Programhelyesség bizonyítás

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

http://www.mit.bme.hu/~majzik/

Motiváció

- Kritikus algoritmusok helyességének bizonyítása
  - Korlátozott feladatok: biztonságkritikus funkciók, hozzáférésvédelem, protokoll mag, …
  - Bizonyítás alapja:
    - Részletes terv: Pszeudo-nyelven leírt algoritmus (a továbbiakban mint „program”)
    - Valós programnyelv: Csak részleges támogatás van
- Tételbizonyító rendszerek használata
  - Bizonyítandó tulajdonság mint bizonyítandó „tétel” adott
  - Klasszikus elő- és utófeltételek a tipikusak (contract)
- Kihívások:
  - (Pszeudo) programból hogyan lesz bizonyítandó tétel?
  - Milyen stratégiák használhatók a bizonyításhoz?

Tételbizonyító rendszerek

- Felépítés:
  - Leíró nyelv, pl.:
    - Elsőrendű logika
    - Típusokkal kiegészített logikák
    - Magasabbrendű logika, …
  - Problémaleírás mint logikai axiómák
  - Következtetési szabályok (levezetési szabályok)
    - Indukció, dedukció, unifikáció, …
- Komponensek:
  - Algoritmikus: Egy-egy következtetési szabály alkalmazása
  - Kereső: Stratégia/taktika a szabályok kiválasztására
    - Cél-vezérelt (visszafelé történő) keresés
    - Mélységi vagy szélességi keresés
    - Interaktív (felhasználói segítséggel)

Tételbizonyító rendszerek alkalmazása

- A tételbizonyítás komplex feladat
  - n operátort tartalmazó tétel,
    $O(2^n)$ hosszú bizonyítási szekvencia,
    $O(2^{2^n})$ időigény a bizonyítás megtalálásához… (worst case, kereséssel)
  - Fontos a bizonyítási stratégia meghatározása!
- További használati esetek
  - Kézi bizonyítás automatikus ellenőrzése (proof checking)
  - Kézi bizonyítás segítése (interaktív szabályalkalmazás)
- Tételbizonyítás szerepe
  - Adat-intenzív alkalmazások tulajdonságainak bizonyítása
  - Paraméter-függőségek kezelése (pl. protokoll résztvevők száma)
    - Indukció használható
  - Együttes használat a modell ellenőrzéssel
    - Kis paraméter értékre (kiindulási eset): modell ellenőrzés
    - Tulajdonság megtartásának bizonyítása indukcióval
- Népszerű eszközök
  - HOL, PVS, ACL2, …

Tulajdonságok

D dedukciós rendszer, c levezetendő tétel (bizonyítandó spec.)

- Szemantikus helyesség (soundness):
  - Ami levezethető D-ben az igaz
  - Szükséges a használhatósághoz
  - Formálisan: $\forall c$: ha $\vdash_D c$ (levezethető) akkor $\models c$ (teljesül, igaz)
- Szemantikus teljesség (completeness):
  - Ami igaz, az levezethető D-ben
  - Hasznos tulajdonság, de nem szükséges
  - Formálisan: $\forall c$: ha $\models c$ akkor $\vdash_D c$
- Totális helyesség és teljesség:
  - Minden interpretációra helyes és teljes
- Konzisztens dedukciós rendszer:
  - Nem lehet egy tételt és az ellenkezőjét is bizonyítani

A probléma leképzése

Források:

- Dedukciós rendszer axiómáihoz:
  - Program utasítások (értékadás)
  - Program domén axiómái
- Dedukciós rendszer levezetési szabályaihoz:
  - Program nyelv szemantika
  - Program domén szemantika
- Bizonyítandó tételhez:
  - Program specifikáció

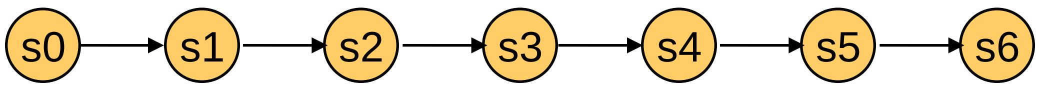
Program és specifikáció kapcsolata

Egy P program és egy Φ specifikáció (követelmény) esetén:

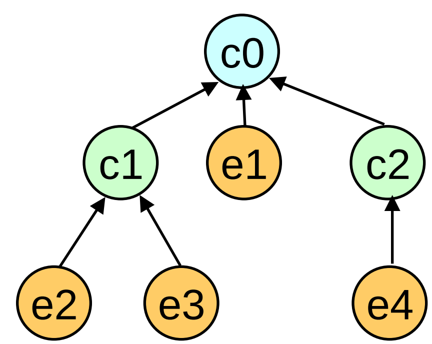
- Szintézis feladat:
  - Φ alapján P konstruálása
- Analízis feladat:
  - P alapján Φ levezetése
- Verifikáció (program helyességbizonyítás):
  - $P \models \Phi$ eldöntése
- Optimalizáció:
  - $P \models \Phi$ alapján olyan P' konstruálása,
    hogy $P' \models \Phi$, ugyanakkor
    P' jobb adott szempontok szerint (kisebb, gyorsabb…)
- Javítás:
  - $P \not\models \Phi$ alapján olyan P' konstruálása,
    hogy $P' \models \Phi$ legyen

Bizonyításhoz leginkább használt módszer: Indukció

- Számítási indukció: Műveleti szemantika esetén
  - Állapotokra:
    Ha a kiindulási állapot tulajdonságát ismerjük, az átmenetek követésével levezetjük a végállapot tulajdonságát



- Strukturális indukció: Axiomatikus szemantikához
  - Szintaktikai konstrukciókra:
    Ha az elemek tulajdonságait ismerjük, a rajtuk alkalmazott szintaktikai konstrukció alapján levezetjük a kompozit struktúra (így majd a teljes program) tulajdonságait



Célkitűzéseink

- Bizonyítási vázak rögzítése program helyesség bizonyításhoz
  - Bizonyítási stratégia megadása
  - Nincs teljesen automatikus, garantált módszer
- Nem konkrét programozási nyelvhez kötve
  - Pszeudo-nyelv (algoritmus leírás)
    - Továbbiakban mint „programozási nyelv” szerepel
  - Pl. saját, domén-specifikus nyelvhez is alkalmazható
- Megkötések a bizonyíthatósághoz
  - Programozási nyelv: Formális szemantikával rendelkezik (műveleti vagy axiomatikus szemantika)
  - Specifikációs nyelv: Elsőrendű kijelentéslogika

