Transitive Closure and the Mechanization of Mathematics

Arnon Avron
School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
*(*aa@math.tau.ac.il*)*

Abstract. We argue that the concept of transitive closure is the key for understanding finitary inductive definitions and reasoning, and we provide evidence for the thesis that logics which are based on it (in which induction is a logical rule) are the right *logical* framework for the formalization and mechanization of Mathematics. We investigate the expressive power of languages with the most basic transitive closure operation TC. We show that with TC one can define all recursive predicates and functions from 0, the successor function and addition, yet with TC alone addition is not definable from 0 and the successor function. However, in the presence of a pairing function, TC does suffice for having all types of finitary inductive definitions of relations and functions. This result is used for presenting a simple version of Feferman's framework FS_0, demonstrating that TC-logics provide in general an excellent framework for mechanizing formal systems. An interesting side effect of these results is a simple characterization of recursive enumerability and a new, concise version of Church thesis. We end with a use of TC for a formalization of Set Theory which is based on purely syntactical considerations, and reflects real mathematical practice.

1. Introduction

There is an old answer to the question: "What is a formal system?" which was supplied by Post [22] and Smullyan [23]. They took the classes of strings closed under sets of inductive definitions as the general notion (in effect, the theory of recursively enumerable classes). Their approach was however believed to be completely unusable in practice, and so it has first been ignored as a possible basis for a practical framework. However, in [9] Feferman gave a new proposal in the same vein, as a competitor to the type theory school of frameworks, like the Edinburgh LF (see e.g. [14, 21]). Feferman's FS_0 follows the approach of Post and Smullyan: basically it identifies the notions of a formal system and of an r.e. set. There is enormous improvement, however, over the Post/Smullyan approach by taking Lisp's S-expressions (rather than strings) as the basic data structure, and by using a very flexible form of inductive definitions. FS_0 is accordingly a version of Pure Lisp, supplemented with facilities for defining recursively enumerable classes and with induction over such classes. Since syntactic categories connected with a formal system L are almost always defined inductively, it is usually straightforward to make an encoding of a logic L in FS_0

as classes of S-expressions. The induction principles of FS_0 can then be used for doing within FS_0 object-level theorem proving in L, and also meta-level reasoning concerning L. That this approach is indeed potentially practical from an implementation point of view has been demonstrated by the works of Matthews [20, 17, 18]). Nevertheless, mechanical theorem proving (even interactively) in FS_0 is still quite difficult, since FS_0 is a first-order *theory*, in which induction is provided as an *axiom*, not as a rule. Moreover: the language of FS_0 is also not so convenient to work with. It is a rather complicated 3-sorted language, including S-expressions, functions on S-expressions, functionals for combining functions, classes and class forming operations. [1]

The main goal of this paper is to show that Feferman's framework can be simplified a lot by turning it into a basic *logic* (rather than a complicated theory within a logic) in a relatively simple language. Since induction becomes a logical rule of inference in this logic, the resulting system (so we believe) should be easier to use, and can really serve as a framework for formalizing mathematical systems and for reasoning about them. The key idea is that instead of using first order logic as an underlying logic we should use an extension of it with an operation for forming transitive closure of binary relations.

A second goal we have, strongly related to the first, is to provide evidence for a broader thesis, according to which a logic of this sort is actually the right *logical* framework for the formalization and mechanization of mathematics. It should be clear that first-order logic is too week for this task. Thus the fact that it fails even to provide a categorical characterization of the natural numbers means that this logic is incapable of telling the whole story that human Logic has to tell. Moreover: even a mathematical theory which does have a first-order formulation often has basic theorems which are not really theorems of that first-order theory, but (officially) theorems *about* it. A trivial example (from [17])[2] is the identity $(ab)^n = a^n b^n$, which is valid for every integer n in every commutative group. This identity is certainly a theorem of commutative group theory (proved by mathematical *induction*), but it is *not* a theorem of first-order commutative group theory!

While first-order logic is too weak, second-order logic is frequently too strong (and much more difficult to mechanically work with). Moreover: it is based on ontological commitments which are not universally accepted. Thus it is counter-intuitive that in order to understand the notion of a natural number (which every child does) one needs to understand also the notion of an arbitrary set of natural numbers (which

[1] A more detailed description of FS_0 is given in subsection 5.1.

[2] Other, more interesting examples are given in subsection 2.2.

most people do not), but this precisely is what the use of a second-order language for characterizing the natural numbers amounts to. It could in fact be claimed with justice that second-order "logic" is actually not a *logic* at all, but set theory in disguise.

The use of of a logic with a transitive closure operation TC seems to me as the correct intermediate level. The fundamental induction principle is a *logical* principle in such a logic, and usually any obvious consequence of a given first-order theory which cannot be formulated and proved in that theory can easily be formulated and proved in a corresponding logic with TC using this principle. On the other hand the construction of the transitive closure of a given relation does not involve the creation of new, abstract objects (like in second order logic), but rather of new *relations* among existing ones [3]. Moreover: the use of TC is *natural* (and so user-friendly), since the correct use of it is a part of everybody's logic. Here are two non-mathematical examples:

- I am a descendant of Jacob. Therefore my son is also a descendant of Jacob. Moreover: The descendants of my son will all be descendants of Jacob.

- If you have sexual relations with someone who had had sexual relations with someone who had had sexual relations with ... someone who had been a HIV carrier, then you might become one too.[4]

Everyone can (and does) understand the inferences made in these two examples. One does not need to be trained in mathematics or even to understand the natural numbers for that. This means that they are based on general (though not first-order) *logical* principles [5]. Now what is involved in the two examples (and in a lot of other examples of this sort[6]) are the following two factors:

1. The construction of a certain new binary relation between objects from a given one. This relation is exactly what mathematicians call "the transitive closure" of the old one.

[3] Note that the use of relations which are not objects can anyhow never been dispensed with. Even in set theory, where relations are usually taken as objects, the most important ones ($\in$, $=$ and $\subseteq$) are *not*!

[4] Taken without permission from a T.V. campaign for safe sex.

[5] I am convinced that most people will indeed classify the conclusions in the examples above as *logical* consequences of their assumptions!

[6] To give just one more example from computer science: in Page 7 of [16] the authors write: "Some appeal to second-order logic appears necessary here because transitive closure is not first-order definable". In fact transitive closure is the only "second-order" feature they need.

2. The use of some induction principle for inferring facts about objects which are related by the new relation. Such induction principle(s) should be the core of the logic of the transitive closure operation.

These consideration lead us to the conviction that the ability to define the transitive closure of any given relation and make appropriate inferences concerning it is essential for any computerized reasoning system, and especially one which is designed for doing mathematics. We further believe that systems which are based on extending first-order logic with TC should suffice for most (if not all) concrete, applicable mathematics, and that such systems should be better suited for automated reasoning than systems which are based on higher order logics. The results below provide some (quite partial) evidence for these beliefs.

The structure of the paper is as follows. In section 2 we review first the basic definitions, and then provide examples of applications in two major mathematical disciplines: geometry and arithmetics. The main result of this section is that using TC, all recursive functions are definable from $0, S$ and $+$ (where S denotes the successor function). In section 3 we present induction as a logical rule of languages with TC. We use for this a Gentzen-type format, generalizing a particular case which has been used by Gentzen. In section 4 we study the expressive power of languages with TC. We show that TC alone is not sufficient for defining addition from 0 and the successor function. However, in the presence of a pairing function, it does suffice for having all types of finitary inductive definitions of relations and functions. This result is used in section 5.1 for presenting a simple version of Feferman's framework FS_0, demonstrating that TC-logics provide in general an excellent framework for mechanizing formal systems. An interesting side effect of the results in that section is a simple characterization of recursive enumerability and a new, concise version of Church thesis. Finally, in section 5.2 we use TC for a formalization of set theory which is based on purely syntactical considerations, and reflects real mathematical practice.

