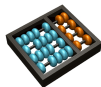


Criação de uma biblioteca padrão para HasCASL

Glauber Módolo Cabral – Orientando
Prof. Dr. Arnaldo Vieira Moura – Orientador

**Universidade Estadual de Campinas
Instituto de Computação**



26 de Abril de 2010

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Métodos Formais

- ▶ Ferramentas de Engenharia de Software que empregam formalismos matemáticos na construção de programas;
- ▶ Compostas por uma ou mais linguagens de especificação e algumas ferramentas auxiliares;
- ▶ Várias linguagens existentes baseadas em diversos formalismos (Extended ML, Z, B Method, Maude, Larch, CASL ...);
- ▶ Variado nível de suporte à verificação automática auxiliada por ferramentas.

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Linguagens Relacionadas

Linguagens Relacionadas



- ▶ Z: linguagem de especificação baseada em teoria dos conjuntos.
- ▶ MÉTODO B: evolução da linguagem Z que engloba desde a especificação até a tradução para linguagem de programação.
- ▶ EML: extensão da linguagem de programação ML para especificação de sistemas.
- ▶ MAUDE: linguagem de especificação executável que permite não-determinismo.
- ▶ LARCH: família de linguagens de especificação com modelagem em duas camadas.

CASL (Common Algebraic Specification Language)

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

- ▶ Linguagem de especificação algébrica criada para ser padrão na área;
- ▶ Possui 4 tipos de especificações semanticamente independentes;
- ▶ Permite extensões e sub-linguagens alterando-se a semântica de apenas 1 tipo de especificação;
- ▶ Extensões e sub-linguagens originam uma família de linguagens;
- ▶ Possui uma biblioteca padrão com especificações para reuso.

- ▶ Linguagem funcional de programação;
- ▶ Implementa o conceito de lógica de segunda ordem: tipos que são funções, polimorfismo e construtores de tipos;
- ▶ Fortemente tipificada: todo elemento possui tipo;
- ▶ Avaliação preguiçosa: um argumento de uma função só é avaliado quando é usado no corpo da função;
- ▶ Pura: não permite efeitos colaterais ou seja, uma função só altera suas variáveis locais.
- ▶ Haskell Prelude: biblioteca padrão com funções de uso comum:
 - ▶ Tipos básicos (Integer, Char, String, Float, ...);
 - ▶ Manipulação de listas;
 - ▶ Manipulação de texto;
 - ▶ Mônadas para operações de E/S em tela e arquivo.

- ▶ Extensão de CASL com conceitos de lógica de segunda ordem: tipos que são funções; polimorfismo e construtores de tipos;
- ▶ Tem a linguagem de programação funcional Haskell como sub-conjunto;
 - ▶ Facilita a transformação da especificação em código Haskell executável;
- ▶ Semântica baseada em teoria dos conjuntos para manter-se próxima de CASL e poder importar suas especificações;
- ▶ Possui avaliação estrita de parâmetros: parâmetros indefinidos sempre resultam em valores de retorno indefinidos;
- ▶ Avaliação preguiçosa de parâmetros pode ser emulada com uma combinação de tipos;
- ▶ Não possui biblioteca padrão com especificações reutilizáveis.

