Programozáselmélet - Szükséges matematikai alapok

Készítette: Borsi Zsolt

1. Halmazok

Matematikában gyakran használt halmazok jelölései:

$\mathbb{N}$	– a természetes számok halmaza, a nullát is beleértve
$\mathbb{N}^+$	– a pozitív egész számok halmaza
$\mathbb{Z}$	– az egész számok halmaza
$\mathbb{L}$	– a logikai értékek kételemű halmaza
$\emptyset$	– az üreshalmaz

A halmazokat gyakran vagy az elemeik felsorolásával (például a logikai értékek halmaza:)

$$\mathbb{L} ::= \{igaz, hamis\}$$

vagy egy tulajdonság megfogalmazásával (például a 100-nál nem nagyobb 5-tel osztható egészek halmaza:)

$$\{\, x \in \mathbb{Z} \mid x \leqslant 100 \wedge 5|x \,\}$$

adjuk meg.

Definíció (Intervallum)***:*** Az $[a..b] ::= \{\, x \in \mathbb{Z} \mid a \leqslant x \wedge x \leqslant b \,\}$ halmazt (ahol a és b egész számok) *intervallumnak* nevezzük. Ami üres, ha $a > b$.

Jelölés: Egy H halmaz számosságát $|H|$ jelöli. Azt, hogy egy H halmaz véges, így is írhatjuk: $|H| < \infty$.

2. Reláció

Definíció: Legyenek A és B tetszőleges nemüres halmazok. Ekkor az $A \times B$ halmaz az A és B *Descartes szorzata*, és

$$A \times B ::= \{\, (a, b) \mid a \in A \wedge b \in B \,\}$$

$A \times B$ elemei tehát olyan rendezett párok, ahol a pár első komponense A-ból, a második B-ből van.

Definíció (Reláció)***:*** Legyenek A és B tetszőleges nemüres halmazok. Ekkor az $A \times B$ halmaz minden R részhalmazát (tehát az üreshalmazt is) *relációnak* nevezzük.

Ha $(x, y) \in R$, akkor azt mondjuk hogy az R reláció x-hez hozzárendeli y-t.

Definíció: Legyenek A és B tetszőleges nemüres halmazok és $R \subseteq A \times B$ tetszőleges reláció. Az R reláció értelmezési tartománya:

$$\mathcal{D}_R ::= \{\, a \in A | \exists b \in B : (a, b) \in R \,\}$$

a reláció értékkészlete:

$$\mathcal{R}_R ::= \{\, b \in B | \exists a \in A : (a, b) \in R \,\}$$

a reláció értéke egy a helyen, vagy másképpen az a pont R reláció szerinti képe:

$$R(a) ::= \{\, b \in B | (a, b) \in R \,\}$$

Az $R \subseteq A \times B$ relációt felfoghatjuk egy leképezésnek, megfeleltetésnek is az A és a B halmaz elemei között.

3. Függvény

Speciális reláció a függvény.

Definíció (Függvény)***:*** Legyenek A és B tetszőleges nemüres halmazok és $R \subseteq A \times B$ tetszőleges reláció. Azt mondjuk hogy az R reláció determinisztikus, ha

$$\forall a \in A : |R(a)| \leqslant 1$$

A determinisztikus relációkat másképpen függvényeknek hívjuk. Az $R \subseteq A \times B$ függvénynek, mint speciális relációnak külön jelölést vezetünk be: $R \in A \to B$.

Jelölés: Ha az $f \in A \to B$ függvényre még az is teljesül, hogy értelmezési tartománya megegyezik az A halmazzal, vagyis ha

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

akkor a következő jelölést alkalmazzuk: $f \colon A \to B$.

Megjegyzés: Az olyan $f \in A \to B$ függvényeket melyek nem rendelnek minden A-beli elemhez egy B-beli elemet (vagyis nincsenek mindenhol értelmezve az A felett), parciális függvényeknek nevezzük.

Megjegyzés: Legyen $f \colon A \to B$ függvény (tehát tudjuk hogy az f reláció az A halmaz minden eleméhez pontosan egy B-belit rendel) és $a \in A$. Legyen továbbá $f(a) = \{b\}$ ahol $b \in B$ az a-hoz rendelt egyetlen elem. Ebben az esetben az $f(a)$ képet sokszor nem a $\{b\}$ egyelemű képhalmazként hanem egyszerűen csak mint b írjuk.

Programozáselmélet - Programozási alapfogalmak

Készítette: Borsi Zsolt

1. Állapottér

A feladat adatokról szól, a program is adatokkal dolgozik. Egy adat típusérték-halmaza az adat lehetséges értékeiből áll.

Definíció (Állapot)***:*** Legyenek $A_1, \ldots, A_n$ (ahol $n \in \mathbb{N}^+$) típusérték-halmazok és $v_1, \ldots, v_n$ a halmazokat azonosító egyedi címkék (változók). Az ezekből képzett, címkézett értékeknek egy $\{v_1{:}a_1, \ldots, v_n{:}a_n\}$ halmazát (ahol $\forall i \in [1..n] : a_i \in A_i$) *állapotnak* nevezzük.

Egy-komponensű állapottér esetén $\{v_1{:}a_1\}$ helyett írhatunk a_1-et is.

Definíció (Állapottér)***:*** Legyenek $A_1, \ldots, A_n$ (ahol $n \in \mathbb{N}^+$) típusérték-halmazok és $v_1, \ldots, v_n$ a halmazokat azonosító egyedi címkék (változók). Az ezekből képzett összes lehetséges $\{v_1{:}a_1, \ldots, v_n{:}a_n\}$ állapot (ahol $\forall i \in [1..n] : a_i \in A_i$) halmazát *állapottérnek* nevezzük és $(v_1{:}A_1, \ldots, v_n{:}A_n)$-nel jelöljük.

$$(v_1{:}A_1, \ldots, v_n{:}A_n) ::= \left\{ \{v_1{:}a_1, \ldots, v_n{:}a_n\} \,|\forall i \in [1..n] : a_i \in A_i \right\}$$

Definíció (Változó)***:*** Az $A = (v_1{:}A_1, \ldots, v_n{:}A_n)$ állapottér cimkéire (*változók*) úgy tekintünk mint $v_i \colon A \to A_i$ függvényekre, ahol $v_i(a) = a_i$ egy $a = \{v_1{:}a_1, \ldots, v_n{:}a_n\}$ állapot esetén.

Definíció (Altér)***:*** Legyenek $A = (v_1{:}A_1, \ldots, v_n{:}A_n)$ és $B = (u_1{:}B_1, \ldots, u_n{:}B_m)$ állapotterek ($n, m \in \mathbb{N}^+$ és $m \leqslant n$). Azt mondjuk, hogy az A állapottérnek *altere* a B állapottér ($B \leq A$), ha van olyan $\varphi \colon [1..m] \to [1..n]$ injekció, amelyre $\forall i \in [1..m] : B_i = A_{\varphi(i)}$.

2. Feladat

Definíció (Feladat)***:*** Legyen A tetszőleges állapottér. Feladatnak nevezünk egy $F \subseteq A \times A$ relációt.

A feladat fenti definíciója természetes módon adódik abból, hogy a feladatot egy leképezésnek tekintjük az állapottéren, és az állapottér minden pontjára megmondjuk, hova kell belőle eljutni, ha egyáltalán el kell jutni belőle valahova.

3. Sorozatok

Jelölés: Ha H tetszőleges halmaz, akkor jelölje H^{**} az olyan (akár véges, vagy akár végtelen) sorozatok halmazát, mely sorozatok elemei mind a H halmazból valók. A H-beli

véges sorozatok halmazát H^*-gal, a végtelen sorozatok halmazát H^∞-nel jelöljük. Tehát $H^{**} = H^* \cup H^\infty$. Egy $\alpha \in H^*$ sorozat hosszát jelölje $|\alpha|$, végtelen sorozat esetén legyen $|\alpha| = \infty$.

