MODELLELLENŐRZÉS ALKALMAZÁSA EGY BIZTONSÁGKRITIKUS RENDSZER VÉDELMI LOGIKÁJÁNAK VERIFIKÁCIÓJÁRA

Vörös András

Abstract
Verifying the correctness of embedded, safety critical systems is an important task, since the failure of these systems may have catastrophic consequences. Such system is for example a safety system in a nuclear power plant. During my work I have created and analyzed the model of a safety function in the Reactor Protection System of the Paks Nuclear Power Plant. I used coloured Petri nets for the modeling and I used model checking for the verification. The analysis was carried out by the model checking tool developed at our department. This is the first case where one could examine the full behavior of this system without any restrictions, and I successfully proved its functional correctness.

Key words:
Safety critical systems, model checking, verification, nuclear power plant, coloured Petri nets

Összefoglalás
Beágyazott biztonságkritikus rendszerek ellenőrzése fontos feladat. Ilyen többek között egy atomerőmű védelmi rendszere is, amely ha hibásan, nem a specifikációnak megfelelően működik, komoly anyagi károkat, emberi tragédiákat okozhat. Munkám során a Paksi Atomerőmű Reaktorvédelmi Rendszerének egyik biztonsági funkcióját modelleztem és vizsgáltam. A modellezést egy elterjedt matematikailag precíz leíró nyelv, a Petri-hálók segítségével végeztem, a verifikációra modellellenőrzést használtam. Az analízist a tanszékünkön fejlesztett modellellenőrző keretrendszer segítségével hajtottam végre. Az elvárt működést temporális logikák segítségével fogalmaztam meg. A modellellenőrzés során a teljes működési tartományt sikerült vizsgálni, míg korábbi megközelítések csak megkötésekkel tudták vizsgálni a modellt.

Kulcsszavak:
Biztonságkritikus rendszerek, modellellenőrzés, verifikáció, Paksi Atomerőmű, Színezett Petri-hálók

1. Bevezetés

A formális verifikáció alkalmazása egyre elterjedtebb a biztonságkritikus, elosztott és beágyazott rendszerek terén. Formális verifikáció segítségével a tervezés korai fázisában megtalálhatjuk a tervezési hibákat, vagy bizonyíthatjuk a tervek helyességét. A modellellenőrzés egy automatikus verifikációs módszer diszkrét, jellemzően véges állapotterű modellek ellenőrzésére. Az elmúlt 20 évben sok fejlődés történt a területen, új algoritmusok és alkalmazások jelentek meg.

A modellellenőrzés során általában a rendszer egy formális modelljéből indulunk ki, amelyet vagy megtervezünk, vagy pedig származtatunk informális és egyéb leírásokból. A modellellenőrzés a specifikációs kritériumok teljesülését ellenőrzi a modellen, azaz azt vizsgálja, hogy a specifikációnak

megfelelően terveztük-e meg a modellünket. Munkám során a Paksi Atomerőmű egyik biztonsági funkcióját modelleztem, és a megadott informális specifikáció alapján elkészítettem a formális követelményeket temporális logikák segítségével, majd a tanszékünkön fejlesztett eszközök segítségével ellenőriztem a rendszer helyességét.

2. Háttérismeretek

A modellellenőrzés előfeltétele a rendszer matematikailag precíz modelljének elkészítése, amelyhez jellemzően valamilyen formális modellezési nyelvet használunk. Ilyen formális leíró nyelv a *Petri-háló*.

A Petri-háló egy irányított, súlyozott, páros gráf, melyben az egyik pontosztály (P) elemeit *helyeknek*, a másik pontosztály (T) elemeit pedig *tranzícióknak* nevezzük. Egy irányított él egy tranzíciót köt össze egy hellyel vagy egy helyet egy tranzícióval. Az élekhez egy-egy pozitív egész számot rendelünk, ezeket *élsúlyoknak* nevezzük. A Petri-háló állapotát helyeken lévő tokenek segítségével fejezzük ki. A háló *tokeneloszlása* (állapota) egy M: P $\rightarrow \mathbb{N}$ függvény, amely minden helyhez egy nemnegatív egész számot rendel. A tokeneloszlás kifejezhető egy $m = [m_1 \dots m_n]^T$ oszlopvektorral is, ahol $m_i = M(p_i)$.

