Tesztsorozat generálás

Csöndes Tibor
Ericsson Kft., R&D,
BME-TMIT
Tibor.Csondes@ericsson.com,
csondes@tmit.bme.hu

Miről lesz szó?

- Mealy modell
- Gráfelméleti alapfogalmak
- FSM (gráf és táblázatos reprezentáció)
- Transition Tour (TT) módszer
- Distinguishing Sequence (DS) módszer
- Characterizing sequence/set vagy W módszer
- Unique Input/Output (UIO) sorozatok

Ajánlott irodalom

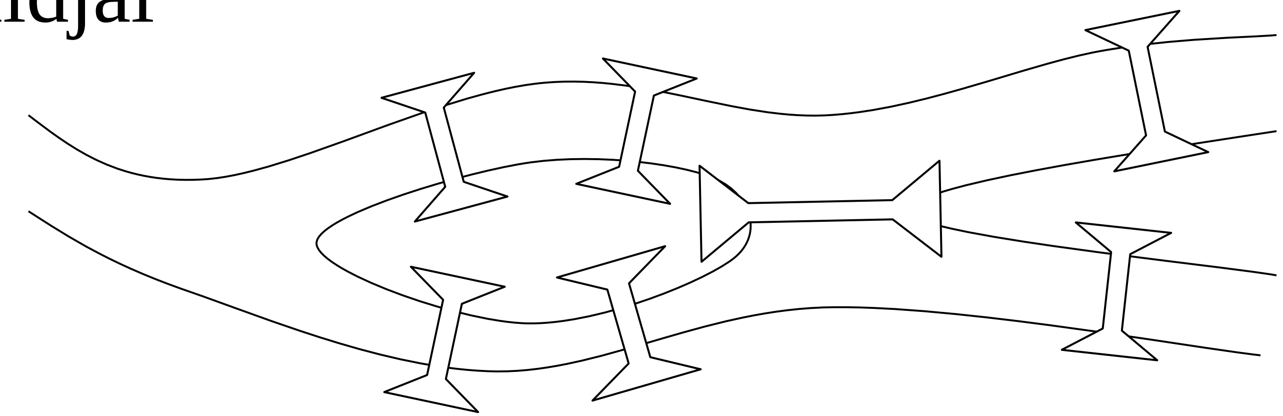
- G.J. Holzmann, Design and Validation of Computer Protocols, Prentice-Hall Inc., 1991.
- Andrásfai Béla: Gráfelmélet, Polygon, 1997.
- Katona Gyula Y., Recski András, Szabó Csaba: Gráfelmélet, algoritmuselmélet és algebra, 1997, tanszéki jegyzet
- Xiao Sun, Yinan Shen, Chao Feng, Fabrizio Lombardi: Protocol Conformance Testing Using Unique Input/Output Sequences, World Scientific, 1997. ISBN 9810228325
- ISO-IEC 9646, Conformance Testing Methodology and Framework

Általános bevezetés

- Mealy modell (Táblázat - gráf)
- Példa
- Nem determinisztikus viselkedés

Gráfelméleti alapfogalmak - 1

- 1736 Euler: Königsberg (Kalinyingrád) hídjai



Hogy lehet olyan sétát tenni, hogy közben mind a hét hídon pontosan egyszer haladjunk át?

Gráfelméleti alapfogalmak - 2

<u>Def:</u> *Egy* **gráf (graph)** *egy rendezett pár, G=(V, E), ahol V egy nemüres halmaz, E pedig ebből a halmazból képezhető párok egy halmaza. V elemeit* **pontoknak** *vagy* **csúcsoknak (vertex)**, *E elemeit* **éleknek (edge)** *nevezzük.*

Ha az $e \in E$ *az* $\{u, v\}$ *párnak felel meg, akkor ez a két pont e* **végpontja**. *Ha* $u = v$ *akkor e* **hurokél**.

Gráfelméleti alapfogalmak - 3

Def: *Egy pont* **izolált pont**, *ha nincsen vele szomszédos másik pont, vagyis nem illeszkedik egyetlen élre sem.*

Def: *Egy pontra illeszkedő élek száma a pont* **fokszáma (degree)**. *Az u pont fokszámát d(u)-val jelöljük.*

Egy esetleges hurokél kettővel növeli a fokszámot.

