様式Ｆ－１９

科学研究費助成事業（学術研究助成基金助成金）研究成果報告書

平成２５年　６月１０日現在

機関番号：12601
研究種目：若手研究(B)
研究期間：2011～2012
課題番号：23700029
研究課題名（和文）　動的モジュールを持つプログラミング言語におけるソフトウェア検証機構

研究課題名（英文）　Verification mechanism in programming languages with dynamic modules

研究代表者　紙名　哲生（KAMINA TETSUO）
東京大学・大学院教育学研究科・特任助教
研究者番号：90431882

研究成果の概要（和文）：文脈に依存したプログラムの振る舞いをモジュール化し、イベント駆動で動的にそれらを合成できる文脈指向プログラミング言語 EventCJ を対象に、動的モジュール合成のための言語機構および基礎理論に関する研究を行った。まず、EventCJ の核言語 Featherweight EventCJ を構築し、その操作的意味論を形式的に定義した。それと同時に、事例研究により文脈とモジュールの対応関係を明らかにし、新たな一貫性保証のための言語機構として合成層を発明し、それを EventCJ プログラムへの変換として実現した。これにより、EventCJ に備わるモデル検査機構を用いることにより、モジュール合成に関する性質を検証することが可能になった。さらに、Featherweight EventCJ を拡張して合成層の意味論及び元の Featherweight EventCJ への変換を形式的に定義し、その変換が正しいという定理を証明した。これにより、合成層に関する性質を EventCJ のモデル検査機構で検証した結果が、理論的に正しいということが保証された。

研究成果の概要（英文）：Language mechanisms for dynamic module composition and their theories are studied through extending EventCJ, a context-oriented programming (COP) language with event-based per-instance layer transition. First, Featherweight EventCJ, a core calculus for EventCJ, is developed to formally define the operational semantics of EventCJ. Next, a new COP mechanism composite layers is developed to enhance the expressive power of representing relationship between contexts and layers (implementations of context-dependent behavior). Composite layers are implemented as a translation into EventCJ programs. By this translation, it is enabled to verify some properties about composite layers by applying the model-checking mechanism equipped with EventCJ. Finally, this translation is formally studied by extending Featherweight EventCJ with composite layers, and the theorem about the soundness of this translation is proven. Thus, it is ensured that the result of application of the model-checking mechanism in EventCJ is correct.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
交付決定額	2,500,000	750,000	3,250,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウェア
キーワード：文脈指向・合成層・EventCJ・プログラム変換・操作的意味論

１．研究開始当初の背景

現在、ソフトウェアは大規模なサーバから小型の端末や自動車・家電の制御に用いる超小型のコンピュータに至るまで、日常生活のあ

らゆる場面に存在する。市場での熾烈な競争から、ソフトウェアの開発コストをいかに下げ、納期を短縮するかが重要となっている一方で、ソフトウェアは社会基盤として欠かせないものであるため、その安全性や信頼性の問題が非常に重要である。そのため、ソフトウェアのモジュール化に関する研究や、安全性の保証に関する研究は、信頼できる情報基盤を構築する上において重要な研究テーマである。

一方で、ここ最近、ソフトウェアを取り巻く環境は大きく変化している。モバイル技術やコンピュータの小型化にともない、いつでもどこでも、実世界に埋め込まれた機器が状況の変化に合わせてサービスを提供するユビキタスコンピューティングの実現が可能になってきた。また開放型分散システム環境や移動計算機環境の普及により、ソフトウェアは多様な実行環境とその動的な状態変化に対応することが求められている。こうした環境の変化だけでなく、例えばユーザインターフェイスの分野でも、ユーザの意図に合わせて適応的に振る舞う UI が注目を集めており、文脈（コンテキスト）に応じた動的な振る舞いの変更は、ソフトウェアが備えるべき機能の一つとして近年急速に認識され始めている。

これらのことから、コンテキストアウェアネス（システムが状況の変化にあわせて適応的に振る舞う性質）の実現は、今日のソフトウェア開発技術において重要な課題である。とくにプログラミング言語の分野では、それを実現するために、開発時やリンク時・ロード時ではなく、実行時に動的にモジュールを合成する仕組みや、動的に活性なモジュールを切り替える仕組みに関する研究が盛んにおこなわれるようになってきた。例えば、古くはオブジェクト指向プログラミングにおける自己反映計算の分野においてこうした取り組みは行われていた。その後、動的アスペクト指向や文脈指向の考え方が広まるに従って、ソフトウェアのモジュールを動的にシステムに組み込むことを可能にするプログラミング言語が生まれてきた。

