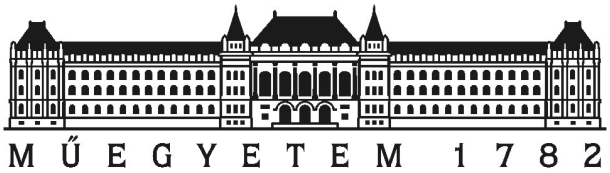


BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

INFORMÁCIÓS RENDSZEREK PETRI HÁLÓ ALAPÚ EGYÜTTES VERIFIKÁCIÓJA ÉS OPTIMALIZÁLÁSA

PHD TÉZISFÜZET

VARRÓ-GYAPAY SZILVIA

TÉMAVEZETŐ:
DR. PATARICZA ANDRÁS, DSC
EGYETEMI TANÁR

BUDAPEST, 2014. SZEPTEMBER

1. Előzmények és célkitűzések

1.1. Biztonságkritikus rendszerek

Napjainkban az információs rendszerek komplexitásának növekedésével egyre inkább előtérbe került a rendszerek magas szintű szolgáltatásminőségének biztosítása. Biztonságkritikus rendszereknél mint az autóipari vagy repülőgépipari szoftverek a szolgáltatás minőségének biztosítása elengedhetetlen, hiszen akár egy apró hiba is hatalmas károkat okozhat mind a rendszerben, mind emberéletben. Az üzleti élet területén a nagy nemzetközi cégek szolgáltatásminőség biztosításának célja azon szolgáltatás kiesésének vagy a rendszer meghibásodásának elkerülése, amelyek következtében a cégeknek anyagi veszteséggel kell számolnia.

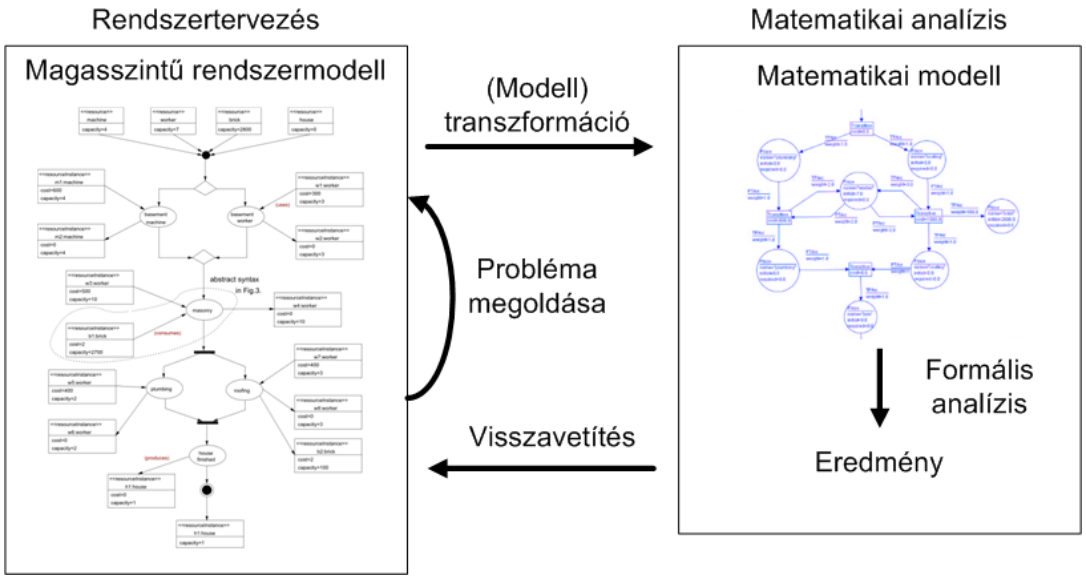
A repülőgépiparban, illetve az autógyártásban használt szabványok, mint a DO-178C, illetve ISO 26262 szabványok szigorú biztonsági követelményeket írnak elő a fejlesztési folyamatra a követelményspecifikációtól kezdve az analízisen, a verifikáción és a validáción keresztül a dokumentációig, melyeknek célja a biztonságos működés garantálása a rendszer teljes életciklusa során. Ennek köszönhetően egy új, biztonságkritikus rendszer minősítési költsége horribilis, melynek 70-80%-át a verifikáció és a validáció költségei teszik ki, így napjainkban kihívás az olyan verifikációs és validációs eljárások kidolgozása amelyek egyszerre garantálják a helyes működést és csökkentik a minősítés költségeit.

Egy megfelelő minőségű szolgáltatásnak több követelménynek is meg kell felelnie. Ilyen a szolgáltatás funkcionális helyességének garantálása, vagyis a specifikáció által előírt kimenet előállítása az adott bemenetből, illetve a szolgáltatás teljesítésére vonatkozó határidők betartása nagy megbízhatóság és rendelkezésre állás mellett. Ugyanakkor elvárás, hogy a szolgáltatás nyújtása optimális legyen, amely jelentheti akár a költséghatékony megoldást, akár a minél rövidebb idő alatti teljesítést. Ennek megfelelően különböztetünk meg a rendszer tervezése során funkcionális, extrafunkcionális és optimalizálási követelményeket.

1.1.1. Kritikus rendszerek modellvezérelt fejlesztése

A kritikus rendszerek tervezésében kulcsszerepet játszanak a modellvezérelt tervezési technikák melyeket számos alkalmazás fejlesztésében használnak mind az autóipar, mind a repülőgépipar területén [40, 41]. A modellvezérelt fejlesztés során a rendszerkövetelményeket és a tervezést magasszintű, vizuális mérnöki modellekkel írják le. Modelltranszformáció segítségével a tervezési modellekből matematikai modell generálható, amelyeken végzett analízis által már a tervezés korai szakaszában felderíthető számos tervezési hiba [29]. A modellvezérelt formális analízis folyamata az 1. ábrán látható.

A modellvezérelt fejlesztés segítségével egyszerre növelhető a minőség és csökkenthetőek a költségek hiszen a formális analízis még a fejlesztés korai fázisában képes felderíteni a tervezési hibákat, így jelentős költséget takarítva meg a költséges újratervezés és implementálás elkerülése révén, amely akkor következik be, ha ezen hibákat például csak az elkészült kód tesztelése során derítjük fel (lásd: hagyományos tesztelés). Egy



1. ábra. Kritikus rendszerek modellvezérelt analízise

biztonságkritikus alkalmazás kódjának validált modellből történő automatikus generálása egyben a generált kód helyességét is garantálja, ezzel is növelve a fejlesztés hatékonyságát. Mindemellett a modellvezérelt fejlesztés segítségével más tervezési produktumok is előállíthatók mint hibafák, tesztesetek, vagy dokumentáció, melyek a nyomonkövethetőséget támogatják [25].

1.1.2. Formális módszerek

A formális módszerek, mint állapottérképek, Petri hálók, adatfolyamhálók, processzalgebrák, vagy időzített automaták precíz szemantikát és végrehajtási modellt adnak a modellvezérelt fejlesztés során létrehozott modellek működésére. Egyúttal az ezen matematikai modelleken végzett analízis eredményének visszavetítésével már a tervezés korai szakaszában hibák deríthetők fel. A formális módszerek használatának előnye a hagyományos teszteléssel szemben, hogy míg az utóbbi a hibák meglétét vizsgálja a kódban addig a formális módszerek sok esetben képesek azt bizonyítani, hogy a hiba nem is léphet fel. Éppen emiatt a legtöbb biztonságkritikus rendszer minősítési szabványában elfogadott a formális módszerek használata. Mindemellett a hagyományos tesztelés és a formális módszerek ötvözése is elterjedt a rendszerfejlesztés során, főleg a repülőgépiparban.

