1 Exercícios de Semântica Operacional e Sistemas de Tipos

1.1 Respostas - Linguagem L1

Exercício 1.1 Mostre as derivações de todos os passos de avaliação da expressão $(2+3)+(3 \geq \texttt{true})$ na seguinte configiração (onde σ é uma memória qualquer):

$$\langle (2+3) + (3 \geq \texttt{true}),\ \sigma \rangle$$

Resposta:

Derivação do 1° passo:

$$\dfrac{\dfrac{[\![5]\!] = [\![2+3]\!]}{\langle 2+3,\ \sigma\rangle \ \longrightarrow\ \langle 5,\ \sigma\rangle}\ \textsc{op+}}{\langle (2+3) + (3 \geq \texttt{true}),\ \sigma\rangle \ \longrightarrow\ \langle 5 + (3 \geq \texttt{true}),\ \sigma\rangle}\ \textsc{op1}$$

Derivação do 2° passo:

$$\dfrac{\langle 3 \geq \texttt{true},\ \sigma\rangle \not\longrightarrow}{\langle 5 + (3 \geq \texttt{true}),\ \sigma\rangle}\ \textsc{op2}$$

Também é possível mostrar os passos acima omitindo as derivações de cada passo

$$\begin{array}{ccc} \langle (2+3) + (3 \geq \texttt{true}),\ \sigma\rangle & \longrightarrow & \langle 5 + (3 \geq \texttt{true}) \quad \sigma\rangle \\ & \not\longrightarrow & \end{array}$$

Exercício 1.2 Mostre as derivações de todos os passos da avaliação para a configuração $\langle (l_0 := 7); l_1 := (!l_0 + 2),\ \{l_0 \mapsto 0, l_1 \mapsto 0\}\rangle$.

Resposta: (parênteses omitidos por clareza)

Derivação do 1° passo:

$$\dfrac{\dfrac{l_0 \in Dom(\{l_0 \mapsto 0, l_1 \mapsto 0\})}{\langle l_0 := 7,\ \{l_0 \mapsto 0, l_1 \mapsto 0\}\rangle \ \longrightarrow\ \langle \texttt{skip},\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle}\ \textsc{atr1}}{\langle l_0 := 7; l_1 := !l_0 + 2,\ \{l_0 \mapsto 0, l_1 \mapsto 0\}\rangle \ \longrightarrow\ \langle \texttt{skip}; l_1 := !l_0 + 2,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle}\ \textsc{seq2}$$

Derivação do 2° passo:

$$\dfrac{}{\langle \texttt{skip}; l_1 := !l_0 + 2,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \ \longrightarrow\ \langle l_1 := !l_0 + 2,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle}\ \textsc{seq1}$$

Derivação do 3° passo: (na derivação abaixo $\sigma = \{l_0 \mapsto 7, l_1 \mapsto 0\}$)

$$\dfrac{\dfrac{\dfrac{l_0 \in Dom(\sigma) \quad \sigma(l_0) = 7}{\langle\ !l_0,\ \sigma\rangle \ \longrightarrow\ \langle 7,\ \sigma\rangle}\ \textsc{deref}}{\langle\ !l_0 + 2,\ \sigma\rangle \ \longrightarrow\ \langle 7 + 2,\ \sigma\rangle}\ \textsc{op2}}{\langle l_1 := !l_0 + 2,\ \sigma\rangle \ \longrightarrow\ \langle l_1 := 7 + 2,\ \sigma\rangle}\ \textsc{atr2}$$

Derivação do 4° passo:

$$\dfrac{\dfrac{[\![9]\!] = [\![7+2]\!]}{\langle 7+2,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \ \longrightarrow\ \langle 9,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle}\ \textsc{op+}}{\langle l_1 := 7+2,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \ \longrightarrow\ \langle l_1 := 9,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle}\ \textsc{atr2}$$

Derivação do 5° passo:

$$\dfrac{l_1 \in Dom(\{l_0 \mapsto 7, l_1 \mapsto 0\})}{\langle l_1 := 9,\ \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \ \longrightarrow\ \langle \texttt{skip},\ \{l_0 \mapsto 7, l_1 \mapsto 9\}\rangle}\ \textsc{atr1}$$

Também é possível mostrar todos os passos acima omitindo as derivações de cada passo

$$\begin{array}{lcll}
\langle (l_0 := 7); l_1 := (\,!l_0 + 2),\ \{l_0 \mapsto 0, l_1 \mapsto 0\}\rangle & \longrightarrow & \langle \texttt{skip}; l_1 := !l_0 + 2, & \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \\
& \longrightarrow & \langle l_1 := !l_0 + 2, & \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \\
& \longrightarrow & \langle l_1 := \ 7 + 2, & \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \\
& \longrightarrow & \langle l_1 := \ 9, & \{l_0 \mapsto 7, l_1 \mapsto 0\}\rangle \\
& \longrightarrow & \langle \texttt{skip}, & \{l_0 \mapsto 7, l_1 \mapsto 9\}\rangle
\end{array}$$

Exercício 1.3 Mostre a derivação de tipo para a expressão $(l_0 := 7); l_1 := (!l_0 + 2)$ em um ambiente de tipo Γ onde $l_0 : \mathsf{int\ ref}$ e $l_1 : \mathsf{int\ ref}$.

