Types and Programming Languages
Chapter 29-30 の途中
(型輪講 :6/29)

米澤研究室
M1 佐藤秀明

概要

- 型についての関数を導入 (cf. 項についての関数)
  - Type operators
  - Kinding
- 型システムの拡張
  - F_ω: System F + type operators
  - F_ω の性質 (type equivalence and reduction)

Type Operator

- ex. ペアの abbreviation

$$Pair\ Y\ Z = \forall\ X.(Y \to Z \to X) \to X\ ;$$

- 型を受け取って型を返す関数とみなす

$$\begin{aligned} Pair &= \lambda\ Y.\lambda\ Z.\forall\ X.(Y \to Z \to X) \to X\ ; \\ Pair\ S\ T &= (\lambda\ Y.\lambda\ Z.\forall\ X.(Y \to Z \to X) \to X)\ S\ T \\ &= \forall\ X.(S \to T \to X) \to X\ ; \end{aligned}$$

- 型に関するラムダ抽象と適用を定義
  - 項に用いられていた notation を流用

Definitional Equivalence(1)

- Type operator は同じ型の多様な表現を可能にする
- ex. $Id = \lambda X.X$

$$Nat \to Bool \qquad Nat \to Id\ Bool \qquad Id\ Nat \to Id\ Bool$$
$$Id\ Nat\ Bool \qquad Id\,(Nat \to Bool) \qquad Id\,(Id\,(Id\ Nat \to Bool))$$

  - 上は全て同じ型を表す
- これらの型が等価であることを Typing の際に認識したい

Definitional Equivalence(2)

- 同じ型どうしの間に成り立つ関係 (≡) を定義
  - 詳細は figure 29-1 の”Type equivalence”に
  - $(\lambda X::K_{11}.T_{12})T_2 \equiv [X \to T_2]T_{12} \qquad (Q-APPABS)$
    - “::”については後述
- (≡) が成り立つなら typing の推論を進める
  - $$\frac{\Gamma \vdash t : S \quad S \equiv T}{\Gamma \vdash t : T} \qquad (T-EQ)$$

Kinds(1)

- Type operator が無意味な型表現を生成することがある
  - ex. (Bool Nat)
- 対策：とりうる引数の数によって型表現を分類
  - 言わば「型の型」

Kinds(2)

- Kind の定義

$$\begin{array}{lll} K & ::= & \textit{kinds} \\ & * & \textit{kind of proper types} \\ & K \Rightarrow K & \textit{kind of operators} \end{array}$$

- Proper type: 項を実際に分類する型
- Kinding の例
  - * ・・・ Bool, Nat→Bool, ∀X.X→X, (λX.X→X)Nat
  - *⇒* ・・・ λX.X→X, Pair Nat
  - *⇒*⇒* ・・・ λX.λY.X→Y, Pair
  - (*⇒*)⇒* ・・・ λX.X Bool

Kinds(3)

- Ill-formed な型表現は kinding できない
  - Kinding の rule は figure 29-1 の”Kinding”に
- ラムダ抽象の束縛変数に kind を添える
  - well-kindedness のチェックを簡単にするため
  - ex. Pair

$$Pair = \lambda Y :: * .\ \ \lambda Z :: * .\ \ \forall X.(Y \to Z \to X) \to X\ ;$$

実際の言語における Type Operator

- 現実の言語については型の型までを考えれば十分
  - 型の型の型を扱う研究もある
- ML の type operator 機能
  - ex. type 'a tyop = Tyoptag of ('a -> 'a)
  - 型表現を多相的に定義できるのは variant のみ
    - 必ず tag がつくので型チェックが楽

λ_ω(1)

- F_ω の前にまず λ_ω
  - λ_ω: 単純型付き + type operator
  - Syntax と rule は figure 29-1 に

$\lambda_\omega(2)$

- Kinding の導入
  - $\Gamma \vdash T :: K$ ・・・ 文脈Γにおいて型 T は kind K を持つ
- 型の well-formedness の確認

$$\frac{\Gamma \vdash T_1 :: * \qquad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1 . t_2 : T_1 \to T_2} \qquad (T-ABS)$$

  - term の束縛変数は proper type をもつ
- Definitional equivalence

$$\frac{\Gamma \vdash t : S \qquad S \equiv T \qquad \Gamma \vdash T :: *}{\Gamma \vdash t : T} \qquad (T-EQ)$$

