1 Introdução a Semântica Operacional e a Sistema de Tipos

As referências para essa introdução a semântica operacional *small step* são os capítulos 3, 4 e 8 do livro *Types and Programming Languages* de Benjamin Pierce.

1.1 Definições indutivas - sintaxe

Terms é o *menor* conjunto definido pela gramática abstrata abaixo (onde $t, t_1, t_2, \ldots$ Terms):

$$
\begin{array}{rcl}
t & ::= & \texttt{true} \\
 & | & \texttt{false} \\
 & | & \texttt{if } (t_1,\ t_2,\ t_3) \\
 & | & 0 \\
 & | & \texttt{succ } t \\
 & | & \texttt{pred } t \\
 & | & \texttt{iszero } t
\end{array}
$$

ou ainda,

Terms é o *menor* conjunto satisfazendo as regras abaixo

$$\texttt{true} \in \text{Terms}$$

$$\texttt{false} \in \text{Terms}$$

$$\frac{t_1 \in \text{Terms} \qquad t_2 \in \text{Terms} \qquad t_3 \in \text{Terms}}{\texttt{if } (t_1,\ t_2,\ t_3) \in \text{Terms}}$$

$$0 \in \text{Terms}$$

$$\frac{t \in \text{Terms}}{\texttt{succ } (t) \in \text{Terms}}$$

$$\frac{t \in \text{Terms}}{\texttt{pred } (t) \in \text{Terms}}$$

$$\frac{t \in \text{Terms}}{\texttt{iszero } (t) \in \text{Terms}}$$

ou ainda,

Terms é o *menor* conjunto satisfazendo o seguinte:

- $\texttt{true} \in$ Terms, $\texttt{false} \in$ Terms, $0 \in$ Terms
- Se $t_1, t_2, t_3 \in$ Terms então $\texttt{if } (t_1,\ t_2,\ t_3) \in$ Terms
- Se $t \in$ Terms então $\texttt{succ}(t) \in$ Terms, $\texttt{pred}(t) \in$ Terms e $\texttt{iszero}(t) \in$ Terms

As formas alternativas acima definem o conjunto de termos através de suas características/propriedades.

Agora considere o conjunto S *construido* da seguinte forma:

$$
\begin{array}{rcl}
S_0 & = & \{\ \} \\
S_{i+1} & = & \{\texttt{true}, \texttt{false}, 0\} \\
 & \cup & \{\texttt{if}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in S_i\} \\
 & \cup & \{\texttt{succ}(t), \texttt{pred}(t), \texttt{iszero}(t) \mid t \in S_i\}
\end{array}
$$

E finalmente:

$$S = \bigcup_{i=0}^{\infty} S_i$$

Ambos as formas de definir conjuntos (por propriedades ou por construção) são equivalentes:

Teorema 1 $S = \mathsf{Terms}$.

Observe que o conjunto R construido abaixo é um exemplo de conjunto que também satisfaz as regras/propriedades das definições anteriores, exceto que ele não é o *menor* conjunto a satisfazelas:

$$R = \bigcup_{i=0}^{\infty} R_i$$

onde:

$$\begin{array}{lcl} R_0 & = & \{\ \} \\ R_{i+1} & = & \{\texttt{true}, \texttt{false}, 0, \texttt{nop}\} \\ & \cup & \{\texttt{if}(t_1, t_2, t_3) \mid t_1, t_2, t_3 \in R_i\} \\ & \cup & \{\texttt{succ}(t), \texttt{pred}(t), \texttt{iszero}(t), \texttt{twice}(t) \mid t \in R_i\} \end{array}$$

Observe que $S \subseteq R$, ou seja R não é o *menor* conjunto.

1.2 Provas por Indução Estrutural

Abaixo segue a regra de inferência para indução estrutural da linguagem Terms (no estilo de dedução natural):

$$\frac{\begin{array}{ll} & \forall t_1, t_2, t_3.\ P(t_1) \wedge P(t_2) \wedge P(t_3) \Rightarrow P(\texttt{if}(t_1, t_2, t_3)) \\ P(\texttt{true}) & \forall t.\ P(t) \Rightarrow P(\texttt{succ}(t) \\ P(\texttt{false}) & \forall t.\ P(t) \Rightarrow P(\texttt{pred}(t) \\ P(0) & \forall t.\ P(t) \Rightarrow P(\texttt{iszero}(t) \end{array}}{\forall t.\ \ P(t)}$$

As tres premissas empilhadas no lado esquerdo são chamadas de *base da indução*. As quatro premissas empilhadas no lado direito da regra acima são chamadas de *passos indutivos*.

Para provar cada uma das fórmulas correspondentes aos passos indutivos usamos a regra de dedução natural *introdução do* $\forall$ (nome apropriado quando lida "de cima para baixo"). Por exemplo, para provarmos a premissa

$$\forall t_1, t_2, t_3.\ P(t_1) \wedge P(t_2) \wedge P(t_3) \Rightarrow P(\texttt{if}(t_1, t_2, t_3))$$

precisamos provar a fórmula

$$P(t_1) \wedge P(t_2) \wedge P(t_3) \Rightarrow P(\texttt{if}(t_1, t_2, t_3))$$

e essa prova não pode depender de outras hipóteses sobre t_1, t_2 e t_3.

Esta fórmula, por sua vez, é uma implicação. Para prová-la (usando a regra de dedução natural *introdução da implicação*) temos que provar o consequente $P(\texttt{if}(t_1, t_2, t_3))$, podendo fazer uso do antecedente $P(t_1) \wedge P(t_2) \wedge P(t_3)$, caso necessário.

Note que o antecedente da implicação diz que t_1, t_2 e t_3 possuem a propriedade P. Dizemos que $P(t_1)$, $P(t_2)$ e $P(t_3)$ são *hipóteses indutivas* (HI).

