3
Provas Modularizadas

No capítulo 2, seção 2.1, foi apresentado um resumo do trabalho contido em [26], o qual possui a prova do teorema 2.4 que garante o processo de síntese construtiva de programas desde que algumas restrições sejam respeitadas.

Neste capítulo, é dado prosseguimento ao estudo sobre síntese construtiva de programas, o qual não possui a restrição sobre a regra de eliminação do existencial, que agora pode ter um conteúdo computacional qualquer. Antes, era restrito a programas escritos recursivamente.

É proposto um processo de síntese construtiva de programas que gera programas modulares a partir de provas modularizadas, bem como a garantia de sua correção.

Essa idéia surgiu a partir de [27] como um modo de reduzir o tamanho das árvores de prova, inserindo "*conteúdos computacionais pré-processados*"através de hipóteses com quantificadores existenciais em suas fórmulas, que estão associados a programas. As fórmulas são provadas separadamente e inseridas como hipóteses no sistema, permitindo cortes em alguns ramos da árvore de prova.

Essa estratégia, torna mais viável a utilização dos provadores de teoremas pois permite uma diminuição do custo computacional, bem como uma menor quantidade de memória para a realização da prova. Entretanto, esta estratégia não reduz a complexidade da prova final.

Neste novo processo de síntese de programas, além da eliminação de uma das restrições [1], é inserido o comando de atribuição, característico das linguagens imperativas, e também a utilização de memória principal,

[1] É mantida a restrição da utilização da regra de introdução da negação.

que poderá ser visto no comando de atribuição e na comunicação entre os módulos via parâmetros.

O processo de geração de programas é dividido em três partes: a *Marcação das configurações de memória*, a *Associação dos comandos com as regras de inferência* e a *Substituição dos rótulos por comandos na linguagem de programação*. Na seção seguinte, será apresentada a primeira parte deste processo, que descreve o processo computacional a partir das modificações de memória efetuadas a cada aplicação de uma regra de inferência.

Observação 3.0.1 *1. Alguns conceitos básicos, como por exemplo: definição de conteúdo computacional, conteúdo lógico, modelo de Herbrand, etc. estão definidos no capítulo 2, seção 2.1;*

2. Serão apresentados apenas os conceitos modificados em relação ao trabalho apresentado na seção 2.1, visto que estas colocações são apenas uma extensão das anteriores.

3.1 Marcação das configurações de memória

Este processo é similar ao que foi apresentado na seção 2.1.1, exceto pelo fato de que:

Quando um termo estiver associado à aplicação mais interna da regra de eliminação do existencial de um ramo da árvore de prova, será adicionado a ele o rótulo ""para assinalar que este possui uma chamada de procedimento associado.*[2]

3.1.1 Marcação das configurações de memória

Este processo é similar ao que foi apresentado na seção 2.1.1, diferenciando-se apenas pela substituição da regra de eliminação do quantificador existencial pela seguinte regra :

[2]Veja regra de eliminação do existencial.

Eliminação do Quantificador Existencial

Se esta regra for a sua aplicação mais interna, a regra de marcação das configurações de memória será:

$$\frac{\exists x \alpha(x)^V_T \qquad \begin{array}{c}\alpha(h)^V_{T\cup h^*}\\ \vdots \\ \gamma^{V_1}_{T_1}\end{array}}{\gamma^{V_1}_{T_1}},$$

senão, a regra será:

$$\frac{\exists x \alpha(x)^V_T \qquad \begin{array}{c}\alpha(h)^V_{T\cup h}\\ \vdots \\ \gamma^{V_1}_{T_1}\end{array}}{\gamma^{V_1}_{T_1}}$$

Antes de apresentar as associações das regras de inferência com comandos da linguagem de programação, será apresentada a semântica dos comandos. Esta informação também será utilizada na prova de correção do processo de síntese de programas.

3.2 Semântica dos comandos da linguagem de programação utilizada no processo de síntese de programas

A semântica dos comandos será expressa em semântica axiomática, onde as regras possuem o formato das triplas de Hoare [10]: $\{\phi\}P\{\psi\}$, onde ϕ e ψ são propriedades ou predicados (assertivas), e P é um segmento do programa. Considerando que, ϕ e ψ são pré e pós condições de P, respectivamente, a tripla de Hoare é válida se, e somente se, dado que a propriedade ϕ é válida, após a execução de P a propriedade ψ será válida.

As regras semânticas são apresentadas da seguinte forma:

Axiomas
$\{\phi\}P\{\psi\}$

Regras de Inferência:

1- $\dfrac{H_1, .., H_n}{H}$

Sempre que $H_1, \ldots, H_n$ forem triplas válidas, H será um tripla válida.

$$2\text{-}\frac{H_1, .., H_n \vdash H_{n+1}}{H}$$

Se H_{n+1} puder ser provada a partir de $H_1, \ldots, H_n$, então H é uma tripla válida.

Na descrição da semântica dos comandos são utilizadas as seguintes notações:

in – Memória de entrada[3];

out – Memória de saída[4];

$\widehat{z\ K}$ – z é o primeiro elemento a ser acessado na memória e K é o conjunto das outras informações em memória.

$\widehat{K\ z}$ – z é o último elemento a ser acessado na memória e K é o conjunto das outras informações em memória.

$\mathbf{P}_y^x$ – Toda ocorrência livre de x na fórmula P será sistematicamente substituída por y.

Segue a descrição da semântica dos comandos utilizados no processo de síntese de programas.

◇ *read* (variável)

Esse comando lê o primeiro valor de entrada na memória secundária (arquivo) atribuindo-o a uma variável(h).

Regra semântica:

$$\forall i \left\{ \left\{ \left(in = \widehat{i\ L} \right) \wedge \phi_i^h \wedge (i = i) \right\} read(h) \left\{ (in = L) \wedge \phi \wedge (h = i) \right\} \right\}$$

◇ *write* (variável)

O comando *write* escreve o valor contido na variável de saída (t) na memória secundária (arquivo) de saída.

Regra semântica:

$$\forall o \left\{ \left\{ \psi_o^t \wedge (out = W) \right\} write(t) \left\{ \psi_o^t \wedge \left(out = \widehat{W\ o} \right) \right\} \right\}$$

[3]Essa memória pode ser principal ou secundária.

[4]Essa memória pode ser principal ou secundária.

◇ *exec* (nome do procedimento, lista das variáveis de entrada)

Esse comando prepara o ambiente para a execução da função $env_{in}(\Lambda, \overrightarrow{v})$, que atribui os valores de entrada ($\overrightarrow{v}$), passados como argumentos, às variáveis de entrada do procedimento (*programa* Λ)), seguido da execução do procedimento e atribuição dos valores de saída às variáveis passadas por referência [45].

Regra semântica:

$$\frac{\dfrac{\left\{\phi\frac{\overrightarrow{v}}{\overrightarrow{i}}\right\} env_{in}(\Lambda, \overrightarrow{v})\,\{\phi\} \qquad \{\phi\}\,\Lambda(\overrightarrow{v})\left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}}\right\}}{\left\{\phi\frac{\overrightarrow{v}}{\overrightarrow{i}}\right\}\Lambda'(\overrightarrow{v})\left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}}\right\}} \qquad \left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}}\right\}\Lambda'(\overrightarrow{v})\,\{\psi\}}{\left\{\phi\frac{\overrightarrow{v}}{\overrightarrow{i}}\right\} exec(\Lambda, \overrightarrow{v})\,\{\psi\}}$$

◇ ←

Esse rótulo[5] marca a adição de uma posição de memória, relacionada aos valores de saída, a uma chamada de procedimento. Essa ação também pode ser descrita pela adição de uma variável passada por referência ao cabeçalho do procedimento.

Regra semântica:

$\left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}} \wedge (t_k = o_k) \wedge out = W\right\} Z \leftarrow \Lambda(\overrightarrow{v}, \overrightarrow{u}) \left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}} \wedge (Z = o_k) \wedge out = \widehat{W\ Z}\right\}$, onde $o_k \in \overrightarrow{o}$ e $k = 1..n$.

◇ ∗

Esse rótulo marca uma variável passada por referência a uma chamada de procedimento. Sua semântica é a idêntica ao do rótulo ... ← ...

◇ →

Esse rótulo assinala a atribuição de um valor armazenado na memória de entrada a uma variável de entrada de um procedimento.

Regra semântica:

$\left\{in = \widehat{h\ L} \wedge \phi_h^v \wedge (v = v)\right\} h \rightarrow \{in = L \wedge \phi \wedge (v = h)\}$

[5]Rótulos são símbolos que serão substituídos por comandos da linguagem de programação que possuem a mesma semântica.

◇ "(comando1) ; (comando2)"

O comando ";" expressa a composição de comandos descrevendo sua sequência de execução.