Programozási nyelv műveleti szemantikával

- Konfiguráció (állapot): C
  - Látható állapot σ (a program vizsgált kimenetében szerepel)
    - $\sigma[x]$ egy x változó értéke egy σ látható állapotban
    - $\sigma[\underline{x}]$ az $\underline{x}$ változó-vektor értéke egy σ látható állapotban
  - Rejtett állapot (helyesség szempontjából érdektelen)
  - Szintaktikus folytatás λ: A további számításokat definiálja
    - „Programszámláló” helyett
    - A továbbiakban végrehajtandó forrásszövegként képzelhető el
- Átmenet reláció a konfigurációk között: $\rightarrow$
  - $\pi(P, \sigma_0)$ egy P program számítása σ_0 kezdőállapotból
    - $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \ldots$ maximális szekvencia (végállapotig/végtelen)
    - $\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \ldots$ látható állapot szekvencia
  - $val(\pi(P, \sigma)) = \sigma_n$ véges esetben a végállapot jelölése
- I domén: A számítások itt értelmezettek

Specifikációs nyelv

- Korlátozások a programra:
  - Determinisztikus
  - Nem folytonos működésű
  - Érték- vagy állapot-transzformációt valósít meg
- Tulajdonság megadása: Predikátumokkal
  - Előfeltétel: $p(\underline{x})$ - a megengedhető kezdeti állapotokra
    - $\underline{x}$ a látható állapot változói
    - $\sigma_0 \models p(\underline{x})$ jelentése: kezdőállapotban igaz $p(\underline{x})$
  - Utófeltétel: $q(\underline{x})$ - az elfogadható végállapotokra
    - true – minden befejeződő számítás kielégíti
    - false – egy szabályos végállapot sem elégíti ki
    - $val(\pi(P,\sigma_0)) \models q(\underline{x})$ jelentése: π számítás végállapotában igaz $q(\underline{x})$
- Specifikáció példák
  - Segédváltozók használata
  - Specifikációs változók használata

Példák specifikációkra

- x és y nagyság szerint rendezve a kimeneten:

  Előfeltétel $p(x,y) = true$, utófeltétel $q(x,y) = x \leq y$

  Tömörebben felírva $(true, x \leq y)$

- x legyen páros a kimeneten:

  $(true, even(x))$ ha a doménen értelmezett $even(x)$

  $(true, \exists y\colon x=2y)$ itt y kötött segédváltozó $q(x)$-ben

- Bemeneti x-et kétszerezze meg:

  $(x=X, x=2X)$ itt X egy specifikációs változó

- x/y poz. bennfoglalás eredménye q és maradéka r:

  $(x=X \wedge x>0 \wedge y=Y \wedge y>0, \quad X=qY+r \wedge 0<=r<Y)$

  ha meg kell őrizni x és y értékét:

  $(x=X \wedge x>0 \wedge y=Y \wedge y>0, \quad X=qY+r \wedge 0<=r<Y \wedge x=X \wedge y=Y)$

Program helyességi kritériumok (1)

- Részleges helyesség: Jelölése $\{p(\underline{x})\}\ P\ \{q(\underline{x})\}$

  Egy P program részlegesen helyes $p(\underline{x})$, $q(\underline{x})$ szerint, ha teljesül a következő:

  $\forall \pi(P,\sigma_0)$ és $\sigma_0 \models p(\underline{x})$ esetén
  ha π befejeződik, akkor $val(\pi(P,\sigma_0)) \models q(\underline{x})$

- Megjegyzések:
  - Az előfeltételt kielégítő állapotból induló számításokra: ha befejeződik, akkor az utófeltétel igaz a végállapotban
  - Nem mond semmit azokról a számításokról, amelyekre $\sigma_0 |\neq p(\underline{x})$
  - $\{true\}P\{true\}$ minden programra igaz
  - $\{true\}P\{false\}$ ha igaz, akkor nincs befejeződő számítás

Program helyességi kritériumok (2)

- Helyesség: Jelölése $\langle p(\underline{x})\rangle\ P\ \langle q(\underline{x})\rangle$

  Egy P program helyes $p(\underline{x})$, $q(\underline{x})$ szerint, ha teljesül a következő:

  $\forall \pi(P,\sigma_0)$ és $\sigma_0 \models p(\underline{x})$ esetén

  π befejeződik és $val(\pi(P,\sigma_0)) \models q(\underline{x})$

- Megjegyzések:
  - Az előfeltételt kielégítő állapotból induló számításokra: befejeződik és az utófeltétel igaz lesz a végállapotban
  - $\langle p(\underline{x})\rangle P \langle true\rangle$ csak befejeződést ír elő
  - Felírható:

    $\langle p(\underline{x})\rangle P \langle q(\underline{x})\rangle$ a.cs.a. $\{p(\underline{x})\}P\{q(\underline{x})\}$ és $\langle p(\underline{x})\rangle P \langle true\rangle$

