90 % Decidable

Sophia Kolak

Problem: We will define a language L to be "90%"-decidable if there exists a Turing machine M that halts and accepts all inputs in L, and for every positive integer n, it halts and rejects on $\geq$ 90 % of the strings of length n that are not in L. On the remaining $<$ 10 % of strings of length n not in L, it may potentially run forever (but will never halt and accept). Suppose a language L is "90%"-decidable, and we have a reduction f that maps another language H into L, with only the usual guarantee that $f(x) \in L$ for $x \in H$, and $f(x) \in L$ for $x \in H$. Explain informally why we cannot conclude that H is also "90%" decidable. If f were a bijection, would this change?

Solution: The function f is defined such that:

$$if x \in H, f(x) \in L$$

$$if x \in H, f(x) \in L$$

We know that L is 90% decidable, but we do not know that H is 90% decidable. To see why this is the case, suppose that 100% of the strings $\in H$ are decidable, meaning that H is recursive. We can define the Turing machine M which halts and rejects on all strings $\in H$. Clearly this language H could still meet the definition that $x \in H$ implies $f(x) \in L$, if a string $\in H$ maps to more than one string $\in L$, or if a string $\in L$ maps to more than one string $\in H$. Since we constructed H to be 100% decidable, though, we know it is clearly not 90% decidable. Thus, we cannot assume that H is 90% decidable.

If we know that f is a bijection, however, this condition changes. Whereas before we could have multiple strings mapping onto the same string, if we have a bijection then there must be a one-to-one correspondence between strings in H and strings in L. This means the mapping of lets say 19 strings onto 9 strings could not possibly happen. Also, the mapping has to go both ways. Thus, L being 90% decidable means that 90% of the strings $\in H$ must be decidable since this is a direct correspondence.