Notes for Nov. 22

Jason Chen

Convention. The spaces we are working with here all have the form of (at most countable) products where each factor is either ω or ω^ω. Sometimes when things are really clear, we might also consider factors other than ω or ω^ω.

A pointset is simply a subset of the spaces under discussion. A pointclass is a collection of pointsets.

Following historical baggage, we tend to use $f, g, x, y, \alpha, \beta$ for variables ranging over ω^ω or whatever else that gets called a "real number" by logicians. And m, n, j, k, i are for variables ranging over natural numbers. Although it occurs rarely[1], when the context is clear, f, g might also range over functions or maps, and α, β over ordinals. (You're welcome, and no, I am not sorry.)

1 Normal forms of Σ^1_1 via Luzin's arithemtical example

Definition 1. A pointset A is in the pointclass Σ^1_1 (for short, A is Σ^1_1) if it can be defined by $A(x) \Leftrightarrow \exists y \forall n R(x, y, n)$, where R is a computable relation.

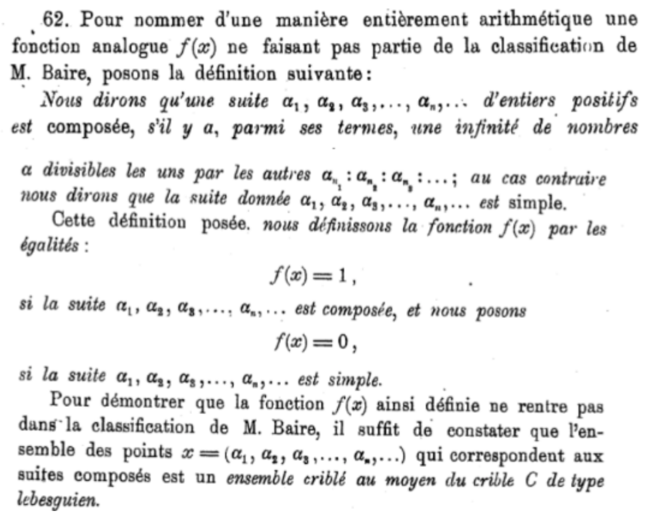
Example 2 (Luzin, 1927)**.** Consider the space $(\omega \setminus \{0\})^\omega$. This is the space of sequences of positive integers. Define a subset A of the space as follows:

$$A(x) \Leftrightarrow \exists n_0 < n_1 < n_2 < \ldots x(n_i) \text{ divides } x(n_{i+1})$$

In other words, $x \in A$ iff there is some increasing $y \in (\omega \setminus \{0\})^\omega$ such that for all $i \in \omega$, we have $x(y(i))$ divides $x(y(i+1))$. This is Σ^1_1, because the relation "$y(m) > y(m+1) \wedge x(y(n)) \mid x(y(n+1))$", with free variables (x, y, m, n), is computable.

The example of the set A is meant to illustrate how we obtain normal forms for Σ^1_1 sets. Intuitively, how would you verify membership in A?

[1]by this I mean the context is rarely clear



62. Pour nommer d'une manière entièrement arithmétique une fonction analogue $f(x)$ ne faisant pas partie de la classification de M. Baire, posons la définition suivante:

Nous dirons qu'une suite $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n, \ldots$ *d'entiers positifs est* composée, *s'il y a, parmi ses termes, une infinité de nombres* α *divisibles les uns par les autres* $\alpha_{n_1} : \alpha_{n_2} : \alpha_{n_3} : \ldots$; *au cas contraire nous dirons que la suite donnée* $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n, \ldots$ *est* simple.

Cette définition posée. *nous définissons la fonction* $f(x)$ *par les égalités* :

$$f(x) = 1,$$

si la suite $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n, \ldots$ *est composée, et nous posons*

$$f(x) = 0,$$

si la suite $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n, \ldots$ *est simple.*

Pour démontrer que la fonction $f(x)$ ainsi définie ne rentre pas dans la classification de M. Baire, il suffit de constater que l'ensemble des points $x = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n, \ldots)$ qui correspondent aux suites composés est un *ensemble criblé au moyen du crible C de type lebesguien.*

Figure 1: Luzin, Sur les ensembles analytiques

Well, A consists of sequences where you can find an infinite subsequence of progressively divisible integers (it's just slick way of saying each intger divides the next). Given a sequence f of positive integers, imagine we are rolling out f in front of us, examining longer and longer initial segments, and finding whether these finite segments have progressively divisible subsequences.

