論文の要旨

A Design and Implementation of Mixin-Based Composition in Strongly Typed Object-Oriented Languages
(強く型付けされたオブジェクト指向言語における mixin 合成の設計と実現)

紙名 哲生

Java をはじめとする多くのオブジェクト指向言語では，プログラムを再利用するための言語機構として単一継承の機構を提供している．しかし単一継承機構では，「複数のスーパークラスに対して行う共通の拡張や修正」を行うモジュールを作成することができない．一方でそのようなモジュールは，Common Lisp のオブジェクト指向的拡張である Flavors や CLOS において，mixin の機構を用いることによって実現されてきた．mixin とは，スーパークラスをパラメータ化することによって抽象化されたクラスのことである．mixin を他のクラスや他の mixin と合成してそのパラメータを埋めることによって，mixin の利用者が，その mixin がどのクラスや mixin を継承するかを決めることができる．ただし CLOS などには静的な型システムが存在しない．本論文では，Java のような型付オブジェクト指向言語における mixin 機構の実現方法について提案する．

本論文で扱われる問題は以下の通りである．まず，既存のオブジェクト指向言語に対する mixin 機構の導入を提案する．このとき，プログラミング言語が持つべき基本的な性質である型健全性について調べる．型健全性とは，コンパイラが受理したプログラムは，実行時にある種のエラーを起こさないことが保証されるという性質である．次に，mixin 機構の実装方法について考える．実行効率と既存のプラットフォームとの互換性がここでの主な論点である．次に，mixin 機構が引き起こす新たな問題点について考察する．mixin 機構には，mixin で定義されたメソッドがスーパークラスのメソッドを非意図的に上書きしてしまい，危険な振る舞いを引き起こす可能性のあることが指摘されており，この問題点を解決する必要がある．最後に，mixin 機構が，近年新たに提案されてきた他の言語機構とどのような関わり方をするかについて調べる．

具体的に，本論文では Java に mixin 機構を導入して拡張したプログラミング言語 McJava

の設計を行った．構文としては，従来の Java の文法に加えて，mixin を宣言するための構文と，mixin を他のクラスや他の mixin と合成するための演算子が新たに加わっている．これら mixin 宣言，mixin 合成，それに mixin の名前を型として扱える mixin 型の機構に加えて，McJava では mixin を mixin と合成し，新たな mixin を作る高階 mixin の機構と，mixin 合成間における柔軟な部分型機構を提供している．特に後者の部分型の機構は，従来のオブジェクト指向言語が部分型関係をクラスの間の継承関係でしか決めていなかったのに対して，本研究では継承関係の間に部分型関係が決められ，継承関係の間に別のクラスが混ざったものでももとの継承関係の部分型として認められるという点で特徴的である．これによりプログラミングの柔軟性が高まる．

型健全性の性質は，McJava のサブセットである Core McJava 上で調べた．Core McJava は，Java のサブセットである FJ の拡張として作られている．FJ は，Java の型システムを特徴づけている基本的な機構のみを持つ小さな言語で，McJava のような Java の拡張の型システムの持つ性質を調べるときに広く用いられている計算体系である．Core McJava は，FJ に，上で述べられた，McJava を特徴づけている言語機構を加えたものである．本論文では，Core McJava が型健全であることを証明した．

次に，McJava の実装方法について述べる．McJava コンパイラの実装は，Java へのソースコード変換によって行った．本論文で提案された変換方法は，Java のみで書かれたプログラムには変換中全く変更が加えられない．また mixin 合成は，それと同じ深さを持つ Java の継承関係へと変換される．以上のことから，本論文で提案された実現方法により，Java と同程度の実行効率が McJava でも保証されることが示された．Java に変換されることから，既存の Java 仮想機械との互換性も保証される．さらに，Java のみで書かれたプログラムには変更が加えられないことから，既存のライブラリとの互換性も理論的には保証される．

次に，非意図的な上書きの問題の解決方法について述べる．本論文では，この問題を解決するために新たなメソッドディスパッチの機構を提案した．通常の Java におけるメソッドディスパッチ機構は，常にメソッド呼び出しのレシーバの，実行時のクラス継承の一番深いところからメソッドが検索される．それに対して本論文で提案された機構では，レシーバの静的な型を実行時に保存し，メソッドはこの静的な型を起点として探される．これによって，レシーバ（実行時オブジェクト）に複数の同じシグネチャを持つメソッドが存在する場合でも，静的な型情報を用いてそれらのうちから適切なメソッドが呼び出されることになり，非意図的な上書き（片方がもう片方を完全に隠すこと）はなくなる．特に McJava の場合，上で述べた柔軟な部分型の機構から，静的に分かるスーパークラスが実行時のスーパークラスと異なる場合がある．つまり，super に対してメソッドを呼び出すときに，静的に分かるスーパークラスのほうへメソッド検索の制御を移すことが必要になる場合がある．本研究ではこの機構を Java へのソースコード変換として実装し，どのように変換されるかを具体例を用いて示した．

最後に，mixin 機構と他の言語機構との関連について述べる．Java を拡張する研究には数多くのものがあるが，本論文では特に，それらの中から総称型及び ThisType の機構と mixin 機構との関わり方について調べた．総称型とは，型定義（クラスや mixin 等の定義）のなかで用いられている型情報を，パラメータによって抽象化したもののことである．ThisType とは，this の型のことである．総称型の機構は実際 Java 言語の正式なリリースに取り入れられたものである．ThisType は Java には今のところ取り入れられてはいないが，モジュールの拡張を

考えるときに重要な役割を果たすものである．本論文ではこれらの言語機構を McJava に取り入れた場合，表現力がどれだけ増し，型健全性の性質はどうなるかについて議論した．結果，プログラムのモジュラリティが高まり，表現力が増すことが例題を用いることによって示された．型健全性に関しては，モジュールを構成する内部要素間の部分型関係についてある種の制約を加えなければ保証されないことが示された．ただしこの制約は，実用上問題ないものであると考えられる性質のものである．

以上，本論文では Java のような型付オブジェクト指向言語における mixin 合成の実現方法について研究した．そして言語が型健全性の性質を持つこと，処理系が妥当な実行効率と後方互換性の性質を持つこと，非意図的なメソッド上書きの問題点が解決されることが示された．さらに，mixin 合成と総称型や `ThisType` との関わり方を調べた結果，これらを含む言語のモジュラリティが高まることがわかった．以上のことから，本論文で提案された手法は Java における mixin 合成の設計及び実現方法として有力なものである．