Soluções prova 1

Questão 1 (Ordem e progresso, 20%)
(Progresso, preservação: 0,8pt, Segurança: 0,4pt)
O que significam as noções “segurança”, “progresso” e “preservação” num sistema de tipos?
Ambas, progresso e preservação são características de uma linguagem que relacionam o sistema de tipos com a semântica operacional.
“Progresso” é a característica, que um programa bem tipado sempre tem um próxima passo na execução até chegar num estado final. No caso da semântica operacional estrutural (small-step) isso significa, que sempre tem mais uma transição na execução; no caso da semântica operacional natural (big-step) que existe uma prova ou a prova não acaba (nem com um contra-exemplo finito). Formalmente (nesse exemplo para expressões)

$$(\gamma \vdash \mathbf{e} : \tau) \rightarrow (\exists v \in V : \mathbf{e}, \sigma \Downarrow v)\,.$$

“Preservação” significa que o sistema de tipos e consiste com a regras da semântica. Se uma expressão (ou um comando) tem um tipo τ, o valor final que resulta da avaliação dessa expressão pertence ao mesmo tipo. Formalmente (para expressões):

$$(\gamma \vdash \mathbf{e} : \tau \wedge \mathbf{e}, \sigma \Downarrow v) \rightarrow \gamma \vdash v : \tau.$$

Segurança é a combinação do progresso e preservação.

Questão 2 (Aplicação da semântica operacional, 20%)
(Cada item 1pt)

(a) Derivação no sistema de tipos sem a derivações triviais (TNum, TBool)

$$\dfrac{\emptyset \vdash_E \texttt{1} : \text{int} \qquad \dfrac{\gamma_0 \vdash_E \texttt{3} : \text{int} \qquad \dfrac{\dfrac{\dfrac{\gamma_1 \vdash_E x : \text{int}\gamma_1 \vdash_E 1 : \text{int}}{\gamma_1 \vdash_E \texttt{x+1} : \text{int}}\,\text{TSom} \qquad \gamma_1(y) = (\text{int}, \text{var})}{\gamma_1[x \mapsto (\text{int}, \text{const})] \vdash_C \texttt{y:=x+1} : \text{void}}\,\text{T:=} \qquad \dfrac{\gamma_2 \vdash_E \texttt{5} : \text{int} \quad \dfrac{\gamma_3(p) = (\text{int}, \text{proc}) \quad \gamma_3 \vdash_E x : \text{int}}{\gamma_3 = \gamma_2[x \mapsto (\text{int}, \text{var})] \vdash_C \texttt{p(x)} : \text{void}}\,\text{TCallProc}}{\gamma_2 = \gamma_1[p \mapsto (\text{int}, \text{proc})] \vdash_C \texttt{let var x:=5 in p(x)} : \text{void}}\,\text{TLetVar}}{\gamma_1 = \gamma_0[x \mapsto (\text{int}, \text{const})] \vdash_C \texttt{let proc p(x:int)=y:=x+1 in let var x:=5 in p(x)} : \text{void}}\,\text{TLetProc}}{\gamma_0 = \{x \mapsto (\text{int}, \text{var}] \vdash_C \texttt{let const x=3 in let proc p(x:int)=y:=x+1 in let var x:=5 in p(x)} : \text{void}}\,\text{TLetCte}}{\emptyset \vdash_E \texttt{let var y:=1 in let const x=3 in let proc p(x:int)=y:=x+1 in let var x:=5 in p(x)} : \text{void}}\,\text{TLetVar}$$

(b) Avaliação na semântica operacional natural, a partir de um ambiente e estado vazio $\rho_0 = \sigma_0 = \emptyset$:

$$\dfrac{\rho_0 \vdash \texttt{1}, \sigma \Downarrow 1 \quad 0 \notin \text{dom}(\sigma) \qquad \dfrac{\rho_1 \vdash \texttt{3}, \sigma_1 \Downarrow 3 \qquad \dfrac{\dfrac{\rho_3 \vdash \texttt{5}, \sigma_1 \Downarrow 5 \quad 1 \notin \text{dom}(\sigma_1) \qquad \dfrac{\rho_4 \vdash \texttt{x}, \sigma_2 \Downarrow 5 \quad \rho_4(p) = (x, \rho_2, \texttt{y:=x+1})\rho_2[x \mapsto 5] \vdash \texttt{y:=x+1}, \sigma_2 \Downarrow \sigma_3}{\rho_4 \vdash \texttt{p(x)}, \sigma_2 \Downarrow \sigma_3}\,\text{CallProc}}{\rho_3 \vdash \texttt{let var x:=5 in p(x)}, \sigma_1 \Downarrow \sigma_3}\,\text{LetVar}}{\rho_2 \vdash \texttt{let proc p(x:int)=y:=x+1 in let var x:=5 in p(x)}, \sigma_1 \Downarrow \sigma_3}\,\text{LetProc}}{\rho_1 \vdash \texttt{let const x=3 in let proc p(x:int)=y:=x+1 in let var x:=5 in p(x)}, \sigma_1 \Downarrow \sigma_3}\,\text{LetConst}}{\rho_0 \vdash \texttt{let var y:=1 in let const x=3 in let proc p(x:int)=y:=x+1 in let var x:=5 in p(x)}, \sigma_0 \Downarrow \sigma_3}\,\text{LetVar}$$

