Eseményalapú rendszertervezés helyességbizonyítással

Majzik István
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék
majzik@mit.bme.hu

2004

Szekvenciális programok konstrukciója

Cél:

- Szekvenciális programok automatikus konstrukciója.
- Konstrukciós szabályok + az előrevetítés (anticipation) technikája.

Szekvenciális programok:

- Állítások: műveletek, adatkezelés, számítások
- Vezérlési szerkezetek: sorrendezés, elágazások, iterációk.

Szekvenciális programok hagyományos fejlesztése

- Kezdeti algoritmus-váz + fokozatosan finomítás a futtatható kódig.
- A finomítás minden lépése ütemezett utasításokból áll.
- A saját absztrakciós szintjén minden lépés teljesnek mondható (bár sokszor még nem-determinisztikus).

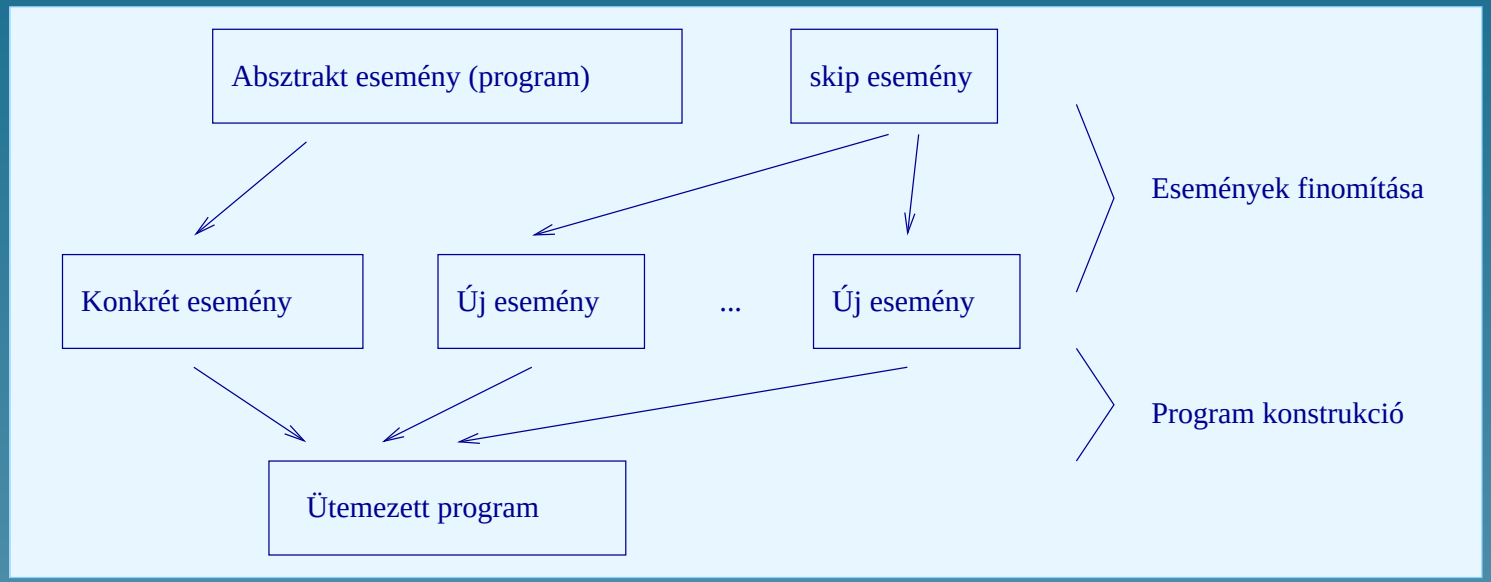
A most bemutatandó megközelítés

- A matematikai logikai alapokon való fejlesztés lényege:
  az utasítások kidolgozásának és a vezérlésnek a szétválasztása.

- Először: Az utasítások felvétele és finomítása.
  - ⋆ Állítások = események: feltételrész és akciórész.
  - ⋆ Párhuzamos, elosztott végrehajtás (nincs ütemezés):
    feltételrész határozza meg a végrehajthatóságot.
  - ⋆ Finomítás: feltételrész szigorúbbá tétele, új esemény felvétele.
  - ⋆ A program egyes részeit függetlenül lehet kidolgozni.
  - ⋆ Bizonyítható a finomítás konzisztenciája.

- Ha ez kész: Ütemezés, vagyis a vezérlési szerkezetek konstrukciója
  - ⋆ Szisztematikusan, konstrukciós szabályok alapján.
  - ⋆ Az egyes események összeillesztése a feltételrészek alapján
    (így a feltételrészek egy része gyakorlatilag el is tűnik).

