Mida tähendab, et programm on korrektne?

- Mis on programm?
- Mida ta teeb?
- Kuidas tema tegevuse üle arutleda?

Olgu meil antud mingi *muutujate* hulk $\mathbf{Var}$.

Aritmeetilist avaldist A defineeriv *abstraktne süntaks* on järgmine:

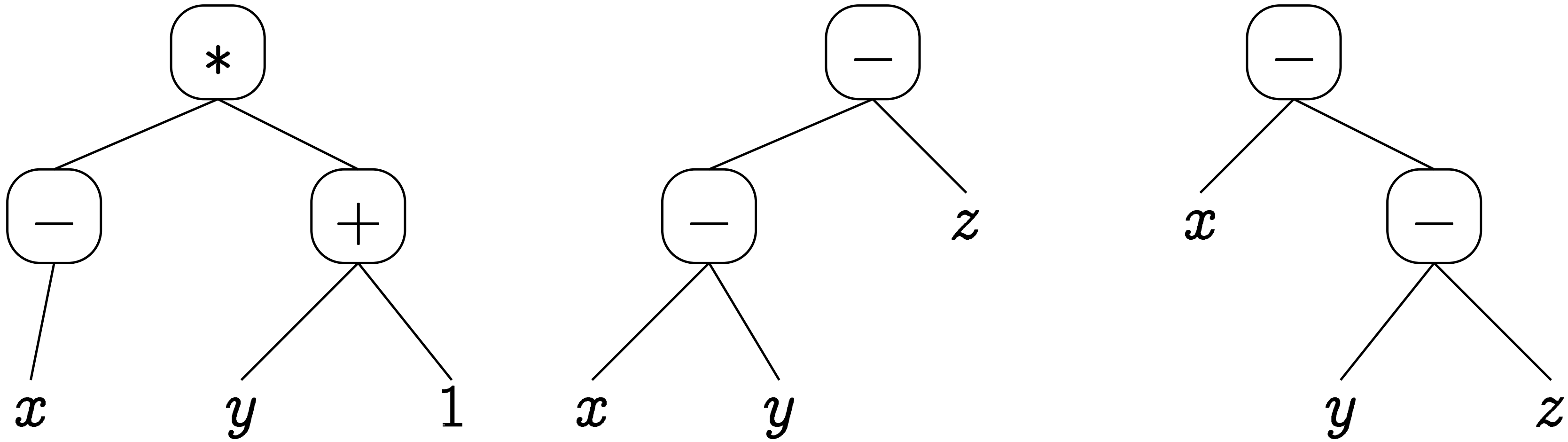
$$
\begin{array}{rcl}
A & ::= & n \\
  & | & x \\
  & | & -A_1 \\
  & | & A_1 + A_2 \mid A_1 - A_2 \mid A_1 * A_2 \mid A_1 / A_2
\end{array}
$$

kus A_1, A_2 on aritmeetilised avaldised, $x \in \mathbf{Var}$ ja $n \in \mathbb{Z}$.

Olgu $\mathbf{Aexp}$ kõigi aritmeetiliste avaldiste hulk.

Eelmisel slaidil on toodud abstraktne süntaks.

Objekte, mida see süntaks defineerib, tuleks ette kujutada termidena / puudena.



$(-x) * (y + 1)$ $\qquad$ $(x - y) - z$ $\qquad$ $x - (y - z)$

Konkreetne süntaks meid ei huvita. Kui vaja, siis lisame mingid grupeerijad (sulud).

Tõeväärtusavaldist B defineeriv abstraktne süntaks on järgmine:

$$\begin{array}{rcl} B & ::= & \mathsf{true} \mid \mathsf{false} \\ & \mid & A_1 = A_2 \mid A_1 \leqslant A_2 \mid \ldots \\ & \mid & \neg B_1 \\ & \mid & B_1 \wedge B_2 \mid B_1 \vee B_2 \mid \ldots \end{array}$$

kus A_1, A_2 on aritmeetilised avaldised, B_1 ja B_2 tõeväärtusavaldised.

Olgu **Bexp** kõigi tõeväärtusavaldiste hulk.

Olgu $\mathbb{B} = \{\mathsf{true}, \mathsf{false}\}$.

Programmi S defineeriv abstraktne süntaks on järgmine:

$$\begin{array}{rcl} S & ::= & x := A \\ & \mid & \mathit{skip} \\ & \mid & S_1 ; S_2 \\ & \mid & \mathit{if}\ B\ \mathit{then}\ S_1\ \mathit{else}\ S_2 \\ & \mid & \mathit{while}\ B\ \mathit{do}\ S_1 \end{array}$$

kus $x \in \mathbf{Var}$, A on aritmeetiline avaldis, B tõeväärtusavaldis ning S_1 ja S_2 programmid.

Teisi konstruktsioone antud kursuses ei vaatle.

Olgu $\mathbf{Prog}$ kõigi programmide hulk.

