COMP0017
Computability and Complexity Theory

Fabio Zanasi
http://www0.cs.ucl.ac.uk/staff/F.Zanasi/

Lecture twelve

Previously on COMP0017

We commenced our investigation of unsolvable problems.

We saw various techniques, like **mapping-reducibility**, for proving that a problem is unsolvable (undecidable or unrecognisable).

In this lecture

We continue our investigation of unsolvable problems, focussing on general results that tell us something about the nature of solvability.

We prove that all the problems decidable by a Turing machine are in a sense trivial (Rice's Theorem).

We prove that there are many more un-recognisable problems than recognisable ones (Cantor's diagonal argument).

Rice's Theorem

Seeking positive results

We saw that we can't decide whether a given Turing machine

- halts on a given input
- halts on empty input (the empty string)
- halts on any input
- is equivalent to another given Turing machine.

What problems about Turing machines we can decide then?

Seeking positive results

For instance, we can decide whether a given Turing machine

- always halts within a certain number of steps.
- has less than a certain number of states.

How much else can we get?

Remainder: languages

Languages are just subsets of Σ^*. Let's fix $\Sigma = \{0,1\}$, which can be done without loss of generality modulo a suitable encoding of Σ^* into $\{0,1\}^*$.

The **language of a Turing machine** $\mathcal{M}$ is $L_{\mathcal{M}} = \{x \in \{0,1\}^* \mid \mathcal{M} \text{ accepts } x\}$.

Properties of TM's languages

A **TM language property** P is a function from the set of Turing machines to {0,1} (false/true), such that $L_{\mathcal{M}} = L_{\mathcal{M}'}$ implies P($\mathcal{M}$) = P($\mathcal{M}'$).

This is just to ensure that P depends only on the language the machine describes. Example: ``halts in 42 steps'' is not a language property.

Such a property is **nontrivial** if there exists a TM $\mathcal{M}$ such that P($\mathcal{M}$) = 1 and a TM $\mathcal{M}'$ such that P($\mathcal{M}'$) = 0.

Formally, we will identify the TMs satisfying property P with the set { y ∈ Σ* | y = code($\mathcal{M}$) and P($\mathcal{M}$) = 1}.

Examples

{ y | y = code($\mathcal{M}$) and $L_{\mathcal{M}}$ is finite}

{ y | y = code($\mathcal{M}$) and $L_{\mathcal{M}}$ is empty}

{ y | y = code($\mathcal{M}$) and $L_{\mathcal{M}} = L$}

These are non-trivial language properties. Some machines have them, some others don't.

{y | y = code($\mathcal{M}$) and $\mathcal{M}$ recognises some language}

This is a trivial language property. Every TM recognises a language by definition.

Rice's Theorem

Theorem If P is a nontrivial language property, then the problem ``Does $\mathcal{M}$ have property P'' is undecidable.

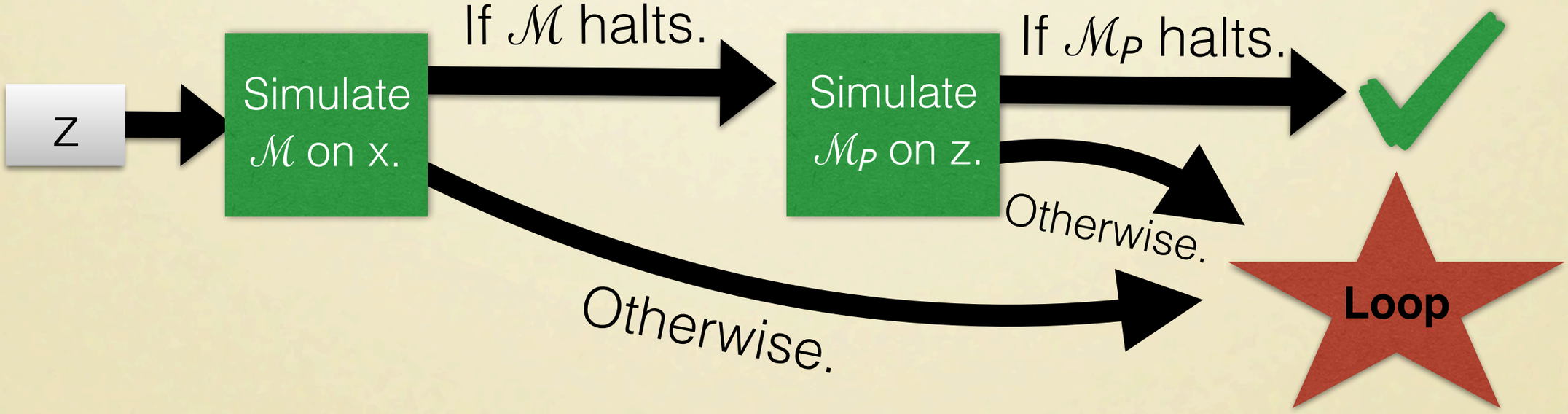
Proof strategy By contradiction. We are going to show that if ``Does $\mathcal{M}$ have property P'' was decidable, then the halting problem would be decidable.

