Total NP Functions II: Provability and Reducibility

Sam Buss (UCSD)
sbuss@math.ucsd.edu

Newton Institute, March 29, 2012

NP Search with Proofs of Totality

.

This talk focuses on classes of NP search problems based on provable totality in weak formal theories.

- *Bounded arithmetic* is often the main focus.
- *Constant depth propositional proofs*. Major goal: independence results for constant depth proofs. Is there a good depth hierarchy?

This talk:

- Bounded arithmetic theories — brief overview.
- Colored PLS.
- Game induction principles.
- Local improvement principles.

Theories of bounded arithmetic

Bounded Arithmetic refers to a collection of subtheories of Peano arithmetic, with severely restricted induction principles.

We work with first- and second-order theories, in the language of Cook's PV, with all polynomial time functions and relations. Adding induction for more formulas gives:

Theory	Induction formulas	Induction type
S_2^1	NP-predicates (Σ_1^b)	length (LIND)/polynomial (PIND)
T_2^1	NP-predicates (Σ_1^b)	successor (IND)
S_2^2	$\mathrm{NP}^{\mathrm{NP}}$-predicates (Σ_2^b)	length (LIND)/polynomial (PIND)
T_2^2	$\mathrm{NP}^{\mathrm{NP}}$-predicates (Σ_2^b)	successor (IND)
S_2^k	Σ_k^p-predicates (Σ_2^b)	length (LIND)/polynomial (PIND)
T_2^k	Σ_k^p-predicates (Σ_2^b)	successor (IND)
U_2^1	NEXPTIME $(\Sigma_1^{1,b})$	length (LIND)/polynomial (PIND)
V_2^1	NEXPTIME $(\Sigma_1^{1,b})$	successor (IND)

Theory	Function definition	Function class
S_2^1	Σ_1^b-definable	polynomial time (P)
$S_2^2 \& T_2^1$	Σ_1^b-definable	polynomial local search (PLS) [BK'94]
$S_2^2 \& T_2^1$	Σ_2^b-definable	P^{NP} [B'85]
$S_2^3 \& T_2^2$	Σ_1^b-definable	Colored PLS [KST'06]
$S_2^3 \& T_2^2$	Σ_2^b-definable	$\mathrm{PLS}^{\mathrm{NP}}$ [BK'94, BB'09]
$S_2^3 \& T_2^2$	Σ_3^b-definable	$\mathrm{P}^{\Sigma_2^b}$ [B'85]
$\vdots$	$\vdots$	$\vdots$
U_2^1	$\Sigma_1^{1,b}$-definable	PSPACE
V_2^1	$\Sigma_1^{1,b}$-definable	EXPTIME

Main theories for this talk:

Theory	Full induction for these predicates	Provably total NP search functions are computable in:
S_2^1	P	polynomial time (P)
T_2^1	NP	polynomial local search (PLS)
T_2^2	$\mathrm{NP}^{\mathrm{NP}}$	Colored PLS
U_2^1	PSPACE+	PSPACE
V_2^1	NEXPTIME	EXPTIME

Let R be one of the above theories. The *provably total NP search problems of* R, also called (projections of) "Σ_1^b-definable" functions, correspond to NP-search problems which are R-provably total by:

$$R \vdash \forall x \exists y \, \varphi(x, y)$$

where $\varphi(x, y)$ is a polynomial time predicate.

But, having $x \mapsto y$ computable in PSPACE, or EXPTIME is not very helpful in understanding the computational complexity of the NP search problem. We want to get more information from the fact that an NP search problem is *provably* total in U^1_2 or V^1_2.

What about PLS or Colored PLS?
How are they more computational?

Answer:
PLS corresponds to a search process, namely iterating N to find y.
Colored PLS corresponds to a two-stage search process: first iterating N to find y, then finding a color $e(y)$, then stepping back through the iteration of N to find an illegal color at $i(x)$.

Definition (Colored PLS [Krajíček-Skelley-Thapen'07)

Given input x and polynomial time computable functions that take x as a side parameter:

neighbor function $N(y) = N(x, y)$,
initial point $i = i(x)$,
color predicate $C(y, \alpha) = C(x, y, \alpha)$ for node y and color α,
and final node color assignment function $e(y) = e(x, y)$.

