CONTROL STRUCTURE

Slides Courtesy of Radu Mardare

§1. Loop constructs

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

We extend the syntax of Stm with loop constructs.

§§1.1. Repeat-loops

Stm: $S ::= \ldots \mid \text{repeat } S \text{ until } b$

Informal semantics: the loop body S is executed, then the condition b is checked $\longrightarrow$ if b evaluates to $\top \Rightarrow$ leave the loop
$\longrightarrow$ if b evaluates to $\perp \Rightarrow$ execute the loop again

Exercise: write a BS-semantics for repeat-loops.

BS-semantics for repeat-loops

$$[\text{Repeat-T}_{BS}]\ \frac{\langle S, s\rangle \longrightarrow s'}{\langle \texttt{repeat}\ S\ \texttt{until}\ b, s\rangle \longrightarrow s'} \quad s' \vdash b \longrightarrow_B \top$$

$$[\text{Repeat-}\bot_{BS}]\ \frac{\langle S, s\rangle \longrightarrow s' \quad \langle \texttt{repeat}\ S\ \texttt{until}\ b, s'\rangle \longrightarrow s''}{\langle \texttt{repeat}\ S\ \texttt{until}\ b, s\rangle \longrightarrow s''} \quad s' \vdash b \longrightarrow_B \bot$$

Notice the side conditions: b is not evaluated in the current state s, but in the next state s', i.e., after S is executed.

Exercise: Build a derivation tree and find the final state for the transition

$$\langle \texttt{repeat}\ y := y * x;\ x := x - \underline{1}\ \texttt{until}\ x = \underline{1}, s\rangle \longrightarrow s'$$

where $s = [x \mapsto 4, y \mapsto 1]$

Observation: we do not need to add the repeat-loop as a new syntactic construct. Bims is already sufficiently expressive to encode it.

Theorem: For any $s \in \mathbf{States}$,

$$\langle \text{repeat } S \text{ until } b, s\rangle \rightarrow s' \quad \text{iff} \quad \langle S; \text{while } \neg b \text{ do } S, s\rangle \rightarrow s'$$

Proof: We prove firstly that if we have $\langle \text{repeat } S \text{ until } b, s\rangle \rightarrow s'$, then we also have that $\langle S; \text{while } \neg b \text{ do } S, s\rangle \rightarrow s'$.

} induction on the size of the derivation tree of the hypothesis

Secondly, we prove that if $\langle S; \text{while } \neg b \text{ do } S, s\rangle \rightarrow s'$ then we also have that $\langle \text{repeat } S \text{ until } b, s\rangle \rightarrow s'$

} induction on the size of the derivation tree of the hypothesis

Homework: Study this proof in Hüttel's book, pg. 67 – 69

Problem: Give an SS-semantics for repeat-loops.

§§ 1.2. For-loops

$$\mathbf{Stm} \qquad S ::= \ldots \mid \texttt{for } x := n_1 \texttt{ to } n_2 \texttt{ do } S \,, \text{ where } n_1, n_2 \in \mathbf{Num}$$

Informal semantics: the initial value of x is $v_1 = \mathcal{N}[\![n_1]\!]$.
If $v_1 \leq v_2 = \mathcal{N}[\![n_2]\!]$, we execute S and increment the value of x by 1. We continue until $v_1 > v_2$. After the for-loop has terminated, the variable x has the value $v_2 + 1$.

BS-semantics for the for-loops

$$[\mathrm{For}-1_{BS}]\quad \frac{\langle S, s[x \mapsto v_1]\rangle \to s'' \qquad \langle \texttt{for } x := n_1' \texttt{ to } n_2 \texttt{ do } S, s''\rangle \to s'}{\langle \texttt{for } x := n_1 \texttt{ to } n_2 \texttt{ do } S, s\rangle \to s'}$$

if $v_1 \leq v_2$ where $v_i = \mathcal{N}[\![n_i]\!]$ and $n_1' = \mathcal{N}^{-1}(v_1+1)$

$$[\mathrm{For}-2_{BS}]\quad \langle \texttt{for } x := n_1 \texttt{ to } n_2 \texttt{ do } S, s\rangle \to s[x \mapsto v_1]$$

if $v_1 > v_2$ where $v_i = \mathcal{N}[\![n_i]\!]$

We have the semantic function $\mathcal{N} : \mathbf{Num} \to \mathbb{Z}$

$$\mathcal{N}[\![\underline{3}]\!] = 3, \quad \mathcal{N}[\![\underline{5}]\!] = 5, \quad \mathcal{N}[\![\underline{0}]\!] = 0$$

