Princeton University Spring'00	COS 598g: Proof Complexity

Lecture 3: First-order theories (cntd.) and Bounded Arithmetic

Lecturer: *Alexander Razborov* Notes: *Sanjeev Arora*

In this lecture Sasha stated several more useful facts about first-order theories, defined Sam Buss's *bounded arithmetic*, a fragment of Peano Arithmetic that intends to capture the notion of "computationally feasible reasoning", and formulated its main theorem.

1 Preliminaries

Recall Gödel's completeness theorem.

Theorem 1 *In any first order theory, $A_1, A_2, \ldots, A_k \vdash B$ if and only if $A_1, A_2, \ldots, A_k \models B$.*

This shows an equivalence between a syntactic notion ("When can we prove B from $A_1, A_2, \ldots, A_k$?") and a semantic notion ("Is B valid in every model in which $A_1, A_2, \ldots, A_k$ are?").

In the sequel we will be sometimes interested in systems with infinite number of axioms. We can say something meaningful about this case as well.

Theorem 2 *Let $\mathcal{A}$ be an infinite set of axioms and B be any formula. Then*

1. *$\mathcal{A} \vdash B$ if and only if there is a finite $\mathcal{A}_0 \subseteq \mathcal{A}$ such that $\mathcal{A}_0 \vdash B$.*
2. *$\mathcal{A} \models B$ if and only if there is a finite $\mathcal{A}_0 \subseteq \mathcal{A}$ such that $\mathcal{A}_0 \models B$.*

Note that part 1 of the theorem is trivial since any proof is finite by definition. Part 2 is the compactness theorem, and will not be proved here. The compactness theorem has many important consequences, and the most immediate consequence is that Gödel's completeness theorem readily extends to the case of infinitely many formulas on the left-hand side. Here comes one example of a different nature.

Example 3 Consider the first-order theory of fields. If a sentence is valid for fields of characteristic 0, then it is valid for all fields of sufficiently large characteristic. To see this, note that sentences that are valid for characteristic 0 are the ones that follow

from the following infinite set of (1st order) axioms:
characteristic $\neq 2$,
characteristic $\neq 3$,
characteristic $\neq 5$,.. etc.

Here is another useful fact from logic.

Theorem 4 (Craig Interpolation Theorem) *Let L_0, L_1 be languages of 1st order logic and A, B respectively be formulae over these two languages. If $A \vdash B$ then there exists an interpolant C over the language $L_0 \cap L_1$ such that $\vdash A \to C$ and $\vdash C \to B$.*

Whenever T is a theory, we will use interchangeably $\vdash_T A$ and $T \vdash A$ for denoting the fact that A is a theorem of T.

Definition 5 *(*Conservative Extension*) Let L_0, L_1 be languages with $L_0 \subseteq L_1$. Let T_0, T_1 respectively be theories in these languages. We say T_1 is* conservative *over T_0 if any theorem of T_1 in the language L_0 can be proven in T_0.*

Example 6 Suppose T is a theory over language L. Suppose $\vdash_T \forall x \exists y A(x, y)$, where $Vars(A) = \{x, y\}$. Then $T \cup \{\forall x A(x, f(x))\}$ is conservative over T, where f is a new function symbol not appearing in L.

Here is a semantic (or model theoretic) proof of this fact. Suppose $T \cup \{\forall x A(x, f(x))\} \vdash B$, where B is a formula that does not contain f. We want to show that $T \vdash B$. This follows from the fact that in every model M of T, $\forall x \exists y A(x, y)$ is true. So M contains an interpretation of f such that $A(x, f(x))$ is true. Thus B is true in M.

Exercise 1: Give a syntactic proof of the fact given in the previous example. Sasha does not know of any way to do this without unraveling the proof of the Completeness theorem. In particular, some form of the axiom of choice seems to be needed.

There are two important cases, however, when such a syntactic proof *is* possible.

