

Modellellenőrzés és tesztelés: egy kombinált megközelítés szoftverek verifikálására

Különböző szoftveres rendszerek életünk szerves részét képezik. Mára még a biztonság-kritikus rendszerekben is egyre inkább jelen vannak különböző, szoftvert használó komponensek, ahol a helyesség biztosítása jelentős feladat. Különböző megközelítések léteznek szoftverekben való hibakeresésre. Egyrészt a formális verifikáció, ami a szoftver matematikai modelljét vizsgálja, és matematikailag bizonyítja a hibák nemlétét. A formális verifikáció egy számításigényes feladat, hiszen megvizsgálja a szoftver összes lehetséges állapotát, viszont még a legegyszerűbb programoknak is hatalmas lehet az állapottere, ha egyenesen nem végtelen. Az elmúlt évtizedben számos megközelítés született, amik csökkenteni tudták ezen feladatok komplexitását, illetve a technológiai fejlődésnek köszönhetően a számításokhoz szükséges idő is csökkent. Viszont mindezek ellenére vannak olyan szoftverek, amiknek hibamentes működését még mindig nem tudjuk formális verifikációval igazolni. Egy másik merőben más megközelítés a tesztelés, ami hatékonyan képes hibákat találni a meglévő rendszerekben. Számításigényét tekintve sokkal szerényebb a formális verifikációnál, és az iparban elterjedten használt, viszont önmagában a helyes működés igazolására nem használható.

A munkám célja, hogy ezt a két különböző megközelítést kombináljam, ötvözve a két módszer előnyeit. Bemutatok egy új algoritmust, ami több különböző absztrakció alapú modellellenőrzési technikát használ a szoftver viselkedésének vizsgálatára. Ha ezek a módszerek képesek bizonyítani a szoftver helyességét, a verifikáció sikeres. Ellenben, ha a valós életben sokkal gyakrabban előforduló eset áll fent, név szerint, hogy a formális módszerek sem hiba létét, sem annak hiányát nem képesek igazolni, tesztelési technikák lesznek alkalmazva. Ehhez az algoritmus leállítja a formális verifikációt egy bizonyos határ után (ami lehet idő vagy memória korlát), és tesztek generál. A tesztek generálásához az algoritmus felhasználja az állapottér bejárása során gyűjtött információkat, és az állapottér csak azon részét teszteli, melynek helyességét a formális verifikáció nem tudta igazolni. Ezen kívül külön fókuszálók olyan hibák megtalálására, amelyeket a formális verifikáció önmagában nem tud megtalálni.

Az algoritmusomat az irodalomban fellelhető referencia modellekhez fogom mérni. Remélem, hogy ez az újszerű kombinációja a formális verifikációnak és tesztelésnek segíteni fog a biztonság-kritikus szoftverek minőségének javításában.