# Cantor's proof

## Theorem 1

*Let $S$ be an infinite set and $\mathcal{P}(S)$ the power set of $S$. Then $S \not\sim \mathcal{P}(S)$.*

**Proof**

The proof is a diagonalization proof that assumes we have a surjection between $S$ and $\mathcal{P}(S)$ and then shows that such a surjection is not possible.

Assume we have a surjection $f : S \twoheadrightarrow \mathcal{P}(S)$.

So, $\forall s \in S$, $s$ is mapped to a subset $f(s)$ where $f(s) \subseteq S$.

# Cantor's proof

## Theorem 1

*Let $S$ be an infinite set and $\mathcal{P}(S)$ the power set of $S$. Then $S \not\sim \mathcal{P}(S)$.*

**Proof**

The proof is a diagonalization proof that assumes we have a surjection between $S$ and $\mathcal{P}(S)$ and then shows that such a surjection is not possible.

Assume we have a surjection $f : S \twoheadrightarrow \mathcal{P}(S)$.

So, $\forall s \in S$, $s$ is mapped to a subset $f(s)$ where $f(s) \subseteq S$.

For any $s$ either $s \in f(s)$ or $s \notin f(s)$.

# Cantor's proof

## Theorem 1

*Let $S$ be an infinite set and $\mathcal{P}(S)$ the power set of $S$. Then $S \not\sim \mathcal{P}(S)$.*

**Proof**

The proof is a diagonalization proof that assumes we have a surjection between $S$ and $\mathcal{P}(S)$ and then shows that such a surjection is not possible.

Assume we have a surjection $f : S \twoheadrightarrow \mathcal{P}(S)$.

So, $\forall s \in S$, $s$ is mapped to a subset $f(s)$ where $f(s) \subseteq S$.

For any $s$ either $s \in f(s)$ or $s \notin f(s)$.

Define the set $\tilde{S} = \{s \in S | s \notin f(s)\}$. So, $\tilde{S}$ is the set of all $s \in S$ such the $s$ itself is not a member of its image set $f(s)$. Observe that $\tilde{S} \subseteq S$ is a properly defined subset of $S$.

Since $f$ is surjective $\tilde{S}$ must have a pre-image point in $S$, say $\tilde{s}$, such that $f(\tilde{s}) = \tilde{S}$. Now $\tilde{S}$ is either empty or non-empty. We look at each case.

Since $f$ is surjective $\tilde{S}$ must have a pre-image point in $S$, say $\tilde{s}$, such that $f(\tilde{s}) = \tilde{S}$. Now $\tilde{S}$ is either empty or non-empty. We look at each case.

**Case $\tilde{S} = \varnothing$:** This implies that $\forall s \in S$, $s \in f(s)$. That is in each case $s$ is a member of the subset $f(s)$ to which it maps. This in turn means that for the set $\varnothing \in \mathcal{P}(S)$ there is no pre-image point in $S$ since $\varnothing$ does not contain any $s \in S$. So, $f$ cannot be a surjection.

Since $f$ is surjective $\tilde{S}$ must have a pre-image point in $S$, say $\tilde{s}$, such that $f(\tilde{s}) = \tilde{S}$. Now $\tilde{S}$ is either empty or non-empty. We look at each case.

**Case $\tilde{S} = \varnothing$:** This implies that $\forall s \in S$, $s \in f(s)$. That is in each case $s$ is a member of the subset $f(s)$ to which it maps. This in turn means that for the set $\varnothing \in \mathcal{P}(S)$ there is no pre-image point in $S$ since $\varnothing$ does not contain any $s \in S$. So, $f$ cannot be a surjection.

**Case $\tilde{S} \neq \varnothing$:** Now we try to determine the status of $\tilde{s}$ w.r.t $\tilde{S}$.

Since $f$ is surjective $\tilde{S}$ must have a pre-image point in $S$, say $\tilde{s}$, such that $f(\tilde{s}) = \tilde{S}$. Now $\tilde{S}$ is either empty or non-empty. We look at each case.

