Lecture 6: A Decision Procedure?

Dr John Levine

CS103 Machines, Languages and Computation
October 12th 2015

Homework for Tutorial this week

- Read Chapter 2 of Gödel, Escher, Bach
- Do Assignment 2, parts (d) and (e)
  - Part (d) is a proof by induction
  - Part (e) asks why proof by induction is like toppling an infinite line of dominoes
- Try to complete Assignment 1, part (d)
  - What is the decision procedure for MIU?
- Hand in your workbooks by 3pm tomorrow

Infinite Sets

- Sets can have an infinite number of elements.
- The most important is the set of *natural numbers*, *N.*
- $N = \{0, 1, 2, 3, \ldots\}$
- Property 1: There is a first element: in this case, 0.
- Property 2: Each element has a natural successor: if I'm looking at element a, the next element is $a+1$.
- Not all infinite sets have these two properties.
- But if our infinite set does have these two properties, then we can do *proof by induction.*

Proof by Induction

- Often used to prove statements of the form:

  "for all $n \in N$, some property holds of n"

- First, prove true for the first case, which is easy enough: just find the smallest possible value of n for which the formula makes sense, and see if the formula holds.
- Next, assume the formula is true for some arbitrary value, k. The inductive step consists of proving that the formula must be true for the value $k+1$.
- The "domino effect" makes the formula true for all n.

Example of Proof by Induction

- The sum of the first n non-zero integers = $n.(n+1)/2$
- In other words, $1 + 2 + 3 + \ldots + n = n.(n+1)/2$
- Proof by induction:
- First case is $n = 1$. $1 = 1.(1+1)/2$, which is true.
- Assume kth case is true:

$$1 + 2 + 3 + \ldots + k = k.(k+1)/2 \qquad (1)$$

- Now prove the $(k+1)$th case is true:

$$1 + 2 + 3 + \ldots + k + (k+1) = (k+1).(k+2)/2 \qquad (2)$$

Example of Proof by Induction

- The sum of the first n non-zero integers = $n.(n+1)/2$
- In other words, $1 + 2 + 3 + \ldots + n = n.(n+1)/2$
- Proof by induction:
- First case is $n = 1$. $1 = 1.(1+1)/2$, which is true.
- Assume kth case is true:

$$1 + 2 + 3 + \ldots + k = k.(k+1)/2 \qquad (1)$$

- Now prove the $(k+1)$th case is true:

$$1 + 2 + 3 + \ldots + k + (k+1) = (k+1).(k+2)/2 \qquad (2)$$

Example of Proof by Induction

- Substitute the right-hand side of (1) into (2):
- $1 + 2 + 3 + \ldots + k + (k+1) = (k+1).(k+2)/2 \qquad (2)$
- $k.(k+1)/2 + (k+1) = (k+1).(k+2)/2$
- $(k^2 + k)/2 + (k+1) = (k+1).(k+2)/2$
- $k^2/2 + k/2 + k+1 = (k+1).(k+2)/2$
- $k^2/2+3k/2+1 = (k+1).(k+2)/2$
- $k^2/2+3k/2+1 = (k^2+3k+2)/2$
- $k^2/2+3k/2+1 = k^2/2+3k/2+1.$ ■

Assignment 2, Part (d)

- Use induction to produce a proof that the sum of the first n odd numbers is n^2:

$$n = 1: \quad 1 = 1$$
$$n = 2: \quad 1 + 3 = 4$$
$$n = 3: \quad 1 + 3 + 5 = 9$$
$$n = 4: \quad 1 + 3 + 5 + 7 = 16$$

.

.

.

A Decision Procedure for MIU?

Assignment 1, parts (c) and (d):

(c) What is meant by a *decision procedure* for strings of the MIU system?

(d) Can you write a simple decision procedure for strings of the MIU system?

A Decision Procedure?

- All theorems start with an M with the rest being a mixture of U and I – we can write this as M(U*I*)*
- Is this enough to characterise all the theorems of the MIU-system?
- If not, can we somehow make the description of theorems more restrictive?
- What we want is a *decision procedure*, i.e. a test for theoremhood that tells us if a string is a theorem and gives us an answer in a finite amount of time

Decidable Problems

- Say we have a question to which the answer is “yes” or “no”, such as “Is *k* a prime number?” or “Is string *S* a theorem of the MIU-system?”
- If we have a procedure for *all cases* which can tell us whether the answer is “yes” or “no” in a *finite amount of time*, then the problem is called *decidable*.
- If no such procedure exists, then the problem is called *undecidable*.
- Note that the program that searches is *not* a decision procedure (why?)

The Challenge

- Can we come up with a decision procedure for strings which are theorems of the MIU-system?
- String = M(U*I*)* is a start, but some strings seem to be very difficult to find…
- Is there any pattern to the theorems my program can produce?
- If there is, can we inspect the rules to find the reason for such a pattern being there?

A possible approach: I-count

- Rule 3 allows III to become a U; let's relax the system and allow U and III to be interchangeable in our string
- This relaxation allows a U to be counted as 3 I's
- Now let the I-count of a string be the number of times we see an I in a string, counting U as 3 I's
- For example, the I-count of MIIUIIU is 10
- What do our 4 rules do to the I-count of a string?

How the rules change the I-Count

- Let's see how the four rules change the I-count of a string:

  I. $x\mathrm{I} \rightarrow x\mathrm{IU}$ `# add 3 to the I-count`
  II. $\mathrm{M}x \rightarrow \mathrm{M}xx$ `# multiply the I-count by 2`
  III. $x\mathrm{III}y \rightarrow x\mathrm{U}y$ `# no change to the I-count`
  IV. $x\mathrm{UU}y \rightarrow xy$ `# subtract 6 from the I-count`

- Starting from an I-count of 1 (i.e. the axiom, MI), what values of the I-count are possible?

- Can we make an I-count of 3 (e.g. MU)?

Possible values of I-count

- So, all we can do to the I-count is add 3 to it, double it, or subtract 6 from it
- Starting from 1, what numbers can you make?
- What numbers are impossible to make?
- Can you use this to make your decision procedure?
- Extra question: if you start with MIII as the axiom rather than MI, how does this change things?
- Reminder: hand in your workbook by 3pm tomorrow