動的型付けプログラミング言語 HSP における静的型付け導入に関する研究

鈴木　颯太

プログラミング言語における型付けシステムは、開発の効率性と安全性に多大な影響を与える要素である。JavaScript や Python などの動的型付け言語は、その柔軟性とスクリプト言語としての扱いやすさにより広く普及している。一方で、C++ や Java などの静的型付け言語は、コードの整合性の検証やリファクタリングの容易性といった利点により、大規模なソフトウェア開発において特に重要な役割を果たしている。

近年、動的型付け言語に静的型付けの利点を取り入れる試みが注目を集めている。その代表例が JavaScript により強固な型の概念を追加する TypeScript であり、トランスコンパイラを介して JavaScript コードを生成することで、型システムを導入しつつ JavaScript の実行環境との互換性を維持している。このような仕組みは、従来のプログラミング言語においてコンパイラに対し実装が必須であったコード生成器や、それに伴うコード最適化や複数ターゲット向けに関する実装などを省略し、従来よりも言語仕様に対する実装や議論、考察に注力することができ、かつ元の言語処理系が持つ実行環境を活用することができる。

本研究では、動的型付け言語である Hot Soup Processor（以下、HSP）を対象とし、TypeScript のようなトランスコンパイラを用いて静的型付けを導入する方法を提案する。HSP は、日本発のプログラミング言語として、初学者や個人開発者に親しまれてきた。その特長は、GUI を構築するための簡潔な環境と直感的な文法にあり、ゲームやツール開発といった分野で広く使用されている。しかし、大規模プロジェクトや長期的なメンテナンスを伴う開発において、動的型付けによるバグの潜在的リスクや、型安全性の欠如が課題となる場合がある。TypeScript のようなトランスコンパイラを使用する手法を採用し、ランタイムの再構築をすることなく、既存の実行環境を損なわずに型安全性を提供することで、HSP の持つ特徴を維持しながら開発体験の向上を実現することができる。

本研究では、Go によるトランスコンパイラの実装を行なった。入力として今回実装した記法である TSP を受け付け、HSP ソースコードを出力する。本研究の評価として、TSP を用いて記述したソースコードと、そのソースコードをトランスコンパイルし、HSP で実行した際に期待される出力結果を照らし合わせたテーブルテストを実施した。内容としては HSP の基本的な文法の大部分を網羅する他、TSP の記法が正しく実装されていることを確認するためのものであり、実行の結果ほとんどが仕様を満たす実装ができていたが、一部意味解析で誤った結果を出力することがあった。

残された課題として、エラーレベルの設定や、未実装である HSP 構文、Shift-JIS への対応や人間による新たな記述が開発体験向上にどのように影響するかの検証などが挙げられる。

（指導教員　　阪口　哲男）