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　２ 年　 ６ 月　 ７ 日現在

機関番号：１２６０１
研究種目：基盤研究(B)（一般）
研究期間：2015～2018
課題番号：１５Ｈ０２６８２

研究課題名（和文）ドメイン専用言語の実装技術

研究課題名（英文）Implementation techniques for Domain Specific Languages

研究代表者
千葉　滋（Chiba, Shigeru）

東京大学・大学院情報理工学系研究科・教授

研究者番号：８０２８２７１３

交付決定額（研究期間全体）：（直接経費）　　13,800,000円

研究成果の概要（和文）：本研究では埋め込み領域特化言語（埋め込みドメイン専用言語）の実装技術の研究をおこなった。(1) 埋め込み領域特化言語の実装を直接支援する言語機構を備えた汎用プログラミング言語を研究開発し、利用者が誤った領域特化言語のプログラムを書きにくくなるようにした。(2) 独自の構文をもった埋め込み領域特化言語の実装を可能にする言語機構 turnstile type を考案した。(3) メソッドチェインを用いた埋め込み領域特化言語の実装時に、チェイン中のメソッドの並びの正しさをホストとなる汎用言語の型検査を利用しておこなう新たな手法を開発した。

研究成果の学術的意義や社会的意義
汎用プログラミング言語上のライブラリとして実装される埋め込み領域特化言語は、独立したミニ・プログラミング言語風のプログラミング・インタフェースをもつライブラリともいえ、次世代の高機能ライブラリとして注目されている。本研究は、そのようなライブラリを実用化するための基礎技術を開発した。これは高品質なソフトウェアを少人数かつ短い時間で開発するための一助になる。

研究成果の概要（英文）：This research project investigated implementation techniques for Embedded Domain Specific Languages (EDSLs). (1) It developed a general-purpose programming language with a language support for implementing EDSLs. The language helps the users avoid making a mistake when writing EDSL code. (2) The project developed a new language construct named turnstile type, which enables an EDSL with dedicated syntactic constructs. (3) The project developed a technique for verifying a sequence of method calls by using the type system of the host language when an EDSL program is represented by a chain of method calls.

研究分野：プログラミング言語、コンピュータ・ソフトウェア

キーワード：ライブラリ　プログラミング言語　ソフトウェア学　ソフトウェア開発の効率化・安定化

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

Fortran や C、Java といったプログラミング言語は汎用プログラミング言語と呼ばれ、特定の分野に限定されない任意の分野のソフトウェアの開発に使われることを想定して設計されている。一方で特定の応用領域のソフトウェア開発に特化して設計された領域特化言語（ドメイン専用言語、Domain Specific Languages、DSL）の有用性もよく知られている。R や SQL が領域特化言語の例である。領域特化言語は汎用言語と同様の独立した言語として実現することもできるが、汎用言語のライブラリとして実現することもできる。汎用言語のライブラリとして実現された領域特化言語は、その汎用言語をホスト言語とする、埋め込み領域特化言語（Embedded DSL）と呼ばれ、その実用性が注目されつつある。埋め込み領域特化言語はライブラリの一種であるので、そのホストである汎用言語のプログラムの中で、その領域に関係する部分の記述だけに使うことができ、既存の開発環境との親和性が高い。またライブラリであるので実装が比較的容易である。領域特化言語を独立したプログラミング言語として実現すると、コンパイラや実行系を開発する他に、エディタやデバッガなど、ソフトウェア開発を支える様々なツール群も整備しなければならず開発が大変である。

２．研究の目的

埋め込み領域特化言語を新世代のライブラリの形態ととらえ、そのようなライブラリを作りやすいプログラミング言語や、そのようなライブラリの作成を支援するツールを開発することが本研究の目的であった。プログラムのライブラリは、Fortran のライブラリに代表されるような関数ライブラリに始まり、現在の主流のデザインといってよいオブジェクト指向ライブラリへと発展してきた。オブジェクト指向ライブラリの普及とともに、アプリケーション・プログラム全体のひな形を提供するライブラリも広く使われるようになり、これらはフレームワークと呼ばれる。埋め込み領域特化言語は、このようなライブラリの進化の先にある最も新しい形態ととらえ、そのさらなる進化と普及につながる研究をおこなった。

