{title= レコード計算の
多相型システムと型推論,

author=  @bd_gfngfn ,

date= 2017年9月30日 }

今回扱う論文

- A. Ohori. ***A Polymorphic record calculus and its compilation***, 1995.
- E. Osinski. ***A Polymorphic Type System and Compilation Scheme for Record Concatenation***, 2006.

今回扱う論文

- A. Ohori. ***A Polymorphic record calculus and its compilation***, 1995.
- E. Osinski. ***A Polymorphic Type System and Compilation Scheme for Record Concatenation***, 2006.

(正直なところ間に合いませんでした)

前者に焦点を当てて説明，後者はごく初歩を紹介

レコードとは

直観的には：

- 各要素にラベルがついたタプル
- いわゆる keyval や dictionary，連想配列に似ている（ただしラベル名は文字列値ではなく識別子的に扱う）

$$\{\mathtt{foo}{=}\ 42,\ \mathtt{bar}{=}\ \mathrm{true}\}$$

レコードに対する演算

とりあえず型を度外視するとして，書けると嬉しそうな操作は：

- **“第1級な” 構築**（これは書けて当然）

$$\mathbf{let}\ r = \{\mathtt{foo}{=}\ 42,\ \mathtt{bar}{=}\ \mathrm{true}\}\ \mathbf{in}\ \cdots$$

- **フィールド取出し**（これも当然）

$$r\,{\scriptstyle\blacksquare}\,\mathtt{foo} \longrightarrow 42$$

- **更新**

$$r\ \mathbf{update}\ \mathtt{bar}\ \mathrm{false} \longrightarrow \{\mathtt{foo}{=}\ 42,\ \mathtt{bar}{=}\ \mathrm{false}\}$$

※ “値を覆い隠している” だけで，破壊的代入ではない

レコードに対する演算

- **拡張** $\{\mathtt{foo}= 42\}\ \mathbf{extend}\ \mathtt{bar}\ \mathrm{true}$

$$\longrightarrow \{\mathtt{foo}= 42,\ \mathtt{bar}= \mathrm{true}\}$$

既に含まれるラベルと被る場合，上書きするか何もしないかでヴァリエーションあり

- **連接** (concatenation)

$$\{\mathtt{foo}= 42,\ \mathtt{bar}= \mathrm{true}\}\ |\&|\ \{\mathtt{foo}= 0,\ \mathtt{baz}= \text{“オッ”}\}$$

$$\longrightarrow \{\mathtt{foo}= 0,\ \mathtt{bar}= \mathrm{true},\ \mathtt{baz}= \text{“オッ”}\}$$

ラベルが重複する場合を許して後者の値を優先するものと許さないものとのヴァリエーションあり

タプルに比べたレコードのありがたみ

- ラベルのおかげで単純にコードのドキュメント性が向上

タプル流

$$(\text{“}\texttt{realDonaldTrump}\text{”},\ \text{“}\texttt{Donald J. Trump}\text{”},\ \mathrm{true},\ \mathrm{true})$$

タプルに比べたレコードのありがたみ

- ラベルのおかげで単純にコードのドキュメント性が向上

タプル流

$$(\text{“}\texttt{realDonaldTrump}\text{”}, \text{“}\texttt{Donald J. Trump}\text{”}, \mathrm{true}, \mathrm{true})$$

レコード流

$$\{\texttt{screen\_name} = \text{“}\texttt{realDonaldTrump}\text{”}, \texttt{name} = \text{“}\texttt{Donald J. Trump}\text{”}, \texttt{is\_official} = \mathrm{true}, \texttt{is\_public} = \mathrm{true}\}$$

タプルに比べたレコードのありがたみ

- “多相性” を持ち，そのうえ “組にして扱う要素” の仕様変更に対してもコードが頑強

組の要素の
仕様変更のたびに
書き換える必要あり

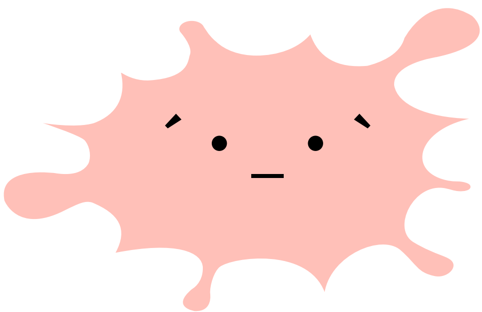
$$
\begin{aligned}
&\mathbf{let}\ \mathit{send\_missile} = \boldsymbol{\lambda}\mathit{account}. \\
&\quad \mathbf{let}\ (\mathit{screen\_name},\ \_,\ \_,\ \_) = \mathit{account}\ \mathbf{in} \\
&\qquad \mathit{send\_reply}\ \mathit{screen\_name}\ \text{“🚀”}
\end{aligned}
$$

