On the Weakness of Sharply Bounded Polynomial Induction

Jan Johannsen
Friedrich-Alexander-Universität Erlangen-Nürnberg

May 4, 1994

Abstract

We shall show that if the theory S_2^0 of sharply bounded polynomial induction is extended by symbols for certain functions and their defining axioms, it is still far weaker than T_2^0, which has ordinary sharply bounded induction. Furthermore, we show that this extended system S_{2+}^0 cannot Σ_1^b-define every function in AC^0, the class of functions computable by polynomial size constant depth circuits.

1 Introduction

The theory S_2 of bounded arithmetic and its fragments S_2^i and T_2^i ($i \geq 0$) were defined in [Bu]. The language of these theories comprises the usual language of arithmetic plus additional symbols for the functions $\lfloor \frac{1}{2}x \rfloor$, $|x| := \lceil \log_2(x+1) \rceil$ and $x \# y := 2^{|x| \cdot |y|}$. Quantifiers of the form $\forall x \leq t$, $\exists x \leq t$ are called *bounded quantifiers.* If the bounding term t is furthermore of the form $|s|$, the quantifier is called *sharply bounded.* The classes Σ_i^b and Π_i^b of the *bounded arithmetic hierarchy* are defined in analogy to the classes of the usual arithmetical hierarchy, where the i counts alternations of bounded quantifiers, ignoring the sharply bounded quantifiers.

S_2^i is axiomatized by a finite set of open axioms (called $BASIC$) plus the schema of polynomial induction ($PIND$) for Σ_i^b-formulae φ:

$$\varphi(0) \wedge \forall x \, (\, \varphi(\lfloor \frac{1}{2}x \rfloor) \rightarrow \varphi(x) \,) \; \rightarrow \; \forall x \, \varphi(x)$$

T_2^i is the same with the ordinary induction scheme for Σ_i^b-formulae replacing $PIND$.

These theories have a close connection to the polynomial hierarchy of Complexity Theory: the main theorem of [Bu] states that for $i \geq 1$, the Σ_i^b-definable functions of S_2^i are exactly the functions from $\square_i^p$, the class of functions computable in polynomial time using an oracle for a set in the $i-1^{\text{th}}$ level of this hierarchy. In that paper it is also shown that $T_2^i \subseteq S_2^{i+1} \subseteq T_2^{i+1}$. It is not known whether any of these inclusions is proper. The paper [K-P-T] shows that this question is related to the separation problem for the polynomial hierarchy.

In [Ta], Takeuti has proved that $S_2^0 \neq T_2^0$ by showing that the former theory cannot define the predecessor function, while the latter can. He uses an interpretation of S_2^0 in S_2 where numbers are coded as descending sequences. We shall use a variant of Takeuti's method to strengthen his results in the following way:

We extend the language of bounded arithmetic by function symbols P and $\dot{-}$ for the predecessor and modified subtraction, as well as *Count* and *MSP* whose meaning is clear from the defining axioms below. Let S^0_{2+} be the theory in this extended language consisting of the *BASIC* axioms, the additional axioms

- $P0 = 0, \qquad P(Sx) = x, \qquad x > 0 \to S(Px) = x$
- $x \dot{-} 0 = x, \qquad x \dot{-} Sy = P(x \dot{-} y), \qquad x \geq y \to (x \dot{-} y) + y = x,$
  $x < y \to x \dot{-} y = 0$
- $Count(0) = 0, \qquad Count(2x) = Count(x),$
  $Count(S(2x)) = S(Count(x))$
- $MSP(x, 0) = x, \qquad MSP(x, Si) = \lfloor \frac{1}{2} MSP(x, i) \rfloor$

and the schema $\Sigma^b_0 - PIND$ (for sharply bounded formulae in the extended language).

We define w *is a sequence of positive numbers* (or "*positive sequence*" for short) by

$$PSeq(w) :\leftrightarrow Seq(w) \wedge \forall i < Len(w)\ \beta(Si, w) \neq 0 \ ,$$

where the predicate *Seq* and the functions *Len* and β are those defined in chapter 2 of [Bu]. From now on, we shall use the functions and predicates defined there without further comment.

