Syntaks og semantik

Lektion 10

27 marts 2007

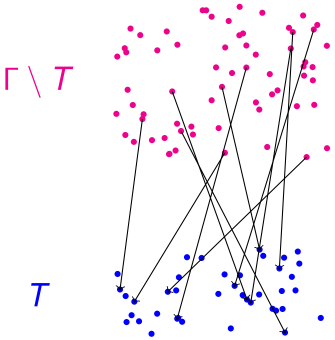
Fra sidst

1. Operationel semantik
2. Big vs. small step
3. At opskrive en operationel semantik
4. Derivationstræer

- Operationel semantik: at oversætte et *program* til et *transitionssystem*:
    - konfigurationer: programtilstande
    - transitioner: programskridt
    - slutkonfigurationer: (succesfuld) terminering af programmet
- Transitionssystemer: $(\Gamma, \to, T)$
    - konfigurationer Γ, transitioner $\to$, slutkonfigurationer T
    - slutkonfigurationer er *terminale*: $\forall \gamma \in T \nexists \gamma' \in \Gamma : \gamma \to \gamma'$
    - men ikke alle terminale konfigurationer er nødvendigvis slutkonfigurationer! – deadlock

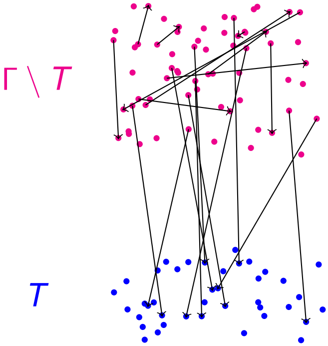
Big-step-semantik:

- at evaluere ting *i ét hug*
- transitioner fra konfigurationer til slutkonfigurationer



Small-step-semantik:

- at evaluere ting *ét skridt ad gangen*
- transitioner fra konfigurationer til konfigurationer og til slutkonfigurationer



At opskrive en operationel semantik for et programmeringssprog:

1. abstrakt syntaks
   - syntaktiske kategorier

$$
\begin{aligned}
n &\in \textbf{Num} && \text{– Numeraler}\\
x &\in \textbf{Var} && \text{– Variable}\\
a &\in \textbf{Aud} && \text{– Aritmetiske udtryk}\\
b &\in \textbf{Bud} && \text{– Boolske udtryk}\\
S &\in \textbf{Kom} && \text{– Kommandoer}
\end{aligned}
$$

   - opbygningsregler

$$
\begin{aligned}
S &::= x{:=}a \mid \texttt{skip} \mid S_1 ; S_2 \mid \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\\
&\qquad \mid \texttt{while}\ b\ \texttt{do}\ S\\
b &::= a_1 = a_2 \mid a_1 < a_2 \mid \neg b_1 \mid b_1 \wedge b_2 \mid (\ b_1\ )\\
a &::= n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2 \mid (\ a_1\ )
\end{aligned}
$$

At opskrive en operationel semantik for et programmeringssprog:

1. abstrakt syntaks
   - syntaktiske kategorier
   - opbygningsregler
2. semantiske mængder og hjælpefunktioner
   - værdier af numeraler er elementer i $\mathbb{Z}$
   - funktionen $\mathcal{N} : \mathbf{Num} \to \mathbb{Z}$ giver værdien af en numeral

At opskrive en operationel semantik for et programmeringssprog:

1. abstrakt syntaks
   - syntaktiske kategorier
   - opbygningsregler
2. semantiske mængder og hjælpefunktioner
3. transitionssystem(er)
   - konfigurationer og slutkonfigurationer

     $\Gamma = \textbf{Aud} \cup \mathbb{Z},\ T = \mathbb{Z}$
   - transitionsrelationen givet ved *transitionsregler*

f.eks. $\dfrac{a_1 \to v_1 \quad a_2 \to v_2}{a_1 + a_2 \to v}$ hvor $v = v_1 + v_2$

