



Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprog-
ramok Tanszék

Osztott rendszerek szintézise

IPM-08sztORSZE

Konzultációs segédanyag

Kopácsi László, Szabó Miklós

Utolsó módosítás: 2018. március 1.

Tartalomjegyzék

1. konzultáció	2
1.1. Áttekintés	2
1.2. Étkező filozófusok	2
1.3. Moziterem	3
2. konzultáció	4
2.1. Áttekintés	4
2.2. Program	7
2.3. Leggyengébb előfeltétel	8
2.4. Feladatok	10
3. konzultáció	11
3.1. Gyakorlás	11
3.2. Stabilitási tulajdonság	12
4. Megoldások	16

1. konzultáció

1.1. Áttekintés

Az előadás során több, temporális logikai relációval találkoztunk, nézzük ezeket át informálisan, kezdve a biztonsági tulajdonságokkal.

- Az első, melyet "háromszög"-ként említünk (\triangleright), bizonyos állapot-átmeneteket megenged, másokat pedig megtilt. A $(P \triangleright Q)$ azt jelenti, hogy a P állapotot **ha** elhagyjuk, akkor ezt csak a Q -n keresztül tehetjük meg. A háromszög azonban nem tesz semmiféle kikötést arról, hogy a P -t el kell hagynunk, csupán biztosít minket arról, hogy ha ez mégis megtörténik, akkor milyen irányba (nem) mozdulhatunk.
- Másik biztonsági tulajdonság az *invariáns*. Ha egy K állítás invariáns, akkor ennek minden állapot-átmenet előtt és után teljesülnie kell.
- A harmadik említett kikötés a *fixpont*. Ezzel leírhatjuk, hogy ha egy rendszerben már nem figyelhetünk meg további állapot-átmeneteket, akkor milyen tulajdonságoknak kell teljesülnie. $(FP \Rightarrow R)$ estén például egy R -el jelölt állítás igaz, amennyiben fixpontba jutottunk. Fixpontba azonban nem csak a kívánt befejezési állapot tarthat, ha holtpont helyzet alakul ki, azt is tekinthetjük fixpontnak.

Természetesen nem csak biztonsági tulajdonságokra van szükségünk - azaz mit (ne) csinálhasson a rendszer -, hanem haladásra is (azért csináljon valamit).

- Az "egyenes nyíl" (\mapsto) néven nevezett reláció egy szigorú kikötés arra vonatkozóan, hogy egy állapotból milyen másik helyzetbe **kell** lépünk. Míg \triangleright esetén csupán azt mondtuk, hogy *ha* elhagyunk egy állapotot, akkor azt milyen irányba tegyük, a $(P \mapsto Q)$ azt mondja, hogy a P állapotból a Q állapotba kell, hogy kerüljünk (véges időn belül).
- Ennél megengedőbb a "görbe nyíl"-ként (\hookrightarrow) ismert reláció. Ebben az esetben a $(A \hookrightarrow B)$ feltétel csupán annyit mond, hogy az A állapotot előbb-utóbb a B állapot fogja követni (azaz A -ból elkerülhetetlenül B -be fogunk érkezni), de itt nincs semmilyen megkötés arra, hogy a két állapot egymás után következzen be. Legális állapot-átmenet sorozat az $(A \hookrightarrow B)$ -ra az $\langle A, G, F, D, F, E, C, D, B \rangle$ is.

1.2. Étkező filozófusok

Tekintsük az előadáson is ismertetett *étkező filozófusok* feladatot (jegyzet^[1] 1.1). Próbáljuk meg kiegészíteni a feltételeket további megkötések formalizálásával:

- Ha a rendszer nyugalmi állapotban van, akkor egy filozófus sem eszik.
- Mindegyik filozófusra igaz, hogy ha hazament, akkor utána már nem kerülhet más állapotba.

1.3. Moziterem

A következő példában egy mozira vonatkozó feladatot fogunk ismertetni, ahol a nézők tevékenységére szeretnénk megkötéseket tenni. A jelölést megkönnyítendő vezessük be az alábbiakat: $n(i)$ jelölje az i -ik nézőt. A moziba látogatók állapotait az alábbiak alapján jelöljük:

- a) megérkezik a moziba - a
- b) jegyet vesz - j
- c) üdítőt és nasit vásárol - b
- d) érvényes jeggyel rendelkezik - t
- e) filmet néz - f
- f) hazamegy - h

Próbáljuk formalizálni az alábbi feltételeket:

- A moziba érkező néző filmet fog nézni.
- Ha valaki érvényes jeggyel rendelkezik, akkor megnézi a filmet.
- Ha a moziban nincs mozgás, akkor minden néző már otthon van.
- A moziba érkező néző jegyet vásárol, vagy a büfébe megy.
- A film után a néző hazamegy.
- Senki nem nézhet filmet úgy, hogy nincs érvényes jegye. (Tipp: próbáljunk invariánst megfogalmazni.)
- Ha valaki hazament, akkor már nem csinál semmit a moziban.

2. konzultáció

2.1. Áttekintés

Ahhoz, hogy a későbbiekben biztos módon számolhassunk programokkal, elkerülhetetlen a számunkra szükséges (alap)fogalmakat tisztázni a halmazelmélet és a relációk témakörében.

Legyenek A és B tetszőleges halmazok. A és B *direkt*-, vagy *Descartes*-szorzatán azt a halmazt értjük, melyben olyan párok találhatóak, melynek első eleme A -, második eleme pedig B -beli.

$$A \times B ::= \{(a, b) | a \in A \text{ és } b \in B\}$$

Jelölje $r \subseteq A \times B$ azt a bináris relációt, mely A elemeihez rendel értékeket a B halmazból (A és B tetszőleges halmazok). A reláció elemeit $(a, b) \in r$ módon fogjuk jelölni.

$$\text{Az } r \text{ reláció értelmezési tartománya: } \mathcal{D}_r ::= \{a \in A | \exists b \in B : (a, b) \in r\} \subseteq A$$

$$\text{Az } r \text{ reláció értékkészlete: } \mathcal{R}_r ::= \{b \in B | \exists a \in A : (a, b) \in r\} \subseteq B$$

$$r(a) \text{ jelölje azt a halmazt, melynek elemei: } \{b \in B | (a, b) \in r\}$$

Világos, hogy az értelmezési tartományban olyan elemek vannak, amikhez rendel valamit r , míg az értékkészletben olyanokat találhatunk, amik valamilyen elemhez hozzá lettek rendelve. Egy elem képe a hozzá rendelt elemek halmazából áll elő.

