



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

Valósídejű biztonságkritikus rendszerek k-indukció alapú verifikációja

TDK-DOLGOZAT

Készítette:

Tóth Tamás
MSc I. évfolyam

Konzulens:

dr. Majzik István
egyetemi docens

Vörös András
tudományos segédmunkatárs

A dolgozat bemutatását a TÁMOP - 4.2.2.B-10/1-2010-0009
Új tehetséggondozó programok és kutatások a Műegyetem tudományos műhelyeiben
című program támogatta.

2012. december 9.

Tartalomjegyzék

Kivonat	4
Abstract	5
Bevezetés	6
1. Rendszerek és követelmények formális modellezése	8
1.1. Tranzíciós rendszerek	8
1.2. Lineáris idejű tulajdonságok	10
1.3. Omega-reguláris tulajdonságok	12
1.4. Nemdeterminisztikus Büchi-automaták	13
1.5. Lineáris temporális logika	13
1.6. Biszimuláció	16
2. Időzített rendszerek modellezése időzített automatákkal	18
2.1. Időzített automaták szemantikája	21
2.2. Időzített automaták tulajdonságai	21
2.3. Időzített automaták véges absztrakciója	22
2.3.1. Óraekvivalencia	22
2.3.2. Régió tranzíciós rendszer	23
2.4. Időzített automaták korlátos modellellenőrzése	24
3. Omega-reguláris tulajdonságok modellellenőrzése	26
3.1. Invariáns tulajdonságok vizsgálata k-indukcióval	26
3.2. Omega-reguláris modellellenőrzés perzisztencia vizsgálatával	28
3.3. Perzisztencia vizsgálata invariáns tulajdonságok ellenőrzésével	29
3.3.1. Szükséges invariáns feltétel a rendszer perzisztenciájára	29
3.3.2. Elégséges invariáns feltétel a rendszer perzisztenciájára	31

4. Magasabb szintű formalizmus időzített rendszerek modellezéséhez	33
4.1. A naptáramomata szemantika	33
4.2. Naptáramomatak szemantikájának módosítása	34
4.3. Magasabb szintű formalizmus naptáramomatakhoz	36
4.3.1. Naptáramomatak kompozíciója	37
4.3.2. A magasabb szintű formalizmus szemantikája	38
4.3.3. Naptáramomatak tulajdonságai	39
4.4. Naptáramomatak biszimulációi	39
4.4.1. Biszimuláció időeltolással	39
4.4.2. Biszimuláció óraekvivalenciával	40
5. A k-indukciós modellellenőrzés módszere	41
5.1. A rendszer modellezése	41
5.1.1. A rendszerkomponensek modellezése	41
5.1.2. A rendszermodell jólformáltsági kritériumai	42
5.1.3. A követelmények modellezése	43
5.1.4. Leképezés tranzíciós rendszerre	44
5.2. A modell ellenőrzése	44
5.2.1. Invariánsok keresése a modellben	44
5.2.2. Ellenpéldák keresése	46
5.2.3. A rendszer helyességének bizonyítása k-indukcióval	47
6. A modellellenőrzés eszköztámogatása	48
6.1. Modellezés szakterület-specifikus nyelven	48
6.2. A modell statikus analízise gráfmintaillesztéssel	49
6.3. A modell leképezése kódgenerálással	51
6.3.1. Rövid betekintés a SAL nyelvbe	51
6.3.2. Kódgenerálás	52
6.4. A modell ellenőrzése	53
7. A ProSigma SCAN protokoll vizsgálata	54
7.1. A ProSigma rendszer bemutatása	54
7.1.1. A SCAN protokoll kapcsolatkezelése	54

7.2. A kapcsolatkezelés modellje	56
7.2.1. Kapcsolatkezelés a terepi oldalon	57
7.2.2. Kapcsolatkezelés az irányítási oldalon	58
7.2.3. Validáció	58
7.3. A kapcsolatkezelés modelljének ellenőrzése	59
7.3.1. A hibamentes rendszer vizsgálata	59
7.3.2. Invariánsok keresése	59
7.3.3. A rendszer helyességének bizonyítása absztrakció segítségével	60
7.3.4. A rendszer kiegészítése egyszerű hibamoddal	61
7.4. A rendszer javítása	62
7.5. Az eredmények általánosítása	64
Összefoglalás	65
Köszönetnyilvánítás	67
Irodalomjegyzék	69

Kivonat

Napjainkban egyre nagyobb szerepet játszanak a beágyazott számítógépes rendszerek, amelyeket elterjedten alkalmaznak biztonságkritikus környezetekben is, például vasúti vagy autópári szakterületen. Ilyen rendszerek hibamentes működése kiemelten fontos, hiszen egy rendszerszintű hibajelenség komoly katasztrófához vezethet. A tervezési hibák felderítése és javítása formális módszerek alkalmazásával lehetséges, ezekkel ugyanis nemcsak a hibák jelenléte mutatható ki, hanem a rendszer hibamentessége is matematikailag megalapozott módon bizonyítható.

A biztonságkritikus rendszerek gyakran időfüggő viselkedésűek, ráadásul valósidejű követelményeknek kell megfelelniük. Ezek a rendszerek sokszor nem modellezhetők diszkrét idejű rendszerként, azaz az ellenőrzés során folytonos, idő dimenziójú változókkal is számolni kell, ami végtelen nagyságú állapottérhez vezet. Az ilyen időzített rendszerekben a végtelen állapottér vizsgálata nem megoldhatatlan feladat: rendszerek biztonsági tulajdonságainak ellenőrzésére alkalmas algoritmus a k-indukció módszere, amely az indukció általánosítása által végtelen állapotterű rendszerekre is alkalmazható.

Céлом egy könnyen használható keretrendszer felépítése volt, amely modellezési és ellenőrzési támogatást nyújt időzített rendszerek verifikációjához. Munkám során az alábbi feladatokat oldottam meg:

- Definiáltam egy új modellezési formalizmust, amely segítségével időzített rendszerek működése magas szinten, a mérnöki gyakorlathoz közelálló módon leírható. A formalizmushoz olyan formális szemantikát rendeltem, amely lehetővé teszi a k-indukció alkalmazását a verifikáció során.
- Megadtam egy statikus analízis módszert, amely segítségével a magas szintű modelltől a rendszer bizonyos invariánsai kinyerhetők. A talált invariánsokkal a követelmények k-indukciós bizonyítása hatékonyan támogatható.
- A magas szintű modell megfelelő leképezésével lehetővé tettem, hogy a rendszernek ne csak biztonsági, hanem élőségi (azaz a működés során elvárt állapotok elérésére vonatkozó) tulajdonságai is vizsgálhatóak legyenek.
- A módszerhez tartozó eszközkészletet implementáltam.

A módszert sikeresen alkalmaztam egy ipari projektben, ezzel demonstrálom a módszer hatékonyságát. Az esettanulmány ismertetése során kitérek a vizsgált rendszerben a módszer segítségével talált hibára és annak kijavítására.

Abstract

Nowadays, embedded computer systems play an ever increasing role, even in safety critical domains like the railway control and automotive industry. The absence of errors in such systems is of especially great importance, since a system level failure may result in catastrophic consequences. Formal methods are appropriate means for detecting and correcting design flaws, as with their usage not only is it possible to reveal errors, it is also possible to prove the correctness of the design in a mathematically sound way.

The behaviour of safety critical systems is often time dependent, furthermore, they frequently need to satisfy real-time requirements. It is often not appropriate to model such systems in the discrete time domain. Therefore the designer must count with continuous variables of time dimension, which results in an infinite state space. The analysis of such timed systems is however not impossible: safety properties of systems can be checked using the k-induction approach, that is applicable on systems of infinite state space by the generalization of the induction principle.

My goal was to develop an easy-to-use framework that provides the means for modelling and verification of timed systems. During my work, I have accomplished the following objectives:

- I have defined a new modelling formalism with which timed systems can be modelled in a high level that fits to the engineering practice. I have assigned to the formalism a formal semantics that allows verification by k-induction.
- I have specified a static analysis method that finds invariants of the system based on the high level model. With the invariants found the k-induction based verification can be efficiently supported.
- With the suitable mapping of the high level model I have made possible the verification of liveness properties (that concern the expected states of the system), in addition to the verification of safety properties.
- I have implemented the toolset that supports these modelling and verification activities.

I have successfully applied the method in an industrial project, by that I demonstrate its efficiency. During the description of this case study I will detail the error found in the system under verification and the corresponding correction steps.

Bevezetés

Napjainkban az élet egyre több területén jelennek meg különböző informatikai rendszerek. Ezeknek a rendszereknek jelentős részét teszik ki a beágyazott rendszerek, melyeknek gyakran biztonságkritikus környezetben kell üzemelniük (például vasúti biztosítóberendezések komponensei). Ilyenkor egy-egy hibajelenség súlyos környezeti vagy anyagi kárral, rosszabb esetben akár emberéletet fenyegető következményekkel járhat, emiatt az ilyen rendszerek specifikációnak megfelelő működése létfontosságú. A specifikációnak megfelelő működés ellenőrzésére különböző verifikációs technikák szolgálnak. Ezek egyike a modellellenőrzés, mely során a rendszer egy matematikai modelljét vizsgálva lehet különböző formalizált követelmények teljesülését ellenőrizni.

Munkám során a Prolan Irányítástechnikai Zrt. ProSigma rendszerének jelenleg fejlesztés alatt álló kommunikációs protokollját, a SCAN protokollt vizsgáltam formális módszerekkel. A ProSigma feladata a vasúti biztosítóberendezés részeként a tévesztésmentes jelátvitel biztosítása, ezt a legmagasabb, SIL4-es biztonságintegritási szinten kell megvalósítania. Vizsgálataimat a protokoll egy szűk, ám mindenképpen fajsúlyos részletére, a kapcsolatkezelésre koncentráltam.

A protokollra a biztonságkritikus környezet kívánalmainak megfelelően valósidejű követelmények is vonatkoznak, így a modellalkotás során az idő modellezését meg kellett oldani. A rendszer modellezéséhez egy létező formalizmusból indultam ki, melyet a hatékony ellenőrzés érdekében módosítottam. A formalizmus egy megszorítása fölé magasabb szintű formalizmust definiáltam, mellyel hatékonyan, a mérnöki szemléletnek megfelelő módon lehet a formalizmusnak megfelelő rendszereket definiálni. A magasabb szintű formalizmust úgy adtam meg, hogy a jól ismert időzített automata formalizmus több jellemzőjét is megtartsa. A formalizmus segítségével a rendszert valósághűen, folytonos időben tudtam modellezni.

A rendszer egy véges absztrakciója hiányában egy ilyen, folytonos idejű rendszer állapottere kontinuum számosságú lesz, ami lehetetlenné teszi az állapottér explicit bejárását. Ugyanakkor a végtelen állapottér fölötti érvelés sem lehetetlen feladat, a k-indukciós modellellenőrző algoritmussal kísérletet lehet tenni az egyes tulajdonságok igazolására vagy cáfolására. Mivel az algoritmus alapvetően elérhetőséget (így közvetve biztonságosságot) vizsgál, a protokoll analízise szempontjából igazán érdekes élősegi tulajdonságok ellenőrzésére (például kiépül-e a kapcsolat) külön megoldást adtam. Ehhez két, a modell kiegészítésén alapuló, egymást jól kiegészítő módszert alkalmaztam, egyet az ellenpéldák keresésére, egyet pedig a tulajdonság bizonyítására.

Az elméletben kidolgozott ellenőrzési folyamathoz a lépések kivitelezését támogató esz-

közöket megterveztem és implementáltam. A magas szintű modellezési formalizmusához EMF alapon szakterület-specifikus nyelvet hoztam létre. A nyelvhez EMF IncQuery technológiával olyan gráfmintákat definiáltam, melyek elvégzik a modell különböző tervezési szempontok szerinti validálását, illetve statikus analízist végezve a k-indukciós bizonyítás támogatására feltérképezik a rendszer néhány invariáns tulajdonságát. Ezen felül XTend2 nyelven készítettem egy kódgenerátort, amely a magas szinten megfogalmazott példánymodellekből előállítja az ellenőrzésre alkalmas, a megfelelően leképezett rendszereket leíró kódot. Az ellenőrzéshez a SAL keretrendszert használtam, egyrészt a SAL nyelv kifejezőereje, másrészt az eszközzel sikeresen verifikált időzített rendszerek és a témában megjelent, az eszközökhöz kapcsolódó hasznos cikkek miatt.

Az eszközök felhasználásával elkészítettem és validáltam a kapcsolatkezelés modellét, valamint kigeneráltam az ellenőrizhető kódot. A modellellenőrzés során feltártam a kapcsolatkezelés hibás működését, melynek okát a generált ellenpélda segítségével azonosítottam. A hibát kijavítottam a rendszerben, majd ismételt ellenőrzéssel igazoltam a módosított rendszer helyességét. A verifikáció eredményei alapján a protokoll módosítására javaslatot tettem, ezt a Prolan Zrt. munkatársai jóváhagyták és bevezették a specifikációba.

A fent leírt folyamatot és a kapcsolódó elméleti ismereteket a dolgozat az alábbi tartalmi felosztásban tárgyalja.

- Az első fejezetben bemutatom a rendszerek és követelményeik formális modellezéséhez szükséges elméleti hátteret.
- A második fejezet bemutatja az időzített automata formalizmusához kapcsolódó alapvető modellezési ismereteket.
- A harmadik fejezet néhány olyan modellellenőrzési eljárást ismertet, melyeket a munkám során használtam.
- A negyedik fejezetben bemutatom a formális modellekhez kapcsolódó eredményeimet: a naptárautomaták alapján általam fejlesztett új formalizmust és a hozzá kapcsolódó matematikai eszközkészletet, amelyek segítségével hatékony k-indukció alapú verifikáció valósítható meg.
- Az ötödik fejezetben bemutatom az általam kidolgozott, a korábbi fejezetekben bemutatott elméleti eredményeimet használó modellellenőrzési módszert.
- A hatodik fejezetben bemutatom azokat a szoftver eszközöket, amelyeket a modellellenőrzési folyamat támogatására és hatékony megvalósítása érdekében fejlesztettem.
- A hetedik fejezetben ismertetem a ProSigma rendszer SCAN protokoll kapcsolatkezelésének a módszer és eszközkészlet segítségével történő modellezését és verifikációját, továbbá a rendszer analízise során elért eredményeimet.

1. fejezet

Rendszerek és követelmények formális modellezése

Ahhoz, hogy egy rendszer működésének helyessége igazolható legyen, mind a rendszer, mind a követelmények formális leírásának rendelkezésre kell állnia. Az alábbi fejezet olyan matematikai módszereket mutat be tömör tárgyalásban, melyek segítségével (alacsony szinten) specifikálható egy rendszer viselkedése, valamint az azzal szemben támasztott elvárások. Az itt bemutatott formalizmusok képezik a később bemutatott, időzített rendszerek magas szintű modellezésére és ellenőrzésére alkalmas formális módszerek alapjait.

1.1. Tranzíciós rendszerek

Informatikai rendszerek modellezésének egyik elterjedt módja a tranzíciós rendszer formalizmus használata. A formalizmus kiválóan alkalmas állapotalapú, eseményvezérelt működésű rendszerek modellezésére.

Szintaxis A tranzíciós rendszer formálisan egy $TS = (S, Act, \longrightarrow, I, AP, L)$ hatos, ahol

- S az állapotok halmaza,
- Act az akciók halmaza,
- $\longrightarrow \subseteq S \times Act \times S$ az úgynevezett tranzíciós reláció,
- $I \subseteq S$ a kezdőállapotok halmaza,
- AP az atomi propozíciók halmaza,
- $L : S \rightarrow 2^{AP}$ pedig a címkéző függvény.

TS véges, ha S , Act és AP véges.

Kényelmi okokból a $(s, \alpha, s') \in \longrightarrow$ kifejezés helyett gyakran a szemléletesebb $s \xrightarrow{\alpha} s'$ jelölést alkalmazzuk.

Egy s állapot terminális, ha nincsen rákövetkező állapota, azaz $\nexists s' \in S, \alpha \in Act$, hogy $s \xrightarrow{\alpha} s'$.

Egy tranzíciós rendszer működése informálisan a következő. A rendszer kezdetben valamely kezdőállapotban tartózkodik. A rendszer állapota a tranzíciós reláció mentén változik, vagyis ha a rendszer állapota s , akkor a rendszer következő állapotát valamely $s \xrightarrow{\alpha} s'$ átmenete fogja nemdeterminisztikusan meghatározni. Ekkor a rendszer s állapotból s' állapotba megy át.

Egy s állapotban a rendszer az L címkéző függvény által meghatározott $L(s)$ atomi propozíciókkal jellemezhető. Formálisan, az s állapot kielégít egy Φ propozicionális logikai formulát ($s \models \Phi$) akkor és csak akkor, ha Φ az s állapot $L(s)$ címkéje által meghatározott kiértékelése igaz, azaz $L(s) \models \Phi$.

A tranzíciókhoz rendelt akciók a rendszer interakciói és a rajtuk értelmezett követelmények modellezésére alkalmasak. Ebben a dolgozatban a rendszer állapot alapú viselkedését vizsgáljuk, így a formalizmus ezen részlete számunkra csak a szemléletesebb modellezés szempontjából bír jelentőséggel.

Végrehajtás Jelölje a továbbiakban $Post(s, \alpha)$ az s állapot α -rákövetkező állapotait, azaz

$$Post(s, \alpha) = \{s' \in S : s \xrightarrow{\alpha} s'\}$$

Egy TS rendszer véges végrehajtásrészlete állapotok és akciók egy $\hat{\rho} = s_0 \alpha_0 s_1 \alpha_1 \dots \alpha_{n-1} s_n$ alternáló sorozata, ahol $s_i \in Post(s_{i-1}, \alpha_{i-1})$ minden $0 < i \leq n$ indexre. Jelölés:

$$s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} s_n$$

Hasonlóképpen, egy végtelen végrehajtásrészlet állapotok és akciók egy $\rho = s_0 \alpha_0 s_1 \alpha_1 s_2 \alpha_2 \dots$ alternáló sorozata. Jelölésben:

$$s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} \dots$$

Egy végrehajtásrészlet maximális, ha nem hosszabbítható meg, vagyis ha terminális állapotban végződő véges útrészlet, vagy végtelen útrészlet.

Egy végrehajtásrészlet kiinduló, ha egy $s_0 \in I$ kezdőállapotból indul.

A rendszer egy végrehajtása egy kiinduló, maximális végrehajtásrészlet.

Egy $s \in S$ állapot elérhető, ha létezik egy $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} s_n = s$ véges, kiinduló ($s_0 \in I$) útrészlet. Az elérhető állapotok halmazát $Reach(TS)$ jelöli.

Út A végrehajtásokkal teljesen analóg módon definiálhatjuk a rendszer útjait. Jelölje $Post(s)$ az s állapot rákövetkező állapotait, azaz

$$Post(s) = \{s' \in S : s \xrightarrow{\alpha} s', \exists \alpha \in Act\}$$

A TS tranzíciós rendszer egy $\hat{\pi}$ véges útrészlete egy $s_0, s_1, \dots, s_n$ állapotsorozat, ahol minden $0 < i \leq n$ esetén $s_i \in Post(s_{i-1})$. Egy végtelen π útrészlet egy $s_0, s_1, \dots$ végtelen állapotsorozat, ahol minden $i > 0$ esetén $s_i \in Post(s_{i-1})$.

A rendszer egy útrészlete maximális, ha terminális állapotban végződő véges útrészlet, illetve ha végtelen. Az s állapotból kiinduló maximális útrészleteket $Paths(s)$ jelöli.

Egy útrészletet kiindulónak nevezünk, ha kezdőállapotból indul, vagyis $s_0 \in I$.

A rendszer egy útja egy kiinduló, maximális útrészlet. A TS tranzíciós rendszer útjainak halmazát $Paths(TS)$ kifejezéssel jelöljük. Formálisan

$$Paths(TS) = \bigcup_{s \in I} Paths(s)$$

A továbbiakban olyan tranzíciós rendszerekkel foglalkozunk, melyek nem rendelkeznek terminális állapotokkal, így minden maximális útrészletük végtelen. Ez a megkötés lényegében nem korlátozza a vizsgálódás érvényességi körét, hiszen a rendszer egy s terminális állapota kiküszöbölhető egy $s \xrightarrow{\alpha} s$ tranzíció felvételével.

Nyom Egy $\pi = s_0, s_1, \dots$ végtelen útrészlet nyoma a $trace(\pi) = L(s_0), L(s_1), \dots$ sorozat. Egy $\hat{\pi} = s_0, s_1, \dots, s_n$ véges útrészlet nyoma a $trace(\hat{\pi}) = L(s_0), L(s_1), \dots, L(s_n)$ sorozat.

Az útrészletek Π halmaza felett értelmezett a $trace(\Pi) = \{trace(\pi) : \pi \in \Pi\}$ leképezés. Az s állapotból kiinduló útrészletek nyoma $Traces(s) = trace(Paths(s))$.

Jelölje $Traces(TS)$ a rendszer útjainak nyomát, vagyis

$$Traces(TS) = \bigcup_{s \in I} Traces(s)$$

Egy tranzíciós rendszer nyoma tehát egy olyan nyelv, mely a 2^{AP} ábécé felett értelmezett végtelen hosszú szavakat tartalmaz. Azaz $Traces(TS) \subseteq (2^{AP})^\omega$.

1.2. Lineáris idejű tulajdonságok

A következőkben a modelleken vizsgálható tulajdonságokra térünk át.

Lineáris idejű tulajdonságnak nevezzük azokat a megkötéseket, melyek a rendszer megengedett lefutásait definiálják. Formálisan egy AP felett értelmezett P lineáris idejű tulajdonság az AP részhalmazaiából álló végtelen sorozatok egy részhalmaza, tehát $P \subseteq (2^{AP})^\omega$.

Lineáris idejű tulajdonság kielégítése Azt, hogy egy adott rendszer eleget tesz egy lineáris idejű tulajdonságnak, a következőképpen fogalmazhatjuk meg. Legyen P egy lineáris idejű tulajdonság AP felett, valamint $TS = (S, Act, \longrightarrow, I, AP, L)$ pedig egy terminális állapotoktól mentes tranzíciós rendszer. Ekkor TS kielégíti a P tulajdonságot ($TS \models P$) akkor és csak akkor, ha $Traces(TS) \subseteq P$. Hasonlóan, egy s állapot kielégíti a P tulajdonságot ($s \models P$) akkor és csak akkor, ha $Traces(s) \subseteq P$.

Biztonságossági tulajdonságok Egy P_{safe} AP feletti lineáris idejű tulajdonságot biztonságossági tulajdonságnak nevezünk, ha tetszőleges, a tulajdonságot sértő $\sigma \in (2^{AP})^\omega \setminus P_{safe}$ szóhoz létezik egy $\hat{\sigma}$ véges prefixum, amely nem előtagja egyetlen P_{safe} nyelven belüli szónak sem.

Az ilyen tulajdonságok tehát véges hosszú szavak felismerésével ellenőrizhetők. Egy biztonságossági követelmény szemléletes jelentése, hogy a rendszer futása során semmi rossz nem következik be.

Invariáns tulajdonságok A legegyszerűbb biztonságossági tulajdonságok az úgynevezett invariáns tulajdonságok, amelyek a rendszer elérhető állapotain fogalmaznak meg megköveteléseket. Formálisan egy AP fölötti P_{inv} lineáris idejű tulajdonság invariáns, ha létezik egy olyan Φ proposícionális logikai formula AP felett, melyre

$$P_{inv} = \left\{ A_0 A_1 A_2 \dots \in (2^{AP})^\omega : A_i \models \Phi, \forall i \geq 0 \right\}$$

vagyis a nyelv összes szavára jellemző, hogy szimbólumai kielégítik a Φ invariáns feltételt.

Könnyű belátni, hogy minden invariáns tulajdonság valóban biztonságossági tulajdonság is, hiszen minden olyan szó, ahol sérül az invariáns feltétel, jellemezhető a feltételsértésig tartó véges előtaggal.

