Curry-Howard correspondence

Typed lambda calculation is well known to correspond well with the theory of proof of mathematical logic, which is called **Curry-Howard correspondence**. In Curry-Howard compliance, proposition type, proof lambda term $A \to B$ is a proposition representing "B if A", and the proof of $A \to B$ is considered to be a procedure to create a proof of B from the proof of A. This is a proof It is the idea of intuitionistic logic that it gives concrete evidence that the process of standardizing the proof which is an important technique to demonstrate the consistency of the logical system in the theory of proof is a typed lambda It corresponds to β reduction in calculation.

As a consequence of Curry-Howard correspondence, the semantics of typed ram calculation by Cartesian closed region described in this chapter can be read as it is as semantics of intuitionistic propositional logic as it is.
In other words, with respect to arbitrary **Cartesian closed category**, the semantics of **intuitionistic logic** which proof is probed for the proposition can be obtained.This is the field theory which is the field which studies mathematical logic theory using categorical theory. It is a basic example of a logical logic / categorical type theory.

CURRY-HOWARD対応

型付きラムダ計算は,数理論理学の証明の理論とうまく対応することが知られています. これを, Curry-Howard対応と呼びます. Curry-Howard対応では,型を命題, ラムダ項を証明と読み替えます. $A \to B$は「AならばB」を表す命題というわけです. そして, $A \to B$の証明は, Aの証明からBの証明を作り出す手続きである, と考えます. これは,証明とは具体的な証拠を与えるものであるとする直観主義論理の考え方です. しかも,証明の理論において論理体系の無矛盾性を示すために重要な技法である証明を標準化するプロセスが,型付きラムダ計算におけるβ簡約に対応します.
Curry-Howard対応の帰結として,本章で解説したカルテジアン閉圏による型付きラムダ計算の意味論は, そっくりそのまま直観主義命題論理の意味論と読み替えることができます.すなわち,任意のカルテジアン閉圏について,命題を対象に,証明を射とする直観主義論理の意味論が得られます.これは,圏論を用いて数理論理学を研究する分野である圏論的論理·圏論的型理論の基本的な例になっています.

Untyped lambda calculation

The untyped lambda calculation is a computational model that has lost all the concept of type from typed lambda calculations as its name, or it is a simple computation model with a special base type like $X = X \to X$. It can also be thought that it is net type typed lambda calculation (all programs have types, so it is the same as not actually thinking about type.) Because we do not have to worry about type, with simple typed lambda calculation we could not write, we can write programs like (λx, x (x)) (x. x (x)).
However, in **Sets** there is only one point set for set X which is $X \cong X^X$, so we can not give non-trivial semantics of unknown lambda calculation. What should we do this? This problem has been unresolved for a long time, but Dana In 1969, Scott found a Cartesian closed region with a nontrivial subject X that would be $X \cong X^X$, and succeeded in giving for the first time the display semantics of untyped lambda calculations, which at the time was a totally unexpected result It seems it was, one month before this discovery, Scott himself wrote that "untyped lambda calculation is too complicated to understand the display semantics and it is awkward to use typed lambda calculation from now on" it's dark.
After that, the results of Scott showed that it is effective not only to lambda calculation but also display semantics of various programming languages, and it has great influence on the development of program semantics.

型なしラムダ計算

型なしラムダ計算は, その名のとおり型付きラムダ計算から型の概念をすべてなくしてしまった計算モデルです.あるいは, $X=X\to X$であるような特別な基底型 を持つ単純型付きラムダ計算だと思うこともできます(すべてのプログラムは型を持つ．だから,実質的に型を考えないのと同じというわけです). 型を気にしなくてもいいので,単純型付きラムダ計算では書けなかった
(λx, x (x)) (x. x (x))のようなプログラムも書けます.
しかし, Setsでは $X \cong X^X$となる集合Xは一点集合しかないので,型なしラムダ計算の非自明な意味論を与えることができません.ではどうしたらいいのか? この問題は長らく未解決でしたが, Dana Scottは, 1969年に, $X \cong X^X$となる非自明な対象Xを持つカルテジアン閉圏を発見し,型なしラムダ計算の表示的意味論をはじめて与えることに成功しました．当時これはまったく予想外の結果だったようで，この発見の1か月前にはScott自身が「型なしラムダ計算は表示的意味論もよくわからなくて厄介だから，これからは型付きラムダ計算を使おうよ」と書いていたくらいです.その後，Scottの結果は，ラムダ計算にとどまらず多様なプログラミング言語の表示的意味論を与えるために有効であることが示され, プログラム意味論の発展に大きな影響を与えました.

