Indecibilidade da inferência de tipos com recursão polimórfica

Gabriela Moreira e Luiz Gustavo Eburneo

Departamento de Ciência da Computação
Centro de Ciências Tecnológicas
Universidade do Estado de Santa Catarina (UDESC)
Joinville, SC – Brasil

Abstract

O cálculo Damas-Milner é um cálculo Lambda tipado utilizado no sistema de tipos de ML e outras linguagens funcionais. Essa definição permite a declaração de funções polimórficas paramétricas somente com a condição de que ocorrências da função recursiva no corpo de sua declaração sejam usadas monomorficamente. O cálculo Milner-Mycroft extende essa definição, sugerindo uma regra de tipagem polimórfica para definições recursivas e, portanto, permitindo que ocorrências da função em sua definição sejam usadas polimorficamente. Inferir os tipos nesse novo sistema, entretanto, é indecidível. A demonstração da indecibilidade se dá pela prova da redução do problema da semi unificação, provado indecidível.

Keywords: Indecibilidade, Inferência de Tipos, Milner-Mycroft, Recursão Polimórfica

1. Introdução

O cálculo lambda foi proposto por Alonzo Church na década de 1930 como um modelo universal de computação, e consiste em substituições de variáveis e reduções sobre expressões lambda, que são dadas por

$$e := v|e_1e_2|\lambda x.e$$

Mais tarde, Luis Damas e Robin Milner propõe uma extensão para esse modelo, baseando-se no sistema de tipos da liguagem de programação funcional ML, que ficou conhecido como cálculo Damas-Milner. Juntamente

com a definição, é proposto o algoritmo W que infere tipos de expressões escritas no sistema.

Esse sistema permite a definição e uso de funções parametricamente polimórficas (uniformes). Ele restringe, contanto, que ocorrências de funções recursivas no corpo de sua própria declaração sejam usadas apenas monomorficamente - ou seja, os argumentos e o retorno devem manter o mesmo tipo em todas as ocorrências da função - impedindo assim a declaração e uso de tipos não uniformes.

Vendo isso, Mycroft (1984) e Meertens (1983) (independentemente) sugerem uma regra polimórfica para definições recursivas, tornando possível que ocorrências da função em seu corpo sejam usadas polimorficamente. A inclusão dessa regra no cálculo Damas-Milner orginou um novo sistema, denominado cálculo Milner-Mycroft. Este novo modelo é consistente (Henglein 1993), porém o algoritmo de inferência de tipos baseado em unificação, como o W, perde sua completude quando adicionada a inclusão polimórfica. Procura-se então um algoritmo que infira tipos nesse novo sistema. Entretanto, será mostrado que esse problema é redutível ao problema da semi unificação, que foi provado ser indecidível (Kfoury et al. 1993). Ou seja, um algoritmo decidível para a inferência não pode existir.

O problema da semi unificação está relacionado ao problema da unificação, e ambos podem ser abordados sobre sistemas de equações e inequações (SEI's). A prova da redução é feita a partir desses SEI's, convertendo-os para expressões em sistemas de tipos de forma que, se porventura houvesse um algoritmo que sempre para inferindo o tipo correto para uma expressão, qualquer SEI poderia ser convertido, obtendo uma solução a partir desse algoritmo. Isso tornaria o problema da semi unificação decidível, e portanto não pode ocorrer.

2. Sistemas de Tipos

Sistemas de tipos são conjuntos de regras sobre sentenças (*judgments*), que tem forma:

$$\Gamma \vdash e : \sigma$$

Ou seja, num contexto Γ, é derivável que a expressão e tem tipo σ. As regras sobre as sentenças tem forma:

$$\frac{\Gamma_1 \vdash e_1 : \sigma_1 \quad \Gamma_2 \vdash e_2 : \sigma_2 \quad \ldots \quad \Gamma_n \vdash e_n : \sigma_n}{\Gamma \vdash e : \sigma}$$

Onde as sentenças acima da linha divisória são premissas, e a sentença abaixo é a conclusão.

2.1. Cálculo Lambda Simplesmente Tipado

Publicado por Alonzo Church (1940), o cálculo lambda simplesmente tipado é uma interpretação do cálculo lambda incluindo teoria de tipos. Nesse sistema, além de expressões lambda, são definidos tipos simples. Um tipo simples τ tem forma:

$$\tau := \alpha \mid \tau \to \tau'$$

Onde α é uma variável de tipo e $\tau \to \tau'$ é o tipo de uma função que recebe um parâmetro do tipo τ e retorna um tipo τ'.

