プログラム言語論

亀山幸義

筑波大学 情報科学類

No. 10b: 型と論理

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.
- 論理式 $(A \wedge B) \supset (B \wedge A)$ は証明可能.
- 型 $(A \times B) \to (B \times A)$ は要素を持つ
  - 引数の左右を逆にしたものを返す関数.

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.
- 論理式 $(A \wedge B) \supset (B \wedge A)$ は証明可能.
- 型 $(A \times B) \to (B \times A)$ は要素を持つ
  - 引数の左右を逆にしたものを返す関数.
- 論理式 $(A \supset B) \supset (B \supset C) \supset (A \supset C)$ は証明可能.
- 型 $(A \to B) \to (B \to C) \to (A \to C)$ は要素を持つ.,
  - 関数を 2 つもらって，それらの合成関数を返す高階関数.

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.
- 論理式 $(A \wedge B) \supset (B \wedge A)$ は証明可能.
- 型 $(A \times B) \to (B \times A)$ は要素を持つ
  - 引数の左右を逆にしたものを返す関数.
- 論理式 $(A \supset B) \supset (B \supset C) \supset (A \supset C)$ は証明可能.
- 型 $(A \to B) \to (B \to C) \to (A \to C)$ は要素を持つ.,
  - 関数を 2 つもらって，それらの合成関数を返す高階関数.

Curry-Howard の対応は，プログラミング言語の型システムの設計に多大な影響を与えてきた.

単純型付きラムダ計算と直観主義命題論理 (1)

命題論理とは？

- 命題: 原子命題 $(P, Q, \cdots)$ を論理結合子 $(\wedge, \vee, \Rightarrow, \ldots)$ で結合。
- 推論規則の例:

$$\frac{\Delta \vdash A_1 \quad \Delta \vdash A_2}{\Delta \vdash A_1 \wedge A_2} \qquad \frac{\Delta \vdash A_1 \wedge A_2}{\Delta \vdash A_i} \qquad \frac{\Delta, A_1 \vdash A_2}{\Delta \vdash A_1 \Rightarrow A_2} \qquad \frac{\Delta \vdash A_1 \Rightarrow A_2 \quad \Delta \vdash A_1}{\Delta \vdash A_2}$$

  ただし、Δ は、有限個の命題の列。
- 推論規則を有限回適用して推論できた命題が証明可能

直観主義論理 (intuitionistic logic):

- $A \vee (\neg A)$ や $\neg(\neg A) \Rightarrow A$ が成立しない論理。
- これら以外は「普通」の論理 (古典論理) と一致。

単純型付きラムダ計算と直観主義命題論理 (2)

定理 1. 単純型付きラムダ計算 (STLC) で $\Gamma \vdash M : T$ が導出できれば、直観主義命題論理 (IPL) で $\overline{\Gamma} \vdash \overline{T}$ が証明できる。

ここで、$\overline{X}$ は、X を以下の表に従って左から右に変換したもの。ただし、基本型と原子命題は同じ記号を使う。

型構成子	論理結合子
$\times$	$\wedge$
$+$	$\vee$
$\rightarrow$	$\Rightarrow$ (ならば)

例: STLC で $\vdash \lambda x.\, \lambda y.\, x : T_1 \rightarrow (T_2 \rightarrow T_1)$ が導出できるので、IPL で、$\vdash T_1 \Rightarrow (T_2 \Rightarrow T_1)$ が証明できる。

単純型付きラムダ計算と直観主義命題論理 (3)

定理 2. IPL で $\Delta \vdash A$ が証明できれば、あるラムダ項 M に対して、STLC で $\underline{\Delta} \vdash M : \underline{A}$ が導出できる。

ここで、$\underline{Y}$ は、Y を以下の表に従って右から左に変換したもの。

型構成子	論理結合子
$\times$	$\wedge$
$+$	$\vee$
$\rightarrow$	$\Rightarrow$ (ならば)

例: IPL で $\vdash T_1 \Rightarrow ((T_1 \Rightarrow T_2) \Rightarrow T_2)$ が証明できるので、STLC で $\vdash M : T_1 \rightarrow ((T_1 \rightarrow T_2) \rightarrow T_2)$ が導出できるラムダ項 M が存在する。
($M = \lambda x.\ \lambda y.\ y\, x$ とすればよい。)

単純型付きラムダ計算と直観主義命題論理 (4)

事実: 前述の定理 1 および定理 2 の「STLC での導出」と「IPL での証明」は、同じサイズで同じ構造のものとすることができる。(以下に例をのべる。)

論理式 $A \Rightarrow (B \Rightarrow A)$ の証明:

$$\dfrac{\dfrac{\dfrac{}{A, B \vdash A}}{A \vdash B \Rightarrow A}}{\vdash A \Rightarrow (B \Rightarrow A)}$$