We consider its inverse $\mathcal{N}^{-1} : \mathbb{Z} \to \mathbf{Num}$

$$\mathcal{N}^{-1}(3) = \underline{3}, \quad \mathcal{N}^{-1}(1+4) = \underline{5}, \quad \mathcal{N}^{-1}(2\cdot 3) = \underline{6}$$

Problem: Give an SS-semantics for the for-loops.

$$[\text{For}-1_{ss}]\ \frac{\langle S, s[x \mapsto v_1]\rangle \Rightarrow \langle S', s'\rangle}{\langle \texttt{for } x := n_1 \texttt{ to } n_2 \texttt{ do } S, s\rangle \Rightarrow \langle S'; \texttt{for } x := n_1' \texttt{ to } n_2 \texttt{ do } S, s'\rangle}$$

if $v_1 \leq v_2$, $v_i = \mathcal{N}[\![n_i]\!]$, $n_1' = \mathcal{N}^{-1}(v_1+1)$

$$[\text{For}-2_{ss}]\ \frac{\langle S, s[x \mapsto v_1]\rangle \Rightarrow s'}{\langle \texttt{for } x := n_1 \texttt{ to } n_2 \texttt{ do } S, s\rangle \Rightarrow \langle \texttt{for } x := n_1' \texttt{ to } n_2 \texttt{ do } S, s'\rangle}$$

if $v_1 \leq v_2$, $v_i = \mathcal{N}[\![n_i]\!]$, $n_1' = \mathcal{N}^{-1}(v_1+1)$

$$[\text{For}-3_{ss}]\ \langle \texttt{for } x := n_1 \texttt{ to } n_2 \texttt{ do } S, s\rangle \Rightarrow s[x \mapsto v_1]$$

if $v_1 > v_2$, $v_i = \mathcal{N}[\![n_i]\!]$

Problem: Consider the more general version of for-loops

$$\text{for } x := a_1 \text{ to } a_2 \text{ do } S$$

Propose a BS and an SS-semantics.

Hint: Use "for $x := n_1$ to n_2 do S" as a more basic syntactic construct.

[Ext. For−BS] $$\frac{\langle \text{for } x := n_1 \text{ to } n_2 \text{ do } S, s\rangle \longrightarrow s'}{\langle \text{for } x := a_1 \text{ to } a_2 \text{ do } S, s\rangle \longrightarrow s'} \quad s \vdash a_i \rightarrow_A v_i \,,\; v_i = \mathcal{N}[\![n_i]\!]$$

[Ext. For−1_{SS}] $$\frac{\langle \text{for } x := n_1 \text{ to } n_2 \text{ do } S, s\rangle \Longrightarrow \langle S', s'\rangle}{\langle \text{for } x := a_1 \text{ to } a_2 \text{ do } S, s\rangle \Longrightarrow \langle S', s'\rangle} \quad s \vdash a_i \rightarrow_A v_i \,,\; v_i = \mathcal{N}[\![n_i]\!]$$

[Ext. For−2_{SS}] $$\frac{\langle \text{for } x := n_1 \text{ to } n_2 \text{ do } S, s\rangle \Longrightarrow s'}{\langle \text{for } x := a_1 \text{ to } a_2 \text{ do } S, s\rangle \Longrightarrow s'} \quad s \vdash a_i \rightarrow_A v_i \,,\; v_i = \mathcal{N}[\![n_i]\!]$$

§ 2. Semantic equivalence

Semantic equivalence = a formal version of the notion of "having the same behaviour".

* two different implementations of the same underlying algorithm
  - if the two have the same behaviour, we have deeper reasons to believe that the algorithm has been correctly implemented.

* an old an a new (optimized) version of a program
  - we want that the two have the same behaviour

* a program written in some high-level language and a machine-code version of it obtained by compiling our high-level program
  - if the compiler is correct, the two must have the same behaviours.