A fél-formális nyelvek, mint a Unified Modeling Language (UML) [33], a System Modeling Language (SysML) [24] vagy a Business Process Modeling Notation (BPMN) [23] olyan magasszintű szabványos modellezési nyelvek, amelyek könnyen érthető, precízen definiált struktúrával rendelkeznek, így használatuk egyfajta kommunikációs csatornát jelent a fejlesztők és az egyéb döntéshozók között. Mindazonáltal ezen nyelvek nem rendelkeznek kiforrott, szabványos szemantikával, így a modell működésére nézve

ezek nem adnak egyértelmű leírást. Ugyanakkor bizonyos követelmények, mint például az időbeli viselkedés leírása csak nehezen, vagy egyáltalán nem írhatók le ezekben a félformális nyelvekben.

Ezzel szemben a formális módszerek olyan precíz matematikai modelleket adnak a tervezéshez, amelyek formális szemantikája egyértelmű és lehetővé teszi a rendszerrel szemben támasztott követelmények formális verifikációját. Ugyanakkor ezen matematikai modellek megértése, analízis módszereik használata erős matematikai hátteret igényel a tervezőtől. A modellvezérelt fejlesztés ezt a rést hivatott áthidalni automatikus modelltranszformáció segítségével. A modelltranszformáció által lehetővé válik, hogy a rendszer tervezői magasszintű nyelven tervezzenek, anélkül, hogy a mögötte lévő matematikai modellt alaposabban ismernék, valamint hogy a matematikai modellen végzett formális verifikáció eredményét visszavetítve az eredeti modellben lássák a felderített hibákat. Szőke Ákos és társai kimutatták [26], hogy a formális módszerek és a modellvezérelt fejlesztés együttes alkalmazása egy nagyságrenddel növeli a minőséget és csökkenti a költségeket a hagyományos V&V fejlesztéssel összehasonlítva, amely nem alkalmaz formális módszereket.

A validáció és verifikáció folyamatában gyakran használt módszer a modellellenőrzés. A modellellenőrzés során a követelmények teljesülését automatikusan vizsgáljuk az adott modell állapotterének kimerítő bejárásával [27]. Egy követelmény vagy teljesül az adott modellben, vagy a bejárás során olyan ellenpélda konstruálható, amely sérti az adott követelményt.

A modellellenőrző bemenete egy véges címkézett tranzíciós rendszer, mely egy gráf, ahol a rendszer állapotai a gráf csúcsai, és a tranzíciók (állapotátmenetek) az élek az állapotok között. A rendszerrel szemben támasztott követelményeket lineáris temporális logikai (LTL) kifejezésekkel adhatjuk meg, amelyek állapotváltozókból és temporális logikai operátorokból állnak. Egy explicit állapot alapú modellellenőrző, mint például a SPIN [36] modellellenőrző eszköz mind a tranzíciós rendszert, mind a lineáris temporális logikai kifejezést egy-egy automatává fordítja le. Az ellenőrzés során a lineáris temporális logikai kifejezés automatájának negáltjának és a tranzíciós rendszer automatájának a szorzatautomatáját vizsgálja: ha a szorzatautomatában nincs olyan állapotátmenet sorozat, amely elfogadó állapothoz vezethet, akkor a követelmény teljesül. Ellenkező esetben az elfogadó állapothoz vezető út egy ellenpélda a rendszermodellben a követelmény nem teljesülésére.

1.2. Optimalizálási módszerek

Információs rendszerek tervezése során gyakori kihívás, hogy a rendszer helyesen *és* egyben hatékonyan is működjön a követelményeknek megfelelően. Tipikus példák a gyártórendszerek és üzleti folyamatok, melyek esetén központi probléma az olyan megoldás kidolgozása, amely optimális költség, illetve idő szempontjából. Ezen kihívásra a formális analízis és optimalizálási módszerek kombinálása adhat megoldást.

A lineáris programozási (LP) módszerek hatékony megoldást nyújtanak optimalizálási problémák megoldására [47]. Egy LP probléma megoldása során valós döntési változók

fölötti lineáris kényszerek kielégítése mellett keressük a döntési változók egy lineáris célfüggvényének optimumát.

Egy lineáris programozási feladatot egész lineáris programozási (ILP) feladatnak nevezünk, ha a döntési változók csak egész értéket vehetnek fel. Ha a változók között megengedett mind valós mind egész értékű változó, akkor a problémát vegyes egész lineáris programozási (MILP) feladatnak nevezzük.

Számos technikát fejlesztettek ki az LP problémák megoldására, mint amilyen a szimplex módszer [47] vagy belső-pont módszerek [37]. A MILP illetve ILP problémák megoldására jól ismert módszer az úgynevezett Branch and Bound módszer, melynek alapötlete a szétválasztás és korlátozás lépések egymás utáni alkalmazása. Ennek során az aktuális (rész)probléma (i) kisebb megoldásterű, az eredeti megoldásteret lefedő problémákra bontható, (ii) melyekre hatékonyan számolható például valamely belső pont módszerrel felső, ill. alsó korlát.

Korlátozás és relaxáció. Az egészértékű programozási feladatok megoldása jóval időigényesebb és összetettebb mint a folytonos (LP) problémáé. A már említett korlátozás lépésben egy minimalizálási, illetve maximalizálási részproblémára egy alsó, illetve felső korlát számolható ki a folytonos LP problémára, amely alsó illetve felső korlát az ILP problémára is. Ha ez az érték nagyobb illetve kisebb, mint egy már ismert megoldáshoz tartozó optimum, akkor ez a részprobléma már nem vezet jobb megoldáshoz. A folytonos LP probléma megoldását a probléma relaxálásának nevezzük.

Számos mérnöki probléma leírható (vegyes egész) lineáris problémaként mint a gyártási folyamatok szintézis problémái, szállítási feladatok, vegyészeti folyamatok vagy VLSI tervezés.

1.3. Az optimális trajektória probléma: együttes optimalizálás és verifikáció

Míg a verifikációs és validációs technikák feladata a rendszermodell helyességének bizonyítása, addig az optimalizálási módszerek a rendszer hatékonyságvizsgálatában játszanak szerepet. Az e két területen ismert megoldások kombinálása gyakran olyan elérhetőségi problémaként írható le, amelyet *optimális trajektória problémaként* nevezhetünk meg: adott kezdőállapotból kiindulva keresünk egy olyan optimális utat, amely megfelel a rendszerrel szemben támasztott követelményeknek.

A hagyományos elérhetőségi probléma annak a kérdésnek az eldöntése, hogy egy adott állapot elérhető-e egy adott kezdőállapotból a rendszerben, vagy sem. Az optimális trajektória problémában nemcsak a kérdés eldöntése szükséges, hanem a két állapot közötti út meghatározása is. A célállapot lehet egy konkrét állapot, ahol az összes állapotváltozó egy előre magadott értéket vesz fel, vagy lehet egy részleges állapot, amikor az állapotváltozók egy részhalmazára adunk csak meg feltételt.

Ilyen problémákkal foglalkoznak a termeléstervezési és ütemezési algoritmusok [30, 42, 46], különböző játékelméleti stratégiák [31], valamint a tervezési tér bejárása a rendszertervezésben [28].

1.4. Az értekezés célja

Kutatásom célja (i) olyan matematikai modellek megadása volt, amelyekben az optimális trajektória probléma formálisan leírható, valamint (ii) stratégiák és algoritmusok kidolgozása az optimális trajektória megoldására verifikációs és optimalizálási módszerek kombinálásával.

2. Rendszermodellek együttes verifikációja és optimalizálása

A hatékony verifikációs és optimalizálási technikák kombinálása egy olyan keretrendszer esetében működhet, amelyben egyaránt modellezhetők a rendszer kritikus tulajdonságai, a rendszer viselkedése, valamint a mennyiségi paraméterek. A kutatásom során erre a célra a Petri hálók és gráftranszformációs rendszerek költséggel, illetve idővel kiterjesztett változatát használtam.

