+s)' = w+s' \to \big(R \wedge |w+s| \leq \mathfrak{b} \to E(w+s)\big) \wedge \big(R \wedge |w+s| \leq \mathfrak{b} \to F(w+s)\big)$
   $\to \big(R \wedge |w+s| \leq \mathfrak{b} \to E(w+s)\big) \wedge \big(R \wedge |w+s'| \leq \mathfrak{b} \to E(w+s')\big)$
   CL4-Instantiation, instance of (100)

10. $\big(|t'| \leq \mathfrak{b} \wedge E(t) \wedge F(t) \to E(t) \wedge E(t')\big) \to$
    $t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to \tilde{E}(s) \wedge \tilde{F}(s) \to \tilde{E}(s) \wedge \tilde{E}(s')$
    abbreviating 9

11. $\Big(|t'| \leq \mathfrak{b} \wedge E(t) \wedge F(t) \to E(t) \wedge \big(E(t') \sqcap F(t')\big)\Big) \to$
    $t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to \tilde{E}(s) \wedge \tilde{F}(s) \to \tilde{E}(s) \wedge \tilde{E}(s')$
    $\sqcup$-Choose: 10

12. $\big(|t'| \leq \mathfrak{b} \wedge E(t) \wedge F(t) \to E(t) \wedge F(t')\big) \to$
    $t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to$
    $\big(R \wedge |w+s| \leq \mathfrak{b} \to E(w+s)\big) \wedge \big(R \wedge |w+s| \leq \mathfrak{b} \to F(w+s)\big) \to$
    $\big(R \wedge |w+s| \leq \mathfrak{b} \to E(w+s)\big) \wedge \big(R \wedge |w+s'| \leq \mathfrak{b} \to F(w+s')\big)$
    CL4-Instantiation, instance of (100)

13. $\big(|t'| \leq \mathfrak{b} \wedge E(t) \wedge F(t) \to E(t) \wedge F(t')\big) \to$
    $t = w+s \to |w+s'| \leq \mathfrak{b} \wedge |w+s| \leq \mathfrak{b} \wedge (w+s)' = w+s' \to \tilde{E}(s) \wedge \tilde{F}(s) \to \tilde{E}(s) \wedge \tilde{F}(s')$
    abbreviating 12

14. $\Big(|t'|\leq\mathfrak{b}\wedge E(t)\wedge F(t)\rightarrow E(t)\wedge\big(E(t')\sqcap F(t')\big)\Big)\rightarrow$
$t=w+s\rightarrow|w+s'|\leq\mathfrak{b}\wedge|w+s|\leq\mathfrak{b}\wedge(w+s)'=w+s'\rightarrow\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\tilde{F}(s')$
$\sqcup$-Choose: 10

15. $\Big(|t'|\leq\mathfrak{b}\wedge E(t)\wedge F(t)\rightarrow E(t)\wedge\big(E(t')\sqcap F(t')\big)\Big)\rightarrow$
$t=w+s\rightarrow|w+s'|\leq\mathfrak{b}\wedge|w+s|\leq\mathfrak{b}\wedge(w+s)'=w+s'\rightarrow$
$\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$
$\sqcap$-Introduction: 11,14

16. $t=w+s\rightarrow|w+s'|\leq\mathfrak{b}\wedge|w+s|\leq\mathfrak{b}\wedge(w+s)'=w+s'\rightarrow$
$\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$ MP: (98),15

17. $\bigsqcup z(z=w+s)\rightarrow$
$|w+s'|\leq\mathfrak{b}\wedge|w+s\leq\mathfrak{b}|\wedge(w+s)'=w+s'\rightarrow\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$
$\bigsqcap$-Introduction: 16

18. $|w+s'|\leq\mathfrak{b}\wedge|w+s|\leq\mathfrak{b}\wedge(w+s)'=w+s'\rightarrow\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$
$\sqcup$-Elimination: 7,8,17

19. $v=w+s'\rightarrow\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$ TR: 6,18

20. $\bigsqcup z(z=w+s')\rightarrow\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$ $\bigsqcap$-Introduction: 19

21. $\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)$ $\sqcup$-Elimination: 1,3,20

22. $\tilde{E}(s)\wedge\big(\tilde{E}(s')\sqcap\tilde{F}(s')\big)\rightarrow\tilde{E}(s')\sqcap\big(\tilde{F}(s')\wedge\tilde{E}(s)\big)$ **CL4**-Instantiation

23. $\tilde{E}(s)\wedge\tilde{F}(s)\rightarrow\tilde{E}(s')\sqcap\big(\tilde{F}(s')\wedge\tilde{E}(s)\big)$ TR: 21,22

24. $s\leq\tau\rightarrow\tilde{E}(s)\wedge\tilde{F}(s)$ PTI: (99), 23

The following is a disabbreviation of (101):

$$\mathbf{PTA}\vdash s\leq\tau\rightarrow\big(R\wedge|w+s|\leq\mathfrak{b}\rightarrow E(w+s)\big)\wedge\big(R\wedge|w+s|\leq\mathfrak{b}\rightarrow F(w+s)\big).$$

It is easy to see that, by **CL4**-Instantiation, we also have

$$\mathbf{PTA}\vdash\Big(s\leq\tau\rightarrow\big(R\wedge|w+s|\leq\mathfrak{b}\rightarrow E(w+s)\big)\wedge\big(R\wedge|w+s|\leq\mathfrak{b}\rightarrow F(w+s)\big)\Big)\rightarrow$$
$$s\leq\tau\wedge R\wedge|w+s|\leq\mathfrak{b}\rightarrow E(w+s)\wedge F(w+s).$$

Hence, by Modus Ponens,

$$\mathbf{PTA}\vdash s\leq\tau\wedge R\wedge|w+s|\leq\mathfrak{b}\rightarrow E(w+s)\wedge F(w+s).\qquad(102)$$

Now, the following sequence is an **PTA**-proof of the target formula $R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$, which completes our proof of the present lemma:

1. $\bigsqcap x\bigsqcap y\bigsqcup z(x=y+z\sqcup y=x+z)$ Lemma 19.9
2. $\bigsqcup z(w=t+z\sqcup t=w+z)$ $\bigsqcap$-Elimination (twice): 1
3. $s=\mathbf{0}\sqcup s\neq\mathbf{0}$ Axiom 8
4. $s=\mathbf{0}\rightarrow(w=t+s\rightarrow t=w)$ **PA**
5. $\big(R\rightarrow E(w)\wedge F(w)\big)\rightarrow(w=t+s\rightarrow t=w)\rightarrow\big(w=t+s\rightarrow R\rightarrow E(t)\wedge F(t)\big)$
CL4-Instantiation
6. $(w=t+s\rightarrow t=w)\rightarrow\big(w=t+s\rightarrow R\rightarrow E(t)\wedge F(t)\big)$ MP: (97),5
7. $s=\mathbf{0}\rightarrow w=t+s\rightarrow R\rightarrow E(t)\wedge F(t)$ TR: 4,6
8. $s=\mathbf{0}\rightarrow w=t+s\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ Weakening: 7
9. $s\neq\mathbf{0}\rightarrow w=t+s\rightarrow\neg w\leq t\leq\tau$ **PA**
10. $s\neq\mathbf{0}\rightarrow w=t+s\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ Weakenings: 9
11. $w=t+s\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ $\sqcup$-Elimination: 3,8,10
12. $|t|\leq\mathfrak{b}$ Axiom 13
13. $t=w+s\wedge w\leq t\leq\tau\rightarrow s\leq\tau$ **PA**
14. $|t|\leq\mathfrak{b}\wedge(t=w+s\wedge w\leq t\leq\tau\rightarrow s\leq\tau)\wedge\big(s\leq\tau\wedge R\wedge|w+s|\leq\mathfrak{b}\rightarrow E(w+s)\wedge F(w+s)\big)$
$\rightarrow t=w+s\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$
CL4-Instantiation
15. $t=w+s\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ MP: 12,13,(102),14
16. $w=t+s\sqcup t=w+s\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ $\sqcap$-Introduction: 11,15
17. $\bigsqcup z(w=t+z\sqcup t=w+z)\rightarrow R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ $\bigsqcap$-Introduction: 16
18. $R\wedge w\leq t\leq\tau\rightarrow E(t)\wedge F(t)$ MP: 2,17 ■

Lemma 20.2 *The following rule is admissible in* **PTA**:

$$\frac{R\rightarrow F(w)\qquad R\wedge w\leq t<\tau\wedge F(t)\rightarrow F(t')}{R\wedge w\leq t\leq\tau\rightarrow F(t)},$$

where R is any elementary formula, w is any variable, t is any variable *not occurring in R and different from $\mathfrak{b}$, $F(t)$ is any formula, τ is any $\mathfrak{b}$-term, and $F(w)$ (resp. $F(t')$) is the result of replacing in $F(t)$ all free occurrences of t by w (resp. t').*

Idea. This rule can be reduced to the rule of Lemma 20.1 by taking $\top$ and $R\wedge w\leq t\leq\tau\to F(t)$ in the roles of $E(t)$ and $F(t)$ of the latter, respectively. ■

Proof. Assume all conditions of the rule, and assume its premises are provable, i.e.,

$$\textbf{PTA}\vdash R\to F(w); \tag{103}$$

$$\textbf{PTA}\vdash R\wedge w\leq t<\tau\wedge F(t)\to F(t'). \tag{104}$$

Our goal is to show that $\textbf{PTA}\vdash R\wedge w\leq t\leq\tau\to F(t)$.

Let us agree on the following abbreviation:

$$\tilde{F}(t)\ =\ R\wedge w\leq t\leq\tau\to F(t).$$

From (103), by Weakening, we have

$$\textbf{PTA}\vdash R\to\tilde{F}(w). \tag{105}$$

We now claim that

$$\textbf{PTA}\vdash |t'|\leq\mathfrak{b}\wedge\top\wedge\tilde{F}(t)\to\top\wedge\big(\top\sqcap\tilde{F}(t')\big). \tag{106}$$

This claim is justified a follows:

1. $\neg|t'|\leq\mathfrak{b}\sqcup\bigsqcup z(z=t')$ Lemma 19.8
2. $|w|\leq\mathfrak{b}$ Axiom 13
3. $|w|\leq\mathfrak{b}\to\neg|t'|\leq\mathfrak{b}\to t'\neq w$ Logical axiom
4. $\neg|t'|\leq\mathfrak{b}\to t'\neq w$ MP: 2,3
5. $\neg|t'|\leq\mathfrak{b}\to t'=w\sqcup t'\neq w$ $\sqcup$-Choose: 4
6. $\sqcap x\sqcap y(y=x\sqcup y\neq x)$ Lemma 18.3
7. $r=w\sqcup r\neq w$ $\sqcap$-Elimination (twice): 6
8. $r=w\to r=t'\to t'=w$ Logical axiom
9. $r=w\to r=t'\to t'=w\sqcup t'\neq w$ $\sqcup$-Choose: 8
10. $r\neq w\to r=t'\to t'\neq w$ Logical axiom

11. $r\neq w\to r=t'\to t'=w\sqcup t'\neq w$ $\sqcup$-Choose: 10
12. $r=t'\to t'=w\sqcup t'\neq w$ $\sqcup$-Elimination: 7,9,11
13. $\bigsqcup z(z=t')\to t'=w\sqcup t'\neq w$ $\sqcap$-Introduction: 12
14. $t'=w\sqcup t'\neq w$ $\sqcup$-Elimination: 1,5,13
15. $\big(R\to F(w)\big)\to t'=w\to\big(R\to F(t')\big)$ **CL4**-Instantiation
16. $t'=w\to\big(R\to F(t')\big)$ MP: (103),15
17. $t'=w\to\big(R\wedge w\leq t\leq\tau\to F(t)\big)\to\big(R\wedge w\leq t'\leq\tau\to F(t')\big)$ Weakenings: 16
18. $t'\neq w\to(w\leq t'\leq\tau\to w\leq t\leq\tau\wedge w\leq t<\tau)$ **PA**
19. $\big(R\wedge w\leq t<\tau\wedge F(t)\to F(t')\big)\to$
$(w\leq t'\leq\tau\to w\leq t\leq\tau\wedge w\leq t<\tau)\to\big(R\wedge w\leq t\leq\tau\to F(t)\big)\to$
$\big(R\wedge w\leq t'\leq\tau\to F(t')\big)$
CL4-Instantiation, instance of $(q\wedge p_3\wedge P\to Q)\to(p_1\to p_2\wedge p_3)\to$ $(q\wedge p_2\to P)\to(q\wedge p_1\to Q)$
20. $(w\leq t'\leq\tau\to w\leq t\leq\tau\wedge w\leq t<\tau)\to$
$\big(R\wedge w\leq t\leq\tau\to F(t)\big)\to\big(R\wedge w\leq t'\leq\tau\to F(t')\big)$ MP: (104),19
21. $t'\neq w\to\big(R\wedge w\leq t\leq\tau\to F(t)\big)\to\big(R\wedge w\leq t'\leq\tau\to F(t')\big)$ TR: 18,20
22. $\big(R\wedge w\leq t\leq\tau\to F(t)\big)\to$
$\big(R\wedge w\leq t'\leq\tau\to F(t')\big)$ $\sqcup$-Elimination: 14,17,21
23. $\tilde{F}(t)\to\tilde{F}(t')$ abbreviating 22
24. $|t'|\leq\mathfrak{b}\wedge\top\wedge\tilde{F}(t)\to\tilde{F}(t')$ Weakenings: 23
25. $\tilde{F}(t')\to\top\wedge\big(\top\sqcap\tilde{F}(t')\big)$ **CL4**-Instantiation
26. $|t'|\leq\mathfrak{b}\wedge\top\wedge\tilde{F}(t)\to\top\wedge\big(\top\sqcap\tilde{F}(t')\big)$ TR: 24,25

From (105) and (106), by the rule of Lemma 20.1, we get $\textbf{PTA}\vdash R\wedge w\leq t\leq\tau\to\top\wedge\tilde{F}(t)$. Of course (by **CL4**-Instantiation) $\textbf{PTA}\vdash\top\wedge\tilde{F}(t)\to\tilde{F}(t)$, so, by Transitivity, $\textbf{PTA}\vdash R\wedge w\leq t\leq\tau\to\tilde{F}(t)$. Disabbreviating the latter, we thus have

$$\textbf{PTA}\vdash R\wedge w\leq t\leq\tau\to R\wedge w\leq t\leq\tau\to F(t).$$

We also have

$$\textbf{PTA}\vdash\big(R\wedge w\leq t\leq\tau\to R\wedge w\leq t\leq\tau\to F(t)\big)\to\big(R\wedge w\leq t\leq\tau\to F(t)\big)$$

(the above formula is an instance of the obviously **CL4**-provable $(p \to p \to Q) \to (p \to Q)$). So, by Modus Ponens, we find that **PTA** proves the desired $R \wedge w \le t \le \tau \to F(t)$.

■

21. THE EXTENSIONAL COMPLETENESS OF PTA

This section is devoted to proving the completeness part of Theorem 12.3. It means showing that, for any arithmetical problem A that has a polynomial time solution, there is a theorem of **PTA** which, under the standard interpretation, equals ("expresses") A.

So, let us pick an arbitrary polynomial-time-solvable arithmetical problem A. By definition, A is an arithmetical problem because, for some formula X of the language of **PTA**, $A = X^{\dagger}$. For the rest of this section, we fix such a formula

$$X,$$

and fix

$$\mathcal{X}$$

as an HPM that solves A (and hence $X^{\dagger}$) in polynomial time. Specifically, we assume that $\mathcal{X}$ runs in time

$$\xi(\mathfrak{b}),$$

where $\xi(\mathfrak{b})$, which we also fix for the rest of this section and which sometimes can be written simply as ξ, is a $\mathfrak{b}$-term (a term containing no variables other than $\mathfrak{b}$).

X may not necessarily be provable in **PTA**, and our goal is to construct another formula $\overline{X}$ for which, just like for X, we have $A = \overline{X}^{\dagger}$ and which, perhaps unlike X, is provable in **PTA**.

Remember our convention about identifying formulas of ptarithmetic with (the games that are) their standard interpretations. So, in the sequel, just as we have done so far, we shall typically write $E, F, \ldots$ to mean either $E, F, \ldots$ or $E^{\dagger}, F^{\dagger}, \ldots$. Similar conventions apply to terms as well. In fact, we have just used this convention when saying that $\mathcal{X}$ runs in time ξ. What was really meant was that it runs in time $\xi^{\dagger}$.

21.1. Preliminary insights

Our proof is long and, in the process of going through it, it is easy to get lost in the forest and stop seeing it for the trees. Therefore, it might be worthwhile to try to get some preliminary insights into the basic idea behind this proof before venturing into its details.

Let us consider the simplest nontrivial special case where X is

$$Y(x) \sqcup Z(x)$$

for some elementary formulas $Y(x)$ and $Z(x)$ (perhaps $Z(x)$ is $\neg Y(x)$, in which case X expresses an ordinary decision problem — the problem of deciding the predicate $Y(x)$).

The assertion "$\mathcal{X}$ does not win X in time ξ" can be formalized in the language of **PA** through as a certain formula $\mathbb{L}$. Then we let the earlier mentioned $\overline{X}$ be the formula

$$\big(Y(x) \vee \mathbb{L}\big) \sqcup \big(Z(x) \vee \mathbb{L}\big).$$

Since $\mathcal{X}$ *does* win game X in time ξ, $\mathbb{L}$ is false. Hence $Y(x) \vee \mathbb{L}$ is equivalent to $Y(x)$, and $Z(x) \vee \mathbb{L}$ is equivalent to $Z(x)$. This means that $\overline{X}$ and X, as games, are the same, that is, $\overline{X}^{\dagger} = X^{\dagger}$. It now remains to understand why **PTA** $\vdash \overline{X}$.

A central lemma here is one establishing that the work of $\mathcal{X}$ is "*provably traceable*". Roughly, this means the provability of the fact that, for any time moment $t \le \xi(\mathfrak{b})$, we can tell ("can tell" formally indicated with $\sqcup$ or $\bigsqcup$ applied to the possible alternatives) the state in which $\mathcal{X}$ will be, the locations of its three scanning heads, and the content of any of the cells of any of the three tapes. Letting $\mathcal{X}$ work for $\xi(\mathfrak{b})$ steps, one of the following four eventual scenarios should take place, and the provable traceability of the work of $\mathcal{X}$ can be shown to imply that **PTA** proves the $\sqcup$-disjunction of formulas describing those scenarios:

Scenario 1: $\mathcal{X}$ makes the move 0 (and no other moves).

Scenario 2: $\mathcal{X}$ makes the move 1 (and no other moves).

Scenario 3: $\mathcal{X}$ does not make any moves.

Scenario 4: $\mathcal{X}$ makes an illegal move (perhaps after first making a legal move 0 or 1).

In the case of Scenario 1, the play over $\overline{X}$ hits $Y(x)\vee\mathbb{L}$. And **PTA** — in fact, **PA** — proves that, in this case, $Y(x)\vee\mathbb{L}$ is true. The truth of $Y(x)\vee\mathbb{L}$ is indeed very easily established: if it was false, then $Y(x)$ should be false, but then the play of $\mathcal{X}$ over X (which, as a game, is the same as $\overline{X}$) hits the false $Y(x)$ and hence is lost, but then $\mathbb{L}$ is true, but then $Y(x)\vee\mathbb{L}$ is true. Thus, $\textbf{PTA}\vdash(\textit{Scenario 1})\rightarrow Y(x)\vee\mathbb{L}$, from which, by $\sqcup$-Choose, $\textbf{PTA}\vdash(\textit{Scenario 1})\rightarrow\overline{X}$.

The case of Scenario 2 is symmetric.

In the case of Scenario 3, (**PTA** proves that) $\mathcal{X}$ loses, i.e. $\mathbb{L}$ is true, and hence, say, $Y(x)\vee\mathbb{L}$ (or $Z(x)\vee\mathbb{L}$ if you like) is true. That is, $\textbf{PTA}\vdash(\textit{Scenario 3})\rightarrow Y(x)\vee\mathbb{L}$, from which, by $\sqcup$-Choose, $\textbf{PTA}\vdash(\textit{Scenario 3})\rightarrow\overline{X}$.

The case of Scenario 4 is similar.

Thus, for each $i\in\{1,2,3,4\}$, $\textbf{PTA}\vdash(\textit{Scenario i})\rightarrow\overline{X}$. And, as we also have

$$\textbf{PTA}\vdash(\textit{Scenario 1})\sqcup(\textit{Scenario 2})\sqcup(\textit{Scenario 3})\sqcup(\textit{Scenario 4}),$$

by $\sqcup$-Elimination, we find the desired $\textbf{PTA}\vdash\overline{X}$.

The remaining question to clarify is how the provable traceability of the work of $\mathcal{X}$ is achieved. This is where PTI comes into play. In the roles of the two formulas E and F of that rule we employ certain nonelementary formulas $\mathbb{E}$ and $\mathbb{F}$. With t being the "current time", $\mathbb{E}(t)$ is a formula which, as a resource, allows us to tell ($\sqcup$ or $\bigsqcup$) the current state of $\mathcal{X}$, and ($\wedge$) the locations of its three heads, and ($\wedge$) the contents of the three cells under the three heads. And $\mathbb{F}(t)$ allows us, for any ($\bigsqcap$) cell of any ($\wedge$) tape, to tell ($\sqcup$) its current content.

In order to resolve $\mathbb{F}(t')$ — that is, to tell the content of any ($\bigsqcap$) given cell $\#c$ at time $t+1$ — all we need to know is the state of $\mathcal{X}$, the content of cell $\#c$, the locations of the scanning heads (perhaps only one of them), and the contents of the three cells scanned by the three heads at time t. The content of cell $\#c$ at time t can be obtained from (a single copy of) the resource $\mathbb{F}(t)$, and the rest of the above information from (a single copy of) the resource $\mathbb{E}(t)$. **PTA** is aware of this, and proves $\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\mathbb{F}(t')$.

Similarly, it turns out that, in order to resolve $\mathbb{E}(t')$, a single copy of $\mathbb{E}(t)$ and a single copy of $\mathbb{F}(t)$ are sufficient, and **PTA**, being aware of this, proves $\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\mathbb{E}(t')$.