Egy Petri-háló dinamikus viselkedését a tranzíciók *tüzelései* határozzák meg. Egy tranzíció akkor tüzelhet, ha engedélyezett, azaz ha minden bemenő helyén legalább annyi token van, mint az él súlya. Az engedélyezett tranzíciók nemdeterminisztikusan tüzelhetnek. A tüzelés során a bemeneti helyekről az élsúlyoknak megfelelő számú tokent elvesznek, és a kimeneti helyekre a kimeneti élek súlyának megfelelő számú tokent kitesznek.

Sok esetben azonban nagyobb kifejező erejű formalizmusra van szükségünk, ilyenkor jelenthet segítséget a *színezett Petri-hálók* használata. A színezett Petri-hálók adatstruktúrákkal egészítik ki a Petri-háló formalizmust, őrfeltételek vezetnek be, amelyek segítségével bonyolultabb logikákat is megfogalmazhatunk, továbbá kompakt modellreprezentációt tesznek lehetővé.

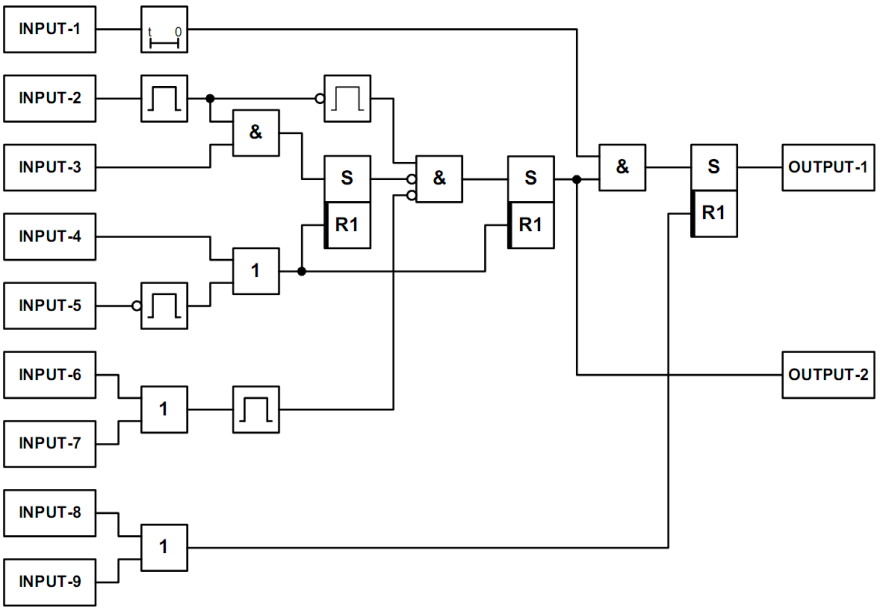
Az elkészített modelleken modellellenőrzést hajthatunk végre, amely során először felderítjük a modellek állapotterét, majd az állapottér által reprezentált viselkedések bejárásával vizsgáljuk a specifikáció teljesülését. Az ellenőrzendő tulajdonságokat *temporális logikák* segítségével tudjuk megadni. Munkám során az elágazó idejű *CTL* (*Computational Tree Logic*) temporális logikát használtam, amely útvonal- és állapotkvantorok segítségével definiálja a viselkedések temporális jellegét. Lehetőséget biztosít invariáns jellegű, élőségi, elérhetőségi vizsgálatokra, továbbá ezeknek a kombinációjára is. Az operátorok párokban fordulnak elő, az útvonalkvantorokat állapotoperátorok követik.

Elosztott, aszinkron rendszerek verifikációjára használt módszer az úgynevezett *szaturációs algoritmus*. Ez az algoritmus hatékonyan tudja aszinkron rendszerek hatalmas állapotterét felderíteni, továbbá a speciális, úgynevezett *szimbolikus technikák* segítségével az állapotteret tárolni. A

szaturációs algoritmus hatékony strukturális, azaz CTL modellellenőrzésre is lehetőséget biztosít, amely során kihasználja a használt iterációs stratégia előnyeit.

3. Biztonságkritikus rendszer verifikációja

Munkám során a Paksi Atomerőmű egyik biztonsági funkcióját [2] ellenőriztem formális módszerek segítségével. A konkrét funkció az atomerőmű Reaktorvédelmi Rendszerének az egyik indító logikája. A modul célja, hogy detektálja, ha a reaktor primer köréből víz szivárog át a szekunder körbe. Ezt a meghibásodást nevezzük *PRISE* eseménynek. Ennek az eseménynek a bekövetkeztét kell detektálnunk, és szükség esetén a védelmi működéseket elindítani.