For at vise at en bestemt transition findes i en operationel semantik, konstrueres et derivationstræ:

$$\dfrac{\dfrac{\dfrac{\underline{2} \longrightarrow 2 \qquad \underline{4} \longrightarrow 4}{\underline{2} + \underline{4} \longrightarrow ?}}{(\underline{2} + \underline{4}) \longrightarrow ?} \qquad \dfrac{\dfrac{\underline{6} \longrightarrow 6 \qquad \underline{1} \longrightarrow 1}{\underline{6} + \underline{1} \longrightarrow ?}}{(\underline{6} + \underline{1}) \longrightarrow ?}}{(\underline{2} + \underline{4}) \star (\underline{6} + \underline{1}) \longrightarrow ?}$$

- aksiomer i bladene
- knude k har sønner $p_1, p_2, \ldots, p_n$ hvis og kun hvis der er en transitionsregel $$\frac{p_1, p_2, \ldots, p_n}{k}$$
- *mekanisk* proces $\Rightarrow$ automatisering!

Operationelle semantikker for **Bims**

Mål: Transitionssystem hvori transitioner beskriver udførelser af **Bims**-kommandoer.

Hvad skal *konfigurationerne* være?

- konfiguration $=$ *programtilstand*
- programmers opførsel kan afhænge af værdier af variable

$\Rightarrow$ programtilstand $=$ de kommandoer vi mangler at udføre
$+$ værdier af alle variable

Definition 4.1: En tilstand er en *partiel* funktion **Var** $\to \mathbb{Z}$.
Definition 4.3: Mængden af alle tilstande kaldes **Tilstande**.

Dvs. **Tilstande** $=$ **Var** $\rightharpoonup \mathbb{Z}$. $\longleftarrow$ mængden af alle *partielle* funktioner fra **Var** til $\mathbb{Z}$

– konfigurationerne vil være *par af kommandoer og tilstande*:
$\Gamma =$ **Kom** $\times$ **Tilstande**

Aritmetiske udtryk med variable:

$$\textbf{Aud:} \qquad a ::= n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2 \mid (\, a_1 \,)$$

- big-step-semantik
- semantikken *afhænger* af tilstanden, men *ændrer den ikke*

$\Rightarrow$ konfigurationer $\Gamma = \textbf{Aud} \cup \mathbb{Z}$ (som før!), men transitionssystemet afhænger af tilstanden!

- transitioner skrives $s \vdash a \rightarrow_a v$: i tilstand s kan a evaluere til v
- slutkonfigurationer $T = \mathbb{Z}$ (også som før)

$$[\text{plus}_{\text{bss}}] \quad \frac{s \vdash a_1 \to_a v_1 \quad s \vdash a_2 \to_a v_2}{s \vdash a_1 + a_2 \to_a v} \quad \text{hvor } v = v_1 + v_2$$

$$[\text{minus}_{\text{bss}}] \quad \frac{s \vdash a_1 \to_a v_1 \quad s \vdash a_2 \to_a v_2}{s \vdash a_1 - a_2 \to_a v} \quad \text{hvor } v = v_1 - v_2$$

$$[\text{mult}_{\text{bss}}] \quad \frac{s \vdash a_1 \to_a v_1 \quad s \vdash a_2 \to_a v_2}{s \vdash a_1 * a_2 \to_a v} \quad \text{hvor } v = v_1 \cdot v_2$$

$$[\text{parent}_{\text{bss}}] \quad \frac{s \vdash a_1 \to_a v_1}{s \vdash (\, a_1 \,) \to_a v_1}$$

$$[\text{num}_{\text{bss}}] \quad s \vdash n \to_a v \quad \text{hvis} \quad \mathcal{N}[\![n]\!] = v$$

$$[\text{var}_{\text{bss}}] \quad s \vdash x \to_a v \quad \text{hvis} \quad s(x) = v$$

- syntaksdirigerede: ethvert sammensat element fra syntaksen optræder som konklusion i en transitionsregel, ethvert basiselement som aksiom
- kompositionelle: præmisserne i en regel udtaler sig om de *umiddelbare bestanddele* af elementet i konklusionen

Boolske udtryk:

$$\textbf{Bud:} \qquad b ::= a_1 = a_2 \mid a_1 < a_2 \mid \neg b_1 \mid b_1 \wedge b_2 \mid (\, b_1 \,)$$