Egy g relációt *parciális függvénynek* (vagy determinisztikus relációnak) nevezhetünk, amennyiben az alábbi teljesül:

$$\forall a \in A : |g(a)| \leq 1,$$

azaz minden elemhez *legfeljebb* egy másikat társítunk. Jelölésünk ekkor: $g \in A \rightarrow B$. Ha minden elemhez pontosan egy értéket rendelünk, akkor az f reláció függvény, azaz:

$$\forall a \in A : |f(a)| = 1.$$

Jelölésünk ekkor: $f : A \rightarrow B$. Ebben az esetben általában $f(a)$ nem az egy elemű halmazt, hanem annak képét jelenti.

Ahhoz, hogy állításokat fogalmazhassunk meg a későbbiekben, szükségünk lesz logikai relációkra is.

$$\text{A } h \subseteq A \times \mathbb{L} \text{ logikai relációnak nevezzük, ahol } \mathbb{L} ::= \{igaz, hamis\}.$$

Ha h függvény, akkor *logikai függvénynek* nevezzük.

Egy reláció inverzét az alábbi módon definiálhatjuk:

$$R^{(-1)} ::= \{(b, a) \in B \times A \mid (a, b) \in R\}$$

A továbbiakban szükségünk lesz egy reláció adott halmazra vonatkozó inverz- és ősképfé definíciójára.

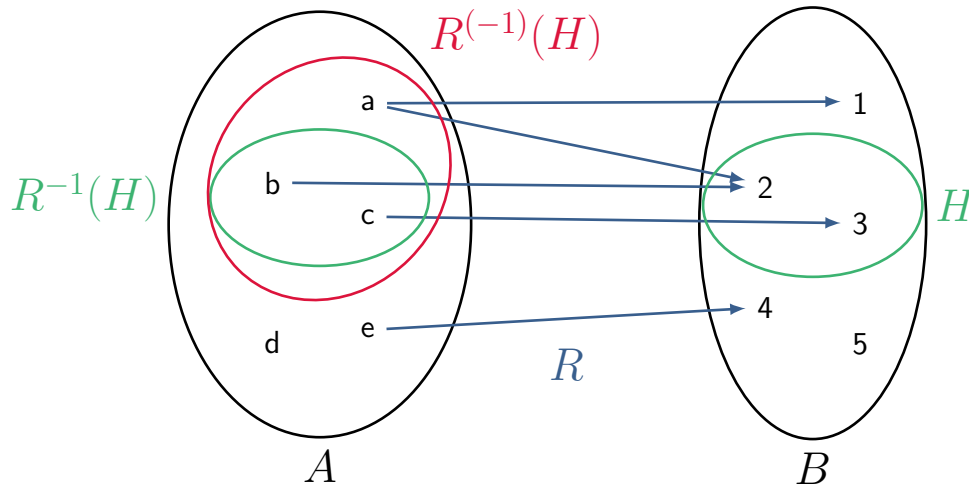
A $H \subseteq B$ halmaz R reláció szerinti *inverzképe*:

$$R^{(-1)}(H) ::= \{a \in A \mid R(a) \cap H \neq \emptyset\}$$

A $H \subseteq B$ halmaz R reláció szerinti *ősképe*:

$$R^{-1}(H) ::= \{a \in A \mid R(a) \subseteq H\}$$

Meggondolva látható, hogy az *inverzkép* megengedőbb, hisz csak annyit kér, hogy egy adott elemhez *létezzen* H -beli elem az R hozzárendelésben, az *ősképe* viszont megköveteli, hogy *minden* ilyen elem a H halmazban legyen.



1. ábra. Nézzünk egy példát a korábban említett fogalmakra. Legyen $A = \{a, b, c, d, e\}$ és $B = \{1, 2, 3, 4, 5\}$ halmazok, valamint $R \subseteq A \times B$, $R = \{(a, 1), (a, 2), (b, 2), (c, 3), (e, 4)\}$ reláció. Ekkor R értelmezési tartománya $\mathcal{D}_R = \{a, b, c, e\}$, értékkészlete pedig $\mathcal{R}_R = \{1, 2, 3, 4\}$. Továbbá legyen $H \subseteq B$, $H = \{2, 3\}$. Ekkor a H halmaz R reláció szerinti ősképe $R^{-1}(H) = \{b, c\}$, inverzképe pedig $R^{(-1)}(H) = \{a, b, c\}$. Itt jól látszik, hogy az *inverzkép* megengedőbb, mint az *ősképe*, hiszen $R^{-1}(H) \subseteq R^{(-1)}(H) = \{b, c\} \subseteq \{a, b, c\}$.

Legyen $R \subseteq A \times \mathbb{L}$ logikai reláció, R igazsághalmaza ekkor:

$$\lceil R \rceil ::= R^{-1}(\{\text{igaz}\}) \text{ azaz: } \lceil R \rceil = \{a \in \mathcal{D}_R \mid R(a) \subseteq \{\text{igaz}\}\}$$

Az igazsághalmazt tehát az $\{\text{igaz}\}$ halmazra vett ősképe szerint definiáljuk.

Ha inverzképet számolunk, akkor juthatunk a *gyenge igazsághalmaz* fogalmához:

$$\lfloor R \rfloor ::= R^{(-1)}(\{igaz\}) \text{ azaz: } \lfloor R \rfloor = \{a \in \mathcal{D}_R \mid R(a) \cap \{igaz\} \neq \emptyset\}$$

A későbbiekben nagyban megkönnyíti a dolgunkat, ha bevezetjük az *azonosan igaz*, és az *azonosan hamis* logikai függvényeket.

$$Igaz : A \rightarrow \mathbb{L} : \forall a \in A : Igaz(a) = \{igaz\}$$

$$Hamis : A \rightarrow \mathbb{L} : \forall a \in A : Hamis(a) = \{hamis\}$$

Könnyű meggondolni, hogy ekkor $\lceil Igaz \rceil = A$ és $\lceil Hamis \rceil = \emptyset$.

Az igazsághalmazzal kapcsolatban fontos megemlíteni néhány tulajdonságot, melyeket a későbbiekben kihasználunk.

