Suppose I have two C programs:
M1.c and M2.c

I remember from a few years ago that one of them halts when I use the Keller 7 graph (which has 16,384 vertices) as input and gives me a Hamilton cycle and the other one has a bug in it and goes into an infinite loop.

But I forget which one is which.

What should I do to figure out which one is the good program and which one is buggy?

Undecidable Problems

Decidable problem:

Yes/No Question such that there exists a TM (or equivalently a C or Java program) which halts on all inputs with the answer to the question.

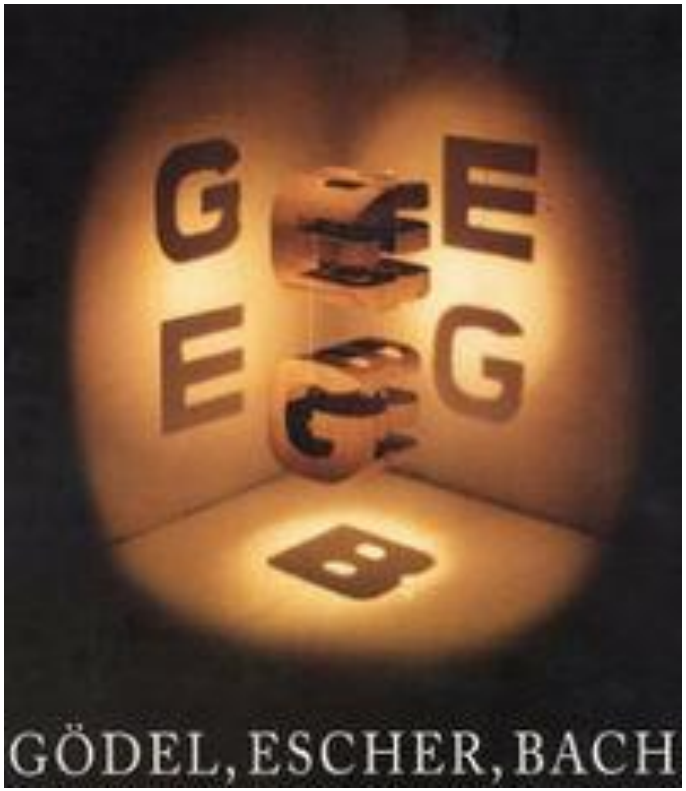
In this lecture, we give a proof that some problems are not decidable.

Proving that our initial problem is not decidable requires self-reference.

Gödel, Escher, Bach: an Eternal Golden Braid is a Pulitzer Prize-winning book by Douglas Hofstadter, described by the author as "a metaphorical fugue on minds and machines in the spirit of Lewis Carroll". On its surface, it examines logician Kurt Gödel, artist M. C. Escher and composer Johann Sebastian Bach, discussing common themes in their work and lives. At a deeper level, the book is a detailed and subtle exposition of concepts fundamental to mathematics, symmetry, and intelligence.