これまでのライブラリは、様々な形の再利用可能なプログラムの断片を提供するが、多種多様な機能のプログラムの断片を多数提供する一方、求める機能をライブラリの中から探し出し、正しく利用するのが難しいという問題があった。あるライブラリは、一つの機能をいくつかの関数やメソッドに分割して提供し、利用者が適切な関数やメソッドを選んで組み合わせることで、利用者が最も望む形の機能を得られるようにしている。そのようなライブラリは便利な一方、組合わされた関数やメソッドを決められた正しい順序で呼び出さないと求める機能を正しく得られないことが多い。しかし正しい呼び出し順序は、ライブラリのマニュアルに書いてあるだけなので、利用者はマニュアルを読み込んで正しい順序を学ばないといけない。これではライブラリを利用するために学ばなければならないことが多く時間がかかり、また利用を始めた後も間違った使い方をして誤った動作をするプログラムを書いてしまいがちになる。

ライブラリを、領域特化言語とはいえ一つのプログラミング言語と捉え、普通のプログラミング言語が備える性質をライブラリにも与えることできれば、そのようなライブラリの問題を低減できる。その領域のプログラムを記述するのに自然な構文を利用できれば、ライブラリの利用の仕方がより直感的になり、利用の仕方に悩む場面が少なくなるだろう。これまでの埋め込み領域特化言語では独自の構文が利用できず、原則全ての記述が関数またはメソッド呼び出し、そしてラムダ式で表現されるので、しばしば直感的には必要性や意味がわからない記述がプログラムに現れがちである。また普通のプログラミング言語が備える構文チェックなどに相当する機能をライブラリがもてば、誤ったライブラリの利用の仕方を静的に検査して見つけ出すことも可能だろう。これまでの埋め込み特化言語では、ホスト言語の構文チェック等は利用できるが、その埋め込み特化言語独自の静的チェックは利用できないのが普通であった。

３．研究の方法

埋め込み領域特化言語のための新しい言語機構を備えたプログラミング言語を、既存のプログラミング言語を拡張して新たに開発するアプローチと、既存の汎用プログラミング言語を改造せずにその機能の範囲内でより優れた埋め込み領域特化言語の作成方法を探るアプローチの、二つのアプローチで研究をおこなった。

前者のアプローチでは、既存のプログラミング言語の拡張とはいえ新しい言語を開発するので、より優れた埋め込み領域特化言語を作成できるようになるが、それを即、実用的なソフトウェア開発に適用することは困難である。そのためにはプログラミング言語のコンパイラ等の処理系だけでなく、その言語専用のエディタやデバッガなど様々な周辺ツールも用意しなければならない。一方、後者のアプローチでは既存のプログラミング言語をそのまま用いるので、研究成果を即実用に供することができる。しかし機能的な制約は大きく、従来のものから大きく進化した埋め込み領域特化言語を実現することは難しい。本研究では、二つのアプローチを両方採用することで、長期的な視点での研究開発と中短期的な視点での研究開発の両方をおこなった。

４．研究成果

4.1. 領域特化言語の開発を支援する新しい汎用プログラミング言語の研究開発

埋め込み領域特化言語は本質的にライブラリであるため、注意深く設計されていない領域特化言語では、利用者が落とし穴にはまりやすい。とくに deep embedding あるいは staged embedding と呼ばれる手法で設計された領域特化言語にその傾向が強い。そのような領域特化言語では、領域特化言語で書かれたプログラム（しばしばライブラリ・メソッドの呼び出しで表現される）が実行されると、まずそのプログラムの抽象構文木が作られ、コンパイル処理（最適化や機械語命令の生成）がおこなわれ、最後に実行される。これら一連の処理はホスト言語の実行中におこなわれる。これら一連の処理がどのタイミングで実行されるか、領域特化言語の利用者側からは必ずしも自明でないため、落とし穴にはまり、意図せず誤ったプログラムを書いてしまいがちである。本研究では具体的にどのような落とし穴があるかを調べ、それを避けるために、埋め込み領域特化言語を実装するための言語機構をそなえた新しいプログラミング言語を開発した。開発したプログラミング言語は Java 言語に新たな言語機構を追加したものである。この研究成果は Scherr と Chiba によって GPCE2015 で論文発表された。