Élőségi tulajdonságok Élőségi tulajdonságnak egy olyan P_{live} AP feletti lineáris idejű tulajdonságot nevezünk, melyre igaz, hogy tetszőleges $w \in (2^{AP})^*$ véges szó kiegészíthető egy $\sigma \in (2^{AP})^\omega$ végtelen hosszú utótaggal, hogy $w\sigma \in P_{live}$.

Egy rendszerre megfogalmazott élőségi jellemző jelentése, hogy a futás során előbb-utóbb bekövetkezik valamilyen kívánatos esemény.

Lineáris idejű tulajdonságok osztályozása Az egyetlen olyan nyelv, mely biztonságossági és élőségi tulajdonság is egyben, a $(2^{AP})^\omega$. Ezen felül tetszőleges P lineáris idejű tulajdonságra igaz, hogy előáll egy biztonságossági és egy élőségi tulajdonság uniójaként, azaz $P = P_{safe} \cup P_{live}$ [2].

A továbbiakban a lineáris idejű tulajdonságok egy speciális osztályával, az ω -reguláris tulajdonságokkal foglalkozunk. A modellellenőrzés szempontjából az ilyen tulajdonságok

jelentősége, hogy segítségével a gyakorlatban felmerülő követelmények széles köre megfogalmazható, ugyanakkor ábrázolásuk egyszerű.

1.3. Omega-reguláris tulajdonságok

Lineáris idejű tulajdonságok jellegzetes osztályát alkotják az ω -reguláris tulajdonságok. A definíciókat az alábbi pont tartalmazza.

Omega-nyelv Jelölje egy tetszőleges $\mathcal{L} \subseteq \Sigma^*$ nyelv szavaiból konkatenációval képzett végtelen hosszú láncokat $\mathcal{L}^\omega$, azaz

$$\mathcal{L}^\omega = \{w_1 w_2 \dots : w_i \in \mathcal{L}, \forall i \geq 1\}$$

Amennyiben $\mathcal{L}$ nem tartalmazza az ε üres szót, az így képzett nyelv végtelen hosszú szavakat fog tartalmazni. Az ilyen $\mathcal{L}^\omega \subseteq \Sigma^\omega$ nyelveket ω -nyelveknek nevezzük.

Az ω -nyelvek legismertebb típusa az ω -reguláris nyelvek osztálya. A ω -reguláris nyelvek a reguláris nyelveket általánosítják végtelen hosszú szavakra.

Omega-reguláris nyelv Egy $\mathcal{L} \in \Sigma^\omega$ ω -nyelv ω -reguláris, ha az alábbi műveletek véges sokszori alkalmazásával képezhető:

- $\mathcal{L}_1^\omega$, ahol $\varepsilon \notin \mathcal{L}_1$ reguláris,
- $\mathcal{L}_1 \mathcal{L}_2 = \{w\sigma : w \in \mathcal{L}_1, \sigma \in \mathcal{L}_2\}$, ahol $\mathcal{L}_1$ reguláris, $\mathcal{L}_2$ ω -reguláris,
- $\mathcal{L}_1 \cup \mathcal{L}_2$, ahol $\mathcal{L}_1$ és $\mathcal{L}_2$ ω -reguláris.

Omega-reguláris tulajdonság A fenti definíciókat felhasználva P lineáris idejű tulajdonság ω -reguláris tulajdonság, ha P ω -reguláris nyelv 2^{AP} felett.

Perzisztencia tulajdonságok Speciális ω -reguláris tulajdonságok az úgynevezett perzisztencia tulajdonságok. Egy ilyen tulajdonság szemléletes jelentése az, hogy a rendszer végrehajtása során idővel minden állapot kielégít valamilyen feltételt.

Egy AP halmaz feletti P_{pers} perzisztencia tulajdonság egy $P_{pers} \subseteq (2^{AP})^\omega$ élőségi tulajdonság, melyhez adható olyan Φ propozicionális logikai formula, melyre

$$P_{pers} = \left\{ A_0 A_1 A_2 \dots \in (2^{AP})^\omega : A_j \models \Phi, \exists i \geq 0, \forall j \geq i \right\}$$

1.4. Nemdeterminisztikus Büchi-automaták

Az olyan automatákat, melyek ω -nyelveket ismernek fel, ω -automatának nevezzük. Mivel ezek az automaták végtelen hosszú szavakon futnak, végtelen lefutásokra vonatkozó elfogadási feltételekkel kell rendelkezniük. A legegyszerűbb ω -automata a nemdeterminisztikus Büchi-automata [7], melyet a következő pont mutat be.

Nemdeterminisztikus Büchi-automata Egy $\mathcal{A}$ nemdeterminisztikus Büchi-automata egy $(Q, \Sigma, \delta, Q_0, F)$ ötös, ahol

- Q az állapotok véges halmaza,
- Σ egy véges ábécé,
- $\delta : Q \times \Sigma \rightarrow 2^Q$ az átmeneti függvény,
- $Q_0 \subseteq Q$ a kezdőállapotok halmaza,
- $F \subseteq Q$ pedig az elfogadó állapotok halmaza.

Az egyszerűség kedvéért a $p \in \delta(q, A)$ tranzícióra bevezetjük a $q \xrightarrow{A} p$ jelölést.

Az automatát nemblokkolónak nevezzük, ha $\delta(q, A) \neq \emptyset$ bármely $q \in Q$ és $A \in \Sigma$ esetén. Egy automata könnyen nemblokkolóvá tehető egy q_{trap} nemelfogadó csapdaállapot felvételével, melybe az összes nem definiált átmenet belevisz, és minden betűhöz önhurkot tartalmaz.

Büchi-automaták által elfogadott nyelv Az automata egy futása a $\sigma = A_0A_1A_2 \dots$ szón az állapotok egy végtelen $q_0q_1q_2 \dots$ sorozata, ahol $q_0 \in Q_0$ és $q_i \xrightarrow{A_i} q_{i+1}$ minden $i \geq 0$ esetén. A $q_0q_1q_2 \dots$ futás elfogadó, ha $q_i \in F$ végtelen sok $i \in \mathbb{N}$ esetén. Az $\mathcal{A}$ által elfogadott $\mathcal{L}_\omega(\mathcal{A})$ nyelv azon $\sigma \in \Sigma^\omega$ szavak halmaza, melyekhez létezik elfogadó futás.

Belátható, hogy tetszőleges ω -reguláris nyelvhez létezik azt elfogadó nemdeterminisztikus Büchi-automata. Ezen felül igaz az is, hogy egy nemdeterminisztikus Büchi-automata által elfogadott nyelv ω -reguláris, így megállapítható, hogy a nemdeterminisztikus Büchi-automaták által elfogadott nyelvek osztálya megegyezik az ω -reguláris nyelvek osztályával. [15]

1.5. Lineáris temporális logika

ω -reguláris tulajdonságok nemcsak Büchi-automatákkal írhatók le. A következő pontban bemutatásra kerül a modellellenőrzés gyakorlatában igen elterjedt LTL nyelv [18], mely segítségével viszonylag szemléletesen fogalmazhatóak meg ω -reguláris tulajdonságok.

Szintaxis Atomi propozíciók egy AP halmaza feletti (propozicionális) lineáris temporális logikai (LTL) kifejezésnek nevezünk egy, az alábbi absztrakt szintaxis alapján megkonstruált kifejezést.

Jelöljön φ egy LTL-kifejezést, továbbá legyen $a \in AP$ atomi propozíció. Ezek alapján egy LTL-kifejezés az alábbi nyelvtani szabályok szerint generálható:

$$\varphi ::= \text{true} \mid a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi \mathbf{U} \varphi$$

A fentiek közül speciális LTL-operátorok a $\bigcirc$ (next, máskor **X**) és az $\mathbf{U}$ (until, **U**), míg a többi operátor a propozicionális logikából ismert. Az alábbi definíciók segítségével vezethetők be további hagyományos Boole-logikai operátorok:

$$\begin{aligned} \text{false} &\stackrel{\text{def}}{=} \neg\text{true} \\ \varphi \vee \psi &\stackrel{\text{def}}{=} \neg(\neg\varphi \wedge \neg\psi) \\ \varphi \Rightarrow \psi &\stackrel{\text{def}}{=} \neg\varphi \vee \psi \\ \varphi \Leftrightarrow \psi &\stackrel{\text{def}}{=} (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi) \end{aligned}$$

További temporális operátorok a $\Diamond$ (finally, **F**) és $\Box$ (globally, **G**):

$$\begin{aligned} \Diamond\varphi &\stackrel{\text{def}}{=} \text{true} \mathbf{U} \varphi \\ \Box\varphi &\stackrel{\text{def}}{=} \neg\Diamond\neg\varphi \end{aligned}$$

Szemantika Legyen $\sigma = A_0A_1A_2\ldots \in (2^{AP})^\omega$ egy végtelen hosszúságú szó. Jelölje továbbá $\sigma[j\ldots] = A_jA_{j+1}A_{j+2}\ldots$ a szó j indextől kezdődő utótagját.

Ekkor a $\models \subseteq (2^{AP})^\omega \times \text{LTL}$ kielégítési reláció a legszűkebb olyan reláció, mely az alábbi szabályoknak eleget tesz:

$$\begin{aligned} \sigma &\models \text{true} \\ \sigma &\models \neg\varphi \Leftrightarrow \neg(\sigma \models \varphi) \\ \sigma &\models a \Leftrightarrow a \in A_0 \\ \sigma &\models \bigcirc\varphi \Leftrightarrow \sigma[1\ldots] \models \varphi \\ \sigma &\models \varphi \wedge \psi \Leftrightarrow \sigma \models \varphi \wedge \sigma \models \psi \\ \sigma &\models \varphi \mathbf{U} \psi \Leftrightarrow \sigma[j\ldots] \models \psi \wedge \sigma[i\ldots] \models \varphi, \exists j \geq 0, \forall 0 \leq i < j \end{aligned}$$

A $\Diamond$ operátor szemantikája a $\mathbf{U}$ operátor szemantikájából közvetlenül adódik:

$$\begin{aligned} \sigma &\models \Diamond\varphi \\ \sigma &\models \text{true} \mathbf{U} \varphi \\ \sigma[j\ldots] &\models \varphi \wedge \sigma[i\ldots] \models \text{true}, \exists j \geq 0, \forall 0 \leq i < j \\ \sigma[j\ldots] &\models \varphi, \exists j \geq 0 \end{aligned}$$

Ebből a $\Box$ operátor szemantikája a következőképpen vezethető le:

$$\sigma \models \Box \varphi$$

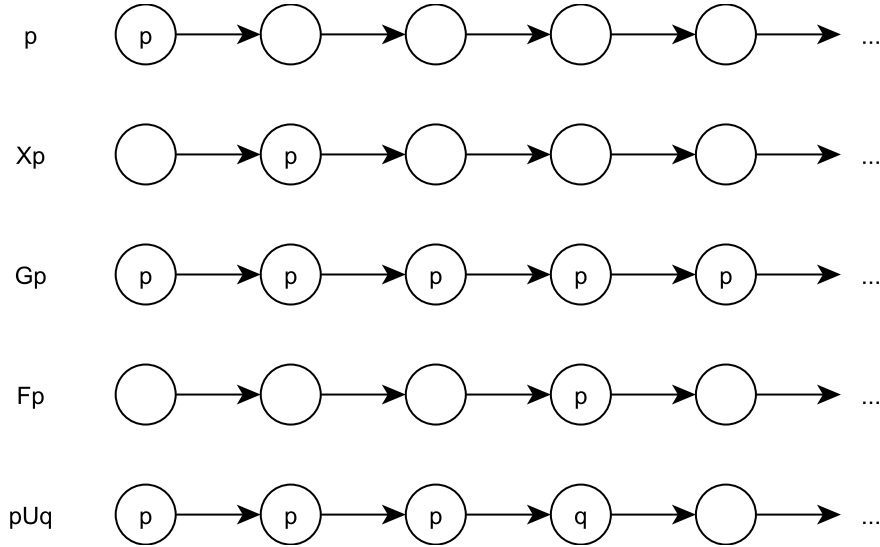
$$\sigma \models \neg \Diamond \neg \varphi$$

$$\sigma[j \dots] \models \neg \varphi, \neg \exists j \geq 0$$

$$\neg(\sigma[j \dots] \models \varphi), \neg \exists j \geq 0$$

$$\sigma[j \dots] \models \varphi, \forall j \geq 0$$

Az egyes LTL-operátorok jelentését szemléletesen foglalja össze a 1.1. ábra.



1.1. ábra. LTL-kijelentések

Ez alapján már definiálhatjuk azt a nyelvet, melyet egy LTL-formula meghatároz.

LTL kifejezések és tranzíciós rendszerek Azokat a szavakat, melyek egy AP feletti φ LTL-kifejezést kielégítenek, a $Words(\varphi) = \{\sigma \in (2^{AP})^\omega : \sigma \models \varphi\}$ nyelv tartalmazza. Mivel $Words(\varphi)$ egy AP feletti lineáris idejű tulajdonság, az ott látott definíciók segítségével könnyen megteremthető a kapcsolat egy LTL-formula és egy tranzíciós rendszer között.

Legyen tehát $TS = (S, Act, \longrightarrow, I, AP, L)$ egy terminális állapot nélküli tranzíciós rendszer, φ pedig egy LTL-kifejezés AP felett. Ekkor TS kielégíti a φ LTL-kifejezést ($TS \models \varphi$) akkor és csak akkor, ha $TS \models Words(\varphi)$. Egy $s \in S$ állapotra pedig $s \models \varphi$ akkor és csak akkor, ha $s \models Words(\varphi)$, vagyis az s állapotból kiinduló összes maximális (ez esetben végtelen) útrészlet kielégíti a φ formulát.

Bizonyítható tétel, hogy tetszőleges φ LTL-formulára $Words(\varphi)$ ω -reguláris nyelv. A tétel konstruktív bizonyítása leírja azt a transzformációt, mely során a kifejezésből a nyelvet felismerő Büchi-automata előáll. [25]

1.6. Biszimuláció

Előfordulhat, hogy egy tranzíciós rendszer nem véges, sőt, állapottere folytonos. Ez meglehetősen nehézíti, bizonyos esetekben pedig akár lehetetlenné is teszi a modellellenőrzést, hiszen sok algoritmus csak véges állapottereken működik. A probléma áthidalható egy olyan véges rendszer vizsgálatával, melynek a nyoma megegyezik az eredeti rendszerével. Egy lehetséges megoldás a biszimuláció reláción keresztül vezet.

Az alábbi pont definiálja a biszimuláció relációt két tranzíciós rendszer között, majd az állapotok felett értelmezett biszimulációkon keresztül megmutatja, hogyan lehet az eredetivel biszimiláris, így azonos működésű rendszereket előállítani.

Biszimuláció tranzíciós rendszereken Legyenek $TS_i = (S_i, Act_i, \longrightarrow_i, I_i, AP, L_i)$, $i = 1, 2$ tranzíciós rendszerek AP felett. Ekkor (TS_1, TS_2) egy biszimulációja egy $\mathcal{R} \subseteq S_1 \times S_2$ bináris reláció, melyre

- $\forall s_1 \in I_1 : \exists s_2 \in I_2 : (s_1, s_2) \in \mathcal{R}$
- $\forall s_2 \in I_2 : \exists s_1 \in I_1 : (s_1, s_2) \in \mathcal{R}$
- $\forall (s_1, s_2) \in \mathcal{R}$:
 - $L_1(s_1) = L_2(s_2)$
 - $s'_1 \in Post(s_1) \Rightarrow \exists s'_2 \in Post(s_2) : (s'_1, s'_2) \in \mathcal{R}$
 - $s'_2 \in Post(s_2) \Rightarrow \exists s'_1 \in Post(s_1) : (s'_1, s'_2) \in \mathcal{R}$

TS_1 és TS_2 biszimiláris ($TS_1 \sim TS_2$), ha létezik egy $\mathcal{R}$ biszimuláció reláció a (TS_1, TS_2) párra.

Belátható, hogy egy így definiált $\mathcal{R}$ reláció ekvivalenciareláció. Ezen felül ha $TS_1 \sim TS_2$, akkor $Traces(TS_1) = Traces(TS_2)$, azaz a biszimuláció úgynevezett nyom-ekvivalencia. Emiatt a két rendszer ugyanazokat a tulajdonságokat elégíti ki.

Biszimuláció állapotokon Legyen $TS = (S, Act, \longrightarrow, I, AP, L)$. Egy $\mathcal{R}$ reláció biszimuláció az S állapothalmazon, ha minden $(s_1, s_2) \in \mathcal{R}$ párra

- $L_1(s_1) = L_2(s_2)$
- $s'_1 \in Post(s_1) \Rightarrow \exists s'_2 \in Post(s_2) : (s'_1, s'_2) \in \mathcal{R}$
- $s'_2 \in Post(s_2) \Rightarrow \exists s'_1 \in Post(s_1) : (s'_1, s'_2) \in \mathcal{R}$

Az s_1 és s_2 állapotok biszimilárisak ($s_1 \sim_{TS} s_2$), ha létezik egy $\mathcal{R}$ biszimuláció az S halmazon, hogy $(s_1, s_2) \in \mathcal{R}$.

A korábbiakhoz hasonló módon itt is bizonyítható, hogy $\mathcal{R}$ ekvivalenciareláció.

Hányados rendszer Alkalmazzuk az alábbi jelöléseket: legyen $[s]_{\mathcal{R}} = \{s' \in S : (s, s') \in \mathcal{R}\}$ az s elem $\mathcal{R}$ szerinti ekvivalenciaosztálya, $S \setminus \mathcal{R} = \{[s]_{\mathcal{R}} : s \in S\}$ pedig az $\mathcal{R}$ -ekvivalenciaosztályok halmaza.

Ekkor a $TS = (S, Act, \longrightarrow, I, AP, L)$ és az $\mathcal{R}$ biszimuláció hányadosa az a $TS \setminus \mathcal{R} = (S \setminus \mathcal{R}, \{\tau\}, \longrightarrow', I', AP, L')$ tranzíciós rendszer, ahol

- $I' = \{[s]_{\mathcal{R}} : s \in I\},$
- $L'([s]_{\mathcal{R}}) = L(s),$
- továbbá az $\longrightarrow'$ tranzíciós relációt az alábbi szabály határozza meg:

$$\frac{s \xrightarrow{\alpha} s'}{[s]_{\mathcal{R}} \xrightarrow{\tau} [s']_{\mathcal{R}}}$$

Bizonyítható, hogy ha $\mathcal{R}$ állapothalmaz feletti biszimuláció, akkor $TS \sim TS \setminus \mathcal{R}$, azaz a rendszer biszimiláris a hányadosával. Ebből következik, hogy ha a rendszer állapotait sikerül egy biszimuláció segítségével osztályozni, akkor az eredetivel biszimiláris, ugyanakkor annál tömörebb reprezentációt tudunk megadni.

2. fejezet

Időzített rendszerek modellezése időzített automatákkal

Mivel a beágyazott rendszerek viselkedése gyakran időfüggő, az előző fejezetben bemutatott tranzíciós rendszer formalizmust ki kell terjeszteni az idő múlását modellezni képes részletekkel. Időzített rendszerek modellezésének a gyakorlatban is jól bevált módja az időzített automata formalizmus [3][13] használata. Az alábbi fejezet bemutatja a formalizmus néhány, a későbbiek szempontjából fontos részletét.

Szintaxis Egy időzített automata egy $TA = (Loc, Act, C, \longrightarrow, Loc_0, Inv, AP, L)$ nyolcas, ahol

- Loc a vezérlési helyek véges halmaza,
- Act az akciók véges halmaza,
- C az órák véges halmaza,
- $\longrightarrow \subseteq Loc \times CC(C) \times Act \times 2^C \times Loc$ a tranzíciós reláció,
- $Inv : Loc \rightarrow CC(C)$ az invariánshozzárendelési függvény,
- AP a megszkott atomi propozíciók véges halmaza,
- $L : Loc \rightarrow 2^{AP}$ pedig a helyeket címkéző függvény.

A továbbiakban a $(\ell, g, \alpha, D, \ell' \in \longrightarrow)$ kifejezésre a $\ell \xrightarrow{g:\alpha,D} \ell'$ jelölést alkalmazzuk.

Órakényszerek $CC(C)$ jelöli az órák C halmaza feletti órakényszerek halmazát. Egy órakényszer az alábbi szintaxis szerint konstruálható meg:

Legyen $x \in C$ egy óra, $c \in \mathbb{N}$ természetes szám, $g \in CC(C)$ pedig jelöljön órakényszer. Ekkor órakényszerek az alábbi nyelvtan szerint generálhatók:

$$g ::= x < c \mid x \leq c \mid x > c \mid x \geq c \mid g \wedge g$$

Az órákényszerek az egyszerűség kedvéért intervallumként is jelölhetők.

Egy TA időzített automatához $ACC(TA)$ jelöli az átmenetekben őrfeltételként előforduló és a vezérlési helyek invariánsaként szereplő órákényszerek halmazát.

Órák kiértékelése Órák egy C halmazának η kiértékelése egy $\eta \rightarrow \mathbb{R}_{\geq 0}$ függvény, mely minden $x \in C$ órához annak aktuális $\eta(x)$ értékét rendeli.

A C halmaz fölötti összes kiértékelés halmazát $Eval(C)$ jelöli.

A fentiek ismeretében kielégítési reláció definiálható az órákényszerekhez. Legyen C órák egy halmaza, $x \in C$ óra, $\eta \in Eval(C)$, $c \in \mathbb{N}$, továbbá $g, h \in CC(C)$. Ekkor $\models \subseteq Eval(C) \times CC(C)$ az alábbi szabályok szerint definiált reláció:

$$\begin{aligned} \eta &\models \text{true} \\ \eta &\models x < c \iff \eta(x) < c \\ \eta &\models x \leq c \iff \eta(x) \leq c \\ \eta &\models \neg g \iff \neg(\eta \models g) \\ \eta &\models g \wedge h \iff \eta \models g \wedge \eta \models h \end{aligned}$$

Egy $d \in \mathbb{R}_{\geq 0}$ számhoz $\eta + d$ jelöli azt a kiértékelést, ahol minden óra értéke d -vel növelt, vagyis

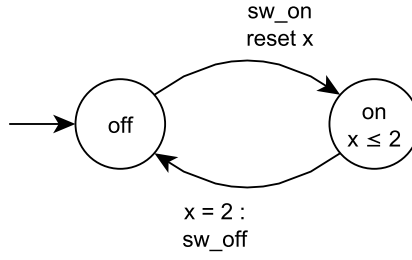
$$(\eta + d)(x) = \eta(x) + d$$

minden $x \in C$ órára. $\text{reset } x \text{ in } \eta$ jelöli azt a kiértékelést, ahol x nulla, a többi óra pedig változatlan, azaz

$$(\text{reset } x \text{ in } \eta)(y) = \begin{cases} \eta(y) & x \neq y \\ 0 & x = y \end{cases}$$

Egy időzített automata működése röviden az alábbiakban foglalható össze. Kezdetben a rendszer egy kiinduló helyen tartózkodik, és az összes óra értéke nulla. Az idő múlásával az órák értéke szinkron növekszik. A helyre megadott invariáns korlátozza az órák értékét, így az idő csak addig telhet, amíg az órák értékei a feltételnek eleget tesznek. Amennyiben időközben valamely tranzíció őrfeltétele teljesül, és az órák a tranzíció során módosuló értéke kielégíti az új vezérlési hely invariáns tulajdonságát, a tranzíció engedélyezett. Az engedélyezett tranzíciók nemdeterminisztikusan tüzelhetnek, ekkor a tranzíció a rendszert az új helybe viszi, valamint az órák egy rögzített részhalmazát nulla értékre állítja.

Egy példát szemléltet a 2.1. ábra. Az automata egy időzített kapcsolót ír le, amely a felkapcsolást követően két időegységgel automatikusan lekapcsol.