Regra semântica:

$$\frac{\{\phi\}\,\Lambda\,\{\theta\} \qquad \{\theta\}\,\Gamma\,\{\psi\}}{\{\phi\}\,\Lambda;\Gamma\,\{\psi\}}$$

◇ if (condição1) then {seqüência de comandos1} else {if (condição2) then {seqüência de comandos2}}

Esse comando, conhecido como condicional, expressa uma escolha entre o conjunto de comandos a serem executados, sendo esta escolha controlada pela satisfação ou não das condições.

Regra semântica:

$$\frac{\{\phi \wedge \alpha\}\,\Lambda\,\{\varphi\} \quad \{\phi \wedge \sim \alpha \wedge \rho\}\,\Psi\,\{\varphi\}}{\{\phi\}\,if\ (\alpha)\ then\ \{\Lambda\}\ else\ \{if\ (\rho)\ then\ \{\Psi\}\}\,\{\varphi\}}$$

◇ Atribuição

Este comando atribui o valor associado à variável do lado direito do comando ao espaço de memória associado à variável do lado esquerdo.

Regra semântica:

$$\{\phi^s_e\}\ s := e\ \{\phi\}.$$

◇ Chamada de Procedimento

A semântica deste comando equivale a semântica da execução do conjunto de comandos associado a ele.

Regra semântica:

$$\frac{\{\phi\}\,\Lambda\,\{\varphi\} \quad body(N) = \Lambda,\ params(N) = \langle I_1, ..., I_k, L_1, ..., L_m\rangle}{\left\{\phi^{I_1,...,I_k}_{E_1,...,E_k}\right\}\ call\ N(E_1, ..., E_k, O_1, ..., O_m)\ \{\varphi^{L_1,...,L_m}_{O_1,...,O_m}\}}$$

◇ Chamada Recursiva

Este comando reflete uma seqüência de chamadas realizadas por um programa a si mesmo, dentro do seu próprio bloco de comandos; de tal forma que o efeito da chamada do procedimento é o mesmo de sua execução.

Regra semântica:

$$A = \forall(I_1, ..., I_k)\forall(L_1, ..., L_m)\,\{\phi\}\ \ call\ N(I_1, ..., I_k, L_1, ..., L_m)\ \{\varphi\}$$

$$B = \{\phi\}\,\Lambda\,\{\varphi\}\,, body(N) = \Lambda,\ params(N) = \langle I_1, ..., I_k, L_1, ..., L_m\rangle$$

$$\frac{A \vdash B}{\left\{\phi^{I_1,...,I_k}_{E_1,...,E_k}\right\}\ call\ N(E_1, ..., E_k, O_1, ..., O_m)\ \{\varphi^{L_1,...,L_m}_{O_1,...,O_m}\}}$$

O processo de associação das regras de inferência com comandos, neste novo processo de síntese de programas, é similar ao apresentado na seção 2.1, sendo necessária apenas a substituição das regras de inferência apresentadas na próxima seção.

3.3 Associação das regras de inferência com comandos da linguagem imperativa

◇ *Fórmulas iniciais*

Axiomas

Os axiomas descrevem a natureza dos objetos utilizados pelo programa para resolver um problema, possuindo assim, somente conteúdos lógicos.

A regra de associação de comandos é:

$\sigma : \beta^V_T$.

Hipóteses

Se a hipótese for indutiva, o programa ainda está sendo construído; caso contrário, existe um programa relacionado com a hipótese que será fornecido pelo usuário. Em ambos os casos, a execução do programa gera a configuração de memória associada à fórmula.
A regra de associação de comandos é:

$p : \delta_T^V$, onde p é o símbolo para programas.

◇ *Introdução do Quantificador Universal*

Pode-se observar que a variável "h"está relacionada ao parâmetro de entrada do programa relacionado à premissa. Este fato é representado pelo rótulo $\rightarrow$:

$$\frac{\Lambda : \alpha(h)_T^{V \cup h}}{h \rightarrow \Lambda : \forall y \alpha(y)_T^V}$$

◇ *Eliminação do Quantificador Existencial*

Como esta regra é aplicada sobre o conjunto de hipóteses, a premissa maior possui um programa associado e uma de suas saídas é o valor referenciado pela aplicação desta regra. Logo, deve-se rotular a variável que receberá o valor após a execução do programa (será uma variável passada por referência para o procedimento).

Existem dois tipos de regras associadas, cuja escolha depende de sua posição na prova, que é assinalada pelo rótulo do termo relacionado com a premissa menor da mesma.

Observação 3.3.1 *Nas regras abaixo, Ψ é uma metavariável, que pode expressar um programa que está sendo construído (derivado a partir de uma hipótese indutiva) ou um programa (derivado a partir de uma dada hipótese) que será fornecido pelo usuário do sistema.*
O comando "exec"assinala uma chamada de procedimento e o símbolo "$\leftarrow$"indica que a variável do seu lado esquerdo será passada por referência para o procedimento que está ao seu lado direto.

Caso 1- Se o termo associado à regra de inferência apresentar o rótulo "*", então esta é a aplicação mais interna da regra de eliminação do quantificador existencial. Logo, além da marcação do termo de saída associado, é inserido o comando relacionado à chamada de procedimentos "*exec*".
A regra de associação de comandos é:

$$\dfrac{\begin{array}{cc} & h \leftarrow exec(\Psi, \overrightarrow{v}) : \alpha(h)^V_{T \cup h^*} \\ \Psi : \exists x \alpha(x)^V_T & \vdots \\ & \Lambda : \gamma^{V_1}_{T_1} \end{array}}{\Lambda : \gamma^{V_1}_{T_1}}$$

Caso 2- Caso contrário, basta inserir o rótulo do termo de saída associado. A regra de associação de comandos é:

$$\dfrac{\begin{array}{cc} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m & h \leftarrow \Psi : \alpha(h)^V_{T \cup h} \\ \vdots & \vdots \\ \Psi : \exists x \alpha(x)^V_T & \Lambda : \gamma^{V_1}_{T_1} \end{array}}{\Lambda : \gamma^{V_1}_{T_1}}$$

onde um dos comandos representados por Ψ é o comando "$\ldots \leftarrow exec(\ldots)$".

Observação 3.3.2 *Se a regra de eliminação do quantificador existencial for aplicada em uma fórmula que representa uma hipótese indutiva, a solução será um programa completo; caso contrário, a solução será um programa parametrizado.*

◇ *Introdução do Quantificador Existencial*

A execução do programa associado à premissa desta regra irá gerar um valor que será a saída do programa associado à conclusão.
A regra de associação de comandos é:

$$\dfrac{\Lambda : \alpha(h)^V_{T \cup h}}{y \leftarrow (\Lambda; y := h) : \exists x \alpha(x)^V_T}$$

◇ *Eliminação da Implicação*

Se a premissa menor apresentar somente conteúdo lógico, o programa gerado será o mesmo da premissa maior, e a regra de associação de comandos será:

$$\frac{\Xi : \alpha_T^V \qquad \Lambda : (\alpha \to \rho)_T^V}{\Lambda : \rho_T^V}$$

Por outro lado, se houver um programa associado com a premissa menor, deve-se substituir a suposta chamada de procedimento (Dec p) pela chamada de procedimento real (Ξ) no programa (Λ). A substituição é descrita na regra abaixo pela sentença: $\Lambda \blacktriangleleft \{exec([p], v) = Y\}$.
Logo, a regra de associação de comandos é:

$$\frac{\Xi : \alpha_T^V \qquad \text{Dec p; } \Lambda : (\alpha \to \rho)_T^V}{\Lambda \blacktriangleleft \{exec([p], v) = \Xi\} : \rho_T^V}$$

◇ *Indução*

Este conceito é análogo ao encontrado em programas recursivos; desta forma, o programa gerado será um procedimento recursivo contendo um comando condicional. Como esta regra é também uma regra de introdução do quantificador universal, é necessário marcar as variáveis do programa que são passadas por valor, utilizando o rótulo $\leftarrow$.
A regra de associação de comandos é:

$$\frac{\begin{array}{ccccc} [z \prec l]_T^V & & p : \alpha(a_i)_{T_1}^{V_1} & & \\ \vdots & & \vdots & & \vdots \\ \Psi : \alpha(z)_T^{V \cup z} & & \Lambda : \alpha(k)_T^{V \cup k} & & a_i \prec k_{T_2}^{V_2} \end{array}}{\begin{array}{l} VarIn \to VarOut \leftarrow \text{Procedure Rec}(VarIn, y, refVarOut)\{ \\ \quad \text{if } (y < l) \text{ then } \Psi \\ \quad \text{else}\{ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad : \forall y \alpha(y)_T^V \\ \Lambda \leftarrowtail (r \leftarrow \text{exec}([\text{p}], \overrightarrow{\text{v}}) = \text{Rec}(VarIn, variante(y), VarOut))^{**}\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \} \end{array}}$$

Observação 3.3.3 *1. - Embora a regra de inferência seja uma forma alternativa da regra de introdução do quantificador universal, o comando "read "estará no corpo do programa principal que executa a chamada do procedimento e os valores de entrada serão passados como parâmetros ($VarIn$) ;*

*2. - O símbolo "**"significa a substituição do programa hipotético (exec([p],$\overrightarrow{v}$)) pela chamada recursiva (Rec) no programa (Λ).*

Pode-se observar que, para simplificar o processo de geração de programas, algumas regras de inferência inserem o comando"*exec*", e os rótulo "←"e "→ ", que serão subtituídos por um comando ou por um conjunto de comandos, como será apresentado na próxima seção.