One more remark before we start: a great deal of attention has already been given in the past to logics with a transitive closure operation in the context of finite model theory (FMT – see [8]), as well as other branches of computer science (like databases and complexity). However, in almost all cases the research has focused on their applications to *finite* structures. Our own interest, in contrast, is in logics which are meaningful and applicable to *every* structure. In fact, infinite structures are by far more interesting and important from the point of view which interests us: the prospect of formalizing and mechanizing mathematics (especially concrete mathematics). Important and useful

as it certainly is, most of the work that was done in FMT (and related areas) on the subject is therefore almost irrelevant to the purposes of this paper. Moreover: most of this research has been dedicated to model theory (with an emphasis on fixed-point logics), while for formalizing mathematics we need useful *proof systems*. I believe that from this point of view logics which are based on TC will prove superior to fixed-point logics, and they are certainly more natural and better suited than these logics to formalize the way mathematics is actually *done*!

2. Logics with a Transitive Closure Operation

2.1. The Language and its Semantics

The language and semantics of the basic extension of first order logic with a transitive closure operation are defined as follows (see, e.g., [15, 13, 12]):

DEFINITION 1. *Let σ be a signature for a first-order language with equality. The language $L^1_{TC}(\sigma)$ is defined like the usual first-order language which is based on σ, but with the addition of the following clause:*

- *If φ is a formula, x, y are distinct variables, and t, s are terms, then $(TC_{x,y}\varphi)(t,s)$ is a formula. In this formula all the free occurrences of x and y in φ become bound.*

The semantics of the new type of formulas is defined as follows. Given a structure $\mathcal{D}$ for σ with domain D and an assignment v in D for the variables of the language, define:

- *$\mathcal{D}, v \models (TC_{x,y}\varphi)(t,s)$ iff there exist $a_0, a_1, \ldots, a_n \in D$ $(n \geq 1)$ such that $a_0 = v(t)$, $a_n = v(s)$, and $\mathcal{D}, v(x := a_i,\ y := a_{i+1}) \models \varphi$ for $i = 0, 1, \ldots, n-1$.*

NOTE 1. Given a formula φ, $(TC_{x,y}\varphi)$ is a new binary *predicate* symbol (in standard first-order languages only primitive predicate symbols are available!).

NOTE 2. In L^1_{TC} we really need only the propositional connectives and the TC operator, since the existential quantifier can be *defined* as follows:

$$\exists x\varphi =_{Df} \Big(TC_{u,v}\big(\varphi(u/x) \vee \varphi(v/x)\big)\Big)(t,s)$$

where t and s are terms. Any two terms would do, but in order for $\exists x\varphi$ to have less free variables than φ (in particular, in order for $\exists x\varphi$ to be

a sentence in case x is the only free variable of φ), t and s should be (not necessarily distinct) closed terms. This is possible, of course, only if the signature σ has at least one constant.

On the other hand, in the presence of the existential quantifier it is possible to provide an alternative (simpler?) version of TC. Since $\varphi(t/x)$ is equivalent to $\exists x.x = t \wedge \varphi$, one can replace the somewhat cumbersome $(TC_{x,y}\varphi)(t,s)$ by the simpler $TC^*_{x,y}\varphi$, where the meaning of the latter is identical to that of $(TC_{x,y}\varphi)(x,y)$ (and so x and y are *free* in this formula). Note however that the resulting language is strange in that it might not be possible to substitute in it terms (not even closed ones) for free variables (although the truth-value of a formula in a structure still depends on the values which are assigned there to its free variables).

NOTE 3. The reflexive transitive closure RTC can be defined as:

$$RTC_{xy}(t,s) \equiv_{Df} (t = s) \vee (TC_{xy}\varphi)(t,s) \ .$$

It is possible in fact to take RTC as primitive, since $(TC_{xy}\varphi)(t,s)$ is logically equivalent to:

$$\varphi(t/x, s/y) \vee \exists z\big(z \neq t \wedge (RTC_{x,y}\varphi)(t,z) \wedge (RTC_{x,y}\varphi)(z,s)\big) \ .$$

Note however that the use of negation is needed for defining TC in terms of RTC, while it is not needed for defining RTC in terms of TC (This difference is crucial for the validity of Theorems 3 and 5 below!). Hence it is better to choose TC as primitive.

2.2. An Example: Applications in Euclidean Geometry

As is well known, elementary Euclidean geometry (EUG) has been given by Tarski a complete (and so decidable) first-order axiomatization (see e.g. [24]). However, some of the most interesting theorems of Euclidean geometry cannot be formulated in the language of EUG. Examples are the negative results concerning construction problems (like squaring the circle), and the Mascheroni-Mohr theorem according to which every point in the plane which can be constructed using a ruler and a compass (given a finite set of points) can be constructed using only a compass[7]. By Theorem 3 below, these theorems can be formulated if TC and a pairing function are added to the language of EUG. As for their proofs — induction plays of course a crucial role in them. It is

[7] An attempt of automatizing my new proof of this theorem in [2, 3] has been one of my major motivations for investigating $L^1_{TC}(\sigma)$ (see [4]).

interesting to note however that at least for Mascheroni-Mohr theorem an extra axiom is needed as well: Archimedes' axiom. This axiom says that given two segments a and b, there is a natural number n such that b is congruent to a partial segment of na. Archimedes' axiom cannot be formulated in the first-order language of EUG. It can however easily and naturally be formulated using TC. Here, e.g., is a formulation of its counterpart in the standard signature of ordered fields (which is somewhat shorter than its formulation in the language of geometry):

$$\forall a \forall b \Big(b > 0 \rightarrow \exists z \big(z > a \wedge (TC_{x,y} x = y + b)(z, 0) \big) \Big) \ .$$

Note that the natural numbers are not even mentioned in this formulation! (Of course, they are *definable* in this signature by the formula $N(n) \equiv_{Df} (TC_{x,y} x = y + 1)(n, 0)$).

2.3. Transitive Closure and the Natural Numbers

In $L^1_{TC}(\langle 0, S, + \rangle)$ one can define:

$$t < s \equiv_{Df} \ \big(TC_{x,y} y = S(x) \big)(t, s) \ .$$

The following finite set of axioms is then *categorical*, with $\mathbb{N}$ as the unique model:

N1.	$\forall x \big(S(x) \neq 0 \big)$
N2.	$\forall x \forall y \big(S(x) = S(y) \Rightarrow x = y \big)$
N3.	$\forall x (x = 0 \vee 0 < x)$
N4.	$\forall x (x + 0 = x)$
N5.	$\forall x \big(x + S(y) = S(x + y) \big)$

Note that this system is of course *not* categorical if we define $t < s$ as $\exists z . S(z) + t = s$!

THEOREM 1. *All recursive functions are definable in* $L^1_{TC}(\langle 0, S, + \rangle)$.[8].

