第９回補足資料

１２月12日（改訂版）まだ音声なし版です。

レポート課題で躓いているかもしれない人用の説明を1ページ足し、誤植でわかり難かった５ページ目を改訂しました。

第９回の上級者向けの内容をさらに少し加えました。

第２回レポート課題問題１、２についての追加のアドバイス

よくある質問２の答えのつづき：定義4.2の２が絡んでいる変項の場合は、定義4.2の２のλ記号導入とともに型表示をλの直後の変項の右肩に付けてください。

問題文の表現が型付けできると分かっても、その中に現れる変項でλで縛られることのないものがある場合には、その変項に対してどんな型を付けたかの説明も付けてもらうとよいです。

第８回補足資料では、(xx)が型付けできないことを指摘しました。

問題１，２で型付けできない場合は、定義4.2に従ってそのことを示してください。

一方、型付けできる場合の解答の仕方もアドバイスしておきます。

λx(yx)は型付けできますので、この例で示します。たとえば、

yをP→R, xをPを型付けしてもそれらはλ項と言えることが定義4.2の1で分かる。

この時(yx)が型Rのλ項であることが定義4,2の3で分かる。

さらに、このことと、xの型がPであったことから、定義4,2の２から、λx(yx)が型P→Rの型の型付きラムダ項であると分かる。

このように解答を書いてもらえばよいです。最後のラムダ項表記を$\lambda x^{P}(yx)$と書いてもらうのがよいです。　なお、PとRかたはじめなくても、AとBなどの型表現を選んでもよいです。

第２回レポート課題についていての追加のアドバイス

- 先週末に質問期間を設定し、皆さんからの質問を受けました。

その時特に目立った質問についてアドバイスしておきます。

これから提出する人で、解答に迷っている人は参考にしてください。

問題１と問題２について：

あくまでも定義4.2に従って回答してください。

逆に言えば、定義4.2が理解できれば十分に解答でいるので、定義4.2の理解に心がけてください。

第２回レポート課題問題１、２についての追加のアドバイス

よくある質問１.

変項だけでもラムダ項ですか。

答え；はいそうです。

定義4,2の１から、どんな変項自体も適当に型を指定すれば変項だけでもラムダ項となることが分かります。

第２回レポート課題問題１、２についての追加のアドバイス

よくある質問２.

問題文のλ記号の直後の変項の肩に型記号がついていないのはどうしてですか。

答え；型記号を付けて定義4.2に合ったλ項になるかどうかを問う問題だからです。

定義4.2の２から、tに型が付けられてxに型を指定すれば、λxtには特定の型が付けられることが分かります。例えばｔにCという型が付けられて、xにDという型をつけるとら、λxtの型はD→Cとなり、$\lambda x^{D}t$のように型を右肩につけることになります。

~~tが型付けできているときには、tの中にλが現れるときはそれらのλの直後の変項にも右肩に定義4.2の２から、ｔに型がtに型が付けられてyに型を指定すれば、λxyには特定の型が付けられることが分かります。例えばｔにCという型が付けられて、ｙにDという型をつけるとら、λxtの型はD →Cとなり、λx Dtのように型を右肩につけることになります。Tが型Cで型付けできているときにはtに現れるmonnsai~~

取り消し線の部分は誤植もあり分かりにくいので消しました。代わりに、２ページ後に、さらにもう少し、解答の仕方の説明を繰り返しました。

よくある質問２のつづき

ある質問２の答えのつづき：定義4.2の２が絡んでいる変項の場は、定義4.2の２のλ記号導入とともに型表示をλの直後の変項の右肩に付けてくれれば、その束縛変項に対してどんな型を宣言したのかこちらで分かります・

問題文の表現が型付けできると分かっても、その中に現れる変項でλで縛られることのないものがある場合には、その変項に対してどんな型を付けたかの説明も付けてもらうとよいです。

