Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Gyakorlatorientált formális módszerek az ipari vezérlőrendszerek szoftverfejlesztésének támogatására

Tézisfüzet

Darvas Dániel

Témavezető:
Majzik István, PhD (BME)

Külső konzulens:
Enrique Blanco Viñuela, PhD (CERN)

Budapest, 2017.

1. Előzmények és célok

A *szolgáltatásbiztonság* (dependability)[1] egy összefoglaló fogalom, amely magába foglalja a rendelkezésre állást, a megbízhatóságot, a biztonságosságot, az integritást és a karbantarthatóságot. Ez egy elvárt tulajdonság, különösen kritikus rendszerek esetén. Egy *hibajelenség* (failure) a rendszer elvárt működésétől való megfigyelhető eltérés, amely tehát rontja a szolgáltatásbiztonságot. A hibajelenség oka egy *hiba* (error) továbbterjedése. A hiba egy hibajelenséghez vezető (belső) rendszerállapot. A hibák okai a *meghibásodások* (fault) [Avi+04].

A kívánt szolgáltatásbiztonsági szint elérésére számos módszer áll rendelkezésre: *hibamegelőzés* (fault prevention), *hibatűrés* (fault tolerance), *hibamegszüntetés* (fault removal) és *hibaelőrejelzés* (fault forecasting) [Avi+04]. A formális módszerek jól ismert technikák a fejlesztési és bizonyos működési hibák megelőzésére azáltal, hogy matematikai alapokon nyugvó, egyértelmű módszereket biztosítanak a rendszerkövetelmények leírására és ellenőrzésére (verifikációjára) [Mar94]. A formális verifikáció és formális specifikáció egyre gyakrabban használatos *biztonságkritikus* felhasználási területeken, ahol a hibák következményei *katasztrofálisak* [Avi+04; Woo+09]. Ez lehet azért, mert egy egyedi hiba balesetet vagy komoly gazdasági kárt okozhat (pl. vasúti rendszereknél [LSP07], repülőiparban [Sou+09], űrbeli alkalmazásoknál [Hav+00]), vagy mert a nemkívánatos következmény nagyszámú rendszert érint (pl. tömeggyártott processzorok [Fix08; Kai+09]), ezáltal okozva összességében magas költséget.

Ipari vezérlőrendszereket számos különböző szituációban használunk. Ha a vezérlőrendszer biztonságkritikus feladatot lát el, az IEC 61508-2 (MSZ EN 61508-2) szabvány [I61508-2] különféle elvárt fejlesztési és ellenőrzési módszereket definiál az egyes *biztonságintegritási szintekhez* (safety integrity level, SIL), az elviselhető veszélygyakoriságtól (veszélyes hibák valószínűségétől) függően. Az ipari vezérlőrendszerek több részből állnak. Az egyik kulcsösszetevő tipikusan egy *programozható logikai vezérlő* (programmable logic controller, PLC): egy robusztus, konfigurálható, specializált számítógép, amely vezérlési feladatokat lát el. Bizonyos esetekben a PLC-k működése biztonságkritikus, de gyakran – további biztonsági rendszereknek vagy eljárásoknak (pl. fizikai akadály, független reléalapú biztonsági rendszer) köszönhetően – a PLC-alapú vezérlők biztonságintegritási szintje SIL 1 (a legalacsonyabb IEC 61508-által definiált szint) alatti. Bár ezekben az esetekben a várható hibajelenségek ritkák vagy nem katasztrofálisak, ez nem jelenti azt, hogy egy meghibásodás (pl. egy *szolgáltatáskiesés*) nem okozhat komoly anyagi kárt. Ugyanakkor az alacsony SIL általában alacsony verifikációs költségvetéssel is együtt jár. Ezekben az esetekben a „nehézsúlyú” formális módszerek (pl. B módszer vagy Z specifikációra; tételbizonyítók kézi használata verifikációra) használata túlzottan sok erőforrást igényelne. A nehéz használat (magas képzési költségek, speciális szakértelem szükségessége) mellett a formális verifikáció alkalmazásának egy másik gyakori akadálya ezen módszerek teljesítménye. Például egy modell viselkedésének (állapotterének) kimerítő ellenőrzése egy igen számításigényes feladat, ezért számos algoritmus nem tud a valós, ipari problémák méretéhez felskálázódni.

Célok. E kutatás fő célja a formális módszerek alkalmazhatóságának vizsgálata az ipari vezérlőrendszerek területén, és ez alapján specifikációs és verifikációs módszerek javaslata. A fentiek alapján az alkalmazás két fő akadálya a *teljesítmény* és *használhatóság*. Ez a disszertáció különböző megoldásokat javasol mindkét problémára. Egyrészt megcélozza a verifikációs algoritmusok hatékonyságának növelését, másrészt pedig könnyen használható, gyakorlatorientált (specifikációs és verifikációs) módszereket javasol, amelyek alkalmazhatók az ipari vezérlőrendszerekben használt PLC-programokra, amelyek esetén a „nehézsúlyú” módszerek használata nem szükséges vagy nem kivitelezhető. A javaslatra kerülő módszereknek figyelembe kell venniük a megcélzott felhasználási terület specialitásait.

[1] Ebben a doktori értekezésben az Avižienis, Laprie *et al.* által javasolt taxonómiát [Avi+04] használom, amely röviden bevezetésre és összefoglalásra kerül a következő két bekezdésben.

A tézisfüzet felépítése. A tézisfüzet felépítése a következő. Az 1. szakasz további része röviden bemutatja a formális verifikáció (1.1. szakasz) és formális specifikáció (1.2. szakasz) fontosabb alapelveit és főbb módszereit. Ezt a részt az 1.3. szakasz zárja le a jelen munkában megcélzott kihívások összefoglalásával. Ezt követően a 2. szakasz bemutatja a követett kutatási módszertant és a doktori értekezés új eredményeit. Utána a 3. szakasz áttekinti az új eredmények gyakorlati felhasználását. Végül a 4. szakasz tartalmazza a szerző publikációinak listáját.

1.1. Bevezetés a formális verifikációba

A *verifikáció* egy „rendszer vagy komponens értékelésének folyamata annak megállapítása céljából, hogy a termék [...] megfelel-e a kezdeti feltételeknek”[2] [I1012]. A *formális verifikáció* olyan, matematikai alapokon nyugvó módszereket foglal magában, amelyek pontosan képesek megállapítani megadott formális követelmények teljesülését.

„A *modellellenőrzés* egy automatizált [formális verifikációs] módszer, amely egy rendszer megadott véges állapotú modelljén egy adott formális tulajdonságra szisztematikusan megállapítja, hogy ez a követelmény az adott modellre teljesül-e”[3] [BK08]. Pontosabban megfogalmazva a modellellenőrzés egy módszer egy (Kripke-struktúraként adott) modell összes olyan állapotának megkeresésére, amelyek teljesítik a megadott temporális logikai kifejezést [Cla08]. Ezt a módszer először Edmund M. Clarke és E. Allen Emerson írta le [CE82], valamint tőlük függetlenül Jean-Pierre Queille és Joseph Sifakis [QS82].

Mivel a modellellenőrzés automatikus, egy megfelelő jelölt lehet „könnyűsúlyú” formális verifikációra, alapjául szolgálhat egy „gombnyomásra működő technikának”, amely használata nem igényel komoly hozzáértést a felhasználótól [BK08]. Továbbá a modellellenőrzés képes diagnosztikai szekvenciákat (ellenpéldákat vagy tanúkat) szolgáltatni, amelyek hasznos visszajelzésül szolgálnak a talált problémákkal kapcsolatban.

Modellezési és követelményleírási formalizmusok. Bár a modellellenőrzés definíció szerint Kripke-struktúrákon alapul, a modellek reprezentációjára gyakran magasabb szintű modellezési nyelveket használunk, például Petri-hálókat [Mur89], (időzített vagy időzítetlen) automatákat [AD94] vagy processzalgebrát [Fok00]. Hasonlóképp többféle temporális logika használható az ellenőrizendő követelmények leírására is. A leggyakrabban használtak a lineáris temporális logika (linear temporal logic, LTL) és a számítási fa logika (computation tree logic, CTL). Az egyes formalizmusok különböző kifejezőerővel bírnak, továbbá a modellellenőrzési algoritmusok közt is jelentős eltérések lehetnek attól függően, mely logiká(ka)t támogatják.

Modellellenőrzési módszerek. E. Clarke, a modellellenőrzés egyik megalapozója szerint a modellellenőrzési algoritmus egy „intelligens kimerítő keresés az állapottérben annak megállapítására, hogy a specifikáció teljesül vagy sem”[4] [CES09]. A rendszer elérhető állapotai számának növekedésével az állapottér-felderítő algoritmusok „intelligenciája” egyre fontosabbá válik. Az ún. *explicit* módszerek az adott modell minden egyes állapotát külön-külön képezik le. Ez lehetővé teszi egyszerű gráfelméleti algoritmusok használatát, de nem képes megfelelő megoldást nyújtani a nagyméretű modellek esetére, ahol az állapotok egyesével történő kezelése nem lehetséges. A nagy állapottereknek számos oka lehet, pl. a nagyszámú bemenet vagy kimenet,

[2] Eredeti idézet: „[t]he process of evaluating a system or component to determine whether the products [...] satisfy the conditions imposed at the start” [I1012].

[3] Eredeti idézet: „Model checking is an automated [formal verification] technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model.” [BK08].

[4] Eredeti idézet: „intelligent exhaustive search of the state space to determine if the specification is true or not” [CES09].

vagy a rendszer konkurens működése. Ez a jól ismert *állapottér-robbanás*. Az évek során számos megoldás született e probléma kezelésére:

- *szimbolikus algoritmusok*, amelyek az állapotteret egy tömörebb, kódolt formában tárolják, pl. döntési diagramok segítségével,
- *absztrakciók*, amelyek egyszerűsítik a modellt, így érve el kisebb állapotteret,
- *korlátos algoritmusok*, amelyek az állapottér-bejárás (kezdőállapottól számított) mélységét korlátozzák a felderített állapottér méretének csökkentése érdekében.

Az első szimbolikus algoritmusok bináris döntési diagramokon alapultak [Bur+92]. Azóta új algoritmusok születtek, különböző bejárási stratégiákat és adatszerkezeteket alkalmazva. Az egyik legbiztatóbb módszer a *szaturáció* (saturation) [CLS01], amely „különösen jól teljesít olyan diszkrét rendszerekre alkalmazva, amelyek relatíve kis alrendszerektől függő, aszinkron eseményeket tartalmaznak”[5] [CZJ12]. Az, hogy az algoritmus „jól teljesít” azt jelenti, hogy a verifikálható modellek és követelmények köre nagyobb, illetve a verifikáció futásideje rövidebb, de nem jelenti azt, hogy ennek az algoritmusnak nincsenek a teljesítményigényből adódó korlátai. Bizonyos esetekben a verifikáció végrehajtásához túlzottan sok memória lenne szükséges, így a verifikáció meghiúsul.

A modellellenőrzés kihívásaira nincsen univerzális megoldás: minden módszernek megvannak a maga hátrányai és korlátai. Egyes megoldások már korábban megpróbálták többféle módszer ötleteit ötvözni, pl. [Cha+02; Cop+01; CNQ05]. A szaturációalapú modellellenőrzés egyik lehetséges továbbfejlesztése lehet a korlátos modellellenőrzéssel való integráció, amely – a szerző tudomása szerint – az itt bemutatott munka előtt nem került tanulmányozásra. Ez a bővítés segítheti az ipari vezérlőrendszerek verifikációját is azáltal, hogy a korábbi verifikációs teljesítmény javul (pl. a [*c*28]-belihez képest). E lehetőség értékelése a jelen értekezés egyik kihívása (1. kihívás).

Modellellenőrzés a PLC-programok fejlesztési folyamatában. A modellellenőrzés már bizonyította hatékonyságát különböző felhasználási területeken [Cla08]. Ugyanakkor a szükséges formális modellek és temporális logikai kifejezésekként megfogalmazott követelmények biztosítása nehéz feladat, különösen a formális módszereket nem ismerő személyek számára. Továbbá a modellellenőrzéshez gyakran kézi módosításokra, finomhangolásra van szükség, hogy megfelelő teljesítményt sikerüljön elérni. Emiatt az alkalmazás erőforrás-igényes, ami meggátolhatja a modellellenőrzés használatát a PLC-programok fejlesztési folyamatában.