Sobre expressões lambda e tipos simples, define-se o seguinte conjunto de regras base:

$$\Gamma \vdash x : \tau \ (VAR) \quad \{x : \tau\} \in \Gamma$$

$$\frac{\Gamma \vdash e : \tau \to \tau' \quad \Gamma \vdash e' : \tau}{\Gamma \vdash e\ e' : \tau'} \ (APP) \qquad \frac{\Gamma, x : \tau' \vdash e : \tau}{\Gamma \vdash \lambda x.e : \tau' \to \tau} \ (ABS)$$

Acrescentando regras, são definidas as interpretações dos tipos **unit** (unitário, produto nulo), produto e união disjunta:

$$\Gamma \vdash \langle\rangle : \text{unit}$$

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash e' : \tau'}{\Gamma \vdash e \times e' : \tau \times \tau'} \ (PAIR)$$

$$\frac{\Gamma \vdash e \times e' : \tau \times \tau'}{\Gamma \vdash \pi_1(e \times e') : \tau} \ (PROJ) \quad \frac{\Gamma \vdash \ e \times e' : \tau \times \tau'}{\Gamma \vdash \pi_2(e \times e') : \tau'}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash e + e' : \tau + \tau'} \ (SUM) \quad \frac{\Gamma \vdash e' : \tau'}{\Gamma \vdash e + e' : \tau + \tau'}$$

$$\frac{\Gamma, x_1 : \tau_1 \vdash e_1 : \tau \quad \Gamma, x_2 : \tau_2 \vdash e_2 : \tau \quad \Gamma \vdash e : \tau_1 + \tau_2}{\Gamma \vdash \text{case } e \text{ of } \{x_1 \Rightarrow e_1 \mid x_2 \Rightarrow e_2\} : \tau} \ (CASE)$$

Tipos recursivos podem ser definidos através do operador de ponto fixo, com o uso de expressões de tipo $\mu\alpha.\tau$, que denota o isomorfismo dos tipos que satisfazem a equação $\mu\alpha.\tau \cong \{\alpha \mapsto \mu\alpha.\tau\}\tau$. Ou seja, α pode ser substituído por uma expressão de tipo $\mu\alpha.\tau$ que, por sua vez, pode ter o α substituído por uma nova expressão desse tipo, formando um tipo recursivo. As regras para definir tal isomorfismo precisam ser acrescentadas ao sistema:

$$\frac{\Gamma \vdash e : \{\alpha \mapsto \mu\alpha.\tau\}\tau}{\Gamma \vdash \text{fold } e : \mu\alpha.\tau} \ (FOLD) \quad \frac{\Gamma \vdash e : \mu\alpha.\tau}{\Gamma \vdash \text{unfold } e : \{\alpha \mapsto \mu\alpha.\tau\}\tau} \ (UNFOLD)$$

O operador $\mu\alpha.\tau$ é denominado **fix**.

2.2. Sistema Damas-Milner

Luis Damas e Robin Milner (1982) definem uma extensão do cálculo lambda simplesmente tipado para aceitação de polimorfismo paramétrico. Eles ainda propõem um algoritmo para a inferência de tipos nesse novo sistema, que também é ficou conhecido como Sistema Hindley-Milner ou Damas–Hindley–Milner.

Para a interpretação de tipos parametricamente polimórficos, são definidos tipos quantificados. Assim, tem-se a combinação de tipos simples (τ) e tipos quantificados (σ):

$$\tau := \alpha \mid \tau \to \tau'$$
$$\sigma := \tau \mid \forall \alpha.\sigma$$

Ao quantificar tipos de uma expressão, é necessário saber quais variáveis são livres no contexto (Γ). A lista de variáveis livres em um contexto é chamada de $ftv(\Gamma)$ e se dá pela união $ftv(\sigma)$ para todo tipo σ que ocorre em Γ. ftv é definido:

$$ftv(\alpha) = \{\alpha\}$$
$$ftv(\tau \to \tau') = ftv(\tau) \cup ftv(\tau')$$
$$ftv(\forall \alpha.\sigma) = ftv(\sigma) - \{\alpha\}$$