Em resumo: para provar que $\forall t.\ P(t)$ é preciso fazer as seguintes subprovas abaixo (é uma boa prática ordenar as provas abaixo de acordo com a ordem em que os termos parecem na gramática):

1. provar $P(\texttt{true})$
2. provar $P(\texttt{false})$
3. provar $P(\texttt{if}(t_1, t_2, t_3))$ - podendo fazer uso das HI $P(t_1)$, $P(t_2)$ e $P(t_3)$
4. provar $P(0)$
5. provar $P(\texttt{succ}(t))$ - podendo fazer uso da HI $P(t)$
6. provar $P(\texttt{pred}(t))$ - podendo fazer uso da HI $P(t)$
7. provar $P(\texttt{iszero}(t))$ - podendo fazer uso da HI $P(t)$

E claro, as provas podem fazer uso de todo conhecimento prévio que não precisa estar explicitado no enunciado do teorema.

A prova de teoremas $\forall t.\ P(t)$, por indução estrutural, tem portanto a seguinte forma:

Teorema 1 (nome do teorema) $\forall t.\ P(t)$.

Prova. Por indução na estrutura de t.

caso $\texttt{true}$

aqui vai a prova de $P(\texttt{true})$

caso $\texttt{false}$

aqui vai a prova de $P(\texttt{false})$

caso $\texttt{if}(t_1, t_2, t_3)$

aqui vai a prova de $P(\texttt{if}(t_1, t_2.t_3))$ *(pode usar as HIs* $P(t_1)$, $P(t_2)$, $P(t_3)$*)*

caso 0

aqui vai a prova de $P(0)$

caso $\texttt{succ}(t)$

aqui vai a prova de $P(\texttt{succ}(t))$ *que pode fazer uso da HI* $P(t)$

caso $\texttt{pred}(t)$

aqui vai a prova de $P(\texttt{pred}(t))$ *que pode fazer uso da HI* $P(t)$

caso $\texttt{iszero}(t)$

aqui vai a prova de $P(\texttt{iszero}(t))$ *que pode fazer uso da HI* $P(t)$

■

1.3 Definições indutivas - semântica operacional

Vamos continuar com mais definições indutivas. Veremos agora a definição de um conjunto de pares ordenados de termos para o qual daremos o seguinte nome:

$$\longrightarrow \quad \text{ou} \quad \mathsf{step}$$

Temos portanto que:

$$\longrightarrow \subseteq \mathsf{Terms} \times \mathsf{Terms}$$

Para um par de termos (t, t') pertencer ao conjunto $\longrightarrow$ o termo t deve *avaliar para o termo* t' *em um passo*. Precisamos, antes de mais nada, termos uma compreensão intuitiva do que é esse *avaliar em um passo* e ao formalizarmos essa compreensão intuitiva teremos definido a *semântica operacional estrutural* (mais conhecida como *semântica operacional small step*) de Terms.

Alguns termos não são avaliados para outros termos pois já estão *prontos* Chamaremos esses termos prontos de *valores*. Para a linguagem em questão os termos *valores* são `true` e `false` (valores booleanos) e $0, \texttt{succ}(0), \texttt{succ}(\texttt{succ}(0)), \ldots$. (valores numéricos).

Por hora vejamos alguns exemplo de pares que **devem** estar no conjunto $\longrightarrow$:

- $(\texttt{iszero}(0),\ \texttt{true}) \in \longrightarrow$
- $(\texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0)),\ 0) \in \longrightarrow$
- $(\texttt{if}(\texttt{iszero}(0),\ 0,\ \texttt{succ}(0)),\ \texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0))) \in \longrightarrow$

De agora em diante vamos preferir escrever $t \longrightarrow t'$ sempre que $(t,\ t') \in \longrightarrow$ (ou ainda $\mathsf{step}(t,\ t')$ em algumas situações). Logo os exemplos acima podem ser reescritos na forma:

- $\texttt{iszero}(0) \longrightarrow \texttt{true}$
- $\texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0)) \longrightarrow 0$
- $\texttt{if}(\texttt{iszero}(0),\ 0,\ \texttt{succ}(0)) \longrightarrow \texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0))$

E abaixo eis alguns exemplos de pares em $\mathsf{Terms} \times \mathsf{Terms}$ que **não devem** pertencer a $\longrightarrow$: (por que?)

- $(\texttt{iszero}(0),\ \texttt{false}) \notin \longrightarrow$
- $(\texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0)),\ \texttt{succ}(0)) \notin \longrightarrow$
- $(\texttt{if}(\texttt{iszero}(0),\ 0,\ \texttt{succ}(0)),\ 0) \notin \longrightarrow$
- $(\texttt{true},\ \texttt{true}) \notin \longrightarrow$
- $(\texttt{iszero}(\texttt{false}),\ 0) \notin \longrightarrow$

Exercício 1 Para cada um dos pares de termos acima explique os motivos pelos quais eles não estão na relação $\longrightarrow$.

Exercício 2 De mais exemplos de pares de termos que pertencem e pares de termos que não pertencem ao conjunto $\longrightarrow$ (observe que, a noção de *avaliar em um passo* ainda não foi definida precisamente. Os exemplo acima nos dão uma compreensão incompleta sobre ela. Sendo assim, estritamente falando, não há como dizer se os exemplos dados estão certos ou errados).

Tendo uma compreensão intuitiva do que deve e do que não deve estar nesse conjunto o próximo passo é partir para uma definição precisa. Um método sistemático para chegarmos a definição desse conjunto de pares $\longrightarrow$ (também chamado de *relação*) é usar a gramática abstrata como um guia. Dessa forma podemos nos certificar de que não esqueceremos de considerar cada uma das possibilidades.

caso `true`

Como vimos nos exemplos anteriores, o termo `true` **não deve** estar relacionado a qualquer outro termo do conjunto Terms pois ele é um valor. Dessa forma não há nenhuma regra a ser explicitada para este caso.

caso `false`

De forma semelhante ao caso anterior, o termo `false` **não deve** estar relacionado a qualquer outro termo do conjunto Terms. Aqui também não há nenhuma regra a ser explicitada para este caso.

caso `if` $(t_1,\ t_2,\ t_3)$

Termos com essa estrutura podem estar relacionados com outros termos na relação de um passo desde que as seguintes condições sejam satisfeitas

$$\frac{t_1 = \texttt{true}}{\texttt{if}\ (t_1,\ t_2,\ t_3) \longrightarrow t_2} \qquad \text{(E-IfTrue)}$$

$$\frac{t_1 = \texttt{false}}{\texttt{if}\ (t_1,\ t_2,\ t_3) \longrightarrow t_3} \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ (t_1,\ t_2,\ t_3) \longrightarrow \texttt{if}\ (t_1',\ t_2,\ t_3)} \qquad \text{(E-If)}$$

