

Formális helyességbizonyítás

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

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

Motiváció

- Kritikus algoritmusok helyességének bizonyítása a részletes tervek alapján
 - Korlátozott feladatok: biztonságkritikus funkciók, hozzáférésvédelem, protokoll mag, ...
 - Részletes terv: Pseudo-nyelven leírt algoritmus
 - A továbbiakban mint „program”
 - (Valós programnyelvek részleges támogatása)
- Tételbizonyító rendszerek használata
 - Bizonyítandó tulajdonság mint bizonyítandó „tétel” adott
 - Klasszikus elő- és utófeltételek a tipikusak (contract)
- Kihívások:
 - (Pseudo) 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**
 - 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 kritérium (tétel)
- **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
- **Dedukciós rendszer konzisztens:**
 - Nem lehet egy tételt és az ellenkezőjét is bizonyítani

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:**

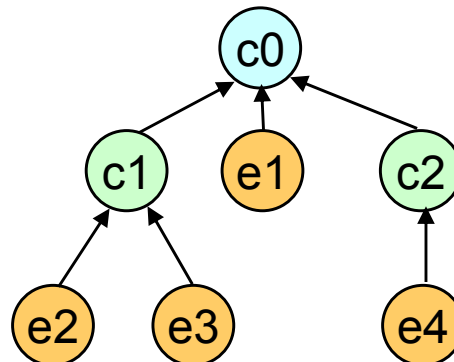
Ha s_0 (kiindulási állapot) esetén igaz a tulajdonság, és az átmenetek megőrzik, akkor végig fennáll



- **Strukturális indukció:** Axiomatikus szemantikához

- **Szintaktikai konstrukciókra:**

Ha elemekre igaz, és a konstrukciók megőrzik, akkor a teljes programra igaz



Célkitűzéseink

- **Bizonyítási vázák rögzítése program helyesség bizonyításhoz**
 - Bizonyítási stratégia megadása
 - Nincs teljesen automatikus, garantált módszer
- **Nem konkrét programozási nyelvhez kötve**
 - Pseudo-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: A további számításokat definiálja
 - „Programszámláló” helyett
 - A továbbiakban végrehajtandó forrásszöveggént képzelhető el
- Átmenet reláció a konfigurációk között: \rightarrow
 - $\pi(P, \sigma_0)$ egy P program számítása σ_0 kezdőállapotból
 - $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow \dots$ maximális szekvencia (végállapotig/végtelen)
 - $\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 \rightarrow \dots$ látható állapot szekvencia
 - $\text{val}(\pi(P, \sigma)) = \sigma_n$ 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: Predikátumok
 - 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
 - $\text{val}(\pi(P, \sigma_0)) \models q(\underline{x})$ jelentése: π számítás végállapotában igaz $q(\underline{x})$
- Specifikáció példák
 - Segédváltozók használata
 - Specifikációs változók használata

Példák specifikációkra

- x és y nagyság szerinti sorba rendezése:
előfeltétel $p(x,y) = \text{true}$, utófeltétel $q(x,y) = x > y$
tömörebben felírva $(\text{true}, x > y)$
- x legyen páros a kimeneten:
 $(\text{true}, \text{even}(x))$ ha a doménen értelmezett $\text{even}(x)$
 $(\text{true}, \exists y: x=2y)$ 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 :
 $(x=X > 0 \wedge y=Y > 0, X=qY+r \wedge 0 \leq r < Y)$
ha meg kell őrizni x és y értékét:
 $(x=X > 0 \wedge y=Y > 0, X=qY+r \wedge 0 \leq r < Y \wedge x=X \wedge y=Y)$

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

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

- **Megjegyzések:**

- Az előfeltételt kielégítő állapotból induló számításokra:
ha befejeződik, akkor az utófeltétel igaz a végállapotban
- Nem mond semmit azokról a számításokról,
amelyekre $\sigma_0 \not\models p(\underline{x})$
- $\{\text{true}\}P\{\text{true}\}$ minden programra igaz
- $\{\text{true}\}P\{\text{false}\}$ 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 $\forall \pi(P, \sigma_0)$ és $\sigma_0 \models p(\underline{x})$ esetén