Rice's Theorem

A TM whose recognised language is empty.

Proof

- Fix a property P. We first suppose that P($\mathcal{M}_\emptyset$) = 0.
- As P is non-trivial, we can pick $\mathcal{M}_P$ with P($\mathcal{M}_P$) = 1.
- Fix $\mathcal{M}$ and *x* as parameters. We construct the following TM $\mathcal{M}_{\mathcal{M},x}$:



Rice's Theorem

Proof

$\mathcal{M}$ halts on x. $\Rightarrow$ $\mathcal{M}_{\mathcal{M},x}$ halts on z whenever $\mathcal{M}_P$ halts on z. $\Rightarrow$ $L_{\mathcal{M}_P} = L_{\mathcal{M}_{\mathcal{M},x}}$ $\Rightarrow$ P($\mathcal{M}_{\mathcal{M},x}$) = P($\mathcal{M}_P$), so $\mathcal{M}_{\mathcal{M},x}$ has property P.

$\mathcal{M}$ does not halt on x. $\Rightarrow$ $\mathcal{M}_{\mathcal{M},x}$ never halts on z. $\Rightarrow$ $L_{\mathcal{M}_{\mathcal{M},x}} = \varnothing$ $\Rightarrow$ P($\mathcal{M}_{\mathcal{M},x}$) = P($\mathcal{M}_\varnothing$), so $\mathcal{M}_{\mathcal{M},x}$ does not have property P.

Conclusion: if we could decide if $\mathcal{M}_{\mathcal{M},x}$ has the property P, we could decide the halting problem.

Therefore, {y | y = code($\mathcal{M}$) and P($\mathcal{M}$) = 1} is undecidable.

Rice's Theorem

Proof We assumed P($\mathscr{M}_\varnothing$) = 0. What if P($\mathscr{M}_\varnothing$) = 1?

If so, repeat the very same argument, but this time for the property ¬P (``$\mathscr{M}$ has not property P'').

Notice that repeating the argument works because:

- as P is non-trivial, ¬P is non-trivial too.
- as P($\mathscr{M}_\varnothing$) = 1, then ¬P($\mathscr{M}_\varnothing$) = 0.

Thus we are able to get to the conclusion that {y | y = code($\mathscr{M}$) and ¬P($\mathscr{M}$) = 1} is undecidable. This implies that also {y | y = code($\mathscr{M}$) and P($\mathscr{M}$) = 1} is undecidable.

Disclaimer: decidable properties

Beware of this common misconception with Rice's Theorem.

Rice's Theorem speaks about **language** properties, not **machine** properties. It's about functions (specifications), not programs (implementations).

For instance, we cannot use Rice's Theorem to derive the undecidability of the halting problem (and related ones). The associated properties would refer to machines rather than languages.

Disclaimer: decidable properties

Loosely speaking, there are three kinds of properties that one may consider about Turing machines.

- **Language properties**. Non-trivial ones are undecidable by Rice's theorem.
- **Structural properties**, like ``$\mathscr{M}$ has 13 states''. These are typically decidable, since they can be checked statically on the given description of a Turing machine in encoded form.
- **Behavioural properties**, like ``$\mathscr{M}$ never moves left on input 0101'' or ``$\mathscr{M}$ halts on code($\mathscr{M}$)''. Some of these are decidable, some others aren't, and the classification is not obvious.

The cardinality of unsolvable problems

Aim

We want to prove that most languages are not recognisable (and, consequently, undecidable).