Proof: Obviously, it is sufficient to show that multiplication is definable in $L^1_{TC}(\langle 0, S, + \rangle)$. For this note that the division relation (defined by: $x \mid y \equiv_{Df} \exists z (x * z = y)$) is definable in $L^1_{TC}(\langle 0, S, + \rangle)$, since $\mathbb{N} \models x \mid y \Leftrightarrow (TC_{a,b} a + x = b)(0, y)$. Next, it is obvious that $x = y^2$ is equivalent in $\mathbb{N}$ to

$$y \mid (x + y) \wedge S(y) \mid (x + y) \wedge \forall z (z < x + y \wedge y \mid z \Rightarrow \neg \big(S(y) \mid z \big) \ .$$

[8] By an $n-$ary function being definable we mean of course that its graph is definable (as an $(n+1)-$ary relation).

Finally, $x = y \cdot z$ is equivalent in $\mathbb{N}$ to

$$\exists u \exists v \exists w \Big(u = y^2 \wedge v = z^2 \wedge w = (y+z)^2 \wedge w = \big((u+v)+x\big) + x \Big)$$

COROLLARY 1. *The set of formulas of $L^1_{TC}(\langle 0, S, + \rangle)$ which are valid in $\mathbb{N}$ is not arithmetical.*

COROLLARY 2. *The set of logically valid formulas of $L^1_{TC}(\langle 0, S, + \rangle)$ is not arithmetical.*

Proof: A formula φ of $L^1_{TC}(\langle 0, S, + \rangle)$ is valid in $\mathbb{N}$ iff the formula $N1 \wedge \cdots \wedge N5 \rightarrow \varphi$ is logically valid.

NOTE 4. For a categorical characterization of the natural numbers $L^1_{TC}(\langle 0, S \rangle)$ and axioms N1-N3 already suffice. Unfortunately, the expressive power of this language is too weak (See Theorem 2 below). Note, however, that $L^1_{TC}(\langle 0, S \rangle)$ does suffice for our definition of $<$.

3. Reasoning with TC: Induction as a Logical Rule

Corollary 2 means that no complete and sound proof system for the logically valid formulas of L^1_{TC} is possible. Instead one should look for a computationally efficient sound proof system which will be strong enough for most (or even all) related mathematical needs. The most important valid *logical* rule of $L^1_{TC}(\sigma)$ which such a system should include is *induction*. Using a Gentzen-type format, a general form of this principle can be formulated as follows:

$$\frac{\Gamma, \psi, \varphi \Rightarrow \Delta, \psi(y/x)}{\Gamma, \psi(s/x), (TC_{x,y}\varphi)(s,t) \Rightarrow \Delta, \psi(t/x)}$$

where x and y should not occur free in Γ, Δ, and y should not occur free in ψ.

To see the connection between this rule and ordinary induction, assume that σ is a signature which contains 0 and S, and take φ to be $y = S(x)$. Substituting 0 for s we get:

$$\frac{\Gamma, \psi, y = S(x) \Rightarrow \Delta, \psi(y/x)}{\Gamma, \psi(0/x), \big(TC_{x,y} y = S(x)\big)(0,t) \Rightarrow \Delta, \psi(t/x)}$$

Using first order rules for $=$ and the definition of $<$ in $L^1_{TC}(\langle 0, S \rangle)$, this is equivalent to:

$$\frac{\Gamma, \psi \Rightarrow \Delta, \psi\big(S(x)/x\big)}{\Gamma, \psi(0/x), 0 < t \Rightarrow \Delta, \psi(t/x)}$$

Since obviously $\Gamma, \psi(0/x),\ 0 = t \Rightarrow \Delta, \psi(t/x)$ is a valid sequent, we get

$$\frac{\Gamma, \psi \Rightarrow \Delta, \psi\big(S(x)/x\big)}{\Gamma, \psi(0/x), 0 = t \vee 0 < t \Rightarrow \Delta, \psi(t/x)}$$

Using N3 from subsection 2.3 , this implies the validity of the following in the context of any system in which N3 is valid:

$$\frac{\Gamma, \psi \Rightarrow \Delta, \psi\big(S(x)/x\big)}{\Gamma, \psi(0/x) \Rightarrow \Delta, \psi(t/x)}$$

This is exactly the form given to the induction rule by Gentzen in the classical [11] (in which the consistency of PA is proved).

The induction rule is an introduction rule for TC on the left hand side of a sequent. Two obvious rules for introducing it on the right hand side are:

$$\frac{\Gamma \Rightarrow \Delta, \varphi(t/x, s/y)}{\Gamma \Rightarrow \Delta, (TC_{x,y}\varphi)(t, s)}$$

$$\frac{\Gamma \Rightarrow \Delta, (TC_{x,y}\varphi)(r, s) \qquad \Gamma \Rightarrow \Delta, (TC_{x,y}\varphi)(s, t)}{\Gamma \Rightarrow \Delta, (TC_{x,y}\varphi)(r, t)}$$

A major research task here is to find out what other rules (if any) should be added in order to make the system "complete" in some reasonable sense.

4. The Expressive Power and Inductive Definitions

4.1. The Need to Use Pairs

We have seen that induction as a method of proof is a part of the logic of $L^1_{TC}(\sigma)$. However, induction is used in Mathematics also as a tool for defining new concepts and relations (including functions, which we take here as a special type of relations). For this $L^1_{TC}(\sigma)$ is in general too weak:

THEOREM 2. *$+$ is not definable in $L^1_{TC}(\langle 0, S\rangle)$.*

Proof: By a well-known result of Büchi ([7]. See also [6], p. 615]), $S1S$, the (monadic) second order theory of the successor function, is

decidable. Now $L^1_{TC}(\langle 0, S\rangle)$ is interpretable in $S1S$, since $(TC_{x,y}\varphi)(t,s)$ is equivalent to

$$\forall Z\Big\{\big[\forall u\big(\varphi(t/x, u/y) \to u \in Z\big) \wedge \forall x \forall y(x \in Z \wedge \varphi \to y \in Z)\big] \to s \in Z\Big\}$$

(where Z is a second order variable). This implies the decidability of the set of formulas of $L^1_{TC}(\langle 0, S\rangle)$ which are valid in $\mathbb{N}$. The theorem follows therefore from Corollary 1.

One possible solution to the problem caused by Theorem 2 is to use stronger transitive closure operations. [13], e.g., considers the languages $L^n_{TC}(\sigma)$, in which $(TC^k_{x_1,\ldots,x_k,y_1,\ldots,y_k}\varphi)(\vec{t},\vec{s})$ is a formula whenever φ is a formula, $x_1,\ldots,x_k,y_1,\ldots,y_k$ $(k \leq n)$ are $2k$ distinct variables, and $\vec{t},\vec{s}$ are k-vectors of terms. The semantics of $(TC^k_{\vec{x},\vec{y}}\varphi)(\vec{t},\vec{s})$ is defined as in the case $k = 1$, only this time we need to refer to vectors of length k of terms or variables. Now in $L^2_{TC}(\langle 0, S\rangle)$ addition (and so every recursive function) *is* definable, since it is easy to see that

$$\mathbb{N} \models x = y + z \Leftrightarrow \big(RTC^2_{x_1,x_2,y_1,y_2} y_1 = S(x_1) \wedge y_2 = S(x_2)\big)(0, y, z, x)$$

