Programa Doutoral MAPi
Proposta de Tema para Doutoramento

Jorge Sousa Pinto
{jsp@di.uminho.pt}

Dezembro de 2010

1 Tema

Verificação Dedutiva de Programas Concorrentes

2 Unidade Proponente

Centro de Ciências e Tecnologias de Computação, Universidade do Minho

3 Enquadramento

Este trabalho é proposto no âmbito das actividades do projecto de investigação financiado pela FCT *FAVAS, A Formal Verification Platform for Realtime Systems*, que teve arranque em Março de 2010. O projecto tem ainda como participantes as Universidades da Beira Interior e da Madeira, bem como a empresa Critical Software.

O projecto visa o desenvolvimento de técnicas de verificação para os sistemas críticos embebidos, em particular para os sistemas de tempo real. O projecto proporcionará, além do apoio da respectiva equipa multidisciplinar de investigadores, também apoio financeiro para deslocações, bem como para a contratação de recursos humanos (bolseiros que colaborarão de forma próxima neste trabalho de doutoramento).

É inegável a importância crescente do desenvolvimento de técnicas de verificação formal de correcção e segurança (*safety*) para os sistemas de software, e em particular os sistemas *críticos*. Ao longo dos anos têm sido propostas diversas técnicas baseadas em diferentes formas de análise estática do código, que podem ser classificadas de acordo com o seu grau de automação. Em geral a prova automática de propriedades interessantes de um programa é um problema indecidível, e tipicamente a um maior grau de automação corresponde uma menor capacidade de lidar com propriedades complexas de modo preciso. Técnicas automáticas incluem nomeadamente *interpretação abstracta* e *verificação de modelos de* software ("software model checking"). Técnicas em que o envolvimento do utilizador é obrigatório, são as baseadas em sistemas de inferência e validade lógica, nomedamente a verificação dedutiva de programas. Não se tratando de uma técnica automática, por requerer a intevenção do utilizador, nomeadamente através da inclusão de *anotações* no código, existe ainda uma gradação no nível de automação possível no processo de estabelecimento da validade lógica, que poderá passar pela prova automática, no caso de propriedades relativamente simples, ou pela prova construída interactivamente, no caso de propriedades mais complexas.

As propriedades do código que se pretende estabelecer são de resto também elas muito diversas, podendo ir das propriedades funcionais de código sequencial até às propriedades sensíveis ao factor tempo, passando pelas propriedades de segurança relativa aos acessos a memória ou à execução de código concorrente (como a ausência de "deadlock").

O tema de tese de doutoramento aqui proposto centra-se na verificação formal de sistemas concorrentes, recorrendo a técnicas baseadas em sistemas de inferência. Estas técnicas, sendo as mais difíceis de utilizar e automatizar, são também as únicas que oferecem garantia total (i.e. com respeito a todas as execuções possíveis de um programa) de satisfação de propriedades. Reconhecidamente, o problema da verificação de aplicações concorrentes é em geral de resolução extremamente difícil, se não impossível. Se no caso das propriedades de código sequencial existem já diversas ferramentas de verificação capazes de produzir resultados muito satisfatórios e com um grau de automação bastante aceitável, já no caso das propriedades de código concorrente os próprios fundamentos teóricos (como os métodos "rely-guarantee", a lógica da separação, ou ainda a chamada análise de forma) encontram-se ainda em fase de desenvolvimento ou aperfeiçoamento.

No contexto dos sistemas críticos, a programação de sistemas concorrentes passa frequentemente pela definição de restrições muito fortes sobre a forma que a concorrência pode assumir, o que tende a simplificar um pouco o problema de verificação formal. No caso da linguagem Ada por exemplo, foi proposto um perfil restrito para o desenvolvimento de aplicações concorrentes, designado por *Ravenscar*. Apesar de um dos grandes objectivos do perfil ser precisamente possiblitar a verificação formal das aplicações, os trabalhos desenvolvidos até agora ficam claramente aquém das expectativas criadas na comunidades Ada.

Acrescente-se ainda que no caso do código Ada *sequencial*, popularizou-se a utilização de uma sub-linguagem do Ada, designada por SPARK. Mais do que um simples sub-conjunto sequencial do Ada, o SPARK é uma linguagem concebida com base nos princípios do *design-by-contract*, suportando verificação estática (funcional e de segurança /ausência de erros de execução) com base num VCGen e ferramenta de prova dedicadas. O *toolset* SPARK inclui também verificação estática de fluxo de dados e de informação.

4 Objectivos

O grande objectivo desta tese é, depois de efectuado um levantamento exaustivo de todo o trabalho existente e publicado sobre verificação dedutiva de programas de programas concorrentes, o desenvolvimento de uma linguagem nuclear para programação concorrente, que inclua restrições do tipo usualmente associadas ao desenvolvimento de sistemas críticos, e de toda a infraestrutura de verificação (em particular um gerador de condições de verificação) que possibilite a verificação de propriedades de programas desta linguagem.

Este grande objectivo pressupõe o desenvolvimento de contribuições fundamentais ao nível dos formalismos e técnicas de verificação de propriedades de código concorrente. As tarefas poderão passar pelo desenvolvimento inicial de uma plataforma de verificação sequencial, após o que se considerará extensões a esta plataforma no sentido do tratamento de estruturas dinâmicas (em "heap") e da execução concorrente de diversas tarefas do programa.