存在型と部分型をもつオブジェクト計算モデル*

1M－4　村山 尚† 児玉 靖司‡ 原田 賢一†

慶應義塾大学 大学院 計算機科学専攻†　東京理科大学 理工学部 情報科学科‡

1 背景

プログラミング言語は，**型付き言語**と**型無しの言語**に，大別できる．型付き言語は，静的型検査によって型誤りを起こさないことが保証される**強く型付けされた言語**と，型誤りを起こす可能性のあるプログラムも型検査に合格してしまう弱く型付けされた言語とに分けられる．強く型付けされた言語では，型に関して安全で，信頼性の高いプログラムを作ることが可能である．型無しの言語では，型を明示的に指定する必要がないので，プログラムが簡潔に記述でき，変更などに対しても柔軟に対応できる．

強く型付けされた言語と型無しの言語の両方の長所を実現するために，**型推論**が考案された．型推論とは，プログラム中に現われる，非明示的な型に関する情報を使って，式の持ち得る型を厳密な推論規則によって，決定することである．型推論を行うことにより，明示的に型を指定しなくても，強い型検査を行うことができ，実行時の型誤りが起きないことが保証できる．

2 目的

オブジェクト指向言語は，データ抽象と継承の概念をもち，プログラムが扱うデータ間の複雑な関係を構造化し，それらの振舞を定義する機能を提供している．さらに，オブジェクト指向言語は，異なる型の値を取れるオブジェクトを定義することができ，多相性を実現している．しかし，多くのオブジェクト指向言語では，型理論による裏付けをもたずに，多相性を実現しているため，弱く型付けされた言語か型無し言語として実現されている．

本研究の目的は，オブジェクトを表現できるモデルに，型推論を取り入れるとにより，オブジェクト指向言語のもつプログラミングの簡潔性に関する利点，強く型付けされた言語の信頼性に関する利点，さらに型無し言語の柔軟性を兼ね備えた計算モデルを提案することである．

3 オブジェクト計算モデル λ^{obj}

本研究では，データの抽象化と継承の概念をもち，オブジェクトを自然に表すことのできる計算モデルを**オブジェクト計算モデル**と呼ぶ．型推論を行う計算モデルに，データ抽象と継承の機能を導入したモデルは，型推論を行うオブジェクト計算モデルと考えることができる．本研究では，型推論を行う言語MLのモデルである，型付きλ計算モデル + let式 に対し，存在型，部分型，拡張レコード操作を導入したオブジェクト計算モデル λ^{obj} を提案する．オブジェクトはレコードで表され，データ抽象と継承は，それぞれ存在型と部分型の概念によって表現される．

3.1 存在型

抽象データ型でのカプセル化は，スコープの調整による方法と，存在型を使用する方法に大別できる．

抽象データ型は，状態変数とそれを参照することのできる関数の組として表現される．スコープを調整する方法は，抽象データ型の状態変数のスコープを，その抽象データ型に付属する関数だけに限定する．そのため，抽象データ型の状態変数は，その抽象データ型に属する関数からしか見えず，他の関数は，直接状態変数を参照することができない．ObjectMLでは，スコープを調整する方法を用いてデータ抽象を実現している．そのため，オブジェクトは，再帰レコードとして表現されなければならないという制限がある．例えば，一次元上の点を表し，座標の取出しと書込みができるオブジェクト $Point$ の型は，次のように表される．

$$Point : \mu\beta\{get : Int, put : Int \rightarrow \beta\}$$

ここで，β は再帰型である $Point$ 自身の型を表し，get, put はそれぞれ座標の取出しと書込みを行うメソッド表す．内部変数は，スコープを調整されているので型には表れない．put の様に内部状態を変更するメソッドは，必ず自分自身の型を返すように記述する必要がある．

---

* A Object Calculus Model with Existential Type and Subtype

Hisashi Murayama†, Yasushi Kodama‡, Ken'ichi Harada†

† Department of Computer Science, Keio University, 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223, JAPAN

‡ Department of Information Sciences, Science of University of Tokyo, 2641 Yamazaki, Noda-City, Chiba 278, JAPAN