第８回補足資料では、(xx)が型付けできないことを指摘しました。

このような例では、定義4.2に従えば、なぜ型付けできないかを示せます。

問題１，２で型付けできない場合は、定義4.2に従ってそのことを示してください。

一方、型付けできる場合の解答の仕方もアドバイスしておきます。

λx(yx)は型付けできますので、この例でどのように解答するか示します。たとえば、

yをP→R, xをPを型付けしてもそれらはλ項と言えることが定義4.2の1で分かる。

この時(yx)が型Rのλ項であることが定義4,2の3で分かる。

さらに、このことと、xの型がPであったことから、定義4,2の２から、λx(yx)が型P→Rの型の型付きラムダ項であると分かる。

このように解答を書いてもらえばよいです。最後のラムダ項表記を$\lambda x^{P}(yx)$と書いてもらうとよりよいです。　なお、PとRという型表現で解答をはじめなくても、AとBなどの型表現を選んでもよいです。

解答の仕方についてのヒントの繰り返し

説明をもう少し詳しく繰り返します。すでに分かった人はこのページを飛ばして構いません。

問題1,2で型付けできる場合は、各変項の型を示してください。（λの後の変項にも右肩に型記号を付けた形も示していただくのが望ましいです。）

例えば問題文が、「λf.(f (fx))は型付きλ項となるように型を付けられますか。型付けできる場合はなる場合は定義4.2に従ってそのことを示してください。べく簡単な型表現で示してください」であったときのの答えとしては、例えば、

xの型としてAとかPとかRとかEというような一字で表せる型を選べば、 x だけの部分が定義4.2の 1 から型付きλ項と分かり、

たとえばEをxの型と選んだ人は、(fx)の部分が型付きλ項ためには、定義4.2の 3 から、fは→の左にEが来る型を選ぶことになり、例えば、→の右にPやEやSなど一字で書けるものを選ぶことになります。

fの型としてE→(R→D)としても、定義4.2の 3 から(fx)の部分は(R→D)型の型付きλ項と言えますが、問題文になるべく単純な型表現を選ぶよう指示されている場合は、例えば

fの型としてE→Rなどの最も単純な形式（ただし形式が→一つで→の左がEで右が一文字であればなんでもよいわけです）を選んで、定義4.2の3により、その時(fx)の部分が型Rのλ項になることを説明してください。ステップを踏んで、最終的に$\lambda f^{E\to R}.(f(fx))$の形の型付きラムダ項となり、その型が、(E→R)→Rとなることを、定義4.2により、ステップを踏んで答えてください。

問題1,2で型付けできない表現であると答える場合は、補助教材で言及した(xx)が型付けできないことをどのように定義4.2により示すかも参考にしてください。この場合、定義4.2の 3 により、一般に(xy)の形の表現が型付けされるためには、xを例えばA→Bの形で、yはこの関数型のxが入力の型として待っているAの型と定義4.2のので指定しておかねばなりませんが、y も x であったとすると、 x の型はA→Bであると同時にAでなければならず、一方で定義4.2の1で変項の型宣言は一意的でなければならにので、型付けできない、というような説明になります。

問題 3 の解答に当たってはすべての β 簡約計算過程でλ記号の直後にある（束縛）変項の右肩の型表示を省略しないでください。

上級者の答え方：よくある質問のつづき

- 前のページでは、ある表現が型付きラムダ項であることを示すのに、定義4.2の定義のテキストを引用しながら答えましたが、定義4.2のすぐ後にある、記号表現だけの型推論を組み合わせて、ちょうどそのラムダ項に対する証明が現れる形で解答してくれても結構です。
- これは上級者に対して期待している解答方式です。

- ただし、どちらの形で解答してもらっても、正解であれば点差はありません。

第２回レポート課題についていての追加のアドバイスのつづき：「問題３」のよくある質問の回答

レポート課題の問題に3ついて：

７回目補足資料と対応する基本教材でこの部分の説明をしましたが、さらに８回目補足資料の（表紙も含めて）３ページ目と４ページ目では、この「問題３」のヒントにもなるような説明を加えています。参考にしてください。

　β簡約のすべてのステップを省かずに書いてください。

　λ記号直後の（束縛）変項の右肩に付ける型表現は計算途中でも省略しないでください。

Ｂ簡約できる箇所（リデックス）が複数あるときは自由に選んだ順で簡約していって構いません。λ計算の合流性により、計算結果は同じになりますので、ご自分で計算しやすい順で計算してください。

それ以上β簡約できないλ項（既約・正規形）になるまで簡約していってください。型付きλ計算の停止性により、必ず既約λ項に到達します。

ここからが今回の本題：ラムダ計算の基本性質

これまでも簡単に触れていましたが、これまで触れていたλ計算の基本性質についてここで整理してまとめてみます。1.