3.3.1
Substituição dos rótulos por comandos

É demonstrado, nesta seção, como é realizada a substituição do comando *exec* e dos rótulos "←"e "→ ", e como será criado o corpo do programa principal.

Deve-se observar que o comando "*exec* "e os rótulos "←"e "→"estão relacionados com a chamada de procedimento, variáveis de entrada e variáveis de saída, respectivamente.

Caso 1 :

Se o comando *exec* estiver relacionado com uma hipótese indutiva, a regra da indução substituirá este comando e os rótulos "←"e "→"por uma chamada recursiva com seus respectivos parâmetros. Desta forma, é necessário somente criar um novo arquivo com o programa que é composto pelo corpo do procedimento, seguido pelo corpo do programa principal que é formado por: cabeçalho do programa, comandos *read* aplicados sobre as variáveis com o rótulo "→", chamada do procedimento e os comandos *write* aplicados sobre as variáveis com o rótulo "←".

Exemplo:

Programa Gerado	Programa transformado

x→y→z←w←Procedure X(x,y,*z,*w)
{ if (x<l)
{ x= x+1;
.....
Procedure X(x,y,*z,*w)
....
}
else
{ .... }
}
	Procedure X(x,y,*z,*w)
	{ if (x<l)
	{ x= x+1;
	.....
	Procedure X(x,y,*z,*w);
	....
	} else
	{ .... }
	{ .... }
	}
	Main Program
	{ read(x); read(y);
	X(x,y,*z,*w);
	write(z); write(w);
	}

Caso 2:

Se o comando *exec* não estiver relacionado com a hipótese indutiva, o usuário terá que substituir este comando e os rótulos "←"e "→"pela respectiva chamada de procedimento.

Nesta substituição não se deve esquecer de fazer a passagem de parâmetros corretamente, dada a lista de variáveis e seus rótulos que estão antes do comando *exec*. As variáveis com rótulo "←"serão parâmetros passados por referência e as variáveis com o rótulo "→"serão parâmetros passados por valor. Além disso, deve-se criar o cabeçalho para o procedimento da mesma forma que foi realizado no caso dos procedimentos recursivos (caso 1).

Após as substituições, um novo arquivo será criado com o programa que é composto pelo corpo do procedimento, seguido pelo corpo do programa principal que é formado por: cabeçalho do programa, comandos *read* aplicados sobre as variáveis com o rótulo "→", chamada de procedimento e os comandos *write* aplicados sobre as variáveis com o rótulo "← ".

Exemplo: Suponha Λ o corpo do procedimento, tem-se então o seguinte programa: x→y→z←w← Λ. Ele será substituído por:

```
Procedure X(x,y ,*z,*w) { Λ }
Main Program {
  read(x);
  read(y);
  X( x,y , *z,*w);
  write(z);
  write(w);
}
```

Após a associação das regras de inferência com comandos da linguagem de programação, pode ser formulada a seguinte pergunta: Será que os comandos refletem realmente o conteúdo computacional das regras?

A resposta para essa pergunta é dada pela prova de correção. Entretanto, para realizá-la, é necessário conhecer o conteúdo computacional de cada símbolo lógico.

3.4 Contéudo semi-computational

No processo de síntese de programas dado uma prova, é extraído o conteúdo computacional de suas fórmulas, mas para facilitar a prova de corretude serão utilizados os conteúdos semi-computacionais (CSC) das fórmulas.

Nas definições sobre CSC, tem-se que:
θ - é um conjunto de fórmulas;
σ - expressa a configuração da memória, ou seja, atribuições de valores para as variáveis, em que certas propriedades são verdadeiras;
Γ- é o conjunto de axiomas que descreve a teoria, onde a solução do problema (conclusão da prova) é representada;
$\widehat{aC}$ - expressa a concatenação do elemento a com o objeto C; e
$\widehat{Ca}$- expressa a concatenação do objeto C com o elemento a.

O $CSC_M^{\theta}\left(\alpha_T^V\right)$ é descrito por triplas com a seguinte estrutura: $\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$, onde L é a memória de entrada, $\overrightarrow{i}$ é a lista das variáveis de entrada, $\overrightarrow{o}$ é a lista das variáveis de saída e W a memória de saída.

É considerado que M é o modelo de $Herbrand$ (H), ou seja uma estrutura sobre o universo de $Herbrand$ para a linguagem $L(\Gamma)$[6], tal que: $M \underset{H}{\models} \Gamma_T^V$

Abaixo segue a definição de $\mathrm{CSC}_M^{\theta}\left(\alpha_T^V\right)$, de acordo com a estrutura α:

(i) Fórmula atômica:

$CSC_M^{\theta}\left(A_T^V\right) =$

$$\left\{ \left\langle nil, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, nil \right\rangle \diagup \begin{array}{l} ((\forall v_k \in V)(\forall i_k \in \overrightarrow{i})\forall\sigma(\sigma(v_k) = i_k)), \\ ((\forall t_k \in T)(\forall o_k \in \overrightarrow{o})\,[[t_k]]_\sigma = o_k) \text{ e} \\ \text{Se } M \underset{H,\sigma}{\models} \theta \text{ então } M \underset{H,\sigma}{\models} A_T^V \end{array} \right\}$$

O conteúdo semi-computacional das fórmulas atômicas expressa que todos os valores que em algum instante estiveram armazenados em memória são os mesmos dos termos e das variáveis da fórmula, e que esses tornam a fórmula válida no modelo da teoria correspondente.

(ii) Quantificador Universal $(\forall)$

$CSC_M^{\theta}\left(\forall x \alpha(x)_T^V\right) =$

$$\left\{ \forall i_k \left(\left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle \diagup \left\langle L, \left\langle \widehat{i_k \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in CSC_M^{\theta}\left(\alpha(h)_T^{V \cup h}\right) \right) \right\}$$

A definição acima mostra que os valores das variáveis de entrada estão nas posições de memória relacionadas a elas.

[6]O modelo de Herbrand para as fórmulas rotuladas é o mesmo utilizado para as fórmulas não rotuladas.

(iii) Quantificador Existencial ($\exists$)

$$CSC_M^{\theta}\left(\exists x\alpha(x)_T^V\right) =$$

$$\left\{\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, \widehat{W o_k} \right\rangle / \left\langle L, \left\langle \overrightarrow{i}, \widehat{\overrightarrow{o} o_k} \right\rangle, W \right\rangle \in CSC_M^{\theta}\left(\alpha(h)_{T\cup h}^V\right)\right\}$$

Na definição do $CSC_M^{\theta}(\exists x\alpha(x)_T^V)$, os valores dos termos de saída estão nas posições de memória relacionadas a eles.

(iv) Conjunção($\wedge$)

$$CSC_M^{\theta}\left((\alpha \wedge \beta)_T^V\right) =$$

$$\left\{\left\langle \widehat{L_1 L_2}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, \widehat{W_1 W_2} \right\rangle / \begin{array}{c} k_\alpha = \left\langle L_1, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W_1 \right\rangle \in CSC_M^{\theta}\left(\alpha_T^V\right) e \\ k_\beta = \left\langle L_2, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W_2 \right\rangle \in CSC_M^{\theta}\left(\beta_T^V\right) \end{array}\right\}$$

Nesta definição, o $CSC_M^{\theta}\left((\alpha \wedge \beta)_T^V\right)$ é formado pelo $CSC_M^{\theta}(\alpha_T^V)$ e pelo $CSC_M^{\theta}(\beta_T^V)$.

(v) Disjunção ($\vee$)

$$CSC_M^{\theta}\left((\alpha \vee \beta)_T^V\right) = CSC_M^{\theta}\left(\alpha_T^V\right) \cup CSC_M^{\theta}\left(\beta_T^V\right)$$

Na definição acima, o $CSC_M^{\theta}\left((\alpha \vee \beta)_T^V\right)$ pode ser formado pelo $CSC_M^{\theta}(\alpha_T^V)$ ou pelo $CSC_M^{\theta}(\beta_T^V)$.

(vi) Implicação ($\rightarrow$)

$$CSC_M^{\theta}\left((\alpha \rightarrow \beta)_T^V\right) = CSC_M^{\theta\cup\alpha}\left(\beta_T^V\right)$$

Nesta definição,o $CSC_M^{\theta}\left((\alpha \rightarrow \beta)_T^V\right)$ é o mesmo $CSC_M^{\theta\cup\alpha}\left(\beta_T^V\right)$ que possue internamente uma referência $CSC_M^{\theta}\left(\alpha_T^V\right)$

(vii) Absurdo Intuicionista ($\perp$)

$$CSC_M^{\theta}\left(\perp_T^V\right) = \{\}$$

O $CSC_M^{\theta}\left(\perp_T^V\right)$ define a não existência de conteúdo semi-computacional.