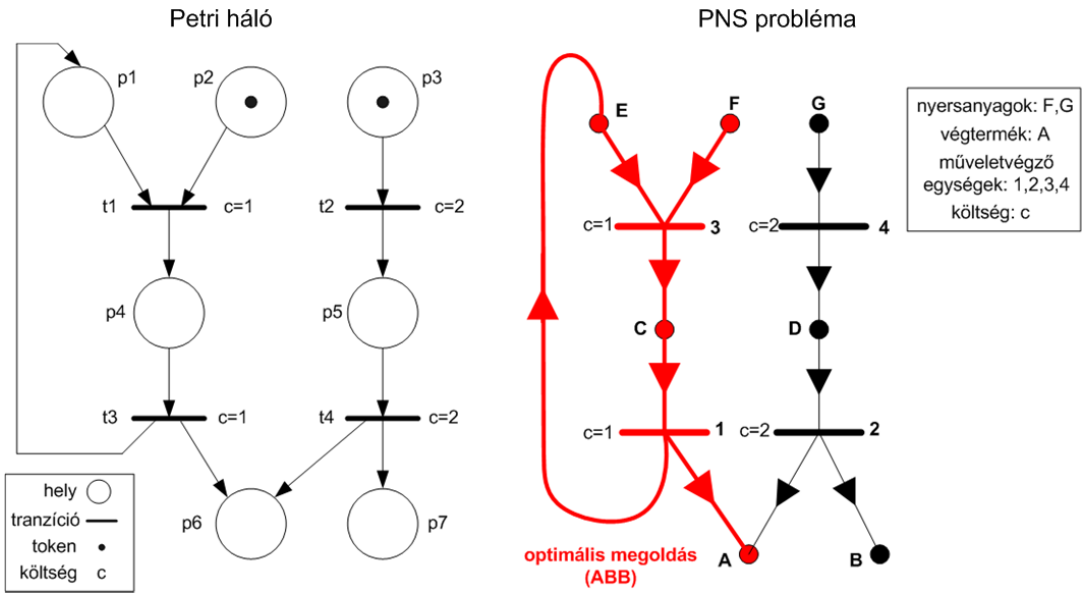
Az optimális trajektória egy Petri hálóban vagy egy gráftranszformációs rendszerben egy olyan út az állapottérben, amely optimális és egyben kielégíti a rendszerrel szemben támasztott követelményeket is. A legegyszerűbb módszer egy ilyen út megtalálására az állapottér kimerítő bejárása, amely azonban nem kivitelezhető végtelen állapottér esetén. Mindazonáltal, a keresési tér véges méretűre korlátozható az utak megszorításával vagy az optimális megoldás értékének korlátozásával.

2.1. Petri hálók

A disszertáció középpontjában a Petri hálók állnak, melyeket Carl Adam Petri vezetett be 1962-ben a disszertációjában [45] folyamatok kommunikációjának modellezésére. A Petri hálók népszerűségüket érthető grafikus reprezentációjuk mellett modellező erejüknek köszönhetik: mind a nemdeterminisztikusság, mind a párhuzamosság modellezését támogatják.

A Petri hálók páros, irányított gráfok, amelyekben a két diszjunkt csúcshalmaz helyeket és tranzíciókat ábrázol (2. ábra). A helyek tokeneket tartalmazhatnak, amelyek eloszlásával jellemezhető a háló állapota. A tokeneket a tranzíciók tüzelése áramoltatja a hálóban: egy tranzíció tüzelése elnyeli a bemeneti helyeken tartalmazott, az élekhez rendelt súlyfüggvénynek megfelelő számú tokent, majd szintén a súlyfüggvény által meghatározott mennyiségben termel tokent a kimeneti helyekre. A háló egy trajektóriája egy végrehajtható tüzelési szekvenciát ír le két állapot között. A háló viselkedése ezen trajektóriákkal írható le.

A Petri hálóknak számos kiterjesztése létezik melyek a modellezett rendszer tulajdonságait ábrázolják, mint például az idő, a költség, vagy a megbízhatósági paraméterek. Jelen disszertáció a költséggel és idővel kiterjesztett modellekkel foglalkozik, ahol a költségek és az időtartamok a rendszer működési fázisaihoz, azaz a tranzícókhoz vannak rendelve: a költség a tranzíció működésének költségét jelenti, míg az idő a tranzíció



2. ábra. Petri háló költség paraméterrel és PNS probléma

működésének időtartamát.

2.1.1. Költségoptimális trajektória

A költségoptimális trajektória probléma egy minimális költségű tüzelési szekvencia keresése a Petri hálóban, amely egy adott kezdőállapotból indulva egy olyan állapotba vezet, amely egy adott állapotot fed, valamint a költsége alacsonyabb egy előre megadott korlátnál.

A Petri hálóban az egyes tranzíciók tüzelésének hatására bekövetkező változás az úgynevezett állapotegyenlettel írható le. Az elérhetőségi problémára, melynek eldöntése során arra keresünk választ, hogy egy konkrét állapot elérhető-e egy adott kezdőállapotból, az állapotegyenlet részleges döntési technikát ad: amennyiben az állapotegyenletnek nincsen megoldása az adott kezdő- és célállapot esetén, a konkrét állapot nem elérhető az adott kezdőállapotból. Ugyanakkor előfordulhat, hogy az állapotegyenlet megoldásával kapott tranzíció tüzelési szám vektorhoz nem tartozik tüzelhető tranzíció szekvencia a Petri hálóban. Ennek oka, hogy a tüzelési szám vektor (Parikh vektor) csak a trajektóriában (ha az létezik egyáltalán) tüzelt tranzíciók számát adja meg, azok tüzelési sorrendjét azonban nem. Így a megoldás megvalósíthatósága nem garantált.

Részleges elérhetőségi probléma esetén az a kérdés, hogy egy olyan állapot elérhető-e a kezdőállapotból, amely lefed egy adott részleges célállapotot, vagyis az állapotváltozóknak csak egy részhalmazára adottak követelmények. Ekkor az állapotegyenlet alapján egy egyenlőtlenségrendszert írhatunk fel.

A költség paraméterrel kibővített Petri hálóban az optimális trajektória keresése során ezt az egyenlőtlenségrendszert egy megfelelő célfüggvénnyel kiegészítve egy ILP problé-

mát kapunk. Ez az ILP probléma egy absztrakciót ad a megoldási térre, mivel (i) minden, a kezdő-, illetve a részleges célállapotnak megfelelő trajektória megoldása az ILP problémának, (ii) de létezhet olyan megoldása is az ILP problémának, amelyhez nem tartozik tüzelhető megoldás a Petri hálóban. Ezáltal az ILP probléma megoldása szintén részleges döntési technikát ad: amennyiben nem létezik megoldása a problémának, az adott részleges célállapot nem érhető el. Ha létezik megoldás, akkor viszont meg kell vizsgálni, hogy a megoldáshoz tartozik-e tüzelhető trajektória. Ha igen, akkor azt meg is kell adni, ha nem, akkor pedig az ILP probléma következő legjobb megoldását kell vizsgálni.

Az ILP probléma megoldására használhatunk hagyományos Branch and Bound módszereket, azonban a megoldási tér tovább szűkíthető egy rokon paradigma, a produkciós hálózatok folyamatszintézisére adott algoritmusok segítségével, melyek az ILP probléma megoldása során figyelembe veszik a hálózat struktúráját is.

2.1.2. Produkciós hálózatok szintézise

A Petri hálók költségoptimális trajektóriájához hasonló produkciós hálózatok szintézisére Friedler Ferenc és társai adtak hatékony megoldást a '90-es években [34, 35]. A folyamatszintézis feladata (röviden PNS) adott nyersanyagok felhasználásával megfelelő mennyiségű végtermék gyártása a műveletvégző egységek segítségével. Ennek mennyiségi paraméterei a műveletvégző egységek anyagfogyasztó- és termelő rátája, valamint a nyersanyagok költsége, és a műveletvégző egységek állandó és működési költsége. Így a feladat nem csupán egy megfelelő termelő hálózat megadása, hanem egyben egy optimális termelő hálózat megadása is.