1.合流性（チャーチ＝ロッサー性とも呼ぶ）

どんな順序でβ簡約して言っても同じ計算結果になる。

２．停止性　（正規化性）

計算が必ず止まること。

３．健全性（Sunjective reductionとの呼ぶ）

Β簡約しても型が変わらないこと。

４．カリー＝ハワード同型

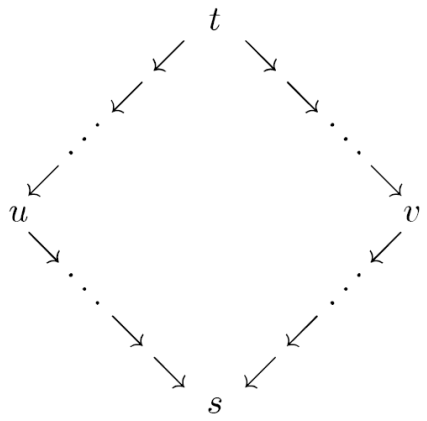
型付きラムダ計算と論理とが対応関係にあること

1．合流性（チャーチロッサー性）
（β 簡約の順序によらず同じ結果になりえる）

ここで**停止性**とは，任意のラムダ項 t に対して，t から始まる任意の順序で β －簡約の適用を繰り返すと常に有限で止まることを意味する．又，**チャーチ・ロッサー性**とは，t からの任意の書き換え $t \to^* u$ 及び $t \to^* v$ で得られたラムダ項 u，v に対してあるラムダ項 s が存在して，

$$u \to^* s \ {}^*\!\leftarrow v$$

が成立する．即ち，



停止性　計算が必ず止まる

- 型付きλ計算の停止性については、基本教材の命題論理証明の強停止性と弱停止性の区別がそのまま成り立ちます。
- 授業で取り扱っている型付きλ計算は強停止性を満たします。つまり、どのような順序でリデックスを選んでβ簡約を進めても、計算は必ず規約（正規形）λ項で停止します。

弱い形の停止性の証明は、命題論理の証明の弱正規化定理の証明に対応することを行えば完結します。上級者はできるだけ自分でやってみてください。

強い停止性の証明は本授業の範囲を超えますが、興味がある人がいれば、参考文献を紹介します。

合流性と強停止性により、λ項は計算順序によらず「一意的」に規約形で停止することが分かります。

追加スライド：弱停止性証明：上級者へ

基本教材の命題論理の証明論の最後の部分にある、証明の弱正規化定理の証明を参照しながら、型付きλ計算の弱停止性証明を与えてみてください。

数学的帰納法を用います。

、あた。この弱正規化定理で証明できた停止性を保証する評価方略（戦略）とはどのような方略であるかをλ計算の観点で説明してください。

簡約計算手続きの方略・戦略をうまく選べば正規形証明や既約形型付きλ項に至ることは、1930年代に証明の正規化や型付きλ項のβ簡約の概念が登場するや否や示されましたが、どんな方略をとって簡約・評価していっても停止することは1960年ころまで未解決でした。

合流性については授業の中でも言及しました（このページは余談です。）

Β簡約できるリデックスが複数あるとき、どの順序で簡約していっても簡約過程が合流する。

内側のREDEXをすべてβ簡約していく評価方略と必要な出るまで評価を控える方略などを選べるのも、結局はどんな順序で評価しても評価過程が合流するという合流性で保証されていた。（前回授業では、Eager　Evaluation、Lazy　Evaluationという用語も紹介しました。

前回のORの評価の時に外側から評価したのは、Tが引数として受け取る((Fm)m)及び((Fm)n)事態は評価しないで受け取る計算の仕方で、これをCall –by-name(名前呼び出し）と呼び、練習問題として眠さんに残した、((Fm)m)と((Fm)n)というTの引数を評価して値を確定した後で、Tに値を渡してTを計算する仕方をCall-by-value(値呼び出し）という区別にもあたります。

合流性と停止性

多くの計算システムは合流性をはじめから満たしていることが多いです。例えば日常の足し算、掛け算などもその例ですから、部分的に複数の箇所で計算を進めることができる場合は、どの順番で計算しても同じ値で計算が停止するわけです。