- samme big-step-semantik som før
- bortset fra at alle transitioner nu er af formen $s \vdash b \rightarrow_b tt$ eller $s \vdash b \rightarrow_b ff$
- det gider vi ikke vise igen …

Kommandoer i **Bims**:

$$S ::= x{:=}a \mid \texttt{skip} \mid S_1 ; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \mid \texttt{while } b \texttt{ do } S$$

- kommandoer kan ændre tilstanden (f.x. kommandoen `x:=2`)

$\Rightarrow$ skal have tilstanden med i konfigurationerne

- dvs. konfigurationer $\Gamma = \textbf{Kom} \times \textbf{Tilstande} \cup \textbf{Tilstande}$ og slutkonfigurationer $T = \textbf{Tilstande}$
- skrives $\langle S, s \rangle$ (S kommando, s tilstand)
- (og transitionsrelationen $\to$ defineres ved transitionsregler; coming up)
- at ændre en tilstand: Definition 4.4: Lad $s \in \textbf{Tilstande}$, $x \in \textbf{Var}$ og $v \in \mathbb{Z}$. Den opdaterede tilstand $s[x \mapsto v]$ er givet ved

$$s[x \mapsto v](y) = \begin{cases} s(y) & \text{hvis } y \neq x \\ v & \text{hvis } y = x \end{cases}$$

[ass_{bss}] $\langle x := a, s\rangle \to s[x \mapsto v]$ hvor $s \vdash a \to_a v$

[skip_{bss}] $\langle \texttt{skip}, s\rangle \to s$

[comp_{bss}] $$\frac{\langle S_1, s\rangle \to s'' \quad \langle S_2, s''\rangle \to s'}{\langle S_1; S_2, s\rangle \to s'}$$

[$\text{if-sand}_{\text{bss}}$] $$\frac{\langle S_1, s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ , s\rangle \to s'}$$
hvis $s \vdash b \to_b tt$

[$\text{if-falsk}_{\text{bss}}$] $$\frac{\langle S_2, s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2\ , s\rangle \to s'}$$
hvis $s \vdash b \to_b ff$

[$\text{while-sand}_{\text{bss}}$] $$\frac{\langle S, s\rangle \to s'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s''\rangle \to s'}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to s'}$$
hvis $s \vdash b \to_b tt$

[$\text{while-falsk}_{\text{bss}}$] $\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to s$ hvis $s \vdash b \to_b ff$

$$[\text{while-sand}_{\text{bss}}] \quad \frac{\langle S, s\rangle \to s'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s''\rangle \to s'}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \to s'}$$

$$\text{hvis} \quad s \vdash b \to_b tt$$

Dén regel er ikke kompositionel: præmissen indeholder ikke kun umiddelbare bestanddele af konklusionens syntakselement

- fordi `while`-løkker er *rekursive*
- reglen skal anvendes *indtil b bliver falsk*
- ellers: *uendelig løkke* – ikke-terminering
- fikspunkt-teori!

Eksempel: Givet kommandoen

$$S = \texttt{i:=6; while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}$$

og tilstanden s ved $s(\texttt{x}) = 5$, konstruer et derivationstræ for at finde en transition $\langle S, s\rangle \to s'$:

1. $$\frac{\langle \texttt{i:=6}, s\rangle \to s_2 \quad \langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_2\rangle \to s'}{\langle \texttt{i:=6; while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s\rangle \to s'}$$

2. $\langle \texttt{i:=6}, s\rangle \to s[\texttt{i} \mapsto 6]$, fordi $s \vdash \texttt{6} \to_a 6$. Så $s_2 = s[\texttt{i} \mapsto 6]$.