Gráfelméleti alapfogalmak - 4

Def: **Irányított gráf (directed graph or digraph)** *melynek élei nem {u,v} alakú rendezetlen párok, hanem (u,v) alakú rendezett párok. Egy ilyen (u,v) élnek u a* **kezdőpontja**, *v a* **végpontja**.

Def: *Legyen G=(V, E) egy cimkézett irányíott gráf. Egy u-ból v-be mutató élet a következő hármas reprezentál:* (u, v, L_k), *ahol* L_k *egy megkülönböztető* **címke (label)**. *Ezt hívhatjuk* **költségnek (cost)** *is.*

Gráfelméleti alapfogalmak - 5

Irányítatlan gráf - Undirected graph

Irányított gráf - Directed graph or Digraph

Cimke - Label

Be-fok ($d_{be}(u)$) - In-degree

Ki-fok ($d_{ki}(u)$) - Out-degree

Gráfelméleti alapfogalmak - 6

Def: *A* $G' = (V', E')$ *gráf a* $G = (V, E)$ *gráf* **részgráfja (subgraph)**, *ha* $V' \subseteq V, E' \subseteq E$ *valamint egy pont és egy él pontosan akkor illeszkedik egymásra* G'*-ben, ha* G*-ben is illeszkedők.*

Def: *Ha* E' *azokból az* E*-beli élekből áll, amelyeknek mindkét végpontja* V'*-ben van, és* E' *az összes ilyen élet tartalmazza, akkor* G' *a* G *gráf* V' *által* **feszített részgráfja (vertex-induced subgraph)**.

Gráfelméleti alapfogalmak - 7

Def: *A* $G' = (V', E')$ **feszítő-részgráfja (edge-induced subgraph)** $G = (V, E)$ *-nek, ha* $E' \subseteq E$ *valamint* V' *az* E'*-re illeszkedő pontok halmaza.*

Def: *A* $G' = (V', E')$ **teljes feszítő-részgráfja (edge-induced spanning subgraph)** $G = (V, E)$ *-nek, ha* G' *feszítő részgráf és* $V' = V$.

Gráfelméleti alapfogalmak - 8

<u>Def:</u> *Egy* $(v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k)$ *sorozatot* **élsorozatnak (walk or tour)** *nevezünk, ha* e_i *a* v_{i-1}*-t és* v_i *-t összekötő él. Ha* $v_0 = v_k$*, akkor az élsorozat* **zárt (closed walk)**.

<u>Def:</u> *Ha a csúcsok mind különbözőek, akkor egy* **utat** *definiáltunk.*

<u>Def:</u> *Ha* $v_0 = v_k$ *és különben a csúcsok mind különbözőek, akkor ez egy* **kör** *a gráfban.*

Gráfelméleti alapfogalmak - 9

Def: *Egy gráf* **összefüggő**, *ha bármely két pontja egymásból úttal elérhető.*

Def: *Egy irányított gráf* **unilaterálisan összefüggő**, *ha létezik irányított út bármely két pontja között.*

Def: *Egy irányított gráf* **gyengén összefüggő**, *ha elhagyva az élek irányítottságát a kapott irányítatlan gráf összefüggő marad.*

Gráfelméleti alapfogalmak – 10

Def: *Irányított gráfban egy* $(v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$ *utat akkor hívunk* **irányított útnak**, *ha* $e_1 = (v_0, v_1), e_2 = (v_1, v_2), \ldots, e_k = (v_{k-1}, v_k)$.

Def: *Egy irányított gráf* **erősen összefüggő**, *ha bármely pontjából bármely más pontjába vezet irányított út.*

Gráfelméleti alapfogalmak – 11

Gyengén összefüggő - Weakly connected

Unilaterálisan összefüggő - Unilaterally connected

Erősen összefüggő - Strongly connected

Példa: Fig. 5.