　λ計算の大きな特徴は、

型付きλ計算の場合：合流性と停止性の両方が成り立つ

ただし、計算力は限定的

型なしλ計算の場合：停止性は成立しない（前回授業で判例を出したとおりです）が、合流性は成立する。あらゆる計算が表現可能（これについてもすでに説明した通りです。）

最上級者向けページ：
合流性が成立しない例

- 命題論理の正規化で合流性が成立しない箇所を探し出そう：「または」の導入規則には二種類（右に入る場合と、左に入る場合）あります。これを区別すると、「または」の極大論理式をA∨Aと置くと（つまり、基本教材の極大論理式のA∨BのBをAととると）、さらにCとしてA∨Aをとると、合流性の反例を作れます。
- より進んだ非合流性と合流化手法：例えば代数の群論公理を計算論的に取り扱おうとする目的で、非合流的群公理を合流性を持ち停止性も持つ形の群論計算規則に変換する手続きなどが有名です。Knuth-Bendix Completion(クヌース＝ベンディックス完備化）という言葉で検索すると多く出てきます。このKnuthはThe Arts of Computer Programmingというとても有名な本の著者でもあり、Latexという自分の原稿作成用にプログラムしたワープロソフトを一般公開した人でもあります。（基本教材でも利用しています。）

型付き λ 計算の健全性

- 　β 簡約前と後とで型が変化しないことを型付き計算システムの健全性と呼ぶことがあります。

- 本授業で取り扱っている単純型付き λ 計算体系は健全背を持ちます。
- 基本教材でも強調しているように、型付き λ 計算の計算実行は証明の正規化に対応しており、どちらもこの性質を持つことが分かります。

型なしラムダ項の例

- 型なしラムダ計算はラムダ項プログラムの停止性保証や論理的制御に限界がありますが、計算力には限界がなく、あらゆる計算ができる（チャーチ＝チューリングのテーゼ）という特徴があります。

このではとても込み入った計算プログラムの例ではなく、ごく簡単なラムダ項プログラムで、型なしラムダ項の表現力の豊かさを見てみましょう。

掛け算TIMESの模様入換え法を使った指数関数の例

- ((Times 3)2)つまり((λm^{Nat}λn^{Nat}λf$^{P\to P}$(m(nf))3)2)は、すこし大雑把に言えば、III X IIの形の掛け算をするのにIIをffにした後で、IIIのそれぞれにffを一遍にに代入され、ff ff ffとなるようなパターン変換をしていました。

同じからくりを使って指数関数を定義してみましょう。

2^3：((Exp 2)3) を次のλ項と考えます。

((λmλn(n(Times m))1)2)3) ここで2回β簡約すると

((3(Times 2))1) この計算は掛け算のパターン変換同様、

上の赤カッコのβ変換で、IIIのそれぞれに(Times 2)[意味は2X]がi一遍に代入され、(λx((Times 2)((Times 2) ((Times 2)x)))) 1)一2X番外カッコのβ簡約をしたあとは、(2X(2X(2X1)))に対応するλ項になったわけです。

つまり、IIIに(Times")をβ簡約で一遍に代入するパターン変換操作は掛け算の時と同様です。3をλfλx(f(f(fx)))としたとき、型付きラムダ計算ではfの型はP→Pでしたが、上のβ簡約で代入した(Times 2)の型はNat→Natとなり、型が合致していません。

Xに1を代入するβ簡約の時も型が全く一致していません。でも、とてもエレガントな指数関数の定義になっています。型なしλ計算の定義の一例です。

型なしλ計算の他の定義例

- 本授業では単純型付きラムダ計算及びその命題論理との一致関係（カリー＝ハワード対応）について議論してきました。
- 型付きの条件を外して計算する場合は、基本教材で取り扱った論理的演算子（論理結合子）はっ単純化できます。型を無視してTを λxλyx , Fを λxλyy とします。
- 例えば補足資料でも議論した　OR「または」は、

Or ≡ λxλy((xT)y)

という単純か形で充分となります。

xにTが代入される場合は、そのTの第一引数であるTが返され、

xにFが代入される場合はyの値が返されます。このことから、一つ目の引数か二つ目の引数のすくなくとも一方がT（真）のときは、β簡約の結果がTとなり、ORの真理表道理となります。これまでの型付きλ計算の立場で見ると、このORは型付きラムダ項となっていないことに注意してください。

