Verificação Formal de Kernel

Glauber Módolo Cabral

Universidade Estadual de Campinas - UNICAMP
Instituto de Computação - IC

MO806 - Tópicos em Sistemas Operacionais
Seminário

Outubro de 2007

Roteiro

Haskell

- Linguagem funcional de programação: Implementa os conceitos de λ-Cálculo (programação através de aplicações de funções);
- Fortemente tipificada: todo elemento possui tipo (pré-definido ou calculado automaticamente no contexto);
- Avaliação preguiçosa: um argumento de uma função só é avaliado quando é usado no cálculo;
- Pura: não permite alterar o estado do sistema (efeitos colaterais) a menos das partes envolvidas no cálculo de uma função.
- Mônadas são utilizadas para seqüenciar operações com efeitos colaterais, encapsulando as alterações;
- Literate Haskell: permite documentação em LaTeXjunto ao código.

Definições

Verificação formal

Técnica que visa a garantir, através de provas matemáticas, que um programa comporta-se da forma especificada.

Micro-kernel

Kernel projetado para ser mínimo em tamanho de código e em funções providas.

Problemas

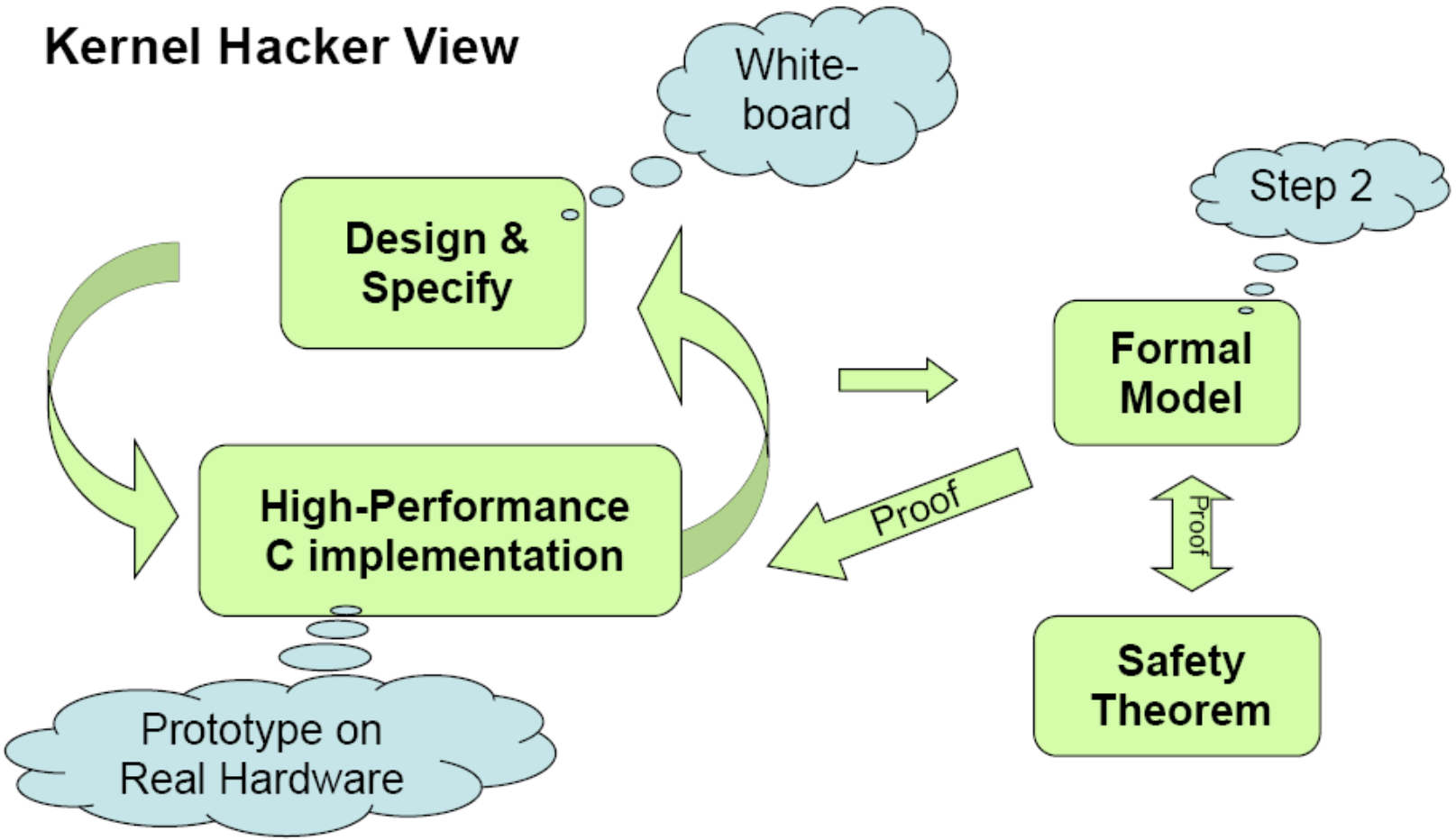
Kernel

- Desenvolvimento bottom-up;
- Código muito grande;
- Interfaces de alto-nível sofrem alterações para acomodar código de mais baixo-nível durante o desenvolvimento;

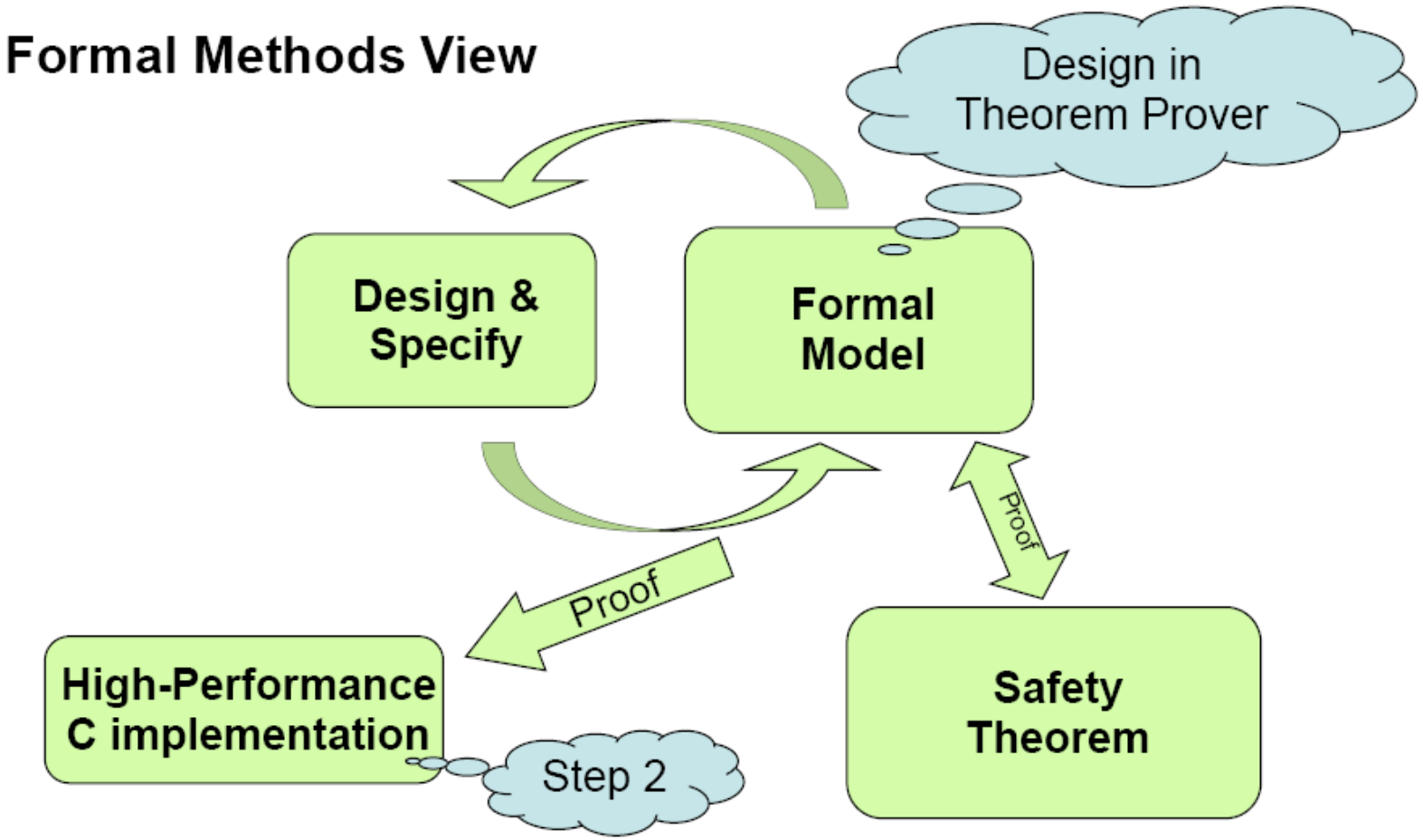
Verificação Formal

- Desenvolvimento top-down;
- Ausência de escalabilidade nas técnicas;
- Ausência de provadores eficientes para projetos grandes;
- Verificar programas com efeitos colaterais é complicado.