型判断 $\vdash \lambda x.\ \lambda y.\ x : A \to (B \to A)$ の導出:

$$\dfrac{\dfrac{\dfrac{}{x : A,\ y : B \vdash x : A}}{x : A \vdash \lambda y.\ x : B \to A}}{\vdash \lambda x.\ \lambda y.\ x : A \to (B \to A)}$$

単純型付きラムダ計算と直観主義命題論理 (5)

論理式 $(A \wedge B) \Rightarrow (B \wedge A)$ の証明:

$$\dfrac{\dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash B} \quad \dfrac{\dfrac{}{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash A}}{\dfrac{A \wedge B \vdash B \wedge A}{\vdash (A \wedge B) \Rightarrow (B \wedge A)}}$$

型判断 $\vdash \lambda x.\ (snd\ x,\ fst\ x) : (A \times B) \to (B \times A)$ の導出:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \times B \vdash x : A \times B}}{x : A \times B \vdash snd\ x : B} \quad \dfrac{\dfrac{}{x : A \times B \vdash x : A \times B}}{x : A \times B \vdash fst\ x : A}}{x : A \times B \vdash (snd\ x,\ fst\ x) : B \times A}}{\vdash \lambda x.\ (snd\ x,\ fst\ x) : (A \times B) \to (B \times A)}$$

命題＝型 (Propositions as Types)

命題 (論理式) と型は本質的に同じ (記号をつけかえるだけで、構造や規則は同じになる。)

型	論理記号
直積型 ($\times$)	論理積 ($\wedge$)
直和型 ($+$)	論理和 ($\vee$)
関数型 ($\rightarrow$)	含意 ($\Rightarrow$)

命題＝型 の拡張

型システム	論理体系
単純型付きラムダ計算	命題論理
＋多相型 (System F, $\lambda 2$)	二階命題論理
＋高階多相型 (Fω, $\lambda\omega$)	高階命題論理
＋依存型 (λP)	一階述語論理
上記全部を含む	高階述語論理

依存型: 値に依存した型;

- 例. $\Pi(x : N).$`int list(n)` (整数 n をもらい、長さ n の整数リストを返す関数の型)

述語論理: $\forall x.\exists y.\ x < y \wedge \mathit{Prime}(y)$

Curry-Howard の同型対応 (まとめ)

論理体系と型システムが「本質的に同じ」ということ。

- 基礎となる型システムはほとんど全て、Curry-Howard 同型対応により、何らかの論理体系と対応している。
- 現実のプログラミング言語の型システムは、基礎となる型システムを拡張していることが多いが、それでも、Curry-Howard 同型対応は多くの示唆を与えてくれる。

型システムとは？

B. Pierce, “Types and Programming Languages”, MIT Press, 2002. (型理論の著名な教科書，TaPL と略記)

> *A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute.*

型システムとは？

B. Pierce, “Types and Programming Languages”, MIT Press, 2002. (型理論の著名な教科書，TaPL と略記)

> *A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute.*

- 「プログラムの (良くない) 振舞いが発生しない」ことを証明する.

型システムとは？

B. Pierce, “Types and Programming Languages”, MIT Press, 2002. (型理論の著名な教科書，TaPL と略記)

> *A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute.*

- 「プログラムの (良くない) 振舞いが発生しない」ことを証明する.
- 手に負える構文的な方法である

型システムとは？

B. Pierce, “Types and Programming Languages”, MIT Press, 2002. (型理論の著名な教科書，TaPL と略記)

> *A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute.*

- 「プログラムの (良くない) 振舞いが発生しない」ことを証明する.
- 手に負える構文的な方法である
- (プログラムの) フレーズを分類することにより,

型システムとは？

B. Pierce, “Types and Programming Languages”, MIT Press, 2002. (型理論の著名な教科書，TaPL と略記)

> *A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute.*

- 「プログラムの (良くない) 振舞いが発生しない」ことを証明する.
- 手に負える構文的な方法である
- (プログラムの) フレーズを分類することにより,
- フレーズたちが計算する値の種類に従って

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.
- 論理式 $(A \wedge B) \supset (B \wedge A)$ は証明可能.
- 型 $(A \times B) \to (B \times A)$ は要素を持つ
  - 引数の左右を逆にしたものを返す関数.

型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.
- 論理式 $(A \wedge B) \supset (B \wedge A)$ は証明可能.
- 型 $(A \times B) \to (B \times A)$ は要素を持つ
  - 引数の左右を逆にしたものを返す関数.
- 論理式 $(A \supset B) \supset (B \supset C) \supset (A \supset C)$ は証明可能.
- 型 $(A \to B) \to (B \to C) \to (A \to C)$ は要素を持つ.,
  - 関数を 2 つもらって，それらの合成関数を返す高階関数.