Goal: find a witness that one of the following is false:

- $\forall \alpha\, [\neg C(i, \alpha)]$.
- $\forall y\, \forall \alpha\, [N(y) < y \wedge C(N(y), e) \rightarrow C(y, e)]$.
- $\forall y\, [N(y) < y \vee C(y, e(y))]$.

Values of y decrease under N until reaching a leaf. The initial point has no color. Any color of a neighbor of y is a color of y. Every local minimum has a color.

Game Induction Principle

This idea for the "feasibility" of Colored PLS leads to a more general idea of k-round Game Induction principles, GI_k. [Skelley-Thapen'11] E.g., $k = 4$:

$$
\begin{array}{lccccccc}
G_1\colon & & x_1^1 & \to & x_2^1 & & x_3^1 & \to & x_4^1 \\
 & & \uparrow & & \downarrow & & \uparrow & & \downarrow \\
G_2\colon & & x_1^2 & & x_2^2 & & x_3^2 & & x_4^2 \\
 & & \uparrow & & \downarrow & & \uparrow & & \downarrow \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
 & & \uparrow & & \downarrow & & \uparrow & & \downarrow \\
G_a\colon & \to & x_1^a & & x_2^a & \to & x_3^a & & x_4^a
\end{array}
$$

G_i is the i-th instance of a two-player game.
Player A (resp. B) has a winning strategy for G_a (resp. G_1).
$\uparrow/\downarrow$'s indicate reductions between games. ....

Still with $k = 4$: Consider a formula

$$\forall x_1 \leq b \, \exists x_2 \leq b \, \forall x_3 \leq b \, \exists x_4 \leq b \, G(x_1, x_2, x_3, x_4).$$

This defines a game, also called G. Player B plays existential variables, Player A plays universal values.
B's goal is to make $G(x_1, x_2, x_3, x_4)$ true.

B has a winning strategy U for G:

$$\forall x_1 \leq b \, \forall x_3 \leq b \, G(x_1, U_2(x_1), x_3, U_4(x_1, x_3)).$$

G: $\quad x_1^1 \quad \rightarrow \quad x_2^1 \qquad x_3^1 \quad \rightarrow \quad x_4^1$

A has a winning strategy V for G:

$$\forall x_2 \leq b \, \forall x_4 \leq b \, \neg G(V_1, x_2, V_3(x_2), x_4).$$

G: $\quad \rightarrow \quad x_1^a \qquad x_2^a \quad \rightarrow \quad x_3^a \qquad x_4^a$

Game Reduction: Reducing G to G' via functions $F = \{f_i\}_i$:

$$\begin{array}{lcccc} G: & x_1 & x_2 & x_3 & x_4 \\ & f_1 \uparrow & f_2 \downarrow & f_3 \uparrow & f_4 \downarrow \\ G': & x'_1 & x'_2 & x'_3 & x'_4 \end{array}$$

"B **has a winning strategy for** G
$\Rightarrow$ B **has a winning strategy for** G'**:**"
is expressed by:

$$\forall x'_1, x_2, x'_3, x_4 [G(f_1(x'_1), x_2, f_3(x'_1, x_2, x'_3), x_4) \rightarrow G'(x'_1, f_2(x'_1, x_2), x'_3, f_4(x'_1, x_2, x'_3, x_4))].$$

B's strategy U' for G' is given by in terms of strategy U for G by:
$U'_2(x'_1) = f_2(x'_1, U_2(f_1(x'_1)))$ and
$U'_4(x'_1, x'_3) = f_4(x'_1, U_2(f_1(x'_1)), x'_3, U_4(f_1(x'_1), f_3(x'_1, U_2(f_1(x'_1)), x'_3)))$.

Definition (Game Induction Principle)

Fix $k > 0$. The GI_k search problem is given parameters a and b and x, a polynomial time predicate G that uniformly defines k-turn games G_i for $i = 1, \cdots a$, and polynomial time functions U, V, W where W uniformly computes W_i's. The value x is a side parameter for all of G, U, V, W.
A *solution* consists of values showing that

(a) U does not give winning strategy for B on G_1,

(b) V does not give winning strategy for A on G_a, or

(c) for some i, W_i does not give a game reduction from G_i to G_{i+1}.