Hets: Heterogeneous Tool Set

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

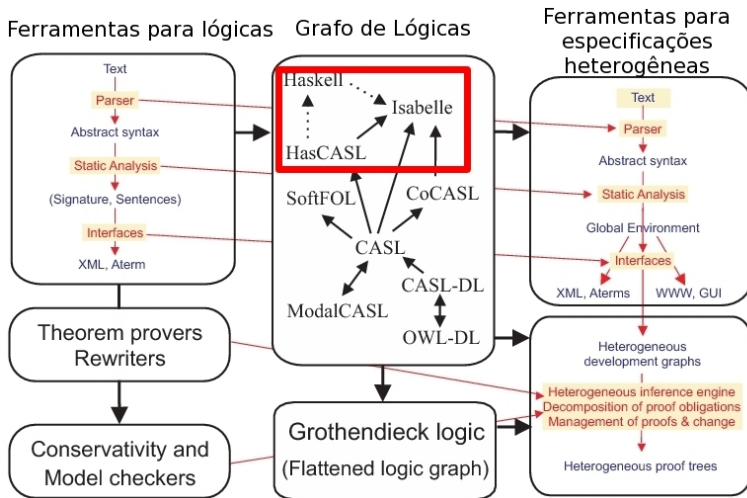
Conclusões

Trabalhos futuros

Agradecimentos

- ▶ Analisador sintático e gerenciador de provas implementado em Haskell;
- ▶ Gerencia ferramentas de provas para as lógicas utilizadas nas extensões e sub-linguagens de CASL:
 - ▶ Lógica de primeira ordem: SPASS;
 - ▶ Lógica de segunda ordem: Isabelle.
- ▶ Gera um Grafo de Desenvolvimento:
 - ▶ Nós: especificações;
 - ▶ Arcos: dependência entre especificações;
 - ▶ Cores indicam o estado das necessidades de prova;
- ▶ Utiliza o editor Emacs como interface:
 - ▶ formatação automática de código;
 - ▶ execução automática da ferramenta *para* de grafos de desenvolvimento.

- ▶ Provedor de teoremas genérico, semi-automático, que permite o uso de várias lógicas como cálculo formal;
- ▶ Automatiza alguns trechos repetitivos de provas: equações, aritmética básica e fórmulas matemáticas;
- ▶ Lógicas para escrita de provas: HOL (Higher-Order Language), HOLCF, etc;
- ▶ A sintaxe de HOL assemelha-se à de Haskell;
- ▶ Hets traduz uma especificação em HasCASL para HOL de forma automática;



Fonte:

http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/index_e.htm

Motivação e Justificativa

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Criação de uma Biblioteca Padrão

- ▶ Contribui para difundir a linguagem;
- ▶ Permite o reuso de especificações;
- ▶ Premissa para uso da linguagem em problemas reais.

Biblioteca Prelude como Ponto de Partida

- ▶ Possui tipos de dados amplamente utilizados em programas Haskell;
- ▶ Permite a importação, em HasCASL, de tipos existentes em Haskell;
- ▶ Facilita a transformação de especificações em HasCASL para código executável em Haskell;

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Objetivo Principal

Especificar uma biblioteca para a linguagem HasCASL com tipos de segunda ordem baseada na biblioteca Prelude da linguagem Haskell.

Objetivo Secundário

Provar teoremas criados pela ferramenta Hets durante a análise da especificação utilizando o provador de teoremas Isabelle.

Escolhas iniciais

Avaliação Estrita

- ▶ Emprego de construções mais simples linguagem HASCASL;
- ▶ Possui alguma documentação e alguns exemplos;
- ▶ Conhecimento básico de HOL para as provas;
- ▶ Escolhida como ponto de partida para a especificação.

Avaliação Preguiçosa

- ▶ Emprego das construções mais avançadas da linguagem HASCASL;
- ▶ Possui pouca documentação e exemplos;
- ▶ Profundo conhecimento prévio das linguagens HOL e HOLCF para as provas;
- ▶ Introduzida em refinamento posterior da especificação.

Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Especificando a Biblioteca

Especificação em HASCASL

- ▶ Modelar o problema *usando* com axiomas em HASCASL e CASL;
- ▶ Criar propriedades a serem verificadas com ISABELLE.

Verificação de Teoremas

- ▶ Necessário para garantir que as especificações se comportam ~~em~~ como esperado;
- ▶ Em algumas provas, os axiomas precisam ser reescritos para que ISABELLE consiga usá-los;
- ▶ Axiomas são reescritos na forma de lemas e também precisam ser provados para garantir consistência;

Especificação Ord em HASCASL

Criação de uma
biblioteca padrão
para HASCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

```
spec Ord = Eq and Bool then
free type Ordering ::= LT | EQ | GT
type instance Ordering: Eq
. (LT == LT) = True      %(IOE01)% %implied
. (LT == EQ) = False    %(IOE04)%
. (LT /= EQ) = True     %(IOE07)% %implied
class Ord < Eq {
  fun __<__ : a * a -> Bool
  var x, y, z, w: a
  . (x == y) = True => (x < y) = False
                                     %(LeIrreflexivity)%
  . (x < y) = True => y < x = False
                                     %(LeTAsymmetry)% %implied
  . (x < y) = True /\ (y < z) = True
    => (x < z) = True                %(LeTTransitive)%
  . (x < y) = True \/ (y < x) = True
    \/ (x == y) = True              %(LeTTTotal)%
}
```

