Decidable and Semi-Decidable Languages

What is a word and what is a language: a brief reminder. In Computer Science:

- a *word* is any combination of symbols, and
- a *language* is any set of words.

Comment. Do not tell this to your friends who study linguistics, they will laugh at a suggestion that, e.g., a set {aha, uh-uh} consisting of two words is a language :-)

Which languages we studied earlier in this class.

- We started with *regular* languages, i.e., languages for which a finite automaton can decide whether a given word belongs to this language or not.
- Then, we proved that some languages are *not* regular. Specifically, we proved that the following language is not regular:

$$\{a^n b^n, n = 0, 1, 2, \ldots\} = \{\Lambda, ab, aabb, \ldots\}.$$

- So, we introduced a more complex idea of a computational device – a pushdown automaton.
- Then, we considered languages that can be described by pushdown automata – these languages also correspond to context-free grammars.
- Then, we had an example of a language that cannot be recognized by a pushdown automation: namely, we proved that the following language is not context-free:

$$\{a^n b^n c^n, n = 0, 1, 2, \ldots\} = \{\Lambda, abc, aabbcc, \ldots\}.$$

- To cover such languages, we introduced a new notion – a Turing machine.
- This notion should be sufficient for all computations: indeed, according to the Church-Turing thesis, anything that can be computed on any computational device can be computed on a Turing machine.

What next?

- Languages for which a finite automaton can decide whether a word belongs to this language or not are called regular.
- Languages for which a pushdown automaton can decide whether a word belongs to this language or not are called context-free.

What about Turing machines? Here we have the following definition.

Definition of a decidable language. A language L is called *decidable* if there exists an algorithm (or, equivalently, a Turing machine) that:

- given a word,
- returns "yes" or "no" depending on whether this word belongs to this language or not.

Not all languages are decidable.

- We have shown that no algorithm is possible that, given a program p and data d, checks whether p halts on d.
- In language terms, this means that the set of all pairs (p, d) for which p halts on d is *not* decidable. We will denote this set by H and call it a *halting set*.

What *can* we say about the halting set? OK, this set is not decidable, but what can we say about this set?

One thing we can do if we have a pair (p, d) consisting of a program p and data d is to run p on d and to wait until it halts. Then:

- if p halts on d, we will eventually know it;
- but, of course, if p does not halt on d, we will never know it.

Let us describe this property in general terms. For the halting language H, there exists a Turing machine that returns "yes" for all words from this language – and only for these words.

Sets with this property are called *semi-decidable*, *Turing-recognizable*, or *recursively enumerable*.

Comment.

- In the beginning, these sets were called recursively enumerable.
- Later on, "recursive" became a programming term – as in recursive methods.
- To avoid confusion, researchers invented a new term: semi-decidable.

Now you can (almost) tell the age of the textbook's author by what term he/she uses:

- older authors usually stick to the original term recursively enumerable, while
- younger authors prefer semi-decidable.

But, of course, it is the same notion (just like United States and Estados Unidos are the two names for the same country).

Let us now summarize what we learned.

Summary.

- A set L is called *decidable* if there exists an algorithm that:
  - given a word w,
  - returns "yes" if and only if the word w belongs to the language L and
  - returns "no" if and only if the word w does not belong to the language L.
- A set L is called *semi-decidable* if there exists an algorithm that:
  - given a word w,
  - returns "yes" if and only if the word w belongs to the language L.

Comment. This comparison explains where the term "semi-decidable" (i.e., "half-decidable") comes from. In both cases, there exists an algorithm that, given a word, returns "yes" or "no" depending on whether this word belongs to the language or not. The difference is that:

- for decidable sets, the algorithm is required to work for *all* the words: for the words that belong to the language and for words that do not belong to the language;
- in contrast, for semi-decidable sets, the algorithm is required to return the correct answer ("yes") only for words that belong to the language,

In other words, the algorithm corresponding to semi-decidable sets is required to work correctly only on one of the two cases – i.e., in one half of the cases. Hence such sets are called semi(half)-decidable.