Programmiolek on kujutus, mis seab igale muutujale vastavusse tema väärtuse.

Võimalike väärtuste hulk on $\mathbf{Val} = \mathbb{Z}$. Võimalike programmiolekute hulk on

$$\mathbf{State} = \mathbf{Var} \to \mathbf{Val} \quad .$$

Kui $s \in \mathbf{State}$ ja $x \in \mathbf{Var}$, siis $s(x)$ on muutuja x väärtus programmiolekus s.

Programmi täitmine kujutab endast programmioleku muutmist vastavalt eeskirjale (milleks on see programm).

Aritmeetilisele avaldisele A ja programmiolekule s seame vastavusse täisarvu $\mathcal{A}[\![A]\!]s$.

Funktsiooni $\mathcal{A}$ tüüp on

$$\mathbf{Aexp} \to (\mathbf{State} \to \mathbb{Z}),$$

$$\begin{aligned}
\mathcal{A}[\![n]\!]s &:= n \\
\mathcal{A}[\![x]\!]s &:= s(x) \\
\mathcal{A}[\![-A]\!]s &:= -\mathcal{A}[\![A]\!]s \\
\mathcal{A}[\![A_1 + A_2]\!]s &:= \mathcal{A}[\![A_1]\!]s + \mathcal{A}[\![A_2]\!]s \\
\mathcal{A}[\![A_1 - A_2]\!]s &:= \mathcal{A}[\![A_1]\!]s - \mathcal{A}[\![A_2]\!]s \\
&\dots
\end{aligned}$$

Tehtemärgid vasakul ja paremal pool on erinevat tüüpi...

Tõeväärtusavaldisele B ja programmiolekule s seame vastavusse tõeväärtuse $\mathcal{B}[\![B]\!]s$.

Funktsiooni $\mathcal{B}$ tüüp on

$$\mathbf{Bexp} \to (\mathbf{State} \to \mathbb{B})$$

$$\begin{aligned}
\mathcal{B}[\![\mathsf{true}]\!]s &:= \mathsf{true} \\
\mathcal{B}[\![A_1 = A_2]\!]s &:= \begin{cases} \mathsf{true}, & \text{kui } \mathcal{A}[\![A_1]\!]s = \mathcal{A}[\![A_2]\!]s \\ \mathsf{false}, & \text{muidu} \end{cases} \\
\mathcal{B}[\![\neg B]\!]s &:= \neg \mathcal{B}[\![B]\!]s \\
\mathcal{B}[\![B_1 \wedge B_2]\!]s &:= \mathcal{B}[\![B_1]\!]s \wedge \mathcal{B}[\![B_2]\!]s \\
&\ldots
\end{aligned}$$

Defineerimaks, mida teeb mingi programm, kasutame tema *struktuurset operatsioonilist semantikat*.

Kui S on programm ja s programmiolek, siis defineerime seose

$$\langle S, s\rangle \Longrightarrow \ldots$$

siin see $\ldots$ iseloomustab programmi ja tema olekut „ühe sammu pärast“.

Paari $\langle S, s\rangle$ nimetame *konfiguratsiooniks*.

- s — muutujate väärtused praegusel hetkel.
- S — programmiosa, mis veel läbi jooksutada tuleb.

„Üks samm“:

- Aritmeetilise avaldise väärtuse arvutamine ja selle omistamine muutujale.
- Tõeväärtusavaldise väärtuse arvutamine ja otsustamine, mida edasi täita.

$$\langle S, s\rangle \Longrightarrow \ldots$$

Siin ... on

- programmiolek, kui programmi S täitmine vajab ühteainsat sammu;
- konfiguratsioon $\langle S', s'\rangle$, kus s' on programmiolek ühe sammu pärast ja S' peale ühte sammu veel täitaolev programm.

Semantika:

$$\langle x := A, s \rangle \Longrightarrow s'$$

kus

$$s'(y) := \begin{cases} s(y), & \text{kui } y \neq x \\ \mathcal{A}[\![A]\!]s, & \text{kui } y = x \end{cases} .$$

Seda s'-i tähistame $s\big[x \mapsto \mathcal{A}[\![A]\!]s\big]$.

$$\langle skip, s \rangle \Longrightarrow s$$

Programmi $S_1; S_2$ semantika defineerime rekursiivselt. Rekursioon on üle süntaksi.

- Kui $\langle S_1, s\rangle \Longrightarrow s'$, siis $\langle S_1; S_2, s\rangle \Longrightarrow \langle S_2, s'\rangle$.
- Kui $\langle S_1, s\rangle \Longrightarrow \langle S_1', s'\rangle$, siis $\langle S_1; S_2, s\rangle \Longrightarrow \langle S_1'; S_2, s'\rangle$.