Lawvere theory

Operators such as groups, rings, monoids and what are defined as structures that satisfy the axioms of the form of equations concerning them are generally called algebraic theory Bill Lawvere, in his doctoral dissertation, he developed a method to treat various algebraic theories uniformly in the form of a category called Lawvere theory, roughly speaking, the Lawvere theory of a certain algebraic theory $\mathcal{A}$ is that all the identities established in $\mathcal{A}$ It is a category that you know and it can be said that it embodies $\mathcal{A}$ itself.

By taking a certain colimit, we can construct a finite monad over **Sets** from the Lawvere theory. especially from the Lawvere theory of algebraic theory $\mathcal{A}$. A finite monad over **Sets** is created that allocates the underlying sets of freedom-A on A to set A.

M and *G* introduced as examples can be obtained in this way, while you can also retrieve the Lawvere theory from a finite monad on **Sets**. The following theorem has characterized the correspondence categorically.

Theorem 5.4　The category of Lawvere theory and the category of finite monads on **Sets** are equivalence of categories.

On the other hand, because the power set monad *P* and the continued monad *C* on **Sets** are not finite, it can not be made from the Lawvere theory, but by increasing the size of the Lawvere theory, those monads will be able to be captured.

LAWVERE理論

群,環，モノイドといった演算子と,それらに関する等式の形の公理を満たす構造として定義されるものを,一般に代数的理論(algebraic theory)と呼びます. Bill Lawvereは博士論文にてさまざまな代数的理論をLawvere理論と呼ばれる圏の形で統一的に扱う方法を開発しました.おおまかにいうと, ある代数的理論$\mathcal{A}$のLawvere理論とは$\mathcal{A}$ において成立する恒等式をすべて知っている圏であり,$\mathcal{A}$そのものを体現した圏ともいえます.

ある種の余極限を取ることで，Lawvere理論からSets上の有限的モナドを構成することができます.特に代数的理論AのLawvere理論からは「集合Aに対してA上の自由-Aの台集合 underlying sets を割り当てる」**Sets**上の有限的モナドが作られます.

例として紹介した*M*，*G*はこのようにして得られます一方，**Sets**上の有限的モナドからLawvere理論を取り出すこともできます. この対応関係を圏論的に特徴づけたのが以下の定理です.

定理 5.4
Lawvere理論の圏とSets上の有限的モナドの圏は圏同値になる.

一方,冪集合モナドPESets上の継続モナドCは有限的でないためLawvere理論から作ることができませんが, Lawvere理論のサイズを大きくすることで, それらのモナドを捉えることが可能になります.

Cayley‘s representation theory

Cayley‘s Cayley‘s representation theory

This theorem states that "Any group is isomorphic to a subgroup of a group permutations.

Theorem 8.1 An arbitrary group G is isomorphic to some of the subgroups of the symmetry group o

Sym$|G|$:={σ : $|G|$→$|G|$ σ is bijection}

on the underlying set.

Some may seem obvious, but I will prove it. Note that the above σ is not necessarily homomorphic.

Proof: Each element $a \in |G|$ of the proof group is bijective σ_a:

$|G| \rightarrow |G|$ is provoked.
$\sigma_a(b) := ab$

In order to see that it is surjective, for once $b \in |G|$

$\sigma_a(a^{-1}b) = a(a^{-1}b) = b$

It is good to do it, and to see that it is an injection

$\sigma_a(b) = \sigma_a(b') \Rightarrow ab \Rightarrow ab'$

$\Rightarrow a^{-1}(ab) \Rightarrow a^{-1}(ab')$

$\Rightarrow b \Rightarrow b'$

Therefore σa is an element of | Sym $|G|$ |, function,
$f : |G| \rightarrow |\ \mathrm{Sym}\ |G|\ |, a \rightarrow \sigma_a$

is determined.

Next, it shows that f is a homomorphism of the group, that is, "keep the structure of the group." Specifically, the following three.

$$\sigma_{eG} = e_{|\ \mathrm{Sym}\ |G|\ |}, \quad \sigma_{ab} = \sigma_a \sigma_b, \quad \sigma_a{}^{-1} = (\sigma_a)^{-1} \tag{8.1}$$

These can be shown easily · After that,

- f is an injection.
- In general, those which are monopolized by the homomorphic map of the group $h : G_1 \rightarrow G_2$ are isomorphic maps of the group

$G_1 \rightarrow \mathrm{Im}\ h$, where Im h is the image of h
Im $h := \{\ h(a) \quad | a \in G_1 \}$

So this will naturally become a subgroup of G.