The above two provabilities, by $\sqcap$-Introduction, imply $\textbf{PTA}\vdash\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\mathbb{E}(t')\sqcap\mathbb{F}(t')$. This is almost the inductive step of PTI. What is missing is a $\wedge$-conjunct $\mathbb{E}(t)$ in the consequent. Not to worry. Unlike $\mathbb{F}(t)$, $\mathbb{E}(t)$ is a recyclable resource due to the fact that it does not contain $\sqcap$ or $\bigsqcap$ (albeit it contains $\sqcup,\bigsqcup$). Namely, once we learn — from the antecedental resource $\mathbb{E}(t)$ — about the state of $\mathcal{X}$, the locations of the three scanning heads and the cell contents at those locations at time t, we can use/recycle that information and "return/resolve back" $\mathbb{E}(t)$ in the consequent. A syntactic equivalent — or rather consequence — of what we just said is that the provability of $\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\mathbb{E}(t')\sqcap\mathbb{F}(t')$ implies the provability of $\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\big(\mathbb{E}(t')\sqcap\mathbb{F}(t')\big)\wedge\mathbb{E}(t)$, and hence also the provability of the weaker $\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\mathbb{E}(t')\sqcap\big(\mathbb{F}(t')\wedge\mathbb{E}(t)\big)$.

Thus, $\textbf{PTA}\vdash\mathbb{E}(t)\wedge\mathbb{F}(t)\rightarrow\mathbb{E}(t')\sqcap\big(\mathbb{F}(t')\wedge\mathbb{E}(t)\big)$. We also have $\textbf{PTA}\vdash\mathbb{E}(\mathbf{0})\wedge\mathbb{F}(\mathbf{0})$, as this formula is essentially just a description of the initial configuration of the machine. Then, by PTI, $\textbf{PTA}\vdash t\leq\xi(\mathfrak{b})\rightarrow\mathbb{E}(t)\wedge\mathbb{F}(t)$. This is exactly what we meant by the provable traceability of the work of $\mathcal{X}$.

The above was about the pathologically simple case of $X=Y(x)\sqcup Z(x)$, and the general case will be much more complex, of course. Among other things, provable traceability would have to account for the possibility of the environment making moves now and then. And showing the provability of $\overline{X}$ would require a certain metainduction on its complexity, which we did not need in the present case. But the idea that we have just tried to explain would still remain valid and central, only requiring certain — nontrivial but doable — adjustments and refinements.

21.2. *The overline notation*

Throughout the rest of this section, we assume that the formula X has no free occurrences of variables other than $\mathfrak{b}$. There is no loss of generality in making such an assumption, because, if X does not satisfy this condition, it can be replaced by the both semantically and deductively equivalent $\bigsqcap$-closure of it over all free variables different from $\mathfrak{b}$.

We shall sometimes find it helpful to write X as

$$X(\mathfrak{b}).$$

When, after that, writing $X(b)$ (where b is a constant), one should keep in mind that it means the result of substituting $\mathfrak{b}$ by b in $X(\mathfrak{b})$ not only where we explicitly see $\mathfrak{b}$, but also in choice quantifiers $\sqcap$ and $\sqcup$, which, as we remember, are lazy ways to write $\sqcap^{\mathfrak{b}}$ and $\sqcup^{\mathfrak{b}}$. So, for instance, if $X(\mathfrak{b})$ is $\sqcup x E(x)$ and $c \neq b$, then $X(c)$ is not the same as $X(b)$ even if $\mathfrak{b}$ does not occur in $E(x)$, because the former is $\sqcup^c x E(x)$ and the latter is $\sqcup^b x E(x)$. The same applies to any formula written in the form $F(\mathfrak{b}, \ldots)$, of course.

Let us say that a formula is **safe** iff no two occurrences of quantifiers in it bind the same variable. For simplicity and also without loss of generality, we further assume that the formula X is safe (otherwise make it safe by renaming variables).

Since X has no free variables other than $\mathfrak{b}$, for simplicity we can limit our considerations to valuations that send every non-$\mathfrak{b}$ variable to 0. We call such valuations **standard** and use a special notation for them. Namely, for an integer b, we write

$$e_b$$

for the valuation such that $e_b(\mathfrak{b}) = b$ and, for any other variable v, $e_b(v) = 0$.

By a **politeral** of a formula we mean a positive occurrence of a literal in it. While a politeral is not merely a literal but a literal L *together* with a fixed occurrence, we shall often refer to it just by the name L of the literal, assuming that it is clear from the context which (positive) occurrence of L is meant.

We assume that the reader is sufficiently familiar with Gödel's technique of encoding and arithmetizing. Using that technique, we can construct a sentence

$$\mathbb{L}$$

of the language of **PA** which asserts — more precisely, implies — "$\mathcal{X}$ does not win X in time ξ".

Namely, let $E_1(\mathfrak{b}, \vec{x}), \ldots, E_n(\mathfrak{b}, \vec{x})$ be all subformulas of X, where all free variables of each $E_i(\mathfrak{b}, \vec{x})$ are among $\mathfrak{b}, \vec{x}$ (but not necessarily vice

versa). Then the above sentence $\mathbb{L}$ is a natural formalization of the following statement:

> *"There is a (finite) run Γ generated by $\mathcal{X}$ on some standard bounded valuation e_b such that:*
>
> *(1) $\top$'s time in Γ is not smaller than $\xi(b)$, or*
>
> *(2) Γ is a $\top$-illegal run of $X(b)$, or*
>
> *(3) Γ is a legal run of $X(b)$ and there is a tuple $\vec{c}$ of constants ($\vec{c}$ of the same length as $\vec{x}$) such that:*
> - *$\langle\Gamma\rangle X(b) = E_1(b, \vec{c})$, and we have $\neg\|E_1(b, \vec{c})\|$ (i.e., $\|E_1(b, \vec{c})\|$ is false),*
> - *or ..., or*
> - *$\langle\Gamma\rangle X(b) = E_n(b, \vec{c})$, and we have $\neg\|E_n(b, \vec{c})\|$ (i.e., $\|E_n(b, \vec{c})\|$ is false)."*

As we remember, our goal is to construct a formula $\overline{X}$ which expresses the same problem as X does and which is provable in **PTA**. For any formula E — including X — we let

$$\overline{E}$$

be the result of replacing in E every politeral L by $L \vee \mathbb{L}$.

Lemma 21.1 *Any literal L is equivalent (in the standard model of arithmetic) to $L \vee \mathbb{L}$.*

Proof. That L implies $L \vee \mathbb{L}$ is immediate, as the former is a disjunct of the latter. For the opposite direction, suppose $L \vee \mathbb{L}$ is true at a given valuation e. Its second disjunct cannot be true, because $\mathcal{X}$ *does* win X in time ξ, contrary to what $\mathbb{L}$ asserts. So, the first disjunct, i.e. L, is true. ■

Lemma 21.2 *For any formula E, including X, we have $E^{\dagger} = \overline{E}^{\dagger}$.*

Proof. Immediately from Lemma 21.1 by induction on the complexity of E. ■

In view of the above lemma, what now remains to do for the completion of our completeness proof is to show that **PTA** $\vdash \overline{X}$. The rest of the present section is entirely devoted to this task.

21.3. *This and that*

Lemma 21.3 *For any formula E,* $\mathbf{PTA} \vdash \mathbb{L} \to \overline{E}$.

Idea. $\overline{E}$ is a logical combination of "quasipoliterals" of the form $L \vee \mathbb{L}$. Under the assumption (of the truth of) $\mathbb{L}$, each such quasipoliteral becomes true and, correspondingly, $\overline{E}$ essentially becomes a logical combination of $\top$s. Any such combination is very easy to solve/prove. ■

Proof. We prove this lemma by induction on the complexity of E.

If E has the form $E[H_1 \sqcup \ldots \sqcup H_n]$, then, by the induction hypothesis, $\mathbf{PTA} \vdash \mathbb{L} \to \overline{E[H_1]}$. From here, by $\sqcup$-Choose, we get the desired $\mathbf{PTA} \vdash \mathbb{L} \to \overline{E[H_1 \sqcup \ldots \sqcup H_n]}$.

Quite similarly, if E has the form $E[\bigsqcup x H(x)]$, then, by the induction hypothesis, $\mathbf{PTA} \vdash \mathbb{L} \to \overline{E[H(v)]}$ (for whatever variable v you like). From here, by $\bigsqcup$-Choose, we get $\mathbf{PTA} \vdash \mathbb{L} \to \overline{E[\bigsqcup x H(x)]}$.

Now assume E has no surface occurrences of $\sqcup$- and $\bigsqcup$-subformulas. The formula $\|\overline{E}\|$ is a $(\wedge, \vee, \forall, \exists)$-combination of $\top$s (originating from $\sqcap$- and $\bigsqcap$-subformulas when elementarizing $\overline{E}$) and formulas $L \vee \mathbb{L}$ (originating from L when transferring from E to $\overline{E}$) where L is a politeral of E. $\top$ is true. If $\mathbb{L}$ is true, then each $L \vee \mathbb{L}$ is also true no matter what the values of the variables of L are (if L contains any variables at all). Therefore, clearly, $\|\overline{E}\|$, as a $(\wedge, \vee, \forall, \exists)$-combination of (always) true formulas, is true. Formalizing this argument in **PA** and hence in **PTA** yields $\mathbf{PTA} \vdash \mathbb{L} \to \|\overline{E}\|$, which, taking into account that $\mathbb{L}$ is an elementary formula and hence $\mathbb{L} = \|\mathbb{L}\|$, is the same as to say that

$$\mathbf{PTA} \vdash \|\mathbb{L} \to \overline{E}\|. \tag{107}$$

Suppose E has the form $E[H_1 \sqcap \ldots \sqcap H_n]$. Then, by the induction hypothesis, **PTA** proves $\mathbb{L} \to \overline{E[H_i]}$ for each $i \in \{1, \ldots, n\}$. Similarly, suppose E has the form $E[\bigsqcap x H(x)]$. Let v be a variable different from $\mathfrak{b}$ and not occurring in $E[\bigsqcap x H(x)]$. Then, again by the induction hypothesis, **PTA** proves $\mathbb{L} \to \overline{E[H(v)]}$. These observations, together with (107), by Wait, yield the desired $\mathbf{PTA} \vdash \mathbb{L} \to \overline{E}$. ■

We shall say that a run generated by the machine $\mathcal{X}$ is **prompt** iff $\bot$'s time in it is 0. In a prompt run, the environment always reacts to a

move by $\mathcal{X}$ instantaneously (on the same clock cycle as that on which $\mathcal{X}$ moved), or does not react at all. An exception is clock cycle #0, on which the environment can move even if $\mathcal{X}$ did not move. Such runs are convenient to deal with, because in them $\top$'s time equals the timestamp of the last move. And this, in turn, means that no moves by either player are made at any time greater or equal to $\xi(b)$, where b is the value assigned to $\mathfrak{b}$ by the valuation spelled on the valuation tape of the machine.

By our assumption, $\mathcal{X}$ wins X (in time ξ), meaning that every run Γ generated by $\mathcal{X}$ on a bounded valuation e is a $\top$-won run of $e[X]$, including the cases when Γ is prompt and e is standard. This allows us to focus on prompt runs and standard valuations only. Specifically, we are going to show that $\overline{X}$ is provable because $\mathcal{X}$ wins (in time ξ) every prompt run of X on every standard bounded valuation.

Further, for our present purposes, environment's possible strategies can be understood as (limited to) fixed/predetermined behaviors seen as finite sequences of moves with non-decreasing timestamps. Let us call such sequences **counterbehaviors**. The meaning of a counterbehavior

$$\langle(\alpha_1, t_1), (\alpha_2, t_2), \ldots, (\alpha_n, t_n)\rangle$$

is that the environment makes move α_1 at time t_1, move α_2 at time t_2, ..., move α_n at time t_n. If two consecutive moves have the same timestamp, the moves are assumed to be made (appear in the run) in the same order as they are listed in the counterbehavior.

Given a standard valuation e and a counterbehavior $C = \langle(\alpha_1, t_1), \ldots, (\alpha_n, t_n)\rangle$, by the (C, e)**-branch** we mean the e-computation branch of $\mathcal{X}$ where the environment acts according to C — that is, makes move α_1 at time t_1, ..., move α_n at time t_n. And the (C, e)**-run** is the run spelled by this branch.

For natural numbers b and d, we say that a counterbehavior C is (b, d)**-adequate** iff the following three conditions are satisfied:

(1) the (C, e_b)-run is not a $\bot$-illegal run of $X(b)$;

(2) the (C, e_b)-run is prompt;

(3) the timestamp of the last move of C (if C is nonempty) is less than d.

Thus, “C is (b,d)-adequate” means that, using this counterbehavior against $\mathcal{X}$ with e_b on the valuation tape of the latter, the environment has played legally (condition 1), acted fast/promptly (condition 2), and made all (if any) moves before time d (condition 3).

Just as with any finite objects, counterbehaviors can be encoded through natural numbers. The **code** (**Gödel number**) of an object O will be denoted by

$$\ulcorner O \urcorner.$$

Under any encoding, the size of the code of a counterbehavior of interest will generally exceed the value of $\mathfrak{b}$. But this is not going to be a problem as we will quantify counterbehaviors using blind rather than choice quantifiers.

For convenience, we assume that every natural number is the code of some counterbehavior. This allows us to terminologically identify counterbehaviors with their codes, and say phrases like “a is a (b,d)-adequate counterbehavior” — as done below — which should be understood as “Where C is the counterbehavior with $a = \ulcorner C \urcorner$, C is (b,d)-adequate”. Similarly, “the (a,e)-branch” (or “the (a,e)-run”) will mean “the (C,e)-branch (or (C,e)-run) where C is the counterbehavior with $a = \ulcorner C \urcorner$”.

Let $E = E(\mathfrak{b},\vec{s})$ be a formula all of whose free variables are among $\mathfrak{b},\vec{s}$ (but not necessarily vice versa). We will write

$$\mathbb{W}^E(z,t_1,t_2,\mathfrak{b},\vec{s})$$

to denote an elementary formula whose free variables are exactly (the pairwise distinct) $z,t_1,t_2,\mathfrak{b},\vec{s}$, and which is a natural arithmetization of the predicate which, for any constants $a,d_1,d_2,b,\vec{c}$, holds — that is, $\mathbb{W}^E(a,d_1,d_2,b,\vec{c})$ is true — iff the following conditions are satisfied:

- $0 < d_1 \leq d_2 \leq \xi(b)$;
- a is a (b,d_1)-adequate counterbehavior;
- where Φ is the initial segment of the (a,e_b)-run obtained from the latter by deleting all moves except those whose timestamps are less than d_1, Φ is a legal position of $E(b,\vec{c})$;
- for the above Φ, we have $\langle\Phi\rangle X(b) = E(b,\vec{c})$, and **PA** proves this fact;

- either $d_1 = 1$ or, in the (a,e_b)-branch, $\mathcal{X}$ has made some move at time $d_1 - 1$ (so that the effect of that move first took place at time d_1);
- for any k with $d_1 \leq k < d_2$, no move is made at time k (i.e. no move has the timestamp k) in the (a,e_b)-run.

Thus, in the context of the (a,e_b)-branch, $\mathbb{W}^E(a,d_1,d_2,b,\vec{c})$ says that, exactly by time d_1,[21] the play has hit the position $E(b,\vec{c})$, and that this position has remained stable (there were no moves to change it) throughout the interval $[d_1,d_2]$. It does not rule out that a move was made at time d_2 but, as we remember, the effect of such a move will take place by time d_2+1 rather than d_2.

It may be worthwhile to comment on the meaning of the above for the special case where t_2 is $\xi(\mathfrak{b})$. Keeping in mind that $\mathcal{X}$ runs in time $\xi(\mathfrak{b})$, the formula

$$\mathbb{W}^E(z,t,\xi(\mathfrak{b}),\mathfrak{b},\vec{s}),$$

for any given values $a,d,b,\vec{c}$ for $z,t,\mathfrak{b},\vec{s}$, asserts — or rather implies — that, in the scenario of the (a,e_b)-branch, at time d, the play (position to which X has evolved) hits $E(b,\vec{c})$ and remains stable ever after, so that $E(b,\vec{c})$ is the final, ultimate position of the play.

We say that a formula E or the corresponding game is **critical** iff one of the following conditions is satisfied:

- E is a $\sqcup$- or $\bigsqcup$-formula;
- E is $\forall y G$ or $\exists y G$, and G is critical;
- E is a $\vee$-disjunction, with all disjuncts critical;
- E is a $\wedge$-conjunction, with at least one conjunct critical.

The importance of the above concept is related to the fact that (**PA** knows that) a given legal run of $\overline{X}$ is lost by $\mathcal{X}$ if and only if the eventual formula/position hit by that run is critical.

Lemma 21.4 *Assume $E = E(\mathfrak{b},\vec{s})$ is a non-critical formula all of whose free variables are among $\mathfrak{b},\vec{s}$. Further assume $\theta,\omega,\vec{\psi}$ are any terms ($\vec{\psi}$ of the same length as $\vec{s}$), and z is a variable not occurring in these terms or in E. Then*

$$\mathbf{PTA} \vdash \exists z \mathbb{W}^E(z,\theta,\xi(\omega),\omega,\vec{\psi}) \rightarrow \|\overline{E(\omega,\vec{\psi})}\|.$$

Idea. The antecedent of the above formula implies that some run of X generated by $\mathscr{X}$ yields the non-critical eventual position $E(\omega,\vec{\psi})$. If $\|E(\omega,\vec{\psi})\|$ is true, then so is $\|\overline{E(\omega,\vec{\psi})}\|$. Otherwise, if $\|E(\omega,\vec{\psi})\|$ is false, $\mathscr{X}$ has lost, so $\mathbb{L}$ is true. But the truth of the formula $\mathbb{L}$, which is disjuncted with every politeral of $\overline{E(\omega,\vec{\psi})}$, easily implies the truth of $\|\overline{E(\omega,\vec{\psi})}\|$. This argument is formalizable in **PA**. ■

Proof. Assume the conditions of the lemma. Argue in **PA**. Consider arbitrary values of θ, ω, $\vec{\psi}$, which we continue writing as θ, ω, $\vec{\psi}$. Suppose, for a contradiction, that the ultimate position — that is, the position reached by the time $\xi(\omega)$ — of some play of $\mathscr{X}$ over X is $E(\omega,\vec{\psi})$ (i.e., $\exists z\mathbb{W}^E(z,\theta,\xi(\omega),\omega,\vec{\psi})$ is true) but $\|\overline{E(\omega,\vec{\psi})}\|$ is false. The falsity of $\|\overline{E(\omega,\vec{\psi})}\|$ implies the falsity of $\|E(\omega,\vec{\psi})\|$. This is so because the only difference between the two formulas is that, wherever the latter has some politeral L, the former has a disjunction containing L as a disjunct.

But ending with an ultimate position whose elementarization is false means that $\mathscr{X}$ does not win X in time ξ (remember Lemma 9.3). In other words,

$$\mathbb{L} \textit{ is true.} \tag{108}$$

Consider any non-critical formula G. By induction on the complexity of G, we are going to show that $\|\overline{G}\|$ is true for any values of its free variables. Indeed:

If G is a literal, then $\|\overline{G}\|$ is $G\vee\mathbb{L}$ which, by (108), is true.

If G is $H_1\sqcap\ldots\sqcap H_n$ or $\sqcap xH(x)$, then $\|\overline{G}\|$ is $\top$ and is thus true.

G cannot be $H_1\sqcup\ldots\sqcup H_n$ or $\sqcup xH(x)$, because then it would be critical.

If G is $\forall yH(y)$ or $\exists yH(y)$, then $\|\overline{G}\|$ is $\forall y\|\overline{H(y)}\|$ or $\exists y\|\overline{H(y)}\|$. In either case $\|\overline{G}\|$ is true because, by the induction hypothesis, $\|\overline{H(y)}\|$ is true for every value of its free variables, including variable y.

If G is $H_1\wedge\ldots\wedge H_n$, then the formulas $H_1,\ldots,H_n$ are non-critical. Hence, by the induction hypothesis, $\|\overline{H_1}\|,\ldots,\|\overline{H_n}\|$ are true. Hence so is $\|\overline{H_1}\|\wedge\ldots\wedge\|\overline{H_n}\|$ which, in turn, is nothing but $\|\overline{G}\|$.

Finally, if G is $H_1\vee\ldots\vee H_n$, then one of the formulas H_i is non-critical. Hence, by the induction hypothesis, $\|\overline{H_i}\|$ is true. Hence so is $\|\overline{H_1}\|\vee\ldots\vee\|\overline{H_n}\|$ which, in turn, is nothing but $\|\overline{G}\|$.

Thus, for any non-critical formula G, $\|\overline{G}\|$ is true. This includes the case $G=E(\omega,\vec{\psi})$ which, however, contradicts our earlier observation that $\|\overline{E(\omega,\vec{\psi})}\|$ is false. ■

Lemma 21.5 *Assume $E=E(\mathfrak{b},\vec{s})$ is a critical formula all of whose free variables are among $\mathfrak{b},\vec{s}$. Further assume $\theta,\omega,\vec{\psi}$ are any terms ($\vec{\psi}$ of the same length as $\vec{s}$), and z is a variable not occurring in these terms or in E. Then*

$$\textbf{PTA}\vdash\exists z\mathbb{W}^E(z,\theta,\xi(\omega),\omega,\vec{\psi})\rightarrow\overline{E(\omega,\vec{\psi})}.$$

Proof. Assume the conditions of the lemma. By induction on complexity, one can easily see that the elementarization of any critical formula is false. Thus, $\|\overline{E(\omega,\vec{\psi})}\|$ is false. Arguing further as we did in the proof of Lemma 21.4 when deriving (108), we find that, if $\exists z\mathbb{W}^E(z,\theta,\xi(\omega),\omega,\vec{\psi})$ is true, then so is $\mathbb{L}$. And this argument can be formalized in **PA**, so that we have

$$\textbf{PTA}\vdash\exists z\mathbb{W}^E(z,\theta,\xi(\omega),\omega,\vec{\psi})\rightarrow\mathbb{L}.$$

The above, together with Lemma 21.3, by Transitivity, implies **PTA** $\vdash$ $\exists z\mathbb{W}^E(z,\theta,\xi(\omega),\omega,\vec{\psi})\rightarrow\overline{E(\omega,\vec{\psi})}$. ■

21.4. Taking care of the case of small bounds

$|\xi(\mathfrak{b})|$ is logarithmic in $\mathfrak{b}$ and hence, generally, it will be much smaller than $\mathfrak{b}$. However, there are exceptions. For instance, when $\mathfrak{b}=1$ and $\xi(\mathfrak{b})=\mathfrak{b}+\mathfrak{b}$, the size of $\xi(\mathfrak{b})$ is 2, exceeding $\mathfrak{b}$. Such exceptions will only occur in a finite number of cases, where $\mathfrak{b}$ is "very small". These pathological cases — the cases with $\neg|\xi(\mathfrak{b})|\leq\mathfrak{b}$ — require a separate handling, which we present in this subsection. The main result here is Lemma 21.11, according to which **PTA** proves $\neg|\xi(\mathfrak{b})|\leq\mathfrak{b}\rightarrow\overline{X}$, i.e. proves the target $\overline{X}$ on the assumption that we are dealing with a pathologically small $\mathfrak{b}$. The remaining, "normal" case of $|\xi(\mathfrak{b})|\leq\mathfrak{b}$ will be taken care of later in Subsection 21.6.