Selle „kui ..., siis ...“ paneme kirja järgmiselt:

$$\frac{\langle S_1, s\rangle \Longrightarrow s'}{\langle S_1; S_2, s\rangle \Longrightarrow \langle S_2, s'\rangle} \qquad \frac{\langle S_1, s\rangle \Longrightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Longrightarrow \langle S_1'; S_2, s'\rangle}$$

Loeme, et $S_1; S_2; S_3$ tähendab $(S_1; S_2); S_3$.

$$\langle \mathit{if}\ \ B\ \ \mathit{then}\ \ S_1\ \ \mathit{else}\ \ S_2, s\rangle \Longrightarrow \langle S_1, s\rangle \quad \text{kui } \mathcal{B}[\![B]\!]s = \mathsf{true}$$

$$\langle \mathit{if}\ \ B\ \ \mathit{then}\ \ S_1\ \ \mathit{else}\ \ S_2, s\rangle \Longrightarrow \langle S_2, s\rangle \quad \text{kui } \mathcal{B}[\![B]\!]s = \mathsf{false}$$

$$\langle \mathit{while}\ \ B\ \ \mathit{do}\ \ S, s\rangle \Longrightarrow \langle S; \mathit{while}\ \ B\ \ \mathit{do}\ \ S, s\rangle \quad \text{kui } \mathcal{B}[\![B]\!]s = \mathsf{true}$$

$$\langle \mathit{while}\ \ B\ \ \mathit{do}\ \ S, s\rangle \Longrightarrow s \quad \text{kui } \mathcal{B}[\![B]\!]s = \mathsf{false}$$

Näide. Olgu $\mathbf{Var} = \{k, m, n\}$. Vaatame programmi S_F, mis on järgmine:

$$m := 1; k := 1; \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big)$$

Olgu programmi algolek $\{k \mapsto 3, m \mapsto 5, n \mapsto 4\}$.

$$\left\langle \begin{array}{c} m := 1; k := 1; while\ k \leqslant n\ do\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 5, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} k := 1; while\ k \leqslant n\ do\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

Tõepoolest, vastavalt $S_1; S_2$ semantika definitsioonile:

$$\dfrac{\dfrac{\langle m := 1, s\rangle \Longrightarrow s\big[m \mapsto 1\big]}{\langle m := 1; k := 1, s\rangle \Longrightarrow \langle k := 1, s\big[m \mapsto 1\big]\rangle}}{\begin{array}{c}\langle m := 1; k := 1; while\ k \leqslant n\ do\ \big(m := m * k; k := k + 1\big), s\rangle \\ \Longrightarrow \\ \langle k := 1; while\ k \leqslant n\ do\ \big(m := m * k; k := k + 1\big), s\big[m \mapsto 1\big]\rangle\end{array}}$$

$$\left\langle \begin{array}{c} k := 1; while\ k \leqslant n\ do\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} while\ k \leqslant n\ do\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 1, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 1, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} m := m * k; k := k + 1; \qquad\qquad\qquad\qquad \\ \qquad\qquad \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 1, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

sest $\mathcal{B}[\![k \leqslant n]\!]\{k \mapsto 1, m \mapsto 1, n \mapsto 4\} = \mathsf{true}$.

$$\left\langle \begin{array}{r} m := m * k; k := k + 1; \qquad\qquad\qquad\qquad\qquad\qquad \\ \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 1, m \mapsto 1, n \mapsto 4\} \qquad\qquad\qquad \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} k := k + 1; \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 1, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 2, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 2, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{l} m := m * k; k := k + 1; \\ \qquad\qquad\qquad \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \qquad\qquad\qquad\qquad \{k \mapsto 2, m \mapsto 1, n \mapsto 4\} \end{array} \right\rangle$$

sest $\mathcal{B}[\![k \leqslant n]\!]\{k \mapsto 2, m \mapsto 1, n \mapsto 4\} = \text{true}$.

$$\left\langle \begin{array}{r} m := m * k; k := k + 1; \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \\ \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 2, m \mapsto 1, n \mapsto 4\} \qquad\qquad\qquad\qquad \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} k := k + 1; \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 2, m \mapsto 2, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 2, n \mapsto 4\} \end{array} \right\rangle$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 2, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} m := m * k; k := k + 1; \\ \qquad \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 2, n \mapsto 4\} \end{array} \right\rangle$$

sest $\mathcal{B}[\![k \leqslant n]\!]\{k \mapsto 3, m \mapsto 2, n \mapsto 4\} = \text{true}$.

$$\left\langle \begin{array}{l} m := m * k; k := k + 1; \\ \qquad\qquad\qquad \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \qquad\qquad \{k \mapsto 3, m \mapsto 2, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} k := k + 1; \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 6, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 4, m \mapsto 6, n \mapsto 4\} \end{array} \right\rangle$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 4, m \mapsto 6, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{l} m := m * k; k := k + 1; \\ \qquad\qquad\qquad \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \qquad\qquad\qquad\qquad \{k \mapsto 4, m \mapsto 6, n \mapsto 4\} \end{array} \right\rangle$$

sest $\mathcal{B}[\![k \leqslant n]\!]\{k \mapsto 4, m \mapsto 6, n \mapsto 4\} = \text{true}$.