However, the use of TC^k is not so natural, and its implementation is a serious problem for proof checkers which are based on strict discipline of types, like LF-style Logical Frameworks (see [14, 1, 21]). To see why, let σ be a signature of some first-order language L. The standard way of representing L in the LF is by using an LF-signature σ^* which includes the types ι and o (representing the set of terms of L and the set of formulas of L, respectively), the judgment $true : o \to Type$, constants for the connectives and quantifiers (where the type of $\forall$, e.g., is $(\iota \to o) \to o$), and constants corresponding to those of σ with appropriate types (thus a binary predicate symbol of σ is assigned the type $\iota \to \iota \to o$, while a binary function symbol — the type $\iota \to \iota \to \iota$). Following this approach, the obvious (and the only natural) way to represent TC in such frameworks is by introducing a constant:

$$TC: \quad (\iota \longrightarrow \iota \longrightarrow o) \longrightarrow (\iota \longrightarrow \iota \longrightarrow o)$$

TC^2, in turn, should be represented by a constant of the form:

$$TC^2 : (\iota \to \iota \to \iota \to \iota \to o) \longrightarrow (\iota \to \iota \to \iota \to \iota \to o)$$

Obviously, the complexity of the types of the constants needed for TC^k according to this standard approach grows with k. Worse: there is no way of including all of them in one finite signature, while the LF (as

well as any other logical framework which is based on strict discipline of types) allows only finite signatures. [9]

An obvious better solution to the problem caused by Theorem 2 is to allow the (explicit or implicit) construction of *pairs* in the language. Using pairs all the TC^k can be reduced to TC^1 in a trivial way. Drawing on Feferman's analysis and results in ([9]), we now show that the availability of both pairs and the basic TC (i.e. TC^1) together suffice for having *all* types of finitary inductive definitions at our disposal.

4.2. Finitary inductive Definitions of Relations

For simplicity, we assume first that a pairing function is explicitly given in the language.

DEFINITION 2. *Let σ be a first-order signature with equality having at least one constant* 0. *Let $\mathcal{L}$ be a language which contains $L^1_{TC}(\sigma)$.*

1. *A structure $\mathcal{D}$ for $\mathcal{L}$ is called* admissible *if there exists a term t(x,y) of $\mathcal{L}$ (denoted below simply as (x,y)) such that:*

$$\mathcal{D} \models (x_1, y_1) = (x_2, y_2) \rightarrow x_1 = x_2 \wedge y_1 = y_2$$

$$\mathcal{D} \models (x, y) \neq 0$$

2. *Let $\mathcal{D}$ with domain D be admissible for $\mathcal{L}$, and let Ω be a class of formulas of $\mathcal{L}$. We denote by $\mathcal{K}^{\Omega}(\mathcal{D})$ the set of subsets of D which are definable by some formula in Ω.*

NOTE 5. Like in [9], by letting $S(x) = (x, 0)$ we get a a copy of the natural numbers in any admissible structure $\mathcal{D}$. In what follows we shall identify the natural numbers with this copy.

NOTE 6. As noted in [9], every subset A of D induces an n-ary relation $A^{(n)}$ on D, defined by: $A^{(n)}(x_1, \ldots, x_n) \Leftrightarrow (x_1, \ldots, x_n) \in A$ (where $(x_1, \ldots, x_n)$ is an abbreviation of $(\cdots((x_1, x_2), x_3)\cdots, x_n)$). Conversely: every n-ary relation $A^{(n)}$ on D is induced by some $A \subseteq D$. Hence it suffices to investigate definability of subsets of D.

DEFINITION 3. *Let $\mathcal{L}$ be as in Definition 2. A class Ω of formulas of $\mathcal{L}$ is called $\mathcal{TC}^+$-closed if it contains all equations in $\mathcal{L}$ and is closed under applications of $\vee, \wedge, \exists$, and TC (i.e., if $\varphi \in \Omega$, and t and s are terms, then $(TC_{x,y}\varphi)(s,t) \in \Omega$).*

[9] The real source of the problem is that the generalized transitive closure operation does not have a unique arity. Hence any system in which every constant should have a fixed arity can directly handle it only by officially using an infinite number of primitive symbols, or by using a very roundabout codification.

NOTE 7. If Ω is closed under substitutions of variables for variables (which is the case in all interesting cases) then by Note 2 closure under $\exists x$ follows from closure under TC and $\vee$.

THEOREM 3. *Let $\mathcal{D}$ be admissible for $\mathcal{L}$, and let Ω be a $\mathcal{TC}^+$-closed class of formulas of $\mathcal{L}$. Then $\mathcal{K}^\Omega(\mathcal{D})$ is closed under finitary inductive definitions (as they are defined in [9]). In other words: Whenever $A_1, \ldots, A_m, B_1, \ldots, B_p$ are all in $\mathcal{K}^\Omega(\mathcal{D})$, then so are the least $X_1, \ldots, X_m$ which satisfy the following conditions:*

(1) $A_i \subseteq X_i \quad (1 \leq i \leq m)$.

(2) If $a_1 \in X^j_{k_1}, \ldots, a_{n_j} \in X^j_{k_{n_j}}$ and $(b, a_1, \ldots, a_{n_j}) \in B_j$ then also $b \in X_j$ $(1 \leq j \leq p)$.

(here $A_1, \ldots, A_m$ provide the initial conditions*, while $B_1, \ldots, B_p$ are the* inductive rules*).*

Proof: The definability of m sets $X_1, \ldots, X_m$ is equivalent to the definability of the single set $Z = \bigcup_{i \leq m} X_i \times \{i\}$, since

$$z \in Z \leftrightarrow \bigvee_{i \leq m} \exists x (x \in X_i \wedge z = (x, i))$$

$$x \in X_i \leftrightarrow \exists z (z \in Z \wedge z = (x, i))$$

Since the initial and inductive conditions concerning $X_1, \ldots, X_m$ in the formulation of the theorem can similarly be transformed into conditions on this set, we may assume in what follows w.l.o.g. that $m = 1$. With this assumption we may use disjunction (and, if necessary, dummy conditions of the form $y_i = y_i$) to further assume that $p = 1$ as well[10]. Hence it remains to show that $\mathcal{K}^\Omega(\mathcal{D})$ is closed under Feferman's operations I_k (from [9]), where for $A, B \subseteq D$ and $k \geq 1$, $I_k(A, B)$ is the least $X \subseteq D$ such that:

(i) $A \subseteq X$

(ii) If $x_1, \ldots, x_k \in X$, and $(y, x_1, \ldots, x_k) \in B$, then $y \in X$.

To show the closure of $\mathcal{K}^\Omega(\mathcal{D})$ under I_k, let (following [9])

$$Seq(X) =_{Df} \{0\} \cup \{(0, x_1, \ldots, x_n) \mid n \geq 1, x_1, \ldots, x_n \in X\}$$

The main idea of the proof is to show that given A and B, $\mathcal{K}^\Omega(\mathcal{D})$ contains the set of finite sequences (represented as elements of $Seq(V_0)$)

[10] This reduction to the case where $m = p = 1$ is exactly like what is done in [9] for Feferman's system FS_0.

which can be viewed as "proofs" that their last element is in $I_k(A,B)$. Obtaining then $I_k(A,B)$ is easy, since for any $a \in Seq(X)$ both n and x_n are uniquely determined, and x_n can easily be obtained from a.

To simplify notation, we do here the case $k=2$ (the case where $k>2$ is almost identical, while the case $k=1$ is done below [11]). The details are given in the following definitions and easily shown facts (where we use $\{(y,x) \mid \varphi\}$ as an abbreviation of $\{z \mid \exists y \exists x (z=(y,x) \wedge \varphi)\}$).