Gráfelméleti alapfogalmak – 12

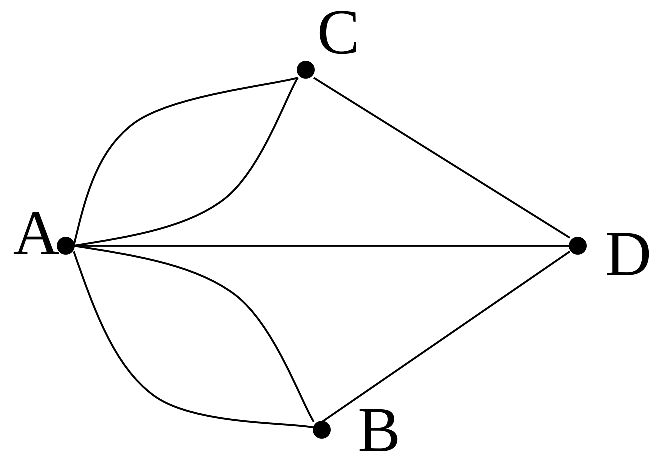
Def: *A G gráf* **Euler-körének** *nevezünk egy zárt élsorozatot, ha az élsorozat pontosan egyszer tartalmazza G összes élét. Ha az élsorozat nem feltétlenül zárt, akkor* **Euler-utat (Euler Tour)** *kapunk.*

Tehát minden Euler-kör egyben Euler-út is!

Def: **Euler gráf** *az ami Euler kör is egyben.*

Gráfelméleti alapfogalmak - 13

Tétel: *Egy G gráfban akkor és csak akkor van Euler-kör, ha G minden pontjának fokszáma páros, és G összefüggő.*



Tehát a Königsbergi hidak problémája nem megoldható: nem létezik olyan út amely egyetlen egyszer megy át minden hídon.

Gráfelméleti alapfogalmak - 14

Def: *Egy* $G=(V,E)$ *irányított gráf* **szimmetrikus**, *ha minden* $u \in V$ *csúcsra* $d_{ki}(u)=d_{be}(u)$.

Gráfelméleti alapfogalmak - 15

Tétel: *Egy* $G=(V,E)$ **irányított gráfban** *akkor és csak akkor van* **Euler-kör** *ha a gráf unilaterálisan összefüggő és minden csúcsra* $d_{ki}(u)=d_{be}(u)$ *vagyis a gráf* $u\in V$ *szimmetrikus*.

Tétel: *Egy* $G=(V,E)$ **irányított gráfban** *akkor és csak akkor van* **Euler-út** *ha a gráf unilaterálisan összefüggő és szimmetrikus kivéve két pontot ahol a ki- és be-fokok különbsége* ± 1.

Gráfelméleti alapfogalmak - 16

Def: *A G gráf* **Postás útjának vagy élsorozatának (Postman Tour)** *nevezünk egy élsorozatot, ha az élsorozat legalább egyszer tartalmazza G összes élét.*

Def: *A* **Kínai postás probléma (Chinese Postman Problem),** *amikor egy optimális (pl. minimális költségű) postás utat keresünk egy irányított, erősen összefüggő gráfban. Az ilyen utat* **Kínai postás útnak (Chinese Postman Tour)** *nevezzük.*

Tehát ha létezik Euler út, akkor az egyben Kínai postás út is.

Gráfelméleti alapfogalmak - 17

<u>Def:</u> *Adott P postás út* $G=(V,E)$*-ben. Legyen a* (v_i,v_j,L_k) *él előfordulásának száma P-ben* $\chi(v_i,v_j,L_k)\geq 1.$ *Ha a* (v_i,v_j,L_k) *élet megismételjük* $\chi(v_i,v_j,L_k)$ *-szor, akkor G gráf* **szimmetrikus kiterjesztett (Symmetric Augmentation)** *gráfját kapjuk, melyet jelöljünk* $\hat{G}$ *-vel.*

<u>Következmény:</u> P egy Euler útja $\hat{G}$-nek.

Gráfelméleti alapfogalmak - 18

Kínai postás probléma megoldása:

1. Ismételjünk meg néhány $(v_i, v_j, L_k) \in E$ élet úgy, hogy az eredmény gráf $\hat{G}$, szimmetrikus kiterjesztett gráf legyen és a létrehozott élek összköltsége minimális legyen. (Polinom időben megoldható!)
2. Keressük meg $\hat{G}$ egy Euler útját. (Lineáris algoritmus!)

Gráfelméleti alapfogalmak - 19

<u>Def:</u> *Legyen* E_c *a gráf éleinek részhalmaza:* $E_c \subseteq E$ **Vidéki kínai postás út (Rural Chinese Postman Tour)** *egy olyan út, ahol* E_c *minden élén legalább egyszer áthaladunk.*

Vidéki Kínai postás probléma (Rural Chinese Postman Problem) keresni egy minimális költségű vidéki postás utat.