$$\left\langle \begin{array}{r} m := m * k; k := k + 1; \\ \mathit{while}\ k \leqslant n\ \mathit{do}\ (m := m * k; k := k + 1) \\ \{k \mapsto 4, m \mapsto 6, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} k := k + 1; \mathit{while}\ k \leqslant n\ \mathit{do}\ (m := m * k; k := k + 1) \\ \{k \mapsto 4, m \mapsto 24, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\left\langle \begin{array}{c} \mathit{while}\ k \leqslant n\ \mathit{do}\ (m := m * k; k := k + 1) \\ \{k \mapsto 5, m \mapsto 24, n \mapsto 4\} \end{array} \right\rangle$$

$$\left\langle \begin{array}{c} \textit{while } k \leqslant n \textit{ do } \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 5, m \mapsto 24, n \mapsto 4\} \end{array} \right\rangle$$

$$\Longrightarrow$$

$$\{k \mapsto 5, m \mapsto 24, n \mapsto 4\}$$

sest $\mathcal{B}[\![k \leqslant n]\!]\{k \mapsto 5, m \mapsto 24, n \mapsto 4\} = \text{false}$.

Muuhulgas näitasime, et

$$\left\langle \begin{array}{c} m := 1; k := 1; \mathit{while}\ k \leqslant n\ \mathit{do}\ \big(m := m * k; k := k + 1\big) \\ \{k \mapsto 3, m \mapsto 5, n \mapsto 4\} \end{array} \right\rangle$$

$$\stackrel{*}{\Longrightarrow}$$

$$\{k \mapsto 5, m \mapsto 24, n \mapsto 4\}$$

Siin $\stackrel{*}{\Longrightarrow}$ tähistab seose $\Longrightarrow$ refleksiivset transitiivset sulundit.

$C \stackrel{*}{\Longrightarrow} C'$, kui leidub $n \geqslant 0$ ja $C_0, C_1, \ldots, C_n$ nii, et $C = C_0 \Longrightarrow C_1 \Longrightarrow \cdots \Longrightarrow C_n = C'$.

Programmi S *osalist korrektsust* väljendab üleskirjutus

$$\{P\}S\{Q\},$$

kus P ja Q on predikaadid programmiolekute hulgal. S.t.

$$P, Q : \mathbf{State} \to \mathbb{B} \ .$$

See kirjutus tähendab: Iga $s, s' \in \mathbf{State}$ jaoks, kus

- $P(s) = \mathsf{true}$,
- $\langle S, s\rangle \stackrel{*}{\Longrightarrow} s'$,

kehtib $Q(s') = \mathsf{true}$.

Näiteks kehtib $\{n \geqslant 0\}S_F\{m = n!\}$.

Samuti kehtib $\{\mathsf{true}\}$*while* true *do skip*$\{\mathsf{false}\}$.

Nimetusi:

- $\{P\}S\{Q\}$ — *Hoare'i kolmik*.
- P — *eeltingimus*.
- Q — *järeltingimus*.

Antud P, S ja Q. Kuidas tõestada, et $\{P\}S\{Q\}$?

Alternatiiv: antud S ja Q, kuidas leida võimalikult nõrka väidet P, et $\{P\}S\{Q\}$?

- P on nõrgem kui P' siis, kui $P' \Rightarrow P$.

Järgnevas tutvustame mõningaid reegleid, mis lubavad $\{P\}S\{Q\}$ tuletada või mingi sobiva P leida teatavate predikaatarvutuse valemite samaselt tõesusest.

$$\frac{P \Rightarrow P' \quad \{P'\}S\{Q'\} \quad Q' \Rightarrow Q}{\{P\}S\{Q\}}$$

Tõepoolest, olgu $s, s' \in \mathbf{State}$ sellised, et $P(s)$ ja $\langle S, s\rangle \stackrel{*}{\Longrightarrow} s'$. Siit järeldub üksteise järel kohe, et $P'(s)$, $Q'(s')$ ja $Q(s')$.

$$\{P\}skip\{P\}$$

Tõepoolest, kui $\langle skip, s\rangle \overset{*}{\Longrightarrow} s'$, siis $s = s'$.

Olgu Q mingi predikaat programmiolekutel, olgu $x \in \mathbf{Var}$ ja $A \in \mathbf{Aexp}$. Olgu Q_A^x predikaat, mis väärtus leitakse järgmiselt:

1. Leides $Q_A^x(s)$ väärtust, leia kõigepealt $a := \mathcal{A}[\![A]\!]s$.
2. Leia $Q(s\left[x \mapsto a\right])$ väärtus ja tagasta see.