- For $B \subseteq D$ let $TC(B)$ be the transitive closure of the binary relation induced by B. In other words:

$$TC(B) = \{(x,y) \mid \big(TC_{u,v} \exists z.z=(u,v) \wedge z \in B\big)(x,y)\}$$

  Obviously, $TC(B) \in \mathcal{K}^{\Omega}(\mathcal{D})$ for all $B \in \mathcal{K}^{\Omega}(\mathcal{D})$.

- $I_1(A,B) = \{x \mid x \in A \vee \exists y.y \in A \wedge (x,y) \in TC(B)\}$. Hence $\mathcal{K}^{\Omega}(\mathcal{D})$ is closed under I_1.

- $Seq(A) = I_1(\{0\}, \{(y,x) \mid \exists w(y=(x,w) \wedge w \in A)\})$. Hence $Seq(A)$ is in $\mathcal{K}^{\Omega}(\mathcal{D})$ whenever $A \in \mathcal{K}^{\Omega}(\mathcal{D})$.

- Let $1^{-}Perm$ be the set of all pairs of the form:

$$((x_1,\ldots,x_l,a,b,y_1,\ldots,y_n),\ (x_1,\ldots,x_l,b,a,y_1,\ldots,y_n))$$

  where $l>0$ and $n \geq 0$. Then $1^{-}Perm \in \mathcal{K}^{\Omega}(\mathcal{D})$, since it is identical to $I_1(S,E)$, where $S(witch)$ and $E(nlarge)$ are defined by:

$$S = \{(y,x) \mid \exists z \exists a \exists b (x=(z,a,b) \wedge y=(z,b,a)\}$$
$$E = \{(y,x) \mid \exists a \exists b \exists v.x=(a,b) \wedge y=((a,v),(b,v))\}$$

  (S is the set of pairs $((x_1,\ldots,x_l,a,b),(x_1,\ldots,x_l,b,a))$ $(l>0)$, while E corresponds to the inductive rule: $\frac{(a,b) \in 1^{-}Perm}{((a,v),(b,v)) \in 1^{-}Perm}$).

- Let $Perm$ be the set of all pairs of the form:

$$((a,x_1,\ldots,x_n),(a,x_{\pi(1)},\ldots,x_{\pi(n)}))$$

  where π is a permutation of $\{1,\ldots,n\}$. Then $Perm \in \mathcal{K}^{\Omega}(\mathcal{D})$, since $Perm = I_1(id^*, 1^{-}Perm)$, where $id^* = \{(x,x) \mid \exists u \exists v.x=(u,v)\}$.

- Let $Proof(A,B)$ be the set of sequences $(0,x_1,\ldots,x_n)$ for which there exists a permutation π of $\{1,\ldots,n\}$ s. t. for every $1 \leq i \leq n$, either $x_{\pi(i)} \in A$ or there exist j,l such that $\pi(j),\pi(l) < \pi(i)$ and $(x_{\pi(i)},x_{\pi(j)},x_{\pi(l)}) \in B$. Then $Proof(A,B) \in \mathcal{K}^{\Omega}(\mathcal{D})$, since it is

[11] In [9] it is shown how to reduce (in a similar framework) any I_k to I_2.

identical to $I_1(Seq(A), Perm \cup B^*)$, where $U \cup V$ is an abbreviation for $\{z \mid z \in U \vee z \in V\}$, and

$$B^* = \{(y,x) \mid \exists z \exists u \exists v \exists w. y = (x,z) \wedge x = (u,v,w) \wedge (z,v,w) \in B\}$$

Note that y is obtained from x in B^* by adding to x the result of applying the rule B to the last two components of x.

– $I_2(A,B) = \{y \mid \exists x \exists z. x \in Proof(A,B) \wedge x = (z,y)\}$. Hence $I_2(A,B) \in \mathcal{K}^\Omega(\mathcal{D})$ whenever $A, B \in \mathcal{K}^\Omega(\mathcal{D})$.

NOTE 8. Feferman has taken I_2 as a primitive of his FS_0. A proof similar to that of Theorem 3 shows that he could have taken instead just I_1 (or TC applied to classes).

NOTE 9. An examination of the proof Theorem 3 shows that it is not necessary to assume that the language has a pairing term for $\mathcal{D}$. It suffices that there is a formula $\psi(x,y,z)$ in Ω such that:

$$\mathcal{D} \models \forall x \forall y \exists z. z \neq 0 \wedge \psi(x,y,z)$$

$$\mathcal{D} \models \psi(x,y,z_1) \wedge \psi(x,y,z_2) \rightarrow z_1 = z_2$$

$$\mathcal{D} \models \psi(x_1,y_1,z) \wedge \psi(x_2,y_2,z) \rightarrow x_1 = x_2 \wedge y_1 = y_2$$

4.3. Finitary Inductive Definitions of Partial Functions

We have concentrated so far on inductive definitions of classes and relations. However, finitary inductive definitions are used also for defining partial functions. To do this within our framework, we follow the modern approach, and identify a partial function f from D to D with its graph. Accordingly, we call an n-ary f *definable* if the following set is definable: $\{(x_1,\ldots,x_n,y) \mid y = f(x_1,\ldots,x_n)\}$. It is not difficult then to show that from our results concerning definability of relations it follows that the usual methods of introducing new functions using finitary inductive definitions are available to us. As an example, we show that for admissible $\mathcal{D}$ and for a $\mathcal{TC}^+$-closed class of formulas Ω, the set $\mathcal{K}^\Omega(\mathcal{D})$ is closed under the three basic functionals which have been used in [9].

THEOREM 4. *If Ω is $\mathcal{TC}^+$-closed, and the partial functions f and g are in $\mathcal{K}^\Omega(\mathcal{D})$ (where $\mathcal{D}$ is admissible), then so are also the partial functions $P(f,g)$, $C(f,g)$, and $R(f,g)$, where $P(f,g) = \lambda x.(fx,gx)$, $C(f,g) = \lambda x.f(gx)$, and $R(f,g)$ is the function h defined by primitive recursion as follows:*

$$h(x) = \begin{cases} 0 & x = 0 \\ f(y) & x = (y, 0) \\ g(y, z, w, h(y, z), h(y, w)) & x = (y, (z, w)) \end{cases}$$

Proof: we have:

- $P(f, g) = \{z \mid \exists x \exists y_1 \exists y_2 . z = (x, (y_1, y_2)) \wedge (x, y_1) \in f \wedge (x, y_2) \in g\}$
- $C(f, g) = \{z \mid \exists x \exists y \exists w . z = (x, y) \wedge (x, w) \in g \wedge (w, y) \in f\}$
- $R(f, g)$ is the least set h which contains the set

  $$\{w \mid w = (0, 0) \vee \exists x \exists y . w = ((x, 0), y) \wedge (x, y) \in f\}$$

  and is closed under the following inductive rule: If $(x, y, u_1) \in h$, $(x, z, u_2) \in h$, and $(x, y, z, u_1, u_2, u_3) \in g$, then $(x, (y, z), u_3) \in h$. By Theorem 3 it follows therefore that $R(f, g)$ is in $\mathcal{K}^{\Omega}(\mathcal{D})$.

