Homotopy Type Theory によるパッチ理論の拡張

佐藤善紀　三好博之

本発表では C. Angiuli，E. Morehouse，D. R. Licata，R. Harper によって行われた Homotopy Type Theory (HoTT) による Darcs スタイルのパッチ理論の定式化を複数ファイルからなるリポジトリの場合に拡張する．複数ファイルに拡張する際のポイントとなるのは，単一ファイルの場合と異なりファイル名（識別子）とファイルの内容を明示的に区別する必要があることである．パッチを適用する対象が複雑になっても，identity type があることによりその対象についてのパッチという概念が容易に定義できることを示す．

1　はじめに

近年ソフトウェアの大規模化に伴い，ソフトウェア開発の多くの現場でバージョン管理システムが使われようになっている．バージョン管理システムは，ファイルの変更記録を管理するためのシステムであり，ファイルについてどのような変更をしたか，新しいファイルを作成したか，存在するファイルを削除したか，などの変更に関するすべての情報をリポジトリと呼ばれるデータベースに保存する．このリポジトリによってバージョン管理システムはファイルを特定の過去の状態に戻すことや，ファイルの異なる複数のバージョンを管理することなど多くの機能を提供する．

本発表ではバージョン管理システムの一つである Darcs について形式化を考える．Darcs でば”パッチ”とはリポジトリの内容の変更を意味する．例えばファイルの内容を編集したり，新しいファイルを追加したり削除したらリポジトリの内容は変更されたことになる．また Darcs ではパッチの処理は元に戻すことができる，すなわち可逆である．パッチを基礎として考えると，Darcs ではリポジトリをパッチのシーケンスの結果と見なすことが出来る．何らかの形式的枠組みをもちいてリポジトリやパッチを表現し，パッチを基礎として構築されたバージョン管理システムの形式化はパッチ理論と呼ばれる．

一方で 2006 年前後に，ホモトピー論と Martin-Löf type theory の間に密接な関係があることが発見され，Homotopy Type Theory (HoTT) と呼ばれる新しい分野が生まれた [1]．例えばホモトピー論で扱う，空間 X，空間 X の点 a，b，空間 X 内の a から b へのパス p をそれぞれ type theory の，type $\mathtt{X}$，type $\mathtt{X}$ の項 $\mathtt{a}$，$\mathtt{b}$，identity type の項 $\mathtt{p} : \mathtt{a} =_{\mathtt{X}} \mathtt{b}$ に対応させることができる．実際，Martin-Löf type theory の任意の type 上の identity type 全体は ∞-groupoid 構造（weak higher category の特別な場合）を持ち [2] [3]，一方で ∞-groupoid 構造はホモトピー論において空間を分類する．HoTT はホモトピー論と higher category theory を踏まえた上で，このような対応関係に基づいて，Univalence Axiom (UA) と higher inductive types (HIT) により Martin-Löf type theory を拡張したものである．

C. Angiuli，E. Morehouse，D. R. Licata，R. Harper は，パッチに identity type の項すなわち空間内のパスを対応させることで，リポジトリ内にファイ

An extension of a patch theory in homotopy type theory.

Yoshinori Sato, 京都産業大学大学院理学研究科, Division of Science, Kyoto Sangyo University.

Hiroyuki Miyoshi, 京都産業大学理学部数理科学科, Dept. of Mathematics, Kyoto Sangyo University.

ルが単一の場合の Darcs のパッチ理論を HoTT を用いて定式化した [4]．本発表ではこの研究を踏まえて，さらに現実的な開発モデルに近付けるためリポジトリに複数のファイルが存在する場合の定式化を与える．

2 パッチ理論

リポジトリとは，ディレクトリ，ファイル，ファイルの中身が集まったものである．このリポジトリを形式的に変更するのをパッチと呼ぶ．各パッチにはパッチが適用できるリポジトリを表す domain と，パッチを適用した後のリポジトリを表す codomain を持つ．例えばあるファイルを消去するパッチはリポジトリにそのファイルが存在しないと適用できないから，domain はそのファイルが存在するリポジトリである．また codomain はそのファイルが存在するリポジトリになる．