Abordagem dos programadores de kernel



Abordagem de métodos formais



Vantagens

- É a unica maneira de garantir a confiabilidade de um kernel pra todo evento possível;
- Garante a ausência de erros de software;
- Com as ferramentas atuais, o custo mantém-se na mesma ordem de grandeza em relação às técnicas tradicionais;

seL4 - Secure Embedded L4

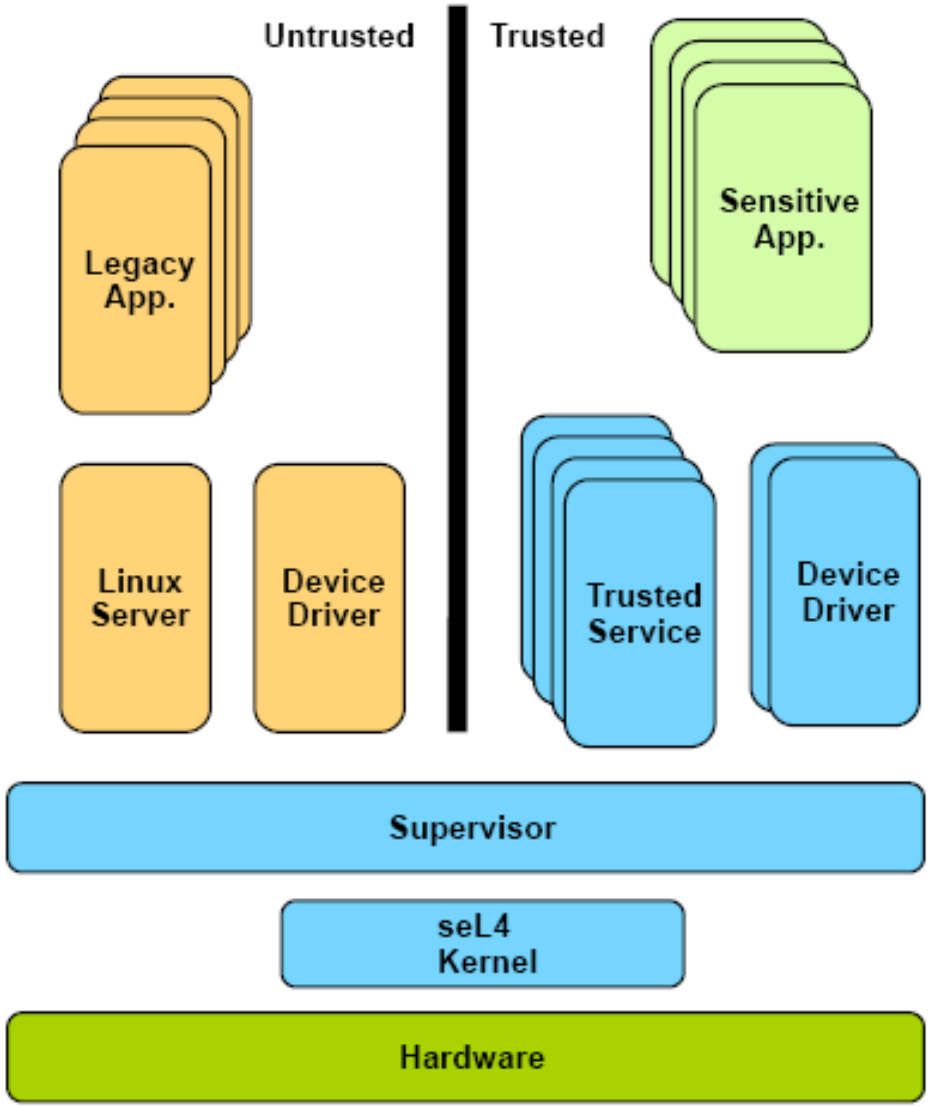
Definição

- Derivado do micro-kernel L4;
- Criado para provêr uma base segura para o desenvolvimento de sistemas embarcados;