This amounts to running some kind of program, in pseudocode (notice "places to be checked" has the same length as the segment; this is just making it pretty so it's easier to define the tree next):

Algorithm 1 Finding progressive divisibility subsequence

```
Input: segment to be checked: $\langle n_0, ..., n_i \rangle$, places to check: $\langle m_0, ...m_i \rangle$
  if $m_0, ..., m_i$ is not increasing then
      halt saying no
  end if
  $j \leftarrow 0$
  while $j \leq i$ do
      if for some $k \leq j$, an $m_k$ is greater than $i$ then
          halt, do nothing        ▷ This is for when we inquire about a place beyond our input
      else if $n_{m_0} \mid n_{m_1}, n_{m_1} \mid n_{m_2}, ..., n_{m_{j-1}} \mid n_{m_j}$ then
          $j \leftarrow j + 1$, go back to the start of the while loop
      else halt saying no
      end if
      $j \leftarrow j + 1$
  end while
  Halt saying yes
```

A casual examination of the program above shows that it can have 3 possible outcomes:

1. it halts saying nothing, in which case we've asked for a place beyond the length of the input, resulting in undefined.

2. it halts saying yes, in which case we succeed in finding a finite subsequence that's progressively divisible.
3. it halts saying no. This is either because the "places to check" is not increasing (we're not checking progressively), or because no such finite subsequence can be found

If you think about it, the only way a sequence f fails to get into A is if it has no infinite subsequence that's progressively divisible. In other words, every attempt to pick out progressively divisible subsequences will be rejected by the program above in finite time.

So if we eliminate these losers, we end up getting a tree coding the verification of membership in A.

Proposition 3. There is a tree T on $\omega \times \omega$ satisfying: for all $f \in (\omega \setminus \{0\})^\omega$, $f \in A$ if and only if there is some $g \in (\omega \setminus \{0\})^\omega$ such that for all n, $(f \restriction n, g \restriction n) \in T$

Remark. In descriptive set theory, a tree on ω is a subset of finite sequences of ω that's closed under initial segments. Simiarly a tree T on $\omega \times \omega$ is a subset of $\omega^{<\omega} \times \omega^{<\omega}$ such that if $(s, t) \in T$, then length$(s) =$ length(t) and if s' is an initial segment of s and t' is an initial segment of t' with the same length as s', then $(s', t') \in T$.

Proof. T will be the tree that has "eliminated the losers". That is,

$$T := \{(s, t) \mid \text{ the program running Algorithm 1 does not halt saying no on input } (s, t)\}$$

If $f \in A$, then fix an increasing g picking out an infinite progressively divisible subsequence of f. Claim: for each n, the pair $(f \restriction n, g \restriction n)$ will not be rejected by the program. This is because the only way for the program to halt saying no on $(f \restriction n, g \restriction n)$ is if $f(g(0)), f(g(1)), f(g(2)), ...f(g(n-1))$ is not a progressively divisible sequence, or if $g \restriction n$ is not increasing. Either way, this will contradict the choice of g.

Conversely, let $g \in (\omega \setminus \{0\})^\omega$ be such that for all n, $(f \restriction n, g \restriction n) \in T$. First of all, it's obvious that g must be increasing. Second, observe the following: whatever $g(k)$ is, there will be a run of the program when n gets large enough so that $f \restriction n$ has the $g(k)$th place. In other words, previously undefined behavior will eventually become defined on large enough n.

Therefore, the assumption that the program running Algorithm 1 does not halt saying no on input $(f \restriction n, g \restriction n)$ for each n can only mean that $\langle f(g(k)) : k < n\rangle$ is a progressively divisible subsequence for each n, but that means (since f, g are fixed) that $\langle f(g(k)) : k < \omega\rangle$ is such a subsequence of f. □

Remark. In fact, Luzin's set A is an example of what's called a complete-Σ^1_1 set. It's like NP-complete, in that every Σ^1_1 set can be continuously (in fact, computably) reduced to it. But we won't say more about that here.

In the proof above, we only used the fact that "$y(m) > y(m+1) \wedge x(y(n)) \mid x(y(n+1))$" is a computable relation. Abstracting away from talks of divisibility and finding sequences, this proof provides the following normal form for Σ^1_1 sets.

Theorem 4. Let A be a pointset, say A is a subset of ω^ω for simplicity. The following are equivalent

1. A is Σ^1_1.
2. There is some computable tree on $\omega \times \omega$ such that $A = p[T]$. That is, $f \in A$ if and only if there is some $g \in \omega^\omega$ such that for all n, $(f \upharpoonright n, g \upharpoonright n) \in T$.
3. There is a computable assignment $f \mapsto T_f$ from reals to trees, such that $f \in A$ if and only if T_f has an infinite branch.

Proof Sketch. $1 \Rightarrow 2$: for this, modify the definition of T in the last proof. Define instead

$$T := \{(s,t) \mid \text{length}(s) = \text{length}(t) \wedge \forall n < \text{length}(t)$$
$$\text{the program deciding } R(x,y,n) \text{ has not halted rejecting } (s,t) \text{ in length}(t) \text{ many steps}\}$$

$2 \Rightarrow 3$: Given a computable tree T on $\omega \times \omega$, the assignment mapping y to its projection T_y (T_y is a tree on ω) is computable. To compute it: given any finite initial segment of y, search through T in the second coordinate to confirm or deny whether it's the t of any $(s,t) \in T$.

