XI. FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2006. március 24-25.

GRÁFTRANSZFORMÁCIÓS BENCHMARKOK DEFINIÁLÁSA

Varró Gergely

Abstract. An overview on the paradigm of graph transformation is given, which can be used for a rule-based description of simulations. The specification tasks that are necessary to quantitatively compare the run-time performance of graph transformation tools are presented. The process of graph transformation benchmark specification is also demonstrated by using a mutual exclusion algorithm as a running example.

Összefoglaló. A szimulációk szabály alapú leírására szolgáló gráftranszformáció paradigmájának áttekintése után, ismertetem, hogy mik szükségesek ahhoz, hogy olyan feladatokat lehessen specifikálni, melyekkel gráftranszformációs eszközök teljesítményét lehet összehasonlítani. Végül a gráftranszformációs benchmarkok specifikálási folyamatának egy példán keresztül történő szemléltetése következik.

1. Bevezetés

A gráftranszformáció [2,3] egy szabály alapú, jól definiált szemantikával rendelkező paradigma, mely kitűnően használható grafikus nyelven megfogalmazott rendszermodellek manipulációjára, így a manapság oly népszerű modellvezérelt rendszerfejlesztésben előkerülő modelltranszformációk leírására is. A gráftranszformáció területén végzett több évtizedes kutatás eredményeként napjainkban elég széles a gráftranszformációs eszközök skálája, így az utóbbi években felmerült az igény arra, hogy ezek teljesítménye (futási ideje) és az általuk biztosított különböző optimalizálási lehetőségek hatásai kvantitatív módon összehasonlíthatóak legyenek. Ezen célkitűzés eléréséhez elkészült néhány gráftranszformációs tesztfeladat igen tömör specifikációja [4] és az implementálásukat követően számos összehasonlító mérés is [5]. Jelen cikk célja, hogy egy példán keresztül szemléltesse azt, hogy hogyan lehet egy gráftranszformációs tesztfeladatot megfogalmazni.

2. Gráftranszformáció

Egy elosztott, kölcsönös kizárást biztosító (Mutex) algoritmuson [1] bemutatom a modellezési nyelvek specifikálásának alapjait, majd az így definiált modelleken végrehajtható szimulációk gráftranszformációval történő leírásának áttekintése következik.

1/a ábra: A kölcsönös kizárást biztosító algoritmus metamodellje

1/b ábra: Az ALAP nevű tesztfeladat kezdeti modellje az $N=4$ esetben

1. ábra: Példa metamodellre és modellre

Metamodellek. A modellezési nyelv absztrakt szintaxisát **metamodell** (1/a ábra) adjuk meg. Ennek formális megadása egy **típus gráf** segítségével történik, melynek csúcsait **osztályoknak** (pl. *Processz*, *Erőforrás*), éleit pedig **asszociációknak** (pl. *igény*, *token*) hívják. Az osztályoknak lehetnek **attribútumai**, melyek az adott osztály valamilyen tulajdonságait fejezik ki. Az osztályok között **öröklés** definiálható, mely azt jelenti, hogy a leszármazott osztály minden olyan tulajdonsággal fog rendelkezni, mely az ősében is megvolt, de ezen kívül tartalmazhat további attribútumokat is. (A példánk metamodelljében nincs attribútum és öröklés sem.)

Modellek. A **modell** (1/b ábra) a modellezési nyelv segítségével definiált konkrét rendszert írja le, mely egy jól formált példánya a metamodellnek. A modell szintén egy gráf, melynek csúcsait **objektumoknak** (pl. *p1*, *p2*, *e1*), éleit pedig **linkeknek** (pl. *k1*, *k2*, *h1*) hívják. Az objektumok és a linkek rendre a metamodellbeli osztályok és asszociációk példányaiként jelennek meg. A modellek szintjén az öröklés azt jelenti, hogy a leszármazott osztály példányaikat minden olyan szituációban lehet használni, ahol egy őosztálybeli példány van előírva.

Gráftranszformáció. A **gráftranszformációs szabály** modellek fejlődését és változását tudja leírni egy, a metamodell szintjén levő grafikus nyelvet használva. Egy gráftranszformációs szabály egy **baloldalból** és egy **jobboldalból**, valamint egy **negatív alkalmazási feltételből** áll, melyek mindegyike egy olyan gráf, melynek minden csúcsa és éle egy-egy változóként fogható fel. Ezek típusa rendre egy-egy metamodellbeli osztály illetve asszociáció. Egy adott gráfon belül a változók elnevezésének egyedinek kell lenni, viszont a különböző gráfokban megengedett az, hogy a változók neve megegyezzen. Ilyen esetekben az azonos nevű változók azonos entitást jelölnek.

Egy **szabály egy adott modellen való alkalmazása** a baloldal egy modellbeli illeszkedésének a jobboldal egy képre való lecserélését jelenti. Ez két lépésben történik: (1) a **mintaillesztési fázisban** megkeressük a baloldal egy típushelyes, modellbeli illeszkedését, és leellenőrizzük a negatív alkalmazási feltétel megthiúsulását; majd (2) a **manipulációs fázisban** kitöröljük azon modellrészeket, melyek olyan baloldalbéli csúcsra vagy élre illeszkednek, melyek a jobboldalon nem szerepelnek, végül létrehozunk olyan elemeket, melyek kizárólag a jobboldalon szereplő változókhoz tartoznak.