Resposta:

$$\dfrac{\dfrac{\Delta(l_0) = \mathsf{int\ ref} \qquad \dfrac{}{\Delta \vdash 7 : \mathsf{int}}\ \textsc{Tint}}{\Delta \vdash l_0 := 7 : \mathsf{unit}}\ \textsc{Tatr} \qquad \dfrac{\Delta(l_1) = \mathsf{int\ ref} \qquad \dfrac{\dfrac{\Delta(l_0) = \mathsf{int\ ref}}{\Delta \vdash !l_0 : \mathsf{int}}\ \textsc{Tderef} \qquad \dfrac{}{\Delta \vdash 2 : \mathsf{int}}\ \textsc{Tint}}{\Delta \vdash !l_0 + 2 : \mathsf{int}}\ \textsc{T+}}{\Delta \vdash l_1 := \ (!l_0 + 2) : \mathsf{unit}}\ \textsc{Tatr}}{\Delta \vdash (l_0 := 7); l_1 := (!l_0 + 2) : \mathsf{unit}}\ \textsc{Tseq}$$

Exercício 1.4 Mostre a derivação de tipo para a expressão

$$(l_2 := \ 0);\ \texttt{while}\ !l_1 \geq 0\ \texttt{do}\ (l_2 := \ (!l_2 + !l_1);\ (l_1 := \ (!l_1 + -1)))$$

com $\Gamma = \{l_1 : \mathsf{int\ ref},\ \ l_2 : \mathsf{int\ ref},\ \ l_3 : \mathsf{int\ ref}\}$

Resposta:

$$\dfrac{\dfrac{\Delta(l_2) = \mathsf{int\ ref} \qquad \dfrac{}{\Delta \vdash 0 : \mathsf{int}}\ \textsc{Tint}}{\Delta \vdash l_2 := \ 0 : \mathsf{unit}}\ \textsc{Tatr} \qquad \dfrac{\dfrac{\dfrac{\Delta(l_1) = \mathsf{int\ ref}}{\Delta \vdash\ !l_1 : \mathsf{int}}\ \textsc{Tderef} \qquad \dfrac{}{\Delta \vdash 0 : \mathsf{int}}\ \textsc{Tint}}{\Delta \vdash\ !l_1 \geq 0 : \mathsf{bool}}\ \textsc{T}\geq \qquad \nabla}{\Delta \vdash \texttt{while}\ !l_1 \geq 0\ \texttt{do}\ l_2 := !l_2 + !l_1;\ l_1 := !l_1\ +\ -1 : \mathsf{unit}}\ \textsc{Twhile}}{\Delta \vdash l_2 := \ 0;\ \texttt{while}\ !l_1 \geq 0\ \texttt{do}\ l_2 := !l_2 + !l_1;\ l_1 := !l_1\ +\ -1 : \mathsf{unit}}\ \textsc{Tseq}$$

derivação ∇:

$$
\dfrac{
\dfrac{
\Delta(l_2) = \mathsf{int\ ref}
\qquad
\dfrac{
\dfrac{\Delta(l_2) = \mathsf{int\ ref}}{\Delta \vdash\, !l_2 : \mathsf{int}}\,\textsc{Tderef}
\qquad
\dfrac{\Delta(l_1) = \mathsf{int\ ref}}{\Delta \vdash\, !l_1 : \mathsf{int}}\,\textsc{Tderef}
}{\Delta \vdash\, !l_2 + !l_1 : \mathsf{int}}\,\mathrm{T}{+}
}{\Delta \vdash l_2 := !l_2 + !l_1 : \mathsf{unit}}\,\textsc{Tatr}
\qquad
\dfrac{
\Delta(l_1) = \mathsf{int\ ref}
\qquad
\dfrac{
\dfrac{\Delta(l_1) = \mathsf{int\ ref}}{\Delta \vdash\, !l_1 : \mathsf{int}}\,\textsc{Tderef}
\qquad
\dfrac{}{\Delta \vdash -1 : \mathsf{int}}\,\textsc{Tint}
}{\Delta \vdash\, !l_1 \;+\; -1 : \mathsf{int}}\,\mathrm{T}{+}
}{\Delta \vdash l_1 := !l_1 \;+\; -1 : \mathsf{unit}}\,\textsc{Tatr}
}{\Delta \vdash l_2 := !l_2 + !l_1;\; l_1 := !l_1 \;+\; -1 : \mathsf{unit}}\,\textsc{Tseq}
$$

Exercício 1.5 Modifique a semântica operacional de L1 de tal forma que a linguagem deixe de ser determinística.