Observe que essas regras são formas mais convenientes/abreviadas que expressam o mesmo que as seguintes fórmulas:

$$\forall t_1, t_2, t_3.\ \ t_1 = \texttt{true} \ \Rightarrow\ \ \texttt{if}(t_1, t_2, t_3) \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\forall t_1, t_2, t_3.\ \ t_1 = \texttt{false} \ \Rightarrow\ \ \texttt{if}(t_1, t_2, t_3) \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

$$\forall t_1, t_2, t_3.\ \ t_1 \longrightarrow t_1' \ \Rightarrow\ \ \texttt{if}(t_1, t_2,\ t_3) \longrightarrow \texttt{if}(t_1', t_2, t_3) \qquad \text{(E-If)}$$

E aqui temos uma oportunidade para usar o nome step para evitar alguma confusão entre a relaçao de avaliação em um passo e a implicação da lógica:

$$\forall t_1, t_2, t_3.\ \ t_1 = \texttt{true} \ \Rightarrow\ \ \mathsf{step}(\texttt{if}(t_1, t_2, t_3),\ t_2) \qquad \text{(E-IfTrue)}$$

$$\forall t_1, t_2, t_3.\ \ t_1 = \texttt{false} \ \Rightarrow\ \ \mathsf{step}(\texttt{if}(t_1, t_2, t_3), t_3) \qquad \text{(E-IfFalse)}$$

$$\forall t_1, t_2, t_3.\ \ \mathsf{step}(t_1,\ t_1') \ \Rightarrow\ \ \mathsf{step}(\texttt{if}(t_1, t_2, t_3),\ \texttt{if}(t_1', t_2, t_3)) \qquad \text{(E-If)}$$

As regras podem ficar mais compactas se as condições expressas nas premissas das regras forem representadas da seguinte forma:

$$\frac{}{\texttt{if}\ (\texttt{true},\ t_2,\ t_3) \longrightarrow t_2} \qquad \text{(E-IfTrue)}$$

$$\frac{}{\texttt{if}\ (\texttt{false},\ t_2,\ t_3) \longrightarrow t_3} \quad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ (t_1,\ t_2,\ t_3) \longrightarrow \texttt{if}\ (t_1',\ t_2,\ t_3)} \quad \text{(E-IfFalse)}$$

caso 0

De forma semelhante aos casos para `true` e `false` o termo 0 **não deve** estar relacionado a qualquer outro termo do conjunto Terms pois ele é um **valor**. Aqui também não há nenhuma regra a ser explicitada.

caso $\texttt{succ}(t)$

Termos com essa estrutura podem estar relacionados com outros termos na relação de um passo desde que a seguinte condições seja satisfeitas

$$\frac{t \longrightarrow t'}{\texttt{succ}(t) \longrightarrow \texttt{succ}(t')} \quad \text{(E-Succ)}$$

Essa regra expressa o mesmo que a seguinte fórmula (usando step ao invés de $\longrightarrow$):

$$\forall t.\ \ \mathsf{step}(t,\ t') \quad \Rightarrow \quad \mathsf{step}(\texttt{succ}(t), \texttt{succ}(t')) \quad \text{(E-Succ)}$$

caso $\texttt{pred}(t)$

$$\frac{t = 0}{\texttt{pred}(t) \longrightarrow 0} \quad \text{(E-PredZero)}$$

$$\frac{t = \texttt{succ}(nv)}{\texttt{pred}(t) \longrightarrow\ nv} \quad \text{(E-PredSucc)}$$

$$\frac{t \longrightarrow t'}{\texttt{pred}(t) \longrightarrow \texttt{succ}(t')} \quad \text{(E-Pred)}$$

Com step ao invés de $\longrightarrow$:

$$\forall t.\ \ t = 0 \quad \Rightarrow \quad \mathsf{step}(\texttt{pred}(t), 0) \qquad \text{(E-PredZero)}$$

$$\forall t.\ \ t = \texttt{succ}(nv) \quad \Rightarrow \quad \mathsf{step}(\texttt{pred}(t), nv) \qquad \text{(E-PredSucc)}$$

$$\forall t.\ \ \mathsf{step}(t,\ t') \quad \Rightarrow \quad \mathsf{step}(\texttt{pred}(t), \texttt{pred}(t')) \qquad \text{(E-Pred)}$$

As regras podem ser escritas de forma mais compacta:

$$\frac{}{\texttt{pred}(0) \longrightarrow 0} \quad \text{(E-PredZero)}$$

$$\frac{}{\texttt{pred}(\texttt{succ}(nv)) \longrightarrow\ nv} \quad \text{(E-PredSucc)}$$

$$\frac{t \longrightarrow t'}{\texttt{pred}(t) \longrightarrow \texttt{pred}(t')} \quad \text{(E-Pred)}$$

caso $\texttt{iszero}(t)$

$$\frac{t = 0}{\texttt{iszero}(t) \longrightarrow \texttt{true}} \qquad \text{(E-IsZeroZero)}$$

$$\frac{t = \texttt{succ}(nv)}{\texttt{iszero}(t) \longrightarrow \texttt{false}} \qquad \text{(E-IsZeroSucc)}$$

$$\frac{t \longrightarrow t'}{\texttt{iszero}(t) \longrightarrow \texttt{iszero}(t')} \qquad \text{(E-IsZero)}$$

E usando step ao invés de $\longrightarrow$:

$$\forall t.\ \ t = 0 \ \ \Rightarrow \ \ \mathsf{step}(\texttt{iszero}(t), \texttt{true}) \qquad \text{(E-IsZeroZero)}$$

$$\forall t.\ \ t = \texttt{succ}(nv) \ \ \Rightarrow \ \ \mathsf{step}(\texttt{iszero}(t), \texttt{false}) \qquad \text{(E-IsZeroSucc)}$$

$$\forall t.\ \ \mathsf{step}(t,\ t') \ \ \Rightarrow \ \ \mathsf{step}(\texttt{iszero}(t), \texttt{iszero}(t')) \qquad \text{(E-IsZero)}$$