$$
\begin{aligned}
&\mathbf{let}\ \mathit{send\_missile} = \boldsymbol{\lambda}\mathit{account}. \\
&\quad \mathit{send\_reply}\ (\mathit{account}{\scriptstyle\blacksquare}\mathtt{screen\_name})\ \text{“🚀”}
\end{aligned}
$$

問題意識

- 型なしだと（当然ながら）取出しなどの際に失敗しうる

$$\{\mathtt{foo}= 42\}.\mathtt{bar} \longrightarrow$$



破滅

適切な型システムを設計して，失敗し得ないプログラムにだけ静的に型がつくようにしたい

ただし，その際に……

問題意識

- できるだけ保守的になりすぎないようにしたい
  (前述の演算のなるべく多くを "自然な範囲で"
  サポートしたい)
- **主要型** (principal type) が存在し,
  かつそれが**型推論**で得られる体系にしたい
- できれば**より計算上効率的な形式へと**
  コンパイルできるようにしたい

[Ohori 1995] 概要

- **SML#** の型システムの基礎づけ
- 双対である多相ヴァリアントもサポート（今回は扱わない）
- 一般的な大きい入力に対しても十分高速に型推論が動作

✳ レコードの**拡張や連接はサポートしない**

✓ **型推論可能**

✓ **コンパイルによる効率化が可能**

✓ （主観ながら）**推論される型が比較的わかりやすい**

[Osinski 2006] 概要

- 博士論文（同著者唯一の publication らしい？）

✓ **前述のすべての演算をサポート**

✓ **型推論可能**

✓ **コンパイルによる効率化が可能**

▲ （主観ながら）推論される型が複雑でわかりにくい？

▲ 実装がまだ無いようで，**実時間の動作に堪えるか不明**
（自分の手で実装して確かめるつもりでしたが間に合わず）

構成

- 背景
- **おさらい：Let多相の型推論**
- [Ohori 1995]
- [Osinski 2006]
- まとめ

項と型

型註釈なしの項を主として使用

- **値** $v \ ::= \ c \ \mid \ \boldsymbol{\lambda} x.\ e$
- **項** $e \ ::= \ v \ \mid \ x \ \mid \ e\ e \ \mid \ \mathbf{let}\ x = e\ \mathbf{in}\ e$

実用上は不動点 $(\mathbf{fix}\ x.\ \boldsymbol{\lambda} x.\ e)$ や条件分岐 $\mathbf{if}\ e\ \mathbf{then}\ e\ \mathbf{else}\ e$ も必要だが，簡単のため今回は含めない

- **単相型** $\tau \ ::= \ b \ \mid \ \alpha \ \mid \ \tau \to \tau$
- **多相型** $\sigma \ ::= \ \tau \ \mid \ \forall \alpha.\ \sigma$

型つけ規則

$$\boxed{\Gamma \vdash e : \tau}$$

型変数の束縛出現の
"インスタンス化"

$$\frac{\left(\sigma :\equiv \Gamma(x), \quad \sigma \geqslant \tau\right)}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma[x \mapsto \tau'] \vdash e : \tau}{\Gamma \vdash (\lambda x.\ e) : \tau' \to \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

型変数の量化

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \left(\sigma :\equiv \mathrm{Gen}(\Gamma, \tau)\right) \qquad \Gamma[x \mapsto \sigma] \vdash e_2 : \tau_2}{\Gamma \vdash \mathbf{let}\ x = e_1\ \mathbf{in}\ e_2 : \tau_2}$$

型つけ規則の読み方

1. 純粋に3項関係の帰納的定義として

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

“仮定側”

“帰結側”

2. アルゴリズムとして： “下から読む”

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

型つけ規則の読み方

1. 純粋に3項関係の帰納的定義として

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

“仮定側”

“帰結側”

2. アルゴリズムとして：“下から読む”

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

型つけ規則の読み方

1. 純粋に3項関係の帰納的定義として

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

“仮定側”

“帰結側”

2. アルゴリズムとして： “下から読む”