4. Program

Definíció (Program)***:*** Legyen A az úgynevezett alap-állapottér ($fail \notin A$). Jelölje $\bar{A}$ azon véges komponensű állapotterek unióját, melyeknek altere az A alap-állapottér: $\bar{A} = \bigcup_{A \leq B} B$.

Az A feletti *programnak* hívjuk az $S \subseteq A \times (\bar{A} \cup \{\textbf{fail}\})^{**}$ relációt, ha

1. $\mathcal{D}_S = A$
2. $\forall a \in A$: $\forall \alpha \in S(a) : |\alpha| \geq 1$ és $\alpha_1 = a$
3. $\forall \alpha \in \mathcal{R}_S : (\forall i \in \mathbb{N}^+ : i < |\alpha| \rightarrow \alpha_i \neq fail)$
4. $\forall \alpha \in \mathcal{R}_S : (|\alpha| < \infty \rightarrow \alpha_{|\alpha|} \in A \cup \{fail\})$

5. Elemi programok

Definíció (Elemi program)***:*** Legyen A tetszőleges állapottér. Az $S \subseteq A \times (\bar{A} \cup \{\textbf{fail}\})^{**}$ programot *elemi programnak* nevezzük, ha

$$\forall a \in A\colon S(a) \subseteq \{< a >, < a, fail >, < a, a, a, \cdots >, < a, b > | b \in A\}$$

A definíció szerint minden programhoz található vele ekvivalens elemi program, csak a sorozatok közbülső elemeit el kell hagyni, így lényegében (egy adott szinten) minden program elemi. Az elemi programok közül kiválasztunk néhány speciális tulajdonsággal rendelkezőt, és a továbbiakban velük foglalkozunk.

Definíció: Legyen A tetszőleges állapottér. $SKIP$ jelöli azt a programot, melyre

$$\forall a \in A\colon SKIP(a) = \{< a >\}$$

A $SKIP$ az állapottér minden a állapotához egyetlen sorozatot, az $< a >$ sorozatot rendeli. Így a-ból indulva a $SKIP$ garantált hogy a-ba jut, és csak oda.

Definíció: Legyen A tetszőleges állapottér. $ABORT$ jelöli azt a programot, melyre

$$\forall a \in A\colon ABORT(a) = \{< a, fail >\}$$

Az $ABORT$ az állapottér minden a állapotához egyetlen sorozatot, az $< a, fail >$ sorozatot rendeli. Így a-ból indulva az $ABORT$ programnak nincs más végrehajtása, mint a $fail$ állapotban végződő végrehajtás.
A harmadik speciális elemi program az értékadás, amivel az állapottér egyes komponenseinek (változóinak) értéke megváltoztatható.

Programozáselmélet - A megoldás fogalma

Készítette: Borsi Zsolt

1. Programfüggvény

Definíció: A $p(S) \subseteq A \times A$ reláció az $S \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ program *programfüggvénye*, ha

1. $\mathcal{D}_{p(S)} = \{a \in A \mid S(a) \subseteq \bar{A}^*\}$
2. $\forall a \in \mathcal{D}_{p(S)} \colon p(S)(a) = \{b \in A \mid \exists \alpha \in S(a) \colon b = \alpha_{|\alpha|}\}$

2. Megoldás

Definíció: Azt mondjuk hogy az S program megoldja az F feladatot (más szavakkal: az S program teljesen helyes az F feladatra nézve), ha

1. $\mathcal{D}_F \subseteq \mathcal{D}_{p(S)}$
2. $\forall a \in \mathcal{D}_F \colon p(S)(a) \subseteq F(a)$

3. Parciális helyesség

Definíció (Gyenge programfüggvény)***:*** A $\tilde{p}(S) \subseteq A \times (A \cup \{fail\})$ reláció az $S \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ program gyenge programfüggvénye, ha

1. $\mathcal{D}_{\tilde{p}(S)} = \{a \in A \mid S(a) \cap (\bar{A} \cup \{fail\})^* \neq \emptyset\}$
2. $\forall a \in \mathcal{D}_{\tilde{p}(S)} \colon \tilde{p}(S)(a) = \{b \in A \cup \{fail\} \mid \exists \alpha \in S(a) \cap (\bar{A} \cup \{fail\})^* \colon b = \alpha_{|\alpha|}\}$

Definíció (Parciális helyesség)***:*** Azt mondjuk hogy az S program parciálisan helyes az F feladatra nézve, ha

1. $\forall a \in \mathcal{D}_F \colon \tilde{p}(S)(a) \subseteq F(a)$

Programozáselmélet - A leggyengébb előfeltétel

Készítette: Borsi Zsolt

1. Nevezetes logikai függvények

Definíció: Legyen A tetszőleges halmaz. $HAMIS$ jelöli azt a logikai függvényt, melyre

$$\forall a \in A\colon HAMIS(a) = \{hamis\}$$

Definíció: Legyen A tetszőleges halmaz. $IGAZ$ jelöli azt a logikai függvényt, melyre

$$\forall a \in A\colon IGAZ(a) = \{igaz\}$$

Azaz a $HAMIS$ logikai függvény egy adott A halmaz minden eleméhez a $hamis$, az $IGAZ$ az $igaz$ értéket rendeli.

Jelölés (Igazsághalmaz)***:*** Legyen $R \in A \to \mathbb{L}$ logikai függvény. Ekkor $\lceil R \rceil$ jelöli az olyan állapottérbeli pontok halmazát ahol R igaz. Azaz

$$\lceil R \rceil = \{a \in A \mid R(a) = \{igaz\}\}$$

Az $\lceil R \rceil$ halmazt az R logikai függvény *igazsághalmazának* nevezzük.

Ne felejtsük el hogy $R \in A \to \mathbb{L}$ nem feltétlenül értelmezett az A minden pontjában. Amennyiben $R\colon A \to \mathbb{L}$, akkor már viszont igaz hogy egy $a \in A$ pontban ha R nem igaz, akkor hamis.

2. A „következik” reláció

Definíció: Legyenek $Q, R \in A \to \mathbb{L}$ tetszőleges logikai függvények. Amennyiben $\lceil Q \rceil \subseteq \lceil R \rceil$ teljesül, akkor azt mondjuk hogy Q maga után vonja R-t (vagy másképp: Q-ból következik R) és a következőképpen jelöljük: $Q \implies R$.

Vegyük észre, hogy ha $Q \implies R$, akkor ez azt jelenti, hogy minden olyan $a \in A$ pontra amire Q igaz, arra igaz R is.

Példa: Legyen $A = \{1, 2, 3, 4\}$ és $Q, R \in A \to \mathbb{L}$ logikai függvények úgy hogy $\lceil Q \rceil = \{1, 3, 4\}$ és $\lceil R \rceil = \{1, 3\}$. Ebben az esetben $Q \implies R$ nem teljesül (mert a 4-re igaz Q de R nem), de $R \implies Q$ igen.

Példa: Legyen $A = (a{:}\mathbb{N}, h{:}\mathbb{N})$ és $Q, R \in A \to \mathbb{L}$ logikai függvények úgy hogy $Q = (a = 10)$ és $R = (h = a^3)$. Ugyan van olyan A-beli pont (az A halmaz most speciálisan egy állapottér, tehát elemei állapotok) amihez Q és R is igazat rendel, méghozzá az $\{a{:}10, h{:}1000\}$ állapot, de az nem igaz hogy $Q \implies R$, hiszen például $\{a{:}10, h{:}82\} \in \lceil Q \rceil$, de az $\{a{:}10, h{:}82\}$ elemhez R hamisat rendel.