リポジトリにパッチを適用してもリポジトリを変化させないパッチを no-op パッチと呼ぶ．これはパッチ操作の単位元である．さらにここで考えるパッチ理論ではすべてのパッチは可逆であるという性質を持つとする．つまりパッチを $\mathtt{p}$ とすると undo パッチ!p が存在し，パッチ p を適用した後に undo パッチ $\mathtt{!p}$ を適用するとリポジトリは元の状態のままである．さらに，一般化して undo パッチ $\mathtt{!p}$ を適用した後にパッチ $\mathtt{p}$ を適用することもできてリポジトリは元のままであるとする。これによりパッチ全体を groupoid とみなすことが出来る．

3 ホモトピー型理論

Homotopy type theory (HoTT) の基本となる考えは Martin-löf 型理論をホモトピー論によって解釈するということである．

型理論の型 $\mathtt{A}$ をホモトピーの空間 A，また型 $\mathtt{A}$ の項 $\mathtt{a}$ を空間 A の点 a だと解釈する．全ての型 $\mathtt{A}$ と項 $\mathtt{a}$，$\mathtt{b}$ について identity type $\mathtt{a}{=}_{\mathtt{A}}\mathtt{b}$ が存在する．この identity type の項 $\mathtt{p} : \mathtt{a}{=}_{\mathtt{A}}\mathtt{b}$ を点 a から b へのパスだと解釈する．さらに $\mathtt{a}{=}_{\mathtt{A}}\mathtt{b}$ は型であるから，この型の任意の項 $\mathtt{p}, \mathtt{q} : \mathtt{a}{=}_{\mathtt{A}}\mathtt{b}$ についても identity type $\mathtt{p}{=}_{\mathtt{a}=_{\mathtt{A}}\mathtt{b}}\mathtt{q}$ が存在する．この型 $\mathtt{p}{=}_{\mathtt{a}=_{\mathtt{A}}\mathtt{b}}\mathtt{q}$ の項がパス p からパス q へのパス，つまりホモトピーに対応する．以下，等号の添字は自明な場合には省略する．

さらに恒等パス，パスの逆，パスの合成などのパスの演算を定義することが出来る．つまり，空間の各点 a から型 $\mathtt{a} = \mathtt{a}$ の項を作ることができ，その項を $\mathtt{refl}_{\mathtt{a}}$ で表し恒等パスだと解釈する．同様に任意の点 a から点 b のパス p に対応する項を $\mathtt{p} : \mathtt{a} = \mathtt{b}$ とし，p の逆向きのパスを考えると型 $\mathtt{b} = \mathtt{a}$ の項を作ることが出来て，その項を $\mathtt{!p}$ と表す．任意の 2 つの項 $\mathtt{p} : \mathtt{a} = \mathtt{b}$ と $\mathtt{q} : \mathtt{b} = \mathtt{c}$ から，型 $\mathtt{a} = \mathtt{c}$ の項を作ることができ，その項をパス p と q の合成に対応するものと見なして $\mathtt{q} \circ \mathtt{p}$ と表す．これは Martin-löf 型理論で推論できる事実と整合性がとれている。

そのように考えてホモトピー論のモデルにおいて成り立つ事実を、型理論に取り入れたものが Homotopy Type Theory である。具体的には新たに導入されたのは Univalence Axiom (UA) と higher inductive types (HIT) である。前者は各型階層で、identity type と依存型を用いて 2 つの型のホモトピー同値が型として定義できることを用いて、$(\mathtt{A} = \mathtt{B}) \cong (\mathtt{A} \cong \mathtt{B})$ という型として表される公理である（同値関係 $\cong$ はそれぞれ適切な型階層に属するとする）。

