Grafikus modellezési nyelvek modelljeinek automatikus helyességellenőrzése

Schmidt Ákos V. Inf.

Konzulens: Varró Dániel, MIT

Napjainkban az MDA (modell-vezérelt architektúra) a szoftverfejlesztés legújabb trendje. Ennek koncepciója, hogy először egy platform független absztrakt modellt készítünk el valamilyen grafikus modellezési nyelv (legtöbbször UML) segítségével, majd a konkrét platformokra illeszkedő modelleket automatikus modelltranszformáció által származtatjuk. Végül automatikus kódgenerátor segítségével állítható elő a célalkalmazás igen jelentős része.

Grafikus modellezési nyelvek körében elterjedtek a metamodellezési és gráftranszformációs technikák, melyek egyszerre nyújtanak a mérnökök számára közérthető és grafikus, ugyanakkor matematikailag precíz specifikációs nyelvet. Mindez azonban nem garantálja, hogy az egyes rendszertervek, azaz a grafikus modellezési nyelvek modellpéldányai nem tartalmaznak tervezési, modellezési hibákat. Fontos szempont, hogy ezen hibákat minél korábbi fázisban automatikusan felismerjük minél több, a tervezés folyamatában előforduló modellezési nyelv esetére.

A dolgozat témája a metamodellezéssel és gráftranszformációval (grafikus modellezési nyelvvel) megadott konkrét modell példányainak helyességellenőrzése, mely tetszőleges, metamodellezéssel és gráftranszformációval definiált modellezési nyelvre adaptálható. A rendszer (CheckVML) elkészítését egy kidolgozott matematikai transzformációra alapoztuk, melyhez konkrét algoritmust készítettünk és Java nyelven implementáltuk. Bemutatjuk a használt algoritmust, mely a bemeneti modellünkből elkészít egy modellellenőrző független matematikai modellreprezentációt, majd ebből a formális leírásból állítjuk elő a konkrét modellellenőrző bemenetére adható kódot.

Maga a modellellenőrzés két lépcsőben történik: először az általunk írt alkalmazás elvégzi az előbb definiált transzformációkat, azaz legenerálja a modellellenőrző nyelvére (esetünkben SPIN bemenetre) a modellünket. Ezután a modellellenőrző segítségével formális követelményeket (LTL kifejezések, élőségi tulajdonság, stb.) vizsgálhatunk, melyekre a modellellenőrző automatikusan megadja a választ a modell állapotterének kimerítő bejárása után. A kimenet lehet elfogadó, ellenkező esetben pedig ellenpéldával szolgál, mely segíti a hiba okának megszüntetését.

A dolgozatban a téma elméleti hátterének bemutatása után ismertetjük a program logikai vázát és fontosabb algoritmusait, majd megvizsgáljuk a programunk és a modell-ellenőrző futási idejét különböző méretű bemeneti modellekre. Ezt két esettanulmány tanulmányozásán keresztül vizsgáljuk meg, mely példák gyakran fordulnak elő ebben a témakörben, mint benchmark (teljesítménymérő) alkalmazások.