Summary

Graph-based models are widely used in the development of complex, safety and business critical systems like automotive, avionics and financial software. Advanced modeling environments based on Domain Specific Languages (DSLs) (1) supports the development of models by continuously validating them, (2) derives different views to highlight task-specific aspects of the model, (3) automates several steps in the development (e.g. by code generators) and (4) provides mathematical analysis to check the correctness of the underlying design (e.g. by model checkers). Thus they can significantly improve the overall productivity of the development and quality of the product.

Despite the wide range of existing tool support, constructing a complex DSL and a modeling environment is a tedious task, and – like any software artifacts – they are error-prone. First, to define the structure of graph models of a language, a DSL is defined with a large number of complex, interacting constraints (i.e. design rules), that can easily be formalized incorrectly. This could result in inconsistent, incomplete or ambiguous languages. Moreover, errors in the implementation of the modeling environment inject errors to the generated code and invalidate the results of any verification process. Therefore, it is important to ensure the correctness of modeling tools themselves. As model-driven tools are frequently used in critical systems design, those tools should be validated with the same level of scrutiny as the underlying system as part of a software tool qualification process in order to provide trust in their output. Therefore software tool qualification raises several challenges for building trusted DSL tools for a specific domain.

The objective of this thesis is to improve validation and testing support for DSLs and modeling tools using automated model generation techniques. For this purpose, this thesis proposes a logic-based approach that able precisely capture the definitions of DSLs as logic theorems. Those theorems are analyzed by advanced logic solver methods in order to derive proofs or counter-examples for expected language properties or to create a set of different models that can be used as test data for modeling environments.

In my thesis, I present three main groups of contributions. (1) I propose an efficient logic solver algorithm for the generation of graph-based models that reason directly over graph structures using partial modeling and 3-valued logic with classic logic solving techniques. (2) I propose a translation and validation techniques for DSL specifications to detect language level inconsistencies, incompleteness, and ambiguity. (3) I propose an iterative, multi-step model generation approach that with three applications. First, (3/A) iterative generation that improves the scalability of existing solver-based model generation techniques. Next, (3/B) an iterative generation is able to measure and control the diversity of the generated models thus improve the quality of test suites. And finally, (3/C) it is able to incrementally resolve view model inconsistencies. I presented my results in the context of three case studies: the validation of an avionics architecture modeling language, the testing of an industrial statechart modeling environment, and view synchronization in a remote health monitoring system.

As a proof-of-concept demonstration of my conceptual results, I developed an open-source modeling tool VIATRA Solver framework, that uses two popular logic solvers (Alloy and Z3), and features the proposed graph solver algorithm, which scales 1-2 orders of magnitude better than existing solutions. The framework natively supports EMF-based (Eclipse Modeling Framework) modeling languages with VIATRA graph patterns for validation and testing and does not require additional theorem-proving skills.

Összefoglaló

Megbízható komplex rendszerek – autók, repülők vagy pénzügyi szoftverek – tervezése során széles körben alkalmaznak gráfalapú modelleket. Szakterület-specifikus nyelvekre (Domain-Specific Language, DSL) épülő fejlett modellezőeszközök jelentősen támogatják a fejlesztési folyamatot azáltal, hogy (1) a fejlesztés alatt álló modelleket folyamatosan ellenőrzik, (2) nézeti modellekkel kiemelik a modell különböző fontos aspektusait, (3) fejlesztési lépéseket automatizálnak például automatikus kódgenerátorok alkalmazásával, valamint (4) a tervek helyességét ellenőrizhetővé tehetik különböző analízis eszközök (például modellellenőrzők) alkalmazásával. Modellezőeszközök alkalmazásával tehát jobb minőségű szoftverek készíthetők, várhatóan kevesebb idő alatt.

Azonban, mint bármely szoftver, maguk a modellezőeszközök és modellezési nyelvek is tartalmaznak hibákat. Mindenekelőtt maguknak a nyelveknek a meghatározásához használt összetett tervezési szabályok is lehetnek ellentmondásosak, többértelműek vagy hiányosak. Továbbá a modellezőeszközben található hibák továbbterjedhetnek a generált kódba, ezáltal érvénytelenítve egy matematikai precíz analízis eredményét. Ezért a modellezőeszközök helyességének ellenőrzése kiemelten fontos. Ez különösen érvényes a biztonságkritikus rendszerek esetén, ahol a felhasznált eszközök alkalmazásához az eszközöknek ugyanolyan szigorú ellenőrzésen kell átesnie, mint magának a végterméknek.

Disszertációm célja modellezőeszközök és modellezőnyelvek ellenőrzésének és tesztelésének támogatása automatikus modellgenerálási technikák segítségével. Munkám során olyan logika-alapú megközelítést alkalmaztam, amely matematikai elméletek formájában képes precízen reprezentálni a vizsgált szakterület-specifikus nyelvek gráf-struktúráját. Ezek az elméletek matematikai következtetési módszerekkel elemezhetővé válnak, ezáltal bizonyítékot vagy ellenpéldát tudunk adni elvárt nyelvi tulajdonságok teljesülésére vagy megszegésére. Ezen felül, az automatikusan előállított helyes modellek tesztbemenetként is hasznosíthatóak.

Disszertációmban három területen mutatok be új tudományos eredményeket. Elsőként készítettem egy újszerű, hatékony modellgenerálásra alkalmas logikai következtető algoritmust, amely háromértékű parciális modelleket és klasszikus logikai algoritmusokat ötvözve közvetlenül gráfalapú logikai struktúrákon következtet. Másodjára, olyan transzformációs és ellenőrzési technikát javasoltam, amely logikai következtetők felhasználásával képes nyelvi szintű ellentmondások, hiányosságok és többértelműségek felfedésére. Végül harmadsorban olyan iteratív többlépéses generálási folyamatot javasoltam, amely nagyban javítja meglévő következtetőkre épülő tesztgenerátorok skálázhatóságát, képes mérni és szabályozni a generált modellek diverzitását ezzel javítani a generált tesztkészlet minőségét, és képes feloldani a nézeti modellekben megfogalmazott inkonzisztenciákat. Eredményeim gyakorlati alkalmazhatóságát három esettanulmány segítségével szemléltetem: egy repülőgép architektúra modellezőnyelv ellenőrzésével, egy ipari állapotgép modellezőeszköz tesztelésével, és távoli egészségügyi felügyeleti rendszer nézeti modelljeinek elemzésével.

Az elméleti eredményeimre építve egy nyílt forráskódú szoftver prototípust is elkészítettem, és publikusan elérhetővé tettem a Viatra Solver modellgenerátor keretrendszerben, ami integráltan használ két népszerű logikai következtetőt (Alloyt és Z3-at), valamint tartalmazza az új gráfalapú következtető algoritmust, amelynek segítségével 1-2 nagyságrenddel nagyobb modellek is előállíthatóak mint más, logikai következtetésen alapuló eszközökben. A keretrendszer közvetlenül támogatja az EMF (Eclipse Modeling Framework) és Viatra gráfmintákon alapuló modellezési nyelvek ellenőrzését.