Lecture 4: MI → MU?

Dr John Levine

CS103 Machines, Languages and Computation
October 5th 2015

University of Strathclyde Science

The Rules of the Game

- Lectures are Mon 10 in RC 345, Fri 10 in RC 641
- Tutorials: Thu 10 in JA 505, Thu 1 in TG 223
- Lectures run Weeks 1-11 (no lectures in Week 12)
- Class materials will appear on the class web page:

  http://www.cis.strath.ac.uk/~johnl/CS103
- Facebook page: MLAC 2015
- Email: John.Levine@strath.ac.uk
- Room LT1420, extension 4524

University of Strathclyde Science

Tutorial Allocation

- You attend **one** tutorial per week
- The Thursday 1pm tutorial is for those students taking:
  - BSc Computer Science
- The Thursday 10am tutorial is for those students taking:
  - MEng Computer Science
  - BSc Mathematics and Computer Science
  - MEng Computer and Electronic Systems
  - BEng Computer and Electronic Systems
  - BSc Software Engineering
  - BSc Business Information Systems
  - Everyone else not already mentioned

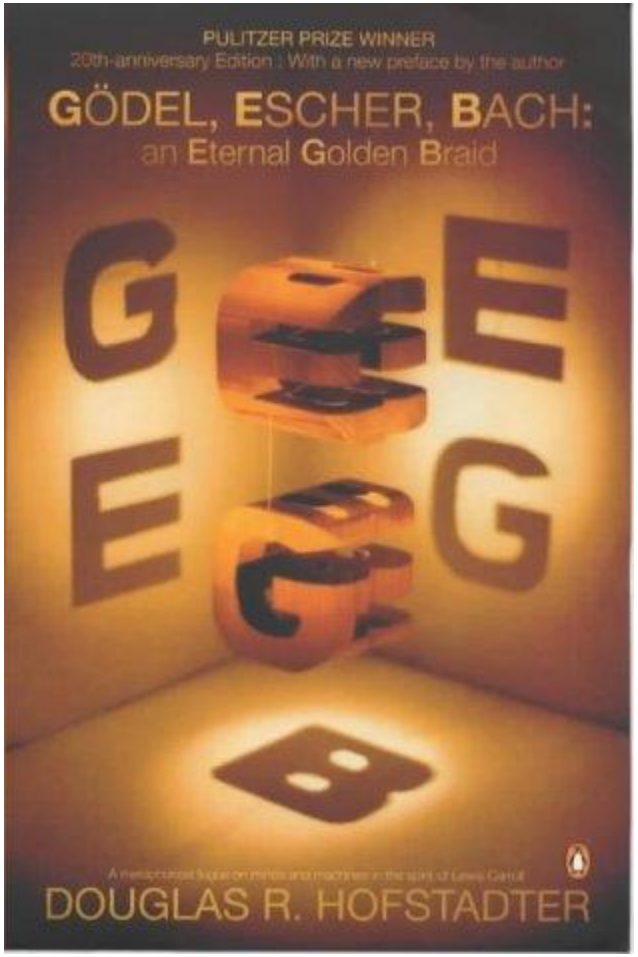
University of Strathclyde Science

Course Book

"Gödel, Escher, Bach: an Eternal Golden Braid"
by
Douglas R. Hofstadter

Publisher: Penguin Books Ltd
ISBN: 0140289208
Cost: £18.99 (amazon.co.uk)

You will need your own copy of this book as we will be reading from it every week.



University of Strathclyde Science

Non-Computable Functions

- The first 5 weeks of the course will build up towards the proof that non-computable functions exist:

  If P is the set of all computer programs and F is the set of all functions $f: n \rightarrow m$ such that n,m are members of the set of natural numbers N, then $|F| > |P|$ and therefore there are some functions in F for which no computer program can exist.

University of Strathclyde Science

Non-Computable Functions

- The first 5 weeks of the course will build up towards the proof that non-computable functions exist:

  If P is the set of all computer programs and F is the set of all functions $f: n \rightarrow m$ such that n,m are members of the set of natural numbers N, then $|F| > |P|$ and therefore there are some functions in F for which no computer program can exist.

University of Strathclyde Science

What is a Mathematical Proof?

- In mathematics, a proof is a logical argument showing that some statement (a *theorem*) is necessarily true.
- A statement is an expression that is either true or false, such as “all even integers > 2 can be expressed as the sum of two prime numbers” (Goldbach’s conjecture)
- A proof starts with *axioms*, which are statements whose truth can be taken for granted, such as “any even number can be exactly divided by 2”.
- *Logical deduction* is the process of creating new true statements from the given axioms.

University of Strathclyde Science

A Simple Proof

Theorem: adding two even numbers always gives an even number.