com

$$\rho_0 = \emptyset, \qquad \rho_1 = \{y \mapsto @0\}$$
$$\rho_2 = \{y \mapsto @0, x \mapsto 3\}$$
$$\rho_3 = \{y \mapsto @0, x \mapsto 3, p \mapsto (x, \rho_2, \texttt{y:=x+1})\}$$
$$\rho_4 = \{y \mapsto @0, x \mapsto @1, p \mapsto (x, \rho_2, \texttt{y:=x+1})\}$$
$$\sigma_0 = \emptyset, \qquad \sigma_1 = \{@0 \mapsto 1\}$$
$$\sigma_2 = \{@0 \mapsto 1, @1 \mapsto 5\}$$
$$\sigma_3 = \{@0 \mapsto 1, @1 \mapsto 6\}$$

(Observação: Para enfatizar a diferença entre o conjunto $\mathbb{Z}$ e End $= \mathbb{N}$ um $n \in$ End está escrito $@n$.)

Questão 3 (Semântica da atribuição condicional, 30%)
(Cada item 1pt)

(a) A gramática extendida é
$$\mathbf{c} ::= \cdots|\mathbf{e_1}\texttt{?}\quad \mathbf{x_1}\texttt{,}\mathbf{x_2}\ \texttt{:=}\ \mathbf{e_2}|\cdots$$

(b) O sistema de tipos precisa mais uma regra
$$\frac{\gamma \vdash_E \mathbf{e_1} : \text{bool} \quad \gamma \vdash_E \mathbf{x_1} : \tau \quad \gamma \vdash_E \mathbf{x_2} : \tau \quad \gamma \vdash_E \mathbf{e_2} : \tau}{\gamma \vdash_C \mathbf{e_1}\texttt{?}\quad \mathbf{x_1}\texttt{,}\mathbf{x_2}\ \texttt{:=}\ \mathbf{e_2} : \text{void}}\ \text{T?:=}$$
que garante que as duas variáveis e a expressão do lado direto tem o mesmo tipo.

(c) Na semântica operacional natural as regras
$$\frac{\mathbf{e_1}, \sigma \Downarrow \text{true} \quad \mathbf{x_1}\texttt{:=}\mathbf{e_2}, \sigma \Downarrow \sigma'}{\mathbf{e_1}\texttt{?}\quad \mathbf{x_1}\texttt{,}\mathbf{x_2}\ \texttt{:=}\ \mathbf{e_2}, \sigma \Downarrow \sigma'}\ \text{?:=}$$
$$\frac{\mathbf{e_1}, \sigma \Downarrow \text{false} \quad \mathbf{x_2}\texttt{:=}\mathbf{e_2}, \sigma \Downarrow \sigma'}{\mathbf{e_1}\texttt{?}\quad \mathbf{x_1}\texttt{,}\mathbf{x_2}\ \texttt{:=}\ \mathbf{e_2}, \sigma \Downarrow \sigma'}\ \text{?:=}$$
definem a avaliação nos dois casos.

Questão 4 (Preservação, 30%)
(Princípo da indução: 1pt, Base: 1pt, Passo: 1pt)
Temos que provar uma característica de todas expressões. A prova é com indução estrutural sobre as expressões. A característica que nos queremos provar é
$$P(\mathbf{e}) = (\gamma \vdash \mathbf{e} : \tau) \land (\gamma \vdash \sigma) \to (\mathbf{e}, \sigma \Downarrow v) \land (\tau = \text{int} \to v \in \mathbb{Z}) \land (\tau = \text{bool} \to v \in \mathbb{B}).$$
Seguindo o princípio da indução estrutural temos a equivalência
$$\begin{aligned}\forall \mathbf{e} \in \text{Exp} : P(\mathbf{e}) \quad \leftrightarrow \quad & \forall \mathbf{n} \in \text{Num} : P(\mathbf{n}) \land \\ & \forall \mathbf{t} \in \text{Bool} : P(\mathbf{t}) \land \\ & \forall \mathbf{x} \in \text{Ident} : P(\mathbf{x}) \land \\ & \forall \mathbf{e} \equiv \mathbf{e_1}\texttt{+}\mathbf{e_2} : P(\mathbf{e_1}) \land P(\mathbf{e_2}) \to P(\mathbf{e}) \\ & \forall \mathbf{e} \equiv \neg\mathbf{e}' : P(\mathbf{e}') \to P(\mathbf{e})\end{aligned}$$

Observe que nos incluimos só as operações $\texttt{+}$ e $\neg$; os outros casos são equivalentes. A base da indução corresponde com os três primeiros casos (Num,Bool,Ident) e o passo com os últimos dois.
Prova da base:

(a) Se $\mathbf{e} \equiv \mathbf{n}$, $\tau = \text{int}$ porque só tem uma única regra de tipo, e a semântica operacional avalie $\texttt{meta}n, \sigma \Downarrow n$ para qualquer σ. Portanto $\tau = \text{int} \to v \in \mathbb{Z}$ é verdadeiro e $P(e)$ também.

