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

Megbízható architektúrák szintézise gráfgenerátor segítségével

TDK dolgozat

Készítette:

Földiák Máté

Konzulens:

Marussy Kristóf

2020

Tartalomjegyzék

Kivonat

A kiberfizikai rendszerek (CPS) terjedésével, mint az edge alkalmazások vagy az 5G hálózatok, egyre nagyobb szerepet kapnak biztonságkritikus elosztott rendszerek, ilyen rendszerek lehetnek például az önvezető autókat támogató infrastruktúrák. Ezeknek a rendszereknek komoly elvárásokat kell teljesíteni nem csak a funkcionális követelmények terén, hanem az extra-funkcionális követelmények terén is. Ilyen extra-funkcionális követelmény a megbízhatóság, a teljesítmény vagy az elkészítési és üzemeltetési költség. Ezekre a jellemzőkre nagy hatással van az, hogy a rendszer milyen architektúrát használ, így kiemelt fontosságú, hogy a tervezés során a megfelelő architektúraterv kiválasztása.

Ezen a területen kihívást jelent, hogy a funkcionális követelményeknek a rendszer méretében exponenciális számú lehetséges architektúra felelhet meg és ezek különböző extra-funkcionális jellemzőkkel rendelkezhetnek. Tehát fontos lenne egy az optimálishoz közel álló kiválasztása. Az extra-funkcionális metrikák közül számos sztochasztikus modellek analízisével határozható meg. Ezt nehezíti, hogy ezeknek az architektúráknak a vizsgálata is sokszor időigényes az állapottér-robbanás jelensége miatt. Erre a problémára léteznek heurisztikus keresésen alapuló megoldások, amik elkerülik az állapottér-robbanás okozta skálázódási problémákat, de nem képesek helyességi és teljességi garanciákat biztosítani, ezáltal nem biztosítható, hogy a használatukkal az optimális megoldás előállítható.

Ezeknek a problémáknak egy lehetséges megoldása, hogy a potenciális architektúrák előállításához formális módszereket használunk. Ilyen lehet egy logikai következtetőrendszer, ennek használatával matematikai úton igazolható, hogy ha a következtető rendszer helyes bemenetet kapott, akkor az eredményül kapott architektúrák is helyesek. Ugyanakkor itt még nincsen információ arról, hogy az extra-funkcionális metrikák szerint mennyire jó. Ezeknek a mérőszámoknak a meghatározására lehetőség van a kész javaslaton, de a nagyszámú javaslat és az elemzés időigénye együtt jelentős nehézségeket okozhat. Olyan technikát javaslok, mellyel már a generálás során ezekre a jellemzőkre becsléseket adunk és a szintézis folyamatát ezekkel irányítjuk. Az eredmény helyességen túl a teljesség is biztosítható, így a használt absztrakciós szinten az összes lehetséges megoldás előállítható. Ezzel garanciát kapunk arról, hogy az optimális megoldás előállítható, illetve bizonyíthatjuk az előállított megoldás optimalitását.

Ebben a dolgozatban egy a NASA JPL-től származó esettanulmányt bővítek hibalehetőségekkel és redundanciával. Ezt a bővített modellt és az esettanulmányban szereplő extra-funkcionális metrikákat felhasználva arra, hogy a szintézis során előálló gráf alapú félkész modellekre alsó és felső becsléseket adjak a szintézishez használt gráfgenerátor számára. Ehhez egy prototípus implementációt megvalósítva vizsgálom a rendszer skálázhatóságát.

Ezzel a megoldással lehetőség nyílik arra, hogy félkész architektúrákat is vizsgálhassunk extra-funkcionális szempontok szerint a szintézis folyamata közben, az elemzések eredményeinek felhasználásával az architektúra generálás skálázhatóságán lehet javítani.

Abstract

With the spreading of cyber-physical systems like edge applications or 5G networks the safety-critical distributed systems have an increasing role. Systems like this are the support infrastructures of support self-driving cars. In addition to their functional requirements, these systems have to satisfy stringent extra-functional requirements, like reliability, performance, or the cost of production and maintenance. The chosen architecture of a system has a significant impact on these characteristics, thus choosing the best architecture plan has a top priority during the design phase.

A major challenge in this field is that the functional requirements of a system can be fulfilled by an exponential number of potential architectures in the size of the system. Also, these architectures have different extra-functional attributes. Therefore, it is important to select a near-optimal solution. Many of the extra-functional characteristics can be determined by analyzing low-level stochastic models constructed from the higher-level architecture model. However, the analysis of these models is very time-consuming due to the so-called state space explosion. Heuristic approaches are widely applied in these design-space exploration problems, but these solutions cannot provide guarantees for correctness and completeness thus the optimal solution may not be provided.

A possible solution to the correctness and completeness requirement is to use formal methods for architecture generation. I propose the use of graph generation based on logic solvers to address the completeness and correctness challenges while evaluating the extra-functional attributes of the candidate architectures. These metrics can be determined after the synthesis but, due to the computational complexity of the requisite stochastic analyzes, this process quickly becomes intractable as the size of the architectures grows. To mitigate this issue, I propose an approach to evaluate the extra-functional attributes of partial architecture candidates during the graph generation process. Therefore, in addition to ensuring completeness, we can also provide formal proofs that the generated architectures are optimal.

In this work, I extend a case study from NASA Jet Propulsion Laboratory with potential failures and with redundancy. I construct stochastic models compute bounds of the extra-functional attributes of partial architecture candidates to guide the graph generator during synthesis. I create a prototype implementation and analyze its scaling properties.

With this solution, we have an opportunity to analyze extra-functional aspects of partial architectures during synthesis and with the results, we may improve the scalability of architecture synthesis by formal methods.

1. fejezet

Bevezetés

A kiberfizikai rendszereknek (CPS) a funkcionális és extra-funkcionális követelmények teljesíthetősége kritikus kérdés a tervezés során. Itt a cél egy olyan architektúra megalkotása, ami megfelel a követelményeknek. Az automatikus szintézis során szeretnénk, ha a használt eszköz konzisztens architektúrákat állítana elő. Nem elvárás, de az is hasznos lenne, ha a generátor felismerné, hogyha a követelmények nem kielégíthetőek.

A probléma másik része, hogy több megoldás is lehetséges lehet és a megoldások nem azonos extra-funkcionális jellemzőkkel rendelkeznek. A megoldás kiválasztásánál a követelmények teljesülésén túl fontos lenne, hogy az valamilyen szempont szerint optimális legyen. Például a megoldás a legolcsóbb legyen, vagy a teljesítménye a lehető legnagyobb legyen. Ezek miatt fontos a lehetséges architektúrák vizsgálata, de a nagy számú potenciális architektúra miatt a legegyszerűbb elemzések is túlzottan időigényesek.

További probléma, hogy az extra-funkcionális követelmények teljesülését is ellenőrizni szeretnénk. Ehhez szükséges, hogy a jellemzőket pontosan határozzuk meg, amihez összetettebbek analízisek kellenek. Sajnos igaz, hogy a sztochasztikus analízisre használt eszközök is érzékenyek az állapottér robbanásra. [9] Ez azt jelenti, hogy egy új elem hozzáadása a sztochasztikus modellhez exponenciálisan növeli a vizsgált rendszer állapotterét és ezáltal az analízis komplexitását. Ezt kombinálva a nagyszámú lehetséges megoldással látható, hogy az a megoldás, hogy minden lehetséges architektúrát előállítunk és ezeket egyesével elemezzük várhatóan nem praktikus.

Erre egy megoldás az, ha a szintézissel nem akarjuk az összes lehetséges megoldást előállítani, hanem csak azokat, amelyek az extra-funkcionális jellemzők alapján kedvezőek. Ennek egy lehetséges módja, hogy a szintézist extra-funkcionális jellemzők becsléseivel próbáljuk segíteni. Ilyen becslés lehet, hogy a szintézis folytatásával mekkora maximális teljesítmény érhető el. Ugyanúgy ezek a jellemzők a jelenlegi állapotra is meghatározhatóak.

Ebben a dolgozatban az alábbi feladatokat végeztem el:

- Kibővítem az esettanulmányt hibával és redundanciával, hogy a rendszert jellemeze megbízhatóság és a hasznossága függjön a megbízhatóságtól.
- Sztochasztikus modell készítek, amivel egy architektúra megbízhatósági jellemzőit vizsgálni lehet.
- Implementálok egy nézeti transzformációt, amivel gráfmodellből automatikusan elő lehet állítani a sztochasztikus modellt.
- Implementálok egy becslőt, amivel egy parciális modellből elérhető maximális hasznosságot lehet becsülni.
- A végén pedig ezeket integrálom a generátorba és vizsgálom meg a skálázhatóságát.

Ezzel a módszerrel lehetőségünk van arra, hogy a generátor állapotterét csökkentsük, úgy, hogy az optimális architektúra meghatározásánál a generátor teljességét nem sértjük meg. Az állapottér csökkenésnek pedig kedvező hatása lehet a skálázhatóságra, azaz az állapottér robbanás mérséklődne. Ugyanakkor azt is észre lehet venni, hogy ha a használt becslések nem közelítik elég jól a valóságot vagy túlzottan komplexek, akkor a felhasználásukkal a skálázhatóság romlani fog.

2. fejezet

Háttérismeretek

2.1. Funkcionális és extra-funkcionális követelmények

2.1.1. Funkcionális követelmények

A funkcionális követelmények feladata [8], hogy leírják a rendszer technikai feladatait és céljait. Ezek adják meg, hogy a rendszernek milyen feladatokat kell tudnia teljesíteni a működése közben. Egy ilyen funkcionális követelmény lehet az, hogy a műhold legyen képes továbbítani a mérési eredményeit. A funkcionális követelményeket lehet pontosítani. Ennek a célja, hogy egy magas szintű követelményt több kisebb követelmény együttesen teljesít. A fenti példa követelménynek egy finomítása lehet az alábbi: A műhold legyen képes egy elsődleges és egy tartalék úton továbbítani a mérési eredményeket. Az egyes követelményekhez tartozhatnak feltételek is, ezekkel lehet pontosítani a rendszer elvárt működését. Egy finomítása lehet az előző követelménynek a következő: Ha az elsődleges úton a földi állomás nem elérhető, akkor a műhold a tartalék úton továbbítsa a mérési eredményeket.

Elvárás a rendszerrel kapcsolatban, hogy az összes követelményt teljesítse, ennek feltétele, hogy a követelményeknek ellentmondás mentesnek kell lennie. Ugyanakkor nagy rendszereknél a követelmények között előfordulhatnak konfliktusok. Példa egy konfliktusos követelményre a következő: Ha egy műhold közvetlenül a földi állomásnak továbbítja a mérési eredményeit, akkor nem kell rendelkeznie tartalék továbbítási útvonallal. Az ilyen konfliktusokat a tervezés során fel kell oldani. Ezeknek a konfliktusoknak a felderítése és feloldása fontos akkor, ha a tervezést emberek végzik, de ha véletlenül ellentmondás marad a követelményrendszerben, akkor az nem feltétlenül nagy jelent problémát. Ugyanakkor automatikus szintézis során, ahol a követelmények teljesülését formális módszerekkel próbáljuk elérni, ott a legkisebb ellentmondás is a követelmények kielégíthetetlenségét vonja maga után, így lehetetlenné téve a szintézist.

Automatikus szintézis során fontos még, hogy a követelmények teljesen specifikusak legyenek, mivel hiányos követelményekkel a szintézis adhat olyan eredményt, ami a követelményeknek megfelel, de valamilyen egyéb, nem magadott elvárásnak nem felel meg. Erre egy példa lehet az, hogy egy követelménnyel meghatározzuk, hogy a rendszer tartalmazzon egy komponenst, de tényleges használata nincsen kikényszerítve.

Általánosságban igaz, hogy egy funkcionális követelményt a rendszer egy-egy komponense elégíti ki, de finomítás során lehet, hogy a finomított követelmények egy részét a komponens egy részkomponense látja el. A fenti példákat folytatva ilyen lehet, hogy a műholdtól várjuk el, hogy a mérési eredményeket továbbítsa, de ezt a feladatot a műhold kommunikációs alrendszere látja el.

2.1.2. Extra-funkcionális követelmények

Az extra-funkcionális követelményekkel olyan elvárásokat fogalmazhatunk meg, amelyeket általában a rendszer egészének kell teljesítenie. Ezek a követelmények között vannak azok, amelyeknek a teljesülése a rendszer komponenseitől és azoknak a kapcsolatától függenek. Az extra-funkcionális követelmények az alábbi csoportra oszthatóak. [8]

- Biztonsági követelmények: Ezek a követelmények határozzák meg a rendszer elvárt viselkedését azokban az esetekben, amikor a rendszert szándékosan olyan módon akarják felhasználni, hogy az az elvárt működéstől eltérően viselkedjen. Ezek leggyakrabban támadások a rendszer ellen.