後者は空間における点と経路に関する帰納的定義を可能にするために導入された機構である。パッチ理論を定式化する際にはこの拡張された部分が主要な点で用いられている。通常の帰納法では，自然数ならば二つの generator，$\mathrm{zero} : \mathrm{Nat}$ と $\mathrm{succ} : \mathrm{Nat} \to \mathrm{Nat}$ によって帰納的に定義できる．この様に inductive type は generator によって帰納的に定義される．Higher Inductive Type は generator として点だけでなく，一つ次元が上の path も含める事により inductive type を一般化したものである．

4 単一のファイル（を持つリポジトリ）に関するパッチ

C. Anguiuli et al. による 1 つのファイルに行を追加したり削除したりする抽象的なパッチの定式化は以下の様なものである．まず m 行のファイルから n 行のファイルへの変化を考える．任意のファイルの変化は，ファイルの特定の場所に 1 行を追加する操作 (パッチ)`ADD_@_::_`と特定の場所の 1 行を削除する操

作（パッチ）RM_::_の繰り返しによって実現することが出来る．よってそのような操作の列（シーケンス）をパッチヒストリー Histroy m n と呼ぶ．同じ変化を実現するヒストリーは複数ありうる．よってそのような History を同一視することにより抽象的なパッチという概念を定式化することが出来る．これは実際には次のように同一視を考慮した HIT (quotient higher inductive type) によって定義できる．

```
space History : Nat → Nat → Type where
  [] : History m n
  ADD_@_::_ :{m n : Nat}(s : String)
      (l : Fin n+1) →
      History m n → History m n+1
  RM_::_ : {m n : Nat}(l : Fin n+1) →
      → History m n+1 → History m n
  ADD-ADD-< : {m n : Nat}(l1 : Fin n+1)
    (l2 : Fin n+2)(s1 s2 : String)
    (h : History m n) → l1<l2 →
    ADD s2 @ l2 :: ADD s1 @ l1 :: h)
    = (ADD s1 @ l1 :: ADD s2 @ (l2-1) :: h)
  ADD-ADD- ≥ : {n : Nat}(l1 : Fin n+1)
    (l2 : Fin n+2)(s1 s2 : String)
    (h : History m n) → l1 → l2 →
    ADD s2 @ l2 :: ADD s1 @ l1 :: h)
    = (ADD s1 @ l1+1 :: ADD s2 @ l2 :: h)
```

ヒストリーの同値関係を定義するためには 2 つの操作の組み合わせとして ADD-ADD 以外にも，RM-ADD, ADD-RM, RM-RM も定義する必要があるが，同様に定義できるのでここでは省略している．

空ファイルに適用できるヒストリー，つまり型 History 0 n の要素は一意的にファイルを決定する．このようなヒストリーは，ファイルの空間のパス，すなわち identity type で表現される抽象化されたパッチの context（その抽象化されたパッチが適用できるファイルを特定するヒストリー）と考えることが出来る．

```
space R : Type where
  doc : {n : Nat} → History 0 n → R
  addP :{n : Nat}(s : String) (l : Fin n+1)
  (h:History 0 n) → doc h = doc (ADD s@l::h)
  rmP : {n : Nat}  (l : Fin n+1)
  (h:History 0 n+1) → doc h = doc (RM l::h)
```

ここで doc h はヒストリー h から生成されるファイルを表す型である．

History の定義と同様に，抽象化されたパッチを適用する順番を交換したとき行番号を調整すればパッチを同一視出来る，ということを表す高次のパスは以下の様に表現される．

```
addP-addP-< : {n : Nat}  (l1 : Fin n+1)
  (l2 : Fin n+2) (s1 s2 : String)
  (h : History 0 n) → l1 < l2 →
  PathOver (x.doc h=doc x) ADD-ADD-<
  (addP s1 l2 ∘ addP s1 l1)
  (addP s1 l1 ∘ addP s2 (l2-1))
addP-addP-≥ : {n:Nat}  (l1:Fin n+1)
  (l2 :Fin n+2)(s1 s2 : String)
   (h:History n) → l1 ≥ l2 →
  PathOver (x.doc h=doc x) ADD-ADD-<
  (addP s1 l2 ∘ addP s1 l1
  (addP s1 (l1+1) ∘ addP s2 l2
```

