MATH230: Tutorial Eight Solutions

Peano Arithmetic & Recursive Functions

Key ideas and learning outcomes
- Further understanding of the expressibility of PA,
- Practice writing natural deductions in PA,
- Recognise when procedures are primitive recursive,
- Informally describe procedures with primitive recursive steps.

Relevant notes: Peano and Primitive Recursion Slides
Relevant reading: *Computability Theory* Enderton, Chapter 2.

Hand in exercises: 2b, 3a, 3f, 3g, 7
Due following Friday @ 5pm to the tutor, or to lecturer.

Discussion Questions

1. The first-order language of Peano Arithmetic is often presented with an extra binary relation symbol $<$ where $x < y$ is given the usual interpretation: x is strictly less than y. In fact it is not necessary to add anything extra, for this relation can be defined using a sentence in PA as stated.

   Write down a wff in PA which, when interpreted in the standard model, defines the binary relation $<$ of being "strictly less than". Use this to write down formulae that represent: less than or equal to, strictly greater than, and greater than or equal to.

   Solution: There are a number of ways to do this. Here is one:

   $$x < y :\equiv \exists z\ (x + s(z) = y)$$

   Following on from this we can make the following definitions:

   (a) $x \geq y :\equiv \neg(x < y)$

   (b) $x > y :\equiv (x \geq y) \wedge \neg(x = y)$

   (c) $x \leq y :\equiv \neg(x > y)$

   It might be better to define $x > y$ directly instead of via $x < y$.

   $$x > y :\equiv \exists z\ (y + s(z) = x)$$

2. If $P(\mathbf{x})$ and $Q(\mathbf{x})$ are two primitive recursive n-ary predicates, then show that the predicate $P(\mathbf{x}) \to Q(\mathbf{x})$ is primitive recursive.

   Solution: The characteristic function of a compound propositional formula can be calculated from the characteristic function of the atomic formulae from which it's built.

   This follows from the work we did with valuations in the topic on propositional logic. There we saw, for example, that for some valuation v

   $$v(P \to Q) = 1 - v(P) + v(P)v(Q)$$

   If we swap the valuation for the characteristic functions χ_P and χ_Q we obtain the formula

   $$\chi_{P \to Q}(x) = 1 - \chi_P(x) + \chi_P(x)\chi_Q(x)$$

If the predicates P, Q are primitive recursive, then the formula above is obtained from the iterated compositions of primitive recursive functions.

$$\begin{aligned}\chi_{P\to Q}(x) &= 1 - \chi_P(x) + \chi_P(x)\chi_Q(x) \\ &= +(1 - \chi_P(x), \chi_P(x)\chi_Q(x)) \\ &= +(-(C_1(x), \chi_P(x)), \times(\chi_P(x), \chi_Q(x)))\end{aligned}$$

3. Discuss whether the search for roots to a given polynomial equation is a primitive recursive procedure.

   Solution: Evaluating a polynomial involves additions, subtractions, and multiplications. Both of which are primitive recursive. Checking for equality is also primitive recursive.

   However, we've seen in class that *bounded* search for the smallest number satisfying a predicate, less than a given number, is a primitive recursive procedure. Therefore, searching for solutions below some bound is primitive recursive. For a general polynomial, there is no bound to put so the process of looking for zeroes of a polynomial is an unbounded search. These are not primitive recursive, because primitive recursive functions are finite i.e. always halt!

   The unbounded μ operator can be used to define a search procedure, but this takes us out of the class of primitive recursive functions. Search for roots of a polynomial is not primitive recursive, but μ-recursive.

   Restricting the search space by looking for zeroes modulo N will make the procedure primitive recursive - as the search space is now finite.

Tutorial Exercises

Peano Arithmetic

1. Write down well-formed formulae in the first-order language of PA corresponding to the following statements.

   (a) Each natural number is either equal to 0 or greater than 0.

   Solution: $\forall x\ [(x = 0) \vee (0 < x)]$

   (b) If x is not less than y, then x equals y or y is less than x.

   Solution: $\forall x\ \forall y\ [\neg(x < y) \rightarrow ((x = y) \vee (y < x))]$

   (c) If x is less than or equal to y and y is less than or equal to x, then $x = y$.