  - 型付け可能な項は kinding 可能な型をもつ

F_ω

- F_ω : λ_ω に universal types を追加
  - Syntax と rule は figure 30-1 に
- (参考)F_ω に existential types を追加
  - Syntax と rule は figure 30-2 に

F_ω with Existential Types の例 (1)

- PairSig = {∃Pair::*⇒*⇒*,

  {pair: ∀X. ∀Y. X→Y→(Pair X Y)

  fst: ∀X. ∀Y. (Pair X Y)→X,

  snd: ∀X. ∀Y. (Pair X Y)→Y}};

F_ω with Existential Types の例 (2)

- pairADT =

  {*λX. λY. ∀R. (X→Y→R) → R,

  {pair = λX. λY. λx:X. λy:Y.

  λR. λp:X→Y→R. p x y,

  fst = λX. λY. λp: ∀R. (X→Y→R)→R.

  p[X] (λx:X. λy:Y. x),

  snd = λX. λY. λp: ∀R. (X→Y→R)→R.

  p[Y] (λx:X. λy:Y. y)}}

  as PairSig;

F_ω with Existential Types の例 (3)

- let {Pair, pair} = pairADT in

  pair.fst[Nat][Bool] (pair.pair[Nat][Bool] 5 true);

  - 5 : Nat

F_ω の性質

- 以降、F_ω の性質について述べる
  - Existential type は考えない

Basic Properties(1)

- Lemma 30.3.1 [Strengthening]

$$If\ \Gamma, x:S, \Delta \vdash T :: K, then\ \Gamma, \Delta \vdash T :: K$$

- Proof: kinding の関係は term の変数束縛関係に依存しない。

Basic Properties(2)

- Lemma 30.3.2 [Permutation and Weakening]
  - 文脈 Δ を、文脈 Γ,Σ の well-formed な permutation とする。このとき
  - $If\ \Gamma \vdash T :: K\ , then\ \Delta \vdash T :: K$
  - $If\ \Gamma \vdash t : T\ , then\ \Delta \vdash t : T$
- Proof: 導出に関する帰納法。

Basic Properties(3)

- Lemma 30.3.3 [Term Substitution]
  - $If\ \Gamma, x:S, \Delta \vdash t:T \wedge \Gamma \vdash s:S,$
    $then\ \Gamma, \Delta \vdash [x \to s]t:T$
- Proof: 導出に関する帰納法。
  - T-ABS, T-EQ に Strengthening を用いる
  - T-ABS, T-TABS に Permutation を用いる
  - T-VAR に Weakening を用いる

Basic Properties(4)

- Lemma 30.3.4 [Type Substitution]
  - 1. $If\ \Gamma, Y::J, \Delta \vdash T::K \wedge \Gamma \vdash S::J,$
    $then\ \Gamma, [Y \to S]\Delta \vdash [Y \to S]T::K$
  - 2. $If\ T \equiv U, then [Y \to S]T \equiv [Y \to S]U$
  - 3. $If\ \Gamma, Y::J, \Delta \vdash t:T \wedge \Gamma \vdash S::J,$
    $then\ \Gamma, [Y \to S]\Delta \vdash [Y \to S]t:[Y \to S]T$
- Proof: 導出に関する帰納法。
  - K-TVAR と T-VAR に weakening を使う
  - Q-APPABS には、$[X \to [Y \to S]T_2]([Y \to S]T_{12})$ と $[Y \to S]([X \to T_2]T_{12})$ が同じであることを使う

Parallel reduction(⇛)

- definitional equivalence に似た概念
  - 対称律 (SYMM) がない
    - reduction が後戻りしないことに対応
  - 推移律 (TRANS) がない
  - APPABS は β 簡約と同時に各要素も reduce できる
    - Diamond property(Lemma 30.3.8) を導くため
- 詳細は Figure 30-3 に

Type Equivalence and Reduction(1)

- Lemma 30.3.5(反射的推移的閉包)
  - $S \equiv T \; iff \; S \Leftrightarrow^{*} T$
- Proof:
  - ⇐ : 明らか。
  - ⇒ : S≡T の導出は Q-TRANS で繋がれた鎖 $S=S_1 \equiv S_2 \equiv \cdots \equiv S_n = T$ に置き換えられる。鎖中の各導出 $S_i \equiv S_{i+1}$ が Q-SYMM を最後だけ使うように、鎖を構成できる。

