発表概要

具象コード実行をともなう抽象解釈器の実行モデル

馬谷 誠二[1,a)]

2021年3月17日発表

言語仕様が厳密には規定されていないプログラミング言語で書かれたプログラムの性質を知りたい場合，言語処理系（リファレンス実装）が生成した低水準コードの命令列を静的解析するという手段を用いることができる．しかし，高水準言語からコンパイルされた低水準コードの命令列に対する解析では，生成されるコードやランタイムの複雑さのせいで有用な情報を十分に取得できないことが多い．そこで本発表では，コンパイルされた低水準コードの直接実行（「具象解釈」）と抽象解釈を混ぜながら実行する抽象解釈器の実行モデルを提案する．純粋な抽象解釈だけでは正確に解析できない一部のコードのみを選択的に具象解釈することで，解析精度の低下を回避できることが提案モデルの特徴である．どのような粒度・頻度で分割されたプログラムであっても，2 つの解釈モジュールの実行環境をシームレスに接続することにより，プログラムを正しく実行（解析）することが可能となっている．また，アプリケーションデータの相互変換においても，個々の解析に依存するデータ型の抽象ドメインの定義を系統的に用いる仕組みが組み込まれており，2 つの解釈モジュール間で，メモリ上のエイリアス関係を保ちつつ任意のデータ構造を適切に相互変換可能である．

Presentation Abstract

Execution Model of an Abstract Interpreter that Intertwines with Concrete Code Execution

Seiji Umatani[1,a)]

Presented: March 17, 2021

Suppose we want to know a program's properties written in a programming language whose specification is not strictly defined. In that case, we can use static analysis of the instruction sequence of low-level code generated by its language processor (reference implementation). However, the analysis of the instruction sequence of low-level code compiled from a high-level language often fails to obtain enough useful information due to the complexity of the generated code and the runtime. This presentation proposes an abstract interpreter's execution model that mixes direct execution ("concrete interpretation") and abstract interpretation of compiled low-level code. The proposed model avoids the degradation of analysis accuracy by selectively performing concrete interpretation only for some code fragments that cannot be analyzed accurately by pure abstract interpretation alone. Even if the target program is split at an arbitrary granularity and frequency, it is executed (analyzed) correctly by seamlessly connecting the two interpretation modules' execution environment. Besides, the systematic use of abstract domain definitions for data types that depend on individual analyses is incorporated in the mutual conversion of application data, allowing appropriate mutual conversion of arbitrary data structures between the two interpretation modules while maintaining alias relations in memory.

This is the abstract of an unrefereed presentation, and it should not preclude subsequent publication.

[1] 神奈川大学理学部情報科学科
Department of Information Sciences, Faculty of Science, Kanagawa University, Hiratsuka, Kanagawa 259–1293, Japan
[a)] umatani@kanagawa-u.ac.jp