<u>NP-teljes!!!!!!!!</u>

FSM – Finite State Machine

<u>Def:</u> *Egy* **determinisztikus véges automata** *a következő hatos:* $FSM = \langle S, I, O, \delta, \theta, st_0 \rangle$ *ahol:*

$S = \{st_1, \ldots, st_n\}$ *az állapotok véges halmaza*

$I = \{i_1, \ldots, i_n\}$ *a bemenetek véges halmaza*

$O = \{o_1, \ldots, o_n\}$ *a kimenetek véges halmaza*

δ *az állapot átmenet függvény, mely leképezi a jelenlegi állapotot és a bemenetet a következő állapotra:* $\delta : S \times I \to S$

θ *a kimeneti függvény, mely leképezi a jelenlegi állapotot és a bemenetet a kimenetre:* $\theta : S \times I \to O$

st_0 *a kezdeti állapot*

<u>Megjegyzés</u>: Különbség a nyelvi automatákhoz képest: a fent definiált FSM egy bemenetre egy kimenetet ad és nincs elfogadó állapot. (Így tudunk tesztelni!)

FSM – Finite State Machine

- Egy FSM reprezentálható irányított gráffal vagy állapot átmenet táblával:

$$G = (V, E)$$

$$S \to V$$

$$\delta \to E$$

- INRES példa

EFSM – Extended Finite State Machine

<u>Def:</u> *Egy* **determinisztikus kiterjesztett véges automata** *a következő tizes:* $EFSM = \langle S, I, O, \delta, \theta, P, VAR, A, st_0, va_0 \rangle$

ahol:

P *a predikátumok véges halmaza, azaz egy bemeneti eseményhez egy adott állapotban és változó kombinációban az igaz, vagy hamis értéket rendeli.*

VAR *a változók véges halmaza*

A *az akciók véges halmaza*

va_0 *a változók kezdeti értéke*

Az SDL EFSM-re épül!!!!!

FSM - EFSM

- A tesztgeneráló módszerek általában FSM automatákon alapulnak nem EFSM-en.
- Az EFSM modellek transzformálása FSM modellre igen gyakran állapotrobbanáshoz vezet!!!! (State-space explosion).
- Ez baj, mert a valós protokollok EFSM-mel adhatók meg: lsd. SDL.

FSM mint gráf - 1

<u>Def:</u> Az *r hosszú Q bemeneti sorozat* **meghatározott (specified)** *egy M FSM-re* st_i *állapotában, ha létezik egy* $w = (v_{i_1}, v_{i_2}, L_1), \ldots, (v_{i_r}, v_{i_{r+1}}, L_r)$ *élsorozat* $v_i = v_{i_1}$ *kiindulóponttal (* st_i v_i *-nek felel meg az FSM-ben és* L_j *megfelel* q_j *-nek a bemeneti sorozatban).*

<u>Def:</u> *Egy FSM* **teljesen meghatározott (Fully Specified),** *ha minden állapotában minden lehetséges bemenethez tartozik állapot átmenet (és kimenet).*

FSM mint gráf - 2

<u>Def:</u> st_i *állapot* **gyengén ekvivalens (weakly equivalent)** *az* $st_j \in S$ *állapottal, ha egy tetszőleges* st_i*-re meghatározott bemeneti sorozat* st_j *-re szintén meghatározott valamint teljesen megegyező kimeneti sorozatot ad.*

<u>Def:</u> *Két állapot* **erősen ekvivalens (strongly equivalent)** *ha kölcsönösen gyengén ekvivalensek egymással.*

FSM mint gráf - 3

Def: *Egy FSM-et* **minimálisnak (minimal)** *hívunk, ha nem tartalmaz erősen ekvivalens állapotokat.*

Def: v_0 **kezdőállapotot (start state)** *az FSM* **null állapotának (null state)** *nevezzük.*

Def: *Egy FSM-nél* **kiinduló állapotba állás képességéről (reset capability)** *beszélünk, ha képes null állapotba kerülni bármely állapotából egy egyszerű* r_i *bemenet hatására.*

Tesztgeneráló módszerek - 1

- **Fekete doboz (black box)** módszer: implementációt egy be- illetve kimeneti porton keresztül érjük csak el. Tehát bemeneti sorozatokat generálunk és figyeljük az arra érkező kimeneti sorozatot, de a belső állapotokról nincs információnk.
- **Holtpont (deadlock)** mentesség
- **Végtelen ciklus (livelock)** mentesség