Especificação Ord traduzida para HOL

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

```
LeIrreflexivity [rule_format] :  
  "ALL (x :: 'a). ALL (y :: 'a).  
    x == ' y = True' --> x < ' y = False'"  
  
lemma LeIrreflContra : " x < ' x = True' ==> False"  
by auto  
  
theorem LeTAsymmetry :  
  "ALL (x :: 'a). ALL (y :: 'a).  
    x < ' y = True' --> y < ' x = False'"  
apply(auto)  
apply(rule ccontr)  
apply(simp add: notNot2 NotTrue1)  
apply(rule_tac x="x" in LeIrreflContra)  
apply(rule_tac y = "y" in LeTTransitive)  
by auto
```

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Estado Inicial da Verificação das Especificações

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos



Passo a passo da Especificação e Verificação

1. Realizar verificação sintática com a ferramenta HETS no arquivo da especificação (extensão `.casl` ou `.het`);

The screenshot displays the HETS application window with the file `Prelude.het` open. The interface is divided into two main panes. The top pane shows the source code of the `Prelude.het` file, which defines a library for basic types and operations. The bottom pane shows the command-line output of the HETS tool, which is executing the verification process.

```
library Prelude
version 0.1
%authors: Glauber M. Cabral <glauber.sp@gmail.com>
%date: 19 Feb 2008

logic HasCASL

from Basic/Numbers get Nat, Int, Rat
from HasCASL/Metatheory/Monad get Functor, Monad
from Basic/CharactersAndStrings get Char l-> IChar

spec Bool = %mono
free type Bool ::= True | False
fun Not__ : Bool -> Bool
fun __&&__ : Bool * Bool -> Bool
fun __||__ : Bool * Bool -> Bool
fun otherwiseH: Bool

--=== Prelude.het Top (10.50) SVN-291 (CASL Ind)
/Users/glauber/Hets/bin/hets -v2 /Users/glauber/GlauberMSc/Codigo/Prelude/Prelude.het
Reading file /Users/glauber/GlauberMSc/Codigo/Prelude/Prelude.het
Analyzing library Prelude version 0.1
logic HasCASL
Downloading Basic/Numbers ...
Reading file /Users/glauber/Hets/Hets-lib/Basic/Numbers.casl
Analyzing library Basic/Numbers version 1.0
U:*** "hets-run" Top (b,U) (hets-compile:exit [U])
Beginning of buffer
```

Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas

Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas

Enfrentados

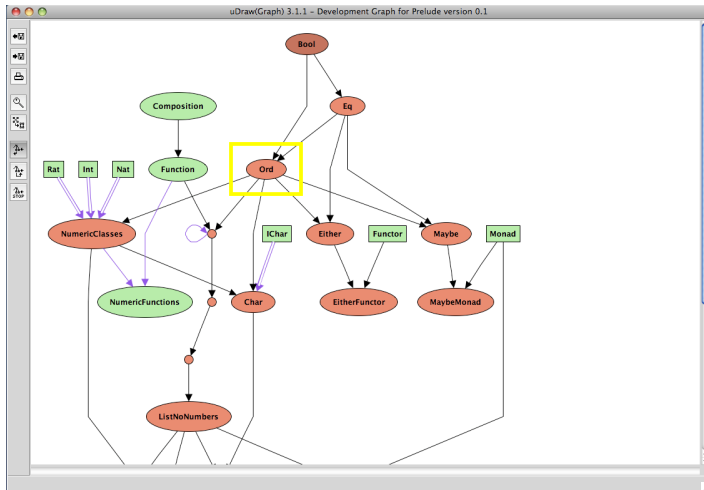
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

2. HETS gera o grafo de desenvolvimento da especificação analisada;



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

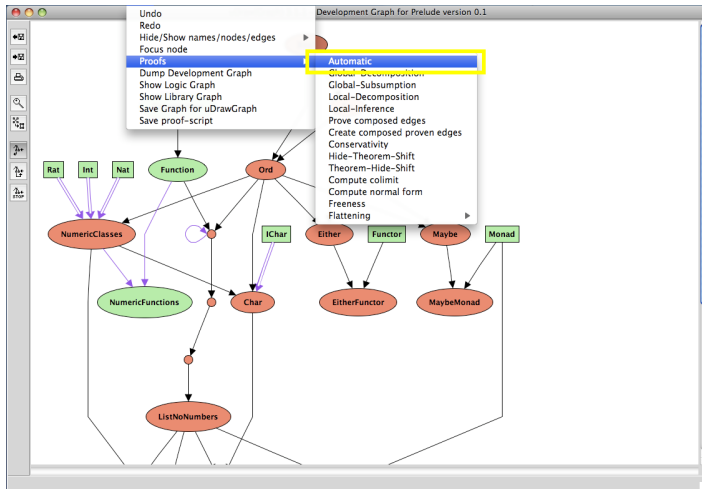
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