Type Equivalence and Reduction(2)

- Lemma 30.3.6
  - $If\ S \Rrightarrow S',\ then [Y \to S]T \Rrightarrow [Y \to S']T\ \ for\ any\ type\ T$
- Proof: T の構造に関する帰納法。

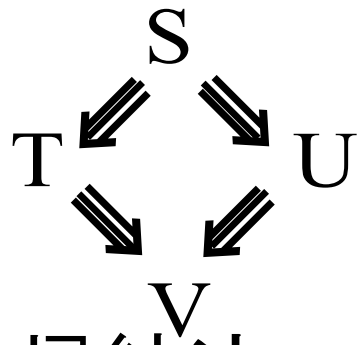
Type Equivalence and Reduction(3)

- Lemma 30.3.7
  - $If\ S \Rrightarrow S' \wedge T \Rrightarrow T', then [Y \to S]T \Rrightarrow [Y \to S']T'$
- Proof: T⇛T' の導出に関する帰納法。
  - QR-REFL には Lemma 30.3.6 を使う
  - QR-APPABS は次のようにおくとよい

$$T = (\lambda X :: K_{11}.\, T_{12})\, T_2, \quad T' = [X \to T_2']\, T_{12}'$$
$$with \quad T_{12} \Rrightarrow T_{12}', \quad T_2 \Rrightarrow T_2'$$

Type Equivalence and Reduction(4)

- Lemma 30.3.8 [Single-step Diamond Property of Substitution]
  - $If\ S \Rrightarrow T \wedge S \Rrightarrow U\ ,then\ there\ is\ some\ type\ V\ such\ that\ T \Rrightarrow V \wedge U \Rrightarrow V$



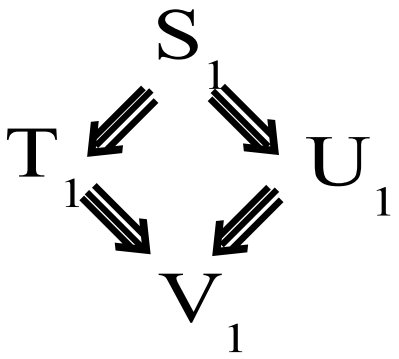
- Proof: 組 (S ⇛ T, S⇛U) の導出に関する帰納法。
  - 両導出が同じ rule を用いている場合 (ex. QR-ARROW)

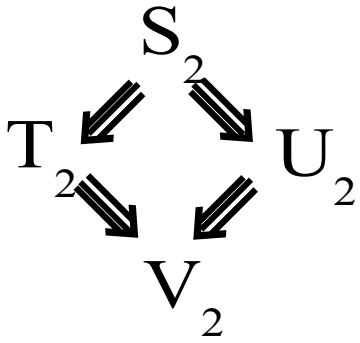
$$\frac{S_1 \Rrightarrow T_1 \qquad S_2 \Rrightarrow T_2}{S_1 \to S_2 \Rrightarrow T_1 \to T_2} \qquad \frac{S_1 \Rrightarrow U_1 \qquad S_2 \Rrightarrow U_2}{S_1 \to S_2 \Rrightarrow U_1 \to U_2}$$

Type Equivalence and Reduction(5)

- Proof(続き)

  帰納法の仮定より





  よって

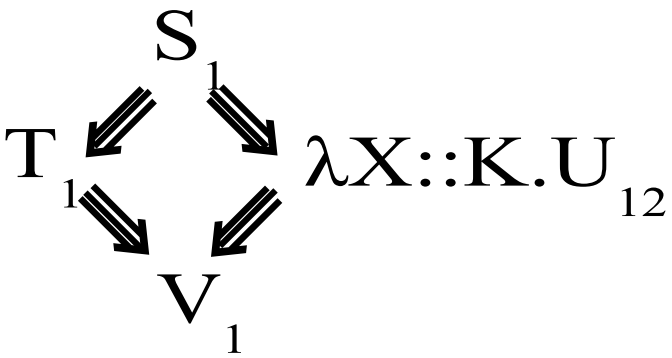
$$\frac{T_1 \Rightarrow V_1 \qquad T_2 \Rightarrow V_2}{T_1 \to T_2 \Rightarrow V_1 \to V_2} \qquad \frac{U_1 \Rightarrow V_1 \qquad U_2 \Rightarrow V_2}{U_1 \to U_2 \Rightarrow V_1 \to V_2}$$