5. Using TC for the Mechanization of Mathematics

5.1. FS_0 and Recursively Enumerable Sets

A formal mathematical system is a finite collection of syntactic categories (over some finite set of basic symbols called the "alphabet"). Each of these categories should be semi-decidable: if something belongs to it, then it should be possible to mechanically show that this is the case. In practice, the various syntactic categories are invariably defined using *finitary inductive definitions*. As was explained in the introduction, this observation has led Feferman in [9] (following previous works of Post [22] and Smullyan [23]) to propose FS_0, a simple theory of inductive definitions, as a general framework for implementing formal systems and for reasoning about them. The main two features which distinguish FS_0 from previous works are:

- Rather than the set of strings of symbols from some finite alphabet U of basic symbols, the universe of expressions in FS_0 is taken to be that of Lisp's S-expressions (or lists), where (without a loss in generality) only one basic symbol 0 is needed. In other words: the pairing function is taken to be primitive, and the universe of expressions is V_0, the least set which includes 0, and includes (a, b) whenever it includes a and b (note that every admissible structure contains an inductively defined substructure which is isomorphic

to V_0). This simplifies definitions and propositions a lot, and allows for greater flexibility (and it is also strongly justified by the results of the previous section!).

– FS_0 provides explicit class terms for denoting subsets of V_0, as well as explicit means for introducing such subsets using finitary inductive definitions. It is not difficult to show that a subset of V_0 is denoted by some class term of FS_0 iff it is recursively enumerable.

The approach of FS_0 for constructing class-denoting terms is based on the use of Combinators rather than the use of abstractions. Moreover: In addition to terms for classes FS_0 has also terms denoting functions, and these terms are used in constructing class terms. The definition of the two classes is as follows:

$C\text{-}FnTm$ (closed function terms)

1. The constants I, D, P_1, P_2, and K_0 are in $C\text{-}FnTm$.
2. If $f, g \in C\text{-}FnTm$ then $P(f,g), C(f,g), R(f,g) \in C\text{-}FnTm$.

$C\text{-}ClTm$ (closed class terms)

1. The constant $\{0\} \in C\text{-}ClTm$.
2. If $f \in C\text{-}FnTm$ and $S \in C\text{-}ClTm$ then $f^{-1}S \in C\text{-}ClTm$.
3. If $S,T \in C\text{-}ClTm$ then $S \cap T, S \cup T, I_2(S,T) \in C\text{-}ClTm$.

Here I, P_1, P_2 and K_0 denote, respectively, the identity function, the projection functions for pairs, and the constant function $\lambda x.0$. D denotes the function that given (x, y, u, v) returns u if $x = y$ and v if $x \neq y$ (it returns 0 if the input is not a 4-tuple). The class term $f^{-1}S$ denotes the set $\{x \mid f(x) \in S\}$. The meaning of the other constructs was explained in the previous section.

We describe now a very simple language with TC, in which the subsets of V_0 which are definable by abstractions are exactly those which are denoted by class terms of FS_0 (i.e., the r.e. subsets of V_0).

DEFINITION 4. *$\mathcal{PTC}^+$, the minimal pure language whose set of formulas is $\mathcal{TC}^+$-closed, is defined as follows:*

Terms of $\mathcal{PTC}^+$

1. *The constant 0 is a term.*
2. *Every (individual) variable is a term.*
3. *If t and s are terms then so is (t, s).*

Formulas of $\mathcal{PTC}^+$

1. *If t and s are terms then $t = s$ is a formula.*
2. *If φ and ψ are formulas then so are $\varphi \vee \psi$ and $\varphi \wedge \psi$.*
3. *If φ is a formula, x, y are two different variables, and t, s are terms, then $(TC_{x,y}\varphi)(t, s)$ is a formula.*

NOTE 10. Recall that the existential quantifier is definable in $\mathcal{PTC}^+$ by Note 2 (that note provides also an alternative, shorter syntax for TC in the presence of this quantifier).

DEFINITION 5. $\Sigma = \mathcal{K}^{\mathcal{PTC}^+}(V_0)$ *(i.e., $S \in \Sigma$ iff there is a formula $\varphi(x)$ of $\mathcal{PTC}^+$ such that $S = \{x \in V_0 \mid \varphi(x)\}$).*

THEOREM 5. *The following are equivalent for a subset S of V_0:*

1. *S is recursively enumerable.*
2. *S is definable by some closed class term of FS_0.*
3. *$S \in \Sigma$.*

Proof: That (1) implies (2) is well-known, and was essentially shown in [9] (see Theorem 10.3 and 20.1 there). That (3) implies (1) is obvious from Church Thesis (and can easily be shown formally by standard methods). We prove here that (2) implies (3).

LEMMA 1. $NEQ = \{(x, y) \in V_0 \mid x \neq y\} \in \Sigma$

Proof of Lemma 1: NEQ is the least subset S of V_0 which contains $\{w \mid \exists x \exists y.w = ((x, y), 0) \vee w = (0, (x, y))\}$ and is closed under the following inductive rules: If $(x, y) \in S$ then $((x, z), (y, w)) \in S$ and $((z, x), (w, y)) \in S$. Hence $Neq \in \Sigma$ by Theorem 3.

LEMMA 2. *If f is a closed function term of FS_0 then the function denoted by f (viewed as a set of pairs) is in Σ.*

Proof: By Theorem 4 it suffices to prove that the basic functions of FS_0 are in Σ. This is obvious for I, P_1, P_2, K_0, since:

$I = \{w \mid \exists x.w = (x, x)\}$

$P_1 = \{w \mid \exists x \exists y.w = ((x, y), x)\}$

$P_2 = \{w \mid \exists x \exists y.w = ((x, y), y)\}$

$$K_0 = \{w \mid \exists x.w = (x, 0)\}$$

Finally, that $D \in \Sigma$ follows from Lemma 1, since $D = D_1 \cup D_2$, where:

$$D_1 = \{w \mid \exists x.w = (x,0) \wedge (x = 0 \vee \exists y.x = (0,y) \vee \exists y \exists z.x = ((0,y),z))\}$$

$$D_2 = \{w \mid \exists x,y,u,v,z.w = (x,y,u,v,z) \wedge .(x = y \wedge z = u) \vee (x \neq y \wedge z = v)\}$$

End of the proof of Theorem 5: By induction on the construction of closed class terms in FS_0. The constant $\{0\}$ denotes the set $\{x \mid x = 0\}$, which is in Σ. Σ is obviously closed under $\cap$ and $\cup$, and by Theorem 3 it is also closed under I_2. It remains to prove that if S denotes a set in Σ, and f is a closed function term of FS_0, then $f^{-1}S \in \Sigma$. This follows from Lemma 2 and the induction hypothesis, since

$$f^{-1}S = \{x \mid \exists y \exists w.w = (x,y) \wedge y \in S \wedge w \in f\}$$

NOTE 11. Theorem 5 leads in a natural way to the most concise (and simplest?) formulation of Church Thesis that I know.