Example 7 (Definable function symbols) Let $\exists!$ be a shorthand for "exists and is unique", i.e.

$$\forall x \exists! y A(x, y) \qquad \equiv \qquad \forall x (\exists y A(x, y) \wedge \forall y_1, \forall y_2, A(x, y_1) \wedge A(x, y_2) \to y_1 = y_2).$$

If $\vdash_T \forall x \exists! y A(x, y)$ then we can efficiently translate every formula B over the language $L \cup \{f\}$ to a formula B^* over L such that:

- $T \cup \{\forall x A(x, f(x))\} \vdash (B^* \equiv B)$;
- every proof in $T \cup \{\forall x A(x, f(x))\}$ gets translated into a (slight modification) of a proof in T.

B^* is defined by induction on the logical complexity of B. For example,

$$(f(t) = y)^* \equiv \exists x((t = x)^* \wedge A(x, y)).$$

$\forall x A(x, f(x))$ is called the *defining axiom* for f, and the theory $T \cup \{\forall x A(x, f(x))\}$ is the *extension by definitions obtained by adding the function (or function symbol) f definable in T.*

Exercise 2: If $\vdash_T \exists y A(y)$, where $Vars(A) = \{y\}$, and c is a new constant added to the language then $T \cup \{A(c)\}$ is a conservative extension of T. Give syntactic proof.

Exercise 3: Give the definition and prove the conservativity result for extensions obtained by adding new *predicate* symbols.

2 Peano Arithmetic

To motivate Bounded Arithmetic, we first recall Peano Arithmetic (PA). The language of PA is $\{0, +, \cdot, S, \leq\}$. It has several open axioms like $x \cdot (y + z) = x \cdot y + x \cdot z$. It has a nonopen axiom: $x \neq 0 \rightarrow \exists y(S(y) = x)$. Then it has an infinite set of axioms (usually called the *induction scheme* of PA), one for each formula $A(x)$ (that *may* in general contain other free variables). The axiom is

$$A(0) \wedge \forall x(A(x) \rightarrow A(x + 1)) \rightarrow \forall x A(x).$$

PA is very powerful and is believed capable of proving all number theoretic facts (i.e., formulae in $Th(\mathbf{N})$) that do not pertain to the existence of infinite sets. To give a simple example, the exponentiation function is definable in PA.

The set of theorems of PA is undecidable. In fact, we can say something stronger.

Theorem 8 *PA can define all recursive functions and predicates. Namely, if $f : \mathbf{N}^k \mapsto \mathbf{N}$ is any recursive function, then there is a formula $A(x_1, x_2, \ldots, x_k, y)$ such that*

$$PA \vdash \forall \vec{x} \exists ! y A(\vec{x}, y)$$

and $\models_{\mathbf{N}} A(n_1, n_2, \ldots, n_k, f(n_1, n_2, \ldots, n_k))$ for all integers $n_1, \ldots, n_k$.

We will sketch the proof. Let M be any Turing machine computing f. Formula $A(x_1, \ldots, x_k, y)$ will "say" the following:

either M doesn't halt on $\vec{x}$ and $y = 0$ or M halts and outputs y.

How can PA reason about Turing machines? Note that a Turing machine is a simple manipulator of sequences. We can use Gödel numbering to encode a finite sequence of numbers by an integer[1] whereupon we can easily encode Turing machine tapes and computation histories. For doing that, we creat a conservative extension of PA by adding to it definable function/predicate symbols like $seq(x), len(x), (x)_i, x**y$. These are supposed to mean the following: $seq(x)$ is a predicate that says that x is a sequence, $len(x)$ is the length of x, $(x)_i$ is the ith element of x, and $x**y$ is the concatenation of x, y. If we stick to Gödel numbers, these are relatively simple arithmetic functions and predicates, and the axioms describing them are relatively simple to express in PA.

Then we prove in PA formulae like

$$seq(x) \wedge seq(y) \wedge (len(x) = len(y)) \wedge (\forall z < len(x)((x)_z = (y)_z)) \to x = y,$$

$$seq(x) \wedge seq(y) \to len(x**y) = len(x) + len(y),$$

etc.

After this, for any definable function f, we want to define the sequence $F(x) =< f(0), f(1), \ldots, f(x-1) >$. We introduce a function symbol F into the language of PA with the defining axiom

$$seq(F(x)) \wedge len(F(x)) = x \wedge \forall y < x((F(x))_y = f(y))$$

and use already proved properties of seg, len etc. to derive the existense and uniqueness conditions for it.