2.1. ábra. Időzített kapcsoló [4]

Időzített automaták kompozíciója A szakterülethez kapcsolódó modellezéshez szükséges, hogy komponensekből álló (elosztott) rendszereket tudjunk leírni. Időzített automaták esetében ezt a célt szolgálja a párhuzamos kompozíció művelet. A művelet eredményeként előálló időzített automatában a komponensek közötti interakciót a kézfogás (handshake) primitív biztosítja, mely informálisan két komponens közös akción történő szinkronizációját (szinkron tüzelését) modellezi.

A művelet formálisan a következőképpen adható meg:

Legyen $TA_i = (Loc_i, Act_i, C_i, \hookrightarrow_i, Loc_{0,i}, Inv_i, AP_i, L_i)$, $i = 1, 2$ olyan, hogy $C_1 \cap C_2 = \emptyset$ és $AP_1 \cap AP_2 = \emptyset$. Legyen továbbá $H \subseteq Act_1 \cap Act_2$ a kézfogások halmaza. Ekkor

$$TA_1 \parallel_H TA_2 = (Loc_1 \times Loc_2, Act_1 \cup Act_2, C_1 \cup C_2, \hookrightarrow, Loc_{0,1} \times Loc_{0,2}, Inv, AP_1 \cup AP_2, L)$$

úgy, hogy

- $L(\langle \ell_1, \ell_2 \rangle) = L_1(\ell_1) \cup L_2(\ell_2)$,
- $Inv(\langle \ell_1, \ell_2 \rangle) = Inv_1(\ell_1) \wedge Inv_2(\ell_2)$,
- a $\hookrightarrow$ tranzíciós relációt pedig az alábbi szabályok határozzák meg:

– ha $\alpha \in H$:

$$\frac{\ell_1 \xrightarrow{g_1:\alpha, D_1}_1 \ell'_1 \wedge \ell_2 \xrightarrow{g_2:\alpha, D_2}_2 \ell'_2}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g_1 \wedge g_2:\alpha, D_1 \cup D_2} \langle \ell'_1, \ell'_2 \rangle}$$

– ha $\alpha \notin H$:

$$\frac{\ell_1 \xrightarrow{g:\alpha, D}_1 \ell'_1}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g:\alpha, D} \langle \ell'_1, \ell_2 \rangle}$$

illetve

$$\frac{\ell_2 \xrightarrow{g:\alpha, D}_2 \ell'_2}{\langle \ell_1, \ell_2 \rangle \xrightarrow{g:\alpha, D} \langle \ell_1, \ell'_2 \rangle}$$

2.1. Időzített automaták szemantikája

Az alábbi pont bemutatja az időzített automata formalizmushoz rendelt formális szemantikát.

Legyen $TA = (Loc, Act, C, \hookrightarrow, Loc_0, Inv, AP, L)$ időzített automata. Ekkor $TS(TA) = (S, Act', \longrightarrow, I, AP', L')$ az a tranzíciós rendszer, melyre

- $S = Loc \times Eval(C)$,
- $Act' = Act \cup \mathbb{R}_{\geq 0}$,
- $I = \{\langle l_0, \eta \rangle : l_0 \in Loc_0 \wedge \eta(x) = 0, \forall x \in C\}$,
- $AP' = AP \cup ACC(TA)$,
- $L'(\langle \ell, \eta \rangle) = L(\ell) \cup \{g \in ACC(TA) : \eta \models g\}$,
- a $\longrightarrow$ tranzíciós relációt pedig az alábbi szabályok határozzák meg:
 - diszkrét tranzíciók:

$$\frac{\ell \xrightarrow{g:\alpha,D} \ell' \wedge \eta \models g \wedge \eta' = \text{reset } D \text{ in } \eta \wedge \eta' \models Inv(\ell')}{\langle \ell, \eta \rangle \xrightarrow{\alpha} \langle \ell', \eta' \rangle}$$

- késleltető tranzíciók valamely $d \in \mathbb{R}_{\geq 0}$ késleltetéssel:

$$\frac{\eta' = \eta + d \wedge \eta' \models Inv(\ell)}{\langle \ell, \eta \rangle \xrightarrow{d} \langle \ell, \eta' \rangle}$$

2.2. Időzített automaták tulajdonságai

Időzített automaták tranzíciós rendszer szemantikája megadja a rendszer lehetséges viselkedéseit. Ezek némelyike azonban nem realizisztikus. Az alábbi pont ezeket a viselkedéseket definiálja.

Végrehajtási idő Legyen TA időzített automata Act akcióhalmazzal. Ekkor $TS(TA)$ egy $\tau \in Act$ akciójának végrehajtási idejét az

$$ExecTime(\tau) = \begin{cases} 0 & \tau \in Act \\ d & \tau = d \in \mathbb{R}_{\geq 0} \end{cases}$$

függvény adja meg. A $\rho = s_0 \xrightarrow{\tau_0} s_1 \xrightarrow{\tau_1} s_2 \xrightarrow{\tau_2} \dots$ végtelen végrehajtásrészlet során eltelt idő alatt az $ExecTime(\rho) = \sum_{i=0}^{\infty} ExecTime(\tau_i)$ számot értjük. Ezek alapján értelmezhető a ρ végrehajtásrészlet által indukált π útrészlet $ExecTime(\pi)$ végrehajtási ideje is.

Idő-divergencia $TS(TA)$ egy π végtelen útrészlet idő-divergens, ha $ExecTime(\pi) = \infty$, egyébként π idő-konvergens.

Mint $TS(TA)$ definíciójából látható, elkerülhetetlen, hogy egy rendszerben idő-konvergens utak legyenek. Olyan modellellenőrzési módszert kell tehát alkotni, mely ezeket az utakat figyelmen kívül hagyja.

Timelock $TS(TA)$ egy s állapota timelockot tartalmaz, ha $Paths(s)$ minden útvonalra idő-konvergens. TA timelockmentes, ha minden $s \in Reach(TS(TA))$ elérhető állapot timelockmentes.

A timelock jelenléte egy rendszerben tervezési hibára utalhat, így a modellezés során a timelockot tartalmazó állapotok azonosítása különös jelentőséggel bír.

Nemzénó tulajdonság $TS(TA)$ egy π végtelen útrészlete zénó, ha idő-konvergens és π mentén végtelen sok $\alpha \in Act$ akció hajtódik végre. TA nemzénó, ha $TS(TA)$ minden útja nemzénó.

2.3. Időzített automaták véges absztrakciója

Az időzített automaták szemantikájánál definiált $TS(TA)$ rendszeren jól látszott, hogy nem véges. A probléma egy megfelelő biszimuláció segítségével hidalható át. Ilyen biszimulációt definiál az óraértékelések között értelmezett óraekvivalencia reláció.

2.3.1. Óraekvivalencia

Legyen TA időzített automata a C órahalmaz felett. Jelölje c_x azt a legnagyobb konstanst, mellyel bármely $g \in ACC(TA)$ órakényszerben az $x \in C$ óra összehasonlításra kerül. Az η, η' óraértékelések ekvivalensek ($\eta \cong \eta'$), ha a következő feltételek teljesülnek:

- $\forall x \in C : \eta(x) > c_x \Leftrightarrow \eta'(x) > c_x$
- bármely $x, y \in C$ órára, melyekre $\eta(x), \eta'(x) \leq c_x$ és $\eta(y), \eta'(y) \leq c_y$ teljesül, érvényesek az alábbiak:
 - $\lfloor \eta(x) \rfloor = \lfloor \eta'(x) \rfloor$
 - $\{\eta(x)\} = 0 \Leftrightarrow \{\eta'(x)\} = 0$
 - $\{\eta(x)\} \leq \{\eta(y)\} \Leftrightarrow \{\eta'(x)\} \leq \{\eta'(y)\}$

ahol $\lfloor \eta(x) \rfloor$ alsó egészrészt, $\{\eta(x)\}$ pedig törtrészt jelöl.

Az $\eta \in Eval(C)$ óraértékelés régiója az $[\eta] \in Eval(C) \setminus \cong$ ekvivalenciaosztály.

Állapotrégió Az óraekvivalencia segítségével definiálható $TS(TA)$ állapotainak ekvivalenciája is. Az $\langle \ell_1, \eta_1 \rangle, \langle \ell_2, \eta_2 \rangle \in S$ állapotok ekvivalensek ($\langle \ell_1, \eta_1 \rangle \cong \langle \ell_2, \eta_2 \rangle$) akkor és csak akkor, ha $\ell_1 = \ell_2 \wedge \eta_1 \cong \eta_2$. Az $s \in S$ állapot régiója az $[s] \in S \setminus \cong$ ekvivalenciaosztály.

Bizonyítható, hogy $\cong$ biszimuláció $TS(TA)$ állapotai felett. [23]

Nem korlátos órarégió Az r_∞ nem korlátos órarégió az az $[\eta] \in Eval(C) \setminus \cong$ órarégió, melyre minden $x \in C$ órára $\eta(x) > c_x$.

Rákövetkező régió Legyenek $r, r' \in Eval(C) \setminus \cong$ órarégiók. Az r' az r régió rákövetkezője ($r' = succ(r)$), ha

- $r = r' \Rightarrow r = r_\infty$
- ha $r \neq r', r \neq r_\infty$, akkor minden $\eta \in r$ kiértékeléshez létezik $d \in \mathbb{R}_{>0}$ késleltetés, hogy $\eta + d \in r'$ úgy, hogy minden $0 \leq d' \leq d$ késleltetésre $\eta + d' \in r \cup r'$.

2.3.2. Régió tranzíciós rendszer

Legyen $TA = (Loc, Act, C, \hookrightarrow, Loc_0, Inv, AP, L)$ nemzénő időzített automata. Ekkor az $RTS(TA) = (S, Act \cup \{\tau\}, \longrightarrow, I, AP', L')$ régió tranzíciós rendszer a következőképpen definiált rendszer:

- S, I, AP' és L' ugyanaz, mint a $TS(TA) \setminus \cong$ rendszerben,
- a $\longrightarrow$ tranzíciós relációt pedig az alábbi szabályok határozzák meg:
 - diszkrét tranzíciók:

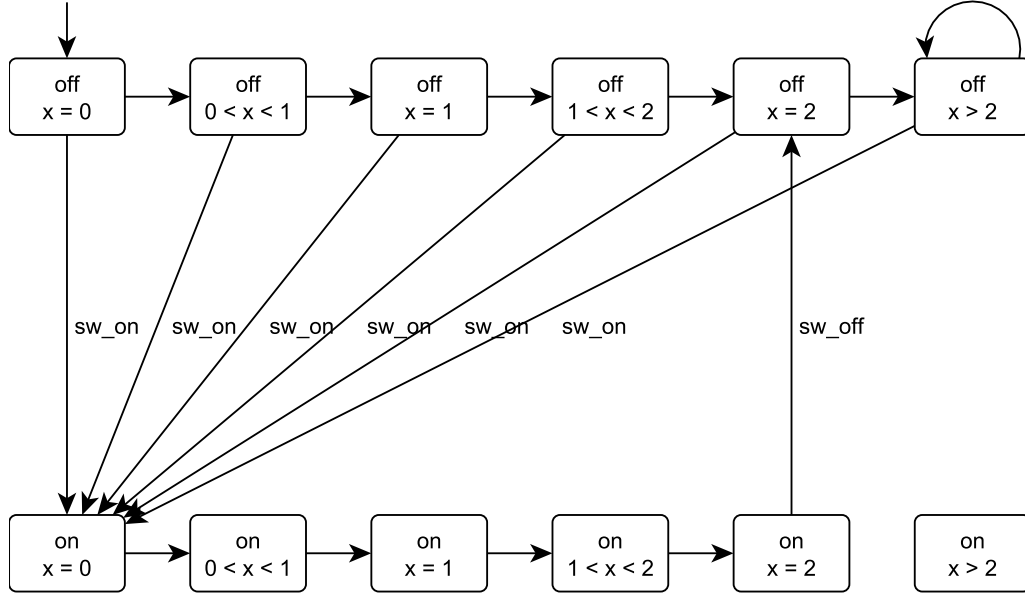
$$\frac{\ell \xrightarrow{g:\alpha,D} \ell' \wedge \eta \models g \wedge \eta' = \text{reset } D \text{ in } \eta \wedge \eta' \models Inv(\ell')}{\langle \ell, [\eta] \rangle \xrightarrow{\alpha} \langle \ell', [\eta'] \rangle}$$

- régiók közötti tranzíciók:

$$\frac{\eta \models Inv(\ell) \wedge [\eta'] = succ([\eta]) \wedge \eta' \models Inv(\ell')}{\langle \ell, [\eta] \rangle \xrightarrow{\tau} \langle \ell, [\eta'] \rangle}$$

A definícióból látható, hogy $RTS(TA)$ és $TS(TA) \setminus \cong$ között a különbség abban rejlik, hogy míg a hányados rendszerben a tranzíciók akciói elvesznek, $RTS(TA)$ megtartja őket. Ennél fontosabb azonban, hogy $RTS(TA)$ csak azokat a késleltető tranzíciókat tartalmazza, melyek egy adott régió rákövetkezőjébe visznek. Ez a konstrukció a több régió átívelő késleltetések, valamint az idő-konvergens útvonalak kiküszöböléséhez szükséges. TA nemzénő időzített automatára bizonyítható ugyanis, hogy egy végtelen útrészlet akkor és csak akkor idő-konvergens, ha annak késleltető tranzíciói egy idő után nem visznek ki egy adott régióból (leszámítva az r_∞ nem korlátos régiót).

A 2.1. ábrán szereplő kapcsoló régió tranzíciós rendszerét mutatja be a 2.2. ábra.



2.2. ábra. A kapcsolóhoz tartozó régió tranzíciós rendszer [4]

2.4. Időzített automaták korlátos modellellenőrzése

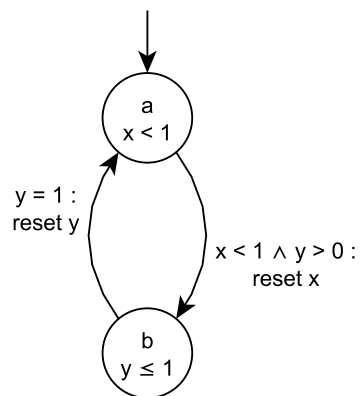
Látható, hogy egy TA időzített automata modellellenőrzése megoldható az $RTS(TA)$ tranzíciós rendszer birtokában, hiszen utóbbi egy olyan véges rendszer, mely (az idő-konvergens utakat és a több régió átívelő késleltetéseket leszámítva) a TA rendszer szemantikájának megfelelően, annak tulajdonságait megtartva működik.

Felmerül ugyanakkor a kérdés, hogy mindenképpen szükség van-e a régió tranzíciós rendszerre, hiszen ellenpélda - egy, a korábbiakban vázolt tulajdonsággal rendelkező hurok - a $TS(TA)$ végtelen rendszerben is kereshető. Ekkor a hurok záródásának figyelésekor nem a két állapot óraekvivalenciáját, hanem egyenlőségét vizsgáljuk.

Ahhoz persze, hogy a módszer valóban teljes legyen (azaz ha van ellenpélda, akkor az kimutatható legyen), teljesülnie kell annak a feltételnek, hogy minden, az $RTS(TA)$ rendszerben található ellenpéldához tartozik ellenpélda a $TS(TA)$ rendszerben.

Ez a feltétel azonban - a korábbi meggyőződés [20] ellenére - nem teljesül általánosan [14]. Tekintsük a 2.3. ábrán szereplő időzített automatát. A következő gondolatmenettel belátható, hogy az automata minden végrehajtása hurokmentes.

Jelölje t_i^a és t_i^b ($i \geq 1$) az a , illetve b vezérlési helyen az i -edik látogatáskor eltöltött időt. Mivel az a helyről kimenő tranzíció két eltüzélése között kevesebb, mint 1 időegység telik el, érvényes a $t_i^b + t_{i+1}^a < 1$ egyenlőtlenség. A másik élnél ugyanezzel az ötlettel élve $t_i^a + t_i^b = 1$ feltétel adódik. A két egyenlőtlenséget kombinálva a $t_i^a > t_{i+1}^a$ és $t_i^b < t_{i+1}^b$ következtetésre jutunk. Ezen felül tudjuk, hogy az a vezérlési helyen az $(i+1)$ -edik látogatáskor a két óra közötti különbség megegyezik t_i^b értékével, illetve fordítva. Emiatt a két egymást követő látogatáskor a különbség szigorúan nő, b esetében pedig szigorúan csökken. Emiatt a rendszer soha nem kerül kétszer ugyanabba az állapotba, amivel beláttuk, hogy az au-



2.3. ábra. *Példa hurokmentes időzített automatára [14]*

tomatának nincsen hurok alakú végrehajtása. Ugyanakkor az automatának van végtelen nemzénó lefutása: a $t_i^a = \frac{1}{i+2}$, $t_i^b = 1 - t_1^a$ megfelelő választás, ekkor $\sum_1^\infty t_i^a + 1 - t_i^a = \infty$. [14]

3. fejezet

Omega-reguláris tulajdonságok modellellenőrzése

Az alábbi fejezetben bemutatom a k-indukciós modellellenőrző algoritmust. Mivel k-indukcióval állapotok elérhetősége vizsgálható, ezért olyan eljárásokat kellett fejlesztenem az irodalom alapján, melyek segítségével bonyolultabb ω -reguláris tulajdonságok is vizsgálhatók.

3.1. Invariáns tulajdonságok vizsgálata k-indukcióval

A k-indukciós algoritmus segítségével lehetőség van végtelen állapotterű rendszerek vizsgálatra. A módszer a matematikai indukción, illetve annak általánosításán alapul.

Matematikai indukció Legyen $P_0, P_1, P_2, \dots$ propozicionális logikai kifejezések végtelen sorozata. Annak bizonyítására, hogy P_n minden $n \in \mathbb{N}$ esetén teljesül, ismert módszer a hagyományos indukció. Ennek sémája a következő:

$$P_0 \wedge \forall n (P_n \Rightarrow P_{n+1}) \Rightarrow \forall n P_n$$

K-indukció A módszer érvényes általánosítása a k-indukció, mely a következőt állítja:

$$\bigwedge_{i=0}^{k-1} P_i \wedge \forall n \left(\left(\bigwedge_{i=0}^{k-1} P_{n+i} \right) \Rightarrow P_{n+k} \right) \Rightarrow \forall n P_n$$

Ez az alábbi eljárást vázolja fel:

1. Kiinduló lépés: belátni, hogy $P_0, P_1, \dots, P_{k-1}$, teljesül.
2. Indukciós lépés: belátni minden $n \in \mathbb{N}$ esetére, hogy ha $P_n, P_{n+1}, \dots, P_{n+k-1}$ teljesül, akkor P_{n+k} is teljesül.

A definícióból adódik, hogy ha egy tulajdonságot sikerül belátni k -indukcióval, akkor a tulajdonság tetszőleges, k -nál nagyobb indukciós mélységben bizonyítható. Igaz az is, hogy a mélység növelésével általában erősebb módszert kapunk. Ez alól azok az esetek kivételek, amikor tetszőleges k mélységben létezik olyan, az indukciós lépésnél talált $P_n, P_{n+1}, \dots, P_{n+k-1}, \neg P_{n+k}$ ellenpélda, mely kiegészíthető visszafelé egy P_{n-1} állítással.

Alkalmazás tranzíciós rendszerekre Legyen $TS = (S, Act, \longrightarrow, I, AP, L)$ egy tranzíciós rendszer terminális állapotok nélkül, P_{inv} pedig invariáns tulajdonság a Φ invariáns feltétellel.

Definíció szerint $TS \models P_{inv}$ pontosan akkor, ha minden $s \in Reach(TS)$ elérhető állapotban $s \models \Phi$. Ennek bizonyítása k -indukcióval a következőképpen történhet [19]:

1. Kiinduló lépés: korlátos modellellenőrzés. Azt kell belátni, hogy minden $\hat{\pi} = s_0, s_1, \dots, s_{k-1}$ kiinduló ($s_0 \in I$) útrészlet olyan, hogy $s_i \models \Phi, \forall 0 \leq i \leq k-1$. Egy olyan útrészlet, mely nem tesz eleget a feltételnek, ellenpélda a P_{inv} tulajdonság teljesülésére.
2. Indukciós lépés: bizonyítandó, hogy minden $\hat{\pi} = s_n, s_{n+1}, \dots, s_{n+k-1}, s_{n+k}$ útrészlet olyan, hogy ha $s_i \models \Phi, \forall n \leq i \leq n+k-1$, akkor $s_{n+k} \models \Phi$. Az ezt a feltételt sértő útrészleteket induktív ellenpéldának hívjuk.

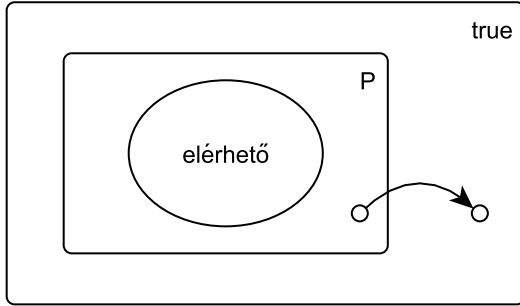
Látható, hogy mindkét lépésnél érdemes kizárólag hurokmentes útrészletek vizsgálatára szorítkozni.

Ha a kiinduló lépés során adódik egy ellenpélda, úgy a feltétel nyilván nem teljesül. Amennyiben se ellenpélda, sem induktív ellenpélda nincsen, úgy a feltétel a k -indukció értelmében teljesül, azaz minden elérhető állapotban Φ érvényes. Ellenkező esetben a bizonyításhoz az alábbi technikák valamelyike alkalmazható:

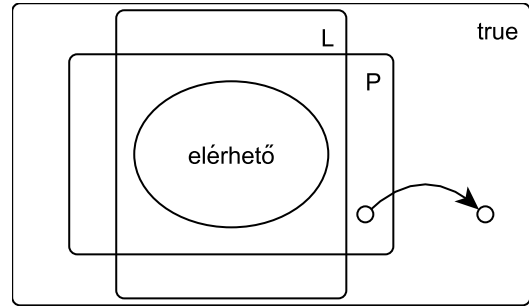
- Indukció mélységének növelése. Ez - ahogy korábban láttuk - az esetek többségében erősebb módszert eredményez. Ugyanakkor általános esetben a nagyobb indukciós mélység lassabb ellenőrzést is okoz, hiszen hosszabb útvonalak vizsgálatát teszi szükségessé.
- A tulajdonság erősítése. Ez azt jelenti, hogy P_{inv} helyett egy erősebb $P'_{inv} \subset P_{inv}$ tulajdonságot próbálunk bizonyítani.
- Kiegészítő lemmák használata. Amennyiben tudjuk, hogy L_{inv} érvényes invariáns tulajdonság, úgy az indukciós lépésnél elég az olyan útrészleteket figyelembe vennünk, ahol az L_{inv} tulajdonság végig teljesül, hiszen egy olyan állapot, ahol a feltétel sérül, nyilván nem elérhető. A séma ekkor így módosul:

$$\forall n \left(\left(\bigwedge_{i=0}^{k-1} P_{n+i} \wedge L_{n+i} \right) \Rightarrow P_{n+k} \right)$$

A kiegészítő invariánsok jelentőségét szemlélteti a 3.1. és 3.2. ábra [10]. Míg az első esetben $k = 1$ mélységben a tulajdonság nem bizonyítható (nem induktív), addig egy induktív invariáns megadásával a bizonyítás sikeres.



3.1. ábra. P invariáns, de nem induktív



3.2. ábra. P induktív L -hez képest

A k -indukció tehát invariáns tulajdonságok vizsgálatának hatékony eszköze. Ahhoz azonban, hogy összetettebb tulajdonságok is vizsgálhatóak legyenek az algoritmussal, további technikákat kell alkalmazni. A következő két pont ilyen módszereket mutat be.

3.2. Omega-reguláris modellellenőrzés perzisztencia vizsgálatával

A problémát egyszerűsíti, hogy minden ω -reguláris tulajdonság ellenőrzése az automataelméleti megközelítés [24] értelmében a modell megfelelő leképezésével visszavezethető valamilyen perzisztencia tulajdonság vizsgálatára. A leképezést az alábbi pont írja le.

