# The *MU* Formal System Automatic Deduction

Pan Baoxiang

## Problem Statement

The *MU* formal system comes from Doctor Douglas Hofstadter's book *Gödel, Escher Bach-An Eternal Golden Braid.* This simple system uses only three letters, namely $M$,$I$ and $U$, to employ the abundance of strings under the constriction of certain rules as stated below:

- If the string ends with $I$, a $U$ could be appended.
- If we have $Mx$, then we also have $Mxx$. e.g. $MUI => MUIUI$.
- The $III$ in the string could be replaced by $U$.
- We could erase $UU$ wherever it appears.

Here comes the question:
We are given $MI$, could we generate $MU$ according to the three rules above?

## Algorithm

The algorithm is constructed in five parts, namely, the axiom definition, derivation rules definition, memory definition, deduction process definition and the test section. We first apply the derivation rules to the axiom to generate legit theorems, restore them into our memory, then, more and more theorems are generated, with the new ones stored to into the memory, until the final MU is generated could we stop our program.

The program is implemented in mit-scheme, which in turn deeply influences the perspective toward this problem.

**Define Axiom**

```
(define init (list (list 'm 'i)))
```

**Define Derivation Rules**

```
(define (operator lis)
(define (rule_1 x)
(define (end lis)
(if (null? (cdr lis))
(car lis)
(end (cdr lis))))
(if (eq? (end x) 'i)
(append x (list u))))
(define (rule_2 x)
(append x (cdr x)))
(define (rule_3 x)
(cond ((> (length x) 3)
(cond ((and (eq? (cadr x) 'i)
(eq? (caddr x) 'i) (eq? (cadddr x) 'i))
(append (list 'm 'u) (cddddr x)))
(else '())))
(else '())))
(define (rule_4 x)
(cond ((or (null? x) (null? (cdr x))) x)
((and (eq? (car x) 'u) (eq? (cadr x) 'u))
(rule_4 (cddr x)))
(else (cons (car x) (rule_4 (cdr x))))))
(define (erase seq)
(filter (lambda(x) (not (null? x))) seq))
(erase (list (rule_1 lis) (rule_2 lis)
(rule_3 lis) (rule_4 lis)))
)
```

**Define Memory**

```
(define dictionary init)

(define (in? value lis)
(or (equal? value (car lis))
(and (not (null? (cdr lis)))
(in? value (cdr lis)))))
```

**Define Deduction**

```
(define (expand seqs)
(define (erase_repeat lis)
(cond ((null? lis) '())
(else (cons (car lis)
(filter (lambda(x) (not (equal? x (car lis))))
(erase_repeat (cdr lis)))))))
(define (pre-expand lis)
(if (null? lis)
'()
(append (operator (car lis))
(pre-expand (cdr lis)))))
(define result
(filter (lambda(x) (not (in? x dictionary)))
(erase_repeat (pre-expand seqs))))
(begin
(set! dictionary (append dictionary result))
result))
```

**Define Test**

```
(define object (list 'm 'u))
(define (f a)
(if (in? object a) 'yeah (f (expand a))))
(f init)
```

## Results

```
1 ]=> (expand init)

;Value 12: ((m i u) (m i i))

1 ]=> dictionary

;Value 13: ((m i) (m i u) (m i i))

1 ]=> (expand (expand (expand init)))
```

```
;Value 14: ((m i u u u) (m i u u i u u) (m i u u i u)
(m i u i u i u i u) (m i u u i)(m i u i i u i)
(m i u i i i) (m i i i i i i i i) (m u i))

1 ]=> dictionary

;Value 15: ((m i) (m i u) (m i i)
(m i u u) (m i u i u) (m i u i) (m i i i i)
(m i u u u) (m i u u i u u) (m i u u i u)
(m i u i u i u i u) (m i u u i) (m i u i i u i)
(m i u i i i) (m i i i i i i i i) (m u i))

1 ]=> (f init)

;Aborting!: out of memory
;GC #97: took: 0.80 (35%) CPU time, 0.80 (36%) real time;
free: 2752857
;GC #98: took: 0.70 (88%) CPU time, 0.70 (99%) real time;
free: 2752937
```

## Discussion

The auto deduction program could generate new theorems according to the rules we give. However, it meets the problem of stack overflow when we test if the *MU* string could be generated. This is not esoteric once we calculate the computation complexity of the algorithm.

Let's suppose every derivation rule is applicable once the string is complex enough. Each application of a certain rule generate a new theorem. the $nth$ layer theorem number is 4 times that of the $(n-1)th$ layer. The quantity of the $0th$ layer theorem, which is the axiom, is 1, thus, without considering duplication, the size of the memory would be $\frac{1}{3}[4^{n+1}-1]$. We are meeting a exponent explosion.

In fact, the *MU* string could never be deduced once we jump out of the formal system to check it with the skill of Gödel Number. The most advantage of human mind over the mechanism learning method today is that it could jump between different levels to find links and contradictions, to make abstraction and deduction, and finally, to reflect the whole process.