様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　５ 年　 ５ 月 ２５ 日現在

機関番号：１２６０１
研究種目：若手研究
研究期間：2018～2022
課題番号：１８Ｋ１８０３０

研究課題名（和文）名前呼び計算体型のための型システムによるプログラム検証

研究課題名（英文）Program verification by refinment type system for call-by-name programs

研究代表者
佐藤　亮介（Sato, Ryosuke）

東京大学・大学院情報理工学系研究科・助教

研究者番号：１０８０４６７７

交付決定額（研究期間全体）：（直接経費）　　3,100,000円

研究成果の概要（和文）：研究計画に基づき研究を進め、名前呼び戦略の言語のための検証手法を構築することができた。具体的には、変数に束縛される項がいつ評価されるのかを部分構造型システムを用いることによって管理し、評価されるタイミングによってその項の情報を使えるかどうかを判断することによって名前呼び評価戦略の言語のための健全な詳細化型システムを構築することができた。さらに、値呼び評価戦略の言語のための型システムでは必要ではなかった型ガードという概念を導入することによって十分な表現力を持った型システムを構築することができた。本手法の効果を確かめるため検証器を実装し、評価を行った。

研究成果の学術的意義や社会的意義
本研究は近年注目されるようになってきた関数型言語、特にHaskellやCleanなどの名前呼び評価戦略の言語に対する全自動検証の第一歩である。本研究では、そのような言語の核となる小さな言語に対する全自動検証手法を提案し、実装、実験を行うことができた。本研究を発展させることによってHaskell等のプログラムを全自動で検証することができる可能性がある。既存研究としてHaskellを検証するための手法はあるが、全自動ではないという点が本研究手法との大きな違いである。

研究成果の概要（英文）：Based on the research plan, we were able to progress with the study and develop a verifier for a call-by-name functional language. Specifically, we managed the evaluation timing of terms bound to variables using a substructural type system, and by determining whether the information of the term can be used based on its evaluation timing, we were able to construct a sound refinement type system for the call-by-value language. Furthermore, we introduced the concept of type guards, which was not necessary in a type system of call-by-value language, allowing us to build a type system with sufficient expressive power. To verify the effectiveness of this approach, we implemented a verifier and conducted evaluations.

研究分野：形式検証

キーワード：型システム　名前呼び評価戦略　関数型言語　全自動検証

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

情報科学の発達に伴い、航空管制システムや証券取引システムなど、人名や財産に関わるようなシステムがコンピュータによって制御されるようになってきた。これらのシステムを制御しているソフトウェアに欠陥があると、コンピュータウィルスや不正侵入などの攻撃によって、財産のみならず人命をも失うことになる。そのため、ソフトウェアが正しく動作することを何らかの手段によって保証しなければならない。しかし、現在の開発現場で用いられているテスト実行の繰り返しによるバグ発見手法では、特殊なケースのみに起こるバグを見逃してしまう恐れがある。全てのケースを漏れ無く確かめるために、ソフトウェアの安全性は形式的な手法に基づいて検証されなければならない。手続き型言語で書かれたプログラムに対しては非常に多くの形式手法が提案されているのに対し、近年注目されるようになってきた関数型言語で書かれたプログラム、および、関数型言語の機能を使用した手続き型プログラムの検証に関する研究は、まだまだ発展途上である。特に、Haskell や Clean などの名前呼び戦略の言語に対する検証に関する研究が進んでいない。これらの言語により書かれたプログラムは今後増えることが予想されるため、名前呼び戦略の言語に対する検証手法の開発が望まれる。

２．研究の目的

名前呼び戦略の言語（以下、名前呼び言語）で書かれたプログラムの正しさを保証するため、名前呼び言語に対する自動検証手法を確立する。通常の手続き型言語や値呼び戦略の関数型言語では、関数の引数を先に評価し、その評価結果に関数を適用するが、名前呼び言語では関数の引数が実際に使われるときに初めて評価される。このような値呼び言語のための型システムは、名前呼びの言語に対しては不健全（異常終了するプログラムに対して正常終了するという結果を返すことがある）であることがわかっている。さらに、一般に名前呼びプログラムを値呼びプログラムに変換できることが知られているが、変換後のプログラムは既存の検証手法では検証できない複雑なものとなってしまうため、名前呼びプログラムをそのまま検証する手法が必要となる。

３．研究の方法

名前呼び戦略の関数型言語を対象とした自動検証手法を確立し、その手法に基づいた検証器を作成する。基本的な機能のみを持つ名前呼び戦略の言語に対して、詳細化型システム（refinement type system）を構築する。具体的な対象言語としては、単純型付きラムダ計算に再帰、整数、および、fail を入れた言語を用いる。この言語に対して、名前呼び言語のための詳細化型システムを定義する。型システムのアイデアとしては、対象の項が必ず評価（簡約）されることを表す様相演算子を型に追加することで、不健全な型付けを防ぎつつ、多くのプログラムを検証できるようにする。これにより、変数に束縛される項がいつ評価されるのかが把握でき、健全な型システムが実現できる。提案した型システムを基に、検証器のプロトタイプを実装し、実験を行う。実験結果を解析することによって、検証手法および実装の問題点を洗いし、それぞれの問題点に対して本質的な改良が必要か否かを判断する。本質的な改良が必要であると判断したものを除き、解消する改良を手法および実装に対して行う。