埋め込み領域特化言語の弱点の一つは、利用できる構文がホスト言語の構文に限られるので、その領域に適した構文を必ずしも用いることができないことである。とくに大半の埋め込み領域特化言語は、ホスト言語のメソッドや関数の呼び出しの連鎖として表現しなくてはならない。If 文や while 文のような制御構造も利用できないので、埋め込み領域特化言語のプログラムはあまり読みやすくないことが多い。この問題を解決するために、構文マクロを用いて構文拡張を可能にする手法は数多く提案されてきた。しかし構文マクロでは新しい種類の字句（トークン）や構文を導入することは難しく、可能であってもマクロによる複数の拡張構文を一緒に組合わせて用いることはできない。本研究では、拡張構文をユーザ定義演算子として実装する手法を研究した。静的型システムを活用することで、異なる拡張構文も柔軟に組合わせて利用することができる。ユーザ定義演算子による構文拡張では、名前束縛（その構文の中でだけ有効な変数の宣言）を扱えなかったが、本研究では turnstile type を考案し、特定のスコープ内でだけ有効なユーザ定義演算子を導入することで、この問題を解決した。この研究成果は Ichikawa と Chiba によって<Programming>2017 で論文発表された。

4.2. 既存の汎用プログラミング言語用の領域特化言語の開発支援技術

メソッドチェインは埋め込み領域特化言語のプログラムの表現によく使われる技法である。メソッドチェインとは、鎖状につながったメソッド呼び出し（あるいは関数呼び出し）のことである。最初のメソッド呼び出しの戻り値に対してまたメソッド呼び出しをおこない、とこれを連続させるとメソッド呼び出しを鎖のようにつなげることができる。鎖状につながった個々のメソッドをトークンとみなすと、チェイン全体を領域特化言語による一つのプログラムと見なせる。チェインはプログラムと見なせるが、そのチェインの正しさを静的に検証する仕組みは従来ほ

とんど提供されてこなかった。通常のプログラミング言語であれば、構文検査や型検査である程度コンパイル時にプログラムの誤りを発見できるが、埋め込み領域特化言語ではそのような検査はほとんど提供されない。ホスト言語の検査はもちろん領域特化言語のプログラムにも適用されるが、領域特化言語独自の構文検査（メソッドチェインの中のメソッドの呼び出し順序の正しさ）や型検査はほとんどおこなわれない。本研究では、メソッドチェイン中のメソッドの正しい並び方を LL 文法あるいは LR 文法で表現できれば、ホスト言語の静的型検査機構を利用することでコンパイル時にメソッドの並び方の正しさを検査できることを示した。また考案したアルゴリズムを用い、そのような検査をおこなえるライブラリのひな形を文法の定義から生成するソフトウェア開発ツールも作成した。作成したツールは、既存の Java 言語向けのものと、Scala、C++、Haskell 言語向けのものであり、実際に実用に使える。この研究成果は Nakamaru、Yamazaki、Chiba 他の論文として、GPCE2017、Journal of Computer Languages、OOPSLA2019、<Programming>2020 で論文発表された。

５．主な発表論文等

〔雑誌論文〕　計13件（うち査読付論文　13件／うち国際共著　0件／うちオープンアクセス　6件）

１．著者名 Tomoki Nakamaru, Kazuhiro Ichikawa, Tetsuro Yamazaki, and Shigeru Chiba	４．巻 vol.52 issue 12
２．論文標題 Silverchain: A Fluent API Generator	５．発行年 2017年
３．雑誌名 Proceedings of the 16th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE 2017)	６．最初と最後の頁 199-211
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3136040.3136041	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Joseph Caldwell and Shigeru Chiba	４．巻 vol.52 issue 12
２．論文標題 Reducing calling convention overhead in object-oriented programming on embedded ARM thumb-2 platforms	５．発行年 2017年
３．雑誌名 Proceedings of the 16th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE 2017)	６．最初と最後の頁 146-156
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3136040.3136057	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Yung Yu Zhuang, Shigeru Chiba	４．巻 24-4
２．論文標題 Expanding Event Systems to Support Signals by Enabling the Automation of Handler Bindings	５．発行年 2016年
３．雑誌名 Journal of Information Processing	６．最初と最後の頁 620-634
掲載論文のDOI（デジタルオブジェクト識別子） https://doi.org/10.2197/ipsjjip.24.620	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 －