Assim, são acrescentadas três novas regras para quantificar, instanciar e utilizar tipos parametricamente polimórficos:

$$\frac{\Gamma \vdash e : \forall \alpha.\sigma \quad (\alpha \notin ftv(\Gamma))}{\Gamma \vdash e : \sigma} \ (GEN) \qquad \frac{\Gamma \vdash e : \sigma \quad (\sigma \geqslant \sigma')}{\Gamma \vdash e : \sigma'} \ (INST)$$

$$\frac{\Gamma \vdash \text{let } x = e \text{ in } e' : \tau}{\Gamma \vdash e : \sigma \quad \Gamma, x : \sigma \vdash e' : \tau} \ (LET)$$

A inferência de tipos para esse sistema é decidível. O algoritmo W foi proposto por Damas e Milner (1982) e infere o tipo mais geral possível para uma expressão nesse sistema. Um tipo mais geral (σ) de uma expressão é tal que qualquer outro tipo (σ') para a mesma expressão é uma instância dele ($\sigma \geqslant \sigma'$). Os autores ainda provam a completude e a corretude do algoritmo.

2.3. Sistema Milner-Mycroft

Em 1984, Alan Mycroft propõe uma extensão ao sistema Damas-Milner com uma regra mais geral para definições recursivas. A modificação proposta trata-se apenas de aceitar o uso de tipos quantificados com operadores de ponto fixo, o que depende de uma alteração nas regras ($FOLD$) e ($UNFOLD$):

$$\frac{\sigma \vdash e : \{\sigma \mapsto \mu\sigma.\tau\}\tau}{\sigma \vdash \text{fold } e : \mu\sigma.\tau}\ (FOLD') \qquad \frac{\Gamma \vdash e : \mu\sigma.\tau}{\sigma \vdash \text{unfold } e : \{\sigma \mapsto \mu\sigma.\tau\}\tau}\ (UNFOLD')$$

Assim, esse sistema aceita que tipos recursivos sejam usados polimorficamente. Com essa alteração, muitas características do sistema Damas-Milner são preservadas, como a propriedade do tipo principal - que garante que qualquer expressão tipável pelo sistema possui um único tipo mais geral. A inferência de tipos, contudo, não é decidível nesse novo sistema.

2.4. Sistema Milner-Mycroft extendido

Para a prova da redução, algumas alterações no sistema serão definidas. A prova de que essas alterações não tornam o sistema mais fraco nem mais forte que o original foi mostrada por Henglein (1993). O sistema trabalhado define-se como se segue:

$$\frac{(\tau, \vec{\tau}) \geqslant (\tau', \vec{\tau})}{\Gamma \cup \{x : \tau\}, \vec{\tau} \vdash x : \tau'}\ (TAUT')$$

$$\frac{\Gamma \cup \{x : \tau_x\}, (\vec{\tau}, \tau_x) \vdash e : \tau \qquad \tau' = \tau_x \to \tau}{\Gamma, \vec{\tau} \vdash \lambda x.e : \tau'}\ (ABS')$$

$$\frac{\Gamma, \vec{\tau} \vdash e : \tau \qquad \Gamma, \vec{\tau} \vdash e' : \tau' \qquad \tau = \tau' \to \tau''}{\Gamma, \vec{\tau} \vdash (ee') : \tau''}\ (APPL')$$

$$\frac{\Gamma, \vec{\tau} \vdash e : \tau \qquad \Gamma \cup \{x : \tau_x\}, \vec{\tau} \vdash e' : \tau' \qquad \tau = \tau', \tau' = \tau''}{\Gamma, \vec{\tau} \vdash \ \textbf{let } x = e \textbf{ in } e' : \tau''}\ (LET')$$

$$\frac{\Gamma \cup \{x : \tau_x\}, \vec{\tau} \vdash e : \tau \qquad \tau_x = \tau, (\tau, \vec{\tau}) \geqslant (\tau', \vec{\tau})}{\Gamma, \vec{\tau} \vdash \textbf{fix } x.e : \tau'}\ (FIX\text{-}P')$$

3. Problemas

Deseja-se mostrar a redução do problema da semi unificação para o problema da inferência de tipos com recursão polimórfica. Inicialmente, serão definidos esses problemas.

