Corollary 3:
Given a finite alphabet A, the set of all Turing-recognisable languages over A is countably infinite.

Proof:
An encoding <M> of a Turing machine M is the 7-tuple $(S, A, \tilde{A}, t, i, S_{acc}, S_{rej})$, which is a finite string over a language B that contains A and is finite.
By the theorem, $B^* = B^0 \cup B^1 \cup \ldots \cup B^\infty$ is countably infinite.
Since $\langle M \rangle \in B^*$, there are at most countably infinitely many Turing machines M that recognise languages over A
⇒ there are at most countably infinitely many Turing-recognisable languages over A.
We know we can build Turing machines with as large a set of states S as we want
⇒ the set of Turing machines that recognises languages over A cannot be finite
⇒ it is countably infinite. *(q.e.d)*

Proposition: Let A be a finite alphabet. Not all languages over A are Turing-recognisable.

Proof:
By *corollary 1*, the set of all languages over A is uncountably infinite.
By *corollary 3*, the set of all Turing-recognisable languages over A is countably infinite
⇒ there are many more languages over A than can be recognised by a Turing machine.
(q.e.d)

Remark:
This result makes a lot of sense because while we normally look at simpler, well-structured problems where there is a pattern, most languages over A have no pattern to them.
To understand more on the set of all Turing machines, we define the language
L_{TM} = {<M, w> | M is a Turing machine and M accepts w}
Here w is a string over the input alphabet A.
We will prove that L_{TM} is a Turing-recognisable language, but L_{TM} is **not** Turing-decidable.

Proposition: L_{TM} is a Turing-recognisable language.

Proof: We define a Turing machine U that recognises L_{TM}:
U = on input <M, w>, where M is a Turing machine and w is a string.
1. Simulate M on string w.
2. If M ever enters its accept state then accept.
   If M ever enters its reject state then reject.

U loops on input <M, w> if M loops on w ⇒ U is a recogniser but not a decider. (*q.e.d*)

Remark:
The Turing machine U is an example of the **universal Turing machine** first proposed by Turing in 1936. This idea of a universal Turing machine led to the development of stored-program computers.

NB: Philosophically, the universal Turing machine we just constructed runs into the following big issues:
1. U itself is a Turing machine. What happens when U is given an input <U, w>?
2. The encoding of a Turing machine is a string. What happens when we input <M, <M>> or even worse <U, <U>>?

We are getting very close to Russell's paradox, the set $\Gamma = \{D \mid D \notin D\}$ which showed the axioms of naive set theory were inconsistent and led to more complicated axioms.
In one case, these issues lead to showing the language L_{TM} cannot possibly be Turing-decidable.