一方で、システムの安全性を保証するためには、モジュールを組み込んだ結果を事前に予測し、エラーを未然に防ぐ必要がある。とくに、開発効率の高さが要求される今日のソフトウェア開発においては、こうした検証は静的に行えるのが望ましい。しかし、動的に振る舞いが変わるようなソフトウェアを静的に検証することは、これまでのプログラミング言語では難しい。よって、動的モジュール合成を実現するプログラミング言語におけるソフトウェア検証機構を研究することは、学術的に見ても、また社会的な重要性から見ても、非常に意義のあることである。

２．研究の目的
本研究課題では、動的モジュール合成を実現するプログラミング言語の中でも、特に文脈指向プログラミング言語に的を絞り、新たな言語機構や言語の形式的意味論を探求することによってプログラムを検証する技術を研究する。具体的には、次の項目を研究する。

(1) 状態遷移モデルに基づく新たな文脈指向核言語の構築。文脈の変化は状態遷移として捉えられるため、それをモデルに含む、文脈指向プログラミング言語の研究において理論的基礎を与える各言語を構築する。

(2) モジュール合成のための言語機構の探求。現実の問題の中から、現在の文脈指向プログラミングモデルの問題点を把握すると同時に、それに対して有効な解決手段となる新たなモジュール合成機構を探求する。

(3) モジュール合成における性質を検証するための機構の構築。動的モジュール合成において保証したい性質を検証するためのアルゴリズムを構築し、そのアルゴリズムが正しいことを理論的に保証する。

３．研究の方法
(1) 研究の目的(1)を達成するため、対象とする既存の文脈指向プログラミング言語モデルを選定する。本研究では対象とするプログラミング言語を EventCJ[1]とした。EventCJ は、イベント駆動で行われるインスタンスごとの状態遷移に基づく文脈変化を表現することができ、その記述を宣言的に行える。そのため、既存の動的モジュール合成機構の中でも、より高い柔軟性とモジュラリティを持ち、本研究の対象として適切である。しかし EventCJ スタイルの文脈遷移を表現できる文脈指向プログラミングの計算体系は存在しなかった。そこで既存の文脈指向言語の核言語のひとつである ContextFJ[2]を EventCJ 独自の機構を取り込むことによって拡張し、新たな計算体系 Featherweight EventCJ を構築した。

(2) 事例研究を通して EventCJ の動的モジュール合成機構の問題点を調査し、それを解決する新たな動的モジュール合成機構である合成層を考案した。合成層はプログラマが明示的に遷移を指定するための層（原子層といい、文脈に対応する）と、原子層の状態の組み合わせによって暗黙的に活性化・非活性化が行われる層を分けるための機構である。

(3) 合成層に関する性質を検証するため、EventCJ に備わるモデル検査機構を用いる。そのために、合成層から、それと等価な合成層なしの EventCJ へと変換するプログラム変換のアルゴリズムを構築した。また、このアルゴリズムの正しさを保証するため、研究の方法(1)で構築した Featherweight EventCJ をもとにこのアルゴリズムを形式化した。具体的には、Featherweight EventCJ を合成層によって拡張し、この拡張からもとの Featherweight EventCJ への変換を形式的に定義し、変換の正しさを保証する定理を記述してその定理を証明した。

４．研究成果

(1) 文脈指向プログラミングの新たな計算体系である Featherweight EventCJ を構築した。これにより、従来では存在しなかったインスタンスベースの層遷移機構を持つ核言語が初めて実現された。また、EventCJ の意味論の中でも複雑な部分、とくに複数のイベントが同時に生成された場合や、適用可能な動的モジュール合成の規則が複数存在した場合において、EventCJ がどのように振る舞うかが明らかにされた。具体的には、複数のイベントが同時に生成された場合や、イベントが一つだけの場合でも複数の動的モジュール合成の規則が一つのイベントに紐付けられていた場合は、適用可能なイベント生成時に適用可能な動的モジュール合成の規則が複数存在することになる。このとき、EventCJ ではそれらが逐次実行されるのではなく、並列に実行される。逐次実行の場合は、先行する規則の適用結果が、次の規則の適用結果に影響を及ぼし、直列化の順序に関してプログラムの振る舞いが変わったりするので望ましくない。EventCJ の言語仕様ではそのようなことが起こらないこと、また言語実装の際には、並列適用される動的モジュール合成の規則を（CPU が一つの場合などに）直列化する際は、それらが互いに影響し合わないようにする必要があることが、Featherweight EventCJ の構築により示された。