A produkciós hálókat P-gráfokkal írják le, melyek irányított páros gráfok: a csomópontok egyik típusa az anyagtárolókat, másik típusa pedig a műveletvégző egységeket reprezentálja, a csomópontok közötti élek pedig a köztük levő kapcsolatot (többnyire anyagmozgást) írják le (2. ábra).

Friedler és társai a folyamatszintézis probléma megoldására hatékony algoritmusokat dolgoztak ki a probléma kombinatorikus jellegét kihasználva. Megmutatták, hogy egy megoldás P-gráf struktúrájának meg kell felelnie öt axiómának. Ezeket a gráfokat kombinatorikusan lehetséges megoldás struktúrának nevezik, amelyek recepteknek tekinthetők: megadják, hogy mely összetevőkből (anyagokból) és eszközökkel (műveleti egységekkel) állítható elő az adott termék. Az Accelerated Branch and Bound algoritmus ezen receptek fölött keresi az optimumot így szűkítve a keresési teret.

A Petri hálók és a P-gráfok szerkezete nagy hasonlóságot mutat: a Petri háló helyei a produkciós hálók anyagtárolóinak, a tokenek az anyagoknak, az állapotátmenetet reprezentáló tüzelések pedig a műveletvégző egységeknek feleltethetők meg. Ezen megfeleltetés mentén szintén párhuzam vonható a Petri háló költségoptimális trajektória problémája és a PNS probléma között, mivel mindkét probléma optimális utat keres egy páros gráfban. A két paradigma összehasonlításának egyik célja volt megvizsgálni, hogy a folyamatszintézisben használt módszerek alkalmazhatóak-e a Petri háló költségoptimális trajektória probléma megoldásában.

Mivel a folyamatszintézis módszereit elsősorban a vegyészetben használják, impli-

cit módon feltételezik egyes, nem fogyó, katalizátor anyagok jelenlétét, amely biztosítja, hogy a pusztán algebrai úton kapott eredménynek egy megvalósítható gyártási sorrend megfeleltethető. Ez információs rendszerek esetében azonban nem mindig teljesül, így vizsgálatokat végeztem, hogy milyen jellegű megkötéseket kell tenni a Petri hálóra ahhoz, hogy a PNS algoritmusok alkalmazhatók legyenek arra, illetve, hogy milyen módok léteznek általános Petri hálók esetén a katalizátornak megfelelő struktúra működőképességét explicit módon lehetővé tenni.

Ugyanakkor problémát jelent a PNS módszerek alkalmazásakor, hogy a folyamatszintézis megengedi az egyes műveleti egységek nemcsak egész számszoros alkalmazását. Ennek egyik oka, hogy a PNS algoritmusokat feldolgozó rendszerek folyamattervezésére találták ki, ahol az egyes anyagok transzformációját (pl. vegyületek előállítása) végző műveleti egységek működési ideje törtrésze is lehet az egységnyi időnek, míg a Petri hálóknál a műveleti egységeknek megfelelő tranzícióknak csak egész számszoros tüzelése megengedett.

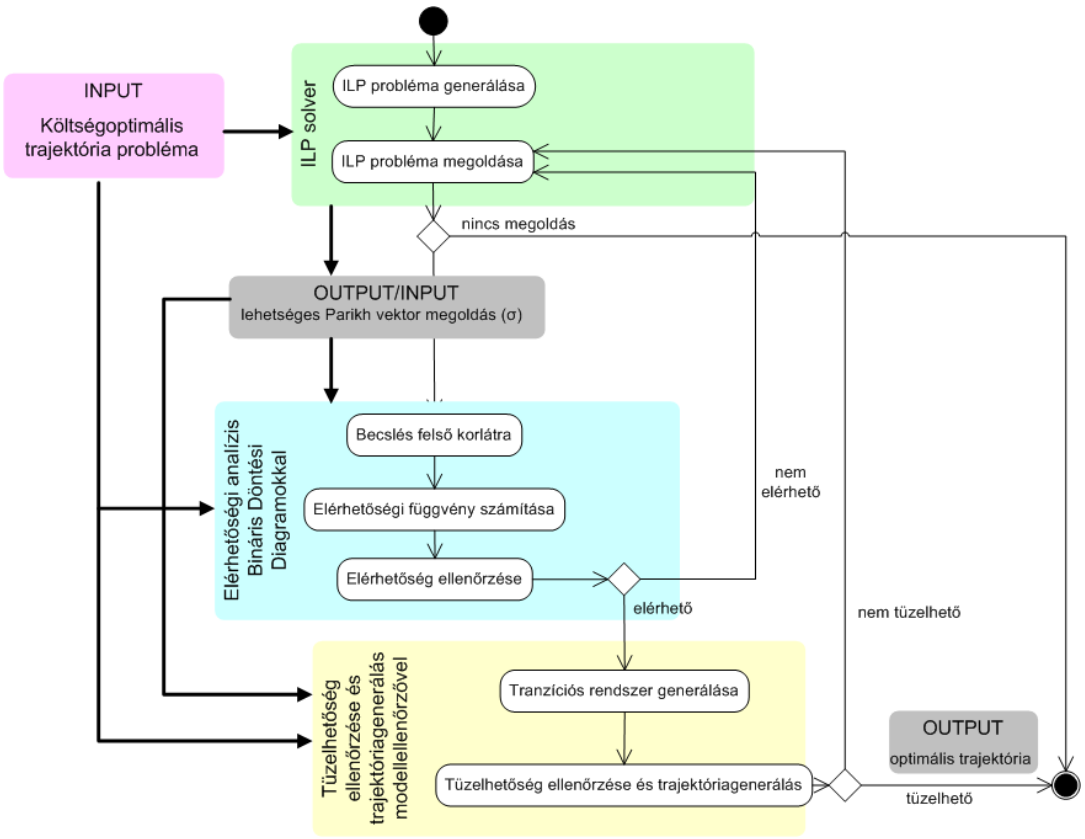
Mivel a PNS algoritmusok által generált megoldások nem feltétlenül tüzelhetőek a Petri hálóban, de alkalmazásuk csökkenti a keresési teret, a PNS algoritmusok használatával egy optimális tüzelési szám vektort, más néven Parikh vektort számolok ki, amelyhez tartozó trajektóriákat két külön fázisban vizsgálom meg. Mivel a Parikh vektorhoz nem biztos, hogy tartozik trajektória, a Parikh vektort lehetséges megoldásnak hívom.

A kezdőállapot és a lehetséges Parikh vektor megoldás alapján az állapotegyenlet segítségével kiszámolható az az állapot, amelyet egy, a lehetséges Parikh vektor megoldásnak megfelelő tüzelési szekvencia által érünk el (ha létezik ilyen). Az első fázisban ennek az állapotnak az elérhetőségét vizsgáljuk az úgynevezett elérhetőségi függvénnyel. Ez a fázis egy előszűrés, amely még mindig nem garantálja egy megfelelő trajektória létezését, hiszen egy konkrét célállapot esetében is több megoldás Parikh vektor létezhet. Ugyanakkor ennek a szűrési fázisnak az előnye, hogy az elérhetőségi függvény egy rákövetkező ellenőrzésben újra felhasználható.

Az elérhetőségi ellenőrzést követi a tüzelhetőségi ellenőrzés amelynek egyúttal az is feladata, hogy egy konkrét tüzelési szekvenciát is előállítson, amely egyben megoldása az optimális trajektória problémának. A 3. ábra ezt a folyamatot mutatja be.

