QA レポート

システム情報工学研究科 コンピュータサイエンス専攻
博士課程前期 1 年 201520621 大石純平

平成 27 年 10 月 30 日

- 研究題目: 多段階 let 挿入を行うコード生成言語の設計
- 主任指導教員: 亀山幸義
- 発表日時 2015/10/22

1 質問

安全性と最適化について，let 挿入を安全に行えるというのは分かったが，最適な位置にどうやって置くのか．安全性と最適化の両立はどのように行うのか．

let 文というものを適切な？安全な位置に挿入するときに，shift0/reset0 というものを使うと，それを安全な場所に挿入できるということだと理解したが，安全であるが最適な場所かは分からないということか

性能のほうで最適な位置を見つける技術はまだないということか

1.1 発表時の回答

Sudo らの型システムのアイデア [1] を用いる事によって，let 文を適切な位置に挿入することを実現したい．

はい．安全性のみを考えている．

1.2 改善した回答

各環境に応じた最適なコード（実行性能の高いコード）を自動的に生成するわけではない．端的にマルチステージの目的を言うと，コンパイラなどが自動的に最適化できないような人間の手でアドホックに行う最適化に対しての安全性を保証したいということである．つまり，その最適化を行う部分は人間の手で書くということになる．表現力を上げた体系（今回の場合だと例えば，多段階 let 挿入が行える体系）において本研究の型システムが生成後のコードに対して型エラーを起こさないようにすることで，生成後のコードの安全性が保たれる．

- コード生成言語に shift0/reset0[2] を加えることで多段階 let 挿入等ができるので表現力があがる．
- すると，shift0/reset0 をプログラマが用いることによって，多段階 let 挿入などの計算エフェクトを含むコードを書くことができるようになる．
- その際，本研究の型システムによってそのコードが型エラーを起こさなければ，生成後のコードが安全にコンパイル，実行できることを保証する．

1.3 コメント

ステージングは最適化を自動的に行うようなものではなく，最適化を行うのはあくまでも我々の言語を使う人であるというステージングの立場を共有する必要があった．

2 質問

shift0/reset0 を入れることで let 挿入が可能となるそうだが，それが，効率のよいプログラムを記述するための表現力を高めつつその安全性が保証されたということにどう結びつくのか？(私がもういちど聞き返す）shift0/reset0 を入れることのメリットを教えてほしい．

2.1 発表時の回答

(let 挿入を行うことのメリットは何かという質問だと勘違いをして) assert 挿入の例を持ちだして，let 挿入をすることのメリットを答えた．

2.2 改善した回答

shift0/reset0 を入れることだけでは安全性には貢献しない．shift0/reset0 を用いることによって得られる計算エフェクトを含むコード（たとえば，多段階 let 挿入）に対しての安全性を保証する型システムを作る必要がある．

3 質問

前の質問 1 に引き続いての質問．安全性をひとまず保証しようとしている？ただ精度（性能）を安全性もそのままにしながら精度（性能）を保証するにはどういう風にしていくのか？例えば，shift0/reset0 でなくて，shift/reset は安全性と精度（性能）はうまく保証されているのか？そうであるならば，同じような手法を使って精度（性能）を保証するということはできないのか？

安全性も保証しながら精度も高いということの意味を教えてほしい．

3.1 発表時の回答

shift/reset で表現できないことがある．それを shift0/reset0 では表現できる．その一つの例が多段階 let 挿入である．その多段階 let 挿入を用いることで安全性を保証しつつ効率のよいコードも書ける．

精度が良いってのは最適なコードということ？(実行性能の高いコード？）ここで作りたいものというのは，最適なアルゴリズムを考えるのは，私ではなく使う人である．コード生成言語を使うことによって多数のパラメータに応じた環境に特化したコードが生成される．そのアルゴリズム自体は使う人が考える．我々は安全性は保証するというのが立ち位置である．

3.2 改善した回答

性能を上げるための最適化の話をしていたわけではない．表現力を上げた体系の安全性を型システムで抑えるということを行いたいということを話していた．

ほとんどの人にとってはステージングでなくコンパイラを使うと思うが，ドメインに特化した高性能なコードを生成したいプログラマに対して自由にプログラムを書かせたい．コンパイラが自動的に最適化できないような，アドホックな人間の手で行うような最適化を我々の体系の言語で書かせる．なおかつその表現力の上がった体系の安全性を型システムで保証する．

shift/reset を含むステージングの体系については，亀山ら [3] が shift/reset の影響が変数スコープを超えることを制限する型システムを構築し，安全性を厳密に保証した．彼らの仕事は本研究でも参考にする．

3.3 コメント

適切な位置に let 挿入を行うということを最適な位置に let 挿入を行うと口頭で話してしまったために混乱させてしまったように思う．全自動で最適化されたコードが生成されるわけではない．質問 1，2 と同様にステージングの立場を共有するべきだった．

参考文献

[1] 須藤悠斗, Oleg Kiselyov, 亀山幸義. コード生成のための自然演繹. 日本ソフトウェア科学会第 31 回大会, 2014 年 9 月, 名古屋大学. PPL4-4, 9 ページ.

[2] Marek Materzok and Dariusz Biernacki. Subtyping delimited continuations. *SIGPLAN Not.*, Vol. 46, No. 9, pp. 81–93, September 2011.

[3] Yukiyoshi Kameyama, Oleg Kiselyov, and Chung-chieh Shan. Shifting the stage: Staging with delimited control. In *Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, PEPM '09, pp. 111–120, New York, NY, USA, 2009. ACM.