Resposta:
Basta acresentar uma regra que reduza uma expressão de forma diferente da forma pela qual ela é reduzida por uma regra já existente. Por exemplo, adicionando a seguinte regra para reduzir expressões condicionais:

$$\langle \texttt{if true then } e_2 \texttt{ else } e_3,\ \sigma\rangle \quad\longrightarrow\quad \langle e_3,\ \sigma\rangle$$

Exercício 1.6 O Teorema da Preservação de Tipo continua valendo caso as regras de avaliação SEQ1 e ATR1 sejam trocadas pelas regras SEQ1' e ATR1' abaixo? Em caso negativo dê um exemplo e modifique o sistema de tipos de forma a reestabelecer Preservação de Tipos.

$$\langle v\texttt{;}e_2,\ \sigma\rangle \quad\longrightarrow\quad \langle e_2,\ \sigma\rangle \tag{SEQ1'}$$

$$\frac{l \in Dom(\sigma)}{\langle l\texttt{:=}\ n,\ \sigma\rangle \quad\longrightarrow\quad \langle n,\ \sigma[l \mapsto n]\rangle} \tag{ATR1'}$$

Resposta:
A regra SEQ1' *não afeta Preservação de Tipo.* ***Se*** *uma expressão $v; e_2$ for bem tipada, ela terá o tipo de e_2 (pela regra de tipo* TSEQ*) e, no lado direito da regra de avaliação* SEQ1'*, temos a expressão e_2, portanto o tipo será mantido.*
Já a regra ATR1' *faz com que Preservação de Tipo não seja mais verdadeira.* ***Se*** *a expressão $l\texttt{:=}n$ for bem tipada ela terá tipo* unit *(de acordo com a regra de tipo* TATR*), mas no lado direito da regra de redução* ATR1' *acima temos a expressão n que é do tipo* int*(de acordo com a regra de tipo* TINT*).*
Mantendo a regra de redução ATR1' *acima, temos que mudar a regra de tipo para atribuição para:*

$$\frac{\Delta(l) = \mathsf{int\ ref} \qquad \Delta \vdash e : \mathsf{int}}{\Delta \vdash l\texttt{:=}\ e : \mathsf{int}}$$

Obs.: a adoção dessa regra de tipo reestabelece Preservação de Tipo, mas impede que expressões tais como $l_1 := 5;\ l_2 := 10$, por exemplo, sejam bem tipadas pois a regra de tipos para seqüência exige que o lado esquerdo do ";"seja do tipo unit*. Uma forma de corrigir esse problema é mudar a regra da seqüência para:*

$$\frac{\Delta \vdash e_1 : T_1 \qquad \Delta \vdash e_2 : T_2}{\Delta \vdash e_1;\ e_2 : T_2}$$

1.2 Respostas - Linguagem L2

Exercício 1.7 Construa derivações para todos os passos da redução e construa a derivação de tipo para a expressão abaixo:

$$\begin{array}{l}\texttt{let}\ x : \text{int} \to \text{int} = \\ \qquad fn\ y : \text{int} \Rightarrow y + 10 \\ \texttt{in} \\ \quad x\ 10 \\ \texttt{end}\end{array}$$

Resposta:
Derivação do 1° passo:

$$\dfrac{}{\langle \texttt{let}\ x{:}\text{int} \to \text{int} = fn\ y{:}\text{int} \Rightarrow y + 10\ \texttt{in}\ x\ 10\ \texttt{end},\ \sigma\rangle \quad \longrightarrow \quad \langle \{fn\ y{:}\text{int} \Rightarrow y + 10/x\}(x\ 10),\ \sigma\rangle}\ \textsc{let1}$$

$\{fn\ y{:}\text{int} \Rightarrow y + 10/x\}(x\ 10) \ \equiv\ (fn\ y{:}\text{int} \Rightarrow y + 10)\ 10$, logo

Derivação do 2° passo:

$$\dfrac{}{\langle (fn\ y{:}\text{int} \Rightarrow y + 10)\ 10,\ \sigma\rangle \quad \longrightarrow \quad \langle \{10/y\}(y + 10),\ \sigma\rangle}\ \beta$$

$\{10/y\}(y + 10) \ \equiv\ 10 + 10$ *logo:*

Derivação do 3° passo:

$$\dfrac{}{\langle 10 + 10,\ \sigma\rangle \quad \longrightarrow \quad \langle 20,\ \sigma\rangle}\ \textsc{op}+$$

Também é possível mostrar todos os passos acima da seguinte forma

$$\begin{array}{lcll}\langle \texttt{let}\ x{:}\text{int} \to \text{int} = fn\ y{:}\text{int} \Rightarrow y + 10\ \texttt{in}\ x\ 10\ \texttt{end},\ \sigma\rangle & \longrightarrow & \langle \{fn\ y : \text{int} \Rightarrow y + 10/x\}\ (x\ 10) & \sigma\rangle \\ & \equiv & \langle (fn\ y : \text{int} \Rightarrow y + 10)\ 10, & \sigma\rangle \\ & \longrightarrow & \langle \{10/y\}\ (y + 10), & \sigma\rangle \\ & \equiv & \langle 10 + 10, & \sigma\rangle \\ & \longrightarrow & \langle 20, & \sigma\rangle\end{array}$$