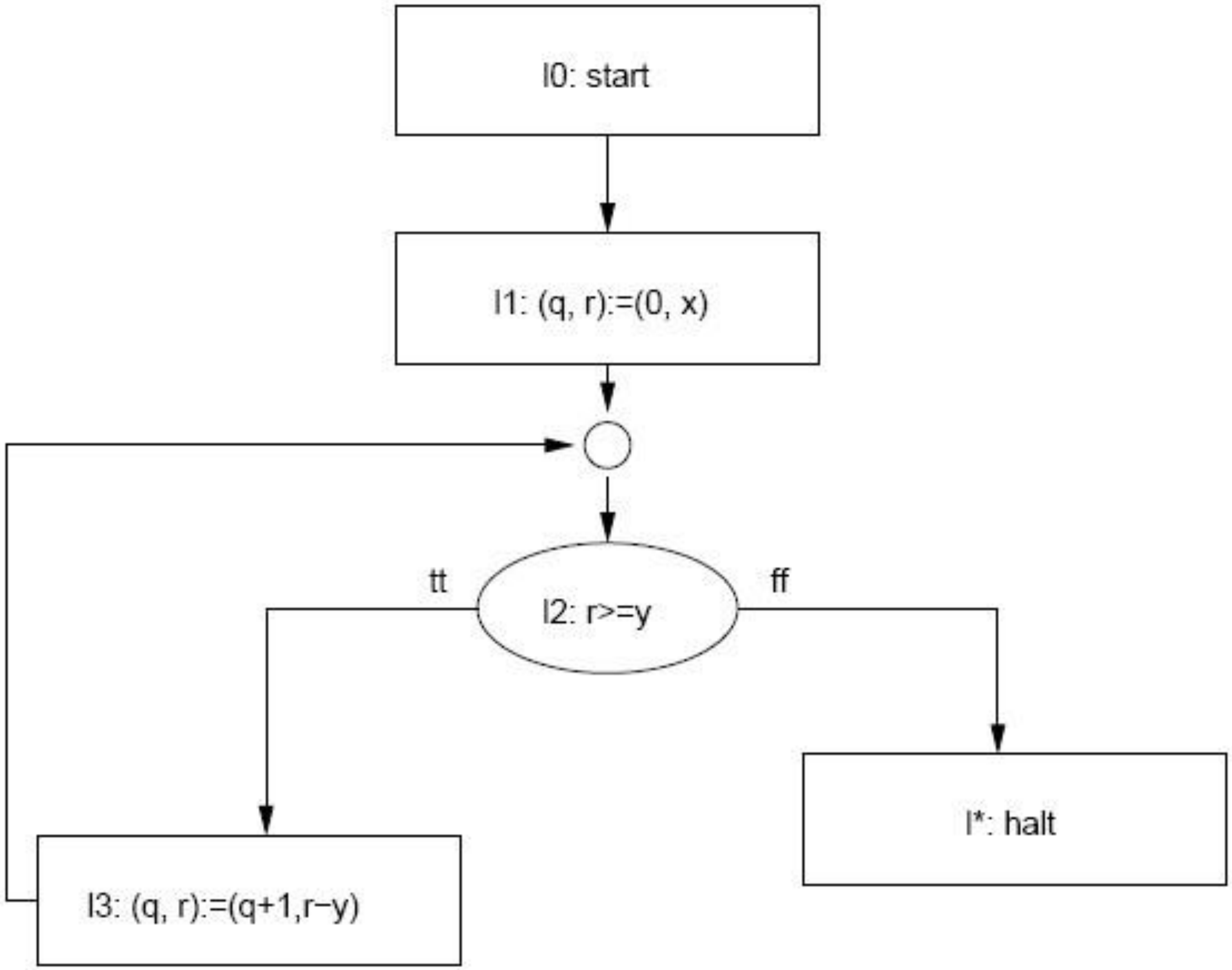
    azaz a program helyes, ha részlegesen helyes és befejeződik

Egyszerű determinisztikus programok

- PLF „flow language”:
  - *start*, $\underline{x}$:=$\underline{e}$, *B*($\underline{x}$), *halt* utasítások egyedi l címkékkel (l_0, l_*, l_i,…)
  - Assembly szinthez közeli pszeudo-nyelv
- Program szintaxis: PLF mint véges irányított gráf
  - succ(l), $succ^+$(l), $succ^-$(l) használható a következő csomóponthoz
  - Minden utasítás egy start $\rightarrow$ halt útvonalon
- Szemantika: C=(σ,λ) konfiguráció és $\rightarrow$ megadása: C(σ,λ) $\rightarrow$ C' (σ',λ') a.cs.a.
  - λ egy *start*: λ'=succ(λ), σ'=σ
  - λ egy $\underline{x}$:=$\underline{e}$ utasítás: λ'=succ(λ), σ'=σ[$\underline{e}$/$\underline{x}$]
    - itt [$\underline{e}$/$\underline{x}$] jelöli, hogy $\underline{x}$ helyébe $\underline{e}$ helyettesítése történik
  - λ egy *B*($\underline{x}$) elágazás:
    - σ|=B($\underline{x}$) esetén: λ'=$succ^+$(λ), σ'=σ
    - σ|$\neq$B($\underline{x}$) esetén: λ'=$succ^-$(λ), σ'=σ

Példa: Maradékos osztás egész számokra

x/y számítása, q hányados, r maradék



Részleges helyesség ciklusmentes esetben (1)

- Ötlet: Számítási indukció {p}P{q} esetén
- Egy véges számításhoz tartozó u útvonal jellemzői:

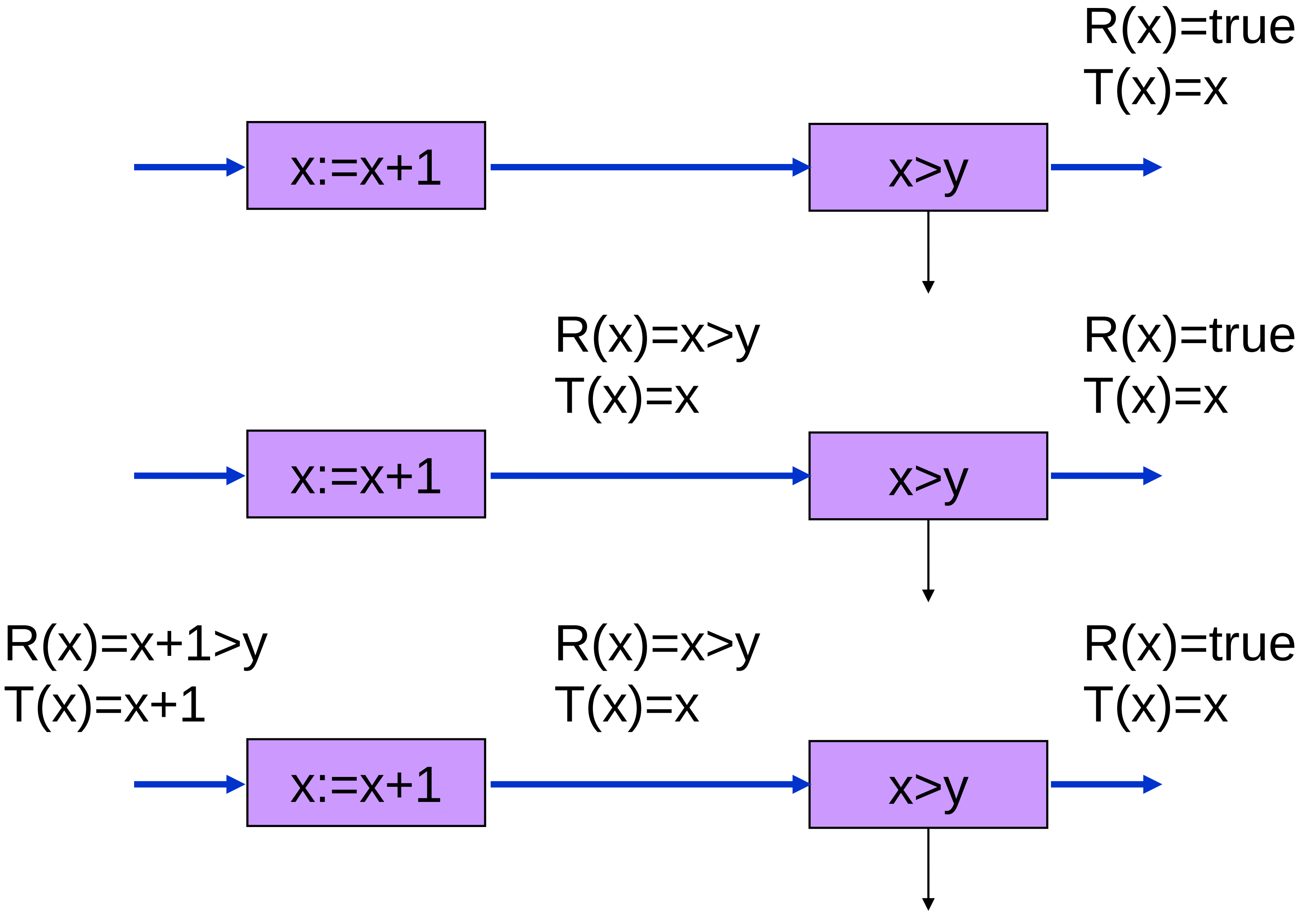
  $u = l_0 \rightarrow l_1 \rightarrow l_2 \rightarrow \ldots \rightarrow l_k$