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

型つけ規則の読み方

1. 純粋に3項関係の帰納的定義として

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

“仮定側”

“帰結側”

2. アルゴリズムとして：“下から読む”

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

型つけ規則の読み方

1. 純粋に3項関係の帰納的定義として

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

“仮定側”

“帰結側”

2. アルゴリズムとして：“下から読む”

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

型つけ規則の読み方

1. 純粋に3項関係の帰納的定義として

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

“仮定側”

“帰結側”

2. アルゴリズムとして：“下から読む”

$$\frac{\Gamma \vdash e_1 : \tau' \to \tau \qquad \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1\ e_2 : \tau}$$

（※不正確，あくまで直観）

型つけ規則から型推論アルゴリズムへ

型推論アルゴリズムとしては，より正確には**型変数**と
代入（型変数を実際の型へ関連づける部分写像）を使う

つまるところ型推論とは「未知の型を型変数で置いておき，
項を走査して種々の制約をもとに代入を育てていく」処理

型が未知のところを
型変数で置いて上る

制約の解消結果として
下りてきた代入

$$\frac{(\alpha\ \ \text{fresh}) \qquad \Gamma[x \mapsto \alpha] \vdash e : \tau \downarrow \theta}{\Gamma \vdash (\boldsymbol{\lambda} x.\ e) : (\theta\alpha \to \tau) \downarrow \theta}$$

下りてきた代入を
使って実際の型を得る

型つけ規則から型推論アルゴリズムへ

単一化：

型の等式制約を解いて
代入にする処理,
型推論の根幹

$$\frac{\begin{array}{c}\Gamma \vdash e_1 : \tau_1 \downarrow \theta_1 \qquad \theta_1 \Gamma \vdash e_2 : \tau_2 \downarrow \theta_2 \\ \alpha \ \text{fresh} \qquad \theta' := \mathrm{Unify}\Big(\big\{\theta_2 \tau_1 \stackrel{?}{=} \tau_2 \to \alpha\big\}\Big),\end{array}}{\Gamma \vdash e_1 \ e_2 : \theta' \alpha \downarrow \theta' \circ \theta_2 \circ \theta_1}$$

単一化

Unify(E) が呼ばれたら, $\bigl(E, \varnothing\bigr)$ から以下の書き換えを始め,
$\bigl(\varnothing, \theta\bigr)$ の形に到達したらその θ を返す
途中で書き換えられなくなったら充足不可能（型エラー相当）

$$\bigl(E \uplus \{\tau \stackrel{?}{=} \tau\}, \theta\bigr) \longrightarrow \bigl(E, \theta\bigr)$$

$$\bigl(E \uplus \{\tau \stackrel{?}{=} \alpha\}\bigr) \longrightarrow \bigl([\tau/\alpha]E, \{\alpha \mapsto \tau\} \circ \theta\bigr)$$

$$\bigl(\text{if } \alpha \notin \mathrm{FTV}(\tau)\bigr)$$

$$\bigl(E \uplus \bigl\{\tau_1 \to \tau_2 \stackrel{?}{=} \tau_1' \to \tau_2'\bigr\}, \theta\bigr) \longrightarrow \bigl(E \cup \bigl\{\tau_1 \stackrel{?}{=} \tau_1', \tau_2 \stackrel{?}{=} \tau_2'\bigr\}, \theta\bigr)$$

構成

- 背景
- おさらい：Let多相の型推論
- **[Ohori 1995]**
- [Osinski 2006]
- まとめ

機能面の概要（再掲）

- **SML#** の型システムの基礎づけ
- 双対である多相ヴァリアントもサポート（今回は扱わない）
- 一般的な入力に対して十分高速に型推論が動作

✴ レコードの**拡張や連接はサポートしない**

✓ **型推論可能**

✓ **コンパイルによる効率化が可能**

✓ （やや主観ながら）**推論される型が比較的わかりやすい**

理論面の概要

- 型システム：
  - **有界量化**による多相
  - 型の間の部分型関係　<:　は陽には扱わず,
    種 (kind) の形で表現する
- 型推論：
  - Let多相の型推論で使われる通常の単一化を
    “種の制約解消と並行して行う単一化” へと変更して実現