3.1. Inferência de tipos com recursão polimórfica

Um tipo recursivamente polimórfico é aquele que ocorre em sua própria definição. Uma árvore binária totalmente balanceada (onde cada nó tem exatamente dois filhos) pode assumir essa representação:

```
data Arvore a = Cons a (Arvore (a,a)) | Nil
```

Uma árvore de altura igual a três pode ser descrita como:

```
Cons 1 (Cons (2,3) (Cons ((4,5),(6,7)) Nil))
```

Pode-se declarar uma função para calcular a altura de uma árvore:

```
altura ::  Arvore a -> Int
altura Nil = 0
altura (Cons n a) = 1 + altura a
```

Essa função requer uma assinatura de tipo. O problema em questão é a inferência do tipo de expressões como a função *altura*, onde ocorre recursão polimófica. Analisando as chamadas dessa função para o exemplo dado, o polimorfismo fica evidente.

```
1ª chamada:  n = 1,             a =Cons (2,3) (Cons ((4,5),(6,7)) Nil)
2ª chamada:  n = (2,3),         a = Cons ((4,5),(6,7)) Nil
3ª chamada:  n = ((4,5),(6,7)), a = Nil
```

Ou seja, o tipo do parâmetro n é diferente para cada chamada da função.

3.2. Semi Unificação

O problema da semi unificação é uma generalização do problema da unificação. Dois tipos σ e σ' são ditos unificáveis se e somente se existe uma substituição S da forma $\{\alpha_1 \mapsto \tau_1, \alpha_2 \mapsto \tau_2, \ldots, \alpha_n \mapsto \tau_n\}$ tal que $S(\sigma) = S(\sigma')$. A aplicação de S em um tipo substitui cada ocorrência de α_i pelo respectivo τ_i em S.

Dado um conjunto de pares de tipos $\{(M_1, N_1), (M_2, N_2), \ldots, (M_n, N_n)\}$ onde $M_i \geqslant N_i$ (M_i é mais geral que N_i). O conjunto é dito semi unificável se e somente se existe um semi unificador S e um conjunto de substituições $\{S_1, S_2, \ldots, S_n\}$ tal que

$$S_i(S(M_i) = S(N_i)$$

Esse problema pode ser descrito em sistemas de equações e inequações (SEI) de tipos. Seja um SEI $\{M_{01} = N_{01}, M_{02} = N_{02}, \ldots, M_{0n} = N_{0n}, M_1 \geqslant N_1, M_2 \geqslant N_2, \ldots, M_m \geqslant N_m\}$. Um semi unificador S e um conjunto de m substituições $\{S_1, S_2, \ldots, S_m\}$ para esse SEI são tais que:

$$\begin{gathered} S(M_{0i}) = S(N_{0i}) \\ S(M_j) \geqslant S(N_j) \\ S_j((S(M_j)) = S(N_j) \end{gathered}$$

Para i de 1 a n e j de 1 a m. Nota-se que um semi unificador para uma equação de tipos é exatamente seu unificador.

4. Redução

Sabe-se que o problema da semi unificação é indecidível, conforme provado por Kfoury et al. (1993). Para provar a indecibilidade do problema da inferência de tipos com recursão polimórfica, será mostrada uma função computável que transforma uma instância do problema da semi unificação - um SEI - em uma expressão do sistema Milner-Mycroft. Essa função é de tal forma que o SEI será unificável se e somente se a expressão for tipável no sistema. Assim, um inferidor de tipos para a expressão decidiria o problema da semi unificação, o que foi provado impossível.

Construindo a redução, a aplicação de um contexto Γ é usada também como uma substituição $\Gamma(M)$. A transformação se dá trivialmente pela troca de cada elemento da forma $x : \tau$ em Γ para um da forma $x \mapsto \tau$.

Também serão utilizados pares e projeções com uma definição alternativa, para trabalhar com igualdade de tipos nas expressões. Define-se:

$$(e, e1) = \lambda x.xee'$$
$$e.1 = e(\lambda x.\lambda y.x)$$
$$e.2 = e(\lambda x.\lambda y.y)$$

com $x \notin ftv(e) \cup ftv(e')$. As regras para derivação de tipos são definidas:

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash e' : \tau'}{\Gamma \vdash (e, e') : (\tau, \tau')_{\tau''}} \ (PAIR')$$

$$\frac{\Gamma \vdash \ e : (\tau, \tau')_{\tau}}{\Gamma \vdash e.1 : \tau} \ (PROJ') \ \frac{\Gamma \vdash \ e : (\tau, \tau')_{\tau'}}{\Gamma \vdash e.2 : \tau'}$$

O tipo correspondente a $(e, e1) = \lambda x.xee'$ é

$$(\tau, \tau')_{\tau''} = (\tau \to \tau' \to \tau'') \to \tau''.$$

O símbolo $\doteq$ será usado nas expressões como um operador para garantir igualdade de tipos. Dadas duas expressões e e e', escreve-se $e \doteq e'$ para a expressão $\lambda y.(ye, ye')$ com $y \notin ftv(e) \cup ftv(e')$. Essa expressão usa a definição de par definida acima para garantir que duas expressões são unificáveis, e o símbolo $\doteq$ deixará a leitura da prova mais limpa.