Natural numbers are coded by positive sequences as follows: 0 is coded by the empty sequence, and a positive number a is coded by a sequence $A = \langle a_1, \ldots, a_k \rangle$ with the following intended meaning: the binary representation of a consists of a block of a_1 ones followed by a block of a_2 zeros etc. E.g. the number 22 is 10110 in binary and is therefore coded by the sequence $\langle 1, 1, 2, 1 \rangle$.

Let *Code* denote this bijection between natural numbers and positive sequences. We shall see that *Code* is polynomial time computable (p.t.c. for short) and hence Σ^b_1-definable in S^1_2. We shall define an ordering $\leq_C$ on positive sequences such that $Code(a) \leq_C Code(b)$ if and only if $a \leq b$.

For a function f, the code-version of f is the function C^f on positive sequences such that for all $x_1, \ldots, x_n$

$$Code^{-1}(C^f(Code(x_1), \ldots, Code(x_n))) = f(x_1, \ldots, x_n) \ .$$

The code-versions of the primitive functions $|.|, \lfloor \frac{1}{2} . \rfloor, S, P, +, \dot{-}, \cdot, \#, Count$ and MSP can be Σ^b_1-defined in S^1_2.

Therefore we can interpret S^0_{2+} in S_2 via this encoding and use this to prove that integer division by three cannot be Σ^b_1-defined in S^0_{2+}, whereas it can be defined in T^0_2 by use of induction for open formulae only.

Furthermore we show that S^0_{2+} cannot Σ^b_1-define every function in a very small complexity class, viz. the class AC^0 of functions computable by uniform families of polynomial size, constant depth unbounded fan-in circuits.

2 Coding Numbers by Sequences

We shall use the fact that S^1_2 can Σ^b_1-define functions by *length bounded summation*, i.e. let f be a Σ^b_1-defined function, then we can define $F(k) = \sum_{i=0}^{|k|} f(i)$. The existence and uniqueness conditions are easily proved in S^1_2 by use of sequence encoding. In particular we can define a function Σ such that

$$\Sigma(w) = \begin{cases} \sum\limits_{i=1}^{Len(w)} \beta(i,w) & \text{if } Seq(w) \\ 0 & \text{else} \end{cases} .$$

The function *Code* is p.t.c. and hence Σ^b_1-definable in S^1_2. To see this, define the functions f_1, f_2 and f as follows:

$$\begin{aligned} f_1(a) &:= \mu i \leq |a| \; (Bit(|a| - Si, a) = 0) \\ f_2(a) &:= f_1((2^{|a|-f_1(a)} - 1) \dot{-} LSP(|a| - f_1(a), a)) \\ f(a) &:= \begin{cases} 0 * f_1(a) * f_2(a) & \text{if } f_1(a) > 0 \text{ and } f_2(a) > 0 \\ 0 * f_1(a) & \text{if } f_1(a) > 0 \text{ and } f_2(a) = 0 \\ 0 & \text{else} \end{cases} \end{aligned}$$

These are Σ^b_1-definable in S^1_2 by theorem 2.9 of [Bu] and hence p.t.c. Now *Code* can be defined by limited iteration from f:

$$\begin{aligned} \tau(a, 0) &:= 0 \\ \tau(a, Si) &:= \tau(a, i) ** f(LSP(a, |a| - \Sigma(\tau(a, i)))) \end{aligned}$$

Then $Code(a) := \tau(a, |a|)$ and for all $i \leq |a| \;:\; |\tau(a,i)| \leq |a| \cdot (2|a| + 2)$. Note that $Code^{-1}$ is *not* p.t.c. since $Code^{-1}(\langle 1, a\rangle) = 2^a$ for $a \geq 1$.

For the rest of this section, let $A := \langle a_1, \ldots a_k\rangle$ and $B := \langle b_1, \ldots, b_\ell\rangle$. The ordering $\leq_C$ of positive sequences is defined by

$$\begin{aligned} A <_C B &:\leftrightarrow \Sigma(A) < \Sigma(B) \vee (\Sigma(A) = \Sigma(B) \wedge \exists i \leq min(\ell, k) \\ &\qquad (\forall j < i \; (a_j = b_j) \wedge (Even(i) \wedge a_i > b_i \;\vee\; Odd(i) \wedge a_i < b_i))) \;, \\ A =_C B &:\leftrightarrow k = \ell \wedge \forall i \leq k \; a_i = b_i \;, \\ A \leq_C B &:\leftrightarrow A <_C B \vee A =_C B \;. \end{aligned}$$

These definitions are obviously Δ^b_1 w.r.t. S^1_2.