Kui Q on antud mingi valemiga, milles esinevad muutujanimed, konstandid, aritmeetiliste ja loogiliste tehete märgid, siis on Q_A^x antud valemiga, kus Q valemis on kõikjal x asendatud A-ga.

S.t. Q_A^x on Q-st lihtsalt leitav.

$$\{Q^x_A\}x := A\{Q\}$$

Tõepoolest, kui $\langle x := A, s\rangle \stackrel{*}{\Longrightarrow} s'$, siis Q^x_A rakendamine s-le on sama, mis Q rakendamine s'-le.

Näide:

$$\{m * k = n!\}m := m * k\{m = n!\}$$

$$\frac{\{P\}S_1\{R\} \quad \{R\}S_2\{Q\}}{\{P\}S_1;S_2\{Q\}}$$

S.t. järjestikuse täitmise jaoks osalise korrektsuse tõestamisel tuleb meil leida mingi „vahepealne“ predikaat R.

Meil on ülesanne: antud S_2 ja Q, leida selline võimalikult nõrk R, nii et $\{R\}S_2\{Q\}$.

$$\frac{\{P \wedge B)\}S_1\{Q\} \qquad \{P \wedge \neg B\}S_2\{Q\}}{\{P\}\mathit{if}\ \ B\ \ \mathit{then}\ \ S_1\ \ \mathit{else}\ \ S_2\{Q\}}$$

Tõestus. Olgu s selline programmiolek, et $P(s)$, ja olgu s' selline, et $\langle \mathit{if}\ \ B\ \ \mathit{then}\ \ S_1\ \ \mathit{else}\ \ S_2, s\rangle \stackrel{*}{\Longrightarrow} s'$. Vaatame väärtust $\mathcal{B}[\![B]\!]s$.

Kui $\mathcal{B}[\![B]\!]s = \mathsf{true}$, siis

$$\langle \mathit{if}\ \ B\ \ \mathit{then}\ \ S_1\ \ \mathit{else}\ \ S_2, s\rangle \Longrightarrow \langle S_1, s\rangle \stackrel{*}{\Longrightarrow} s' \ \ .$$

Reegli esimese eelduse järgi siis $Q(s')$.

Variant $\mathcal{B}[\![B]\!]s = \mathsf{false}$ on analoogiline.

Eelmise reegli variant:

$$\frac{\{P \land B\}S_1\{Q\} \qquad P \land \neg B \Rightarrow Q}{\{P\}if\ \ B\ \ then\ \ S_1\{Q\}}$$

$$\frac{\{R \wedge B\}S\{R\}}{\{R\}while\ B\ do\ S\{R \wedge \neg B\}}$$

Predikaat R on *tsükliinvariant*. Ta kehtib tsükli keha alguses.

Väite $\{P\}while\ B\ do\ S\{Q\}$ tõestamiseks on meil lisaks tarvis

$$P \Rightarrow R$$
$$R \wedge \neg B \Rightarrow Q$$

(esimesena toodud reegel)

Sobiva tsükliinvariandi leidmine P-st, Q-st, B-st ja S-st on üldiselt mittelahenduv.

Reegli tõestus. Olgu s selline, et $R(s)$, ning olgu s' selline, et $\langle while\ \ B\ \ do\ \ S, s\rangle \overset{k}{\Longrightarrow} s'$ mingi $k \in \mathbb{N}$ jaoks. Teeme induktsiooni k järgi.

Baas. Kui $k = 1$, siis

$$\langle while\ \ B\ \ do\ \ S, s\rangle \Longrightarrow s'$$

ja seega siis $\mathcal{B}[\![B]\!]s = \mathsf{false}$ ja $s' = s$. Seega olemegi näidanud, et $R(s') \wedge (\mathcal{B}[\![B]\!]s' = \mathsf{false})$.

Samm. Olgu $k > 1$. Siis $\mathcal{B}[\![B]\!]s = \mathsf{true}$ ja

$$\langle while\ \ B\ \ do\ \ S, s\rangle \Longrightarrow \langle S; while\ \ B\ \ do\ \ S, s\rangle \overset{k-1}{\Longrightarrow} s' \ \ .$$

Peavad leiduma selline s'' ning sellised i, j, et

$$\begin{aligned} \langle S, s\rangle &\overset{i}{\Longrightarrow} s'' \\ \langle while\ \ B\ \ do\ \ S, s''\rangle &\overset{j}{\Longrightarrow} s' \\ i + j &= k - 1 \ \ . \end{aligned}$$

Reegli eeldusest siis $R(s'')$ ning induktsiooni eeldusest $R(s') \wedge (\mathcal{B}[\![B]\!]s' = \mathsf{false})$.

Tähistagu üleskirjutus

$$\{P\}S\downarrow$$

seda, et iga $s \in \mathbf{State}$ jaoks, kus $P(s)$, leidub selline $s' \in \mathbf{State}$, et $\langle S, s\rangle \stackrel{*}{\Longrightarrow} s'$.

