II. Introduction to
Bounded Arithmetic and Witnessing

Sam Buss, UCSD
sbuss@math.ucsd.edu

Prague, September 2009
Language of first-order theory of bounded includes:

\[
0, S, +, \cdot, \leq, |x| := \lceil \log_2(x + 1) \rceil, \lfloor \frac{1}{2}x \rfloor, x \# y := 2^{|x| \cdot |y|}.
\]

Sometimes also add all polynomial time functions and relations.

Axioms can include (among others):
(a) Defining (equational) axioms for functions and relations, “\text{BASIC}”.
(b) Restricted forms of induction.
Definition

A *bounded* quantifier is of the form \((\forall x \leq t)\) or \((\exists x \leq t)\). It is *sharply bounded* provided \(t\) has the form \(|s|\). A formula is *bounded* or *sharply bounded* provided all its quantifiers are bounded or sharply bounded (resp.).

Definition

\[ \Delta^b_0 = \Sigma^b_0 = \Pi^b_0: \text{Sharply bounded formulas} \]
\[ \Sigma^b_{i+1}: \text{Closure of } \Pi^b_i \text{ under existential bounded quantification and arbitrary sharply bounded quantification, modulo prenex operations.} \]
\[ \Pi^b_{i+1} \text{ is defined dually.} \]

\(\Sigma^b_i, \Pi^b_i\) define exactly the predicates at the \(i\)-th level of the polynomial hierarchy (PH), if \(i \geq 1\). Thus, \(\Sigma^b_1\) and \(\Pi^b_1\) define exactly the NP and coNP sets.
Let formulas $A$ be in $\Psi$, we have the following kinds of induction:

$\Psi$-IND: $A(0) \land (\forall x)(A(x) \rightarrow A(x + 1)) \rightarrow (\forall x)A(x)$.

$\Psi$-PIND: $A(0) \land (\forall x)(A(\lfloor \frac{1}{2}x \rfloor) \rightarrow A(x)) \rightarrow (\forall x)A(x)$.

$\Psi$-LIND: $A(0) \land (\forall x)(A(x) \rightarrow A(x + 1)) \rightarrow (\forall x)A(|x|)$.

**Definition (Fragments of bounded arithmetic, B’85)**

$S_i^2$: BASIC $+$ $\Sigma_i^b$-PIND.

$T_i^2$: BASIC $+$ $\Sigma_i^b$-IND.

$S_2 = \bigcup_i S_i^2$ and $T_2 = \bigcup_i T_i^2$.

Note $T_2$ is essentially $I\Delta_0 + \Omega_1$. [Parikh’71, Wilkie-Paris’87]
Theorem (B’85, B’90)

(a) $S_2^1 \subseteq T_2^1 \preceq_{\forall \Sigma^b_2} S_2^2 \subseteq T_2^2 \preceq_{\forall \Sigma^b_3} S_2^3 \subseteq \cdots$

(b) Thus, $S_2 = T_2$.

(c) $S_2^1 + \Sigma^b_i$-LIND equals $S_2^i$.

More axioms:

**Φ-MIN:** $(\exists x)A(x) \rightarrow (\exists x)(A(x) \land (\forall y < x)\neg A(y))$.

**Φ-LMIN:** $(\exists x)A(x) \rightarrow (\exists x)(A(x) \land (\forall y)(|y| < |x| \rightarrow \neg A(y)))$.

**Φ-replacement:**

$(\forall x \leq |t|)(\exists y \leq s)A(x, y) \rightarrow (\exists w)(\forall x \leq |t|)A(x, \beta(x, w))$.

**Φ-strong replacement:**

$(\exists w)(\forall x \leq |t|)[(\exists y \leq s)A(x, y) \leftrightarrow A(x, \beta(x, w))]$. 
\[ \Sigma^b_i \text{-IND} \iff \Pi^b_i \text{-IND} \iff \Sigma^b_i \text{-MIN} \iff \Pi^b_{i-1} \text{-MIN} \iff \Delta^b_{i+1} \text{-IND} \]
\[ \Sigma^b_i \text{-PIND} \iff \Pi^b_i \text{-PIND} \iff \Sigma^b_i \text{-LIND} \iff \Pi^b_i \text{-LIND} \]
\[ \Sigma^b_i \text{-LMIN} \iff (\Sigma^b_{i+1} \cap \Pi^b_{i+1}) \text{-PIND} \iff_1 \text{strong } \Sigma^b_i \text{-replacement} \]
\[ \Sigma^b_{i-1} \text{-IND} \]