Ours is a **size** argument: there are **many more** languages than there are Turing machines.

In order to make this precise, we introduce an elegant proof method that turns out to be useful also in other settings: Cantor's diagonalisation argument.

Countably infinite sets

A set *S* is **countably infinite** if there is a total bijective function f: $\mathbb{N} \to S$.

Idea: a countably infinite set has as many elements as there are natural numbers.

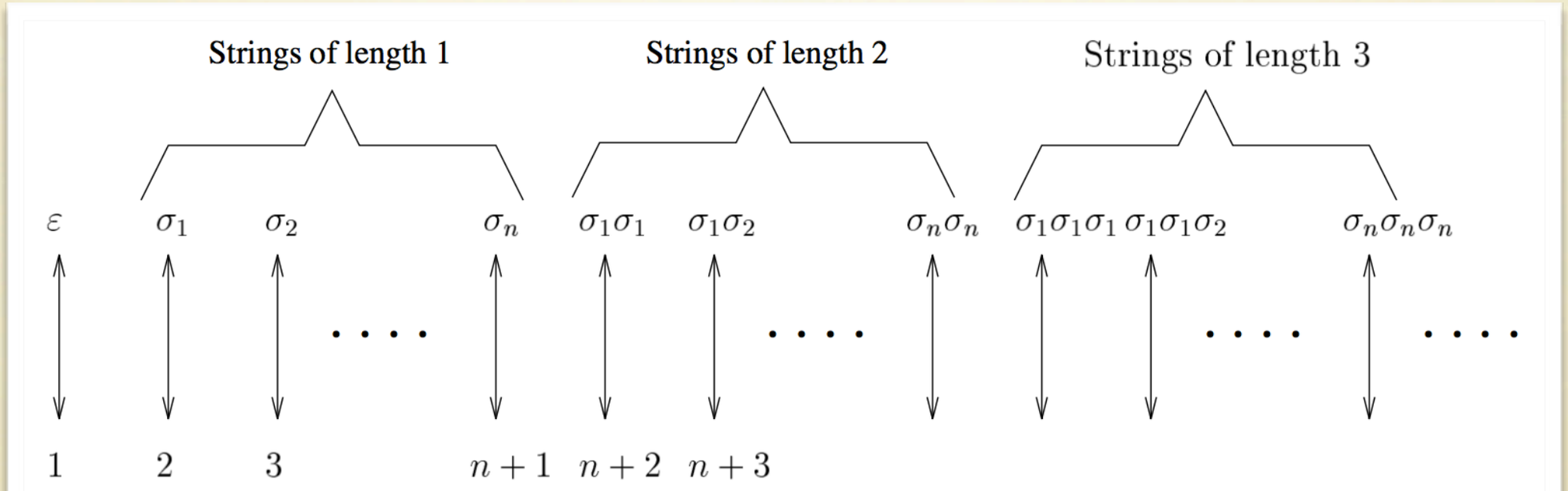
Example: the set of odd natural numbers, with the function f(n) = 2n ÷ 1.

Lemma If S_1 and S_2 are countably infinite, $S_1 \cup S_2$ is countably infinite.