Az akadémiai algoritmusfejlesztés nagy teljesítményű, általános modellellenőrző eszközöket eredményezett (pl. UPPAAL [Amn+01], LTSmin [Kan+15], NuSMV/nuXmv [Cav+14]). Ezek az általános célú eszközök viszont nem tudják javítani a verifikációs módszerek szakterület-specifikus használhatóságát. A használhatóság javításához és a formális módszerek ipari vezérlőrendszerek fejlesztési folyamatába történő integrációjához a fókuszt specifikusan erre a szakterületre kell helyezni.

Bár a PLC-alapú ipari vezérlőszoftverek modellellenőrzését már tanulmányozták pl. a [GSF08; SD08; BBK12] munkákban, ezek nem nyújtottak általános, valós problémákra alkalmazható megoldásokat, vagy ezt nem demonstrálták. A modellellenőrzés adaptálása a PLC-programok fejlesztésének területére, annak a PLC-fejlesztők általi közvetlen felhasználhatóságának biztosítása és a módszer skálázhatóvá tétele jelen munka kihívásai (2. és 3. kihívás). Ezek mellett külön figyelmet kell fordítani a PLC-k egy speciális ágára, a biztonsági PLC-kre (fail-safe vagy safety PLC). Annak érdekében, hogy ezek a szoftverek megfelelő minőséget érjenek el, speciális korlátozások és fejlesztési módszerek tartandók be (pl. kódolási konvenciók, programozási megkötések) és ezeket is figyelembe kell venni a javasolandó megoldásnál (4. kihívás).

[5] Eredeti idézet: „tends to perform extremely well when applied to discrete-event systems having multiple asynchronous events that depend and affect only relatively small subsystems” [CZJ12].

1.2. Bevezetés a formális specifikációba

A *követelménymérnökség* (requirements engineering) tevékenységek egy csoportja a tervezendő rendszer céljainak, képességeinek, korlátainak és feltételezéseinek feltárására, kiértékelésére és dokumentálására [Lam09]. A *(követelmény)specifikáció* a „létrehozandó rendszer elfogadott jellemzőinek részletezése, strukturálása és dokumentálása”[6] [Lam09]. A [Lam00] alapján a *formális specifikáció* „egy rendszer által teljesítendő tulajdonságok kifejezése valamilyen formális nyelven, bizonyos absztrakciós szinten”[7]. Az egyik tényező, ami miatt a formális specifikáció folyamata nehéz az, hogy a „specifikációk sosem formálisak a kezdetekben”[8] és „nehéz [a formális specifikációkat] elkészíteni és értékelni”[9] [Lam00]. A specifikációk formalizálásától várt haszon „a formalizálás precízségének egy magasabb foka [...], precíz értelmezési szabályok és sokkal kifinomultabb validációs és verifikációs módszerek”[10] [Lam09].

A formális specifikációt már az 1960-as évek végétől tanulmányozzák és azóta számos módszer született. A Petri-hálók [Mur89], Lotos [I8807], a B módszer [Abr96], Z [I13568], vagy a kommunikáló szekvenciális folyamatok (communicating sequential processes, CSP) [Hoa85] széles körben ismert technikák. Bár ismertek, ezek a módszerek nem terjedtek el széles körben az iparban [Kni+97], mivel túl összetettek, mély matematikai előismereteket igényelnek, vagy túl magas az absztrakciós szintjük. Emiatt ezek a specifikációs módszerek főként a különösen kritikus területeken használatosak, pl. vasúti [But02] vagy légi [HLR98] iparban.

Az ipari vezérlőrendszerek területén a napjainkban általánosan használatos, legkorszerűbb fejlesztési módszerek még mindig informális specifikációkon és rejtett feltételezéseken alapulnak. Ezek a specifikációk gyakran nem egyértelműek, ami félreértésekhez és a megvalósított rendszer elvárásoknak nem megfelelő viselkedéséhez vezethetnek. Az egyértelmű specifikáció hiánya a formális verifikáció számára is problémát jelent: hogyan dönthető el, hogy a megvalósítás helyes-e, ha nem ismert a helyesség definíciója, azaz az elvárt tulajdonságok?

Különböző próbálkozások történtek jobb, PLC-specifikus specifikációs módszerek készítésére, pl. [Lju+10; Luk+13]. A létező módszerek vizsgálata és egy megfelelő specifikációs formalizmus keresése jelen munka kihívása (5. kihívás). További kihívás a kiválasztott specifikációs módszerhez formális verifikáció biztosítása PLC-programokra (6. kihívás).

1.3. Az új kihívások áttekintése

1. kihívás: Korlátos és szaturációalapú technikákat integráló modellellenőrző algoritmusok tervezése a teljesítményük javítása céljából. Mind a korlátos modellellenőrzés, mind a szaturációalapú technikák bővítették a verifikálható modellek körét az egyszerű explicit modellellenőrző algoritmusokhoz képest. Lehetséges ezt a két megoldást ötvözni? Javítja ez a teljesítményt az eredeti szaturációalapú modellellenőrzéshez képest?

2. kihívás: A modellellenőrzés elérhetővé tétele PLC-fejlesztők számára. Ipari vezérlőszoftverek fejlesztése során a modellellenőrzés ritkán használatos, főként mivel a formális modellek és követelmények létrehozása, valamint a modellellenőrző eszközök használatának megismerése sok erőfeszítést igényel. Hogyan tehető a modellellenőrzés elérhetővé és gyakorlatban felhasználhatóvá a PLC-programok fejlesztési folyamatában? Hogyan lehet a modellellenőrzést túlzott mértékű erőfeszítés nélkül, a

[6] Eredeti idézet: „detailing, structuring and documenting the agreed characteristics of the system-to-be” [Lam09].

[7] Eredeti idézet: „is the expression, in some formal language and at some level of abstraction, of a collection of properties some system should satisfy” [Lam00].

[8] Eredeti idézet: „[s]pecifications are never formal in the first place” [Lam00].

[9] Eredeti idézet: „hard to develop and assess” [Lam00].

[10] Eredeti idézet: „a higher degree of precision in the formulation [...], precise rules for their interpretation and much more sophisticated forms of validation and verification” [Lam09].

felhasználókat (PLC-fejlesztőket) komplex matematikai formalizmusoktól megkímélve használni?

3. **kihívás: A PLC-modellellenőrzés használhatóvá tétele valós PLC-programokra.** A valódi PLC-modulok vagy programok formális modelljeinek állapottere tipikusan óriási, ami gyakran lehetetlenné teszi az általános célú modellellenőrzők használatát a túlzott erőforrásigény miatt. Segíthetnek-e heurisztikus modellredukciók az erőforrásigény csökkentésében és ezáltal abban, hogy nagyobb modellek ellenőrzése is lehetővé váljon?
4. **kihívás: A modellellenőrzési folyamat kiterjesztése biztonságkritikus PLC-programokra.** Az eredeti PLC-modellellenőrzési folyamat csak a Siemens SCL nyelvet célozta meg, amely magas szintű programnyelvként megfelelő összetett programok implementációjára. Ugyanakkor a biztonságkritikus PLC-k fejlesztése során speciális eljárásokat és korlátozásokat kell követni, mint például az FBD vagy LAD nyelvek használata (Siemens PLC-k esetén). Hogyan adaptálható a felvázolt verifikációs folyamat ezekre a biztonságkritikus PLC-programok fejlesztésben használt, alacsony szintű programnyelvekre?
5. **kihívás: Könnyűsúlyú formális specifikáció biztosítása PLC-program-modulokhoz.** Egyértelmű követelményekre bármilyen fejlesztési vagy verifikációs tevékenységhez szükség van. A formális specifikációk használata segíthet az egyértelműségben, de az általános célú formális specifikációs módszerek túlzottan összetettek vagy nem elég intuitívak, így a PLC-programok fejlesztésében nem lennének használhatók túlzott erőfeszítés nélkül. Milyen követelményeknek kell egy PLC-specifikus formális specifikációs nyelvnek megfelelnie? Milyen formális specifikációs módszer tudná segíteni a PLC-k szoftverfejlesztési folyamatát?
6. **kihívás: Verifikációs megoldások nyújtása a formális specifikációra építve.** Hogyan segítheti a formális specifikáció a PLC-programok verifikációját? Milyen verifikációs módszerek használhatók a PLC-programok és formális specifikációik közti konformancia vizsgálatára? Hogyan tehetők ezek a módszerek hasznossá a gyakorlatban túlzott mennyiségű hamis pozitív nélkül (túlzott mennyiségű olyan eltérés jelzése nélkül, amelyeket a fejlesztők elfogadható különbségnek tartanak)?

Ezek a kihívások egy kutatási projekthez vezettek, amely három különböző területen nyújt új tudományos eredményeket: kifejlesztésre került egy új, szaturációalapú korlátos modellellenőrző algoritmus különféle iterációs stratégiákkal (1. tézis); egy új módszer a PLC-programok modellellenőrzésére (2. tézis); és definiálásra került egy formális specifikációs nyelv PLC-programok moduljaihoz, a hozzá kapcsolódó verifikációs módszerekkel együtt (3. tézis). Az 1. táblázat bemutatja a tárgyalt kihívások és a leírt munka eredményei közti összerendeléseket.

1. táblázat. A tárgyalt kihívások és a leírt munka eredményei közti összerendelések

		1.	**2.**	**3.**	**4.**	**5.**	**6.**
		kihívás					
1. tézis	*Az értekezés 2. fejezete*	•					
2. tézis	*Az értekezés 3. fejezete*		•	•	•		
3. tézis	*Az értekezés 4. fejezete*					•	•

2. Kutatási módszerek és új eredmények

Kutatási módszerek. J.N. Amaral [Ama] öt kategóriába sorolta az informatikatudományi kutatásokat: *formális*, *kísérleti*, *építő*, *folyamatot leíró* és *modellt alkotó* módszerek. Ez a doktori értekezés a kísérleti és építő módszereket követi. Az építő módszerben a kutató egy új dolgot állít elő azért, hogy „demonstrálja, hogy lehetséges”[11] [Ama]. A kísérleti módszer során a kutató először azonosítja „a kiértékelendő rendszerrel kapcsolatban felteendő kérdéseket”[12] [Ama], majd a második fázisban megválaszolja ezeket kísérletekkel, mérésekkel.

Jelen munkában az új algoritmusok tervezése és az új formalizmusok, nyelvek definíciója a matematikai logikán, gráfelméleten és az automataelméleten alapul. A doktori értekezésben bemutatott kutatás kibővíti és épít a korábbi módszerekre és eredményekre. A rendszerek modellezése ebben a munkában Petri-hálókra, véges állapotgépekre és időzített automatákra támaszkodik. A tulajdonságok specifikációjára részben a CTL és LTL temporális logikák kerülnek felhasználásra. A verifikációs algoritmusok felhasználják a modellellenőrzést, viselkedési ekvivalenciarelációkat és gráfalgoritmusokat.

Az eredményeket különböző ipari példákon kerültek kiértékelésre. A bemutatott kutatás jelentős része a CERN (European Organization for Nuclear Research, Európai Nukleáris Kutatási Szervezet) Beams és Engineering Department ipari vezérlőrendszerekkel foglalkozó csoportjaiban készült. Ez lehetővé tette számomra a gyakori interakciót az ipari és folyamatirányítási mérnökökkel. Ez hasznos tudást biztosított a szakterület specialitásairól és szükségleteiről, illetve később értékes visszajelzéseket az eredményekkel kapcsolatban.

2.1. Szaturációalapú korlátos modellellenőrzés

A szimbolikus modellellenőrzési módszerek egy sikeres alosztálya az ún. szaturációalapú technikák családja [CZJ12; ZC09]. A szaturációalapú technikák már bizonyították hatékonyságukat különféle felhasználási területeken. Egy fontos példát mutat be a [*c*28], amely elsőként biztosított teljes modellellenőrzést az ún. *PRISE biztonsági logikára*, amely a Paksi Atomerőmű egyik ipari vezérlő alrendszere.