４．研究成果

研究計画に基づき研究を進め、名前呼び戦略の言語のための検証手法を構築することができた。具体的には、変数に束縛される項がいつ評価されるのかを部分構造型システムを用いることによって管理し、評価されるタイミングによってその項の情報を使えるかどうかを判断することによって名前呼び評価戦略の言語のための健全な詳細化型システムを構築することができた。さらに、値呼び評価戦略の言語のための型システムでは必要ではなかった型ガードという概念を導入することによって十分な表現力を持った型システムを構築することができた。型ガードはホーア論理の事前条件のようなもので、型ガードが付いた型を持つ項を安全に評価できるための条件を表すものである。値呼び評価戦略の言語では項が関数適用のタイミングで常に評価されるため、事前条件に相当するものは引数の条件（述語）として記述することができた。対して名前呼び評価戦略の言語では評価のタイミングがそのプログラムの構文からはわからないため、型ガードの導入が必要であることがわかった。

本手法の効果を確かめるために簡単な名前呼び関数型言語に対する検証器を実装し、既存のベンチマークセットおよび自作のベンチマークセットによる評価を行った。表１がその実験結果である。本手法は安全なプログラムに対しては本研究手法では正しく「安全」と判断でき、かつ、危険なプログラムに対しては「安全かどうかわからない」という結果を返した（カラム Ours）。本研究手法を含め詳細化型システムは通常、健全だが完全な型システムではないため危険なプログラムに対しては「安全かどうかわからない」という結果を返すのみとなる。既存研究を基にした Haskell の検証器である Liquid Haskell では、安全なプログラムに対して「安全かどうかわからない」という出力をしたり、場合によっては危険なプログラムを「安全」と判断したりする結果になっている。

本研究の結果をまとめた論文は JIP に採録が決定している[1]。

表 1 実験結果

ベンチマーク	プログラム	Ours		LiquidHaskell		LiquidHaskell-nt	
オリジナル（名前呼びと値呼びで危険性が変わる）	div_intro	Safe	0.020s	Unsafe	0.337s	Unsafe	0.358s
	l-zip	Safe	0.046s	Unsafe	0.327s	Unsafe	0.338s
	collatz	Safe	0.034s	Unsafe	0.318s	Safe	0.318s
	isort_max	Safe	0.158s	Unsafe	0.406s	Unsafe	0.367s
	loop	Safe	0.006s	Unsafe	0.308s	Safe	0.318s
	loop-e	Unknown	0.007s	Unsafe	0.318s	Safe	0.307s
	ack	Safe	9.735s	Unsafe	0.358s	Safe	0.357s
	copy_intro	Safe	0.032s	Unsafe	0.308s	Safe	0.318s
	l-zipmap	Safe	0.053s	Unsafe	0.359s	Safe	0.348s
	l-zipunzip	Safe	0.052s	Unsafe	0.338s	Safe	0.358s
MoCHi [Kobayashi+ 2011, Sato+ 2013]	max	Safe	0.022s	Safe	0.390s	Safe	0.390s
	mc91	Safe	0.043s	Unsafe	0.329s	Unsafe	0.329s
	mc91-e	Unknown	0.032s	Unsafe	0.318s	Unsafe	0.318s
	mult	Safe	0.042s	Unsafe	0.328s	Safe	0.317s
	mult-e	Unknown	0.032s	Unsafe	0.328s	Unsafe	0.317s
	sum	Safe	0.029s	Safe	0.307s	Safe	0.327s
	sum-e	Unknown	0.033s	Unsafe	0.308s	Unsafe	0.317s
	sum2	Safe	0.024s	Safe	0.328s	Safe	0.318s
	sum_intro	Safe	0.030s	Safe	0.317s	Safe	0.317s

参考文献：[1] Ryosuke Sato. Refinement types for call-by-name programs. Journal of Information Processing. （採録決定済み）

5．主な発表論文等

〔雑誌論文〕　計1件（うち査読付論文　1件／うち国際共著　0件／うちオープンアクセス　1件）

1．著者名 Ryosuke Sato	4．巻 -
2．論文標題 Refinement types for call-by-name programs	5．発行年 2023年
3．雑誌名 Journal of Information Processing	6．最初と最後の頁 -
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 有
オープンアクセス オープンアクセスとしている（また、その予定である）	国際共著 －

〔学会発表〕　計1件（うち招待講演　0件／うち国際学会　0件）

1．発表者名 Ryosuke Sato
2．発表標題 Refinement types for call-by-name programs
3．学会等名 第143回情報処理学会プログラミング研究発表会
4．発表年 2023年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

－

6．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考

7．科研費を使用して開催した国際研究集会

〔国際研究集会〕　計0件

8．本研究に関連して実施した国際共同研究の実施状況

共同研究相手国	相手方研究機関