Legyenek $P, Q \subseteq A \times \mathbb{L}$, ekkor:

- $\lfloor P \wedge Q \rfloor = \lfloor P \rfloor \cap \lfloor Q \rfloor$
- $\lfloor P \vee Q \rfloor = \lfloor P \rfloor \cup \lfloor Q \rfloor$
- $\lfloor \neg P \rfloor = A \setminus \lfloor P \rfloor$
- $\lfloor P \rightarrow Q \rfloor = \lfloor \neg P \vee Q \rfloor = (A \setminus \lfloor P \rfloor) \cup \lfloor Q \rfloor$
- $P \Rightarrow Q = \lfloor P \rfloor \subseteq \lfloor Q \rfloor$

Egyszerűbben megfogalmazhatóak állítások, ha tudjuk, hogy $A \Rightarrow B$. Ekkor ugyanis:

- $A \vee B = B$
- $A \wedge B = A$

Nézzünk erre egy példát, legyenek $A, B : \mathbb{N} \times \mathbb{L}$ úgy, hogy:

$\lceil A \rceil := \{10\text{-nél nagyobb szám}\}$ és

$\lceil B \rceil := \{\text{pozitív szám}\}$.

Világos, hogy $A \Rightarrow B$, hiszen ha egy egész szám 10-nél nagyobb, akkor pozitív. Az $A \vee B$ állítást úgy fogalmazhatjuk meg, hogy azokat az egész számokat keressük, melyek 10-nél nagyobbak, **vagy** pozitívak. Érződik, hogy a *vagy* kapcsolat miatt a gyengébb feltétellel is megelégszünk, így a bővebb halmaz, azaz a pozitív számok halmazát kapjuk ($= B$). Ha azonban a 10-nél nagyobb **és** pozitív számokra vagyunk kíváncsiak, akkor a szigorítás miatt a szűkebb halmazt kapjuk, tehát a 10-nél nagyobb számokat kell vizsgálnunk ($= A$).

2.2. Program

Röviden tekintsük át, hogy a Fóthi^[2]-Horváth - féle modellben hogyan is definiáltuk a programot és annak hatásrelációját.

Jelölje A^* az A elemeiből képzett véges, A^∞ pedig a végtelen sorozatokat. A későbbiekben A^{**} jelenti az $A^* \cup A^\infty$ halmazt, azaz a véges és végtelen sorozatok halmazát. Ha alaphalmaznak a természetes számokat választjuk, akkor az $\langle 1, 5, 3, 2 \rangle \in A^*$ egy véges, míg az $\langle 1, 2, 3, 4, \dots \rangle \in A^\infty$ végtelen sorozatot jelöl.

Utasítás alatt egy olyan $s \subseteq A \times A^{**}$ relációt értünk, melyre:

1. $\mathcal{D}_s = A$
2. $\forall a \in A : \forall \alpha \in s(a) : |\alpha| \neq 0 \wedge \alpha_1 = a$
3. $(\alpha \in \mathcal{R}_s \wedge \alpha \in A^*) \Rightarrow (\forall i(1 \leq i < |\alpha|) : \alpha_i \neq \alpha_{i+1})$
4. $(\alpha \in \mathcal{R}_s \wedge \alpha \in A^\infty) \Rightarrow (\forall i \in \mathbb{N} : (\alpha_i = \alpha_{i+1} \rightarrow (\forall k \in \mathbb{N}^+ : \alpha_i = \alpha_{i+k})))$

A fenti definíció a *működés* fogalmát próbálja absztrakt módon szemléltetni. A négy pont jól jellemzi az utasítást: elsőként szeretnénk, ha az utasítás minden állapotér-beli pontban értelmezve lenne (azaz az utasítás mindenhol el tud indulni).

Második pontban azt fogalmazzuk meg, hogy egy sorozat a működése teljes történetét írja le, kezdve a kiindulási állapottal.

A harmadik pontunk a *redukáltságra* vonatkozik: ha véges hosszú sorozattal dolgozunk, akkor egymás után kétszer ne szerepelhessen ugyanaz az elem (hiszen az nem egy jó véges utasítás, amelyik úgy lép egy következő állapotba, hogy nem történt állapotváltozás - ez a megfogalmazáson is érződik).

A negyedik pont a végtelen utasításra utal: ha egy utasítás futása nem fejeződik be (végtelen ciklus, stb.), akkor ezt a hozzárendelt sorozatban úgy jelzi, hogy egy adott ponttól kezdve nem történik állapotváltozás, folyton ugyan abban az állapotban ragad (pl. $s(4) = \langle 4, 3, 2, 1, 0, 0, 0, 0, 0, 0, \dots \rangle$)

Egy $s \subseteq A \times A^{**}$ utasítás *hatásrelációja* az a $p(s) \subseteq A \times A$ reláció, melyre:

1. $\mathcal{D}_{p(s)} = \{a \in A \mid s(a) \subseteq A^*\}$
2. $p(s)(a) = \{b \in A \mid \exists \alpha \in s(a) : \tau(\alpha) = b\}$

ahol $\tau : A^* \rightarrow A; \tau(\alpha) ::= \alpha_{|\alpha|}$, azaz a *tau* függvény egy véges sorozathoz annak utolsó tagját rendeli.

A hatásrelációt tehát csak olyan pontokban definiáljuk, ahol a program *megáll*, azaz véges sorozatot rendel, a hozzárendelési szabály pedig az, ahová az utasítás eljut, tehát az adott sorozatok utolsó eleme.

Az *absztrakt programot* egy $S = (s_0, \{s_1, s_2, \dots, s_n\})$ párként tudjuk megadni, ami egy párhuzamos programot takar. Az első tagja (s_0) a kezdeti utasítás, ami a program indulásakor hajtódik végre. A második tagja ($\{s_1, s_2, \dots, s_n\}$) pedig *atomi* (vagyis párhuzamos futás során nem akadnak össze, szekvenciális futás eredményét adó) utasítások halmaza. Ezeket valamilyen *feltétlenül pártatlan ütemezés* szerint (kiéheztetés nélkül, azaz az egyes végrehajtások során minden utasítást végtelen sokszor véve) végtelen sokáig futtatunk. A programra pedig akkor mondjuk, hogy terminált (befejeződött), ha fixpontba jut, azaz már nem történik állapotváltozás.

Nézzünk erre egy egyszerű példát:

Legyen $S = (x := 0, \{x := 2 \cdot x, x := x + 1\})$. Ekkor az alábbi sorozatot ennek egy lehetséges kiértékelésének tekintjük:

$$\begin{array}{ccccccccc} x := 0, & x := x + 1, & x := 2 * x, & x := 2 * x, & x := x + 1, & x := x + 1, & \dots \\ 0, & 1, & 2, & 4, & 5, & 6, & \dots \end{array}$$

2.3. Leggyengébb előfeltétel

Érezhető, hogy a fenti definíciókkal történő számolások nagyon nehezzé fogják tenni a későbbiekben a feladatok és az azt megoldó programok közötti kapcsolat megteremtését, ezért bevezetünk egy új fogalmat, a *leggyengébb előfeltételt*.