As seguintes definições são utilizadas na prova de correção do processo de síntese de programas.

Definição 3.1 *$U \subseteq CSC_M^{\theta}\left(\alpha_T^V\right)$, U é completo se, e somente se, $\forall k_1 \in U$, tal que $k_1 = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$ e $\forall k_2 \in U$, tal que $k_2 = \left\langle L', \left\langle \overrightarrow{i}, \overrightarrow{o}' \right\rangle, W' \right\rangle$, logo $L = L'$, $\overrightarrow{o} = \overrightarrow{o}'$ e $W = W'$*

Observação 3.4.1 *O conceito de U completo equivale a dizer que o conteúdo computacional é restrito ao domínio em que U é funcional.*

Definição 3.2 *Seja $k = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$, onde $\overrightarrow{i}, \overrightarrow{o} \in M$ (Modelo de Herbrand), então Λ é um programa que calcula k se, e somente se,*
$\underset{Hoare}{\vdash} \{in = L \wedge (v_1 = i_1, ..., v_n = i_n)\}\, \Lambda \{(t_1 = o_1, ..., t_m = o_m) \wedge out = W\}$
onde $\underset{Hoare}{\vdash}$ significa derivabilidade no cálculo de Hoare, $V = \{v_1, ..., v_n\}$ e $T = \{t_1, ..., t_m\}$.

Definição 3.3 *$\theta \underset{M,\sigma}{\models} \Lambda : \left(\alpha_T^V\right)$ se, e somente se, $\exists U$ completo $\subseteq CSC_M^{\theta}\left(\alpha_T^V\right)$, tal que,$\forall u \in U$, o programa Λ calcula u.*

Dado o conhecimento do processo de síntese e de todas as definições apresentadas, têm-se as ferramentas necessárias para a prova de correção do processo de síntese construtivo de programas.

3.4.1
Prova de correção do processo de síntese

O processo de síntese é composto por três partes: a marcação das configurações de memória (L), a extração do conteúdo computacional da prova (G) e a substituição dos rótulos (R).

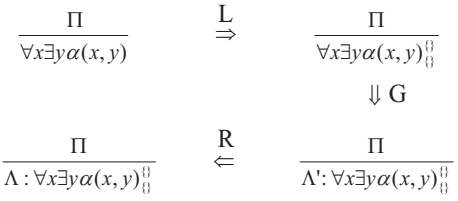

Figura 3.1: Esquema do processo de Síntese

Assim, obtém-se um programa que possui a mesma semântica da prova. Esse fato é garantido pela prova de corretude do sistema, obtido a partir da prova do teorema que utiliza o seguinte lema:

Lema 3.4.2 *Seja M um modelo de Herbrand para um conjunto de axiomas* Δ, Π *uma prova da fórmula* α: $\begin{matrix}\Delta\\ \Pi\\ \alpha\end{matrix}$ *e* Π*'=R(G(L(*Π*))).*

Sendo $\Pi_1 \prec \Pi$[7]*, então:*

a - As fórmulas, que apresentam conteúdo lógico, derivadas a partir das hipóteses, que também possuem conteúdo lógico, têm o mesmo modelo. Isto é:

Se $\begin{matrix}\{\beta_1, ..., \beta_n\}\\ \Pi_1\\ \sigma : \gamma\end{matrix}$ *e* $\beta_1, ..., \beta_n \notin \Delta$ *então* $\beta_1, ..., \beta_n \underset{M,\sigma}{\models} \sigma : \gamma_T^V$

b - Os conteúdos computacionais das fórmulas derivadas a partir dos axiomas e hipóteses, todos válidos no mesmo modelo, refletem a semântica dos programas associados a elas. Em outras palavras:

Se $\begin{matrix}\{\beta_1, ..., \beta_n, p_1 : \delta_1, , ..., p_k : \delta_k\}\\ \Pi_1\\ \Lambda : \alpha\end{matrix}$ *e* $\Delta \underset{M,\sigma}{\models} \forall_j(p_j : (\delta_j)_T^V)$ *então:*

$\theta \underset{M,\sigma}{\models} \Lambda : \alpha_T^V$, *onde* $\theta = \{\beta_1, ..., \beta_n\}$, *i.e.,* $\exists U$ *completo* $\subseteq CSC_M^\theta\left(\alpha_T^V\right)$ *tal que,* $\forall\ k \in U$ *sendo* $k = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$, *temos que:*

$\underset{Hoare}{\vdash} \{in = L \wedge (v_1 = i_1, ..., v_n = i_n)\}\, \Lambda \{(t_1 = o_1, ..., t_m = o_m) \wedge out = W\}$, *onde* $V = \{v_1, ..., v_n\}$, $T = \{t_1, ..., t_m\}$ *e* p_j *é o nome do programa associado a* δ_j.

[7] $\Pi_1 \prec \Pi$ expressa que Π_1 é uma derivação contida em Π.

Prova. A prova do lema[8] é realizada por indução no tamanho da prova, através da comparação da semântica das modificações sintáticas do programa com os conteúdos semi-computacionais das regras de inferência.

Observação 3.4.3 Δ *representa um conjunto de axiomas e* σ *símboliza a configuração da memória, ou seja, atribuições de valores para as variáveis, em que as propriedades expressas pela fórmula associada são verdadeiras;*

Caso Base:

$\diamond$ *Fórmulas Iniciais*

1-Axiomas

De acordo com regra de extração do conteúdo semi-computacional relacionado com os axiomas, eles expressam conteúdo lógico. Pela hipótese do lema, M é um modelo de $Herbrand$ para eles. Seja $\gamma \in \Delta$

$\models_M \sigma : \gamma$

2- Hipótese (não axiomas)

a) Hipótese com conteúdo lógico

De acordo com a hipótese (a) do lema, este possui apenas conteúdo lógico, logo: $\beta_i \models_{M,\sigma} \sigma : (\beta_i)^V_T$

b) Hipótese com conteúdo semi-computational

Pelas hipóteses do lema, essas hipóteses são corretas por construção, logo: $\Delta \models_{M,\sigma} p_i : (\delta_i)^V_T$

[8]Nesta seção, são apresentadas apenas as regras de inferência que tiveram os comandos associados modificados quando comparados à prova de correção citada em na seção 2.1. Entretanto, a prova para as regras não modificadas estão no apêndice na seção A.1.

Caso Indutivo :

Nota: Para facilitar a leitura da prova será usada a notação: δ_ι, ao invés de $p_i : \delta_\iota$.

$\diamond$ *Introdução do Quantificador Universal*

Suponha que esta regra seja a última aplicada na derivação D:

$$D = \left\{ \begin{array}{c} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \\ \vdots \\ \Lambda : \alpha(h)_T^{V \cup h} \\ \hline h \rightarrow \Lambda : \forall y \alpha(y)_T^V \end{array} \right.$$

Pode-se observar que a regra de construção de programas associado a essa regra gera um programa formado pela adição do rótulo "$h \rightarrow$"ao programa associado à premissa da regra (Λ), que reflete a atribuição de um valor em memória a uma variável de entrada (h).
Pela hipótese indutiva:

$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : \alpha(h)_T^{V \cup h}$ se, e somente se, $\exists U$ completo $\subseteq CSC_M^\theta \left(\alpha(h)_T^{V \cup h}\right)$ tal que, $\forall Q \in U$ sendo $Q = \left\langle L, \left\langle \widehat{h \ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (h = i_k) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$, onde $i_k \in \overrightarrow{i}$ e $k = 1..n$.

O resultado da composição do rótulo $h \rightarrow$ com o programa gera a seguinte regra semântica:
(1) $\dfrac{\left\{in = \widehat{i_k \ L} \wedge \phi_h^{i_k} \wedge (i_k = i_k)\right\} h \rightarrow \{in = L \wedge \phi \wedge (h = i_k)\} \quad \left\{in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (i_k = h)\right\} \Lambda \{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}}{\left\{in = \widehat{h \ L} \wedge \phi_h^{i_k} \wedge (i_k = i_k)\right\} h \rightarrow \Lambda \{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}}$

Dado que:
$CSC_M^\theta \left(\forall x \alpha(x)_T^V\right) =$

$$\left\{ \forall i_k \left(\left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle \diagup \left\langle L, \left\langle \widehat{i_k \ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in CSC_M^\theta \left(\alpha(h)_T^{V \cup h}\right) \right) \right\}$$

Se $U\prime \subseteq CSC_M^{\theta}\left(\forall x \alpha(x)_T^V\right)$, então:

$$U\prime = \left\{ \forall i_k \left(\left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle \diagup \left\langle L, \left\langle \widehat{i_k \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in U \right) \right\}$$

Como $U\prime$ é formado a partir de U, que pela hipótese indutiva é completo, tem-se que $U\prime$ é completo.