For a natural number n, by the **formal numeral for** n, denoted $\hat{n}$, we will mean some standard variable-free term representing n. For clarity, let us say that the formal numeral for zero is **0**0, the formal numeral for one is **0**1, the formal numeral for two is **0**10, the formal numeral for three is **0**11, the formal numeral for four is **0**100, etc.

The above-mentioned provability of $\neg|\xi(\mathfrak{b})|\leq\mathfrak{b}\rightarrow\overline{X}$ will be established through showing (Lemma 21.10) that, for each particular positive integer b, including all of the finitely many b's with $\neg|\xi(\hat{b})|\leq\hat{b}$, **PTA** proves $\mathfrak{b}=\hat{b}\rightarrow\overline{X}$. But we need a little preparation first.

Lemma 21.6 *Let r be any variable, b any positive integer, and N the set of all natural numbers a with $|a|\leq b$. Then*

$$\mathbf{PTA}\vdash\mathfrak{b}=\hat{b}\rightarrow\sqcup\{r=\hat{a}\mid a\in N\}.$$

Idea. On the assumption $\mathfrak{b}=\hat{b}$ and due to Axiom 13, **PTA** knows that, whatever r is, its size cannot exceed $\hat{b}$. In other words, it knows that r has to be one of the elements of N. The main technical part of our proof of the lemma is devoted to showing that this knowledge is, in fact, constructive, in the sense that **PTA** can tell exactly which ($\sqcup$) element of N the number r is. ■

Proof. Assume the conditions of the lemma. Obviously we have

$$\mathbf{PA}\vdash|r|\leq\mathfrak{b}\rightarrow\mathfrak{b}=\hat{b}\rightarrow\vee\{r=\hat{a}\mid a\in N\},$$

modus-ponensing which with Axiom 13 yields

$$\mathbf{PTA}\vdash\mathfrak{b}=\hat{b}\rightarrow\vee\{r=\hat{a}\mid a\in N\}.\qquad(109)$$

Next, consider any $a\in N$. We claim that

$$\mathbf{PTA}\vdash\mathfrak{b}=\hat{b}\rightarrow r=\hat{a}\sqcup r\neq\hat{a},\qquad(110)$$

which is justified as follows:

1. $\neg|\hat{a}|\leq\mathfrak{b}\sqcup\bigsqcup z(z=\hat{a})$ Lemma 19.8
2. $\neg|\hat{a}|\leq\mathfrak{b}\rightarrow\mathfrak{b}\neq\hat{b}$ **PA**
3. $\neg|\hat{a}|\leq\mathfrak{b}\rightarrow\big(\mathfrak{b}=\hat{b}\rightarrow\bigsqcup z(z=\hat{a})\big)$ Weakening: 2
4. $\bigsqcup z(z=\hat{a})\rightarrow\big(\mathfrak{b}=\hat{b}\rightarrow\bigsqcup z(z=\hat{a})\big)$ **CL4**-Instantiation, instance of $P\rightarrow(q\rightarrow P)$
5. $\mathfrak{b}=\hat{b}\rightarrow\bigsqcup z(z=\hat{a})$ $\sqcup$-Elimination: 1,3,4
6. $\bigsqcap x\bigsqcap y(y=x\sqcup y\neq x)$ Lemma 18.3
7. $s=r\sqcup s\neq r$ $\bigsqcap$-Elimination (twice): 6
8. $s=r\rightarrow s=\hat{a}\rightarrow r=\hat{a}$ Logical axiom
9. $s=r\rightarrow s=\hat{a}\rightarrow r=\hat{a}\sqcup r\neq\hat{a}$ $\sqcup$-Choose: 8
10. $s\neq r\rightarrow s=\hat{a}\rightarrow r\neq\hat{a}$ Logical axiom
11. $s\neq r\rightarrow s=\hat{a}\rightarrow r=\hat{a}\sqcup r\neq\hat{a}$ $\sqcup$-Choose: 10
12. $s=\hat{a}\rightarrow r=\hat{a}\sqcup r\neq\hat{a}$ $\sqcup$-Elimination: 7,9,11
13. $\bigsqcup z(z=\hat{a})\rightarrow r=\hat{a}\sqcup r\neq\hat{a}$ $\bigsqcap$-Introduction: 12
14. $\mathfrak{b}=\hat{b}\rightarrow r=\hat{a}\sqcup r\neq\hat{a}$ TR: 5,13

Now, with a little thought, the formula

$$(\mathfrak{b}=\hat{b}\rightarrow\vee\{r=\hat{a}\mid a\in N\})\wedge\wedge\{\mathfrak{b}=\hat{b}\rightarrow r=\hat{a}\sqcup r\neq\hat{a}\mid a\in N\}\rightarrow$$
$$(\mathfrak{b}=\hat{b}\rightarrow\sqcup\{r=\hat{a}\mid a\in N\})$$

can be seen to be provable in **CL3** and hence in **PTA**. Modus-ponensing the above with (109) and (110) yields the desired $\mathbf{PTA}\vdash\mathfrak{b}=\hat{b}\rightarrow\sqcup\{r=\hat{a}\mid a\in N\}$.
■

Lemma 21.7 *Let r be any variable, b any positive integer, and $E(r)$ any formula. Assume that, for each natural number a with $|a|\leq b$, **PTA** $\vdash E(\hat{a})$. Then **PTA** $\vdash\mathfrak{b}=\hat{b}\rightarrow E(r)$.*

Proof. Assume the conditions of the lemma. Let N be the set of all numbers a with $|a|\leq b$. Consider any $a\in N$. Clearly, by **CL4**-Instantiation, **PTA** $\vdash E(\hat{a})\rightarrow r=\hat{a}\rightarrow E(r)$. Modus-ponensing this with the assumption **PTA** $\vdash E(\hat{a})$ yields **PTA** $\vdash r=\hat{a}\rightarrow E(r)$. This holds for all $a\in N$, so, by $\sqcap$-Introduction, **PTA** $\vdash\sqcup\{r=\hat{a}\mid a\in N\}\rightarrow E(r)$. But, by Lemma 21.6, **PTA** $\vdash\mathfrak{b}=\hat{b}\rightarrow\sqcup\{r=\hat{a}\mid a\in N\}$. Hence, by Transitivity, **PTA** $\vdash\mathfrak{b}=\hat{b}\rightarrow E(r)$. ■

Below and elsewhere, for a tuple $\vec{c}=c_1,\ldots,c_n$ of constants, $\vec{\hat{c}}$ stands for the tuple $\hat{c}_1,\ldots,\hat{c}_n$.

Lemma 21.8 *Assume $E=E(\mathfrak{b},\vec{s})$ is a formula all of whose free variables are among $\mathfrak{b},\vec{s}$, b is any positive integer, and $a,d_1,d_2,\vec{c}$ are any natural* *numbers ($\vec{c}$ of the same length as $\vec{s}$). Then $\mathbb{W}^E(\hat{a},\hat{d}_1,\hat{d}_2,\hat{b},\vec{c})$ is true iff it is provable in* **PA***.*

Proof. **PA** only proves true sentences. **PA** is also known to prove all "mechanically verifiable" (of complexity $\Sigma^0{}_1$, to be precise) true sentences, such as $\mathbb{W}^E(\hat{a},\hat{d}_1,\hat{d}_2,\hat{b},\vec{c})$ is, if true. ■

Lemma 21.9 *Under the conditions of Lemma 21.8, if $\mathbb{W}^E(\hat{a},\hat{d}_1,\hat{d}_2,\hat{b},\vec{c})$ is true, then* **PTA** $\vdash \mathfrak{b}=\hat{b}\to\overline{E(\hat{b},\vec{c})}$.

Idea. In the context of the (a,e_b)-branch, the assumptions of the lemma imply that, at some (d_1) point, the play hits the position $E(b,\vec{c})$. $\mathscr{X}$ may or may not make further moves to modify this position.

If a move is made, it brings us to a new position expressed through a simpler formula, from which $E(b,\vec{c})$ follows by $\sqcap$-Choose or $⨅$-Choose. This allows us to apply the induction hypothesis to that formula, and then find the provability of $\mathfrak{b}=\hat{b}\to\overline{E(\hat{b},\vec{c})}$ by the corresponding Choose rule.

Suppose now no moves are made, so that the play ends as $E(b,\vec{c})$. This position has to be non-critical, or otherwise $\mathscr{X}$ would be the loser. Then Lemmas 21.4 and 21.8 allow us to find that the elementarization of the target formula is provable. Appropriately manipulating the induction hypothesis, we manage to find the provability of all additional premises from which the target formula follows by Wait. ■

Proof. Our proof proceeds by induction on the complexity of $E(\mathfrak{b},\vec{s})$. Assume $\mathbb{W}^E(\hat{a},\hat{d}_1,\hat{d}_2,\hat{b},\vec{c})$ is true. We separately consider the following two cases.

Case 1: $\mathbb{W}^E(\hat{a},\hat{d}_1,\xi(\hat{b}),\hat{b},\vec{c})$ is not true. On the other hand, by our assumption, $\mathbb{W}^E(\hat{a},\hat{d}_1,\hat{d}_2,\hat{b},\vec{c})$ is true. The latter implies that, in the (a,e_b)-branch, the play reaches (by time d_1) the position $E(\hat{b},\vec{c})$ which persists up to time d_2; and the former implies that this situation changes sometime afterwards (at the latest by time $\xi(b)$). So, a move is made at some time m with $d_2\le m<\xi(b)$. Such a move β (the earliest one if there are several) cannot be made by the environment, because, as implied by the assumption $\mathbb{W}^E(\hat{a},\hat{d}_1,\hat{d}_2,\hat{b},\vec{c})$, a is a (b,d_1)-adequate

counterbehavior. So, β is a move by $\mathscr{X}$. Since $\mathscr{X}$ wins X, β cannot be an illegal move of the play. It is obvious that then one of the following conditions holds:

(i) There is a formula $H=H(\mathfrak{b},\vec{s})$ which is the result of replacing in $E(\mathfrak{b},\vec{s})$ a surface occurrence of a subformula $G_1\sqcup\ldots\sqcup G_n$ by one of the G_i's, such that $\mathbb{W}^H(\hat{a},\hat{m}',\hat{m}',\hat{b},\vec{c})$ is true.

(ii) There is formula $H=H(\mathfrak{b},\vec{s},r)$, where r is a variable not occurring in $E(\mathfrak{b},\vec{s})$, such that $H(\mathfrak{b},\vec{s},r)$ is the result of replacing in $E(\mathfrak{b},\vec{s})$ a surface occurrence of a subformula $⨆yG(y)$ by $G(r)$, and $\mathbb{W}^H(\hat{a},\hat{m}',\hat{m}',\hat{b},\vec{c},\hat{k})$ is true for some constant k with $|k|\le b$.
.

Thus, $H(b,\vec{c})$ (in case (i)) or $H(b,\vec{c},k)$ (in case (ii)) is the game/position to which $E(b,\vec{c})$ is brought down by the above-mentioned legal labmove $\top\beta$.

Assume condition (i) holds. By the induction hypothesis, **PTA** $\vdash \mathfrak{b}=\hat{b}\to\overline{H(\hat{b},\vec{c})}$. Then, by $\sqcup$-Choose, **PTA** $\vdash \mathfrak{b}=\hat{b}\to\overline{E(\hat{b},\vec{c})}$.

Assume now condition (ii) holds. Again, by the induction hypothesis,

$$\textbf{PTA}\vdash \mathfrak{b}=\hat{b}\to\overline{H(\hat{b},\vec{c},\hat{k})}. \tag{111}$$

Obviously **CL4** $\vdash \big(p\to Q(f)\big)\to\big(r=f\to p\to Q(r)\big)$ whence, by **CL4**-Instantiation,

$$\textbf{PTA}\vdash \big(\mathfrak{b}=\hat{b}\to\overline{H(\hat{b},\vec{c},\hat{k})}\big)\to\big(r=\hat{k}\to\mathfrak{b}=\hat{b}\to\overline{H(\hat{b},\vec{c},r)}\big).$$

Modus-ponensing the above with (111) yields

$$\textbf{PTA}\vdash r=\hat{k}\to\big(\mathfrak{b}=\hat{b}\to\overline{H(\hat{b},\vec{c},r)}\big)$$

from which, by $⨆$-Choose,

$$\textbf{PTA}\vdash r=\hat{k}\to\big(\mathfrak{b}=\hat{b}\to\overline{E(\hat{b},\vec{c})}\big)$$

and then, by $⨅$-Introduction,

$$\textbf{PTA}\vdash ⨆z(z=\hat{k})\to\big(\mathfrak{b}=\hat{b}\to\overline{E(\hat{b},\vec{c})}\big). \tag{112}$$

We also have

$$\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \sqcup z(z = \hat{k}), \tag{113}$$

justified as follows:

1. $\neg|\hat{k}| \le \mathfrak{b} \sqcup \sqcup z(z = \hat{k})$ Lemma 19.8
2. $\neg|\hat{k}| \le \mathfrak{b} \to \mathfrak{b} \neq \hat{b}$ **PA**
3. $\neg|\hat{k}| \le \mathfrak{b} \to \mathfrak{b} = \hat{b} \to \sqcup z(z = \hat{k})$ Weakening: 2
4. $\sqcup z(z = \hat{k}) \to \mathfrak{b} = \hat{b} \to \sqcup z(z = \hat{k})$ **CL4**-Instantiation, instance of $P \to q \to P$
6. $\mathfrak{b} = \hat{b} \to \sqcup z(z = \hat{k})$ $\sqcup$-Elimination: 1,3,4

From (113) and (112), by Transitivity, $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \mathfrak{b} = \hat{b} \to \overline{E(\hat{b}, \hat{\vec{c}})}$. But, by **CL4**-Instantiation, we have

$$\mathbf{PTA} \vdash \big(\mathfrak{b} = \hat{b} \to \mathfrak{b} = \hat{b} \to \overline{E(\hat{b}, \hat{\vec{c}})}\big) \to \big(\mathfrak{b} = \hat{b} \to \overline{E(\hat{b}, \hat{\vec{c}})}\big)$$

(this matches $(p \to p \to Q) \to (p \to Q)$). Hence, by Modus Ponens, we find $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \overline{E(\hat{b}, \hat{\vec{c}})}$, as desired.

Case 2: $\mathbb{W}^E(\hat{a}, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}})$ is true. Then, by Lemma 21.8, **PTA** proves $\mathbb{W}^E(\hat{a}, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}})$. **PTA** also proves the following formula because it is a logical axiom:

$$\mathbb{W}^E(\hat{a}, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}}) \to \exists z \mathbb{W}^E(z, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}}).$$

Hence, by Modus Ponens,

$$\mathbf{PTA} \vdash \exists z \mathbb{W}^E(z, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}}). \tag{114}$$

$\mathbb{W}^E(\hat{a}, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}})$ implies that $E(b, \vec{c})$ is the final position of the play over X according to the scenario of the (a, e_b)-branch. Note that, therefore, $E(\mathfrak{b}, \vec{s})$ cannot be critical. This is so because, as observed earlier, the elementarization of any critical formula is false, and having such a formula as the final position in some play would make $\mathscr{X}$ lose, contrary to our assumption that $\mathscr{X}$ (always) wins X. Therefore, by Lemma 21.4,

$$\mathbf{PTA} \vdash \exists z \mathbb{W}^E(z, \hat{d}_1, \xi(\hat{b}), \hat{b}, \hat{\vec{c}}) \to \|\overline{E(\hat{b}, \hat{\vec{c}})}\|.$$

Modus-ponensing the above with (114) yields $\mathbf{PTA} \vdash \|\overline{E(\hat{b}, \hat{\vec{c}})}\|$, from which, by Weakening, $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \|\overline{E(\hat{b}, \hat{\vec{c}})}\|$, which is the same as to say that

$$\mathbf{PTA} \vdash \|\mathfrak{b} = \hat{b} \to \overline{E(\hat{b}, \hat{\vec{c}})}\|. \tag{115}$$

Claim 1. *Assume $\overline{E(\hat{b}, \hat{\vec{c}})}$ has the form $H[G_1 \sqcap \ldots \sqcap G_m]$, and $i \in \{1, \ldots, m\}$. Then* $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to H[G_i]$.

Proof. Assume the conditions of the claim. Let $F = F(\mathfrak{b}, \vec{s})$ be the formula such that $\overline{F(\hat{b}, \hat{\vec{c}})} = H[G_i]$. Let β be the environment's move that brings $H[G_1 \sqcap \ldots \sqcap G_m]$ to $H[G_i]$. Let k be the (code of the) counterbehavior obtained by appending the timestamped move $(\beta, d_1 - 1)$ to (the counterbehavior whose code is) a. Since $\mathbb{W}^E(\hat{a}, \hat{d}_1, \hat{d}_2, \hat{b}, \hat{\vec{c}})$ is true, obviously $\mathbb{W}^F(\hat{k}, \hat{d}_1, \hat{d}_1, \hat{b}, \hat{\vec{c}})$ also has to be true. Then, by the induction hypothesis, $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \overline{F(\hat{b}, \hat{\vec{c}})}$, i.e. $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to H[G_i]$. ■

Claim 2. *Assume $\overline{E(\hat{b}, \hat{\vec{c}})}$ has the form $H[\sqcap y G(y)]$, and r is an arbitrary non-$\mathfrak{b}$ variable not occurring in $E(\mathfrak{b}, \vec{x})$. Then* $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to H[G(r)]$.

Proof. Assume the conditions of the claim. Let $F = F(\mathfrak{b}, \vec{s}, r)$ be the formula such that $\overline{F(\hat{b}, \hat{\vec{c}}, r)} = H[G(r)]$. For each constant m whose size does not exceed b, let β_m be the environment's move that brings $H[\sqcap y G(y)]$ to $H[G(m)]$, and let k_m be the (code of the) counterbehavior obtained by appending the timestamped move $(\beta_m, d_1 - 1)$ to (the counterbehavior whose code is) a. Since $\mathbb{W}^E(\hat{a}, \hat{d}_1, \hat{d}_2, \hat{b}, \hat{\vec{c}})$ is true, obviously, for each constant m with $|m| \le b$, $\mathbb{W}^F(\hat{k}_m, \hat{d}_1, \hat{d}_1, \hat{b}, \hat{\vec{c}}, \hat{m})$ is also true. Then, by the induction hypothesis, $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \overline{F(\hat{b}, \hat{\vec{c}}, \hat{m})}$, i.e. $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to H[G(\hat{m})]$. But then, by Lemma 21.7, $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \mathfrak{b} = \hat{b} \to \overline{F(\hat{b}, \hat{\vec{c}}, r)}$, i.e. $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \mathfrak{b} = \hat{b} \to H[G(r)]$. By **CL4**-Instantiation, we also have

$$\mathbf{PTA} \vdash \big(\mathfrak{b} = \hat{b} \to \mathfrak{b} = \hat{b} \to H[G(r)]\big) \to \big(\mathfrak{b} = \hat{b} \to H[G(r)]\big)$$

(this is an instance of $(p \to p \to Q) \to (p \to Q)$). So, by Modus Ponens, $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to H[G(r)]$. ■

From (115), Claim 1 and Claim 2, by Wait, we find the desired $\mathbf{PTA} \vdash \mathfrak{b} = \hat{b} \to \overline{E(\hat{b}, \hat{\vec{c}})}$. ■

Lemma 21.10 *For any positive integer b,* **PTA** $\vdash \mathfrak{b}=\hat{b} \to \overline{X(\mathfrak{b})}$.

Proof. Consider any positive integer b. Let a be (the code of) the empty counterbehavior. Of course, $\mathbb{W}^X(\hat{a}, \hat{1}, \hat{1}, \hat{b})$ is true. Then, by Lemma 21.9, **PTA** $\vdash \mathfrak{b}=\hat{b} \to \overline{X(\hat{b})}$. But the formula

$$\big(\mathfrak{b}=\hat{b} \to \overline{X(\hat{b})}\big) \to \big(\mathfrak{b}=\hat{b} \to \overline{X(\mathfrak{b})}\big)$$

is an instance of the **CL4**-provable $\big(\mathfrak{b}=f \to P(f)\big) \to \big(\mathfrak{b}=f \to P(\mathfrak{b})\big)$ and, by **CL4**-Instantiation, is provable in **PTA**. Hence, by Modus Ponens, **PTA** $\vdash \mathfrak{b}=\hat{b} \to \overline{X(\mathfrak{b})}$. ■

Lemma 21.11 **PTA** $\vdash \neg|\xi(\mathfrak{b})| \leq \mathfrak{b} \to \overline{X(\mathfrak{b})}$.