Definition [Big-Step Semantic equivalence]:

Let $(\Gamma, \rightarrow, F)$ be the transition system for our BS-semantics of Bims. We say that two statements $S_1, S_2 \in \mathbf{Stm}$ are <u>semantically-equivalent</u>, written

$$S_1 \sim_{BS} S_2$$

iff for all states $s, s' \in \mathbf{States}$,

$$\langle S_1, s\rangle \rightarrow s' \quad \text{iff} \quad \langle S_2, s\rangle \rightarrow s'$$

Observe that our previous Theorem stated that

$$\text{repeat } S \text{ until } b \sim_{BS} S; \text{while } \neg b \text{ do } S$$

Theorem: $\sim_{BS}$ is an equivalence relation.

Proof: – exercise

Definition [Small-Step Semantic Equivalence]:

Let $(\Gamma, \Rightarrow, F)$ be the transition system given by the SS-semantics of Bims. We say that two statements $S_1, S_2 \in \mathbf{Stm}$ are semantically-equivalent in SS, written

$$S_1 \sim_{SS} S_2$$

iff for all states $s, s' \in \mathbf{States}$ and all statements $S' \in \mathbf{Stm}$,

$$\langle S_1, s\rangle \Rightarrow \langle S', s'\rangle \text{ iff } \langle S_2, s\rangle \Rightarrow \langle S', s'\rangle$$

We say that S_1 and S_2 are semantically-equivalent to termination written

$$S_1 \sim^{*}_{SS} S_2$$

iff for all $s, s' \in \mathbf{States}$,

$$\langle S_1, s\rangle \Rightarrow^{*} s' \text{ iff } \langle S_2, s\rangle \Rightarrow^{*} s'$$

Theorem: $\sim_{SS}$ is an equivalence relation.

Theorem: $\sim^*_{SS}$ is an equivalence relation.

Theorem: In $\mathbb{B}$ims, for arbitrary statements $S_1, S_2 \in \mathbb{S}tm$ we have that

$$S_1 \sim_{BS} S_2 \text{ iff } S_1 \sim^*_{SS} S_2.$$

Proof We have proven during the previous lecture that for arbitrary $S \in \mathbb{S}tm$ and $s, s' \in \mathbb{S}tates$, $\langle S, s\rangle \rightarrow s'$ iff $\langle S, s\rangle \Rightarrow^* s'$ (*)

($\Rightarrow$) Supp. $S_1 \sim_{BS} S_2$. Then for any $s, s' \in \mathbb{S}tates$,

$$\langle S_1, s\rangle \rightarrow s' \text{ iff } \langle S_2, s\rangle \rightarrow s'$$

Applying (*): $\langle S_1, s\rangle \rightarrow s'$ iff $\langle S_1, s\rangle \Rightarrow^* s'$

$\langle S_2, s\rangle \rightarrow s'$ iff $\langle S_2, s\rangle \Rightarrow^* s'$ $\Rightarrow$

$\Rightarrow$ $\langle S_1, s\rangle \Rightarrow^* s'$ iff $\langle S_2, s\rangle \Rightarrow^* s'$

Hence, $S_1 \sim^*_{SS} S_2$

Theorem: $\sim_{SS}$ is an equivalence relation.

Theorem: $\sim^*_{SS}$ is an equivalence relation.

Theorem: In Bims, for arbitrary statements $S_1, S_2 \in \mathbf{Stm}$ we have that

$$S_1 \sim_{BS} S_2 \text{ iff } S_1 \sim^*_{SS} S_2.$$

Proof We have proven during the previous lecture that for arbitrary $S \in \mathbf{Stm}$ and $s, s' \in \mathbf{States}$, $\langle S, s\rangle \longrightarrow s'$ iff $\langle S, s\rangle \Longrightarrow^* s'$ (*)

($\Leftarrow$) Supp. $S_1 \sim^*_{SS} S_2$. Then, for any $s, s' \in \mathbf{States}$,

$$\left.\begin{array}{r} \langle S_1, s\rangle \Longrightarrow^* s' \text{ iff } \langle S_2, s\rangle \Longrightarrow^* s' \\ \text{Applying(*): } \langle S_1, s\rangle \Longrightarrow^* s' \text{ iff } \langle S_1, s\rangle \longrightarrow s' \\ \langle S_2, s\rangle \Longrightarrow^* s' \text{ iff } \langle S_2, s\rangle \longrightarrow s' \end{array}\right\} \Longrightarrow$$

$$\Longrightarrow \langle S_1, s\rangle \longrightarrow s' \text{ iff } \langle S_2, s\rangle \longrightarrow s'$$