As regras podem ser escritas de forma mais compacta:

$$\frac{}{\texttt{iszero}(0) \longrightarrow \texttt{true}} \qquad \text{(E-IsZeroZero)}$$

$$\frac{}{\texttt{iszero}(\texttt{succ}(nv)) \longrightarrow \texttt{false}} \qquad \text{(E-IsZeroSucc)}$$

$$\frac{t \longrightarrow t'}{\texttt{iszero}(t) \longrightarrow \texttt{iszero}(t')} \qquad \text{(E-IsZero)}$$

Segue abaixo um resumo da definição da semântica operacional de Terms na notação compacta:

Sintaxe

$$t \quad ::= \quad \texttt{true} \mid \texttt{false} \mid \texttt{if}\ (t_1, t_2, t_3) \mid 0 \mid \texttt{succ}(t) \mid \texttt{pred}(t) \mid \texttt{iszero}(t)$$

$$v \quad ::= \quad \texttt{true} \mid \texttt{false} \mid nv$$
$$nv \quad ::= \quad 0 \mid \texttt{succ}(nv)$$

Semântica Operacional

$$\frac{}{\texttt{if}\ (\texttt{true},\ t_2,\ t_3) \longrightarrow t_2} \qquad \text{(E-IfTrue)}$$

$$\frac{}{\texttt{if}\ (\texttt{false},\ t_2,\ t_3) \longrightarrow t_3} \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ (t_1,\ t_2,\ t_3) \longrightarrow \texttt{if}\ (t_1',\ t_2,\ t_3)} \qquad \text{(E-If)}$$

$$\frac{t \longrightarrow t'}{\texttt{succ}(t) \longrightarrow \texttt{succ}(t')} \qquad \text{(E-Succ)}$$

$$\frac{}{\texttt{pred}(0) \longrightarrow 0} \qquad \text{(E-PredZero)}$$

$$\frac{}{\texttt{pred}(\texttt{succ}(nv)) \longrightarrow nv} \qquad \text{(E-PredSucc)}$$

$$\frac{t \longrightarrow t'}{\texttt{pred}(t) \longrightarrow \texttt{pred}(t')} \qquad \text{(E-Pred)}$$

$$\frac{}{\texttt{iszero}(0) \longrightarrow \texttt{true}} \qquad \text{(E-IsZeroZero)}$$

$$\frac{}{\texttt{iszero}(\texttt{succ}(nv)) \longrightarrow \texttt{false}} \qquad \text{(E-IsZeroSucc)}$$

$$\frac{t \longrightarrow t'}{\texttt{iszero}(t) \longrightarrow \texttt{iszero}(t')} \qquad \text{(E-IsZero)}$$

É preciso também definir a relação $\longrightarrow^*$ (ou ainda steps). Assim como a relação $\longrightarrow$, a relação também é composta por um conjunto de pares ordenados de termos:

$$\longrightarrow^* \subseteq \mathsf{Terms} \times \mathsf{Terms}$$

Para um par de termos (t, t') pertencer a $\longrightarrow^*$ o termo t deve avaliar para o termo t' em *zero ou mais passos*, onde a noção de *passo* corresponde a definição de $\longrightarrow^*$. Exemplo de pares que **devem** estar nesse conjunto:

- $(\texttt{iszero}(0),\ \texttt{true}) \in \longrightarrow^*$
- $(\texttt{true},\ \texttt{true}) \notin \longrightarrow$
- $(\texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0)),\ 0) \in \longrightarrow^*$
- $(\texttt{if}(\texttt{iszero}(0),\ 0,\ \texttt{succ}(0)),\ \texttt{if}(\texttt{true},\ 0,\ \texttt{succ}(0))) \in \longrightarrow^*$
- $(\texttt{if}(\texttt{iszero}(0),\ 0,\ \texttt{succ}(0)),\ 0) \notin \longrightarrow$

Segue abaixo a definição da relação $\longrightarrow^*$:

$$\frac{t \longrightarrow t'}{t \longrightarrow^* t'}$$

$$\frac{}{t \longrightarrow^* t}$$

$$\frac{t \longrightarrow^* t' \qquad t' \longrightarrow^* t''}{t \longrightarrow^* t''}$$

A relação $\longrightarrow^*$ é portanto o *fecho reflexivo e transitivo* da relação $\longrightarrow$.

1.4 Propriedades

Exercício 3 Um termo está em *forma normal* se nenhuma regra da semântica operacional se aplica a ele. Formalizar a definição de *termo em forma normal* em lógica de predicados

Exercício 4 Formalizar cada uma das afirmações abaixo e investigar se a linguagen Terms possui a propriedade expressa pela afirmação. Para os casos negativos dar um contra-exemplo e para os casos afirmativos fornecer uma prova formal:

1. Todo termo tem no máximo um termo como resultado da avaliação em um passo
2. Todo valor da linguagem está em *forma normal*
3. Toda forma normal da linguagem é um valor
4. A avaliação de todo termo sempre termina
5. A forma normal de um termo é única

Definição 1 (Forma Normal) Um termo t está em forma normal se nenhuma regra da semântica operacional se aplica a ele.

$$\mathsf{FN}(t) \quad \equiv \quad \neg\exists t'.\ \mathsf{step}(t, t')$$

Teorema 2 (Determinismo) Se $t \longrightarrow t'$ e $t \longrightarrow t''$ então $t' = t''$.

$$\forall t, t', t''.\ \mathsf{step}(t, t') \wedge \mathsf{step}(t, t'') \Rightarrow t' = t''$$

O teorema acima significa que a relação de avaliação em um passo é uma função de elementos de Terms para Terms (mostre que essa função não é total).

Teorema 3 Todo valor de Terms é uma forma normal.

$$\forall t.\ \mathsf{valor}(t) \rightarrow \mathsf{FN}(t)$$

1.5 Animação das regras da semântica operacional

Abaixo segue código em OCaml com a implementação da avaliação de termos da linguagem Terms. Observe que esse programa segue **estritamente** as regras.

Exercício 5 Experimente com o programa abaixo na interface interativa do OCaml chamando a função *step* e *eval* para árvores abstrata de teste (crie suas próprias árvores abstratas além daquelas ao final do código).