A szaturációalapú CTL-modellellenőrzés egy két lépésből álló algoritmus: először felderíti az elérhető állapotteret, majd megvizsgálja a megadott követelményt. Ugyanakkor gyakran a követelmény kiértékelhető lenne (pl. egy a követelményt sértő állapot található lenne) az állapottér egy részének felderítésével is (csak azon állapotok felderítésével, amelyek legfeljebb b távolságra vannak a kezdőállapotból, azaz legfeljebb b lépésben elérhetők). Ez a korlátos modellellenőrzési algoritmusok alapvető felvetése. A mi ötletünk az, hogy a korlátos modellellenőrzés ötvözése a szaturációalapú technikákkal javíthatja a szaturációalapú modellellenőrzés teljesítményét bizonyos esetekben.

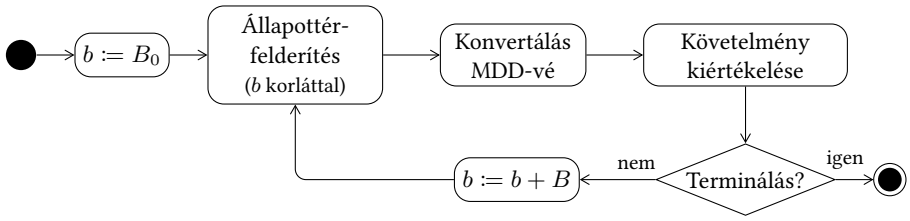
Egy explicit korlátos modellellenőrző algoritmust viszonylag könnyű készíteni, például az állapotgráf mélységi bejárásának korlátozásával. Ezzel szemben a szaturáció kódolt állapothalmazokon működik és egy speciális bejárási sorrendet használ a teljesítmény javítása érdekében, amely kihasználja a dekomponált modellek lokalitását. Ezek miatt nehéz az állapottérben megtett lépések (tranzíciók) számolása, valamint az egyes állapothalmazok különböző távolságú állapotokat is tartalmazhatnak.

Yu *et al.* [YCL09] javasolt egy szaturációalapú korlátos állapottér-felderítési algoritmust, de ennek modellellenőrzésre való felhasználását nem vizsgálták. Ez a módszer közvetlenül nem használható modellellenőrzésre két fő oknál fogva: a CTL-kifejezések kiértékelését nem célozták (csak elérhetőségi problémák vizsgálatát biztosították, nem temporális logikai kifejezések kiértékelését), és a módszer nem iteratív (azaz nem vették figyelembe, hogy az állapottér-felderítés egyre növekvő korlátokkal folytatandó, amíg a követelmény ki nem értékelhető).

[11] Eredeti idézet: „to demonstrate that it is possible” [Ama].

[12] Eredeti idézet: „the questions to be asked about the system under evaluation” [Ama].

Vörös András és Bartha Tamás felügyelete alatt kidolgoztam egy folyamatot, amely integrálja a szaturációalapú korlátos állapottér-felderítést és a CTL-kifejezések szaturációalapú kiértékelését. A nemkorlátos szaturációalapú modellellenőrzés egyszerű, kétlépéses folyamatával szemben ez iteratív, ahogy az 1. ábrán látható.



1. ábra. Az iteratív szaturációalapú korlátos modellellenőrzés (B-I-Sat) áttekintése

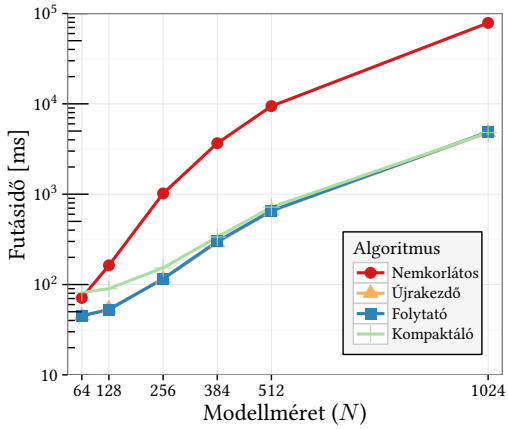
Az algoritmus újszerű módon integrálja a már létező építőelemeket:

- A *korlátos állapottér-felderítés* a Yu *et al.* által [YCL09] javasolt algoritmuson alapul (B_0 jelöli a kezdő korlátot, B pedig a korlát növekményét).
- A *temporális logikai kifejezés kiértékelése* hasonló algoritmust használ, mint [*c*28; *j*26], amelyek a [CS03; ZC09] munkákon alapulnak.
- Az élcímkézett döntési diagramok (edge-valued decision diagram, EDD) és a többértékű döntési diagramok (multivalued decision diagram, MDD) közti konverzió a diagramok definícióin alapul. Megjegyzendő, hogy EDD-ket a korlátos állapottér távolságinformációinak tárolása céljából használunk, de ezt a kiegészítő információt a követelmény kiértékelése során nem használjuk fel.

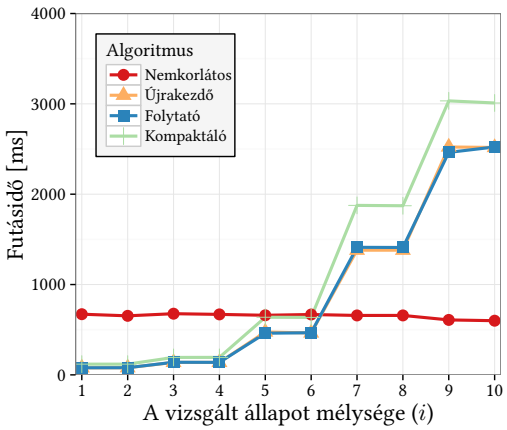
Ez az algoritmus és az implementációja megmutatta, hogy az alapötlet megvalósítható. Ugyanakkor bizonyos kihívások új kontribúciókat tettek szükségessé a verifikáció hatékonnyá és helyessé tétele érdekében.

- **Iterációk.** Yu *et al.* algoritmusát [YCL09] iteratívvá kellett tenni, hogy így a felderítés folytatható legyen addig, amíg a követelmény kiértékelése meg nem tehető. Különböző iterációs stratégiák lehetségesek. A bemutatott munka során három különböző stratégiát javasoltam: az újrakezdő, folytató és kompaktáló stratégiákat. Ezek főként az egyes iterációk közt megőrzött információk típusában és mennyiségében, illetve az iterációk kezdő állapothalmazaiban különböznek.
- **Terminálás.** A *terminálás* az algoritmus kritikus pontja. Annak érdekében, hogy a módszer teljes és helyes legyen, a korlátos modellellenőrzés csak akkor állítható le, ha egy mérvadó eredmény adható a felderített (részleges) állapottér alapján, vagy ha a teljes állapottér felderítésre került. E munka keretében terminálási feltételeket adtam a szaturációalapú korlátos CTL-modellellenőrzéshez. Ez háromértékű logikán alapul, amely képes kifejezni a részleges állapottér által okozott bizonytalanságot a kifejezés teljesülésében.
- **Távolságinformáció tárolása.** A korlátos állapottér-felderítés egyik problémáját az állapotok *távolságinformációjának* tárolása okozza. Ennek következményeként a szimbolikus állapothalmazok kisebb részekre töredeznek, amely megnövekedett számítási és tárolási igényt jelent. Munkám során fejlesztettem egy inkrementális bejárási stratégiát (az ún. kompaktáló stratégiát), amely csökkentheti a memóriaigényt (és ezáltal a futásidőt is) azáltal, hogy csak azokhoz az állapotokhoz tárolja el a távolságinformációt, amelyekhez ez szükséges.

Ezen megfontolások alapján született meg a B-I-Sat (korlátos iteratív szaturáció, Bounded Iterative Saturation) algoritmus, egy szaturációalapú korlátos CTL-modellellenőrző algoritmus. A B-I-Sat algoritmus és a különböző stratégiák többféle benchmark modellen voltak kiértékelve. Az algoritmus felhasználásra került egy ipari vezérlőrendszer vizsgálatában is, amelyeket korábban a nemkorlátos szaturációalapú algoritmusokkal sikerült elsőként verifikálni.



(a) Az EF ($bit_{12} = 1$) kifejezés kiértékelésének futásideje a Counter–N modellen



(b) Az EF ($q_i = 0$) kifejezés kiértékelésének futásideje a Queen–10 modellen

2. ábra. A nemkorlátos szaturáció és a B-I-Sat algoritmus skálázódásának összehasonlítása

1. tézis Kidolgoztam a B-I-Sat (korlátos iteratív szaturáció, Bounded Iterative Saturation) algoritmust, ami egy új, CTL-modellellenőrző algoritmus, amely hatékonyan ötvözi a korlátos modellellenőrzést a szaturáció-alapú technikákkal.

- 1.1 Definiáltam az algoritmus építőelemeit, majd ezek alapján a teljes B-I-Sat algoritmust korlátos szaturációalapú CTL-modellellenőrzés biztosítására. Két stratégiát definiáltam a B-I-Sat algoritmushoz: az újrakezdő és a folytató stratégiákat.
- 1.2 A háromértékű logikát felhasználva terminálási feltételeket definiáltam a B-I-Sat algoritmushoz.
- 1.3 Kifejlesztettem egy inkrementális keresési stratégiát, az ún. kompaktáló stratégiát a B-I-Sat algoritmus memóriaigényének csökkentése érdekében.
- 1.4 Kiértékeltem a B-I-Sat algoritmus és a különféle stratégiák teljesítményét változatos benchmark modelleken és egy ipari példán.

Az eredmények jelentőségét az adja, hogy a B-I-Sat algoritmus bizonyos esetekben gyorsabb verifikációt, vagy nagyobb modellek vizsgálatát teszi lehetővé a nemkorlátos szaturáció-alapú CTL-modellellenőrzőhöz képest. A 2(a). ábrán látható, hogy a különféle B-I-Sat stratégiák jobban skálázódnak a növekvő modellmérettel, mint a nemkorlátos algoritmus. A 2(b). ábra demonstrálja, hogy a B-I-Sat stratégiák jelentősen jobban teljesítenek, amikor „sekély” (kis i-értékű) követelmény kerül kiértékelésre, míg a nemkorlátos modellellenőrző hatékonyabb, amennyiben a követelmény „mély”, azaz az állapottér nagy részét fel kell deríteni a kiértékeléshez.

Vörös András és Bartha Tamás a szakdolgozatom és diplomatervem témavezetőiként vettek részt ebben a kutatásban. A munka egyes részei bemutatásra kerültek a szakdolgozatomban és diplomatervemben [*a*31; *a*30].

Az 1. tézis eredményei az értekezés 2. fejezetében kerülnek tárgyalásra. A tézishez az alábbi publikációk kötődnek: [*j*2; *j*4; *c*10; *c*17; *c*18; *e*20; *e*21].

2.2. Kritikus PLC-programok modellellenőrzése

A modellellenőrzés ipari fejlesztési folyamatokba történő integrálása és használatának széles körű elterjedése (nem csak különlegesen kritikus esetekben történő használata) nem várható, amennyiben a felhasználóktól elvárt a követelmények vagy az implementáció kézi formalizálása. A formális követelmények és modellek elrejtéséhez olyan módszerek szükségesek, amelyek ezeket automatikusan generálják olyan bemenetek alapján, amelyek a megcélzott felhasználói

kör által könnyen érthetőek és biztosíthatóak. A formális modellek elrejtése azt is jelenti, hogy a gyakran szükséges kézi modelloptimalizációt is automatikussá kell tenni.

Az itt bemutatott munka keretében egy általános modellellenőrzés-alapú verifikációs folyamatot (ld. a 3. ábrán) dolgoztunk ki, amely specifikusan a PLC-programok ellenőrzését célozza meg. A PLC-programok különböző nyelveken íródnak: az IEC 61131 szabvány [I61131-3] öt programozási nyelvet definiál. Ezek közül kettő, a magas szintű ST és az alacsony szintű IL szöveges nyelv, míg az FBD, LD és SFC grafikus nyelvek. Az egyes gyártók kisebb vagy nagyobb eltérésekkel valósítják meg ezeket a szabványos nyelveket. Ebben a munkában a Siemens által készített megvalósítást vettem alapul (ezek nevei rendre: SCL, STL, FBD, LAD és GRAPH/SFC). A nyelvek teljes palettája nem feltétlen áll a fejlesztők rendelkezésére, például a biztonságkritikus Siemens PLC-k szoftverfejlesztése esetén a használható nyelvek köre és azok elemei korlátozottak.