Legyen $s \subseteq A \times A^{**}$ utasítás, $R : A \rightarrow \mathbb{L}$ állítás. Az s utasítás R utófeltételhez tartozó *leggyengébb előfeltétele* az az $lf(s, R) : A \rightarrow \mathbb{L}$ **függvény**, melyre:

$$\lceil lf(s, R) \rceil = \{a \in \mathcal{D}_{p(s)} \mid p(s)(a) \subseteq \lceil R \rceil\}.$$

Az lf tehát egy olyan függvény, mely pontosan azokhoz a pontokhoz rendel igazat, melyből elindítva az s utasítást az biztosan megáll, és az összes ilyen állapotban az R tulajdonság teljesül. Magának a függvénynek a definícióját legtöbbször nehéz megadni, de az igazsághalmazát könnyedén kifejezhetjük. Az igazsághalmaz definícióját, a kompozíció és az öskép tulajdonságait felhasználva kapjuk, hogy:

$$\lceil lf(s, R) \rceil = \lceil R \circ p(s) \rceil.$$

Az lf -et tehát az utófeltételbe helyettesítés módszerével tudjuk kifejezni (ennek bizonyítása megtalálható a *tankönyv*^[2] 44-ik oldalán, a 3.1-es definíciónál).

Mivel a leggyengébb előfeltételt nem csak utasításokra, hanem programokra is ki szeretnénk tudni számolni, a fenti definíciót picit általánosítani kell. Tehát legyen $S = (s_0, \{s_1, s_2, \dots, s_n\})$ program, $R : A \rightarrow \mathbb{L}$ állítás. Ekkor az S program R utófeltételhez tartozó *leggyengébb előfeltétele* az egyes utasításai által adott leggyengébb előfeltételek

konjugáltja lesz:

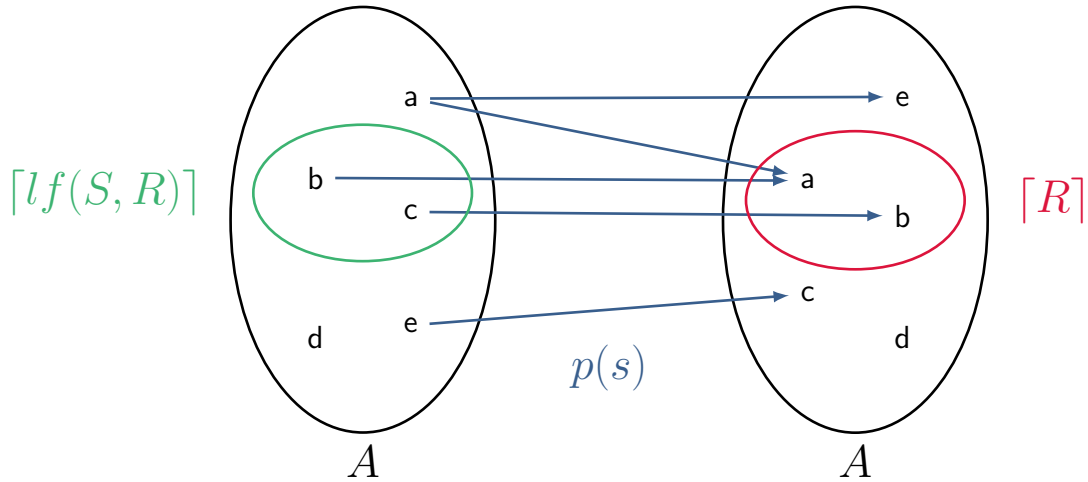
$$lf(S, R) = \bigwedge_{i=1}^n lf(s_i, R).$$

A későbbiekben több alaptulajdonságra is szükségünk lesz, ha az lf -fel akarunk számolni. Nézzük ezeket:

Legyen $S = (s_0, \{s_1, s_2, \dots, s_n\})$ program, $Q, R : A \rightarrow \mathbb{L}$ állítások. Ekkor:

1. $lf(S, Hamis) = Hamis$ (csoda kizárásának elve),
2. Ha $Q \Rightarrow R$, akkor $lf(S, Q) \Rightarrow lf(S, R)$ (monotonitás),
3. $lf(S, Q) \vee lf(S, R) \Rightarrow lf(S, Q \vee R)$ (gyenge additivitás),
4. $lf(S, Q) \wedge lf(S, R) = lf(S, Q \wedge R)$ (multiplikativitás)

A bizonyításokat szintén a jegyzetben lehet olvasni.



2. ábra. Legyen $A = \{a, b, c, d, e\}$ halmaz és $p(s) := \{(a, a), (a, e), (b, a), (c, b), (e, c)\}$ hatásreláció, $[R] = \{a, b\}$. $lf(s, R)$ ekkor azokban a pontokban lesz igaz, melyben $p(s)$ értelmezve van és minden ezekhez rendelt elem R igazsághalmazában található. Végiggondolva tehát $[lf(S, R)]$ ekkor $= \{b, c\}$.

Kiszámítása: Az lf kiszámolását a már fentebb említett utófeltételbe helyettesítés módszerével tudjuk megtenni.

Az **egyszerű értékadások** során ez a következőt jelenti:

$$lf(x := y, R) = R^{x \leftarrow y}$$

Például az $s = (x := 3), R = (1 \leq x \leq 5)$ esetben:

$lf(s, R) = lf(x := 3, (1 \leq x \leq 5)) = R^{x \leftarrow y} = (1 \leq x \leq 5)^{x \leftarrow 3} = (1 \leq 3 \leq 5)$. Az s utasítás tehát olyan állapotokban tudjuk biztonságosan végrehajtani úgy, hogy R -be

érkezzen, melyre teljesül az, hogy $1 \leq 3 \leq 5 \equiv igaz$, azaz tetszőleges pontból indítva helyesen működő utasítást kaphatunk.