\[ S^i_2 \preccurlyeq \forall \Sigma^b_i \ T^{i-1}_2 \quad S^i_2 \preccurlyeq_{\forall B(\Sigma^b_i)} T^{i-1}_2 + \Sigma^b_i \text{-replacement} \]

\[ \Sigma^b_i \text{-PIND} + \Sigma^b_{i+1} \text{-replacement} \implies \Sigma^b_i \text{-PIND} \implies \Sigma^b_i \text{-replacement} \]

**Open:** The exact relative strength of \( \Sigma^b_i \text{-replacement} \).
**Definition**

Let $R$ be a bounded theory. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is **provably total** in $R$ provided there is a formula $A_f(x, y)$ that defines the graph of $f$ such that $R \vdash (\forall x)(\exists ! y)A_f(x, y)$, with $A_f$ polynomial time computable.

**Definition**

$f$ is $\Sigma^b_i$-definable by $R$, provided there is a $\Sigma^b_i$-formula $A(x, y)$ such that

1. $R \vdash (\forall x)(\exists y \leq t)A(x, y)$ for some term $t$.
2. $R \vdash (\forall x)(A(x, y) \land A(x, z) \rightarrow y = z)$.
3. $A(x, y)$ defines the graph of $f$. 
Thm. Any $\Sigma^b_1$-definable function in $S^i_2$ or $T^i_2$ can be introduced conservatively into the language of the theory with its defining axiom, and be used freely in induction formulas.

**Theorem (B’85)**

1. $S^1_2$ can $\Sigma^b_1$-define every polynomial time function.
2. $S^i_2$ can $\Sigma^b_i$-define every function which is polynomial time computable with an oracle from $\Sigma^p_{i-1}$.

(The converse holds too.)

Hence, we can w.l.o.g. assume that all polynomial time functions are present in the language of bounded arithmetic.
Similar definitions and results hold for predicates.

**Definition**

A predicate $P$ is $\Delta^b_i$-definable in $R$ provided there are a $\Sigma^b_i$-formula $A$ and $\Pi^b_i$-formula $B$ which are $R$-provably equivalent and which define the predicate $P$.

**Theorem (B’85)**

*Every polynomial time predicate is $\Delta^b_1$-definable by $S^1_2$. Every predicate which is polynomial time computable with an oracle from $\Sigma^b_{i−1}$ is $\Delta^b_i$-definable in $S^i_2$.*

(Again, a converse holds.)

Thus, every polynomial time predicate can be conservatively introduced to $S^i_2$ or $T^i_2$ with its defining axioms, and used freely in induction axioms.
Witnessing Theorem for $S_2^i$

Theorem (Main Theorem for $S_2^i$, B’85)

Let $i \geq 1$. Suppose $f$ is $\Sigma^b_i$-defined by $S_2^i$. Then $f$ is computable in $P^{\Sigma^p_{i-1}}$, that is, in polynomial time with an oracle for $\Sigma^p_{i-1}$.

For $i = 1$, $f$ is in $P$, polynomial time computable.

This gives an exact characterization of the functions that are $\Sigma^b_i$-definable in $S_2^i$.

For $i = 1$, the $\Sigma^b_1$-definable functions of $S_2^1$ are precisely the polynomial computable functions.

Likewise, the $\Delta^b_1$-definable predicates of $S_2^1$ are precisely the predicates that are provably in $NP \cap coNP$.

Open: Give a more satisfactory account of the functions that are $\Sigma^b_1$-definable in $S_2^i$, $i > 1$. That is, of the provably total functions of these theories. (Note the uniqueness condition.)
We now start the proof of the Main Theorem.

**Proof idea:** Form a free-cut free proof, in which all formulas are in $\Sigma^b_i$. The free-cut free proof is then essentially an algorithm for the function $f$.