これに対して**存在型**[1]は，抽象データ型の内部構造ではなくその型を隠蔽する．例えば前に示した $Point$ オブジェクトの型は，存在型を用いて次のように表される．

$$Point : \exists\alpha.\{state : \alpha, get : \alpha \rightarrow Int,$$
$$put : \alpha \rightarrow Int \rightarrow \alpha\}$$

ここで，α は存在型である．ある型 α が存在し，それを使って上のように表すことができる値は，すべて $Point$ 型である．存在型は，型変数 α がある型に束縛されていることを保証しながら，その型を隠す役割をする．型 $Point$ は，$state$ というフィールドをもつことは分かっても，その型を特定することができないので，直接 $state$ フィールドの値を参照することはできない．

型システムに対してデータ抽象の概念を導入するためには，再帰型を使うよりも，存在型を使用する方がより自然である．そのため，本研究では存在型を用いてデータ抽象を行う．

3.2 部分型

継承は，再利用性と多相性が密接に関係しあって実現されている．多くのオブジェクト指向言語では，多相性によってサブクラスのインスタンスは，スーパークラスのインスタンスと同様に扱うことができる．このため，スーパークラスのインスタンスに対して適用可能な関数やメソッドは，同様にサブクラスのインスタンスに対しても適用できる．スーパークラスのメソッドをサブクラスで再利用することができる．

継承を表現するために**部分型**という概念を導入する．"型 S が型 T の部分型である" とは，型 S に属するすべてのデータを型 T に属するデータと見なすことができることをいう．部分型の定義を論理式で記述すると，次のように表すことができる．

$$(\forall t \in T, P(t) \Rightarrow \forall s \in S, P(s)) \Rightarrow T \succeq S$$

ここで，T, S は型，t, s はそれぞれの要素，P は論理式を表す．このため，型 S が型 T の部分型ならば，型 T に対して適用できる関数は，型 S に対しても適用できる．継承は部分型で表現することができる．

3.3 オブジェクトの表現方法

本研究で提案するオブジェクトモデルでは，オブジェクトはレコードによって表現される．　レコードとは，次のような構造をもつ，フィールド (ラベル l_i と値 v_i の組) の直積である．

$$\{l_1 : v_1, \cdots, l_n : v_n\}$$

レコードは通常の直積とは異なり，フィールドの順序は可換である．レコードは，一つのレコードの中に，同名にラベルをもつフィールドを複数含むことができない．

オブジェクトをレコードで表現するためには，レコードの特定のフィールドの値を取り出す操作やレコードの特定のフィールドの値を更新する操作が必要になる．しかし，現在型推論を行う言語の多くは，このような**拡張レコード操作**に対する型推論を行うことができない．

そのためオブジェクト計算モデルには，このようなレコード操作を拡張する必要がある．

4 計算モデルの比較

型推論を行なう計算モデル，およびそのモデルを使用した言語を表1に示す．ObjectML は，ML で使われているモデルを大幅に拡張し，レコードとその部分型を自然に表せるレコード計算モデル[2]を使用している．そのモデルだけでは存在型を扱えないため，データ抽象を表現できないが，言語でスコープを調整することによりデータ抽象を表現し，オブジェクトを扱うことができる．

表 1: 計算モデルの比較

モデル	言語
型付きλ計算モデル + *let*式	ML
型付きλ計算モデル + *let*式 + 存在型	ExML
型付きλ計算モデル + *let*式 + 拡張レコード操作	ΠML
型付きλ計算モデル + *let*式 + 拡張レコード操作 +制限された部分型	ObjectML

5 まとめ

本研究では，ML の計算モデルである型付き λ 計算モデル + *let*式 に，存在型，部分型，拡張レコード操作，を導入し，オブジェクト計算モデル λ^{obj} へと拡張した．λ^{obj} は，オブジェクト指向言語のもつプログラミングの簡潔性に関する利点，強く型付けされた言語の，型に関する信頼性，さらに型無し言語の，変更に対する柔軟性をもつ．

参考文献

[1] K.Läufer and M.Odersky : Polymorphic Type Inference and Abstract Data Types, ACM Transaction on Programming Languages and Systems, Vol.16, No.5, pp.1411∼1430

[2] A Ohori : A Polymorphic Record Calculus and its Computation,
ftp://ftp.kurims.kyoto-u.ac.jp/pub/paper/member/ohori/recordcalc.dvi