   Solution: $\forall x\ \forall y\ [(x \leq y) \wedge (y \leq x)] \rightarrow x = y$

2. Provide natural deductions of the following theorems of Peano Arithmetic.

   (a) PA, $0 < a \ \vdash\ 0 < s(a)$

   Solution:

$$\dfrac{\dfrac{0 < a}{\exists y\ (a = 0 + s(y))} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a = 0 + s(b)}\ 1 \quad \dfrac{\dfrac{}{a = 0 + s(b) \rightarrow s(a) = s(0 + s(b))}\ \text{THM}}{s(a) = s(0 + s(b))}\ \text{MP} \quad \dfrac{\text{PA4}}{0 + s(s(b)) = s(0 + s(b))}\ \forall E}{s(a) = 0 + s(s(b))}\ = E}{\exists x\ s(a) + 0 + s(x)}\ \exists I}{(a = 0 + s(b)) \rightarrow \exists x\ (s(a) = 0 + s(x))}\ \rightarrow I, 1}{\exists x\ (s(a) = 0 + s(x))}\ \exists E}{0 < s(a)}$$

   (b) PA, $a < b \ \vdash\ s(a) < s(b)$

   Solution:

   This should go similar to the proof above :)

(c) PA, $(a < b) \wedge (b < c) \vdash a < c$

Solution:

The proof that follows contains a lot of terms with the sugar left on. Remember that $a < b$ is really an $\exists$-statement. Therefore to introduce or eliminate it, one must use $\exists$ rules of inference.

The THM being called is the proof that $+$ is associative i.e. one can rebracket as done in that step. This was proved in the previous tutorial.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{b = a + s(u)}\,1 \quad \dfrac{}{c = b + s(v)}\,2}{c = (a + s(u)) + s(v)}\,=}{c = a + (s(u) + s(v))}\,\textsf{THM} \quad \dfrac{\textsf{PA4}}{s(u) + s(v) = s(s(u) + v)}\,\forall E}{c = a + s(s(u) + v)}\,=E}{a < c}\,\exists I}{(b = a + s(u)) \to a < c}\,\to I, 1}{(c = b + s(v)) \to ((b = a + s(u)) \to (a < c))}\,\to I, 2 \qquad \dfrac{\dfrac{(a < b) \wedge (b < c)}{b < c}\,\wedge E_r}{b = a + s(u) \to a < c}\,\exists E \qquad \dfrac{\dfrac{a < b \wedge b < c}{a < b}\,\wedge E_l}{a < c}\,\exists E$$

(d) PA $\vdash \forall x\ [(x = 0) \vee (0 < x)]$ (Challenge!)

Solution: For this proof we will use induction on the formula

$$P(x) :\equiv (x = 0 \vee 0 < x)$$

First we provide the base case deduction $\mathcal{D}_{BC}$ for the sequent:

$$\text{PA} \vdash\ 0 = 0 \vee 0 < 0$$

$$\dfrac{\dfrac{}{0 = 0}\ =}{0 = 0 \vee 0 < 0}\ \vee I$$

Next we provide the induction step $\mathcal{D}_{IS}$ for the sequent

$$\text{PA},\ (a = 0) \vee (0 < a) \vdash\ s(a) = 0 \vee 0 < s(a)$$

This requires the use of $\vee$E since the induction hypothesis is a disjunction. Therefore the induction step will be of the following form:

$$\dfrac{\dfrac{}{(a = 0) \vee (0 < a)}\ \text{IH} \quad \dfrac{\begin{array}{c}\mathcal{D}_1\\ \vdots\\ (a = 0) \to \gamma\end{array}}{}\ \to I \quad \dfrac{\begin{array}{c}\mathcal{D}_2\\ \vdots\\ (0 < a) \to \gamma\end{array}}{}\ \to I}{s(a) = 0\ \vee\ 0 < s(a)}\ \vee E$$

Where $\gamma :\equiv s(a) = 0 \vee 0 < s(a)$. So the induction step is split into two deductions $\mathcal{D}_1$ and $\mathcal{D}_2$ which are presented below.