- Biztonságossági követelmények: Ezek határozzák meg, hogy a rendszer a működése során természetes módon vagy véletlenül előálló zavaró hatásokra milyen választ kell adnia. Általában ezek közé tartoznak azok, amelyek a rendszer által okozható anyagi károkat vagy a lehetséges emberi sérüléseket határozzák meg.

- Teljesítmény: Ezek határozzák meg, hogy a rendszer üzemeltetése során a feladatait milyen keretek között kell ellátnia. Ilyen lehet például a rendszer áteresztő képessége, feladatvégrehajtási ideje vagy az energia fogyasztása.

- Megbízhatóság: Ezek határozzák meg, hogy a rendszernek a működése során bekövetkező hibák hogyan befolyásolhatják a működését. Ezek a követelmények határozhatják meg, hogy mennyi ideig lehet a rendszer nem elérhető adott idő alatt, mennyi idő alatt kell visszaállnia normális működésre egy hiba után vagy hogy a működés során hány hibát kell kezelnie úgy, hogy a rendszer teljesítménye ne essen egy adott küszöb alá.

- Kölség: Ez takarhat valós költséget, mint az architektúra megvalósításának és üzemeltetésének költsége. Magasabb absztrakciós szinten ez lehet akár a rendszerben találkató komponensek száma is.

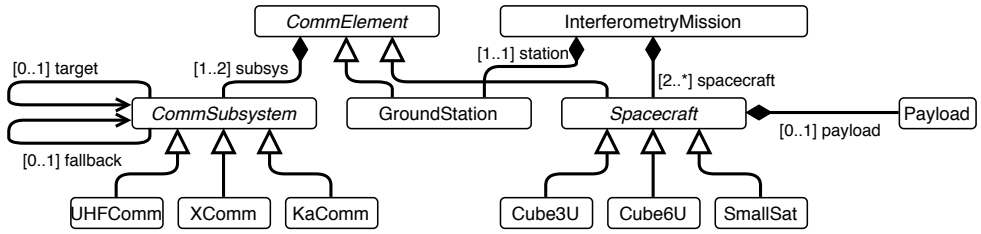
- Használhatóság: Ezek a követelmények határozzák meg az elvárásokat a rendszer interfészeivel szemben. Ezek közé tartoznak például a felhasználói felületek átláthatóságával és kezelhetőségével kapcsolatos elvárások.

- Támogathatóság: Ezek határozzák meg a rendszer fenntarthatóságával kapcsolatos elvárásokat. Ezek közé tartoznak a karbantarthatósággal, bővíthetőséggel és tesztelhetőséggel kapcsolatos elvárások.

2.2. Esettanulmány

A vizsgálat alapjául a Model-transformation-based computational design synthesis for mission architecture optimization [11] cikk szolgál.

Ebben vannak lefektetve az alapok, ugyanakkor ez nem foglalkozik a hibák lehetőségével, így ez ki lett egészítve, hogy alapjául szolgálhasson megbízhatóság vizsgálatoknak is. Ez használtam kiindulási alapnak és egészítettem ki úgy, hogy lehessenek benne a rendszer üzemeléséből származó hibák és a hálózat redundáns legyen. Ugyanakkor ezek a kiegészítések konfliktusba kerülnek a cikk[11] néhány másik elemével. Ezeknek feloldásával érdemben nem foglalkoztam, mivel megbízhatósági szempontból kis jelentőséggel bírnak.



2.1. ábra. A bővített rendszer metamodellje

2.2.1. A rendszer célja

Az esettanulmány egy műholdakból álló hálózatot vizsgál, amelynek feladata, hogy méréseket végezzen tárgyakról a 30MHz alatti tartományban. Ennek legfontosabb következménye az, hogy a feladat végrehajtásához legalább két mérő eszközre van szükség. A mérést jellemzi, hogy az nem pillanatszerű, hanem időbeli kiterjedése van és a mérés ideje hatással van a gyűjtött adatok mennyiségére és ezáltal a mérés hasznosságára. Továbbá az is igaz, hogy több műhold használatával a mérés pontossága javítható, ami a mérési hasznosság növekedéséhez vezet[11].

A mérés során a műholdak egy ideig mérnek, majd mérés után a saját és más műholdaktól kapott eredményeket továbbítják vagy egy másik műholdnak, vagy pedig a földi állomásnak. Itt jelenik meg az elromlás legfontosabb hatása, mivel, ha egy műhold elromlik, akkor annak nem lehetséges adatot küldeni, továbbá a saját mérési eredménye sem lesz felhasználható. A mérés hasznossága a mérés hosszától és attól, hogy hány mérőműszer eredményét sikerült eljuttatni a földi állomásra.

2.2.2. A rendszer komponensei

A rendszer alapvető felépítése az alábbi ábrának megfelelő, de néhány esetben további megszorításoknak is meg kell felelnie.

CommElement: Ez reprezentál minden olyan komponenst, ami befejezett modellek esetében képes kommunikációra. Félkész modellek esetén ide tartoznak azok a komponensek is, amik még nem képesek, de a befejezéskor már mindenképpen azok lesznek.

InterferometryMission: Ez felel meg az egész missziónak. Parciális modelleknél a jelentősége alacsony. Befejezett modelleknél viszont gyökérelemként szolgál minden további komponens eléréséhez.

CommSubsystem: Ez reprezentálja a kommunikációs rendszert. Minden CommElementnek rendelkeznie kell legalább egy darabbal, de kettő is megengedett. Egy ilyen kommunikációs rendszer alkalmas lehet egyszerre küldésre és fogadásra is, de fontos, hogy mind a küldőnek és a fogadónak azonos típusúnak kell lennie. Egy példánya vagy rendelkezik egy fő és egy tartalék célponttal, amiknek tud adatot küldeni vagy nem rendelkezik egyikkel sem.

KaComm: Ka tartományban működő kommunikációs rendszer. Ez alkalmas a műholdak közötti nagysebességű kommunikációra Ka tartományban és képes lehet a földi állomáson lévő Ka kommunikációs rendszerrel is kommunikálni.

XComm: X sávú kommunikációs rendszer. Ez is képes lehet kommunikálni más X sávú rendszerekkel mind a földi állomáson, mind pedig másik műholdon lévő eszközzel. Bár a kommunikációs sebessége lassú, de más nem figyelembe vett szempontok miatt indokolt lehet a használata.

UHFComm: UHF tartományú kommunikációs rendszer, kizárólag műholdak közötti közepes sebességű kommunikációra alkalmas.

GroundStation: Földi állomás, ide kell a mérési eredményeket eljuttatni. Megbízhatósági szempontból úgy módosítja a viszonyokat, hogy az ide szerelt kommunikációs rendszerek nem romlanak el. Ennek a döntésnek az a fő oka, hogy itt lehet lehetőség a kommunikációs rendszer javítására tönkremenetel esetén. Illetve a kommunikációs rendszer is vélhetően kevesebb kártékony hatásnak van kitéve.

Spacecraft: A műhold általános reprezentációja. A missziónak legalább kettőt tartalmaznia kell. Kommunikációs rendszerek tekintetében pontosan egy olyannal rendelkezik, ami küld és fogadhat adatot, illetve rendelkezhet egy másodikkal is, de a második csak fogadhat.

SmallSat: Kis műhold, de nagyobb, mint a többi. A nagy mérete miatt több lehetőség lehet benne redundáns rendszerekre, így ezt a műholdtípust tekintem a legmegbízhatóbbnak.

Cube6U: Hat egység méretű műhold. A közepes mérete miatt közepes elromlási rátát feltételezünk.

Cube3U: Három egység méretű műhold. A kis mérete miatt nagyobb elromlási rátát feltételezünk.

Payload: Ez reprezentálja a mérőműszert, amit a műholdra lehet szerelni. A feladat elvégzése során ebből legalább kettő kell. A feladat során nem rendelkezett az elromlás lehetőségével, de akár rendelkezhetne is.

A meghibásodási ráták nem rendelkeznek valós alappal. Az egyes meghibásodásra képes komponensek meghibásodási rátáját az alábbi táblázat tartalmazza.

Komponens	Meghibásodási ráta
SmallSat	1/70
6U CubeSat	1/66
3U CubeSat	1/62
Ka kommunikációs rendszer	1/10
UHF kommunikációs rendszer	1/12
X kommunikációs rendszer	1/13

2.2.3. A rendszer jólformáltsági kényszerei

A mérés fizikai paraméterei miatt [11] szükséges, hogy legalább két mérőműszerrel ellátott műhold legyen a hálózatban. A hálózat tartalmazhat további mérőműszereket is, amelyek a mérés hasznosságát javítják.

Triviális elvárás, hogy a műholdhálózat minden eleme számára elérhető legyen a földi állomás. Ez a nem redundáns esetben mega után vonja, hogy a hálózat a kommunikációs élek mentén nem tartalmazhat kört. A redundanciával kiegészített esetnél ezt a követelményt megtartottam, hogy egy ilyen rendszerben se alakulhasson ki kommunikációs kör.

További kényszer, hogy a kommunikációs rendszerek kompatibilitását be kell tartani, így kell jólformáltsági kényszer arra, hogy a kommunikációs rendszerek csak a velük megegyező típusú kommunikációs rendszerekkel léphessenek kapcsoltabba.

A kommunikációs rendszereket jellemzi, hogy csak műhold-műhold adattovábbításra képes, vagy pedig képes-e az adatokat a földi állomásra eljuttatni. Emiatt hibás minden olyan architektúra, ahol egy UHF tartományú kommunikációs rendszer közvetlen kapcsolattal rendelkezik egy földi állomáson lévő kommunikációs rendszerrel.

2.2.4. Rendszerszintű hiba modusok

A rendszerben előforduló hibák és következményeik:

- Kommunikációs rendszer romlik el: Ennek a hatása, hogy az a kommunikációs rendszer nem képes adatot fogadni és küldeni. Ha ez második, azaz adatot nem küldő kommunikációs rendszer volt, akkor a hatása, az, hogy azok a kommunikációs rendszerek, amik eddig ide továbbították az adatokat kénytelenek átállni a tartalék kapcsolatra, ha pedig ez volt a tartalék kapcsolat célpontja, akkor a küldő műhold kiesik a hálózatból. Ennél az esetnél a kommunikációs rendszert tartalmazó műhold rajta marad a hálózaton. Ha a tönkrement kommunikációs rendszer küldött is adatot, akkor az egész műhold kiesik a hálózatból, és minden olyan kommunikációs rendszer, ami ehhez a műholdhoz tartozó kommunikációs rendszernek küldött adatot szintén kénytelen átállni a tartalék kapcsolatra vagy kiesik a hálózatból.

- Általános rendszer romlik el: Ekkor az érintett műhold kiesik a hálózatból, függetlenül attól, hogy van-e olyan kommunikációs rendszere, ami képes lehet az adatokat továbbítani és a kommunikációs rendszerei nem képesek adatot fogadni.

- Elérhetetlenségi hiba: Ekkor a műhold és a hozzá tartozó kommunikációs rendszerek is üzemelnek, de sem a rendes sem pedig a tartalék kapcsolaton keresztül nem juttatható el a mérési eredmény a földi állomásra, mert egy másik érintett műhold része vagy egésze ment tönkre. A műhold ekkor is kiesik a hálózatból.

2.3. Szakterületspecifikus modellezés

2.3.1. Metamodell és példánymodell

A metamodell a szakterület fogalmait és az ezek között fennálló relációkat foglalja össze egy szakterület specifikus nyelven. Ebben a dolgozatban architektúrákkal foglalkozunk ezért a metamodell egyrészt katalógusként szolgál arra, hogy megadjuk, hogy a rendszer milyen komponenseket tartalmazhat, másrészt pedig, hogy ezek a komponensek milyen hierarchiába kell, hogy szerveződjenek.

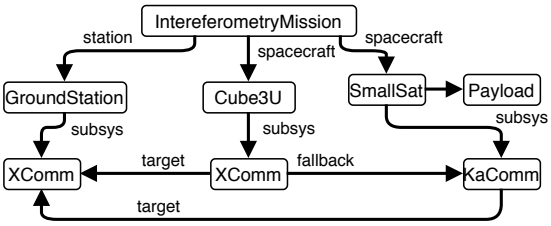
A metamodell grafikus reprezentációjában az Ecore [2] szintaxisát használja, ami az UML [4] szintaxis egy dialektusának tekinthető. A 2.1 árba alapján alapján így leolvasható, hogy például a CommSubsystem helyettesíthető az UHFComm, XComm vagy KaComm bármelyikével.