3. Leggyengébb előfeltétel

Definíció: Legyen $S \subseteq A \times (\bar{A} \cup \{\textbf{fail}\})^{**}$ program, $R \in A \to \mathbb{L}$ logikai függvény. Ekkor az S program R utófeltételhez tartozó leggyengébb előfeltétele az az $lf(S, R)\colon A \to \mathbb{L}$ függvény, amelyre

$$\lceil lf(S, R) \rceil = \{a \in A \mid a \in D_{p(S)} \wedge p(S)(a) \subseteq \lceil R \rceil\}$$

A leggyengébb előfeltétel tehát pontosan azokban a pontokban igaz, ahonnan kiindulva az S program biztosan hibátlanul terminál, és az összes lehetséges végállapotban igaz R.

Tétel (Az lf tulajdonságai)***:*** Legyen $S \subseteq A \times (\bar{A} \cup \{\textbf{fail}\})^{**}$ program, $Q, R \in A \to \mathbb{L}$ logikai függvények. Ekkor

1. $lf(S, HAMIS) = HAMIS$
2. ha $Q \implies R$ akkor $lf(S, Q) \implies lf(S, R)$
3. $lf(S, Q) \wedge lf(S, R) = lf(S, Q \wedge R)$
4. $lf(S, Q) \vee lf(S, R) \implies lf(S, Q \vee R)$

Példa: Legyen $A = (x{:}\mathbb{N})$. $R\colon A \to \mathbb{L}$ logikai függvény adott, $R = (x < 10)$. Számoljuk ki az $x := x - 5$ értékadásnak az R utófeltételhez tartozó leggyengébb előfeltételét.

Először megvizsgáljuk hogyan viselkedik az $x := x - 5$ program az állapottér néhány pontjában: az $\{x{:}8\}$ ponthoz az $< \{x{:}8\}, \{x{:}3\} >$ sorozatot, míg az $\{x{:}2\}$ állapothoz az $< \{x{:}2\}, fail >$ sorozatot rendeli. A program programfüggvénye olyan $\{x{:}a_1\}$ állapotokban van értelmezve, ahol $a_1 \geqslant 5$, innen indulva a program garantáltan olyan pontban terminál ahol x értéke $a_1 - 5$. Egyéb állapotból indulva a program a $fail$ állapotban terminál.

Felhasználva a leggyengébb előfeltétel definícióját, és az $x := x - 5$ értékadást S-sel jelölve, felírhatjuk:

$$\begin{aligned}\lceil lf(S, R) \rceil = \{a \in A \mid a \in D_{p(S)} \wedge p(S)(a) \subseteq \lceil R \rceil\} = \\ \{a \in A \mid x(a) \geqslant 5 \wedge \{x(a) - 5\} \subseteq \lceil R \rceil\} = \\ \{a \in A \mid x(a) \geqslant 5 \wedge x(a) - 5 \in \lceil R \rceil\} = \\ \{a \in A \mid x(a) \geqslant 5 \wedge x(a) - 5 < 10\}\end{aligned}$$

Azaz azt kaptuk, hogy az $lf(S, R)$ pontosan akkor igaz ha $(5 \leqslant x < 15)$. Ne felejtsük el hogy az A állapottéren az egyetlen változónk neve x és most számoltuk ki azon állapotok halmazát ahol a leggyengébb előfeltétel igaz.

A leggyengébb előfeltétel fogalma nagyon fontos, ugyanakkor nagyon egyszerű. Vegyük észre hogy az előbbi példában azt számoltuk ki, hogy az x értéke 15-nél kisebb kell legyen, hogy az $x := x - 5$ értékadást végrehajtva olyan pontban terminálunk ahol x értéke 10-nél kisebb. Továbbá az x értéke legalább 5 kell legyen, hogy az $x := x - 5$ értékadás hibátlanul működjön (ne ffeljtsük: az x típusa természetes szám).
Természetesen igaz az is hogy $x \in [8..12]) \implies lf(x := x - 5, x < 10$, azaz hogy ha az x-hez tartozó érték a $[8..12]$ halmazból van, akkor az $x := x - 5$ biztos hogy hibátlanul terminál, méghozzá olyan állapotban ahol $x < 10$ teljesül. Mindez azért van, mert az $x \in [8..12]$ feltétel szigorúbb mint az a leggyengébb előfeltétel amit előbb kiszámoltunk.
Általánosan: ha valamely P logikai függvényre teljesül hogy $P \implies lf(S, R)$ (azaz P szigorúbb mint az $lf(S, R)$ feltétel) akkor a P tulajdonságú pontokból indulva az S program biztos hogy helyesen terminál és a végpontokban igaz R. A leggyengébb előfeltételt ezért hívják „leggyengébb előfeltételnek”.

Programozáselmélet - A specifikáció tétele

Készítette: Borsi Zsolt

1. Specifikáció tétele

Definíció: Azt mondjuk hogy a B halmaz az $F \subseteq A \times A$ feladat egy paramétertere, ha léteznek olyan $F_1 \subseteq A \times B$ és $F_2 \subseteq B \times A$ relációk, melyekre $F = F_2 \circ F_1$.

Megjegyzés: Bármely $F \subseteq A \times A$ feladatnak létezik paramétertere. Hiszen legyen $B = A$, és válasszuk az $F_1 \subseteq A \times B$ és $F_2 \subseteq B \times A$ relációkat úgy, hogy $F_1 = id$ (azaz a minden A-beli elemhez önmagát rendelő reláció) és $F_2 = F$. Ekkor nyilvánvalóan teljesül hogy $F \circ id = F$.

Definíció: Legyenek A és B tetszőleges nemüres halmazok és $R \subseteq A \times B$ tetszőleges reláció. Az R reláció inverze:

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

azaz olyan B-ről A-ra képező reláció, ami pontosan akkor tartalmaz egy $(b, a) \in B \times A$ párt, ha $(a, b) \in R$.

Tétel: Legyen $F \subseteq A \times A$ feladat, B az F egy paramétertere (azaz léteznek olyan $F_1 \subseteq A \times B$ és $F_2 \subseteq B \times A$ relációk úgy hogy $F = F_2 \circ F_1$). Legyen $b \in B$ tetszőleges paraméter, amihez definiáljuk a $Q_b \colon A \to \mathbb{L}$ és $R_b \colon A \to \mathbb{L}$ logikai függvényeket az igazsághalmazaik megadásával:

$$\lceil Q_b \rceil ::= F_1^{(-1)}(b)$$

$$\lceil R_b \rceil ::= F_2(b)$$

Ekkor ha $\forall b \in B : Q_b \implies lf(S, R_b)$ akkor az S program megoldja az F feladatot.

$\lceil Q_b \rceil = \{a \in A \mid (a, b) \in F_1\}$, azaz Q_b igazsághalmazában olyan $a \in A$ állapotok vannak, melyekhez az F_1 reláció hozzárendeli a $b \in B$ paramétert.
$\lceil R_b \rceil = \{a \in A \mid (b, a) \in F_2\}$, azaz R_b igazsághalmazában olyan $a \in A$ állapotok vannak, melyeket az F_2 reláció a $b \in B$ paraméterhez rendel.