Objetivos de segurança

- Controle da comunicação entre aplicações:
  - garantir isolamento entre sub-sistemas
  - garantir isolamento das informações de uma aplicação em relação a outras.
- Gerenciamento da memória física utilizada pelo kernel:
  - garantir disponibilidade de memória para serviços do kernel;
  - prevêr o tempo de execução das operações.

seL4 - Visão Geral



seL4 - Propriedades Desejadas na Metodologia

- API com especificação precisa, em oposição a descrições feitas em linguagem natural e em forma de manuais;
- Expôr detalhes de implementação para mostrar a possibilidade de uma implementação de alta performance;
- Permitir a construção de sistemas mais complexos usando a API resultante;
- Permitir formalização do sistema;
- Tornar possível o uso da metodologia por programadores de kernel, não adeptos a métodos formais.

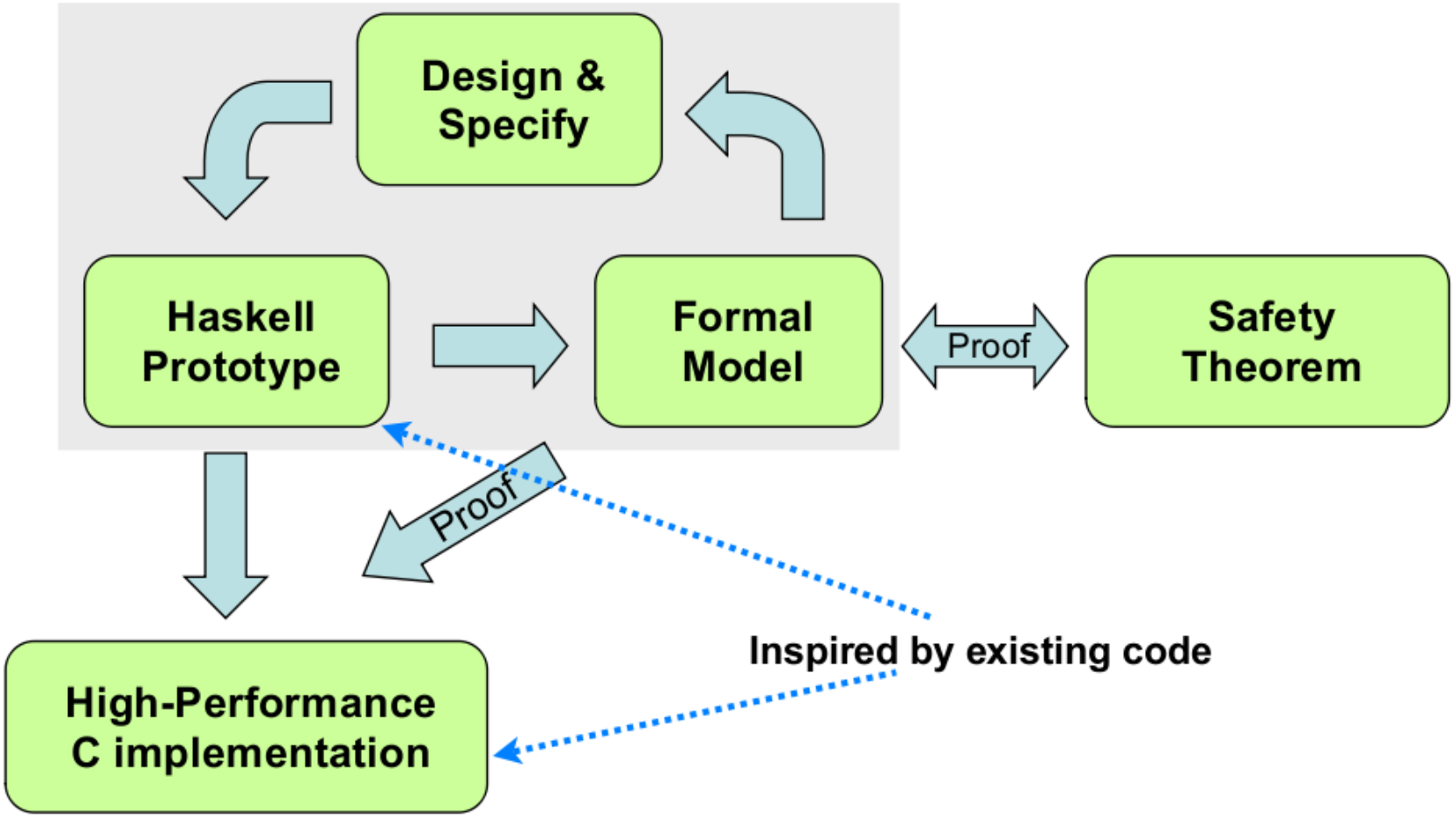
seL4 - Metodologia empregada (1)