Exercício 6 Modifique o programa de tal forma que ele imprimia o termo inicial submetido para avaliação e também o termo que resultou da avaliação dizendo se ele é um valor ou um *erro de execução*.

```
(* Gramatica:

    t ::= true
        | false
        | if(t1,t2,t3)
        | 0
        | succ(t)
        | pred(t)
        | iszero(t)
```

```
 *)

type term =
    TmTrue
  | TmFalse
  | TmIf of  term * term * term
  | TmZero
  | TmSucc  of term
  | TmPred  of term
  | TmIsZero of  term

(* Exceção a ser ativada quando termo for uma FORMA NORMAL. O que, para
   essa linguagem significa que:
     termo pode ser um VALOR, ou
     termo pode ser um ERRO de EXECUÇÃO
*)

exception NoRuleApplies

(* Função auxiliar para determinar se um termo é um VALOR NUMÉRICO *)

let rec isnumericval t = match t with
    TmZero → true
  | TmSucc(t1) → isnumericval t1
  | _ → false

  (* Implementação da função STEP /     → de avaliação em um passo *)

let rec step t = match t with

(* CASO IF(t1,t2,t3)    *)

    TmIf(TmTrue,t2,t3) → (* regra E-IfTrue *)
      t2
  | TmIf(TmFalse,t2,t3) → (* regra E-False *
      t3
  | TmIf(t1,t2,t3) →  (* regra E-If *)
      let t1' = step t1 in
      TmIf(t1', t2, t3)

 (* CASO SUCC(t1)    *)

  | TmSucc(t1) →  (* regra E-Succ *)
      let t1' = step t1 in
      TmSucc(t1')

 (* CASO  PRED(t1)    *)

  | TmPred(TmZero) → (* regra E-PredZero *)
      TmZero
  | TmPred(TmSucc(nv1)) when (isnumericval nv1) → (* regra E-PredSucc *)
      nv1
```

```
  | TmPred(t1) → (* regra E-Pred *)
      let t1' = step t1 in
      TmPred(t1')

 (* CASO ISZERO(t1)    *)

  | TmIsZero(TmZero) → (* regra E-IsZeroZero *)
      TmTrue
  | TmIsZero(TmSucc(nv1)) when (isnumericval nv1) →  (* regra E-IsZeroSucc *)
      TmFalse
  | TmIsZero(t1) → (* regra E-IsZero *)
      let t1' = step t1 in
      TmIsZero(t1')

(* CASO Nenhuma regra se aplique ao termo  *)

  | _ →
      raise NoRuleApplies

(* Implementação de EVAL  *)

let rec eval t =
  try let t' = step t
      in eval t'
  with NoRuleApplies → t

(*  ASTs para teste *)

let t1 =  TmIsZero(TmZero)
let t2 =  TmZero
let t3 =  TmSucc(TmZero)
let tif =  TmIf(t1,t2,t3)

let t4 =  TmIsZero(TmSucc(TmZero))
let t5 =  TmIsZero(TmFalse)
```

1.6 Sistema de Tipos

Vimos nas seções anteriores que a avaliação de todo termo da linguagem Terms sempre termina produzindo uma forma normal e que essa forma normal é única. Para a linguagem Terms a forma normal poder ser um valor ou pode ser o que chamaremos de *erro de execução* ou ainda *stuck term*.

Definição 2 Na linguagem Terms um termo é considerado um *erro de execução* se ele estiver em forma normal e não for um valor.

Exercício 7 De 5 exemplos de termos que são ou que, quando avaliados, leverão a *erros de execução*.

$$T \quad ::= \quad \mathsf{nat} \mid \mathsf{bool}$$

Vamos definir uma relação binária ":"entre um termo t e um tipo T. Escreve-se $t : T$ e lê-se *o termo t é do tipo T*:

$$: \subseteq \mathsf{Terms} \times \{\mathsf{nat}, \mathsf{bool}\}$$

A relação de tipagem " : " é a menor relação contida em $\mathsf{Terms} \times \{\mathsf{nat}, \mathsf{bool}\}$ que satisfaz as seguintes propriedades:

$$\texttt{true} : \mathsf{bool} \qquad \text{(T-True)}$$

$$\texttt{false} : \mathsf{bool} \qquad \text{(T-False)}$$

$$\forall t_1, t_2, t_3, T.\ \ t_1 : \mathsf{bool} \ \wedge\ t_2 : T \ \wedge\ t_3 : T \ \ \Rightarrow\ \ \texttt{if}(t_1, t_2, t_3) : T \qquad \text{(T-If)}$$

$$0 : \mathsf{nat} \qquad \text{(T-Zero)}$$

$$\forall t.\ \ t : \mathsf{nat} \ \ \Rightarrow\ \ \texttt{succ}(t) : \mathsf{nat} \qquad \text{(T-Succ)}$$

$$\forall t.\ \ t : \mathsf{nat} \ \ \Rightarrow\ \ \texttt{pred}(t) : \mathsf{nat} \qquad \text{(T-Pred)}$$

$$\forall t.\ \ t : \mathsf{nat} \ \ \Rightarrow\ \ \texttt{iszero}(t) : \mathsf{bool} \qquad \text{(T-IsZero)}$$

Na forma de regras de inferência temos:

$$\frac{}{\texttt{true} : \mathsf{bool}} \qquad \text{(T-True)}$$

$$\frac{}{\texttt{false} : \mathsf{bool}} \qquad \text{(T-False)}$$

$$\frac{t_1 : T_1 \qquad t_2 : T_2 \qquad t_3 : T_3 \qquad T_1 = \mathsf{bool} \qquad T_2 = T_3}{\texttt{if}(t_1, t_2, t_3) : T_2} \qquad \text{(T-If)}$$

$$\frac{}{0 : \mathsf{nat}} \qquad \text{(T-Zero)}$$

$$\frac{t : T \qquad T = \mathsf{nat}}{\texttt{succ}(t) : \mathsf{nat}} \qquad \text{(T-Succ)}$$

$$\frac{t : T \qquad T = \mathsf{nat}}{\texttt{pred}(t) : \mathsf{nat}} \qquad \text{(T-Pred)}$$

$$\frac{t : T \qquad T = \mathsf{nat}}{\texttt{iszero}(t) : \mathsf{bool}} \qquad \text{(T-IsZero)}$$