Elérhetőség ellenőrzése. Az első lépésben a lehetséges Parikh vektor megoldás és az állapotegyenlet által számolt állapot elérhetőségét vizsgálom. Ehhez egy elérhetőségi függvényt állítok elő, amelybe behelyettesítve a kezdő–, illetve lehetséges Parikh vektor megoldás által meghatározott végállapotot, gyors választ kapunk arra, hogy ez a végállapot elérhető-e. Az elérhetőségi függvény generálása során a Petri háló, mint nemdeterminisztikus véges automata állapotátmeneti relációját építem fel bináris döntési diagramok (BDD) segítségével, amelyből az állapotátmeneti függvény tranzitív lezártját generálom. A tranzitív lezárt egy logikai formulával adott, amely bármely két állapotra, mint bemeneti paraméterekre igazra értékelődik ki, ha a Petri háló működésével konzisztens út létezik közöttük. Az elérhetőségi függvény előnye, hogy a későbbi vizsgálatok során is felhasználható.

Amennyiben a lehetséges Parikh vektor megoldás által meghatározott állapot nem el-



3. ábra. Költségoptimális trajektória keresésének menete

érhető a kezdőállapotból, a Parikh vektorhoz nem tartozik tüzelhető trajektória, így az ILP probléma következő legjobb megoldását számoljuk ki, majd ellenőrizzük ismét. Mivel egy állapot több trajektória illetve több Parikh vektor által is elérhető, a tüzelhetőséget még ellenőriznünk kell.

Tüzelhetőség ellenőrzése és a trajektória generálása. Kidolgoztam egy algoritmust a lehetséges Parikh vektor megoldáshoz tartozó a kezdő–, és végállapot közötti trajektória előállítására (amennyiben létezik) modellellenőrző eszközt használva. Ebbe a lépésbe integráltam a lineáris temporális logikával megfogalmazható verifikációs kritériumok ellenőrzését is.

Amennyiben a tüzelhetőségi ellenőrzés nem talál megfelelő trajektóriát, az ILP probléma következő legjobb Parikh vektor megoldását generáljuk, és azon végezzük el az ellenőrzéseket, amíg megoldást nem találunk.

2.1.3. Modellellenőrzés alapú optimális trajektóriagenerálás

A SPIN modellellenőrző eszközt felhasználva egy másik megoldást is adtam a költségoptimális trajektória generálására, amely az optimális trajektória problémát közvetlenül a modellellenőrző segítségével generálja. Ezen megközelítés előnye, hogy az optimalitási feltétel az ellenőrizendő lineáris temporális logikai kifejezésbe beágyazva dinamikusan változik a Petri háló állapotterének bejárása során, vagyis a mennyiségi kritériumot is a modellenőrzés során vesszük figyelembe. A megoldás hátránya, hogy a modellellenőrző bemenetként véges tranzíciós rendszert vár, ami nem feltétlenül teljesül egy általános Petri háló esetében. Ugyanakkor a tranzíciós rendszer méretére a költségkorlát alapján tudunk felső korlátot megadni.

2.1.4. Petri hálók minimális idejű trajektórájának generálása

A költségoptimális esethez hasonlóan szintén két megoldást adtam a minimális időtartamú trajektóriák keresésére. Az első megoldásban a maximális időtartamra egy felső korlátot számítok ki, amelyet, mint időhorizontot használok az ILP probléma felírásában. Az ILP probléma (i) a tokenáramlást az egyes helyeken időegységenként írja le ezen az időhorizonton belül, és (ii) a tranzíciók tüzelési feltételét is belekódolom az ILP problémába. Ezen megoldás révén az ILP probléma megoldása egyben tüzelhető megoldást határoz meg.

A második megoldás a költségoptimális esethez hasonlóan közvetlenül a modellellenőrzőt használja fel az optimális trajektória generálására a mennyiségi kritérium LTL kifejezésbe történő beágyazásával. A Petri háló tranzíciós rendszer méretére itt ugyanazt a felső korlátot alkalmazom, amelyet az előző megoldásban.

A kidolgozott módszer egy szintén rokon területen, gráftranszformációs rendszerek tervezésében is használható.

2.2. Gráftranszformációs rendszerek

Napjainkban egyre elterjedtebb a *gráftranszformációs (GT) rendszerek* használata komplex rendszerek tervezésében [22, 39]. A gráftranszformációs rendszerek előnye, hogy míg a Petri hálók (tranzíciók tüzelése által) inkább a rendszerek vezérlési struktúráját írják le, addig a GT rendszerek által leírt rendszerek többnyire adatvezéreltek, a transzformációs lépések során az adatstruktúra változik.

Mivel a gráftranszformációs rendszerek és a Petri hálók közötti kapcsolatot, illetve ezen két terület analízisének hasonlóságát, az analízis módszerek alkalmazhatóságát többen kutatták [32, 38, 39], a kutatás során a Petri hálók területén kidolgozott módszer adaptálhatóságát vizsgáltam a gráftranszformációs rendszerek modellellenőrzésére, optimalizálására.

Egy gráftranszformációs modellben a rendszer állapota egy gráffal írható le, mely állapotok közötti átmenetet a gráftranszformációs szabályokkal adunk meg, melyek a rendszer lehetséges működését írják le, a költség, illetve az időtartam paraméterek pedig a transzformációs lépésekhez vannak hozzárendelve. A kezdőállapot a kiindulási gráf, az

elérendő (részleges) végállapot pedig szintén egy gráffal adott. Az OT probléma GT rendszerek esetében így szintén egy elérhetőségi probléma, ahol a kezdő és a (részleges) elérendő állapot gráffal adott, és ezek között keressük az optimális utat, illetve optimális gráftranszformációs szabályszekvenciát.

A Petri hálókra kidolgozott optimális trajektória probléma megoldásának adaptációjánál egy már létező transzformációt vettem alapul [22]. A gráftranszformációs rendszert ezen transzformáció segítségével egy Petri hálóvá transzformálom, melyen a fent bemutatott módon elvégzem az optimalizálást. Mivel a gráftranszformáció Petri hálóba történő transzformációja nem biszimuláció, így a Petri hálón kapott megoldás végrehajthatósága a gráftranszformációs rendszerben nem garantált, azonban a gráftranszformációs rendszer állapotterének bejárásának irányításához felhasználható, így levágva a nem megfelelő zsákutcákat a mennyiségi és logikai korlátok alapján.

3. Tudományos eredmények

A Petri hálók optimális trajektória probléma megoldása általános Petri hálókra ad megoldást. A bemutatott megoldás az irodalomból ismert megoldásokhoz képest más abban, hogy