**Case $\tilde{S} = \varnothing$:** This implies that $\forall s \in S$, $s \in f(s)$. That is in each case $s$ is a member of the subset $f(s)$ to which it maps. This in turn means that for the set $\varnothing \in \mathcal{P}(S)$ there is no pre-image point in $S$ since $\varnothing$ does not contain any $s \in S$. So, $f$ cannot be a surjection.

**Case $\tilde{S} \neq \varnothing$:** Now we try to determine the status of $\tilde{s}$ w.r.t $\tilde{S}$.

Let $\tilde{s} \in \tilde{S} = f(\tilde{s})$. But by definition $\tilde{S}$ contains only those $s$ that are not in $f(s)$, that is $\tilde{s} \notin \tilde{S} = f(\tilde{s})$. A contradiction.

Since $f$ is surjective $\tilde{S}$ must have a pre-image point in $S$, say $\tilde{s}$, such that $f(\tilde{s}) = \tilde{S}$. Now $\tilde{S}$ is either empty or non-empty. We look at each case.

**Case $\tilde{S} = \varnothing$:** This implies that $\forall s \in S$, $s \in f(s)$. That is in each case $s$ is a member of the subset $f(s)$ to which it maps. This in turn means that for the set $\varnothing \in \mathcal{P}(S)$ there is no pre-image point in $S$ since $\varnothing$ does not contain any $s \in S$. So, $f$ cannot be a surjection.

**Case $\tilde{S} \neq \varnothing$:** Now we try to determine the status of $\tilde{s}$ w.r.t $\tilde{S}$.

Let $\tilde{s} \in \tilde{S} = f(\tilde{s})$. But by definition $\tilde{S}$ contains only those $s$ that are not in $f(s)$, that is $\tilde{s} \notin \tilde{S} = f(\tilde{s})$. A contradiction.

Let $\tilde{s} \notin \tilde{S}$. Then again by definition of $\tilde{S}$ since $\tilde{s}$ is not in $\tilde{S}$ it must be a member of $\tilde{S}$, that is $\tilde{s} \in \tilde{S}$. Again a contradiction.

This means $\tilde{s}$ does not exist. So, there is no $s \in S$ which is a pre-image of $\tilde{S}$ which is a subset of $S$ and therefore is in $\mathcal{P}(S)$.

So, this again means $f$ is not a surjection. $\square$

# Paradoxes of self reference: semantic paradoxes

- I am lying. Am I telling the truth or lying?

  If telling truth then by my own admission "I am lying". On the other hand if I am lying then I am telling the truth that "I am lying".

# Paradoxes of self reference: semantic paradoxes

- A barber is a person who shaves all those who do not shave themselves. Does the barber shave himself?

  If the barber shaves himself then he is a person who does not shave himself. And if he does not shave himself then the barber (that is himself) shaves him so he shaves himself.

# Paradoxes of self reference: semantic paradoxes

- Autological: a predicate/word that is applicable to itself. E.g. English, short,polysyllabic, word etc.

  Heterological: a predicate/word that does not apply to itself. E.g. Hindi, long,monosyllabic, red etc.

  Is heterological autological or heterlogical?

  Let it be autological. Then it applies to itself so heterological is heterological.

  Let it be heterological. So, heterological is heterlogical so it applies to iself. That is it is autological.

# Set theoretic paradoxes: Russell's paradox

**Normal set:** A set that does not contain itself as an element. E.g. Set of 2D triangles.
**Abnormal set:** A set that contains itself as an element. E.g. The complement of the set of 2D triangles.

Let R $\equiv$ set of all normal sets.

What is the status of R? Normal or abnormal.

Assume R is normal. Then it must be an element of R (the set of all normal sets) and is therefore abnormal. Contradiction.