2. Feladat specifikációja

Tekintsük azt a feladatot, amikor egy adott pozitív egész egy osztóját kell megadnunk. A feladat állapottere $A = (n{:}\mathbb{N}^+, d{:}\mathbb{N}^+)$. Ezt a feladatot le tudjuk formálisan írni, mint olyan $(u, v) \in A \times A$ párok halmaza ahol u és v állapotok n változó szerinti értékei megegyeznek és

v célállapot d változóhoz tartozó értéke osztója az u kiindulási állapot n változóhoz tartozó értéknek:

$$\{(u, v) \in A \times A \mid n(u) = n(v) \wedge d(v)|n(u)\}$$

Felhasználva a specifikáció tételének jelöléseit, írjuk fel más módon - de formálisan - a feladat specifikációját. Azt vesszük észre, hogy minden $a \in A$ állapothoz melyekre $n(a)$ megegyezik, a feladat ugyanazt rendeli; nem függ a kiindulási állapot d változó szerinti értékétől. Írjuk fel az F feladatot a F_1 és F_2 relációk kompozíciójaként, úgy hogy azokhoz az állapotokhoz melyeknek F szerinti képe azonos, F_1 ugyanazt a paramétert rendelje. Mivel ezek az állapotok megegyeznek az n változóhoz tartozó értékükben, célszerű hozzájuk ugyanezt a (címkézett) értéket rendelni, F_1 reláció szerint. Azaz, a feladat egy paramétertere legyen a pozitív egész számok halmaza, ahol az értékre az n' változó segítségével (egy komponens lévén, nem lenne szükség változóra, de általános esetben kell) hivatkozhatunk: $B = (n'{:}\mathbb{N}^+)$.
Azt, hogy F_1 pontosan akkor rendel egy $a \in A$ állapothoz egy $b \in B$ értéket, a specifikáció tételében szereplő Q_b logikai függvénnyel adhatjuk meg. Legyen $b \in B$ tetszőleges, ekkor
$\forall a \in A : Q_b(a) = (n(a) = n'(b))$.
Természetesen úgy kapjuk meg F feladatot az F_1 és F_2 relációk kompozíciójaként, ha F_2 a $b \in B$ paraméterhez olyan állapottérbeli a pontot rendel, melynek d változóhoz tartozó értéke osztója a kiindulási állapot n változóhoz tartozó értékének. Épp ezért tetszőleges $b \in B$ esetén legyen R_b olyan logikai függvény, melyre
$\forall a \in A : R_b(a) = (n(a) = n'(b) \wedge d(a)|n(a))$.
Vegyük észre hogy a $n(a) = n'(b)$ kikötésre szükségünk van, anélkül csak annyit mondanánk hogy a célállapotban a d változóhoz tartozó érték osztója kell legyen az n változóhoz tartozó aktuális értéknek, nem fogalmaznánk meg szorosabb kapcsolatot a kiindulási állapot és a célállapot között. A feladat specifikációja tehát

$A = (n{:}\mathbb{N}^+, d{:}\mathbb{N}^+)$
$B = (n'{:}\mathbb{N}^+)$
$\forall b \in B : Q_b(a) = (n(a) = n'(b))$ (ahol $a \in A$ tetszőleges állapot)
$\forall b \in B : R_b(a) = (n(a) = n'(b) \wedge d(a)|n(a))$ (ahol $a \in A$ tetszőleges állapot)

A továbbiakban a feladatnak ezen leírását (tehát ami tartalmazza a feladat állapotterét és a feladat egy paraméterterének megadását; minden $b \in B$ paraméterre a hozzá tartozó Q_b és R_b logikai függvények definícióját) a feladat specifikációjának nevezzük. Mivel d az A állapottérről $\mathbb{N}$-re képező függvény (azaz argumentumába egy $a \in A$ elemet írhatunk), hasonlóan Q_b egy $b \in B$ paraméterhez definiált, $a \in A$ állapotokhoz logikai értéket rendelő függvény; az egyértelmű jelöléseket elhagyva a következőket kapjuk:
$A = (n{:}\mathbb{N}^+, d{:}\mathbb{N}^+)$
$B = (n'{:}\mathbb{N}^+)$
$Q = (n = n')$
$R = (Q \wedge d|n)$

Programozáselmélet - Programkonstrukciók

Készítette: Borsi Zsolt

Ebben a fejezetben azzal foglalkozunk, hogy lehet meglévő programokból új programokat készíteni. Háromféle konstrukciót engedünk meg: szekvencia, elágazás és ciklus. Ezek definícióit úgy adjuk meg, hogy a velük képzett relációk illeszkedjenek a korábban bevezetett program fogalmához. A programkonstrukciókat struktogrammal ábrázoljuk.

1. Szekvencia

Szekvencia esetében két programot egymás után végzünk el. Amennyiben az első program végrehajtása adott állapotból indulva nem véges vagy hibásan terminál, a második program nem tudja folytatni a végrehajtást a végpontból; az első program által generált végrehajtás egy lehetséges végrehajtása lesz a szekvenciának is.
Nevezzük csatlakozási pontnak azt az állapotot egy végrehajtási sorozatban, ahol az egyik program terminál és a másik program ugyaninnen indul el. Nem szeretnénk ha például az $x := x + 1$ és $x := x + 2$ programok szekvenciája (ahol x egész típusú) az állapottér $\{x{:}5\}$ eleméhez az $< \{x{:}5\}, \{x{:}6\}, \{x{:}6\}, \{x{:}8\} >$ sorozatot rendelné. Ezért az egymás után véges sokszor ismétlődő csatlakozási pontokból egyet hagyjunk el.

Jelölés: *Véges hosszú α sorozat utolsó elemét jelölje $\tau(\alpha)$. Tehát ha α véges, $\tau(\alpha) = \alpha_{|\alpha|}$.*

Jelölés: *Legyen A tetszőleges állapottér. $\alpha \in \bar{A}^*$ és $\beta \in (\bar{A} \cup \{fail\})^{**}$ úgy hogy α és β nem üres sorozatok továbbá $\tau(\alpha) = \beta_1$. Ekkor $\alpha \otimes \beta$ jelölje az α és β sorozatok összefűzésében β első elemének elhagyásával kapott sorozatot.*
Általánosítsuk a jelölést $n \in \mathbb{N}^+$ darab vagy akár végtelen sok sorozat esetére. A sorozatok összefűzése (konkatenációja) után az egymás után véges sokszor ismétlődő csatlakozási pontokból egyet hagyjunk el.

Példa: *Legyen $A = \{1, 2, 3, 4\}$. Ekkor*

$$\otimes_4(<1,2,3,1>,<1,2,3,1>,<1,2,3,1>,<1,2,3,1>) =<1,2,3,1,2,3,1,2,3,1,2,3,1>$$

$$\otimes_4(<1>,<1>,<1>,<1,2,3,1>) =<1,1,1,2,3,1>$$

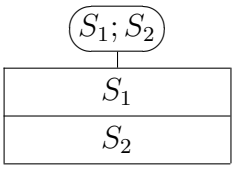
$$\otimes_\infty(<1>,<1>,<1>,<1>,\ldots) =<1,1,\ldots>$$

$$\otimes_\infty(<1,2,3,1>,<1,2,3,1>,<1,4>,<4>,<4>,<4>,\ldots) =<1,2,3,1,2,3,1,4,4,4,\ldots>$$

Definíció: *Legyen A közös alap-állapottere az S_1 és S_2 programoknak. Az $(S_1; S_2)$ relációt az S_1 és S_2 programok szekvenciájának nevezzük, ha*

$$\begin{aligned}(S_1;S_2)(a) =&\{\alpha \in \bar{A}^\infty \mid \alpha \in S_1(a)\} \cup \\ &\{\alpha \in (\bar{A} \cup \{fail\})^* \mid \alpha \in S_1(a) \wedge \alpha_{|\alpha|} = fail\} \cup \\ &\{\gamma \in (\bar{A} \cup \{fail\})^{**} \mid \gamma = \alpha \otimes \beta \wedge \alpha \in S_1(a) \wedge |\alpha| < \infty \wedge \alpha_{|\alpha|} \neq fail \wedge \beta \in S_2(\alpha_{|\alpha|})\}\end{aligned}$$