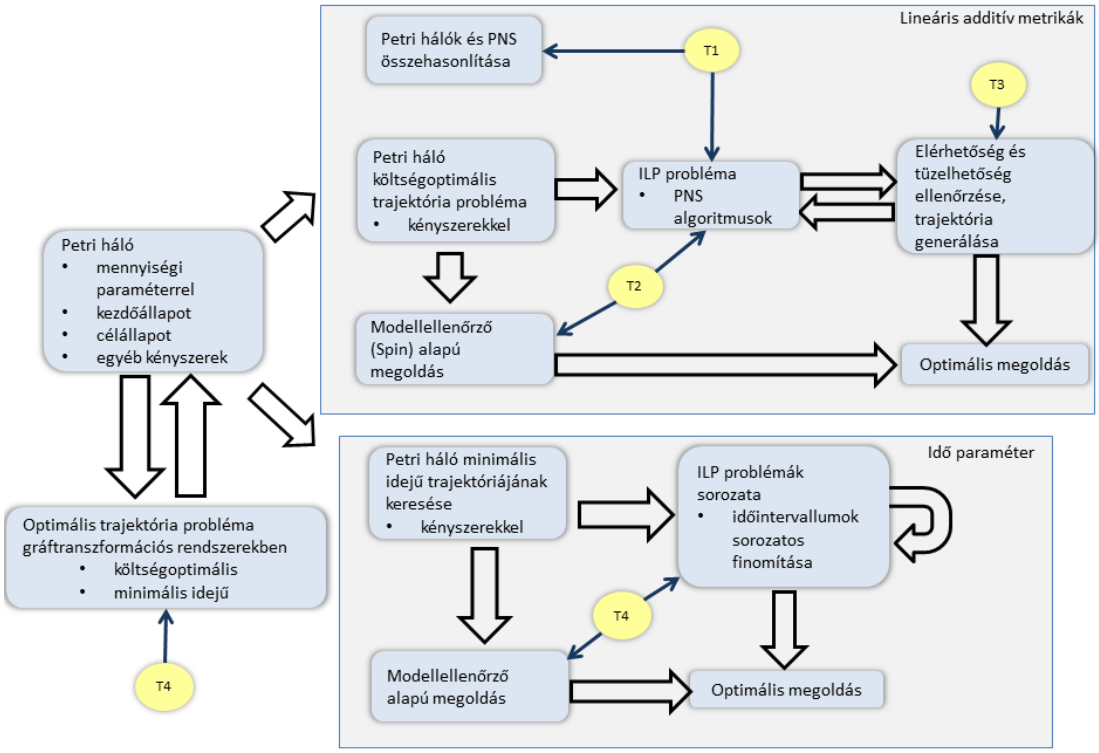
- nem csak Petri hálók egy (strukturális) alosztályára alkalmazható, hanem általános Petri hálókra is, valamint
- nem csak közelítő megoldást ad, hanem optimálisat.
- A bemutatott ellenőrzési technikákban használt elérhetőségi függvény és tranzíciós rendszer a rendszer megváltoztatása esetén is újrahasznosítható, valamint a keresés is inkrementálisan folytatható.

A Petri háló alapú optimális trajektória keresés gráftranszformációs rendszerekre is adaptálható a gráftranszformációs rendszer Petri hálóvá történő transzformációjával. A Petri hálóra adott megoldás által generált optimális trajektória visszavetítésével a gráftranszformációs rendszerben is meghatározható az optimális trajektória.

A 4. ábra az értekezésben ismertetett eredményeket mutatja be, ahol T1-T4 a téziseket jelölik.

3.1. Petri hálók és produkciós hálók szintézise

Tézis 1. *Megvizsgáltam a Petri hálók elérhetőségi probléma és a produkciós hálózatok szintézis (PNS) probléma közötti hasonlóságokat és különbözőségeket, amelyek alapján meghatároztam a PNS algoritmusok alkalmazásának feltételeit a Petri háló optimális trajektória problémára [4, 7–9].*



4. ábra. Tézisek áttekintése

1/1 Definiáltam egy strukturális megfeleltetést a Petri hálók és a P-gráf reprezentáció között. Definiáltam egy megfeleltetést, amely a Petri háló költségoptimális trajektória problémát egy PNS problémának felelteti meg.

1/2 A PNS területen egy lehetséges megoldási struktúrát meghatározó öt axióma mintájára a Petri hálók területén definiáltam a strukturálisan helyes Petri háló fogalmát.

1/3 Kidolgoztam egy algoritmust, amely a PNS probléma méretét csökkenti a Petri háló redukciós szabályainak adaptálásával. Megmutattam, hogy a redukciós szabályok ezen alkalmazása megtartja az elérhetőséget és a költséget a PNS problémában.

3.2. PNS algoritmusok a Petri háló költségoptimális trajektória probléma megoldásában

Tézis 2. *A PNS algoritmusokat adaptáltam a Petri háló költségoptimális trajektória probléma megoldásában a lehetséges Parikh vektor megoldás előállítására.*

2/1 A Petri háló költségoptimális probléma maximális struktúrájának generálására a PNS Maximal Structure Generation algoritmust adaptáltam. A strukturálisan helyes Petri hálók generálására pedig a PNS Solution Structure Generation algoritmust használtam. Az eredményeket a [4, 12, 13, 15–17] publikációkban mutattam be.

2/2 Kidolgoztam egy algoritmust amelynek bemenete a strukturálisan helyes Petri hálók halmaza, míg a kimenete egy lehetséges Parikh vektor megoldás. Kidolgoztam egy másik algoritmust is, amely a strukturálisan helyes Petri hálók minimális bázisa alapján generál egy lehetséges Parikh vektor megoldást. A kapcsolódó eredményeket a [4] publikációban mutattam be.

2/3 A PNS Accelerated Branch and Bound algoritmus Petri háló költségoptimális problémájának megoldására történő alkalmazásával kidolgoztam egy algoritmust, amelynek bemenete a maximális Petri háló struktúra, amelyből a strukturális követelmények (axiómák) folyamatos vizsgálata mellett származtatok egy lehetséges Parikh vektor megoldást, mint kimenetet. A kapcsolódó eredményeket a [4, 12, 13, 15–17] publikációkban mutatom be.

3.3. A lehetséges Parikh vektor megoldás elérhetőségi és tüzelhetőségi ellenőrzése

Tézis 3. *A lehetséges Parikh vektor megoldás ellenőrzésére szűrőlépéseket vezettem be, melyek a keresési tér méretét hivatottak csökkenteni olyan módon, hogy ezek a módszerek a későbbi vizsgálatok során újra felhasználhatóak legyenek. Mind az elérhetőségi függvény, mind a tüzelhetőségi ellenőrzés véges modellekkel dolgozik, melyek méretére felső korlátokat adtam az ILP probléma és az adott költségkorlát alapján. A tüzelhetőségi ellenőrzés során generálom az optimális trajektóriát, amennyiben az létezik.*

3/1 A Petri hálók elérhetőségi függvényét használva megoldást adtam a lehetséges Parikh vektor megoldás által adott célállapot elérhetőségének ellenőrzésére. Az elérhetőségi függvény generálása során a Petri háló, mint nemdeterminisztikus

véges automata állapotátmeneti relációját építem fel bináris döntési diagramok segítségével, amelyből az állapotátmeneti függvény tranzitív lezártját generálom. Ezzel egy gyors és újra felhasználható szűrést adtam a lehetséges Parikh vektor megoldások ellenőrzésére.

A kapcsolódó eredményeket a [4, 12–16] publikációkban mutattam be.

3/2 Kidolgoztam egy algoritmust a SPIN *modellellenőrzőt használva az optimális trajektória generálására, amely (1) megfelel a lehetséges Parikh vektor megoldásnak, (2) az elérhetőségi függvénynek és (3) a további (temporális logikai) kényszereknek.*

A kapcsolódó eredményeket a [4, 12–14, 16, 19]. publikációkban mutattam be.

3/3 Megoldást adtam korlátos Petri hálók költségoptimális trajektória problémájának közvetlenül a modellellenőrző segítségével történő megoldására. Ezen megoldásban az optimalitási feltétel az ellenőrizendő lineáris temporális logikai kifejezésbe beágyazva dinamikusan változik a Petri háló állapotterének bejárása során, vagyis a mennyiségi kritériumot is a modellenőrzés során vesszük figyelembe. A kapcsolódó eredményeket a [3, 4] publikációkban mutattam be. .

3.4. Minimális időtartamú trajektória probléma

Tézis 4. *Két megoldást dolgoztam ki a minimális időtartamú trajektória probléma megoldására, amelyek (1) a Petri háló működésének időegységekre lebontott állapotegyenlet alapú leírásán, valamint (2) a költségoptimális ILP probléma megoldásával számított felső időkorlát alkalmazásán alapul.*

4/1 Kidolgoztam egy algoritmust a minimális időtartamú trajektória probléma megoldására, amely az optimális megoldást egy ILP probléma megoldásaként állítja elő. Az ILP probléma (1) a tokenek számának változását az egyes helyeken és (2) a tranzíciók tüzelési feltételét időegységenként írom le az állapotegyenletet alkalmazva. Az ILP probléma méretére felső korlátot a költségoptimális trajektória ILP problémájából származtatok.