3. $$\frac{\begin{array}{c}\langle \texttt{x:=x+i; i:=i-2}, s_2\rangle \to s_3 \\ \langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_3\rangle \to s'\end{array}}{\langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_2\rangle \to s'}$$

   fordi $s_2 \vdash \texttt{i}\neq\texttt{0} \to$

4. $$\frac{\langle \texttt{x:=x+i}, s_2\rangle \to s_4 \quad \langle \texttt{i:=i-2}, s_4\rangle \to s_3}{\langle \texttt{x:=x+i; i:=i-2}, s_2\rangle \to s_3}$$

5. $\langle \texttt{x:=x+i}, s_2 \rangle \to s_2[\texttt{x} \mapsto 11]$, fordi $s_2 \vdash \texttt{x+i} \to_a 11$ (anvend [plus$_{\text{bss}}$]!)
$\Rightarrow s_4 = s_2[\texttt{x} \mapsto 11] = s[\texttt{i} \mapsto 6, \texttt{x} \mapsto 11]$
6. $\langle \texttt{i:=i-2}, s_4 \rangle \to s_4[\texttt{i} \mapsto 4]$, fordi $s_4 \vdash \texttt{i-2} \to_a 4$ (anvend [plus$_{\text{bss}}$]!)
$\Rightarrow s_3 = s_4[\texttt{i} \mapsto 4] = s[\texttt{i} \mapsto 4, \texttt{x} \mapsto 11]$
7. $$\frac{\langle \texttt{x:=x+i; i:=i-2}, s_3 \rangle \to s_5 \qquad \langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_5 \rangle \to s'}{\langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_3 \rangle \to s'}$$ fordi $s_3 \vdash \texttt{i}\neq\texttt{0} \to$
8. ...
9. ...
10. ... $s_5 = s[\texttt{i} \mapsto 2, \texttt{x} \mapsto 15]$
11. $$\frac{\langle \texttt{x:=x+i; i:=i-2}, s_5 \rangle \to s_7 \qquad \langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_7 \rangle \to s'}{\langle \texttt{while i}\neq\texttt{0 do (x:=x+i; i:=i-2)}, s_5 \rangle \to s'}$$ fordi $s_5 \vdash \texttt{i}\neq\texttt{0} \to$

12. ...
13. ...
14. ... $s_7 = s[\mathtt{i} \mapsto 0, \mathtt{x} \mapsto 17]$
15. $\langle \mathtt{while\ i{\neq}0\ do\ (x:=x+i;\ i:=i-2)}, s_7 \rangle \to s_7$, fordi $s_7 \vdash \mathtt{i{\neq}0} \to_b$

$\Rightarrow$ $s' = s_7 = s[\mathtt{i} \mapsto 0, \mathtt{x} \mapsto 17]$, dvs.

$$\langle \mathtt{i:=6;\ while\ i{\neq}0\ do\ (x:=x+i;\ i:=i-2)}, s \rangle \to s[\mathtt{i} \mapsto 0, \mathtt{x} \mapsto 17$$

- at konstruere derivationstræer = *kedeligt*, mekanisk

$\Rightarrow$ automatisering $\Rightarrow$ fortolker!

Definition: Givet $S \in \textbf{Kom}$ og $s \in \textbf{Tilstande}$:

- S siges at terminere fra s hvis der findes $s' \in \textbf{Tilstande}$ så $\langle S, s \rangle \to s'$.
- S siges at gå i uendelig løkke på s hvis S ikke terminerer fra s.
- S terminerer altid hvis S terminerer fra alle $s \in \textbf{Tilstande}$.
- S går altid i uendelig løkke hvis S går i uendelig løkke på alle $s \in \textbf{Tilstande}$.

Opgave 4.8: Vis at $S =$ `while 0=0 do skip` altid går i uendelig løkke.

- (Husk: **Tilstande** $=$ **Var** $\rightharpoonup \mathbb{Z}$)
- konfigurationer $\Gamma =$ **Kom** $\times$ **Tilstande** $\cup$ **Tilstande**, slutkonfigurationer $T =$ **Tilstande**
- transitionsregler for $\Rightarrow$ coming up
- transition $\langle S, s\rangle \Rightarrow s'$: terminering i s' *efter ét skridt*
- transition $\langle S, s\rangle \Rightarrow \langle S', s'\rangle$: efter *ét skridt* kommer vi fra S i tilstand s til S' i tilstand s'