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

- **Megjegyzések:**

- Az előfeltételt kielégítő állapotból induló számításokra: **befejeződik** és az utófeltétel igaz lesz a végállapotban

- $\langle p(\underline{x}) \rangle P \langle \text{true} \rangle$ 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 $\langle p(\underline{x}) \rangle P \langle \text{true} \rangle$

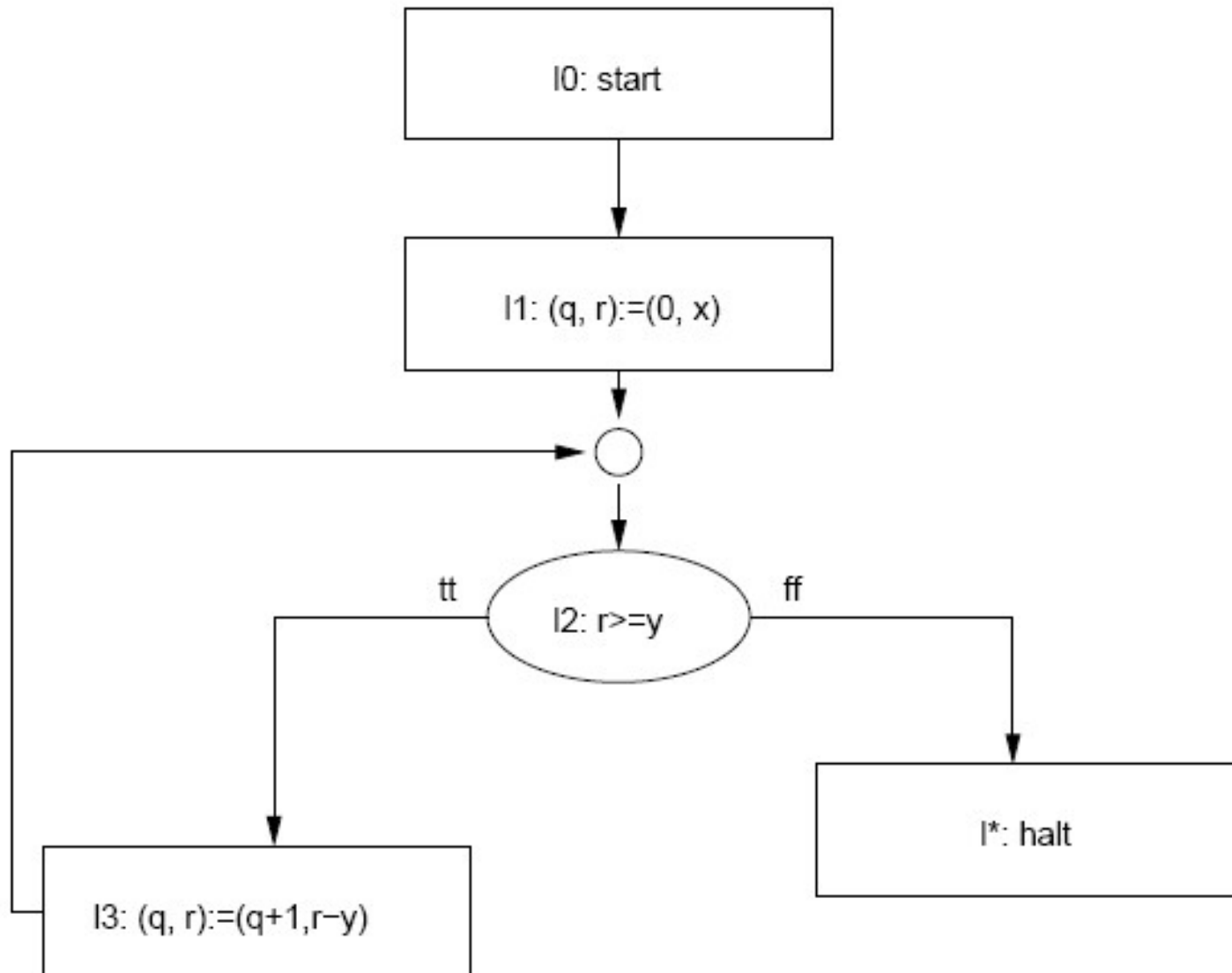
azaz a program helyes, ha részlegesen helyes és befejeződik