Feltételes értékadás esetén figyelembe kell venni a feltételt is, hiszen ha ez nem teljesül, abban az esetben nem kell az értékadást végrehajtanunk.

$$lf(\{x := y, \text{ha } \pi\}, R) = (\pi \rightarrow R^{x \leftarrow y}) \wedge (\neg \pi \rightarrow R^{SKIP})$$

Például az $s = (x := -x, \text{ha } x < 0), R = (x > 0)$ esetben:

$$\begin{aligned} lf(s, R) &= lf((x := -x, \text{ha } x < 0), x > 0) = (x < 0 \rightarrow (x > 0)^{x \leftarrow -x}) \wedge (x \geq 0 \rightarrow x > 0) \\ &= (x < 0 \rightarrow -x > 0) \wedge (x \geq 0 \rightarrow x > 0) \end{aligned}$$

Ekkor az $A \rightarrow B = \neg A \vee B$ logikai szabályt alkalmazva az alábbiakat kapjuk:

$$(x \geq 0 \vee -x > 0) \wedge (x < 0 \vee x > 0) = (x \geq 0 \vee x < 0) \wedge (x \neq 0) = (\uparrow) \wedge (x \neq 0) = (x \neq 0)$$

Vagyis az s utasítást minden $x \neq 0$ állapotban biztonságosan végre tudjuk hajtani úgy, hogy R -be érkezzen.

Szimultán értékadás során egyszerre hajtjuk végre az adott értékadásokat:

$$lf(\{x_1, \dots, x_n := y_1, \dots, y_n\}, R) = lf(\{ \parallel_{i=1}^n x_i := y_i \}, R) = R^{x_1 \leftarrow y_1 \dots x_n \leftarrow y_n}$$

Például az $s = (x, y := x + 1, y - 1), R = (2|x + y)$ esetben:

$$\begin{aligned} lf(s, R) &= lf((x, y := x + 1, y - 1), (2|x + y)) \\ &= (2|x + y)^{x \leftarrow x+1, y \leftarrow y-1} \\ &= (2|(x + 1) + (y - 1)) = (2|x + y) \end{aligned}$$

Azaz az s utasítást olyan x, y állapotokban tudjuk biztonságosan végrehajítani úgy, hogy R -be érkezzen, melyre teljesül az, hogy $2|x + y$.

A **feltételes szimultán értékadás** kiszámításának módja ezek után egyértelműen megállapítható az előzőek alapján:

$$\begin{aligned} lf(\{x_1, \dots, x_n := y_1, \dots, y_n, \text{ha } \pi\}, R) &= lf(\{ \parallel_{i=1}^n x_i := y_i, \text{ha } \pi \}, R) = \\ &= (\pi \rightarrow R^{x_1 \leftarrow y_1 \dots x_n \leftarrow y_n}) \wedge (\neg \pi \rightarrow R^{SKIP}) \end{aligned}$$

2.4. Feladatok

A következő példákkal a leggyengébb előfeltétel számítását tudjuk gyakorolni.

1. Legyen $S = (SKIP, \{x := x \cdot 2, \text{ha } 2 \nmid x, \\ x := x + 1, \text{ha } x < 10\})$,
és $R = (2|x)$. Számoljuk ki az S program R -hez tartozó leggyengébb előfeltételét.
2. Legyen $S = (m, n := 0, 0, \{m, n := m + 1, n - 1, \\ n := n + 1, \text{ha } n < m\})$.
Számoljuk ki az S program R -hez tartozó leggyengébb előfeltételét.
 - a) Ha $R = (m = n)$.
 - b) Ha $R = (n > m)$.

3. konzultáció

3.1. Gyakorlás

Először gyakoroljunk pár további lf számolást.

1. Legyen $s = \{x := -x, \text{ha } x > 0\}$, és
 $R = (x > 1)$.
 $lf(s, R) = ?$

$$\begin{aligned}
 [lf(s, R)] &= (x > 0 \rightarrow (x > 1)^{x \leftarrow -x}) \wedge (x \leq 0 \rightarrow x > 1) \\
 &= (x > 0 \rightarrow -x > 1) \wedge (x \leq 0 \rightarrow x > 1) \\
 &= (x \leq 0 \vee -x > 1) \wedge (x > 0 \vee x > 1) \\
 &= (x \leq 0 \vee x < 1) \wedge (x > 0) \\
 &= (x \leq 0) \wedge (x > 0) \equiv Hamis.
 \end{aligned}$$

Átgondolva mit is akartunk kiszámolni, milyen állapotból indulva teljesíthető az a követelmény, hogy $(x > 1)$, ha a programunk a pozitív számokon előjelet vált, a negatívakat (és nullát) pedig nem változtatja?

Ha pozitív bemenetet kap, akkor negatívát csinál belőle, negatívakra pedig nem csinál semmit, tehát sehogy sem tudjuk elérni, hogy tetszőleges bemenetre elérjünk a kívánt állapotba, ezért az azonosan *Hamis* az eredmény.

2. Legyen $S = (SKIP, \{x := 0, \text{ha } y \neq 0, \\ x := x * z, \text{ha } z = 0\})$
 $R = (x = 0)$
 $lf(S, R) = ? \dots \#magic$
 $lf(S, R) = (x = 0) \vee (y \neq 0 \wedge z = 0)$

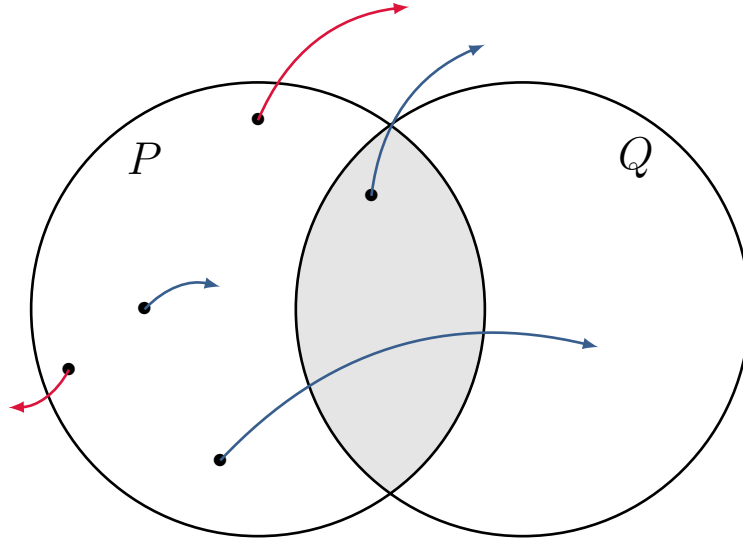
3.2. Stabilitási tulajdonság

Biztonsági tulajdonságot már néztünk a feladatok kapcsán, ám utasításokra (és programokra) is szeretnénk ilyen definiálni.