Tesztgeneráló módszerek - 2

Ennek a vizsgálatnak alapvetően három lépése van:

1. Az automata st_i állapotba vitele
2. i_k bemeneti esemény előidézése és o_l kimenet ellenőrzése
3. A várt st_j állapot ellenőrzése

Def: Tesztelő él (Testing edge) $(v_i, v_j, i_k / o_l)$ *legyen a következő:* $TEST(v_i, v_j, i_k / o_l)$

Tesztgeneráló módszerek - 3

Gyakorlati problémák:

- Korlátozott vezérelhetőség: nehéz az implementációt elvinni egy adott állapotba, sokszor nagyon sok átmenet kell hozzá (1. lépés)
- Korlátozott megfigyelhetőség: nehéz ellenőrizni, hogy az implementáció a várt állapotba került-e. (3. lépés)

Tesztgeneráló módszerek - 4

<u>Def:</u> *Minden* $st_i \in S$ *állapotra létezik a következő él a gráf reprezentációban:*

$(v_i, v_i, status / i) \in E$ *melyet* **státus üzenetnek (Status message)** *hívunk.*

A státus üzenet bevezetése megoldja a megfigyelhetőség (3. lépés) problémáját!

Tesztgeneráló módszerek - 5

Def: *Bemenetek és kimenetek sorozatát* **meghatározó sorozatnak (Characterizing Sequence)** *hívjuk, ha az adott állapotra megkülönböztető sorozatot ad, vagyis megkülönbözteti az adott állapotot a többi állapottól.*

DS módszer - 1

<u>Def:</u> *Egy M FSM* **megkülönböztető sorozata (Distinguishing Sequence)** *különböző kimeneteteket ad minden állapotra.*

Egy megkülönböztető sorozat egyben meghatározó sorozat is!

DS módszer - 2

- Az FSM-nek minimálisnak és erősen összefüggőnek kell lennie.
- A módszer megadja a végállapotot is.
- Nincs minden protokollnak DS-e. Sőt a gyakorlatban nagyon ritka, hogy létezik DS.
- Felel a következő kérdésre: “Melyik állapotban volt az automata?”

W módszer - 1

Def: *A* **meghatározó halmaz (Characterizing set, W-set)** *meghatározó sorozatokat tartalamaz, melyek képesek páronként megkülönböztetni az automata egyes állapotait.*

W módszer - 2

- Az FSM-nek minimálisnak és erősen összefüggőnek kell lennie.
- Minden ilyen automata rendelkezik meghatározó halmazzal,
- tehát a gyakorlatban sokkal jobban alkalmazható mint DS módszer.

UIO sorozatok módszere - 1

Def: *Az* **egyedi ki/be sorozat (Unique Input/Output Sequence)** *M FSM egy adott állapotához megkülönböztető ki-/bemeneti viselkedést ad.*

Nem használható bármely állapotból kiindulva!

UIO sorozatok módszere - 2

- Az FSM-nek minimálisnak és erősen összefüggőnek kell lennie.
- Minden állapotra külön UIO sorozat van.
- Rövidebb mint a DS,
- viszont csak a következő kérdésre felel: “Ebben az adott állapotban volt az automata?”
- Nem minden állapot rendelkezik UIO sorozattal.
- Majdnem minden FSM-nek vannak UIO sorozatai, így a gyakorlatban sokkal jobban használható, mint a DS vagy a W módszer.

TT módszer - 1

<u>Def:</u> *Az automata összes állapotátmenetének bejárása egyidejűleg a kimenetek ellenőrzésével a* **TT (Transition Tour)** *módszer.*

Tulajdonképpen egy gráfbejárást jelent.

TT módszer - 2

- Az FSM-nek minimálisnak és erősen összefüggőnek kell lennie.
- A legrövidebb vizsgáló sorozatot generálja.
- Tulajdonképpen a kínai postás problémának a megoldása.
- Nem minden irodalom említi önálló módszerként.
- Kimeneti hibákat fölfedez, de nem képes detektálni az állapot átmenetekből adódó hibákat (errors of the next state).
- Status üzenet nélkül a gyakorlatban nem nagyon alkalmazható.