A PLC-programok modellellenőrzésének különböző részeivel már több szerző is foglalkozott. Legtöbben az alacsony szintű nyelveket célozták meg (pl. az IL nyelvet [Can+00; LNN13]), míg a magas szintű ST nyelv ritkán támogatott [GSF08; BBK12]. A legtöbb munka az IEC 61131 szabvány szerinti PLC-programokat feltételezi, holott a vezető PLC-hardver gyártók jelentősen eltérnek ezektől. Számos formális verifikációs munka modellellenőrzőket használ alacsonyabb szintű verifikációs motorként (pl. CaSMV, NuSMV, UPPAAL [Can+00; GSF08; SF11]), míg mások például közvetlenül SAT-megoldókra építenek (pl. [LNN13]).

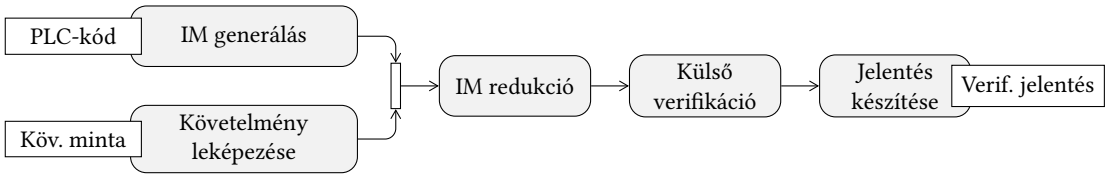
A fő problémák, a skálázhatóság és a való életbeli felhasználhatóság ritkán kapnak különleges figyelmet ezekben a munkákban. Mindössze néhány szerző számolt be eszközök fejlesztéséről, amelyek az ötleteiket és algoritmusaikat megvalósítják. Ezek közül is csupán egy, az Arcade.PLC eszköz érhető el [BBK12]. Sajnos az Arcade.PLC nem alkalmazható a figyelembe vett alkalmazási területen, részben mivel a felhasználók még az eszközt használva is túlzottan ki vannak téve a formális részleteknek, például a követelmények formalizálásakor.

A mi verifikációs folyamatunk (ld. 3. ábra) dedikáltan a skálázódási és felhasználhatósági kihívásokat célozza. A verifikációs folyamat főbb lépései és tulajdonságai az alábbiakban kerülnek kiemelésre.

- **Általános belső reprezentáció.** A verifikációs folyamat egy köztes modell (intermediate model, IM) köré épül. Ez egy általános, egyszerű leképezése a PLC-program viselkedését leíró formális modellnek. Ez a reprezentáció független a használt verifikációs motortól. A köztes modell egy kiterjesztett automata formalizmusra épít, amely megfelelő programok viselkedésének leírására és számos modellellenőrző bemeneti nyelvéhez közel áll.
- **A verifikációs folyamat bemenetei.** A verifikációs folyamat két bemenete a PLC-program (Siemens SCL nyelven a nem biztonságkritikus és STL nyelven a biztonságkritikus esetekben) és a követelmény egy kitöltött követelményminta formájában, hasonlóan [DAC99; CMS08]-hoz. Minden egyes követelményminta egy precízen megfogalmazott angol mondat, amelynek bizonyos kihagyott részeit a felhasználónak kell kitöltenie a bemeneti és kimeneti változókból (jelekből) felépített logikai kifejezésekkel. Ez alapján a temporális logikai kifejezés automatikusan generálásra kerül.
- **Modellredukciók.** A modellellenőrzési folyamat erőforrásigényének csökkentése (és így a használhatóságának növelése) érdekében automatikus tulajdonságmegőrző modellredukciókat alkalmazunk a köztes modellen. Ezek közt találhatók például strukturális redukciók, amik hasonlók a fordítók által végzett optimalizációkhoz. Más redukciós szabályok a követelmény figyelembevételével eltávolítják a modell azon részeit, amelyek nem befolyásolják a megadott követelmény teljesülését.
- **Külső verifikációs eszközök.** Ebben a projektben nem került sor új modellellenőrző fejlesztésére annak érdekében, hogy a fejlesztési szükségleteket megfelelő szinten tartsuk és profitálni tudjunk a számos helyen zajló korszerű modellellenőrző-fejlesztésekből. Ehelyett széles körben ismert, általános célú modellellenőrzőket „csomagoltunk be” és használunk alacsonyabb szintű verifikációs motorként. A csomagolás megvalósítása felel a köztes és

konkrét (kimeneti és bemeneti) reprezentáció közti transzformációért.

- **Jelentés.** Az egyes modellellenőrzők különböző, modellellenőrző-függő, gyakran komplex kimeneteket biztosítanak. Ezeket gyakran nehéz értelmezni. Emiatt a modellellenőrzők kimeneteit is átfordítjuk egy köztes reprezentációra, majd ezen alapulva egy egyszerűsített verifikációs jelentést generálunk, amely érthetőbb a PLC-fejlesztők számára.



3. ábra. A PLC-programok verifikációs folyamatának egyszerűsített áttekintése

2. tézis Hozzájárultam egy általános, rugalmas, PLC-programok modellellenőrzését végző verifikációs folyamat kidolgozásához, amely nem igényel mély ismereteket a felhasználóktól a formális módszerek területén. Ennek a folyamatnak az alábbi fontos részeit dolgoztam ki:

2.1 Megterveztem egy köztes nyelvet (intermediate model, IM), amely PLC-programokat képes reprezentálni verifikációs célokból és köztes reprezentációként tud szolgálni többféle modellellenőrző felé. Az IM-alapú modellellenőrzés teljesen automatizált, így a formális verifikációhoz nem értő fejlesztők által is használható.

2.2 Modellredukciós heurisztikákat dolgoztam ki a verifikációs folyamathoz, amelyek a köztes modell méretét hatékonyan tudják csökkenteni, így téve a modellellenőrzési folyamatot kevésbé erőforrás-igényessé.

2.3 Kibővítettem a verifikációs folyamatot (amely eredetileg csak az SCL programozási nyelvet célozta) a biztonságkritikus PLC-programok ellenőrzéséhez szükséges programnyelvek (FBD és LAD) támogatásával az STL nyelven keresztül.

2.4 Implementáltam a teljes definiált verifikációs folyamatot a *PLCverif* eszközben, amely így elérhetővé teszi a PLC-programok forráskódján és követelménymintákon alapuló modellellenőrzést az ahhoz nem értő fejlesztők számára is anélkül, hogy mélyreható képzésre lenne ehhez szükség. Megvizsgáltam a verifikációs folyamat valós körülmények közti használhatóságát különböző, a CERN-ben fejlesztett és használt PLC-modulok és alkalmazások segítségével.

Az eredmények jelentőségét az adja, hogy a PLCverifben implementált formális verifikációs folyamat lehetővé teszi a formális verifikációs tudással nem rendelkező fejlesztők számára is PLC-programok modellellenőrzését. Így javítható a fejlesztett PLC-programok minősége. A PLCverif eszköz[13] használatban van a CERN-ben, és használata bebizonyította, hogy a modellellenőrzés képes olyan hibákat felfedni a PLC-programok specifikációjában és implementációjában, amelyeket a tesztelés nem [*j*3; *c*9; *c*16]. A javasolt verifikációs folyamat más projektek mellett egy biztonságkritikus rendszer fejlesztésében is felhasználásra került. Ez a rendszer felel a szupravezető mágnesek vizsgálatának biztonságáért [*c*9]. Modellellenőrzés segítségével 12 különböző hibát sikerült felfedni az implementációban, közülük többet gyakorlatilag lehetetlen lett volna megtalálni az általánosan alkalmazott tesztelési módszerekkel.

A 2. tézisben leírt munka egyes részei közösek voltak Borja Fernández Adiegoval (University of Oviedo és CERN). Az ő fő kontribúciói a verifikációs folyamat magas szintű koncepciója (ld. korábbi cikkeiben [Fer+13; FBM13]), a PLC-k viselkedésének vizsgálata, és a változóabsztrakció (variable abstraction), egy a javasolt IM-redukciós módszerek közül [*j*3]. Az eredményeit egy doktori értekezésben foglalta össze [Fer14].

[13] http://cern.ch/plcverif/

Az én hozzájárulásom volt a köztes modell (IM) nyelv kidolgozása és megvalósítása, a modellredukció kidolgozása (kivéve a változóabsztrakciót), továbbá a verifikációs folyamat kiterjesztése és adaptálása a biztonságkritikus PLC-programok ellenőrzéséhez. Az SCL nyelven írt programok IM-re történő leképezése közös munka volt. A kutatás során Enrique Blanco Viñuela és Jean-Charles Tournier a CERN részéről konzulensként vettek részt. Víctor M. González Suárez, Jan Olaf Blech, Simon Bliudze, Vörös András és Bartha Tamás szintén segítették a munkát megjegyzéseikkel és visszajelzéseikkel.

A 2. tézis eredményei az értekezés 3. fejezetében kerülnek tárgyalásra. A tézishez az alábbi publikációk kötődnek: [*j*1; *j*3; *c*8; *c*9; *c*11; *c*13; *c*14; *c*15; *c*16; *r*23; *r*24].

2.3. PLC-modulok formális specifikációja

A PLCverif eszköz és a mögötte álló verifikációs folyamat elérhetővé tette a formális verifikációt a PLC-fejlesztőknek anélkül, hogy különleges ismeretekre vagy a fejlesztési folyamat módosítására lenne szükség. Ugyanakkor azt is felismertük, hogy gyakran nehéz a követelmények megfogalmazása, formalizálása a jelenleg elérhető informális specifikációk és dokumentációk alapján.

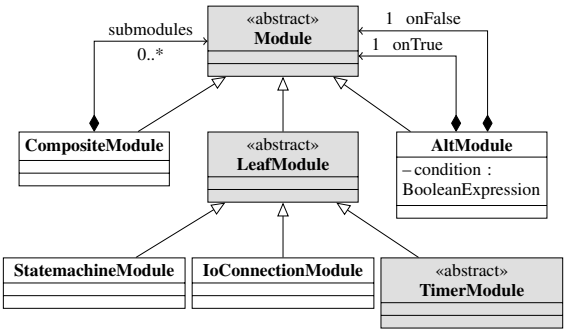
A PLCverifben használt mintaalapú követelményspecifikáció csak egy korlátozott megoldást nyújt a formalizálás nehézségére, mivel egy teljes viselkedést lefedő, mintaalapú követelményhalmaz biztosítása nehéz feladat. Ez különösen fontos probléma két esetben: a PLC-programok alapvető építőelemeinek (újrahasználható modulok) fejlesztése során, ahol egy hiba vagy egy nem egyértelmű viselkedés nagy számú alkalmazást érinthet; illetve biztonságkritikus rendszerek esetén. Ezen esetekben a deklaratív követelményminták egy halmaza (amely a tulajdonságok egy részére koncentrál) nem elégséges, a teljes viselkedés leírására van szükség.

A megfelelő, precíz specifikációs nyelvek biztosítása ipari vezérlőrendszerekhez egy régóta zajló munka. Ezen a területen az egyik legismertebb specifikációs nyelv a Grafcet [I60848], amely fejlesztésének gyökerei az 1970-es évek közepéig nyúlnak vissza [Bla77]. Ez a nyelv egy biztos Petri-hálókon alapuló, precíz specifikációs nyelvnek készült, ugyanakkor nem alkalmazható általánosan minden típusú PLC-programra, továbbá az összetett logikák leírása gyakran félreérthető és érzékeny a hibákra. Későbbi munkák egyszerűbb megoldásokat javasoltak, mint például a ProcGraph [Luk+13], amely véges állapotgépekre épül. E nyelv esetében viszont gyakran nagy mennyiségű forráskódot kell írni a specifikáció részeként. Más megoldások a formális nyelvek felől közelítik meg a problémát, pl. az ST-LTL [Lju+10] az LTL temporális logika adaptációja PLC-programokra. Ez a módszer formális és egyértelmű, de teljes rendszerek komplex viselkedésének leírása helyett inkább bizonyos invariáns tulajdonságok kifejezésére alkalmas. Léteznek további formális és nemformális, PLC-specifikus specifikációs módszerek is, de egyik sem bizonyult kellőképp formálisnak, egyértelműnek és érthetőnek ahhoz, hogy a CERN fejlesztési folyamatába integrálható legyen.