For some of the function symbols, the code-versions can easily defined, viz.

$$\begin{aligned} C^{|\cdot|}(A) &:= Code(\Sigma(A)) \\ C^{\lfloor \frac{1}{2} \cdot \rfloor}(A) &:= \begin{cases} 0 & \text{if } A = 0 \text{ or } A = \langle 1\rangle \\ \langle a_1, \ldots, a_{k-1}\rangle & \text{if } a_k = 1 \\ \langle a_1, \ldots, a_k - 1\rangle & \text{if } a_k > 1 \end{cases} \\ C^{\#}(A, B) &:= \langle 1, \Sigma(A) \cdot \Sigma(B)\rangle \end{aligned}$$

$$C^S(A) \quad := \quad \begin{cases} \langle 1 \rangle & \text{if } A = 0 \\ \langle 1, a_1 \rangle & \text{if } k = 1 \\ \langle a_1, \ldots, a_{k-1} - 1, 1, a_k \rangle & \text{if } k \geq 3 \text{ is odd and } a_{k-1} > 1 \\ \langle a_1, \ldots, a_{k-2} + 1, a_k \rangle & \text{if } k \geq 3 \text{ is odd and } a_{k-1} = 1 \\ \langle a_1, \ldots, a_k - 1, 1 \rangle & \text{if } k \text{ is even and } a_k > 1 \\ \langle a_1, \ldots, a_{k-1} + 1 \rangle & \text{if } k \text{ is even and } a_k = 1 \end{cases}$$

$$C^{Count}(A) \quad := \quad Code\left(\sum_{\substack{i \leq k \\ i \text{ odd}}} a_i \right)$$

These are all obviously p.t.c. and can thus be Σ_1^b-definable in S_2^1. To obtain the code-version of addition, define the following functions:

$$\begin{aligned} Cut(A,n) \quad &:= \quad \max\{i \leq k \,;\, \sum_{j=i}^{k} a_j \geq n\} \\ Head(A,n) \quad &:= \quad \begin{cases} \langle a_1, \ldots, a_{m-1} \rangle & \text{if } \ h = 0 \\ \langle a_1, \ldots, a_{m-1}, h \rangle & \text{else} \end{cases} \\ Tail(A,n) \quad &:= \quad \langle p, t, a_{m+1}, \ldots, a_k \rangle \end{aligned}$$

where $m := Cut(A,n)$, $h := \sum_{j=m}^{k} a_j - n$, $t := a_m - h$ and $p := Mod2(m) + 1$.

$$Merge(A,B) := \begin{cases} \langle a_1, \ldots, a_k, b_2, \ldots b_\ell \rangle & \text{if } \ Mod2(k) = Mod2(b_1) \\ \langle a_1, \ldots, a_k + b_2, \ldots b_\ell \rangle & \text{else} \end{cases}$$

Now we can define

$$C^{+}(A,B) := \begin{cases} Add(A,B) & \text{if } \ A \leq_C B \\ Add(B,A) & \text{else} \end{cases}$$

where *Add* is recursively defined by

$$Add(A,B) := \begin{cases} A & \text{if } \ B = \langle \, \rangle \\ Merge(\, Add(\, Head(A, b_\ell) \,,\, \langle b_1, \ldots, b_{\ell-1} \rangle \,) \,,\, Tail(A, b_\ell) \,) & \text{if } \ell \text{ is even} \\ Add(\, Step(A,B) \,,\, \langle b_1, \ldots, b_{\ell-1} + b_\ell \rangle \,) & \text{if } \ell \text{ is odd} \end{cases}$$

and $Step(A,B) = Code(Code^{-1}(A) + 2^{b_\ell} - 1)$ is given by Table 1.