  - **Elérhetőségi (bejárási) feltétel**: $R_u(\underline{x})$ predikátum u útvonalhoz
    - Ha fennáll l_0 esetén, akkor éppen az u útvonalat járja be a program
  - **Állapot transzformáció**: $T_u(\underline{x})$ egy állapotvektort ad u útvonalhoz
    - $\underline{x}$ állapotvektorból indulva az u útvonal bejárása utáni állapot
    - $\underline{x} := T_u(\underline{x})$ a végállapotot eredményező állapot transzformáció
- Jelölések:
  - $l_m \rightarrow \ldots \rightarrow l_k$ szuffixe az útvonalnak az m indextől
  - $R_u^m(\underline{x})$ és $T_u^m(\underline{x})$ erre a szuffixre vonatkoznak
- Ismert a végállapotra (utolsó szuffixre):
  - $R_u^k(\underline{x})=\text{true}$ - hiszen már a végállapotban vagyunk
  - $T_u^k(\underline{x})=\underline{x}$ - hiszen nincs további transzformáció

Részleges helyesség ciklusmentes esetben (2)

- Visszalépéses indukció:
  - Feltéve: Ismert $R_u^{m+1}(\underline{x})$ és $T_u^{m+1}(\underline{x})$ egy szuffixre
  - Lépés: Az I_m utasítása alapján $R_u^{m}(\underline{x})$ és $T_u^{m}(\underline{x})$ számítás
    - *$\underline{x}$:=$\underline{e}$* hozzárendelés:
      $R_u^m(\underline{x}) = R_u^{m+1}(\underline{x})[\underline{e}/\underline{x}]$, $T_u^m(\underline{x}) = T_u^{m+1}(\underline{x})[\underline{e}/\underline{x}]$
    - *B($\underline{x}$)* feltétel pozitív ága:
      $R_u^m(\underline{x}) = R_u^{m+1}(\underline{x}) \wedge B(x)$, $T_u^m(\underline{x}) = T_u^{m+1}(\underline{x})$
    - *B($\underline{x}$)* feltétel negatív ága:
      $R_u^m(\underline{x}) = R_u^{m+1}(\underline{x}) \wedge \neg B(x)$, $T_u^m(\underline{x}) = T_u^{m+1}(\underline{x})$
    - *start*:
      $R_u(\underline{x}) = R_u^0(\underline{x})$, $T_u(\underline{x}) = T_u^0(\underline{x})$
  - Így a végállapotban ismert $R_u^k(\underline{x})$=true és $T_u^k(\underline{x})=\underline{x}$ alapján a kezdőállapotra $R_u(\underline{x})$ és $T_u(\underline{x})$ *levezethető*

Részleges helyesség ciklusmentes esetben (példa)



Részleges helyesség ciklusmentes esetben (3)

- Részleges helyesség megmutatása:

  $\{p(\underline{x})\}\ P\ \{q(\underline{x})\}$ a.cs.a. ha minden teljes u útra:

  $$\forall \underline{x}: p(\underline{x}) \wedge R_u(\underline{x}) \Rightarrow q(T_u(\underline{x}))$$

Doménen értelmezett elsőrendű logikai kifejezés;
ez tételbizonyítónak beadható:

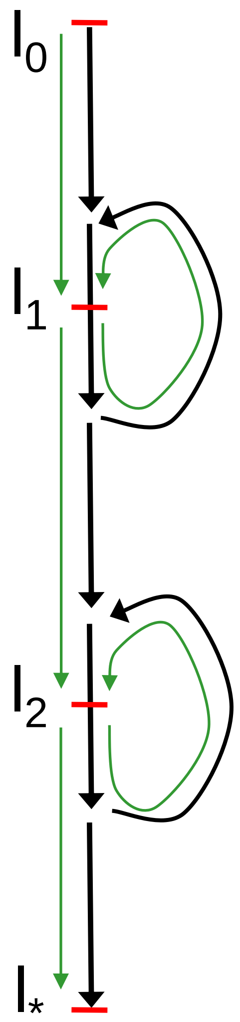
$$\overbrace{\boxed{\forall \underline{x}: p(\underline{x}) \wedge R_u(\underline{x}) \Rightarrow q(T_u(\underline{x}))}}$$

p és q a specifikáció alapján

R és T a program forrás alapján,
visszalépéses számítási indukció
alapján megadható minden útra

Részleges helyesség ciklusos programokra (1)

- Ötlet: Ciklusok felvágása
  - Minden ciklusban egy l_i utasítás kijelölése, ami azt (ciklusmentes) szegmensekre osztja
  - $I_{li}(\underline{x})$ predikátum, ún. induktív állítás hozzárendelése a vágási ponthoz
    - Első belépés után l_i-nél igaz
    - Ciklus futása során igaz marad (ciklus invariáns)
    - Kilépés esetén következő szegmens bejárási feltételét igazzá teszi
→ majd az utófeltételt igazzá teszi
  - A szegmensek mint ciklusmentes utak az előző módszer alapján vizsgálhatók
    - Elérhetőségi feltétel és
    - állapot-transzformáció számítható



Részleges helyesség ciklusos programokra (2)