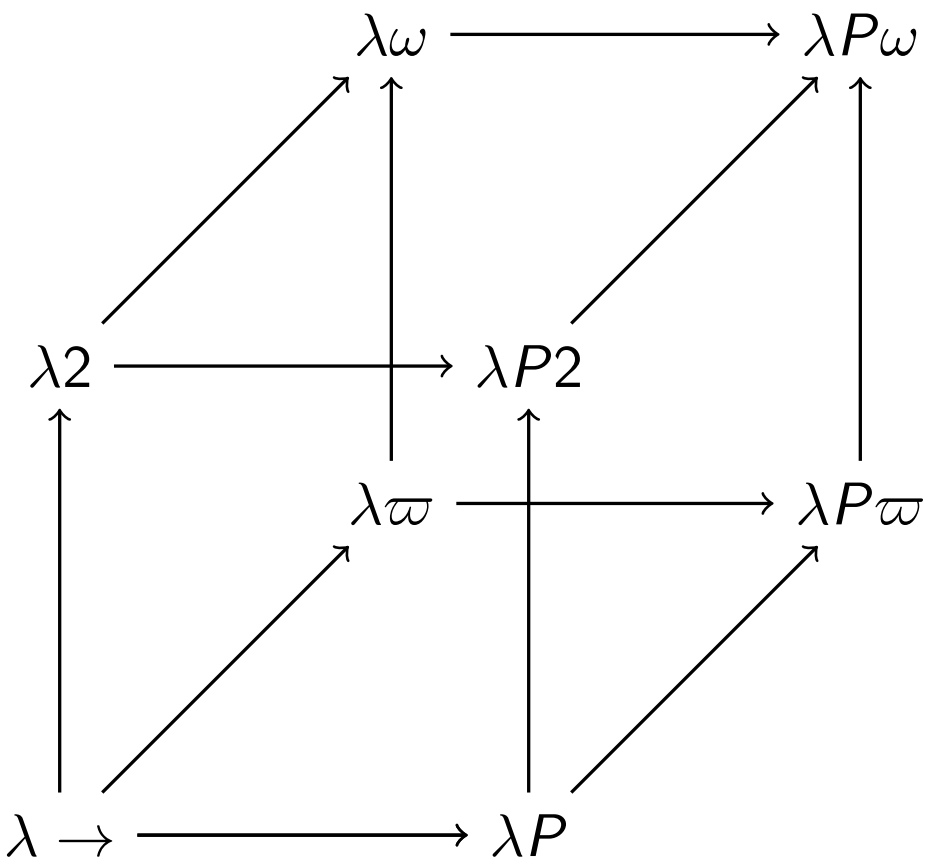
型と論理

「ある種の論理の世界と型の世界は同型である.」(Curry-Howard の対応)

- 論理式 $A \supset A$ は証明可能.
- 型 $A \to A$ は要素を持つ.
  - 恒等関数.
- 論理式 $(A \wedge B) \supset (B \wedge A)$ は証明可能.
- 型 $(A \times B) \to (B \times A)$ は要素を持つ
  - 引数の左右を逆にしたものを返す関数.
- 論理式 $(A \supset B) \supset (B \supset C) \supset (A \supset C)$ は証明可能.
- 型 $(A \to B) \to (B \to C) \to (A \to C)$ は要素を持つ.,
  - 関数を 2 つもらって，それらの合成関数を返す高階関数.

Curry-Howard の対応は，プログラミング言語の型システムの設計に多大な影響を与えてきた.

ラムダ・キューブ


$\lambda\omega$
$\lambda P\omega$
$\lambda 2$
$\lambda P2$
$\lambda\underline{\omega}$
$\lambda P\underline{\omega}$
$\lambda\to$
λP

まとめ

この授業では，プログラム言語の型システムを理解してそれを通して様々なプログラム言語について語るための概念を修得する．

- Java,C,TypeScript,ML,Haskell, ...

まとめ

この授業では，プログラム言語の型システムを理解してそれを通して様々なプログラム言語について語るための概念を修得する．

- Java,C,TypeScript,ML,Haskell, ...

演習では，特定のプログラム言語だけを一貫して使いたいので，型について議論しやすい言語を採用する．

まとめ

この授業では，プログラム言語の型システムを理解してそれを通して様々なプログラム言語について語るための概念を修得する．

- Java,C,TypeScript,ML,Haskell, ...

演習では，特定のプログラム言語だけを一貫して使いたいので，型について議論しやすい言語を採用する．

- OCaml (関数型プログラミング言語 ML の一族のひとつ)