The proof is considerably simplified by working with *strict* $\Sigma^b_i$-formulas, denoted $s\Sigma^b_i$ for short. These are of the form:

$$(\exists x_1 \leq t_1)(\forall x_2 \leq t_2)\cdots (Qx_i \leq t_i)B(\vec{x})$$

where $B$ is sharply bounded, and the quantifiers alternate in type (and subformulas of these formulas).

**Thm.** $S^i_2$ can equivalently be formulated with $s\Sigma^b_i$-PIND, provided $-\,$ and MSP are added to the language.

**Proof idea:** Careful bootstrapping, plus use of replacement.
To prove the witnessing theorem, by free-cut elimination, it suffices to consider sequent calculus proofs in which every formula is an $s\Sigma_{i}^{b}$-formula (including, via pairing functions, the final, proved formula). Henceforth, fix $i > 0$.

**Definition**

Let $A(\vec{c})$ be $s\Sigma_{i}^{b}$. The predicate $Wit_{A}(\vec{c}, u)$ is defined so that

- If $A$ is $(\exists x \leq t)B(\vec{c}, x)$, $B \notin \Sigma_{i-1}^{b}$, then $Wit_{A}(\vec{c}, u)$ is the formula $u \leq t \land B(\vec{c}, u)$.
- If $A$ is in $\Pi_{i-1}^{b}$, then $Wit_{A}(\vec{c}, u)$ is just $A(\vec{c})$.

The following is trivial since we are working with strict formulas.

**Fact:** $A(\vec{c}) \leftrightarrow (\exists u)Wit_{A}(\vec{c}, u)$.

**Fact:** $Wit_{A}$ is a $\Pi_{i-1}^{b}$-formula (or $\Delta_{1}^{b}$, when $i = 1$.)
A cedent is a set of formulas. If $\Gamma$ and $\Delta$ are cedents, then $\Gamma \rightarrow \Delta$ is a sequent. Its meaning is that the conjunction of $\Gamma$ implies the disjunction of $\Delta$.

Letting $\Gamma = A_1, \ldots, A_k$, then $\text{Wit}_\Gamma(\vec{c}, u)$ is the statement:

$$\bigwedge_{i=1}^{k} \text{Wit}_{A_i}(\vec{c}, (u)_i).$$

For $\Delta = B_1, \ldots, B_\ell$, $\text{Wit}_\Delta(\vec{c}, u)$ is the statement

$$\bigvee_{j=1}^{\ell} ((u)_1 = j \land \text{Wit}_{B_j}(\vec{c}, (u)_2))$$

The notation $(u)_i$ means $\beta(i, u)$, the $i$-entry in the sequence coded by $u$. That is, $u = \langle u_1, \ldots, u_k \rangle$ in the first case, and $u = \langle u_1, u_2 \rangle$ in the second case.
Theorem (Witnessing Lemma)

If $\Gamma \rightarrow \Delta$ is an $S^i_2$-provable sequent of $s\Sigma^b_i$ formulas with free variables $\vec{c}$, then there is a function $f(\vec{c}, u)$ which is $\Sigma^b_i$-definable in $S^i_2$ and computable in polynomial time with an oracle for $\Sigma^b_{i-1}$ such that $S^i_2$ proves

$$Wit_\Gamma(\vec{c}, u) \rightarrow Wit_\Delta(\vec{c}, f(\vec{c}, u)).$$

The theorem is proved by induction on the number of lines in a free-cut free $S^i_2$-proof $P$ of $\Gamma \rightarrow \Delta$. The base cases are the equational axioms defining the symbols of the language. Since witnesses for $\Delta^b_0$-formulas are trivial, these cases are all trivial.

The induction step splits into cases depending on the last inference of the proof $P$. 
Case (1): Last inference is $\exists \leq$:right.

\[
\Gamma \rightarrow \Delta, A(\vec{c}, s) \\
\quad s \leq t, \Gamma \rightarrow \Delta, (\exists x \leq t)A(\vec{c}, x)
\]

The formula $A$ is $s\Pi_{i-1}^b$. The induction hypothesis gives a function $f$, which accepts witnesses for $\Gamma$ and produces a witness either making a formula in $\Delta$ true or making $A(\vec{c}, s)$ true. Modify $f$, so that in the latter case, it returns $\langle \ell, s \rangle$.