Since the computation of $Add(A,B)$ terminates after ℓ recursions, and the space required to store intermediate values is bounded by a polynomial in $|A|, \ell$ and $Size(A)$, the recursive definition could be written as a limited iteration, and hence *Add* is p.t.c. and so is C^+.

We will now define the code-version of modified subtraction :

$$C^{\dot{-}}(A,B) := \begin{cases} 0 & \text{if } A \leq_C B \\ Sub(A,B) & \text{else} \end{cases}$$

Since $a - b = C - ((C - a) + b)$, if we choose $C = 2^{|a|+1} - 1$ (i.e. do subtraction by taking the twos complement and addition), we can define

$$\begin{aligned} Red(A) \quad &:= \quad \langle a_2, \ldots, a_k \rangle \\ Sub(A,B) \quad &:= \quad Red(Add(\langle 1 \rangle ** A \,,\, B)) \ . \end{aligned}$$

Table 1: Definition of $Step(A, B)$. To decrease the number of cases, zero entries in a sequence are treated as if they were deleted and the entries left and right of them were added then.

k even	m even	$m=k$			$\langle a_1, \dots, a_{k-1}, a_k - b_\ell, b_\ell \rangle$
		$m+1 = k-1$	$h>0$		$\langle a_1, \dots, a_{k-3}, h-1, 1, t, a_{k-1}-1, 1, a_k \rangle$
			$h=0$	$m \geq 4$	$\langle a_1, \dots, a_{k-5}, a_{k-4}-1, 1, a_{k-3}+a_{k-2}, a_{k-1}-1, 1, a_k \rangle$
				$m=2$	$\langle 1, a_1+a_2, a_3-1, 1, a_4 \rangle$
		$m+1 < k-1$	$h>0$		$\langle a_1, \dots, a_{m-1}, h-1, 1, t, a_{m+1}, \dots, a_{k-1}-1, 1, a_k \rangle$
			$h=0$	$m \geq 4$	$\langle a_1, \dots, a_{m-3}, a_{m-2}-1, 1, a_{m-1}+a_m, a_{m+1}, \dots, a_{k-1}-1, 1, a_k \rangle$
				$m=2$	$\langle 1, a_1+a_2, a_3, \dots, a_{k-1}-1, 1, a_k \rangle$
	m odd	$m+1 = k$	$m \geq 3$		$\langle a_1, \dots, a_{k-3}, a_{k-2}-1, 1, h, t-1, 1, a_k \rangle$
			$m=1$		$\langle 1, h, t-1, 1, a_2 \rangle$
		$m+1 < k$	$m \geq 3$		$\langle a_1, \dots, a_{m-2}, a_{m-1}-1, 1, h, t, a_{m+1}, \dots, a_{k-1}-1, 1, a_k \rangle$
			$m=1$		$\langle 1, h, t, a_2, \dots, a_{k-1}-1, 1, a_k \rangle$
k odd	m even	$m+1 = k$	$h \geq 1$		$\langle a_1, \dots, a_{k-2}, h-1, 1, t, a_k-1, 1 \rangle$
			$h=0$	$m \geq 4$	$\langle a_1, \dots, a_{k-4}, a_{k-3}-1, 1, a_{k-2}+a_{k-1}, a_k-1, 1 \rangle$
				$m=2$	$\langle 1, a_1+a_2, a_3-1, 1 \rangle$
		$m+1 < k$	$h \geq 1$		$\langle a_1, \dots, a_{m-1}, h-1, 1, t, a_{m+1}, \dots, a_k-1, 1 \rangle$
			$h=0$	$m \geq 4$	$\langle a_1, \dots, a_{m-3}, a_{m-2}-1, 1, a_{m-1}+a_m, a_{m+1}, \dots, a_k-1, 1 \rangle$
				$m=2$	$\langle 1, a_1+a_2, a_3, \dots, a_k-1, 1 \rangle$
	m odd	$m<k$	$m \geq 3$		$\langle a_1, \dots, a_{m-2}, a_{m-1}-1, 1, h, t, a_{m+1}, \dots, a_k-1, 1 \rangle$
			$m=1$		$\langle 1, h, t, a_2, \dots, a_k-1, 1 \rangle$
		$m=k$	$k \geq 3$		$\langle a_1, \dots, a_{k-1}-1, 1, a_k-b_\ell, b_\ell-1, 1 \rangle$
			$k=1$		$\langle 1, a_1-b_\ell, b_\ell-1, 1 \rangle$