Segue abaixo a derivação de tipo para essa expressão:

$$\dfrac{\dfrac{\dfrac{\dfrac{\{y : \text{int}\}(y) = \text{int}}{y : \text{int} \vdash y : \text{int}}\ \textsc{TVar} \qquad \dfrac{}{y : \text{int} \vdash 10 : \text{int}}\ \textsc{TInt}}{y : \text{int} \vdash y + 10 : \text{int}}\ \textsc{T}+}{\vdash fn\ y{:}\text{int} \Rightarrow y + 10 : \text{int} \to \text{int}}\ \textsc{TAbs} \qquad \dfrac{\dfrac{\{x : \text{int} \to \text{int}\}(x) = \text{int} \to \text{int}}{x : \text{int} \to \text{int} \vdash x : \text{int} \to \text{int}}\ \textsc{TVar} \qquad \dfrac{}{x : \text{int} \to \text{int} \vdash 10 : \text{int}}\ \textsc{TInt}}{x : \text{int} \to \text{int} \vdash x\ 10 : \text{int}}\ \textsc{TApp}}{\vdash \texttt{let}\ x{:}\text{int} \to \text{int} = fn\ y{:}\text{int} \Rightarrow y + 10\ \texttt{in}\ x\ 10\ \texttt{end} : \text{int}}\ \textsc{TLet}$$

Exercício 1.8 Escreva um programa em L2 que define a função fatorial e calcula o fatorial de 5 (suponha que o operador para multiplicação e subtração já tenham sido adicionados a linguagem)

Resposta:

```
let rec fat : int → int =
      fn x : int ⇒ if 0 ≥ x then 1 else x ∗ fat (x − 1)
in
      fat 5
end
```

Exercício 1.9 Construa as derivações para todos os passos de redução do programa do exercício anterior (antes defina regras da semântica para as operações de multiplicação e subtração).

Exercício 1.10 Construa a derivação de tipo do programa do exercício anterior (antes defina regras de tipo para as operações de multiplicação e subtração).

Exercício 1.11 Suponha L2 **sem** as operações de soma (+) e comparação (≥) mas equipada com operação `iszero` que, dado argumento inteiro, retorna verdadeiro caso o argumento seja igual a zero e falso caso contrário. Suponha também a existência de operadores `pred` e `succ` que decrementam e incrementam de um o seus argumento. Com essa linguagem defina as funções recursivas `equal` (entre inteiros), `plus`, `times` (soma e multiplicação dos seus dois argumentos inteiros respectivamente). **Suponha que essas funções serão chamadas sempre para operarem com inteiros maiores ou iguais a zero.**

```
let rec equal : int → int → bool =
      fn x : int ⇒ fn y : int ⇒
            if iszero x then iszero y
            else if iszero y then false
            else equal (pred x)(pred y)
in
      equal 3 4
end

let rec plus : int → int → int =
      fn x : int ⇒ fn y : int ⇒
            if iszero x then y
            else succ (plus (pred x) y)
in
let rec times : int → int → int =
      fn x : int ⇒ fn y : int ⇒
            if iszero x then 0
            else plus y (times (pred x) y)
in
      times 3 4
end
end
```

1.3 Respostas - Linguagem L3

Exercício 1.12 Escreva um programa em L3 que quando avaliado produz a seguinte memória (que contém um ciclo): $\{l_1 \mapsto fn\ x{:}\mathsf{int} \Rightarrow (!l_2)\ x, l_2 \mapsto fn\ x{:}\mathsf{int} \Rightarrow (!l_1)\ x\}$

Resposta:

$\texttt{let}\ r_1 : (\mathsf{int} \to \mathsf{int})\ \mathsf{ref} = \mathsf{ref}\ (fn\ x : \mathsf{int} \Rightarrow 0)\ \texttt{in}$
$\texttt{let}\ r_2 : (\mathsf{int} \to \mathsf{int})\ \mathsf{ref} = \mathsf{ref}\ (fn\ x : \mathsf{int} \Rightarrow (!r_1)\ x)\ \texttt{in}$
$\quad r_1 := (fn\ x : \mathsf{int} \Rightarrow (!r_2)\ x)$

Exercício 1.13 Proponha uma regra de tipo para a expressão $e{+}{+}$. A regra deve estar de acordo com a semântica da expressão dada pelas regras abaixo:

$$\frac{\langle e,\ \sigma\rangle \quad\longrightarrow\quad \langle e',\ \sigma'\rangle}{\langle e{+}{+},\ \sigma\rangle \quad\longrightarrow\quad \langle e'{+}{+},\ \sigma'\rangle}$$

$$\frac{\sigma(l) = n \qquad n' = n+1}{\langle l{+}{+},\ \sigma\rangle \quad\longrightarrow\quad \langle \texttt{skip},\ \sigma[l \mapsto n']\rangle}$$

Resposta:

$$\frac{\Gamma;\Delta \vdash e : \mathsf{int\ ref}}{\Gamma;\Delta \vdash e{+}{+} : \mathsf{unit}}$$

Exercício 1.14 Programas escritos em L3 não podem conter endereços $l \in \mathbb{L}$. Por que então incluir endereços na gramática de expressões e definir uma regra de tipo para eles?