S.t. kui s rahuldab tingimust P, siis S, rakendatuna olekule s, termineerub.

Reeglid, mis peaksid olema ilmsed:

$$\frac{P \Rightarrow P' \quad \{P'\}S \downarrow}{\{P\}S \downarrow} \qquad \{\mathsf{true}\}x := A \downarrow \qquad \{\mathsf{true}\}\mathit{skip} \downarrow$$

$$\frac{\{P\}S_1 \downarrow \quad \{P\}S_1\{R\} \quad \{R\}S_2 \downarrow}{\{P\}S_1; S_2 \downarrow} \qquad \frac{\{P \wedge B\}S_1 \downarrow \quad \{P \wedge \neg B\}S_2 \downarrow}{\{P\}\mathit{if}\ \ B\ \ \mathit{then}\ \ S_1\ \ \mathit{else}\ \ S_2 \downarrow}$$

Vaatame programmi $while\ B\ do\ S$. Olgu s programmi algolek. On kolm võimalust.

- Programm termineerub.
- Programm ei termineeru, sest iteratsioone on lõpmata palju.
- Programm ei termineeru, sest mingil iteratsioonil S ei termineeru.

while B *do* S termineerumise uurimisel võtame S-i termineerumise eelduseks.

Iteratsioonide arvu lõplikkuse näitamiseks, on meil tarvis mingit *mõõdufunktsiooni* E, mis

- seab igale programmiolekule s vastavusse naturaalarvu $E(s)$;
  - S.t. $E(s)$-l leidub vähim võimalik väärtus.
- väheneb igal iteratsioonil.
  - ... ja vähemalt 1 võrra, sest on naturaalarv.

$$\frac{\{R \wedge B\}S \downarrow \qquad \{R \wedge B \wedge (e = E)\}S\{R \wedge (E < e)\}}{\{R\}while\ \ B\ \ do\ \ S \downarrow}$$

Siin R on jälle tsükliinvariant.

e-d võib interpreteerida kui täiendavat muutujat, mida tsükli keha ei muuda.

Korrektsuse tõestust võib kirja panna järgmiselt:

1. Programmi üleskirjutus:

Järjestikusel täitmisel $(S_1; S_2; \cdots ; S_n)$ kirjuta kõik laused ühekaupa üksteise alla:

$$S_1$$

$$S_2$$

$$\dots$$

$$S_n$$

(jäta ridade vahele piisavalt ruumi, et mahuks valemeid kirjutama)

$if \;\; B \;\; then \;\; S_1 \;\; else \;\; S_2$ kirjuta üles järgmiselt:

$$\begin{array}{l} if \;\; B \;\; then \\ \quad S_1 \\ else \\ \quad S_2 \\ fi \end{array}$$

ja $while \;\; B \;\; do \;\; S_1$ järgmiselt:

$$\begin{array}{l} while \;\; B \;\; do \\ \quad S_1 \\ od \end{array}$$

2. Tsükliinvariandid:

Iga *while* B *do* S_1 *od* jaoks

- Leia tsükliinvariant R (ja mõõdufunktsioon E).
- Rea *while* B *do* ette kirjuta R.
- S_1 ette (s.t. vahetult peale rida *while* B *do*) kirjuta $R \wedge B \wedge (E = e)$.
- S_1 järele (s.t. vahetult enne rida *od*) kirjuta $R \wedge (E < e)$.
- Rea *od* järele kirjuta $R \wedge \neg B$.

3. Omistamised:

Kui lause $x := A$ järele on kirjutatud mingi Q, siis kirjuta tema ette Q^x_A.

4. *if*-laused:

Kui lause *if* B *then* S_1 *else* S_2 *fi* jaoks on

- …peale rida *fi* kirjutatud mingi Q, siis kirjuta nii S_1 kui ka S_2 järele Q.
- …enne rida *if* B *then* kirjutatud mingi P, siis kirjuta S_1 ette $P \wedge B$ ja S_2 ette $P \wedge \neg B$.
- …enne S_1 kirjutatud mingi P_1 ja enne S_2 kirjutatud mingi P_2, siis kirjuta enne rida *if* B *then* valem $(B \Rightarrow P_1) \wedge (\neg B \Rightarrow P_2)$.

Punkte 3 ja 4 tuleb täita seni kaua, kuni igal lausel on olemas eel- ja järeltingimus.

Kui kuskil on kaks valemit üksteise järel ilma vahepealse lauseta, siis tuleb näidata, et esimesest järeldub teine.