$C^P(A)$ is then simply defined as $C^{\dot{-}}(A, \langle 1\rangle)$. Now for the code-version of multiplication: We use an iterated version of the so-called Russian Peasant Algorithm, i.e.:

$$\begin{aligned} x \cdot 0 &:= 0 \\ x \cdot 2^i y &:= 2^i x \cdot y \quad \text{where } y \text{ is odd} \\ x \cdot t^{(i)}(y) &:= 2^i x \cdot y + \sum_{j=0}^{i-1} 2^j x \quad \text{where } y \text{ is even and } t(x) := 2x+1. \end{aligned}$$

An operation corresponding to multiplication by powers of two is easily defined by

$$MPT(A,n) := \begin{cases} 0 & \text{if } A = 0 \\ \langle a_1, \dots, a_k + n\rangle & \text{if } k \text{ is even} \\ \langle a_1, \dots, a_k, n\rangle & \text{if } k \text{ is odd} \end{cases}.$$

Then since $\sum_{j=0}^{i-1} 2^j x = (\sum_{j=0}^{i-1} 2^j) \cdot x = (2^i - 1) \cdot x = 2^i x - x$, $C^{\cdot}$ can be recursively defined by

$$C^{\cdot}(A,B) := \begin{cases} 0 & \text{if } B = 0 \\ C^{\cdot}(MPT(A, b_\ell), \langle b_1, \dots, b_{\ell-1}\rangle) & \text{if } \ell \text{ is even} \\ C^{+}(C^{\cdot}(MPT(A, b_l), \langle b_1, \dots, b_{\ell-1}\rangle), Sub(MPT(A, b_l), A)) & \text{if } \ell \text{ is odd} \end{cases}.$$

Just as in the case of *Add*, the number of recursions used to compute $C^{\cdot}(A,B)$ is ℓ, and the space required can be bounded by a polynomial in values that are bounded by lengths, thus $C^{\cdot}$ is computable in polynomial time.

To define the code-version of MSP, we need the possibility to decode sequences representing small numbers, i.e. numbers bounded by a length. So let

$$Decode(A,B) := \begin{cases} 0 & \text{if } A >_C C^{|\cdot|}(B) \\ Code^{-1}(A) & \text{else} \end{cases}.$$

But this function is p.t.c. since in the case where it has to be computed (i.e. if $A \leq_C C^{|\cdot|}(B)$), we have $Code^{-1}(A) \leq \Sigma(B)$ and this can be computed as

$$Code^{-1}(A) = \sum_{i=0}^{|\Sigma(B)|+1} Par(B,i) \cdot 2^i$$

where $Par(B,i) := Cut(A,i) \bmod 2$, and exponentiation can be used since $i \leq |\Sigma(B)| + 1$ and therefore 2^i can be replaced by $2^{\min(i, |2\Sigma(B)|)}$. Hence the function *Decode* is Σ_1^b-definable in S_2^1, and we can define

$$C^{MSP}(A,B) := \begin{cases} 0 & \text{if } B \geq_C C^{|\cdot|}(A) \\ Head(A, Decode(B,A)) & \text{else} \end{cases}.$$

3 Interpretation of S^0_{2+} in S_2

We shall now use the coding defined above to interpret S^0_{2+} in S_2. For a term t, the interpretation t^C is defined as follows:

- If t is 0 or a variable, then $t^C := t$.

- If t is $f(s)$, where $f \in \{|.|\, , \lfloor \frac{1}{2} . \rfloor\, , S\, , P\, , \mathit{Count}\}$, then $t^C := C^f(s^C)$.
- If t is $s_1 \circ s_2$, where $\circ \in \{+, \, , \dot{-}\, , \cdot, \#\, , MSP\}$, then $t^C := C^{\circ}(s_1^C, s_2^C)$.