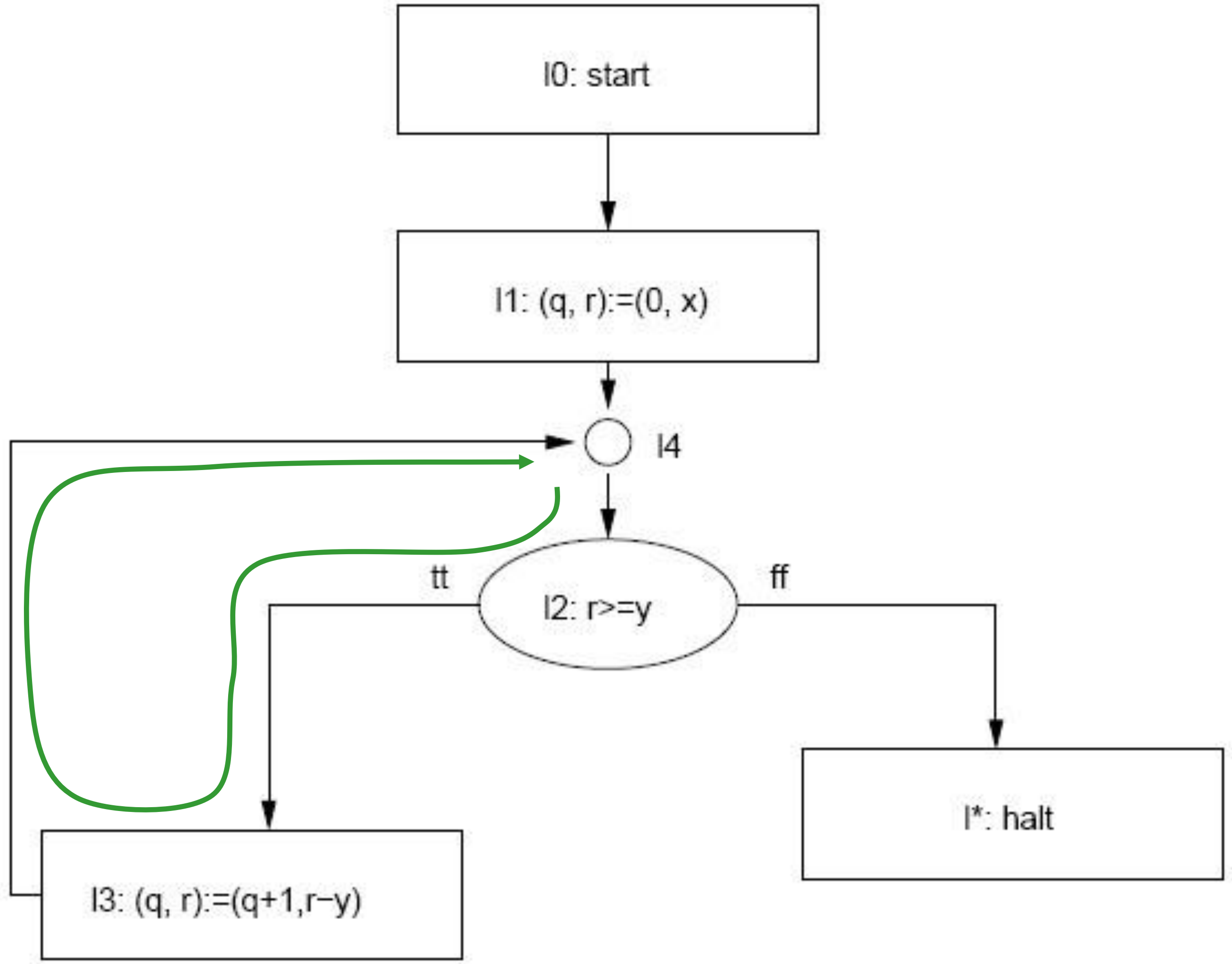
- Bizonyítási váz:
  - Vágási pontok kijelölése a ciklusokban (legalább egy-egy)
  - Induktív állítások felírása: $I_{li}(\underline{x})$
    - $I_{l0}(\underline{x}) = p(\underline{x})$ - kezdőállapotra
    - $I_{l*}(\underline{x}) = q(\underline{x})$ - végállapotra
    - Ciklusokban: ld. előbb (ciklus invariáns)
  - Verifikációs feltételek: A szomszédos vágási pontokra, azaz minden l, l' vágási pontok által kijelölt u szegmensre:

$$\boxed{\forall \underline{x}: I_l(\underline{x}) \wedge R_u(\underline{x}) \Rightarrow I_{l'}(T_u(\underline{x})) \text{ bizonyítandó}}$$

    - $R_u(\underline{x})$ és $T_u(\underline{x})$ a szegmensekre kiszámíthatók
- Helyes és teljes módszer
  - Mindig találhatók megfelelő vágási pontok és induktív állítások (de ennek bizonyítása nem konstruktív ☹)
  - Heurisztikus az induktív állítások felvétele

Részleges helyesség ciklusos programokra (példa)

$$I_{l4}(x,y,q,r) = (x\geq 0 \wedge y>0 \wedge x=q\cdot y+r \wedge r\geq 0)$$



Részleges helyesség ciklusos programokra: Példa ☺

$I_{l_4}(x,y,q,r) = (x = X \geq 0 \wedge y = Y \geq 0 \wedge X = q \cdot Y + r \wedge r \geq 0)$.
Be kell látnunk, hogy a fentebb felsorolt három feltétel teljesül:

- l_4 első elérésekor I_{l_4} teljesül. Mivel tudjuk, hogy $R_{(l_0,l_4)}(x,y,q,r) = \text{true}$ és $T_{(l_0,l_4)}(x,y,q,r) = (x,y,0,x)$, ezért a vizsgálandó $p(x,y) \wedge R_{(l_0,l_4)}(x,y,q,r) \Rightarrow I_{l_4}(T_{(l_0,l_4)}(x,y,q,r))$ alakú kifejezés itt $(x = X \geq 0 \wedge y = Y \geq 0 \wedge \text{true}) \Rightarrow (x = X \geq 0 \wedge y = Y \geq 0 \wedge X = 0 \cdot Y + x \wedge x \geq 0)$, ami belátható.
- A ciklus bejárása során a program megőrzi $I_l(\overline{x})$-et. Mivel $R_{(l_4,l_4)}(x,y,q,r) = r \geq y$ és $T_{(l_4,l_4)}(x,y,q,r) = (x,y,q+1,r-y)$, ezért a vizsgálandó $I_{l_4}(x,y,q,r) \wedge R_{(l_4,l_4)}(x,y,q,r) \Rightarrow I_{l_4}(T_{(l_4,l_4)}(x,y,q,r))$ kifejezés itt $(x = X \geq 0 \wedge y = Y \geq 0 \wedge X = q \cdot Y + r \wedge r \geq 0 \wedge r \geq y) \Rightarrow (x = X \geq 0 \wedge y = Y \geq 0 \wedge x = (q+1) \cdot Y + r - y \wedge r - y \geq 0)$ lesz, ami belátható.
- Ha a program kilép a ciklusból, akkor (a további szegmensekkel) biztosítja az utófeltételt. Mivel $R_{(l_4,l_*)}(x,y,q,r) = \neg r \geq y$ (azaz $r < y$), valamint $T_{(l_4,l_*)}(x,y,q,r) = (x,y,q,r)$ (identitás), ezért a vizsgálandó $I_{l_4}(x,y,q,r) \wedge R_{(l_4,l_*)}(x,y,q,r) \Rightarrow q(x,y,q,r)$ kifejezéshez azt kell belátnunk, hogy $x = X \geq 0 \wedge y = Y \geq 0 \wedge X = q \cdot Y + r \wedge r \geq 0 \wedge r < y \Rightarrow X = q \cdot Y + r \wedge 0 \leq r < Y$, ami triviális.

Befejeződés bizonyítása ciklusok esetén (1)

- Ötlet: Az induktív állítások paraméterezése
  - Paraméter egy (W,>) ún. jól megalapozott halmazból
    - Nem létezik végtelen $w_0 > w_1 > \ldots$ csökkenő szekvencia, $w_i \in W$
    - Példák:
      - Természetes számok, és az ezeken értelmezett > reláció
      - Véges halmaz valódi részhalmazai, és a tartalmazás reláció
      - Véges lista, és a prefix reláció
      - …
  - A ciklus befejeződik, ha a paraméterről kimutatható, hogy csökken a ciklus végrehajtása során
  - A paraméter sok esetben lehet maga a ciklusváltozó, de segédváltozót is használhatunk
    - Megválasztása heurisztikus

Befejeződés bizonyítása ciklusok esetén (2)

- Bizonyítási váz:
  - Jól megalapozott halmaz(ok) választása: $(W,<)$
  - Vágási pontok kijelölése a ciklusokban: l_0, l_i, l_*
  - Paraméterezett induktív állítások: $I_l(\underline{x},w)$ ahol $w \in W$
  - Verifikációs feltételek (bizonyítandók):
    - Inicializálás: $\forall \underline{x}$: $p(\underline{x}) \Rightarrow \exists w$: $I_{l0}(\underline{x},w)$ (előfeltétel bővítés)
    - Terminálás: $\forall \underline{x}$: $I_{l*}(\underline{x},w) \Rightarrow q(\underline{x})$
    - Csökkenés: Szomszédos l, l' pontok által kijelölt u szegmensre:

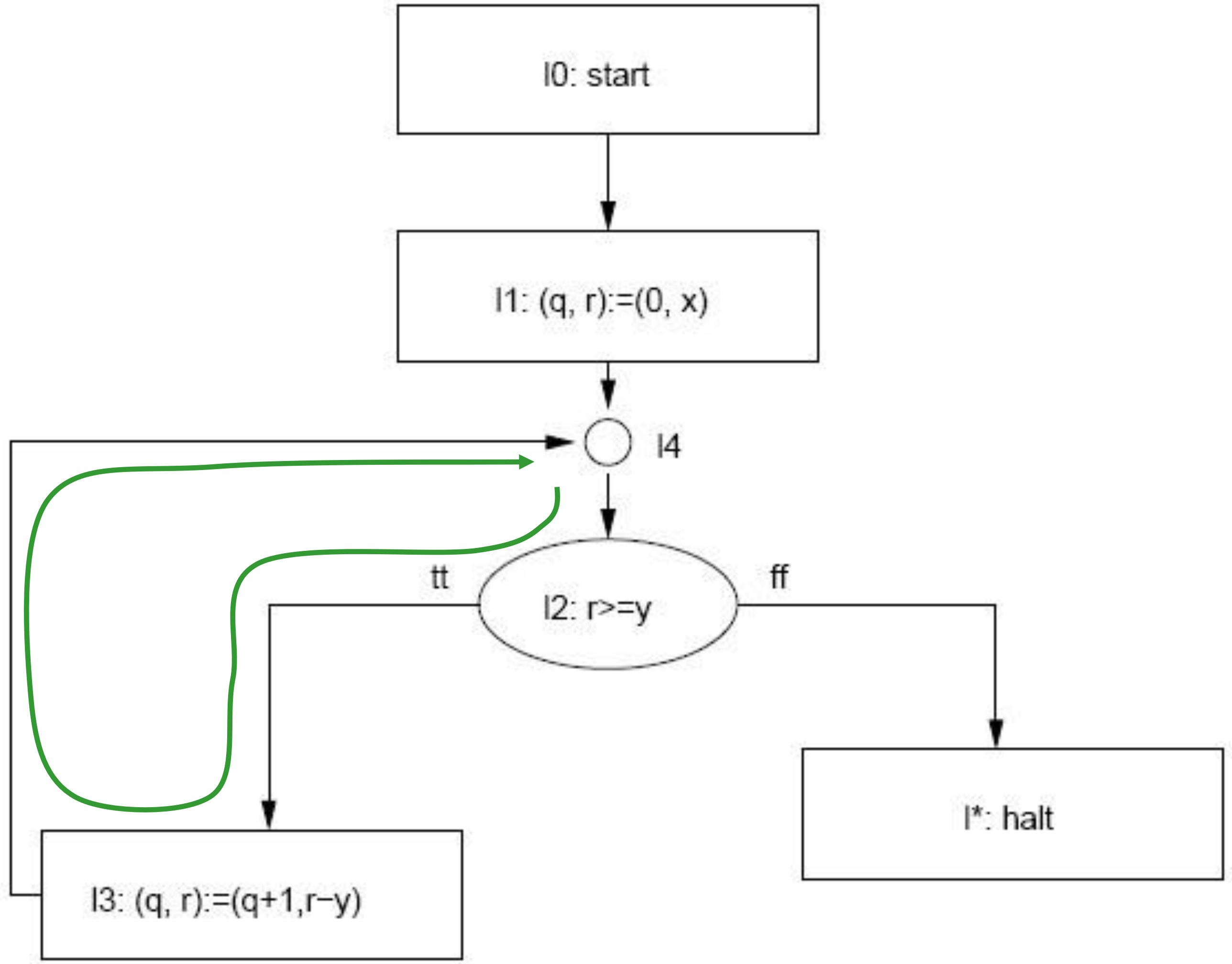
$$\boxed{\forall \underline{x}: \ I_l(\underline{x},w) \wedge R_u(\underline{x}) \Rightarrow \exists w'<w: \ I_{l'}(T_u(\underline{x}),w')}$$

