Formal Methods in Software Engineering

Theory of Computation — Spring 2025

Konstantin Chukharev

§1 Computability

Computable functions

Definition 1 (Church–Turing thesis): *Computable functions* are exactly the functions that can be calculated using a mechanical (that is, automatic) calculation device given unlimited amounts of time and storage space.

"*Every model of computation that has ever been imagined can compute* only *computable functions, and* all *computable functions can be computed by any of several* models of computation *that are apparently very different, such as Turing machines, register machines, lambda calculus and general recursive functions.*"

Example: A partial function $f : \mathbb{N}^k \hookrightarrow \mathbb{N}$ is *computable* ("can be calculated") if there exists a computer program with the following properties:

- If $f(x)$ is defined, then the program terminates on the input x with the value $f(x)$ stored in the computer memory.
- If $f(x)$ is undefined, then the program never terminates on the input x.

Effective procedures

Definition 2: An **effective procedure** is a finite, deterministic, mechanical algorithm that guarantees to terminate and produce the correct answer in a finite number of steps.

§2 Decidability

Decidable sets

Definition 3 (Decidable set): Given a universal set $\mathcal{U}$, a set $S \subseteq \mathcal{U}$ is **decidable** (or **computable**) if there exists a computable function $f : \mathcal{U} \longrightarrow \{0, 1\}$ such that $f(x) = 1$ iff $x \in S$.

Examples:

- The set of all WFFs is decidable.
  - *We can check if a given string is well-formed by recursively verifying the syntax rules.*
- For a given finite set Γ of WFFs, the set $\{\alpha \mid \Gamma \vDash \alpha\}$ of all tautological consequences of Γ is decidable.
  - *We can decide $\Gamma \vDash \alpha$ using a truth table algorithm by enumerating all possible interpretations (at most $2^{|\Gamma|}$) and checking if each satisfies all formulas in Γ.*
- The set of all tautologies is decidable.
  - *It is the set of all tautological consequences of the empty set.*

TODO: undecidable sets (existence proof)

Semi-decidability

Suppose we want to determine $\Gamma \vDash \alpha$ where Γ is infinite. In general, it is undecidable.

However, it is possible to obtain a weaker result.

Definition 4 (Semi-decidable set): A set S is **computably enumerable** if there is an *enumeration procedure* which lists, in some order, every member of S: $s_1, s_2, s_3 \ldots$

Equivalently, a set S is **semi-decidable** if there is an algorithm such that the set of inputs for which the algorithm halts is exactly S.

Note that if S is infinite, the enumeration procedure will *never* finish, but every member of S will be listed *eventually*, after some finite amount of time.

Some properties:

- Decidable sets are closed under union, intersection, Cartesian product, and complement.
- Semi-decidable sets are closed under union, intersection, and Cartesian product.

Semi-decidability [2]

Theorem 1: A set S is computably enumerable iff it is semi-decidable.

Proof ($\Rightarrow$): If S is computably enumerable, we can check if $\alpha \in S$ by enumerating all members of S and checking if α is among them. If it is, we answer "yes"; otherwise, we continue enumerating. Thus, if $\alpha \in S$, the procedure produces "yes". If $\alpha \notin S$, the procedure runs forever. □

Proof ($\Leftarrow$): On the other hand, suppose we have a procedure P which, given α, terminates and produces "yes" iff $\alpha \in S$. To show that S is computably enumerable, we can proceed as follows.

1. Construct a systematic enumeration of **all** expressions (for example, by listing all strings over the alphabet in length-lexicographical order): $\beta_1, \beta_2, \beta_3, \ldots$
2. Break the procedure P into a finite number of "steps" (for example, by program instructions).
3. Run the procedure on each expression in turn, for an increasing number of steps (see dovetailing):
   - Run P on β_1 for 1 step.
   - Run P on β_1 for 2 steps, then on β_2 for 2 steps.
   - ...

Semi-decidability [3]

- Run P on each of $\beta_1, \ldots, \beta_n$ for n steps each.
- ...

4. If P produces "yes" for some β_i, output (yield) β_i and continue enumerating.

This procedure will eventually list all members of S. □

Semi-decidability [4]

Theorem 2: A set is decidable iff both it and its complement are semi-decidable.

Proof: Alternate between running the procedure for the set and the procedure for its completement. One of them will eventually produce "yes". □

Semi-decidability [5]

Theorem 3: If Σ is an effectively enumerable set of WFFs, then the set $\{\alpha \mid \Sigma \vDash \alpha\}$ of tautological consequences of Σ is effectively enumerable.

Proof: Consider an enumeration of the elements of Σ: $\sigma_1, \sigma_2, \sigma_3, \ldots$

By the compactness theorem, $\Sigma \vDash \alpha$ iff $\{\sigma_1, \ldots, \sigma_n\} \vDash \alpha$ for some n.

Hence, it is sufficient to successively test:

- $\emptyset \vDash \alpha$
- $\{\sigma_1\} \vDash \alpha$
- $\{\sigma_1, \sigma_2\} \vDash \alpha$
- ...

If any of these tests succeeds (each is decidable), then $\Sigma \vDash \alpha$.

This demonstrates that there is an effective procedure that, given any WFF α, will output "yes" iff α is a tautological consequence of Σ. Thus, the set of tautological consequences of Σ is effectively enumerable. □

§3 Complexity Zoo

Complexity classes

TODO

See also: https://complexityzoo.net/Petting_Zoo

TODO

- [x] Computability
- [x] Decidability
- [ ] Undecidable sets
- [x] Semi-decidability
- [ ] Complexity classes
- [ ] NP-completeness
- [ ] Polytime reductions
- [ ] Cook theorem