A conversão será feita em duas partes, dividindo equações e inequações no SEI. Equações serão convertidas em expressões do sistema Damas-Milner, onde a inferência de tipos é decidível. Inequações serão convertidas em expressões do sistema Milner-Mycroft da forma **fix** $y.e$.

4.1. Conversão de equações

Dada uma equação de tipos $M = N$, constroe-se um contexto Γ e uma expressão $\lambda\vec{x}.(m \doteq n)$ onde m e n são expressões tal que $\Gamma \vdash m : M$, $\Gamma \vdash n : N$, e $\vec{x} = ftv(M) \cup ftv(N)$. Ou seja, $M = N$ é semi unificável se e somente se $\lambda\vec{x}.(m \doteq n)$ é tipável. Como $M = N$ é uma equação, seu semi unificador é igual ao seu unificador, formando o teorema a seguir.

Teorema 1. *$M = N$ é unificável $\iff$ $\lambda\vec{x}.(m \doteq n)$ é tipável*

Prova. ($\Rightarrow$) Seja S um unificador para M e N. Seja Γ' um novo contexto tal que $\Gamma'(x) = \Gamma(S(x))$ para todo x no domínio de Γ (tal operação pode ser entendida como a aplicação da substituição no contexto). Assim, deriva-se $\Gamma' \vdash m : \Gamma(S(M))$ e $\Gamma' \vdash n : \Gamma(S(N))$. Pela definição de unificação,

$S(M) = S(N)$, o que satisfaz os tipos esperados pela função ($\doteq$). Assim, a expressão $\lambda\vec{x}.(m \doteq n)$ é tipável no contexto Γ'.

($\Leftarrow$) Se $\lambda\vec{x}.(m \doteq n)$ é tipável, então existe um contexto Γ e um tipo τ tal que $\Gamma \vdash m : \tau$ e $\Gamma \vdash n : \tau$. Escrevendo Γ como uma substituição, tem-se $\Gamma(M) = \tau$ e $\Gamma(N) = \tau$, e então $\Gamma(M) = \Gamma(N)$. Assim, Γ é um unificador para os tipos M e N.

□

4.2. Conversão de inequações

Sem perder generalidade, conforme Henglein (1993), é possível reduzir a prova e mostrar a conversão de um sistema de apenas duas inequações. Assim, um sistema de inequações $M_1 \geqslant N_1, M_2 \geqslant N_2$ é convertido para uma expressão do sistema Milner-Mycroft da forma.

$$e = \mathbf{fix} f.\lambda\vec{x}.K(m_1 \doteq m_2)(\lambda\vec{y}.((f\vec{y}).1 \doteq n_1), \lambda\vec{y}.((f\vec{y}).2 \doteq n_2)$$

Onde $\vec{x} = ftv(M_1) \cup ftv(M_2) \cup ftv(N_1) \cup ftv(N_2)$ e $K = \lambda x.\lambda y.x$. Assim como um contexto Γ tal que $\Gamma \vdash m_1 : M_1, \Gamma \vdash m_2 : M_2, \Gamma \vdash n_1 : N_1$ e $\Gamma \vdash n_2 : N_2$.

Teorema 2. *$\{M_1 \geqslant N_1, M_2 \geqslant N_2\}$ é semi unificável $\iff$ e é tipável*

Prova. Analisando as regras do sistema Milner-Mycroft extendido, sabe-se que e é tipável se e somente se existem os tipos $\vec{\tau}, \vec{\tau}', \vec{\tau}'', \tau_1, \tau_2, \tau_1', \tau_2', \tau_1'', \tau_2''$ tal que as seguintes derivações sejam possíveis (com ϵ representando a sequência vazia):

$$\{f : \vec{\tau} \to (\tau_1, \tau_2)\}, \epsilon \vdash f : \vec{\tau}' \to (\tau_1', \tau_2')$$
$$\{f : \vec{\tau} \to (\tau_1, \tau_2)\}, \epsilon \vdash f : \vec{\tau}'' \to (\tau_1'', \tau_2'')$$
$$\{\vec{x} : \vec{\tau}\}, \vec{\tau} \vdash m_1 : \tau_1$$
$$\{\vec{x} : \vec{\tau}\}, \vec{\tau} \vdash m_2 : \tau_2$$
$$\{\vec{x} : \vec{\tau}\}, \vec{\tau} \vdash n_1 : \tau_1'$$
$$\{\vec{x} : \vec{\tau}\}, \vec{\tau} \vdash n_2 : \tau_2''$$