Hence, $S_1 \sim_{BS} S_2$

§3. Abnormal termination

Stm $\quad S ::= \ldots \mid \text{abort}$

Informal semantics: abort stops the execution of a program

Example: Suppose that we extent the Aexp to include division of integers.
we can use abort to use correctly the division operation:

$$\text{if } \neg(x=0) \text{ then } x := \underline{25}/x \text{ else abort}$$

$\langle \text{abort}, s\rangle$ has no transition
- there is no BS-rule for abort
- there is no SS-rule for abort

Observe that

(i) $\text{abort} \not\sim_{BS} \text{skip}$ (ii) $\text{abort} \not\sim_{SS} \text{skip}$ (iii) $\text{abort} \not\sim^{*}_{SS} \text{skip}$

What is the relation between "abort" and "while 0=0 do skip" regarding the BS-semantics?

$$\text{while } 0=0 \text{ do skip} \sim_{BS} \text{abort}$$

Our BS-semantics cannot distinguish between abnormal termination and infinite loops.

What is the SS-semantic relation between "abort" and "while 0=0 do skip"?

Since $\langle \text{while } 0=0 \text{ do skip}, s\rangle \Rightarrow^3 \langle \text{while } 0=0 \text{ do skip}, s\rangle$

$$\text{while } 0=0 \text{ do skip} \not\sim_{SS} \text{abort}$$

What is the SS-semantic relation on termination between "abort" and "while 0=0 do skip"?

$$\text{while } 0=0 \text{ do skip} \sim^{*}_{SS} \text{abort}$$

§4. Nondeterminism

Nondeterminism – the possibility of choosing between two different branches of execution of a program.
– if a branch is chosen, then the other one disappears.

Stm: $S ::= \ldots \mid S_1 \text{ or } S_2$

Notice that this "or" is different of the disjunction from Bexp.
In fact, "b_1 or b_2" and "$S_1 \vee S_2$" are both illegal in Bims.

BS-semantics of Nondeterminism

$$[\text{OR-1}_{BS}] \quad \frac{\langle S_1, s\rangle \to s'}{\langle S_1 \text{ or } S_2, s\rangle \to s'}$$

$$[\text{OR-2}_{BS}] \quad \frac{\langle S_2, s\rangle \to s'}{\langle S_1 \text{ or } S_2, s\rangle \to s'}$$

Example:

$$\begin{tikzcd}[column sep=small, row sep=small] & s[x \mapsto 1] \\ \langle x := \underline{1} \text{ or } (x := \underline{2};\ x := x + \underline{3}), s\rangle \arrow[ur] \arrow[dr] & \\ & s[x \mapsto 5] \end{tikzcd}$$

SS-semantics for Nondeterminism

$$[\text{OR-}1_{ss}] \quad \langle S_1 \text{ or } S_2, s\rangle \Rightarrow \langle S_1, s\rangle$$

$$[\text{OR-}2_{ss}] \quad \langle S_1 \text{ or } S_2, s\rangle \Rightarrow \langle S_2, s\rangle$$

Observe that in this SS-semantics the choice between S_1 and S_2 is treated as a proper transition.

Example: $\langle x := \underline{1} \text{ or } (x := \underline{2};\ x := x + \underline{3}), s\rangle \Rightarrow \langle x := \underline{1}, s\rangle \Rightarrow s[x \mapsto 1]$

and $\langle x := \underline{1} \text{ or } (x := \underline{2};\ x := x + \underline{3}), s\rangle \Rightarrow \langle x := \underline{2};\ x := x + \underline{3}, s\rangle$

$$\Rightarrow \langle x := x + \underline{3}, s[x \mapsto 2]\rangle$$

$$\Rightarrow s[x \mapsto 5]$$

Example:

$$\langle x := \underline{1} \text{ or } (x := \underline{2}\,;\ x := x + \underline{3}),\ s\rangle \begin{array}{l} \nearrow s[x \mapsto 1] \\ \searrow s[x \mapsto 5] \end{array}$$

SS-semantics for Nondeterminism

$[\text{OR-}1_{SS}]$ $\langle S_1 \text{ or } S_2, s\rangle \Rightarrow \langle S_1, s\rangle$

$[\text{OR-}2_{SS}]$ $\langle S_1 \text{ or } S_2, s\rangle \Rightarrow \langle S_2, s\rangle$

Observe that in this SS-semantics the choice between S_1 and S_2 is treated as a proper transition.