A szekvencia struktogramja:



Tétel: *Legyen A közös alap-állapottere az S_1 és S_2 programoknak. Az $(S_1; S_2)$ szekvencia program.*

Tétel: *Legyen A közös alap-állapottere az S_1 és S_2 programoknak és jelölje S az $(S_1; S_2)$ szekvenciát. Ekkor*

$$p(S) = p(S_2) \odot p(S_1)$$

2. Elágazás

Definíció: *Legyen A közös alap-állapottere az $S_1, \ldots S_n$ programoknak. Legyenek továbbá $\pi_1, \ldots \pi_n \in A \to \mathbb{L}$ logikai függvények. Ekkor az $IF \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ relációt az S_i programokból képzett π_i feltételek által meghatározott elágazásnak nevezzük és $(\pi_1{:}S_1, \ldots \pi_n{:}S_n)$-nel jelöljük, ha*

$$\forall a \in A : IF(a) = \omega_0(a) \cup \bigcup_{i=1}^{n} \omega_i(a)$$

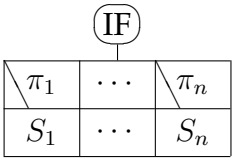
ahol $\forall i \in [1..n]$:

$$\omega_i(a) = \begin{cases} S_i(a), & \text{ha } a \in \mathcal{D}_{\pi_i} \wedge \pi_i(a) \\ \emptyset, & \text{ha } a \in \mathcal{D}_{\pi_i} \wedge \neg\pi_i(a) \\ \{< a, fail >\}, & \text{ha } a \notin \mathcal{D}_{\pi_i} \end{cases}$$

és

$$\omega_0(a) = \begin{cases} \{< a, fail >\}, & \text{ha } \forall i \in [1..n] : (a \in \mathcal{D}_{\pi_i} \wedge \neg\pi_i(a)) \\ \emptyset, & \text{különben} \end{cases}$$

Az elágazás struktogramja:



Az elágazást szokás még a következő módon is leírni:

if
$\pi_1 \to S_1 \ \square$
$\ldots$
$\pi_{n-1} \to S_{n-1} \ \square$
$\pi_n \to S_n$
fi

Tétel: *Legyen A közös alap-állapottere az $S_1, \ldots S_n$ programoknak. Legyenek továbbá $\pi_1, \ldots \pi_n \in A \to \mathbb{L}$ logikai függvények. Az $IF = (\pi_1{:}S_1, \ldots \pi_n{:}S_n)$ elágazás program.*

Tétel: *Legyen A közös alap-állapottere az $S_1, \ldots S_n$ programoknak. Legyenek továbbá $\pi_1, \ldots \pi_n \in A \to \mathbb{L}$ logikai függvények. $IF = (\pi_1{:}S_1, \ldots \pi_n{:}S_n)$. Ekkor*

$$\mathcal{D}_{p(IF)} = \{a \in A \mid a \in \bigcap_{i=1}^{n} \mathcal{D}_{\pi_i} \wedge a \in \bigcup_{i=1}^{n} \lceil \pi_i \rceil \wedge \forall i \in [1..n] : a \in \lceil \pi_i \rceil \implies a \in \mathcal{D}_{p(S_i)}\}$$

és

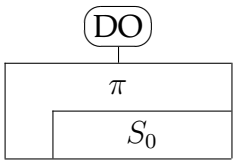
$$\forall a \in \mathcal{D}_{p(IF)} \colon p(IF)(a) = \bigcup_{i=1}^{n} p(S_i)|_{\lceil \pi_i \rceil}$$

3. Ciklus

Definíció: *Legyen $\pi \in A \to \mathbb{L}$ feltétel és $S_0 \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ program. A $DO \subseteq A \times (\bar{A} \cup \{fail\})^{**}$ relációt az S_0 programból π feltétellel képzett ciklusnak nevezzük és (π, S_0)-lal jelöljük, ha $\forall a \in A$:*

$$DO(a) = \begin{cases} (S_0; DO)(a) & \text{ha} \quad a \in \mathcal{D}_\pi \wedge \pi(a) \\ \{< a >\} & \text{ha} \quad a \in \mathcal{D}_\pi \wedge \neg\pi(a) \\ \{< a, fail >\} & \text{ha} \quad a \notin \mathcal{D}_\pi \end{cases}$$

A ciklus struktogramja:



A szakirodalomban a következő mód is elterjedt a ciklus leírására:

while π **do**
S_0
od

A szekvenciához és az elágazáshoz hasonló módon is definiálhatjuk a ciklust.

Definíció: $\forall a \in A$:

$$DO(a) = \begin{cases} \{< a, fail >\}, & \textit{ha } a \notin \mathcal{D}_\pi \\ \{< a >\}, & \textit{ha } a \in \mathcal{D}_\pi \wedge \neg\pi(a) \\ \{\alpha \in (\bar{A} \cup \{fail\})^{**} \mid \exists \alpha^1, \ldots, \alpha^n \in (\bar{A} \cup \{fail\})^{**} \colon \alpha = \otimes_n(\alpha^1, \ldots, \alpha^n) \wedge \\ \alpha^1 \in S_0(a) \wedge \forall i \in [1..n-1] \colon (\alpha^i \in \bar{A}^* \wedge \tau(\alpha^i) \in \lceil \pi \rceil \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i))) \wedge \\ ((\alpha^n \in \bar{A}^\infty \vee (\alpha^n \in (\bar{A} \cup \{fail\})^* \wedge \tau(\alpha^i) = fail) \vee \\ (\alpha^n \in \bar{A}^* \wedge \tau(\alpha^i) \in \mathcal{D}_\pi \wedge \tau(\alpha^i) \notin \lceil \pi \rceil)\} \\ \cup \\ \{\alpha \in \bar{A}^\infty \mid \exists \alpha^1, \alpha^2, \ldots \in \bar{A}^* \colon \alpha = \otimes_\infty(\alpha^1, \alpha^2, \ldots) \wedge \alpha^1 \in S_0(a) \wedge \forall i \in \mathbb{N}^+ \colon \\ (\alpha^i \in \bar{A}^* \wedge \tau(\alpha^i) \in \lceil \pi \rceil \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i)))\} \\ \cup \\ \{\alpha \in (\bar{A} \cup \{fail\})^* \mid \exists \alpha^1, \ldots, \alpha^n \in (\bar{A} \cup \{fail\})^{**} \colon \alpha = \otimes_n(\alpha^1, \ldots, \alpha^n) \wedge \\ \alpha^1 \in S_0(a) \wedge \forall i \in [1..n-2] \colon (\alpha^i \in \bar{A}^* \wedge \tau(\alpha^i) \in \lceil \pi \rceil \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i))) \\ \wedge (\alpha^{n-1} \in \bar{A}^* \wedge \tau(\alpha^{n-1}) \notin \mathcal{D}_\pi \wedge \alpha^n =< \tau(\alpha^{n-1}), fail >)\}, & \textit{ha } a \in \mathcal{D}_\pi \wedge \pi(a) \end{cases}$$

Első ránézésre a definíció kissé bonyolultnak tűnik. Amennyiben sorra vesszük hogy az állapottér egy a pontjához milyen sorozatokat rendelhet a ciklus, már nem is fogjuk bonyolultnak találni a definíciót!