GI_k is a total NP search problem.

GI_k defines a k-stage search process, similar to how Colored PLS defined a two-stage search process.

Theorem (Skelley-Thapen'11)

Let $k \geq 1$. T_2^k proves GI_k is total. Conversely, any provably total NP search problem of T_2^k is polynomial time many-one reducible to GI_k.

This also holds in the relativized setting.

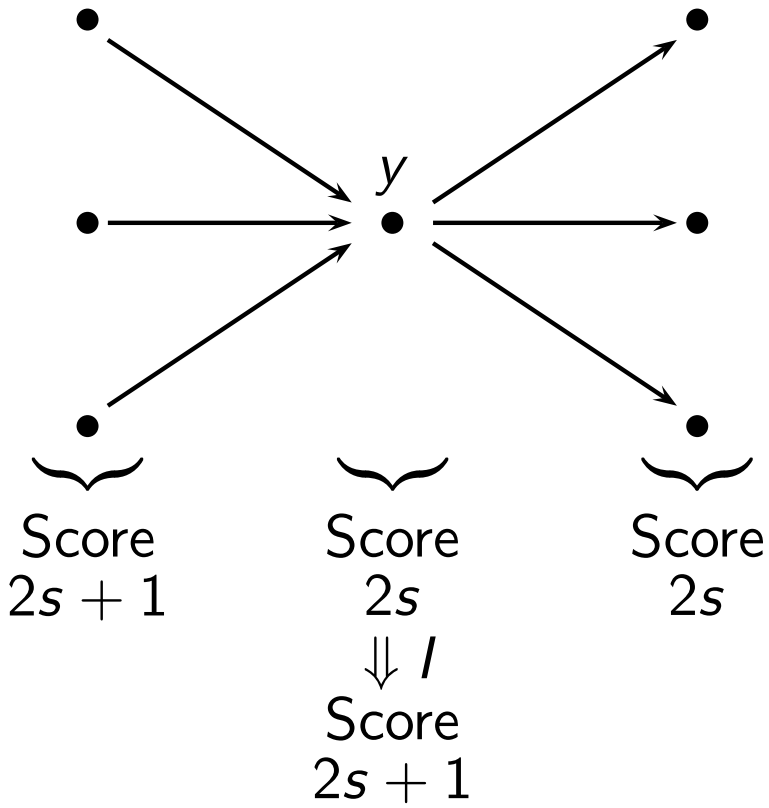
Local Improvement Properties

The **Local Improvement Properties** are motivated by trying to extend the GI_k principles to non-constant values for k. [Kołodziejczyk-Nguyen-Thapen'11,Beckmann-Buss' ??]

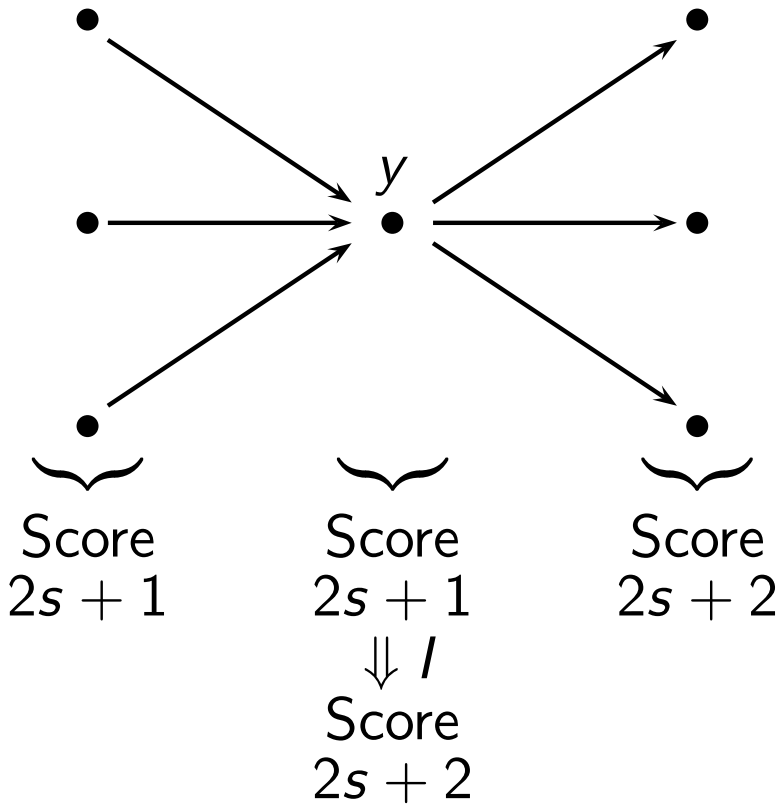
Intuition: An (implicitly defined) directed acyclic labeled graph G has bounded in-/out-degree. All nodes of G are initialized with labels of score $s = 0$. By sweeping forward in G, labels may be "locally improved" from even score $2s$ to score $2s + 1$. By sweeping backward in G, labels may be locally improved from odd score $2s + 1$ to score $2s + 2$. However, no label can have score $s > c$.

