Lista de exercícios 3

Exercício 1 (Derivações)
Com σ tal que $\sigma(x) = 2$, quais são as derivações na semântica operacional estrutural começando com

1. `if x<0 then s:=-1 else (if x = 0 then s:=0 else s:=1 )`, σ
2. `f1:=1; f2:=1; while ¬(x<1) do ( f:=f1+f2; f1:=f2; f2:=f; x:=x-1 )`, σ

Justifique cada passo com o nome da regra correspondente da semântica operacional estrutural (as provas detalhadas podem ser omitidas).

Exercício 2 (Expressões booleanas condicionais)
Aumente as expressões booleanas com a expressão

$$\mathbf{b} ::= \cdots | \texttt{if } \mathbf{b_1} \texttt{ then } \mathbf{b_2} \texttt{ else } \mathbf{b_3} | \cdots.$$

O valor dessa expressão é o valor de $\mathbf{b_2}$, se o valor de $\mathbf{b_1}$ é true, e $\mathbf{b_3}$ senão.

1. Defina a semântica operacional natural de `if`.
2. Usando a sua semântica mostre a derivação completa para a seguinte expressão:

$$\neg(\texttt{if } (\text{false} \wedge \text{false}) \texttt{ then } (\texttt{if } \text{true } \texttt{then } (\text{false} \wedge \text{true}) \texttt{ else } \text{false}) \texttt{ else } \neg\text{true})$$

Exercício 3 (Estados intermediarios)
1. Prove o seguinte fato: Se $\mathbf{c_1};\mathbf{c_2}, \sigma_0 \to^k \sigma_2$ na semântica operacional estrutural, existe um estado σ_1 e números k_1, k_2 tal que $\mathbf{c_1}, \sigma_0 \to^{k_1} \sigma_1$ e $\mathbf{c_2}, \sigma_1 \to^{k_2} \sigma_2$ com $k = k_1 + k_2$. (Use indução sobre o número de passos de uma derivação).
2. Suponha que $\mathbf{c_1};\mathbf{c_2}, \sigma \to^* \mathbf{c_2}, \sigma'$. Mostre que não necessariamente $\mathbf{c_1}, \sigma \to^* \sigma'$.

Exercício 4 (Expressões com efeitos colaterais)
1. Aumente as expressões aritméticas com a seguinte expressão:

$$\mathbf{a} ::= \cdots | \mathbf{x}\texttt{++} | \cdots \qquad \mathbf{x} \in \text{Loc}, \mathbf{a} \in \text{AExp}.$$

   Modifique a semântica operacional estrutural tal que a avaliação de $\mathbf{x}$`++` tem o efeito colateral de incrementar $\mathbf{l}$. O valor do $\mathbf{l}$`++` num estado σ é o mesmo pela avaliação do $\mathbf{l}$ no estado σ.

2. Aumente as expressões aritméticas com a seguinte expressão:

$$\mathbf{a} ::= \cdots | \texttt{do } \mathbf{c} \texttt{ result is } \mathbf{a} | \cdots \qquad \mathbf{a} \in \text{AExp}, \mathbf{c} \in \text{Com}.$$

   Nessa expressão o comando $\mathbf{c}$ é executado e resulta num novo estado. O valor da expressão $\mathbf{a}$ nesse estado é o valor da expressão inteira.

3. Prove ou mostre um contra-exemplo: `do` $\mathbf{x}$`:=`$\mathbf{x}$`+1 result is` $\mathbf{x}$`-1` $\sim \mathbf{x}$`++`

Dicas: Use a semântica operacional estrutural para expressões aritméticas e booleanas (veja material suplementar na página que tem todas as regras). O estado nessa semântica nunca mudou e simplesmente foi "copiado" nas transições. Como as expressões agora tem efeitos colaterais, um passo da avaliação de uma expressão também pode resultar em um estado diferente!

Exercício 5 (Interpredador de IMP)
Na aula vimos a definiçaõ da sintaxe do IMP em OCaml e um exemplo de um avaliador de expressões aritméticas. Complete a implementação com (a) uma função `eval_bExpr` para avaliar expressões booleanas e (b) uma função `eval_com` para avaliar comandos. Use a semântica operacional natural para os comandos, isto é, o avaliador não produz passos intermediarios, mas só o estado final do comando.
Teste o implementação com o seguinte programa:

```
n := 1
x := 1
while n < 10 do
  x := x * n;
  n := n + 1
```

Qual é o valor final do n?
Observação: Entregue o programa eletronicamente (email, ...) ou entregue uma URL para baixar o programa.