E em uma forma mais compacta:

$$\frac{}{\texttt{true} : \mathsf{bool}} \qquad \text{(T-True)}$$

$$\frac{t_1 : \mathsf{bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if}(t_1, t_2, t_3) : T} \qquad \text{(T-If)}$$

$$\frac{t : \mathsf{nat}}{\texttt{succ}(t) : \mathsf{nat}} \qquad \text{(T-Succ)}$$

$$\frac{t : \mathsf{nat}}{\texttt{iszero}(t) : \mathsf{bool}} \qquad \text{(T-IsZero)}$$

$$\frac{}{\texttt{false} : \mathsf{bool}} \qquad \text{(T-False)}$$

$$\frac{}{0 : \mathsf{nat}} \qquad \text{(T-Zero)}$$

$$\frac{t : \mathsf{nat}}{\texttt{pred}(t) : \mathsf{nat}} \qquad \text{(T-Pred)}$$

Esse conjunto de regras especifica diretamente a parte central de um *verificador de tipos* que pode ser incorporado, juntamente com o avaliador de expressões, em um interpretador. A idéia é que o avaliador de expressões só seja executado se o verificador de tipos, após a análise da árvore de sintaxe abstrata, conclui que o termo é bem tipado.

Exercício 8 Implemente em OCaml uma função de verificação de tipos que siga estritamente o que as regras do sistema de tipos prescrevem.

1.7 Propriedades do Sistema de Tipos

Exercício 9 Formalize, investigue se são verdadeira e prove/de um contra-exemplo para as seguintes afirmações:

1. Todo termo tem no máximo um tipo (unicidade de tipo)
2. se um termo é bem tipado ele é um valor ou ele progride em um passo (progresso de termos bem tipados)
3. se um termo bem tipado progride em um passo o termo resultante também é bem tipado e do mesmo tipo (preservação de tipo com progressão)

Exercício 10 O sistema de tipos definido especifica uma análise estática, ou seja, uma análise que é feita sobre a árvore de sintaxe abstrata sem executar/avaliar termos. Dessa forma ele é necessariamente *conservador* (por que?) e pode concluir que um termo é *mal tipado* mesmo que a avaliação desse termo **não** leve a *erro de execução*. De exemplos de termos que ilustram o caráter conservador do sistema de tipos.

Teorema 4 (Unicidade de Tipo) Se $t : T$ e $t : T'$ então $T = T'$.

(ou seja: $\forall t,\ \forall T, T'.\ \ t : T \ \ \wedge \ \ t : T' \ \ \Rightarrow \ \ T = T'$)

Teorema 5 (Progresso) Se $t : T$ então (i) t é valor ou (ii) existe t' tal que $t \longrightarrow t'$.

(ou seja: $\forall t.\ \forall T.\ \ t : T \ \Rightarrow \ (\mathsf{valor}(t) \ \vee \ \exists t'.\ t \longrightarrow t')$.

Prova. Por indução na estrutura de t.

$P(t) \ \equiv \ \forall T.\ \boxed{\mathrm{t}} : T \ \Rightarrow \ (\mathsf{valor}(\boxed{\mathrm{t}}) \ \vee \ \exists t'.\ \boxed{\mathrm{t}} \longrightarrow t')$.

caso `true`

(*prova de* $P($`true`$)$ *ou seja prova de* $\forall T.\ $ `true` $: T \ \Rightarrow \ (\mathsf{valor}($`true`$) \ \vee \ \exists t'.$ `true` $\longrightarrow t')$.)

Após a aplicaçao da regra $\forall I$ na fórmula acima, temos que provar a seguinte fórmula:

$$\texttt{true} : T \ \Rightarrow \ (\mathsf{valor}(\texttt{true}) \ \vee \ \exists t'.\ \texttt{true} \longrightarrow t') \tag{1}$$

Pela regra $\Rightarrow I$ aplicada a fórmula (1) acima assumimos `true` $: T$ e temos que provar

$$\mathsf{valor}(\texttt{true}) \ \vee \ \exists t'.\ \texttt{true} \longrightarrow t'$$

Que é trivialmente verdadeira pois $\mathsf{valor}($`true`$)$ é verdadeiro.

caso `false`

(*prova de* $P($`false`$)$ *ou seja prova de* $\forall T.\ $ `false` $: T \ \Rightarrow \ (\mathsf{valor}($`false`$) \ \vee \ \exists t'.$ `false` $\longrightarrow t')$.)

Essa prova segue exatamente a mesma estrutura do caso anterior trocando `true` por `false`.

caso $\texttt{if}(t_1, t_2, t_3)$

(*prova de* $P(\texttt{if}(t_1, t_2, t_3))$ *ou seja prova de*:

$\forall T.\ \ \texttt{if}(t_1, t_2, t_3) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{if}(t_1, t_2, t_3)) \ \vee\ \exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t')$.

podendo usar as seguintes hipóteses indutivas, caso necessário:

$P(t_1) \ \equiv\ \forall T.\ \ t_1 : T \ \Rightarrow\ (\mathsf{valor}(t_1) \ \vee\ \exists t'.\ t_1 \longrightarrow t')$,
$P(t_2) \ \equiv\ \forall T.\ \ t_2 : T \ \Rightarrow\ (\mathsf{valor}(t_2) \ \vee\ \exists t'.\ t_2 \longrightarrow t')$,
$P(t_3) \ \equiv\ \forall T.\ \ t_3 : T \ \Rightarrow\ (\mathsf{valor}(t_3) \ \vee\ \exists t'.\ t_3 \longrightarrow t'))$

Após a aplicaçao da regra $\forall I$ na fórmula acima, temos que provar a seguinte fórmula:

$$\texttt{if}(t_1, t_2, t_3) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{if}(t_1, t_2, t_3)) \ \vee\ \exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t') \tag{2}$$

Pela regra $\Rightarrow I$ aplicada a (2) acima assumimos:

$$\texttt{if}(t_1, t_2, t_3) : T \tag{3}$$

e temos que provar

$$\mathsf{valor}(\texttt{if}(t_1, t_2, t_3)) \ \vee\ \exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t'$$