Ez a felismerés vezetett el a PLCspecif tervezéséhez és fejlesztéséhez, amely egy szakterület-specifikus formális specifikációs nyelv PLC-modulok teljes viselkedésének leírására. Az elérhető módszerek és a CERN-beli kódbázis áttekintése, valamint a fejlesztőkkel folytatott megbeszélések alapján összegyűjtöttem általános és szakterület-specifikus követelményeket a javasolandó specifikációs nyelvvel kapcsolatban. Általános követelmények a PLCspecif felé, hogy legyen formális, precíz, „könnyűsúlyú” (könnyen tanulható). Emellett segítse az érthetőséget, a verifikációt és a dokumentációt. Szakterület-specifikus követelmények például az események és alacsony szintű jelek támogatása, a vezérlő logika leírásának tisztán tartása a bemenetek és kimenetek kezelésének leválasztásával, többféle formalizmus támogatása a leírandó viselkedés és a specifikáció közti szemantikus rés minimalizálása érdekében és a PLC-kben használt időzítőkhöz adaptált időkezelés [*e*19].

A követelmények alapján a PLCspecif nyelvet egy hierarchikus, moduláris nyelvnek terveztem meg. Minden PLCspecif modul egy kompozit modul (amely viselkedését almodulok finomítják), alternatív modul (amely viselkedése bizonyos feltételtől függ) vagy levél modul (amely a

rendszer egy részének viselkedését írja le). A modulstruktúra magas szintű áttekintése a 4. ábrán látható. A levél modulok három fő részre vannak osztva: bemeneti jelek kezelése, vezérlő logika leírása és kimeneti jelek kezelése. Mindhárom résznél többféle formalizmus támogatott, így a fejlesztő kiválaszthatja az adott esetben legmegfelelőbbet. A bemeneti és kimeneti jelek kezelésénél egyszerű logikai vagy aritmetikai kifejezések mellett két táblázatos formalizmus is használható. A vezérlő logika leírására egy speciális állapotgép-formalizmus, egy adatfolyam-leírási módszer és PLC-időzítő modulok használhatók. Ezek a formalizmusok a PLC-fejlesztők által használt informális és félformális módszerek alapján kerültek kiválasztásra. Minden modul kiegészíthető invariáns tulajdonságokkal, így segítve azon követelmények leírását, amelyeket nehéz a főként imperatív logikai leíró nyelvekkel kifejezni.



4. ábra. A PLCspecif modulstruktúra metamodellje [*e*19]

A PLCspecif absztrakt és konkrét szintaxisa precízen definiálásra került [*r*22], csakúgy mint a formális szemantikája, amely egy kiterjesztett automata formalizmuson alapul.

Az, hogy a PLCspecif egy precíz, formális szemantikával rendelkezik, biztosítja, hogy a leírt viselkedés egyértelmű legyen, továbbá lehetővé teszi a specifikációk felhasználását verifikációs és kódgenerálási célokra is. Munkám során az alábbi kapcsolódó módszereket dolgoztam ki:

- **Statikus analízis.** Definiáltam egy szabálykészletet a specifikáció jólformáltságának ellenőrzésére. Ezek többsége gráfalgoritmusokkal került megvalósításra, míg a többihez SAT-problémákra történő leképezést biztosítottam, így ezek pl. a Microsoft Z3 SAT/SMT-megoldó eszközzel [MB08] vizsgálhatók.

- **Invariáns tulajdonságok ellenőrzése.** Ha egy PLC-fejlesztő nem bizonyos abban, hogy az imperatív viselkedésleírás kielégít bizonyos kívánt invariáns tulajdonságokat, lehetősége van ezen invariáns tulajdonságok explicit leírására a specifikációban. A korábbiakban tárgyalt mintaalapú követelményspecifikációk ezen invariáns tulajdonságok leírására is használhatók. A PLCspecif szemantikadefiníciója alapján megadtam egy leképezést a PLCverifben használt köztes modellre, amely lehetővé teszi az invariáns tulajdonságok modellellenőrzését a specifikáción [*c*5].

- **Kódgenerálás.** Mivel a PLCspecif specifikáció a PLC-program teljes viselkedését leírja, lehetőség nyílik az implementációjának automatikus előállítására. A PLCspecif formális szemantikáját úgy definiáltam, hogy az közel álljon programok vezérlésfolyam-gráfjához (control flow graph, CFG). Ez alapján megterveztem és kifejlesztettem egy kódgenerálási módszert, amely egy Siemens SCL nyelvű implementációt állít elő a megadott PLCspecif specifikációhoz. Ugyanakkor nem elégséges, hogy a kódgenerálás eredményeként előálló programkód viselkedése megfelel a specifikált viselkedésnek. Karbantarthatósági okokból az is szükséges, hogy a generált kód olvasható és érthető legyen. Ennek érdekében a generált kód követi a specifikáció struktúráját, valamit a kódgenerátor konfigurálható (pl. a generált kód struktúrája, az állapotgépek reprezentációja vagy az enumerációtípusok le-

képezése beállítható), így a generált kód beilleszthető a kódbázisba.

- **Konformanciaellenőrzés.** Számos valós felhasználási esetben az implementáció (PLC-program) és a specifikáció is adott, de a kettő közti ekvivalencia nem ismert. Ilyen eset lehet például az, ha korábbi implementációhoz később készül a formális specifikáció, vagy ha a generált programkód kézzel kerül módosításra, majd a formális specifikációba később kerül a módosítás átvezetésre. A viselkedés ekvivalenciaellenőrzése megmutathatja az implementáció és a specifikáció egyezőségét vagy a köztük lévő különbségeket. Kidolgoztam egy módszert a PLCspecif specifikációk viselkedési ekvivalenciájának ellenőrzésére. Ehhez a specifikációt a PLCverif eszközben használt IM formalizmusra képezem le – hasonlóan a PLC-programhoz –, majd a két összehasonlítandó modellből egy kompozit modellt képezek. Ezen a kompozit modellen a PLCverif egy modellellenőrzőt használva bizonyíthatja vagy cáfolhatja a modellek viselkedési ekvivalenciáját.

  A gyakorlatban a PLC-k gyakran lassú folyamatokat irányítanak, ahol a kimenet rövid késlekedése nem okoz megfigyelhető különbséget az irányított folyamatban. A szigorú viselkedési ekvivalencia megkövetelése a specifikáció és az implementáció közt ezen esetekben hamis pozitív eredményekhez vezethet, azaz olyan különbségeket mutathat, amelyek a felhasználó szempontjából elfogadhatók. A hamis pozitív eredmények nagy száma ellehetetleníti a gyakorlatban az ekvivalenciaellenőrzést. Emiatt azonosítottam azokat a főbb eseteket, amikor a szigorú ekvivalencia nem követelmény bizonyos kimenetpárok közt és ezek alapján *megengedő konformanciarelációkat* definiáltam. Így a fejlesztők minden egyes kimenetpárra megadhatják a konformancia elvárt szintjét. A konformanciarelációk formálisan definiálásra kerültek, valamint definiáltam a megsértésüket leíró temporális logikai kifejezéseket is, ennek köszönhetően a konformanciarelációk teljesülése modellellenőrző segítségével vizsgálható [*c*6].

3. tézis Kidolgoztam a *PLCspecif* formális viselkedésspecifikációs nyelvet, amely a PLC-programok fejlesztésének igényeihez van adaptálva. A nyelv kódgenerálási és verifikációs felhasználást céloz meg.

3.1 Megterveztem a főbb nyelvi elveket, majd definiáltam a nyelv precíz szintaxisát és szemantikáját. A nyelv tervezése azokon a követelményeken alapult, amelyeket a formális specifikációs módszerek irodalmának tanulmányozásával és a CERN PLC-fejlesztői közössége által nyújtott visszajelzések elemzésével sikerült összegyűjteni.

3.2 Kifejlesztettem egy leképezést a PLCspecif specifikációról a PLCverif által használt köztes modell (IM) nyelvre. Ez lehetővé teszi különféle modellellenőrzők használatát az invariáns tulajdonságok ellenőrzésére.

3.3 Megterveztem és megvalósítottam a PLCspecif specifikációs nyelvhez egy automatikus kódgenerálási módszert, amely a nyelv formális szemantikáján alapul. Ez a rugalmas és konfigurálható kódgenerálási módszer Siemens SCL programkódot állít elő a formális specifikáció alapján.

3.4 Új konformanciarelációkat terveztem meg, amelyek a PLC szoftverfejlesztésben használhatók és lehetővé teszik a fejlesztők számára az elfogadható eltérések definícióját a specifikált és implementált viselkedések közt (pl. a kimeneti jelek rövid késleltetését). Definiáltam egy modellellenőrzésen alapuló konformanciaellenőrzési módszert, amely képes megállapítani az egyes konformanciarelációk teljesülését megadott implementáció–specifikáció párokra. Így lehetővé válik a korábban elkészített implementációk vagy kézzel módosított generált kódok összevetése egy formális specifikációval. E relációk ellenőrzését implementáltam és megvizsgáltam a használhatóságukat.

Az eredmények jelentőségét az adja, hogy a javasolt specifikációs nyelv lehetővé teszi PLC-

modulok viselkedésének precíz, teljes, formális definícióját anélkül, hogy a felhasználótól alacsony szintű formális módszerekkel vagy matematikai logikával kapcsolatos tudást igényelne. A precíz és teljes specifikáció lehetővé teszi továbbá ekvivalencia- és konformanciaellenőrzés alkalmazását, amely egy kiegészítő verifikációs módszert nyújt a mintaalapú modellellenőrzés mellett. A [*c*9] munkában bemutatott esettanulmány megmutatta, hogy a konformanciaellenőrzés valódi, biztonságkritikus, CERN-ben használt PLC-programokban képes olyan problémákat azonosítani, amelyeket a mintaalapú követelményleírások modellellenőrzésével nem sikerült megtalálni. A megengedő konformanciarelációk bevezetése a gyakorlatban is használhatóvá teszi ezt a módszert olyan esetekben is, ahol a szigorú egyezés nem követelmény. Továbbá a PLCspecif specifikációs nyelv csökkentheti a kézi implementáció mennyiségét azáltal, hogy a megcélzott szakterülethez adaptált kódgenerálási megoldást biztosít.

Ebben a munkában Majzik István doktori témavezetőként, Enrique Blanco Viñuela külső konzulensként vett részt a CERN részéről.

A 3. tézis eredményei az értekezés 4. fejezetében kerülnek tárgyalásra. A tézishez az alábbi publikációk kötődnek: [*c*5; *c*6; *c*7; *c*8; *c*9; *c*12; *e*19; *r*22].

2.4. A javasolt formális verifikációs módszerek áttekintése

A bemutatott tézisek a modellellenőrzés különféle felhasználását javasolták, pl. különböző modelleket és követelményeket ellenőrizve. A 2. táblázat a javasolt megközelítések főbb tulajdonságait hasonlítja össze.

1. **Követlen modellellenőrzés** (1. tézis). Az első tézisben a modellellenőrzés minden további segítség vagy adaptáció nélkül került felhasználásra. A formális modellek és követelmények kézzel készültek, ebben az esetben a Petri-háló és CTL formalizmusokat felhasználva. Ezeken a B-I-Sat modellellenőrző algoritmust (amely a PetriDotNet keretrendszerben került implementálásra [*c*10]) használtuk, amely eredménye kézzel került kiértékelésre. Ez egy megfelelő módszer lehet például akkor, ha egy nemdeterminisztikus modellre van szükség a modellezett viselkedések felülbecslésére, vagy ha a verifikáció ritkán történik és nem hatékony az adott platformhoz teljes, dedikált eszköztámogatást fejleszteni, mint például a PRISE biztonsági logika esetében.

