



Uma Certificação em Coq do Algoritmo W Monádico

Rafael Castro

rafaelcgs10@gmail.com

Departamento de Ciência da Computação
Centro de Ciências e Tecnológicas
Universidade do Estado de Santa Catarina

22 de Agosto de 2019

Sumário

- Introdução
- Fundamentos
- Certificação da Unificação
- Certificação do Algoritmo W

Introdução

Introdução - Sistemas de Tipos

- Sistemas de tipos são relações que ditam quais expressões podem ser associadas a quais tipos.
- Nas linguagens de programação, classificavam os dados para serem tratados corretamente pelo processador;
- Passaram a ser utilizados como uma técnica para identificar falhas na consistência de programas.
- Tipagem estática e dinâmica:
 - Estática: atribuições de tipo não mudam em *run time*; uma forma de verificação automática.
 - Dinâmica: atribuições de tipo **podem mudar** em *run time*; mais permissível; mais permissível a erros.

[1, 'a', objeto, [2]]

Introdução - Inferência de Tipos e Polimorfismo

- A **inferência de tipos** é o processo de encontrar a assinatura de tipo de um programa.
- Liberdade ao programador de escolher anotar ou não os tipos das funções.
- Polimorfismo é quando uma função pode assumir vários usos com diferentes tipos de dados. Em especial, o polimorfismo paramétrico é feito pela quantificação de variáveis de tipos.

reverse :: [a] \rightarrow [a]

Introdução - Algoritmo para Inferência de Tipos

- O **algoritmo W** é um algoritmo para a inferência de tipos no sistema Damas-Milner.
Uma das bases para a inferência em Haskell e OCaml.
- *Idealmente* um algoritmo de inferência deve *representar* o que o seu sistema de tipos define.
- Formalmente isso é *Soundness* e *Completeness*.
- *Essas propriedades são diretamente refletidas nas implementações?*

Introdução - Provas mecanizadas

- O uso de assistentes de provas na pesquisa de linguagens de programação.
- Aproximam formalização e implementação.
- Reduzem o trabalho da verificação da prova.
- *PoplMark Challenge* e *Principles of Programming Languages* (POPL)

Introdução - O que é este trabalho?



Introdução - O que é este trabalho?

Uma Certificação em Coq do
Algoritmo W Monádico

Introdução - O que é este trabalho?

Uma Certificação em Coq do Algoritmo W Monádico

- Uma completa certificação de uma implementação monádica do **algoritmo W** no assistentes de provas Coq.
- Certificação do algoritmo de unificação.
- Uso da *Hoare State Monad* (HSM) para certificar o código monádico.

Introdução - Objetivos específicos

- Uma implementação certificada do **algoritmo de unificação**: consistência, completude e terminação.
- Modificar a *Hoare State Monad* para lidar com o efeito de exceção necessário no algoritmo W e avaliar as qualidades dessa técnica.
- Provar a consistência e a completude do algoritmo W monádico.

Fundamentos

Fundamentos - Damas-Milner e algoritmo W

- Damas-Milner: polimorfismo via *let*: forma restrita de polimorfismo.
- O algoritmo W é consistente e completo em relação as regras de Damas-Milner.
- Consistência (*Soundness*): O que o algoritmo W infere pode ser demostrado pelas regras de Damas-Milner.
- Completude (*Completeness*): Se o sistema Damas-Milner mostra que e tem o tipo τ , então o algoritmo infere para e o tipo τ' , de maneira que $\tau = \mathbb{R}(\tau')$ para alguma substituição \mathbb{R} .

Fundamentos - Trabalhos relacionados

- [Dubois and Ménissier-Morain, 1999] forneceram provas em Coq de consistência e completude do algoritmo W. As propriedades da unificação foram tomadas como axiomas e *binding* de variáveis é resolvido com *de Bruijn indices*.
- [Naraschewski and Nipkow, 1999] também apresentaram provas de consistência e completude do algoritmo W, mas em Isabelle/HOL. As propriedades da unificação também foram assumidas como axiomas e *de Bruijn indices* também foi utilizado.
- [Kothari and Caldwell, 2009] relataram um resultado parcial na certificação em Coq da extensão polimórfica do algoritmo de Wand.
- [Tan et al., 2015] verificaram a inferência de tipos de CakeML, uma linguagem baseada em SML (Standard ML) cujo compilador tem um *backend* completamente verificado [Kiam Tan et al., 2019]. As provas de consistência e completude da inferência de tipos foram feitas no assistente de provas HOL4.

Fundamentos - Assistentes de provas

- São programas para o desenvolvimento de provas formais.
- O núcleo é um verificador, que verifica a consistência lógica da prova.
- Os principais apelos são:
 - a verificação mecânica é rápida e evita as falhas humanas;
 - interatividade, permite visualizar informações sobre os estados da prova;
 - comandos para busca de teoremas e lemas para o progresso da prova;
 - automatização de provas com métodos não-deterministas;
 - potencialização da capacidade humana de realizar provas;
 - extração de programas verificados.