Legyenek $P, Q : A \rightarrow \mathbb{L}$ állítások, és $s \subseteq A \times A^{**}$ utasítás.

$$P \triangleright_s Q ::= (P \wedge \neg Q) \Rightarrow lf(s, P \vee Q)$$

A P állapotot az utasítás ekkor *ha* elhagyja, akkor ezt csak Q -n keresztül teheti meg.



3. ábra. Nézzünk egy szemléletesebb példát a háromszög tulajdonságra. $P \triangleright_s Q$ akkor és csak akkor teljesül, ha $(P \wedge \neg Q)$ -ból bárhogy választunk állapotokat, az s utasítás hatására ugyanúgy $(P \wedge \neg Q)$ -ban maradunk, vagy pedig Q -ba jutunk. Jó átmeneteknek számítanak a kék nyilak, viszont a piros nyilakról ez nem mondható el.

A *háromszög*re fontos néhány tulajdonságot megjegyezni, melyet a definíció alapján igazolhatunk.

- $P \triangleright_s P = P \wedge \neg P \Rightarrow lf(s, P \vee P) = \downarrow \Rightarrow lf(s, P) = \uparrow \vee lf(s, P) = \uparrow$
- $P \triangleright_s \neg P = P \wedge P \Rightarrow lf(s, P \vee \neg P) = P \Rightarrow lf(s, Igaz) = \uparrow$
- $Igaz \triangleright_s P = Igaz \wedge \neg P \Rightarrow lf(s, Igaz \vee P) = \neg P \Rightarrow lf(s, Igaz) = \uparrow$
- $Hamis \triangleright_s P = Hamis \wedge \neg P \Rightarrow lf(s, Hamis \vee P) = Hamis \Rightarrow lf(s, P) = \uparrow$
- $P \triangleright_s Igaz = P \wedge Hamis \Rightarrow lf(s, P \vee Igaz) = Hamis \Rightarrow lf(s, Igaz) = \uparrow$

Fontos megjegyezni, hogy általános esetben a $P \triangleright_s Hamis$ nem teljesül.

$P \triangleright_s Hamis = P \wedge Igaz \Rightarrow lf(s, P \vee Hamis) = P \Rightarrow lf(s, P)$, de erről nem tudunk tetszőleges program és állítás esetén semmit garantálni.

Egy példa erre, mikor nem teljesül:

$s = (x := x + 1)$ és $P = (2|x)$

$P \triangleright_s Hamis = P \Rightarrow lf(s, P) = (2|x) \Rightarrow lf(x := x + 1, 2|x) = (2|x) \Rightarrow (2|x)^{x \leftarrow x+1} = (2|x) \Rightarrow (2|x + 1)$, ami nyilvánvaló ellentmondás, hiszen ha egy szám páros, akkor a nála eggyel nagyobb szám nem lehet szintén páros (mivel az páratlan).

Abban az esetben azonban, ha $P \triangleright_s Hamis$ teljesül, akkor azt mondjuk, hogy s rendelkezik a P stabil tulajdonsággal, azaz P -ből nem tud kivezetni a program. Jelölésünk erre: $Pstabil_s$ (triviálisan stabil tulajdonság az $Igaz$, hiszen $Igaz \Rightarrow lf(s, Igaz)$ tetszőleges utasításra teljesül).

A háromszög azonban nem tranzitív, azaz $P \triangleright_s Q$ és $Q \triangleright_s R$ -re általában **nem igaz**, hogy $P \triangleright_s R$.

Feladat Számoljuk ki $P \triangleright_s Q$ -t, ha:

$$\begin{aligned} S &= (SKIP, \{s_1 : x := x + 2, \text{ ha } x < 50, \text{ és} \\ &\quad s_2 : x := x + 1, \text{ ha } x \geq 50\}) \\ P &= (2|x) \\ Q &= (x \geq 50) \end{aligned}$$

$$\begin{aligned} P \triangleright_s Q &= lf(S, P \vee Q) \\ (2|x) \wedge (x < 50) &\Rightarrow lf(S, (2|x \vee x \geq 50)) \\ \text{Az } lf \text{ multiplikativitását felhasználva:} \\ lf(S, P \vee Q) &= lf(s_1, P \vee Q) \wedge lf(s_2, P \vee Q) \\ s_1 : lf(x := x + 2, \text{ ha } x < 50, (2|x \vee x \geq 50)) &= \\ &= (x < 50 \rightarrow (2|x \vee x \geq 50)^{x \leftarrow x+2}) \wedge \\ &\quad \wedge (x \geq 50 \rightarrow (2|x \vee x \geq 50)) \\ &= (x \geq 50 \vee 2|x + 2 \vee x + 2 \geq 50) = (x \geq 50 \vee 2|x \vee x \geq 48) = (x \geq 48 \vee 2|x) \\ s_2 : lf(x := x + 1, \text{ ha } x \geq 50, (2|x \vee x \geq 50)) &= \\ &= (x \geq 50 \rightarrow (2|x \vee x \geq 50)^{x \leftarrow x+1}) \wedge \\ &\quad \wedge (x < 50 \rightarrow (2|x \vee x \geq 50)) \\ &= (x \geq 50 \rightarrow 2|(x + 1) \vee x + 1 \geq 50) \wedge (x < 50 \rightarrow 2|x \vee x \geq 50) \\ &= (x < 50 \vee 2|x + 1 \vee x \geq 49) \wedge (x \geq 50 \vee 2|x \vee x \geq 50) \\ &= (\uparrow) \wedge (2|x \vee x \geq 50) \end{aligned}$$

A két eredményt észelve: $lf(S, P \vee Q) = lf(s_1, P \vee Q) \wedge (s_2, P \vee Q)$

$$= (x \geq 48 \vee 2|x) \wedge (x \geq 48 \vee 2|x) = 2|x \vee x \geq 50$$

$$P \wedge \neg Q \Rightarrow lf(S, P \vee Q) =$$

$$(2|x \wedge x < 50) \Rightarrow (2|x \vee x \geq 50) \text{ pedig igaz.}$$

Háromszög esetén szerencsére lehetőségünk van egyszerűsítések használatára, mellyel valamennyire megkönnyíthetjük a számolást.