[ass$_{sss}$] $\langle x := a, s\rangle \Rightarrow s[x \mapsto v]$ hvor $s \vdash a \rightarrow_a v$

[skip$_{sss}$] $\langle \texttt{skip}, s\rangle \Rightarrow s$

[comp-1$_{sss}$] $$\frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s'\rangle}$$

[comp-2$_{sss}$] $$\frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

[if-sand$_{sss}$] $\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \Rightarrow \langle S_1, s\rangle$ hvis $s \vdash b \rightarrow_b tt$

[if-falsk$_{sss}$] $\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2, s\rangle \Rightarrow \langle S_2, s\rangle$ hvis $s \vdash b \rightarrow_b ff$

[while$_{sss}$] $\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \Rightarrow$
$\langle \texttt{if}\ b\ \texttt{then}\ (S; \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, s\rangle$

– reglen for `while`-løkken indeholder igen rekursion

Ikke-terminering svarer nu til *uendelige transitionsfølger*:

$$\langle \texttt{while 0=0 do skip}, s\rangle \stackrel{3}{\Rightarrow} \langle \texttt{while 0=0 do skip}, s\rangle \stackrel{3}{\Rightarrow} \dots$$

(eller til *løkker i transitionssystemet*!)

Definition: Givet $S \in$ **Kom** og $s \in$ **Tilstande**:

- S siges at terminere fra s hvis der findes $s' \in$ **Tilstande** så $\langle S, s\rangle \stackrel{*}{\Rightarrow} s'$.
- S siges at gå i uendelig løkke på s hvis der findes en uendelig transitionsfølge

$$\langle S, s\rangle \Rightarrow \langle S_1, s_1\rangle \Rightarrow \langle S_2, s_2\rangle \Rightarrow \dots$$

Sætning 4.11 / 4.13 : Lad $S \in$ **Kom** og $s, s' \in$ **Tilstande.** Da har vi $\langle S, s\rangle \to s'$ hvis og kun hvis $\langle S, s\rangle \overset{*}{\Rightarrow} s'$.

– dvs. kommandoen S terminerer fra tilstand s i tilstand s' i big-step-semantikken hvis og kun hvis den gør det i small-step-semantikken.

– dvs. big-step- og small-step-semantikken er ækvivalent.

Vi viser her sætning 4.11 med tilhørende lemma 4.12. Beviserne for sætning 4.13 og lemma 4.14 springes over.

Lemma 4.12: Lad $S_1, S_2 \in \textbf{Kom}$ og $s, s' \in \textbf{Tilstande}$. Hvis $\langle S_1, s\rangle \overset{*}{\Rightarrow} s'$ så $\langle S_1; S_2, s\rangle \overset{*}{\Rightarrow} \langle S_2, s'\rangle$.

Bevis ved *induktion* i transitionsfølgers længde:
(Bemærk forskellen fra bogens bevis!)

1. Lad $\langle S_1, s\rangle \overset{*}{\Rightarrow} s'$, dvs. $\langle S_1, s\rangle \overset{k}{\Rightarrow} s'$ for et eller andet $k \in \mathbb{N}_0$.
2. Vi må have $k \neq 0$, da $\langle S_1, s\rangle \neq s'$. ($\overset{0}{\Rightarrow}$ er defineret som $=$!)
3. *Induktionsbasis:* Lad $k = 1$. Reglen [comp-2_{sss}] giver at $\langle S_1, s\rangle \Rightarrow s'$ medfører $\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle$. ✓
4. *Induktionsskridt:* Lad $k \geq 1$ og antag at vi har vist påstanden for alle transitionsfølger af længde k.
5. Lad $\langle S_1, s\rangle \overset{k+1}{\Rightarrow} s'$. Vi må have $S_1' \in \textbf{Kom}$ og $s'' \in \textbf{Tilstande}$ med $\langle S_1, s\rangle \Rightarrow \langle S_1', s''\rangle \overset{k}{\Rightarrow} s'$.
6. Pga. induktionsantagelsen kan vi konkludere $\langle S_1'; S_2, s''\rangle \overset{k}{\Rightarrow} \langle S_2, s'\rangle$. Og med [comp-$1_{sss}$] har vi $\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle$. Dvs.

$$\langle S_1; S_2, s\rangle \Rightarrow \langle S_1'; S_2, s''\rangle \overset{k}{\Rightarrow} \langle S_2, s'\rangle \quad ✓$$

Sætning 4.11: Lad $S \in$ **Kom** og $s, s' \in$ **Tilstande.** Hvis $\langle S, s\rangle \to s'$ så $\langle S, s\rangle \stackrel{*}{\Rightarrow} s'$.