- Literate Haskell: especificação e implementação da API;
- Protótipo escrito em Haskell;
- Validação através de um emulador ARM: binários de códigos em C são executados sobre o kernel;

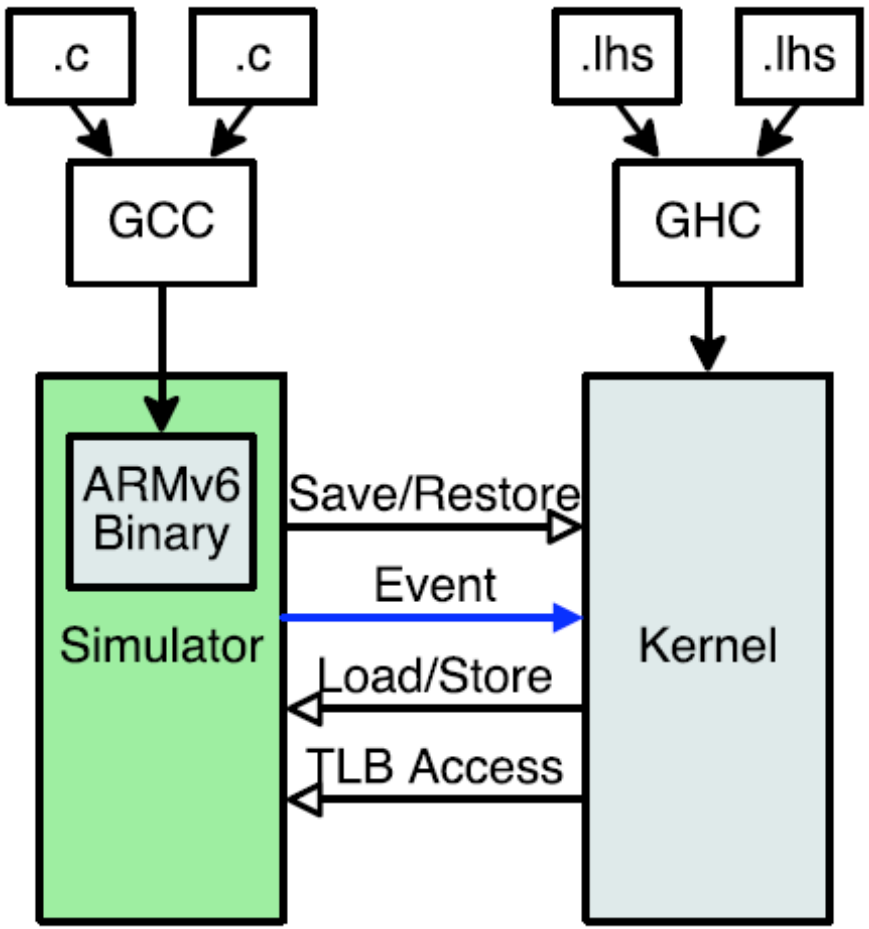
Justificativa

- Haskell não permite efeitos colaterais;
- Permite explorar detalhes de implementação de maneira fácil.

Metodologia empregada (2)



Simulador ARM



Problemas para a verificação

- Kernel grande (3000 loc);
- Extensões do compilador GHC não possuem modelos formais;
- Ferramentas existentes não fazem análise do código (Haskell + HOL);
- É possível não haver terminação (recursão infinita);
- Sistema de tipos de HOL é fraco para modelar mônadas de forma abstrata.