項の定義の拡張

- 値　$v \ ::= \ c \ \mid \ \boldsymbol{\lambda} x.\ e \ \mid \ \{\ell = v,\ \ldots,\ \ell = v\}$
- 項　$e \ ::= \ v \ \mid \ x \ \mid \ e\ e \ \mid \ \textbf{let}\ x = e\ \textbf{in}\ e$
  $\quad \mid \ \{\ell = e,\ \ldots,\ \ell = e\} \ \mid \ e \,\blacksquare\, \ell \ \mid \ e\ \textbf{update}\ \ell\ e$

フィールド取り出し

フィールド更新

型つけのための考察

以下の函数にどんな型をつければよさそうか？

$$\mathbf{let}\ \mathit{send\_missile} = \boldsymbol{\lambda}\mathit{account}.$$
$$\quad \mathit{send\_reply}\ (\mathit{account}.\mathtt{screen\_name})\ \text{“🚀”}$$

引数 $\mathit{account}$ は，$\mathtt{screen\_name}$ のフィールドが文字列値として使われているのみ

$\{\mathtt{screen\_name} =$ (文字列値)$, \ldots\}$ **の形の値は全部受け取ってよい**，これを反映した型をつける

型と種：2階の体系

- 単相型 $\tau \ ::= \ b \mid \alpha \mid \tau \to \tau \mid \{\ell : \tau, \ \ldots, \ \ell : \tau\}$
- 多相型 $\sigma \ ::= \ \tau \mid \forall \alpha :: \kappa . \sigma$

種による "制約"
つきの量化

- 種 $\kappa \ ::= \ \mathsf{U} \mid \{\!\{\ell : \tau, \ \ldots, \ \ell : \tau\}\!\}$

$\forall \alpha :: \mathsf{U} . \sigma$ が従来の $\forall \alpha . \sigma$ に相当（つまり "制約" なし）

レコード種 $\{\!\{\ell_1 : \tau_1, \ \ldots, \ \ell_n : \tau_n\}\!\}$ は直観的には
「**少なくとも各** $\ell_i : \tau_i$ **は全部含むレコード型全体**」を指す

レコード種の直観

$\boxed{例}$

$$\{\mathtt{foo} : \mathtt{int}\} :: \{\!\{\mathtt{foo} : \mathtt{int}\}\!\}$$

$$\{\mathtt{foo} : \mathtt{int},\ \mathtt{bar} : \mathtt{bool}\} :: \{\!\{\mathtt{foo} : \mathtt{int}\}\!\}$$

$$\{\mathtt{foo} : \mathtt{int},\ \mathtt{bar} : \mathtt{bool},\ \mathtt{baz} : \mathtt{string}\} :: \{\!\{\mathtt{foo} : \mathtt{int}\}\!\}$$

$\boxed{例}$

$$\{\mathtt{foo} : \mathtt{int},\ \mathtt{bar} : \mathtt{bool}\} :: \{\!\{\mathtt{foo} : \mathtt{int},\ \mathtt{bar} : \mathtt{bool}\}\!\}$$

$$\begin{aligned}\{\mathtt{foo} : \mathtt{int},\ &\mathtt{bar} : \mathtt{bool},\\ &\mathtt{baz} : \mathtt{string}\} :: \{\!\{\mathtt{foo} : \mathtt{int},\ \mathtt{bar} : \mathtt{bool}\}\!\}\end{aligned}$$

型つけの例

$$\textbf{let}\ \mathit{send\_missile} = \boldsymbol{\lambda}\mathit{account}.$$
$$\quad \mathit{send\_reply}\ (\mathit{account}_{\blacksquare}\texttt{screen\_name})\ \text{“🚀”}$$

引数 $\mathit{account}$ の型は $\tau :: \{\!\{\texttt{screen\_name} : \texttt{string}\}\!\}$

なる任意の τ でよい

$\mathit{send\_missile}$ につける型は

$$\forall\alpha :: \{\!\{\texttt{screen\_name} : \texttt{string}\}\!\}.\ \alpha \to \texttt{unit}$$

量化に関する若干の注意

通常のLet多相と違い，**量化の順番も意味を持つ**

$$\mathbf{let}\ get\_name = \boldsymbol{\lambda} x.\ x \,\blacksquare\, \texttt{name}$$

get_name につく型は

$\forall\beta :: \mathsf{U}.\,\forall\alpha :: \{\!\{\texttt{name} : \beta\}\!\}.\ \alpha \to \beta$ であって

$\forall\alpha :: \{\!\{\texttt{name} : \beta\}\!\}.\,\forall\beta :: \mathsf{U}.\ \alpha \to \beta$ ではない