\[
g(\vec{c}, u) = \begin{cases} 
  f(\vec{c}, \text{cdr}(u)) & \text{if } (f(\vec{c}, \text{cdr}(u)))_1 < \ell \\
  \langle \ell, s(\vec{c}) \rangle & \text{if } (f(\vec{c}, \text{cdr}(u)))_1 = \ell.
\end{cases}
\]

(The “\text{cdr}” operation strips the first entry from a sequence.)
Case (2): Last inference is $\exists \leq$:

\[
\begin{array}{c}
b \leq t, A(\vec{c}, b), \Gamma \rightarrow \Delta \\
(\exists x \leq t) A(\vec{c}, x), \Gamma \rightarrow \Delta
\end{array}
\]

where $A$ is $s\Pi_{i-1}^b$ but not $s\Sigma_{i-1}^b$. Let $f$ be given by the induction hypothesis. Define $g$ by

\[
g(\vec{c}, u) = f(\vec{c}, (u)_1, \langle 0 \rangle \ast u)
\]

(The "\ast" operation is sequence concatenation.)
Case (2'): Last inference is $\exists \leq$ : left.

$$b \leq t, A(\vec{c}, b), \Gamma \rightarrow \Delta$$

$$\overline{(\exists x \leq t)A(\vec{c}, x), \Gamma \rightarrow \Delta}$$

where $A$ is $s\Pi_{i-2}^b$. Let $f$ be given by the induction hypothesis. Let $\mu_A(\vec{c})$ equal the least $x \leq t(\vec{c})$ such that $A(\vec{c}, x)$ is true, or equal $t + 1$ if no such $x$ exists.

Define $g$ as

$$g(\vec{c}, u) = f(\vec{c}, \mu_A(\vec{c}), \langle 0 \rangle \ast u).$$

Note that $\mu_A$ is computable in polynomial time with an oracle for $s\Sigma_{i-1}^b$.

A similar argument applies for $\forall \leq$: right inferences.
Case (3): Last inference is PIND.

\[
\frac{A(\lfloor \frac{1}{2} b \rfloor), \Gamma \rightarrow \Delta, A(b)}{A(0), \Gamma \rightarrow \Delta, A(t)}
\]

where \( A \in \Sigma^b_i \setminus \Sigma^b_{i-1} \). Let \( f \) be given by the induction hypothesis. Define

\[
h(\vec{c}, b, u) = \begin{cases} 
  h(\vec{c}, \lfloor \frac{1}{2} b \rfloor, u) & \text{if } (h(\vec{c}, \lfloor \frac{1}{2} b \rfloor, u))_1 < \ell \\
  f(\vec{c}, b, \langle (h(\vec{c}, \lfloor \frac{1}{2} b \rfloor, u))_2 \rangle \ast \text{cdr}(u)) & \text{otherwise}
\end{cases}
\]

and \( h(\vec{c}, 0, u) = \langle \ell, (u)_1 \rangle \). \( h \) can be defined by \textit{limited iteration on notation} and is polynomial time computable relative to \( f \). Here, \( \ell \) is the number of formulas in the antecedent.

Then set \( g(\vec{c}, u) = h(\vec{c}, t(\vec{c}), u) \). Q.E.D.
Corollary

If $R(x, y) \in \mathbb{P}$ and $S^1_2 \vdash (\forall x)(\exists y)R(x, y)$, then $R(x, y)$ is computable by some polynomial time function, provably in $S^1_2$. That is, for some $\Sigma^b_1$-defined, hence ptime, function $f$, $S^1_2 \vdash \forall xR(x, f(x))$.

Proof: Parikh’s theorem gives a polynomial bound on $y$ that is provable in $S^1_2$. Then, the corollary is immediate from the Witnessing Lemma.
Next we sketch the proof of the fact that $S_{2}^{i+1}$ is $\forall \Sigma_{i+1}$-conservative over $T_{2}^{i}$.

**Lemma**

$T_{2}^{i} \vdash \Pi_{i}^{b}$-IND.

**Proof.** Given $A(x)$ in $\Pi_{i}^{b}$, instead of using induction on $A(x)$ from $x = 0$ up to $x = t$, use induction on $\neg A(t \div x)$ with $t$ fixed.

**Lemma**

$T_{2}^{i} \vdash \Sigma_{i}^{b}$-minimization.