Exercise: Propose an SS-semantics for OR such that the choice is not treated as a transition.

Which are the possible transitions of

$$\langle (x := \underline{1}) \text{ or } (\text{while } \underline{0} = \underline{0} \text{ do skip}), s \rangle \quad ?$$

Since there exists no transition $\langle \text{while } \underline{0} = \underline{0} \text{ do skip}, s \rangle \longrightarrow s'$

$$\langle (x := \underline{1}) \text{ or } (\text{while } \underline{0} = \underline{0} \text{ do skip}), s \rangle \longrightarrow s[x \mapsto 1]$$

Hence, $(x := \underline{1}) \text{ or } (\text{while } \underline{0} = \underline{0} \text{ do skip}) \sim_{BS} x := \underline{1}$

The BS-semantics suppresses infinite loops, i.e., undesired choices do not result in a transition — <u>angelic nondeterminism</u>

Which is the SS-relation between

"$x := \underline{1}$" and "$(x := \underline{1}) \text{ or } (\text{while } \underline{0} = \underline{0} \text{ do skip})$"?

$$\langle (x := \underline{1}) \text{ OR } (\text{while } \underline{0} = \underline{0} \text{ do skip}), s \rangle \Longrightarrow \langle x := \underline{1}, s \rangle \Longrightarrow s[x \mapsto 1]$$

and

$$\langle (x := \underline{1}) \text{ or } (\text{while } \underline{0} = \underline{0} \text{ do skip}), s \rangle \Longrightarrow \langle \text{while } \underline{0} = \underline{0} \text{ do skip}, s \rangle \Longrightarrow$$

$$\Longrightarrow^3 \langle \text{while } \underline{0} = \underline{0} \text{ do skip}, s \rangle \Longrightarrow \ldots$$

The SS-semantics exibits infinite loops - <u>demonic nondeterminism</u>

What about the SS-relation to termination?

$$(x := \underline{1}) \sim^*_{SS} (x := \underline{1}) \text{ or } (\text{while } \underline{0} = \underline{0} \text{ do skip})$$

§ 5. Concurrency

$\mathbf{Stm}$: $\quad S ::= \ldots \mid S_1 \| S_2$

Informal semantics:

$\langle x := \underline{1} \,\|\, (x := \underline{2};\ x := x + \underline{3}), s\rangle$

$\Rightarrow \langle x := \underline{2};\ x := x + \underline{3},\ s[x \mapsto 1]\rangle \Rightarrow^2 s[x \mapsto 5]$

OR

$\Rightarrow \langle x := \underline{1} \,\|\, x := x + \underline{3},\ s[x \mapsto 2]\rangle$

$\qquad \Rightarrow \langle x := x + \underline{3},\ s[x \mapsto 1]\rangle \Rightarrow s[x \mapsto 4]$

$\qquad$ OR

$\qquad \Rightarrow \langle x := 1,\ s[x \mapsto 5]\rangle \Rightarrow s[x \mapsto 1]$

§ 5. Concurrency

Stm : $S ::= \ldots \mid S_1 \| S_2$

SS-semantics for parallel

$$[\text{PAR}-1_{ss}] \quad \frac{\langle S_1, s\rangle \Rightarrow \langle S_1', s'\rangle}{\langle S_1 \| S_2, s\rangle \Rightarrow \langle S_1' \| S_2, s'\rangle}$$

$$[\text{PAR}-2_{ss}] \quad \frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1 \| S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

$$[\text{PAR}-3_{ss}] \quad \frac{\langle S_2, s\rangle \Rightarrow \langle S_2', s'\rangle}{\langle S_1 \| S_2, s\rangle \Rightarrow \langle S_1 \| S_2', s'\rangle}$$

$$[\text{PAR}-4_{ss}] \quad \frac{\langle S_2, s\rangle \Rightarrow s'}{\langle S_1 \| S_2, s\rangle \Rightarrow \langle S_1, s'\rangle}$$