1. ábra. *Indító logika blokkdiagramja*

Az 1. ábra mutatja be az indító logika funkcionális blokkdiagramját. A modell a következő elemekből épül fel: pulzusgenerátorok, késleltetők, logikai ÉS kapuk, logikai VAGY kapuk, inverterek és SR flip-flopok. A késleltető elem a bemenetén megjelenő felfutó élet a kimenetére csak bizonyos idő után engedi át. Ezzel szemben a pulzusgenerátor a bemeneten detektált felfutó él hatására a kimenetén adott hosszúságú impulzust generál. Az 1. ábrán az **Input-1** bemenethez egy késleltető, az **Input-2** bemenethez pedig egy pulzusgenerátor kapcsolódik. A logika a bemenő jeleket vizsgálja, azok temporális jellemzőiből következtet a PRISE esemény bekövetkeztére. Erre azért van szükség, mert nem mindegyik bemeneti jel származik megbízható szenzorból, továbbá a bemenő jelek meghatározott sorrendben történő aktiválódása jelzi, hogy a PRISE esemény bekövetkezett. Ezáltal kiszűrhető más események hibásan PRISE eseményként történő azonosítása.

3. Vizsgálatok

A funkcionális blokkdiagram alapján elkészítettem a vezérlő színezett Petri-háló alapú modelljét a *PetriDotNet* [1] keretrendszerben. Az elkészített modell állapotterét az algoritmus paramétereit megfelelően kiválasztva sikerült felderítenie az eszköznek: az állapottér nagyságrendileg 10^{12} állapotot tartalmaz. Megvizsgáltam, hogy előfordulhat-e a modellben holtpont, amikor a működés indokolatlanul leáll. Ezt a következő CTL kifejezéssel vizsgáltam: **AG** (**EX** *true*). A kifejezés jelentése: minden állapotban igaz, hogy továbbléphetünk egy következő állapotba. A modellellenőrzés igaz eredményt adott. Emellett megvizsgáltam, történhet-e téves riasztás, azaz csak akkor aktiválódik-e a logika, ha bekövetkezett a PRISE esemény. Ezt a következő CTL kifejezés segítségével vizsgáltam: ¬(**E** (¬[*PRISE-esemény*] **U** [*Riasztás*])). Ez a vizsgálat is sikeres eredményt adott, azaz a tervek megfelelnek a specifikációnak. A helyes működés szempontjából kritikus, hogy ha bekövetkezett PRISE esemény, akkor riasztást kell kiadni. Ehhez a következő CTL specifikációt vizsgáltam: **EF**(*PRISE-esemény* & **EG**(¬*Riasztás* & ¬*Reset-esemény*)). Ilyen állapotot nem tudunk elérni, a modellellenőrzés hamis eredményt adott. Tehát ha bekövetkezett PRISE esemény, akkor mindenképpen riasztást fogunk kiadni. Mivel az ellenőrzés nem fedett fel sem téves riasztást, sem indokolatlan riasztás elmaradást, az indító logika helyesen fog működni.

4. Összefoglaló

Munkám során a Paksi Atomerőmű egyik védelmi funkciója indító logikájának helyes működését vizsgáltam. Elkészítettem a logika formális modelljét, majd a specifikációs kritériumokat temporális logikák használatával megfogalmaztam. Ezután elvégeztem a modellellenőrzést, amely során bebizonyosodott, hogy az indító logika a specifikációnak megfelelően működik.

Irodalom

[1] Vörös, A., Darvas, D., Bartha, T.: Bounded Saturation Based CTL Model Checking, Tallinn University of Technology, Institute of Cybernetics, Proceedings of the 12th Symposium on Programming Languages and Software Tools (SPLST'11), Tallinn, 2011, pages 149–160.

[2] Németh, Erzsébet; Bartha, Tamás: *Formal verification of safety functions by reinterpretation of functional block based specification*, FMICS 2008. 13th international workshop on formal methods for industrial critical systems. L'Aquila, 2008. 219-234

Vörös András, doktorandusz

Munkahely: Budapesti Műszaki és Gazdaságtudományi Egyetem, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszereke Tanszék

Cím: 1117 Budapest, Magyar Tudósok körútja 2.

E-mail: vori@mit.bme.hu