Program konstrukció eseménybizonyítással



Fejlesztési folyamat

- A feltételrészek szigorúbbá tétele + új események felvétele.
- A finomítás konzisztenciáját szolgáló szabályok betartása.
- Élő és terminálló programok létrehozásához:
  - ⋆ A feltételrészek diszjunkciója igaz (az adott invariánsok mellett).
  - ⋆ Az újonnan bevezetett események nem gátolják meg állandó jelleggel más események végrehajtását.
  - ⋆ Az akciórészek egy jól megalapozott halmaz elemét csökkentik (olyan, részben rendezett halmaz, amelyben nem létezik végtelen, csökkenő elemekből álló szekvencia; lásd pl. a természetes számok).

A finomítás konstrukciós szabálya

- skip eseményt finomító új U esemény bevezetése.
- Egy természetes szám csökkentése:
  U nem gátol folyamatosan más eseményeket.

$$
\begin{array}{rcl}
S & \rightsquigarrow & S' \sqcup U, \text{ ahol} \\
S & \sqsubseteq & S' \\
skip & \sqsubseteq & U \\
I & \Rightarrow & \mathrm{guard}(S') \vee \mathrm{guard}(U) \\
I & \Rightarrow & V \in N \\
I & \Rightarrow & (V = n \Rightarrow [U](V < n))
\end{array}
$$

- Itt $\sqcup$ jelöli a választást (események implicit ütemezése),
  $S \sqsubseteq S'$ jelöli azt, hogy S' finomítása S-nek.

Új esemény bevezetése

- Finomítás: új U esemény (skip finomítása) bevezetése.
- Természetes szám változó csökkentése
  (vagy más, jól megalapozott halmazból választott változó).
- Ez a változó a finomítás során újonnan bevezetett is lehet.

Tegyük fel: a finomítás $i+1$-edik lépésében vezetjük be

- az Eventx eseményt,
- az S_y halmazon értelmezett y változót
  (az ehhez tartozó, állapotra vonatkozó finomítási invariánssal együtt),
- y-t az Eventx esetén az említett csökkentés előírására fogjuk felhasználni,
  mégpedig a $V(y)$ kifejezésben.

Az előrelátás technikája

- Már az *előző finomítási lépésben* (i-edik lépésben) bevezetjük az y változót és az Eventx eseményt a következő formában:

$$\text{Eventx} \mathrel{\widehat{=}} \text{BEGIN } y : (y \in S_y \wedge V(y) < V(y_0)) \text{ END}$$

  Ez teljesíti a finomítási szabályokat (skip finomítása).

- Az $i+1$-edik lépésben, "normál" finomításként kerül sorra Eventx konkrét formájának megfogalmazása.

- Ekkor bizonyítjuk be a finomítás konzisztenciáját.

- Egyszerűsödnek a bizonyítási lépések.

A konstrukciós szabályok

- Egy vagy több eseményből a program vezérlési szerkezetét állítják össze.
- Itt a strukturált programokban megszokott szerkezeteket használjuk:
  - ⋆ IF Q THEN S ELSE T END
  - ⋆ WHILE Q DO S END
- Jelölés: $P \Longrightarrow S$ rövidíti a SELECT P THEN S END eseményt.

A feltételes elágazás szabálya

Két eseményből egy IF ... THEN ... ELSE ... END elágazást állít össze:

$$\begin{aligned}(P \wedge Q &\implies S) \sqcup (P \wedge \neg Q \implies T) \sqcup U \rightsquigarrow \\ (P &\implies \text{IF } Q \text{ THEN } S \text{ ELSE } T \text{ END}) \sqcup U\end{aligned}$$

Az iteráció szabálya

Eseményeket alakít át WHILE ... DO ... END iterációvá:

$$\begin{aligned} (P \wedge Q &\implies S) \sqcup (P \wedge \neg Q \implies T) \sqcup U \rightsquigarrow \\ (P &\implies \text{WHILE } Q \text{ DO } S \text{ END}; T) \sqcup U \\ \text{ahol } I \wedge P \wedge Q &\Rightarrow [S]P \text{ valamint } S \text{ és } T \text{ esetén nincs feltételrész} \end{aligned}$$

Az utolsó feltétel: Az iteráció során P invariáns.

Egyszerűsített forma:

$$\begin{aligned} (Q &\implies S) \sqcup (\neg Q \implies \text{skip}) \sqcup U \rightsquigarrow \\ &(\text{WHILE } Q \text{ DO } S \text{ END}) \sqcup U \\ &\text{ahol } S \text{ esetén nincs feltételrész.} \end{aligned}$$

Mintapélda: Adatkeresés

Egy indexekkel ellátott tömbben keressük egy megadott tömbelem indexét.