Seja $Q_1 \in U\prime$; $Q_1 = \left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$ tal que:

$$Q_2 = \left\langle L, \left\langle \widehat{i_k \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in CSC_M^{\theta}\left(\alpha(h)_T^{V \cup h}\right)$$

Pela hipótese indutiva, o programa Λ calcula Q_2.

Assim, a partir de (1), tem-se que $(h \rightarrow \Lambda)$ calcula Q_1.
Seja Q_1 uma tupla arbitrária que pertença a $U\prime$, conclui-se que:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} h \rightarrow \Lambda : \forall x \alpha(x)_T^V$$

◇ *Eliminação do Quantificador Universal*

Suponha que esta regra seja a última aplicação na derivaçao D:

$$D = \left\{ \begin{array}{c} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \\ \vdots \\ \Xi : \forall x \alpha(x)_T^V \\ \hline \Xi : \alpha(h)_T^{V \cup h} \end{array} \right.$$

Onde Ξ é uma metavariável que pode expressar conteúdos lógicos ou um programa.

Caso 1 : Ξ expressa conteúdo lógico.

Neste caso, como a conclusão tem apenas conteúdo lógico, pela hipótese do lema(a), ele é correto por construção.

Caso 2: Ξ expressa conteúdos computacionais.

Neste caso, o programa relacionado com a conclusão será o mesmo da premissa, pois essa regra de inferência está relacionada com a alocação de uma variável (h) em memória e sabendo que o programa relacionado com a premissa já aloca este espaço de memória, não é necessária nenhuma alteração no programa relacionado à premissa para refletir o CSC da conclusão.

Pela hipótese indutiva:

$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Xi : \forall x \alpha(x)_T^V$ se, e somente se,
$\exists U$ completo $\subseteq CSC_M^\theta \left(\forall x \alpha(x)_T^V\right)$ tal que, $\forall Q \in U$ sendo
$Q = \left\langle L, \left\langle \overrightarrow{i}\ , \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$ *onde* $o_k \in \overrightarrow{o}$ e $k = 1..n$.

Dado que:
$CSC_M^{\theta}\left(\forall x \alpha(x)_T^V\right) =$

$$\left\{\forall i_k \left(\left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle \diagup \left\langle L, \left\langle \widehat{i_k \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in CSC_M^{\theta}\left(\alpha(h)_T^{V \cup h}\right)\right)\right\}$$

Pela definição acima, pode-se observar que para um termo estar presente no arquivo dos valores de entrada, este deverá pertencer à lista dos valores de entrada que estão em memória. Logo, pode-se concluir que a alocação de memória que seria inserida ao programa está presente no programa relacionado à premissa (Ξ).

Assim, utilizando o mesmo U completo que pertence à hipótese indutiva, pode-se concluir:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Xi : \left(\alpha(h)_T^{V \cup h}\right)$$

◇ Introdução do Quantificador Existencial

Suponha que esta regra seja a última a ser aplicada na derivação D:

$$D = \left\{ \frac{\Lambda : \alpha(h)^V_{T \cup h}}{x \leftarrow (\Lambda; x := h) : \exists x \alpha(x)^V_T} \right.$$

Pode-se observar que a regra de construção de programas, associada a esta regra, libera um valor resultante da execução do programa relacionado com a premissa. E este fato é refletido pela construção de um programa formado pelo programa associado à premissa da regra (Λ) mais a atribuição do valor do termo de saída à posição de memória relacionada com a variável de saída ($x := h$), bem como a adição do rótulo $x \leftarrow$ ao programa gerado, simbolizando que este será um valor passado por referência ao procedimento gerado.

Pela hipótese indutiva:

$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : \alpha(h)^V_{T \cup h}$ se, e somente se:

$\exists U$ completo $\subseteq CSC^{\theta}_M \left(\alpha(h)^V_{T \cup h}\right)$ tal que, $\forall Q \in U$ sendo $Q = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o_i} \right\rangle, W \right\rangle$, tem-se que:

$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = \widehat{W\ h} \}$ onde $o_k \in \overrightarrow{o}$, $t_k \in \overrightarrow{t}$ e $k = 1..n$.

Dada a semântica dos comandos:

1- $x := h : \{\phi^x_h\}\, x := h\, \{\phi\}$, e

2- $... \leftarrow ... : \left\{ \psi^{\overrightarrow{t}}_{\overrightarrow{o}} \wedge (t_k = o_k) \right\} Z \leftarrow \Lambda(\overrightarrow{v}, \overrightarrow{u}) \left\{ \psi^{\overrightarrow{t}}_{\overrightarrow{o}} \wedge (Z = o_k) \right\}$, onde $o_k \in \overrightarrow{o}$, $t_k \in \overrightarrow{t}$ e $k = 1..n$.

E sabendo que:

B=$\{(\overrightarrow{t} = \overrightarrow{o}) \wedge (t_k = x) \wedge (h = o_k) \wedge \Upsilon^x_h \wedge out = W\} x := h \left\{ (\overrightarrow{t} = \overrightarrow{o}) \wedge (t_k = x) \wedge (h = o_k) \wedge \Upsilon \wedge out = W \right\}$,

A=$\left\{ in = L \wedge \phi^{\overrightarrow{v}}_{\overrightarrow{i}} \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge (t_k = x) \wedge (h = o_k) \wedge \Upsilon^x_h \wedge out = W \}$,

o resultado da concatenação dos comandos $(\Lambda; x := h)$ é:

$$(\mathrm{C})\frac{\overset{\mathrm{A}}{} \quad \overset{\mathrm{B}}{}}{\left\{in{=}L \wedge \phi\frac{\overrightarrow{v}}{\overrightarrow{i}}\right\}\Lambda; x:=h\left\{(\overrightarrow{t}=\overrightarrow{o}) \wedge (t_k{=}x) \wedge (h{=}o_k) \wedge \Upsilon \wedge out{=}W\right\}}$$

Como a semântica do comando ... ← ... é:
$(\mathrm{D})\left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}} \wedge (t_k = o_k) \wedge out = W\right\} Z \leftarrow \Lambda(\overrightarrow{v}, \overrightarrow{u}) \left\{\psi\frac{\overrightarrow{t}}{\overrightarrow{o}} \wedge (Z = o_k) \wedge out = \widehat{W\ Z}\right\}$,
onde $o_k \in \overrightarrow{o}$ e $k = 1..n$.

A semântica dos comandos $x \leftarrow (\Lambda; x := h)$ é expressa por:
$$\frac{\overset{\mathrm{C}}{} \quad \overset{\mathrm{D}}{}}{\left\{in{=}L \wedge \phi\frac{\overrightarrow{v}}{\overrightarrow{i}}\right\}x \leftarrow (\Lambda; x:=h)\left\{(\overrightarrow{t}=\overrightarrow{o}) \wedge (x{=}o_k) \wedge \Upsilon \wedge out{=}\widehat{W\ x}\right\}} \ (\mathrm{E})$$

Dado que:

$$CSC_M^{\theta}\left(\exists x \alpha(x)_T^V\right) =$$

$$\left\{\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, \widehat{W o_k}\right\rangle / \left\langle L, \left\langle \overrightarrow{i}, \widehat{\overrightarrow{o} o_k}\right\rangle, W\right\rangle \in CSC_M^{\theta}\left(\alpha(h)_{T\cup h}^V\right)\right\}$$

Se $U\prime \subseteq CSC_M^{\theta}\left(\exists x \alpha(x)_T^V\right)$, então:

$$U\prime = \left\{\forall i_k \left(\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, \widehat{W\ o_k}\right\rangle / \left\langle L, \left\langle \overrightarrow{i}, \widehat{\overrightarrow{o}\ o_k}\right\rangle, W\right\rangle \in U\right)\right\}$$

Como $U\prime$ é formado a partir de U, que pela hipótese indutiva é completo, tem-se que $U\prime$ é completo.

Seja $Q_1 \in U\prime$; $Q_1 = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, \widehat{W\ o_k}\right\rangle$ tal que:
$Q_2 = \left\langle L, \left\langle \overrightarrow{i}, \widehat{\overrightarrow{o}\ o_k}\right\rangle, W\right\rangle \in CSC_M^{\theta}\left(\alpha(h)_{T\cup h}^V\right)$

Pela hipótese indutiva, o programa Λ calcula Q_2.

Assim, a partir da semântica descrita em (E), tem-se que o programa $(x \leftarrow (\Lambda; x := h))$ calcula Q_1.