3. Executar o comando de prova automático para que ele interprete o grafo e as necessidades de prova;



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas

Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas

Enfrentados

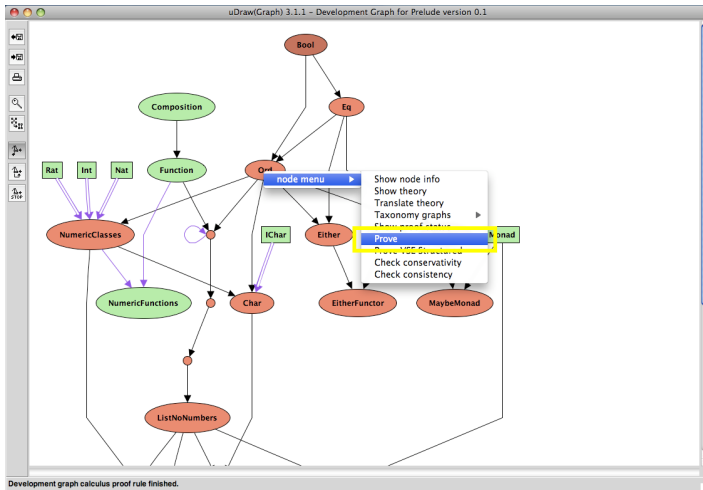
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

4. Selecionar um nó vermelho (com provas em aberto) para verificar seus teoremas;



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

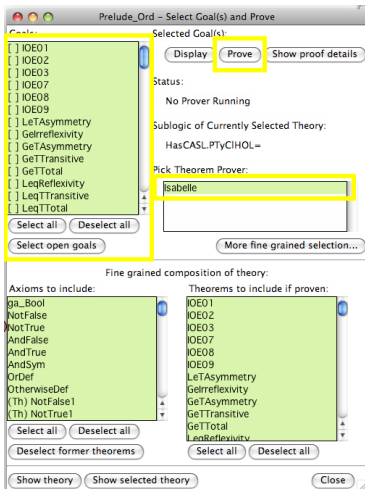
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

- Escolher os teoremas a serem provados, o provador de teorema e iniciar a prova;



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

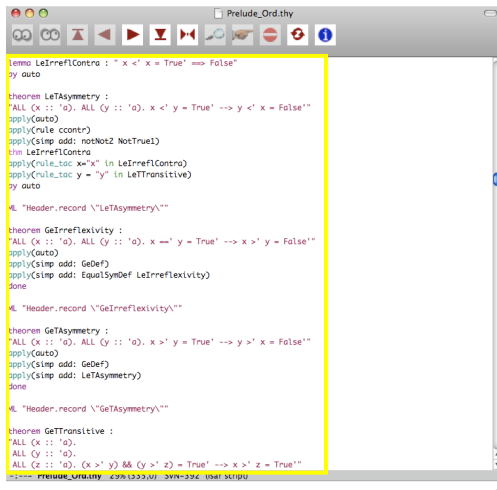
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