Fundamentos - Hoare State Monad

- *Hoare State Monad* (HSM): apresentada por [Swierstra, 2009]
Uma variante da usual mònada de estados.
- Verificação de programas com efeitos de estado.
- Um tipo HoareState $p \ a \ q : a$ pré-condição, o tipo do valor de retorno e a pós-condição.

```
Program Definition HoareState (pre : Pre) (a : Type) (post : Post a) : Type :=
  forall i : {t : st | pre t}, {(x, f) : a * st | post i x f}.
```

```
Program Definition ret (a : Type) : forall x,
  @HoareState top a (fun i y f => i = f /\ y = x) := fun x s => (x, s).
```

```
Program Definition bind : forall a b P1 P2 Q1 Q2,
  (@HoareState P1 a Q1) -> (forall (x : a), @HoareState (P2 x) b (Q2 x)) ->
  @HoareState (fun s1 => P1 s1 /\ forall x s2, Q1 s1 x s2 -> P2 x s2)
    b
    (fun s1 y s3 => exists x, exists s2, Q1 s1 x s2 /\ Q2 x s2 y s3) :=
      fun a b P1 P2 Q1 Q2 c1 c2 s1 => match c1 s1 as y with
        | (x, s2) => c2 x s2
      end.
```

Figura: A Mônada de Estado de Hoare.

Fundamentos - Hoare Exception-State Monad

- O algoritmo W tem dois tipos de efeitos: estado e exceção.
 - *Hoare Exception-State Monad* (HESM):
embute-se o tipo *sum* na definição de HoareState.

```

Program Definition HoareState (B:Prop) (pre:Pre) (a:Type) (post:Post a) : Type :=
  forall i : {t : st | pre t},
    {e : sum (prod a st) B | match e with
      | inl (x, f) => post (proj1_sig i) x f
      | inr _ => True
      end}.

```

```

Program Definition bind : forall a b P1 P2 Q1 Q2 B,
  (@HoareState B P1 a Q1) -> (forall (x:a), @HoareState B (P2 x) b (Q2 x)) ->
  @HoareState B (fun s1 => P1 s1 /\ forall x s2, Q1 s1 x s2 -> P2 x s2) b
    (fun s1 y s3 => exists x, exists s2, Q1 s1 x s2 /\ Q2 x s2 y s3) :=
  fun B a b P1 P2 Q1 Q2 c1 c2 s1 => match c1 s1 as y with
    | inl (x, s2) => c2 x s2
    | inr R => _
  end.

```

Figura: A Mônada de Estado de Hoare.

Certificação da unificação

Certificação da unificação - Aspectos da certificação

- São programas para o desenvolvimento de provas formais.
- O núcleo é um verificador, que verifica a consistência lógica da prova.
- Os principais apelos são:
 - a verificação mecânica é rápida e evita as falhas humanas;
 - interatividade, permite visualizar informações sobre os estados da prova;
 - comandos para busca de teoremas e lemas para o progresso da prova;
 - automatização de provas com métodos não-deterministas;
 - potencialização da capacidade humana de realizar provas;
 - extração de programas verificados.

Certificação do algoritmo W

Certificação do algoritmo W - Estabilidade da substituição

- O lema da estabilidade da substituição é uma propriedade clássica em sistemas de tipos e é fundamental na prova da consistência.
- Se é verdade que $\Gamma \vdash e : \tau$, então para qualquer substituição \mathbb{S} tem-se $\mathbb{S}\Gamma \vdash e : \mathbb{S}\tau$.
- Casos monomórficos são fáceis. Além do caso polimórfico do `let_ht` há também o caso do `fix_ht`.

Conclusão

- Fundamentos teóricos para formalização do MMo.
- Assistentes de provas Coq.
- Como o MMo funciona.
- Formalização do MMo parcial.
- Principais dificuldades encontradas.
- Alternativas para a prova de terminação.

References I

-  Dubois, C. and Ménissier-Morain, V. (1999).
Certification of a Type Inference Tool for ML: Damas-Milner
within Coq.
Journal of Automated Reasoning, 23(3-4):319–346.
-  Kiam Tan, Y., Myreen, M. O., Kumar, R., Fox, A., Owens, S.,
and Norrish, M. (2019).
The Verified CakeML Compiler Backend.
Journal of Functional Programming, 29.
-  Kothari, S. and Caldwell, J. L. (2009).
Toward a machine-certified correctness proof of Wand's type
reconstruction algorithm.

References II

-  Naraschewski, W. and Nipkow, T. (1999).
Type Inference Verified: Algorithm script W sign in Isabelle/HOL.
Journal of Automated Reasoning, 23(3-4):299–318.
-  Swierstra, W. (2009).
The Hoare State Monad.
Technology, pages 440–451.
-  Tan, Y. K., Owens, S., and Kumar, R. (2015).
A verified type system for CakeML.
pages 1–12.