Tranzíciós rendszer és Büchi-automata szorzata Legyen $TS = (S, Act, \longrightarrow, I, AP, L)$ egy terminális állapotokat nem tartalmazó tranzíciós rendszer, $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ pedig egy nemblokkoló nemdeterminisztikus Büchi-automata. Ekkor a $TS \otimes \mathcal{A}$ szorzat egy olyan $(S \times Q, Act, \longrightarrow', I', AP', L')$ tranzíciós rendszer, ahol

- a $\longrightarrow'$ tranzíciós reláció a

$$\frac{s \xrightarrow{\alpha} t \wedge q \xrightarrow{L(t)} p}{\langle s, q \rangle \xrightarrow{\alpha'} \langle t, p \rangle}$$

szabály szerinti legszűkebb reláció,

- $I' = \left\{ \langle s_0, q \rangle : s_0 \in I, \exists q_0 \in Q : q_0 \xrightarrow{L(s_0)} q \right\}$, vagyis a szorzatban minden olyan $\langle s_0, q \rangle$ állapotpár kezdőállapot, ahol s_0 kezdőállapot a TS rendszerben, és az s_0 által meghatározott $L(s_0)$ szó az $\mathcal{A}$ automatát valamely q_0 kezdőállapotából q állapotba tudja vinni,
- $AP' = Q$,
- az $L' : S \times Q \rightarrow 2^Q$ címkéző függvényt pedig az $L(\langle s, q \rangle) = \{q\}$ hozzárendelés adja meg.

A modellellenőrzési probléma visszavezetése perzisztencia vizsgálatára Legyen TS egy AP feletti, terminális állapotokat nem tartalmazó tranzíciós rendszer, P pedig ω -reguláris tulajdonság AP felett.

Legyen adott továbbá az $\mathcal{A}$ nemblokkoló nemdeterminisztikus Büchi-automata az $\mathcal{L}_\omega = (2^{AP})^\omega \setminus P$ elfogadott nyelvvel.

Jelölje ezen felül $P_{pers(\mathcal{A})}$ azt az $AP' = Q$ feletti perzisztencia tulajdonságot, melyet a

$$\Phi = \bigwedge_{q \in F} \neg q$$

perzisztencia feltétel határoz meg. A korábban ismertetett definíció alapján a $P_{pers(\mathcal{A})}$ tulajdonság így azt fogja kifejezni, hogy előbb-utóbb már csak nemelfogadó állapotok fordulnak elő az $\mathcal{A}$ automata futásában.

Ekkor az alábbi állítások ekvivalensek:

1. $TS \models P$
2. $Traces(TS) \cap \mathcal{L}_\omega(\mathcal{A}) = \emptyset$
3. $TS \otimes \mathcal{A} \models P_{pers(\mathcal{A})}$

A tétel értelmében tetszőleges ω -reguláris tulajdonság modellellenőrzése visszavezethető perzisztencia ellenőrzésére. Ehhez elő kell állítanunk a kívánt tulajdonság komplementerét leíró nemdeterminisztikus Büchi-automata és a rendszer szorzatát, majd meg kell vizsgálni, hogy az így kapott rendszerben elfogadó állapottal címkézett állapot csak véges sokszor fordul-e elő.

3.3. Perzisztencia vizsgálata invariáns tulajdonságok ellenőrzésével

Az előző pont értelmében, ha a perzisztencia ellenőrzését sikerül visszavezeti elérhetőség vizsgálatára, akkor már kísérletet lehet tenni ω -reguláris tulajdonságok k-indukciós algoritmussal történő ellenőrzésére.

Legyen tehát $TS = (S, Act, \longrightarrow, I, AP, L)$ terminális állapotokat nem tartalmazó tranzíciós rendszer, P_{pers} pedig a Φ propozicionális logikai kifejezés által meghatározott AP feletti perzisztencia tulajdonság.

Az alábbi pont két olyan módszert ismertet, amelyeket alkalmazva a $TS \models P_{pers}$ eldöntési problémát invariáns tulajdonságok ellenőrzésével lehet megválaszolni.

3.3.1. Szükséges invariáns feltétel a rendszer perzisztenciájára

Véges rendszerek esetén minden, egy adott perzisztencia tulajdonságot sértő út jellemezhető egy olyan, lasszót formáló véges prefixummal [6], melynek belsejében a perzisztencia

feltétel sérül. Ilyen rendszerek esetében tehát az ω -reguláris modellellenőrzés egy speciális körkeresési problémaként fogható fel. Bár végtelen rendszerekben a fenti állítás nem feltétlenül teljesül, az ötlet alkalmas lehet ellenpéldák keresésére.

Az eljárást a modell szintjén, a prefixumok kiszűrésével valósítjuk meg [5]. A módosítás két lépésben valósul meg: először előáll a lasszókat jelezni képes TS' rendszer, majd ebből adódik az a TS'' rendszer, mely felismeri, ha egy lasszón belül sérül a jellemző.

Legyen tehát $TS' = (S \times S \times \mathbb{B} \times \mathbb{B}, Act, \longrightarrow', I', AP, L)$ az a tranzíciós rendszer, melyre

- $I' = \{\langle s, s, l_b, 0 \rangle : s \in I, l_b \in \mathbb{B}\},$
- továbbá a $\longrightarrow'$ reláció az alábbi szabályok segítségével adható meg:

$$\frac{s \xrightarrow{\alpha} s'}{\langle s, t, 0, 0 \rangle \xrightarrow{\alpha}' \langle s', t, 0, 0 \rangle \wedge \langle s, t, 0, 0 \rangle \xrightarrow{\alpha}' \langle s', s', 1, 0 \rangle}$$

$$\frac{s \xrightarrow{\alpha} s', s' \neq t}{\langle s, t, 1, 0 \rangle \xrightarrow{\alpha}' \langle s', t, 1, 0 \rangle \wedge \langle s, s', 1, 0 \rangle \xrightarrow{\alpha}' \langle s', s', 1, 1 \rangle}$$

valamint

$$\frac{s \xrightarrow{\alpha} s'}{\langle s, t, 1, 1 \rangle \xrightarrow{\alpha}' \langle s', t, 1, 1 \rangle}$$

A rendszer működését a következőképpen magyarázhatjuk. A módosított rendszer állapota azon kívül, hogy leköveti az eredeti rendszer működését, képes a rendszerben található hurkokat kimutatni. Ehhez a rendszerállapot második elemébe nemdeterminisztikusan elmentjük az eredeti rendszer aktuális állapotát, és a harmadik elembe egyest írva jelezzük, hogy mentés történt. Ezzel gyakorlatilag kijelöltük azt az állapotot, amely egy esetleges hurok kezdete lehet. A negyedik elemet akkor állítjuk, ha már mentettünk, és a mentett állapot megegyezik az aktuális állapottal, vagyis a hurok záródott.

A TS' rendszer birtokában megkonstruálhatjuk azt a

$$TS'' = (S \times S \times \mathbb{B} \times \mathbb{B} \times \mathbb{B}, Act, \longrightarrow'', I'', AP'', L'')$$

rendszert, melyben képesek vagyunk eldönteni a $\neg\Phi$ kijelentés hurkon belüli teljesülését. A rendszer az alábbi:

- $I'' = I' \times \{0\},$
- $AP'' = \{bad\}$
- $L''(\langle s, t, l_b, l_c, f \rangle) = \{bad\},$ ha $l_c = 1 \wedge f = 1,$ egyébként $\emptyset,$

- valamint a $\longrightarrow''$ relációra vonatkozó szabályok:

$$\frac{\langle s, t, l_b, l_c \rangle \xrightarrow{\alpha'} \langle s', t', l'_b, l'_c \rangle, l_c = 1 \vee l'_b = 0 \vee s' \models \Phi}{\langle s, t, l_b, l_c, f \rangle \xrightarrow{\alpha''} \langle s', t', l'_b, l'_c, f \rangle}$$

és

$$\frac{\langle s, t, 1_b, 0 \rangle \xrightarrow{\alpha'} \langle s', t', 1, l'_c \rangle, s' \models \neg \Phi}{\langle s, t, l_b, 0, f \rangle \xrightarrow{\alpha''} \langle s', t', 1, l'_c, 1 \rangle}$$

Látható, hogy a TS'' rendszer ötödik eleme azt követi nyomon, hogy a hurok kezdetének kijelölése óta előfordult-e olyan állapot, mely sérti a Φ perzisztenciafeltételt.

Ezek után legyen P_{inv} a $\Psi = \neg bad$ invarinacia feltétel által meghatározott invariáns tulajdonság AP'' felett. Tudjuk, hogy a TS'' rendszerben $s \models \neg \Psi$ pontosan akkor, ha a TS rendszerben van egy olyan hurok, melynek belsejében Φ sérül. Egy ilyen hurok pedig ellenpélda a vizsgált perzisztencia tulajdonságra, hiszen ilyenkor van olyan útvonal, mely során $\neg \Phi$ végtelen sokszor teljesül. Formálisan:

$$TS \models P_{pers} \Rightarrow TS'' \models P_{inv}$$

Ahogy korábban szerepelt, véges rendszerek esetén az állítás megfordítása is igaz.

A fentiek értelmében végtelen rendszerek vizsgálatánál a módszer nem feltétlenül vezet eredményre, hiszen akár az is előfordulhat, hogy a rendszerben nincsen a fenti módon definiált hurok. Ilyenkor járható út a rendszer egy megfelelő biszimulációjának vizsgálata, azonban ez nem feltétlenül szükséges. Egy állapotok felett értelmezett $\mathcal{R}$ biszimuláció reláció birtokában ugyanis megtehetjük azt, hogy a hurok záródásánál nem az $s = s'$ feltételt vesszük alapul, hanem az $[s]_{\mathcal{R}} = [s']_{\mathcal{R}}$ feltételt. Ekkor TS' rendszer tranzíciós relációjának képzésénél a második szabály az alábbiak szerint módosul:

$$\frac{s \xrightarrow{\alpha} s', [s']_{\mathcal{R}} \neq [t]_{\mathcal{R}}, [s']_{\mathcal{R}} = [s'']_{\mathcal{R}}}{\langle s, t, 1, 0 \rangle \xrightarrow{\alpha'} \langle s', t, 1, 0 \rangle \wedge \langle s, s'', 1, 0 \rangle \xrightarrow{\alpha} \langle s', s'', 1, 1 \rangle}$$

Természetesen amennyiben a $TS \setminus \mathcal{R}$ rendszer nem véges, úgy ez az eljárás sem biztosítja a módszer teljességét.

3.3.2. Elégséges invariáns feltétel a rendszer perzisztenciájára

Egy rendszer perzisztenciájának bizonyítása intuitív módon történhet úgy, hogy megpróbálunk korlátot adni a rendszer útvonalai során érintett rossz állapotok számára. Ez véges rendszerek esetében mindig teljes eljárás, hiszen ha a korlátot az állapotok számánál na-

gyobbra választjuk [8], azzal az előző pont értelmében az összes rossz viselkedést sikerül megfogni.

A módszer a modell egyszerű számlálóval történő kiegészítésével történhet:

Legyen $l \in \mathbb{Z}^+$ tetszőleges pozitív egész, $TS' = (S \times \mathbb{N}, Act, \longrightarrow', I', AP', L')$ pedig az alábbi rendszer:

- $I' = \{\langle s, l \rangle : s \in I\},$
- $AP' = \{bad\},$
- $L(\langle s, n \rangle) = \{bad\},$ ha $n = 0$, egyébként $\emptyset,$
- $\longrightarrow'$ reláció pedig az alábbi:

$$\frac{s \xrightarrow{\alpha} s', s' \models \Phi}{\langle s, n \rangle \xrightarrow{\alpha} \langle s', n \rangle}$$

illetve

$$\frac{s \xrightarrow{\alpha} s', s' \models \neg\Phi}{\langle s, n \rangle \xrightarrow{\alpha} \langle s', \max\{n-1, 0\}\rangle}$$

Jól látható, hogy a rendszer l -től visszafelé számolja a végrehajtása során érintett, a Φ feltételt sértő állapotokat (a kezdőállapotot kivéve). A rendszer akkor kerül rossz állapotba, ha a számláló eléri a nulla értéket. A $\Psi = \neg bad$ feltétel által meghatározott P_{inv} invariáns tulajdonság tehát azt fogalmazza meg, hogy a végrehajtás során csak véges sok (l -nél kevesebb) kedvezőtlen állapot fordul elő. Ebből adódik az alábbi feltétel:

$$\exists l \in \mathbb{Z}^+ : TS' \models P_{inv} \Rightarrow TS \models P_{pers}$$

Igaz természetesen az is, hogy ha egy l érték bizonyítja az állítást, akkor $l+1$ is. Ezen felül a fentiek értelmében véges rendszer esetén itt is érvényes az állítás megfordítása, hiszen a $l > |S|$ jó választás.

4. fejezet

Magasabb szintű formalizmus időzített rendszerek modellezéséhez

Az előző fejezetben bemutatott időzített automata formalizmus lehetővé teszi időzített rendszerek modellezését. Bizonyos esetekben azonban a formalizmus túl merev ahhoz, hogy a segítségével a vizsgált rendszert (illetve bizonyos esetekben a követelményeit) természetes módon modellezzük.

A naptáramtomata formalizmus [12] tranzíciós rendszerek egy osztálya, mely alkalmas időzített rendszerek a k -indukciós algoritmushoz illeszkedő modellezésére. A formalizmust a hatékonyabb modellellenőrzés érdekében módosítottam, majd a módosított formalizmus főlé a mérnöki szemlélethez illeszkedő, magasabb szintű nyelvet definiáltam. A formalizmust úgy definiáltam, hogy az időzített automata formalizmus bizonyos tulajdonságait megtartsa. Az új formalizmushoz definiálom azokat az ekvivalencia relációkat, amelyek segítségével a hatékony modellellenőrzést megvalósítottam.

Az alábbi fejezet munkám kapcsolódó részeit mutatja be.

4.1. A naptáramtomata szemantika

Jelölje Var változók egy véges halmazát, $dom(x)$ pedig az $x \in Var$ változó típusát. Ekkor a változók egy σ kiértékelése egy olyan függvény, mely minden $x \in Var$ változóhoz egy $dom(x)$ halmazból vett értéket rendel. A Var változók feletti kiértékelések halmazát $Eval(Var)$ jelöli.

Tekintsünk egy $TS = (S \times Eval(Var), Act, \longrightarrow, I, AP, L)$ tranzíciós rendszert

$$\{t, x_1, x_2, \dots, x_n, C\} \subseteq Var$$

változókkal, ahol

- t az aktuális idő, $dom(t) = \mathbb{R}_{\geq 0}$,

- $x_i \in T$ időzítés, $\text{dom}(x_i) = \mathbb{R}_{\geq 0}$,
- C pedig az eseménynaptárt tároló változó. Ennek megfelelően $\text{dom}(C)$ az összes lehetséges eseménynaptár halmaza, ahol eseménynaptár alatt egy

$$c = \{\langle m_i, t_i \rangle : m_i \in M, t_i \in \mathbb{R}_{\geq 0}\}$$

alakú véges halmazt értünk (M a naptáresemények vagy üzenetek véges halmaza).

Jelölje $\min(\sigma(T)) = \min\{\sigma(x_i) : x_i \in T\}$ egy adott $\langle s, \sigma \rangle$ állapotban az időzítések minimumát, $\min(c) = \min\{t_i : \langle m_i, t_i \rangle \in c\}$ a c eseménynaptár legkorábban esedékes eseményének időpontját (üres naptár esetén a $+\infty$ értéket), $Ev_u(c) = \{\langle m_i, t_i \rangle : t_i = u, \langle m_i, t_i \rangle \in c\}$ pedig a c eseménynaptár u időpontra ütemezett eseményeit tároló eseménynaptárt.

A TS tranzíciós rendszer naptárautomata [12], ha az alábbi feltételek teljesülnek:

- minden $\langle S, \sigma \rangle \in I$ kezdőállapotban $\sigma(t) \leq \min(\sigma(T))$ és $\sigma(t) \leq \min(\sigma(C))$,
- $a \longrightarrow$ tranzíciós reláció $\langle S, \sigma \rangle \xrightarrow{\alpha} \langle S', \sigma' \rangle$ tranzícióira az alábbi szabályok érvényesek:
 - diszkrét tranzíciók:
 - * $\sigma(t) = \min(\sigma(T))$ vagy $\sigma(t) = \min(\sigma(C))$,
 - * $\sigma'(t) = \sigma(t)$,
 - * minden $y \in T$ időzítésre $\sigma'(y) = \sigma(y)$ vagy $\sigma'(y) > \sigma'(t)$,
 - * ha $\sigma(t) = \min(\sigma(C))$, akkor $Ev_{\sigma'(t)}(\sigma'(C)) \subseteq Ev_{\sigma(t)}(\sigma(C))$
 - * létezik $x \in T$ időzítés, hogy $\sigma(x) = \sigma(t)$ és $\sigma'(x) > \sigma'(t)$; vagy $Ev_{\sigma'(t)}(\sigma'(C)) \subset Ev_{\sigma(t)}(\sigma(C))$
 - késleltető tranzíciók:
 - * $S' = S$,
 - * $\sigma(t) < \min(\sigma(T))$ és $\sigma(t) < \min(\sigma(C))$,
 - * $\sigma'(t) = \min\{\min(\sigma(T)), \min(\sigma(C))\}$

A fentiek alapján diszkrét tranzíciók akkor engedélyezettek, ha az aktuális idő megegyezik az időzítések és a naptáresemények esedékességi idejének minimumával. Az ilyen tranzíciók végrehajtásakor az idő nem telik. A tranzíció állíthatja az időzítéseket, de csak jövőbeli értékre. Ezen felül az eseménynaptárba kerülhetnek jövőben esedékes események. Az utolsó feltétel biztosítja, hogy minden időpontban csak véges sok esemény következhesen be. A késleltető tranzíciók az időt a legközelebb esedékes esemény bekövetkezési időpontjára állítják, ezzel garantálva az időzített automatáknál nem biztosított maximális időelőrehaladást. [12]

4.2. Naptárautomaták szemantikájának módosítása

A modellellenőrzés hatékonyabbá tétele érdekében a fenti szemantikán módosításokat vezettem be. Egyrészt a diszkrét és időzített tranzíciók összevonásával lerövidítettem a rendszer

útrészleteit, ami nemcsak az ellenpéldák értelmezését könnyíti meg, hanem az ellenőrzéshez szükséges k indukciós mélységet is jelentősen csökkenti [17]. Másrészt egy extra változó bevezetésével megoldottam, hogy az időzítéseknek mindig csak egy, a rendszer működése szempontjából éppen releváns halmazát kelljen állítani. Ezzel a modellezés a mérnöki szemléletnek jobban megfelelő módon történhet, illetve a rendszerben nem jelennek meg felesleges holtpontok a nem releváns időzítések szükségszerű állítása miatt.

A diszkrét és késleltető tranzíciók összevonásával a rendszer útrészletei lerövidíthetők [17]. Látható, hogy a rendszer egy $\langle s, \sigma \rangle$ állapotában $\sigma(t)$ értéke vagy megegyezik az időzítések értékének és a naptáreseemények időpontjának minimumával, vagy a következő állapotban egy késleltető tranzíció hatására meg fog egyezni. Így a tranzíciós reláció $\langle S, \sigma \rangle \xrightarrow{\alpha} \langle S', \sigma' \rangle$ tranzícióira vonatkozó szabályokat az alábbiak szerint módosítva az idő múlása valóban összevonható a diszkrét állapotváltozásokkal:

- minden $y \in T$ időzítésre $\sigma'(y) = \sigma(y)$ vagy $\sigma'(y) > \sigma(t)$,
- ha $\sigma(t) = \min(\sigma(C))$, akkor $Ev_{\sigma(t)}(\sigma'(C)) \subseteq Ev_{\sigma(t)}(\sigma(C))$
- létezik $x \in T$ időzítés, hogy $\sigma(x) = \sigma(t)$ és $\sigma'(x) > \sigma(t)$; vagy $Ev_{\sigma(t)}(\sigma'(C)) \subset Ev_{\sigma(t)}(\sigma(C))$
- $\sigma'(t) = \min\{\min(\sigma'(T)), \min(\sigma'(C))\}$

Látható, hogy mind az eredeti, mind a fent módosított szemantika szerint a rendszer invariáns tulajdonsága, hogy az időzítések értéke nem kisebb az aktuális időnél, hiszen minden esedékes időzítés értéke előbb-utóbb jövőbeli értékre módosul. Ez az eljárás a modellezés és ellenőrzés szempontjából azonban nem mindig szerencsés: egy időzítést akkor is jövőbeli értékre kell állítani, ha annak értéke a rendszer aktuális állapotában nem érdekes, amivel feleslegesen szaporítjuk a rendszer trajektóriáinak számát. Ezen felül még extra holtpontokat is beleviszünk a rendszer működésébe, hiszen előfordulhat, hogy egy időzítés, melyet nem megfelelő értékkel növeltünk, szándékunk ellenére esedékessé válik egy adott állapotban.

A problémák a kiküszöböléséhez az alábbi módosítást vezetjük be. Egészítsük ki a (már módosított) formalizmust egy T_{valid} változóval, mely az adott állapotban érvényes időzítések halmazát tárolja (ennek megfelelően $dom(T_{valid}) = 2^T$). Jelölje $\min(\sigma(T_{valid})) = \min\{\sigma(x_i) : x_i \in \sigma(T_{valid})\}$ az $\langle s, \sigma \rangle$ állapotban érvényes időzítések értékeinek minimumát (üres halmaz esetén pedig a $+\infty$ értéket).

A tranzíciók szemantikája módosuljon az alábbiak szerint:

- minden $y \in \sigma'(T_{valid})$ időzítésre $\sigma'(y) = \sigma(y)$ vagy $\sigma'(y) > \sigma(t)$,
- minden $y \notin \sigma(T_{valid})$ időzítésre, ahol $y \in \sigma'(T_{valid})$, érvényes, hogy $\sigma'(y) > \sigma(t)$,
- ha $\sigma(t) = \min(\sigma(C))$, akkor $Ev_{\sigma(t)}(\sigma'(C)) \subseteq Ev_{\sigma(t)}(\sigma(C))$

- létezik $x \in \sigma(T_{valid})$ időzítés, hogy $\sigma(x) = \sigma(t)$ és ha $x \in \sigma'(T_{valid})$, akkor $\sigma'(x) > \sigma'(t)$; vagy $Ev_{\sigma(t)}(\sigma'(C)) \subset Ev_{\sigma(t)}(\sigma(C))$
- $\sigma'(t) = \min\{\min(\sigma'(T_{valid})), \min(\sigma'(C))\}$

A fenti módosítással élve a szemantika tehát úgy módosul, hogy mindig csak az időzítések egy releváns T_{valid} halmaza határozza meg a következő eseményt.

4.3. Magasabb szintű formalizmus naptáratomatókhoz

A fent megadott viselkedési szemantika túl általános ahhoz, hogy minden, a szemantikának megfelelő modell véges módon leírható legyen. Ezért szükség van egy olyan, a szemantikának eleget tevő magasabb szintű formalizmus definiálására, mellyel rendszerek kellően széles köre leírható, ugyanakkor implementálható és a gyakorlatban is jól használható.

A megszorítások célja tehát elsősorban az, hogy a rendszer tömören leírható legyen. Ehhez célszerű az állapotok nem időzítéshez kapcsolódó komponenseit végessé tenni, az események időzítésébe pedig valamilyen szabályosságot meghatározni. Gondoskodni kell továbbá arról, hogy a rendszer kezdőállapotai egyszerűen megadhatóak legyenek. Alkalmazzuk tehát az alábbi megszorításokat:

- Az S halmaz véges.
- Minden kezdőállapotban az eseménynaptár üres.
- Egy lépésben legfeljebb egy üzenet helyezhető az eseménynaptárba. Az üzenet az aktuális időhöz viszonyítva valamilyen d időegységgel később lesz esedékes, ahol $d \in I$ és $I \subseteq (0, +\infty)$ egész végpontú intervallum.
- Kezdőállapotban minden időzítés valamilyen I intervallumból veszi az értékét, ahol I a fenti tulajdonságú.
- Egy adott időzítés minden kezdőállapotban vagy érvényes vagy érvénytelen.
- Amikor változik egy időzítés értéke, az új érték az aktuális időnél valamilyen d értékkel nagyobb, ahol $d \in I$ és I a fenti tulajdonságokkal rendelkező intervallum.