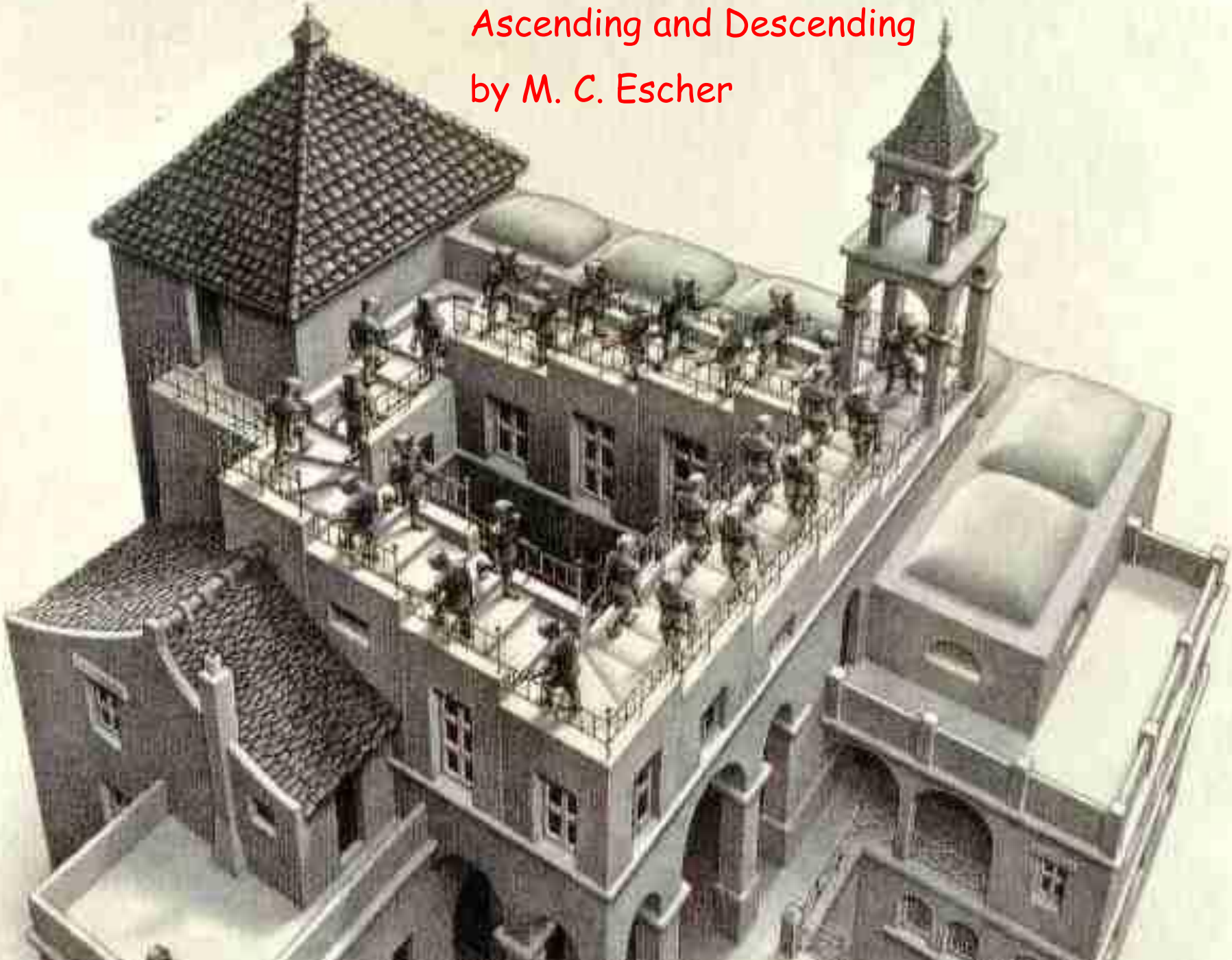
Through illustration and analysis, the book discusses how self-reference and formal rules allow systems to acquire meaning despite being made of "meaningless" elements. It also discusses what it means to communicate, how knowledge can be represented and stored, the methods and limitations of symbolic representation, and even the fundamental notion of "meaning" itself.

From Wikipedia, the free encyclopedia.