A **gráftranszformáció** a szabályok alkalmazásának egy olyan sorozata (transzformációs szekvencia), melyet egy kezdeti modellből indítva lefuttatunk.

Példa. A Mutex algoritmus 13 szabállyal írható le, melyek közül 4 a 2. ábrán szerepel.

2. ábra: Gráftranszformációs szabályok

A fenti ábrákon az X-ek a negatív alkalmazási feltételeket jelölik. A *Felszabadít* szabály (2. ábra bal felső sarka) akkor alkalmazható, ha a modellben van olyan erőforrás, melyet olyan processz használ, mely nem jelentett be igényt semmilyen erőforrásra. Az 1/b ábra modelljén a *Felszabadít* szabály 4 különböző helyen ($e1, h1, p1$; $e2, h2, p2$; $e3, h3, p3$; $e4, h4, p4$) is alkalmazható.

3. Teljesítményvizsgálathoz kapcsolódó fogalmak

Jelen fejezet bemutatja a teljesítményvizsgálathoz kapcsolódó főbb fogalmakat, és azok példaalgoritmusunkhoz kapcsolódó reprezentánsait.

- A **forгатókönyv** a gráftranszformáció paradigmájának egy széles alkalmazási területét jelenti. [4] három ilyen alkalmazási területet említ: a modellanalízist, a modelltranszformációt, és a dinamikus operációs szemantikával rendelkező grafikus nyelvek szimulációját. A forгатókönyvet kizárólag informális jellemzők írják le. *A Mutex algoritmus egy grafikus nyelven leírt szimulációt valósít meg.*
- A **tesztfeladat** az adott alkalmazási terület egy széles körben ismert, gráftranszformációval leírható feladata, mely értelemszerűen teljesíti a forгатókönyv összes informális jellemzőjét. Egy tesztfeladatnál annak metamodellje és gráftranszformációs szabályai rögzítettnek tekinthetők, míg a kiindulási modell és a konkrét transzformációs szekvencia nem definiált. *A Mutex algoritmus esetén az 1/a és a 2. ábra jelenti a tesztfeladat definícióját.*
- Egy tesztfeladat több **teszthalmazból** áll. A teszthalmaz egy parametrikus, de egyébként teljes specifikációként fogható fel. Ez azt jelenti, hogy nem csak a feladathoz kapcsolódó metamodel és gráftranszformációs szabályok adóttak, hanem a kiindulási modell és a transzformációs szekvencia struktúrája is, melyek numerikus paramétereket tartalmaznak. Egy

adott teszthalmaz esetén az eszközszer specifikus optimalizálási stratégiák [5] be- ill. kikapcsolása még nem eldöntött. A *Mutex tesztfeladat ALAP* nevű teszthalmaza esetén a processzek száma (N) a paraméter. A kiindulási modell struktúráját az 1/b ábra szemlélteti az $N=4$ esetben, míg a transzformációs sorozat a Felszabadít szabály N -szeri alkalmazásával kezdődik, melyet az Átad, Igényel, Lefoglal szabályok N -szer és ilyen sorrendben történő alkalmazása követ.

- Egy **tesztesetben** a kiindulási modell és a transzformációs szekvencia még mindig csak paraméteresen adott, viszont az eszközszer specifikus optimalizálási stratégiák [5] bekapcsoltsági állapota már rögzített. Hely hiányában ezeket a stratégiákat nem ismertetem, ugyanakkor érdemes megemlíteni, hogy a bármilyen optimalizálás nélküli teszteset igen gyakori szokott lenni, mivel ez minden eszközön futtatható.
- A **futási sorozatnál** már a numerikus paraméterek is rögzítettek, ezáltal teljesíti a legfőbb gráftranszformációs teljesítményparaméter, a futási idő vizsgálatának azon követelményét, mely szerint a méréseknek megismételhetőeknek kell lenniük. Az *ALAP tesztfeladat* egy rögzített tesztesetének futási sorozata az N értékének (pl. 4-re történő) lerögzítése után kapható.

4. Irodalomjegyzék

- [1] R. Heckel. Compositional verification of reactive systems specified by graph transformation. In E. Astesiano, editor, *Fundamental Approaches to Software Engineering: First International Conference, FASE*, volume 1382 of *LNCIS*, pages 138-153, Springer-Verlag, 1998.
- [2] A. Pataricza, T. Bartha, Gy. Csértán, Sz. Gyapay, I. Majzik, D. Varró: *Formális módszerek az informatikában*, Typotex, 2004.
- [3] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, volume 1: Foundations*. World Scientific, 1997.
- [4] G. Varró, A. Schürr, D. Varró: Benchmarking for graph transformation. Technical report, Budapest University of Technology and Economics, 2005. <http://www.cs.bme.hu/~gervarro/publication/TUB-TR-05-EE17.pdf>
- [5] G. Varró, A. Schürr, D. Varró: Benchmarking for graph transformation. In *Proc. of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 79-88, 2005.

Név: **Varró Gergely**

Munkahelyi cím: H-1117, Budapest, Magyar tudósok körútja 2.

Telefon: +36-1-463-2585

Fax: +36-1-463-3157

E-mail: gervarro@cs.bme.hu