(b) Se $\mathbf{e} \equiv \mathbf{t}$, $\tau = \text{bool}$ porque só tem uma única regra de tipo, e a semântica operacional avalie $\texttt{meta}t, \sigma \Downarrow t$ para qualquer σ. Portanto $\tau = \text{bool} \to v \in \mathbb{B}$ é verdadeiro e $P(e)$.

(c) Seja $\mathbf{e} \equiv \mathbf{x}$. Se $\gamma \vdash_E \mathbf{x} : \tau$ temos $\gamma(x) = \tau$. Logo $\sigma(x) = v$ é definido, e $\tau = \text{int} \to v \in \mathbb{Z}$ e $\tau = \text{bool} \to v \in \mathbb{B}$, porque $\gamma \vdash \sigma$. A regra Tid da semântica operacional garante que $\mathbf{x}, \sigma \Downarrow v$. Portanto a característica está verdadeira para identificadores.

Prova do passo:

(a) Seja $\mathbf{e} \equiv \mathbf{e_1}\texttt{+}\mathbf{e_2}$ e $P(\mathbf{e_1})$, $P(\mathbf{e_2})$ verdadeiros. Se $\gamma \vdash_E \mathbf{e} : \tau$, temos $\tau = \text{int}$ porque só tem uma única regra de tipo. Como as premissas da regra de tipo TSom tem tipo int também, $P(\mathbf{e_1})$ e $P(\mathbf{e_2})$ permitem concluir que $\mathbf{e_1}, \sigma \Downarrow n_1$ e $\mathbf{e_2}, \sigma \Downarrow n_2$. Logo, a regra da semântica operacional Som se aplica: $\mathbf{e}, \sigma \Downarrow n$ com $n = n_1 + n_2$ e a característica é verdadeiro para $\mathbf{e}$ também.

(b) Seja $\mathbf{e} \equiv \neg\mathbf{e}'$ e $P(\mathbf{e}')$ verdadeiro. Se $\gamma \vdash_E \mathbf{e} : \tau$, temos $\tau = \text{bool}$ porque só tem uma única regra de tipo. Como a premissa da regra de tipo TNeg tem tipo bool também, $P(\mathbf{e}'$ permite de concluir que $\mathbf{e}', \sigma \Downarrow b$. A regra Neg da semântica se aplica e temos $\mathbf{e}', \sigma \Downarrow b'$ com $b' = \neg b$ e a característica é verdadeiro para $\mathbf{e}'$ também.

Questão 5 (Registros, 20%)
(Cada item 1pt)

1. No sistema de tipos a regra

$$\frac{\gamma \vdash_E \mathbf{x_i} : \tau_i}{\gamma \vdash_E \{\mathbf{x_1} = \mathbf{e_1}, \ldots, \mathbf{x_n} = \mathbf{e_n}\} : \{\mathbf{x_1} : \tau_1, \ldots, \mathbf{x_n} : \tau_n\}}\ \text{TReg}$$

   define o tipo de um registro e

$$\frac{\gamma \vdash_E \{\mathbf{x_1} = \mathbf{e_1}, \ldots, \mathbf{x_n} = \mathbf{e_n}\} : \{\mathbf{x_1} : \tau_1, \ldots, \mathbf{x_n} : \tau_n\} \quad \mathbf{x} = \mathbf{x_i}}{\gamma \vdash_E \mathbf{e.x} : \tau_i}\ \text{TProj}$$

   checa as projeções.

2. Na semântica operacional temos que definir uma novo tipo de valor no estado

$$v \in V = \mathbb{Z} \cup \mathbb{B} \cup \bigcup_{i \geq 0} (\text{Ident} \times V)^i$$

   e temos que avaliar registros (observe que eles são definidos com expressões)

$$\frac{\mathbf{x_i}, \sigma \Downarrow \mathbf{v_i}}{\{\mathbf{x_1}\mathbf{=}\mathbf{e_1}, \ldots, \mathbf{x_n}\mathbf{=}\mathbf{e_n}\}, \sigma \Downarrow ((\mathbf{x_1}, \mathbf{v_1}) \ldots (\mathbf{x_n}, \mathbf{v_n}))}\ \text{Reg}$$

   e projeções

$$\frac{\mathbf{e}, \sigma \Downarrow ((\mathbf{x_1}, \mathbf{v_1}) \ldots (\mathbf{x_n}, \mathbf{v_n})) \quad \mathbf{x} = \mathbf{x_i}}{\mathbf{e}.\mathbf{x}, \sigma \Downarrow \mathbf{v_i}}\ \text{Proj}$$

Exemplos de registros:

```
let const p = { x=3, y=5, ok=true } in
  let var c:=0 in
    if (p.ok) then
      c := p.x
    else
      c:= p.y

let const circ = { centro = { x=5, y=6 }, raio= 3 } in
  let var dist_sqr:=(circ.centro.x*circ.centro.x+circ.centro.y*circ.centro.y) in
    skip
```