Porque, com semântica operacional small-step, os passos intermediários da redução de uma configuração $\langle e,\ \sigma\rangle$ são configurações $\langle e',\ \sigma'\rangle$, onde e' é uma expressão, por esse motivo temos que incluir endereços l na gramática. Já que endereços podem aparecer em passos intermediários de avaliação. A técnica de prova de segurançã usa o próprio sistema de tipos para tipar expressões que surgem em passos intermediários da avaliação. Por esse motivo é necessário definir uma regra de tipo para endereços.

Exercício 1.15 Por que não generalizar as operações de projeção de pares ordenados para $\sharp e\ e'$ ao invés de termos somente $\sharp 1\ e'$ e $\sharp 2\ e'$?

A linguagem L3 é estaticamente tipada, ou seja, toda verificação de tipos é feita sobre a árvore de sintaxe abstrata sem que nada seja avaliado. Sendo assim não teríamos como concluir se o tipo de $\sharp e\ e'$ é o tipo do primeiro ou do segundo componente do par e'.

Exercício 1.16 Tente construir uma derivação de tipo para o programa abaixo usando as regras do sistema de tipos para L3. Explique porque o programa não é bem tipado mostrando porque o processo de construção da derivação de tipo falha.

$$(fn\ x{:}\{B : \mathsf{int}, A : \mathsf{bool}\} \Rightarrow \texttt{if}\ \sharp A\ x\ \texttt{then}\ \sharp B\ x\ \texttt{else}\ 3)\ \ \{A = \texttt{true}, B = 10\}$$

Exercício 1.17 Utilizando as regras da semântica operacional de L3 avalie o programa do exercícios anterior.

Exercício 1.18 Repita os dois exercícios anteriores para o programa abaixo:

$$(fn\ x{:}\{A : \mathsf{bool}\} \Rightarrow \texttt{if}\ \sharp A\ x\ \texttt{then}\ 2\ \texttt{else}\ 3)\ \ \{A = \texttt{true}, B = 10\}$$

Exercício 1.19 Diga se preservação e progresso continuam verdadeiros com a adição das seguintes regras a definição de uma linguagem. Quando for falso dê um contra-exemplo:

- adição da regra

$$\frac{\Gamma; \Delta \vdash e_1 : \mathsf{int} \qquad \Gamma; \Delta \vdash e_2 : \mathsf{int}}{\Gamma; \Delta \vdash (e_1, e_2) : \mathsf{int}}$$

  Resposta: Preservação de Tipos se mantém verdadeiro. Já Progresso é falso como mostra o seguinte contra-exemplo: $(5, 3) + 5$ *é bem tipado tipo* int *já que tanto o par* $(5, 3)$ *como* 5 *são do tipo* int*, mas a valiação desse este termo não progride*

- adição de $\langle (e_1, e_2) + 1,\ \sigma \rangle \quad \longrightarrow \quad \langle e_1,\ \sigma \rangle$
  Resposta: Preservação de Tipos e Progresso se mantêm verdadeiros

Exercício 1.20 Note que as regras da semântica operacional de L3 não fazem nenhum tipo de *garbage collection*: elas permitem que memória seja alocada sem nenhuma limitação (regra para ref e). Como as regras de avaliação poderiam ser refinadas para modelar *garbage collection*? Como seria o enunciado de um teorema afirmando que esse refinamento está correto ?

1.4 Respostas - Exceções

Os exercícios sobre exceções 1.22 até 1.26 abaixo consideram a versão *sem* valor passado ao tratador

Exercício 1.21 Explique porque a linguagem deixaria de ser determinística caso `raise` fosse considerada um valor. Sugestão: pense como ficaria a avaliação da expressão $(fn\ x : \mathsf{int} \Rightarrow\ 0)\ \texttt{raise}$.

Resposta: se `raise` *fosse considerado um valor qualquer uma das regras abaixo seria aplicável quando* `raise` *aparecesse no lado esquerdo de uma aplicação*

$$\langle \texttt{raise}\ e_2,\ \sigma \rangle \ \rightarrow\ \langle \texttt{raise},\ \sigma \rangle \qquad \text{(APPRS)}$$

$$\langle v\ \texttt{raise},\ \sigma \rangle \quad \longrightarrow \quad \langle \texttt{raise},\ \sigma \rangle \qquad \text{(FNRS)}$$

Exercício 1.22 Como fica a avaliação do programa abaixo:

$$((fn\ x : \mathsf{unit} \Rightarrow \texttt{while true do}\ x)\ \texttt{skip})\ \texttt{raise}$$

Resposta: o programa acima entra em loop.

Exercício 1.23 Note que em uma implementação o tipo de uma determinada ocorrência de `raise` deverá ser inferido de acordo com o contexto no qual ela aparece. Não seria mais simples exigir do programador que ele anote as ocorrências de `raise` com o tipo necessário? qual o problema com isso?