A tartalmazást a teli rombusz fejű nyilak jelölik. Ezeknek a kapcsolatoknak jellemzője egy kardinalitás és egy név. A kardinalitás az, ami megadja, hogy a tartalmazó a tartalmazottból hány darabbal rendelkezik. Ezt a szögletes zárójel tartalmazza. Az első szám az alsó korlát, a második a felső, a * pedig a tetszőlegeset jelöli. Ez alapján leolvasható, hogy egy InterferometryMission tartalmaz pontosan egy GroundStation komponenst és legalább kettő Spacecraft kompatibilis komponenssel.

A vonal végű nyíl hivatkozást jelöl egy másik komponensre, a feliratozása megegyezik a tartalmazásával.

További jelölés a dőlt betű, ezzel ezt jelöljük, hogy az adott komponens egy bsztrakt osztály. Például a Spacecraft csak úgy szerepelhet, ha helyettesítve van a Cube3U, Cube6U vagy SmallSat valamelyikével.

Egy metamodell példánymodelljének nevezzük azokat a modelleket, amelyek megfelelnek a metamodell által előírt feltételeknek. A metamodell segítségével formalizálhatjuk a funkcionális kényszereket. Ugyanakkor a a 2.2 ábrán látható, hogy a metamodellnek való megfelelés még hordozhat szemantikai hibákat. Egy ilyen hiba, hogy a KaComm kapcsolatban áll XComm alrendszerekkel, ami szemantikusan nem lehetséges.



2.2. ábra. Egy a 2.1 ábrán látható metamodellnek megfelelő példánymodell

2.3.2. Jólformáltsági kényszerek

A jólformáltsági kényszerek szerepe azt, hogy az architektúrák strukturális kényszereit finomítsa. Ez abból áll, hogy tovább korlátozza a metamodellben meghatározott feltételeket.

A jólformáltsági kényszerek feladata, hogy a metamodellt kiegészítve finomítsa az elvárásokat a helyes példánymodellekkel szemben. Itt egyrészt olyan korlátozásokra kell gondolni, amelyek a metamodell hierarchiájában helyesek, de az eredményben nem szerepelhet. Ilyen korlátozás vonatkozhat a metamodellben szereplő öröklésekből származó kompatibilitásokra. A metamodell ugyanis megengedhet egy kapcsolatot, de az a valós architektúrában nem jöthetne létre. Ezt orvosolhatja egy olyan tiltó jólformáltsági kényszer, ami megköti a kapcsolatban szereplő elemek típusát.

A kényszerek megadására a VQL [5] nyelvet használjuk. Egy kényszer VQL nyelven az alábbi, aminek az a célja, hogy ne álljon elő olyan modell, amiben kör található. Ennek

```
@Constraint(severity = "error", key = {Element},
  message = "Communication loop.")
pattern communicationLoop(Element : CommunicatingElement) {
  find indirectCommunicationLink(Element, Element);
}
private pattern indirectCommunicationLink(Source : CommunicatingElement, Target :
    CommunicatingElement) {
  find directCommunicationLink+(Source, Target);
}
private pattern directCommunicationLink(Source : CommunicatingElement, Target : CommunicatingElement)
     {
  CommSubsystem.target(SourceSubsystem, TargetSubsystem);
  CommunicatingElement.commSubsystem(Source, SourceSubsystem);
  CommunicatingElement.commSubsystem(Target, TargetSubsystem);
} or {
  CommSubsystem.fallback(SourceSubsystem, TargetSubsystem);
  CommunicatingElement.commSubsystem(Source, SourceSubsystem);
  CommunicatingElement.commSubsystem(Target, TargetSubsystem);

}
```

elején az @Constraint jelöli, hogy kényszerről van szó és nem csak egy mintáról. Meg kell adni még, hogy a minta illeszkedése mivel jár, ez itt error, tehát az a cél, hogy a modellben sehol ne illeszkedjen. utána pedig a patterneken keresztül leírjuk a keresett mintát. A mintákban lehetnek ÉS és VAGY kapcsolatok, felírhatunk tranzitív mintákat, hivatkozhatunk másik mintára és metamodell elemeit és kapcsolatait is felhasználhatjuk.

2.4. Gráfgenerálás parciális modellekkel

2.4.1. Parciális modellek

Parciális modellek vagy másnéven félkész modellek, olyan modellek a széntézis során, amik nem feltétlenül felelnek meg a metamodellnek és a jólformáltsági kényszereknek.

A metamodellel való ellentmondás főleg hiányosságokat takar. Ilyen hiányosság lehet, hogy egy kapcsolat kardinalitása legalább valamekkora, de a modellben ahhoz a kapcsolathoz nem tartozik elég elem. Ugyanakkor egy parciális modelltől el lehet várni, hogy ne tartalmazzon olyan hibát, ami a bővítésével ne lenne korrigálható. Ilyen lehet, hogy egy komponens egy kapcsolatának kardinalitása ne legyen nagyobb a metamodell által megengedett maxmálisnál. Ezek az elvárások azokra a csomópontokra és kapcsolatokra vonatkozik, amelyekről biztosan állítható, hogy a parciális modell bővítéséből kapott példánymodellben biztosan szerepelni fognak.

Egy parciális modellben szerepelhetnek olyan élek és csomópontok is, amelyekről nem lehet tudni, hogy a végleges modell tartalmazni fogja-e vagy sem. Ennek a szintézis során helyettesíthetnek olyan csomópontokat és kapcsolatokat, amelyekről még nem lett eldöntve, hogy a végleges modellben szerepelnek-e. Ezeknek a reprezentálására háromértékű logikát használunk, ami a logikai igaz és hamis értékeken túl felvehet egy 1/2 értéket, ha még ismeretlen.

A másik konfliktus lehetőség az, hogy a parciális modell a jólformáltsági kényszereknek nem felel meg. A metamodellel való ellentmondásokkal ellentétben a parciális modell bővítése során új ellentmondás kerülhet a rendszerbe. Az, hogy ez az új ellentmondás tényleg létrejöhet-e, az az adott kényszertől függ.

Ugyanakkor a bizonytalan csomópontok és kapcsolatok létezése fontos információkat szolgáltathat a parciális modellről, ami a feldolgozás során felhasználható.

A formális definíciók a [14] cikkben találhatóak.

2.4.2. VIATRA Generator

A VIATRA [16] [15] egy nyílt forráskódú gráfgenerátor. A feladata, hogy a metamodellből és jólformáltsági kényszerekből konzisztens modelleket állítson elő.

A szintézis során a generátor egy kiindulási parciális modellen hajt végre finomításokat[14] és közben ellenőrzi, hogy létre jöttek-e feloldhatatlan ellentmondások.[15] A parciális modellek finomítása addig történik, amíg a kilépési feltétel nem teljesül vagy ha a probléma nem megoldható inkonzisztencia vagy időhiány miatt.

A szintézis kimenete egy EMF [1] példánymodell, egy gráfvizualizáció és statisztikák a szintézisről.

2.5. Sztochasztikus modellezés

A sztochasztikus jelenségek modellezésére a PRISM [3] modellezési nyelvet és eszközt használtam. Ez a nyelv és eszköz lehetőséget biztosít arra, hogy folytonos idejű sztochasztikus jelenségeket vizsgáljunk.

2.5.1. PRISM modellek

A PRISM modellezési nyelv legkisebb eleme a változó. Egy változónak kétféle típusa lehet, az egyik logikai változó, ez a klasszikus igaz és hamis. A másik típus pedig az egész szám. A változók az értékeiket csak meghatározott zárt intervallumból vehetik fel. Opcionálisan lehetőség van arra, hogy egy változó rendelkezzen kezdeti értékkel. Ilyen változó például a sat1_sys és a com1.

Második fontos eleme a nyelvnek a tranzíciók. Ezek a tranzíciók egy feltétel mellett képesek elsülni és egy megadott valószínűséggel megváltoztatni a változók értékeit. Ilyen tranzíciók találhatóak például a sat1 modul utolsó két sorában. Ezeknek a tranzícióknak van lehetőségük megváltoztatni a modul változóit. Az első tranzíció például 1/62 rátával a sat1_sys változót nullába állítja.

A változók olvasásra globális láthatóságúak, így biztosítani kell az egyediséget, ugyanakkor egy változót csak modulon belülről lehet megváltoztatni.

A PRISM modell tartalmazhat még formulákat, ezeknek lényegében helyettesítési szabályok. Ezekkel adhatunk meg logikai összefüggéseket és ezeket használhatjuk fel számításokhoz. Az utolsó sorban például egy olyan formula van, ami megadja, hogy hány műholdra teljesül az online formula.

Az alábbi példakód egy két műholdból álló hálózat PRISM modellje.

```
ctmc

module sat1
    sat1_sys : [0..1] init 1;
    com1 : [0..1] init 1;
    [] sat1_sys=1 -> 1/62:(sat1_sys'=0);
    [] com1=1 -> 1/13:(com1'=0);
endmodule

module sat2
    sat2_sys : [0..1] init 1;
    com2 : [0..1] init 1;
    [] sat2_sys=1 -> 1/66:(sat2_sys'=0);
    [] com2=1 -> 1/13:(com2'=0);
endmodule

formula gnd_ready = true;

formula sat1_main = com1=1 & gnd_ready;
formula sat1_fallback = (!sat1_main) & com1=1 & gnd_ready;
formula sat1_online = sat1_sys=1 & com1=1 & (sat1_main | sat1_fallback);
formula com1_ready = sat1_online;

formula sat2_main = com2=1 & gnd_ready;
formula sat2_fallback = (!sat2_main) & com2=1 & com1_ready;
formula sat2_online = sat2_sys=1 & com2=1 & (sat2_main | sat2_fallback);
formula com2_ready = sat2_online;

formula working = (sat1_online?1:0) + (sat2_online?1:0);
```

2.5.2. PRISM lekérdezések

A PRISM nyelv többféle lekérdezés is támogat. A lekérdezhető egy esemény valószínűsége, adott időpillanatban vagy időponthoz képest. Igazolhatóak állítások események valószínűségire. Vizsgálható események hosszútávú valószínűsége.

Ami számunkra a legjelentősebb, hogy vizsgálható egy állapot fennállása egy adott időpillanatban. A fenti példához kapcsolódóan az alábbi lekérdezéssel azt lehet megtudni, hogy mennyi a valószínűsége annak, hogy az 1 időpillanatban a working formula 2 értékkel rendelkezik.

```
P=? [F=1.0 working=2];
```

A nyelv ennél sokkal többféle lekérdezést is támogat. Lehet vizsgálni hosszútávú valószínűségeket, verifikálni valószínűségeket, várható értéket számolni, elérhetőséget és holtpontot keresni.

2.6. Kapcsolódó munkák

Az architektúra szintézis problémák megoldására sokfélte módszer léteik. Ezek közül mindegyiknek megvan a saját előnye és hátránya.

Egyik ilyen lehetőség, hogy a szintézis feladatot egy logikai következtető rendszerrel oldunk meg. Ennek a megközelítésnek a lényege, hogy a szintézis feladatot egy logikai problémára. Ez a visszavezetés történhet egy logikai kielégíthetősig, azaz SAT problémára. Ennek során a logikai következtetőrendszer olyan paramétereket keres a kapott logikai kifejezéshez, amelyekkel a logikai kifejezés igazat ad vissza. A SAT problémának általánosítása az SMT probléma, ennek előnye, hogy le lehet cserélni predikátumokra, így szélesebb körben alkalmazható modellezési lehetőséget biztosít. Ezeknek a megoldásoknak a hátránya, hogy rosszul skálázódnak, így csak kis méretű kifejezésekre lehet. A logikai következtetőrendszerek előnye, hogy léteznek már meglévő implementációk, mint a Z3 [10] SMT megoldó vagy az Alloy[12].

További lehetőség egy szabály alapú következtetőrendszer használata. [11] Ennek működése során egy kezdeti megoldáson alkalmazunk egy tervezési szabályt. Ezután a kapott megoldást kiértékelhetjük a saját szempontjaink szerint, majd megtarthatjuk vagy újra alkalmazhatunk rajta egy mintát. A megoldás előnye, hogy tetszőleges extra-funkcionális metrika felhasználható a szintézishez [16]. Ennek az optimalizációs eljárásnak a hátránya, hogy a rendszerhez ismerni kell az alkalmazható szabályokat [13].