Itt $R_u(\underline{x})$ és $T_u(\underline{x})$ a szegmensekre kiszámíthatók

- Helyes módszer $<p(\underline{x})>P<true>$ bizonyítására
  - A paraméterezett induktív állítások felvétele heurisztikus

Befejeződdés bizonyítása ciklusok esetén (példa)

$$I_{l4}(x,y,n) = (y>0 \wedge n=r\geq 0)$$



Rész-összefoglaló

- Alacsony szintű pszeudo-nyelvek:
  - Részleges helyesség ciklusmentes programokra:
    - Visszalépéses számítási indukció
  - Részleges helyesség ciklust tartalmazó programokra:
    - Induktív állítások módszere
  - Helyesség ciklust tartalmazó programokra:
    - Paraméterezett induktív állítások módszere
- Következő lépés: Strukturált programnyelvek

Helyességbizonyítás strukturált programokra

- Cél a „komponálhatóság”:
  - Ha egy P program P_1 és P_2 szintaktikai egységekből áll, akkor P tulajdonságai P_1 és P_2 tulajdonságai alapján bizonyíthatók
  - Strukturális indukció elve
- Strukturált programok: PLW nyelv

P::= x:=e | skip | P_1; P_2 | if B then P_1 else P_2 fi | while B do P od

- Példa:

P_{div}: r:=x; q:=0; while r≥y do r:=r-y; q:=q+1 od

PLW szemantikája

- Konfiguráció: $C_i=(P_i, \sigma_i)$ ahol
  - P_i a szintaktikus folytatás (E az üres folytatás jelölése)
  - σ_i a látható állapot (állapotváltozók)
- Átmenet reláció: $C \to C'$
  - $(x:=e, \sigma) \to (E, \sigma[e/x])$
  - $(\text{skip}, \sigma) \to (E, \sigma)$
  - $(P_1; P_2, \sigma) \to (P_1'; P_2, \sigma')$ ha $(P_1, \sigma) \to (P_1', \sigma')$
  - $(\text{if } B \text{ then } P_1 \text{ else } P_2 \text{ fi}, \sigma) \to (P_1, \sigma)$ ha $\sigma[B]=\text{true}$

    $\to (P_2, \sigma)$ ha $\sigma[B]=\text{false}$
  - $(\text{while } B \text{ do } P \text{ od}, \sigma) \to (P; \text{while } B \text{ do } P \text{ od}, \sigma)$ ha $\sigma[B]=\text{true}$

    $\to (E, \sigma)$ ha $\sigma[B]=\text{false}$

Itt az $E;P \equiv P$ azonosság alkalmazható a végén

H dedukciós rendszer részleges helyesség bizonyításához (1)

- Axiómák:
  - ASS: $\{p[e/x]\}\ x:=e\ \{p\}$
  - SKIP: $\{p\}\ \text{skip}\ \{p\}$

Utófeltételként p teljesül, ha előfeltételként p[e/x] teljesül

- Szabályok a szintaktikai struktúrákhoz:
  - SEQ: $$\frac{\{p\}\ P_1\ \{r\}\ \text{ és }\ \{r\}\ P_2\ \{q\}}{\{p\}\ P_1;\ P_2\ \{q\}}$$
  - COND: $$\frac{\{p \wedge B\}\ P_1\ \{q\}\ \text{ és }\ \{p \wedge \neg B\}\ P_2\ \{q\}}{\{p\}\ \text{if } B \text{ then } P_1 \text{ else } P_2 \text{ fi}\ \{q\}}$$
  - REP: $$\frac{\{p \wedge B\}\ P\ \{p\}}{\{p\}\ \text{while } B \text{ do } P \text{ od}\ \{p \wedge \neg B\}}$$

$\frac{\text{Ha (feltétel)}}{\text{akkor (köv.)}}$

p ciklus invariáns

H dedukciós rendszer részleges helyesség bizonyításához (2)

- Általános szabályok:
  - CONS: $$\frac{p \Rightarrow p_1 \text{ és } \{p_1\}\, P_1\, \{q_1\} \text{ és } q_1 \Rightarrow q}{\{p\}\, P\, \{q\}}$$
  - AND: $$\frac{\{p\}\, P\, \{q_1\} \text{ és } \{p\}\, P\, \{q_2\}}{\{p\}\, P\, \{q_1 \wedge q_2\}}$$
  - OR: $$\frac{\{p_1\}\, P\, \{q\} \text{ és } \{p_2\}\, P\, \{q\}}{\{p_1 \vee p_2\}\, P\, \{q\}}$$

Előfeltétel bővítés és utófeltétel szűkítés

Utófeltétel részekre bontása

Előfeltétel szétválasztása esetekre

- Domén axiómái és szabályai:
  - A tételbizonyítónak ismernie kell
  - {true}skip{p} alakú állítások is

Részleges helyességbizonyítás példa ☺

$\{x \geq 0 \wedge y \geq 0\}$ r:=x; q:=0; while r≥y do r:=r-y; q:=q+1 od $\{x = q \cdot y + r \wedge 0 \leq r < y\}$