**Proof.** Suppose $(\exists x)A(x)$, but there is no least such $x$. Use induction on the $\Pi_{i}^{b}$-formula $(\forall x < a)\neg A(x)$ to get a contradiction.
Lemma

$T^i_2$ can $\Sigma^{b}_{i+1}$-define every function in $P\Sigma^b_i$.

**Proof.** (Idea.) Let $f$ be in $P\Sigma^b_i$. Without loss of generality, $f$ is computed using a “witness oracle” that when queried “$\exists x \leq t. A(x, n)$?” either returns a value for $x \leq t$ that makes $A$ true, or returns $t + 1$ indicating no such $x$ exists.

A consistent computation for $f$ is a computation based on a sequence of oracle answers such that any response $x \leq t$ does satisfy $A$ (but answers “$t + 1$” may be incorrect).

The property of being a consistent computation is $\Pi^{b}_{i-1}$. Order consistent computations lexicographically; $T^i_2$, via $\Sigma^b_i$-minimization, proves there exists a minimum consistent computation. And, that this consistent computation has all oracle answers correct. It is straightforward to check that the minimum consistent computation is $\Sigma^{b}_{i+1}$-definable.
**Theorem (B’90)**

\[ S_{2}^{i+1} \text{ is } \forall \Sigma_{i+1}^{b}-\text{conservative over } T_{2}^{i}. \]

**Proof. (Idea)** Repeat the proof of the Witnessing Lemma for \( S_{2}^{i+1} \), but now the conclusion is that \( T_{2}^{i} \) proves the witnessing sequent (instead of \( S_{2}^{i+1} \)):

\[
\text{Wit}_{\Gamma}(\vec{c}, u) \rightarrow \text{Wit}_{\Delta}(\vec{c}, f(\vec{c}, u)).
\]

It can be checked that \( T_{2}^{i} \) can formalize all the reasoning that was earlier formalized in \( S_{2}^{i+1} \).
A *Polynomial Local Search* PLS is formalized in $S^1_2$ provided its feasible set, initial point function, neighborhood function, and cost function are $\Sigma^b_1$-defined (as ptime functions).

**Theorem**

$T^1_2$ can prove that any (formalized) PLS problem is total.

**Proof:** By $\Sigma^b_1$-minimization, $T^1_2$ can prove there is a minimum cost value $c_0$ satisfying

$$(\exists s \leq b(x))(F(x, s) \land c(x, s) = c_0).$$

Choosing $s$ that realizes the cost $c_0$ gives either a solution to the PLS problem or a place where the PLS conditions are violated. □

**Open:** Can $T^1_2$ witness any PLS problem with a $\Sigma^b_1$-definable (single-valued) function?
Theorem (BK’94)

If $A \in \Sigma^b_1$ and $T_2^1 \vdash (\forall x)(\exists y)A(x, y)$, then there is a PLS problem $R$ such that $T_2^1$ proves

$$(\forall x)(\forall y)(R(x, y) \rightarrow A(x, (y)_1)).$$

If $A \in \Delta^b_1$, then can replace “$(y)_1$” with just “$y$”.

This gives an exact complexity characterization of the $\forall \Sigma^b_1$-definable functions of $T_2^1$, in terms of PLS-computability.
Theorem (Witnessing Lemma)

If $\Gamma \rightarrow \Delta$ is a $T_2^1$-provable sequent of $s\Sigma_1^b$ formulas with free variables $\vec{c}$, then there is a PLS problem $R(\langle \vec{c}, u \rangle, v)$ so that $T_2^1$ proves

$$Wit_\Gamma(\vec{c}, u) \land R(\langle \vec{c}, u \rangle, v) \rightarrow Wit_\Delta(\vec{c}, v).$$

Proof idea: Use a free-cut free $T_2^1$-proof, proceed by induction on number of inferences in the proof. Arguments are similar to what was used to prove the witnessing lemma for $S_2^i$ ($i = 1$ case). Most cases just require closure of PLS under polynomial time operations. However, induction ($\Sigma_1^b$-IND inference) now requires exponentially long iteration: this is handled via the exponentially many possible cost values.

The Theorem and Witnessing Lemma generalize to $i > 1$ with $PLS^{\Sigma_i^b\Sigma_{i-1}}$. The fourth talk will improve on this, however.
Some selected references