Szintén lehetőség van iteratív[11] megoldásoknak a használatára. Ennek során a használt megoldó valamilyen módszerrel többször egymás után bővít egy példánymodellt. A megoldás előnye, hogy a bővítéshez nincsen megkötve, hogy milyen megoldó kell, ezért jól kombinálható más módszerekkel. Egy ilyen módszer használatakor érdemes figyelni kell arra, hogy ne állítsunk elő izomorf modelleket, mert ezek csökkentik a megoldások változatosságát. A megoldás hátránya, hogy a jólformáltsági kényszerek betartása nem garantált az iteráció során, így azok ellenőrzésére szükség van az iterációs lépések befejezése után.

Az architektúra szintézishez hasonló állapottér bejárási feladat egy optimális konfiguráció megkeresése. A konfigurációk elemzésére egy megoldás, hogy a rendszerről mintákat veszünk és valamilyen módszerrel következtetéseket próbálunk levonni. Ilyen módszer lehet a Bayesi megközelítés vagy mesterséges intelligencia használata. [7] Ennek előnye, hogy jól skálázóik, viszont az csak közelítő eredményeket ad, ami egy ezen alapuló keresés teljességét kompromitálhatja. Heurisztikus keresésen alapuló eszköz például a PerOpteryx [13].

Az elkészített megoldás az előzőekhez képeset abban tér el, hogy a szintézis feladatot gráfkét kezeli[16] [18] [17]. A szintézis során a generátor a problémát reprezentáló gráfon végez transzformációkat és figyeli a jólformáltsági kényszerek kielégíthetőségét. Legfontosabb előnye, hogy a heurisztikus megoldásokkal ellentétben a generátor az optimalizációt egzakt módon végzi, így teljesíti a keresés teljességi elvárását, ezáltal, az optimális megoldás megtalálása formálisan igazolható.

3. fejezet

Áttekintés

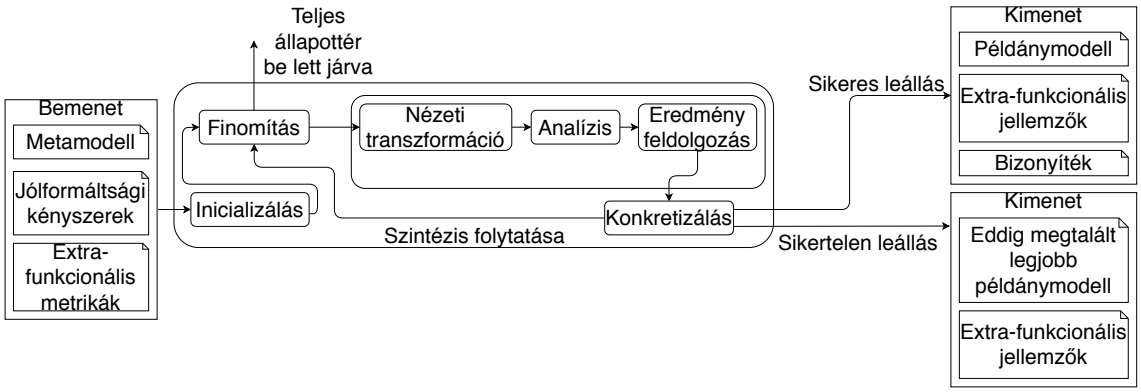
Ebben a fejezetben szeretném bemutatni az általam javasolt megoldás be- és kimeneteit továbbá a legfontosabb komponenseit. A javasolt megoldás felépítése a 3.1 ábrán látható.

3.1. Bemenetek

Először a bemeneteket szeretném bemutatni. Az általános szintézishez két alapvető bemeneti állomány kell. Egyik a metamodell, a másik pedig a jólformáltsági kényszerek. Továbbá ahhoz, hogy az architektúra szintézis során extra-funkcionális jellemzőket is figyelembe tudjunk venni, ahhoz szükségünk van még az extra-funkcionális metrikákat leíró állományokra is.

3.1.1. Metamodell

A bevezetésben elmondottak szerint a metamodell szerepe, hogy leírja az architektúra strukturális jellemzőit. Ez határozza meg, hogy az architektúrában milyen elemek lehetnek és adhat néhány magas szintű megkötést azzal kapcsolatban, hogy ezek az elemek milyen kapcsolatban lehetnek egymással. Fontos hangsúlyozni, hogy ennek az a célja, hogy az architektúra egy absztrakt struktúrát határozzon meg. Ennek önmagában nem kell teljesítenie azt, hogy az ebből elálló architektúrák helyesek lesznek. Ezek a hiányosságok jólformáltsági kényszerekkel javíthatóak. Ugyanakkor érdemes olyan absztrakciót választani, hogy az az elvárt megoldás architektúráját jól közelítse és ne engedjen meg sok hibás kapcsolatot.



3.1. ábra. Javasolt megoldás felépítése

3.1.2. Jólformáltsági kényszerek

A jólformáltsági kényszerek feladata a szintézisnél az, hogy olyan megkötéseket adjon az architektúrára, ami a metamodellben nem ábrázolható. Ezek elsősorban logikai kényszerek. Ezekkel lehet megfogalmazni olyan feltételeket, hogy az architektúrának például milyen szempontok szerint kell összefüggőnek lennie. Ezekkel a kényszerekkel támaszthatunk határozhatunk meg olyan megkötéseket, amelyeknek nem szabad teljesülnie, ha egy másik feltétel teljesül. Erre egy példa lehet, hogy ha két elem között már létezik valamilyen kapcsolat, akkor ugyanazon két elem közé ne hozzon létre egy az előzőhez hasonló új kapcsolatot, akkor sem, ha a metamodell ezt megengedné.

Észre lehet venni, hogy a metamodell által meghatározott strukturális feltételek és a jólformáltsági kényszerek által szabott korlátozások között van átjárhatóság, de egyes feltételek meghatározása emberek számra egyszerűbb a metamodell használatával, míg más feltételek a jólformáltsági kényszerekkel fejezhetőek jól ki. Olyan korlátozások is lehetnek, amelyek a kettő kombinációjával valósíthatóak meg a legegyszerűbben, ezért érdemes ezeket úgy érdemes megválasztani, hogy az a bemenetek átláthatóságát megtartása, mivel egy áttekinthető bemenetről egyszerűbb belátni, hogy helyes. A gráfgenerálás során viszont a metamodell is elemei is jólformáltsági kényszerekre fordulnak le.

3.1.3. Extra-funkcionális metrikák

A harmadik bemenet az extra-funkcionális metrikák. Ennek szerepe, hogy megadja, hogy az egy kész architektúrát jellemző paramétereket hogyan lehet meghatározni. Az egyik legfontosabb ilyen metrika egy költségfüggvény, ami valamilyen módon a kész architektúrában szereplő elemek számát korlátozza. Ennek legegyszerűbb módja egy lineáris költségfüggvény.[14] Ilyen lehet, hogy az elemek száma nem haladhat meg egy megadott darabszámot.

Az extra-funkcionális metrikák másik szerepe a költségkorláton túl a szolgáltatásbiztonsági jellemzők meghatározása. Ilyen jellemző lehet például az architektúra teljesítménye, százalékos elérhetősége, vagy a leállások közötti átlagos idő. Ezeknek a jellemzőknek a szintézis során többféle elvárásnak kellhet megfelelnie. Elvárás lehet, az, hogy egy jellemző feleljen meg egy küszöbérteknek. Ez lehet olyan, hogy ne lépje át azt vagy hogy érje el.

Egy architektúra szintézis folyamat során elvárás lenne, hogy ezeket az extra-funkcionális jellemzőket ne csak befejezett architektúrákra tudjuk számolni, hanem félkész megoldásokra becsülni is. A becslési módszer előállítása a generátor feladata lenne, de a bemenet akár tartalmazhat direktívákat is a becslésre, azzal a céllal, hogy a generátor pontosabb becsléseket tudjon előállítani, vagy a becslés időigénye csökkenjen.

3.2. Feladatok

Egy architektúra generálás többféle céllal történhet. Lehet olyan feladat, ahol csak az optimális megoldás fogadható el, de olyan feladat is elképzelhető, hogy egy közel optimális megoldási is jó, illetve olyan is, ahol csak annyi a cél, hogy az extra-funkcionális jellemző egy tartományon belül legyenek. Ezek alapján az alábbi szintézis feladatok képzelhetőek el:

- Maximális teljesítmény költséghatárral: Ennek egyik lényege, hogy adott egy költséghatár, amit az architektúra nem léphet át. Ilyen költség lehet a maximális elemek száma. Ennél egy valósabb metrika, ha az elemek nem azonos költséggel rendelkeznek és felhasznált elemek összköltségére szabunk határt. Ennek a feladattípusnak a során az optimalizációs eljárással pedig a lehető legnagyobb teljesítményt próbáljuk elérni, természetesen a költséghatár betartásával.

- Teljesítmény küszöb költséghatárral: A fentihez hasonlóan itt is adott egy költséghatár, de az architektúrától nem várunk el optimálását.
- Formális bizonyítás arra, hogy nincs hatékonyabb architektúra adott költségkorlát mellett
- Teljesítmény küszöb költségminimalizálással: Ez arról szól, hogy az architektúrának teljesítenie kell egy hatékonyság követelményt, de a lehetséges megoldások közül a legolcsóbbat keressük.
- Pareto-front meghatározása: Ennek a lényege, hogy azoknak a megoldásoknak a halmazát keressük, amelyekre igaz, hogy az egyik optimalizálandó paramétere nem javítható anélkül, hogy a másik paraméter ne romolna. Logikusnak tűnhet, hogy ezek közül a felhasználó válassza ki a számára legkedvezőbbet, de a Pareto-front elemszáma lehet, hogy túl nagy, illetve a meghatározás is nagyon számításigényes feladat. A továbbiakban ezt a feladatot nem vizsgáljuk.

A dolgozatban az architektúra szintézist az első két esetet vizsgáljuk, ahol a teljesítménynek egy el kell érnie vagy egy minimum értéket, vagy pedig a lehető legmagasabbat egy megadott csomópontszám korlát mellett.

3.3. Generátor felépítése

3.3.1. Inicializálás

Az inicializálási lépés lényege, hogy a bemeneti állományokat a belső működéshez szükséges formátumra. Ennek a lépésnek a lényege, hogy a metamodell és a jólformáltsági kényszerekből feldolgozza, hogy egy parciális modellen milyen finomítási lépések hajthatóak végre. További feladata, hogy a jólformáltsági kényszerek ellenőrzésére szolgáló lekérdezéseket is meghatározza.

A másik feladata, hogy a kész architektúrákra vonatkozó extra-funkcionális metrikákból létrehozzon egy nézeti transzformációkat, amelyekkel egy parciális modell jellemzőire jó alsó és felső becsülést lehet adni.

Szintén ehhez a lépéshez tartozik a kilépési feltétel meghatározása.

3.3.2. Finomítás

Ennek során választunk egy meglévő parciális modellt és azon egy bővítést hajtunk végre. A bővítés lehet egy új komponens hozzáadása vagy egy döntés meglévő komponensek közötti él behúzásáról.

A lépés során fontos, hogy finomítással a parciális modell ne kerülhessen feloldhatatlan ellentmondásba a metamodellel és a jólformáltsági kényszerekkel. Továbbá, érdemes egy finomítási lépésnél figyelembe venni, hogy a döntésnek van-e olyan hatása a parciális modell más elemeire, hogy az érintett elemről egyértelmű döntés hozható. Ekkor ezt a döntést is meghozni és újra vizsgálni az egyértelmű döntések lehetőségét addig, amíg a parciális modell már nem tartalmaz ilyet.

3.3.3. Nézeti transzformáció

A nézeti transzformáció során a parciális architektúra modellt leképezzük az adott extra-funkcionális metrika elemzéséhez szükséges modellre. A lépés során létrehozunk egy-egy modellt a metrika alsó és felső becslésére.

A modelleken túl ebben a lépésben állítjuk elő a modell elemzéséhez szükséges lekérdezéseket is.

Ez a transzformációs lépés használható arra, hogy a szintézis során olyan elemzési módszereket és akár külső eszközöket is fel lehessen használni, amelyet a generátor nem tartalmaz.

3.3.4. Analízis

Az analízis lépés során történik meg a nézeti modellek elemzése az előállított lekérdezések szerint. Ennek a működésére fekete dobozként tekintünk.

