# Összefoglaló

Különböző szoftveres rendszerek életünk szerves részét képezik. A biztonságkritikus rendszerekben is egyre nagyobb teret kapnak a szoftver alapú megoldások, amelyek gyakran a korábbi, hardver alapú megoldások helyett kerülnek alkalmazásra.

Biztonságkritikus környezetben a helyesség biztosítása, illetve az előforduló logikai és programozási hibák megtalálása fontos és nehéz feladat. A gyakorlatban többféle megközelítést is használnak a szoftver hibák felderítésére, továbbá a hibamentesség bizonyítására. A megközelítések egy része leginkább hibák megtalálására használható, míg más megközelítések a helyesség bizonyítására is alkalmasak.

Egy gyakorlatban használt módszer a formális verifikáció, ami a szoftver matematikai modelljét vizsgálja, és matematikailag bizonyítja a specifikációnak megfelelő működést. A formális verifikáció egy számításigényes feladat, hiszen megvizsgálja a szoftver összes lehetséges állapotát, viszont még a legegyszerűbb programoknak is hatalmas lehet az állapottere, vagy sok esetben akár végtelen is. Az elmúlt évtizedben számos megközelítés született, amik hatékonyabbá tették a verifikációs algoritmusokat futási idő és memória felhasználás szempontjából. Ezen előrelépések, és a tudomány gyors fejlődése ellenére sem teljesen megoldott feladat azonban a formális verifikáció: mind az elméleti korlátok, azaz a számításelméleti komplexitásból fakadó nagy számításigény, mind pedig a gyakorlati rendszerek összetettsége, bonyolultsága a sikeres verifikáció ellen dolgoznak.

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 elegendő.

A munkám célja, hogy ezt a két különböző megközelítést kombináljam, ötvözve a két módszer előnyeit. Bemutatok egy új algoritmust, ami egy 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, azaz, hogy a formális módszerek sem hiba létét, sem annak hiányát nem voltak képesek igazolni, tesztelési technikák kerülnek alkalmazásra. Ehhez az algoritmus leállítja a formális verifikációt egy bizonyos határ után (ami lehet idő vagy memória korlát), és a formális verifikáció alapján nyert információkat felhasználva teszteket 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 fókuszálok olyan számábrázolási hibák megtalálására, amelyeket a formális verifikáció önmagában nem tud mindig megtalálni.