2. **PLC-programok modellellenőrzése** (2. tézis). A PLCverif eszközben megvalósított [*c*13] verifikációs folyamat bemenetei az implementáció (PLC-kód) és a kitöltött követelménysminták. Mindkettő könnyen érthető a PLC-fejlesztők számára. Az implementáció ez után automatikusan a köztes modellre kerül leképezésre. A modellellenőrző csomagolója felel a modell és a követelmény leképezéséért a köztes nyelvről a modellellenőrző konkrét szintaxisára. A verifikációs folyamat eredménye egy verifikációs jelentés, amely olvasható és érthető a felhasználók számára. Ez a folyamat megfelelő olyan felhasználók számára, akik nem rendelkeznek mély formális verifikációs tudással. Az automatizálásnak és adaptációnak köszönhetően ez a módszer hatékonyan beilleszthető a PLC-k szoftverfejlesztési folyamatába, túlzott ráfordítás nélkül.

3. **Konformanciaellenőrzés** (3. tézis). A konformanciaellenőrzés a PLC-programon kívül egy formális specifikációt vár bemenetként. Mindkettő leképezésre kerül a köztes modellre, majd egy kompozit modell készül el, amelyen a konformanciaellenőrzés elvégezhető. A kiválasztott konformanciarelációk alapján a megfelelő temporális logikai követelmények automatikusan előállnak. Ezek teljesülését a becsomagolt modellellenőrzők vizsgálják meg, hasonlóak az előző, 2. esethez. Szintén az előző esethez hasonlóan az eredmény egy verifikációs jelentésben kerül bemutatásra. Ha rendelkezésre áll egy formális specifikáció, a konformanciaellenőrzés egy egyszerű és alapos módszer az implementált és elvárt viselkedés összevetésére.

4. **Kódgenerálás és invariánsellenőrzés** (3. tézis). Ha az implementáció automatikusan generálásra kerül egy helyes kódgenerátorral, nincsen szükség a specifikált és implementált viselkedések összevetésére. Ugyanakkor ebben az esetben modellellenőrzés használható a specifikációs fázisban az invariáns vagy biztonsági követelmények teljesülésének vizsgálatára. Ehhez a formális specifikációt leképezzük a köztes modell nyelvre, mint az előző (3.) esetben. Az invariáns tulajdonságok követelménymintákkal írhatók le. Ezek alapján egy becsomagolt modellellenőrző képes kiértékelni az invariáns tulajdonság teljesülését a specifikáción, amelyről automatikusan egy verifikációs jelentés is készül.

2. táblázat. A javasolt modellellenőrzési módszerek áttekintése

	1. Közvetlen modellellenőrzés	**2. PLC-programok modellellenőrzése**	**3. Konformancia-ellenőrzés**	**4. Invariáns-ellenőrzés**
Modell	Közvetlen modellezés	IM (implementáció) + Automatikus redukciók	Kompozit IM (impl. és formális spec.) + Automatikus redukciók	IM (formális spec.) + Automatikus redukciók
Követelmény forrása	Kézi megadás	Követelményminta	Konformanciareláció	Követelményminta
Modellellenőrzés	Közvetlen	„Rejtett”	„Rejtett”	„Rejtett”
Eredmény	Nyers	Verifikációs jelentés	Verifikációs jelentés	Verifikációs jelentés

3. Az új eredmények gyakorlati felhasználása

3.1. Szaturációalapú korlátos modellellenőrzés a gyakorlatban

A szaturációalapú korlátos modellellenőrző algoritmusok a PetriDotNet keretrendszerben[14] kerültek megvalósításra [*c*10]. Ezen algoritmusok hatékonyságát számos benchmark modellen vizsgáltuk. Emellett az algoritmus felhasználásra került a Paksi Atomerőmű PRISE biztonsági logikájának verifikációjára is [*j*2]. A biztonsági logikát korábbi munkánk során már sikeresen vizsgáltuk más szaturációalapú algoritmusokkal [*c*28; *j*26], de bizonyos esetekben a korlátos modellellenőrzés alacsonyabb futásidőt eredményezett [*j*2].

A B-I-Sat algoritmus bizonyos változatait tesztgenerálási célokra [*e*20] is felhasználtuk az R3-COP Artemis projekt keretében [R3C].

3.2. Ipari vezérlőrendszerek modellellenőrzése a PLCverif segítségével

A kutatásom keretében elkészítettem a PLCverif eszközt[15], amely implementálja a bemutatott verifikációs folyamatot [*c*13]. Ez az eszköz lehetővé tette a modellellenőrzés használatát a CERN különféle ipari vezérlőrendszer-fejlesztési projektjeiben. Ezek közt megtalálható volt az UNICOS keretrendszer építőelemeinek vizsgálata [*c*16; *j*3]; nagyobb vezérlőprogramok vizsgálata, mint például a Nagy hadronütköztető (Large Hadron Collider, LHC) egyik kriogén alrendszerének vezérlése [*j*3] és egy speciális, biztonságkritikus vezérlő biztonsági logikájának ellenőrzése, amely a SM18 kriogén tesztüzemben (SM18 Cryogenics Test Facility) használatos [*c*9].

3.3. Ipari vezérlőrendszerek formális specifikációja a PLCspecif segítségével

Munkám során elkészítettem a PLCspecif[16] formális specifikációs nyelv prototípus implementációját. Ezzel sikerült bemutatni az elkészített kódgenerálási módszer használhatóságát [*c*7].

[14] http://petridotnet.inf.mit.bme.hu/
[15] http://cern.ch/plcverif/
[16] http://cern.ch/plcspecif/

A prototípus implementáció integrációra került a PLCverif eszközzel a verifikáció (modellellenőrzés és konformanciaellenőrzés) megvalósítása érdekében. Ez a kombináció felhasználásra került például a már említett SM18 kriogén tesztüzembeli biztonsági logika ellenőrzésében [*c*9]. A PLCspecif specifikáción alapuló konformanciaellenőrzés sikeresen azonosított problémákat a biztonságkritikus implementációban, amely már korábban modellellenőrzésre került, így demonstrálva, hogy a tulajdonságorientált (mintaalapú) és a viselkedési (PLCspecif-alapú) verifikációs módszerek egymás kiegészítői.

4. Publikációs lista

Publikációk száma:	29
Angol nyelvű lektorált folyóiratcikkek száma:	6
WoS vagy Scopus által indexelt folyóiratcikkek száma:	6
Angol nyelvű publikációk a szerző legalább 50%-os hozzájárulásával:	3
Lektorált publikációk száma:	22
Független hivatkozások száma:	23

4.1. Tézisekhez kapcsolt publikációk

	Folyóirat-cikkek	Nemzetközi konferenciacikkek	Helyi cikkek	Kutatási jelentések
1. tézis	[*j*2],[*j*4]	[*c*10],[*c*17],[*c*18]	[*e*20],[*e*21]	—
2. tézis	[*j*1],[*j*3]	[*c*8]*,[*c*9]*,[*c*11],[*c*13],[*c*14],[*c*15],[*c*16]	—	[*r*23],[*r*24]
3. tézis	—	[*c*5],[*c*6],[*c*7],[*c*8]*,[*c*9]*,[*c*12]	[*e*19]	[*r*22]

* A megjelölt publikációk több tézishez is kapcsolódnak.

A cikkek besorolása a doktori iskola publikációs pontozásához használt besorolást követi.

Folyóiratcikkek

[*j*1] Dániel Darvas, István Majzik, and Enrique Blanco Viñuela. PLC program translation for verification purposes. *Periodica Polytechnica, Electrical Engineering and Computer Science*, 2017. URL: http://mit.bme.hu/~darvas/publications/PerPol2017_DarvasEtAl.pdf. Elfogadva, megjelenés alatt.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*j*2] Dániel Darvas, András Vörös, and Tamás Bartha. Improving saturation-based bounded model checking. *Acta Cybernetica* 22(3), 2016, pp. 573–589. DOI: 10.14232/actacyb.22.3.2016.2.

▷ *Saját kontribúció, közös cikk a diplomaterv-konzulenseimmel.*

[*j*3] Borja Fernández Adiego, Dániel Darvas, Enrique Blanco Viñuela, Jean-Charles Tournier, Simon Bliudze, Jan Olaf Blech, and Víctor M. González Suárez. Applying model checking to industrial-sized PLC programs. *IEEE Transactions on Industrial Informatics* 11(6), 2015, pp. 1400–1410. DOI: 10.1109/TII.2015.2489184. $IF_{2014} = 8.78$.

▷ *Az SFC nyelv köztes reprezentációja (IM) és a köztes modell nyelv definíciója saját kontribúció. A PLC-k viselkedésének leírása és a változóabsztrakció B. Fernández Adiego munkája. A verifikációs módszer többi része közös munka B. Fernández Adiegoval. J.O. Blech kontribúciója a verifikációs módszer helyességének vizsgálata. J-C. Tournier, S. Bliudze és V.M. González Suárez konzulensként segítették a munkát.*

[*j*4] András Vörös, Dániel Darvas, and Tamás Bartha. Bounded saturation-based CTL model checking. *Proceedings of the Estonian Academy of Sciences* 62(1), 2013, pp. 59–70. DOI: 10.3176/proc.2013.1.07. $IF_{2013} = 0.37$.

▷ *Saját kontribúció, közös cikk a diplomaterv-konzulenseimmel.*

Nemzetközi konferenciacikkek

[*c*5] Dániel Darvas, István Majzik, and Enrique Blanco Viñuela. Well-formedness and invariant checking of PLCspecif specifications. In: *Proceedings of the 24th PhD Mini-Symposium*, Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2017. In press.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*c*6] Dániel Darvas, István Majzik, and Enrique Blanco Viñuela. Conformance checking for programmable logic controller programs and specifications. In: *11th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pp. 29–36. IEEE, 2016. DOI: 10.1109/SIES.2016.7509409.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*c*7] Dániel Darvas, Enrique Blanco Viñuela, and István Majzik. PLC code generation based on a formal specification language. In: *14th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 389–396. IEEE, 2016. URL: http://mit.bme.hu/~darvas/publications/INDIN2016_DarvasEtAl.pdf.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*c*8] Dániel Darvas. Practice-oriented formal methods for PLC programs of industrial control systems. In: *Proceedings of the PhD Symposium at iFM'16 on Formal Methods: Algorithms, Tools and Applications (PhD-iFM'16)*, Reykjavik University, 2016. URL: http://mit.bme.hu/~darvas/publications/PhD-iFM2016_Darvas.pdf. Bővített absztrakt. Elfogadva és bemutatva, megjelenés alatt.

[*c*9] Dániel Darvas, István Majzik, and Enrique Blanco Viñuela. Formal verification of safety PLC based control software. In: Erika Ábrahám and Marieke Huisman (eds.), *Integrated Formal Methods*, Lecture Notes in Computer Science, vol. 9681, pp. 508–522. Springer, 2016. DOI: 10.1007/978-3-319-33693-0_32.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*c*10] András Vörös, Dániel Darvas, Vince Molnár, Attila Klenik, Ákos Hajdu, Attila Jámbor, Tamás Bartha, and István Majzik. PetriDotNet 1.5: Extensible Petri net editor and analyser for education and research. In: Fabrice Kordon and Daniel Moldt (eds.), *Application and Theory of Petri Nets and Concurrency*, Lecture Notes in Computer Science, vol. 9698, pp. 123–132. Springer, 2016. DOI: 10.1007/978-3-319-39086-4_9.

▷ *A PetriDotNet keretrendszer megvalósítása a cikk szerzőinek közös munkája, amely Szilvási Bertalan eredeti munkáján alapul. A PetriDotNetben használható szaturációalapú modellellenőrzési algoritmusok fejlesztése Jámbor A. és Darvas D. közös munkája volt, Vörös A. és Bartha T. felügyelete alatt. A szaturációalapú korlátos modellellenőrző algoritmus saját kontribúció, Vörös A. és Bartha T. felügyelete alatt.*

[*c*11] Dániel Darvas, István Majzik, and Enrique Blanco Viñuela. Generic representation of PLC programming languages for formal verification. In: *Proceedings of the 23rd PhD Mini-Symposium*, pp. 6–9. Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2016. DOI: 10.5281/zenodo.51064.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*c*12] Dániel Darvas, Enrique Blanco Viñuela, and István Majzik. A formal specification method

for PLC-based applications. In: Lou Corvetti, Kathleen Riches, and Volker R.W. Schaa (eds.), *Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems*, pp. 907–910. JACoW, 2015. DOI: 10.18429/JACoW-ICALEPCS2015-WEPGF091.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*c*13] Dániel Darvas, Borja Fernández Adiego, and Enrique Blanco Viñuela. PLCverif: A tool to verify PLC programs based on model checking techniques. In: Lou Corvetti, Kathleen Riches, and Volker R.W. Schaa (eds.), *Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems*, pp. 911–914. JACoW, 2015. DOI: 10.18429/JACoW-ICALEPCS2015-WEPGF092.