NOTE 12. In addition to closed terms, Feferman allows in FS_0 also terms depending on parameters of three different types: individual parameters, function parameters and class parameters. Theorem 5 can easily be generalized to this more general case. Thus it is possible to prove by the same method that if S is a class term depending on the individual parameters $y_1, \ldots, y_k$ then the set $\{(y_1, \ldots, y_k, x) \mid x \in S\}$ is in Σ. For handling the use of class and function parameters we have to generalize Theorem 5 to signatures which includes extra function and predicate symbols (in addition to $0, =$ and the pairing function). To do this all is needed is to include any atomic formula in the class of formulas whivh defines Σ, and the proof can proceed as before. *This generalization amounts to a logical characterization of relative recursive enumerability in a given set of functions and relations.*

NOTE 13. The structure V_0 is a substructure of any admissible structure, and it can be defined inductively. A careful examination of the proof of Theorem 3 for the particular case of the definability of V_0 reveals that there exists a formula $V_0(x)$ of $\mathcal{PTC}^+$ which defines V_0 in *all* admissible structures (such a formula may be called *absolute*). The structure V_0 can therefore be categorically axiomatized in $L^1_{TC}(\{0, P\})$ (where P is a binary function symbol) by the following theory PTC:

$$\forall x_1 \forall x_2 \forall y_1 \forall y_2 . P(x_1, y_1) = P(x_2, y_2) \rightarrow x_1 = x_2 \wedge y_1 = y_2$$

$$\forall x \forall y . \neg P(x, y) = 0$$

$$\forall x . V_0(x)$$

The strength of a formal system which is based on PTC depends on that of the underlying TC-logic. It might be instructive to determine what logical principles are needed to make it equivalent to PA.

NOTE 14. Feferman's FS_0 provides of course not only means for defining subsets of V_0, but also an axiomatic system for reasoning about V_0 and its definable subsets. The central axiom of this system is an induction axiom which is formulated using a class variable. This axiom is equivalent in strength to $\Sigma^1_0 - IA$ (since only r.e. sets are definable). Now in $\mathcal{PTC}^+$ the induction rule is apriorily restricted to Σ^1_0 formulas, since only such formulas are available. On the other hand the axioms of PTC are of course not in $\mathcal{PTC}^+$, and the logical connectives which are excluded from $\mathcal{PTC}^+$ ($\neg$ and $\rightarrow$) are used in them in an essential way. The best way to develop a corresponding proof system which remains within the language $\mathcal{PTC}^+$ again seems to be the use of a Gentzen-type calculus (with substitution of terms for free variables as one of the rules). It might again be instructive to find out whether one should actually use weaker logical principles for TC in order to get a system which is equivalent in its power to FS_0 (and so to PRA - see [9]), or whether the use of a limitted language suffices here.

5.2. A Formalization of Set Theory

We finally turn to the formalization of the whole of Mathematics, or at least Set Theory, in an appropriate language containing TC. Unlike standard formalizations in books on axiomatic set theory, we want our language to be as close as possible to that used in actual mathematical practice, and at the same time easy for mechanical manipulations and interactive theorem proving. This means that the language should provide a rich class of *terms* denoting sets on one hand, but be based on syntactical (rather than semantic) considerations on the other. We present here one version, in which the language has (in addition to variables) terms of the form $\{x \mid \varphi\}$. Of course, not every formula φ can be allowed in $\{x \mid \varphi\}$. The main novelty in what we present now is in providing a purely *syntactic* characterization of the class of formulas which one can safely use (according to ZF) in abstractions. In order to do so we introduce a safety *relation* between formulas and finite sets of variables (rather than treating safety as a property of formulas). The intended meaning of "The formula $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_k)$ is safe with respect to $\{x_1, \ldots, x_n\}$" is that for any assignment of sets to the parameters $y_1, \ldots, y_k$, the class $\{(x_1, \ldots, x_n) \mid \varphi\}$ is a set.

The Formal definition of our language is the following (where $Fv(A)$ denotes the set of free variables of A):

Terms:

- Every variable is a term.
- If x is a variable, and φ is a formula which is safe w.r.t. $\{x\}$, then $\{x \mid \varphi\}$ is term.

Formulas:

- If t and s are terms than $t = s$, $t \in s$ and $t \subseteq s$ are formulas.
- If φ and ψ are formulas, x and y are variables, and s, t are terms, then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, $\forall x\varphi$, $\exists x\varphi$, and $(TC_{x,y}\varphi)(t, s)$ are formulas.

The safety relation:

- Every formula is safe w.r.t $\emptyset$.
- If x is a variable, t is a term, and $x \notin Fv(t)$, then $x = t$, $t = x$, $x \in t$, and $x \subseteq t$ are safe w.r.t $\{x\}$.
- If φ and ψ are both safe w.r.t. X, then so is $\varphi \vee \psi$.
- If φ is safe w.r.t. X, and ψ is safe w.r.t. Y, then $\varphi \wedge \psi$ and $\psi \wedge \varphi$ are safe w.r.t. $X \cup (Y - Fv(\psi))$.
- If $y \in X$ and φ is safe w.r.t. X, then $\exists y\varphi$ is safe w.r.t. $X - \{y\}$.
- If ψ is safe w.r.t. X, $y \in Fv(\varphi)$, and $X \cap Fv(\varphi) = \emptyset$, then $\exists y\varphi \wedge \forall y(\varphi \rightarrow \psi)$ is safe w.r.t. X.
- If φ is safe w.r.t X, and $\{x, y\} \cap X \neq \emptyset$, then $\big(TC_{x,y}\varphi\big)(x, y)$ is safe w.r.t. X.

The axioms of the system are the following:

The extensionality axioms :

- $x \subseteq y \wedge y \subseteq x \rightarrow x = y$
- $z \in x \wedge x \subseteq y \rightarrow z \in y$
- $x \subseteq y \vee \exists z(z \in x \wedge z \notin y)$

The comprehension schema :

- $\forall x(x \in \{x \mid \varphi\} \leftrightarrow \varphi)$

Other axioms:

- The axiom of choice
- The regularity axiom

What is the connection between this system and ZF, and what is the role of TC here? Well, to start with, it is not too difficult to prove (see [5]) that the TC-free fragment of this system (which is essentially its first-order fragment) is equivalent to the system obtained from ZF by deleting the axiom of infinity. The use of TC, on the other hand, enables us to categorically introduce the class ω of the finite ordinals:

$$\omega = \{x \mid x = \emptyset \vee \exists y.y = \emptyset \wedge \big(TC_{x,y}(x = \{z \mid z = y \vee z \in y\})\big)(x, y)\}$$

(Here $\emptyset$ may be defined, e.g., as $\{x \mid (\exists y.y = y) \wedge \forall y(y = y \rightarrow x \in y)\}$). It follows that with an appropriate logical system for first order logic strengthened with TC, our system is at least as strong as ZF (and most probably equivalent to it in case we employ a *natural* system).

NOTE 15. To understand the clause concerning TC in the definition of the safety relation, note that intuitively $\big(TC_{x,y}\varphi\big)(x, y)$ is equivalent to the infinitary disjunction:

$$\begin{aligned} &\varphi(x, y) \\ \vee&\exists w_1(\varphi(x, w_1) \wedge \varphi(w_1, y)) \\ \vee&\exists w_1 \exists w_2(\varphi(x, w_1) \wedge \varphi(w_1, w_2) \wedge \varphi(w_2, y)) \\ \vee&\ldots \end{aligned}$$

Now if φ is safe w.r.t. X and $x \in X$ (say) then by the clauses concerning $\vee$, $\wedge$ and $\exists$, each disjunct defines a set (for every assignment of values to the parameters, if any exist). Hence the collection of sets which satisfy $\big(TC_{x,y}\varphi\big)(x, y)$ is a countable union of sets, and so it is a set itself.

NOTE 16. The clause in the definition of the safety relation concerning $\forall$ and $\rightarrow$ is needed only for getting the full power of the replacement axiom (see [5]). It should be noted that by deleting it we get a system which is still *stronger* than the original system Z of Zermelo. Thus the existence of the transitive closure (in the usual sense of set theory) of any set is an easy corollary of the clause concerning TC, but cannot be proved in Z[12]. It is interesting to note also that if we add to the language Hilbert's ε-operation, and use it to formulate the axiom of choice as $\exists x\varphi \rightarrow \varphi(\varepsilon x\varphi/x)$ (the axiom of global choice), then the clause concerning $\forall$ and $\rightarrow$ becomes redundant.