3.3.5. Eredmény feldolgozás

Mivel az elemzések eredményeit közvetlenül biztos, hogy fel lehet használni, így ebben a lépésben az analízis során kapott eredményeket átalakítjuk a generátor által felhasználható értékekre. A feldolgozás után a generátor számára egy metrikához mindenképpen egy számot kell rendelni. A szám bármi lehet a $[-\infty; +\infty]$ tartományban vagy lehet egy helyettesítő érték arra az esetre, ha az analízis vagy a feldolgozás alatt hiba lépett fel és ezért az eredményt nem szabad figyelembe venni vagy más módon kell eljárni.

A lépés további lehetőséget ad, hogy az analízis eredményein utófeldolgozást hajtsunk végre. Elképzelhető például olyan eset, hogy egy analízishez több lekérdezés is tartozik és a lekérdezések eredményeit egy metrika szerint súlyozni kell. Az ilyen lépések végrehajtása is ehhez a lépéshez tartozik.

3.3.6. Konkretizálás

Ennek a lépésnek az a lényege, hogy az elemzett parciális modellről eldöntse, hogy lehet-e végleges modell. Ha lehet, akkor hozzon döntést arról, hogy a szintézis folyamata folytatódjon-e. A lépés során, születhet döntés arról, hogy a parciális modell benne maradjon-e a finomítható modellek között vagy a generátor azon már ne hajtson végre finomítást.

Ha a parciális modell nem további finomítás nélkül kész architektúra, akkor a lépés során döntést kell hozni arról, hogy érdemes-e a finomításával tovább foglalkozni. Ennek a döntésnek akkor van jelentősége, ha a szintézis során figyelembe veszünk optimalizálandó metrikákat vagy egy extra-funkcionális metrika szerint el kell érni egy küszöbértéket.

3.4. Kimenetek

A szintézis az alábbi eredményekkel fejeződhet be:

- Nem megoldható a feladat: Ebben az esetben a generálás során bizonyítékot nyer az, hogy a feladat az adott kényszerekkel nem teljesíthető. Erre egy kimenet lehet, egy bizonyíték arra, hogy a követelményeket tényleg nem lehet teljesíteni.
- Sikertelen optimalizáció: Ekkor nem sikerült az optimálisat megtalálni. Itt lehetséges két aleset aszerint, hogy a szintézis során volt-e olyan parciális modell, ami konkretizálható. Ha volt ilyen, akkor sikertelen leállás mellett is vissza lehet adni olyan architektúra javaslatokat, amelyről nem tudjuk bizonyítani, hogy optimális és várhatóan nem is az, de a metamodellnek és jólformáltsági kényszereknek megfelel.
- Sikeres leállás: Ez az, ha találtunk olyan architektúrát, ami az extra-funkcionális küszöbértékeknek és az optimalizációnak is megfelel. Ekkor a kimenet az architektúra

mellett egy bizonyíték az optimalitásra. Ez a bizonyíték származhat abból, hogy a generátorról bizonyított a helyes működés és optimalizáció.

A megoldás limitációja, hogy a generátor nem tudja meghatározni, hogy a szintézisnek miért nincsen megoldása. A sikertelenségnek több oka lehet, például, hogy nem volt elég a rendelkezésre álló idő vagy kényszerek konfliktusba kerültek egymással.

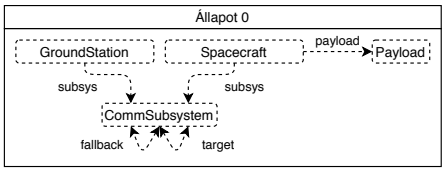
4. fejezet

Megvalósítás

4.1. Gráfgenerálás parciális modellekkel

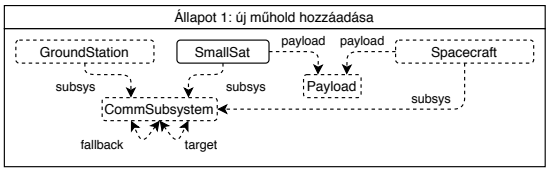
A gráfgenerálás folyamatát egy példa lefutás egy részén keresztül szeretném bemutatni.

A szintézis a 4.1 ábrán látható állapotból indul. Kezdetben a parciális modellben nincsen biztosan létező komponens. Ekkor a parciális modell csak a bizonytalan elemeket tartalmaz. A bizonytalan elemek takarhatnak egy és több elemet is. Ugyanígy a kapcsolatok is bizonytalan állapotban vannak.



4.1. ábra. A kiindulási állapot

A szintézis során a generátor a bizonytalan csomópontokról és élekről döntéseket hoz. A döntés lehet egy bizonytalan csomópontból egy biztos leválasztása. Ennek a lépésnek a során a bizonytalan elem megmaradhat. Ilyen lehet egy új műhold hozzáadása, ami a 4.2 ábrán látható.

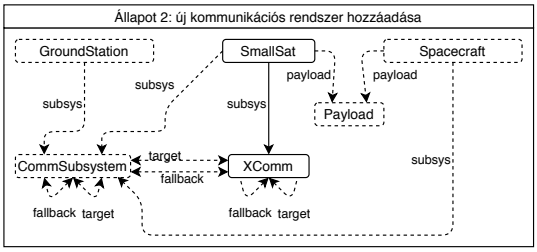


4.2. ábra. Finomítás: műhold hozzáadása

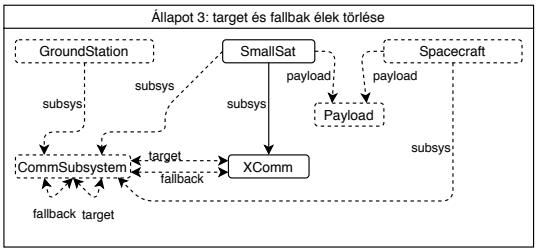
Egy új csomópont behúzása során létrejöhetnek új élek a parciális modellben. Például ha a 4.2 ábrán látható állapotban azt a döntést hozza a generátor, hogy hozzáad a rendszerhez egy új műholdat, akkor a 4.3 ábrán látható állapot jön létre.

A szintézis során történhet a generátor hozhat olyan döntést egy élről, hogy az biztosan nem fog szerepelni a végleges modellben. Ilyen lehet, hogy a 4.3 ábrán látható állapotból eltávolít egy target majd egy fallback élet. Ekkor a 4.4 ábrán látható állapotba jutunk.

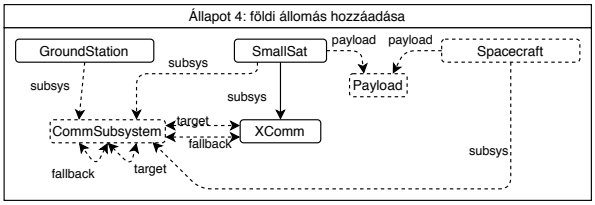
Ez után hozzá lehet adni egy új földi állomást, ezzel a 4.5 ábrán látható állapotba jutunk el.



4.3. ábra. finomítás: kommunikációs rendszer hozzáadása

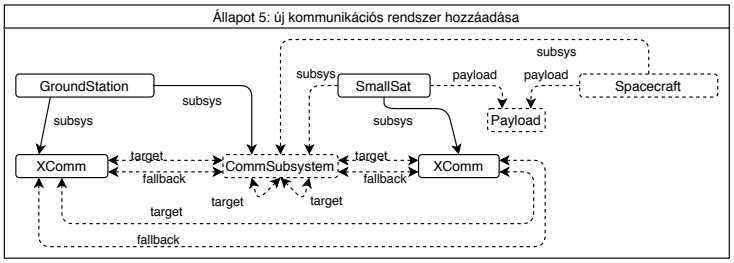


4.4. ábra. Finomítás: élek törlése



4.5. ábra. Finomítás: földi állomás hozzáadása

Innen egy kommunikációs rendszer hozzáadásával a 4.6 ábrán látható állapotba kerülhetünk.

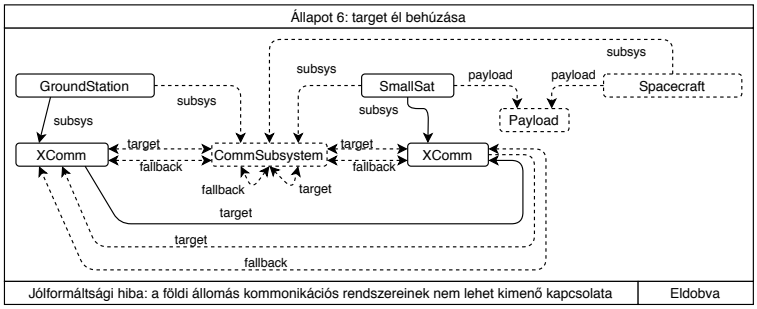


4.6. ábra. Finomítás: kommunikációs rendszer hozzáadása

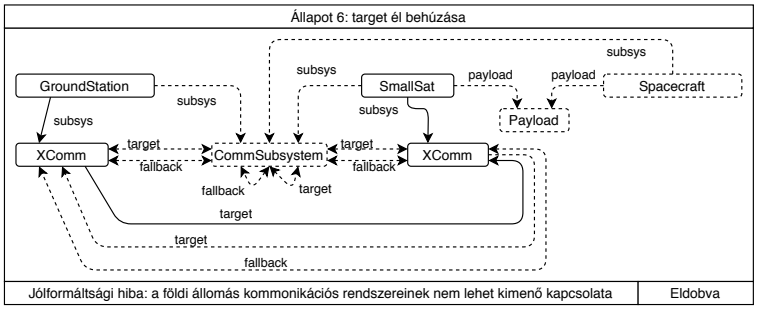
A döntéshozás során olyan döntést is hozhat a generátor, aminek hatására a parciális modell ellentmondásba kerül a kényszerekkel. Ilyen döntés a 4.8 ábrán látható. Ekkor a behúzott él nem felel meg a jólformáltsági kényszerek által támasztott követelményeknek. Ekkor a generátor az így kapott parciális modellt eldobja.

A szintézis során a generátor a meglévő parciális modellek közül véletlenszerűen választ egyet és azon hajt végre egy véletlen finomítási lépést, így egy valós futás csak nagyon kis valószínűséggel ennyire következetes.

A szintézis alatt a parciális modellek már hordozhatnak információkat a belőlük készíthető modellek extra-funkcionális jellemzőiről. A parciális modellek elemzésével azt a



4.7. ábra. Finomítás: target él behúzása



4.8. ábra. Finomítás: Target él behúzása

célt szeretnénk elérni, hogy a szintézis során a generátor olyan parciális modelleket is eldobjon, amelyekről formális úton belátható, hogy nem létezik olyan finomítási lépéssorozat, ami úgy változtatna a parciális modellen, hogy az megfeleljen a kitűzött extra-funkcionális metrikák célértékeinek.

4.2. Optimalizáló gráfgenerálás

Optimalizáló gráfgenerálás során a parciális modellekhez figyelembe vesszük az extra-funkcionális jellemzőket is. Ennek egyik kiindulási feltétele, hogy legyen egy befejezett modellekre számítható metrika. Az esettanulmány alapjául szolgáló cikk [11] a mérési hasznosságra az alábbi összefüggést határozza meg:

$$cov(N,T) = \left(1 - \frac{2}{N}\right)^{\left(1+\frac{9}{T}\right)} + 0.05\frac{T}{3} \tag{4.1}$$

A képlet olyan architektúrákra vonatkozik, amelyek nem tartalmaznak meghibásodást, így módosítani kellett. Erre azt a megoldást adtam, hogy a hasznosságot súlyozom a misszió végén a hálózatban maradt műholdak számának valószínűségével. Ez alapján a felhasznált várható hasznossági metrika a következő:

$$hasznosság(N,T,A) = \sum_{w=2}^{N} P(w,T,A) \cdot cov(w,T) \tag{4.2}$$

Ahol T a mérés hossza órában, A a mérés elvégzéséhez használt architektúra, az N padig az architektúrában lévő mérőműszerek száma.

A szintézis során a parciális modellhez hozzá akarjuk rendelni ennek az alsó és felső becslését.

Az első optimalizációs lehetőség, hogy a kész modellektől azt várjuk el, hogy a hozzájuk tartozó hasznosság érték elérjen egy K küszöbértéket. Ekkor azokat a parciális modelleket kell eldobni, amelyekről belátható, hogy nem létezik hozzájuk olyan finomítási lépéssorozat, amellyel a kapott architektúra a elérné a küszöbértéket.