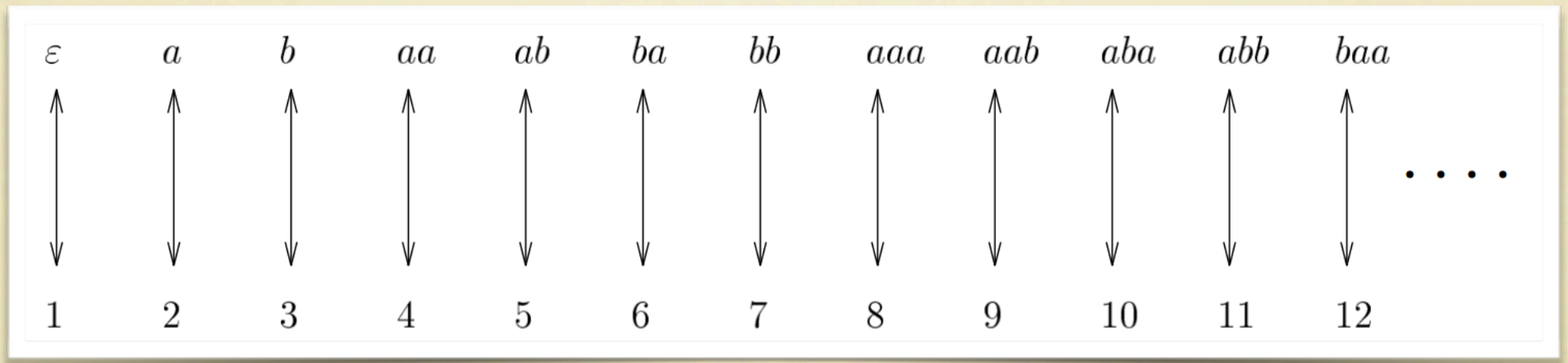
Countably infinite sets

Example: the set Σ^* of strings over a finite alphabet Σ.

Assuming $|\Sigma| = n$, the bijection with $\mathbb{N}$ is constructed as:



For instance, for $\Sigma = \{a,b\}$:



Counting Turing machines

We have already seen that any TM can be encoded as a string for an alphabet Σ with $|\Sigma| = 2$ (e.g., $\Sigma = \{0,1\}$).

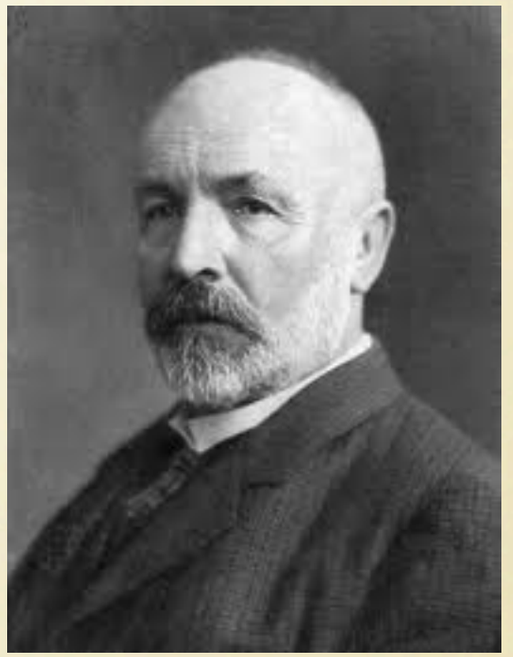
Therefore, the **set of all Turing machines** is countably infinite.

Also, the **set of all recognisable languages** is countably infinite. This is because any language is by definition recognisable if there is a TM recognising it.

Moreover, for the same reason, **the set of functions** $\mathbb{N}\to\mathbb{N}$ **computable** by a Turing machine is countably infinite.

Uncountable sets

Recap: countably infinite sets (including the set of TMs) are those that have as many elements as the natural numbers (they can be “counted”).



However, Georg Cantor (1845-1918) taught us that there are infinite sets that are **uncountable**. They have “more” elements than the natural numbers.

He introduced a technique (called **diagonalisation**) to show that a set is uncountable. He used it to show that the **real numbers** are uncountable.

Languages are uncountable

Call S_Σ the set of all languages over a finite alphabet Σ.

Theorem The set S_Σ is *not* countable.

Proof Recall that a language L is a subset of Σ*. We have already seen that Σ* is countably infinite, so we can write it as $\Sigma^* = \{\sigma_1, \sigma_2, \sigma_3, \ldots\}$.

Then a language, say $L_1 = \{\sigma_1, \sigma_4\}$, can be represented by

	σ_1	σ_2	σ_3	σ_4	σ_5	. . . .
L_1	1	0	0	1	0	. . . .