Idea. **PTA** knows that, if $\neg|\xi(\mathfrak{b})| \leq \mathfrak{b}$, then $\mathfrak{b}=\hat{b}$ for one of finitely many particular ("very small") positive integers b. Furthermore, as in Lemma 21.6, we can show that such knowledge is constructive, in the sense that **PTA** can tell ($\sqcup$) exactly for which b do we have $\mathfrak{b}=\hat{b}$. Then the desired conclusion easily follows from Lemma 21.10. ■

Proof. The size of $\xi(\mathfrak{b})$ can be greater than $\mathfrak{b}$ for only a certain finite number of "small" non-0 values of $\mathfrak{b}$. Let N be the set of all such values. Obviously

$$\textbf{PA} \vdash \mathfrak{b} \neq \mathbf{0} \to \neg|\xi(\mathfrak{b})| \leq \mathfrak{b} \to \vee\{\mathfrak{b}=\hat{a} \mid a \in N\},$$

modus-ponensing which with Lemma 13.2 yields

$$\textbf{PTA} \vdash \neg|\xi(\mathfrak{b})| \leq \mathfrak{b} \to \vee\{\mathfrak{b}=\hat{a} \mid a \in N\}. \tag{116}$$

By Lemma 21.10, for each $a \in N$ we have **PTA** $\vdash \mathfrak{b}=\hat{a} \to \overline{X(\mathfrak{b})}$. Hence, by $\sqcap$-Introduction,

$$\textbf{PTA} \vdash \sqcup\{\mathfrak{b}=\hat{a} \mid a \in N\} \to \overline{X(\mathfrak{b})}. \tag{117}$$

Next we claim that

$$\textit{for each } a \in N,\ \textbf{PA} \vdash \mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}. \tag{118}$$

Below is a justification of this claim for an arbitrary $a \in N$:

1. $\neg|\hat{a}| \leq \mathfrak{b} \sqcup \bigsqcup z(z=\hat{a})$ Lemma 19.8
2. $\neg|\hat{a}| \leq \mathfrak{b} \to \mathfrak{b} \neq \hat{a}$ **PA**
3. $\neg|\hat{a}| \leq \mathfrak{b} \to \mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}$ $\sqcup$-Choose: 2
4. $\bigsqcap x \bigsqcap y(y=x \sqcup y \neq x)$ Lemma 18.3
5. $s=\mathfrak{b} \sqcup s \neq \mathfrak{b}$ $\bigsqcap$-Elimination (twice): 4
6. $s=\mathfrak{b} \to s=\hat{a} \to \mathfrak{b}=\hat{a}$ Logical axiom
7. $s=\mathfrak{b} \to s=\hat{a} \to \mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}$ $\sqcup$-Choose: 6
8. $s \neq \mathfrak{b} \to s=\hat{a} \to \mathfrak{b} \neq \hat{a}$ Logical axiom
9. $s \neq \mathfrak{b} \to s=\hat{a} \to \mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}$ $\sqcup$-Choose: 8
10. $s=\hat{a} \to \mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}$ $\sqcup$-Elimination: 5,7,9
11. $\bigsqcup z(z=\hat{a}) \to \mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}$ $\bigsqcap$-Introduction: 10
12. $\mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a}$ $\sqcup$-Elimination: 1,3,11

The following formula can be easily seen to be provable in **CL3** and hence in **PTA**:

$$\textbf{PTA} \vdash \wedge\{\mathfrak{b}=\hat{a} \sqcup \mathfrak{b} \neq \hat{a} \mid a \in N\} \to \vee\{\mathfrak{b}=\hat{a} \mid a \in N\} \to \sqcup\{\mathfrak{b}=\hat{a} \mid a \in N\}.$$

Modus-ponensing the above with (118) yields

$$\textbf{PTA} \vdash \vee\{\mathfrak{b}=\hat{a} \mid a \in N\} \to \sqcup\{\mathfrak{b}=\hat{a} \mid a \in N\}. \tag{119}$$

Now, from (116), (119) and (117), by Transitivity applied twice, we get **PTA** $\vdash \neg|\xi(\mathfrak{b})| \leq \mathfrak{b} \to \overline{X(\mathfrak{b})}$ as desired. ■

21.5. Ptarithmetizing HPM-computations

In this subsection we prove the earlier-mentioned "provable traceability" of the work of $\mathcal{X}$, in a certain technically strong form necessary for our further treatment. As we remember, roughly it means the constructive knowledge by **PTA** of the configurations of $\mathcal{X}$ in its interaction with a given adversary (the latter thought of as a counterbehavior). The present elaboration is the first relatively advanced example of "*ptarithmetization*" or, more generally, "*clarithmetization*" — extending Gödel's *arithmetization* technique from the classical context to the context of computability logic.

Let *STATES* be the set of all states of the machine $\mathscr{X}$, and *SYMBOLS* be the set of all symbols that may appear on any of its tapes. As we know, both sets are finite. We assume that the cells of each of the three tapes are numbered consecutively starting from 0 (rather than 1).

Below we introduce elementary formulas that naturally arithmetize the corresponding metapredicates.

- $Adequate(z,w,t)$ means "z is a (w,t)-adequate counterbehavior".
- For each $a \in STATES$, $State_a(z,w,t)$ means "In the (z,e_w)-branch, at time t, $\mathscr{X}$ is in state a".
- For each $a \in SYMBOLS$, $VSymbol_a(z,w,t,u)$ means "In the (z,e_w)-branch, at time t, cell $\#u$ of the valuation tape contains symbol a". Similarly for $WSymbol_a(z,w,t,u)$ (for the work tape) and $RSymbol_a(z,w,t,u)$ (for the run tape).
- $VHead(z,w,t,u)$ means "In the (z,e_w)-branch, at time t, the head of the valuation tape is over cell $\#u$". Similarly for $WHead(z,w,t,u)$ (for the work tape) and $RHead(z,w,t,u)$ (for the run tape).
- $Runsize(z,w,t,u)$ means "In the (z,e_w)-branch, at time t, the leftmost blank cell of the run tape is cell $\#u$".
- $\mathbb{E}(z,t)$ abbreviates

$$\begin{array}{l}
Adequate(z,\mathfrak{b},t)\ \wedge\\
\sqcup\{State_a(z,\mathfrak{b},t) \mid a\in STATES\}\ \wedge\\
\Big(\exists x\big(Runsize(z,\mathfrak{b},t,x)\wedge |x|\le\mathfrak{b}\big)\sqsupset \bigsqcup x Runsize(z,\mathfrak{b},t,x)\Big)\ \wedge\\
\bigsqcup x\big(VHead(z,\mathfrak{b},t,x)\wedge \sqcup\{VSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big)\ \wedge\\
\bigsqcup x\big(WHead(z,\mathfrak{b},t,x)\wedge \sqcup\{WSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big)\ \wedge\\
\bigsqcup x\big(RHead(z,\mathfrak{b},t,x)\wedge \sqcup\{RSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big).
\end{array}$$

- $\mathbb{F}(z,t)$ abbreviates

$$\begin{array}{l}
\bigsqcap x\big(\sqcup\{VSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big)\ \wedge\\
\bigsqcap x\big(\sqcup\{WSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big)\ \wedge\\
\Big(\bigsqcap x\big(\sqcup\{WSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big)\\
\sqcap \bigsqcap x\big(\sqcup\{RSymbol_a(z,\mathfrak{b},t,x)\mid a\in SYMBOLS\}\big)\Big).
\end{array}$$

Note that both formulas $\mathbb{E}(z,t)$ and $\mathbb{F}(z,t)$, in addition to z and t, contain $\mathfrak{b}$ as a free variable, which we however do not explicitly indicate as it will never be replaced by any other term.

We use $\exists!$ as a standard abbreviation, defined by

$$\exists! z T(z) \;=\; \exists z\Big(T(z)\wedge \forall y\big(T(y)\to y=z\big)\Big).$$

Let z be any variable and T — let us (also) write it in the form $T(z)$ — any elementary formula. We say that T is **functional for** z iff **PTA** $\vdash \exists! z T(z)$.

For variables z,t and an elementary formula $T = T(z)$ functional for z, we will be using $\mathbb{E}(z^T,t)$ as an abbreviation defined by

$$\mathbb{E}(z^T,t) \;=\; \forall z\big(T(z)\to \mathbb{E}(z,t)\big).$$

Similarly for $\mathbb{F}(z^T,t)$. It is our convention that, whenever using these abbreviations, the variables z and t are not the same, so that t does not get bound by the external $\forall z$. Similarly, if we write $\mathbb{E}(z^T,\theta)$ or $\mathbb{F}(z^T,\theta)$ where θ is a term, it will be assumed that θ does not contain z.

Lemma 21.12 *For any elementary formula T functional for z,* **PTA** *proves*

$$\mathbb{E}(z^T,t)\to \mathbb{E}(z^T,t)\wedge \mathbb{E}(z^T,t).$$

Idea. As explained in Subsection 21.1, $\mathbb{E}$ — whether in the form $\mathbb{E}(z,t)$ or $\mathbb{E}(z^T,t)$ — is essentially a "recyclable" resource because it does not contain $\sqcap$,$\bigsqcap$. ■

Proof. Bottom-up, a proof of the target formula goes like this. Keep applying $\bigsqcap$-Introduction and $\sqcap$-Introduction until the antecedent (in the given branch of the proof tree) becomes

$$\begin{array}{l}
\forall z\Big(T\to\\
Adequate(z,\mathfrak{b},t)\ \wedge\\
State_a(z,\mathfrak{b},t)\ \wedge\\
Runsize(z,\mathfrak{b},t,u)\ \wedge\\
\big(VHead(z,\mathfrak{b},t,v)\wedge VSymbol_b(z,\mathfrak{b},t,v)\big)\ \wedge\\
\big(WHead(z,\mathfrak{b},t,w)\wedge WSymbol_c(z,\mathfrak{b},t,w)\big)\ \wedge\\
\big(RHead(z,\mathfrak{b},t,r)\wedge VSymbol_d(z,\mathfrak{b},t,r)\big)\Big)
\end{array}$$

— or, maybe, the same but with "$\neg\exists x\big(Runsize(z,\mathfrak{b},t,x)\wedge|x|\leq\mathfrak{b}\big)$" instead of "$Runsize(z,\mathfrak{b},t,u)$" — for some variables u,v,w,r, state a and symbols b,c,d. Then apply a series of $\sqcup$-Chooses and $\bigsqcup$-Chooses and bring the consequent to a conjunction of two copies of the antecedent. Now we are dealing with a classically valid and hence provable elementary formula of the form $F\rightarrow F\wedge F$. ■

Lemma 21.13 *For any elementary formula T functional for z,* **PTA** *proves*

$$\mathbb{E}(z^T,t)\wedge\mathbb{F}(z^T,t)\rightarrow\mathbb{F}(z^T,t'). \tag{120}$$

Idea. For reasons in the spirit of an explanation given in Subsection 21.1, a single copy of the resource $\mathbb{E}(z^T,t)$ and a single copy of the resource $\mathbb{F}(z^T,t)$ turn out to be sufficient so solve the problem $\mathbb{F}(z^T,t')$. ■

Proof. The following formula is provable in **CL4** by Match applied three times:

$$\forall z\big(P_1(z)\wedge P_2(z)\rightarrow P_3(z)\big)\ \rightarrow\ \Big(\forall z\big(q(z)\rightarrow P_1(z)\big)\wedge\forall z\big(q(z)\rightarrow P_2(z)\big)\rightarrow\forall z\big(q(z)\rightarrow P_3(z)\big)\Big). \tag{121}$$

Consider the formula

$$\forall z\Big(\mathbb{E}(z,t)\wedge\mathbb{F}(z,t)\rightarrow\mathbb{F}(z,t')\Big). \tag{122}$$

The formula (122) → (120), which — after disabbreviating z^T in (120) — is

$$\forall z\Big(\mathbb{E}(z,t)\wedge\mathbb{F}(z,t)\rightarrow\mathbb{F}(z,t')\Big)\rightarrow \Big(\forall z\big(T(z)\rightarrow\mathbb{E}(z,t)\big)\wedge\forall z\big(T(z)\rightarrow\mathbb{F}(z,t)\big)\rightarrow\forall z\big(T(z)\rightarrow\mathbb{F}(z,t')\big)\Big),$$

can be seen to be an instance of (121) and hence, by **CL4**-Instantiation, provable in **PTA**. Therefore, if **PTA** proves (122), then, by Modus Ponens, it also proves the target (120). Based on this observation, we now forget about (120) and, in what follows, exclusively devote our efforts to showing that **PTA** ⊢ (122).

This is one of those cases where giving a full formal proof in the style practiced earlier is not feasible. But by now we have acquired enough experience in working with **PTA** to see that the informal argument provided below can be translated into a strict **PTA**-proof if necessary.

Argue in **PTA**. Consider an arbitrary (∀) counterstrategy z. The context of our discourse will be the play of $\mathcal{X}$ against z on the standard valuation $e_{\mathfrak{b}}$ — the $(z,e_{\mathfrak{b}})$-branch, that is. Assume that a single copy of the antecedental resource $\mathbb{E}(z,t)\wedge\mathbb{F}(z,t)$ is at our disposal. We need to show how to resolve the consequental problem $\mathbb{F}(z,t')$.

For resolving the first conjunct of $\mathbb{F}(z,t')$, we need to tell, for an arbitrary (⨅) given x, the content of cell $\#x$ of the valuation tape at time t'. This is very easy: the content of the valuation tape never changes. So, the symbol in cell $\#x$ at time t' will be the same as at time t, and what symbol it is we can learn from the first conjunct of (the antecedental resource) $\mathbb{F}(z,t)$. In more detailed terms, a solution/deduction strategy corresponding to the above outline is to wait (bottom-up ⨅-introduction) till the environment specifies a value (syntactically, a "fresh" variable) s for x in the first $\wedge$-conjunct of $\mathbb{F}(z,t')$; then, using the same s (bottom-up $\bigsqcup$-Choose), specify the value of x in the first $\wedge$-conjunct of $\mathbb{F}(z,t)$; after that, wait (bottom-up $\sqcap$-introduction) till the environment selects one of the $\sqcup$-disjuncts in the first $\wedge$-conjunct of $\mathbb{F}(z,t)$ (or rather of what that formula has become), and then select (bottom-up $\sqcup$-Choose) the same $\sqcup$-disjunct in the first $\wedge$-conjunct of $\mathbb{F}(z,t')$. Henceforth we will no longer provide such details, and will limit ourselves to just describing strategies, translatable (as we just saw) into bottom-up **PTA**-deductions.

For resolving the second conjunct of $\mathbb{F}(z,t')$, we need to tell, for an arbitrary (⨅) given x, the content of cell $\#x$ of the work tape at time t'. This is not hard, either. At first, using the fifth conjunct of $\mathbb{E}(z,t)$, we determine the location m of the work-tape head and the tape symbol c_W at that location at time t. If $m\neq x$ (Lemma 18.3 can be used to tell whether this is the case or not), then the symbol in cell $\#x$ at time t' will remain the same c_W. Suppose now $m=x$. Then we further use the second, fourth and sixth conjuncts of $\mathbb{E}(z,t)$ to learn about the state a of the machine at time t and the symbols c_V and c_R scanned at that time by the heads of the valuation and the run tapes.

Now, knowing c_V, c_W, c_R and a, based on the transition function of $\mathcal{X}$, we can tell what symbol will be written in cell $\#x$ of the work tape by time t'.

The left $\sqcap$-conjunct of the third $\wedge$-conjunct of $\mathbb{F}(z,t')$ is identical to the second $\wedge$-conjunct of $\mathbb{F}(z,t')$, and it can be resolved as we just saw above. However, to avoid an (unacceptable/unavailable) repeated usage of resources, we will employ the first $\sqcap$-conjunct of the third $\wedge$-conjunct of $\mathbb{F}(z,t)$ instead of the second $\wedge$-conjunct of $\mathbb{F}(z,t)$ as was done in the previous case. Of course, we will also need to use some parts of the resource $\mathbb{E}(z,t)$ which were already used by the procedure of the previous case. This, however, does not create any resource conflicts. Because any information extracted from $\mathbb{E}(z,t)$ earlier is still there, so the relevant parts of $\mathbb{E}(z,t)$ do not really need to be "queried" again, as we already know answers. That (re)using $\mathbb{E}(z,t)$ does not create any competition for resources should be remembered through the remaining part of this proof and the proof of the following lemma as well. This phenomenon of the "recycleability" of $\mathbb{E}(z,t)$ was, in fact, already established in Lemma 21.12.

Finally, for resolving the right $\sqcap$-conjunct of the third $\wedge$- conjunct of $\mathbb{F}(z,t')$, we need to tell, for an arbitrary ($\sqcap$) given x, the content of cell $\#x$ of the run tape at time t'. This is how it can be done. Let us call j the location of the leftmost blank cell of the run tape at time t. At first, we wait till the environment selects one of the $\sqcup$-disjuncts of the third $\wedge$-conjunct of $\mathbb{E}(z,t)$. If the left disjunct is selected, then $\mathfrak{b}<|j|$ (or else the selected disjunct is false and we win). Then we also have ($|x|<|j|$ and hence) $x<j$, because the size of (the $\sqcup$-bound) x cannot exceed $\mathfrak{b}$. If the right disjunct is selected instead, the environment will have to further provide the actual value of j. Then, using Lemma 19.9, we can figure out whether $x<j$ or not. Thus, in either case, we will know whether $x<j$ or $x\geq j$ and, if $x\geq j$, we will also know the value of j. First, suppose $x<j$. Then the content of cell $\#x$ at time t' is obviously the same as at time t, and information about this content can be obtained from the right $\sqcap$-conjunct of the third $\wedge$-conjunct of $\mathbb{F}(z,t)$.[22] Similarly if the state of $\mathcal{X}$ was not a move state at time t (and information about whether this was the case is available from the second conjunct of $\mathbb{E}(z,t)$). Now assume (we know the value of j and) $x\geq j$, and also assume the state of $\mathcal{X}$ at time t was a move state. If

$x=j$ (use Lemma 18.3 to tell if this is so or not), then the content of cell $\#x$ at time t' will be the symbol $\top$. Otherwise, if $x\neq j$, meaning that $x>j$, then the content of cell $\#x$ at time t' will be the content c of cell $\#(x-j-1)$ of the work tape at time t (Lemma 19.9 can again be used to compute the value of $x-j-1$). Such a c can be found using the left $\sqcap$-conjunct of the third $\wedge$-conjunct of $\mathbb{F}(z,t)$. Well, what we just said is true unless $x-j-1$ is greater than or equal to the location of the work-tape head at time t (known from $\mathbb{E}(z,t)$), in which case the content of cell $\#x$ of the run tape at time t' will be blank. ■

Lemma 21.14 *For any elementary formula T functional for z,* **PTA** *proves*

$$|t'|\leq\mathfrak{b}\wedge\mathbb{E}(z^T,t)\wedge\mathbb{F}(z^T,t)\rightarrow\mathbb{E}(z^T,t'). \tag{123}$$

Idea. As in the previous lemma, a single copy of the resource $\mathbb{E}(z^T,t)$ and a single copy of the resource $\mathbb{F}(z^T,t)$ turn out to be sufficient so solve the problem $\mathbb{E}(z^T,t')$. A minor additional technical condition for this in the present case is that the size of t' should not exceed $\mathfrak{b}$. ■

Proof. For reasons similar to those given at the beginning of the proof of Lemma 21.13, it would be sufficient to show the **PTA**-provability of the following formula instead of (123):

$$\forall z\Big(|t'|\leq\mathfrak{b}\wedge\mathbb{E}(z,t)\wedge\mathbb{F}(z,t)\rightarrow\mathbb{E}(z,t')\Big). \tag{124}$$

Argue in **PTA**. Consider an arbitrary ($\forall$) counterstrategy z. As in the proof of the previous lemma, the context of our discourse will be the play according to the scenario of the $(z,e_{\mathfrak{b}})$-branch. Assume $|t'|\leq\mathfrak{b}$. And assume that a single copy of the resource $\mathbb{E}(z,t)\wedge\mathbb{F}(z,t)$ is at our disposal. We need to show how to resolve $\mathbb{E}(z,t')$.

The first conjunct of $\mathbb{E}(z,t)$ is *Adequate*$(z,\mathfrak{b},t)$. It implies that the environment does not move at t or any greater time, so that z will remain adequate for any value greater than t as well. Thus, *Adequate*$(z,\mathfrak{b},t')$ is true, which takes care of the first conjunct of $\mathbb{E}(z,t')$.

The resource $\mathbb{E}(z,t)$ contains full information about the state of the machine at time t, the locations of the three scanning heads, and the symbols at those three locations. This allows us to determine the next state, and the next locations of the heads ("next" means "at time t'"). Note that we will have no problem naming those locations, as they cannot exceed t' (moving a head farther than cell $\#t'$ would require more than t' steps) and hence, in view of the assumption $|t'|\leq\mathfrak{b}$, their sizes cannot exceed $\mathfrak{b}$. What we just said fully takes care of the second conjunct of $\mathbb{E}(z,t')$, and partially takes care of the fourth, fifth and sixth conjuncts. To turn this "partial care" into a full one, we need to show how to tell the symbols looked at by the three heads at time t'.

The content of the cell scanned by the valuation-tape head at time t' will be the same as the content of that cell at time t, and this information can be obtained from the first conjunct of $\mathbb{F}(z,t)$.

Since scanning heads (almost) always move left or right, the content of the cell scanned by the work-tape head at time t' will generally also be the same as the content of that cell at time t, which can be obtained from the second conjunct of $\mathbb{F}(z,t)$. An exception is when the head is at the beginning of the tape at time t, writes a new symbol and tries to move left which, however, results in staying put. In such a case, we can obtain the symbol just written (i.e., the content of the cell scanned by the head at time t') directly from our knowledge of the transition function and our knowledge — already obtained earlier from $\mathbb{E}(z,t)$ — of the state of $\mathcal{X}$ and the contents of the three cells scanned at time t.

Let the cell scanned by the head of the run tape at time t' be cell $\#i$ (the value of i has already been established earlier). Let the leftmost blank cell of that tape at time t be cell $\#j$. Since the run-tape head can never move past the leftmost blank cell, we have either $i=j$ or ($i\neq j$ and hence) $i<j$. The third conjunct of $\mathbb{E}(z,t)$ in combination with Lemma 18.3 can be used to tell which of these two alternatives is the case. If $i<j$, then the content of the run-tape cell $\#i$ at time t' will be the same as at time t, and this information can be obtained from the right $\sqcap$-conjunct of the third $\wedge$-conjunct of $\mathbb{F}(z,t)$. Similarly if the state of $\mathcal{X}$ was not a move state at time t (and information about whether this was the case is available from the second conjunct of $\mathbb{E}(z,t)$). Assume now $i=j$, and the state of $\mathcal{X}$ at time t was a move state. Then the content of cell $\#i$ at time t' will be the symbol $\top$ (the label of the move made at time t).