4/2 A SPIN *modellellenőrzőn alapuló megoldást adtam a minimális időtartamú trajektória problémára az optimalitási feltétel LTL kifejezésbe való integrálásával. A megoldás ugyanazt a felső korlátot használja, mint az előző megoldások a tranzíciós rendszer méretének korlátozására.*

A kapcsolódó eredményeket a [6] publikációban mutattam be.

3.5. Optimalizálás gráftranszformációs rendszerekben

Tézis 4. *Az optimális trajektória problémát kiterjesztettem költség és idő paraméterekkel kibővített gráftranszformációs rendszerekre.*

4/3 A gráftranszformációs rendszerekbe a költség paramétert mint a szabály alkalmazásának költségét vezettem be, valamint kidolgoztam ennek a formális szemantikáját is. A kapcsolódó eredményeket a [5, 21, 22] publikációkban mutattam be.

4/4 A gráftranszformációs rendszerekben az idő paramétert mint az egyes gráfelemek keletkezésének, illetve utolsó módosításának időbélyegét vezettem be. A gráftranszformációs rendszerekben az idő múlását a szabály alkalmazásának időtartamával modelleztem, valamint formális szemantikát adtam a szabályok alkalmazására. A kapcsolódó eredményeket a [2, 3, 10, 11] publikációkban mutattam be.

4/5 A Petri háló definíció általánosításaként definiáltam az optimális trajektória problémát gráftranszformációs rendszerekben. A kapcsolódó eredményeket a [6] publikációban mutattam be.

A gráftranszformációs rendszerek Petri hálóvá történő transzformációját szerzőtársam, Dr. Varró Dániel dolgozta ki [22]. Az ELTE informatika tanári szakán készült szakdolgozatom témája volt az idő bevezetése a gráftranszformációs rendszerekbe, melyet egy Erasmus ösztöndíj keretében a Paderboni Egyetemen dolgoztam ki Reiko Heckel professzor és később Magyarországon Dr. Varró Dániel témavezetők segítségével. A Petri háló optimális trajektória probléma megoldásának adaptálása a gráftranszformációs rendszerekre saját kontribúció.

4. Új tudományos eredmények alkalmazhatósága

Az optimális trajektória probléma modellezése magasszintű modellezési nyelvekben. Az irodalomban számos automatikus transzformáció létezik magasszintű modellezési nyelvekről (mint például UML vagy BPMN) Petri hálókra [44, 48] melyek segítségével a magasszintű modellezési nyelven definiált optimális trajektória probléma Petri háló optimális trajektóriává transzformálható, és az ott kapott optimális trajektória visszavetíthető az eredeti modellbe.

Maximum likelihood diagnosztika. A kidolgozott Petri háló megoldások a megbízhatósági modellezésben is hasznosíthatók. Egy példa erre a diagnosztikai problémák Petri

hálókkal leírt hibaterjedésének vizsgálata, ahol az egyes tranzíciókhoz a hiba bekövetkezésének a valószínűsége a mennyiségi paraméter [43]. Ebben az esetben az optimális trajektória egy olyan út a Petri hálóban, amelynek a bekövetkezése a legnagyobb valószínűségű.

BDD–vezérelt állapottér bejárás. A PNS algoritmusok alkalmazásakor az ABB algoritmus az állapottér bejárását az elérendő helyekből kiindulva vizsgálja. Az eljárásban használt hely-tranzícióhalmaz párok neutrális kiterjesztése hasonlóságot mutat a Petri háló bináris döntési diagrammjának felépítésével. Ez a hasonlóság felhasználható lehetne az ABB algoritmus elágaztató mechanizmusában ezzel is gyorsítva a keresést.

Kapcsolódó publikációk

Összes publikációk száma: 22
Ebből lektorált: 13
Ismert független idézetek száma: 107

Könyvfejezet és folyóiratcikk

[1] A. Pataricza, T. Bartha, G. Csertán, S. Gyapay, I. Majzik, and D. Varró, *Formális módszerek az informatikában*. Typotex, 2004. In Hungarian.

[2] S. Gyapay, D. Varró, and R. Heckel, „Graph transformation with time," *Fundam. Inform.*, vol. 58, no. 1, pp. 1–22, 2003.

[3] S. Gyapay, A. Schmidt, and D. Varró, „Joint optimization and reachability analysis in graph transformation systems with time," *Electronic Notes in Theoretical Computer Science*, vol. 109, pp. 137–147, 2004.

[4] S. Gyapay and A. Pataricza, „Optimal trajectory generation for Petri nets," *Acta Cybernetica*, vol. 17, pp. 225–245, January 2005.

[5] S. Varró-Gyapay and D. Varró, „Optimization in graph transformation systems using Petri net based techniques," *Electronic Communications of the EASST (ECEASST)*, vol. 2, 2006. Selected papers of Workshop on Petri Nets and Graph Transformations.

[6] S. Varró-Gyapay, „Optimization in graph transformation systems with time using Petri net based techniques," *Electronic Communications of the EASST*, vol. 51, pp. 1–12, 2012.

Konferenciakiadványban megjelent cikk

[7] S. Gyapay, „Operation research methods in Petri net–based models of IT systems," in *Mini–Symposium 2002*, (Budapest University of Technology and Economics, Department of Measurement and Information Systems), pp. 38–39, IEEE Hungary Section (BUTE Student Branch), February 4–5 2002.

[8] S. Gyapay, „Operation research methods in Petri net–based analysis of IT systems," in *The Third Conference of PhD Students in Computer Science* (T. Csendes, ed.), (Szeged, Hungary), p. 36, July 1–4 2002.

[9] S. Gyapay, A. Pataricza, J. Sziray, and F. Friedler, „Petri net-based optimization of production systems," in 6^{th} *IEEE International Conference on Intelligent Engineering Systems* (A. Lovrenčić and I. J. Rudas, eds.), pp. 465–469, Organized and published by Faculty of Organization end Informatics, University of Zagrev, Croatia, May 26–28 2002.

[10] S. Gyapay, R. Heckel, and D. Varró, „Graph transformation with time: Causality and logical clocks," in *Proc. ICGT 2002: 1st International Conference on Graph Transformation* (H.-J. Kreowski and P. Knirsch, eds.), LNCS, (Barcelona, Spain), pp. 120–134, Springer-Verlag, October 7–12 2002.

[11] S. Gyapay and R. Heckel, „Towards graph transformation with time," in *Proc. AGT 2002: Workshop on Applied Graph Transformation* (H.-J. Kreowski and P. Knirsch, eds.), (Grenoble, France), pp. 131–140, 2002.

[12] S. Gyapay, „Process Network Synthesis-based analysis of Petri net models," in *Mini–Symposium 2003*, (Budapest University of Technology and Economics, Department of Measurement and Information Systems), pp. 20–21, IEEE Hungary Section (BUTE Student Branch), February 4–5 2003.

[13] S. Gyapay and A. Pataricza, „A gradual filtering method for simultaneous verification and optimization of Petri nets," in *The John von Neumann PhD Conference*, (Budapest University of Technology and Economics, Faculty of Electrical Engineering and Informatics), pp. 27–30, October 2 2003.

[14] S. Gyapay and A. Pataricza, „A combination of Petri nets and Process Network Synthesis," in *2003 IEEE International Conference on Systems, Man & Cybernetics, Invited Sessions/Track on "Petri Nets and Discrete Event Systems"*, (Washington, D.C., USA), pp. 1167–1174, IEEE Press, October 5-8 2003.

[15] S. Gyapay and A. Pataricza, „Optimization methods for reachability analysis of Petri net models," in *Formal Methods for Railway Operation and Control Systems (Proceedings of Symposium FORMS-2003, Budapest, Hungary, May 15-16)* (G. Tarnai and E. Schneider, eds.), pp. 53–60, L' Harmattan, Budapest, May 17–23 2003.