▷ *A verifikációs folyamat magas szintű elvei a cikk szerzőinek közös munkája. Az eszköz részletes tervezése és fejlesztése saját kontribúció.*

[*c*14] Borja Fernández Adiego, Dániel Darvas, Enrique Blanco Viñuela, Jean-Charles Tournier, Víctor M. González Suárez, and Jan Olaf Blech. Modelling and formal verification of timing aspects in large PLC programs. In: Edward Boje and Xiaohua Xia (eds.), *Proceedings of the 19th IFAC World Congress*, IFAC Proceedings Volumes, vol. 47 (3), pp. 3333–3339. Elsevier, 2014. DOI: 10.3182/20140824-6-ZA-1003.01279.

▷ *A konkrét időreprezentációs módszer („realistic approach", Section 4.1) közös munka B. Fernández Adiegoval. Az absztrakt időreprezentációs módszer (Section 4.2) saját kontribúció. A két megközelítés közti finomítási reláció definiálása J.O. Blech munkája.*

[*c*15] Dániel Darvas, Borja Fernández Adiego, András Vörös, Tamás Bartha, Enrique Blanco Viñuela, and Víctor M. González Suárez. Formal verification of complex properties on PLC programs. In: Erika Ábrahám and Catuscia Palamidessi (eds.), *Formal Techniques for Distributed Objects, Components, and Systems*, Lecture Notes in Computer Science, vol. 8461, pp. 284–299. Springer, 2014. DOI: 10.1007/978-3-662-43613-4_18.

▷ *A verifikációs folyamat magas szintű elvei a cikk szerzőinek közös munkája. A modellredukciós algoritmusok fejlesztése és vizsgálata (Section 3) a saját kontribúcióm.*

[*c*16] Borja Fernández Adiego, Dániel Darvas, Jean-Charles Tournier, Enrique Blanco Viñuela, and Víctor M. González Suárez. Bringing automated model checking to PLC program development – A CERN case study. In: Jean-Jacques Lesage, Jean-Marc Faure, José E. Ribiero Cury, and Bengt Lennartson (eds.), *Proceedings of the 12th International Workshop on Discrete Event Systems*, IFAC Proceedings Volumes, vol. 47 (2), pp. 394–399. Elsevier, 2014. DOI: 10.3182/20140514-3-FR-4046.00051.

▷ *A bemutatott esettanulmány közös munka az általam részletesen definiált és megvalósított verifikációs folyamat alapján.*

[*c*17] Dániel Darvas, András Vörös, and Tamás Bartha. Efficient saturation-based bounded model checking of asynchronous systems. In: Ákos Kiss (ed.), *Proceedings of the 13th Symposium on Programming Languages and Software Tools, SPLST'13*, pp. 259–273. Szeged, Hungary: University of Szeged, 2013. URL: http://petridotnet.inf.mit.bme.hu/publications/SPLST2013_DarvasVorosBartha.pdf.

▷ *Saját kontribúció, közös cikk a diplomaterv-konzulenseimmel.*

[*c*18] András Vörös, Dániel Darvas, and Tamás Bartha. Bounded saturation based CTL model checking. In: Jaan Penjam (ed.), *Proceedings of the 12th Symposium on Programming Languages and Software Tools, SPLST'11*, pp. 149–160. Tallinn, Estonia: Tallinn University of Technology, Institute of Cybernetics, 2011. URL: http://petridotnet.inf.mit.bme.hu/publications/SPLST2011_VorosDarvasBartha.pdf.

▷ *Saját kontribúció, közös cikk a szakdolgozat-konzulenseimmel.*

Helyi konferenciacikkek

[*e*19] Dániel Darvas, István Majzik, and Enrique Blanco Viñuela. Requirements towards a formal specification language for PLCs. In: *Proceedings of the 22nd PhD Mini-Symposium*, pp. 18–21. Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2015. DOI: 10.5281/zenodo.14907.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*e*20] Dániel Darvas and András Vörös. Szaturációalapú tesztbemenet-generálás színezett Petri-hálókkal [in Hungarian; Saturation-based test input generation using coloured Petri nets]. In: *Mesterpróba 2013. Konferenciakiadvány*, pp. 48–51. Budapest University of Technology and Economics, 2013. URL: http://petridotnet.inf.mit.bme.hu/publications/Mesterproba2013_Darvas.pdf.

▷ *Saját kontribúció az R3-COP projekt keretében, közös cikk a diplomaterv-témavezetőmmel.*

[*e*21] Dániel Darvas. Szaturáció alapú korlátos modellellenőrzési technikák Petri-hálók analízisére [in Hungarian; Saturation based bounded model checking methods for the analysis of Petri nets]. In: *XVII. Fiatal Műszakiak Tudományos Ülésszaka*, pp. 83–86. Cluj Napoca, Romania: Erdélyi Múzeum-Egyesület Műszaki Tudományok Szakosztálya, 2012. URL: http://petridotnet.inf.mit.bme.hu/publications/FMTU2012_Darvas.pdf.

Kutatási jelentések

[*r*22] Dániel Darvas, Enrique Blanco Viñuela, and István Majzik. *Syntax and semantics of PLC-specif*. Report EDMS 1523877. CERN, 2015. URL: https://edms.cern.ch/document/1523877.

▷ *Saját kontribúció, közös cikk a doktori témavezetőmmel és külső konzulensemmel.*

[*r*23] Borja Fernández Adiego, Dániel Darvas, Jean-Charles Tournier, Enrique Blanco Viñuela, Jan Olaf Blech, and Víctor M. González Suárez. *Automated generation of formal models from ST control programs for verification purposes*. Internal Note CERN-ACC-NOTE-2014-0037. CERN, 2014. URL: http://cds.cern.ch/record/1708853/.

▷ *A köztes modell nyelv formális definíciója saját kontribúció. A PLC-programokhoz tartozó IM generálása a beszámoló szerzőinek közös munkája volt.*

[*r*24] Dániel Darvas, Borja Fernández Adiego, and Enrique Blanco Viñuela. *Transforming PLC programs into formal models for verification purposes*. Internal Note CERN-ACC-NOTE-2013-0040. CERN, 2013. URL: http://cds.cern.ch/record/1629275/.

▷ *A köztes modell nyelv definíciója saját kontribúció. A PLC-programokhoz tartozó IM, és az IM-et leíró NuSMV-modellek generálása a beszámoló szerzőinek közös munkája volt.*

4.2. Tézisekhez nem kapcsolt publikációk

Folyóiratcikkek

[*j*25] Vince Molnár, András Vörös, Dániel Darvas, Tamás Bartha, and István Majzik. Component-wise incremental LTL model checking. *Formal Aspects of Computing* 28(3), 2016, pp. 345–379. DOI: 10.1007/s00165-015-0347-x. $IF_{2014} = 0.80$.

[*j*26] András Vörös, Dániel Darvas, Attila Jámbor, and Tamás Bartha. Advanced saturation-based model checking of well-formed coloured Petri nets. *Periodica Polytechnica, Electrical Engineering and Computer Science* 58(1), 2014, pp. 3–13. DOI: 10.3311/PPee.2080.

Nemzetközi konferenciacikkek

[*c*27] Vince Molnár, Dániel Darvas, András Vörös, and Tamás Bartha. Saturation-based incremental LTL model checking with inductive proofs. In: Christel Baier and Cesare Tinelli (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, vol. 9035, pp. 643–657. Springer, 2015. DOI: 10.1007/978-3-662-46681-0_58.

[*c*28] Tamás Bartha, András Vörös, Attila Jámbor, and Dániel Darvas. Verification of an industrial safety function using coloured Petri nets and model checking. In: *Proceedings of the 14th International Conference on Modern Information Technology in the Innovation Processes of the Industrial Enterprises (MITIP 2012)*, pp. 472–485. Hungarian Academy of Sciences, Computer and Automation Research Institute, 2012. URL: http://petridotnet.inf.mit.bme.hu/publications/MITIP2012_BarthaEtAl.pdf.

[*c*29] András Vörös, Tamás Bartha, Dániel Darvas, Tamás Szabó, Attila Jámbor, and Ákos Horváth. Parallel saturation based model checking. In: *Proceedings of the 10th International Symposium on Parallel and Distributed Computing (ISPDC)*, pp. 94–101. IEEE, 2011. DOI: 10.1109/ISPDC.2011.23.

4.3. További munkák

[*a*30] Dániel Darvas. Incremental extension of the saturation algorithm-based bounded model checking of Petri nets. Master's thesis. Budapest University of Technology and Economics, 2014. URL: http://petridotnet.inf.mit.bme.hu/publications/Diplomaterv2013_Darvas.pdf.

[*a*31] Dániel Darvas. Petri-háló alapú formális modellek analízise hatékony korlátos modellellenőrzési technikák segítségével [in Hungarian; Efficient bounded model checking techniques for Petri net based formal models]. Bachelor's thesis. Budapest University of Technology and Economics, 2011.

[*a*32] Dániel Darvas and Attila Jámbor. Komplex rendszerek modellezése és verifikációja [in Hungarian; Modeling and verification of complex systems]. Scientific Students' Association Report. 2011. URL: http://petridotnet.inf.mit.bme.hu/publications/TDK2011_DarvasJambor.pdf.

▷ *A szaturációs algoritmus kiterjesztése Petri-hálókra Jámbor A. munkája volt. A szaturációalapú korlátos modellellenőrző algoritmus saját kontribúció.*

[*a*33] Dániel Darvas. Szaturáció alapú automatikus modellellenőrző fejlesztése aszinkron rendszerekhez [in Hungarian; Implementing a saturation-based model checker of asynchronous systems]. Scientific Students' Association Report. 2010. URL: http://petridotnet.inf.mit.bme.hu/publications/OTDK2011_Darvas.pdf.

Hivatkozások

[Abr96] Jean-Raymond Abrial. *The B-book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science* 126(2), 1994, pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[Ama] José Nelson Amaral. About Computing Science Research Methodology. URL: https://webdocs.cs.ualberta.ca/~c603/readings/research-methods.pdf.

[Amn+01] Tobias Amnell et al. UPPAAL – Now, next, and future. In: *Modeling and Verification of Parallel Processes*, Lecture Notes in Computer Science, vol. 2067, pp. 99–124. Springer, 2001. DOI: 10.1007/3-540-45510-8_4.

[Avi+04] Algirdas Avižienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1(1), 2004, pp. 11–33. DOI: 10.1109/TDSC.2004.2.

[BBK12] Sebastian Biallas, Jörg Brauer, and Stefan Kowalewski. Arcade.PLC: A verification platform for programmable logic controllers. In: *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 338–341. IEEE, 2012. DOI: 10.1145/2351676.2351741.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[Bla77] Michel Blanchard. Le GRAFCET pour une representation normalisée de cahier des charges d'un automatisme logique. *Automatique et Informatique Industrielle* (61), 1977, pp. 27–32.

[Bur+92] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation* 98(2), 1992, pp. 142–170. DOI: 10.1016/0890-5401(92)90017-A.

[But02] Michael Butler. A system-based approach to the formal development of embedded controllers for a railway. *Design Automation for Embedded Systems* 6(4), 2002, pp. 355–366. DOI: 10.1023/A:1016503426126.

[Can+00] Géraud Canet, Sandrine Couffin, Jean-Jacques Lesage, Antoine Petit, and Philippe Schnoebelen. Towards the automatic verification of PLC programs written in instruction list. In: *IEEE International Conference on Systems, Man & Cybernetics*, vol. 4, pp. 2449–2454. IEEE, 2000. DOI: 10.1109/ICSMC.2000.884359.