If you show the above two (not difficult), the proof is over □

Notice that in Expression (8.1), "abstract and unknown product" ab in G is expressed as a "specially imaginable product" $\sigma_a \sigma_b$ that it is given a permutation σ_a after substitution .

ケイリーの表現定理

この定理は、「**任意の群は群置換の部分群と同型**である」と述べています.

定理8. 1　任意の群Gは,台集合!Gl上の対称群

$$\mathrm{Sym}|G| := \{\sigma : |G| \to |G| \ \sigma\text{は全単射}\}$$

の,部分群のうちのあるものに同型.

あたりまえに思える人もいるかもしれませんが,証明してみます。上のσは準同型写像とは限らないことに注意.

証明　群の各元 $a \in |G|$は,次のように全単射σ_a :

$|G| \to |G|$をひきおこす.

$\sigma_a (b) := ab$

全射であることをみるには,かつてな $b \in |G|$に対して

$$\sigma_a (a^{-1}b) = a (a^{-1}b) = b$$

とすればよいし,単射であることをみるには

$$\sigma_a (b) = \sigma_a (b') \Rightarrow ab \Rightarrow ab'$$
$$\Rightarrow a^{-1}(ab) \Rightarrow a^{-1}(ab')$$
$$\Rightarrow b \Rightarrow b'$$

とすればよい. よってσ_aは | Sym |G | | の元であり, 関数

$f : |G| \to |\ \mathrm{Sym}\ |G|\ |,\ a \to \sigma_a$

が定まる.

次にfが群の準同型写像であること,つまり「群の構造を保つ」ことを示す. 具体的には次の3つ.

$$\sigma_{eG} = e_{|\ \mathrm{Sym}\ |G|\ |}, \quad \sigma_{ab} = \sigma_a \sigma_b, \quad \sigma_a{}^{-1} = (\sigma_a)^{-1} \qquad (8.1)$$

これらは簡単に示せる·あとは,

- fは単射.
- 一般に,群の準同型写像で単射になっているもの $h : G_1 \to G_2$ は, 群の同型写像

$G_1 \to \mathrm{Im}\ h$　をひきおこす. ここでIm hはhの像

$$\mathrm{Im}\ h := \{\ h(a) \quad | a \in G_1 \}$$

で,これは自然にGの部分群になる.

という2つを示せば(難しくない), 証明終了. □

式(8.1)において, Gにおける「抽象的でよくわからない積」abが,置換 を施した後に置換σ_aを施すという「具体的にイメージできる積」$\sigma_a \sigma_b$として**表現**されていることに注意してください.

Derived category

Functor Hn: K (A) → A is homomorphism of a complex $f^\bullet: X^\bullet \to Y^\bullet$(all f n are homotypic of $\mathcal{A}$) homomorphic homomorphism

$H^n(f^\bullet) : H^n(X^\bullet) \to H^n(Y^\bullet)$

In this way, the shooting of a complex that leads to the homomorphism of arbitrary n is called a pseudo-conformal type All complex isomorphisms are quasi-symmetric, but the reverse is not generally correct.

Compounds with homomorphic homology have the same homology algebraic properties, so if you can construct a zone where pseudo-isomorphism is a complex isomorphism as it is, it is best suited as a stage for developing a complex homology algebra It is supposed to be the one where the pseudo-shaped inversion is added, and this is the arrival category D($\mathcal{A}$).
D($\mathcal{A}$) becomes a triangular category like K($\mathcal{A}$), the guiding category was devised by Grotan Diek, a systematic theory was built by Verdi, the structure of the triangle is clearly defined for the first time It is stated in.
At first glance, it is reported that deep relationships are found by observation through the derived category even for two objects which seem to be irrelevant.In the lead-in area or triangle area, It is used in various fields such as algebraic geometry.

導来圏

関手$H^n : K(\mathcal{A}) \to \mathcal{A}$は複体の同型射$f^\bullet: X^\bullet \to Y^\bullet$(すべての$f^n$が、$\mathcal{A}$の同型射になっている)をホモロジーの同型射

$$H^n(f^\bullet) : H^n(X^\bullet) \to H^n(Y^\bullet)$$

に移します. このように,任意のnでホモロジーの同型を導く複体の射を**擬同型**と呼びます.複体の同型射はすべて擬同型ですが,逆は一般に正しくありません.