Soluções encontradas

Terminação

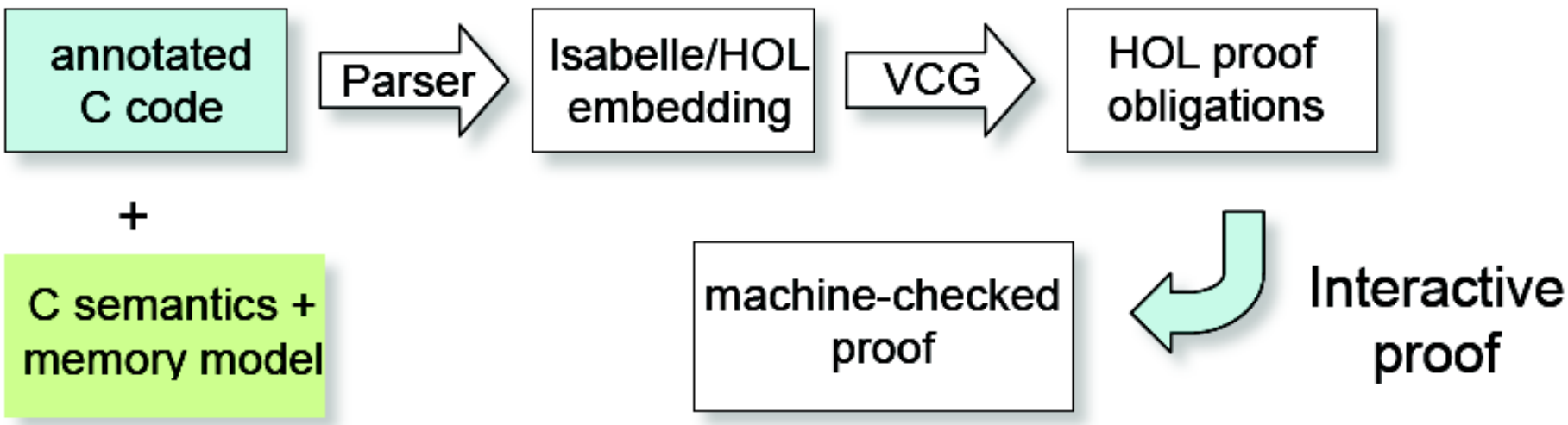
- Há pouca recursão;
- Todas as chamadas de sistemas terminam;
- Prova manual de terminação.

Mônadas

- Sistema de tipos de HOL permite modelar instâncias concretas de mônadas;
- Mônadas concretas bastam para modelar as propriedades desejadas;
- HOL possue uma sintaxe parecida com a de Haskell para seqüenciar ações.

Refinando para código C

- C é bem conhecido pelos programadores de Kernel;
- Permite otimizações posteriores;
- Permite incluir Assembly para alta performance.