- Ha a állapotban a ciklusfeltétel nem kiértékelhető, akkor a ciklus hibásan terminál.
- Ha a állapotban a ciklusfeltétel kiértékelhető de nem teljesül a-ra, akkor a ciklus semmit nem csinál, végrehajtása befejeződik az a állapotban.
- A ciklusmagot véges sokszor elvégezzük egymás után, úgy hogy a ciklusmag utolsó végrehajtásához tartozó sorozat
  - végtelen; vagy
  - véges és a $fail$ állapotban végződik; vagy
  - véges és utolsó elemében kiértékelhető a π feltétel, de az $hamis$.
- A ciklusmagot végtelen sokszor elvégezzük egymás után, mert egy végrehajtás után mindig olyan állapotba jutunk ahol a ciklusfeltétel teljesül.
- Az utolsó lehetőség az, hogy a ciklusmagot azért véges sokszor (de legalább egyszer) végezzük el egymás után, mert utoljára egy olyan állapotba jutunk ahol a ciklusfeltétel nem kiértékelhető.

Programozáselmélet - Levezetési szabályok

Készítette: Borsi Zsolt

Tétel: *A szekvencia levezetési szabálya*
Legyen A közös alap-állapotterű S_1 és S_2 programok szekvenciája $S = (S_1; S_2)$. Legyenek Q, Q' és R logikai függvények A-n. Ha

1. $Q \implies lf(S_1, Q')$ *és*
2. $Q' \implies lf(S_2, R)$

akkor $Q \implies lf(S, R)$.

Tétel: *Az elágazás levezetési szabálya*
Legyen $IF = (\pi_1{:}S_1, \ldots \pi_n{:}S_n)$ a közös A alap-állapotterű S_i programokból képzett, A feletti π_i logikai függvényekkel meghatározott elágazás. Legyenek továbbá Q és R logikai függvények A-n. Ha

1. $Q \implies \bigwedge_{i=1}^{n} (\pi_i \vee \neg\pi_i)$ *és*
2. $Q \implies \bigvee_{i=1}^{n} \pi_i$ *és*
3. $\forall i \in [1..n] : Q \wedge \pi_i \implies lf(S_i, R)$

akkor $Q \implies lf(IF, R)$.

Tétel: *A ciklus levezetési szabálya*
Legyen $DO = (\pi, S_0)$ az A alap-állapottér feletti S_0 programból és a $\pi \in A \to \mathbb{L}$ feltétellel képzett ciklus. Továbbá legyenek P, Q és R logikai függvények A-n és $t\colon A \to \mathbb{Z}$ függvény adottak. Ha

1. $Q \implies P$ *és*
2. $P \wedge \neg\pi \implies R$ *és*
3. $P \implies \pi \vee \neg\pi$ *és*
4. $P \wedge \pi \implies t > 0$ *és*
5. $P \wedge \pi \implies lf(S_0, P)$
6. $P \wedge \pi \wedge t = t_0 \implies lf(S_0, t < t_0)$ *bármely t_0 egész számra*

akkor $Q \implies lf(DO, R)$.

A levezetési szabályban szereplő P állítást a ciklus invariáns tulajdonságának, a t függvényt terminálófüggvénynek nevezzük.
Az utolsó két pont összevonható:
$5-6.$ $P \wedge \pi \wedge t = t_0 \implies lf(S_0, P \wedge t < t_0)$ bármely t_0 egész számra

Programozáselmélet - Párhuzamos programok

Készítette: Borsi Zsolt

Továbbra is szekvenciális programokkal foglalkozunk, de bevezetünk három új programkonstrukciót. Ezek definícióját úgy adjuk meg, hogy a velük képzett relációk megfeleljenek a program korábban kimondott definíciónak. Azaz, a három új programkonstrukció segítségével további programokat tudunk előállítani. Mivel célunk továbbra is az, hogy belássuk programok helyességét (tehát hogy igazoljuk hogy adott program megold egy adott feladatot), az új programkonstrukcióknak is megadjuk a levezetési szabályait.

1. Multiprogramozási modell

Három új programkonstrukció egyike a párhuzamos blokk, ami lehetővé teszi párhuzamosan végrehajtott programok reprezentálását. Egy párhuzamos program végrehajtása alatt a párhuzamos blokkot alkotó programok elemi utasításainak valamilyen ütemezés szerinti sorbarendezését és egy processzoron (magon) való egymás utáni végrehajtását értjük. A multiprogramozási modellben az elemi programok végrehajtása (így az értékadás is) atomi módon történik, mint ahogy a logikai feltételek kiértékelése sem szakítható meg.

2. Új programkonstrukciók

- Atomi utasítás: $[S]$
  Megszakítás nélkül hajtjuk végre. S nem tartalmazhat ciklust vagy várakoztató utasítást.

- Várakoztató utasítás: $\mathbf{await}\ B\ \mathbf{then}\ S\ \mathbf{ta}$
  Amennyiben a β őrfeltétel igaz, az S program atomi utasításként a β kiértékelésével egyszerre hajtódik végre. S nem tartalmazhat ciklust vagy várakoztató utasítást. Ha β kiértékelhető de hamis, az utasítást tartalmazó folyamat blokkolttá válik; a végrehajtás nem tud továbblépni a várakoztató utasításon, amíg egy másik folyamat nem „segít" és változtatja meg az állapotot úgy hogy β teljesüljön.

- Párhuzamos blokk: $\mathbf{parbegin}\ S_1 \| \ldots \| S_n\ \mathbf{parend}$
  Befejeződik, ha minden komponensprogramja terminál. Végrehajtása valamely ágnak kiválasztását és az ott lévő komponens első utasításának, illetve a kapott maradék programnak egymás utáni végrehajtását jelenti.

Definíció: *Legyen A tetszőleges állapottér, S program az A állapottér felett.*
$\forall a \in A : [S](a) = S(a)$

Definíció: *Legyen A közös állapottere az S programnak és a β logikai függvénynek (őrfeltételnek).*

$$\forall a \in A : (\textbf{await } \beta \textbf{ then } S \textbf{ ta})(a) \ = \begin{cases} [S](a) & \text{, ha } a \in \mathcal{D}_\beta \wedge \beta(a) \\ (\textbf{skip}; \textbf{await } \beta \textbf{ then } S \textbf{ ta})(a) & \text{, ha } a \in \mathcal{D}_\beta \wedge \neg\beta(a) \\ \{< a, \textbf{fail} >\} & \text{, ha } a \notin \mathcal{D}_\beta \end{cases}$$

Amikor az n komponensből álló párhuzamos blokkot végrehajtjuk, akkor egy adott állapotból indulva bármely S_i komponenssel ($i \in [1..n]$) megkezdődhet a párhuzamos program végrehajtása. Amennyiben az S_i programot atomiként kell végrehajtani (mert vagy elemi program, vagy a [ ] atomi művelet programkonstrukcióval nem megszakíthatóvá lett téve), akkor annak végrehajtása után a **parbegin** $S_1 \| \ldots \| S_{i-1} \| S_{i+1} \| \ldots \| S_n$ **parend** párhuzamos blokkot kell elvégezni.
Amennyiben S_i végrehajtása megszakítható, akkor S_i program felbontható egy u atomi utasításra és egy T programra. Vagyis S_i felfogható az $u;T$ szekvenciaként, ahol u az S_i első (nem megszakítható) utasítása, T pedig az S_i maradék része. Ekkor a párhuzamos blokk egy végrehajtását kapjuk az u utasítás majd azt követően a
parbegin $S_1 \| \ldots \| S_{i-1} \| T \| S_{i+1} \| \ldots \| S_n$ **parend** párhuzamos blokkot elvégezve.
Mivel a párhuzamos blokk végrehajtását bármely komponenst kiválasztva elkezdhetjük, a párhuzamos programok szükségképpen nem-determinisztikusak. Az összes végrehajtási sorozatot megkapjuk a párhuzamos blokk elvégzését az elsőként az S_i komponens aktiválásával kapott végrehajtási sorozatok összességeként.