6. Escrever as provas utilizando a interface *ProofGeneral* para o provador ISABELLE;



```

Prelude_Ord.thy
[Icons: Undo, Redo, Find, etc.]

lemma LeIrreflContra : "x <' x = True' ==> False"
  by auto

theorem LeAsymmetry :
  "ALL (x :: 'a). ALL (y :: 'a). x <' y = True' --> y <' x = False'"
  apply(auto)
  apply(rule ccontr)
  apply(simp add: notNot2 NotTrue1)
  then LeIrreflContra
  apply(rule_tac x="x" in LeIrreflContra)
  apply(rule_tac y = "y" in LeTransitive)
  by auto

ML "Header.record \"LeAsymmetry\""

theorem GeIrreflexivity :
  "ALL (x :: 'a). ALL (y :: 'a). x ==' y = True' --> x >' y = False'"
  apply(auto)
  apply(simp add: GeDef)
  apply(simp add: EqualSymDef LeIrreflexivity)
  done

ML "Header.record \"GeIrreflexivity\""

theorem GeAsymmetry :
  "ALL (x :: 'a). ALL (y :: 'a). x >' y = True' --> y >' x = False'"
  apply(auto)
  apply(simp add: GeDef)
  apply(simp add: LeAsymmetry)
  done

ML "Header.record \"GeAsymmetry\""

theorem GeTransitive :
  "ALL (x :: 'a).
  ALL (y :: 'a).
  ALL (z :: 'a). (x >' y) && (y >' z) = True' --> x >' z = True'"
  done
Prelude_Ord.thy 299(122,0) 304-392 UseF StipU

```

Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

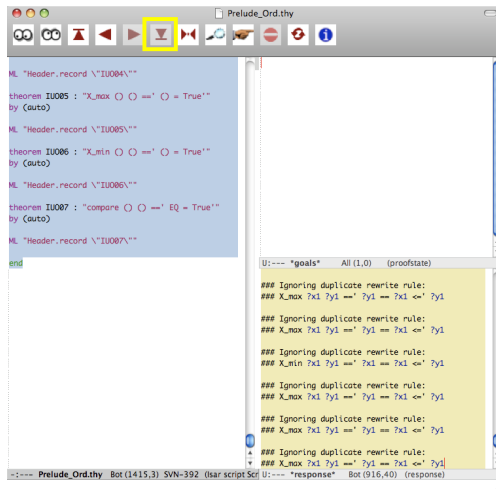
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

7. Executar a verificação completa do arquivo de prova e fechar a janela;



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

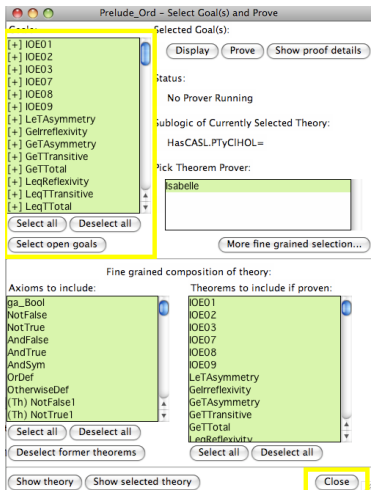
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

8. Os teoremas que foram provados ~~possuem~~ ^{tem} as respectivas caixas de seleção marcadas;



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

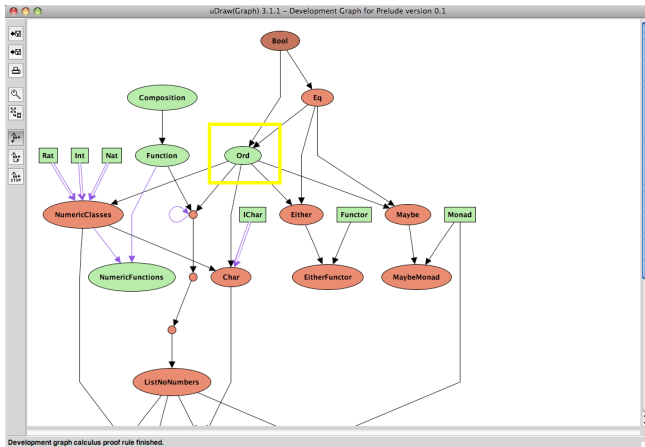
Conclusões

Trabalhos futuros

Agradecimentos

Passo a passo da Especificação e Verificação

9. Resultado da verificação no grafo: verde (totalmente verificado), vermelho (teoremas em aberto), amarelo (falta verificação de consistência).