Solution: A solution is consists of showing values for which the local improvement process fails to correctly increase the score, or for which the score is $> c$.

Forward sweep improvement by I: (even to odd)



Reverse sweep improvement by /: (odd to even)



Formally, an instance **Local Improvement Principle** consists of the following, polynomial-time computable, items:

- A finite directed acyclic graph G with in-/out-degree $O(1)$. Acyclicity is enforced by node numbering.
- A neighborhood function, $N(y)$, giving the neighbors of node y in G.
- A initial labeling function $E(y)$ giving a score zero label to each node y.
- A scoring function $S(\ell)$ giving the score for label ℓ.
- An improvement function I which takes as input y, and labels for y and all nodes adjacent to y, and produces a new label for y.
- A predicate wf testing whether the labels on y and its neighbors are "well-formed".
- Upper bounds b and c on labels and scores.

A solution to the Local Improvement Principles consists of finding values (a) where the graph G is not well-formed, or (b) where the initialization $E(y)$ does not give score zero labels that are well-formed in a neighborhood, or (c) for a valid label with score $> c$, or (d) where the improvement function I does **not** give valid labels satisfying:

- If the predecessors of y have labels with score $2s+1$, and if y and its successors have labels with score $2s$, then I gives a new label for y with score $2s+1$.
- If the successors of y have labels with score $2s+2$, and if and y and its predecessors have labels with score $2s+1$, then I gives a new label for y with score $2s+2$.
- Replacing the label on y with the value given by I preserves the property that the distance-two neighborhood of y has well-formed labels. (I.e., in the neighborhoods of y and its neighbors.)

Local Improvement Principles

Definition (General Local Improvement)

Let $k \geq 1$. LI_k is the NP search problem as defined above with $c = k$ the bound on score values s.
LI is the local improvement search problem with no limitation on the value c.

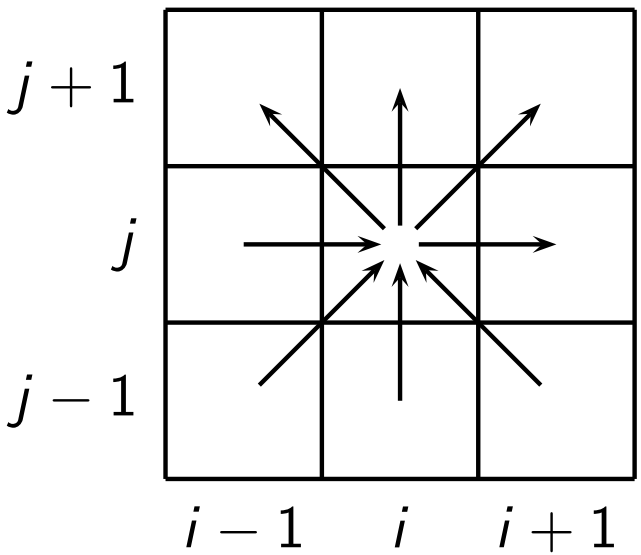
Definition (Linear Local Improvement)

If G is the linear directed graph on $\{0, \ldots, a\}$ with edges from i to $i+1$, then the local improvement condition is denoted LLI or LLI_k.

LLI_k corresponds to k-stages of iteration, analogous to GI_k.
LI_k generalizes to general dags instead of paths.

