6
Conclusão e Trabalhos futuros

No estudo realizado, foi proposto um processo de geração de programas imperativos a partir de provas[1] em lógica intuicionista, que utiliza como sistema dedutivo a dedução natural. O trabalho aqui concluído torna mais robusto o sistema apresentado em [26] graças à possibilidade da regra de eliminação do quantificador existencial poder possuir qualquer conteúdo computacional. Além disso, apresenta a prova de que, para qualquer teorema, na aritmética de Heyting, existe uma prova Π normal[2] , que não possui a regra do absurdo com conteúdo computacional; e que, se essa prova[3] Π utilizar a regra ω computacional, esta pode ser transformada em uma prova Π' que, ao invés da regra ω computacional, utiliza a indução finita.

Neste processo de síntese, para remover a restrição que afirma que a regra de eliminação do quantificador existencial só pode ser aplicada sobre a hipótese indutiva, foi inserido no processo de geração de programas o conceito de modularização de programas, que é um aspecto essencial no desenvolvimento de programas, permitindo que diversos programadores desenvolvam módulos independentes que se comunicam via chamadas de procedimentos. Esse conceito é aplicado ao se assumir que toda hipótese que contém o quantificador existencial em sua fórmula possui um programa associado, resultante do processo de síntese aplicado sobre a fórmula dessa hipótese. Mas, neste caso, se o programa não for escrito recursivamente a chamada do procedimento será fornecida pelo usuário do sistema. Além de resolver o problema relacionado com a geração dos programas, indiretamente é amenizado um do pontos críticos do processo de síntese, que é a geração das provas, um trabalho que tende a ser muito complexo. A correção de todo esse processo é garantida através da prova formal.

Deve-se salientar que os outros sintetizadores da literatura utilizam ITT e cálculo de sequentes, e em alguns casos, lógica de reescrita. Estes necessitam

[1] Além das regras de inferência usuais, utiliza as regras de inferência da igualdade, da indução e da regra ω.

[2] Além das regras de inferência usuais, utiliza as regras de inferência da igualdade, da indução e da regra ω.

[3] No contexto da aritmética de Heyting com a propriedade de reflexividade

de um processo de tradução de suas provas para dedução natural, visto que o isomorfismo de Curry-Howard é estabelecido para este sistema dedutivo. O método proposto além de não necessitar dessa transformação gera programas legíveis e escritos em linguagem imperativa. Enquanto que os outros geram programas em linguagens lógicas ou funcionais, que não são tão legíveis. A legibilidade do programa gerado é garantida pelo fato de que todos os comandos que dizem respeito a uma determinada propriedade estão descritos em um mesmo bloco de comandos, visto que para compreender uma prova em dedução natural, basta analisar as sub-árvores que a compõem. Desta forma, o raciocínio é encadeado, exceto no caso das eliminações das hipóteses. Porém, de qualquer modo, todas as informações estão contidas na mesma sub-árvore. Pode-se tentar justificar que as provas em cálculo de seqüentes e reescrita podem ser organizadas de forma similar a uma prova em dedução natural, porém o custo de tal processo seria equivalente ao de construir um provador em dedução natural.

Pode-se considerar um ponto positivo, em relação a outros sintetizadores encontrados na literatura, o fato de que este sintetizador recebe como entrada especificações declarativas em lógica de predicados, que descreve apenas o problema e os tipos abstratos de dados utilizados, permitindo expressar os problemas de maneira mais simples do que os que trabalham com ITT (Nuprl [28], Oyster [17] e NJL [12]) e lógica equacional (Lemma [21]), nos quais a especificação é muito próxima da própria implementação do programa, sendo desta forma necessário ter conhecimento sobre como funciona a implementação do problema.

No processo de síntese construtiva proposto, baseado no isomorfismo Curry-Howard, a regra do $\perp$ possui somente conteúdo lógico. Entretanto, neste trabalho, em função da especificação dos tipos abstratos de dados, surgiu a dúvida sobre o conteúdo computacional desta regra. Essa questão motivou a apresentação da prova de que para todo teorema[4] da forma $\forall x_1 \ldots \forall x_n \exists y \alpha(x_1, \ldots, x_n, y)$ existe uma prova na qual a regra do $\perp$ sempre apresenta somente conteúdo lógico. Sabendo que toda computação pode ser descrita através de funções recursivas, e que o programa gerado sempre pára, é realizada uma verificação da utilização da regra do $\perp$ nas provas que representam as funções recursivas totais, onde além das regras de inferência usuais foi utilizada a regra ω computacional. Nessas provas foi observada a não necessidade do $\perp$ com conteúdo computacional, independente do fato dos axiomas que definem os tipos abstratos de dados terem ou não conteúdo computacional. Além da prova de correção, como efeito deste procedimento foi obtida a completude do método de síntese de programas para a aritmética de Heyting(HA).

[4]No contexto da arimética de Heyting.

O resultado obtido serviu como semente para se estudar as implementações de provadores de teoremas automáticos que possuam a regra ω computacional. Entretanto, a maior parte dos provadores de teoremas implementam apenas a indução finitária, surgindo, então, a seguinte pergunta:

Será possível que todas as provas, na aritmética de Heyting com a propriedade de reflexividade, realizadas em um sistema que possui regra ω computacional com as devidas restrições, expressando todas funções recursivas totais, podem ser realizadas em sistemas que possuem somente a indução finita, na qual a regra do $\perp$ preserva a leitura computacional apresentada pelo teorema 4.1?

Uma resposta parcial é dada no capítulo 4, onde é apresentada uma prova de que esse resultado é possível, desde as provas com a regra ω tenham sido realizadas no contexto da aritmética de Heyting com a propriedade da reflexividade.

Os resultados aqui obtidos apontam para vários caminhos a serem percorridos no futuro, sendo o objetivo final a implementação de um processo de síntese construtiva que torne viável a implantação efetiva no processo de desenvolvimento de sistemas embutidos. Mas, para isso, tem-se que estudar uma implementação para a regra ω computacional ou, então, se o processo de transformação de provas com a regra ω para indução finitária gera provas normais para o caso de teoremas que representam funções totais.

Seguem alguns tópicos a serem explorados:

✓ Investigar a profundidade da exigência da condição de reflexividade utilizada na prova do teorema 4.5 em [8];

✓ Estudar as condições para a extração do conteúdo computacional de provas não normais;

✓ Análise da complexidade das provas geradas com e sem o absurdo intuicionista;

✓ Utilização de outros cálculos no processo de síntese de programas, por exemplo *Deduction Modulo*;

✓ Investigar a associação do método de síntese de programas a partir de provas na lógica clássica, com o tratamento de exceções no processo de desenvolvimento de programas;

✓ Fazer um estudo sobre o fato de se poder expressar qualquer computação com um único *loop* , que funciona como um interpretador (Kleene), e o resultado de Gentzen que afirma que qualquer prova na aritmética pode ser expressa por uma única indução.