The above three paragraphs complete taking care of the fourth, fifth and sixth conjuncts of $\mathbb{E}(z,t')$.

Finally, to solve the remaining third conjunct of $\mathbb{E}(z,t')$, wait till the environment selects one of the two $\sqcup$-disjuncts of the third conjunct of $\mathbb{E}(z,t)$. If the left disjunct is selected there, do the same in the third conjunct of $\mathbb{E}(z,t')$. Suppose now the right conjunct is selected. Wait till the environment further specifies a value j for x there. If $\mathcal{X}$ is not in a move state at time t, do the exact same selections in the third conjunct of $\mathbb{E}(z,t')$. Suppose now $\mathcal{X}$ is in a move state at time t. Then the location of the leftmost blank cell at time t' will be $j+i+1$, where i is the location of the work-tape head at time t. Using the results of Section 19, try to compute m with $m=j+i+1$. If $|m|$ turns out to exceed $\mathfrak{b}$, select the left $\sqcup$-disjunct of the third conjunct of $\mathbb{E}(z,t')$. Otherwise select the right disjunct, and specify x as m there. ■

Lemma 21.15 *For any elementary formula T functional for z,* **PTA** *proves*

$$|t'|\leq\mathfrak{b}\wedge\mathbb{E}(z^T,t)\wedge\mathbb{F}(z^T,t)\rightarrow\mathbb{E}(z^T,t)\wedge\big(\mathbb{E}(z^T,t')\sqcap\mathbb{F}(z^T,t')\big).$$

Idea. This is a logical consequence of the previous three lemmas (i.e. a consequence exclusively due to logical axioms and rules, without appealing to induction or any nonlogical axioms of **PTA**). Correspondingly, the proof given below is a purely syntactic exercise. ■

Proof. The following sequence is a **CL4**-proof:

1. $(p_1\rightarrow p_2\wedge\top)\wedge(\bot\wedge\bot\rightarrow\top)\wedge(q\wedge\bot\wedge\bot\rightarrow\top)\rightarrow q\wedge p_1\wedge\top\rightarrow p_2\wedge\top$
Tautology

2. $(p_1\rightarrow p_2\wedge p_3)\wedge(\bot\wedge\bot\rightarrow\top)\wedge(q\wedge p_3\wedge p_4\rightarrow p_5)\rightarrow q\wedge p_1\wedge p_4\rightarrow p_2\wedge p_5$
Tautology

3. $(p_1\rightarrow p_2\wedge p_3)\wedge(Q_1\wedge Q_2\rightarrow Q_4)\wedge(q\wedge p_3\wedge p_4\rightarrow p_5)\rightarrow q\wedge p_1\wedge p_4\rightarrow p_2\wedge p_5$
Wait: 2

4. $(p_1\rightarrow p_2\wedge Q_1)\wedge(Q_1\wedge Q_2\rightarrow Q_4)\wedge(q\wedge Q_1\wedge Q_2\rightarrow Q_3)\rightarrow q\wedge p_1\wedge Q_2\rightarrow p_2\wedge Q_3$
Match (3 times): 3

5. $(p_1\rightarrow p_2\wedge p_3)\wedge(p_3\wedge p_4\rightarrow p_5)\wedge(q\wedge\bot\wedge\bot\rightarrow\top)\rightarrow q\wedge p_1\wedge p_4\rightarrow p_2\wedge p_5$
Tautology

6. $(p_1 \to p_2 \wedge p_3) \wedge (p_3 \wedge p_4 \to p_5) \wedge (q \wedge Q_1 \wedge Q_2 \to Q_3) \to q \wedge p_1 \wedge p_4 \to p_2 \wedge p_5$
Wait: 5
7. $(p_1 \to p_2 \wedge Q_1) \wedge (Q_1 \wedge Q_2 \to Q_4) \wedge (q \wedge Q_1 \wedge Q_2 \to Q_3) \to q \wedge p_1 \wedge Q_2 \to p_2 \wedge Q_4$
Match (3 times): 6
8. $(p_1 \to p_2 \wedge Q_1) \wedge (Q_1 \wedge Q_2 \to Q_4) \wedge (q \wedge Q_1 \wedge Q_2 \to Q_3) \to$
$q \wedge p_1 \wedge Q_2 \to p_2 \wedge (Q_3 \sqcap Q_4)$
Wait: 1,4,7
9. $(Q_1 \to Q_1 \wedge Q_1) \wedge (Q_1 \wedge Q_2 \to Q_4) \wedge (q \wedge Q_1 \wedge Q_2 \to Q_3) \to$
$q \wedge Q_1 \wedge Q_2 \to Q_1 \wedge (Q_3 \sqcap Q_4)$
Match (twice): 8

The following formula matches the last formula of the above sequence and hence, by **CL4**-Instantiation, it is provable in **PTA**:

$$\begin{aligned}&\big(\mathbb{E}(z^T,t) \to \mathbb{E}(z^T,t) \wedge \mathbb{E}(z^T,t)\big) \wedge \big(\mathbb{E}(z^T,t) \wedge \mathbb{F}(z^T,t) \to \mathbb{F}(z^T,t')\big)\\ &\wedge \big(|t'| \leq \mathfrak{b} \wedge \mathbb{E}(z^T,t) \wedge \mathbb{F}(z^T,t) \to \mathbb{E}(z^T,t')\big) \to\\ &|t'| \leq \mathfrak{b} \wedge \mathbb{E}(z^T,t) \wedge \mathbb{F}(z^T,t) \to \mathbb{E}(z^T,t) \wedge \big(\mathbb{E}(z^T,t') \sqcap \mathbb{F}(z^T,t')\big).\end{aligned}$$

But, by Lemmas 21.12, 21.13 and 21.14, the three conjuncts of the antecedent of the above formula are also provable. Hence, by Modus Ponens, so is (the desired) consequent. ■

Lemma 21.16 *Assume R is an elementary formula, w is any variable, t is a variable other than $\mathfrak{b}$, z is a variable other than $\mathfrak{b}, w, t$, T is an elementary formula functional for z, and*

$$\mathbf{PTA} \vdash R \to \mathbb{E}(z^T,w) \wedge \mathbb{F}(z^T,w). \tag{125}$$

Then

$$\mathbf{PTA} \vdash R \wedge w \leq t \leq \xi(\mathfrak{b}) \to \mathbb{E}(z^T,t) \wedge \mathbb{F}(z^T,t). \tag{126}$$

Proof. Immediately from Lemmas 21.15 and 20.1. ■

21.6. Taking care of the case of large bounds

We will be using

$$\mathbb{A}(z,r,t)$$

for a natural formalization of the predicate saying that $r \leq t$, z is a $(\mathfrak{b}, r)$-adequate counterbehavior (so that $\mathfrak{b}$ is a hidden free variable of this formula) and, in the $(z, e_\mathfrak{b})$-branch, $\mathscr{X}$ is not in a move state at any time v with $r \leq v < t$.

Next, we will be using

$$\mathbb{B}(z,r,t)$$

as an abbreviation of

$$t < \xi(\mathfrak{b}) \wedge \mathbb{A}(z,r,t) \wedge \neg\mathbb{A}(z,r,t').$$

In the context of the $(z, e_\mathfrak{b})$-branch, $\mathbb{B}(z,r,t)$ thus asserts that, on the interval $[r,t]$, one single move β was made, and it was made exactly at time t. Note that, since the condition of the $(\mathfrak{b}, r)$-adequacy of z is implied by ($\mathbb{A}(z,r,t)$ and hence) $\mathbb{B}(z,r,t)$, **PA** knows that the above move β can only be made by $\mathscr{X}$.

For a variable z and an elementary formula T functional for z, as we did in the case of $\mathbb{E}$ and $\mathbb{F}$, we will write $\mathbb{A}(z^T,r,t)$ as an abbreviation of $\forall z(T \to \mathbb{A}(z,r,t))$. Similarly for $\mathbb{B}(z^T,r,t)$ and $\mathbb{W}^E(z^T,t_1,t_2,\mathfrak{b},\vec{s})$.

Lemma 21.17 *Assume $x,u,z,w,\vec{s}$ are pairwise distinct non-$\mathfrak{b}$ variables, R is an elementary formula, T is an elementary formula functional for z, $E = E(\mathfrak{b},\vec{s})$ is a safe formula all of whose free variables are among $\mathfrak{b},\vec{s}$, and the following provabilities hold:*

$$\mathbf{PTA} \vdash R \to \xi(\mathfrak{b}) = u; \tag{127}$$

$$\mathbf{PTA} \vdash R \to \mathbb{W}^E(z^T,w,w,\mathfrak{b},\vec{s}); \tag{128}$$

$$\mathbf{PTA} \vdash R \to \mathbb{E}(z^T,w) \wedge \mathbb{F}(z^T,w). \tag{129}$$

Then **PTA** *proves*

$$R \to \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \sqcup \bigsqcup x\mathbb{B}(z^T,w,x). \tag{130}$$

Idea. According to (128), **PTA** knows that, under the assumptions (of the truth of) R and T, z is a $(\mathfrak{b}, w)$-adequate counterbehavior and, in the context of the $(z, e_\mathfrak{b})$-branch, by time w, the play is legal and it has evolved to the position $E(\mathfrak{b},\vec{s})$. Under the above assumptions, the target (130) is the problem of telling whether the same situation persists up to time u (the left $\sqcup$-disjunct of the consequent), or whether a (legal or illegal) move is made at some time x with $w \leq x < \xi(\mathfrak{b})$ (the right $\sqcup$-disjunct), i.e. — in view of (127) — at some time x with $w \leq x < u$.

Solving this problem is not hard. Conditions (127) and (129), by Lemma 21.16, imply full knowledge of the configurations of the machine at any time t with $w \leq t < u$. Using this knowledge, we can trace the work of the machine step-by-step starting from w and ending with $u-1$ and see if a move is made or not. Technically, such "tracing" can be implemented relying on the induction rule of Lemma 20.2. ■

Proof. Assume all conditions of the lemma. We shall point out that the condition on the safety of E is not relevant to the present proof, and it is included in the formulation of the lemma merely for the convenience of future references.

By Lemma 21.16, condition (129) implies

$$\mathbf{PTA} \vdash R \wedge w \leq t \leq \xi(\mathfrak{b}) \rightarrow \mathbb{E}(z^T, t) \wedge \mathbb{F}(z^T, t) \tag{131}$$

which, in turn, in view of condition (127), can be easily seen to further imply

$$\mathbf{PTA} \vdash R \wedge w \leq t \leq u \rightarrow \mathbb{E}(z^T, t) \wedge \mathbb{F}(z^T, t). \tag{132}$$

Obviously $\mathbf{PA} \vdash \mathbb{W}^E(z^T, w, w, \mathfrak{b}, \vec{s}) \rightarrow \mathbb{A}(z^T, w, w)$. This, together with (128), by Transitivity, yields $\mathbf{PTA} \vdash R \rightarrow \mathbb{A}(z^T, w, w)$, whence, by $\sqcup$-Choose,

$$\mathbf{PTA} \vdash R \rightarrow \mathbb{A}(z^T, w, w) \sqcup \bigsqcup x \mathbb{B}(z^T, w, x). \tag{133}$$

Claim 1: PTA *proves*

$$R \wedge \mathbb{B}(z^T, w, t) \rightarrow \mathbb{G}_1(z^T) \sqcup \ldots \sqcup \mathbb{G}_m(z^T) \sqcup \neg \big(\mathbb{G}_1(z^T) \vee \ldots \vee \mathbb{G}_m(z^T)\big). \tag{134}$$

Proof. As in the case of Lemmas 21.13 and 21.14, we will have to limit ourselves to an informal reasoning within **PTA**. Assume $R \wedge w \leq t < \xi(\mathfrak{b})$, and (a single copy) of the resource

$$\mathbb{A}(z^T, w, t) \sqcup \bigsqcup x \mathbb{B}(z^T, w, x) \tag{135}$$

from the antecedent of (**??**) is at our disposal. Our task is to solve the consequental problem

$$\mathbb{A}(z^T, w, t') \sqcup \bigsqcup x \mathbb{B}(z^T, w, x). \tag{136}$$

The environment will have to choose one of the two $\sqcup$-disjuncts of (135). If the right disjunct is chosen, then we also choose the identical right disjunct in (136), thus reducing the (relevant part of the) overall play to $\bigsqcup x \mathbb{B}(z^T, w, x) \rightarrow \bigsqcup x \mathbb{B}(z^T, w, x)$ which, having the form $F \rightarrow F$, is, of course, solvable/provable.

Suppose now the left disjunct of (135) is chosen, bringing the latter to $\mathbb{A}(z^T, w, t)$. If this formula is false, we win. So, assume it is true. In view of (131), we have access to the resource $\mathbb{E}(z^T, t)$, which contains information about the state of the machine at time t in the play against the counterbehavior z ("*the*" due to the functionality of T for z) for which T is true. If that state is not a move state, then we resolve (136) by choosing its left component. And if that state is a move state, then we resolve (136) by choosing its right component and specifying x as t in it. With a little thought, this can be seen to guarantee a win. ■

From (133) and Claim 1, by the rule of Lemma 20.2, we find

$$\mathbf{PTA} \vdash R \wedge w \leq t \leq \xi(\mathfrak{b}) \rightarrow \mathbb{A}(z^T, w, t) \sqcup \bigsqcup x \mathbb{B}(z^T, w, x)$$

which, in view of condition (127), obviously implies

$$\mathbf{PTA} \vdash R \wedge w \leq t \leq u \rightarrow \mathbb{A}(z^T, w, t) \sqcup \bigsqcup x \mathbb{B}(z^T, w, x).$$

Applying first $\sqcap$-Introduction and then $\sqcap$-Elimination to the above formula, we get

$$\mathbf{PTA} \vdash R \wedge w \leq u \leq u \rightarrow \mathbb{A}(z^T, w, u) \sqcup \bigsqcup x \mathbb{B}(z^T, w, x). \tag{137}$$

But the condition $w \leq \xi(\mathfrak{b})$ is part of $\mathbb{W}^E(z^T, w, w, \mathfrak{b}, \vec{s})$ and hence, in view of (128) and (127), **PTA** obviously proves $R \rightarrow w \leq u \leq u$. This, in conjunction with (137), can be easily seen to imply the **PTA**-provability of

$$R \rightarrow \mathbb{A}(z^T, w, u) \sqcup \bigsqcup x \mathbb{B}(z^T, w, x). \tag{138}$$

Clearly $\mathbf{PA} \vdash \mathbb{W}^E(z^T, w, w, \mathfrak{b}, \vec{s}) \rightarrow \mathbb{A}(z^T, w, u) \rightarrow \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s})$. This, together with (128), by Transitivity, implies that **PTA** proves

$$R \rightarrow \mathbb{A}(z^T, w, u) \rightarrow \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}). \tag{139}$$

One can easily verify that **CL4** proves

$$(p \rightarrow q_1 \sqcup Q) \wedge (p \rightarrow q_1 \rightarrow q_2) \rightarrow (p \rightarrow q_2 \sqcup Q).$$

Now, (138) $\wedge$ (139) $\rightarrow$ (130) can be seen to be an instance of the above formula and hence provable in **PTA**. Modus-ponensing it with (138) and (139) yields (the **PTA**-provability of) the desired (130). ■

Assume E is a safe formula. We say that a formula G is a **$\sqcup$-deletion** of E iff G is the result of replacing in E some surface occurrence of a subformula $H_1 \sqcup \ldots \sqcup H_m$ by H_i (some $i \in \{1,\ldots,m\}$). And we say that a formula $G(y)$ is a **$\bigsqcup$-deletion** of E iff $G(y)$ is the result of replacing in E some surface occurrence of a subformula $\bigsqcup yH(y)$ by $H(y)$ (deleting "$\bigsqcup y$", that is). Note that $\sqcup$-deletions and $\bigsqcup$-deletions of a safe formula remain safe, and do not create free occurrences of variables that also have bound occurrences, which would otherwise violate Convention 8.1.

Lemma 21.18 *Assume the conditions of Lemma 21.17 are satisfied. Let $G_1 = G_1(\mathfrak{b},\vec{s}),\ldots,G_m = G_m(\mathfrak{b},\vec{s})$ be all of the $\sqcup$-deletions of E, and $H_1 = H_1(\mathfrak{b},\vec{s},y_1),\ldots,H_n = H_n(\mathfrak{b},\vec{s},y_n)$ be all of the $\bigsqcup$-deletions of E (each H_i is obtained from E by deleting a surface occurrence of "$\bigsqcup y_i$"). Let t be a fresh variable, and $\mathbb{C}(t)$ and $\mathbb{D}(t)$ be abbreviations defined by*

$$\begin{aligned}\mathbb{C}(t) &= \mathbb{W}^{G_1}(z^T,t',t',\mathfrak{b},\vec{s}) \sqcup \ldots \sqcup \mathbb{W}^{G_m}(z^T,t',t',\mathfrak{b},\vec{s});\\ \mathbb{D}(t) &= \bigsqcup y_1 \mathbb{W}^{H_1}(z^T,t',t',\mathfrak{b},\vec{s},y_1) \sqcup \ldots \sqcup \bigsqcup y_n \mathbb{W}^{H_n}(z^T,t',t',\mathfrak{b},\vec{s},y_n).\end{aligned}$$

Then **PTA** *proves*

$$R \wedge \mathbb{B}(z^T,w,t) \rightarrow \mathbb{L} \sqcup \mathbb{C}(t) \sqcup \mathbb{D}(t). \tag{140}$$

Idea. By the conditions of the lemma plus the additional condition expressed by the antecedent of (140), and in the context of the play according to the scenario of the $(z,e_{\mathfrak{b}})$-branch (for the counterbehavior z satisfying T), we — **PTA**, that is — know that, by time w, the play has evolved to the position E, and that, at time t with $w \leq t < \xi(\mathfrak{b})$, some new move β has been made by the machine. From (131), we have all information necessary to determine whether β is legal or not and — if β is legal — what move exactly it is. If β is illegal, the machine does not win X after all, so we can choose $\mathbb{L}$ in the consequent of (140). And if β is legal, then, depending on what it is, we can choose $\mathbb{C}(t)$ or $\mathbb{D}(t)$ in the consequent of (140), and then further choose in it the corresponding subcomponent $\mathbb{W}^{G_i}(z^T,t,t,\mathfrak{b},\vec{s})$ or $(\bigsqcup y_i \mathbb{W}^{H_i}(z^T,t',t',\mathfrak{b},\vec{s},y_i)$ and then) $\mathbb{W}^{H_i}(z^T,t',t',\mathfrak{b},\vec{s},c)$. ■

Proof. Assume the conditions of the lemma. Let us fix the two sets $\{\alpha_1,\ldots,\alpha_m\}$ and $\{\beta_1,\ldots,\beta_n\}$ of strings such that the move that brings $E(\mathfrak{b},\vec{s})$ down to $G_i(\mathfrak{b},\vec{s})$ is α_i,[23] and the move that brings $E(\mathfrak{b},\vec{s})$ down to $H_i(\mathfrak{b},\vec{s},c)$ (whatever constant c) is $\beta_i.c$.

For each $i \in \{1,\ldots,m\}$, let $\mathbb{G}_i(z)$ be an elementary formula saying that $\mathbb{B}(z,w,t)$ is true and the move made by the machine at time t in the $(z,e_{\mathfrak{b}})$-branch is α_i. Extending our notational practice to this formula, $\mathbb{G}_i(z^T)$ will be an abbreviation of $\forall z\big(T \rightarrow \mathbb{G}_i(z)\big)$.

Claim 1. PTA *proves*

$$R \wedge \mathbb{B}(z^T,w,t) \rightarrow \mathbb{G}_1(z^T) \sqcup \ldots \sqcup \mathbb{G}_m(z^T) \sqcup \neg\big(\mathbb{G}_1(z^T) \vee \ldots \vee \mathbb{G}_m(z^T)\big). \tag{141}$$

Proof. Let k be the greatest of the lengths of the moves $\alpha_1,\ldots,\alpha_m$. Argue in **PTA**. Assume $R \wedge \mathbb{B}(z^T,w,t)$. Consider the counterbehavior z for which T is true, and consider the play according to the scenario of the $(z,e_{\mathfrak{b}})$-branch. $\mathbb{B}(z^T,w,t)$ implies $w \leq t < \xi(\mathfrak{b})$. Therefore, in view of (131), full information is available about the situation in the machine at time t. Using this information, we first find the location l of the work-tape head and, using the results of Section 19, find a with $a = min(l,k)$. Then we construct a full picture of the contents of cells #0 through #$(a-1)$ of the work tape at time t. From this picture, we can determine whether it shows making one of the moves α_i (and which one), or none, and accordingly choose the true $\sqcup$-disjunct of the consequent of (141). ■

For each $i \in \{1,\ldots,n\}$, let $\mathbb{H}_i(z)$ be an elementary formula saying that $\mathbb{B}(z,w,t)$ is true and the move made by the machine at time t in the $(z,e_{\mathfrak{b}})$-branch has the prefix "$\beta_i.$". $\mathbb{H}_i(z^T)$ will be an abbreviation of $\forall z\big(T \rightarrow \mathbb{H}_i(z)\big)$.

Claim 2. PTA *proves*

$$R \wedge \mathbb{B}(z^T,w,t) \rightarrow \mathbb{H}_1(z^T) \sqcup \ldots \sqcup \mathbb{H}_n(z^T) \sqcup \neg\big(\mathbb{H}_1(z^T) \vee \ldots \vee \mathbb{H}_n(z^T)\big). \tag{142}$$

Proof. Similar to the proof of Claim 1. ■

For each $i \in \{1,\ldots,n\}$, let $\mathbb{H}'_i(z,y)$ be an elementary formula saying that $\mathbb{H}_i(z)$ is true and the move made by the machine at time

t in the $(z, e_{\mathfrak{b}})$-branch is $\beta_i.y$. $\mathbb{H}'_i(z^T, y)$ will be an abbreviation of $\forall z\big(T \to \mathbb{H}'_i(z, y)\big)$.