(2) 合成層を実現した。これにより、EventCJ における、文脈とそれに対応する層の関係が複雑化し、同じ文脈に関するコードが各所に散らばったり異なる文脈に関するコードが混ざり合ったりする問題（scattering and tangling）が解決された。また、合成層の実現方法として、合成層から EventCJ へのプログラム変換手法を実現した。これにより、合成層に関する性質（例えばある合成層 A と B は同時に活性化しない、あるいはある合成層 C が活性化されている間は必ず D も活性化されている必要がある、などのような性質）の検証は、EventCJ に備わるモデル検査手法を用いることで可能となった。さらに、合成層の活性化は、モデルの上ではその合成層が依存する他の層が活性化した際に、その合成層を活性化すべきかどうかをチェックして必要なら活性化するというステップを踏むが、EventCJ へのプログラム変換を採用することで、そのようなステップは実行時には行われなくなり、効率的な合成層の活性化が実現された。

(3) Featherweight EventCJ を合成層によって拡張した新たな計算体系 FECJ° を構築した。これにより、合成層の活性化に関する操作的意味論が初めて構築された。また、FECJ° の基本的な性質として、ある合成層が依存する他の層が活性化している間は、必ずその合成層は活性化されているという定理を構築し、その定理を証明した。さらに、合成層から EventCJ へのプログラム変換を、FECJ° から Featherweight EventCJ への変換として形式化し、FECJ° の操作的意味論における簡約規則を実行してから Featherweight EventCJ に変換した場合と、Featherweight EventCJ に変換してから Featherweight EventCJ の操作的意味論における簡約規則を実行した場合とで、結果が等しくなる（つまり合成層から EventCJ へのプログラム変換が意味論的に正しい）という定理を構築し、その定理を証明した。この証明により、合成層に関する性質を EventCJ のモデル検査機構で検証した結果が正しいということが、理論的に保証された。

参考文献：

[1] Tetsuo Kamina, Tomoyuki Aotani, and Hidehiko Masuhara, EventCJ: A Context-Oriented Programming Language with Declarative Event-based Context Transition. In AOSD.11, pp.256-264, 2011.
[2] Robert Hirschfeld, Atsushi Igarashi, and Hidehiko Masuhara, ContextFJ: A Minimum Core Calculus for Context-oriented Programming. In FOAL'11, 2011.

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 4 件）

(1) Tomoyuki Aotani, Tetsuo Kamina, and Hidehiko Masuhara. Featherweight EventCJ: A Core Calculus for a Context-Oriented Language with Event-Based Per-Instance Layer Transition. In COP'11, pp.1:1-1:6, 2011, 査読有り

(2) Tetsuo Kamina, Tomoyuki Aotani, and Hidehiko Masuhara. Bridging Real-World Contexts and Units of Behavioral Variatios by Composite Layers. In COP'12, pp.4:1-4:6, 2012, 査読有り

(3) Tetsuo Kamina, Tomoyuki Aotani, and Hidehiko Masuhara. Introducing Composite Layers in EventCJ. IPSJ Transactions on Programming, 6(1), pp.1-8, 2013, 査読有り

(4) Tetsuo Kamina, Tomoyuki Aotani, and Hidehiko Masuhara. A Core Calculus of Composite Layers. In FOAL'13, pp.7-12, 2013, 査読有り

〔学会発表〕(計 6 件)
(1) 紙名哲生, 青谷知幸, 増原英彦, 玉井哲雄. ユースケースを用いた文脈指向ソフトウェア開発. ソフトウェアエンジニアリングシンポジウム 2011(SES2011), 2011 年 9 月 13 日. 東京女子大学. 査読有り

(2) 青谷知幸, 紙名哲生, 増原英彦. オブジェクト毎の層遷移を宣言的に記述できる文脈指向言語 EventCJ. 日本ソフトウェア科学会第 28 回大会. 2011 年 9 月 28 日. 沖縄産業センター.

(3) Tetsuo Kamina, Tomoyuki Aotani, Hidehiko Masuhara, and Tetsuo Tamai, COSE: Context-Oriented Software Engineering with Use Cases and Event-Based Context Transition. In AOAsia/Pacific'11, 2011 年 10 月 17 日. 上海交通大学 (中国).

(4) Tetsuo Kamina, Tomoyuki Aotani, Hidehiko Masuhara. Modeling Context Changes and Layers: A Use Case Driven Approach. In COP'12, 2012 年 6 月 11 日, Beijing, China.

(5) 紙名哲生, 青谷知幸, 増原英彦. 文脈指向言語 EventCJ への合成層の導入. 情報処理学会第 89 回プログラミング研究発表会. 2012 年 6 月 21 日. 小樽.

(6) 紙名哲生, 青谷知幸, 増原英彦. JavaCat: Realizing Context as Fluent. 第 15 回プログラミングおよびプログラミング言語ワークショップ (PPL2013), 2013 年 3 月 5 日. 会津若松.

６．研究組織
(1)研究代表者
紙名　哲生 (KAMINA TETSUO)
東京大学・大学院教育学研究科・特任助教
研究者番号：90431882