Bevis ved *transitionsinduktion*:
Vis at egenskaben gælder for alle aksiomer, og at den bevares ved opbygning af derivationstræer.

[ass$_{\text{bss}}$]: Hvis $\langle S, s\rangle \to s'$ kommer fra [ass$_{\text{bss}}$], må vi have $S = x := a$, $s \vdash a \to_a v$ og $s' = s[x \mapsto v]$ for nogle x, a og v. [ass$_{\text{sss}}$] medfører $\langle S, s\rangle \Rightarrow s'$ $\checkmark$

[skip$_{\text{bss}}$]: Hvis $\langle S, s\rangle \to s'$ kommer fra [skip$_{\text{bss}}$], må vi have $S = \texttt{skip}$ og $s' = s$. [skip$_{\text{sss}}$] medfører $\langle S, s\rangle \Rightarrow s'$ $\checkmark$

[comp$_{\text{bss}}$]: Hvis $\langle S_1; S_2, s\rangle \to s''$ kommer fra reglen

$$\frac{\langle S_1, s\rangle \to s' \quad \langle S_2, s'\rangle \to s''}{\langle S_1; S_2, s\rangle \to s''}$$

og vores påstand gælder for præmisserne, må vi have $\langle S_1, s\rangle \stackrel{*}{\Rightarrow} s'$ og $\langle S_2, s'\rangle \stackrel{*}{\Rightarrow} s''$.
Med lemma 4.12 bliver den første til $\langle S_1; S_2, s\rangle \stackrel{*}{\Rightarrow} \langle S_2, s'\rangle$, sammensæt $\Rightarrow \checkmark$

[if-falsk$_{bss}$]: Hvis $\langle \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 , s\rangle \to s'$ kommer fra reglen

$$\frac{\langle S_2, s\rangle \to s'}{\langle \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 , s\rangle \to s'} \qquad s \vdash b \to_b ff$$

giver [if-falsk$_{sss}$] transitionen

$$\langle \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 , s\rangle \Rightarrow \langle S_2, s\rangle$$

Med induktionsantagelsen har vi $\langle S_2, s\rangle \stackrel{*}{\Rightarrow} s'$, sammensæt $\Rightarrow$ ✓

[if-sand$_{bss}$]: tilsvarende

[while-sand$_{bss}$]: Hvis $\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \rightarrow s'$ kommer fra reglen

$$\frac{\langle S, s\rangle \rightarrow s'' \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s''\rangle \rightarrow s'}{\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \rightarrow s'} \quad s \vdash b \rightarrow_b tt$$

kan vi per antagelse konkludere at

$$\langle S, s\rangle \stackrel{*}{\Rightarrow} s'' \quad \text{og} \quad \langle \texttt{while}\ b\ \texttt{do}\ S, s''\rangle \stackrel{*}{\Rightarrow} s'$$

dvs. med lemma 4.12: $\langle S; \texttt{while}\ b\ \texttt{do}\ S, s\rangle \stackrel{*}{\Rightarrow} s'$
Og med [if-sand$_{sss}$] og [while$_{sss}$] har vi så

$$\begin{aligned}
&\langle \texttt{while}\ b\ \texttt{do}\ S, s\rangle \\
&\Rightarrow \langle \texttt{if}\ b\ \texttt{then}\ (S;\ \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else}\ \texttt{skip}, s\rangle \\
&\Rightarrow \langle S; \texttt{while}\ b\ \texttt{do}\ S, s\rangle \\
&\stackrel{*}{\Rightarrow} s'
\end{aligned}$$

[while-falsk$_{bss}$]: tilsvarende

Færdig!