Egyszerű determinisztikus programok

- PLF „flow language”:
 - $start, \underline{x}:=\underline{e}, B(\underline{x}), halt$ utasítások egyedi l címkékkel (l_0, l_*, l_j, \dots)
 - Assembly szinthez közeli pseudo-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: $C=(\sigma, \lambda)$ konfiguráció és \rightarrow megadása:
 $C(\sigma, \lambda) \rightarrow C'(\sigma', \lambda')$ a.cs.a.
 - λ egy $start$: $\lambda' = succ(\lambda), \quad \sigma' = \sigma$
 - λ egy $\underline{x}:=\underline{e}$ utasítás: $\lambda' = succ(\lambda), \quad \sigma' = \sigma[\underline{e}/\underline{x}]$
 - itt $[\underline{e}/\underline{x}]$ jelöli, hogy \underline{x} helyébe \underline{e} helyettesítése történik
 - λ egy $B(\underline{x})$ elágazás:
 - $\sigma \models B(\underline{x})$ esetén: $\lambda' = succ^+(\lambda), \quad \sigma' = \sigma$
 - $\sigma \not\models B(\underline{x})$ esetén: $\lambda' = succ^-(\lambda), \quad \sigma' = \sigma$

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:
 - $u = l_0 \rightarrow l_1 \rightarrow l_2 \rightarrow \dots \rightarrow l_k$
 - **Elérhetőségi (bejárási) feltétel:** $R_u(\underline{x})$ predikátum
 - Ha fennáll l_0 esetén, akkor éppen az u útvonalat járja be a program
 - **Állapot transzformáció:** $T_u(\underline{x})$ egy állapotvektort ad
 - \underline{x} állapotvektorból indulva az u útvonal bejárása utáni állapot
 - $\underline{x} := T_u(\underline{x})$ a végállapotot eredményező állapot transzformáció
- Jelölések:
 - $l_m \rightarrow \dots \rightarrow l_k$ 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):
 - $R_u^k(\underline{x}) = \text{true}$ - már a végállapotban vagyunk
 - $T_u^k(\underline{x}) = \underline{x}$ - 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:

$$R_u^m(\underline{x}) = R_u^{m+1}(\underline{x}) \wedge B(\underline{x}),$$

$$T_u^m(\underline{x}) = T_u^{m+1}(\underline{x})$$

- $B(\underline{x})$ feltétel negatív ága:

$$R_u^m(\underline{x}) = R_u^{m+1}(\underline{x}) \wedge \neg B(\underline{x}),$$