Example

$$\langle x:=\underline{5}; (x:=x+\underline{2} \parallel x:=x*\underline{3}; x:=x*\underline{2}), s\rangle$$

$$\Downarrow$$

$$\langle x:=x+\underline{2} \parallel x:=x*\underline{3}; x:=x*\underline{2}, S[x\mapsto 5]\rangle$$

OR

$$\langle x:=x*\underline{3}; x:=x*\underline{2}, S[x\mapsto 7]\rangle \qquad \langle x:=x+\underline{2} \parallel x:=x*\underline{2}, S[x\mapsto 15]\rangle$$

$$\langle x:=x*\underline{3}; x:=x*\underline{2}, S[x\mapsto 7]\rangle \Rightarrow \langle x:=x*\underline{2}, S[x\mapsto 21]\rangle \Rightarrow S[x\mapsto 42]$$

OR

$$\langle x:=x+\underline{2} \parallel x:=x*\underline{2}, S[x\mapsto 15]\rangle \Rightarrow \langle x:=x+\underline{2}, S[x\mapsto 30]\rangle \Rightarrow S[x\mapsto 32]$$

$$\langle x:=x+\underline{2} \parallel x:=x*\underline{2}, S[x\mapsto 15]\rangle \Rightarrow \langle x:=x*\underline{2}, S[x\mapsto 17]\rangle \Rightarrow S[x\mapsto 34]$$

Can we have a BS-semantics for parallel?

$$[\text{PAR-1}_{BS}] \quad \frac{\langle S_1, s\rangle \rightarrow s' \quad \langle S_2, s'\rangle \rightarrow s''}{\langle S_1 \| S_2, s\rangle \rightarrow s''}$$

$$[\text{PAR-2}_{BS}] \quad \frac{\langle S_1, s'\rangle \rightarrow s'' \quad \langle S_2, s\rangle \rightarrow s'}{\langle S_1 \| S_2, s\rangle \rightarrow s''}$$

Example: $\langle x := \underline{5}; (x := x + \underline{2} \,\|\, x := x * \underline{3};\ x := x * \underline{2}), s\rangle \rightarrow s'$

only if $\langle x := x + \underline{2} \,\|\, x := x * \underline{3},\ x := x * \underline{2},\ s[x \mapsto 5]\rangle \rightarrow s'$

only if either $\begin{cases} \langle x := x + \underline{2},\ s[x \mapsto 5]\rangle \rightarrow s[x \mapsto 7] \\ \text{and} \\ \langle x := x * \underline{3},\ x := x * \underline{2},\ s[x \mapsto 7]\rangle \rightarrow s', \text{ i.e., } \boxed{s' = s[x \mapsto 42]} \end{cases}$

or $\begin{cases} \langle x := x * \underline{3},\ x := x * \underline{2},\ s[x \mapsto 5]\rangle \rightarrow s[x \mapsto 30] \\ \text{and} \\ \langle x := x + \underline{2},\ s[x \mapsto 30]\rangle \rightarrow s', \quad \text{i.e., } \boxed{s' = s[x \mapsto 32]} \end{cases}$

We cannot obtain $s' = s[x \mapsto 34]$!

The semantics of concurrency that allows a statement of type $S_1 \| S_2$ to be executed by executing alternatively comands from S_1, then from S_2, then from S_1, then from S_2, etc is known as the interleaving semantics.

We observed that in Bims we cannot provide a BS-semantics for the parallel operator with the interleaving semantics.

There exists operators that one can define in a language for which it is not possible to have a BS-semantics.

Which are the semantic relations between

$S_1 = x := \underline{1} \parallel (x := \underline{2} \,;\, x := x + \underline{3})$

and

$S_2 = (x := \underline{1} \,;\, x := \underline{2} \,;\, x := \underline{3}) \text{ OR } (x := \underline{2} \,;\, x := \underline{1} \,;\, x := x + \underline{3}) \text{ OR } (x := \underline{2} \,;\, x := x + \underline{3} \,;\, x := \underline{1})$?

$$S_1 \sim_{SS} S_2, \quad S_1 \sim_{SS}^{*} S_2 \quad \text{and} \quad S_1 \sim_{BS} S_2$$

In some cases the parallel operator can be simulated by a disjunction of sequential compositions. This, however, is not true in general

Example: $(\text{while } b_1 \text{ do } S_1) \parallel (\text{while } b_2 \text{ do } S_2)$
cannot be represented as a conjunction of sequential compositions.