$$\begin{aligned}
& P \wedge \neg Q \Rightarrow lf(S, P \vee Q) \\
& = P \wedge \neg Q \Rightarrow \bigwedge_{s_i \in S} \left[\left(\pi_i \rightarrow (P \vee Q)^{s_i} \right) \wedge \left(\neg \pi_i \rightarrow (P \vee Q)^{SKIP} \right) \right] \\
& = P \wedge \neg Q \Rightarrow \bigwedge_{s_i \in S} \left[\left(\pi_i \rightarrow (P \vee Q)^{s_i} \right) \wedge \left(\pi_i \vee (P \vee Q)^{SKIP} \right) \right] \\
& \text{felhasználva, hogy } A \Rightarrow (B \wedge C) = (A \Rightarrow B) \wedge (A \Rightarrow C) \\
& = \left(P \wedge \neg Q \Rightarrow \bigwedge_{s_i \in S} \left[\pi_i \rightarrow (P \vee Q)^{s_i} \right] \right) \wedge \\
& \quad \left(P \wedge \neg Q \Rightarrow \bigwedge_{s_i \in S} \left[\pi_i \vee (P \vee Q)^{SKIP} \right] \right) \text{ (skip ág)}
\end{aligned}$$

A skip ágot tovább tudjuk egyszerűsíteni. Ebben az esetben meggondolható, hogy ha a bal oldal hamis, akkor az implikáció igaz lesz. Ha a bal oldal igaz, akkor P értéke igaz és $\neg Q$ is teljesül, tehát Q hamis. A következtetés jobb oldalán álló nagy konjunkciót triviálisan igazzá tudjuk tenni, hiszen ha P igaz a bal oldalt, akkor minden esetben szintén igaz lesz a jobb oldalon is, és *igaz* feltétel mellé bármit "vagyolhatunk", az igazat ad, emellett igaz állítások és is igazat adnak, tehát a "skip" ágra biztosan igaz a feltétel. A későbbiekben ezt a "skip ág elhagyása"-ként említjük (ez nem azt jelenti, hogy az lf-ben a skippel sosem kell foglalkozni, csupán azt, hogy *háromszög* számolása esetén a SKIP ág minden esetben igazat ad). Visszatérve az eredeti lf-hez ekkor:

$$\begin{aligned}
& = \left(P \wedge \neg Q \Rightarrow \bigwedge_{s_i \in S} \left[\pi_i \rightarrow (P \vee Q)^{s_i} \right] \right) \wedge \uparrow \\
& = P \wedge \neg Q \Rightarrow \bigwedge_{s_i \in S} \left[\pi_i \rightarrow (P \vee Q)^{s_i} \right] \\
& = \forall s_i \in S : (P \wedge \neg Q) \Rightarrow (\pi_i \rightarrow (P \vee Q)^{s_i})
\end{aligned}$$

A mostani formulát "Ha igaz A , és akkor ha igaz B , teljesül -e C ?" módon tudjuk értelmezni, amit szóban (és logikában is) át tudunk fogalmazni "Ha igaz A és B , igaz -e C ?" módra. Ezt fogjuk a későbbiekben *feltétel átvitelének* nevezni.

$$P \triangleright_S Q = \forall s_i \in S : (P \wedge \neg Q \wedge \pi_i) \Rightarrow (P \vee Q)^{s_i}$$

Ismét kiemelnénk, hogy a "*skip+feltétel*" módszerrel való számolás csak és kizárólag a *háromszög* számolása esetén használható.

Teljesül -e a stabilitási tulajdonság az itt megadott állításokra és programra nézve? (Azaz számoljuk ki $P \triangleright_S Q$ -t, ha:)

$S = (SKIP, \{s_1 : x := x + 2, \text{ ha } x < 50, \text{ és}$

$s_2 : x := x + 1, \text{ ha } x \geq 50\})$

$P = (2|x)$

$Q = (x \geq 50)$

SKIP + feltétel miatt:

$$P \triangleright_S Q = \forall s_i \in S : (P \wedge \neg Q \wedge \pi_i) \Rightarrow (P \vee Q)^{s_i}$$

$$s_1 : (2|x) \wedge (x < 50) \wedge (x < 50) \Rightarrow (2|x \vee x \geq 50)^{x \leftarrow x+2} =$$

$$(2|x \wedge x < 50) \Rightarrow (2|x \vee x \geq 48) = \uparrow$$

$$s_2 : (2|x) \wedge (x < 50) \wedge (x \geq 50) \Rightarrow (2|x \vee x \geq 50) =$$

$$\downarrow \Rightarrow (2|x \vee x \geq 50) = \uparrow$$

$$P \triangleright_S Q = \uparrow \wedge \uparrow = \uparrow$$

Látszik, hogy így mennyivel egyszerűbben megkapjuk ugyan azt az eredményt, amit korábban egy oldalon keresztül kellett számolni.

A biztonság kedvéért nézzünk meg még egy példát.

Legyen $S = (SKIP, \{s_1 : x := x + 1,$

$s_2 : x := x - 1, \text{ ha } 2|x\})$,

valamint $P = (x = 3)$ és $Q = (x = 4)$. Számoljuk ki $P \triangleright_S Q$ -t! Ekkor

$$P \triangleright_S Q = (P \triangleright_{s_1} Q) \wedge (P \triangleright_{s_2} Q)$$

$$P \triangleright_{s_1} Q = (P \wedge \neg Q) \Rightarrow lf(s_1, P \vee Q)$$

$$= (x = 3 \wedge x \neq 4) \Rightarrow (x = 3 \vee x = 4)^{x \leftarrow x+1}$$

$$= (x = 3 \wedge x \neq 4) \Rightarrow (x + 1 = 3 \vee x + 1 = 4)$$

$$= (x = 3) \Rightarrow (x = 2 \vee x = 3)$$

Mivel $\{3\} \subseteq \{2, 3\}$:

$$= \uparrow$$

$$P \triangleright_{s_2} Q = (P \wedge \neg Q) \Rightarrow lf(s_2, P \vee Q)$$

SKIP + feltétel miatt:

$$= (P \wedge \neg Q \wedge \pi_2) \Rightarrow (P \vee Q)^{s_2}$$

$$= (x = 3 \wedge x \neq 4 \wedge 2|x) \Rightarrow (x = 3 \vee x = 4)^{x \leftarrow x-1}$$

$$= (x = 3 \wedge x \neq 4 \wedge 2|x) \Rightarrow (x - 1 = 3 \vee x - 1 = 4)$$

$$= (x = 3 \wedge 2|x) \Rightarrow (x = 4 \vee x = 5)$$

$$= \downarrow \Rightarrow (x = 4 \vee x = 5)$$

$$= \uparrow$$

$$P \triangleright_S Q = (P \triangleright_{s_1} Q) \wedge (P \triangleright_{s_2} Q) = \uparrow \wedge \uparrow = \uparrow$$