$3 \Rightarrow 1$: say the assignment is computed by the e^{th} Turing machine. Then saying "$A(f) \Leftrightarrow \exists y$ the tree computed by the e^{th} Turing machine on input f has a path that can be traced by y (i.e., $\forall n\ y \upharpoonright n \in T_f$)" puts A in Σ^1_1 form. □

The following facts are also quite useful. Given what we've seen, proving them is just a matter of using de Morgan's law, taking complements, and basic propositional logic.

Definition 5. A pointset is Π^1_1 iff it's the complement of a Σ^1_1 set.

Theorem 6. Let A be a pointset, say A is a subset of ω^ω for simplicity. The following are equivalent

1. A is Π^1_1.
2. There is some computable tree on $\omega \times \omega$ satisfying: for all f, $f \in A$ if and only if for all $g \in \omega^\omega$, there is some n with $(f \upharpoonright n, g \upharpoonright n) \notin T$.
3. There is a computable assignment $f \mapsto T_f$ from reals to trees, such that $f \in A$ if and only if T_f has no infinite branch (we say T_f is well-founded in this case).

The preceding theorem provides a useful object: a Π^1_1-complete set.

Definition 7. The set WF is the subset of ω^ω that are codes of well-founded relations. This means that the elements of WF are all $f : \omega \to \{0,1\}$ such that (for a fixed-in-advance computable encoding/pairing function $p : \omega \leftrightarrow \omega \times \omega$) the subset of natural numbers having characteristic function f encodes a well-founded relation on the natural numbers.

Theorem 8. For every Π^1_1 set $A \subseteq \mathcal{X}$, there is a computable function $F : \mathcal{X} \to \omega^\omega$ such that $A = F^{-1}[WF]$. (This is like NP-complete with polynomial-time functions).

Proof Sketch. Use the computable assignment $f \mapsto T_f$ above. So $f \in A$ iff T_f is well-founded. Let $F(f) =$ the unique element g in ω^ω, such that (where p is the computable pairing function on naturals) $g(p^{-1}(m,n)) = \begin{cases} 1 & \text{if } t_m, t_n \in T_f \wedge t_m \prec t_n \\ 0 & \text{otherwise} \end{cases}$

Here, $\{t_m \mid m \in \omega\}$ is a computable encoding of $\omega^{<\omega}$(i.e., nodes on a tree). □

2 Norms and Prewellorderings

Recall:

Definition 9. Let Γ be a pointclass. $\varphi : A \to \kappa$ a norm on some pointset A. We say φ is a Γ-norm iff there are relations $P(x, y)$ in Γ and $Q(x, y)$ in $\breve{\Gamma}$ such that for all $y \in A$ and for all x, the following are equivalent:

1. $x \in A$ and $\varphi(x) \leq \varphi(y)$
2. $P(x, y)$
3. $Q(x, y)$

In many texts, $P(x, y)$ is written as $\leq^{\varphi}_{\Gamma}$ instead and $Q(x, y)$ as $\leq^{\varphi}_{\breve{\Gamma}}$. Since in actual arguments these symbols are introduced via existential instantiation, whatever symbols we choose use is just a matter of stylistics.

Definition 10. Given a pointclass Γ, Prewellordering(Γ) is the statement "every Γ-set A has a Γ-norm".

Definition 11. Given pairs of sets $(A, B), (A^*, B^*)$, we say (A^*, B^*) *reduces* (A, B) iff $A^* \cap B^* = \emptyset \wedge A^* \cup B^* = A \cup B$.

We say a pointclass Γ has the reduction property iff every pair of Γ-sets can be reduced by a pair of Γ-sets.

At the moment we are interested in getting the reduction property from the prewellordering property. But let's first see a prototypical case:

Theorem 12. The class of recursively enumerable subsets of ω has the reduction property.

Proof. We appeal to the Church-Turing thesis freely. Let A, B be recursively enumerable. We now describe an enumeration procedure for sets A^*, B^*.

Begin by enumerating A and B on the side alternately (that is, run a separate program, such that, on the even steps of that computation, enumerate A, on the odd steps B). On even steps, if we find an element of A, and this element has not been found in B yet, put it in A^*; if this element has been found in B, then ignore and move on.

Similarly, on an odd step that an element of B is found which is not yet found to be in A, then put it in B^*; if this element has been found in A, then ignore and move on. Don't put anything else in A^* or B^*.