First $\mathcal{D}_1$ is a proof of the sequent:

$$\text{PA},\ a = 0 \vdash\ s(a) = 0 \vee 0 < s(a)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{s(0) = 0 + 1}\ \text{THM} \quad \dfrac{\dfrac{}{a = 0}\ 1 \quad \dfrac{}{a = 0 \to s(a) = s(0)}\ =}{s(a) = s(0)}\ =}{s(a) = 0 + s(0)}\ =}{\exists x\ (s(a) = 0 + s(x))}\ \exists I}{0 < s(a)}}{s(a) = 0 \vee 0 < s(a)}\ \vee I}{a = 0 \to (s(a) = 0 \vee 0 < s(a))}\ \to I, 1$$

Next the deduction $\mathcal{D}_2$ is a proof of the sequent

$$\text{PA},\ 0 < a \vdash\ s(a) = 0 \vee 0 < s(a)$$

However, this is (pretty much) exercise 2a; just add an $\vee$I at the end of the proof.

(e) PA $\vdash\ \forall x\ \forall y\ [\neg(x < y) \to ((x = y) \vee (y < x))]$ (Challenge!!!)

Solution: Needs RAA for the disjunction.

(f) PA $\vdash\ \forall x\ \forall y\ [(x \leq y) \wedge (y \leq x)] \to x = y$ (Challenge!!!)

Solution:

Primitive Recursive Functions

For each of the procedures below it is sufficient to provide an informal explanation of how they can be computed from primitive recursive functions using composition and recursion. For an extra challenge, you can also provide the formal definition from the base primitive recursive functions.

3. Give an informal description of a primitive recursive procedure that computes the following predicates and functions. You may refer to any primitive recursive procedure from class or earlier in the tutorial.

   (a) Strictly less than relation $<?(x, y) = 1$ if $x < y$ otherwise $<?(x, y) = 0$.

   Solution:

   This can be defined in a similar way to the predicate defined in the first discussion question. Here the point is that $x < y$ is the limited difference $\mathsf{ld}(x + 1, y) = 0$. Notice that we need the successor on the x in order to get *strictly* less than. Therefore the strictly-less-than predicate can be defined as a composition of primitive recursive functions, and is therefore itself primitive recursive.

   $$<?(x, y) :\equiv \mathsf{Zero?}(\mathsf{LD}(s(x), y))$$

   (b) less than relation $\leq?(x, y) = 1$ if $x \leq y$ otherwise $\leq?(x, y) = 0$.

   Solution: This is similar. Only this time the successor function is not needed.

   (c) Greater than relation $>?(x, y) = 1$ if $x > y$ otherwise $>?(x, y) = 0$.

   Solution:

   This can be computed with x,y swapped in the procedure for <?

   (d) Strictly greater than relation $\geq?(x, y) = 1$ if $x \geq y$ otherwise $\geq?(x, y) = 0$.

   Solution:

   This can be computed with x,y swapped in the procedure $\leq?(x, y)$.

   (e) Identity relation $=?(x, y) = 1$ if $x = y$ otherwise $=?(x, y) = 0$.

   Solution:

   Two numbers are equal if they are both less-than or equal to each other. In that case both $\leq?(x, y) = 1$ and $\geq?(x, y) = 1$. Therefore $=?(x, y)$ can be computed as the product of these two procedures:

   $$=?(x, y) :\equiv \leq?(x, y) \times \geq?(x, y)$$

   (f) Absolute difference $|x - y|$

   Solution:

   Absolute value functions are piecewise functions i.e. they branch depending on the values of some predicates.

   $$|x - y| :\equiv \begin{cases} \mathsf{LD}(x, y) & x \geq y \\ \mathsf{LD}(y, x) & x < y \end{cases}$$

In lectures we showed that such branching can be obtained by a combination of multiplication and addition.

$$|x - y| = (\geq?(x, y)) \times (\mathsf{LD}(x, y)) + (<?(x, y)) \times (\mathsf{LD}(y, x))$$

(g) Min(x,y)

Solution:

As before, this function is a branching procedure:

$$\mathsf{min}(x, y) :\equiv \begin{cases} x & x \leq y \\ y & x > y \end{cases}$$

$$\mathsf{min}(x, y) :\equiv \leq?(x, y) \times x + >?(x, y) \times y$$

(h) Max(x,y)

Solution:

$$\mathsf{max}(x, y) :\equiv \geq?(x, y) \times x + <?(x, y) \times y$$

4. Addition is iterated succession. Multiplication is iterated addition. Exponentiation is iterated multiplication. Use recursion and exponentiation to define a new primitive recursive function $f(a, b + 1) = exp(a, f(a, b))$, such that $f(a, 0) = 1$. What does this function do? Compute $f(5, 3)$.