１．著者名 福室 嶺・千葉 滋	４．巻 9-4
２．論文標題 有効範囲を既知のコールパスに限定する Ruby 向けの安全な クラス拡張 Method seals	５．発行年 2016年
３．雑誌名 情報処理学会論文誌 プログラミング	６．最初と最後の頁 16-26
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 －

1．著者名 Kazuhiro Ichikawa and Shigeru Chiba	4．巻 1-2
2．論文標題 User-Defined Operators Including Name Binding for New Language Constructs	5．発行年 2017年
3．雑誌名 The Art, Science, and Engineering of Programming	6．最初と最後の頁 Article 15
掲載論文のDOI（デジタルオブジェクト識別子） 10.22152/programming-journal.org/2017/1/15	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 －

1．著者名 Maximilian Scherr and Shigeru Chiba	4．巻 51-3
2．論文標題 Almost First-Class Language Embedding: Taming Staged Embedded DSLs	5．発行年 2015年
3．雑誌名 ACM SIGPLAN Notices - GPCE '15	6．最初と最後の頁 21-30
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/2936314.2814217	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Hiroshi Yamaguchi and Shigeru Chiba	4．巻 51-3
2．論文標題 Inverse Macro in Scala	5．発行年 2015年
3．雑誌名 ACM SIGPLAN Notices - GPCE '15	6．最初と最後の頁 85-94
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/2814204.2814213	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Tetsuro Yamazaki and Shigeru Chiba	4．巻 -
2．論文標題 Buffered Garbage Collection for Self-Reflective Customization	5．発行年 2018年
3．雑誌名 Proc. of 33rd Annual ACM Symposium on Applied Computing (SAC 2018)	6．最初と最後の頁 1256-1259
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3167132.3167416	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Tomoki Nakamaru, Kazuhiro Ichikawa, Tetsuro Yamazaki, and Shigeru Chiba	4．巻 50
2．論文標題 Generating fluent embedded domain-specific languages with subchaining	5．発行年 2019年
3．雑誌名 Journal of Computer Languages	6．最初と最後の頁 70-83
掲載論文のDOI（デジタルオブジェクト識別子） 10.1016/j.jvlc.2018.11.001	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Yamazaki Tetsuro、Chiba Shigeru	4．巻 27
2．論文標題 Buffered Garbage Collection: An Approach to Collecting Self-Created Garbage Objects	5．発行年 2019年
3．雑誌名 Journal of Information Processing	6．最初と最後の頁 479-488
掲載論文のDOI（デジタルオブジェクト識別子） 10.2197/ipsjjip.27.479	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 －

1．著者名 Chiba Shigeru	4．巻 -
2．論文標題 Foreign language interfaces by code migration	5．発行年 2019年
3．雑誌名 The 18th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE 2019)	6．最初と最後の頁 1-13
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3357765.3359521	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Yamazaki Tetsuro、Nakamaru Tomoki、Ichikawa Kazuhiro、Chiba Shigeru	4．巻 3
2．論文標題 Generating a fluent API with syntax checking from an LR grammar	5．発行年 2019年
3．雑誌名 Proceedings of the ACM on Programming Languages	6．最初と最後の頁 1-24
掲載論文のDOI（デジタルオブジェクト識別子） 10.1145/3360560	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 －

１．著者名 Nakamaru Tomoki、Chiba Shigeru	４．巻 4-3
２．論文標題 Generating a Generic Fluent API in Java	５．発行年 2020年
３．雑誌名 The Art, Science, and Engineering of Programming	６．最初と最後の頁 9:1-9:23
掲載論文のDOI（デジタルオブジェクト識別子） 10.22152/programming-journal.org/2020/4/9	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2件（うち招待講演　0件／うち国際学会　0件）

１．発表者名 Antoine Tu, Shigeru Chiba
２．発表標題 Outlook on Composite Type Labels in User-Defined Type Systems
３．学会等名 34th JSSST conference, Keio university, September, 2017
４．発表年 2017年

１．発表者名 松永 智將, 市川 和央, 山崎 徹郎, 中丸 智貴, 千葉 滋
２．発表標題 型検査を用いたコンパイル時LR構文解析手法の提案
３．学会等名 日本ソフトウェア科学会第34回大会
４．発表年 2017年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

Silverchain
https://github.com/csg-tokyo/silverchain
https://www.youtube.com/watch?v=3f0n8cbhFZU

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