Definíció: *Legyen A közös alap-állapottere az $S_1 \ldots S_n$ programoknak.*

$$\forall a \in A : (\textbf{parbegin } S_1 \parallel \cdots \parallel S_i \parallel \cdots \parallel S_n \textbf{ parend})(a) \ = \bigcup_{i=1}^{n} B_i(a)$$

ahol

$$B_i(a) = \begin{cases} (S_i; \textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend})(a) \\ \quad \textit{ha } S_i \textit{ nem megszakítható az } a \textit{ állapotból indulva} \\ (u_i; \textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel T_i \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend})(a) \\ \quad \textit{ha } S_i(a) = (u_i; T_i)(a) \textit{ ahol } u_i \textit{ nem megszakítható az } a \textit{ állapotból indulva} \end{cases}$$

Lásd a fejezet végén lévő kiegészítés részt, hogy mit értünk a különböző programszerkezetek első utasításaként egy adott a állapotból indulva.

3. Az új programkonstrukciók levezetési szabályai

Tétel: *Atomi művelet levezetési szabálya*
Ha
$Q \implies lf(S, R)$ *akkor* $Q \implies lf([S], R)$

Tétel: *Várakozó utasítás levezetési szabálya*
Ha

1. $Q \implies \beta \vee \neg\beta$
2. $Q \wedge \beta \implies lf(S, R)$

akkor $Q \implies lf(\textbf{await } \beta \textbf{ then } S \textbf{ ta}, R)$

Tétel: *Párhuzamos blokk levezetési szabálya*
Legyenek $Q, Q_1, \ldots Q_n$ *és* $R, R_1, \ldots R_n$ *logikai függvények. Ha*

1. $Q \implies Q_1 \wedge \cdots \wedge Q_n$ *és*
2. $R_1 \wedge \cdots \wedge R_n \implies R$ *és*
3. $\forall i \in [1..n] : Q_i \implies lf(S_i, R_i)$ *és*
4. *a* $Q_1 \implies lf(S_1, R_1), \ldots, Q_n \implies lf(S_n, R_n)$ *teljes helyességi formulák interferenciamentesek és*
5. *a párhuzamos blokk holtpontmentes*

akkor $Q \implies lf(\textbf{parbegin } S_1 \| \ldots \| S_n \textbf{ parend}, R)$

A párhuzamos blokk levezetési szabályának első három feltétele szemléletesen azt fejezi ki, hogy

- ha Q előfeltétel teljesül, akkor a párhuzamos blokk bármely S_i komponensprogramjának végrehajtása elkezdhető, mert teljesül a Q_i előfeltétele (ezt nevezhetjük „belépési feltételnek”).
- amikor a párhuzamos blokk terminál (az összes S_i komponens befejeződött, elérte utófeltételét) akkor jó helyen terminált: ott ahol R utófeltétel igaz (ezt nevezhetjük „kilépési feltételnek”).
- minden S_i komponens önmagában teljesen helyes a Q_i előfeltételével és R_i utófeltételével adott feladatra nézve.

4. Interferencia-mentesség

Ezzel a fogalommal azt akarjuk kifejezni, hogy ha valamit már beláttunk az S_i komponens esetén, az a bizonyítás nem veszti érvényét egy S_i komponenssel párhuzamosan végrehajtott S_j komponensben lévő u utasítás elvégzésével. Például, ne legyen az hogy miután már beláttuk hogy S_i egy ciklusának magjában a terminálló függvény értéke csökken, a bizonyításunkat tönkreteszi u végrehajtása, azáltal hogy meg tudja növelni az adott ciklus terminálló függvényének értékét. Különben érvényét vesztené az a bizonyításunk, melyben beláttuk hogy a ciklus terminál.
Mivel értéket megváltoztatni, és így egy logikai állítást hamissá tenni csak az értékadás tud, így u utasításként azon utasításokat kell kell figyelembe venni, amik vagy értékadások vagy atomi módon végrehajtott programként értékadást tartalmaznak. Nevezzük őket kritikus utasításnak!

Jelölés: *Teljes helyességi formula Ha valamilyen S program és Q, R logikai függvények esetén teljesül hogy $Q \implies lf(S,R)$, akkor nevezzük az $Q \implies lf(S,R)$ alakot az S egy teljes helyességi formulájának! Mivel az interferencia-mentesség igazolásához azt akarjuk megmutatni hogy egy bizonyítás nem veszti érvényét, S program helyett egy hozzá tartozó $Q \implies lf(S,R)$ teljes helyességi formulát veszünk figyelembe (tehát azt amellyel bizonyítottuk hogy S program a Q feltételnek eleget tevő állapotokból indulva biztos hogy helyesen terminál, és a végállapotokban teljesül R).*

Definíció: *A $Q_1 \implies lf(S_1,R_1), \ldots, Q_n \implies lf(S_n,R_n)$ teljes helyességi formulák interferencia-mentesek, ha bármely i és j esetén ($i,j \in [1..n]$ és $i \neq j$) az S_i komponens egyik kritikus utasítása sem interferál a $Q_j \implies lf(S_j,R_j)$ teljes helyességi formulával.*

Definíció: *Azt mondjuk hogy az S_i komponens u kritikus utasítása (aminek előfeltételét jelölje pre_u) nem interferál az $Q_j \implies lf(S_j,R_j)$ teljes helyességi formulával, ha*

- $pre_u \wedge R_j \implies lf(u,R_j)$
  Azaz, u végrehajtása nem rontja el S_j utófeltételét: ha R_j igaz volt a végrehajtás előtt akkor utána is igaz lesz.
- $pre_u \wedge pre_s \implies lf(u,pre_s)$ *bármely S_j-beli s utasítás (aminek előfeltételét jelölje pre_s) esetén*
  Azaz u végrehajtása igaznak tartja meg S_j bármely s utasításának előfeltételét: ha s elvégezhető volt u végrehajtása előtt akkor utána is az lesz.
- $pre_u \wedge t = t_0 \implies lf(u, t \leq t_0)$, $\forall t_0 \in \mathbb{Z}$ *esetén ahol t egy S_j-beli ciklus terminálló függvénye*

Azaz u végrehajtása S_j bármely ciklusának t terminálófüggvényének értékét nem növeli meg.

5. Holtpont

Legyen $A = (a{:}\mathbb{Z}, b{:}\mathbb{Z})$ és tekintsük az A alap-állapottér feletti következő programot:

```
a:=0; b:=0;
parbegin
  while IGAZ do
    α0: a:=a+1;
    α1: await b ≠ 0 then
          α2:   a:=a+3
        ta;
  od
||
  while IGAZ do
    β0: a:=2*a;
    β1: await a ≠ 1 then
          β2:   b:=b+1
        ta;
  od
parend
```

Ez a program az állapottér bármely állapotából indulva, sorrendben a β_0 majd α_0 címkével jelzett utasításokat végrehajtva, az $\{a{:}1, b{:}0\}$ állapotba kerülve holtpontba jut.