同型なホモロジーを持つ複体どうしは同じホモロジー代数的性質を持ちますから, もし擬同型がそのまま複体の同型射であるような圏が構成できれば,複体のホモロジー代数を展開する舞台として最も適したものになるはずです. こうして擬同型の逆射を加えて構成される圏が**導来圏**$D(\mathcal{A})$です.
$D(\mathcal{A})$は$K(\mathcal{A})$と同じく三角圏になります.導来圏はグロタンディークによって考案され, ヴェルディエによって体系的な理論が構築されました. その中ではじめて三角圏の構造が明確に述べられています.
一見,無関係に見えるふたつの対象でも,導来圏を通して観察することにより,深い関係性が見出されるということが多く報告されています.導来圏あるいは三角圏は, このような形で環論や代数幾何学などいろいろな分野で用いられています.

補遺

LAMBDA CALCULUS

Lambda calculus (also written as λ-calculus) is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution. It is a universal model of computation that can be used to simulate any Turing machine. It was first introduced by mathematician Alonzo Church in the 1930s as part of his research of the foundations of mathematics. Lambda calculus

Lambda calculus consists of constructing lambda terms and performing reduction operations on them. In the simplest form of lambda calculus, terms are built using only the following rules:

Syntax	Name	Description
x	Variable	A character or string representing a parameter or mathematical/logical value
(λx.M)	Abstraction	Function definition (M is a lambda term). The variable x becomes bound in the expression.
(M N)	Application	Applying a function to an argument. M and N are lambda terms.

producing expressions such as: (λ*x*.λ*y*.(λ*z*.(λ*x*.*z x*) (λ*y*.*z y*)) (*x y*)). Parentheses can be dropped if the expression is unambiguous. For some applications, terms for logical and mathematical constants and operations may be included.

The reduction operations include:

Operation	Name	Description
(λx.M[x]) → (λy.M[y])	α-conversion	Renaming the bound (formal) variables in the expression. Used to avoid name collisions.
((λx.M) E) → (M[x:=E])	β-reduction	Replacing the bound variable with the argument expression in the body of the abstraction

If De Bruijn indexing is used, then α-conversion is no longer required as there will be no name collisions. If repeated application of the reduction steps eventually terminates, then by the Church-Rosser theorem it will produce a beta normal form.

ラムダ計算

ラムダ計算は,計算模型のひとつで,計算の実行を関数への引数の評価と適用としてモデル化・抽象化した計算体系です. ラムダ算法ともいいます. 関数を表現する式に文字ラムダ (λ) を使うという慣習からその名がある.アロンゾ・チャーチとスティーヴン・コール・クリーネによって1930年代に考案された.1936年にチャーチはラムダ計算を用いて一階述語論理の決定可能性問題を(否定的に)解いた。ラムダ計算は「計算可能な関数」とはLambda calculusなにかを定義するために用いられることもある.計算の意味論や型理論など,計算機科学のいろいろなところで使われ

ており,特にLISP、ML、Haskellといった関数型プログラミング言語の理論的基盤として、その誕生に大きな役割を果たしました.
ラムダ計算は1つの変換規則（変数置換）と1つの関数定義規則のみを持つ、最小の（ユニバーサルな）プログラミング言語であるということもできます.ここでいう「ユニバーサルな」とは、全ての計算可能な関数が表現でき正しく評価されるという意味です.これは、ラムダ計算がチューリングマシンと等価な数理モデルであることを意味しています.チューリングマシンがハードウェア的なモデル化であるのに対し,ラムダ計算はよりソフトウェア的なアプローチをとっています.この記事ではチャーチが提唱した元来のいわゆる「型無しラムダ計算」について述べています.その後これを元にして「型付きラムダ計算」という体系も提唱されています.

ラムダ計算(λ計算とも呼ばれる)は,関数の抽象化と変数の束縛と代入によるアプリケーションに基づく計算を表現するための数学的論理における形式的システムです.それはあらゆるチューリング機械をシミュレートするために使用することができる計算の普遍的なモデルです.それは最初に数学の基礎の彼の研究の一部として1930年代に数学者アロンゾチャーチによって導入されました.
（λｘ・λｙ（λｚ・（λｘ・ｚｘ）（λｙ・ｚｙ））（ｘｙ））のような表現を生成します.式が明確な場合は,括弧を省略することができます.いくつかの用途では,論理的および数学的定数および演算のための用語が含まれてもかまいません.ラムダ計算は,ラムダ項を構築し,それらに対して簡約演算を実行することで構成されています.最も簡単な形式のラムダ計算では,次の規則のみを使用して用語が作成されます. De Bruijnインデックスが使用されている場合は,名前の衝突が発生しないため, α変換は不要になりました.縮小ステップの繰り返し適用が最終的に終了すると,Church-Rosserの定理により,ベータ正規形が生成されます.