Definition (Rectangular Local Improvement)

If G is the rectangular directed graph on $\{0, \ldots, a\}^2$ with edges as shown below (omitting edges at boundary points as needed), then the local improvement condition is denoted RLI or RLI_k.



Relativized Theories

.

We often work with *relativized* theories of bounded arithmetic U_2^1 (corresponding to PSPACE computation) and V_2^1 (corresponding to EXPTIME computation).

The relativization means that there is a new second-order predicate symbol α added to the language. α is like an oracle: there are no axioms for α, except it may be used in induction formulas. The effect is that all computational classes are relativized to α; e.g., P^α, PSPACE^α, $\mathrm{EXPTIME}^\alpha$.

Σ_1^b-definable functions now correspond to relativized TFNP search problems.

Theorem (KNT'11)

The LLI_k *search problem is provably total in* T_2^k*. Furthermore, every NP search problem provably total in* T_2^k *is polynomial time, many-one reducible to* LLI_k*.*

Theorem (KNT'11)

The $\mathrm{LLI}_{\log}$ *search problem is provably total in* U_2^1*. Furthermore, every NP search problem provably total in* U_2^1 *is polynomial time, many-one reducible to* $\mathrm{LLI}_{\log}$*.*

Theorem (KNT'11)

The LI *search problem is provably total in* V_2^1*. Furthermore, every NP search problem provably total in* V_2^1 *is polynomial time, many-one reducible to* LI *and to* RLI*.*

Theorem (BB'??)

The LLI *search problem is provably total in* U^1_2*. (As before, every NP search problem provably total in* U^1_2 *is polynomial time, many-one reducible to* $\mathrm{LLI}_{\mathrm{log}}$ *and hence to* LLI*.)*

Theorem (BB'??)

(The LI*,* RLI *and hence* LI_1 *and* $\mathrm{RLI}_{\mathrm{log}}$ *search problems are provably total in* V^1_2*.) Furthermore, every NP search problem provably total in* V^1_2 *is polynomial time, many-one reducible to* LI_1 *and to* $\mathrm{RLI}_{\mathrm{log}}$*.*

In particular, the "log" versions of RLI and LLI are equivalent to the "non-log" versions. Also $\mathrm{LI}_1 \equiv \mathrm{LI}_{\mathrm{log}} \equiv \mathrm{LI}$.

Next: "new-style" witnessing theorems which help prove the second parts of the theorems.

Let $R \supseteq S^1_2$ be a theory of bounded arithmetic, and C the associated complexity class of functions.

Theorem (Old-Style Witnessing Theorem (Template))

Suppose R proves $(\forall x)(\exists y)\phi(x,y)$. Then there is a function $f \in C$ such that R proves $\forall x\, \phi(x, f(x))$.

Theorem (New-Style Witnessing Theorem (Template))

Suppose R proves $(\forall x)(\exists y)\phi(x,y)$. Then there is a function $f \in C$ such that S^1_2 proves

If there exists an encoding of a computation of $f(x)$

then it outputs a value y s.t. $\phi(x,y)$.

and such that R proves that

$\forall x$[*"There exists an encoding of a computation of $f(x)$"*]

Theorem (New-style witnessing for V_2^1, BB'??)

Suppose V_2^1 proves $(\exists y)\phi(y, \vec{a}, \vec{A})$ for ϕ a $\Sigma_0^{1,b}$-formula. Then there is an exponential time oracle Turing machine M such that S_2^1 proves "If Y encodes a complete computation of $M^{\vec{A}}(\vec{a})$, then $\phi(out(Y), \vec{a}, \vec{A})$ is true."

Corollary

Assume the above holds with ϕ a polynomial time predicate. Then there is a polynomial time procedure, which given oracle access to the complete computation Y of $M^{\vec{A}}(\vec{a})$ and $\vec{A}$ either produces the value y such that $\phi(y, \vec{a}, \vec{A})$ or finds a point where encoding Y of the computation contains a (local) mistake.

Proof of corollary uses the polynomial time witnessing theorem for S_2^1.