For a formula φ, the interpretation φ^C is defined by:

- If φ is $s = t$ or $s \le t$, then $\varphi^C := s^C =_C t^C$ or $s^C \le_C t^C$ respectively.
- The interpretation commutes with the logical connectives as usual.
- If φ is $\exists x\, \psi$ or $\forall x\, \psi$, then $\varphi^C := \exists x\, (PSeq(x) \wedge \psi^C)$ or $\forall x\, (PSeq(x) \to \psi^C)$ respectively.
- If φ is $\exists x \le t\, \psi$ or $\forall x \le t\, \psi$, then φ^C is defined as $\exists x\, (PSeq(x) \wedge x \le_C t^C \;\to\; \psi^C)$ or $\forall x\, (PSeq(x) \wedge x \le_C t^C \wedge \psi^C)$ respectively.

Note that the interpretation of a bounded formula is not necessarily equivalent to a bounded formula. Nevertheless, the interpretation of a sharply bounded formula is equivalent to a bounded formula since we can prove

$$PSeq(x) \wedge x \le_C C^{|\cdot|}(t^C) \;\to\; x \le SqBd(|\Sigma(t^C)|, \Sigma(t^C)) \;\; .$$

Theorem 1 If $\varphi(a_1, \ldots, a_n)$ is provable in S^0_{2+}, then

$$PSeq(a_1) \wedge \ldots \wedge PSeq(a_n) \to \varphi^C(a_1, \ldots, a_n)$$

can be proved in S_2.

Proof: It suffices to prove the interpretations of the non-logical axioms of S^0_{2+} in S_2. The axioms from $BASIC$ and the additional axioms for the function symbols P, $\dot{-}$, $Count$ and MSP are all verified by long but straightforward computations. It remains to show that the interpretation of every instance of $\Sigma^b_0 - PIND$ can be proved, or more general

$$S_2 \vdash \; \varphi(0) \wedge \forall x\, (PSeq(x) \wedge \varphi(C^{\lfloor \frac{1}{2} \cdot \rfloor}(x)) \to \varphi(x)) \;\to\; \forall x\, (PSeq(x) \to \varphi(x))$$

where $\varphi(x)$ is a bounded formula. So suppose

$$\varphi(0) \wedge \forall x\, (PSeq(x) \wedge \varphi(C^{\lfloor \frac{1}{2} \cdot \rfloor}(x)) \to \varphi(x)) \;\; .$$

Suppose furthermore that $\exists x\, (PSeq(x) \wedge \neg\varphi(x))$. Then by MIN, which is provable in S_2 by Thm. 2.20 of [Bu], we have

$$\exists x\, (PSeq(x) \wedge \neg\varphi(x) \wedge \forall y {<} x (PSeq(y) \to \varphi(y))) \;\; .$$

Let this minimal x be a, then since $PSeq(a)$, we also have $PSeq(C^{\lfloor \frac{1}{2} \cdot \rfloor}(a))$, and $C^{\lfloor \frac{1}{2} \cdot \rfloor}(a) < a$, hence $\varphi(C^{\lfloor \frac{1}{2} \cdot \rfloor}(a))$, and the first assumption leads to a contradiction. □ □

Let $f(x)$ denote the function $\lfloor \frac{1}{3} x \rfloor$. f can be defined by the open formula

$$b = f(a) \;:\leftrightarrow\; 3b = a \vee 3b + 1 = a \vee 3b + 2 = a \;\; .$$

In T^0_2, integer division $\lfloor \frac{a}{b} \rfloor$ can be defined: to prove the existence, use the induction axiom for the quantifier free formula $b \cdot x > Sa$.

Theorem 2 $\forall x\, \exists y\, y = f(x)$ cannot be proved in S^0_{2+}.