Változók és konstansok:

- S a tömbelemek halmaza,
- n (tömbelemek száma), f (a tömb) és x (a megadott tömbelem) három konstans,
- i (a keresett index) egy változó:

$$\begin{aligned} n &\in N_1 \\ f &\in 1..n \to S \\ x &\in \mathrm{ran}(f) \\ i &\in 1..n \end{aligned}$$

Kiindulás

- A kezdeti esemény:

$$\text{aprog} \mathrel{\widehat{=}} \text{BEGIN } i : (i \in 1..n \wedge f(i) = x) \text{ END}$$

- Az előre látott esemény:

$$\text{progress} \mathrel{\widehat{=}} \text{BEGIN } \ i : (i \in 1..n \wedge n - i < n - i_0) \text{ END}$$

- Nem használunk külön változót a finomított eseményben, magát az i változót növeljük majd (így $n - i$ értékét csökkentve).

- Megközelítés: Az indexeket növelve haladunk a tömbben, ha az $i - 1$ indexig nem találtuk meg a keresett elemet, akkor lépünk tovább az i-edik elemre.

A finomítás eredménye

- A finomított események:

$$\text{init} \mathrel{\widehat{=}} \text{BEGIN } i := 1 \text{ END}$$

$$\text{aprog} \mathrel{\widehat{=}} \text{SELECT } f(i) = x \text{ THEN skip END}$$
$$\text{progress} \mathrel{\widehat{=}} \text{SELECT } \ f(i) \neq x \text{ THEN } i := i + 1 \text{ END}$$

- A finomítás konzisztenciája bizonyítható.

- Az aprog esemény nem "keres" többé,
  hanem tulajdonképpen a megállást írja le.

A program konstrukciója

- Az iteráció szabálya: az aprog és progress eseményekből egy iteráció áll elő.
- Ehhez fűzhető az inicializálást végző esemény.

$$\mathrm{cprog} \mathrel{\widehat{=}} \mathrm{BEGIN}\ i := 1; \mathrm{WHILE}\ f(i) \neq x\ \mathrm{DO}\ \ i := i + 1\ \mathrm{END}\ \ \mathrm{END}$$

Mintapélda: Maximumkeresés

Egy természetes számokat tartalmazó tömbben keressük meg a maximális értéket!

Változók és konstansok:

- n (tömbelemek száma) és f (maga a tömb) egy-egy konstans,
- m (a maximum érték) egy változó,
- míg k (a maximum érték indexe) egy előrevetített változó:

$$\begin{aligned} n &\in N_1 \\ f &\in 1..n \to N \\ m &\in \mathrm{ran}(f) \\ k &\in 1..n \end{aligned}$$

Kiindulás

- A kezdeti esemény:

$$\text{aprog} \mathrel{\widehat{=}} \text{BEGIN } m : (m \in \text{ran}(f) \wedge \forall x : (x \in 1..n \Rightarrow f(x) \leq m)) \text{ END}$$

- Két előrevetített esemény:

$$\text{test1} \mathrel{\widehat{=}} \text{BEGIN } m, k : (m \in \text{ran}(f) \wedge k \in 1..n \wedge n - k < n - k_0) \text{ END}$$

$$\text{test2} \mathrel{\widehat{=}} \text{BEGIN } m, k : (m \in \text{ran}(f) \wedge k \in 1..n \wedge n - k < n - k_0) \text{ END}$$

  Mindkét esemény csökkenti $n - k$ értékét;
  m változó pedig szabadon választható az értelmezési tartományából.

- Kimerítő keresés: A következő invariánst írjuk fel:

$$\forall x : (x \in 1..k \Rightarrow f(x) \leq m)$$

A finomítás

A finomítás során a következő eseményeket írjuk fel:

$$\text{init} \mathrel{\widehat{=}} \text{BEGIN } k := 1 \parallel m := f(1) \text{ END}$$

$$\text{aprog} \mathrel{\widehat{=}} \text{SELECT } k = n \text{ THEN skip END}$$

$$\text{test1} \mathrel{\widehat{=}} \text{SELECT } k \neq n \wedge f(k+1) \leq m \text{ THEN } k := k+1 \text{ END}$$

$$\text{test2} \mathrel{\widehat{=}} \text{SELECT } k \neq n \wedge f(k+1) > m \text{ THEN } k := k+1 \parallel m := f(k+1) \text{ END}$$

A finomítás konzisztenciája itt is bizonyítható.

A program vezérlési struktúra

- A test1 és test2 eseményekből a feltételes elágazás szabálya,
- az aprog eseménnyel az iterációs szabály,
- majd az init esemény:

cprog $\widehat{=}$ BEGIN $k, m := 1, f(1)$

WHILE $k \neq n$ DO

IF $f(k+1) \leq m$ THEN $k := k+1$ ELSE $k, m := k+1, f(k+1)$ END

END

END