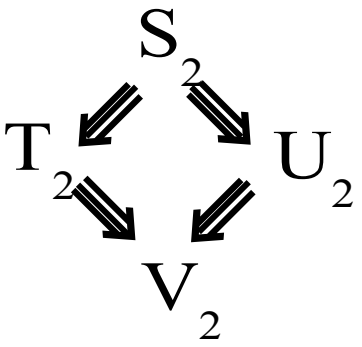
Type Equivalence and Reduction(6)

- Proof(続き)
  - 一方が QR-APP で他方が QR-APPABS の場合

$$\frac{S_1 \Rightarrow T_1 \qquad S_2 \Rightarrow T_2}{S_1 S_2 \Rightarrow T_1 T_2} \qquad \frac{S_{12} \Rightarrow U_{12} \qquad S_2 \Rightarrow U_2}{(\lambda X::K.S_{12})S_2 \Rightarrow [X \rightarrow U_2]U_{12}}$$

S_1 はラムダ抽象だから、帰納法の仮定より



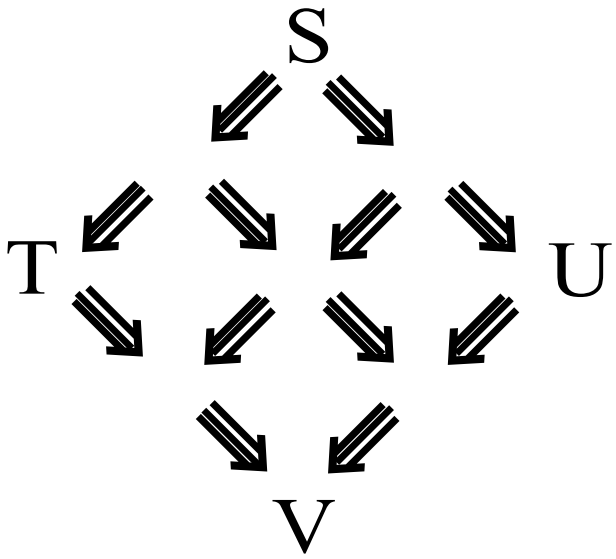


$[X \rightarrow U_2]U_{12} = (\lambda X::K.U_{12})U_2$ だから

$$\frac{T_1 \Rightarrow V_1 \qquad T_2 \Rightarrow V_2}{T_1 T_2 \Rightarrow V_1 V_2} \qquad \frac{\lambda X::K.U_{12} \Rightarrow V_1 \qquad U_2 \Rightarrow V_2}{(\lambda X::K.U_{12})U_2 \Rightarrow V_1 V_2}$$

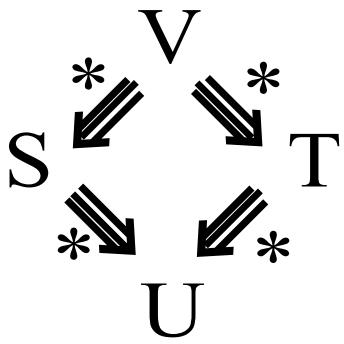
Type Equivalence and Reduction(7)

- Lemma 30.3.9 [Confluence, Church-Rosser Property]
  - $If\ S \Rrightarrow^* T \wedge S \Rrightarrow^* U\ ,$
    $then\ there\ is\ some\ type\ V\ \ such\ that\ T \Rrightarrow^* V \wedge U \Rrightarrow^* V$
- Proof: Lemma 30.3.8 の diamond を格子状に組み合わせればよい。



Type Equivalence and Reduction(8)

- Proposition 30.3.10
  - $If\ S \Leftrightarrow^* T\ ,$
    $then\ there\ is\ some\ U\ \ such\ that\ S \Rightarrow^* U \wedge T \Rightarrow^* U$
- Proof: S $\Leftrightarrow^*$ T より、ある型 V が存在して、V $\Rightarrow^*$ S かつ V $\Rightarrow^*$ T が成り立つ。ここで Confluence を用いる。



- Definitionally equivalent な types は共通の reduct をもつ

次回予告

- Chapter 30 の続き
  - F_{ω} の性質 (preservation, progress, decidability)
  - 型システムの hierarchy
  - Dependent types