Látható, hogy az események (napáresemények és időzítések) állításánál az új érték mindig az aktuális időhöz képest valamilyen egész végpontú intervallumból kerül ki. Ahogy később bemutatásra kerül, ezt azért így definiáltam, hogy az időzített automatáknál megismert néhány fogalom alkalmazható legyen az alább definiált formalizmusra.

Jól látható, hogy a fenti megszorítások alkalmazásával a szemantikában meghatározott feltételek nem sérülnek. Ennek megfelelően a megszorításoknak eleget tevő rendszert az alábbi módon definiálhatjuk. Jelölje $\mathcal{I}$ a $(0, +\infty)$ valós intervallum természetes szám végpontú

részintervallumainak halmazát (ilyen intervallumnak tekintve minden nemnulla természetes számot is). Ekkor egy magasabb szintű naptáráutomata egy

$$CA = (Loc, M, T, \hookrightarrow, Loc_0, T_0, T_{valid,0}, AP, L)$$

kilenc, ahol

- Loc a vezérlési helyek véges halmaza,
- M az üzenetek véges halmaza, mely tartalmazza az $m_\emptyset$ üres üzenetet,
- T az időzítések véges halmaza
- $\hookrightarrow \subseteq Loc \times (M \cup T) \times (M \times \mathcal{I}) \times 2^{(T \times \mathcal{I})} \times 2^T \times Loc$ tranzíciós reláció,
- $Loc_0 \subseteq Loc$ a kezdő vezérlési helyek halmaza,
- $T_0 : T \rightarrow \mathcal{I}$ az időzítések lehetséges kezdőértékét meghatározó függvény,
- $T_{valid,0} \subseteq T$ az kezdetben érvényes időzítések halmaza,
- AP atomi proposíciók halmaza,
- $L : Loc \rightarrow 2^{AP}$ pedig a címkéző függvény.

A tranzíciós reláció egy $(\ell, e, \tilde{m}, \tilde{T}, T^-, \ell' \in \hookrightarrow)$ elemére bevezetjük az $\ell \xrightarrow{e:\tilde{m}, \tilde{T}, T^-} \ell'$ jelölést. Itt $e \in M \cup T$ a tranzíciót kiváltó esemény, $\tilde{m} \in M \times \mathcal{I}$ üzenetküldési akció, $\tilde{T} \subseteq T \times \mathcal{I}$ az időzítésbeállítási akciók halmaza, $T^- \subseteq T$ pedig az érvénytelenítendő időzítések halmaza. Megkötés, hogy az $m_\emptyset$ üres üzenetet nem lehet esemény, továbbá ha a tranzíciónak van egy $\tilde{x} = \langle x, I \rangle, \tilde{x} \in \tilde{T}$ időzítésbeállítási akciója, akkor $x \notin T^-$.

4.3.1. Naptáráutomaták kompozíciója

Modellezési formalizmusok esetében mindig hasznos, ha a nyelv lehetőséget biztosít rendszerek komponensenként történő modellezésére. Ennek érdekében a fent definiált formalizmushoz kompozíció műveletet adtam meg.

Legyen $CA_i = (Loc_i, M_i, T_i, \hookrightarrow_i, Loc_{i,0}, T_{i,0}, T_{i,valid,0}, AP_i, L_i)$, $i = 1, 2$ két naptáráutomata, ahol $T_1 \cap T_2 = \emptyset$ és $AP_1 \cap AP_2 = \emptyset$. Ekkor $CA_1 ||| CA_2$ az a

$$(Loc_1 \times Loc_2, M_1 \cup M_2, T_1 \cup T_2, \hookrightarrow, Loc_{0,1} \times Loc_{0,2}, T_0, T_{1,valid,0} \cup T_{2,valid,0}, AP_1 \cup AP_2, L)$$

naptáráutomata, melyre

- $T_0(x) = T_{0,1}(x)$, ha $x \in T_1$, illetve $T_0(x) = T_{0,2}(x)$, ha $x \in T_2$,

- $L(\langle \ell_1, \ell_2 \rangle) = L_1(\ell_1) \cup L_2(\ell_2)$,
- a $\hookrightarrow$ tranzíciós reláció pedig az alábbiak szerint alakul:

$$\frac{\ell_1 \xrightarrow{e:\tilde{m}, \tilde{T}, T^-} \ell'_1}{\langle \ell_1, \ell_2 \rangle \xrightarrow{e:\tilde{m}, \tilde{T}, T^-} \langle \ell'_1, \ell_2 \rangle}$$

illetve

$$\frac{\ell_2 \xrightarrow{e:\tilde{m}, \tilde{T}, T^-} \ell'_2}{\langle \ell_1, \ell_2 \rangle \xrightarrow{e:\tilde{m}, \tilde{T}, T^-} \langle \ell_1, \ell'_2 \rangle}$$

A művelet lényege tehát, hogy a két komponens a továbbiakban közös eseménynaptár felett működik. Ezzel lehetővé válik komponensek közötti üzenetváltás modellezése.

4.3.2. A magasabb szintű formalizmus szemantikája

Legyen $CA = (Loc, M, T, \hookrightarrow, Loc_0, T_0, T_{valid,0}, AP, L)$ naptáráutomata. Ekkor $TS(CA) = (Loc \times Eval(Var), Act, \longrightarrow, I', AP, L')$ az alábbiak szerint definiált tranzíciós rendszer:

- $Var = \{t, x_1, x_2, \dots, x_n, T_{valid}, C\}$, ahol $T = \{x_1, x_2, \dots, x_n\}$,
- $\langle \ell, \sigma \rangle \in I'$, ahol $\ell \in Loc_0, \sigma(t) = 0, \sigma(x_i) \in T_0(x_i), \sigma(T_{valid}) = T_{valid,0}$ és $\sigma(C) = \emptyset$,
- $Act = \mathbb{R}_{>0}$,
- $L'(\langle \ell, \sigma \rangle) = L(\ell)$,
- továbbá a $\longrightarrow$ tranzíciós reláció egy $\langle \ell, \sigma \rangle \xrightarrow{\sigma'(t) - \sigma(t)} \langle \ell', \sigma' \rangle$ tranzíciója a következő megkötéseknek tesz eleget:

- minden $\ell \xrightarrow{e:\tilde{m}, \tilde{T}, T^-} \ell'$ tranzíció esetén, ahol $\tilde{m} = \langle m, I \rangle$
 - * $\sigma'(t) = \min\{\min(\sigma'(T_{valid})), \min(\sigma'(C))\}$,
 - * ha $\langle y, J \rangle \in \tilde{T}$, akkor $\sigma'(y) = \sigma(t) + d$, ahol $d \in J$,
 - * ha $\langle y, J \rangle \in \tilde{T}$, akkor $y \in \sigma'(T_{valid})$,
 - * $\sigma'(T_{valid}) = \sigma(T_{valid}) \setminus T^-$,
- ha $e \in T$:
 - * $\sigma(e) = \sigma(t)$
 - * ha $m \neq m_\emptyset$, akkor $\sigma'(C) = \sigma(C) \cup \langle m, \sigma(t) + d \rangle$, ahol $d \in I$,
 - * ha $m = m_\emptyset$, akkor $\sigma'(C) = \sigma(C)$
- ha $e \in M, e \neq m_\emptyset$:
 - * $\langle e, \sigma(t) \rangle \in Ev_{\sigma(t)}(\sigma(C))$,

- * ha $m \neq m_\emptyset$, akkor $\sigma'(C) = \sigma(C) \setminus \{\langle e, \sigma(t) \rangle\} \cup \langle m, \sigma(t) + d \rangle$, ahol $d \in I$,
- * ha $m = m_\emptyset$, akkor $\sigma'(C) = \sigma(C) \setminus \{\langle e, \sigma(t) \rangle\}$,
- a többi változó pedig változatlan;

4.3.3. Naptáráutomaták tulajdonságai

Legyen CA naptáráutomata. Ekkor a $TS(CA)$ rendszer egy

$$\rho = s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\delta_1} s_2 \xrightarrow{\delta_2} \dots$$

végtelen végrehajtásrészlete során eltelt idő alatt az $ExecTime(\rho) = \sum_{i=0}^{\infty} \delta_i$ értéket értjük.

Ez alapján megadható a ρ végrehajtásrészlet által indukált π útrészlet $ExecTime(\pi)$ végrehajtási ideje is, így az időzített automatáknál definiált tulajdonságok (timelockmentesség, idő-divergencia, nemzénó tulajdonság) itt is értelmezhetők.

4.4. Naptáráutomaták biszimulációi

Az alábbiakban definiálok két biszimuláció relációt, amelyet később, a hatékony modellel-leőrzési algoritmusok létrehozásakor használtam.

4.4.1. Biszimuláció időeltolással

A naptáráutomaták definíciójából könnyen látható, hogy az automata egy adott állapotban lehetséges viselkedéseit az események időpontját rögzítő változók aktuális időhöz képest vett értéke határozza meg.

Ezzel egy egyszerű biszimuláció definiálható az automata állapotai között. Legyen két állapot ekvivalens ($\langle \ell, \sigma \rangle \triangleq \langle \ell', \sigma' \rangle$), ha az alábbi feltételek teljesülnek:

- $\ell = \ell'$,
- $\forall x \in T : x \in \sigma(T_{valid}) \Leftrightarrow x \in \sigma'(T_{valid})$
- minden $x \in T$ időzítésre, melyre $x \in \sigma(T_{valid}), x \in \sigma'(T_{valid}), \sigma(x) - \sigma(t) = \sigma'(x) - \sigma'(t)$,
- létezik $\pi : \sigma(C) \rightarrow \sigma'(C)$ bijekció, hogy bármely $\langle m, t_0 \rangle$ esetén, ahol $\pi(\langle m, t_0 \rangle) = \langle m', t'_0 \rangle$, teljesülnek az alábbiak:
 - $m = m'$,
 - $t_0 - \sigma(t) = t'_0 - \sigma'(t)$

A π leképezést a két állapot naptáreseményeinek összerendeléséhez definiáltuk.

4.4.2. Biszimuláció óraekvivalenciával

Láttuk, hogy időzített automaták állapotai felett biszimuláció definiálható az óraekvivalencia reláció segítségével. Ott két állapotot akkor tekintettünk ekvivalensnek, ha az állapotok vezérlési helye megegyezett, valamint a két állapotban a releváns órák (azok, amelyek kiértékelése még nem haladta meg az órára vonatkozó őrfeltételekben szereplő maximális értéket) egészrészei megegyeztek, törtreszei pedig ugyanúgy voltak sorrendezhetők.

Mivel az eredeti naptáratomata formalizmus megszorítását úgy definiáltam, hogy minden időzítés az aktuális időhöz viszonyított új értéke valamilyen egész végpontú intervallumból kerül ki, ugyanez az ötletet alkalmazható naptáratomaták esetében is. Itt az órák kiértékelésének megfelelői az események (azaz az időzítések és az eseménynaptár elemei) bekövetkezéséig hátralévő idő lesz. Egy időzítést akkor tekintünk relevánsnak, ha érvényes.

Formálisan, $\langle \ell_1, \sigma_1 \rangle \cong \langle \ell_2, \sigma_2 \rangle$ pontosan akkor, ha

- $\ell_1 = \ell_2$,
- $\forall x \in T : x \in \sigma_1(T_{valid}) \Leftrightarrow x \in \sigma_2(T_{valid})$
- bármely $x, y \in T$ időzítésre, melyekre $x \in \sigma_1(T_{valid}), x \in \sigma_2(T_{valid})$ és $y \in \sigma_1(T_{valid}), y \in \sigma_2(T_{valid})$ teljesül, érvényesek az alábbiak:
 - $\lfloor \sigma_1(x) - \sigma_1(t) \rfloor = \lfloor \sigma_2(x) - \sigma_2(t) \rfloor$
 - $\{\sigma_1(x) - \sigma_1(t)\} = 0 \Leftrightarrow \{\sigma_2(x) - \sigma_2(t)\} = 0$
 - $\{\sigma_1(x) - \sigma_1(t)\} \leq \{\sigma_1(y) - \sigma_1(t)\} \Leftrightarrow \{\sigma_2(x) - \sigma_2(t)\} \leq \{\sigma_2(y) - \sigma_2(t)\}$
- létezik $\pi : \sigma_1(C) \rightarrow \sigma_2(C)$ bijekció, hogy bármely $\langle m_1, t_1 \rangle, \langle m'_1, t'_1 \rangle \in \sigma_1(C)$ esetén, ahol $\pi(\langle m_1, t_1 \rangle) = \langle m_2, t_2 \rangle$ és $\pi(\langle m'_1, t'_1 \rangle) = \langle m'_2, t'_2 \rangle$, teljesülnek az alábbiak:
 - $m_1 = m_2$
 - $\lfloor t_1 - \sigma_1(t) \rfloor = \lfloor t_2 - \sigma_2(t) \rfloor$
 - $\{t_1 - \sigma_1(t)\} = 0 \Leftrightarrow \{t_2 - \sigma_2(t)\} = 0$
 - $\{t_1 - \sigma_1(t)\} \leq \{t'_1 - \sigma_1(t)\} \Leftrightarrow \{t_2 - \sigma_2(t)\} \leq \{t'_2 - \sigma_2(t)\}$
- ezen felül minden $x \in \sigma_1(T_{valid}), \sigma_2(T_{valid}), \langle m_1, t_1 \rangle \in \sigma_1(C), \pi(\langle m_1, t_1 \rangle) = \langle m_2, t_2 \rangle \in \sigma_2(C)$ esetén
 - $\{t_1 - \sigma_1(t)\} \leq \{\sigma_1(x) - \sigma_1(t)\} \Leftrightarrow \{t_2 - \sigma_2(t)\} \leq \{\sigma_2(x) - \sigma_2(t)\}$
 - $\{\sigma_1(x) - \sigma_1(t)\} \leq \{t_1 - \sigma_1(t)\} \Leftrightarrow \{\sigma_2(x) - \sigma_2(t)\} \leq \{t_2 - \sigma_2(t)\}$

5. fejezet

A k-indukciós modellellenőrzés módszere

A naptáramatoma formalizmus és a korábban ismertett modellellenőrzési eljárások felhasználásával definiálok azt a módszert, mellyel időzített működésű beágyazott rendszerek analizálhatók.

Ebben a fejezetben bemutatom a k-indukció alapú modellellenőrzés folyamatát. Ismertetem az általam definiált, tervezési hibák kiszűrésére használható modellezési kritériumokat, amelyek vizsgálatára statikus analízis algoritmusokat hoztam létre. Emellett kitérek a k-indukció alapú bizonyítások elméleti és gyakorlati használatára és korlátaira.

5.1. A rendszer modellezése

A modellezési folyamatot a 5.1. ábra szemlélteti. A folyamat egyes lépéseit az alábbi pontok ismertetik.

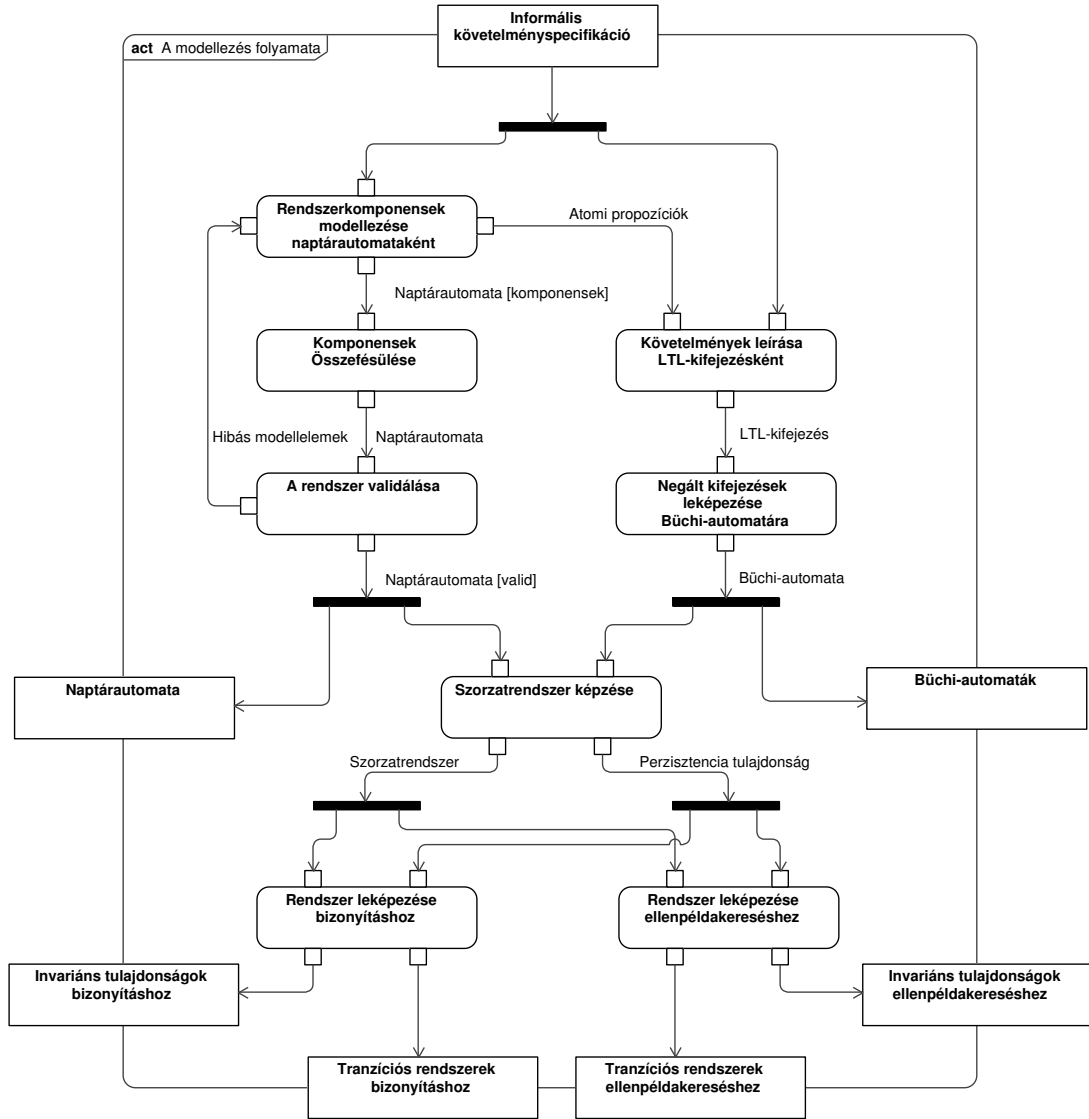
5.1.1. A rendszerkomponensek modellezése

A folyamat első lépéseként el kell készíteni a rendszer formális modelljét. Ehhez a módosított naptáramatoma szemantikához definiált magasabb szintű formalizmust használjuk.

A formalizmushoz definiált egyszerű kompozíció művelet segítségével a modell kényelmesen, komponensenként definiálható. A feladat tehát a rendszer $CA_1, CA_2, \dots, CA_n$ komponenseinek modellezése, majd ebből a teljes

$$CA = CA_1 \parallel \parallel CA_2 \parallel \parallel \dots \parallel \parallel CA_n$$

rendszer előállítás.



5.1. ábra. A modellezés folyamata

5.1.2. A rendszermodell jólformáltsági kritériumai

A magasszintű formalizmusok egyik nagy előnye, hogy a segítségükkel leírt modellek bizonyos tulajdonságai a formalizmus ismert jellegzetességei mentén könnyen, statikus vizsgálattal megfoghatók. Ez kihasználható többek között a modellek jólformáltsági ellenőrzésére, ami által a modellek helyességéről már tervezési időben részben megbizonyosodhatunk.

Az alábbiakban néhány olyan kényszerített definiálók a rendszer CA_i komponenseire, melyek teljesülése biztonságkritikus környezetben működő rendszerek tervezésekor kulcsfontosságú. A kényszerek kiértékelésének módját a következő fejezetben ismertetem.

Teljesség

Fontos, hogy a rendszer minden állapotában minden lehetséges eseményre definiálva legyen valamilyen akció.

- Ha egy komponens rendelkezik egy x időzítéssel, és egy adott állapotban érvényes, akkor az állapot valamely kimenő tranzíciója kezelje az x időzítéshez kapcsolódó eseményt.
- Ha egy komponens valamely vezérlési helyen kezel egy m üzenet fogadásához kapcsolódó eseményt, akkor minden vezérlési helyen kezelje.

Egyértelműség

A rendszer nemdeterminizmusa kiszámíthatatlan működéshez vezethet, ezért minden komponens viselkedésének egyértelműnek kell lennie:

- Ha egy állapotban az aktuális vezérlési helyen egy x időzítés érvényes, akkor minden, a vezérlési helyhez kötődő állapotban érvényes.
- Egy x időzítést minden állapotban legfeljebb egy tranzíció kezelhet.
- Ha egy komponens kezel egy m üzenetet, akkor azt kizárólagos komponensként kezeli.
- Egy, a komponens által kezelt m üzenetet minden állapotban legfeljebb egy kimenő tranzíció kezelhet.

Felesleges modellelemek kiszűrése

A modell egy felesleges eleme tervezési hibára utalhat, ezért az ilyeneket azonosítani kell:

- Ha egy komponensben egy x időzítés definiált, akkor a komponens valamely állapotában érvényes.
- Ha egy m üzenet definiált, akkor valamelyik komponens egyik tranzíciója küld ilyen üzenetet.
- Ha egy vezérlési hely szerepel egy komponensben, akkor az valamelyik kezdőállapottól a tranzíciós reláció mentén elérhető.

5.1.3. A követelmények modellezése

Amennyiben a CA rendszer komponensei előálltak, az egyes propozícióhalmazok uniója felett LTL-kifejezések fogalmazhatók meg, melyek rögzítik a modellel szemben támasztott követelményeket. Egy φ_i LTL-kifejezésből ezt követően képezhető a $\neg\varphi_i$ negált kifejezést leíró $\mathcal{A}_i$ Büchi-automata, ahogy az a 1. fejezetben említettem.

5.1.4. Leképezés tranzíciós rendszerre

Ahhoz, hogy a modellt ellenőrizni tudjuk, elő kell állítani a megfelelő tranzíciós rendszereket. Ez két lépésben történik:

- Először képezni kell a $TS(CA)$ rendszert.
- Ezt követően meg kell adni minden követelményhez a $TS_i = TS(CA) \otimes \mathcal{A}_i$ rendszert és a megfelelő $P_i = P_{pers(\mathcal{A}_i)}$ perzisztencia tulajdonságot.

A feladat ezek után az egyes P_i tulajdonságok bizonyítása. Ismétlésképpen, a P_i tulajdonság azt fejezi ki, hogy a rendszer minden lefutása során a $\neg\varphi_i$ LTL-kifejezés teljesülését monitorozó $\mathcal{A}_i$ Büchi-automata előbb-utóbb csak nemelfogadó állapotokat látogat meg, azaz a φ_i kifejezés teljesül.