束縛されていない出現

型つけ規則のための種環境の導入

種環境 K ：型変数に種を関連づける部分写像

$$K : \alpha \longmapsto \kappa$$

cf. 型環境 Γ ：変数に型を関連づける部分写像

$$\begin{tikzcd}
\textbf{項} & \textbf{型} & \textbf{種} \\
e & \sigma & \kappa \\
x \arrow[ur, mapsto, "\Gamma"'] & \alpha \arrow[ur, mapsto, "K"'] &
\end{tikzcd}$$

型つけ規則（抜粋1）

$$\boxed{K \mid \Gamma \vdash e : \tau}$$

“種環境 K の下で，τ は σ のインスタンスである”
（形式化は複雑なので割愛）

$$\frac{\left(\sigma \equiv \Gamma(x), \qquad K \vdash \sigma \geqslant \tau\right)}{K \mid \Gamma \vdash x : \tau}$$

$$\frac{K \mid \Gamma \vdash e_i : \tau_i \quad (\text{for each } i)}{K \mid \Gamma \vdash \{\ell_1 = e_1, \ \ldots, \ \ell_n = e_n\} : \{\ell_1 : \tau_1, \ \ldots, \ \ell_n : \tau_n\}}$$

型つけ規則（抜粋2）

$$\boxed{K \mid \Gamma \vdash e : \tau}$$

“種環境 K の下で
τ に種 $\{\!\{\ell : \tau'\}\!\}$ がつく”

$$\frac{K \mid \Gamma \vdash e : \tau \qquad K \vdash \tau :: \{\!\{\ell : \tau'\}\!\}}{K \mid \Gamma \vdash (e \,\blacksquare\, \ell) : \tau'}$$

$$\frac{K \mid \Gamma \vdash e : \tau \qquad K \mid \Gamma \vdash e' : \tau' \qquad K \vdash \tau :: \{\!\{\ell : \tau'\}\!\}}{K \mid \Gamma \vdash (e \ \mathbf{update}\ \ell\ e') : \tau}$$

型つけ規則（抜粋2）

$$\boxed{K \mid \Gamma \vdash e : \tau}$$

"種環境 K の下で
τ に種 $\{\!\{\ell : \tau'\}\!\}$ がつく"

$$\frac{K \mid \Gamma \vdash e : \tau \qquad K \vdash \tau :: \{\!\{\ell : \tau'\}\!\}}{K \mid \Gamma \vdash (e \,\blacksquare\, \ell) : \tau'}$$

$$K \vdash \{R\} :: \{\!\{R'\}\!\}$$
$$:\Longleftrightarrow \quad R \supseteq R'$$

$$K \vdash \alpha :: \{\!\{R'\}\!\}$$
$$:\Longleftrightarrow \quad R \supseteq R'$$
$$\big(\{\!\{R\}\!\} :\equiv K(\alpha)\big)$$

$$\frac{K \mid \Gamma \vdash e : \tau \qquad K \mid \Gamma \vdash e' : \tau' \qquad K \vdash \tau :: \{\!\{\ell : \tau'\}\!\}}{K \mid \Gamma \vdash (e \ \mathbf{update}\ \ell\ e') : \tau}$$

型安全性

値呼び戦略では，計算が止まるときは必ず値になっている：

定理

$\varnothing \mid \varnothing \vdash e : \tau$ かつ $e \Downarrow e'$ ならば，e' は τ 型の値である．

値呼び戦略で
簡約できなくなるまで
簡約し続ける

証明は**論理関係** (logical relation) の手法による

型つけ規則から型推論アルゴリズムへ

この体系の型推論では，代入と共に種環境も降りてくる
（つまり**種環境も型変数に対する制約の解消結果**）

$$\frac{(\alpha\ \text{fresh}) \qquad \boxed{K[\alpha \mapsto \mathsf{U}]} \mid \mathit{\Gamma}[x \mapsto \alpha] \vdash e : \tau \downarrow \boxed{K'} \mid \theta}{\boxed{K} \mid \mathit{\Gamma} \vdash (\boldsymbol{\lambda} x.\ e) : (\theta\alpha \to \tau) \downarrow \boxed{K'} \mid \theta}$$

$$\frac{\boxed{K} \mid \mathit{\Gamma} \vdash e_1 : \tau_1 \downarrow \boxed{K_1} \mid \theta_1 \qquad (\beta_1, \beta\ \text{fresh}) \qquad (\boxed{K'}, \theta') := \mathrm{Unify}\Big(\boxed{K_1[\beta \mapsto \mathsf{U}][\beta_1 \mapsto \{\!\{\ell : \beta\}\!\}]}, \{\tau_1 \overset{?}{=} \beta_1\}\Big)}{\boxed{K} \mid \mathit{\Gamma} \vdash (e_1 \,\blacksquare\, \ell) : \theta'\beta \downarrow \boxed{K'} \mid \theta' \circ \theta_1}$$