Very sketchy proof outline for V_2^1 and $\mathrm{RLI}_{\log}$: (based on KNT'11)

Suppose that a NP search problem is provably total (Σ_0^b-definable by $\exists y \phi(x, y).$) in V_2^1. The corollary is used to build an instance of the $\mathrm{RLI}_{\log}$ principle.

The first scan creates labels with score 1 encoding the entire EXPTIME computation Y, rectangularly arranged by time and tape position. This is done with local improvement based on Turing machine transition rules.

The polynomial time algorithm of the corollary makes polynomially many queries to Y. Each query is handled by a back-and-forth scan in the $RLI_{\log}$-instance. (This is a tricky part!) This gives the "log" bound on score.

No local mistake will be discovered in Y, thus any solution to the $\mathrm{RLI}_{\log}$-instance will give a value y such that $\phi(x, y)$ holds.

Proof Sketch: $U_2^1 \vdash$ "LLI is total"

For LLI, for a given input with $a = c$, the graph G is linear, of length a, and each node is repeatedly given new labels, for a total of c left-right/right-left scans. The intuition is that this should be enough to simulate an *exponential time* computation, since we can let the nodes of G label the a tape positions for a Turing machine, and let the labels of vertices of score s encode the exponential time Turing machine's configuration time s.

This would be done by initializing the nodes of G to have label with score zero, scanning left-to-right to give all nodes labels with score 1, then right-to-left to give all nodes labels with score 2, and repeat until scores reach value c. Once score c is surpassed, we must have found a solution to the local improvement principle.

Clearly this process, at least as defined, could simulate an exponential time computation.

The problem is that U_2^1 can formalize (and use induction on) polynomial space predicates, not exponential time predicates. So the previous slide's proof sketch is seemingly beyond the reasoning power of U_2^1.

To fix this, we assign labels to nodes of G using a *non-deterministic* PSPACE procedure instead of an exponential time procedure. The nondeterministic PSPACE procedure may not give a consistent wellformed set of labels, but this will not prevent us from proving the totality of the LI problem.

Furthermore, our argument can be carried out within U_2^1, since it is not hard to show that Savitch's theorem on the equivalence of PSPACE and NPSPACE can be carried out by U_2^1. [BB'??]

• The NSPACE algorithm works by alternately scanning left-to-right and right-to-left and setting values for labels using the improvement function I. At any given point, it "knows" well-formed values of the labels for the nodes $x-2, x-1, x, x+1, x+2$, and uses I to set the new label for x.

• The algorithm checks if the well-formedness still holds: if not, then a solution to the LI problem has been found. Also, if the new label has cost $> c$, a solution to the LI problem as been found.

• Otherwise, the algorithm needs to move one step, say rightward to node $x+1$. For this it "forgets" the label of $x-2$ and guesses the label for $x+3$. If the well-formedness property fails for the guessed label, the algorithm aborts (and does not have a solution to the LI-problem).

• If it does not abort, the algorithm proceeds to handle $x+1$.

• The obvious adjustments are made at the ends of the graph, and the algorithm reverses directions when it reaches the ends of the linear graph.

• We must avoid the case where the algorithm aborts without a solution.

• Suppose for contradiction, that all runs of the NSPACE machine abort in this way. Set s_0 to be maximum score which can be reached by any nondeterministic computation before aborting without a solution. We can prove s_0 exists, since it is definable by a NPSPACE (hence PSPACE) procedure.

Wlog, s_0 is odd, so the algorithm is scanning rightward.

• A computation is s_0 consistent at x if it uses the same score $s_0 - 1$ labels for vertices $x-2, x-1, x, x+1, x+2$ when setting labels of score $s_0 - 1$ as when setting labels of score s_0.

• Choose the maximum value x_0 such that there exists an NSPACE computation which is s_0-consistent at x_0.

• This computation is not forced to abort after setting the score s_0 label of x. This contradicts the choices of s_0 and x_0. QED

Penultimate slide

Other characterizations:

- Combinatorial principles [Pudlák'03]
- Π^b_k-PLS problems [Beckmann-Buss'09]
- Alternating Min/Max characterization [Pudlák-Thapen'12]

Final remark:
If one could give (relativized) separations of the NP search problems provably total in bounded arithmetic theories T^k_2, this would resolve the major open question about finding superquasipolynomial separations of constant-depth propositional proofs.

Thank you!