Definíció: *Holtpont*

- *Egy várakoztató utasítás egy adott állapotban blokkolt, ha az őrfeltétele hamis az állapotban.*
- *Egy szekvenciális program blokkolt, ha egy általa közvetlenül tartalmazott várakoztató utasítás blokkolt.*
- *Egy párhuzamos blokk holtpontban van, ha minden még be nem fejeződött komponense blokkolttá vált.*

6. A holtpontmentesség egy elégséges feltétele

Elégséges feltételt szeretnénk arra adni, hogy a programunk garantáltan nem juthat holtpontba, nincs olyan végrehajtása amely során holtponthelyzet állna fent. Hangsúlyozzuk, hogy egy általános párhuzamos program alatt olyan programot értünk ami az eddig megengedett programkonstrukciókból épül fel (nagy valószínűséggel szekvencia) de tartalmazza az új programkonstrukciók valamelyikét is.
Tekintsünk egy ilyen programot. A holtpontmentesség szempontjából számunkra a várakoztató utasítások és a párhuzamos blokkok az érdekesek. Tegyük fel hogy az általános programunkban m olyan várakoztató utasítás szerepel ami nincs egy párhuzamos blokkon belül, a párhuzamos blokkok száma pedig legyen n.
A k-adik párhuzamos blokkot jelöljük így (tehát n_k komponense van a $k$adik párhuzamos blokknak):
T_k: **parbegin** $S_1^k \| \ldots \| S_{n_k}^k$ **parend**, ahol a komponensek akár újabb várakoztató utasításokat tartalmazhatnak.

S :

$\vdots$

A_1 : **await** β_1 then S_1 ta;

$\vdots$

A_i : **await** β_i then S_i ta;

$\vdots$

T_1 : **parbegin** $S_1^1 \| \ldots \| S_{n_1}^1$ **parend**

$\vdots$

A_j : **await** β_j then S_j ta;

$\vdots$

T_n : **parbegin** $S_1^n \| \ldots \| S_{n_n}^n$ **parend**

$\vdots$

A_m : **await** β_m then S_m ta;

$\vdots$

Sorra vesszük a lehetséges holtpont helyzeteket:

- Az S szekvenciális folyamat blokkolt, ha valamely (nem párhuzamos blokkon belüli) várakoztató utasításánál várakozunk. Ezt úgy írjuk le hogy az A_j ($j \in [1..m]$) várakoztató utasítást készek vagyunk elvégezni (a $pre(A_j)$ előfeltétele teljesül) de a β_j őrfeltétel $hamis$.

- A szekvenciális S program valamely párhuzamos blokkja blokkolt. Tegyük fel hogy a T_k párhuzamos blokkról van szó. Azaz T_k minden S_i^k ($i \in [1..n_k]$) komponensprogramja terminált vagy blokkolt, de legalább az egyik blokkolt.

Definiáljuk $D(S)$-et és $D1(T_k)$-et a következőképpen: (A $D1$ alkalmazásakor tudjuk hogy az argumentuma egy párhuzamos blokk.)

- $$D(S) = \left[\bigvee_{j=1}^{m}(pre(A_j) \wedge \neg\beta_j)\right] \vee \left[\bigvee_{k=1}^{n} D1(T_k)\right]$$

  A formula azt fejezi ki hogy van benne (de nem párhuzamos blokkon belüli, hanem közvetlenül a „főprogramban") blokkolt várakoztató utasítás, vagy valamely párhuzamos blokkja blokkolt.

- $$D1(T_k) = \left[\bigwedge_{j=1}^{n_k}(post(S_i^k) \vee D(S_i^k))\right] \wedge \left[\bigvee_{i=1}^{n_k} D(S_i^k)\right]$$

  A T_k párhuzamos blokk blokkolt, ha minden S_i^k komponense blokkolt vagy véget ért, miközben van legalább egy komponense ami blokkolt.

Tétel: *Amennyiben $D(S)$ azonosan $HAMIS$, nem lehet az S programnak olyan végrehajtása hogy holtpont előfordulhatna; S holtpontmentes.*

7. Kiegészítés: S program felbontása első utasításra és az azt követő maradék programra

Itt megadjuk hogy írható fel egy a kezdőállapotot tekintve az S program (ahol S nem elemi program és nem is atomiként végrehajtható) az $u;T$ szekvenciaként, ahol u az S nem megszakítható első utasítása, T pedig a maradék program.

- Ha $S =$ **await** β **then** S_0 **ta** és $a \in \mathcal{D}_\beta \wedge \neg\beta(a)$, akkor u a **SKIP** program míg T az **await** β **then** S_0 **ta** várakoztató utasítás.

- Ha $S = (S_1; S_2)$ és S_1 végrehajtása nem megszakítható a-ból indulva, akkor u az S_1 és T az S_2.

- Ha $S = (S_1; S_2)$ és S_1 végrehajtása megszakítható a-ból indulva, akkor u az S_1 első (atomi módon végrehajtható) utasítása, míg T a $(T_1; S_2)$ szekvencia (ahol T_1 az S_1 maradék része).

- Ha $S = \textbf{if } \pi_1 \to S_1 \,\square \ldots \square\, \pi_n \to S_n \textbf{ fi}$ és van olyan π_i feltétel ami nem értelmezett a állapotban vagy mindegyik feltétel hamis a-ban ($\exists i \in [1..n] : a \notin \mathcal{D}_{\pi_i} \vee \forall i \in [1..n] : a \in \mathcal{D}_{\pi_i} \wedge \neg\pi_i(a)$), akkor u az **ABORT** program és T szintén az **ABORT**.

- Ha $S = \textbf{if } \pi_1 \to S_1 \,\square \ldots \square\, \pi_n \to S_n \textbf{ fi}$ és az a állapotban mindegyik feltétel értelmezett és legalább egy közülük teljesül ($\forall i \in [1..n] : a \in \mathcal{D}_{\pi_i} \wedge \exists i \in [1..n] : \pi_i(a)$), akkor u a **SKIP** program míg T az S_i (feltéve hogy azt a végrehajtást nézzük ami az a állapotból indulva az i-edik ág kiválasztásával történik).

- Ha $S = \textbf{while } \pi \textbf{ do } S_0 \textbf{ od}$ és az a állapotban a ciklusfeltétel nem értelmezett ($a \notin \mathcal{D}_\pi$), akkor u és T is az **ABORT** program.

- Ha $S = \textbf{while } \pi \textbf{ do } S_0 \textbf{ od}$ és a állapotban a ciklusfeltétel hamis ($a \in \mathcal{D}_\pi \wedge \neg\pi(a)$), akkor u és T is az **ABORT** program.

- Ha $S = \textbf{while } \pi \textbf{ do } S_0 \textbf{ od}$ és a ciklusfeltétel teljesül az a állapotban ($a \in \mathcal{D}_\pi \wedge \pi(a)$), akkor u a **SKIP** program és T az $(S_0; \textbf{while } \pi \textbf{ do } S_0 \textbf{ od})$ szekvencia.

- Ha $S = \textbf{parbegin } S_1 \parallel \cdots \parallel S_i \parallel \cdots \parallel S_n \textbf{ parend}$ és az a állapotból indulva nem megszakíthatóan végrehajtható S_i elvégzésével kezdjük meg a párhuzamos blokk végrehajtását, akkor u az S_i és T a $\textbf{parbegin } S_1 \parallel \cdots \parallel S_{i-1} \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend}$ párhuzamos blokk.

- Ha $S = \textbf{parbegin } S_1 \parallel \cdots \parallel S_i \parallel \cdots \parallel S_n \textbf{ parend}$ és az a állapotból indulva a megszakítható S_i elvégzésével kezdjük meg a párhuzamos blokk végrehajtását, akkor u az S_i első (atomi módon végrehajtható) utasítása, és T az $\textbf{parbegin } S_1 \parallel \ldots \parallel S_{i-1} \parallel T_i \parallel S_{i+1} \parallel \cdots \parallel S_n \textbf{ parend}$ párhuzamos blokk, ahol T_i az S_i maradék programja.