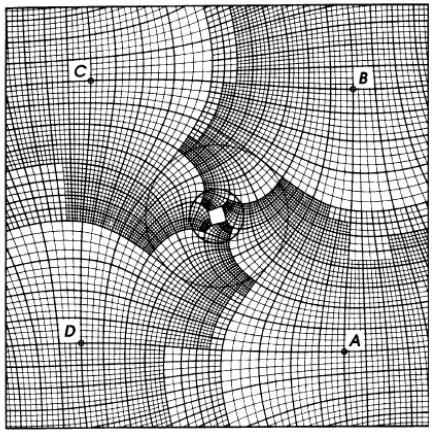


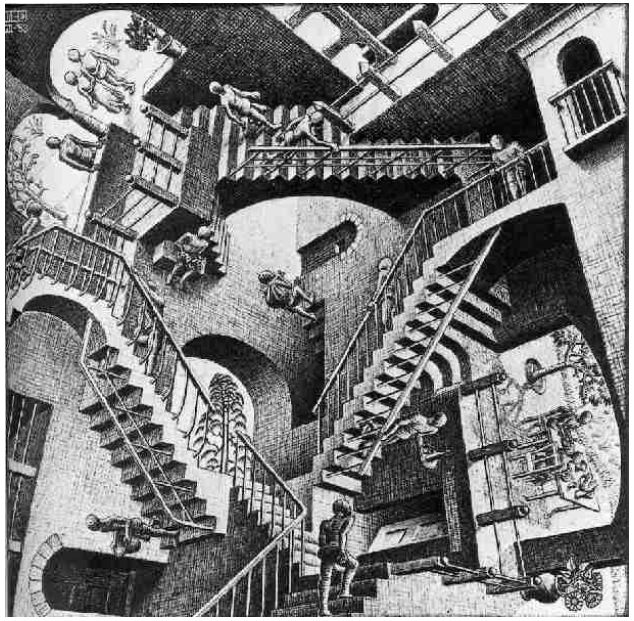
Ascending and Descending

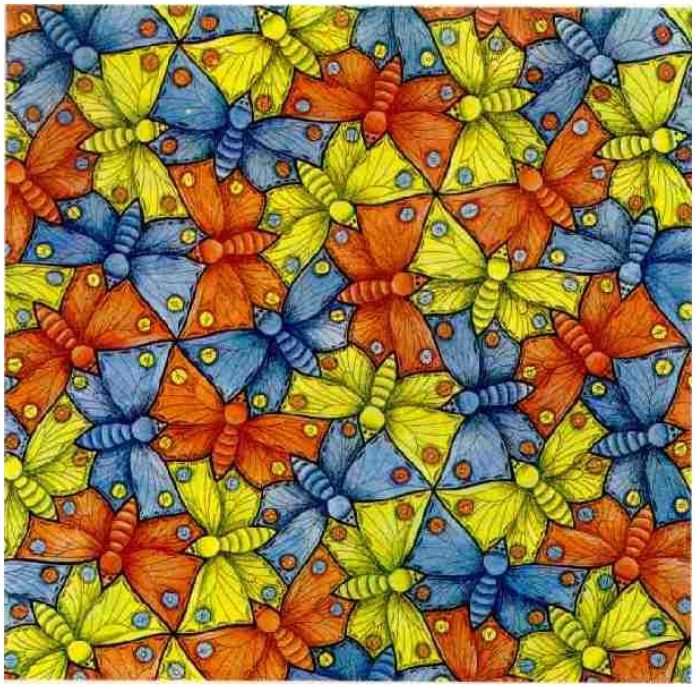
by M. C. Escher

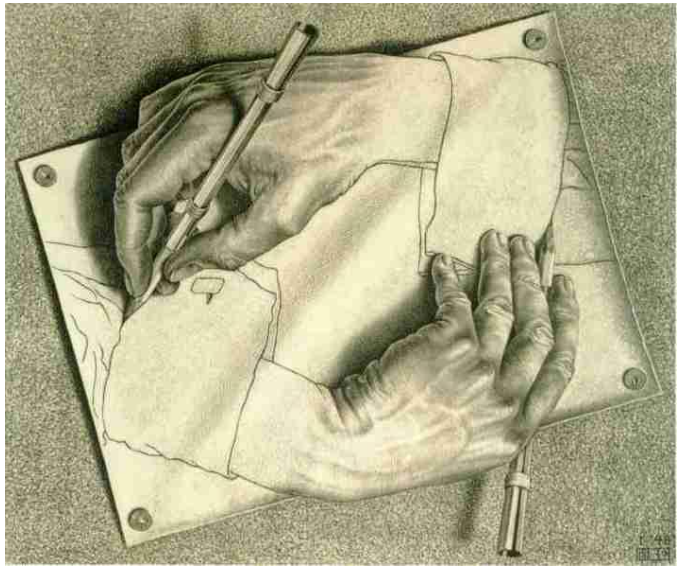


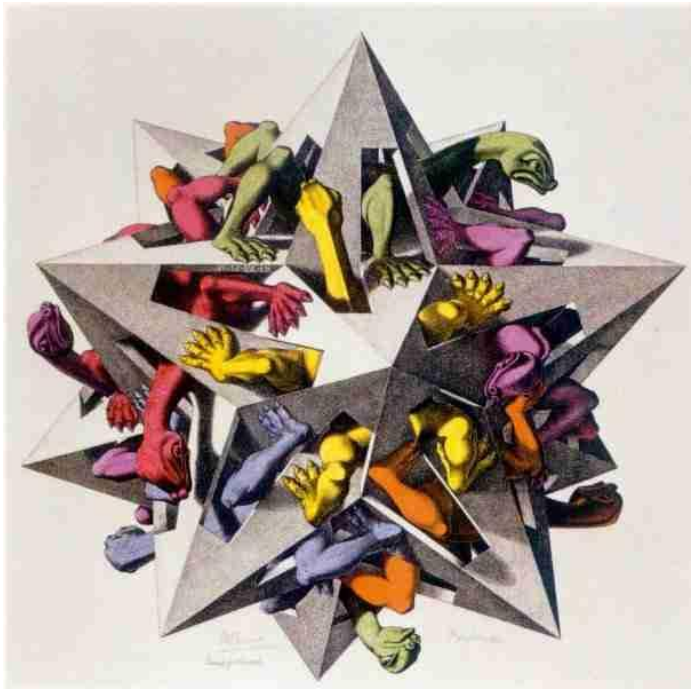
Print Gallery by M. C. Escher











A Strange Language

L= { w : w is a valid C program that prints "no" as its first output on input w}

Can we design a C program P which decides if a string w is in L or not?