Claim 3. *For each $i \in \{1, \ldots, n\}$,* **PTA** *proves*

$$R \wedge \mathbb{H}_i(z^T) \to \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \mathbb{L}. \tag{143}$$

Proof. Take any $i \in \{1, \ldots, n\}$. Let k be the length of the string "β_i.". Let $\mathbb{J}(z, v, y)$ be a formula saying

> *" $\mathbb{H}_i(z)$ (is true) and, in the $(z, e_{\mathfrak{b}})$-branch, at time t, on the work tape, cells $\#k$ through $\#(k+v)$ spell constant y, and the location of the head is not any of the cells $\#0, \#1, \ldots, \#(k+v+1)$".*

$\mathbb{J}(z^T, v, y)$ will be an abbreviation of $\forall z\big(T \to \mathbb{J}(z, v, y)\big)$.

Argue in **PTA**. We want to prove, by WPTI induction on v, that

$$v \leq \hat{k} + \mathfrak{b} \to R \wedge \mathbb{H}_i(z^T) \to \mathbb{L} \sqcup \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \sqcup y \mathbb{J}(z^T, v, y). \tag{144}$$

The basis is

$$R \wedge \mathbb{H}_i(z^T) \to \mathbb{L} \sqcup \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \sqcup y \mathbb{J}(z^T, \mathbf{0}, y). \tag{145}$$

Assume the (truth of the) antecedent of the above. Consider the counterbehavior z for which T is true, and consider the play according to the scenario of the $(z, e_{\mathfrak{b}})$-branch. We will implicitly rely on the fact that, in view of (131) (whose antecedent is implied by $R \wedge \mathbb{H}_i(z^T)$), full information is available about the situation in the machine at time t. The problem (145) is solved as follows, where "head" and "cell" always mean those of the work tape, and "located" or "contains" mean that this is so at time t.

(1) Using the results of Section 19, figure out whether $|k+1| \leq \mathfrak{b}$ ($|\hat{k}'| \leq \mathfrak{b}$, that is) and, if so, find the values of k and $k+1$ and then continue according to Steps 2-4. If, however, $|k+1| > \mathfrak{b}$, then choose $\mathbb{L}$ in the consequent of (145) and you are done as it is guaranteed to be true. This is so because, from Axiom 13, we know that $|t| \leq \mathfrak{b}$, and thus $k+1 > t$; this, in turn, means that the head would not have enough time to go as far as cell $\#(k+1)$; and, if so, the machine cannot make a legal move at time t, so it loses.

(2) If the location of the head is not greater than k, then we are dealing with the fact of $\mathcal{X}$ having just (at time t) made an illegal move which is "β_i." or some proper initial substring of it, so choose $\mathbb{L}$ in the consequent of (145) because $\mathcal{X}$ loses.

(3) Suppose the head is located at cell $\#(k+1)$. Then:

- If cell $\#k$ contains 0, then we are dealing with the fact of $\mathcal{X}$ having made the move $\beta_i.0$, so choose $\sqcup y_i \mathbb{H}'_i(z^T, y_i)$ in the consequent of (145) and specify y_i as 0 in it.
- If cell $\#k$ contains 1, then we are dealing with the fact of $\mathcal{X}$ having made the move $\beta_i.1$, so choose $\sqcup y_i \mathbb{H}'_i(z^T, y_i)$ in the consequent of (145) and specify y_i as 1 in it.
- If cell $\#k$ contains any other symbol, then we are dealing with the fact of $\mathcal{X}$ having made an illegal move, so choose $\mathbb{L}$.

(4) Suppose the location of the head is greater than $k+1$. Then:

- If cell $\#k$ contains 0, choose $\sqcup y \mathbb{J}(z^T, \mathbf{0}, y)$ in the consequent of (145) and specify y as 0 in it.
- If cell $\#k$ contains 1, choose $\sqcup y \mathbb{J}(z^T, \mathbf{0}, y)$ in the consequent of (145) and specify y as 1 in it.
- If cell $\#k$ contains any other symbol, choose $\mathbb{L}$.

The inductive step is

$$\begin{array}{l}\big(R \wedge \mathbb{H}_i(z^T) \to \mathbb{L} \sqcup \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \sqcup y \mathbb{J}(z^T, v, y)\big) \to \\ \big(R \wedge \mathbb{H}_i(z^T) \to \mathbb{L} \sqcup \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \sqcup y \mathbb{J}(z^T, v', y)\big).\end{array} \tag{146}$$

Assume $R \wedge \mathbb{H}_i(z^T)$ is true (otherwise (146) is won). Under this assumption, solving (146) essentially means solving the following problem:

$$\begin{array}{l}\mathbb{L} \sqcup \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \sqcup y \mathbb{J}(z^T, v, y) \to \\ \mathbb{L} \sqcup \sqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \sqcup y \mathbb{J}(z^T, v', y).\end{array} \tag{147}$$

This problem is solved as follows. Wait for the environment to choose a $\sqcup$-disjunct in the antecedent. If that choice is one of the first two disjuncts, choose the identical disjunct in the consequent, and then resolve the resulting problem of the form $F \rightarrow F$. Suppose now the third disjunct $\bigsqcup y \mathbb{J}(z^T, v, y)$ is chosen. Wait till it is further brought to $\mathbb{J}(z^T, v, c)$ for some c. Consider the counterbehavior z for which T is true, and consider the play according to the scenario of the $(z, e_{\mathfrak{b}})$-branch. As was done when justifying the basis of induction, we will rely on the fact that, in view of (131), full information is available about the situation in the machine at time t. In our subsequent discourse, "head" and "cell" always mean those of the work tape, and "located" or "contains" mean that this is so at time t. Note that, as implied by $\mathbb{J}(z^T, v, c)$, the location of the head is greater than $k + v'$. So, using the results of Section 19, we can tell ($\sqcup$) whether that location is $k + v' + 1$ or greater than $k + v' + 1$. We correspondingly consider the following two cases and resolve the consequent of (147) accordingly:

(1) Suppose the head is located at cell $\#(k + v' + 1)$. Then:

- If cell $\#(k + v')$ contains 0, then we are dealing with the fact of $\mathcal{X}$ having made the move $\beta_i.c0$, so choose $\bigsqcup y_i \mathbb{H}'_i(z^T, y_i)$ in the consequent of (147) and specify y_i as $c0$ in it.
- If cell $\#(k + v')$ contains 1, then we are dealing with the fact of $\mathcal{X}$ having made the move $\beta_i.c1$, so choose $\bigsqcup y_i \mathbb{H}'_i(z^T, y_i)$ in the consequent of (147) and specify y_i as $c1$ in it.
- If cell $\#k$ contains any other symbol, then we are dealing with the fact of $\mathcal{X}$ having made an illegal move, so choose $\mathbb{L}$.

(2) Suppose the location of the head is greater than $k + v' + 1$. Then:

- If cell $\#k$ contains 0, choose $\bigsqcup y \mathbb{J}(z^T, v', y)$ in the consequent of (147) and specify y as $c0$ in it.
- If cell $\#k$ contains 1, choose $\bigsqcup y \mathbb{J}(z^T, v', y)$ in the consequent of (147) and specify y as $c1$ in it.
- If cell $\#k$ contains any other symbol, choose $\mathbb{L}$.

Now, (144) follows by WPTI from (145) and (146).

We continue our proof of Claim 3 by arguing in **PTA** towards the goal of justifying (143). Assume (the truth of) the antecedent of the latter. As before, we let z be the counterbehavior satisfying T, and let the context of our discourse be the play according to the scenario of the $(z, e_{\mathfrak{b}})$-branch. From (131), we find the location l of the work-tape head at time t. If $l = \mathbf{0}$, we are dealing with the fact of the machine having made an illegal move (the empty move), so choose $\mathbb{L}$ in the consequent of (143). Otherwise, we find the number a with $l = a'$ (the results of Section 19 will allow us to do so). From (144), we get

$$a \leq \hat{k} + \mathfrak{b} \rightarrow R \wedge \mathbb{H}_i(z^T) \rightarrow \mathbb{L} \sqcup \bigsqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \bigsqcup y \mathbb{J}(z^T, a, y). \quad (148)$$

Next, we figure out (again relying on the results of Section 19) whether $a \leq k + \mathfrak{b}$ or not. If not, we are obviously dealing with the case of the machine having made an illegal ("too long a") move, so we choose $\mathbb{L}$ in the consequent of (143). Suppose now $a \leq k + \mathfrak{b}$. Then, from (148) by Modus Ponens applied twice, we get

$$\mathbb{L} \sqcup \bigsqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \bigsqcup y \mathbb{J}(z^T, a, y). \quad (149)$$

Our remaining task is to show how to solve the consequent

$$\bigsqcup y_i \mathbb{H}'_i(z^T, y_i) \sqcup \mathbb{L} \quad (150)$$

of (143) using the resource (149). This is very easy. Wait till the environment selects a $\sqcup$-disjunct of (149). If one of the first two disjuncts is selected, select the identical disjunct in (150) and, having brought things down to a problem of the form $F \rightarrow F$, solve it. And if the third disjunct $\bigsqcup y \mathbb{J}(z^T, a, y)$ of (149) is selected, we win. That is because, no matter what c the environment further selects for y in it, the resulting formula $\mathbb{J}(z^T, a, c)$ will be false as it implies that the work-tape head at time t is not located at cell $\#(a + 1)$, which is a contradiction — as we remember, $l = a'$ is exactly the location of the head.

Our proof of Claim 3 is now complete. ■

Claim 4. **PA** $\vdash R \wedge \mathbb{B}(z^T, w, t) \rightarrow \neg\big(\mathbb{G}_1(z^T) \vee \ldots \vee \mathbb{G}_m(z^T)\big) \wedge \neg\big(\mathbb{H}_1(z^T) \vee \ldots \vee \mathbb{H}_n(z^T)\big) \rightarrow \mathbb{L}$.

Proof. This and the following two claims can be proven by a straightforward argument within **PA** based on the meanings of the predicates involved in the formula. Assume $R \wedge \mathbb{B}(z^T, w, t)$ and

$$\neg\big(\mathbb{G}_1(z^T) \vee \ldots \vee \mathbb{G}_m(z^T)\big) \wedge \neg\big(\mathbb{H}_1(z^T) \vee \ldots \vee \mathbb{H}_n(z^T)\big). \quad (151)$$

Consider the counterbehavior z satisfying T, and the play according to the scenario of the $(z, e_{\mathfrak{b}})$-branch. According to (128), by time w the play has evolved to position $E(\mathfrak{b}, \vec{s})$. And, according to $\mathbb{B}(z^T, w, t)$, a (first new) move β has been made by the machine at time t. Obviously the assumption (151) precludes the possibility of such a β being a legal move of $E(\mathfrak{b}, \vec{s})$. So, β is illegal, which makes the machine lose the game, and hence $\mathbb{L}$ is true. ■

Claim 5. *For each* $i \in \{1, \ldots, m\}$, $\mathbf{PA} \vdash R \to \mathbb{G}_i(z^T) \to \mathbb{W}^{G_i}(z^T, t', t', \mathfrak{b}, \vec{s})$.

Proof. Argue in **PA**. Assume R and $\mathbb{G}_i(z^T)$. Consider the counterbehavior z satisfying T, and the play according to the scenario of the $(z, e_{\mathfrak{b}})$-branch. According to (128), by time w the play has evolved to position $E(\mathfrak{b}, \vec{s})$. And, according to $\mathbb{G}_i(z^T)$, a (first new) move has been made by the machine at time t, and such a move is α_i. But this move brings $E(\mathfrak{b}, \vec{s})$ down to $G_i(\mathfrak{b}, \vec{s})$. This, in turn, implies that $\mathbb{W}^{G_i}(z^T, t', t', \mathfrak{b}, \vec{s})$ is true. ■

Claim 6. *For each* $i \in \{1, \ldots, n\}$, $\mathbf{PA} \vdash R \to \mathbb{H}'_i(z^T, y_i) \to \mathbb{W}^{H_i}(z^T, t', t', \mathfrak{b}, \vec{s}, y_i)$.

Proof. Very similar to the proof of Claim 5. Argue in **PA**. Assume R and $\mathbb{H}'_i(z^T, y_i)$. Consider the counterbehavior z satisfying T, and the play according to the scenario of the $(z, e_{\mathfrak{b}})$-branch. According to (128), by time w the play has evolved to position $E(\mathfrak{b}, \vec{s})$. And, according to $\mathbb{H}_i(z^T, y_i)$, a (first new) move has been made by the machine at time t, and such a move is $\beta_i.y_i$. But this move brings $E(\mathfrak{b}, \vec{s})$ down to $H_i(\mathfrak{b}, \vec{s}, y_i)$. This, in turn, implies that $\mathbb{W}^{H_i}(z^T, t', t', \mathfrak{b}, \vec{s}, y_i)$ is true. ■

To complete our proof of Lemma 21.18, it remains to observe that (140) is a logical consequence of Claims 1-6. Since we have played more than enough with **CL4**, here we only schematically outline how to do this purely syntactic exercise.

First of all, Claims 1 and 2 can be easily seen to imply

$$\mathbf{PTA} \vdash R \wedge \mathbb{B}(z^T, w, t) \to \mathbb{G}_1(z^T) \sqcup \ldots \sqcup \mathbb{G}_m(z^T) \sqcup \mathbb{H}_1(z^T) \sqcup \ldots \sqcup \mathbb{H}_n(z^T) \sqcup \Big(\neg\big(\mathbb{G}_1(z^T) \vee \ldots \vee \mathbb{G}_m(z^T)\big) \wedge \neg\big(\mathbb{H}_1(z^T) \vee \ldots \vee \mathbb{H}_n(z^T)\big)\Big).$$

The above, together with Claim 4, further implies

$$\mathbf{PTA} \vdash R \wedge \mathbb{B}(z^T, w, t) \to \mathbb{G}_1(z^T) \sqcup \ldots \sqcup \mathbb{G}_m(z^T) \sqcup \mathbb{H}_1(z^T) \sqcup \ldots \sqcup \mathbb{H}_n(z^T) \sqcup \mathbb{L}.$$

This, in turn, together with Claim 5, further implies

$$\mathbf{PTA} \vdash R \wedge \mathbb{B}(z^T, w, t) \to \mathbb{C}(t) \sqcup \mathbb{H}_1(z^T) \sqcup \ldots \sqcup \mathbb{H}_n(z^T) \sqcup \mathbb{L}.$$

The above, together with Claim 3, further implies

$$\mathbf{PTA} \vdash R \wedge \mathbb{B}(z^T, w, t) \to \mathbb{C}(t) \sqcup \bigsqcup y_1 \mathbb{H}'_1(z^T, y_i) \sqcup \ldots \sqcup \bigsqcup y_n \mathbb{H}'_n(z^T, y_n) \sqcup \mathbb{L}. \tag{152}$$

Claim 6 can be seen to imply

$$\mathbf{PTA} \vdash R \to \bigsqcup y_i \mathbb{H}'_i(z^T, y_i) \to \bigsqcup y_i \mathbb{W}^{H_i}(z^T, t', t', \mathfrak{b}, \vec{s}, y_i)$$

for each $i \in \{1, \ldots, n\}$. This, together with (152), can be seen to imply the desired (140). ■

Lemma 21.19 *Under the conditions of Lemma 21.17 and using the abbreviations of Lemma 21.18,* **PTA** *proves*

$$R \to \mathbb{L} \sqcup \bigsqcup x\big(\mathbb{C}(x) \sqcup \mathbb{D}(x)\big) \sqcup \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}). \tag{153}$$

Idea. This is a logical consequence of the previous two lemmas. ■

Proof. Assume the conditions of Lemma 21.17. Then, according to Lemmas 21.17 and 21.18, **PTA** proves (130) and (140) (where, in the latter, t is a fresh variable). The target formula (153) is a logical consequence of those two formulas, verifying which is a purely syntactic exercise. As we did in the proof of the previous lemma, here we only provide a scheme for such a verification. It is rather simple. First, applying $\sqcup$-Choose and $\sqcap$-Introduction to (140), we get

$$\mathbf{PTA} \vdash R \wedge \bigsqcup x \mathbb{B}(z^T, w, x) \to \mathbb{L} \sqcup \bigsqcup x\big(\mathbb{C}(x) \sqcup \mathbb{D}(x)\big).$$

And then we observe that the above, together with $\mathbf{PTA} \vdash$ (130), implies $\mathbf{PTA} \vdash$ (153). ■

Lemma 21.20 *Under the conditions of Lemma 21.17,* $\mathbf{PTA} \vdash R \rightarrow \overline{E(\mathfrak{b},\vec{s})}$.

Idea. Under the assumption of the truth of R, one of the three $\sqcup$-disjuncts of the consequent of (153) is available as a resource. In each case, we need to show (in **PTA**) how to solve the target $\overline{E} = \overline{E(\mathfrak{b},\vec{s})}$.

1. The case of $\mathbb{L}$ is taken care of by Lemma 21.3, according to which $\mathbf{PTA} \vdash \mathbb{L} \rightarrow \overline{E}$.

2. The case of $\bigsqcup x\big(\mathbb{C}(x) \sqcup \mathbb{D}(x)\big)$, depending on which of its $\bigsqcup$- and $\sqcup$-components are further chosen, allows us to jump to a formula F (one of the G_i, $1 \leq i \leq m$ or H_i, $1 \leq i \leq n$) from which E follows by $\sqcup$-Choose or $\bigsqcup$-Choose. With appropriately readjusted R and certain other parameters, by the induction hypothesis, we know how to solve $\overline{F}$. Then (by $\sqcup$-Choose or $\bigsqcup$-Choose) we also know how to solve $\overline{E}$.

3. Finally, consider the case of $\mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s})$. E can be critical or non-critical. The case of E being critical is almost immediately taken care of by Lemma 21.5, according to which $\mathbf{PTA} \vdash \exists z \mathbb{W}^E(z,w,u,\mathfrak{b},\vec{s}) \rightarrow \overline{E}$. Suppose now E is non-critical. Then, by Lemma 21.4, according to which $\mathbf{PTA} \vdash \exists z \mathbb{W}^E(z,w,u,\mathfrak{b},\vec{s}) \rightarrow \|\overline{E}\|$, the elementarization of $\overline{E}$ is true/provable. Relying on the induction hypothesis as in the previous case, and replacing $T(z)$ by a formula $S(z)$ saying that z is a certain one-move extension of the counterbehavior satisfying T, we manage to show that any other (other than $\|\overline{E}\|$) necessary Wait-premise of $\overline{E}$ is also solvable/provable. Then, by Wait, we know how to solve/prove $\overline{E}$.

■

Proof. We prove this lemma by (meta)induction on the complexity of $E(\mathfrak{b},\vec{s})$. Assume the conditions of Lemma 21.17. Then, by Lemma 21.19, $\mathbf{PTA} \vdash$ (153).

By Lemma 21.3, $\mathbf{PTA} \vdash \mathbb{L} \rightarrow \overline{E(\mathfrak{b},\vec{s})}$, whence, by Weakening,

$$\mathbf{PTA} \vdash \mathbb{L} \wedge R \rightarrow \overline{E(\mathfrak{b},\vec{s})}. \tag{154}$$

In what follows, we will rely on the additional assumptions and abbreviations of Lemma 21.18.

Claim 1. *For each* $i \in \{1,\ldots,m\}$, $\mathbf{PTA} \vdash \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{E(\mathfrak{b},\vec{s})}$.

Proof. Pick any $i \in \{1,\ldots,m\}$ and a fresh variable v.

From condition (127), by Weakenings, we have

$$\mathbf{PTA} \vdash v = t' \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow \xi(\mathfrak{b}) = u. \tag{155}$$

And, of course, we also have

$$\mathbf{PTA} \vdash v = t' \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow \mathbb{W}^{G_i}(z^T,v,v,\mathfrak{b},\vec{s}). \tag{156}$$

Condition (129) and Lemma 21.16 imply

$$\mathbf{PTA} \vdash R \wedge w \leq v \leq \xi(\mathfrak{b}) \rightarrow \mathbb{E}(z^T,v) \wedge \mathbb{F}(z^T,v). \tag{157}$$

In view of condition (128), we also obviously have

$$\mathbf{PTA} \vdash v = t' \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow R \wedge w \leq v \leq \xi(\mathfrak{b}). \tag{158}$$

From (158) and (157), by Transitivity, we get

$$\mathbf{PTA} \vdash v = t' \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow \mathbb{E}(z^T,v) \wedge \mathbb{F}(z^T,v). \tag{159}$$

By the induction hypothesis of our lemma, with v, $G_i(\mathfrak{b},\vec{s})$ and $v = t' \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R$ in the roles of w, $E(\mathfrak{b},\vec{s})$ and R, (155), (156) and (159) — which correspond to (127), (128) and (129), respectively — imply

$$\mathbf{PTA} \vdash v = t' \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{G_i(\mathfrak{b},\vec{s})}.$$

The above, by $\sqcap$-Introduction, yields

$$\mathbf{PTA} \vdash \bigsqcup x(x = t') \wedge \mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{G_i(\mathfrak{b},\vec{s})}. \tag{160}$$

Remembering the definition of $\mathbb{W}$, the condition $t' \leq \xi(\mathfrak{b})$ is one of the conjuncts of $\mathbb{W}^{G_i}(z^T,t',t',\mathfrak{b},\vec{s})$. Hence $\mathbf{PA} \vdash \mathbb{W}^{G_i}\big(z^T,t',t',\mathfrak{b},\vec{s}\big) \rightarrow t' \leq \xi(\mathfrak{b})$. Together with condition (127), this implies

$$\mathbf{PTA} \vdash \mathbb{W}^{G_i}\big(z^T,t',t',\mathfrak{b},\vec{s}\big) \wedge R \rightarrow t' \leq u.$$

But, by Axiom 13, $\mathbf{PTA} \vdash |u| \leq \mathfrak{b}$. Hence, obviously, $\mathbf{PTA} \vdash \mathbb{W}^{G_i}\big(z^T,t',t',\mathfrak{b},\vec{s}\big) \wedge R \rightarrow |t'| \leq \mathfrak{b}$. This, together with in Axiom 10, by Transitivity, yields $\mathbf{PTA} \vdash \mathbb{W}^{G_i}\big(z^T,t',t',\mathfrak{b},\vec{s}\big) \wedge R \rightarrow \bigsqcup x(x = t')$. And the latter, in turn, in conjunction with (160), can be seen to imply

$$\mathbf{PTA} \vdash \mathbb{W}^{G_i}(z^T, t', t', \mathfrak{b}, \vec{s}) \wedge R \rightarrow \overline{G_i(\mathfrak{b}, \vec{s})}. \qquad (161)$$

Now, it remains to notice that the desired $\mathbb{W}^{G_i}(z^T, t', t', \mathfrak{b}, \vec{s}) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}$ follows from (161) by $\sqcup$-Choose. ■

Claim 2. *For each $i \in \{1, \ldots, n\}$,* $\mathbf{PTA} \vdash \bigsqcup y_i \mathbb{W}^{H_i}(z^T, t', t', \mathfrak{b}, \vec{s}, y_i) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}$.