[Cav+14] Roberto Cavada et al. The nuXmv symbolic model checker. In: *Computer Aided Verification*, Lecture Notes in Computer Science, vol. 8559, pp. 334–342. Springer, 2014. DOI: 10.1007/978-3-319-08867-9_22.

[CE82] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In: *Logic of Programs*, Lecture Notes in Computer Science, vol. 131, pp. 52–71. Springer, 1982. DOI: 10.1007/BFb0025774.

[CES09] Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis. Model checking: Algorithmic verification and debugging. *Communications of the ACM* 52(11), 2009, pp. 74–84. DOI: 10.1145/1592761.1592781.

[Cha+02] Pankaj Chauhan, Edmund M. Clarke, James Kukula, Samir Sapra, Helmut Veith, and Dong Wang. Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis. In: Mark D. Aagaard and John W. O'Leary (eds.), *Formal Methods in Computer-Aided Design*, Lecture Notes in Computer Science, vol. 2517, pp. 33–51. Springer, 2002. DOI: 10.1007/3-540-36126-X_3.

[Cla08] Edmund M. Clarke. The birth of model checking. In: Orna Grumberg and Helmut Veith (eds.), *25 Years of Model Checking*, Lecture Notes in Computer Science, vol. 5000, pp. 1–26. Springer, 2008. DOI: 10.1007/978-3-540-69850-0_1.

[CLS01] Gianfranco Ciardo, Gerald Lüttgen, and Radu Siminiceanu. Saturation: An efficient iteration strategy for symbolic state-space generation. In: *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, vol. 2031, pp. 328–342. Springer, 2001. DOI: 10.1007/3-540-45319-9_23.

[CMS08] José C. Campos, José Machado, and Eurico Seabra. Property patterns for the formal verification of automated production systems. In: *Proceedings of the 17th IFAC World Congress*, IFAC Proceedings Volumes, vol. 41 (2), pp. 5107–5112. Elsevier, 2008. DOI: 10.3182/20080706-5-KR-1001.00858.

[CNQ05] Gianpiero Cabodi, Sergio Nocco, and Stefano Quer. Are BDDs still alive within sequential verification? *International Journal on Software Tools for Technology Transfer* 7(2), 2005, pp. 129–142. DOI: 10.1007/s10009-004-0172-7.

[Cop+01] Fady Copty, Limor Fix, Ranan Fraer, Enrico Giunchiglia, Gila Kamhi, Armando Tacchella, and Moshe Y. Vardi. Benefits of bounded model checking at an industrial setting. In: Gérard Berry, Hubert Comon, and Alain Finkel (eds.), *Computer Aided Verification*, Lecture Notes in Computer Science, vol. 2102, pp. 436–453. Springer, 2001. DOI: 10.1007/3-540-44585-4_43.

[CS03] Gianfranco Ciardo and Radu Siminiceanu. Structural symbolic CTL model checking of asynchronous systems. In: *Computer Aided Verification*, Lecture Notes in Computer Science, vol. 2725, pp. 40–53. Springer, 2003. DOI: 10.1007/978-3-540-45069-6_4.

[CZJ12] Gianfranco Ciardo, Yang Zhao, and Xiaoqing Jin. Ten years of saturation: A Petri net perspective. In: *Transactions on Petri Nets and Other Models of Concurrency V*, Lecture Notes in Computer Science, vol. 6900, pp. 51–95. Springer, 2012. DOI: 10.1007/978-3-642-29072-5_3.

[DAC99] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In: *Proceedings of the 21st International Conference on Software Engineering*, pp. 411–420. ACM, 1999. DOI: 10.1145/302405.302672.

[FBM13] Borja Fernández Adiego, Enrique Blanco Viñuela, and Alexey Merezhin. Testing & verification of PLC code for process control. In: *Proceedings of the 14th International Conference on Accelerator & Large Experimental Physics Control Systems*, pp. 1258–1261. JACoW, 2013. URL: http://accelconf.web.cern.ch/AccelConf/ICALEPCS2013/papers/thppc080.pdf.

[Fer+13] Borja Fernández Adiego, Enrique Blanco Viñuela, Jean-Charles Tournier, Víctor M. González Suárez, and Simon Bliudze. Model-based automated testing of critical PLC programs. In: *11th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 722–727. 2013. DOI: 10.1109/INDIN.2013.6622973.

[Fer14] Borja Fernández Adiego. Bringing automated formal verification to PLC program development. Ph.D. thesis. University of Oviedo, 2014. URL: http://cds.cern.ch/record/1983193.

[Fix08] Limor Fix. Fifteen years of formal property verification in Intel. In: Orna Grumberg and Helmut Veith (eds.), *25 Years of Model Checking*, Lecture Notes in Computer Science, vol. 5000, pp. 139–144. Springer, 2008. DOI: 10.1007/978-3-540-69850-0_8.

[Fok00] Wan Fokkink. *Introduction to Process Algebra*. 1st edition. Springer, 2000. DOI: 10.1007/978-3-662-04293-9.

[GSF08] Vincent Gourcuff, Olivier de Smet, and Jean-Marc Faure. Improving large-sized PLC programs verification using abstractions. In: *Proceedings of the 17th IFAC World Congress*, IFAC Proceedings Volumes, vol. 41 (2), pp. 5101–5106. Elsevier, 2008. DOI: 10.3182/20080706-5-KR-1001.00857.

[Hav+00] Klaus Havelund, Mike Lowry, Seung Joon Park, Charles Pecheur, John Penix, Willem Visser, and Jon L. White. Formal analysis of the Remote Agent before and after flight. In: C. Michael Holloway (ed.), *Lfm2000: Fifth NASA Langley Formal Methods Workshop*, pp. 163–174. NASA, 2000.

[HLR98] Mats P.E. Heimdahl, Nancy G. Leveson, and Jon D. Reese. Experiences from specifying the TCAS II requirements using RSML. In: *Proceedings of the 17th AIAA/IEEE/SAE Digital Avionics Systems Conference*, vol. 1, pp. C43/1–C43/8. IEEE, 1998. DOI: 10.1109/DASC.1998.741499.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985. URL: http://www.usingcsp.com/cspbook.pdf.

[I1012] *IEEE Std 1012-2012 – IEEE Standard for system and software verification and validation*. IEEE, 2012.

[I13568] *ISO/IEC 13568 Information technology – Z formal specification notation – Syntax, type system and semantics*. ISO/IEC, 2002.

[I60848] *IEC 60848 – GRAFCET specification language for sequential function charts*. IEC, 2013.

[I61131-3] *IEC 61131-3 Programmable controllers – Part 3: Programming languages*. IEC, 2013.

[I61508-2] *IEC 61508-2 Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems*. IEC, 2010.

[I8807] *ISO 8807 Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour*. ISO, 1989.

[Kai+09] Roope Kaivola et al. Replacing testing with formal verification in Intel® Core™ i7 processor execution engine validation. In: Ahmed Bouajjani and Oded Maler (eds.), *Computer Aided Verification*, Lecture Notes in Computer Science, vol. 5643, pp. 414–429. Springer, 2009. DOI: 10.1007/978-3-642-02658-4_32.

[Kan+15] Gijs Kant, Alfons Laarman, Jeroen Meijer, Jaco van de Pol, Stefan Blom, and Tom van Dijk. LTSmin: High-performance language-independent model checking. In: Christel Baier and Cesare Tinelli (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, vol. 9035, pp. 692–707. Springer, 2015. DOI: 10.1007/978-3-662-46681-0_61.

[Kni+97] John C. Knight, Colleen L. DeJong, Matthew S. Gibble, and Luís G. Nakano. Why are formal methods not used more widely? In: C. Michael Holloway and Kelly J. Hayhurst (eds.), *Fourth NASA Langley Formal Methods Workshop (LFM)*, pp. 1–12. 1997. URL: http://www.cs.virginia.edu/~jck/publications/lfm.97.pdf.

[Lam00] Axel van Lamsweerde. Formal specification: A roadmap. In: Anthony Finkelstein (ed.), *Proceedings of the Conference on The Future of Software Engineering (ICSE)*, pp. 147–159. ACM, 2000. DOI: 10.1145/336512.336546.

[Lam09] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[Lju+10] Oscar Ljungkrantz, Knut Åkesson, Martin Fabian, and Chengyin Yuan. A formal specification language for PLC-based control logic. In: *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 1067–1072. 2010. DOI: 10.1109/INDIN.2010.5549591.

[LNN13] Tim Lange, Martin R. Neuhäußer, and Thomas Noll. Speeding up the safety verification of programmable logic controller code. In: *Hardware and Software: Verification and Testing*, Lecture Notes in Computer Science, vol. 8244, pp. 44–60. Springer, 2013. DOI: 10.1007/978-3-319-03077-7_4.

[LSP07] Thierry Lecomte, Thierry Servat, and Guilhem Pouzancre. Formal methods in safety-critical railway systems. In: *Proceedings of the 10th Brasilian Symposium on Formal Methods (SBMF)*, 2007. URL: http://rodin.cs.ncl.ac.uk/Publications/fm_sc_rs_v2.pdf.

[Luk+13] Tomaž Lukman, Giovanni Godena, Jeff Gray, Marjan Heričko, and Stanko Strmčnik. Model-driven engineering of process control software – Beyond device-centric abstractions. *Control Engineering Practice* 21(8), 2013, pp. 1078–1096. DOI: 10.1016/j.conengprac.2013.03.013.

[Mar94] John J. Marciniak. *Encyclopedia of Software Engineering*. Vol. 1. John Wiley & Sons, 1994. DOI: 10.1002/0471028959.

[MB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In: C. R. Ramakrishnan and Jakob Rehof (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer, 2008. DOI: 10.1007/978-3-540-78800-3_24.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4), 1989, pp. 541–580. DOI: 10.1109/5.24143.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In: Mariangiola Dezani-Ciancaglini and Ugo Montanari (eds.), *International Symposium on Programming*, Lecture Notes in Computer Science, vol. 137, pp. 337–351. Springer, 1982. DOI: 10.1007/3-540-11494-7_22.

[R3C] *R3-COP final dissemination, demonstration, PR and E&T report (Period 3+, M42)*. Tech. rep. WP 8, D 8.1.4c+d, D 8.1.6. 2013. URL: http://www.r3-cop.eu/wp-content/uploads/2013/10/R3-COP_D-8.1.4d_-D-8.1.6_1.2r_Dissemination_Final.pdf.

[SD08] André Sülflow and Rolf Drechsler. Verification of PLC programs using formal proof techniques. In: Géza Tarnai and Eckehard Schnieder (eds.), *Formal Methods for Automation and Safety in Railway and Automotive Systems*, pp. 43–50. L'Harmattan, 2008. URL: http://www.informatik.uni-bremen.de/agra/doc/konf/08_forms_VerificationPLCPrograms.pdf.

[SF11] Doaa Soliman and Georg Frey. Verification and validation of safety applications based on PLCopen safety function blocks. *Control Engineering Practice* 19(9), 2011, pp. 929–946. DOI: 10.1016/j.conengprac.2011.01.001.

[Sou+09] Jean Souyris, Virginie Wiels, David Delmas, and Hervé Delseny. Formal verification of avionics software products. In: Ana Cavalcanti and Dennis R. Dams (eds.), *FM 2009: Formal Methods*, Lecture Notes in Computer Science, vol. 5850, pp. 532–546. Springer, 2009. DOI: 10.1007/978-3-642-05089-3_34.

[Woo+09] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. Formal methods: Practice and experience. *ACM Computing Surveys* 41(4), 2009, 19:1–19:36. DOI: 10.1145/1592434.1592436.

[YCL09] Andy Jinqing Yu, Gianfranco Ciardo, and Gerald Lüttgen. Decision-diagram-based techniques for bounded reachability checking of asynchronous systems. *International Journal on Software Tools for Technology Transfer* 11(2), 2009, pp. 117–131. DOI: 10.1007/s10009-009-0099-0.

[ZC09] Yang Zhao and Gianfranco Ciardo. Symbolic CTL model checking of asynchronous systems using constrained saturation. In: *Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science, vol. 5799, pp. 368–381. Springer, 2009. DOI: 10.1007/978-3-642-04761-9_27.