Such a program should halt on w with output "yes" if w is in L and "no" if w is not in L. A deciding program has no other outputs.

p1.c

```
main()
{
   int count;
   char c;

/* Count number of a's in the input. */

   count=0;
   while (scanf("%1c", &c)==1)
   {
      if (c== 'a') count++;
   }
   if (count == 6) printf("yes ");
   else printf("no ");
   printf("Number of a's is: %5d\n",
  count);
}
```

To run p1.c on p1.c

```
gcc p1.c
a.out < p1.c

The output is:

yes Number of a's is:     6
```

p1.c

```
main()
{
   int count;
   char c;

/* Count number of a's in the input. */
   count=0;
   while (scanf("%1c", &c)==1)
   {
      if (c== 'a') count++;
   }
   if (count == 6) printf("yes ");
   else printf("no ");
   printf("Number of a's is: %5d\n",
  count);
}
```

yes Number of a's is: 6

p2.c

```
main()
{
   int count;
   char c;

/* Count the number of a's in the input.
  */
   count=0;
   while (scanf("%1c", &c)==1)
   {
      if (c== 'a') count++;
   }
   if (count != 6) printf("yes ");
   else printf("no ");
   printf("Number of a's is: %5d\n",
  count);
}
```

no Number of a's is: 6

p3.c

```
/*  23  The program tells us if this number if prime */
main()
{
   char *c; int i, n;

/* Read in the "/" and then the "*" from the comment at the top. */

   scanf("%1c", &c); scanf("%1c", &c);

/* Read in the integer at the top. */

   scanf("%d", &n);
   for (i=2; i < n; i++)
   {
      if (n % i == 0)
      {
         printf("no- %5d is a divisor of %5d\n", i, n);
         exit(0);
      }
   }
   printf("yes- %5d is prime.\n", n);
}
```

yes- 23 is prime.

p4.c

```
/*  35  The program tells us if this number if prime */
main()
{
   char *c; int i, n;

/* Read in the "/" and then the "*" from the comment at the top. */

   scanf("%1c", &c); scanf("%1c", &c);

/* Read in the integer at the top. */

   scanf("%d", &n);
   for (i=2; i < n; i++)
   {
      if (n % i == 0)
      {
         printf("no- %5d is a divisor of %5d\n", i, n);
         exit(0);
      }
   }
   printf("yes- %5d is prime.\n", n);
}
```

```
no-     5 is a divisor of    35
```

p5.c

```
main()
{
   int i

   for i from 2 to 6 do
       i++
}
```

```
p5.c: In function `main':
p5.c:6: syntax error before `for'
```

p6.c

```
main()
{
   int i;

   i= 1;

   while (i > 0)
   {
       if ( i==1 ) i++;
       else i--;
       // printf("i= %5d\n", i);
   }
   printf("no\n");
}
```

L= { w : w is a valid C program that prints "no" as its first output on input w}

p1.c:	yes Number of a's is: 6	no
p2.c:	no Number of a's is: 6	yes
p3.c:	yes- 23 is prime.	no
p4.c:	no- 5 is a divisor of 35	yes
p5.c:	Does not compile	no
p6.c:	No output- infinite loop	no

L= { w : w is a valid C program that prints
"no" as its first output on input w}

What might P.c which decides L look like?

1. Read input into a string variable w.
2. Check if w is a valid C program using a compiler.
   If not, print "no" and stop.
3. Decide if w's first output when the input is w

   is "no"- if this is true, print "yes" and stop,

   otherwise, print "no" and stop.

L= { w : w is a valid C program that prints "no" as its first output on input w}

What output does this give: gcc P.c ; a.out < P.c

Answer: it should have output "yes" or "no".

L= { w : w is a valid C program that prints "no" as its first output on input w}

What output does this give: gcc P.c ; a.out < P.c

Answer: it should have output "yes" or "no".

Case 1: The output is "yes".
If P.c is working correctly, this means P.c is in L.
But being in L means that a.out < P.c should have "no" as its first output. CONTRADICTION.

Case 2: The output is "no".
This means that P.c is not in L. But by the definition of L, the first output of a.out < P.c cannot be "no". CONTRADICTION.

Conclusion

It is impossible to design a C program P.c which decides if the input is in:

L= { w : w is a valid C program that prints "no" as its first output on input w}

Since any reasonable algorithm can be implemented in C, this means that no algorithm exists for deciding if a string w is in L.

The problem:

Given a string w, is w in L?

is UNDECIDEABLE.

M is a TM with start state s defined over the alphabet {#,(,),0,1,a,q,,}:

State	Sym.	Next State	Head
s	#	s	L
s	)	t	L
t	0	h	R
t	1	t	1

Num	State	Head
0	h	L
1	s	R
2	t	#
3		(
4		)
5		0
6		1
7		a
8		q
9		,

1. What is "M"? Use the table given to number the states and head instructions.

2. Does M halt on input "M"?