This completes the toolkit for reasoning about sequences. Now we can encode (in an obviously finite way) the working of M to construct the desired formula A.

3 Bounded Arithmetic

The language of Bounded Arithmetic is essentially the same as that of PA, with the following two additions.

New function symbols. The following are function symbols in BA (we give their intuitive meaning in parentheses):

1. $|x|$ (the bit length of x).
2. $\lfloor x/2 \rfloor$ (the largest integer $\leq x/2$).

[1] Thus a sequence $i_1, i_2, i_3, \ldots, i_k$ would be encoded by $p_1^{i_1} p_2^{i_2} \cdots p_k^{i_k}$, where $p_1, p_2, \ldots, p_k$ are the first k primes. What is important, however, is that we do not care about the exact form of the encoding. The only thing we demand is that PA proves several simple facts (described below) about it.

3. $x \sharp y$ ($2^{|x||y|}$).

Only the $\sharp$ adds something fundamentally new. We need it because in PA, for every term t, $t(n_1, n_2, \ldots, n_k)$ is a polynomial and hence has length that is linear in the bit length of $n_1, n_2, \ldots, n_k$. Adding $\#$ allows us to write terms which talk about numbers whose bit size is polynomial in the input size.

Bounded quantifiers. $(\forall x \leq t)$ denotes universal quantification among numbers up to a certain limit expressed by the term t. That is, $(\forall x \leq t)A(x)$ is the abbreviation for $\forall x(x \leq t \rightarrow A(x))$.

We similarly define $(\exists x \leq t)$, as well as *sharply bounded quantifiers*, which are of the form $(\forall x \leq |t|), (\exists x \leq |t|)$.

Now we define a hierarchy of formulae, $\cup_{i \geq 0} \Sigma_i^b \cup \Pi_i^b$. For simplicity we do it only for formulae that do not contain $\{\rightarrow, \equiv\}$, and, moreover, the negation $\neg$ occurs only at atomic formulae (every formula can be easily translated into this form: exercise!)

Definition 9 Σ_0^b, Π_0^b *are the set of all formulae with only propositional connectives and sharply bounded quantifiers.*

Definition 10 **(Σ_i^b, Π_i^b)** Σ_i^b *is the smallest class of formulae that contains* $\Sigma_{i-1}^b, \Pi_{i-1}^b$, *and is closed under* $\wedge, \vee$, *sharply bounded quantifiers, and* $(\exists x \leq t)$.

Π_i^b *is defined similarly in an inductive fashion except we use* $(\forall x \leq t)$ *instead of* $(\exists x \leq t)$.

Now we are finally ready to define Buss's theory S_2^1. It has 32 open axioms (PA axioms plus axioms about the new function symbols such as $|x\#y| = |x| \cdot |y| + 1$.)

It has an infinite set of induction axioms called $\Sigma_1^b - PIND$. For each formula $A \in \Sigma_1^b$, it has an axiom called

$$A(0) \wedge \forall x(A(\lfloor x/2 \rfloor) \rightarrow A(x)) \rightarrow \forall x A(x).$$

(Aside: the "1" in S_2^1 stands for the fact that induction is allowed only on Σ_1^b-formulas, and the "2" stands for the use of the $\#$ function which is the second function in a hierarchy $\#_1, \#_2, \#_3, \ldots$. Here $\#_1$ stands for multiplication, $x \#_3 y = 2^{2^{||x|| \cdot ||y||}}$ etc.)

Next time we will prove Buss's main theorem.

Theorem 11 1. *For any polytime computable function* f, *there is a* Σ_1^b*-formula* $A(x, y)$ *such that*

$$S_2^1 \vdash \forall x \exists ! y A(x, y) \qquad \textit{and} \qquad \models_{\mathbf{N}} \forall n A(n, f(n)).$$

2. *Suppose that* $S_2^1 \vdash \forall x \exists y A(x,y), A \in \Sigma_1^b$. *Then there exists another* $B(x,y)$ *such that*

$$S_2^1 \vdash B \to A \qquad \textit{and} \qquad S_2^1 \vdash \forall x \exists! y B(x,y),$$

and furtheremore, there is a polynomial-time computable function f *such that*

$$\models_{\mathbf{N}} \forall n B(n, f(n)).$$