Proof. Pick any $i \in \{1, \ldots, n\}$. Arguing as we did for (161) in the proof of Claim 1, we find

$$\mathbf{PTA} \vdash \mathbb{W}^{H_i}(z^T, t', t', \mathfrak{b}, \vec{s}, y_i) \wedge R \rightarrow \overline{H_i(\mathfrak{b}, \vec{s}, y_i)}.$$

Applying first $\bigsqcup$-Choose and then $\bigsqcap$-Introduction to the above, we get the desired conclusion $\mathbf{PTA} \vdash \bigsqcup y_i \mathbb{W}^{H_i}(z^T, t', t', \mathfrak{b}, \vec{s}, y_i) \wedge R \rightarrow \overline{E}$. ■

Claims 1 and 2, by $\sqcap$-Introductions, imply

$$\mathbf{PTA} \vdash \Big(\big(\mathbb{W}^{G_1}(z^T, t', t', \mathfrak{b}, \vec{s}) \sqcup \ldots \sqcup \mathbb{W}^{G_m}(z^T, t', t', \mathfrak{b}, \vec{s})\big) \sqcup \big(\bigsqcup y_1 \mathbb{W}^{H_1}(z^T, t', t', \mathfrak{b}, \vec{s}, y_1) \sqcup \ldots \sqcup \bigsqcup y_n \mathbb{W}^{H_n}(z^T, t', t', \mathfrak{b}, \vec{s}, y_n)\big) \Big) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}$$

which, using the abbreviations of Lemma 21.18, is written as $\mathbf{PTA} \vdash \big(\mathbb{C}(t) \sqcup \mathbb{D}(t)\big) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}$. The latter, by $\bigsqcap$-Introduction, yields

$$\mathbf{PTA} \vdash \bigsqcup x \big(\mathbb{C}(x) \sqcup \mathbb{D}(x)\big) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}. \qquad (162)$$

Claim 3. *If $E(\mathfrak{b}, \vec{s})$ is not critical, then* $\mathbf{PTA} \vdash \|\mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}\|$.

Proof. Assume $E(\mathfrak{b}, \vec{s})$ is not critical. Then, Lemma 21.4, together with the fact of T being functional for z, can be easily seen to imply $\mathbf{PTA} \vdash \mathbb{W}^E(z^T, w, \xi(\mathfrak{b}), \mathfrak{b}, \vec{s}) \rightarrow \|\overline{E(\mathfrak{b}, \vec{s})}\|$. Remembering condition (127), the latter can be seen to further imply $\mathbf{PTA} \vdash \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \wedge R \rightarrow \|\overline{E(\mathfrak{b}, \vec{s})}\|$, which is the same as to say that $\mathbf{PTA} \vdash \|\mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \wedge R \rightarrow \overline{E(\mathfrak{b}, \vec{s})}\|$, because both $\mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s})$ and R are elementary. ■

Claim 4. *Assume $E(\mathfrak{b}, \vec{s})$ has the form $F[J_1 \sqcap \ldots \sqcap J_k]$, and $i \in \{1, \ldots, k\}$. Then*

$$\mathbf{PTA} \vdash \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \wedge R \rightarrow \overline{F[J_i]}.$$

Proof. From (127), by Weakening, we have

$$\mathbf{PTA} \vdash \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \wedge R \rightarrow \xi(\mathfrak{b}) = u. \qquad (163)$$

Assume $E(\mathfrak{b}, \vec{s}) = F[J_1 \sqcap \ldots \sqcap J_k]$ and $1 \leq i \leq k$. Let α be the move whose effect is turning $F[J_1 \sqcap \ldots \sqcap J_k]$ into $F[J_i]$. Let us write our formula T in the form $T(z)$. Let $S = S(z)$ be a formula saying that z is the code of the counterbehavior resulting by adding the timestamped move $(\alpha, w-1)$ to the counterbehavior a for which $T(a)$ holds. Of course, S is functional for z. It is not hard to see that **PA** proves $\mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \rightarrow \mathbb{W}^E(z^T, w, w, \mathfrak{b}, \vec{s})$ and $\mathbb{W}^E(z^T, w, w, \mathfrak{b}, \vec{s}) \rightarrow \mathbb{W}^{F[J_i]}(z^S, w, w, \mathfrak{b}, \vec{s})$. Therefore it proves $\mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \rightarrow \mathbb{W}^{F[J_i]}(z^S, w, w, \mathfrak{b}, \vec{s})$, whence, by Weakening,

$$\mathbf{PTA} \vdash \mathbb{W}^E(z^T, w, u, \mathfrak{b}, \vec{s}) \wedge R \rightarrow \mathbb{W}^{F[J_i]}(z^S, w, w, \mathfrak{b}, \vec{s}). \qquad (164)$$

Next, we claim that

$$\mathbf{PTA} \vdash R \rightarrow \mathbb{E}(z^S, w) \wedge \mathbb{F}(z^S, w). \qquad (165)$$

Here is a brief justification of (165) through reasoning in **PTA**. Let a be the counterbehavior for which $T(a)$ is true, and let d be the counterbehavior for which $S(d)$ is true. Assume R. Then, in view of (129), the resource $\mathbb{E}(z^T, w) \wedge \mathbb{F}(z^T, w)$ is available for us in unlimited supply. That is, we have full information about the configuration of $\mathcal{X}$ at time w in the $(a, e_{\mathfrak{b}})$-branch. Solving (165) means being able to generate full information about the configuration of $\mathcal{X}$ at time w in the $(d, e_{\mathfrak{b}})$-branch. Since the time w is fixed and is the same in both cases, let us no longer explicitly mention it. Note that the two configurations are identical, for the exception of the contents of the run tape. So, from the resource $\mathbb{E}(z^T, w) \wedge \mathbb{F}(z^T, w)$ which describes the configuration of the $(a, e_{\mathfrak{b}})$-branch, we can directly tell the (identical) state of $\mathcal{X}$ in the configuration of the $(d, e_{\mathfrak{b}})$-branch, as well as the locations of all three scanning heads, and the contents of any cells of the valuation and work tapes. Next, in order to tell the location of the leftmost blank cell on the run tape in the configuration of the $(d, e_{\mathfrak{b}})$-branch (or tell that the size of this location exceeds $\mathfrak{b}$), all we need is to compute $i+j+1$, where i is the location of the leftmost blank cell of the run tape in the configuration of the $(a, e_{\mathfrak{b}})$-branch (unless $|i| \geq \mathfrak{b}$, in which case the size of the sought value also exceeds $\mathfrak{b}$), and j is the location of the work-tape head in the configuration of the $(a,e_{\mathfrak{b}})$-branch. Finally, consider any cell $\#c$ of the run tape. If c is less than the above i, then the content of cell $\#c$ in the configuration of the $(d,e_{\mathfrak{b}})$-branch is the same as in the $(a,e_{\mathfrak{b}})$-branch. Otherwise, if $c \geq i$, then the sought content is the $(c - i)$th symbol (starting the count of those symbols from 0 rather than 1) of the labmove $\bot\alpha$ — unless $c - i$ is greater or equal to the length of this labmove, in which case the sought content of cell $\#c$ is blank.

From (165), by Weakening, we have

$$\mathbf{PTA} \vdash \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R \rightarrow \mathbb{E}(z^S,w) \wedge \mathbb{F}(z^S,w). \qquad (166)$$

By the induction hypothesis of our lemma, with $F[J_i]$ and $\mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R$ in the roles of E and R, (163), (164) and (166) — which correspond to (127), (128) and (129), respectively — imply the desired $\mathbf{PTA} \vdash \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{F[J_i]}$.

■

Claim 5. *Assume $E(\mathfrak{b},\vec{s})$ has the form $F[\sqcap xJ(x)]$, and v is a variable not occurring in $E(\mathfrak{b},\vec{s})$. Then*

$$\mathbf{PTA} \vdash \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{F[J(v)]}.$$

Proof. Assume $E = F[\sqcap xJ(x)]$, and v is a fresh variable. Let α be the string such that, for whatever constant c, $\alpha.c$ is the move which brings $F[\sqcap xJ(x)]$ down to $F[J(c)]$. Arguing almost literally as in the proof of Claim 4, only with "$\alpha.v$" instead of α and "$J(v)$" instead of "J_i", we find that $\mathbf{PTA} \vdash \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{F[J(v)]}$. The only difference and minor complication is related to the fact that, while in the proof of Claim 4 the labmove $\top\alpha$ was constant, the corresponding labmove $\top\alpha.v$ in the present case is not. Hence, its size is not given directly but rather needs to be determined (while arguing within **PTA**). No problem, this (for the "v" part of the labmove) can be done using Lemma 19.1. Similarly, various symbols of the labmove that were given directly in the proof of Claim 4 will now have to be determined using some general procedure. Again no problem: this (for the "v" part of the labmove) can be done using Lemma 19.10. ■

Now we claim that

$$\mathbf{PTA} \vdash \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{E(\mathfrak{b},\vec{s})}. \qquad (167)$$

Indeed, if $E(\mathfrak{b},\vec{s})$ is not critical, then the above follows from Claims 3, 4 and 5 by the closure of **PTA** under Wait. Suppose now $E(\mathfrak{b},\vec{s})$ is critical. Then, by Lemma 21.5, $\mathbf{PTA} \vdash \exists z\mathbb{W}^E(z,w,\xi(\mathfrak{b}),\mathfrak{b},\vec{s}) \rightarrow \overline{E(\mathfrak{b},\vec{s})}$. This, in view of the functionality of T for z and condition (127), can be easily seen to imply $\mathbf{PTA} \vdash \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s}) \wedge R \rightarrow \overline{E(\mathfrak{b},\vec{s})}$, as claimed.

From (154), (162) and (167), by $\sqcap$-Introduction, we find that **PTA** proves

$$\Big(\mathbb{L} \sqcup \sqcup x\big(\mathbb{C}(x) \sqcup \mathbb{D}(x)\big) \sqcup \mathbb{W}^E(z^T,w,u,\mathfrak{b},\vec{s})\Big) \wedge R \rightarrow \overline{E(\mathfrak{b},\vec{s})}. \qquad (168)$$

In turn, the **PTA**-provability of (153) and (168) can be easily seen to imply the desired **PTA**-provability of $R \rightarrow \overline{E(\mathfrak{b},\vec{s})}$. This completes our proof of Lemma 21.20. ■

Lemma 21.21 $\mathbf{PTA} \vdash \sqcup x\big(x = \xi(\mathfrak{b})\big) \rightarrow \overline{X(\mathfrak{b})}$.

Idea. We take $u = \xi(\mathfrak{b}) \wedge w = \mathbf{0}'$ in the role of R, X in the role of E and show that the conditions of Lemma 21.17 are satisfied. Then, by Lemma 21.20, **PTA** proves $u = \xi(\mathfrak{b}) \wedge w = \mathbf{0}' \rightarrow \overline{X}$. And the target formula $\sqcup x\big(x = \xi(\mathfrak{b})\big) \rightarrow \overline{X}$ is an almost immediate logical consequence of the latter and Lemma 13.3. ■

Proof. Let R be the formula $u = \xi(\mathfrak{b}) \wedge w = \mathbf{0}'$. Then, of course, we have

$$\mathbf{PTA} \vdash R \rightarrow \xi(\mathfrak{b}) = u. \qquad (169)$$

Let $T(z)$ be an elementary formula saying that z is (the code of) the empty counterbehavior. Obviously **PA** proves $\mathfrak{b} \neq \mathbf{0} \rightarrow \mathbb{W}^X(z^T,\mathbf{0}',\mathbf{0}',\mathfrak{b})$ and hence, in view of Lemma 13.2, **PTA** proves $\mathbb{W}^X(z^T,\mathbf{0}',\mathbf{0}',\mathfrak{b})$. Therefore, as R contains the condition $w = \mathbf{0}'$,

$$\mathbf{PTA} \vdash R \rightarrow \mathbb{W}^X(z^T,w,w,\mathfrak{b}). \qquad (170)$$

Next, we observe that $\mathbf{PTA} \vdash \mathbb{E}(z^T,\mathbf{0}) \wedge \mathbb{F}(z^T,\mathbf{0})$. Indeed, arguing in **PTA**, solving both $\mathbb{E}(z^T,\mathbf{0})$ and $\mathbb{F}(z^T,\mathbf{0})$ is very easy as we know exactly and fully the situation in the machine at time 0, which is nothing but the start configuration of the machine. The observation that we just made, of course, implies

$$\mathbf{PTA} \vdash v = \mathbf{0} \rightarrow \mathbb{E}(z^T, v) \wedge \mathbb{F}(z^T, v). \tag{171}$$

From (171), by Lemma 21.16, we get

$$\mathbf{PTA} \vdash v = \mathbf{0} \wedge v \leq w \leq \xi(\mathfrak{b}) \rightarrow \mathbb{E}(z^T, w) \wedge \mathbb{F}(z^T, w). \tag{172}$$

Since $w = \mathbf{0}'$ is a conjunct of R, **PA** obviously proves $\mathfrak{b} \neq \mathbf{0} \rightarrow v = \mathbf{0} \rightarrow R \rightarrow v = \mathbf{0} \wedge v \leq w \leq \xi(\mathfrak{b})$.[24] But, by Lemma 13.2, $\mathbf{PTA} \vdash \mathfrak{b} \neq \mathbf{0}$. Hence, by Modus Ponens, $\mathbf{PTA} \vdash v = \mathbf{0} \rightarrow R \rightarrow v = \mathbf{0} \wedge v \leq w \leq \xi(\mathfrak{b})$. From here and (172), by Transitivity, we get

$$\mathbf{PTA} \vdash v = \mathbf{0} \rightarrow R \rightarrow \mathbb{E}(z^T, w) \wedge \mathbb{F}(z^T, w)$$

whence, by $\sqcap$-Introduction,

$$\mathbf{PTA} \vdash \sqcup x(x = \mathbf{0}) \rightarrow R \rightarrow \mathbb{E}(z^T, w) \wedge \mathbb{F}(z^T, w),$$

modus-ponensing which with Axiom 8 yields

$$\mathbf{PTA} \vdash R \rightarrow \mathbb{E}(z^T, w) \wedge \mathbb{F}(z^T, w). \tag{173}$$

Now, with $X(\mathfrak{b})$ in the role of $E(\mathfrak{b}, \vec{s})$, the conditions (169), (170) and (173) are identical to the conditions (127), (128) and (129) of Lemma 21.17. Hence, by Lemma 21.20, we have $\mathbf{PTA} \vdash R \rightarrow \overline{X(\mathfrak{b})}$, i.e.

$$\mathbf{PTA} \vdash u = \xi(\mathfrak{b}) \wedge w = \mathbf{0}' \rightarrow \overline{X(\mathfrak{b})}.$$

From the above, by $\sqcap$-Introduction, we get

$$\mathbf{PTA} \vdash u = \xi(\mathfrak{b}) \wedge \sqcup x(x = \mathbf{0}') \rightarrow \overline{X(\mathfrak{b})}.$$

But the second conjunct of the antecedent of the above formula is provable by Lemma 13.3. Hence, we obviously have $\mathbf{PTA} \vdash u = \xi(\mathfrak{b}) \rightarrow \overline{X(\mathfrak{b})}$ which, by $\sqcap$-Introduction, yields the desired $\mathbf{PTA} \vdash \sqcup x\big(x = \xi(\mathfrak{b})\big) \rightarrow \overline{X(\mathfrak{b})}$.
■

21.7. *Completing the completeness proof*

By Lemma 19.8, $\mathbf{PTA} \vdash \neg|\xi(\mathfrak{b})| \leq \mathfrak{b} \sqcup \sqcup x\big(x = \xi(\mathfrak{b})\big)$. By Lemmas 21.11 and 21.21, we also have $\mathbf{PTA} \vdash \neg|\xi(\mathfrak{b})| \leq \mathfrak{b} \rightarrow \overline{X}$ and $\mathbf{PTA} \vdash \sqcup x\big(x = \xi(\mathfrak{b})\big) \rightarrow \overline{X}$. From these three facts, by $\sqcup$-Elimination, $\mathbf{PTA} \vdash \overline{X}$.

22. INHERENT EXTENSIONAL INCOMPLETENESS IN THE GENERAL CASE

The extensional completeness of **PTA** is not a result that could be taken for granted. In this short section we argue that, if one considers computability-in-general instead of polynomial time computability, extensional completeness is impossible to achieve for whatever recursively axiomatizable sound extension of **PTA**.

First of all, we need to clarify what is meant by considering computability-in-general instead of polynomial time computability. This simply means a minor readjustment of the semantics of ptarithmetic. Namely, such a readjusted semantics would be the same as the semantics we have been considering so far, with the only difference being that the time complexity of the machine solving a given problem would no longer be required to be polynomial, but rather it would be allowed to be arbitrary without any restrictions. Alternatively, we can treat $\sqcap, \sqcup$ as the ordinary $\sqcap, \sqcup$ of computability logic (rather than $\sqcap^{\mathfrak{b}}, \sqcup^{\mathfrak{b}}$ as done throughout the present paper), and then forget about any complexity altogether.

In either case, our extensional incompleteness argument goes like this. Consider any system **S** in the style of **PTA** whose proof predicate is decidable[25] and hence the theoremhood predicate recursively enumerable. Assume **S** is sound in the same strong sense as **PTA** — that is, there is an effective procedure that extracts an algorithmic solution (HPM) for the problem represented by any formula F from any **S**-proof of F.

Let then $A(s)$ be the predicate which is true iff:

- s is (the code of) an **S**-proof of some formula of the form $\sqcap x\big(\neg E(x) \sqcup E(x)\big)$, where E is elementary,
- and $E(s)$ is false.

On our assumption of the soundness of **S**, $A(s)$ is a decidable predicate. Namely, it is decided by a procedure that first checks if s is the code of an **S**-proof of some formula of the form $\sqcap x\big(\neg E(x) \sqcup E(x)\big)$, where E is elementary. If not, it rejects. If yes, the procedure extracts from s an HPM $\mathscr{H}$ which solves $\sqcap x\big(\neg E(x) \sqcup E(x)\big)$, and then simulates the play of $\mathscr{H}$ against the environment which, at the very beginning of the play, makes the move s, thus bringing the game down to $\neg E(s) \sqcup E(s)$. If, in this play, $\mathscr{H}$ responds by choosing $\neg E(s)$, then the procedure accepts s; and if $\mathscr{H}$ responds by choosing $E(s)$, then the procedure rejects s. Obviously this procedure indeed decides the predicate A.

Now, assume that **S** is extensionally complete. Since A is decidable, the problem $\sqcap x\big(\neg A(x) \sqcup A(x)\big)$ has an algorithmic solution. So, for some formula F with $F^{\dagger} = \sqcap x\big(\neg A(x) \sqcup A(x)\big)$ and some c, we should have that c is an **S**-proof of F. Obviously F should have the form $\sqcap x\big(\neg E(x) \sqcup E(x)\big)$, where E is an elementary formula with $E^{\dagger}(x) = A(x)$. We are now dealing with the absurdity of $A(c)$ being true iff it is false.

23. ON THE INTENSIONAL STRENGTH OF PTA

Theorem 23.1 *Let X and $\mathbb{L}$ be as in Section 21. Then* **PTA** $\vdash \neg\mathbb{L} \rightarrow X$.

Proof. As established in Section 21, **PTA** $\vdash \overline{X}$. By induction on the complexity of X, details of which we omit, it can also easily be seen that **PTA** $\vdash \overline{X} \rightarrow \neg\mathbb{L} \rightarrow X$. So, by Modus Ponens, **PTA** $\vdash \neg\mathbb{L} \rightarrow X$. ■

Remember that, in Section 21, X was an arbitrary **PTA**-formula assumed to have a polynomial time solution under the standard interpretation †. And $\neg\mathbb{L}$ was a certain true sentence of the language of classical Peano arithmetic. We showed in that section that **PTA** proved a certain formula $\overline{X}$ with $\overline{X}^{\dagger} = X^{\dagger}$. That is, we showed that X was "extensionally provable".

According to our present Theorem 23.1, in order to make X also provable in the intensional sense, all we need is to add to the axioms of **PTA** the true elementary sentence $\neg\mathbb{L}$.

In philosophical terms, the import of Theorem 23.1 is that the culprit of the intensional incompleteness of **PTA** is the (Gödel's) incompleteness of its classical, elementary part. Otherwise, the "nonelementary rest" — the extra-Peano axioms and the PTI rule — of **PTA**, as a bridge from classical arithmetic to ptarithmetic, is as perfect/strong as it could possibly be: it guarantees not only extensional but also intensional provability of every polynomial time computable problem as long as all necessary true elementary formulas are taken care of. This means that if, instead of **PA**, we take the truth arithmetic **Th(N)** (the set of all true sentences of the language of **PA**) as the base arithmetical theory, the corresponding version of **PTA** will be not only extensionally, but also intensionally complete. Unfortunately, however, such a system will no longer be recursively axiomatizable.

So, in order to make **PTA** intensionally stronger, it would be sufficient to add to it new true elementary (classical) sentences, without any need for also adding some nonelementary axioms or rules of inference that deal with nonelementary formulas. Note that this sort of an extension, even if in a language more expressive than that of **PA**, would automatically remain sound and extensionally complete: virtually nothing in this paper relies on the fact that **PA** is not stronger than it really is. Thus, basing applied theories on computability logic allows us to construct ever more expressive and intensionally strong (as well as extensionally so in the case of properly more expressive languages) theories without worrying about how to preserve soundness and extensional completeness. Among the main goals of this paper was to illustrate the scalability of computability logic rather than the virtues of the particular system **PTA** based on it. The latter is in a sense arbitrary, as is **PA** itself: in the role of the classical part of **PTA**, we could have chosen not only any true extension of **PA**, certain weaker-than-**PA** theories as well, for our proof of the extensional completeness of **PTA** does not require the full strength of **PA**. The reason for not having done so is purely "pedagogical": **PA** is the simplest and best known arithmetical theory, and reasoning in it is much more relaxed, easy and safe than in weaker versions. **PTA** is thus the simplest and nicest representative of the wide class of "ptarithmetics", all enjoying the same relevant properties as **PTA** does.