4. Megoldások

1.2 - Étkező filozófusok

- $FP \Rightarrow (\forall i : \neg f(i).e)$
- $f(i).o \triangleright \perp$

1.3 - Moziterem

- $n(i).a \hookrightarrow n(i).f$
- $n(i).t \hookrightarrow n(i).f$
- $FP \Rightarrow \forall i : n(i).h$
- $n(i).a \triangleright (n(i).j \vee n(i).b)$
- $n(i).f \mapsto n(i).h$
- $(\forall i : n(i).f \Rightarrow n(i).t) \in inv$
- $n(i).h \triangleright \perp$

2.4 - Feladatok

1. Ahogy korábban láttuk 2.4 bekezdésben, egy program leggyengébb előfeltétele az utasításai által adott leggyengébb előfeltételek konjugáltja. Az egyszerűség kedvéért nevezzük el az $x := x \cdot 2$, ha $2 \nmid x$ utasítást s_1 -nek, az $x := x + 1$, ha $x < 10$ utasítást

pedig s_2 -nek. Ekkor

$$\begin{aligned} lf(s_1, R) &= lf(x := x \cdot 2, ha\ 2 \nmid x; (2|x)) \\ &= (2 \nmid x \rightarrow (2|x)^{x \leftarrow x \cdot 2}) \wedge (2|x \rightarrow 2|x) \\ &= (2 \nmid x \rightarrow 2|x \cdot 2) \wedge (2|x \rightarrow 2|x) \end{aligned}$$

Alkalmazva a $A \rightarrow B = \neg A \vee B$ szabályt a következőt kapjuk:

$$= (2|x \vee 2|x \cdot 2) \wedge (2 \nmid x \vee 2|x)$$

Mivel $A \vee \neg A = \uparrow$, valamint egy szám kétszeresét véve osztható lesz kettővel:

$$\begin{aligned} &= (2|x \vee \uparrow) \wedge (\uparrow) \\ &= (\uparrow) \wedge (\uparrow) \\ &= \uparrow \end{aligned}$$

$$\begin{aligned} lf(s_2, R) &= lf(x := x + 1, ha\ x < 10; (2|x)) \\ &= (x < 10 \rightarrow (2|x)^{x \leftarrow x+1}) \wedge (x \geq 10 \rightarrow 2|x) \\ &= (x < 10 \rightarrow 2|x + 1) \wedge (x \geq 10 \rightarrow 2|x) \\ &= (x \geq 10 \vee 2|x + 1) \wedge (x < 10 \vee 2|x) \end{aligned}$$

Picit tovább alakítva a pedig a következő kifejezést kapjuk:

$$= (x \geq 10 \vee 2|x + 1) \wedge (x < 10 \vee 2|x)$$

$$\begin{aligned} lf(S, R) &= lf(s_1, R) \wedge lf(s_2, R) \\ &= (\uparrow) \wedge ((x \geq 10 \vee 2|x + 1) \wedge (x < 10 \vee 2|x)) \end{aligned}$$

Mivel $\uparrow \wedge A = A$:

$$= (x \geq 10 \vee 2|x + 1) \wedge (x < 10 \vee 2|x)$$

2. Hasonlóan az előző feladathoz, nevezzük el $m, n := m + 1, n - 1$ és $n := n + 1, ha\ n < m$ utasításokat rendre s_1 -nek és s_2 -nek.

$$\text{a) } R = (m = n)$$

$$\begin{aligned} lf(s_1, R) &= lf(m, n := m + 1, n - 1; (m = n)) \\ &= (m = n)_{n \leftarrow n-1}^{m \leftarrow m+1} \\ &= (m + 1 = n - 1) \\ &= (m = n - 2) \end{aligned}$$

$$\begin{aligned} lf(s_2, R) &= lf(n := n + 1, \text{ha } n < m; (m = n)) \\ &= (n < m \rightarrow (m = n)_{n \leftarrow n-1}^{m \leftarrow m+1}) \wedge (n \geq m \rightarrow m = n) \\ &= (n < m \rightarrow m + 1 = n - 1) \wedge (n \geq m \rightarrow m = n) \\ &= (n \geq m \vee m + 1 = n - 1) \wedge (n < m \vee m = n) \\ &= (n \geq m \vee m = n - 2) \wedge (n \leq m) \end{aligned}$$

Alkalmazva a $(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$ szabályt:

$$\begin{aligned} &= (n \geq m \wedge n \leq m) \vee (m = n - 2 \wedge n \leq m) \\ &= (n = m) \vee \downarrow \end{aligned}$$

Mivel $A \vee \downarrow = A$:

$$= (n = m)$$

$$\begin{aligned} lf(S, R) &= lf(s_1, R) \wedge lf(s_2, R) \\ &= (m = n - 2) \wedge (n = m) \\ &= \downarrow \end{aligned}$$

b) $R = (m < n)$

$$\begin{aligned} lf(s_1, R) &= lf(m, n := m + 1, n - 1; (m < n)) \\ &= (m < n) \stackrel{m \leftarrow m+1}{n \leftarrow n-1} \\ &= (m + 1 < n - 1) \\ &= (m + 2 < n) \\ lf(s_2, R) &= lf(n := n + 1, ha\ n < m; (m < n)) \\ &= (n < m \rightarrow (m < n) \stackrel{m \leftarrow m+1}{n \leftarrow n-1}) \wedge (n \geq m \rightarrow m < n) \\ &= (n < m \rightarrow m + 1 < n - 1) \wedge (n \geq m \rightarrow m < n) \\ &= (n \geq m \vee m + 1 < n - 1) \wedge (n < m \vee m < n) \\ &= (n \geq m \vee m + 2 < n) \wedge (n \neq m) \\ &= (n \geq m) \wedge (n \neq m) \\ &= (n > m) \\ lf(S, R) &= lf(s_1, R) \wedge lf(s_2, R) \\ &= (m + 2 < n) \wedge (n > m) \\ &= (m + 2 < n) \wedge (m < n) \\ &= (m + 2 < n) \end{aligned}$$

Hivatkozások

- [1] dr. Horváth Zoltán: Párhuzamos és elosztott programozás
(<http://people.inf.elte.hu/hz/parh/jegyzet.ps>)
- [2] Fóthi Ákos: Bevezetés a programozáshoz
(<http://bzsr.web.elte.hu/progmod2/konyv.pdf>)