A döntéshozáshoz adottak a következőek:

- p: parciális modell
- L: Azok a finomítási lépéssorozatok halmaza, amellyel p parciális modell helyesen befejezhető
- F(p, l): p parciális modellen az l lépéssorozat végrehajtásával kapott helyes modell
- N(m): m modellben lévő mérőműszerek száma
- K: küszöbérték

Ekkor az eldobás feltétele:

$$K > \max_{\{l \in L\}} \{hasznosság(N(F(p,l)), T, F(p,l))\} \tag{4.3}$$

Ez szintézis közben nem segítség, mert meg kell határozni hozzá az összes lehetséges lépéssorozatot, ami egyenértékű a szintézissel. Ugyanakkor, ha tudunk adni egy helyes felső becslést a maximális hasznosságra a parciális modell alapján, akkor az eldobást a lépéssorozatok meghatározás nélkül el lehet dönteni.

Ennek a következő feltétele van:

- $E_{max}(p,T)$: p parciális modell T missióidőre vett hasznosságának felső becslése
- $$E_{max}(p,T) >= \max_{\{l \in L\}} \{hasznosság(N(F(p,l)), T, F(p,l))\} \tag{4.4}$$

Ha a feltételek teljesülnek, akkor az eldobás új feltétele:

$$K > E_{max}(p,T) \tag{4.5}$$

Végig lehet gondolni, hogy ha a becslés nem közelíti eléggé a valós értéket, akkor a becsléssel történő eldobásnál lehetséges, hogy meg fogunk tartani olyan parciális modelleket, amelyeknek a maximális elérhető hasznossága kisebb a küszöbértéknél. Ezért fontos, hogy a becslő jól közelítse a tényleges értéket.

Az is látszik, hogy ha nincsen olyan parciális modell, aminek van nem kifejtett finomítási lépése és csak olyan modelleket dobtunk el, akár becslés alapján, amelyek nem teljesíthetik az extra-funkcionális kényszereket, akkor megállapítható, hogy a követelmény rendszer inkonzisztens és a feladatnak nem létezik megoldása.

A felső becsléshez hasonlóan a parciális modellre alsó becslést is lehet adni. Ezt a kilépési feltétel ellenőrzésénél lehet felhasználni. A küszöbérték alapján történő optimalizációnak a kilépési feltétele a strukturális és jólformáltsági kényszereken túl a következő:

$$K <= hasznosság(N(F(p,\emptyset)), T, F(p,\emptyset)) \tag{4.6}$$

Az alsó becslésre a következő feltételnek kell teljesülnie:

- $E_{min}(p,T)$: p parciális modell T misszióidőre vett hasznosságának alsó becslése
- $$E_{min}(p,T) <= \min_{\{l \in L\}} \{hasznosság(N(F(p,l)), T, F(p,l))\} \tag{4.7}$$

Ekkor a kilépés feltétele becsült alsó korláttal:

$$K <= E_{min}(p, T) \tag{4.8}$$

Ennek közvetlen hatása, hogy innentől akárhogyan fejezzük be a modellt, az mindenképpen eleget fog tenni a küszöbérték követelményének. Ekkor komplexitást lehet spórolni a generátorban azzal, hogy a továbbiakban a parciális modellből finomított modellekre nem kell elvégezni a feltétel ellenőrzését vagy becslését, mert az már mindenképpen teljesülni fog.

A fenti gondolatmenetben hasznosság helyett bármilyen olyan metrika használható, amelynél a nagyobb érték a kedvezőbb. Abban az esetben, ha olyan metrikát vizsgálnánk, ahol a kisebb érték a kedvezőbb, ott ha a metrika és a küszöbérték ellentétjét vesszük, akkor az eljárás ezzel vissza lett vezetve a fenti gondolatmenetre.

Másik optimalizációs eljárás, hogy egy küszöbérték elérése helyett a legjobb megoldást keressük. Ez az eset visszavezethető a küszöbértékes optimalizációra azzal az eltéréssel, hogy az elérendő küszöbérték a legnagyobb nem minden lehetséges módon kifejtett parciális modell maximális elérhető értéke.

Ez azt jelenti, hogy egy parciális modellt el lehet dobni, ha létezik másik olyan parciális modell, aminek az alsó becslése nagyobb, mint a parciális modell felső becslése. Formálisan p eldobható, ha

$$\exists k \neq p : E_{max}(p, T) < E_{min}(k, T) \tag{4.9}$$

4.3. Lineáris költségfüggvény becslése

A parciális modellben lehetnek olyan komponensek, amelyek lehet, hogy szerepelni fognak a kész modellben, ebbe beletartoznak azok is, amelyekről biztosan tudjuk, hogy szerepelni fognak a kész modellben. Ugyanígy lehetnek olyan komponensek is, amelyekről biztosan tudjuk, hogy szerepelni fognak a modellben. A parciális modell különböző komponenseire ezeknek a száma meghatározhat és alapján a generátor képes lineáris egyenlőtlenség rendszereket kezelni.

Ezekből az egyenlőtlenség rendszerekből meghatározható egy intervallum, ami megadja, hogy az egyes komponensekből még hány darab kerülhet a modellbe. Szintén ezek felhasználhatóak inkonzisztens állapotok detektálására.

A modellbe bekerülő új csomópontok minimális és maximális számára szintén része ennek az egyenlőtlenség rendszernek, így a generátor fel tudja ismerni azokat az eseteket, amikor a parciális modell nem konkretizálható, mert nem tartalmaz elég komponenst és inkonzisztens állapotba jutna, ha túl sok komponens szerepelne a modellben.

4.4. Szolgáltatásbiztonsági metrika becslése

4.4.1. Nézeti transzformáció általánosítása

A nézeti transzformáció lényege, hogy a parciális modellt leképezzük egy olyan sztochasztikus modellre, aminek a segítségével a vizsgált extra-funkcionális jellemző értéke meghatározható.

A sztochasztikus analízishez szükségünk van egy olyan nézeti modellre, ami az elemzés szempontjából fontos információkat tartalmazza. Ennek létrehozásához először a parciális modellből ki kell szűrni azokat az elemeket, amelyeket, amelyek befolyásolhatják a sztochasztikus analízis eredményét. Ez a vizsgált esettanulmányban a következőket jelenti:

- Lista a műholdakról

- Egy-több hozzárendelés műholdak és a hozzájuk tartozó kommunikációs rendszerekről
- Két egy-egy hozzárendelés az elsődleges és tartalék kapcsolatokról küldő és fogadó kommunikációs rendszer között
- Lista a földi állomáshoz tartozó kommunikációs rendszerekről
- Egy-egy hozzárendelés az elromlásra képes komponensek és az elromlási ráta értéke között
- Lista a mérőműszerrel ellátott műholdakról
- Dinamikus struktúra, ami módosítások paramétereit tartalmaz

A nézeti transzformáció második lépése, hogy előállítjuk a PRISM modellfájlt. Ennek első transzformációs lépése, hogy a műholdakat leképezzük modulokra, a kommunikációs rendszereket alapján pedig kiegészítjük a modulokat. Ez minden műholdra külön történik.

Ebben a lépésben létrehozunk egy modult a műhold nevével. A modul mindenképpen tartalmaz egy változót a műhold általános rendszereinek a reprezentálására, ami 1 érték esetén működik, 0 esetén nem. A változóhoz tartozik egy tranzíció is, ami az elromlását modellezi. A tranzíció akkor következhet be, ha a műhold változója 1. Ez a tranzíció a műholdhoz rendelt elromlási rátának megfelelően a változót 0 értékre állítja.

A műhold tartalmazhat kommunikációs rendszereket, amiket szintén a modulba kell tenni. Ennek során minden a műhold által tartalmazott kommunikációs rendszernek létrehozzuk a saját változóját és az elromlási tranzícióját a műholdéhoz hasonló módon.

A következő feladat, hogy logikai kapcsolatokat is belevegyük a modellbe. Ez azzal kezdődik, hogy a földi állomás kommunikációs rendszereihez létrehozunk formulákat, amelyek mindig igazak, mivel a földi állomáson lévő kommunikációs rendszereket ideálisnak tekintjük. Ezek a formulák azt állítják, hogy a kommunikációs rendszerek mindig képesek adatot fogadni.

Ezután a műholdakon lévő kommunikációs rendszerekhez tartozó formulákat hozzuk létre. Az első ilyen formula az elsődleges kapcsolatot írja le, ez akkor igaz, ha a műhold egyetlen küldő kommunikációs rendszere működik és annak van elsődleges kapcsolata és a fogadó oldalon lévő kommunikációs rendszer tudja fogadni az adatokat.

A második formula a tartalék kapcsolatot írja le. Ez akkor igaz, ha az elsődleges kapcsolat hamis és a kommunikációs rendszer működik és a kommunikációs rendszernek van tartalék kapcsolata és a fogadó oldalon lévő kommunikációs rendszer tudja fogadni az adatokat.

Harmadik formula a műholdnak azt a tulajdonságát adja meg, hogy a hálózaton van-e. Ez logikailag akkor teljesül, ha a műhold általános rendszere működik és ezen kívül vagy az elsődleges vagy a tartalék kapcsolata igaz. Ebből és az előző két formulából minden műholdhoz egy tartozik.

Módosító eset, ha a műhold rendelkezik ideális kommunikációs rendszerrel. Ekkor az elsődleges és tartalék kapcsolatot leíró formulák nem jönnek létre és a műhold hálózati elérhetősége attól függ csak, hogy az általános rendszere működik-e.

Negyedik formula minden, beleértve a küldő kommunikációs rendszereket is, kommunikációs rendszer után létrejön. Ez a formula azt határozza meg, hogy a kommunikációs rendszer fogadhat-e adatot. Ez akkor teljesül, ha kommunikációs rendszert tartalmazó műhold rajta van a hálózaton és a kommunikációs rendszer működik.

Utolsó feladat a modell előállításához a lekérdezésekhez tartozó formula előállítása. Ez a formula a korábbiaktól eltérően számot ad vissza és nem logikai értéket. A formula minden mérőműszert tartalmazó műholdhoz 1 értéket rendel, ha az a hálózaton van és 0

értéket, ha nem, majd ezeket összegzi. Módosító paraméterrel a kezdőértéke megváltoztatható.

Második feladat a nézeti transzformáció során, az, hogy a lekérdezéseket előállítsuk. Ekkor hozzuk létre azokat a lekérdezéseket, amelyek meghatározzák, annak a valószínűségét, hogy a vizsgált pillanatban adott számú kommunikációs rendszerrel ellátott műhold van a hálózaton.

Futásidő optimalizációs céllal nem hozunk létre lekérdezést azokra az esetekre, amelyekben a hálózaton kettőnél kevesebb kommunikációs rendszer lehetne, mivel az ezekre az esetekre a vizsgált hasznosság függvény nem értelmezett. Szintén nem hozunk létre lekérdezéseket olyan esetekre, amelyek valószínűségéről tudjuk, hogy 0, Ez akkor fordulhat elő, ha legalább 3 ideális műhold van a hálózatban. Ideális műhold az, aminek van mérőműszere és mindig elérhető. Ilyen műhold a hálózaton lévő műholdakat számláló formula módosító paraméterén keresztül adható hozzá. Ilyenkor nem kell azokat az eseteket vizsgálni, amikor kevesebb műhold lenne elérhető, mint a hálózatban lévő ideális műholdak száma.

Felső határ a hálózatban szereplő mérőműszerek száma az ideális műholdakon lévőket is beleszámolva.

A szintézis során történő felhasználáshoz szükség van arra, hogy a fent leírt transzformációs eljárásnak olyan bemenetet adjunk, amivel az előállított sztochasztikus modell alsó és felső becslésre használható.

Az alsó becslés során az alábbi állításokat használjuk fel:

- Az optimalizációt fix misszióidővel végezzük.
- A hálózat hasznossága csak akkor csökkenhet, ha az elérhető mérőműszerek száma csökken. Ez könnyen belátható a 4.3 képelt alapján. A mérőműszer kiesése pedig csak akkor következhet be, ha egy műhold esik ki a hálózatból. Így egy műhold hasznosság hozzájárulása monoton növekvő az elérhetőségének valószínűségében.
- Ha egy műhold nem rendelkezik kommunikációs rendszerrel, az alulról becsülhető azzal, hogy nincsen a hálózaton. Ekkor az elérhetőségének valószínűsége 0, ami legfeljebb akkora, mint a működésének valószínűsége tetszőleges feltételek mellett.
- Ha egy műhold nem rendelkezik küldő kommunikációs rendszerrel, az alulról becsülhető azzal, hogy nincsen a hálózaton. Ekkor az előzővel azonos módon bizonyítható, hogy az elérhetőségének valószínűsége 0, ami legfeljebb akkora, mint a működésének valószínűsége tetszőleges feltételek mellett.
- Egy műhold elérhetőségének valószínűsége monoton növekvő a küldő kommunikációs rendszerének valószínűségében, mivel a műhold elérhetőségének valószínűsége szorzata a következő három független valószínűségnek: a műhold rendszerének működési valószínűsége, a kommunikációs rendszerműködésének valószínűsége és annak, a valószínűsége, hogy legalább az egyik kapcsolaton a fogadó tudja fogadni az adatokat. Utóbbi függetlensége abból következik, hogy a hálózattól körmentességet várunk el.
- Ha egy küldő kommunikációs rendszer nem rendelkezik elsődleges kapcsolattal, az alulról becsülhető azzal, hogy az elsődleges kapcsolata sohasem működik. Ez abból látszik, hogy egy kapcsolatnak két állapota lehet és a nem működik a kedvezőtlen.
- Ha egy küldő kommunikációs rendszer nem rendelkezik tartalék kapcsolattat, az alulról becsülhető azzal, hogy a tartalék kapcsolat sosem működik. Pontosan ugyanazért, amiért az előző.

