Távközlési szoftverek

Tesztelés kiterjesztett véges állapotgépekkel

Kovács Gábor

Miről lesz szó?

- FSM általánosításai
- Nemdeterminisztikus FSM
- Egyéb FSM-ek
- EFSM
- CFSM
- EFSM-ek tesztelése

Felhasznált irodalom

- Lee, D., Yiannakakis, M.: Principles and methods of testing finite state machines – a survey. Proceedings of the IEEE 84(8) (1996) 1090–1123
- Bourhfir, C., Dssouli, R., and Aboulhamid, El M.: Automatic Test Generation for EFSM-based Systems, Technical Report, 1997

FSM általánosítások 1

- Eddigi feltételezések az állapotgépekről: determinisztikus, minimális, erősen összefüggő, teljesen specifikált
- A konformancia reláció értelmezéséhez szükséges feltételek:
  - A specifikáció állapotgépe erősen összefüggő
  - A specifikáció állapotgépe minimális
  - Az implementáció nem változik a megfelelőség-vizsgálat alatt
  - A specifikáció és az implementáció azonos I/O ábécével rendelkeznek
  - Az implementációnak nincs több állapota, mint a specifikációnak

FSM általánosítások 2

- Több állapot: az implementáció FSM-je nem minimális Δ további állapotot tartalmaz
- A konformancia reláció egyik feltétele, hogy a specifikációs és implementációs állapotgépek azonos számú állapottal rendelkeznek – ez sérül
- A specifikációs és implementációs gép az extra állapotok kivételével egy állapotátmenettől eltekinve izomorf
- Ez azt jelenti, hogy az implementáció funkcionális dekompozíciója nem volt teljes, ugyanazt a funkciót több helyen is megvalósították
- Következmények:
  - A redundáns Δ klikkbe vezető útvonal hossza legalább $p^{\Delta+1}$ (Moore).
  - Az ellenőrző sorozat (Checking Sequence) hossza p^{Δ}-szorosára nő (Vasilevskii), és $\Omega(p^{\Delta+1}n^3)$ lesz.

FSM általánosítások 3

- Részben specifikált véges állapotgépek
- $\exists s \in S$ és $\exists i \in I : \nexists \delta(s, i)$
- Ekkor legyen: $\delta(s, i) = s$ és $\lambda(s, i) = \varepsilon$
- Ez az ún. implicit állapotátmenet. Ennek az alkalmazását teljesség feltételezésnek hívják.
- Alternatív, nem használt megoldás: $\delta(s, i) = s_{trap}$ és $\lambda(s, i) = \varepsilon$
- Alternatív megoldás – később – a gyenge konformancia: $\forall x \in I_s*, \lambda_s(s_1, x) = \lambda_i(q_1, x)$

FSM általánosítások 4

- Nemdeterminisztikus véges állapotgépek
- $\exists s, s', s'' \in S$ és $\exists i \in I : \delta(s, i) = s', \delta(s, i) = s''$ és $s' \neq s''$
- A nemdeterminizmus forrása lehet belső (random döntések), külső (versenyhelyzet) – később
- A nemdeterminisztikus állapotgép átalakítható determinisztikus állapotgéppé hatványhalmaz konstrukcióval: $\forall s \in S : \sigma \subseteq 2^S : \forall s' \in S : \delta(s, i) = s', s' \in \sigma$.
- Az I/O ábécé változatlan, az állapotok száma legrosszabb esetben 2^n-re nő
- A D-, U-, W- stb. módszerek értelmezhetőek NFSM-re is. Alapelv FSM-mé alakítás nélkül: determinisztikus útvonalak halmazának meghatározása, a nemdeterminisztikus átmenetek „elég sokszori'' bejárása

FSM általánosítások 5

- Időzített véges állapotgépek
- $TFSM = (S, I, O, h, s_0)$, $h \in S \times (I \times \Pi) \times (O \times \aleph) \times S$, ahol Π input őrfeltétel, $\aleph$ output őrfeltétel, úgy hogy $g \in \Pi$, illetve $f \in \aleph = [min, max], min \leq max$, így $(s, (i, g), (o, f), s') \in h$.
- Az időzített véges állapotgép átalakítható FSM-mé az I/O ábécé particionálásával őrfelétételek összes lehetséges részhalmaza alapján figyelembe véve a Π és $\aleph$ őrfeltételek értékkészletének természetes rendezését. Tehát az állapottérrobbanást itt az I/O ábécé méretének elszállása okozza.
- A D-, U-, W- stb. módszerek értelmezhetőek TFSM-re is. Alapelv: FSM-mé kiterítés.