Criação de uma biblioteca padrão para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Estado Final da Verificação das Especificações

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

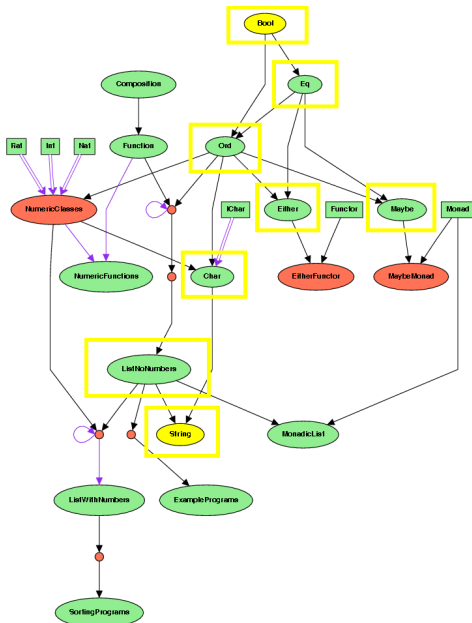
Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos



Principais Contribuições

- ▶ Biblioteca especificada possui os tipos de dados booleano, listas, caracteres e cadeias de caracteres.
- ▶ Especificações de exemplo empregam listas e booleanos;
- ▶ Duas versões para a biblioteca (aprox. 1000 LOC cada):
 - 1ª Versão Tipos com avaliação estrita devido à complexidade do uso de tipos com avaliação preguiçosa;
 - 2ª Versão Refinamento para suportar tipos com avaliação preguiçosa sem suporte a tipos infinitos;
- ▶ A maioria das necessidades de prova foram verificadas:
 - ▶ 9 especificações verificadas totalmente;
 - ▶ 8 especificações possuem alguns teoremas em aberto.

Subtipos

- ▶ Subtipos em CASL ainda não estão mapeados para ISABELLE;
- ▶ Especificações em HASCASL não podem importar especificações em CASL que utilizam subtipos. Ex: tipos de dados numéricos;

Especificações de Tipos Numéricos

- ▶ Funções envolvendo tipos numéricos não foram especificadas na versão atual da biblioteca.
- ▶ A implementação exigiria mapeamentos entre os tipos de dados da biblioteca da linguagem CASL e seus respectivos tipos de dados na linguagem HOL.
- ▶ O mapeamento para o tipo de dados ~~Nat~~^{Int} existente está implementado com funções obsoletas e não pode ser usado dentro do provador ISABELLE.

Tipos Contínuos e Estruturas Infinitas

- ▶ Suporte a tipos de dados contínuos exigiria tipos de dados complexos e uma nova lógica no provador de teoremas ISABELLE.
- ▶ Sem tipos de dados contínuos e estruturas infinitas, não é possível refinar as especificações para o subconjunto executável da linguagem HASCASL.

Problemas Enfrentados

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Mudanças na Tradução para HOL

Mudança na tradução de tipos com avaliação preguiçosa e funções parciais mudou e exigiu adaptação e correção de todas as provas desenvolvidas.

Mapeamento Inicial

```
theory MainHCPairs
```

```
types 'a partial = "bool * 'a"
```

Mapeamento Posterior

```
theory MainHC
```

```
types 'a partial = "'a option"
```

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

Dificuldades com o Provador ISABELLE

- ▶ Alguns teoremas permanecem sem provas;
- ▶ Necessidade de construir e aplicar vários lemas auxiliares;
- ▶ Dificuldade em compreender a aplicação de regras para provar um objetivo em aberto.

Objetivos Iniciais

- ▶ Especificar uma biblioteca para HASCASL baseada na biblioteca PRELUDE;
- ▶ Verificar propriedades da especificação.

Objetivos Alcançados

- ▶ Biblioteca possui os tipos de dados booleano, listas, caracteres e cadeias de caracteres
- ▶ Tipos numéricos não foram especificados;
- ▶ Estado das necessidades de prova geradas:
 - ▶ 9 totalmente verificadas;
 - ▶ 8 verificadas parcialmente.

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos

- ▶ Escrever novos mapeamentos entre os tipos de dados da biblioteca da linguagem CASL e os tipos de dados da linguagem HOL
- ▶ Especificar e verificar tipos de dados numéricos e funções que os envolvam.
- ▶ Especificar tipos de dados infinitos;
- ▶ Especificar estruturas de dados mais complexas implementadas por alguns compiladores da linguagem HASKELL.

Agradecimentos

Apoio Financeiro:



Contato:

glauber.sp@gmail.com

Criação de uma
biblioteca padrão
para HasCASL

Glauber M. Cabral

Roteiro

Introdução

Métodos Formais

Linguagens Relacionadas

Linguagem e Ferramentas
Utilizadas

Motivação

Objetivos

Desenvolvimento

Contribuições

Problemas
Enfrentados

Conclusões

Trabalhos futuros

Agradecimentos