Sendo Q_1 uma tupla arbitrária que pertence a $U\prime$, conclui-se que:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} x \leftarrow (\Lambda; x := h) : \exists x \alpha(x)_T^V$$

◇ Eliminação do Quantificador Existencial

Suponha que esta regra seja a última a ser aplicada na derivação D:

Caso 1: Se esta for primeira aplicação desta regra.

$$D = \left\{ \begin{array}{c} \begin{array}{cc} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m & Z \leftarrow exec(\Psi, \overrightarrow{v}) : \alpha(Z)^V_{T \cup Z^*} \\ \vdots & \vdots \\ \Psi : \exists x \alpha(x)^V_T & \Lambda : \gamma^{V_1}_{T_1} \end{array} \\ \hline \Lambda : \gamma^{V_1}_{T_1} \end{array} \right.$$

Pode-se observar que o comando associado à aplicação desta regra é o comando *exec*, que prepara o ambiente para a execução do procedimento, atribuindo os valores de entrada ($\overrightarrow{v}$), passados como parâmetros, às variáveis de entrada do procedimento (*programa* Ψ), seguido da execução do procedimento e da atribuição dos valores de saída as variáveis passadas por referência. Além do comando *exec*, é adicionado o rótulo "$Z \leftarrow$"que marca qual é a variável a ser passada por referência.

Pela hipótese indutiva:

1 - $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Psi : \exists x \alpha(x)^V_T$ se, e somente se:
$\exists U$ completo $\subseteq CSC^{\theta}_M \left(\exists x \alpha(x)^V_T\right)$ tal que, $\forall Q \in U_1$ sendo $Q = \left\langle L, \left\langle \overrightarrow{i}\ , \overrightarrow{o} \right\rangle, \widehat{W\ o_k} \right\rangle$, tem-se que :
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Psi \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = \widehat{W\ o_k} \}$ onde $o_k \in \overrightarrow{o}$ e $k = 1..n$.

2- $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : \gamma^{V_1}_{T_1}$ se, e somente se:
$\exists U_2$ completo $\subseteq CSC^{\theta}_M \left(\gamma^{V_1}_{T_1}\right)$ tal que, $\forall Q \in U$ sendo $Q = \left\langle L, \left\langle \overrightarrow{i}\ , \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$

Na hipótese indutiva 2, a hipótese δ_i está associada ao comando $Z \leftarrow exec(\Delta, \overrightarrow{v})$

O comando $exec(\dots)$ possui a seguinte regra semântica:

$$\frac{\dfrac{\left\{\phi^{\overrightarrow{v}}_{\overrightarrow{i}}\right\} env_{in}(\Psi, \overrightarrow{v})\,\{\phi\} \qquad \{\phi\}\,\Psi(\overrightarrow{v})\,\{\psi^z_o\}}{\left\{\phi^{\overrightarrow{v}}_{\overrightarrow{i}}\right\}\Psi'(\overrightarrow{v})\,\{\psi^z_o\}} \qquad \{\psi^z_o\}\,\Psi'(\overrightarrow{v})\,\{\psi\}}{\left\{\phi^{\overrightarrow{v}}_{\overrightarrow{i}}\right\} exec(\Psi, \overrightarrow{v})\,\{\psi\}}$$

O comando ... ← ... possui regra semântica:

$$\left\{\psi^{\overrightarrow{t}}_{\overrightarrow{o}} \wedge (t_k = o_k) \wedge out = W\right\} Z \leftarrow \Psi(\overrightarrow{v}, \overrightarrow{u}) \left\{\psi^{\overrightarrow{t}}_{\overrightarrow{o}} \wedge (Z = o_k) \wedge out = \widehat{W\ Z}\right\},$$

onde $o_k \in \overrightarrow{o}$ e $k = 1..n$.

Observação 3.4.4 *A semântica de ... ← .... está embutida no comando exec(...), logo ... ← .... é utilizado apenas como um rótulo.*

Dado que:

$$CSC^{\theta}_{M}\left(\exists x \alpha(x)^{V}_{T}\right) =$$

$$\left\{\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, \widehat{W o_k}\right\rangle / \left\langle L, \left\langle \overrightarrow{i}, \widehat{\overrightarrow{o} o_k}\right\rangle, W\right\rangle \in CSC^{\theta}_{M}\left(\alpha(h)^{V}_{T\cup h}\right)\right\}$$

Pela hipótese indutiva (2): $\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, W\right\rangle \in U_2$, a partir da hipótese que $Z \leftarrow exec(....)$, como $Z = o_k$, pode-se escrever a hipótese $o_k \leftarrow exec(....)$, tal que $o_k \in \overrightarrow{o}$.

Pela hipótese indutiva (1) o programa Ψ calcula $CSC^{\theta}_{M}\left(\exists x \alpha(x)^{V}_{T}\right)$, que é descrito pela tupla: $Q = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, \widehat{W o_k}\right\rangle$.

Pode ser observado com base na semântica do comando $Z \leftarrow exec(....)$, que atribui o termo o_k para Z, é realizado pela execução do programa Ψ. Então, $CSC^{\theta}_{M}(\gamma^{V_1}_{T_1}) = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o}\right\rangle, W\right\rangle$, onde $i_k = o^{\Psi}_k$ e $o^{\Psi}_k \in U_1$.

Sendo k uma tupla arbitrária que pertence a U_1, pode-se concluir que: $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : \gamma^{V_1}_{T_1}$

Caso 2: Se esta não for a primeira aplicação desta regra.

$$D = \left\{ \begin{array}{c} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \quad Z \leftarrow \Upsilon : \alpha(Z)^V_{T \cup Z} \\ \vdots \qquad\qquad\qquad\quad \vdots \\ \Upsilon : \exists x \alpha(x)^V_T \qquad\qquad \Lambda : \gamma^{V_1}_{T_1} \\ \hline \Lambda : \gamma^{V_1}_{T_1} \end{array} \right.$$

A regra de construção de programas relacionado a essa regra de inferência apenas adiciona o rótulo "$Z \leftarrow$"que assinala qual é a variável a ser passada por referência. Como esta não é a primeira aplicação da regra, tem-se que o programa Υ contém o comando *exec* que prepara o ambiente e executa o procedimento que irá gerar o termo de saída a ser associado à variável Z.

Pela hipótese indutiva:

1 - $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Upsilon : \exists x \alpha(x)^V_T$ se, e somente se:
$\exists U$ completo $\subseteq CSC^{\theta}_M \left(\exists x \alpha(x)^V_T\right)$ tal que, $\forall Q \in U_1$ sendo
$Q = \left\langle L, \left\langle \overrightarrow{i} \ , \overrightarrow{o} \right\rangle, \widehat{W \ o_k} \right\rangle$, tem-se que:
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Upsilon \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = \widehat{W \ o_k} \}$ onde $o_k \in \overrightarrow{o}$ e $k = 1..n$.

2- $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : \delta^{V_1}_{T_1}$ se, e somente se:
$\exists U_2$ completo $\subseteq CSC^{\theta}_M \left(\delta^{V_1}_{T_1}\right)$ tal que, $\forall Q \in U$ sendo
$Q = \left\langle L, \left\langle \overrightarrow{i} \ , \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$

Na hipótese indutiva (2) a hipótese δ_i é associada com o comando $Z \leftarrow \Upsilon$.

O comando ... $\leftarrow$ ... possui a seguinte regra semântica:
$\left\{ \psi^{\overrightarrow{t}}_{\overrightarrow{o}} \wedge (t_k = o_k) \wedge out = W \right\} Z \leftarrow \Psi(\overrightarrow{v}, \overrightarrow{u}) \left\{ \psi^{\overrightarrow{t}}_{\overrightarrow{o}} \wedge (Z = o_k) \wedge out = \widehat{W \ Z} \right\}$, onde $o_k \in \overrightarrow{o}$ e $k = 1..n$.

Dado que:

$$CSC_M^{\theta}\left(\exists x \alpha(x)_T^V\right) =$$

$$\left\{\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, \widehat{W o_k} \right\rangle / \left\langle L, \left\langle \overrightarrow{i}, \widehat{\overrightarrow{o} o_k} \right\rangle, W \right\rangle \in CSC_M^{\theta}\left(\alpha(h)_{T \cup h}^V\right)\right\}$$

Pela hipótese indutiva (2): $\left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle \in U_2$ a partir da hipótese que $Z \leftarrow \Upsilon$, como $Z = o_k$ pode-se escrever a hipótese como $o_k \leftarrow \Upsilon$, tal que: $o_k \in \overrightarrow{o}$.

Pela hipótese indutiva (1) o programa Υ calcula $CSC_M^{\theta}\left(\exists x \alpha(x)_T^V\right)$, que é descrito pela tupla: $Q = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, \widehat{W o_k} \right\rangle$.

Com base na semântica do comando $Z \leftarrow \Upsilon$ pode-se observar que ele atribui à Z o termo de saída correspondente gerado pela execução do programa Υ. Dessa forma, tem-se que: $CSC_M^{\theta}(\gamma_{T_1}^{V_1}) = \left\langle L, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$, onde $i_k = o_n^{\Upsilon}$ se $o_n^{\Upsilon} \in U_1$

Seja k uma tupla arbitrária que pertence a U_1, pode-se concluir que:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : \gamma_{T_1}^{V_1}$$

◇ Introdução da Implicação

Suponha que esta regra seja a última a ser aplicada na derivação D:

$$D = \left\{ \begin{array}{c} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, \Xi : \alpha^{V_1}_{T_1} \\ \vdots \\ \Lambda : \rho^V_T \\ \hline \Lambda : (\alpha \to \rho)^V_T \end{array} \right. ,$$

onde Ξ pode representar um programa ou um contéudo lógico.