Ezekből az következik, hogy ha a sztochasztikus modellt hiányos parciális modell alapján állítjuk elő, akkor a sztochasztikus modell elemzésével kapott eredmény jó alsó becslés lesz.

A felső becslés előállításához a sztochasztikus modell kiegészítésére van szükség. Itt az egyik részfeladat, hogy a parciális modellben szereplő műholdak elérhetőségének valószínűségét maximalizáljuk. Egy műholdat elérhető, ha van kimenő kapcsolata, amin keresztül a földi állomás elérhető. Másik, hogy a hálózatban szereplő mérőműszerek számát is maximalizáljuk. Szintén egy mérőműszer elérhető, ha az azt tartalmazó műhold elérhető.

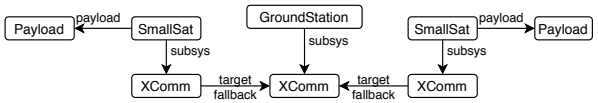
A becsléshez használt első módosítás a sztochasztikus modellbe az, hogy meghatározzuk, hogy a parciális modellhez legfeljebb hány új mérőműszert lehetne hozzáadni. Ezt szerencsére nem kell elvégezni, mivel a generátor képes ezt a számot meghatározni. A sztochasztikus modellt ekkor úgy módosítjuk, hogy minden lehetséges új mérőműszerhez ideális műholdat feltételezünk, azaz a műhold mindig elérhető. Ez abból következik, hogy egy mérőműszer kontribúciója a mérési hasznossághoz csak úgy csökkenhet, ha a műszert tartalmazó műhold kiesik a hálózatból. Az ideális műhold elérhetőségének a valószínűsége 1, ami legalább akkora, mint tetszőleges nem ideális műhold elérhetőségének valószínűsége, így ez egy jó felső becslés.

A másik dolog, amit a parciális modelleken változtatni kell, az a hiányos műholdak elérhetőségének felső becslése.

Itt felhasználjuk, hogy a műholdnak csak egy küldő kommunikációs rendszere lehet. Küldő kommunikációs rendszer az, amelyiknek létezik kimentő kapcsolata. A metamodellben ez nincsen külön megkülönböztetve, az a szerep akkor dől el, amikor a generátor a kommunikációs rendszerhez kimenő kapcsolatot rendel.

Az első ilyen hiány az, ha egy műholdnak nincsen küldő kommunikációs rendszere. Ez felülről becsülhető azzal, hogy a műholdhoz hozzárendelünk egy ideális kommunikációs rendszert. Ideális kommunikációs rendszer az, amin keresztül 1 valószínűséggel érhető el a földi állomás. Ekkor mivel a műhold elérhetőségének valószínűsége az általános rendszerének működési valószínűségének és a küldő kommunikációs rendszeren történő adattovábbítás lehetőségének valószínűségének a szorzata, így ez a szorzat legfeljebb a műhold rendszerének működési valószínűsége lehet.

Másik hiány az lehet, ha egy műhold rendelkezik kommunikációs rendszerrel, de valamelyik kapcsolatának nincsen fogadója. Ez felül becsülhető azzal, ha a kapcsolat fogadójának a földi állomást tekintjük, aminek a működési valószínűsége 1, ami legalább akkora, mint tetszőleges másik fogadó.



4.9. ábra. A legisebb ideális architektúra

4.4.2. Automatikus sztochasztikus analízis

Az analízishez a PRISM programot a generátoron belülről a parancssori interfésszel használjuk. Paraméterként megkapja az előző pontban előállított modellfájlt és a lekérdezéseket tartalmazó fájlt. A futáshoz az alábbi parancsot használjuk:

```
prism model.prism queries.props
```

Az adatok feldolgozását egy állapotgép végzi el. Az első állapotában az állapotgép arra vár, hogy olyan kimenet jöjjön, ami egy lekérdezést tartalmaz, ekkor kiolvassa a lekérdezés

paramétereit. A két fontos paraméter a lekérdezésben vizsgált időpillanat és az, hogy hány mérőműszer elérhetőségét vizsgáljuk.

A feldolgozás után átlép abba az állapotba, hogy egy eredményre vár. Ha felismer egy sort, ami eredmény, akkor kiolvassa a valószínűséget és súlyozza a hozzá tartozó ideális mérési hasznossággal. A kapott eredményt pedig hozzáadja az eddigi várható hasznossághoz. Ezután visszalép az előző állapotba.

Ha a használt process lezárja a kimenetét, ami azt jelenti, hogy az elemzés befejeződött, akkor visszaadja a kiszámított várható hasznosságot.

5. fejezet

Értékelés

Elvégeztem egy mérést, amelyben az architektúra szintézis során a parciális modellek megbízhatóságának becslésével állít elő architektúrákat. Ennek során az alábbi kérdésekre kerestem a válaszokat:

1. Hogyan változik a kapott architektúrák hasznoságának eloszlása optimalizáció nélküli szintézishez képest?
2. Hogyan változik a sikeresség a becslések használatával?
3. Hogyan részesedik a megoldási időből a parciális modellek alsó és felső becslése?

5.1. Felhasznált szakterület specifikus nyelvek

A kérdések megválaszolásához a háttérismeretek fejezetben bemutatott esettanulmányt használtam fel. Az esettanulmány metamodelljét az Ecore [2] felhasználásával adatam meg, jólformáltsági kényszereket VQL [5] nyelven fogalmaztam meg. A nézeti transzformációkat és az eredmények feldolgozását Xtend [6] nyelven implementáltam.

A generátorral háromféle optimalizációs szintézis folyamatot futtattam. NO optimalizációs során a generátor a szintézis során nem veszi figyelembe a hasznosságot, mint optimalizálandó paramétert, a sikerességének egyetlen feltétele, hogy a parciális modell konkretizálható legyen.

THR optimalizáció esetén a generátor figyelembe veszi a parciális modellek alsó és felső becsléseit, ennek a szintézis folyamatnak a során a kilépési feltétel a konkretizáláson kívül az, hogy az eredmény hasznosságának el kell érnie a kitűzött célértéke.

OPT optimalizációnál a lehető legnagyobb hasznosságú architektúrát keressük.

A szintézisek során költségfüggvénynek a csomópontokat használjuk. A szintézis során a generátor a futástól függően pontosan 8 vagy 11 új csomóponttal egészítheti ki a kiindulási parciális modellt.

A generátor konfigurációjában 8 csomópontos futásokra 300 másodperc időlimitet adtam meg, ami az előzetes futások alapján hozzávetőleg 2-3-szorosa volt az átlagosnak. Ekkor a futás során 120 szintetizálást próbál meg a generátor egymás után.

10 csomópontos futásokra a megnövekedett időigényre számítottam, ezért az időlimit itt 900 másodperc és futásonként 50 szintézist hajt végre a generátor.

Az esettanulmány missióidő paraméterét 1 órának állítottam be.

A küszöbértéket 8 csomópontos esetre 0.0135 értékűre állítottam, 11 csomópontos esetben pedig 0.0139 értékre állítottam.

optimalizáció	új csomópontok	időlimit	misszióidő	futások	küszöbérték
NO	8	300s	1h	120	-
OPT	8	300s	1h	120	-
THR	8	300s	1h	120	0.0135
NO	11	900s	1h	50	-
OPT	11	900s	1h	50	-
THR	11	900s	1h	50	0.0139

5.2. Mérési elrendezés

A méréseket egy HP Pavilion Power 15 típusú laptopon végeztem el. A számítások elvégzésére egy Intel i7-7700HQ (2.8 GHz, 8 szál) processzor áll rendelkezésre. A mérést végző laptop 16GB memóriával rendelkezik.

A mérést Ubuntu 20.04 LTS operációs rendszeren végeztem el. A generátor futtatása java virtuális gépen történt, a használt verzió az openjdk 10.0.8. Az sztochasztikus modellek elemzésre a PRISM modellellenőrző [3] program 4.6 verzióját használtam.

5.3. Eredmények

Az eredményekhez hatféle beállításokkal futtattam a generátort. A szám jelöli, hogy hány új csomópontot kellett a generátornak létrehoznia. Utána a betű kódok közül a NO jelenti, hogy a futás során a generátor nem használta a becsléseket, OPT jelöli, hogy a generátor az optimális megoldást kereste a becslések felhasználásával és a THR jelenti, hogy a generátornak egy küszöbértéket kellett elérnie.

5.3.1. Hogyan változik a kapott architektúrák hasznosságának eloszlása optimalizáció nélküli szintézishez képest?

A szintetizált architektúrák hasznosság eloszlása az 5.1 grafikonon láthatóak.

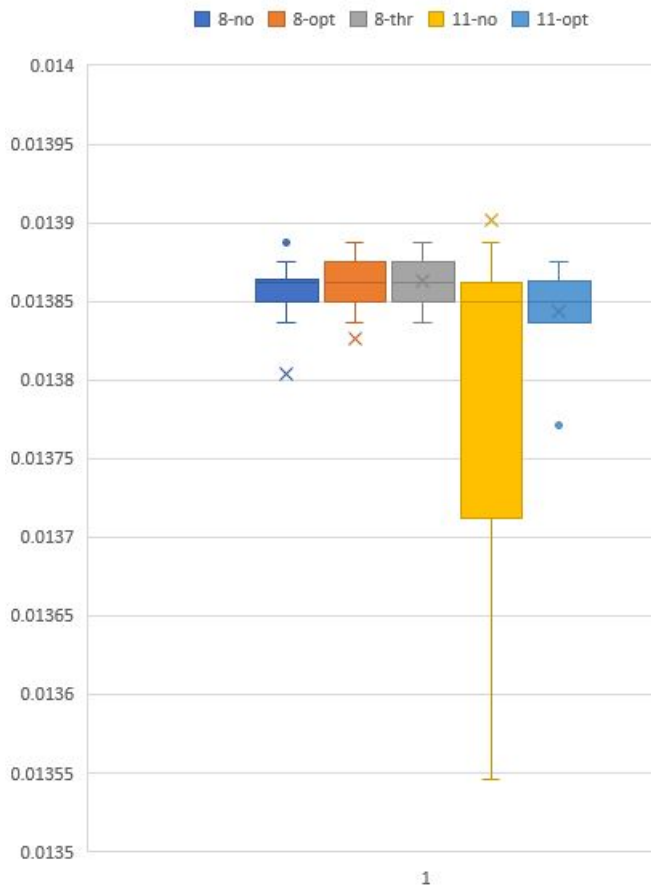
A 8 új csomópont esetére az optimalizáló futás jelentősen javít a kapott megoldások hasznosságának eloszlásán. Ugyanakkor az, hogy az eredmények nem csak az optimális értéknél vannak az azt bizonyítja, hogy a becslések implementációjában hiba van, ami ahhoz vezet, hogy a generátor eldobja az optimális megoldást.

A küszöbérték elérését célzó futás is kedvező, de ennek a fő oka, hogy a beállított 0.0135 hasznosság érték 8 csomópontos esetben nem fordulhat elő, így lényegében felesleges ellenőrizni.