E utilizando o contexto $\Gamma = \{\vec{x} : \vec{\tau}\}$ como substituição nos tipos, obtem-se:

$$\tau_1 = \Gamma(M_1)$$
$$\tau_2 = \Gamma(M_2)$$
$$\tau_1' = \Gamma(N_1)$$
$$\tau_2'' = \Gamma(N_2)$$

Assim, pela regra ($TAUT'$), os tipos para f são deriváveis se e somente se

$$\Gamma(\vec{x}), \Gamma(M_1), \Gamma(M_2)) \geqslant (\vec{\tau}', \Gamma(N_1), \tau_1''),$$
$$\Gamma(\vec{x}), \Gamma(M_1), \Gamma(M_2)) \geqslant (\vec{\tau}'', \tau_2', \Gamma(N_2)).$$

E como $\vec{\tau}', \vec{\tau}'', \tau_1''$ e τ_2' podem assumir qualquer valor, resta:

$$(\Gamma(M_1) \geqslant \Gamma(N_1),$$
$$(\Gamma(M_2) \geqslant \Gamma(N_2).$$

Pela definição de semi unificação, isso acontece se e somente se $\{M_1 \geqslant N_1, M_2 \geqslant N_2\}$ é semi unificável.

□

5. Conclusão

Sistemas de tipos são projetados para evitar erros em tempo de execução, que poderiam acontecer com operações entre tipos incompatíveis. Isso aumenta o nível de segurança que um programador tem quando utiliza uma linguagem. O custo disso, em muitas linguagens, é a exigência da assinatura de tipos em vários lugares do código fonte.

Assim, surgem os algoritmos de inferência de tipos, que poupam do programador o tempo e a preocupação que tem em assinar tipos durante a produção de código, mantendo a segurança de uma linguagem tipada. Contanto, alguns mecanismos mais complexos dessas linguagens tornam a inferência de tipos longe de trivial. Diante desse problema, muitos cientistas da computação procuram algoritmos melhores para inferir tipos cada vez mais complexos.

No caso de tipos com recursão polimórfica, um algoritmo para inferência de tipos que pare e dê a resposta correta em todos os casos não existe e nunca existirá. Como provado, é um problema indecidível. Sabendo disso, ainda é

possível encontrar algoritmos que infiram os tipos de uma vasta maioria de casos, como Vasconcellos et al. (2003) fez para o caso da recursão polimórfica em Haskell.

References

Church, A., 1940. A formulation of the simple theory of types. Journal of Symbolic Logic 5 (2), 56–68.

Damas, L., Milner, R., 1982. Principal type-schemes for functional programs. In: Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL '82. ACM, New York, NY, USA, pp. 207–212.

Henglein, F., July 1988. Type inference and semi-unification. In: Proc. ACM Conf. on LISP and Functional Programming (LFP), Snowbird, Utah. ACM Press, pp. 184–197.

Henglein, F., April 1989. Polymorphic type inference and semi-unification. Ph.D. thesis, Rutgers University, available as NYU Technical Report 443, May 1989, from New York University, Courant Institute of Mathematical Sciences, Department of Computer Science, 251 Mercer St., New York, N.Y. 10012, USA.

Henglein, F., April 1993. Type inference with polymorphic recursion. ACM Transactions on Programming Languages and Systems (TOPLAS) 15 (2), 253–289.

Kfoury, A., Tiuryn, J., Urzyczyn, P., May 1990. The undecidability of the semi-unification problem. In: Proc. 22nd Annual ACM Symp. on Theory of Computation (STOC), Baltimore, Maryland. pp. 468–476.

Kfoury, A. J., Tiuryn, J., Urzyczyn, P., 1993. The undecidability of the semi-unification problem. Information and Computation 102 (1), 83–101.

Meertens, L., 1983. Incremental polymorphic type checking in b. In: Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. POPL '83. ACM, New York, NY, USA, pp. 265–275.

Mycroft, A., 1984. Polymorphic type schemes and recursive definitions. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 217–228.

Vasconcellos, C., Figueiredo, L., Camarão, C., 08 2003. Practical type inference for polymorphic recursion: An implementation in haskell 21, 873.