   Solution:

$$\begin{aligned} f(5,3) &= exp(5, f(5,2)) && 5^{f(5,2)} \\ &= exp(5, exp(5, f(5,1))) && 5^{5^{f(5,1)}} \\ &= exp(5, exp(5, exp(5, f(5,0)))) && 5^{5^{5^{f(5,0)}}} \\ &= exp(5, exp(5, exp(5,1))) && 5^{5^5} \end{aligned}$$

   This iterated exponentiation is called tetration.

5. Give an informal description of the primitive recursive definitions of the function which returns the maximum (resp. minimum) of three inputs.

   Solution:

$$\mathsf{max}(x, y, z) :\equiv \mathsf{max}(\mathsf{max}(x, y), z)$$

$$\mathsf{min}(x, y, z) :\equiv \mathsf{min}(\mathsf{min}(x, y), z)$$

6. Use bounded minimisation (along with other functions already shown to be primitive recursive) to show that bounded maximisation is primitive recursive.

7. Give informal descriptions of how the following functions could be defined using functions and predicates that were shown to be primitive recursive in class. Hence conclude that each of these procedures are primitive recursive.

   (a) Remainder(n, d): remainder when n divided by d.

   Solution:

   This can be computed as follows:

$$\text{remainder}(n, d) :\equiv \begin{cases} n & n < d \\ \text{remainder}(n - d, d) & \text{otherwise} \end{cases}$$

   This procedure uses composition and recursion of functions and predicates already shown to be primitive recursive.

   (b) Divides?(n, d): 0 or 1 according to whether n divides d.

   Solution:

   This can be computed as a ZERO? test on the remainder.

   DIVIDES?$(n, d) =$ ZERO?(remainder(n, d))

   (c) Prime?(x): 0 or 1 according to whether x is prime.

   Solution:

   This is a bounded search for DIVISORS? of x. One can bound by n-1. Or, one can bound by the integer square-root. However, you have to then show that the integer square-root can be computed by a primitive recursive procedure.

   (d) nextPrime(x): returns the smallest prime greater than[1] x.

   Solution:

   This is a search, starting from $x + 1$, for the next number satisfying PRIME? Following the hints, one can put an upper bound on this search. We are guaranteed to come across a prime by the time we get to $2x$. Therefore this procedure is primitive recursive.

   (e) $p : \mathbb{N} \to \mathbb{N}$ such that $p(n) =$ nth prime.

   Solution:

   This function can be computed recursively as follows:

$$p(n) :\equiv \begin{cases} 2 & n = 0 \\ \text{nextPrime}(p(n - 1)) & \text{otherwise} \end{cases}$$

   As such the procedure is primitive recursive.

[1] Hint 1: Prove that there must be a prime, p, between $x < p \leq x! + 1$.
Hint 2: Chebyshev said it, so I'll say it again, there's always a prime, between N and $2N$.

8. Propositional logic is primitive recursive. Suppose $P(\mathbf{x})$ and $Q(\mathbf{x})$ are two primitive n-ary predicates i.e. the characteristic functions χ_P and χ_Q are primitive recursive. Show that the (characteristic functions of the) following predicates are primitive recursive.

   (a) $\neg P(\mathbf{x})$

   Solution:

   $$\chi_{\neg P}(x) :\equiv 1 - \chi_P(x)$$

   (b) $P(\mathbf{x}) \vee Q(\mathbf{x})$

   Solution:

   $$\chi_{P \vee Q}(x) :\equiv \chi_P(x) + \chi_Q(x) - \chi_P(x) \times \chi_Q(x)$$

   (c) $P(\mathbf{x}) \wedge Q(\mathbf{x})$

   Solution:

   $$\chi_{P \wedge Q}(x) :\equiv \chi_P(x) \times \chi_Q(x)$$

   (d) $P(\mathbf{x}) \to Q(\mathbf{x})$

   Solution:

   See discussion questions.

   (e) NAND$(P(\mathbf{x}), Q(\mathbf{x}))$

   Solution:

   $$\chi_{\neg(P \wedge Q)}(x) :\equiv 1 - \chi_{P \wedge Q}(x) = 1 - \chi_P(x) \times \chi_Q(x)$$