$$T_u^m(\underline{x}) = T_u^{m+1}(\underline{x})$$

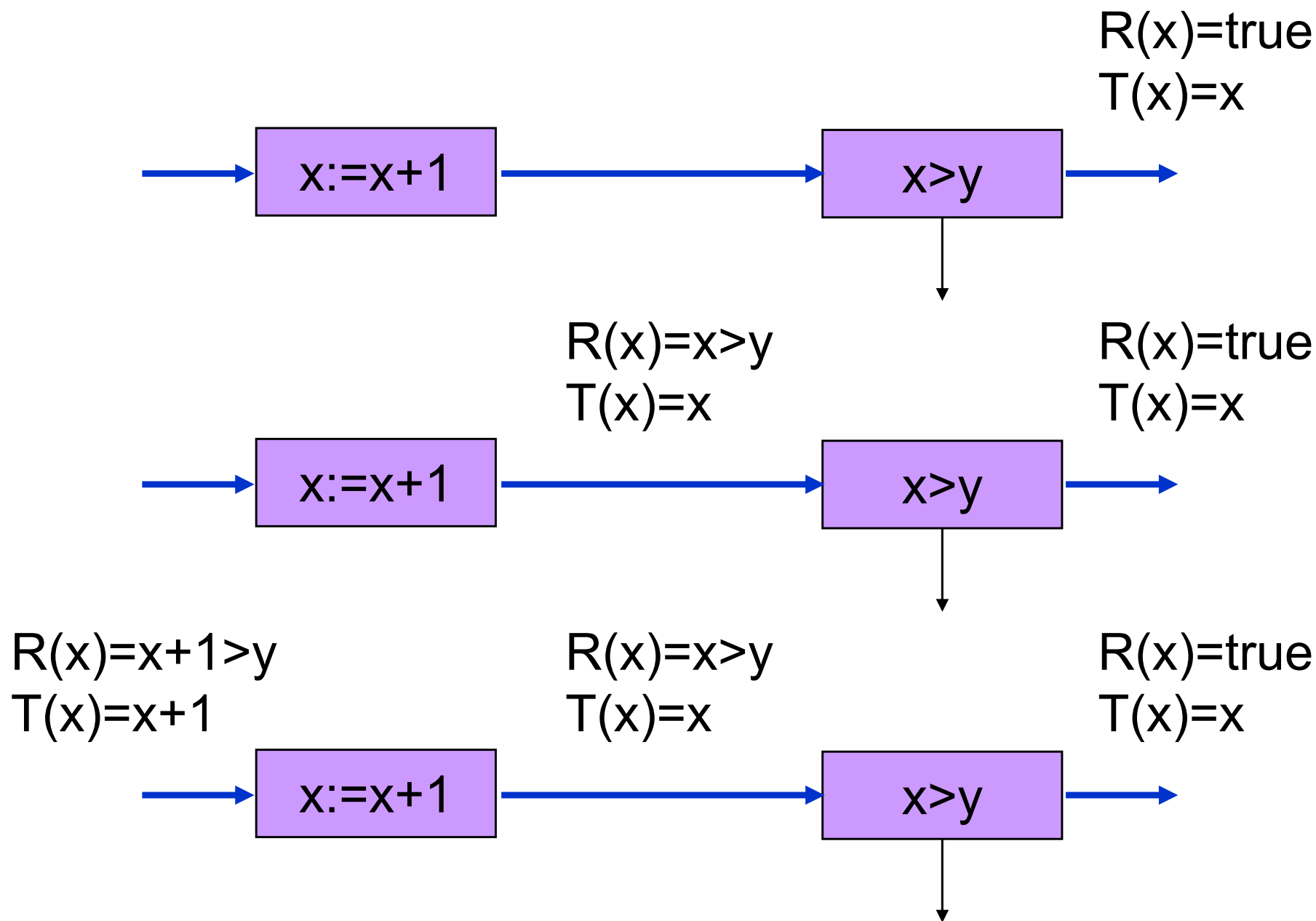
- *start:*

$$R_u(\underline{x}) = R_u^0(\underline{x}),$$

$$T_u(\underline{x}) = T_u^0(\underline{x})$$

- Így a végállapotban ismert $R_u^k(\underline{x}) = \text{true}$ 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)



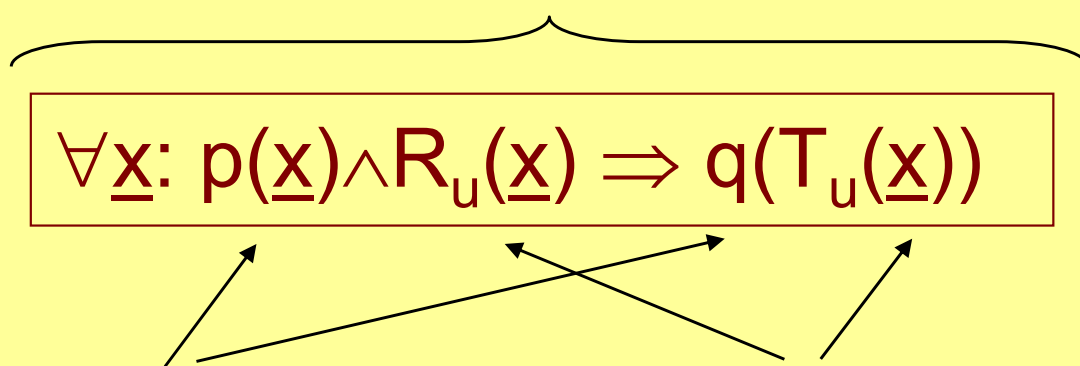
Részleges helyesség ciklusmentes esetben (3)

- 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;
tételbizonyítónak beadható:

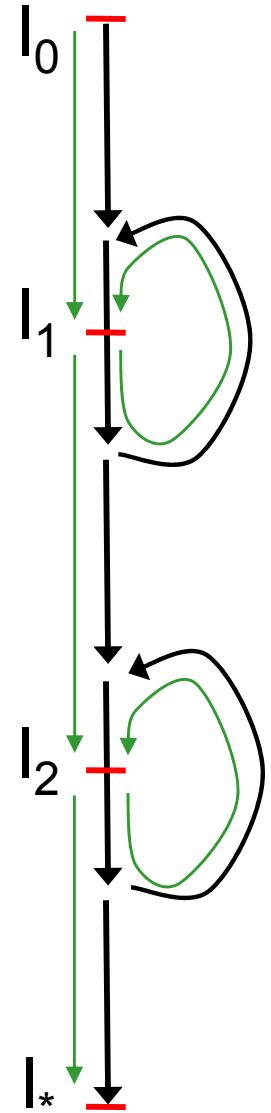

$$\forall \underline{x}: p(\underline{x}) \wedge R_u(\underline{x}) \Rightarrow q(T_u(\underline{x}))$$

p és q a specifikáció alapján

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

Részleges helyesség ciklusos programokra (1)

- **Ötlet: Ciklusok felvágása**
 - Minden ciklusban egy I_i utasítás kijelölése, ami azt (ciklusmentes) szegmensekre osztja
 - $I_i(\underline{x})$ predikátum, ún. **induktív állítás** hozzárendelése a vágási ponthoz
 - Első belépés után I_i -nél igaz
 - Ciklus futása során igaz marad (**ciklus invariáns**)
 - Kilépés után a további szegmensekkel 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: $I_i(\underline{x})$
 - $I_{i_0}(\underline{x}) = p(\underline{x})$ - kezdőállapotra
 - $I_{i^*}(\underline{x}) = q(\underline{x})$ - végállapotra
 - Ciklusokban: ld. előbb (ciklus invariáns)
- Verifikációs feltételek: A szomszédos vágási pontokra, azaz minden I, I' vágási pontok által kijelölt u szegmensre:

$$\forall \underline{x}: I_i(\underline{x}) \wedge R_u(\underline{x}) \Rightarrow I_{i'}(T_u(\underline{x})) \text{ bizonyítandó}$$