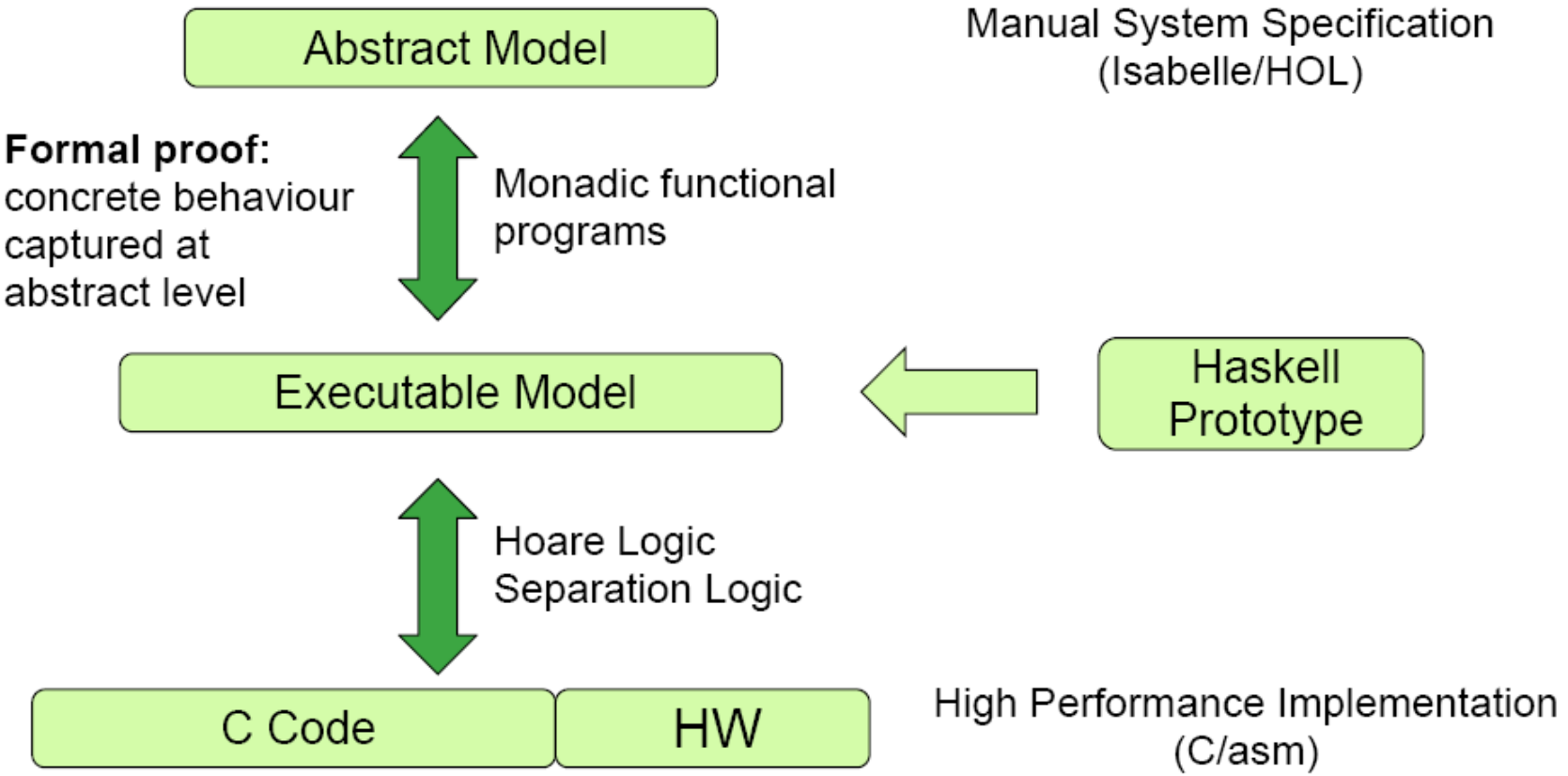
Problemas em C evitados

- Pode-se assumir como dados os detalhes de máquina: endian-ness, comportamento dos operadores aritméticos, tamanho dos bytes (8 bits ou não);
- Ausência de efeitos colaterais permite ignorar a ordem de avaliação de expressões;
- Comportamentos indefinidos são ilegais: divisão por zero, escrever em memória não alocada;
- Aritmética de ponteiros: adicionar guardas para garantir que o resultado seja um endereço válido para o tipo de dado.

Guardas

- Expressão booleana sobre os estados do programa;
- Simulam verificações complicadas em tempo de execução;
- Isabele/HOL exige que a guarda seja verdadeira sempre que o código posterior estiver prestes a ser executado;
- Flexibilidade na definição:
  - Em sistemas onde o endereço zero guarda o início de um vetor de exceções, escrita e leitura de ponteiros nulos devem ser permitidos;
  - Escrita de memória não-alocada pode ser parte do gerenciador de memória; a leitura ainda pode ser tratada como erro de tempo de execução.

Verificação formal



Referências I


Kevin Elphinstone, Gerwin Klein, Philip Derrin, Timothy Roscoe, and Gernot Heiser.
Towards a practical, verified kernel.
In *Proc. 11th Workshop on Hot Topics in Operating Systems*, page 6, San Diego, CA, USA, may 2007.
Online proceedings at `http://www.usenix.org/events/hotos07/tech/`.


Kevin Elphinstone, Gerwin Klein, and Rafal Kolanski.
Formalising a high-performance microkernel.
In Rustan Leino, editor, *Workshop on Verified Software: Theories, Tools, and Experiments (VSTTE 06)*, Microsoft Research Technical Report MSR-TR-2006-117, pages 1–7, Seattle, USA, August 2006.

Referências II

Gerwin Klein, Michael Norrish, Kevin Elphinstone, and Gernot Heiser.
Verifying a high-performance micro-kernel.
In *7th Annual High-Confidence Software and Systems Conference*, Baltimore, MD, USA, May 2007.

Harvey Tuch, Gerwin Klein, and Gernot Heiser.
Os verification — now!
In Margo Seltzer, editor, *Proc. 10th Workshop on Hot Topics in Operating Systems (HotOS X)*, 2005.
to appear.

Agradecimentos

Obrigado!

Perguntas?

Contato:
glauber.cabral@students.ic.unicamp.br