- Proof: let the sum be $s = a + b$ where *a* and *b* are *any* two even numbers
- Let $p = a/2$ and $q = b/2$. Because *a* and *b* are even, *p* and *q* are both whole numbers (integers)
- The sum is now: $s = 2p + 2q$
- We can rewrite this as: $s = 2(p + q)$
- Because *p* and *q* are integers, $(p + q)$ is also an integer
- *s* is *2* times an integer, and so this means that *s* must be an even number. ■

University of Strathclyde Science

Types of Proof

- *Direct proof* is where the conclusion is established by logically combining the axioms, definitions and earlier theorems.
- *Proof by contradiction* is where it is shown that if some statement were false, a logical contradiction occurs, hence the statement must be true.
- *Proof by induction* is where a “base case” is proved, and an “induction rule” used to prove an (often infinite) series of other cases. Since the base case is true, the infinity of other cases must also be true.

University of Strathclyde Science

Prime Numbers

- The usual definition of a prime number is “any integer greater than 1 that divides exactly only by itself and 1”
- 1 is not prime, because of The Fundamental Theorem of Arithmetic (due to Euclid): “Every positive integer greater than one can be written *uniquely* as a product of primes”
- Those numbers which can be written as a product of prime factors are known as *composite* numbers
- So, all integers > 1 are either prime or composite

University of Strathclyde Science

Prime Numbers

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

University of Strathclyde Science

Prime Numbers

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

University of Strathclyde Science

Proof by Contradiction

Theorem: There are infinitely many prime numbers.

- Proof: assume that there are finitely many prime numbers, and call the biggest prime number *N*.
- Now form the number $K = 1 \times 2 \times 3 \times \ldots \times N\text{-}1 \times N$.
- *K* divides exactly by every integer up to *N*.
- Now form the number $M = K + 1$.
- *M* can't be a multiple of 2, as it leaves 1 over when you divide it by 2.
- Or a multiple of 3, 4, 5 , … , *N*-1, *N.*
- So either *M* is prime, or it has a prime divisor > *N*. ■

University of Strathclyde Science

Formal Systems

- The proofs we've seen are a mixture of *formal* symbol manipulation (e.g. rewrite "$s = kx + ky$" as "$s = k(x + y)$") and more informal logical argument (e.g. the smallest divisor of M must be prime).
- To be totally sure of our arguments, we can try to make *all* of the steps be formal symbol manipulation.
- If we have symbol manipulation rules which can only produce true statements from true statements, our proofs are cast-iron guaranteed.
- This could also allow computers to construct proofs!

University of Strathclyde Science

Assignment 1 (taken from GEB)

- We are given the axiom MI and we have to make the theorem MU, using these four inference rules:

  I. $x\mathrm{I} \rightarrow x\mathrm{IU}$
  II. $\mathrm{M}x \rightarrow \mathrm{M}xx$
  III. $x\mathrm{III}y \rightarrow x\mathrm{U}y$
  IV. $x\mathrm{UU}y \rightarrow xy$

- Hofstadter asks: can you make MU?
- Levine asks: given some arbitrary string, can I write a computer program to determine whether or not it is a theorem?

University of Strathclyde Science

Approaches to MU

- Begin by trying out the system, to see how it works
- Start deriving a few theorems:

MI → MII	(Rule II)
MII → MIIII	(Rule II)
MIIII → MUI	(Rule III)
MUI → MUIU	(Rule I)
MUIU → MUIUUIU	(Rule II)
MUIUUIU → MUIIU	(Rule IV)

- Exercise: try to derive MUIUIU

University of Strathclyde Science

Answer to the Exercise

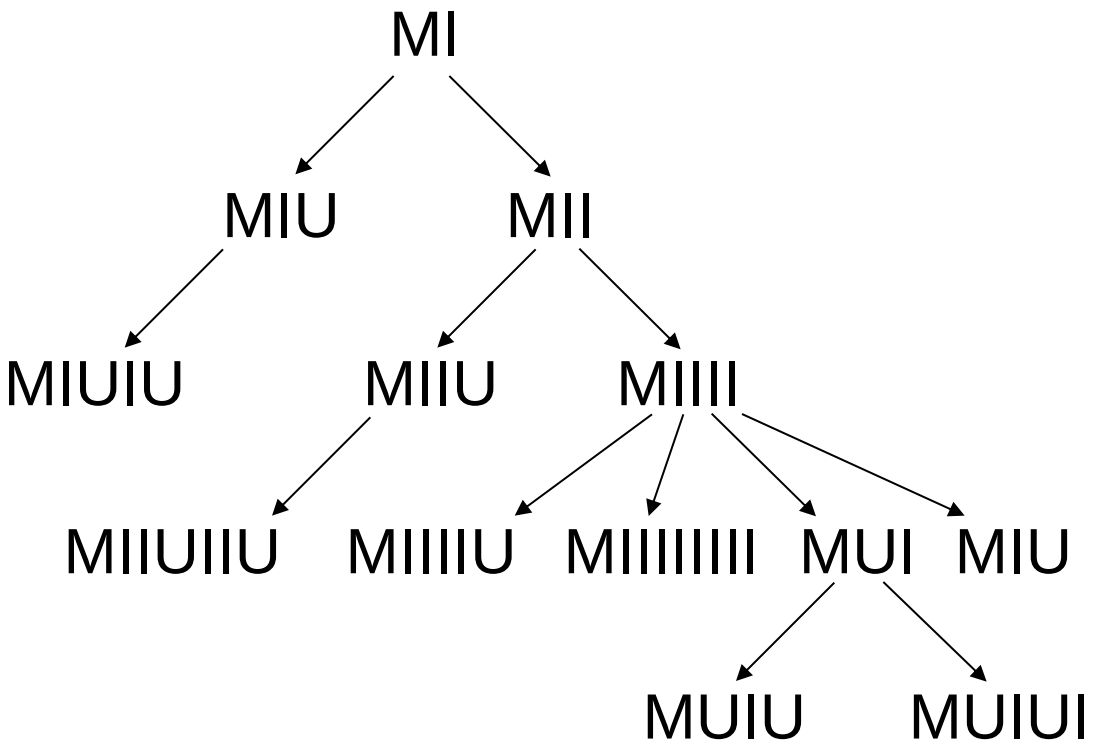
- Here's one way of doing it:

MI → MII	(Rule II)
MII → MIIII	(Rule II)
MIIII → MUI	(Rule III)
MUI → MUIUI	(Rule II)
MUIUI → MUIUIU	(Rule I)

- But can we derive MU?
- Maybe we can write a program to do it for us?

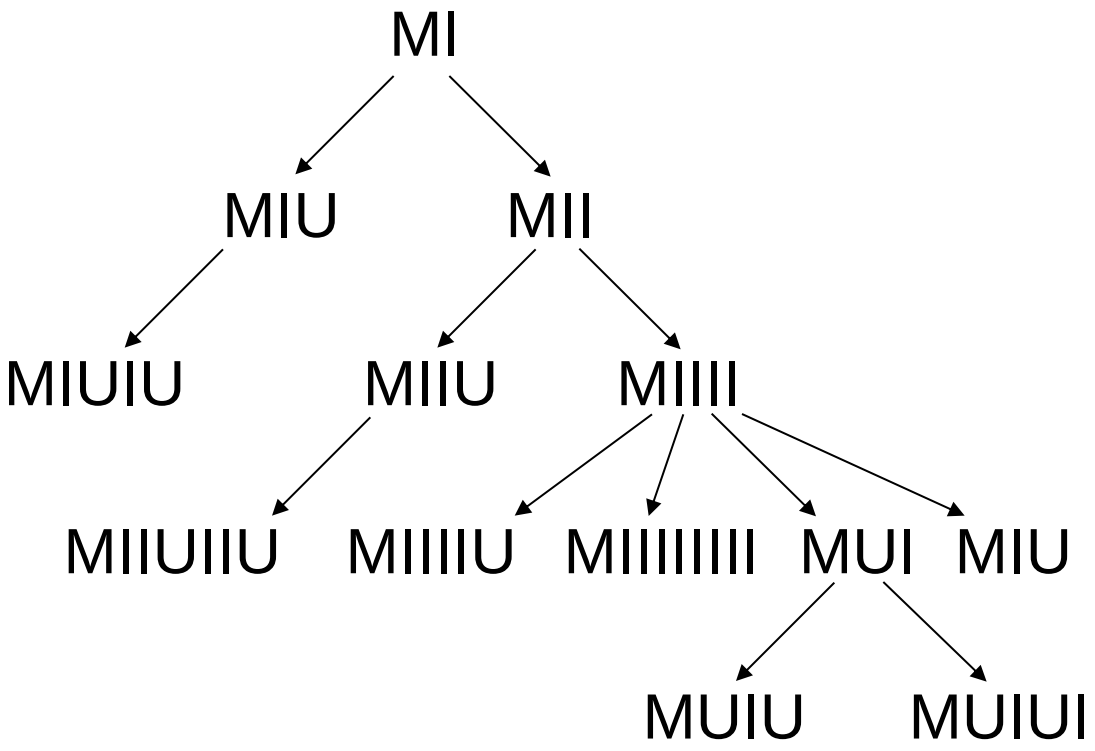
University of Strathclyde Science

A Program that Searches?



University of Strathclyde Science

A Program that Searches?



MU???

University of Strathclyde Science

Observing the System

After working with the system for a while, we start making observations about the system:

- All theorems start with an M with the rest being a mixture of U and I – we can write this as M(U*I*)*
- So some strings are definitely not theorems, such as IIUM, MUMII, MUMUMUII, and so on
- There are infinitely many theorems – Rule II alone is enough to ensure this
- Rules I and II make longer strings, Rules III and IV make shorter strings

University of Strathclyde Science

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

University of Strathclyde Science

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?)

University of Strathclyde Science

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?

University of Strathclyde Science

More Homework

- Assignment 1
  - Part (c): [fairly easy] define what is meant by the term *decision procedure* in the context of the MIU system
  - Part (d): [hard!] try to define a decision procedure for the MIU system
- Assignment 2
  - Part (c): [moderate] without looking, try to reproduce the proof that there are an infinite number of primes
- Hand your workbook in to the office by 3pm on Tuesday