1. $\{x = 0 \cdot y + x \wedge x \geq 0\}\ q := 0\ \{x = q \cdot y + x \wedge x \geq 0\}$ (ASS)
2. $\{x = q \cdot y + x \wedge x \geq 0\}\ r := x\ \{x = q \cdot y + r \wedge r \geq 0\}$ (ASS)
3. $\{x = 0 \cdot y + x \wedge x \geq 0\}\ q := 0; r := x\ \{x = q \cdot y + r \wedge r \geq 0\}$ (1)(2)(SEQ)
4. $x \geq 0 \wedge y \geq 0 \Rightarrow x = 0 \cdot y + x \wedge x \geq 0$ (ARITHMETIC)
5. $\{x \geq 0 \wedge y \geq 0\}\ q := 0; r := x\ \{x = q \cdot y + r \wedge r \geq 0\}$ (3)(4)(CONS)
6. $\{x = (q+1) \cdot y + r - y \wedge r - y \geq 0\}\ r := r - y\ \{x = (q+1) \cdot y + r \wedge r \geq 0\}$ (ASS)
7. $\{x = (q+1) \cdot y + r \wedge r \geq 0\}\ q := q + 1\ \{x = q \cdot y + r \wedge r \geq 0\}$ (ASS)
8. $\{x = (q+1) \cdot y + r - y \wedge r - y \geq 0\}\ r := r - y; q := q + 1\ \{x = q \cdot y + r \wedge r \geq 0\}$ (6)(7)(SEQ)
9. $x := q \cdot y + r \wedge r \geq 0 \wedge r \geq y \Rightarrow x = (q+1) \cdot y + r - y \wedge r - y \geq 0$ (ARITHMETIC)
10. $\{x = q \cdot y + r \wedge r \geq 0 \wedge r \geq y\}\ r := r - y; q := q + 1\ \{x = q \cdot y + r \wedge r \geq 0\}$ (8)(9)(CONS)
11. $\{x = q \cdot y + r \wedge r \geq 0\}$ while $r \geq y$ do $r := r - y; q := q + 1$ od $\{x = q \cdot y + r \wedge 0 \leq r < y\}$ (10)(REP)
12. $\{x \geq 0 \wedge y \geq 0\}\ q := 0; r := x;$ while $r \geq y$ do $r := r - y; q := q + 1$ od $\{x = q \cdot y + r \wedge 0 \leq r < y\}$ (5)(11)(SEQ)

Dedukciós rendszer részleges helyesség bizonyításához (3)

- Egy C állítás bizonyítása: $Tr_I \vdash_H C$ ahol
  - I a domén, Tr_I a domén axiómái és szabályai
  - H a dedukciós rendszer
- Példa az állításra:
  - $\{x \geq 0 \wedge y \geq 0\}\ P_{div}\ \{x = q \cdot y + r \wedge 0 \leq r < y\}$
- Gyakorlati problémák:
  - Domén szabályait a tételbizonyítónak ismernie kell
  - Szabályok alkalmazásához stratégia (vagy keresés) kell
- Jellemzők:
  - Az előbb felvázolt H helyessége bizonyítható
    - $Tr_I \vdash_H \{p\}P\{q\}$ következménye $\models_I \{p\}P\{q\}$
  - H teljessége:
    - Ha a domén axióma- és szabályrendszere kellően bonyolult (elegendően erős): Gödel első nemteljességi tétele érvényes, azaz lehet olyan igaz állítás, ami nem bizonyítható
  - Specifikációs nyelv kifejezőképessége elegendő-e?

Specifikációs nyelv kifejezőképessége

- Definíció:
  - SL specifikációs nyelv kifejező egy PL programnyelv és I domén esetén, ha $\forall p \in SL$, $\forall P \in PL$ esetén $post_I(p,P)$ kifejezhető SL-ben

P futása után elért állapotok predikátuma

Kiindulási állapot predikátuma

P program

  - Egy D dedukciós rendszer relatív teljes a részleges helyesség bizonyításához, ha $\forall SL$, $\forall PL$, $\forall I$ esetén, ahol SL kifejező PL-re és I-re, fennáll: $\models_I \{p\}P\{q\}$ következménye $Tr_I \vdash_D \{p\}P\{q\}$

H* dedukciós rendszer helyesség bizonyításhoz

- Cél: PLW programok helyességének bizonyítása
  - while B do P od ciklusok befejeződése kérdéses
- Ötlet (itt is): Paraméterezett állítások
  - Jól megalapozott halmaz (pl. n természetes szám)
  - $pi(\underline{x},n)$ ciklus invariáns választása kell
- Módosuló REP szabály ebben az esetben:
  - REP*:

$$\frac{pi(\underline{x},n)\Rightarrow B \quad \text{és} \quad <pi(\underline{x},n)>P<pi(\underline{x},n-1)> \quad \text{és} \quad pi(\underline{x},0) \Rightarrow \neg B}{<\exists n:pi(\underline{x},n)> \text{ while B do P od } <pi(\underline{x},0)>}$$

  - Többi szabály:
    {…} helyett egyszerűen <…> kell

Aritmetikai kiterjesztés

- A módosult REP* szabály miatt:
  - SL-nek támogatnia kell a jól megalapozott halmaz (itt: természetes számok) használatát
- Tipikus eset: Peano aritmetika támogatása
  - Természetes számok és +, *, < műveletek
  - Tételbizonyító erre felkészítve
- Definíciók:
  - SL aritmetikai kiterjesztése SL^+, ha minimális bővítésként a Peano aritmetikát tartalmazza
  - I domén aritmetikai kiterjesztése I^+, ha minimális bővítésként a Peano aritmetika doménjét tartalmazza

Aritmetikai helyesség és teljesség

- Aritmetikai helyesség $p,q \in SL^+$ mellett:
  - Tr_{I+} $\vdash_D$ <p>P<q> következménye $\models_{I+}$ <p>P<q>

- Aritmetikai teljesség $p,q \in SL^+$ mellett:
  - $\models_{I+}$ <p>P<q> következménye Tr_{I+} $\vdash_D$ <p>P<q>

- Itt: H* aritmetikai helyessége bizonyítható

Összefoglalás

- Alacsony szintű pszeudo-nyelvek:
  - Részleges helyesség ciklusmentes programokra:
    - Visszalépéses számítási indukció
  - Részleges helyesség ciklust tartalmazó programokra:
    - Induktív állítások módszere
  - Helyesség ciklust tartalmazó programokra:
    - Paraméterezett induktív állítások módszere
- Strukturált programnyelvek (while programok):
  - Részleges helyesség:
    - Dedukciós rendszer (strukturális indukció)
  - Helyesség:
    - Dedukciós rendszer paraméterezett állításokkal
    - Aritmetikai kiterjesztés, aritmetikai helyesség és teljesség

Program helyességbizonyítás a gyakorlatban

Néhány példa:

- Spec# Programming System: C# kiterjesztés
  - Előfeltételek, utófeltételek megadása (metódusokhoz)
  - Objektum invariánsok (pl. adattartományok)
  - Boogie: Az utófeltételek automatikus ellenőrzése
- JML: Java Modelling Language
  - Előfeltételek, utófeltételek, invariánsok megadása
  - ESC/Java2: JML részhalmazhoz automatikus utófeltétel bizonyítás
- SPARK: Ada nyelvi részhalmaz
  - Interaktív tételbizonyítóval támogatott ellenőrzés
- B módszer: Speciális modellezési nyelv és megközelítés
  - Bizonyítandó tételek származtatása és a tételbizonyítás nagy részben automatikus