Resumo

Nos últimos tempos a utilização do Prolog tem-se generalizado, tanto nos meios académicos, como no meio comercial. A eficiência do Prolog é pois uma das características da linguagem que tem sido ao longo do tempo aperfeiçoada. O Yap[1] é neste momento um compilador de Prolog para YAAM[2] (uma variação de WAM [AK91] mais optimizada [Cos19]). Após ter gerado as instruções YAAM, o Yap procede a execução do código. Ao código YAAM chama-se também código da máquina abstracta. A arquitectura do Yap permite que sejam facilmente implementados predicados que alteram o próprio programa em tempo de execução. O Yap não consegue tirar partido de algumas optimizações actualmente existentes, que só fazem sentido quando se gera código nativo (tirando partido das potencialidades de uma máquina especifica). A existência de novos algoritmos e de novas arquitecturas de máquinas, vem novamente colocar o desafio da geração de código nativo. Para assegurar a portabilidade do Yap, este encontra-se escrito em linguagem C [Conb]. Por outro lado a tentativa de tornar o Prolog tão eficiente como as linguagens imperativas [Roy84] leva-nos à necessidade de criação de compiladores de Prolog para código nativo. Este novo compilador, de código nativo, deve tirar partido da arquitectura da máquina, usando as instruções Assembly que são mais adequadas [Cona]. Esta dissertação tem como base a alteração do Yap para que gere código nativo, em vez de executar as instruções da máquina abstracta. Apresento ainda um algoritmo de divisão de código [3] que obtêm um código mais eficiente em termos de velocidade de execução, na geração de código nativo. Apresento algumas optimizações que podem ser feitas no final da aplicação deste algoritmo, bem como alguns exemplos. Sugiro no final uma futura alteração que permite também uma maior eficiência no código nativo gerado, através da eliminação de uma variável[4] internada WAM.

Abstract

In the last decade, the Prolog has been used both in academic and industrial places. The efficiency of Prolog is one of the many language characteristics that has been throw the time refined. Yap [5] is, at this moment, a compiler from Prolog to YAAM[6] (a kind of WAM[AK91], but more optimized[Cos19]). After Yap has generated the YAAM instructions, it proceeds with the code execution. To the YAAM code we will also call the abstract machine code. The architecture of YAP permits the easy implementation of dynamic clauses, allowing the program to modify the code in runtime. Yap, has not being a native code compiler, can not take advantage of the machine architecture. The existence of new algorithms and new machines in the past years has been place again the challenge of native code generation for Prolog. To insure the portability of Yap, it was written in C [Conb] language. By the other side, the attempt to make Prolog as efficiently has the imperative languages[Roy84] takes us to the need of create native code compilers for Prolog. This new native code compiler should take advantage of the machine architecture, using the assembly instructions more appropriated [Cona].

This dissertation has on it base the change from Yap to generate native code for a specific machine, instead of just execute the abstract machine instructions. I also present an implementation of an algorithm to the YAAM instructions. This algorithm is called Two Streams[7]. Some simple optimizations we can implement, in the end of this algorithm, are also demonstrated. I present in the last chapter an optimization to the WAM, that can lead to a best code. This optimization eliminates the WAM variable S.