Independente do tipo da hipótese, o programa relacionado com a conclusão é o mesmo da premissa desta regra de inferência, visto que o $CSC(\alpha)$ já está embutido no programa relacionado à premissa.

Pela hipótese indutiva:
$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, \alpha \underset{M,\sigma}{\models} \Lambda : \rho^V_T$ se, e somente se:
$\exists U$ completo $\subseteq CSC^{\theta \cup \alpha}_M \left(\Lambda : \rho^V_T\right)$ tal que, $\forall Q \in U$ sendo $Q = \left\langle L, \left\langle \overrightarrow{i}\ , \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:
$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$.

Dado que:
$CSC^{\theta}_M \left((\alpha \to \rho)^V_T\right) = CSC^{\theta \cup \alpha}_M \left(\rho^V_T\right)$.

Por definição, $CSC^{\theta}_M \left((\alpha \to \rho)^V_T\right)$ possui o mesmo U da hipótese indutiva, logo:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, \underset{M,\sigma}{\models} \Lambda : (\alpha \to \rho)^V_T$$

◇ Eliminação da Implicação

Suponha que esta regra seja a última aplicação na derivação D:

$$D = \left\{ \begin{array}{c} \begin{array}{cc} \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m & \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \\ \vdots & \vdots \\ \Xi : \alpha_T^V & \Lambda : (\alpha \rightarrow \rho)_T^V \end{array} \\ \hline \Lambda : \rho_T^V \end{array} \right. ,$$

onde Ξ pode representar um programa ou um contéudo lógico.

Independente do conteúdo (lógico ou computacional) da premissa menor, o programa gerado é o mesmo programa da premissa maior desta regra, que por sua vez possui as informações sobre a premissa menor embutida em seu corpo. Se ele vier de uma hipótese que tem um programa associado, o comando *exec* presente nos comandos da premissa maior será substituído pela chamada do procedimento relacionado com a premissa menor.

Pela hipótese indutiva:

$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \Lambda : (\alpha \rightarrow \rho)_T^V$ se, e somente se:

$\exists U$ completo $\subseteq CSC_M^\theta \left(\Lambda : (\alpha \rightarrow \rho)_T^V\right)$ tal que, $\forall\ Q \in U$ sendo $Q = \left\langle L, \left\langle \overrightarrow{i}\ , \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:

$\underset{Hoare}{\vdash} \left\{ in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \right\} \Lambda \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$.

Dado que:

$$CSC_M^\theta \left((\alpha \rightarrow \rho)_T^V\right) = CSC_M^{\theta \cup \alpha} \left(\rho_T^V\right)$$

Por definição $CSC_M^\theta \left(\rho_T^V\right)$ possui o mesmo U da hipótese indutiva, logo: $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, \underset{M,\sigma}{\models} \Lambda : \rho_T^V$

◇ Regra de Indução

Suponha que a regra de inferência seja a última utilizada na derivação D:

$$D = \left\{ \begin{array}{l} \dfrac{\begin{array}{lll} & \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, p : \alpha(a)^{V \cup a}_{T_2} & \\ & \vdots & \\ \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, [z \prec l]^{V_1}_{T_1} & p : \exists x \beta(x)^{V_3}_{T_3} & \\ \vdots & s \leftarrow exec(p, \overrightarrow{v}) : \beta(s)^{V_3}_{T_3} & \vdots \\ \Psi : \alpha(z)^{V \cup z}_{T} & \vdots & a_i \prec r^{V_2}_{T_2} \\ & \Lambda : \alpha(r)^{V \cup r}_{T} & \end{array}}{\begin{array}{l} VarIn \rightarrow VarOut \leftarrow \text{Procedure Rec}(VarIn, y, refVarOut)\{ \\ \quad \text{if } (y < l) \text{ then } \Psi \\ \quad \text{else}\{ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad : \forall y \alpha(y)^{V}_{T} \\ \Lambda \leftarrow (r \leftarrow \text{exec}([\text{p}], \overrightarrow{\text{v}}) = \text{Rec}(VarIn, variante(y), VarOut))^{**}\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \} \end{array} \end{array} \right.$$

Onde : β é a subfórmula de $\alpha(a)$, z é o termo relacionado com o caso base, k o termo relacionado com o caso indutivo, y é a variável de entrada que será igual a z ou a k, "**"representa que no programa Λ a chamada do procedimento hipotético($exec$) será susbtituída por uma chamada recursiva.

O programa associado à aplicação desta regra de inferência, possuirá um programa recursivo que contém um comando condicional. Como esta regra é também uma regra de introdução do quantificador universal, é necessário marcar as variáveis do programa que são passadas por valor, utilizando o rótulo ←.

Pela hipótese indutiva:

1 - $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, z \prec l \underset{M,\sigma}{\models} \Psi : \alpha(z)^{V \cup z}_{T}$ se, e somente se:

$\exists U_1$ completo $\subseteq CSC^{\theta \cup (z \prec l)}_{M}\left(\alpha(z)^{V \cup z}_{T}\right)$ tal que, $(\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m) = \theta$ e $\forall Q \in U_1$ sendo $Q = \left\langle L, \left\langle \widehat{i \ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:

$\underset{Hoare}{\vdash} \left\{ in = L \wedge \phi \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (z = i_k) \right\} \Psi \{ (\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \}$,

onde $i_k \in \overrightarrow{i}$ e $k = 1...n$.

2 - $\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m, p : \alpha(a) \underset{M,\sigma}{\models} \Lambda : \alpha(r)_T^{V \cup r}$ se, e somente se:

$\exists U_2$ completo $\subseteq CSC_M^{\theta \cup \alpha(a)} \left(\alpha(r)_T^{V \cup r}\right)$ tal que, $(\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m) = \theta$ e $\forall Q \in U_2$ sendo $Q = \left\langle L, \left\langle \widehat{r \ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle$, tem-se que:

$\underset{Hoare}{\vdash} \left\{in = L \wedge \phi \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (r = i_k)\right\} \Lambda \{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}$,

onde $i_k \in \overrightarrow{i}$ e $k = 1...n$.

Pela derivação de D, pode-se observar que as conclusões $\alpha(z)$ e $\alpha(r)$ são provadas por hipóteses disjuntas($z \prec l$ e $\alpha(a)$, respectivamente), logo, será associado o comando condicional a essas conclusões. Este terá a seguinte regra semântica:

Sendo:

$A = \left\{in = L \wedge \phi \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (z = i_k) \wedge (z \prec l)\right\} \Psi \{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}$ e

$B = \left\{in = L \wedge \phi \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (r = i_k) \wedge \neg(z \prec l)\right\} \Lambda \{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}$

(a) $\dfrac{A \quad B}{\left\{in = L \wedge \phi \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (y = i_k)\right\} if \ (z \prec l) \ then \ \{\Psi\} \ else \ \{\Lambda\} \left\{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\right\}}$

Pela hipótese indutiva, a partir da prova de $a(z)$ com a hipótese $(z \prec l)$ pode-se extrair o $CSC_M^{\theta \cup (z \prec l)} \left(\alpha(z)_T^{V \cup z}\right)$, e a partir da prova de $\alpha(r)$ com $\alpha(a)$ pode-se extrair o $CSC_M^{\theta \cup \alpha(a)} \left(\alpha(r)_T^{V \cup r}\right)$.

Como $(z \prec l)$ e $\alpha(a)$ são hipóteses disjuntas, tem-se que:

$$CSC_M^{\theta \cup (z \prec l)} \left(\alpha(z)_T^{V \cup z}\right) \cup CSC_M^{\theta \cup \alpha(a)} \left(\alpha(r)_T^{V \cup r}\right) = CSC_M^{\theta} \left(\alpha(x)_T^{V \cup x}\right),$$

onde $x = r$ (quando $x \geqslant l$) ou $x = b$ (quando $x < l$).

Logo, se $U' \subseteq CSC_M^{\theta} \left(\alpha_T^V\right)$ então $U' = U_1 \cup U_2$.

Como $U\prime$ é formado a partir U_1 e U_2, que pela hipótese indutiva são completos, tem-se que $U\prime$ é completo, já que a lista de valores de entrada de U_1 (que é relacionado com os elementos menores que l) e a de U_2 (que é relacionado com os elementos maiores ou iguais a l) são disjuntas.