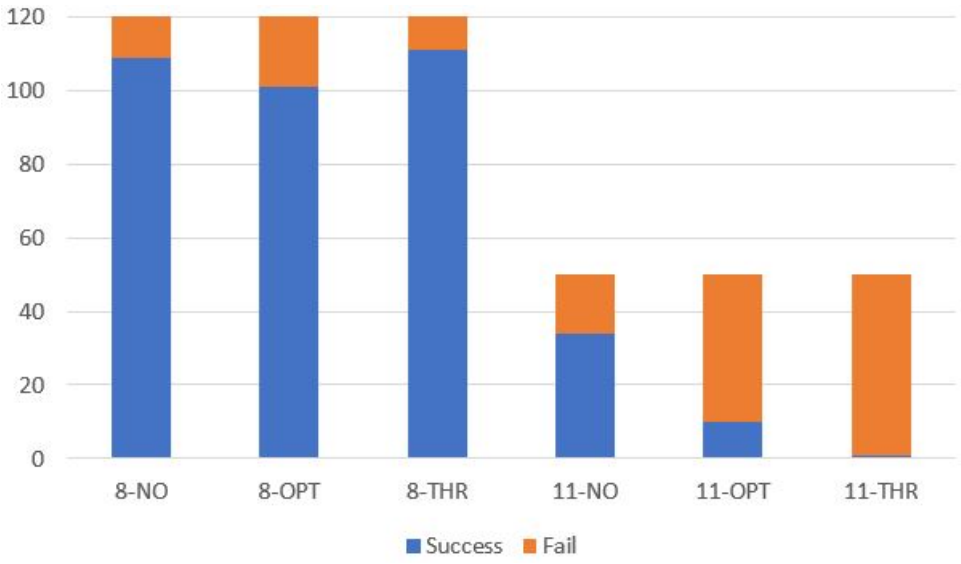
A 11 csomópontos esetek már érdekesebb eredményeket mutatnak. Optimalizáció nélkül az eredmények jelentős része rosszabb, mint a két műholdas esetek. Ennek oka, hogy 11 csomópontnál csak akkor lehet 3 mérőműszert hozzáadni a rendszerhez, ha egyik CommElement sem tartalmaz két kommunikációs rendszert. Továbbá már lehetőség van kedvezőtlen kommunikációs útvonalakat is használni.

Az optimalizáló futás a hibás implementációnak köszönhetően viszont csak kis mértékű romlást szenved el, a 8-NO esethez képest. Ennek oka, hogy a generátor törekszik a jó megoldások keresésére és így a rossz modellek közül sokat eldob.

A küszöbérték alapú optimalizáció egy esetben talált megoldást, ami még mindig nem az optimális, de attól már csak két műhold típusában tér el. Ennek hasznosság értéke 0.016315, ami tényleg nagyobb a beállított 0.0139-nél, ebből az látszik, hogy a tényleg csak olyan modelleket kapunk, amelyek megfelelnek a követelménynek, de hibás implementáció miatt a teljesség sérül.



5.1. ábra. A szintézis eredményeinek hasznosság eloszlása



5.2. ábra. A szintézisek sikerességének eloszlása

5.3.2. Hogyan változik a sikeresség a becslések használatával?

A szintézisek sikeressége az 5.2 grafikonon látható.

A minimális modellek esetén a sikeresség optimalizáló futás esetén a sikertelen lehutások száma 11 darabról 19 darabra nő, ami 70%-os növekedés, de az esetek több mint 80%-ában még mindig sikeres lesz, ami kis mértékű romlásnak tekinthető. A küszöbérték elérését célzó futásnál a becslés nélküli futáshoz képest javul az eredmény, de nem jelentősen. Ezt betudható annak, hogy csak 120 futásból származik a mérési eredmény.

A küszöbérték elérését célzó futásnál az eredmény nem jelentős, mivel nem lehetséges 8 műholdból olyan architektúrát előállítani, ami nem teljesíti a küszöbértéket, így a generátor a becslő eredményeit lényegében figyelmen kívül hagyja.

A 11 új csomópontot tartalmazó modelleknél az változás közel sem ennyire elhanyagolható. Optimalizáció során a sikeresség 20%-ra csökken, ami számottevő csökkenés. Ugyanakkor az eredmények minősége jobb, mint optimalizáció nélkül, így legalább lehet értelme a használatának.

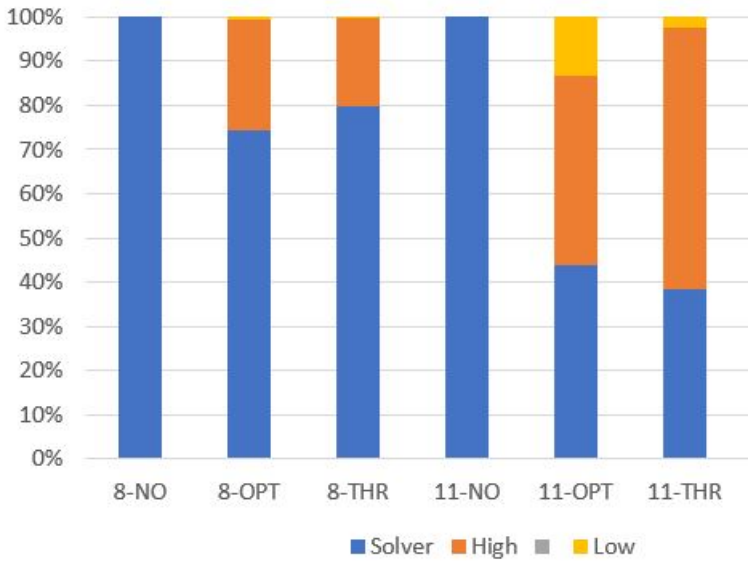
A küszöbérték futásnál a 2%-os sikeresség nem jó, ennek az lehet az oka, hogy a beállított küszöbértéket csak 3 mérőműszeres tényleges architektúrák érhetnek el, ugyanakkor a beállított küszöbérték kisebb, mint a két mérőműszeres ideális architektúrák által elérhető hasznosság, így a generátor nem tud elég hatékonyan vágni az állapottérből.

5.3.3. Hogyan részesedik a megoldási időből a parciális modellek alsó és felső becslése?

A szintézis során felhasznált idő eloszlása az 5.3 grafikonon láthatóak.

A mérés eredményéből egyértelműen látszik, hogy a csomópontszám növekedésével a becslő egyre hosszabb időt igényel a szintézisből. Ebből le lehet vonni azt a következtetést, hogy a generátor jó skálázhatóságához a becslő skálázhatósága is fontos.

Azt is érdemes észrevenni, hogy az alsó becslő számottevően kevesebb időt használ fel, mint a felső becslő. Ennek legvalószínűbb oka, hogy nem fut le a tényleges elemzés, ha nem lehetséges olyan eset, hogy legalább két mérőműszer elérhető.



5.3. ábra. Futásidő eloszlása a sikeres szintézisek során

5.4. A mérés helyességét befolyásoló negatív körülmények

A kezdeti tranziens jelenségek lehetőségét és a java szemétgyűjtő algoritmusának a hatását nem vettem figyelembe a szintézis során. Ennek oka, hogy 120 és 50 modell viszonylag kevés egy reprezentatív adathalmaz előállítására, így a pontosságot a több futástól vártam.

Pontatlanságot eredményező tényező, hogy a használt algoritmusok erre az esettanulmányra lettek specializálva, így más esettanulmánnyal a vizsgálat eltérő eredményeket adhat.

Súlyos hátráltató körülmény, hogy az operációs rendszer és a futáshoz használt ideiglenes fájlok tárolási helye egy USB 3.0 külső merevlemez, ami így jelentősen lassíthatja a fájlokhoz való hozzáférést. Továbbá nagyon kis valószínűséggel néha 5-10 másodperccel késlelteti az írási és olvasási műveleteket. Ennek hatását nem áll módomban kompenzálni.

6. fejezet

Összefoglalás

A dolgozatban kibővítettem az esettanulmányt, elkészítettem hozzá egy sztochasztikus modellt és a nézeti transzformációkat alsó és felső becsléshez továbbá ezeket integráltam a generátorba. Majd megvizsgáltam a generátor működést ennek a felhasználásával.

A jövőben fejlesztésre lehetőség, hogy a nézeti transzformációt a generátor állítsa elő a megadott bemenetek alapján. Ez egy jelentős lépés lenne a generátor tetszőleges esetben történő alkalmazhatósága felé.

További lehetőség, hogy a használt elemzések pontosságát javítjuk például intervallum logikával. Ha a következtetés helyes volt, hogy a hibák egyik oka az eredmények pontatlansága, akkor ez a fejlesztés kulcsfontosságú lehet a helyes működéshez.

Ugyanúgy lehetőség a generátor által a finomítás során hozott döntések hatásának figyelembevétele a szintézis során. Ezzel el lehetne kerülni olyan az eseteket, amikor az alsó becslés nagy eltérést mutat a tényleges legrosszabb esethez képest lehetséges. Ez elképzelhető, hogy javítana a skálázhatóságon.

Szintén lehetséges fejlesztés, olyan alsó becslés elkészítése, ami jobban közelíti a legrosszabb lehetséges eredményt. Itt kiemelt fontosságú a pontosság a korai lépések során.

Irodalomjegyzék

[1] *Eclipse Modeling Framework*.
URL https://www.eclipse.org/modeling/emf/docs/.

[2] *Eclipse Modeling Framework: Ecore*.
URL https://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html.

[3] *Prism Model Checker*. URL https://www.prismmodelchecker.org/.

[4] *Unified Modeling Language*. URL https://www.uml.org/.

[5] *The Viatra Query Lnguage*.
URL https://www.eclipse.org/viatra/documentation/query-language.html.

[6] *Xtend - Modernized Java*. URL https://www.eclipse.org/xtend/.

[7] Sven Apel: Performance feature interactions. 2020.
URL https://www.youtube.com/watch?v=gFTuoiH6e4A.

[8] Algirdas Avizienis – Jean-Claude Laprie – Brian Randell – Carl E. Landwehr: Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1. évf. (2004) 1. sz., 11–33. p.
URL https://doi.org/10.1109/TDSC.2004.2.

[9] Milan Ceska – Nils Jansen – Sebastian Junges – Joost-Pieter Katoen: Shepherding hordes of markov chains. In Tomás Vojnar – Lijun Zhang (szerk.): *Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II*, Lecture Notes in Computer Science konferenciasorozat, 11428. köt. 2019, Springer, 172–190. p. URL https://doi.org/10.1007/978-3-030-17465-1_10.

[10] Leonardo Mendonça de Moura – Nikolaj Bjørner: Z3: an efficient SMT solver. In C. R. Ramakrishnan – Jakob Rehof (szerk.): *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, Lecture Notes in Computer Science konferenciasorozat, 4963. köt. 2008, Springer, 337–340. p.
URL https://doi.org/10.1007/978-3-540-78800-3_24.

[11] Sebastian J. I. Herzig – Sanda Mandutianu – Hongman Kim – Sonia Hernandez – Travis Imken: Model-transformation-based computational design synthesis for mission architecture optimization. 2017.

[12] Daniel Jackson: Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11. évf. (2002) 2. sz., 256–290. p.
URL https://doi.org/10.1145/505145.505149.

[13] Anne Martens – Heiko Koziolek – Steffen Becker – Ralf H. Reussner: Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In Alan Adamson – Andre B. Bondi – Carlos Juiz – Mark S. Squillante (szerk.): *Proceedings of the first joint WOSP/SIPEW International Conference on Performance Engineering, San Jose, California, USA, January 28-30, 2010* (konferenciaanyag). 2010, ACM, 105–116. p.
URL https://doi.org/10.1145/1712605.1712624.

[14] K. Marussy – O. Semerath – D. Varro: Automated generation of consistent graph models with multiplicity reasoning. *IEEE Transactions on Software Engineering*, 2020., 1–1. p.

[15] Oszkár Semeráth – Aren Babikian – Sebastian Pilarski – Dániel Varró: Viatra solver: A framework for the automated generation of consistent domain-specific models. In *41st International Conference on Software Engineering* (konferenciaanyag). Montreal, Canada, 2019. 2019, ACM/IEEE, ACM/IEEE. URL https://2019.icse-conferences.org/event/icse-2019-demonstrations-viatra-solver-a-framework-for-the-automated-generation-of

[16] Oszkár Semeráth – András Szabolcs Nagy – Dániel Varró: A graph solver for the automated generation of consistent domain-specific models. In *40th International Conference on Software Engineering (ICSE 2018)* (konferenciaanyag). Gothenburg, Sweden, 2018. 2018, ACM, ACM.

[17] Oszkár Semeráth – Aren A. Babikian – Anqi Li – Kristóf Marussy – Dániel Varró: Automated generation of consistent models with structural and attribute constraints. In Eugene Syriani – Houari A. Sahraoui – Juan de Lara – Silvia Abrahão (szerk.): *MODELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020* (konferenciaanyag). 2020, ACM, 187–199. p.
URL https://doi.org/10.1145/3365438.3410962.

[18] Dániel Varró – Oszkár Semeráth – Gábor Szárnyas – Ákos Horváth: *Towards the Automated Generation of Consistent, Diverse, Scalable and Realistic Graph Models*. Springer LNCS sorozat, 10800. köt. 2018, Springer.