Näide (Eukleidese algoritm):

$$a \geqslant 1 \land b \geqslant 0 \land a_0 = a \land b_0 = b$$

```
if a < b then
    t := a
    a := b
    b := t
fi
while b > 0 do
    t := a
    a := b
    b := t mod a
od
```

$$a = \gcd(a_0, b_0)$$

```
while b > 0 do
    t := a
    a := b
    b := t mod a
od
```

Tsükliinvariant:

- $R \equiv \gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$;
- $E \equiv \max(b, 0)$.

$$a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b$$

```
if a < b then
    t := a
    a := b
    b := t
fi
```

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$$

```
while b > 0 do
```

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b \wedge b > 0 \wedge e = \max(b, 0)$$

```
    t := a
    a := b
    b := t mod a
```

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b \wedge e > \max(b, 0)$$

```
od
```

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b \wedge b \leqslant 0$$

$$a = \gcd(a_0, b_0)$$

Tsükli sees:

$$\gcd(a,b) = \gcd(a_0,b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b \wedge b > 0 \\ \wedge e = \max(b,0)$$

$$t := a$$

$$a := b$$

$$b := t \bmod a$$

$$\gcd(a,b) = \gcd(a_0,b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b \wedge e > \max(b,0)$$

$$\gcd(a,b)=\gcd(a_0,b_0)\wedge a\geqslant 1\wedge b\geqslant 0\wedge a\geqslant b\wedge b>0$$
$$\wedge e=\max(b,0)$$
$$\gcd(b,a \bmod b)=\gcd(a_0,b_0)\wedge b\geqslant 1\wedge a \bmod b\geqslant 0$$
$$\wedge b\geqslant a \bmod b\wedge e>\max(a \bmod b,0)$$

$t := a$

$$\gcd(b,t \bmod b)=\gcd(a_0,b_0)\wedge b\geqslant 1\wedge t \bmod b\geqslant 0$$
$$\wedge b\geqslant t \bmod b\wedge e>\max(t \bmod b,0)$$

$a := b$

$$\gcd(a,t \bmod a)=\gcd(a_0,b_0)\wedge a\geqslant 1\wedge t \bmod a\geqslant 0$$
$$\wedge a\geqslant t \bmod a\wedge e>\max(t \bmod a,0)$$

$b := t \bmod a$

$$\gcd(a,b)=\gcd(a_0,b_0)\wedge a\geqslant 1\wedge b\geqslant 0\wedge a\geqslant b\wedge e>\max(b,0)$$

if-lauses:

$$a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b$$

```
if a < b then
    t := a
    a := b
    b := t
else
    skip
fi
```

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$$

$$a \geqslant 1 \land b \geqslant 0 \land a_0 = a \land b_0 = b$$

1 *if* $a < b$ *then*

$$\gcd(b, a) = \gcd(a_0, b_0) \land b \geqslant 1 \land a \geqslant 0 \land b \geqslant a$$

2 $t := a$

$$\gcd(b, t) = \gcd(a_0, b_0) \land b \geqslant 1 \land t \geqslant 0 \land b \geqslant t$$

3 $a := b$

$$\gcd(a, t) = \gcd(a_0, b_0) \land a \geqslant 1 \land t \geqslant 0 \land a \geqslant t$$

4 $b := t$

$$\gcd(a, b) = \gcd(a_0, b_0) \land a \geqslant 1 \land b \geqslant 0 \land a \geqslant b$$

else

$$\gcd(a, b) = \gcd(a_0, b_0) \land a \geqslant 1 \land b \geqslant 0 \land a \geqslant b$$

skip

$$\gcd(a, b) = \gcd(a_0, b_0) \land a \geqslant 1 \land b \geqslant 0 \land a \geqslant b$$

5 *fi*

$$\gcd(a, b) = \gcd(a_0, b_0) \land a \geqslant 1 \land b \geqslant 0 \land a \geqslant b$$

if-lause eeltingimusest:

$$a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b$$

$$\mathit{if}\ \ a < b\ \mathit{then}$$

$$a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b \wedge a < b$$

$$\gcd(b, a) = \gcd(a_0, b_0) \wedge b \geqslant 1 \wedge a \geqslant 0 \wedge b \geqslant a$$

$$t := a$$

$$\gcd(b, t) = \gcd(a_0, b_0) \wedge b \geqslant 1 \wedge t \geqslant 0 \wedge b \geqslant t$$

$$a := b$$

$$\gcd(a, t) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge t \geqslant 0 \wedge a \geqslant t$$

$$b := t$$

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$$

$$\mathit{else}$$

$$a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b \wedge a \geqslant b$$

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$$

$$\mathit{skip}$$

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$$

$$\mathit{fi}$$

$$\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$$

Kolmes kohas esineb kaks valemit üksteise järel, meil tuleb tõestada, et esimesest järeldub teine:

- $a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b \wedge a < b \Rightarrow \gcd(b, a) = \gcd(a_0, b_0) \wedge b \geqslant 1 \wedge a \geqslant 0 \wedge b \geqslant a$;

- $a \geqslant 1 \wedge b \geqslant 0 \wedge a_0 = a \wedge b_0 = b \wedge a \geqslant b \Rightarrow \gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b$;

- $\gcd(a, b) = \gcd(a_0, b_0) \wedge a \geqslant 1 \wedge b \geqslant 0 \wedge a \geqslant b \wedge b > 0 \wedge e = \max(b, 0) \Rightarrow \gcd(b, a \bmod b) = \gcd(a_0, b_0) \wedge b \geqslant 1 \wedge a \bmod b \geqslant 0 \wedge b \geqslant a \bmod b \wedge e > \max(a \bmod b, 0)$.