種を加味した単一化（抜粋1）

$$\left(K, E \uplus \left\{\alpha_1 \overset{?}{=} \alpha_2\right\}, \theta\right) \longrightarrow (K', E', \theta')$$

where

$\{\!\{R_1\}\!\} :\equiv K\big(\alpha_1\big)$

$\{\!\{R_2\}\!\} :\equiv K\big(\alpha_2\big)$

$K' := \big([\alpha_2/\alpha_1]K\big)\big[\alpha_1 \mapsto [\alpha_2/\alpha_1]\{\!\{R_1 \cup R_2\}\!\}\big]$

$E' := [\alpha_2/\alpha_1]\Big(E \cup \big\{R_1(\ell) \overset{?}{=} R_2(\ell) \;\big|\; \ell \in \mathrm{dom}\, R_1 \cap \mathrm{dom}\, R_2\big\}\Big)$

$\theta' := \big([\alpha_2/\alpha_1]\theta\big)\big[\alpha_1 \mapsto \alpha_2\big]$

両型変数の
現在の種

種を "合併"させる
（重複する部分は一方だけ）

ラベルが重複する部分は
同一の型でないといけない

種を加味した単一化（抜粋2）

$$\big(K, E \uplus \{\alpha \overset{?}{=} \{R\}\}, \theta\big) \longrightarrow (K', E', \theta')$$

$$
\begin{aligned}
&\text{if } \operatorname{dom} R' \subseteq \operatorname{dom} R \text{ and } \alpha \notin \mathrm{FTV}\big(\{R\}\big)\\
&\text{where}\\
&\{\!\{R'\}\!\} :\equiv K(\alpha)\\
&K' := [\{R\}/\alpha]K\\
&E' := [\{R\}/\alpha]\Big(E \cup \Big\{R'(\ell) \overset{?}{=} R(\ell) \;\Big|\; \ell \in \operatorname{dom} R'\Big\}\Big)\\
&\theta' := \big([\{R\}/\alpha]\theta\big)\big[\alpha \mapsto \{R\}\big]
\end{aligned}
$$

型変数の
現在の種

ラベルが重複する部分は
同一の型でないといけない

構成

- 背景
- おさらい：Let多相の型推論
- [Ohori 1995]
- **[Osinski 2006]**
- まとめ

機能面の概要（再掲）

- 博士論文（同著者唯一の publication らしい？）

✓ **前述のすべての演算をサポート**

✓ **型推論可能**

✓ **コンパイルによる効率化が可能**

▲ （主観ながら）推論される型が複雑でわかりにくい？

▲ 実装がまだ無いようで，**実時間の動作に堪えるか不明**
（自分の手で実装して確かめるつもりでしたが間に合わず）

理論面の概要

- **質化型** (qualified type) を土台としている：

$$\forall X.\, P \Rightarrow \tau$$

型変数の集合

型変数が満たす述語の集合

- レコードやラベル集合の排反性や等価性を述語とし，
  単一化に似た処理で述語集合の制約を解消

- アルゴリズムの正当性の証明するために
  レコードの**多重集合による意味論**を定義しているっぽい

構成

- 背景
- おさらい：Let多相の型推論
- [Ohori 1995]
- [Osinski 2006]
- まとめ

まとめ

- レコード計算の型つけ，たのしい！
- [Ohori 1995]
  - SML# の型システムの基礎
  - 連接をサポートしないかわりに効率的に型推論が可能,
    加えて効率的コードへとコンパイルできる
  - レコード計算のうちでは比較的型がわかりやすい？
- [Osinski 2006]
  - 機能面では “万能” らしい（証明未読）
  - 実装しないことには動作時間や使用感がわからない

蛇足

- 実は諏訪が実装した Macrodown という言語にも [Ohori 1995] の型システムが取り込まれています

https://github.com/gfngfn/Macrodown