From guessing numbers to Cantor’s Theorem

It’s all down to numbers  From http://www.asciitable.com/:

ASCII stands for American Standard Code for Information Inter-change. *Computers can only understand numbers*, so an ASCII code is the numerical representation of a character such as ‘a’ or ‘@’ or an action of some sort.

**Game 1**  Guess my favorite natural number: one of 0,1,2,..., coded below as 0,s(0),s(s(0)),...

```prolog
    game1 :- guess(0).
    guess(X) :- write(X), write(’?’), nl, read(yes).
    guess(X) :- guess(s(X)).
```

<table>
<thead>
<tr>
<th>?- game1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0?</td>
</tr>
<tr>
<td>no.</td>
</tr>
<tr>
<td>0(s)?</td>
</tr>
<tr>
<td>no.</td>
</tr>
<tr>
<td>s(0)?</td>
</tr>
<tr>
<td>no.</td>
</tr>
<tr>
<td>s(s)?</td>
</tr>
<tr>
<td>no.</td>
</tr>
<tr>
<td>s(s)?</td>
</tr>
<tr>
<td>yes.</td>
</tr>
<tr>
<td>yes</td>
</tr>
</tbody>
</table>

**Game 2**  Guess my favorite pair of natural numbers

```prolog
    game2 :- guess2([0,0]).
    guess2(X) :- write(X), write(’?’), nl, read(yes).
    guess2(X) :- ???
```

<table>
<thead>
<tr>
<th>?- game2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,0]?</td>
</tr>
<tr>
<td>no.</td>
</tr>
<tr>
<td>[s(0),0]?</td>
</tr>
<tr>
<td>no.</td>
</tr>
<tr>
<td>[0,s(0)]?</td>
</tr>
<tr>
<td>[s(s(0)),0]?</td>
</tr>
</tbody>
</table>
Game 3  Guess my list of natural numbers of length N

\[
\text{game3}(N) :- \text{??？}
\]
Game X  Guess my list of natural numbers of arbitrary finite length.

    gameX :- ???

-----------------------------------------------

  | ?- gameX.
 | []?
 | [0]?
 | [0,0]?
 | [s(0)]?
 | [0,0,0]?
 | [s(0),0]?
 | [0,s(0)]?
 | [s(s(0))]?
 | [0,0,0,0]?
 | [s(0),0,0]?
 | [0,s(0),0]?
 | [s(s(0)),0]?
 | [0,0,s(0)]?
 | [s(0),s(0)]?
 | [0,s(s(0))]?
 | [s(s(0)))]
 | [0,0,0,0,0]?
 | [s(0),0,0,0]?
 | [0,s(0),0,0]?
 | [s(s(0)),0,0]?
 | [0,0,0,0,0]?
 | [s(0),0,0,0]?
 | [0,s(0),0,0]?
 | [s(s(0)),0,0]?
 | [0,0,0,0,0]?
 | [s(0),0,s(0)]?
 | [0,s(0),s(0)]?
 | [s(s(0)),s(0)]?
 | [0,0,s(s(0))]?
 | [0,0,s(s(0))]?
Guess my favorite infinite bitstring $b_0b_1b_2\cdots$ where $b_n \in \{0,1\}$.

We can identify $b_0b_1b_2\cdots$ with the function mapping $n$ to $b_n$ — i.e., a function from the set $\omega = \{0, 1, 2, \ldots \}$ of natural numbers to $\{0,1\}$.

Let us write $2^\omega$ for the set of subsets of $\omega$, and identify each subset $A \subseteq \omega$ with the function $\chi_A : \omega \to \{0,1\}$ mapping $n \in \omega$ to

$$\chi_A(n) := \begin{cases} 1 & \text{if } n \in A \\ 0 & \text{otherwise.} \end{cases}$$

That is, $b_0b_1b_2\cdots$ corresponds to the subset $\{n \in \omega \mid b_n = 1\}$ of $\omega$, making Game $\infty$ equivalent to

guess my favorite set $A$ of natural numbers.

Playing Game $\infty$ defines a sequence $A_0, A_1, A_2, \ldots$ of guesses $A_i \subseteq \omega$, corresponding to a function $f : \omega \to 2^\omega$ with $f(n) = A_n$.

Cantor’s Theorem

There is no function $f : \omega \to 2^\omega$ such that for every $A \subseteq \omega$, there is an $n \in \omega$ with $f(n) = A$.

Proof Given $f : \omega \to 2^\omega$, define

$$A_f := \{n \in \omega \mid n \not\in f(n)\}$$

so that for all $n \in \omega$,

$$n \in A_f \text{ iff } n \not\in f(n)$$

whence $f(n) \neq A_f$. □

N.B. If we write out $f(0), f(1), \ldots$ as an infinite $(0,1)$-matrix for the infinite bitstrings

$$\begin{array}{ccccccc}
\chi_{f(0)}(0) & \chi_{f(0)}(1) & \chi_{f(0)}(2) & \cdots \\
\chi_{f(1)}(0) & \chi_{f(1)}(1) & \chi_{f(1)}(2) & \cdots \\
\chi_{f(2)}(0) & \chi_{f(2)}(1) & \chi_{f(2)}(2) & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{array}$$

then the idea is that $\chi_{A_f}$ diagonalizes out of this matrix.

The significance of Cantor’s theorem for search is that it says the set of infinite bitstrings cannot be searched one at a time.

This is an absolute limitation (just as the unsolvability of the Halting Problem is) in that it does not rely on some vague notion of feasibility.