Assim:

$$(1)\ \Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} if\ (y \prec l)\ then\ \{\Psi\}\ else\ \{\Lambda\} : \alpha(y)_T^{V \cup y}$$

Dado que esta regra é uma variação da regra de introdução do quantificador universal, é necessário a composição do comando $\rightarrow$ com o comando condicional que terá seguinte regra semântica:

(b) Sendo
$A = \left\{in = \widehat{i_k\ L} \wedge \phi_{i_k}^h \wedge (i_k = i_k)\right\} y \rightarrow \{in = L \wedge \phi \wedge (y = i_k)\}$ e
$B = \left\{in = L \wedge \phi \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (y = i_k)\right\} if\ (y \prec l)\ then\ \{\Psi\}$ $else\ \{\Lambda\} \left\{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\right\}$

$$\frac{A \quad B}{\left\{in = \widehat{h\ L} \wedge \phi_h^{i_k} \wedge (i_k = i_k)\right\} h \rightarrow if\ (y \prec l)\ then\ \{\Psi\}\ else\ \{\Lambda\}\{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}}$$

Dado que:

$$CSC_M^{\theta}\left(\forall x \alpha(x)_T^V\right) =$$

$$\left\{\forall i_k \left(\left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle / \left\langle L, \left\langle \widehat{i_k\ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in CSC_M^{\theta}\left(\alpha(h)_T^{V \cup h}\right)\right)\right\}$$

Se $U_{\forall} \subseteq CSC_M^{\theta}\left(\forall x \alpha(x)_T^V\right)$, então

$$U_{\forall} = \left\{\forall i_k \left(\left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle / \left\langle L, \left\langle \widehat{i_k\ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in U\right)\right\}$$

Como $U_{\forall}$ é formado a partir de U', tem-se que $U_{\forall}$ é completo.

Seja $Q_1 \in U_{\forall}$; $Q_1 = \left\langle \widehat{i_k L}, \left\langle \overrightarrow{i}, \overrightarrow{o} \right\rangle, W \right\rangle$ tal que:

$$Q_2 = \left\langle L, \left\langle \widehat{i_k\ \overrightarrow{i}}, \overrightarrow{o} \right\rangle, W \right\rangle \in \left(U' \subseteq CSC_M^{\theta}\left(\alpha(x)_T^{V \cup x}\right)\right)$$

Pela hipótese indutiva e pela regra semântica (a), o programa $if\ (y \prec l)\ then\ \{\Psi\}\ else\ \{\Lambda\}$ calcula Q_2.

Logo, a partir da regra semântica (b), tem-se que $(h \rightarrow if\ (y \prec l)\ then\ \{\Psi\}\ else\ \{\Lambda\})$ calcula Q_1. Sendo Q_1 uma tupla arbitrária de $U_{\forall}$, pode-se concluir:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} h \rightarrow if\ (y \prec l)\ then\ \{\Lambda\}\ else\ \{\Psi\} : \forall y \alpha(y)_T^V$$

Dado que o programa gerado é associado com a hipótese indutiva ($\alpha(a)$), que tem o quantificador existêncial em sua fórmula, o corpo do programa Λ possui o comando: "$\ldots exec(p, \ldots)$" e pode-se observar que o programa associado à hipótese indutiva é o ponto fixo do programa gerado, logo este é recursivo.

Assim, na construção deste programa, esse comando será substituído pela chamada de procedimento associado, que neste caso é o próprio programa construído, gerando, desta forma uma chamada recursiva.

Dado a semântica das chamadas recursivas:

$A = \left\{in = L \wedge (\overrightarrow{v} = \overrightarrow{i})\right\} h \leftarrow exec(F, \overrightarrow{v}, \overrightarrow{u}) \left\{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \wedge (Z = o_k)\right\}$
e
$B = \left\{in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (r = i_k)\right\} \Upsilon\{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W\}, body(F) = \Upsilon, params(F) = \langle \overrightarrow{v}, \overrightarrow{u} \rangle$:

$$\frac{A \underset{Hoare}{\vdash} B}{\left\{in{=}L{\wedge}(\overrightarrow{v}{=}\overrightarrow{i}){\wedge}(r{=}i_k)\right\}h{\leftarrow}F(\overrightarrow{v},\overrightarrow{u})\{(\overrightarrow{t}{=}\overrightarrow{o}){\wedge}out{=}W{\wedge}(Z{=}o_k)\}}.$$

No programa acima, será adicionado a marcação (Procedure ($\ldots$)) de forma que o programa $\left\{in = L \wedge (\overrightarrow{v} = \overrightarrow{i}) \wedge (r = i_k)\right\} h \leftarrow F(\overrightarrow{v}, \overrightarrow{u})\{(\overrightarrow{t} = \overrightarrow{o}) \wedge out = W \wedge (Z = o_k)\}$ será o corpo do procedimento.

Assim:

$$\Delta, \beta_1, ..., \beta_n, \delta_1, ..., \delta_m \underset{M,\sigma}{\models} \begin{array}{l} \text{Procedure Rec}(VarIn, y, refVarOut)\{ \\ \quad \text{if } (y < l) \text{ then } \Psi \\ \quad \text{else}\{ \\ \quad \Lambda \leftarrow (r \leftarrow \text{exec}([\text{p}], \overrightarrow{v}) = \text{Rec}(VarIn, variante(y), VarOut))^*\} \\ \quad \} \end{array} : \forall y \alpha(y)_T^V$$

Observação 3.4.5 *Após este passo na geração dos programas (seção 3.3), o comando "exec"e os rótulos "$\leftarrow$"e "$\rightarrow$"serão substituídos pela chamada de procedimento com suas respectivas variáveis de entrada e saída, o que não invalida a prova de correção para o processo de síntese para o programa gerado, visto que a substituição preserva semântica. O mesmo raciocínio é válido para adição do programa principal que prepara o ambiente para a execução dos programas (seção 3.3.1)*

□

Teorema 3.4 *Seja Δ um conjunto de axiomas(definindo os tipos de dados pretendidos), δ um conjunto de hipóteses que não são axiomas e Π a prova de $\forall x \exists y \alpha(x, y)$ na HA (Aritmética de Heyting). Se Λ é o programa gerado pela função $R(G(L(\Pi)))$[9] então: $\delta, \Delta \underset{M,\sigma}{\models} \Lambda : \forall x \exists y \alpha(x, y)^{\{\ \}}_{\{\ \}}$, isto é, se M satisfaz o modelo de δ, Δ com σ então satisfaz Λ.*

Prova. Aplicando o lema anterior obtém-se a prova do teorema. □

3.5 Exemplo

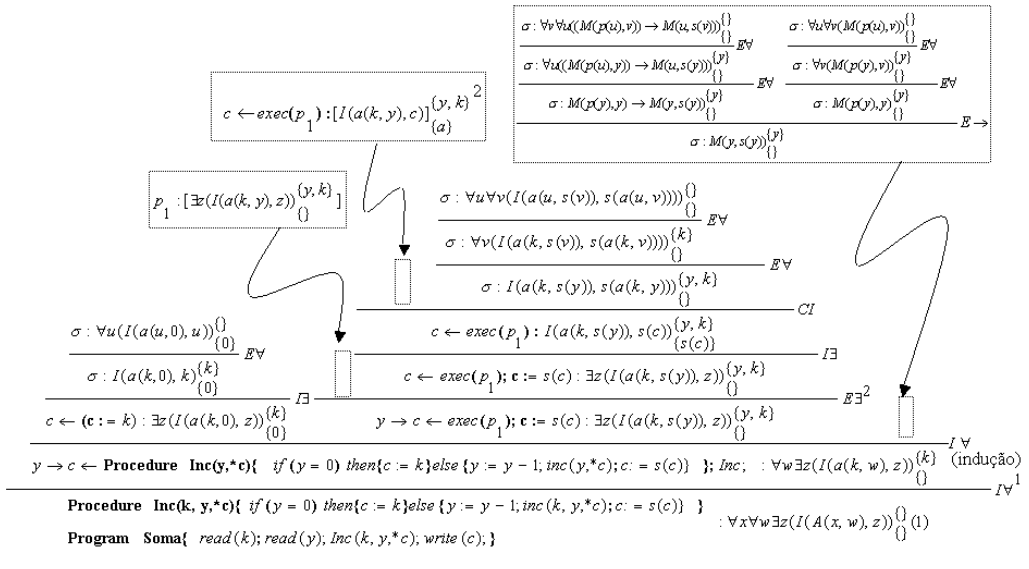

Figura 3.2: Extração do conteúdo computacional para a prova da adição de dois números naturais

Neste capítulo foi apresentado a possibilidade da geração de programas a partir de provas modularizadas, o que ocasionou a eliminação da restrição na utilização da regra de eliminação do quantificador existencial, mas tem-se ainda a restrição sobre a regra de introdução da negação, sobre a qual não se sabe se possui conteúdo computacional ou não. No capítulo seguinte, será apresentada uma argumentação que dará uma solução parcial a esse problema.

[9] A função L realiza a marcação das configurações de memória das provas das formulas, a função G faz a extração do conteúdo computacional da prova e a função R substitui os rótulos.