Assume R is abnormal. Then R must be a member of R. But R is the set of all normal sets so R must be normal. Contradiction.

# Problem with axiom of unrestricted comprehension

Normally set membership is defined by a predicate $P$.

$\exists S \forall x.\ x \in S \leftrightarrow P(x)$

Let $P \equiv x \notin x$ and instantiate $x$ to $S$ giving:

$S \in S \leftrightarrow S \notin S$

ZFC has a restricted axiom of comprehension. $x$ has to be a member of a set and we can only create subsets.

$\{x \in U | P(x)\}$. It cannot create general sets like: $\{x | P(x)\}$

# Application to computability

Define a function *hetero* that checks if a predicate $p$ applies to itself.

*hetero(p)=not p(p)*

Calling *hetero* with predicate *hetero* as argument gives:

$$\begin{aligned} \textit{hetero(p)} &= \textit{not hetero(hetero)} \\ &= \textit{not (not hetero(hetero))} \\ &= \textit{not (not(not(hetero(hetero)))} \\ &= \ldots \end{aligned}$$

We have a non-terminating computation. So, *hetero* does not terminate on all inputs.

# Termination

Termination is a more general problem. Consider the function:

*factorial (x) = if (x==0 || (x==1)) then return 1*
*else return x * factorial(x-1)*

*factorial* does not terminate for negative $x$.

Assume we can define a boolean function *halts?(p, i)* which for any program $p$ and any input $i$ says whether or not $p$ will terminate on $i$ as input. It will return *True* if $p$ terminates on $i$ else *False*. *halts?(p, i)* terminates on all inputs.

Now we define *hetero* using halts
*hetero(p)= if halts?(p, p) then not p(p else True)*

So, a predicate $p$ is *hetero* if either it does not terminate when applied to itself or is false when applied to itself. So *not* $p(p)$ is true.

Let us calculate *hetero(hetero)*.

$$\begin{aligned} \textit{hetero(hetero)} &= \textit{if (halts?(hetero hetero)) then} \\ &\quad \textit{not hetero(hetero) else } \texttt{True} \\ &= \textit{not hetero(hetero)} \end{aligned}$$

Note that *hetero()* always terminates so executes the then statement. We get a contradiction:

*hetero(hetero)=not hetero(hetero)*.

This means that we cannot have a function like *halts?(p, i)* that itself terminates and can always tell whether or not $p$ terminates on $i$ for any $p$ and any $i$.

# Representation and interpreter

Supposing one says that passing functions and predicates as arguments is giving too much power to a language then we can do the following.

Assume that we pass a representation $r(p)$ to a changed function *hetero** which is passed a representation $r(p)$ of the predicate (which is data e.g. text of the function). Similarly, we have *halt?** which is passed $r(p)$.*hetero** now looks like:

```
hetero*(r(p))   =   if (halts*?(r(p),r(p))) then
                    not interpret(r(p),r(p))
                    else True
```

where *interpret* is an interpreter function such that:
$\forall p \forall i$ interpret$(r(p), i) = p(i)$.

Let us find the value of *hetero** with *p=hetero**:

$$
\begin{aligned}
\textit{hetero*(r(hetero*))} \quad &= \quad \textit{if (halts*?(r(hetero*),r(hetero*))) then} \\
&\qquad \textit{not interpret(r(hetero*), r(hetero*))} \\
&\qquad \textit{else } \texttt{True} \\
&= \quad \textit{not interpret(r(hetero*), r(hetero*))} \\
&= \quad \textit{not hetero*(r(hetero*))}
\end{aligned}
$$

A contradiction.

Note that *hetero** always halts and we are using the semantics of the *interpret* function.

# Halts function not computable

So we have the following theorem:

## Theorem 2

*In any programming language L that is powerful enough to write an interpreter for a representation of programs in L it is not possible to program the halts? or halts*? function in L.*

This theorem is true of all widely used programming languages.