NOTE 17. Except for the clause concerning TC, all the other clauses in the definition of the safety relation remain valid if by a "set" we

[12] This fact has been shown, e.g., in an unpublished note of Martin Goldstern.

mean a *finite* set (retaining otherwise the intended meaning of the safety relation as it has been explained above). This is not true, of course, for the clause concerning TC. What *is* true in the finite case is that if φ is safe w.r.t. X, and $\{x, y\} \subseteq X$, then $(TC_{x,y}\varphi)(x, y)$ is safe w.r.t. X.

NOTE 18. The set of hereditary finite sets is a model of the system which is obtained from our version of ZF by deleting the clause concerning TC from the definition of the safety relation. A complete (and categorical) theory of the hereditary finite sets can be obtained by adding instead of that clause an axiom saying that every set is in $I_2(\{\emptyset\}, \{(y, x_1, x_2) \mid y = x_1 \vee y \in x_2\})$ (this axiom is expressible using TC by Theorem 3, since the standard pairing function of set theory is available here). The axiom of choice and the regularity axiom should become redundant in the resulting system.

NOTE 19. With TC it is possible to formulate some weak versions of the regularity axiom, like:

$$\Big(\forall x \neg \big(TC_{x,y}(x \in y)\big)(x, x)\Big) \wedge \Big(\forall x \big(TC_{x,y}(x \in y)\big)(\emptyset, x)\Big)$$

It might be interesting to investigate the resulting theory.

NOTE 20. Our language is expressive enough for introducing most (all?) standard abbreviations and constructs used in normal mathematical texts. Thus the definite article (the unique x such that φ, in case such exists) can be defined by:

$$\iota x\varphi = \{y \mid \exists x\varphi \wedge \forall x(\varphi \rightarrow y \in x)\}$$

where y is a new variable, not occurring in φ (note that according to this definition $\iota x\varphi$ is $\emptyset$ if there is no set which satisfy φ, and it is the intersection of all the sets which satisfy φ otherwise). $\lambda x \in s.t$ (where $x \notin Fv(s)$) can then be defined as $\{z \mid \exists x(x \in s \wedge z = (x, t)\}$, while the application of a function f to an argument t can be defined as $\iota x.(t, x) \in f$.

6. Conclusion and Further Research

Our main subject in this work was the expressive power and the reasoning potential of logics with transitive closure operations. As noted above, our next major goal is to work out this potential by developing

computationally efficient sound proof system(s) for logics with TC that will be strong enough for various mathematical needs. A promising direction to follow here is to try to solve the various problems raised above of how to get formal systems of *logic* which will be equivalent to (or at least as strong as) some basic formal mathematical *theories*, like: PRA, PA, ZF, and others.

As we have already declared, We further believe that an appropriate logic of TC might be sufficient for most of applicable mathematics. Showing this belief to be true is a further future project.

References

1. Avron A., Honsell F. A,, Mason I.A., and Pollack R., *Using Typed Lambda Calculus to Implement Formal Systems on a Machine*, Journal of Automated Deduction, vol. 9 (1992) pp. 309-354.
2. Avron A., *Theorems on Strong Constructibility with a Compass alone*, Journal of Geometry, vol. 30 (1987), pp. 28-35.
3. Avron A., *On Strict Strong Constructibility with a Compass Alone*, Journal of Geometry, vol. 38 (1990), pp. 12-15.
4. Avron A., *An Exercise in An Interactive Geometrical research*, Annals of Mathematics and Artificial Intelligence, vol. 9 (1993), pp. 239-252.
5. Avron A., *Partial Safety of Formulas as a Unifying Foundational Principle*, To appear.
6. Barwise J., Ed., **Handbook of Mathematical Logic**, Studies in Logic and the Foundations of Mathematics, vol. 90, North-Holland Publishing Company, 1977.
7. Büchi, J.R., *On a Decision Method in Restricted Second Order Arithmetic*, in: **Logic Methodology and Philosophy of Science**, Proceedings of the 1960 Congress, Stanford University Press, Stanford, CA, (1962), pp. 1-11.
8. Ebbinghaus H. D., and Flum J., **Finite Model Theory**, Perspectives in Mathematical Logic, Springer, 1995.
9. Feferman S., *Finitary Inductively Presented Logics*, in: **Logic Colloquium 1988** (1989), Amsterdam, North-Holland, pp. 191-220. Reprinted in [10], pp. 297-328.
10. Gabbay D., editor, **What is a Logical System?** Oxford Science Publications, Clarendon Press, Oxford, 1994.
11. Gentzen G., *Neue Fassung des Widerspruchsfreiheitsbeweises für die reine Zahlentheorie*, Forschungen zur Logik, N.S., No. 4, pp. 19-44 (English translation in: **The collected work of Gerhard Gentzen**, edited by M.E. Szabo, North-Holland, Amsterdam, (1969)).
12. Grädel E., *On Transitive Closure Logic*, in: **Computer Science Logic (Berne 1991)**, Springer LNCS 626, 1992, pp. 149-163.
13. Gurevich Y., *Logic and the Challenge of Computer Science*, in: Börger E., ed., **Trends in Theoretical Computer Science**, Computer Science Press Inc., Rockville, Maryland, USA (1988), pp. 1-58.
14. Harper R., Honsell F. and Plotkin G., *A Framework for Defining Logics*, Journal of the Association for Computing Machinery, vol. 40 (1993), pp. 143-184.

15. Immerman, N., *Languages which Capture Complexity Classes*, in: 15th Symposium on Theory of Computing, Association for Computing Machinary (1983), pp. 347-354.
16. Levesque H., Reiter R., Lesperance Y., Lin F., and Scherl R., *Golog: A logic programming language for dynamic domains*, Journal of Logic Programming vol. 31 (1997), pp. 59-84.
17. Matthews S., *A Theory and Its Metatheory in FS_0*, in [10], pp. 329-352, 1994.
18. Matthews S., *Implementing FS_0 in Isabelle: Adding Structure at the Metalevel*, in: J. Calmet and C. Limongelli (eds), **Proc. Disco'96**, Springer, Berlin, 1996
19. Moschovakis Y., *Abstract Recursion as a Foundation for the Theory of Algorithms*, Lecture Notes in Mathematics, vol. 1104 (1984), Springer, pp. 289-364.
20. Matthews S., Smaill A., and Basin D., *Experience with FS0 as a Framework Theory*, in: G. Huet and G. Plotkin, editors, **Logical Environments**, Cambridge University Press, 1993, pp. 61-82.
21. Pfenning F., *The Practice of Logical Frameworks*, in H. Kirchner (ed.): **Proceedings of the Colloquium on Trees in Algebra and Programming, Linkvping, Sweden, April 1996**, Springer-Verlag LNCS 1059, pp. 119-134.
22. Post E., *Formal Reductions of the General Combinatorial Decision Problem*, American J. of Mathematics, pp, 197-214, 1943.
23. Smullyan R., **Theory of Formal Systems**, Princeton University Press, Princeton, 1961.
24. Tarski A., and Givant S., *Tarski's System of Geometry*, Bulletin of Symbolic Logic, vol. 5 (1999), pp. 175-214.