Como $\mathsf{valor}(\texttt{if}(t_1, t_2, t_3))$ é evidentemente falso para provar essa disjunção acima temos que provar:

$$\exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t' \tag{4}$$

De (3) acima e pela regra de tipo T-If temos que:

$$t_1 : \mathsf{bool} \tag{5}$$

Pela hipótese indutiva para t_1 e pela regra $\forall E$ (instanciando T com bool) temos:

$$t_1 : \mathsf{bool} \ \Rightarrow\ (\mathsf{valor}(t_1) \ \vee\ \exists t'.\ t_1 \longrightarrow t') \tag{6}$$

Por *modus ponens* com (5) e (6) acima temos:

$$\mathsf{valor}(t_1) \ \vee\ \exists t'.\ t_1 \longrightarrow t' \tag{7}$$

Por $\vee E$, para provar (4) a partir da disjunção (7) temos que provar (4) a partir de cada um dos disjuntos:

subcaso $\mathsf{valor}(t_1)$:

Por (5) e pelas regras do sistema de tipos temos que $t_1 = \texttt{true}$ ou $t_1 = \texttt{false}$.

Se $t_1 = \texttt{true}$: pela regra E-IfTrue com $t_1 = \texttt{true}$ como premissa temos que:

$$\texttt{if}(t_1, t_2, t_3) \longrightarrow t_2 \tag{8}$$

Por (8) acima e pela regra $\exists I$ provamos, como desejado, que:

$$\exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t'$$

Se $t_1 = \texttt{false}$: pela regra E-IfFalse com $t_1 = \texttt{false}$ como premissa temos que:

$$\texttt{if}(t_1, t_2, t_3) \longrightarrow t_3 \tag{9}$$

Por (9) acima e pela regra $\exists I$ provamos, como desejado, que:

$$\exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t'$$

subcaso $\exists t'.\ t_1 \longrightarrow t'$:

Pela regra $\exists E$ usando t_1' como *elemento representativo*, temos que

$$t_1 \longrightarrow t_1' \tag{10}$$

Com (10) como premissa para a regra E-If temos que:

$$\texttt{if}(t_1, t_2, t_3) \longrightarrow \texttt{if}(t_1', t_2, t_3) \tag{11}$$

E por (11) e pela regra $\exists I$ provamos como desejado que;

$$\exists t'.\ \texttt{if}(t_1, t_2, t_3) \longrightarrow t'$$

caso 0

prova de $P(0)$ ou seja prova de $\forall T.\ \ 0 : T \ \Rightarrow\ (\mathsf{valor}(0)\ \vee\ \exists t'.\ 0 \longrightarrow t')$.

Essa prova segue exatamente a mesma estrutura do caso para `true`, trocando `true` por 0.

caso $\texttt{succ}(t)$

(*prova de* $P(\texttt{succ}(t))$ *ou seja prova de*:

$\forall T.\ \ \texttt{succ}(t) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{succ}(t))\ \vee\ \exists t'.\ \texttt{succ}(t) \longrightarrow t')$.

podendo usar a seguinte hipótese indutiva, caso necessário:

$P(t)\ \ \equiv\ \ \forall T.\ \ t : T \ \Rightarrow\ (\mathsf{valor}(t)\ \vee\ \exists t'.\ t \longrightarrow t'))$

Após a aplicaçao da regra $\forall I$ na fórmula acima, temos que provar a seguinte fórmula:

$$\texttt{succ} : T \ \Rightarrow\ (\mathsf{valor}(\texttt{succ}(t))\ \vee\ \exists t'.\ \texttt{succ}(t) \longrightarrow t') \tag{12}$$

Pela regra $\Rightarrow I$ aplicada a (2) acima assumimos:

$$\texttt{succ}(t) : T \tag{13}$$

e temos que provar

$$\mathsf{valor}(\texttt{succ}(t))\ \vee\ \exists t'.\ \texttt{succ}(t) \longrightarrow t' \tag{14}$$

De (13) acima e pela regra de tipo T-Succ temos que:

$$t : \mathsf{nat} \tag{15}$$

Pela hipótese indutiva para t e pela regra $\forall E$ (instanciando T com nat) temos:

$$t : \mathsf{nat} \ \Rightarrow\ (\mathsf{valor}(t)\ \vee\ \exists t'.\ t \longrightarrow t') \tag{16}$$

Por *modus ponens* com (15) e (16) acima temos:

$$\mathsf{valor}(t)\ \vee\ \exists t'.\ t \longrightarrow t' \tag{17}$$

Por $\vee E$, para provar (14) a partir da disjunção (17) temos que provar (14) a partir de cada um dos disjuntos:

subcaso $\mathsf{valor}(t)$:

Por (15) e pelas regras do sistema de tipos temos que $t = 0$ ou $t = \texttt{succ}(nv)$ onde nv é valor numérico. Em ambos os casos temos que $\mathsf{valor}(\texttt{succ}(t))$ é verdadeiro provando assim (14).

subcaso $\exists t'.\ t \longrightarrow t'$:

Pela regra $\exists E$ usando t' como *elemento representativo*, temos que

$$t \longrightarrow t' \tag{18}$$

Com (18) como premissa para a regra E-Succ temos que:

$$\texttt{succ}(t) \longrightarrow \texttt{succ}(t') \tag{19}$$

E por (19) e pela regra $\exists I$ provamos como desejado que

$$\exists t'.\ \texttt{succ}(t) \longrightarrow t'$$

caso $\texttt{pred}(t)$

prova de $P(\texttt{pred}(t))$ ou seja prova de:

$\forall T.\ \ \texttt{pred}(t) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{pred}(t)) \ \vee\ \exists t'.\ \texttt{pred}(t) \longrightarrow t')$.

podendo usar a seguinte hipótese indutiva, caso necessário:

$P(t) \ \equiv\ \forall T.\ \ t : T \ \Rightarrow\ (\mathsf{valor}(t) \ \vee\ \exists t'.\ t \longrightarrow t')$

Após a aplicaçao da regra $\forall I$ na fórmula acima, temos que provar a seguinte fórmula:

$$\texttt{pred}(t) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{pred}(t)) \ \vee\ \exists t'.\ \texttt{pred}(t) \longrightarrow t') \tag{20}$$