A tesztgeneráló módszerek összehasonlítása

- TT a legrövidebb sorozat, de nem képes a következő állapotra vonatkozóan minden hibát felismerni. Jó lenne, ha lenne status üzenet a protokollokban (tesztelhetőre tervezés – DSS1 LAPD)
- A DS és az UIO nagyságrendileg azonos hosszú sorozatokat generál. DS nem alkalmazható minden FSM-re, de előnye, hogy többszörös hibákat is képes felfedezni.
- Az UIO minden protokollra alkalmazható, de csak egyszeres hibákat detektál.
- A W módszer gazdaságtalanul hosszú sorozatokat generál.

UIO halmaz - 1

Def: *Az* **UIO halmaz (UIO-Set)** *az UIO sorozatok halmaza M FSM egy adott állapotára olymódon, hogy az állapothoz tartozó ki-/bemeneti viselkedés az automata semelyik más állpotára nem jellemző*.

UIO halmaz mindig létezik, ha az FSM minimális!

UIO halmaz - 2

- Az UIO halmaz általánosítása az UIO sorozatnak (hasonlóan a W halmazhoz, ami általánosítása a DS-nek).

(Mostantól az UIO-val foglakozunk csak!)

Teszt részsorozat - 1

<u>Def:</u> *A* **teszt részsorozat (Test Subsequence - TSS)** *egy bemeneti sorozat, amely egyben képes ellenőrizni egy E_i átmenethez tartozó kimenetet és az E_i-t követő állapotot.*

Teszt részsorozat - 2

TSS_i generálás három lépése:

1. Vigyük el az FSM-et $HEAD(E_i)$ állapotba
2. Idézzük elő az $INPUT_i$ bemenetet és ellenőrizzük, hogy a kimenet $OUTPUT_i$-e
3. Egy UIO sorozattal ellenőrizzük, hogy a végállapot $NEXTSTATE_i$-e

Teszt részsorozat - 3

<u>Def:</u> *Definiáljuk a teszt részsorozatot a következőképpen:* $L_{ij} \bullet CS(v_j)$, *ahol* L_{ij} *ki-/bemeneti cimkék sorozata,* $L_{l_1} \bullet L_{l_2} \bullet \ldots \bullet L_{l_\lambda}$ *és* $\bullet$ *a konkatenációs operátor, valamint* $CS(v_j)$ *egy meghatározó sorozat* v_j *állapotra nézve.*

$\alpha, \beta, \gamma -$ részsorozat

A részsorozatoknak több kategóriájuk van a hosszuktól függően, melyek különböző osztályú hibák felderítésére képesek:

$L_{ij} = 0,1,2,3,\ldots,$ megfelel a $\alpha-, \beta-, \gamma-, \delta-, \ldots,$ részsorozatnak

$\alpha -$részsorozat

$L_{ij} = \varepsilon$, ahol ε az üres sztring

- A protokoll állapotait teszteli
- Ki-/bemeneti sztringekből áll, melyeket az FSM állapotaira meghatározó szorozat alkalmazásával nyerhetünk
- UIO esetén UIO sorozatok az FSM állapotaira
- DS esetén ki-/bemeneti párok az automata minden állapotára

$\beta -$ részsorozat

$L_{ij} = L_{l_1} = (i/o)$, ahol i/o egy ki/bemeneti érték az FSM egy állapotátmenetéhez rendelve

- A protokoll egyedi állapotátmeneteit teszteli
- Egy állapotátmenet ki-/bemenetének és az állapotátmenet végén lévő állapot állapotellenőrző sorozatának a konkatenációja

$\gamma -$ részsorozat

$L_{ij} = L_{l_1} \bullet L_{l_2} = (i_1 / o_1) \bullet (i_2 / o_2)$, ahol i_1 és i_2 valamint o_1 és o_2 a két bemenet valamint kimenet az FSM két egymást követő állapotátmenetén

- Előállítása hasonló $\beta -$ részsorozathoz azzal a különbséggel, hogy az állapotellenőrző sorozatot a második állapotátmenet végéhez tartozó állapotra kell meghatározni

Teszt részsorozat előállítása

TSS_i előállítása E_i élre különböző módszerekkel:

