Kivonat

Összetett kiberfizikai rendszerek tervezésére széles körben alkalmaznak modellvezérelt tervezést, amely fejlett modellező nyelvek bevezetésével megkönnyíti a rendszerek megvalósítását. A modellek alkalmasak formális módszerekkel való ellenőrzésre, ami biztonságkritikus feladatkörben - például okos járművek esetében - kiemelten fontos. Továbbá automatikus kódgenerátorok alkalmazásával a rendszer legfőbb komponensei automatikusan előállíthatóak, beleértve a komponensek helyes viselkedéséért felelős monitorozást is.

Azonban számos szoftverkomponensnek – különös tekintettel a korábban vagy külső felek által fejlesztetteknek – nem áll rendelkezésre megfelelően precíz modellje, ami megakadályozza a formális módszerek alkalmazását. Továbbá az elvárt viselkedés specifikációjának hiányában a monitorkomponensek sem képesek detektálni az esetleges hibákat. Különösképpen az időzített komponensek viselkedését nehéz formalizálni.

Dolgozatunk célja egy olyan specializált algoritmus kidolgozása, mely - megfigyelve egy időzített rendszer viselkedését - a megfigyelt események alapján képes formális modelleket szintetizálni. A tanulás támogatására az irodalomból ismert korszerű automatatanuló algoritmusokra támaszkodunk. Célunk ezen algoritmusok kiterjesztése a mérnöki tervezésben gyakran alkalmazott absztrakció hatékony kezelésével, ezáltal lehetővé téve a fókuszált tanulást.

Megközelítésünk újszerűségét a modellalapú bemeneti nyelv és ennek az időzített automatatanuló algoritmusokkal való kombinációja adja. Munkánk során elkészítettünk egy modellalapú automatatanuló keretrendszert időzített szoftverkomponensek mérnöki modelljének szintetizálására. Gráfmintaillesztő nyelvet használunk a tanulás fókuszálására, azaz az absztrakció definiálására. A definiált absztrakció alapján az automatatanuló algoritmus előállítja a rendszer időzített formális modelljét. Az elkészült szoftver felhasználását esettanulmányokkal demonstráljuk.

Módszerünk által lehetővé válik olyan szoftverkomponensek viselkedésének megtanulása, amelyekre eddigiekben nem volt lehetőség komplexitásuk, illetve időtől függő viselkedésük miatt. Az így létrejövő specifikációból a komponens viselkedését formálisan ellenőrizhetjük, anomáliadetektáló monitort származtathatunk, valamint dokumentációt, illetve teszteket generálhatunk.

Abstract

Model-driven engineering is a widely used approach for designing complex cyber-physical systems that is based on the use of high-level system modeling languages. Models can be used for formal verification, which is important for safety-critical systems, e.g. vehicular automation. Moreover, the components of the system can be generated using automatic code generation, including monitoring components that continously check the expected behavior.

However, for numerous software components - mainly legacy and third-party components - there is often no precise specification given. The lack of a formal specification prevents the use of formal verification. Without a specification the expected behavior cannot be monitored. The construction of real-time systems is especially challenging.

The goal of our work is to create a specialized algorithm to synthesize formal models of real-time systems by observing the events of a black-box system. For this purpose we use state-of-the-art automaton learning algorithms. Our goal is to extend these algorithms to focus the learning to efficiently capture the abstraction techniques used in engineering.

The novelty of our approach is a model-based input language and its combination with timed automaton learning algorithms. We developed a model-based automaton learning framework for synthesizing formal models of timed software components. We use a graph query language to focus the learning on important properties of the target component, while excluding unimportant details. Therefore the automaton learning algorithm produces the timed formal model of the target system. We demonstrate our approach with case studies.

As a result of our work, it is possible to learn the behavior of software components that have not been possible yet due to their complexity or time-dependent behavior. Additionally, the model produced during the learning can be used as documentation, to derive monitors for anomaly detection, or for automatically create test inputs.