[16] S. Gyapay and A. Pataricza, „Eine Kombination von Petrinetzen und Optimierun," in *Wissenschaftlichen Mitteilungen der 15. Frühlingsakademie, Balatonfüred, Hungary*, pp. 24–29, 2003.

[17] S. Gyapay, „Petri háló modellek folyamatszintézis–alapú analízise," in *Fiatal Mûszakiak Tudományos Ülésszaka*, (Cluj-Napoca, Romania), March 2003.

[18] S. Gyapay, „Solving the optimal trajectory problem using SPIN," in *Mini–Symposium 2004*, (Budapest University of Technology and Economics, Department of Measurement and Information Systems), pp. 18–19, IEEE Hungary Section (BUTE Student Branch), February 2004.

[19] S. Varró-Gyapay, A. Pataricza, and Á. Nagy, „Optimization of production nets under temporal constraints," in *Veszprém Optimization Conference: Advanced Algorithms*, (Veszprém, Hungary), December 13-15 2004.

[20] S. Gyapay, „Model–based optimization and verification of IT systems," in *Conference of PhD Students in Computer Science* (T. Csendes, ed.), (Szeged, Hungary), p. 52, July 1–4 2004.

[21] H. Ehrig, K. Ehrig, J. de Lara, G. Taentzer, D. Varró, and S. Varró-Gyapay, „Termination criteria for model transformation," in *Proc. FASE 2005: Internation Conference on Fundamental Approaches to Software Engineering* (M. Cerioli, ed.), vol. 3442 of *LNCS*, (Edinburgh, UK,), pp. 49–63, Springer, April 2005.

[22] D. Varró, S. Varró-Gyapay, H. Ehrig, U. Prange, and G. Taentzer, „Termination analysis of model transformations by Petri nets," in *Proc. Third International Conference on Graph Transformation (ICGT 2006)* (A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, eds.), vol. 4178 of *LNCS*, (Natal, Brazil), pp. 260–274, Springer, 2006.

Köszönetnyilvánítás

Hálás vagyok Pataricza András professzornak a témavezetéséért, mind a doktoranduszi, mind a későbbi évek során kapott kapott ötletekért, irányításáért, támogatásáért. Köszönettel tartozom a szakmai beszélgetésekért és ötletekért mind mostani kollégáimnak a Budapesti Műszaki és Gazdaságtudományi Egyetemen, mind korábbi kollégáimnak a Paderborni Egyetemről, a Berlini Műszaki Egyetemről, valamint az Eötvös Loránd Tudományegyetemről. Köszönettel tartozom Reiko Heckelnek, aki a Paderborni Egyetemen vállalta a szakdolgozatom témavezetését.

Különösen szerencsésnek mondhatom magamat a családom miatt, akiknek azt a hátteret köszönhetem, aminek segítségével eljutottam idáig. Köszönöm férjemnek, akivel a családunk mellett a szakmában is vannak közös ötleteink, hogy folyamatosan támogatott mind a családunk menedzselésében, mind a szakmai életben. Köszönöm szüleimnek, és férjem szüleinek a segítséget a gyermekeink felügyeletében, valamint a türelmet és megértést mind az ő, mind a gyermekeim részéről, amit a disszertáció írása alatt tanúsítottak. Köszönettel tartozom többi családtagomnak, barátaimnak és kórustársaimnak, hogy a munka hevében kikapcsolódásképpen általuk töltődhettem fel újra és újra.

A kutatásomat és későbbi munkámat az alábbi projektek segítségével végeztem: OTKA T038027, SEGRAVIS European Research Training Network, az Európai Unió DECOS (IST-511764, FP6) és SENSORIA (IST-3-016004, FP6) kutatási projektjei.

Bibliography

[23] Business Process Model and Notation. http://www.bpmn.org.

[24] OMG Systems Modeling Language. http://www.omgsysml.org.

[25] N. Aizenbud-Reshef, B. T. Nolan, and Y. S.-G. J. Rubin. Model traceability. *IBM Systems Journal*, 45:515–526, 2006.

[26] Andras Pataricza, Orsolya Doban, Akos Szoke. Costs/benefits of using formal methods. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 104–105, 2004.

[27] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[28] G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, Z. Ujhelyi, and D. Varró. Implementing efficient model validation in EMF tools: Tool demonstration. In *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, Lawrence, Kansas, USA, 11/2011 2011. IEEE Computer Society.

[29] A. Bondavalli, M. D. Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML-based system design. *Comput. Syst. Sci. Eng.*, 16(5):265 – 275, 2001.

[30] E. Brinksma and A. Mader. Verification and optimization of a PLC control schedule. In *SPIN Model Checking and Software Verification*, volume 1885 of *Lecture Notes in Computer Science*, pages 73–92, Berlin, Germany, 2000. Springer.

[31] C. F. Camerer. Behavioral game theory. experiments in strategic interaction. *The Journal of Socio-Economics*, 32(6):717–720, 2003.

[32] A. Corradini and U. Montanari. Specification of concurrent systems: from Petri nets to graph grammars. *Quality of Communication-Based Systems*, pages 35–52, 1995.

[33] M. Fowler and K. Scott. *UML Distilled. Applying the Standard Object Modeling Language*. Addison-Wesley, 1997.

[34] F. Friedler, K. Tarjan, Y. W. Huang, and L. Fan. Combinatorial algorithms for Process Synthesis. *Computers Chemical Engineering*, 16:313–320, 1992.

[35] F. Friedler, J. B. Varga, E. Feher, and L. T. Fan. Combinatorially accelerated Branch–and–Bound method for solving MIP model of Process Network Synthesis, nonconvex optimization and its applications. *State of the Arts in Global Opimization, Computational Methods and Applications*, pages 609–626, 1996.

[36] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[37] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, New York, NY, USA, 1984. ACM.

[38] H. G. Knehler. *Transformation von Graph-Grammatiken in Petri-Netze*. TU Berlin, FB Informatik, Bericht Nr. 88-02, September 1987. NewsletterInfo: 30.

[39] H.-J. Kreowski. A comparison between Petri-nets and graph grammars. In *Proceedings of the International Workshop on Graph theoretic Concepts in Computer Science*, WG '80, pages 306–317, London, UK, 1981. Springer-Verlag.

[40] S. P. Miller. Certification issues in model based development. Technical report, NASA, Rockwell Collins.

[41] S. P. Miller. Proving the shalls: Requirements, proofs, and model-based development. In *14th IEEE Int. Requirements Engineering Conference (RE'06)*, page 266, 2006.

[42] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., 2004.

[43] A. Pataricza. Model–based dependability analysis, DSc Thesis, 2008.

[44] A. Pataricza, A. Balogh, and L. Göczy. Verification and validation of nonfunctional aspects in enterprise modeling. *Enterprise Modeling and Computing with UML, Idea Group*, pages 261–303, 2006.

[45] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Darmstadt University of Technology, 1962.

[46] T. C. Ruys. Optimal scheduling using Branch-and-Bound with SPIN 4.0. In *Proc. 10th International SPIN Workshop*, volume 2648 of *LNCS*, pages 1–17, Portland, Oregon, USA, May 9–10 2003. Springer.

[47] W. L. Winston. *Operációkutatás, módszerek és alkalmazások*, volume 1. Aula Kiadó, 2003.

[48] M. Wirsing, M. Hölzl, L. Acciai, F. Banti, A. Clark, A. Fantechi, S. Gilmore, S. Gnesi, L. Gönczy, N. Koch, et al. Sensoria patterns: Augmenting service engineering with formal analysis, transformation and dynamicity. In *Leveraging Applications of Formal Methods, Verification and Validation*, pages 170–190. Springer, 2009.