addP-rmP，rmP-addP，rmP-rmP についても同様に定義できるので省略している．

5 複数のファイルを持つリポジトリ

この節では複数のファイルの集まりすなわちリポジトリに対するパッチを考える．前節と同様に History m n を m 行のファイルに適用でき，n 行のファイルにするパッチのシーケンスの型とする．パッチシーケンスはパッチがどのように合成されてきたかを辿ることが出来るからパッチヒストリーと呼ぶ．History 0 n の項は 0 行のファイルつまり，空ファイルに適用できて n 行のファイルにするパッチヒストリーである．このパッチヒストリーは一意にファイルを決定するからパッチヒストリーはファイルだと見なせる．ここまでは前節と同様の議論である．

ここで，複数のファイルを持つリポジトリにファイルを追加するリポジトリパッチとファイルを削除するリポジトリパッチを考える．さらに同一ファイルの内容を変更するパッチを別に扱うことも考えられるが，ここではそれは一旦ファイルを削除してから新たに

ファイルを追加すると考えて単純化する．

複数のファイルを持つリポジトリ内のファイルには名前（識別子）が必要である．なぜなら同一内容でファイル名が異なるファイルが存在しうるからである．ここでは議論を単純にするために名前として自然数を用いると名前付きのファイルの型は依存和を用いて `Σ(n : Nat) R` と表せる．簡略化のためにこの型を `D` と呼ぶことにする．つまり名前が付いたファイルを `d` とすると `d : D` である．

次に複数のファイルを持つリポジトリの型 `Rep` を定義する．Σ 型の要素はペアなので，リポジトリを名前付きファイルのリスト（もしくは有限集合）と考えることにより，名前付きファイルをリポジトリに追加する関数 `f-set-plus` と，リポジトリから名前付きファイルを削除する関数 `f-set-minus` はリスト操作と場合分けを用いて容易に定義できる．するとリポジトリの型は以下のように定義できる．

```
space Rep : Type where
  [] : Rep
  f-set-plus : D → Rep → Rep
  f-set-minus : D → Rep → Rep
```

以上からリポジトリに対するパッチ理論は以下のように同一視を考慮した HIT を用いることにより `ARep` 上の identity types をパッチとして定義できる．

```
space ARep : Type where
  doc : Rep → ARep
  addf : (d : D) → (RD : Rep)
    → doc RD = doc(f-set-plus d RD)
  rmf : (d : D) → (RD : Rep)
    → doc RD = doc(f-set-minus d RD)
```

さらに `addf-addf`，`addf-rmf`，`rmf-addf`，`rmf-rmf`，を用いた同値性も，同名のファイルの有無によって場合分けすることにより，高次のパスを用いることによって検証が可能であると思われる．

6 結論と展望

本発表では複数のファイルを持つリポジトリに対するパッチも単一ファイルと同様に定式化することを示した．今後は単一のファイルの時の実装への翻訳 `interp` や複数のパッチのマージ `merge` について議論し，Agda において検証を行いたい．本稿では UA は直接は現れなかったが，`interp` を議論するときには本質的にその解釈が重要になる．

7 謝辞

本研究は JSPS 科研費 23320008 の助成を受けている．

参 考 文 献

[1] IAS Project, *Homotopy Tyep Theory*, 2014.

[2] P. L. Lumsdaine “Weak ω-category from intensional type theory, *Typed lambda calculi and applications*, 6:1-19,2010.

[3] B. van den Berg and R. Garner, “Types are weak ω-groupoids, *Proc. of the London Mathematical Society*, 102(2):370-394, 2011.

[4] C. Anguiuli, E. Morehouse, D. R. Licata and R. Harper, “Homotopy Patch Theory (Expanded Version)“, *Proc. of ICFP2014*, 2014.