Se isso fosse feito não teríamos mais Preservação de Tipos. O termo bem tipado abaixo

$$(fn\ x : \mathsf{int} \Rightarrow x) \quad ((fn\ y : \mathsf{bool} \Rightarrow 5) \quad (\texttt{raise} : \mathsf{bool}))$$

seria reduzido em um passo para o termo mal tipado

$$(fn\ x : \mathsf{int} \Rightarrow x) \quad (\texttt{raise} : \mathsf{bool})$$

Exercício 1.24 Dê o tipo e o resultado da avaliação da seguinte expressão:

$$(fn\ x\mathtt{:}\mathsf{bool} \Rightarrow fn\ y\mathtt{:}\mathsf{bool} \Rightarrow \mathtt{raise})\ \mathtt{false}\ \mathtt{false}\ \mathtt{false}\ \mathtt{false}\ \mathtt{false}$$

Resposta: a expressão avalia para **`raise`** *e ela pode ser de qualquer tipo.*

Exercício 1.25 A propriedade da boa tipagem continua sendo preservada com a extensão de L3 com exceções. O enunciado do teorema do **Progresso** entretanto, precisa ser modificado. Escreva esse enunciado.

*Resposta: o enunciado do **Progresso** fica assim:*
Se $\Gamma; \Delta \vdash e : T$ e e é fechado então e é valor, ou é **`raise`***, ou existe e', σ', tal que $\langle e,\ \sigma\rangle \longrightarrow \langle e',\ \sigma'\rangle$ com $Dom(\Delta) \subseteq Dom(\sigma)$ e e' fechado.*

Os exercícios sobre exceções abaixo consideram a versão na qual o tratador recebe um valor quando a a exceção é ativada

Exercício 1.26 Defina as regras da semântica operacional da extensão de L3 com $\mathtt{raise}\ e$ e $\mathtt{try}\ e_1\ \mathtt{with}\ e_2$ para exceções com passagem de valor.

Resposta: a expressão **`raise`** *v é uma expressão já avaliada (mas não é valor). As seguintes regras devem adicionadas:*

$$\frac{\langle e,\ \sigma\rangle \longrightarrow \langle e',\ \sigma'\rangle}{\langle \mathtt{raise}\ e,\ \sigma\rangle \longrightarrow \langle \mathtt{raise}\ e',\ \sigma'\rangle}$$

$$\langle \mathtt{raise}\ (\mathtt{raise}\ v),\ \sigma\rangle \longrightarrow \langle \mathtt{raise}\ v,\ \sigma\rangle$$

$$\frac{\langle e_1,\ \sigma\rangle \longrightarrow \langle e_1',\ \sigma'\rangle}{\langle \mathtt{try}\ e_1\ \mathtt{with}\ e_2,\ \sigma\rangle \longrightarrow \langle \mathtt{try}\ e_1'\ \mathtt{with}\ e_2,\ \sigma\rangle}$$

$$\langle \mathtt{try}\ v\ \mathtt{with}\ e_2,\ \sigma\rangle \longrightarrow \langle v,\ \sigma\rangle$$

$$\langle \mathtt{try}\ \mathtt{raise}\ v\ \mathtt{with}\ e_2,\ \sigma\rangle \longrightarrow \langle e_2\ v,\ \sigma\rangle$$

E para cada grupo de regras ja existentes acrescentar regra(s) para propagar exceçoes. Abaixo exemplo de regra para sequencia:

$$\langle \mathtt{raise}\ v;\ e_2,\ \sigma\rangle \longrightarrow \langle \mathtt{raise}\ v,\ \sigma\rangle$$

Exercício 1.27 Usando as regras do sistema de tipos e da semântica operacional prove que o termo abaixo é bem tipado e o avalie

$$\begin{array}{l}\mathtt{try}\\ \qquad (fn\ x\mathtt{:}\mathsf{bool} \Rightarrow x)\ (\mathtt{raise}\ 1)\\ \mathtt{with}\\ \qquad fn\ z\mathtt{:}\mathsf{int} \Rightarrow \mathtt{if}\ z = 0\ \mathtt{then}\ \mathtt{true}\ \mathtt{else}\ \mathtt{false}\end{array}$$

1.5 Respostas - Subtipos

Nos exercícios envolvendo sistemas de tipos o ambiente Δ será omitido por clareza.

Exercício 1.28 Dê três exemplos de programas de L3 (sem a extensão de subtipos) não tipados mas bem comportados de acordo com a semântica operacional.