- DS módszer esetén: $TSS_i = E_i \bullet DS$
- UIO módszer esetén: $TSS_i = E_i \bullet UIO_k$, ahol UIO_k egy UIO sorozat k állapotra, ahol $k = TAIL(E_i)$
- W módszer esetén: $TSS_i = E_i \bullet \{W\}$

Tesztsorozat

A teszt részsorozatokat összerakhatjuk tesztsorozattá (Test Sequence) úgy, hogy hosszúságra nézve minimális legyen az eredmény. Egy tesztsorozatnak a következő feltételeknek kell megfelelnie:

1. Tartalmaznia kell az összes TSS-t
2. A TSS-eket olymódon kell konkatenálni, hogy bármely TSS_i kezdődjön a saját HEAD állapotából

Tesztsorozat előállítás - 1

Reset Transition Method - Ha a protokollnak van egy reset (*ri*) állapotátmenete minden állapotból a kiinduló állapotba a tesztsorozat a következőképp adható meg:

- Készítsük el az összes TSS_i-t
- Mindegyiket terjesszük ki a következő módon: $ri \bullet TS_{init,i} \bullet TSS_i$, ahol $TS_{init,i}$ a legrövidebb sorozat *init* kiinduló állapotból, *i* állapotba
- Konkatenáljuk össze az összes ilyen kiterjesztett TSS_i teszt részsorozatot

Tesztsorozat előállítás - 2

Touring Method – Optimális hosszúságú tesztsorozat készíthető: ahelyett, hogy visszavisszük az automatát minden alkalommal a kiindulási állapotba egy következő TSS-t hajtunk végre.

Így rövidebb tesztsorozatot kapunk!

Tesztsorozat előállítás - 3

Két TSS a következőképpen fűzhető egybe:

$$TSS_i \bullet BS_{i',j'} \bullet TSS_j$$

$BS_{i',j'}$ (Bridge Sequence) egy sorozat, mely

$TAIL(TSS_i) = i'$ részsorozat végállapotából

$HEAD(TSS_j) = j'$ részssorozat kezdőállapotába vezet

Tesztsorozat előállítás - 4

Sok cikk foglalkozik a Touring módszerrel, mivel sokkal rövidebb tesztsorozatokat lehet vele előállítani mint más módszerekkel. Kétféle módon közelíthető meg a feladat:

- **Heurisztikával**: Egyszerű és gazdaságos megoldás, de nem ad optimális tesztsorozatot
- **Optimalizálással**: Vidéki kínai postás (RCP) probléma megoldásán alapul a feladat. Bizonyos megkötések mellett létezik polinom rendű algoritmus!

Optimalizálás RCP-vel - 1

Készítsük el G' **teszt részsorozat gráfot (Test Subsequence Graph)** G gráfból:

$$G' = (V', E')$$

Ha $V' \equiv V$ és $E' \equiv E \cup E_c$, ahol

$$E_c = \left\{ (v_i, v_j; L_l) \bullet UIO_j \mid (v_i, v_j; L_l) \in E \text{ és } TAIL(UIO_j) = v_k \right\}$$

Optimalizálás RCP-vel - 2

Egy minimális költségű tesztsorozatot keresni, amely leteszteli az FSM összes állapotátmenetét, egyenértékű azzal, hogy keresünk egy Vidéki kínai postás (RCP) utat G' gráf E_c élein:

Minimális költségű út, mely az FSM kezdeti állapotából indul, végighalad E_c-ben lévő éleken legalább egyszer és visszatér a kezdeti állapotba. (Természetesen E-ben lévő éleket is használhatunk a bejárás során.)

Optimalizálás RCP-vel - 3

RCP megoldása G' gráfra:

1. Készítsük el $\hat{G}^* = (\hat{V}^*, \hat{E}^*)$ gráfot G' gráfból, ahol $\hat{V}^* \equiv V'$, minden E-ben lévő élet $\hat{E}^*$ is tartalmazhat és E_c minden élét $\hat{E}^*$ is tartamazza legalább egyszer olymódon, hogy az élek összköltsége $\hat{E}^*$-ban minimális és $\hat{G}^*$ szimmetrikus gráf legyen. $\hat{G}^*$ **vidéki szimmetrikus kiterjesztett (Rural Symmetric augmenetation) gráfja** G'-nek.
2. Keressünk Euler utat $\hat{G}^*$-ben, amely $\hat{E}^*$ éleken pontosan egyszer megy át.