5.2. A modell ellenőrzése

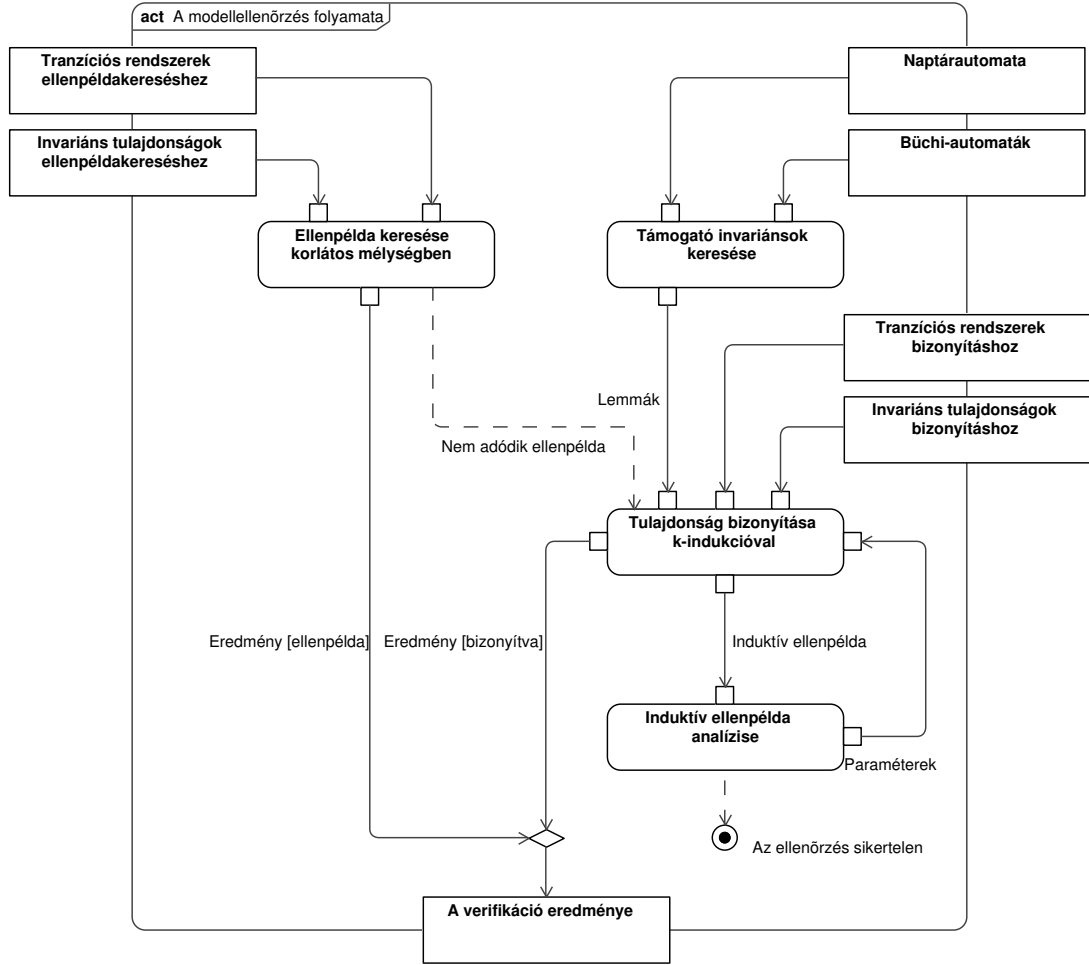
A modellellenőrzés folyamatát a 5.2. ábra írja le. Az egyes lépéseket a következő pontok részletezik.

5.2.1. Invariánsok keresése a modellben

Ahogy az a k-indukció leírásánál szerepelt, a rendszer ismert invariánsai hatékonyan tudják támogatni a modellellenőrzést. Bizonyos invariáns tulajdonságok közvetlenül a formalizmusból adódnak, így minden modellre érvényesek. Ilyen például az az egyszerű jellemző, hogy minden $\langle \ell, \sigma \rangle$ állapotban $0 \leq \sigma(t) \leq \min\{\min(\sigma'(T_{valid})), \min(\sigma'(C))\}$.

Vannak azonban modellfüggő invariánsok is. Hasonlóan ahhoz, ahogy a modell validálásánál a rendszer magasszintű modelljén egyszerűen tudunk kényszereket megfogalmazni, a komponensek néhány ilyen invariáns tulajdonsága is vizsgálható statikusan ezen a szinten. Tipikusan ilyen jellemzők a rendszer egy adott változójának lehetséges értékei valamilyen megkötés alapján:

- Az érvényes időzítések halmazának elemei egy adott vezérlési helyen: rögzített $\ell \in Loc$ vezérlési helyre, ha $\langle \ell, \sigma \rangle$ elérhető, akkor $x \in \sigma(T_{valid})$.
- Az időzítések lehetséges értékei az aktuális időhöz viszonyítva egy adott vezérlési helyen: rögzített $\ell \in Loc$ vezérlési helyre, ha $\langle \ell, \sigma \rangle$ elérhető, akkor $0 \leq \sigma(x) - \sigma(t) \leq c$ valamilyen c konstansra.
- Az eseménynaptár elemeinek esedékességi ideje az aktuális időhöz viszonyítva egy konkrét üzenettípusra: rögzített $m \in M$ üzenetre, ha $\langle \ell, \sigma \rangle$ elérhető, továbbá $(m, t_0) \in \sigma(C)$, akkor $0 \leq t_0 - \sigma(t) \leq c$ valamilyen c konstansra.



5.2. ábra. A modellellenőrzés folyamata

A fenti jellemzők a magas szintű modell által meghatározott gráfban a megfelelő vezérlési helyeket rekurzívan bejárva meghatározhatók. Az első típusú lemmára például az alábbi eljárást dolgoztam ki: a kezdő vezérlési helyen a kezdetben érvényes időzítések halmaza egyértelmű. Az időzítés egy helyen akkor is lehet érvényes, ha a helyre vezet olyan él, mely állítja (így a szemantika alapján implicit érvényesíti) az időzítést. Ezek után ha egy helyen egy időzítés érvényes lehet, és vezet ki belőle olyan él, mely nem érvényteleníti az időzítést, akkor a szomszédjában is érvényes lehet az időzítés.

A következő fejezetben bemutatott eszköztámogatás megtervezésével és implementálásával megvalósítottam a fenti típusú invariánsok a modellből történő automatikus kinyerését.

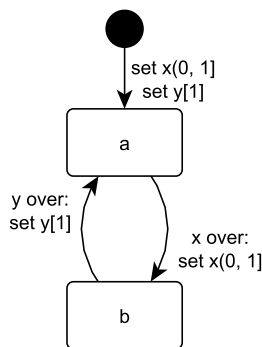
A k-indukciós modellellenőrzés szempontjából hasonlóan fontos a $TS(CA)$ rendszer és a követelményeket modellező $\mathcal{A}_i$ Büchi-automáták állapotai közötti összefüggés feltérképezése a $TS(CA) \otimes \mathcal{A}_i$ szorzat rendszerben. Az összefüggés a Büchi-automata tranzíciói és az L címkéző függvény alapján könnyen meghatározható: ha az $\mathcal{A}_i$ automata $q \in Q$ állapotban van, akkor a rendszer a szorzat definíciója alapján nyilván olyan $s \in S$ állapotban van, hogy $L(s) \in \{\sigma \in 2^{AP} : p \xrightarrow{\sigma} q\}$, hiszen a Büchi-automata csak egy ilyen szimbólum olvasásával juthatott q állapotba.

5.2.2. Ellenpéldák keresése

A korábban vázolt hurokkeresési módszerrel meg lehet kísérelni az egyes P_i tulajdonságok teljesülésére ellenpéldát találni. Mivel egy naptáráutomatában az idő folyamatos növekedése miatt nincsen hurok, valamilyen biszimuláció segítségével kell figyelni az utak záródását.

Rögtön adódik az állapotrégió-ekvivalencia alkalmazása, azonban ez túl összetett ahhoz, hogy hatékonyan lehessen alkalmazni a gyakorlatban, ezért időeltolás-ekvivalencia segítségével kerestem a rendszerben található hurkokat.

Ez ugyanazt a kérdést veti fel, amelyet időzített automaták esetében már felmerült: igaz-e, hogy ha a régió absztrakció segítségével képzett véges rendszerben található egy hurok, akkor ehhez található megfelelő hurok egyszerűen az időben vett eltolás figyelembevételével? A válasz itt is nemleges. Analóg módon konstruálható ugyanis az időzített automatáknál ismertetett ellenpéldának megfelelő naptáráutomata, mely cáfolja az állítást (5.3. ábra).



5.3. ábra. Hurokmentes naptáráutomata

Így bizonyítottam, hogy a módszer valóban csak ellenpéldák keresésére használható. Felmerül persze a kérdés, hogy nem lehet-e olyan biszimulációt megadni, mely elég durva felbontású ahhoz, hogy a hurkok kivétel nélkül megtalálhatók legyenek, és elég egyszerű ahhoz, hogy hatékonyan számolható legyen. Munkám során arra a következtetésre jutottam, hogy a k-indukciós algoritmus alkalmazásával egy ilyen reláció birtokában sem lehetne a rendszeren a feltétel teljesülését bizonyítani ezzel a módszerrel. Erre két lehetőség volna ugyanis:

- Korlátos modellellenőrzés. Ekkor a teljességhez tudni kellene egy tetszőleges véges hánycados átmérőjét, ami megadná a bizonyításhoz szükséges mélységkorlátot.
- K-indukció. Itt a probléma az, hogy a feltétel teljesülésére tetszőlegesen hosszú induktív ellenpélda konstruálható: jó ellenpélda bármilyen útrészlet az állapottérben, mely során valamelyik s_i állapotban a perzisztenciafeltétel sérül. Ilyenkor ugyanis az útrészlet első s_0 állapotában már ki lehet jelölve tetszőleges s^* állapot a hurok kezdetének, így az útrészlet az $[s_n] = [s^*]$ feltétel teljesülésekor hurokká formálódik.

5.2.3. A rendszer helyességének bizonyítása k -indukcióval

Mint láttuk, azzal, hogy a rendszerben nem találunk ellenpéldát egy adott tulajdonság teljesülésére, még nem tudjuk bizonyítani a tulajdonság teljesülését. Tudjuk ugyanakkor, hogy egy perzisztencia tulajdonság bizonyítható úgy, hogy belátjuk, hogy minden út során a feltételt sértő állapotok száma kisebb, mint valami l felső korlát.

Ez az eljárás véges rendszerek esetében mindig célra vezet, azaz mindig létezik megfelelő l korlát. Bár egy naptáratautomatából képzett tranzíciós rendszer nem véges, naptáratautomaták régió-ekvivalenciájánál láttuk, hogy tetszőleges ilyen rendszerhez létezik véges biszimiláris rendszer. Mivel a biszimuláció nyom-ekvivalencia, egy véges rendszer esetében megfelelő korlát jó a vele biszimiláris végtelen rendszer esetében is.

Amennyiben a bizonyításhoz k -indukciót használunk, felmerül a kérdés, hogy hogyan érdemes a két paramétert (az indukció k mélységét és a megengedett rossz állapotok l számát) megválasztani. Azt invariánsként előre kitűzhetjük, hogy az ellenőrzés alapjául szolgáló kiegészített rendszer számlálója legfeljebb l . Az invariáns hatékonyan támogatja az ellenőrzést, ugyanis az indukciós lépésben szűri azokat az $s_0, s_1, \dots, s_n$ útrészleteket, ahol valamilyen i indexet megelőzően az állapotokban a számláló értéke l -et meghaladja. Emiatt érdemes l értékét a lehető legkisebben tartani, és csak akkor növelni, ha a korlátos modellellenőrzés során ellenpélda adódik. A másik paraméter, k értékét természetesen az indukciós lépés sikeressége határozza meg. Az nyilvánvaló, hogy $l \leq k$, mivel ellenkező esetben a korlátos modellellenőrzésnél ellenpélda nem adódhat, és az indukciós lépésben sem tudunk a lemmával ellentmondásra jutni.

Természetesen, mint minden k -indukciós bizonyításnál, itt sem garantálható a tulajdonság sikeres bizonyítása. Előfordulhat ugyanis, hogy bár a tulajdonság igaz, a lemmák által behatárolt állapotterben tetszőlegesen hosszú induktív ellenpélda található. Így a további lemmák megválasztása - mint k -indukció esetében általában is - továbbra is kritikus a bizonyítás sikeressége szempontjából.

Jól látható, hogy a fent ismertetett két módszer jól kiegészíti egymást: míg hurkok keresésével ellenpéldákat tudunk kimutatni, addig egy megfelelő k korlát létezésének bizonyításával a tulajdonság teljesülését igazolhatjuk. A módszereim sikeres alkalmazására a 7. fejezetben az esettanulmány során mutatok példát.

6. fejezet

A modellellenőrzés eszköztámogatása

Az előző fejezetben ismertetett modellellenőrzési módszerhez megterveztem és implementáltam a folyamatot támogató eszközkészletet. Az implementációhoz az Eclipse Modeling Framework keretrendszerhez kapcsolódó technológiákat használtam, többek között a Hibatűrő Rendszerek Kutatócsoport által fejlesztett EMF IncQuery inkrementális gráfmin-taillesztő eszközt.

Az alábbi fejezet bemutatja a technológiákat és az általam elkészített eszközöket és szerepüket a modellellenőrzési folyamatban.

6.1. Modellezés szakterület-specifikus nyelven

Az előző szakaszban bemutattam, hogy az általam kidolgozott ellenőrzési folyamat kiindulópontjaként modellezni kell a rendszert és a rendszerrel szemben támasztott követelményeket. Ennek a folyamatnak a támogatására EMF alapon szakterület-specifikus nyelvet definiáltam, melyben a teljes modell alábbi részletei írhatók le:

- Az egyes CA_i naptárautomaták, melyek kompozíciójából a teljes rendszer előáll. Az automaták AP_j propozícióhalmaza megegyezik a Loc_i helyek halmazával, ennek megfelelően $L_i(\langle \ell, \sigma \rangle) = \{\ell\}$. A különböző idő dimenziójú változók tehát nem szerepelnek a komponensek megfigyelhető jellemzői között, vagyis a rendszert idő-absztrakt módon fogjuk kezelni. Ez a rendszer kvalitatív vizsgálata szempontjából nem jelent megkötést.
- Az $\mathcal{A}_i$ nemdeterminisztikus Büchi-automaták, melyek a $\neg\varphi_j$ negált követelményeket írják le. Az automaták a rendszert monitorozzák, ennek megfelelően a $\Sigma = 2^{AP}$ ábécé felett működnek, ahol $AP = \bigcup AP_i$. A metanyelvben két adott állapot közötti összes átmenet egy átmenetként definiálható, ahol az átmeneteket kiváltó szavak halmaza propozicionális logikai formulaként adható meg.

Az automaták definiálásánál nem szükséges teljes automatát megadni, a leképezési fázisnál a teljessé tétel automatikusan megtörténik. Ugyanakkor a leképezés egyszerűbb implemen-

tációja érdekében az automaták kezdőállapotaira két megkötést definiáltam:

- Az automatának egy kezdőállapota lehet. Ez nem korlátozza az általánosságot, hiszen egy új (mondjuk nemelfogadó) q'_0 kezdőállapot felvéve, majd a régi q_0^i kezdőállapotokból kimenő $q_0^i \xrightarrow{A} r$ élek alapján új $q'_0 \xrightarrow{A} r$ átmenetek definiálásával a probléma kiküszöbölhető.
- Az automata kezdőállapota kimenő éleinek általános esetben determinisztikusnak kell lenniük. Ez a megkötés sem megy az általánosság rovására. Ilyenkor fel lehet venni egy új q'_0 kezdőállapotot, amely pontosan akkor elfogadó, ha a régi kezdőállapot is az volt. Ezek után az új kezdőállapotnak fel kell venni az eredeti q_0 kezdőállapot minden egyértelmű kimenő élének megfelelő tranzíciót. Ezen felül ha adott néhány $q_0 \xrightarrow{A} r_i$ átmenet, akkor ezek alapján egy $q'_0 \xrightarrow{A} q'$ átmenetet kell felvenni, ahol az új q' állapot kimenő tranzíciói az r_i állapotok kimenő tranzícióinak megfelelőek.

A metamodellben a fentiek felül alkalmazott legfontosabb megkötés, hogy az esemény-naptár korlátos méretű. A megkötéseken kívül a metamodell a hatékony modellezés érdekében bizonyos bővítéseket is definiál. Ilyen a hierarchikus állapotok támogatása, ami lehetővé teszi az ismétlődő tranzíciók tömör megadását: egy hierarchikus állapotból kimenő tranzíciót úgy kell értelmezni, mintha minden gyerekelemére vonatkozna. Ezen felül meg lehet határozni, hogy egy adott modell folytonos vagy diszkrét időben értelmezett-e, mivel ellenpéldák kereséséhez utóbbi megközelítés is elég lehet. A metamodell naptáratomata leírására definiált részletét szemlélteti a 6.1. diagram.

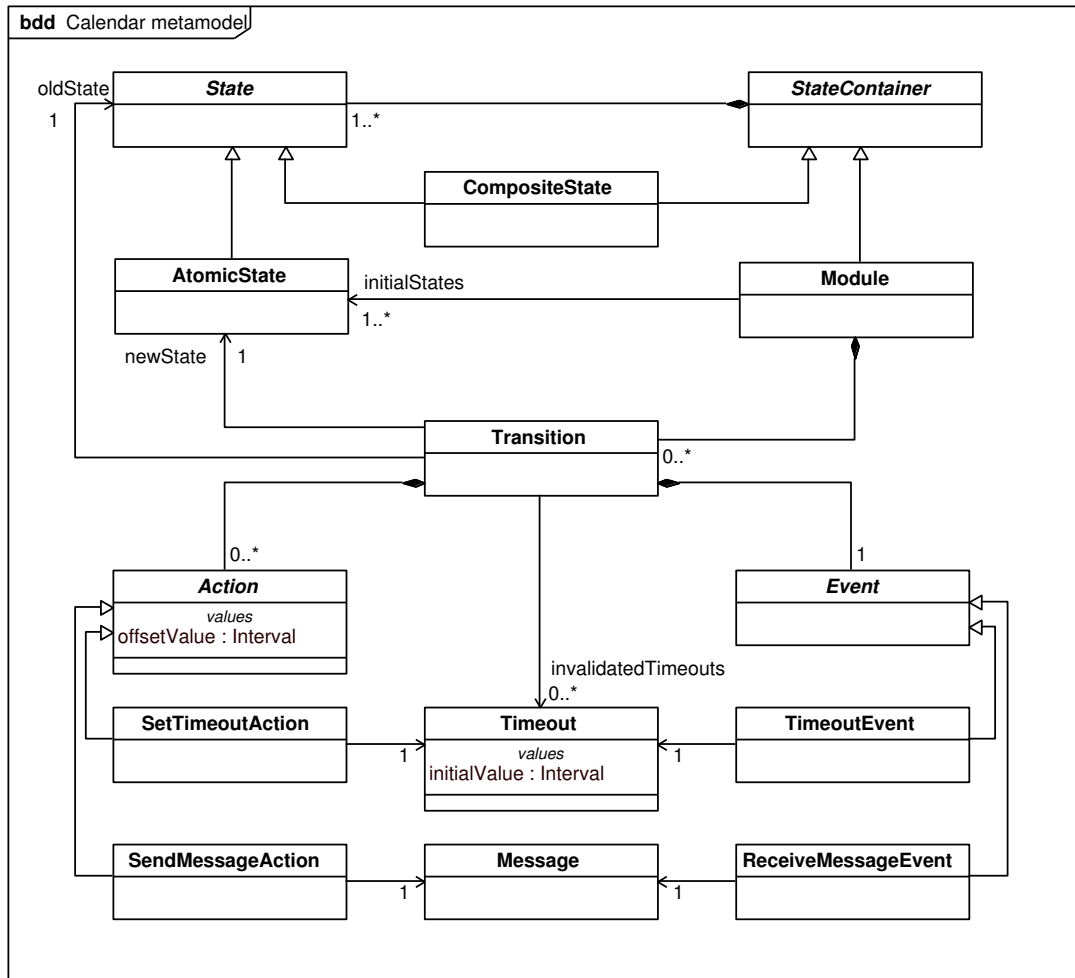
A metamodellből a keretrendszer segítségével fa nézetű szerkesztőt generáltam, amely kisebb példánymodellek szerkesztésére kiválóan alkalmas. Természetesen a metamodellhez a megfelelő technológiákat alkalmazva grafikus szerkesztőt is lehetne készíteni, azonban erre a később bemutatásra kerülő verifikációs folyamat során nem volt szükségünk.

Az előző fejezetben ismertetett folyamat a követelmények modellezésének kiindulópontjaként a követelmények LTL-nyelven történő leírását jelöli meg. A formulák Büchi-automatára történő leképezéséhez több eszköz rendelkezésre áll, többek között az ltl2ba. Az eszköz akár böngészőből is elérhető a megfelelő weboldalon [16] keresztül.

6.2. A modell statikus analízise gráfmintaillesztéssel

A módszer ismertetése során a magasszintű naptáratomata modell statikus, végrehajtás nélküli vizsgálata két ponton is szerepelt: az egyik a modell validálása volt, a másik az invariánsok keresése. A szakterület-specifikus nyelv birtokában a kitűzött feladatok mindegyike kényelmesen megoldható a példánymodell által meghatározott gráfon történő gráfmintaillesztéssel.

A statikus analízis megvalósításához az EMF IncQuery inkrementális gráfmintaillesztő eszközt használtam. Az eszköz segítségével az alábbihoz hasonló gráfminták írhatók le:



6.1. ábra. Szakterület-sepcifikus nyelv naptáratomaták modellezéséhez

```

pattern InvalidTimeoutInState(
  atomicState : AtomicState, timeout : Timeout
) = {
  Module.timeouts(module, timeout);
  Module.initialStates(module, atomicState);
  neg find TimeoutHasInitialValue(timeout);
} or {
  find FlattenedNewState(transition, atomicState);
  find TransitionInvalidatesTimeout(transition, timeout);
} or {
  find InvalidTimeoutInState(otherAtomicState, timeout);
  find FlattenedOldState(transition, otherAtomicState);
  find FlattenedNewState(transition, atomicState);
  neg find TransitionSetsTimeout(transition, timeout);
}
  
```

A fenti gráf minta azokra az állapot-időzítés párokra illeszkedik, amelyek esetében az időzítés az adott állapotban érvénytelen lehet. Ahogy látható, az ilyen feltételek rekurzívan

megfogalmazhatók: az időzítés az állapotban akkor lehet érvénytelen, ha

- az állapot kezdőállapot és az időzítés kezdetben érvénytelen,
- ha az időzítést valamelyik befelé jövő tranzíció érvényteleníti,
- ha az időzítés érvénytelen egy olyan állapotban, ahonnan van befelé jövő, az időzítést nem állító tranzíció.

Az IncQuery eszköz lehetőséget biztosít arra, hogy a mintákat validációs kényszerként annotáljuk. Ennek hatására kigenerálódnak azok az osztályok, melyek segítségével a validáció a már említett fa nézetű szerkesztőbe integrálva elvégezhető.

6.3. A modell leképezése kódgenerálással

A modellellenőrzést az SRI International által fejlesztett Symbolic Analysis Laboratory (SAL) keretrendszerrel hajtottam végre. Választásom az alábbi okokból esett az eszközre:

- A SAL hatékony, state-of-the-art modellellenőrző keretrendszer.
- Az eszközzel több jelentős rendszert (például TTEthernet [22][21]) is sikeresen verifikáltak.
- Időzített rendszerek modellezésével és verifikációjával kapcsolatban több hasznos, az eszközhöz kapcsolódó publikáció jelent meg.[12][11][17]

Az ellenőrzés megvalósításához a szemantika alapján le kellett képezni a rendszert leíró példánymodellt az eszköz bemeneti nyelvére. Az ellenőrzési folyamat automatizálása érdekében a leképezést egy, az XTend2 technológia segítségével létrehozott kódgenerátorral valósítottam meg mely a rendszert leíró példánymodellből kigenerálja az ellenőrzendő rendszert és követelményeit leíró SAL-kódot.

6.3.1. Rövid betekintés a SAL nyelvbe

A SAL nyelv lehetővé teszi tranzíciós rendszerek kompakt leírását. A rendszer közös változókon keresztül kommunikáló modulok szinkron és aszinkron kompozíciójaként definiálható. Egy adott modul változóit a modul őrfeltétellel ellátott tranzíciói manipulálják. A nyelv lehetőséget ad ezen felül a követelmények LTL nyelven történő specifikálására.