Among the potential applications of ptarithmetic-style systems is using them as formal tools for finding efficient solutions for problems (after developing reasonable theorem-provers, which, at this point, only belongs to the realm of fantasy, of course). One can think of those systems as ideally declarative programming languages, where human "programming" simply means stating the problem/formula whose efficient solution is sought (for systematic usage in the future), and hence the program verification problem is non-existent. Compiling such a "program" means finding a proof, followed by the easy step of translating it into an assembly-language program/solution. This process of compiling may take long but, once compiled, the program runs fast ever after. The stronger such a system is, the better the chances that a solution for a given problem will be found. Of course, what matters in this context is intensional rather than extensional strength. So, perfect strength is not achievable, but we can keep moving ever closer to it.

One may ask why not think of simply using **PA** (or even, say, **ZFC**) instead of **PTA** for the same purposes: after all, **PA** is strong enough to allow us to reason about polynomial time computability. This is true, but **PA** is far from being a reasonable alternative to **PTA**. First of all, as a tool for finding solutions, **PA** is very indirect and hence hopelessly inefficient. Pick any of the basic arithmetical functions of Section 19 and try to generate, in **PA**, a full formal proof of the fact that the function is polynomial-time computable (or even just *express* this fact) to understand the difference. Such a proof would have to proceed by clumsy reasoning about *non-number* objects such as Turing machines and computations, which, only by good luck, happen to be amenable to being understood as numbers through encoding. In contrast, reasoning in **PTA** would be directly about numbers and their properties, without having to encode any foreign beasts and then try to reason about them as if they were just kind and innocent natural numbers. Secondly, even if an unimaginably strong theorem-prover succeeded in finding such a proof, there would be no direct use for it, because from a proof of the existence of a solution we cannot directly extract a solution. Furthermore, even knowing that a given HPM $\mathcal{X}$ solves the problem in *some* polynomial time ξ, would have no practical significance without knowing *what* particular polynomial ξ is, in order to asses whether it is "reasonable" (such as $\mathfrak{b}^2$, $\mathfrak{b}^3$, ...) or takes us beyond the number of nanoseconds in the lifespan of the universe (such as $\mathfrak{b}^{9999999999}$). In order to actually obtain a solution and its polynomial degree, one would need a **constructive** proof, that is, not just a proof that a polynomial

ξ and a ξ-time solution exist, but a proof of the fact that certain particular numbers a and b are (the codes of) a polynomial term ξ and a ξ-time solution $\mathcal{X}$. This means that a theorem-prover would have to be used not just once for a single target formula, but an indefinite (intractably many) number of times, once per each possible pair of values of a, b until the "right" values is encountered. To summarize, **PA** does not provide any reasonable mechanism for handling queries in the style "*find* a polynomial time solution for problem A": in its standard form, **PA** is merely a YES/NO kind of a "device".

The above dark picture can be somewhat brightened by switching from **PA** to Heyting's arithmetic **HA** — the version of **PA** based on intuitionistic logic instead of classical logic, which is known to allow us to directly extract, from a proof of a formula $\exists x F(x)$, a particular value of x for which $F(x)$ is true. But the question is: why intuitionistic logic and not computability logic? Both claim to be "constructive logics", but the constructivistic claims of computability logic have a clear semantical meaning and justification, while intuitionistic logic is essentially an ad hoc invention whose constructivistic claims are mainly based on certain syntactic and hence circular considerations,[26] without being supported by a convincing and complete constructive semantics. And, while **HA** is immune to the second one of the two problems pointed out in the previous paragraph, it still suffers from the first problem. At the same time, as a reasoning tool, **HA** is inferior to **PA**, for it is intensionally weaker and, from the point of view of the philosophy of computability logic, is so for no good reasons. As a simple example, consider the function f defined by "$f(x) = x$ if **PA** is either consistent or inconsistent, and $f(x) = 2x$ otherwise". This is a legitimately defined function, and we all — just as **PA** — know that extensionally it is the same as the identity function $f(x) = x$. Yet, **HA** can be seen to fail to prove — in the intensional sense — its computability.

A natural question to ask is: *Is there a formula X of the language of* **PTA** *whose polynomial time solvability is constructively provable in* **PA** *yet X is not provable in* **PTA**? Remember that, as we agreed just a while ago, by constructive provability of the polynomial time solvability of X in **PA** we mean that, for some particular HPM $\mathcal{X}$ and a particular polynomial (term) ξ, **PA** proves that $\mathcal{X}$ is a ξ-time solution of X. If the answer to this question was negative, then **PA**, while indirect and inefficient, would still have at least *something* to say in its defense when competing with **PTA** as a problem-solving tool. But, as seen from the following theorem, the answer to the question is negative:

Theorem 23.2 *Let X be any formula of the language of* **PTA** *such that* **PA** *constructively proves (in the above sense) the polynomial time solvability of X. Then* **PTA** $\vdash X$.

Proof. Consider any formula X of the language of **PTA**. Assume **PA** constructively proves the polynomial time solvability of X, meaning that, for a certain HPM $\mathcal{X}$ and a certain term ξ (fix them), **PA** proves that $\mathcal{X}$ solves X in time ξ. But this is exactly what the formula $\mathbb{L}$ of Section 21 denies. So, **PA** $\vdash \neg\mathbb{L}$. But, by Theorem 23.1, we also have **PTA** $\vdash \neg\mathbb{L} \rightarrow X$. Consequently, **PTA** $\vdash X$. ■

An import of the above theorem is that, if we tried to add to **PTA** some new nonelementary axioms in order to achieve a properly greater intensional strength, the fact that such axioms are computable in time ξ for some particular polynomial ξ would have to be unprovable in **PA**, and hence would have to be "very nontrivial". The same applies to attempts to extend **PTA** through some new rules of inference.

24. GIVE CAESAR WHAT BELONGS TO CAESAR

The idea of exploring versions of Peano arithmetic motivated by and related to various complexity-theoretic considerations and concepts is not new. In this connection one should mention a solid amount of work on studying *bounded arithmetics*, with the usage of the usual quantifiers $\forall, \exists$ of **PA** restricted to forms such as $\forall x\big(x \leq \tau \rightarrow F(x)\big)$ and $\exists x\big(x \leq \tau \wedge F(x)\big)$, where τ is a term not containing x. Parikh (1971) was apparently the first to tackle bounded quantifiers in arithmetic. A systematic study of bounded arithmetics and their connections to complexity theory was initiated in the seminal work (1986a) by Buss. Hajek and Pudlak (1993) give an extensive survey of this area. The main relevant results in it can be summarized by saying that, by appropriately weakening the induction axiom of **PA** and then further restricting it to bounded formulas of certain forms, and correspondingly readjusting the nonlogical vocabulary and axioms of **PA**, certain

soundness and completeness results for the system(s) **S** thus generated can be achieved. Such soundness results typically read like "If **S** proves a formula of the form $\forall x \exists y F(x,y)$, where F satisfies such and such constraints, then there is function of such and such computational complexity which, for each a, returns a b with $F(a,b)$". And completeness results typically read like "For any function f of such and such computational complexity, there is an **S**-provable formula of the form $\forall x \exists y F(x,y)$ such that, for any a and b, $F(a,b)$ is true iff $b = f(a)$".

Among the characteristics that make our approach very different from the above, one should point out that it *extends* rather than *restricts* the language and the deductive power of **PA**. Restricting the language and power of **PA** in the style of the approach of bounded arithmetics throws out the baby with the bath water. Not only does it expel from the system many complexity-theoretically unsound yet otherwise meaningful and useful theorems, but it apparently also reduces — even if only in the intensional rather than extensional sense — the class of complexity-theoretically correct provable principles. This is a necessary sacrifice in such cases, related to the inability of the underlying classical logic to clearly differentiate between constructive ($\sqcap, \sqcup, \bigsqcap, \bigsqcup$) and "ordinary", non-constructive versions ($\wedge, \vee, \forall, \exists$) of operators. Classical logic has never been meant to be a constructive logic, let alone a logic of efficient computations. Hence an attempt to still make it work as a logic of computability or efficient computability cannot go forward without taking a toll, and results such as the above-mentioned soundness can only be partial.

The problem of the partiality of the soundness results has been partially overcome in Buss (1986b) through basing bounded arithmetic on intuitionistic logic instead of classical logic. In this case, soundness extends to all formulas of the form $\forall x \exists y F(x,y)$, without the "$F$ satisfies such and such constraints" condition (the reason why we still consider this sort of soundness partial is that it is still limited to formulas of the form $\forall x \exists y F(x,y)$, even if for arbitrary Fs; similarly, completeness is partial because it is limited only to functions which, for us, are only special cases of computational problems). However, for reasons pointed out in the previous section, switching to intuitionistic logic signifies throwing out even more of the "baby" from the bath tub, further decreasing the intensional strength of the theory. In any case, whether being based on classical or intuitionistic logic, bounded arithmetics do not offer the flexibility of being amenable to strenghthening without loss of soundness, and are hence "inherently weak" theories.

In contrast, computability logic avoids all these troubles and sacrifices by giving Caesar what belongs to Caesar, and God what belongs to God. As we had a chance to see throughout this paper, classical ($\wedge, \vee, \forall, \exists$) and constructive ($\sqcap, \sqcup, \bigsqcap, \bigsqcup$) logical constructs can peacefully coexist and complement each other in one natural system that seamlessly extends the classical, constructive, resource- and complexity-conscious visions and concepts, and does so not by mechanically putting things together, but rather on the basis of one natural, all-unifying game semantics. Unlike most other approaches where only a few, special-form expressions (if any) have clear computational interpretations, in our case every formula is a meaningful computational problem. Further, we can capture not only computational problems in the traditional sense, but also problems in the more general — interactive — sense. That is, ptarithmetic or computability-logic-based theories in general, are by an order of magnitude more expressive and deductively powerful than the classical-logic-based **PA**, let alone the far more limited bounded arithmetics.

Classical logic and classical arithmetic, so close to the hearts and minds of us all, do not at all need to be rejected or tampered with (as done in Heyting's arithmetic or bounded arithmetic) in order to achieve constructive heights. Just the opposite, they can be put in faithful and useful service to this noble goal. Our heavy reliance on reasoning in **PA** throughout this paper is an eloquent illustration of this.

25. THOUGHTS FOR THE FUTURE

The author wishes to hope that the present work is only the beginning of a longer and more in-depth line of research on exploring computability-logic-based theories (arithmetic in particular) with complexity-conscious semantics. There is an ocean of problems to tackle in this direction.

First of all, it should be remembered that the particular language of ptarithmetic employed in this paper is only a modest fragment of the otherwise inordinately expressive and, in fact, open-ended formal-

ism of computability logic. Attempting to extend the present results to more expressive versions of ptarithmetic is one thing that can be done in the future. Perhaps a good starting point would be considering the language employed in Japaridze (2010) which, in addition to the present connectives, has the operator $\bullet\!\!-$, with $A \bullet\!\!- B$ being the problem of reducing B to A where any finite number of reusages of A is allowed. From a more ambitious perspective, a development along these lines may yield the discovery of a series of new, complexity-conscious operators that are interesting and useful in the context of interactive computational complexity, while not quite so useful in the ordinary context of computability-in-principle.

Another direction to continue the work started in this paper would be to try to consider complexity concepts other than polynomial time complexity. Who knows, maybe these studies can eventually lead to the discovery of substantially new, not-yet tried weapons for attacking the famous and notorious open problems in complexity theory. Two of the most immediate candidates for exploration are logarithmic space and polynomial space computabilities. While the precise meaning of logarithmic space computability in our interactive context has yet to be elaborated, a definition of polynomial space computability comes almost for free. It can be defined exactly as we defined polynomial time computability in Section 7, only, instead of counting the number of steps taken by the machine ($\top$'s time, to be more precise), we should count the number of cells visited by the head of the work tape. What, if any, variations of the PTI rule (and perhaps also the nonlogical axioms) would yield systems of **psarithmetic** ("polynomial space arithmetic") or **larithmetic** ("logarithmic space arithmetic"), sound and complete with respect to polynomial space or logarithmic space computability in the same sense as **PTA** is sound and complete with respect to polynomial time computability?

Acknowledgement: Parts of this material are reprinted from "Introduction to clarithmetic I", Information and Computation, vol. 209 (2011), pp. 1312-1354, with permission from Elsevier.

Notes

[1]The paper by Xu & Liu (2009) (in Chinese) is apparently another exception, focused on applications of CL in AI.

[2]For simplicity, here we treat "Composite" as the complement of "Prime", even though, strictly speaking, this is not quite so: the numbers 0 and 1 are neither prime nor composite. Writing "Nonprime" instead of "Composite" would easily correct this minor inaccuracy.

[3]According to our conventions, such a natural number i is identified with its binary representation. The same applies to the other clauses of this definition.

[4]It is important to note that, unlike the case with the choice quantifiers, here we are not imposing any restrictions on the size of such a constant.

[5]The concept of an interpretation in CL is usually more general than the present one. Interpretations in our present sense are called **perfect**. But here we omit the word "perfect" as we do not consider any nonperfect interpretations, anyway.

[6]That is, $=^*$ is a congruence relation. More commonly classical logic simply treats $=$ as the identity predicate. That treatment of $=$, however, is known to be equivalent — in every respect relevant for us — to our present one. Namely, the latter turns into the former by seeing any two $=^*$-equivalent constants as two different *names* of the same *object* of the universe, as "Evening Star" and "Morning Star" are.

[7]Remember Convention 4.4, according to which $\sqcap$ means $\sqcap^{\mathfrak{b}}$ and $\sqcup$ means $\sqcup^{\mathfrak{b}}$.

[8]Since we need to construct $\mathcal{M}$ no matter whether those assumptions are true or not, we can let $\mathcal{M}$ simply stop making any moves as soon as it detects some illegal behavior.

[9]Here and later, of course, an implicit stipulation is that the position spelled on the imaginary run tape of the machine ($\mathcal{M}_i$ in the present case) that made the move is also correspondingly updated (in the present case, by appending $\top\alpha$ to it).

[10]Remember Convention 3.4.

[11]In fact, an essentially the same logic, under the name **L**, was already known as early as in Japaridze (2002), where it emerged in the related yet quite different context of the *Logic of Tasks*.

[12]To ensure that Convention 8.1 is respected, here we can safely assume that, if E is obtained by $\sqcup$-Choose and this rule (in the bottom-up view) introduced a fresh variable s, then s has no (bound) occurrences in $G^{\heartsuit}$, or otherwise rename s into some neutral variable.

[13]Only Axiom 7 is a scheme in the proper sense. Axioms 1-6 are "schemes" only in the sense that x and y are metavariables for variables rather than particular variables. These axioms can be painlessly turned into particular formulas by fixing some particular variables in the roles of x and y. But why bother.

[14]Warning: here we do not follow the standard convention, according to which $|0|$ is considered to be 0 rather than 1.

[15]Do you see or feel a possible application of MP, or TR, or Match behind this informal phrase?

[16]The condition $s<\tau$ would not automatically imply $|s'|\leq\mathfrak{b}$: in pathological cases where $\mathfrak{b}$ is "very small", it may happen that the first condition holds but the second condition is still violated.

[17]Those familiar with bounded arithmetics will notice a resemblance between BPI and the version of induction axiom known as PIND (Buss 1986a; Hajek & Pudlak 1993). An important difference, however, is that PIND assumes s to be (actually or potentially) $\forall$-bound, while in our case s, as a free variable, can be seen as $\sqcap$-bound but by no means as $\forall$-bound.

[18]In view of Convention 8.1, it is implicitly assumed here that τ does not contain z, for otherwise the formula would have both bound and free occurrences of z. Similarly, since z is quantified, it cannot be $\mathfrak{b}$.

[19]Strictly speaking, Axiom 13 or Lemma 13.2 will also be needed here to be sure that, if the size of θ_1 exceeds $\mathfrak{b}$, then $\theta_1\neq\mathbf{0}$.

[20]For instance, the rule of Lemma 20.1 can be easily strengthened by weakening the consequent of its right premise to the more PTI-style $E(t')\sqcap(F(t')\wedge E(t))$, and/or strengthening the antecedent of that premise by adding the conjuncts R and $w\leq t<\tau$ (on the additional condition that t does not occur in R).

[21]Only considering nonzero times in this context.

[22]The third $\wedge$-conjunct of $\mathbb{F}(z,t)$ was already used in the previous paragraph. But there is no resource conflict here, as we have a choice (rather than parallel) conjunction between the problems whose solutions are described in the present and the previous paragraphs, so that only one of them will actually have to be solved.

[23]Strictly speaking, more than one move can bring E to the same $\sqcup$-deletion (e.g., think of the case $E=Y\sqcup Y$). But this is not a serious problem, and is easily taken care of by assuming that the list $G_1,\ldots,G_m$ has repetitions if necessary so that, for each move that turns E into one of its $\sqcup$-deletions, the list contains a separate copy of the corresponding $\sqcup$-deletion.

[24]Remember that $\mathcal{X}$ runs in time $\xi(\mathfrak{b})$. By definition, this means that $\top$'s time in any play is *less* than $\xi(\mathfrak{b})$. Hence, the term $\xi(\mathfrak{b})$ cannot be $\mathbf{0}$, or $\mathbf{0}\times\mathfrak{b}$, or anything else that always evaluates to 0. Therefore, of course, $\mathbf{PA}\vdash\xi(\mathfrak{b})\geq\mathfrak{b}$.

[25]**PTA** can easily be readjusted to satisfy this condition by requiring that each logical axiom in a **PTA**-proof be supplemented with a proof of that axiom in some known (fixed) sound and complete recursively axiomatized calculus for classical logic.

[26]What creates circularity is the common-sense fact that syntax is merely to serve a meaningful semantics, rather than vice versa. It is hard not to remember the following words from Japaridze (2009a) here: "The reason for the failure of $P\sqcup\neg P$ in computability logic is not that this principle . . . is not included in its axioms. Rather, the failure of this principle is exactly the reason why this principle, or anything else entailing it, would not be among the axioms of a sound system for computability logic".

References

Abramsky, S. & Jagadeesan, R. 1994. 'Games and full completeness for multiplicative linear logic'. *Journal of Symbolic Logic* 59: 543–574.

Babai, L. & Shlomo, M. 1988. 'Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes'. *Journal of Computer System Sciences* 36: 254–276.

Blass, A. 1972. 'Degrees of indeterminacy of games'. *Fundamenta Mathematicae* 77: 151–166.

——. 1992. 'A game semantics for linear logic'. *Annals of Pure and Applied Logic* 56: 183–220.

Buss, S. 1986a. *Bounded arithmetic*. Ph.D. thesis, Bibliopolis. (revised version of PhD thesis).
——. 1986b. 'The polynomial hierarchy and intuitionistic bounded arithmetic'. *Lecture Notes in Computer Science* 223: 77–103.
Chandra, A., Kozen, D. & Stockmeyer, L. 1981. 'Alternation'. *Journal of the ACM* 28: 114–133.
Goldwasser, S., Micali, S. & Rackoff, C. 1989. 'The knowledge complexity of interactive proof systems'. *SIAM Journal on Computing* 18: 186–208.
Hajek, P. & Pudlak, P. 1993. *Metamathematics of First-Order Arithmetic*. Springer.
Japaridze, G. 2002. 'The logic of tasks'. *Annals of Pure and Applied Logic* 117: 263–295.
——. 2003. 'Introduction to computability logic'. *Annals of Pure and Applied Logic* 123: 1–99.
——. 2006a. 'Introduction to cirquent calculus and abstract resource semantics'. *Journal of Logic and Computation* 16: 489–532.
——. 2006b. 'Computability logic: a formal theory of interaction'. In D. Goldin, S. Smolka & P. Wegner (eds.) 'Interactive Computation: The New Paradigm', 183–223. Springer Verlag.
——. 2006c. 'From truth to computability I'. *Theoretical Computer Science* 357: 100–135.
——. 2006d. 'Propositional computability logic I'. *ACM Transactions on Computational Logic* 7: 302–330.
——. 2006e. 'Propositional computability logic II'. *ACM Transactions on Computational Logic* 7: 331–362.
——. 2007a. 'The logic of interactive Turing reduction'. *Journal of Symbolic Logic* 72: 243–276.
——. 2007b. 'From truth to computability II'. *Theoretical Computer Science* 379: 20–52.
——. 2007c. 'The intuitionistic fragment of computability logic at the propositional level'. *Annals of Pure and Applied Logic* 147: 187–227.
——. 2007d. 'Intuitionistic computability logic'. *Acta Cybernetica* 18: 77–113.
——. 2008a. 'Cirquent calculus deepened'. *Journal of Logic and Computation* 18: 983–1028.
——. 2008b. 'Sequential operators in computability logic'. *Information and Computation* 206: 1443–1475.
——. 2009a. 'In the beginning was game semantics'. In O. Majer, A.-V. Pietarinen & T. Tulenheimo (eds.) 'Games: Unifying Logic and Philosophy', 249–350. Springer.
——. 2009b. 'Many concepts and two logics of algorithmic reduction'. *Studia Logica* 91: 1–24.
——. 2010. 'Towards applied theories based on computability logic'. *Journal of Symbolic Logic* 565–601.
——. 2011. 'Introduction to clarithmetic I'. *Information and Computation* 1312–1354.
——. forth. 'Introduction to clarithmetic III'. Annals of Pure and Applied Logic, to appear.
——. unpublished. 'Introduction to clarithmetic II'. Manuscript at http://arxiv.org/abs/1004.3236.
Mezhirov, I. & Vereshchagin, N. 2010. 'On abstract resource semantics and computability logic'. *Journal of Computer and System Sciences* 76: 356–372.
Parikh, R. 1971. 'Existence and feasibility in arithmetic'. *Journal of Symbolic Logic* 36: 494–508.
Xu, W. & Liu, S. 2009. 'Knowledge representation and reasoning based on computability logic'. *Journal of Jilin University* 47: 1230–1236.