型なしλ計算の場合のANDの定義も考えてください。ところで教材で扱った型付きラムダ計算のOR、ANDの定義はもちろん型なしラムダ計算でもOR,ANDとして使用することもできます。すべての型付きλ計算は型情報を消しても型なしλ計算として通用します。

以下は初版になく、今回付け加えた部分です。、

型付きλ計算に固有な基本性質の続き

基本性質の４つ目は既に何度も補足資料で述べていました。
基本教材では、「ならば」だけの型（関数型と呼ばれることが多いです）λ計算の箇所と、「かつ」積型を加えたλ計算の箇所でつぃぎの型付きλ計算と命題論理との正確な対応（同型）関係が紹介されています。

定理 2.24 (カリー=ハワード同型定理 (Curry-Howard-Lambek isomorphism)) タイプ A を命題論理と同一視し、$t : A$(「ラムダ項 t がタイプ A を持つ」) を「t が命題 A の証明構造である」と読むことにより、命題論理とラムダ計算との同一視をすることができる。特に次のような同一視ができる。

タイプ A	$=$	論理式 A
ラムダ項 t	$=$	証明構造 t
ラムダ項の正規化過程	$=$	証明構造の正規化過程

型付き λ 計算に固有な基本性質の続き

「ラムダ項の正規化過程」と下に書いてあるのは、β 簡約による簡約過程であり、ラムダ項プログラムの計算実行過程のことです。この過程の各ステップもラムダ計算と証明正規化の間に正確な対応関係がありわけです。

定理 2.24 (カリー=ハワード同型定理 (Curry-Howard-Lambek isomorphism)) タイプ A を命題論理と同一視し、$t : A$(「ラムダ項 t がタイプ A を持つ」) を「t が命題 A の証明構造である」と読むことにより、命題論理とラムダ計算との同一視をすることができる。特に次のような同一視ができる。

タイプ A	$=$	論理式 A
ラムダ項 t	$=$	証明構造 t
ラムダ項の正規化過程	$=$	証明構造の正規化過程

対応関係が分かるコツ（上級者へ）：ラムダ項がどの証明と正確に対応しているか。例4.1にいくつ示しています。これらはラム項の定義を型推論を用いた形で型チェックしていくと現れてきます。逆に、証明が一つ与えられれば、それを型推論による型チェックをしているとみなすことによって、(束縛変項記号の選び方を無視すれば）唯一のラムダ項が得られます。
この対応関係は「上級者」の人にはマスターしてもらえれば充分です。
両者の同一視のために、一つだけ有用なトリックがあります。これがコツです。
下の型チェックの初段で、x;A（変項 x は型Aである）と型宣言しています。証明としてみるときにはこれを$[A]^x$（つまり前提Aが証明のどこかでx番という記号を付けて前提でなくなる）と読みかえます。意味のあるラムダ項（プログラムであれ、データであれ）の変項は通常束縛された変項として表れますので、対応する証明も開いた前提が残らず、すべてカギカッコでどこかで閉じると考えます。前提にははじめじゃら番号を付けておきます。証明では数字で番号を付けましたが、特に数字である必要はないので変項を用います。これがトリック（コツ）です。

$$\dfrac{\dfrac{x:A}{\lambda y^B x:B\to A}}{\lambda x^A\lambda y^B x:A\to(B\to A)}$$

上級者向き

型なしλ計算に基づく具体的な関数型言語の最初のものはまかーしーのLISPでした。その方言のSchemeは米国を中心に記号処理的人工知能言語として採用されました。

一方型付きλ計算に基づいた関数型プログラム言語はミルナーらによるMLが初期からあるものです。カリー＝ハワード同型定理に基づく型理論的関数型言語としてHaskell,また、この同型性を定理証明支援言語として発展させたものに、Coq, Agdaなどの代表的なものがあります。その他、オブジェクト指向的Ocaml, Java仮想マシン動作の関数型言語、MSのF#、等多くの有名な型付き関数型言語があります。どの言語にも入門者用のチュートリアル情報ページがあると思いますので、興味があれば見てみてください。上記のように特徴が違いますが、HaskellとCoqはお勧めです。

（皆さんの多くが学習しているPythonでもユーザーによる関数の型指定はできます。ただし、通常の環境では厳密に型チェックされません。）