A nyelv leíróerejébe nyújt betekintést az alábbi egyszerű példa [9].

```

short: CONTEXT =
BEGIN
  State: TYPE = {ready, busy};

main: MODULE =
BEGIN
  INPUT request : BOOLEAN
  OUTPUT state : State
  INITIALIZATION
    state = ready
  TRANSITION
    state' IN IF (state = ready) AND request THEN
      {busy}
    ELSE
      {ready, busy}          %nemdeterminisztikus választás
    ENDIF
END;

th: THEOREM main |- G(request => F(state = busy));
END

```

A nyelv e fenti példában szereplő konstrukciókon felül még számos, a modellezést kényelmessé tevő nyelvi elemmel rendelkezik, például strukturált (akár rekurzív) adattípusok, elsőrendű logikai kifejezések őrfeltételekben, aritmetikai és logikai operátorok, függvények.

6.3.2. Kódgenerálás

A kódgenerátor feladata a rendszert leíró példánymodellből a megfelelő SAL-specifikáció előállítását. A kigenerált implementáció moduláris felépítésben tartalmazza az eredeti rendszer specifikációját, valamint azoknak a kiegészítő moduloknak a leírását, melyekből a rendszerrel szinkron vagy aszinkron szorzatot képezve előáll a modellellenőrzés módszerénél ismertetett, az ellenőrzéshez áttanszformált rendszer.

Mivel az eredeti naptáráutomata formalizmust a SAL eszközzel történő hatékony modellellenőrzéshez találták ki, a naptáráutomaták generálásához rendelkezésemre állt egy konkrét naptáráutomata modell SAL-nyelvű specifikációja [11]. Ehhez képest azonban számos módosítást alkalmaztam. Ezek közül a legfontosabbak az alábbiak:

- A publikált naptáráutomata modellben az eseménynaptár legfeljebb egy eseményt tartalmazhat. A metamodel azonban lehetővé teszi ennél nagyobb kapacitású eseménynaptárak definiálását is, így ennek leképezéséhez egyéni megoldást kellett alkalmazni. A problémát egy eseménynaptár modul bevezetésével oldottam meg. Az üzenetek kezelése így a modullal történő, speciális őrfeltételekkel biztosított szinkronizációval oldható meg.

- Ahogy a szemantika módosításánál szerepelt, a k -indukció hatékonyabbá tétele érdekében a diszkrét és késleltető tranzíciókat összevontan kezeltem. A problémát az ötletet bevezető cikkben [17] bemutatott példa alapján oldottam meg, azonban az ott megadottnál strukturáltabb és jóval hatékonyabb implementációt hoztam létre. Ezt egy speciális óramodul implementálásával értem el.

Az ennek megfelelően a modellben megadott naptárautomata-komponensekből generált modulok aszinkron kompozíciójaként előáll a komponensek szorzatának szemantikájaként definiált $TS(CA)$ rendszer.

Az LTL-modellellenőrzés perzisztenciavizsgálatra történő visszavezetéshez ezt követően elő kell állítani a követelmények alapján konstruált Büchi-automaták SAL-nyelvű leírását. A szakterület-specifikus nyelvből az ezeket megvalósító SAL-modulok is automatikusan generálhatóak. A generálandó kódot úgy definiáltam, hogy a leképezés során előálló modulok és a rendszer szinkron kompozíciója $TS(CA) \otimes \mathcal{A}_i$ rendszert specifikálja.

Az ellenőrzéshez az így kapott rendszerből képezni kell a (biszimiláris rendszerben történő) hurokkeresésre és a lefutás során érintett elfogadó állapotok számlálására alkalmas rendszert. Ez a kódgenerátor kimenetében - az előző leképezéséhez hasonló módon - modulárisan, a rendszer és az azt monitorozó modulok szinkron kompozíciójával valósul meg. A kódgenerátor ezen felül előállítja a rendszer ellenőrzését lehetővé tevő, a teljes rendszerre vonatkozó invariáns tulajdonságok SAL-nyelvű leírását.

6.4. A modell ellenőrzése

A módszerhez készített eszköztámogatás lehetővé teszi időzített rendszerek magas szintű modellezését. Az elkészített modellen kiértékelhetők a definiált gráfminták által megvalósított jólformáltsági kényszerek. Ezen felül a modellből az ellenőrzést nagy mértékben támogató invariáns tulajdonságok automatikus kinyerésére is lehetséges. A validált modellből automatikusan SAL-kód generálható, melyet a talált invariánsokkal a `sal-inf-bmc` modellellenőrző eszköznek beadva a modellellenőrzés elvégezhető.

A következő fejezet egy ipari esettanulmány segítségével demonstrálja a modellellenőrzési módszer és a fenti eszközök használhatóságát.

7. fejezet

A ProSigma SCAN protokoll vizsgálata

A ProSigma a Prolan Zrt. magas rendelkezésre állású, SIL4-es biztonságintegritási szintet megvalósító, még fejlesztés alatt álló rendszere, melynek feladata egy vasúti biztosítóberendezés moduljai közötti tévesztésmentes jelátvitel biztosítása. Az alábbi fejezet nagy vonalakban bemutatja a rendszer architektúráját, majd ismerteti a rendszer kapcsolatkezelési funkciójának modellezése és verifikációja során elért eredményeimet, melyeket a korábbi fejezetben bemutatott elméleti apparátus, valamint az elkészített eszközkészlet tett lehetővé.

7.1. A ProSigma rendszer bemutatása

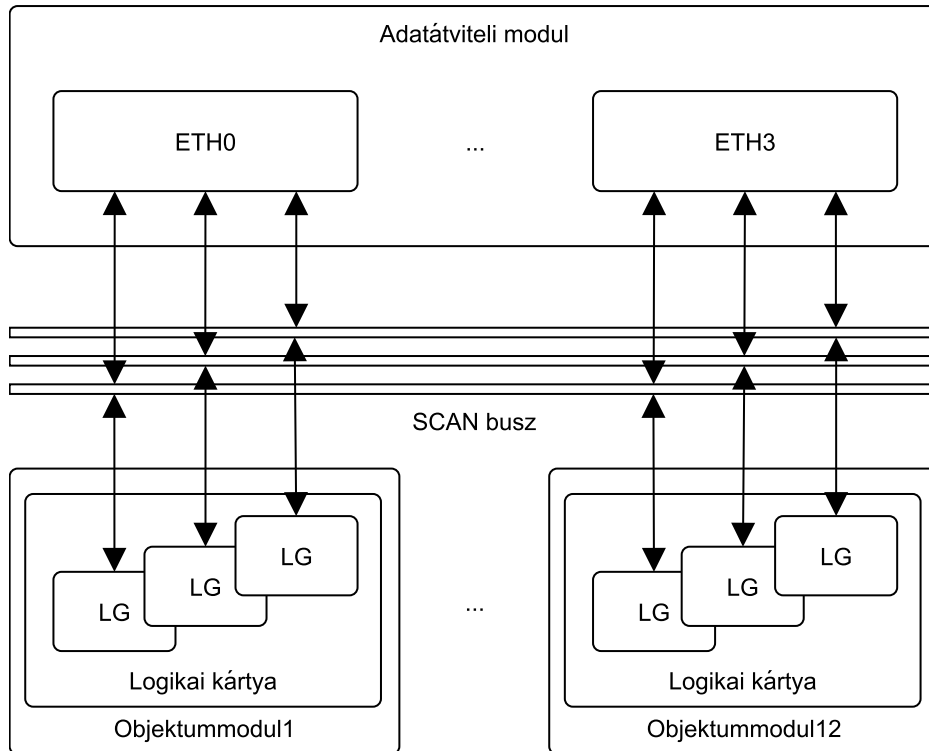
A rendszer ProSigma berendezésekből épül fel, melyek között a kommunikáció IP alapú hálózaton történik. Egy berendezés a kommunikációt biztosító adatátviteli modult (azon belül ETH egységeket) és a terepi vagy irányítási berendezéseket (például váltó, jelző) kezelő objektummodulokat (azon belül LG egységeket) tartalmaz. A berendezés felépítését a 7.1. ábra szemlélteti. A diagramon jól megfigyelhető a berendezés redundáns felépítése.

A modulok között a kommunikáció a SCAN protokoll szerint zajlik. Az üzenetek egy része az adatátviteli modul és az objektum modulok között továbbítódik egy ProSigma berendezésen belül. Az üzenetek másik része az adatátviteli modulon keresztül jut el a célul megadott IP című ProSigma berendezés SCAN-buszára.

7.1.1. A SCAN protokoll kapcsolatkezelése

A kapcsolatkezelés során a terepi és irányítási funkciót ellátó oldalak (konkrétan egy-egy LG egység) között létesül kapcsolat. A kapcsolatkezelés két funkciót foglal magába:

- Kapcsolatfelvétel: a két oldal közötti kiépül a kapcsolat.



7.1. ábra. A ProSigma berendezés felépítése

- Objektumállapot-átvétel: az oldalak elküldik a túloldalra az állapotukat. Ez a kapcsolat fenntartását is biztosítja.

A kapcsolatkezelés során egy adott oldalon nyilván kell tartani a túloldallal kialakított kapcsolatok állapotát. Egy kapcsolatot az alábbiak határoznak meg:

- Az oldal párja (a terepi oldalnak legfeljebb négy párja, az irányítási oldalnak egy párja lehet).
- A kapcsolatot biztosító ETH egység.

A kapcsolat akkor él egy oldallal, ha azzal legalább egy ETH egységen keresztül él.

Általános kapcsolatkezelés

A kapcsolatkezelés OBJ üzenetekkel (*OBJ1*, *OBJ2*, *OBJUP*, *OBJDOWN*) történik. A kapcsolatkezelésben résztvevő OBJ üzenetek feldolgozása kétszintű. Az első szinten azt kell vizsgálni, hogy az üzenetet az időbélyegei alapján kapcsolatfelvételre, illetve kapcsolatfenntartásra elfogadható-e. A második szinten már az objektumállapot elfogadhatóságának vizsgálata történik (újabb-e az objektumállapot az utoljára elfogadottól, megfelelő oldaltól jött-e, stb.).

Kapcsolatfelvétel

A kapcsolatfelvételnek két fázisa van: az első a kapcsolatot kezdeményező, a második a kapcsolatot nyugtázó. A terepi oldalon ez az *OBJ1* küldésével és az *OBJ2* vételével, az irányítási oldalon pedig az *OBJ2* küldésével és az *OBJUP* vételével történik.

A terepi oldal LG egységeinek minden irányítási oldallal minden ETH egységen keresztül, egymástól függetlenül fel kell venni a kapcsolatot.

Objektumállapot-átvitel

Az objektumállapotok küldése a kapcsolatfelvételt követően két üzenet (*OBJUP*, *OBJDOWN*) segítségével történik. Az objektumállapotok küldése és fogadása egy ETH csatornán keresztül csak azután lehetséges, miután a terepi és az irányítási oldal LG egységei az adott ETH csatornán keresztül felvették egymással a kapcsolatot. Az objektumállapot küldését a terepi oldal ETH egységeinek kell kezdeményeznie a hozzájuk tartozó LG egységeknek küldött *TIMESYNC* üzenetekkel.

A rendszer követelményspecifikációja [1] alapján, a tervezőkkel egyeztetve elkészítettem a kapcsolatkezelés modelljét. A kapcsolatkezelés részleteit az elkészült modellen keresztül mutatom be.

7.2. A kapcsolatkezelés modellje

A modell egy terepi és egy irányítási LG modul által nyilvántartott kapcsolatot modellez. A modellben az üzenetek minden esetben e két egység valamelyikétől származnak, és legfeljebb *TPropMax* ideig terjednek. Amennyiben a terjedési késleltetések a többi időparaméterhez viszonyítva ésszerű korláton belül maradnak, a modellben nem fordulhat elő olyan eset, hogy egy üzenet (vagy a benne tárolt objektumállapot) nem elfogadható. Emiatt az üzenetek elfogadhatóságának vizsgálatát, és ennek megfelelően az üzenetek időbélyegeit a modell nem tartalmazza.

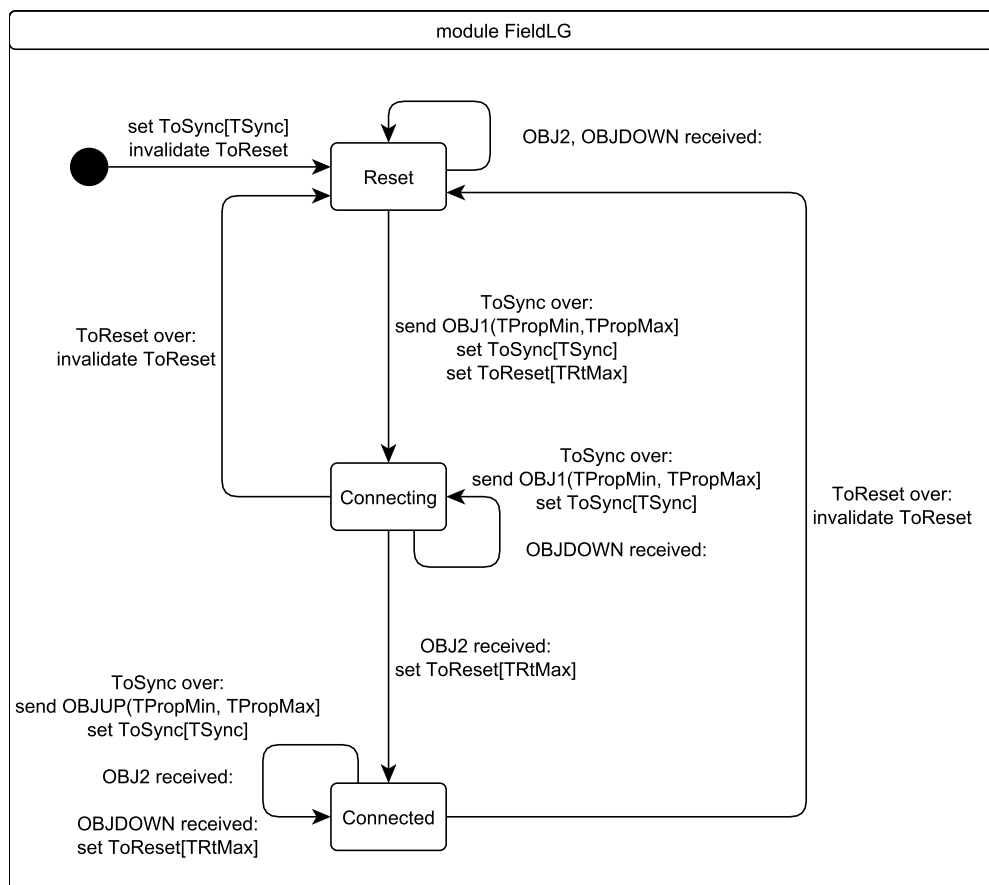
A modell ellenőrzéséhez az időparaméterek értékét az alábbiak szerint rögzítettem:

Paraméter neve	Leírás	Érték
<i>TPropMin</i>	Minimális üzenetkésleltetés	0
<i>TPropMax</i>	Maximális üzenetkésleltetés	1
<i>TSync</i>	Szinkronizációs üzenetek érkezésének periódusideje	3
<i>TRtMax</i>	Maximálisan tolerált körbefordulási idő	6

Az egységek viselkedését az alábbi pontok részletezik.

7.2.1. Kapcsolatkezelés a terepi oldalon

A terepi oldal kapcsolatkezelését modellező magasabb szintű naptárautomatát a 7.2. ábra szemlélteti.



7.2. ábra. A terepi LG kapcsolatkezelése

A rendszer indulásakor a terepi oldalon a kapcsolat *Reset* állapotban van. Ahogy az általános leírásban szerepelt, a kapcsolatfelvételi folyamatot a terepi ETH egység az LG egységnek küldött *TIMESYNC* szinkronizációs üzenete indítja. Az egyszerűség kedvéért ezt az eseményt a terepi LG lokális, konstans, *TSync* időközönként bekövetkező időzítési eseményként modelleztem (ezt az ábrán a *ToSync* eseményhez kapcsolódó tranzíciók fejezik ki). Az esemény hatására a terepi modul *OBJ1* üzenettel kezdeményezi az irányítási oldalnál a kapcsolatfelvételt, majd a kapcsolatot *Connecting* állapotba állítja.

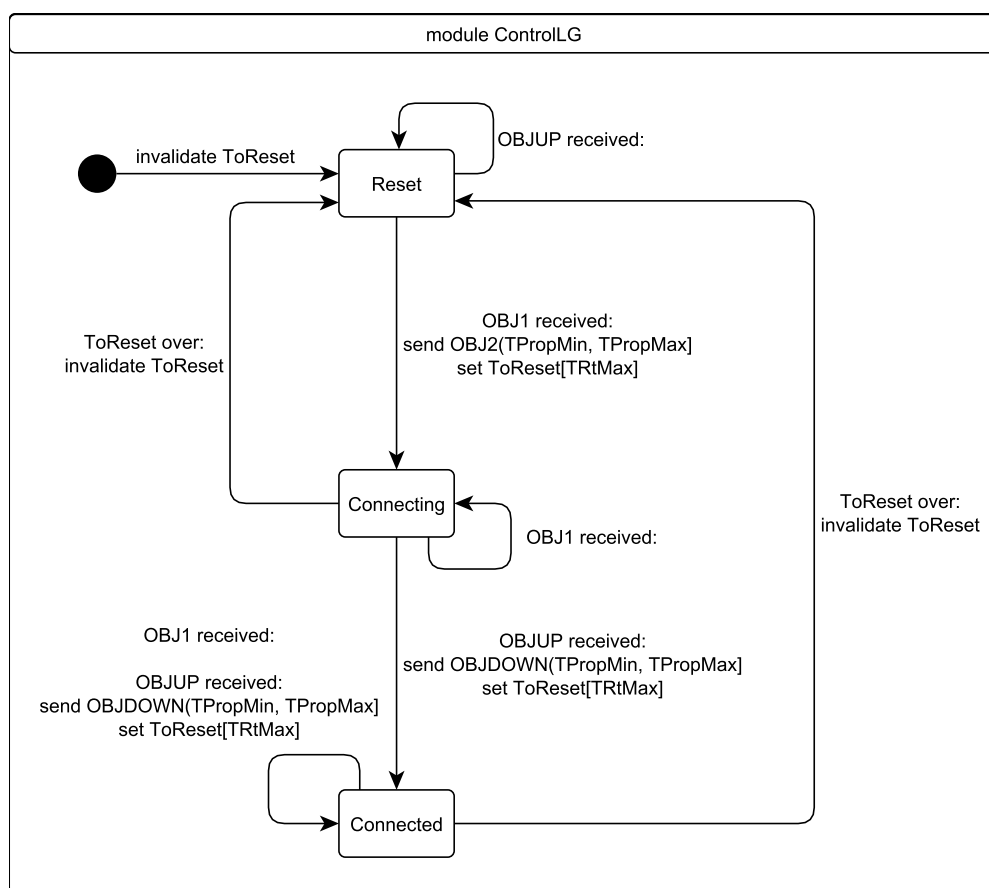
Connecting állapotban az LG egység *TRtMax* ideig várakozik a kapcsolatfelvételt nyugtázó *OBJ2* üzenetre. Ha ez idő alatt nem érkezik ilyen üzenet, a kapcsolat alaphelyzetbe áll. Az üzenet időben történő megérkezésekor a kapcsolat *Connected* állapotba megy át. A terepi oldalon ekkor elkezdődik az objektumállapot-átviteli fázis.

Az objektumátvitelt szintén a terepi ETH egység szinkronizációs üzenetei ütemezik. Ennek hatására az LG egység *OBJUP* üzenetben elküldi állapotát az irányítás oldalnak. Amennyiben az utoljára beérkezett *OBJDOWN* üzenet óta több, mint *TRtMax* idő telt el, a kapcsolat bomlik terepi oldalon.

7.2.2. Kapcsolatkezelés az irányítási oldalon

Az irányítási oldal kapcsolatkezelését a 7.3. ábra mutatja be.

A terepi oldalhoz hasonlóan az irányítási oldalon a kapcsolat kezdetben *Reset* állapotban van. A kapcsolatfelvétel az *OBJ1* üzenet fogadásával kezdődik, melynek hatására az egység *OBJ2* üzenetet küld és átmegy felvétel állapotba. *Connecting* állapotban - hasonlóan a terepi oldalhoz - a kapcsolat legfeljebb *TRtMax* ideig tartózkodhat. Amennyiben ez idő alatt nem érkezik *OBJUP* üzenet, a kapcsolat visszaáll *Reset* állapotba. Az időben érkező *OBJUP* hatására az egység *OBJDOWN* objektumállapot-üzenetet küld, és a kapcsolat *Connected* állapotba megy át. Ha két *OBJUP* üzenet között több, mint *TRtMax* idő telik el, a kapcsolat *Reset* állapotba áll.



7.3. ábra. Az irányítási LG kapcsolatkezelése

7.2.3. Validáció

A korábban bemutatott eszköz segítségével a fent bemutatott példánymodellen végrehajtottam a validációt. A fenti specifikáció kielégíti a definiált teljességi, egyértelműségi és egyéb kényszereket. Ezek alapján már a modellezési fázisban kijelenthető, hogy a kapcsolatkezelés specifikációja megfelel a biztonságkritikus környezetben elvárt legalapvetőbb jólformáltsági szempontoknak.

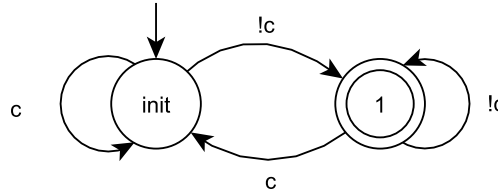
7.3. A kapcsolatkezelés modelljének ellenőrzése

7.3.1. A hibamentes rendszer vizsgálata

A rendszermodell birtokában már modellezhetőek a rendszerrel szemben támasztott követelmények. A modell alapján nyilvánvaló, hogy a kapcsolatkezelés protokollja akkor tekinthető jónak, ha a kapcsolat mindkét oldalon előbb-utóbb eljut úgy *Connected* állapotba, hogy a kapcsolat a későbbiek során sem bomlik fel. Ennek megfelelően a rendszeren a $\varphi = \Diamond \Box c$ LTL-formula teljesülését vizsgáltam, ahol

$$c = \text{FieldLG.Connected} \wedge \text{ControlLG.Connected}$$

a rendszer csatlakozott állapotát kifejező proposíció. A kifejezés negáltja a $\neg \varphi = \Box \Diamond \neg c$ feltétel, melyet a 7.4. ábrán szemléltetett Büchi-automata ír le.



7.4. ábra. A $\text{GF} \neg c$ tulajdonságot leíró Büchi-automata

7.3.2. Invariánsok keresése

A sikeres ellenőrzéshez meg kell határozni a rendszer néhány invariáns tulajdonságát. Az ismertetett eszköz kimenete alapján a naptárazomata modellből az alábbi táblázatban szereplő tulajdonságok adódnak. A táblázatban szerepel az adott invariáns bizonyításához szükséges indukciós hossz és az ellenőrzés futásideje. Az egyes invariánsok a minden naptárazomata jellemző $0 \leq \sigma(t) \leq \min\{\min(\sigma'(T_{\text{valid}})), \min(\sigma'(C))\}$ lemmákhoz relatívan kerültek meghatározásra, ehhez viszonyítva az összes invariáns induktív (azaz $k = 1$).