Need kõik on üsna lihtsalt tõestatavad.

Järgnevate näidete jaoks toome keelde sisse massiivid, mida saab ainult lugeda.

Olgu $\mathbf{MVar}$ massiivitüüpi muutujate hulk. Programmiolek s seab lisaks igale muutujale $\mathrm{x} \in \mathbf{MVar}$ ja igale indeksile $k \in \mathbb{Z}$ vastavusse $\mathrm{x}[k]$ väärtuse $s(\mathrm{x}, k)$.

Täiendame aritmeetiliste avaldiste süntaksit: avaldis A võib olla ka $\mathrm{x}[A_1]$ mingi $\mathrm{x} \in \mathbf{MVar}$ ja aritmeetilise avaldise A_1 jaoks.

Defineerime $\mathcal{A}[\![\mathrm{x}[A_1]]\!]s = s(\mathrm{x}, \mathcal{A}[\![A_1]\!]s)$.

Ülejäänud osa semantikast jääb samaks.

Samuti jääb samaks arutelu Hoare'i kolmikute üle.

Massiivi **a** elementide (k-st l-ni) summa leidmine:

```
s := 0
i := k
while i ⩽ l do
    s := s + a[i]
    i := i + 1
od
```

Eel- ja järeltingimus:

$$k \leqslant l$$

$$\begin{array}{l}
s := 0 \\
i := k \\
\textit{while } i \leqslant l \textit{ do} \\
\qquad s := s + \mathbf{a}[i] \\
\qquad i := i + 1 \\
\textit{od}
\end{array}$$

$$s = \sum_{j=k}^{l} \mathbf{a}[j]$$

while $i \leqslant l$ *do*

  $s := s + \mathbf{a}[i]$

  $i := i + 1$

od

Tsükliinvariant:

- $R \equiv k \leqslant i \leqslant l + 1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j]$;
- $E \equiv \max(l - i + 1, 0)$.

$$k \leqslant l$$

$$s := 0$$

$$i := k$$

$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j]$$

$$\mathit{while}\ i \leqslant l\ \mathit{do}$$

$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge i \leqslant l \wedge e = \max(l-i+1, 0)$$

$$\quad s := s + \mathbf{a}[i]$$

$$\quad i := i + 1$$

$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge e > \max(l-i+1, 0)$$

$$\mathit{od}$$

$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge i > l$$

$$s = \sum_{j=k}^{l} \mathbf{a}[j]$$

$$k \leqslant l$$
$$k \leqslant k \leqslant l+1 \wedge 0 = \sum_{j=k}^{k-1} \mathbf{a}[j]$$
$$s := 0$$
$$k \leqslant k \leqslant l+1 \wedge s = \sum_{j=k}^{k-1} \mathbf{a}[j]$$
$$i := k$$
$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j]$$
$$\mathit{while}\ i \leqslant l\ \mathit{do}$$
$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge i \leqslant l \wedge e = \max(l-i+1,0)$$
$$k \leqslant i+1 \leqslant l+1 \wedge s + \mathbf{a}[i] = \sum_{j=k}^{i} \mathbf{a}[j] \wedge e > \max(l-i,0)$$
$$\quad s := s + \mathbf{a}[i]$$
$$k \leqslant i+1 \leqslant l+1 \wedge s = \sum_{j=k}^{i} \mathbf{a}[j] \wedge e > \max(l-i,0)$$
$$\quad i := i+1$$
$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge e > \max(l-i+1,0)$$
$$\mathit{od}$$
$$k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge i > l$$
$$s = \sum_{j=k}^{l} \mathbf{a}[j]$$

Seal, kus esineb kaks valemit üksteise järel, tuleb näidata, et esimesest järeldub teine:

- $k \leqslant l \Rightarrow k \leqslant k \leqslant l+1 \wedge 0 = \sum_{j=k}^{k-1} \mathbf{a}[j]$;
- $k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge i \leqslant l \wedge e = \max(l-i+1, 0) \Rightarrow k \leqslant i+1 \leqslant l+1 \wedge s + \mathbf{a}[i] = \sum_{j=k}^{i} \mathbf{a}[j] \wedge e > \max(l-i, 0)$;
- $k \leqslant i \leqslant l+1 \wedge s = \sum_{j=k}^{i-1} \mathbf{a}[j] \wedge i > l \Rightarrow s = \sum_{j=k}^{l} \mathbf{a}[j]$.