Languages are uncountable

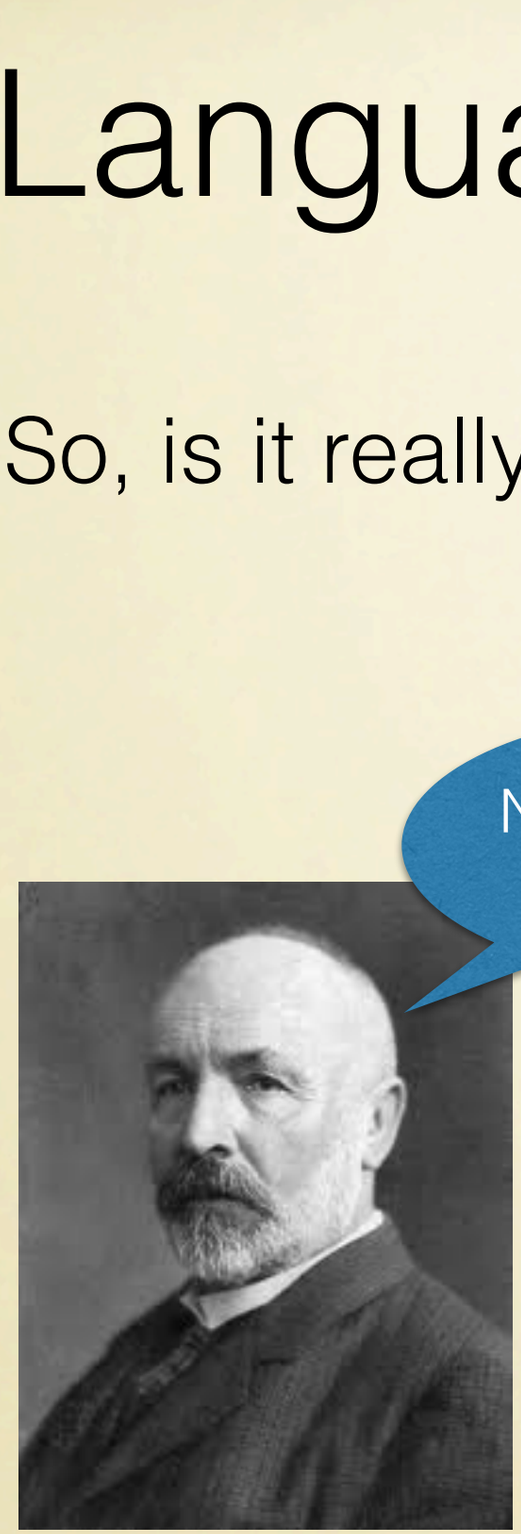
Any language over Σ can be represented in this way.

Towards a contradiction, suppose now that S_Σ is a countable set. If so, we can assign natural numbers to its elements, so that any $L_i \in S_\Sigma$ appears as one of the rows on the right.

	σ_1	σ_2	σ_3	σ_4	σ_5	$\cdots$
L_1	1	0	0	1	0	$\cdots$
L_2	0	1	1	0	1	$\cdots$
L_3	0	0	0	0	0	$\cdots$
L_4	1	1	1	0	1	$\cdots$
L_5	1	1	1	1	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Languages are uncountable

So, is it really any element L of S_Σ on some row?



	σ_1	σ_2	σ_3	σ_4	σ_5	$\cdots$
L_1	1	0	0	1	0	$\cdots$
L_2	0	1	1	0	1	$\cdots$
L_3	0	0	0	0	0	$\cdots$
L_4	1	1	1	0	1	$\cdots$
L_5	1	1	1	1	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Languages are uncountable

So, is it really any element L of S_Σ on some row?

Define L as 00110…, so that $\sigma_i \in L$ if and only if $\sigma_i \notin L_i$.

	σ_1	σ_2	σ_3	σ_4	σ_5	$\cdots$
L_1	1	0	0	1	0	$\cdots$
L_2	0	1	1	0	1	$\cdots$
L_3	0	0	0	0	0	$\cdots$
L_4	1	1	1	0	1	$\cdots$
L_5	1	1	1	1	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Then L is different from any language L_i on a row.

So L itself cannot be on a row! **Contradiction**.

Therefore, S_Σ is not countable.

Recap

For a finite alphabet Σ, we have seen that

- the set of languages recognisable by a Turing machine is countably infinite.
- the set of all languages is uncountable.

We have seen a few already:
$HALT^-$, EQ, EQ^-

Therefore, there exist languages that are not recognisable by any Turing machine.

Further question: how many un-recognisable languages are there?

How many unrecognisable sets?

Our answer will stem from a more general result.

Theorem If *S* is an infinite set that is not countable and *S'* is a countably infinite subset of *S*, then *S* \ *S'* is not countably infinite.

Proof Suppose that *S* \ *S'* is countably infinite. Then, because countably infinite languages are closed under union (see previous Lemma), (*S* \ *S'*) ∪ *S'* = *S* is countable. Contradiction!

Corollary The set of un-recognisable languages is not countably infinite — so there are more un-recognisable than recognisable languages.