CLaim: (A^*, B^*) reduces (A, B). Proof: obviously, $A^* \subseteq A$ and $B^* \subseteq B$ and $A^* \cap B^* = \emptyset$. To see $A \cup B \subseteq A^* \cup B^*$, we notice that the separate program we run must enumerate every element of $A \cup B$ since it alternates between A and B and both are recursively enumerable. Every first encounter with an element in A or B, we will put it either A^* or B^* depending on where we first find it. □

The preceding proof is not hard. The key point is we have some enumeration procedure of the sets in question and the ability to look back on such a procedure to check if anything has been enumerated up to this moment. Norms and prewellorderings are the abstract tools for that.

Theorem 13. For any pointclass Γ, if Prewellordering(Γ), then Γ has the reduction property.

Lemma 14. Let Γ be an adequate pointclass and let C be a Γ set. If φ is a Γ-norm on C, then the following relations are also in Γ:

$$x \leq^*_\varphi y \Leftrightarrow_{df} x \in C \wedge (y \in C \to \varphi(x) \leq \varphi(y))$$
$$x <^*_\varphi y \Leftrightarrow_{df} x \in C \wedge (y \in C \to \varphi(x) < \varphi(y))$$

Proof of Lemma. Simply observe that

$$x \leq^*_\varphi y \leftrightarrow x \in C \wedge (x \leq^\varphi_\Gamma y \vee y \not\leq^\varphi_{\breve{\Gamma}} x)$$
$$x <^*_\varphi y \leftrightarrow x \in C \wedge y \not\leq^\varphi_{\breve{\Gamma}} x$$

□

Proof of Theorem. By the lemma, the relations $\leq^*_\varphi$ and $<^*_\varphi$ are also in Γ. Now suppose $A, B \subseteq \mathcal{X}$ are Γ sets in the space $\mathcal{X}$. We want to find A^*, B^* that reduce them. The idea is to abstractly simulate the computer programs above, using the norm and these two relations.

Define a pointset $C \subseteq \mathcal{X} \times \omega$ by (this is like the alternating enumeration)

$$C(x,n) \Leftrightarrow ((A(x) \wedge n = 0) \vee (B(x) \wedge n = 1))$$

By the closure properties of adequate Γ ($n = 0, n = 1$ are both recursive), the set C thus defined is also a Γ set. So by Prewellordering(Γ), it has a Γ-norm φ.

Define A^*, B^* as

$$A^*(x) \Leftrightarrow (x,0) \leq^*_\varphi (x,1)$$
$$B^*(x) \Leftrightarrow (x,1) <^*_\varphi (x,0)$$

(This is like checking if an element is first found in A or in B)

By the lemma, A^* and B^* are Γ sets. It is obvious from the definitions that $A^* \subseteq A, B^* \subseteq B$. And $A^* \cap B^* = \emptyset$ because for those $(x,n) \in C$, the conditions $(x,0) \leq^*_\varphi (x,1)$ and $(x,1) <^*_\varphi (x,0)$ are mutually exclusive (this is just a definition chase).

To see $A \cup B \subseteq A^* \cup B^*$: let's say $x \in A$, on the one hand, if $x \notin B$, then $(x,1) \notin C$, and hence $(x,0) \leq^*_\varphi (x,1)$ holds vacuously (expand the full definition in the statement of Lemma 14 to see this), and so $A^*(x)$ holds. On the other hand, if $x \in B$, then both $(x,0)$ and $(x,1)$ are in C. Now we can compare the ranking between $(x,0)$ and $(x,1)$ given by the Γ-norm φ on C. If $(x,1) <^*_\varphi (x,0)$ then x falls into B^*, and if $(x,0) \leq^*_\varphi (x,1)$, then x falls into A^* instead.

Exchanging A and B in the argument above finishes the proof. □

Theorem 15. If Γ is adequate, then Prewellordering(Γ) implies Prewellordering($\underset{\sim}{\Gamma}$).

Proof. Say A is a $\underset{\sim}{\Gamma}$ set in the space $\mathcal{X}$, this means that there is some Γ set B in the space $\mathcal{X} \times \omega^\omega$ and $\alpha_0 \in \omega^\omega$ such that $A = \{\alpha \mid (\alpha, \alpha_0) \in B\}$. By Prewellordering($\Gamma$), fix a Γ-norm $\bar{\varphi}(x,y)$ on B, and define, for $\alpha \in A$, $\psi(\alpha) = \bar{\varphi}(\alpha, \alpha_0)$.

It might be the case that ψ skips over some ordinals (so it's not really onto a ordinal). But that's okay, we can always define a surjective $\varphi : A \to \kappa$ by setting $\varphi(\alpha) = \text{otp}\{\psi(\beta) \mid \beta < \alpha\}$. Finally, check that φ inherits all the relevant properties from $\bar{\varphi}$, making it a $\underset{\sim}{\Gamma}$-norm (this step involves writing strings of iffs). □