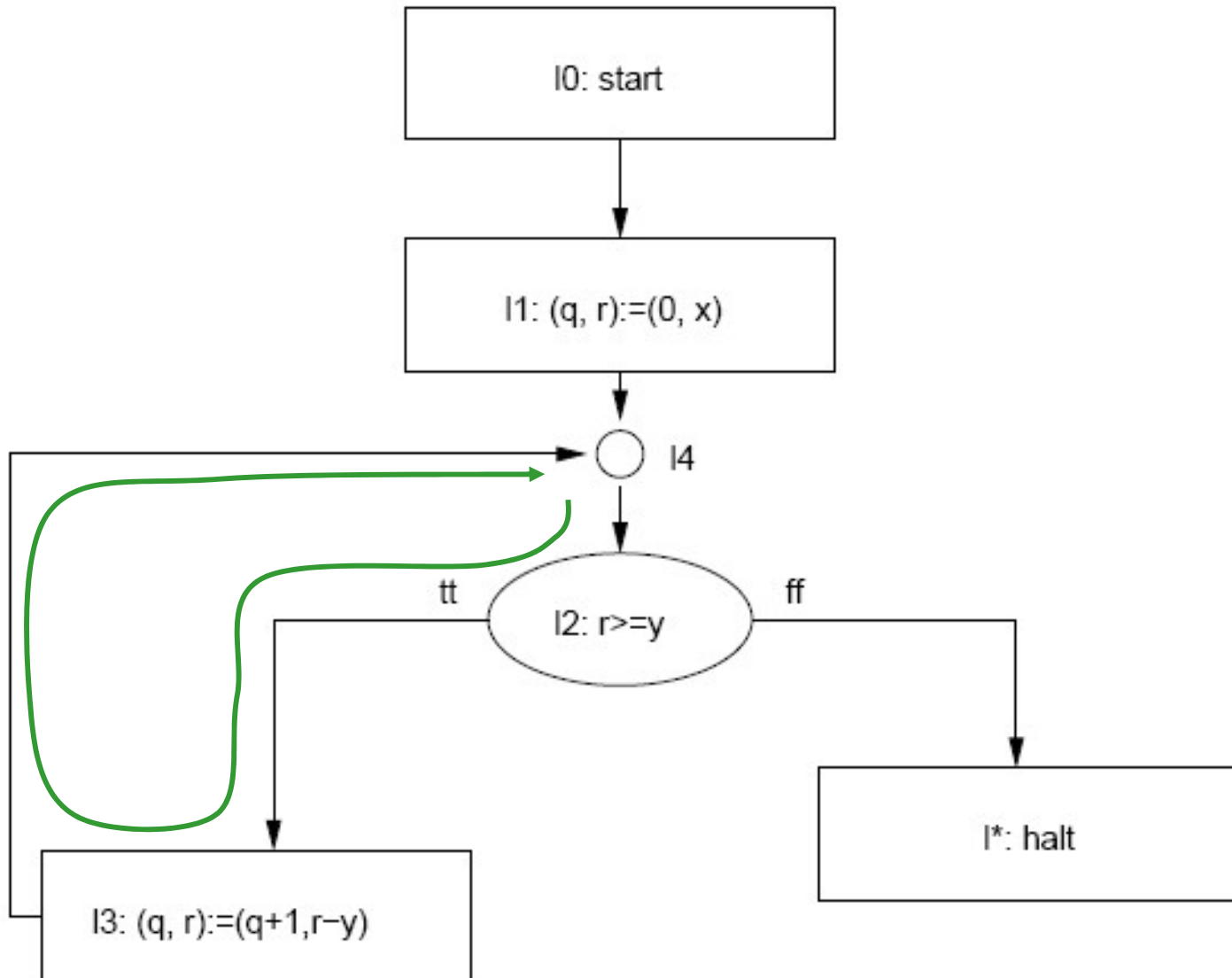
- $R_u(\underline{x})$ és $T_u(\underline{x})$ a szegmensekre kiszámíthatók

- Helyes és teljes módszer

- Mindig található megfelelő vágási pontok és induktív állítások (de ennek bizonyítása nem konstruktív ☹)
- Heurisztikus az induktív állítások felvétele

Részleges helyesség ciklusos programokra (példa)

$$I_{l_4}(x,y,q,r) = (x \geq 0 \wedge y > 0 \wedge x = q \cdot y + r \wedge r \geq 0)$$



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 $I_l(\bar{x})$ -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 $R_{(l_4, l_s)}(x, y, q, r) = \neg r \geq y$ (azaz $r < y$), valamint $T_{(l_4, l_s)}(x, y, q, r) = (x, y, q, r)$ (identitás), ezért a vizsgálandó $I_{l_4}(x, y, q, r) \wedge R_{(l_4, l_s)}(x, y, q, r) \Rightarrow q(x, y, q, r)$ 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 $w_0 > w_1 > \dots$ csökkenő szekvencia
 - 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: I_0, I_i, I_*
- Paraméterezett induktív állítások: $I_i(\underline{x}, w)$ ahol $w \in W$
- Verifikációs feltételek (bizonyítandók):

- Inicializálás: $\forall \underline{x}: p(\underline{x}) \Rightarrow \exists w: I_{I_0}(\underline{x}, w)$ (előfeltétel bővítés)
- Terminálás: $\forall \underline{x}: I_{I_*}(\underline{x}, w) \Rightarrow q(\underline{x})$
- Csökkenés: Szomszédos I, I' pontok által kijelölt u szegmensre:

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

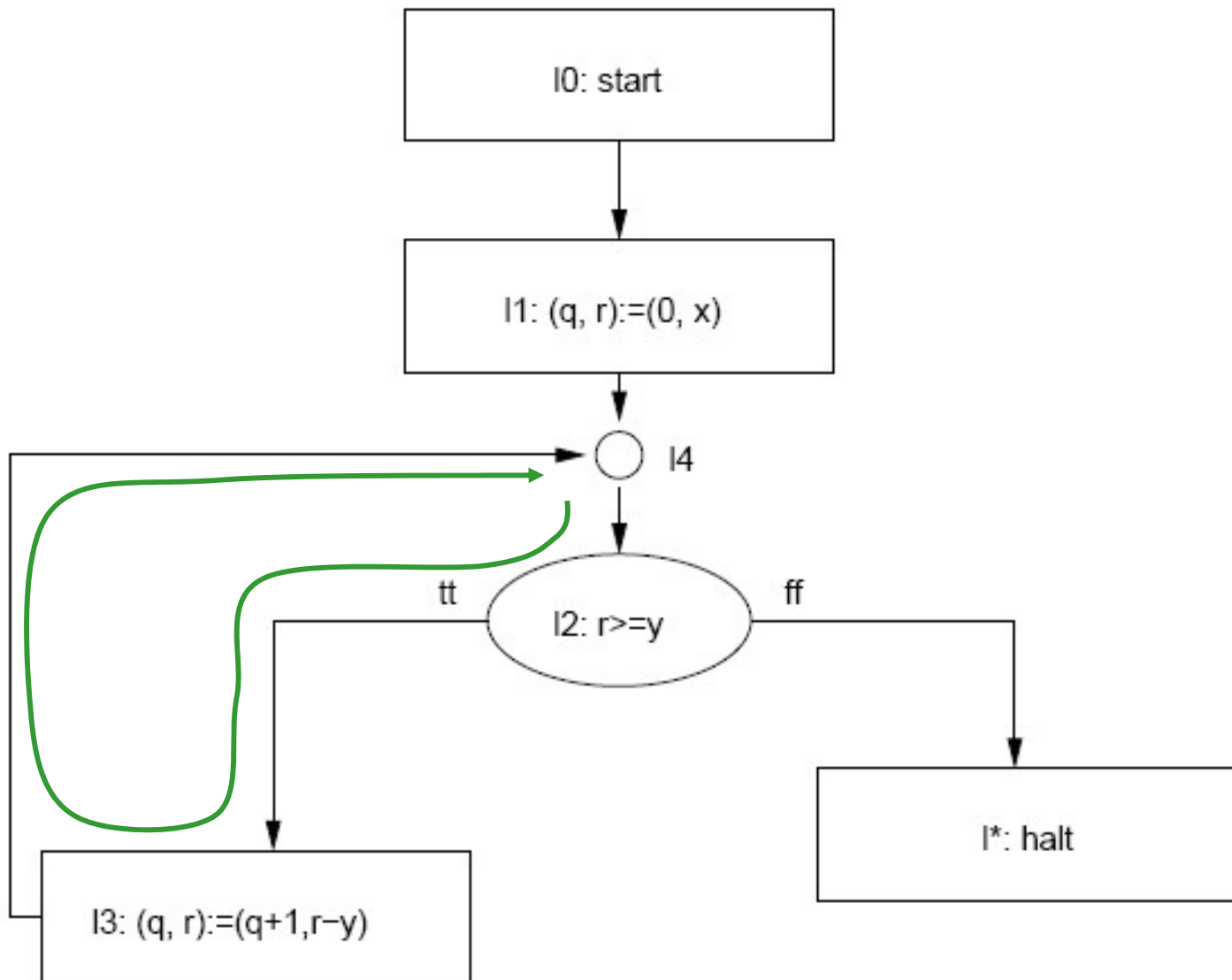
Itt $R_u(\underline{x})$ és $T_u(\underline{x})$ a szegmensekre kiszámíthatók

- Helyes módszer $\langle p(\underline{x}) \rangle P \langle \text{true} \rangle$ bizonyítására

- A paraméterezett induktív állítások felvétele heurisztikus

Befejeződés bizonyítása ciklusok esetén (példa)

$$I_{l_4}(x,y,n) = (y > 0 \wedge n=r \geq 0)$$



Rész-összefoglaló

- Alacsony szintű pseudo-nyelvek (blokk diagram):
 - Részleges helyesség ciklusmentes programokra:
 - Visszalépéses számítási indukció
 - Részleges helyesség ciklust tartalmazó programokra:
 - Induktív állítások
 - Helyesség ciklust tartalmazó programokra:
 - Paraméterezett induktív állítások
- Következő lépés: Strukturált programnyelvek