Pela regra $\Rightarrow I$ aplicada a (20) acima assumimos:

$$\texttt{pred}(t) : T \tag{21}$$

e temos que provar

$$\mathsf{valor}(\texttt{pred}(t)) \ \vee\ \exists t'.\ \texttt{pred}(t) \longrightarrow t'$$

Como $\mathsf{valor}(\texttt{pred}(t))$ é evidentemente falso para provar essa disjunção acima temos que provar:

$$\exists t'.\ \texttt{pred}(t) \longrightarrow t' \tag{22}$$

De (21) acima e pela regra de tipo T-Pred temos que:

$$t : \mathsf{nat} \tag{23}$$

Pela hipótese indutiva para t e pela regra $\forall E$ (instanciando T com nat) temos:

$$t : \mathsf{nat} \ \Rightarrow\ (\mathsf{valor}(t) \ \vee\ \exists t'.\ t \longrightarrow t') \tag{24}$$

Por *modus ponens* com (23) e (24) acima temos:

$$\mathsf{valor}(t) \ \vee\ \exists t'.\ t \longrightarrow t' \tag{25}$$

Por $\vee E$, para provar (22) a partir da disjunção (25) temos que provar (22) a partir de cada um dos disjuntos:

subcaso $\mathsf{valor}(t)$:

Por (23) e pelas regras do sistema de tipos temos que $t = 0$ ou $t = \texttt{succ}(nv)$ com nv um valor numérico.

Se $t = 0$: pela regra E-PredZero com $t = 0$ como premissa temos que:

$$\texttt{pred}(t) \longrightarrow 0 \tag{26}$$

Por (26) acima e pela regra $\exists I$ provamos, como desejado, que:

$$\exists t'.\ \texttt{pred}(t) \longrightarrow t'$$

Se $t = \texttt{succ}(nv)$: pela regra E-PredSucc com $t = \texttt{succ}(nv)$ como premissa temos que:

$$\texttt{pred}(t) \longrightarrow nv \tag{27}$$

Por (27) acima e pela regra $\exists I$ provamos, como desejado, que:

$$\exists t'.\ \texttt{pred}(t) \longrightarrow t'$$

subcaso $\exists t'.\ t \longrightarrow t'$:

Pela regra $\exists E$ usando t' como *elemento representativo*, temos que

$$t \longrightarrow t' \tag{28}$$

Com (28) como premissa para a regra E-Pred temos que:

$$\texttt{pred}(t) \longrightarrow \texttt{pred}(t') \tag{29}$$

E por (29) e pela regra $\exists I$ provamos como desejado que;

$$\exists t'.\ \texttt{pred}(t) \longrightarrow t'$$

caso $\texttt{iszero}(t)$

prova de $P(\texttt{iszero}(t))$ ou seja prova de:

$\forall T.\ \ \texttt{iszero}(t) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{iszero}(t)) \ \vee\ \exists t'.\ \texttt{iszero}(t) \longrightarrow t')$.

podendo usar a seguinte hipótese indutiva, caso necessário:

$P(t) \ \equiv\ \forall T.\ \ t : T \ \Rightarrow\ (\mathsf{valor}(t) \ \vee\ \exists t'.\ t \longrightarrow t')$

Após a aplicaçao da regra $\forall I$ na fórmula acima, temos que provar a seguinte fórmula:

$$\texttt{iszero}(t) : T \ \Rightarrow\ (\mathsf{valor}(\texttt{iszero}(t)) \ \vee\ \exists t'.\ \texttt{iszero}(t) \longrightarrow t') \tag{30}$$

Pela regra $\Rightarrow I$ aplicada a (30) acima assumimos:

$$\texttt{iszero}(t) : T \tag{31}$$

e temos que provar

$$\mathsf{valor}(\texttt{iszero}(t)) \ \vee\ \exists t'.\ \texttt{iszero}(t) \longrightarrow t'$$

Como $\mathsf{valor}(\texttt{iszero}(t))$ é evidentemente falso para provar essa disjunção acima temos que provar:

$$\exists t'.\ \texttt{iszero}(t) \longrightarrow t' \tag{32}$$

De (31) acima e pela regra de tipo T-IsZero temos que:

$$t : \mathsf{nat} \tag{33}$$

Pela hipótese indutiva para t e pela regra $\forall E$ (instanciando T com nat) temos:

$$t : \mathsf{nat} \;\Rightarrow\; (\mathsf{valor}(t) \;\vee\; \exists t'.\, t \longrightarrow t') \tag{34}$$

Por *modus ponens* com (33) e (34) acima temos:

$$\mathsf{valor}(t) \;\vee\; \exists t'.\, t \longrightarrow t' \tag{35}$$

Por $\vee E$, para provar (32) a partir da disjunção (35) temos que provar (32) a partir de cada um dos disjuntos:

subcaso $\mathsf{valor}(t)$:

Por (33) e pelas regras do sistema de tipos temos que $t = 0$ ou $t = \mathtt{succ}(nv)$ com nv um valor numérico.

Se $t = 0$: pela regra E-IsZeroZero com $t = 0$ como premissa temos que:

$$\mathtt{iszero}(t) \longrightarrow \mathtt{true} \tag{36}$$

Por (36) acima e pela regra $\exists I$ provamos, como desejado, que:

$$\exists t'.\, \mathtt{iszero}(t) \longrightarrow t'$$

Se $t = \mathtt{succ}(nv)$: pela regra E-IsZeroSucc com $t = \mathtt{succ}(nv)$ como premissa temos que:

$$\mathtt{iszero}(t) \longrightarrow FALSE \tag{37}$$

Por (37) acima e pela regra $\exists I$ provamos, como desejado, que:

$$\exists t'.\, \mathtt{iszero}(t) \longrightarrow t'$$

subcaso $\exists t'.\, t \longrightarrow t'$:

Pela regra $\exists E$ usando t' como *elemento representativo*, temos que

$$t \longrightarrow t' \tag{38}$$

Com (38) como premissa para a regra E-IsZero temos que:

$$\mathtt{iszero}(t) \longrightarrow \mathtt{iszero}(t') \tag{39}$$

E por (39) e pela regra $\exists I$ provamos como desejado que;

$$\exists t'.\, \mathtt{iszero}(t) \longrightarrow t'$$

■