Optimalizálás RCP-vel - 4

Definiáljuk $G[E_c]$ gráfot, mint G' feszítő-részgráfját úgy, hogy pontjai megegyeznek a G'-beli pontokkal és élei csak E_c-beliek.

<u>Tétel:</u> *Ha* $G[E_c]$ *gyengén összefüggő feszítő gráfja* G'*-nek, akkor* G' *bármely vidéki szimmetrikus kiterjesztett gráfja* $\hat{G}^*$*, erősen összefüggő és ezért létezik benne Euler út.*

Optimalizálás RCP-vel - 5

<u>Tétel:</u> *Ha egy FSM rendelkezik a kiinduló állapotba állás képességével (reset capability), akkor a* $G[E_c]$ *feszítő részgráf teljes feszítő részgráfja* G'*-nek és gyengén összefüggő.*

Optimalizálás RCP-vel - 6

<u>Tétel:</u> *Ha egy M FSM gráf reprezentációja erősen összefüggő és minden* $v_i \in V$ *pontjára létezik egy* $(v_i, v_i; L_l) \in E$ *él (hurokél), akkor a* $G[E_c]$ *feszítő részgráf teljes feszítő részgráfja* G' *-nek és gyengén összefüggő.*

Optimalizálás RCP-vel - 7

Legyen $v_i \in G'$ élre

$$d_{in}^{E_c}(v_i) = \left\| \left\{ (v_j, v_i; L_k) \mid (v_j, v_i; L_k) \in E_c \right\} \right\|$$

és legyen

$$d_{out}^{E_c}(v_i) = \left\| \left\{ (v_i, v_j; L_k) \mid (v_i, v_j; L_k) \in E_c \right\} \right\|$$

ahol || a halmaz elemeinek a számát jelöli.

$\xi(v_i)$ v_i pontba befutó illetve onnan kifutó E_c

-beli élek számának a különbségét jelöli:

$$\xi(v_i) = d_{in}^{E_c}(v_i) - d_{out}^{E_c}(v_i)$$

Optimalizálás RCP-vel - 8

- Ha egy adott v_i pontra $\xi(v_i) = 0$, akkor nem szükséges az 1. lépésben plusz élet betenni az adott ponthoz,
- ellenkező esetben szimmetrikussá kell tegyük a gráfot élek megismétlésével
- Minimális költség, maximális folyam algoritmus segítségével vehetünk be további éleket minimális költséggel

Optimalizálás RCP-vel – 9

Def: *Hozzuk létre* $G_F = (V_F, E_F)$ *gráfot* G' *gráfból a következőképpen:*

Legyen $V_F \equiv V' \cup \{s,t\}$, *ahol s és t a forrás és a nyelő* G_F*-ben, valamint*

$$E_F = E \cup \{(s,v_i) \mid v_i \in D\} \cup \{(v_j,t) \mid v_j \in C\},$$

ahol $C \subset V_F$ *a pozitív indexű (*$\xi(v_i) > 0$*),* $D \subset V_F$ *a negatív indexű (*$\xi(v_i) < 0$*) pontok halmaza* G' *-ben.*

Optimalizálás RCP-vel – 10

Egyszerűsítések:

- A hurokéleket elhagyjuk
- Többszörös él esetén csak a minimális költségű élet vesszük be
- Mivel nincsenek többszörös élek, így a cimkéket elhagyjuk

Optimalizálás RCP-vel – 11

Legyen minden (s, v_i) él költsége nulla és a kapacitása: $\gamma(s, v_i) = \xi(v_i)$, és minden (v_j, t) él költsége szintén nulla kapacitása pedig: $\gamma(v_j, t) = -\xi(v_j)$

A többi él G_F-ben azonos költséggel szerepel mint G'-ben a kapacitásuk pedig végtelen.

Optimalizálás RCP-vel – 12

Adott egy F minimális költségű maximális folyam G_F-ben. Definiáljuk a χ függvényt E' élekre:

$$\chi(v_i, v_j; L_i) = \begin{cases} 1 & ha\,(v_i, v_j; L_i) \in E_c; \\ F(v_i, v_j; L_i), & ha\,(v_i, v_j; L_i) \in E. \end{cases}$$

Belátható, hogy χ függvény minimális költségű vidéki szimmetrikus kiterjesztett gráfját adja G_F-nek