Invariáns	k	Idő (s)
$\text{FieldLG.Reset} \Rightarrow \text{FieldLG.ToReset}$ érvénytelen	1	0.72
$\text{FieldLG.Connecting} \vee \text{FieldLG.Connected} \Rightarrow 0 \leq \text{FieldLG.ToReset} - \text{time} \leq \text{TRtMax}$	1	0.72
$0 \leq \text{FieldLG.ToSync} - \text{time} \leq \text{TSync}$	1	0.72
$\text{ControlLG.Reset} \Rightarrow \text{ControlLG.ToReset}$ érvénytelen	1	0.72
$\text{ControlLG.Connecting} \vee \text{ControlLG.Connected} \Rightarrow 0 \leq \text{ControlLG.ToReset} - \text{time} \leq \text{TRtMax}$	1	0.72

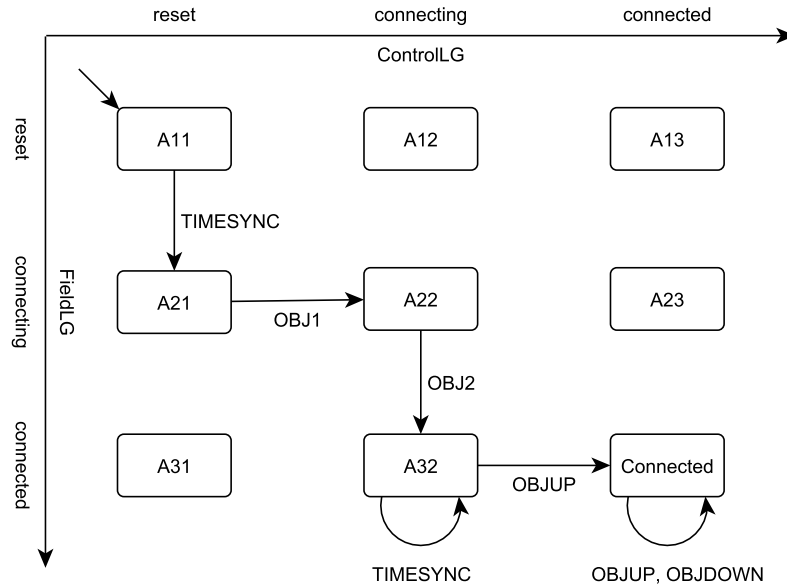
Ezen felül a fent szereplő Büchi-automatára felírható az $init \Leftrightarrow c$ invariáns.

7.3.3. A rendszer helyességének bizonyítása absztrakció segítségével

A sikeres ellenőrzés érdekében további, összetettebb invariánsokat határoztam meg. Ezt egy absztrakciós technika [12] alkalmazásával értem el. A technika lényege, hogy egy monitorral megadunk egy olyan rendszert, mely leírja, hogy ha a rendszer aktuális állapotára egy Φ_i propozíció jellemző, akkor a következő állapotban teljesül a Ψ_i propozíció. Formálisan egy $\square \left(\bigwedge_i (\Phi_i \Rightarrow \bigcirc \Psi_i) \right)$ tulajdonságról van szó, ahol Φ_i és Ψ_i az egyes állapotokra vonatkoztatott invariáns feltételek.

A módszer természetesen általánosabban is megfogalmazható, hiszen az ötlet tulajdonképpen az, hogy lemmaként nem egy, az eredeti modellen megfogalmazott invariáns tulajdonságot adunk be a modellellenőrzőnek, hanem egy monitorral és az arra megfogalmazott invariánssal leírt biztonságossági tulajdonságot. Ez a meglévő eszköztámogatással könnyen megvalósítható, hiszen a kívánt biztonságossági tulajdonsághoz megalkotva a negált feltételt leíró Büchi-automatát, majd a kódot kigenerálva egy megfelelő monitort kapunk. Ebből következik, hogy egy adott biztonságossági tulajdonsághoz tartozó, sikeresen bizonyított feltétel alkalmazható a későbbi bizonyítási lépések támogatására. Ehhez természetesen érdemes a tulajdonsághoz tartozó automatán minél hasznosabb lemmákat megfogalmazni.

Ezek alapján elkészítettem a rendszerhez egy egyszerű absztrakciót. Ezt a 7.5. ábra szemlélteti. Az ábrán az absztrakt állapotok egy koordinátarendszerben vannak elhelyezve, ahol a tengelyek a terepi, illetve irányítási oldalon nyilvántartott kapcsolat állapotát fejezik ki.

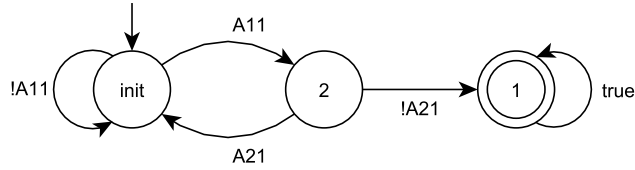


7.5. ábra. A hibamentes rendszer absztrakciója

A fenti absztrakciót több biztonságossági követelmény megadásával definiáltam. Így például egy követelmény, hogy az A_{11} absztrakt állapot rákövetkezője mindig az A_{21} állapot, egy másik, hogy az A_{12} absztrakt állapot nem elérhető. Ennek a megoldásnak az előnye,

hogy az absztrakció már bizonyított mozaikjai felhasználhatók a többi rész bizonyításához.

Az egyes mozaikok bizonyítása a megfelelő invariánsok birtokában a 3. fejezetben vázolt számlálós megoldással történik. Biztonságossági tulajdonságokról van szó, tehát a Büchi-automaták a rossz prefixumokat ismerik fel. Ez akár determinisztikus véges automatával is megoldható, így, ha a Büchi-automaták kellően tömörek (azaz nincsenek benne felesleges elfogadó állapotok), a végrehajtás során megengedett elfogadó állapotok $l - 1$ száma nullának is választható. Az absztrakció darabjait az alábbihoz hasonló automaták írják le:



7.6. ábra. $A \neg G(A_{11} \Rightarrow X A_{21})$
tulajdonságot leíró büchi-automata

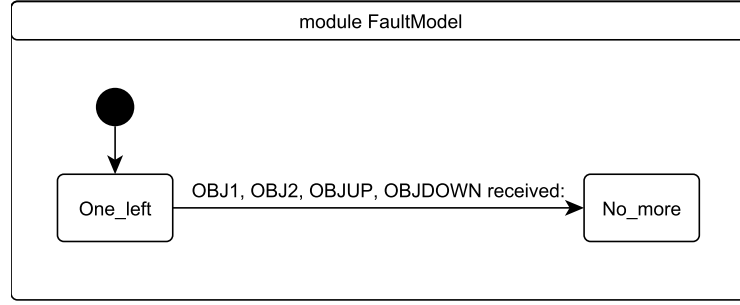
Az egyes tulajdonságokat külön-külön belátva, majd invariánsként (az automatákra vonatkozó kiegészítő invariánsokkal együtt) a modellellenőrzőnek megadva $a\varphi = \Diamond \Box c$ követelmény könnyen bizonyítható.

Tulajdonság	k	l	Idő (s)
$\Box(A_{11} \Rightarrow \bigcirc A_{21})$	18	1	8.93
$\Box(A_{21} \Rightarrow \bigcirc A_{22})$	18	1	5.66
$\Box(A_{22} \Rightarrow \bigcirc A_{32})$	19	1	4.28
$\Box(A_{32} \Rightarrow \bigcirc(A_{32} \vee c))$	6	1	1.26
$\Box(c \Rightarrow \bigcirc c)$	7	1	1.39
$\Box(\neg A_{12})$	16	1	3.02
$\Box(\neg A_{13})$	8	1	1.48
$\Box(\neg A_{23})$	9	1	1.58
$\Box(\neg A_{31})$	6	1	1.25
$\Diamond \Box c$	6	5	1.24

7.3.4. A rendszer kiegészítése egyszerű hibamodellel

Mivel a rendszernek biztonságkritikus környezetben kell üzemelnie, nem elég a hibamentes forgatókönyvekre garanciát adni. Ezért a rendszer modelljét egyszerű, egy üzenetvesztést megengedő hibamodellel egészítettem ki. Ezt a 7.7. ábra mutatja be.

A szemantika értelmében a modul a csatornából kivehet egy OBJ-üzenetet, így a modul egyszeres üzenetvesztést modellez.



7.7. ábra. Egyszerű, üzenetvesztéses hibamodell

Az így kiegészített modellen az ellenpéldakeresési eljárást futtattam. Az ellenőrzés hatékonysága érdekében a keresést diszkrét időre megszorított modellen hajtottam végre. A modellellenőrző $k = 18$ mélységben a 7.8. ábrán szereplő hurkot találta. Az ábrán a időzí-tési eseményeket jelképező nyilak állása az események feldolgozási sorrendjét fejezi ki.

Az ellenőrzés kimutatta, hogy a kapcsolat kiépülését a megfelelő eseménysorrendezés mellett a vártakkal ellentétben már egy üzenet kiesése is meg tudja akadályozni. Az egyes üzenetek késleltetésingadozása miatt ilyen szituáció a gyakorlatban is elképzelhető.

Az ellenpéldában a problémát az okozza, hogy a terepi oldal csak akkor kap *OBJ2* üzenetet, ha ott a kapcsolat bontva állapotban van. Ez pedig arra vezethető vissza, hogy az *OBJ1* üzenetek többségét az irányítási oldal felvétel állapotban fogadja, így az üzenetre nem reagál.

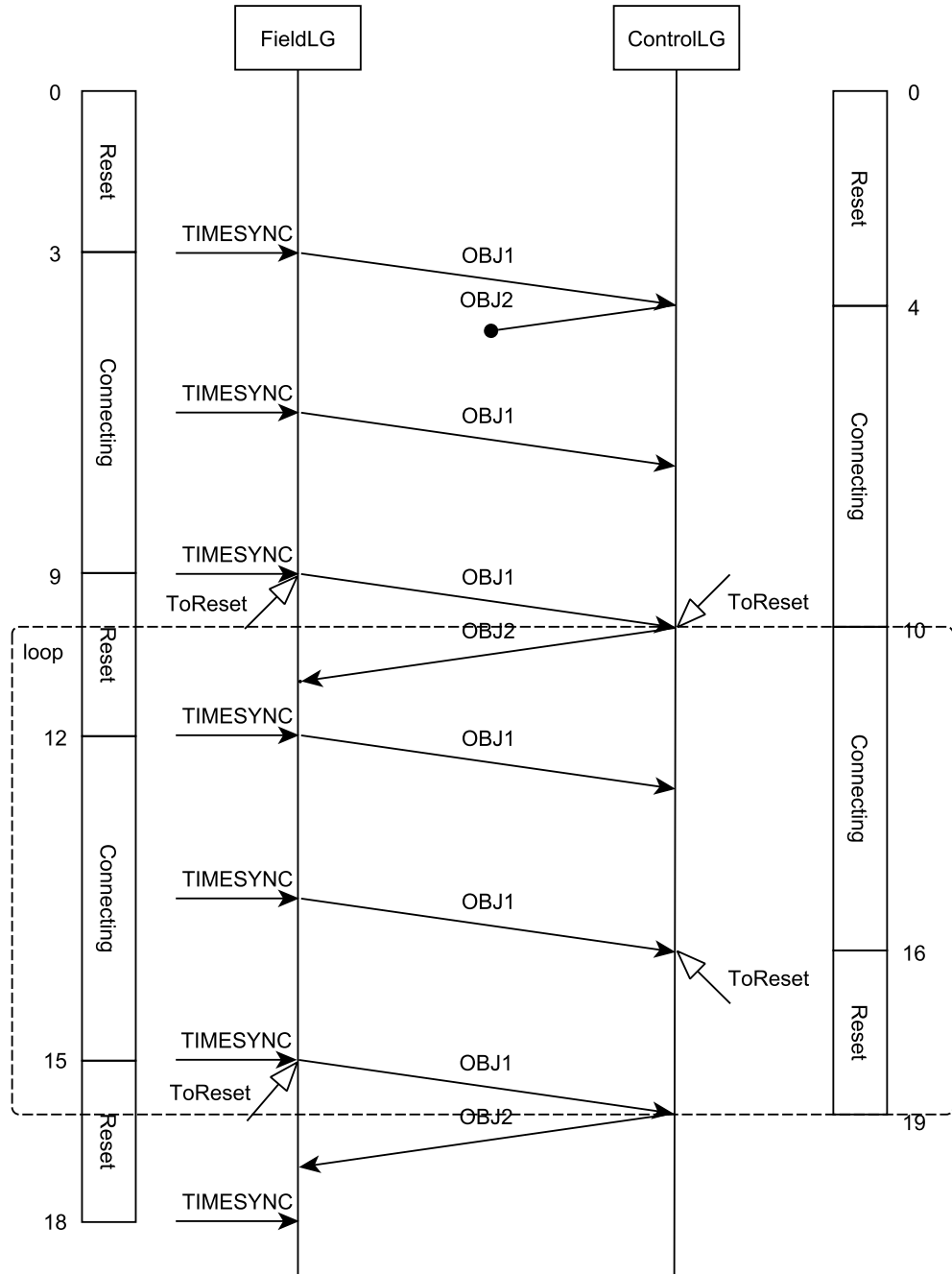
Tulajdonság	k	Idő (s)
$\Diamond \Box c$ tulajdonságra ellenpélda	18	4.93

7.4. A rendszer javítása

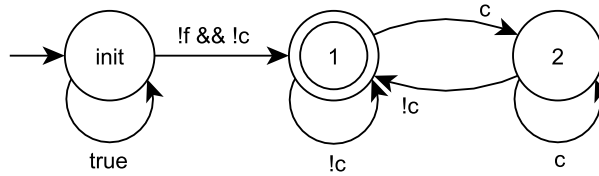
A fenti probléma javítására adódik az ötlet, hogy az irányítási oldal az *OBJ1* üzenetekre Connecting állapotban is válaszoljon *OBJ2*-vel. Annak eldöntésére, hogy ez megoldja-e a problémát, az ellenőrzést meg kell ismételni.

Mivel az előbbi $k = 18$, illetve az ennél valamivel nagyobb mélységű ellenőrzésekre nem adódott ellenpélda, megkíséreltem a tulajdonság bizonyítását. Mivel a hibamentes esetre a tulajdonság bizonyított (a módosított tranzíció az absztrakció alapján ekkor nem tüzel soha), elég a $\Box(\neg f \wedge \neg c \Rightarrow \Diamond \Box c)$ feltételt vizsgálni, ahol $f = \text{FaultModel.One_left}$, azaz arra vonatkozó proposíció, hogy még történhet üzenetvesztés. A tulajdonság negáltját a 7.9. ábrán szereplő Büchi-automata fejezi ki.

Az automata az ellenőrzés szempontjából két szerencsés tulajdonsággal bír. Egyrészt az állapotokra a korábbiakhoz hasonló módon invariánsok írhatók fel. Másrészt a számlálós bizonyítás segítségéhez felírható az a lemma, hogy a számláló értéke a kiindulási érték pontosan akkor, ha az automata *init* állapotban van.



7.8. ábra. Ellenpéllda a rendszer helyességére



7.9. ábra. A $\neg G(\neg c \wedge \neg f \Rightarrow FGc)$ tulajdonságot leíró Büchi-automata

A lemmák segítségével sikerült a tulajdonságot bizonyítani. Azaz a modellen végrehajtott módosítás megfelelő.

Tulajdonság	k	l	Idő (s)
$\Box(\neg f \wedge \neg c \Rightarrow \Diamond\Box c)$	32	30	49.36

7.5. Az eredmények általánosítása

A k-indukciós ellenőrzés sikeressége alapján sejteni lehet a javított rendszernek azt a nagyon kedvező tulajdonságát, hogy a lemmákban rögzített változó-értékhatárok mellett a rendszer állapota tetszőleges állapotból kiindulva a kiépített kapcsolat felé konvergál.

A sejtés igazolásához elég a hibamodell nélküli modell kezdőállapottainak körét kibővíteni. Ez nagyrészt a magas szintű modellen elvégezhető, mivel azonban a $T_{valid,0}$ halmaz a kezdőhelyek függvényében változik, a generált kódot is némileg (két sor hozzáadásával) bővíteni kell. Itt megmutatkozik a folyamatnak az az előnye, hogy a generált kód birtokában a rendszer működése az igényeknek megfelelően rugalmasan alakítható.

Az eredeti, $\Diamond\Box c$ tulajdonságot a módosított modellen sikeresen bizonyítottam. Ez alapján megállapítható, hogy a rendszer véges sok üzenet elvesztése esetén is megfelelően működik, hiszen a lemmák által behatárolt állapottérből az üzenetvesztés nem vezet ki. Ezen felül arra is következtetni lehet, hogy a rendszer kapcsolatkezelése egy leállást követően sem zavarodik meg.

Tulajdonság	k	l	Idő (s)
$\Diamond\Box c$ tetszőleges állapotból kiindulva	32	30	78.47

Az ellenőrzések eredménye alapján a Prolan Zrt. ProSigma rendszert fejlesztő munkatársainak javasoltam a vizsgált módosítás alkalmazását a rendszeren. A javasolt módosítás a SCAN protokoll követelményspecifikációjának 121-es verziójában bevezetésre került.

Összefoglalás

Biztonságkritikus környezetben üzemelő beágyazott rendszerek helyes működése létfontosságú az esetleges hibajelenségek súlyos következményei miatt. A követelményeknek megfelelő viselkedés igazolásának lehetséges módja a modellellenőrzés. Ehhez elő kell állítani egy, a rendszert valamilyen szempont alapján megfelelően leíró modellt. Amennyiben a rendszer működése időzített, az elkészített modell állapottere a megfelelő biszimuláció hiányában nem lesz véges. Ilyen végtelen rendszerek is ellenőrizhetők azonban k-indukció segítségével.

Munkám során egy olyan módszert dolgoztam ki, mellyel a fenti ellenőrzési probléma megoldható. Időzített rendszerek modellezéséhez a k-indukció módszeréhez jól illeszkedő szemantikával bíró, továbbá a mérnöki gondolkodáshoz jól illeszkedő formalizmust definiáltam. A formalizmust úgy határoztam meg, hogy az időzített rendszerek modellezésénél etalonnak számító időzített automaták számos tulajdonságát megörökölje. A formalizmushoz a biztonságkritikus alkalmazási környezet által megkívánt validációs kényszereket fogalmaztam meg.

K-indukciós algoritmussal csak invariáns (így biztonságossági) tulajdonságok vizsgálhatók, így élőségi tulajdonságok ellenőrzéséhez a rendszer megfelelő leképezéséről gondoskodni kellett. A probléma megoldásához lényegében perzisztenciaellenőrzést kellett visszavezetni invariáns tulajdonságok ellenőrzésére. Erre két, egymást jól kiegészítő technikát alkalmaztam, az egyik a tulajdonság bizonyítására, a másik ellenpéldák generálására alkalmas. Mindkét módszer érvényességi körét vizsgálat alá vontam mind a formalizmus, mind a k-indukciós eljárás tükrében. A k-indukciós modellellenőrzés támogatására statikus analízis módszert írtam le, mellyel a rendszer magas szintű modelljéből kinyerhető a rendszer néhány tipikus invariáns tulajdonsága.

A folyamat alkalmazhatósága érdekében Eclipse modellezési technológiák felhasználásával megterveztem és implementáltam az egyes lépésekhez kapcsolódó eszközöket. A magas szintű rendszermodell leírására szakterületspecifikus nyelvet definiáltam Ecore nyelven. A nyelvhez az EMF IncQuery inkrementális gráfmintaillesztő eszközt felhasználva implementáltam a formalizmushoz definiált validációs kényszereket. Ezen felül olyan gráfmintákat definiáltam, melyek meghatározzák a definiált invariánsok egy részét. XTend2 technológiával olyan kódgenerátort hoztam létre, amely a magas szintű modelltől kigenerálja az ellenőrzéshez szükséges, származtatott modelleket modulárisan leíró SAL-nyelvű kódot. A generált kódot a sal-inf-bmc k-indukciós modellellenőrző eszköznek bemenetként megadva a modellezett rendszer verifikálható.

A folyamatot és eszközkészletet egy ipari projekten, a Prolan Zrt. ProSigma rendszer SCAN

protokollján alkalmaztam. A rendszer specifikációja alapján megalkottam a protokoll kapcsolatkezelésének a definiált formalizmus szerinti modelljét. A modellt egyszerű, üzenetvesztéses hibamoddellel egészítettem ki, majd az ellenpéldakeresési eljárással kimutattam a kapcsolatkezelés hibás működését. A hiba okát a generált ellenpélda alapján azonosítva módosítottam a modellt, majd a bizonyítási eljárás segítségével igazoltam az új modell helyességét. Ezt követően a generált kódot manuálisan módosítva és ellenőrizve a javított rendszer helyességét általánosan igazoltam. A modellellenőrzés eredményei alapján a rendszer működésének módosítására konkrét javaslatot tettem, amely alapján a Prolan Zrt. munkatársai korrigálták a rendszer követelményspecifikációját.

Természetesen az elkészült munka több ponton bővíthető további megoldásokkal. A magas szintű formalizmushoz készített szakterületspecifikus nyelvhez grafikus szerkesztő készíthető. Egy ilyen szerkesztő funkcionalitása bővíthető a rendszer szimulációjának és a rendszer végrehajtásai vizualizálásának képességével. A követelmények megfogalmazására az LTL-nél magasabb szintű nyelvet lehet bevezetni. Ezen felül új algoritmusokat és formalizmusokat lehet kipróbálni az ilyen jellegű problémák megoldására.

Köszönetnyilvánítás

Ez úton szeretném megköszönni konzulenseim, dr. Majzik István és Vörös András segítségét, akik sok hasznos tanáccsal és észrevétellel támogatták e dolgozat létrejöttét. Szeretnék továbbá köszönetet mondani Gál Gábornak, Dr. Mosó Tamásnak és Pecsérke Tamásnak, a Prolan Zrt. munkatársainak, akik sokat segítettek a SCAN protokoll megértésében.

Irodalomjegyzék

- [1] Prosigma rendszer scan protokoll. Követelményspecifikáció, v120. Szerkesztő: Pecsérke Tamás.
- [2] Bowen Alpern and Fred B. Schneider. Defining liveness. Technical report, 1984.
- [3] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, April 1994.
- [4] Christel Baier and Joost-Pieter Katonen. *Principles of Model Checking*. The MIT Press, 2008.
- [5] Armin Biere, Cyrille Artho, and Viktor Schuppan. Liveness checking as safety checking. FMICS, 2002.
- [6] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without bdds. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS '99, pages 193–207. Springer-Verlag, 1999.
- [7] Julius R. Buechi. On a Decision Method in Restricted Second-Order Arithmetic. In *International Congress on Logic, Methodology, and Philosophy of Science*, pages 1–11. Stanford University Press, 1962.
- [8] Koen Claessen and Niklas Sörensson. A liveness checking algorithm that counts. To appear, 2012.
- [9] Leonardo de Moura, Sam Owre, and N. Shankar. The sal language manual. Technical Report SRI-CSL-01-02, Computer Science Laboratory, August 2003.
- [10] Bruno Dutertre. Smt for fault-tolerant real-time systems. Lecture slides on SAT/SMT Summer School, 2012.
- [11] Bruno Dutertre and Maria Sorea. Modeling and verification of a fault-tolerant real-time startup protocol using calendar automata. FORMATS/FTRTFT'04, pages 119–214, September 2004.
- [12] Bruno Dutertre and Maria Sorea. Timed systems in sal. Technical Report SRI-SDL-04-03, Computer Science Laboratory, July 2004.
- [13] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:394–406, 1992.

- [14] Roland Kindermann, Tommi Junttila, and Ilkka Niemelä. Beyond lassos: complete smt-based bounded model checking for timed automata. *FMOODS'12/FORTE'12*, pages 84–100, 2012.
- [15] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.
- [16] Denis Oddoux and Paul Gastin. Ltl 2 ba : fast translation from ltl formulae to büchi automata. <http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/>.
- [17] Lee Pike. Real-time system verification by k -induction. Technical Report TM-2005-213751, NASA Langley Research Center, May 2005.
- [18] Amir Pnueli. The temporal logic of programs. *FoCS*, pages 46–57, September 1977.
- [19] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. Checking safety properties using induction and a sat-solver. *FMCAD*, pages 108–125, November 2000.
- [20] Maria Sorea. Bounded model checking for timed automata. Technical Report SRI-CSL-02-03, Computer Science Laboratory, May 2002.
- [21] Wilfried Steiner and Bruno Dutertre. Smt-based formal verification of a *ttethernet* synchronization function. *FMICS*, pages 148–163, 2010.
- [22] Wilfried Steiner and Bruno Dutertre. Automated formal verification of the *ttethernet* synchronization quality. *NASA Formal Methods*, pages 375–390, 2011.
- [23] Stavros Tripakis and Sergio Yovine. Analysis of timed systems using time-abtracting bisimulations. In *Formal Methods in System Design*, 1999.
- [24] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, Lecture Notes in Computer Science, pages 238–266. Springer-Verlag, 1996.
- [25] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.