FSM általánosítások 6

- Véletlen események figyelembe vétele egy P eloszlásfüggvénnyel
- $PFSM = (S, I, O, T, P)$, $T \subseteq S \times I \times O \times S$, ahol $P(t) \in [0, 1], t \in T$, és $\forall s \in S, i \in I : \sum_{s,i} P(s, i, o, s') = 1$.
- A PFSM valójában egy Markov döntési folyamat.
- A PFSM tesztelése hasonlít az NFSM teszteléshez.

EFSM 1

- Minden egyes lokális változó értelmezési tartományának minden bitje duplikálja az állapotteret
- Az FSM állapotai számának csökkentése változók, értékadások és feltételek bevezetésével
- Az FSM konfigurációja: az állapot és a változók vektorának aktuális pillanatképe
- Analógia: állapot $\sim$ változó, bemenet $\sim$ feltétel, kimenet $\sim$ értékadás
- $EFSM = (S, V, I, O, T)$, ahol $T \subseteq (S \times V) \times (I \times P*) \times (O \times A*) \times (S \times V)$
- Gráf reprezentáció:
  - FSM élek címkézése: input (I), őrfeltétel-sorozat (P), outputsorozat ($O*$), értékadás-sorozat ($A*$)
  - Új csomóponttípus bevezetése feltételekre, az abból kivezető éleket a lokális feltétel triggereli. Az élek címkézése ekkor: input (I) vagy lokális feltétel (P), outputsorozat ($O*$), értékadás-sorozat ($A*$)

EFSM 2

- Az üzenetformátum minden egyes mezője értelmezési tartományának minden bitje duplikálja az I/O ábécét
- Paraméterezett EFSM: az I/O ábécé méretének csökkentése
- Minden input és output szimbólum, feltétel és értékadás paraméterezhető a V változóhalmaz tetszőleges részhalmazával
- Analógia:
  - Input: függvény formális paraméterlistája vagy PDU/SDU üzenetformátum
  - Output: függvény aktuális paraméterlistája vagy egy konkrét PDU/SDU
  - Feltétel: maga a feltételes kifejezés
  - Értékadás: a balérték és a jobb oldali kifejezés

EFSM 3

- Példa: bármely programozási nyelv egy függvénye egy EFSM
  - Állapotok: mozgás a stacken
  - I/O ábécé: kimenő függvényhívások és azok visszatérése, a függvény meghívása és visszatérése
  - Változók: lokális változók, formális paraméterek
  - Feltételek: if, switch C-ben
  - Értékadás

CFSM

- A rendszer dekompozíciója egymással kommunikáló FSM-ek halmazára, pl. protokoll alrétegek vagy több teszter
- Az FSM-ek ún. portokon keresztül kommunikálnak egymással (vö. osztályok közötti kapcsolatok), egy porton a rendszer teljes I/O ábécéjének egy részhalmaza küldhető át
- Probléma: az egyik állapotgép egy állapotátmenet során két portot is aktivál üzenetküldés céljából. A kapuk késleltetése miatt a küldés sorrendje nem feltétlenül egyezik meg az üzenetet fogadó FSM-ek feldolgozási sorrendjével. Ez az egész rendszert nézve nemdeterminisztikus viselkedéshez vezethet. Ez az ún. versenyhelyzet. Pl. TCP Karn-algoritmus.
- A CFSM átalakítható NFSM-mé: szorzatállapotgép képzésével.
- A CFSM átalakítható EFSM-mé: egy változóval kódolhatjuk a komponens FSM-eket.