Proof: Suppose $S^0_{2+} \vdash \forall x\, \exists y\, y = f(x)$, then by Theorem 1

$$S_2 \vdash \forall x\, PSeq(x) \to \exists y\, (PSeq(y) \wedge y = C^f(x)) \ .$$

Then by Parikh's Theorem it follows that there is a term $t(x)$ in the language of bounded arithmetic s.t. in particular

$$S_2 \vdash \exists y \leq t(a)\, (PSeq(y) \wedge y = C^f(\langle a+1 \rangle)) \ .$$

But $\langle a+1 \rangle = Code(2^{a+1} - 1)$, and one easily sees that $f(2^{a+1} - 1)$ is such that y must be of the form $y = \langle 1, \dots, 1 \rangle$ with a ones, hence $Len(y) = a$, so $y > 2^a$. $\Box$ $\Box$

Hence T^0_2 is not $\forall\exists\Sigma^b_0$-conservative over S^0_{2+}, and Parikhs Theorem immediately yields

Corollary 3 T^0_2 is not $\forall\Sigma^b_1$-conservative over S^0_{2+}.

We conjecture that for any number k that is not a power of two, S^0_{2+} cannot define the function $\lfloor \frac{1}{k} x \rfloor$. Clearly it would suffice to prove this for k an odd prime number.

S^0_{2+} and Circuit Complexity

AC^0 denotes the class of functions computable by uniform families of polynomial size constant depth unbounded fan-in circuits. In [Cl], Clote shows that it is reasonable to consider this class to be equal to Immerman's class FO, which is known to be equal to the alternating logarithmic time hierarchy LH (cf. [B-I-S]).

Theorem 4 *S^0_{2+} cannot Σ^b_1-define every function in AC^0.*

Proof: According to Clote [Cl], AC^0 is the smallest class containing the initial functions 0, $2x$, $2x+1$, projections, $|x|$, $\#$ and Bit and closed under composition and CRN, which is the following scheme: f is defined by CRN from g , h_0, , h_1 if $h_i(\underline{x}, y) \leq 1$ for $i = 0, 1$ and every $\underline{x}, y$, and

$$\begin{aligned} f(\underline{x}, 0) &= g(\underline{x}) \\ f(\underline{x}, 2y) &= 2f(\underline{x}, y) + h_0(\underline{x}, y) \quad \text{for } y > 0 \\ f(\underline{x}, 2y+1) &= 2f(\underline{x}, y) + h_1(\underline{x}, y) \ . \end{aligned}$$

Consider the function $f(x) := \lfloor \frac{1}{3} P(1\#x) \rfloor$. $P(1\#x) = 2^{|x|} - 1$ is the number of length $|x|$ where every bit is 1. Thus $f(x)$ is a number of length $|x| - 1$ with every second bit set 1, the remaining bits set 0.

Now this function f can be defined by CRN from $g(x) = 0$ and $h_0(x) = h_1(x) = |x| \bmod 2 = Bit(|x|, 0)$:

$$\begin{aligned} f(0) &:= 0 \\ f(x) &:= 2f(\lfloor \frac{1}{2} x \rfloor) + |\lfloor \frac{1}{2} x \rfloor| \bmod 2 \text{ for } x > 0 \end{aligned}$$

The same argument as in the proof of Theorem 2 shows that this function f cannot be Σ_1^b defined in S_{2+}^0, since for the crucial numbers b with $Code(b) = \langle a+1 \rangle$ for some a we have $f(b) = \lfloor \frac{1}{3} b \rfloor$. $\Box$ $\Box$

On the other hand, multiplication does not belong to AC^0 (cf. [Cl]). Hence the Σ_1^b-definable functions of S_{2+}^0 seem to correspond to no reasonable complexity class.

References

[B-I-S] Barrington, D. A. M., Immermann, N., Straubing, H.: On uniformity within NC^1. Journal of Computing and Systems Sciences **41**, 274–306 (1990)

[Bu] Buss, S. R.: Bounded arithmetic. Napoli: Bibliopolis 1986

[Cl] Clote, P.: Sequential, machine independent characterizations of the parallel complexity classes $ALogTIME$, AC^k, NC^k and NC. In: Buss, S. R., Scott, P. J. (eds.), Feasible Mathematics, pp. 49–69. Birkhäuser 1990

[K-P-T] Krajíček, J., Pudlák, P., Takeuti, G.: Bounded arithmetic and the polynomial hierarchy. Annals of Pure and Applied Logic **52**, 143–153 (1991)

[Ta] Takeuti, G.: Sharply bounded arithmetic and the function $a \dot{-} 1$. In: Logic and Computation, pp. 281–288. Providence: American Mathematical Society 1990