Resposta:

- $(fn\ x{:}\{B : \mathsf{int}, A : \mathsf{bool}\} \Rightarrow \texttt{if}\ \sharp A\ x\ \texttt{then}\ \sharp B\ x\ \texttt{else}\ 3)\ \ \{A = \texttt{true}, B = 10\}$
- $(fn\ x{:}\{A : \mathsf{bool}\} \Rightarrow \texttt{if}\ \sharp A\ x\ \texttt{then}\ \sharp B\ x\ \texttt{else}\ 3)\ \ \{A = \texttt{true}, B = 10\}$
- $\texttt{if true then}\ 5\ \texttt{else (true, true)}$

Exercício 1.29 Construa uma derivação de tipo para o programa abaixo usando as regras do sistema de tipos para L3 extendido com subtipos. Considere somente a relação de subtipos envolvendo tipos registro. Refaça o exercício considerando relação de subtipo entre tipos função.

$$(fn\ x{:}\{B : \mathsf{int}, A : \mathsf{bool}\} \Rightarrow \texttt{if}\ \sharp A\ x\ \texttt{then}\ \sharp B\ 2\ \texttt{else}\ 3)\ \ \{A = \texttt{true}, B = 10\}$$

Resposta: no que segue usamos:

- T_{BA} para o tipo $\{B : \mathsf{int}, A : \mathsf{bool}\}$
- T_{AB} para o tipo $\{A : \mathsf{bool}, B : \mathsf{int}\}$
- $e_{\texttt{if}}$ para a expressão $\texttt{if}\ \sharp A\ x\ \texttt{then}\ \sharp B\ x\ \texttt{else}\ 3$
- Γ' para o ambiente de tipo $x : T_{BA}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'(x) = T_{BA}}{\Gamma' \vdash x : T_{BA}}\,\textsc{TVar}}{\Gamma' \vdash \sharp B\ x : \mathsf{bool}}\,\textsc{TPjr} \quad \dfrac{\dfrac{\Gamma'(x) = T_{BA}}{\Gamma' \vdash x : T_{BA}}\,\textsc{TVar}}{\Gamma' \vdash \sharp A\ x : \mathsf{int}}\,\textsc{TPjr} \quad \dfrac{}{\Gamma' \vdash 3 : \mathsf{int}}\,\textsc{TInt}}{\Gamma' \vdash e_{\texttt{if}} : \mathsf{int}}\,\textsc{TIf}}{\vdash fn\ x{:}T_{BA} \Rightarrow e_{\texttt{if}} : T_{BA} \to \mathsf{int}}\,\textsc{TFn} \quad \dfrac{\dfrac{\dfrac{}{\vdash \texttt{true} : \mathsf{bool}}\,\textsc{TBool} \quad \dfrac{}{\vdash 10 : \mathsf{int}}\,\textsc{TInt}}{\vdash \{A = \texttt{true}, B = 10\} : T_{AB}}\,\textsc{TRcd} \quad \dfrac{}{T_{AB} <: T_{BA}}}{\vdash \{A = \texttt{true}, B = 10\} : T_{BA}}\,\textsc{TSub}}{\vdash (fn\ x{:}T_{BA} \Rightarrow e_{\texttt{if}})\ \ \{A = \texttt{true}, B = 10\} : \mathsf{int}}\,\textsc{TAp}$$

Derivação de tipo para a mesma expressão só que agora usando a relação de subtipo entre tipos função (o tipo int *é abreviado para* $\mathtt{i}$ *na derivação abaixo)*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'(x) = T_{BA}}{\Gamma' \vdash x : T_{BA}}\,\textsc{TVar}}{\Gamma' \vdash \sharp B\ x : \mathsf{bool}}\,\textsc{TPjr} \quad \dfrac{\dfrac{\Gamma'(x) = T_{BA}}{\Gamma' \vdash x : T_{BA}}\,\textsc{TVar}}{\Gamma' \vdash \sharp A\ x : \mathtt{i}}\,\textsc{TPjr} \quad \dfrac{}{\Gamma' \vdash 3 : \mathtt{i}}\,\textsc{TInt}}{\Gamma' \vdash e_{\texttt{if}} : \mathtt{i}}\,\textsc{TIf}}{\vdash fn\ x{:}T_{BA} \Rightarrow e_{\texttt{if}} : T_{BA} \to \mathtt{i}}\,\textsc{TFn} \quad \dfrac{\dfrac{}{T_{AB} <: T_{BA}} \quad \dfrac{}{\mathtt{i} <: \mathtt{i}}}{T_{BA} \to \mathtt{i} <: T_{AB} \to \mathtt{i}}}{\vdash fn\ x{:}T_{BA} \Rightarrow e_{\texttt{if}} : T_{AB} \to \mathtt{i}}\,\textsc{TSb} \quad \dfrac{\dfrac{}{\vdash \texttt{true} : \mathsf{bool}}\,\textsc{TBol} \quad \dfrac{}{\vdash 10 : \mathtt{i}}\,\textsc{TInt}}{\vdash \{A = \texttt{true}, B = 10\} : T_{AB}}\,\textsc{TRcd}}{\vdash (fn\ x{:}T_{BA} \Rightarrow e_{\texttt{if}})\{A = \texttt{true}, B = 10\} : \mathtt{i}}\,\textsc{TAp}$$

Exercício 1.30 Repita o exercício anterior para o programa abaixo:

$$(fn\ x{:}\{A : \mathsf{bool}\} \Rightarrow \mathtt{if}\ \sharp A\ x\ \mathtt{then}\ 2\ \mathtt{else}\ 3)\ \ \{A = \mathtt{true}, B = 10\}$$

Resposta: no que segue usamos:

- T_A para o tipo $\{A : \mathsf{bool}\}$
- T_{AB} para o tipo $\{A : \mathsf{bool}, B : \mathsf{int}\}$
- $e_{\mathtt{if}}$ para a expressão $\mathtt{if}\ \sharp A\ x\ \mathtt{then}\ 2\ \mathtt{else}\ 3$
- Γ para o ambiente de tipo $x : T_A$

E começamos com a derivação abaixo onde usamos a relação de subtipos entre tipos registros: (o tipo int é abreviado para i)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma(x) = T_A}{\Gamma \vdash x : T_A}\,\textsc{TVar}}{\Gamma \vdash \sharp B\ x : \mathsf{bool}}\,\textsc{TPjr} \quad \dfrac{\dfrac{\Gamma(x) = T_A}{\Gamma \vdash x : T_A}\,\textsc{TVar}}{\Gamma \vdash \sharp A\ x : \mathtt{i}}\,\textsc{TPjr} \quad \dfrac{}{\Gamma \vdash 3 : \mathtt{i}}\,\textsc{TInt}}{\Gamma \vdash e_{\mathtt{if}} : \mathtt{i}}\,\textsc{TIf}}{\vdash fn\ x{:}T_A \Rightarrow e_{\mathtt{if}} : T_A \to \mathtt{i}}\,\textsc{TFn} \quad \dfrac{\dfrac{\dfrac{}{\vdash \mathtt{true} : \mathsf{bool}}\,\textsc{TBol} \quad \dfrac{}{\vdash 10 : \mathtt{i}}\,\textsc{TInt}}{\vdash \{A = \mathtt{true}, B = 10\} : T_{AB}}\,\textsc{TRcd} \quad \dfrac{}{T_{AB} <: T_A}}{\vdash \{A = \mathtt{true}, B = 10\} : T_A}\,\textsc{TSub}}{\vdash (fn\ x{:}T_A \Rightarrow e_{\mathtt{if}})\ \ \{A = \mathtt{true}, B = 10\} : \mathtt{i}}\,\textsc{TApp}$$

E repetimos usando agora a relação de subtipos entre tipos função:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma(x) = T_A}{\Gamma \vdash x : T_A}\,\textsc{TVar}}{\Gamma \vdash \sharp B\ x : \mathsf{bool}}\,\textsc{TPjr} \quad \dfrac{\dfrac{\Gamma(x) = T_A}{\Gamma \vdash x : T_A}\,\textsc{TVar}}{\Gamma \vdash \sharp A\ x : \mathtt{i}}\,\textsc{TPjr} \quad \dfrac{}{\Gamma' \vdash 3 : \mathtt{i}}\,\textsc{TInt}}{\Gamma \vdash e_{\mathtt{if}} : \mathtt{i}}\,\textsc{TIf}}{\vdash fn\ x{:}T_A \Rightarrow e_{\mathtt{if}} : T_A \to \mathtt{i}}\,\textsc{TFn} \quad \dfrac{\dfrac{}{T_A <: T_{AB}} \quad \dfrac{}{\mathtt{i} <: \mathtt{i}}}{T_{AB} \to \mathtt{i} <: T_A \to \mathtt{i}}}{\vdash fn\ x{:}T_A \Rightarrow e_{\mathtt{if}} : T_{AB} \to \mathtt{i}}\,\textsc{TSub} \quad \dfrac{\dfrac{}{\vdash \mathtt{true} : \mathsf{bool}}\,\textsc{TBol} \quad \dfrac{}{\vdash 10 : \mathtt{i}}\,\textsc{TInt}}{\vdash \{A = \mathtt{true}, B = 10\} : T_{AB}}\,\textsc{TRcd}}{\vdash (fn\ x{:}T_A \Rightarrow e_{\mathtt{if}})\ \ \{A = \mathtt{true}, B = 10\} : \mathsf{int}}\,\textsc{TApp}$$

Exercício 1.31 $\mathtt{Top}$ é supertipo de todos os tipos: ou seja $S <: \mathtt{Top}$ onde S é um tipo qualquer. Quantos supertipos diferentes há para o tipo $\{a : \mathtt{Top}, b : \mathtt{Top}\}$?

Resposta:

São os seis tipos a seguir:
$\{a : \mathtt{Top}, b : \mathtt{Top}\}$, $\{b : \mathtt{Top}, a : \mathtt{Top}\}$, $\{a : \mathtt{Top}\}$, $\{b : \mathtt{Top}\}$, $\{\ \}$, e $\mathtt{Top}$.

1.6 Respostas do Capítulo 7 - Orientação a Objetos

Exercício 1.32 A cópia explícita da maioria dos campos da superclasse no registro da subclasse ainda é inconveniente. Como está evita-se repetir todo o código dos métodos da superclasse na subclasse, mas mesmo assim requer muita digitação. Para programas OO maiores será útil dispormos de uma construção como

$$\texttt{super with}\ \{\texttt{reset} = fn\ \_ {:} \texttt{unit} \Rightarrow r.x := 1\}$$

representando um registro como `super` mas com o campo `reset` redefinido. Defina a sintaxe, semântica operacional e regras de tipo para essa nova construção.