EFSM-ek tesztelése 1

- Az FSM-mé kiterítés nem működik
- Vezérlési és adatfolyam szétválasztása
- A D-, U-, W- és HIS-módszerek teljes FSM lefedést biztosítanak a vezérlési folyamra, a TT-módszer csak részlegeset
- Adatfolyam-tesztelés:
  - A használandó útvonalak halmazának meghatározása
  - Ugyanazon vezérlési folyam tesztesetek végrehajtása üzenetek más-más paraméterezésével
  - Célja az őrfeltételek minden kombinációjának triggerelése

EFSM-ek tesztelése 2

- Szimbolikus állapottér: $(I \cup O)*$
- Keresés:
  - Véletlen séta
  - Irányított séta
  - Kimerítő keresés
- Irányítás metrikái
  - Állapot, állapotátmenet, útvonal, elágazás, az FSM modellje szöveges reprezentációjának sora, hiba, szimbólum stb. lefedettségi aránya

EFSM-ek tesztelése 3

- Útvonal alapú stratégia: minden elágazást minden irányban legalább egyszer be kell járni, ezért címkézést vezetünk be
- Definíció: $\mathsf{A-Use}(x)$, ha x változóhoz új értéket rendelünk
- Definíció: $\mathsf{I-Use}(x)$, ha x-hez egy input üzenet aktuális paramétere rendel értéket
- Definíció: $\mathsf{P-Use}(x)$, ha x egy feltétel kifejezésében szerepel
- Definíció: $\mathsf{C-Use}(x)$, ha x egy értékadás jobb oldalán lévő kifejezésben szerepel
- Definíció: $\mathsf{def-clear-path}((t_1,\ldots,t_k),x)$, ha a $(t_1,\ldots,t_k)$ útvonal t_i egyik eleme sem bír $A-Use(x)$ vagy $I-Use(x)$ címkével
- Definíció: $\mathsf{du-path}((t_1,\ldots,t_k),x)$, ha a $(t_1,\ldots,t_k)$ útvonal t_i eleme $\mathsf{P-Use}(x)$ vagy $\mathsf{C-Use}(x)$ címkével bír
- Az egyszerűség kedvéért feltételezzük, hogy minden változó globális

EFSM-ek tesztelése 4

- Definíció-használat gráf G definiálható az EFSM útvonalain $*-\mathsf{Use}$ csomóponthalmazzal és változókkal, mint élekkel
- Az útvonalak P halmaza akkor teljes, ha növekvő erősségi sorrendben a következő feltételek teljesülnek:
  - G minden csomópontja szerepel P-ben
  - G minden éle szerepel P-ben
  - G minden csomópontjából létezik $\mathsf{def}-\mathsf{clear}-\mathsf{path}$ P-beli útvonak egy $\mathsf{P}-\mathsf{Use}(x)$ vagy $\mathsf{C}-\mathsf{Use}(x)$ csomópontba
  - G minden csomópontjából léteznek $\mathsf{def}-\mathsf{clear}-\mathsf{path}$ P-beli útvonalak minden $\mathsf{P}-\mathsf{Use}(x)$ csomópontba
  - G minden csomópontjából léteznek $\mathsf{def}-\mathsf{clear}-\mathsf{path}$ P-beli útvonalak minden $\mathsf{P}-\mathsf{Use}(x)$ és $\mathsf{C}-\mathsf{Use}(x)$ csomópontba
  - P tartalmazza minden változó összes definícióját
  - P a G gráf minden útvonalát tartalmazza. Ebből végtelen sok lehet tekintettel a ciklusokra.

EFSM-ek tesztelése 5

- Az útvonalak az adatfolyam-tesztelésnek köszönhetően rendelkezésre állnak, azonban a változók értékeit úgy kell beállítanunk, hogy az útvonal mentén teljesüljenek az őrfeltételek
- Tesztadatok kiválasztása
  - Mi legyen az input üzenet paraméterlistája?
  - Mi az arra adott válaszüzenet paraméterlistája?

EFSM-ek tesztelése 6

- Módszerek
  - Szimbolikus végrehajtás: az EFSM futtatása egy szimulációs környezetben
  - ILP, ha csak egész és boolean típusok, és additív operátorok vannak
  - Constraint Logic Programming (CLP): az útvonalak mentén összegyűjtött feltételekből összeállított egyenlőtlenségrendszer kielégítése logikai programozással (Prolog), a probléma maga NP-nehéz
  - Mutáció elemzés: hibamodell alapján hibák beszúrása az EFSM-be, az a tesztadathalmaz a jó, amelyik az összes hibát megtalálja
  - Heurisztikus és random módszerek
  - Irányított módszerek a tesztcélok definíciójával