様式 C-19

科学研究費補助金研究成果報告書

平成２１年　５月２９日現在

研究種目：若手研究（Ｂ）
研究期間：　2007　～　2008
課題番号：19700031
研究課題名（和文）ドメイン特化型言語開発環境の研究

研究課題名（英文）Research of Development Environment for Domain Specific Languages

研究代表者
佐々木　晃（SASAKI AKIRA）
法政大学･情報科学部･准教授
研究者番号：90396870

研究成果の概要：コンピュータは様々な場面で用いられるようになっているが，高度な利用は限られた専門家にしか可能ではない．これは，コンピュータ言語と，人間が問題解決のために用いる言語の構造が違うからである．本課題では，この溝を埋める「ドメイン特化型言語（特定の問題解決に目的を絞った言語)」をスムーズに設計，作成，運用を可能にする枠組みについて研究を行った．特に，このフレームワークを実現するため，新しく作成する言語の構造設計図から，必要となる一連のソフトウェアを半自動で作成する手法について研究した．

交付額

（金額単位：円）

	直接経費	間接経費	合　計
2007 年度	800,000	0	800,000
2008 年度	1,600,000	480,000	2,080,000
年度			
年度			
年度			
総　計	2,400,000	480,000	2,880,000

研究分野：情報科学
科研費の分科・細目：情報学･ソフトウェア
キーワード：プログラミング言語，ドメイン特化型言語，属性文法，コンパイラ，エディタ，言語開発環境

１．研究開始当初の背景

様々な学術，研究分野においてコンピュータを援用した問題解決が不可欠となっている．しかし，それぞれの分野の研究者は，必ずしもコンピュータ言語などを扱えるわけではないため，彼らにとって，計算機科学の知見や技術を存分に生かした研究を行うことは簡単ではない．

例えば，コンピュータシミュレーションを利用し，問題解決を計るという例は，様々な分野ですでに行われている．しかし，システム開発の側面から見ると，新たなシミュレーション言語を定義し処理系を実装することは困難である．また，実際に開発した言語処理系の利用を促進するためには，ＧＵＩを含む一連の補助的なツールを用意する必要があり，このようなシステムを実現するのは容易ではない．言語処理系の開発支援ツールとして，yacc や lex などの構文解析器生成系や字句解析器の自動生成システムが古くから研究されている．これらは言語解析のためのフロントエンド部を，形式的な記述から自動

生成することができるが，言語処理系のフェーズのごく最初の部分しか扱うことができない．また，これより少し広い範囲の言語処理系アプリケーションの開発を支援するSableCC,JavaCC 等の研究があるがその用途はコンパイラという分野に限られているという問題点がある．

２．研究の目的

本研究の全体的な目的は，上記で挙げたコンピュータシミュレーションのような典型例をはじめとした，様々な学術，研究分野で必要となるソフトウェア・アプリケーションを効率的に開発するための手法を確立することを全体的な目標とする．特に，本研究課題では，各分野の専門家が利用するアプリケーションとして，ドメイン特化型言語(Domain Specific Language)を対象とすることとし，ドメイン特化型言語処理系およびその周辺のツールを効率良く開発するためのフレームワークを明らかにすることを目的とする．ドメイン特化型言語は，特定の領域の問題を，その領域に適したモデルの記述方法で表現するための言語であり，近年注目されている．一般には，汎用プログラミング言語とは異なるが，ドメイン特化型言語で書かれたモデル記述は，適切な処理系によってコンピュータ上で実行可能である，という利点をもつ．したがって，その領域の専門家が，たとえ汎用プログラム言語を書くことができない場合でも，ドメイン特化型言語を用いて問題（すなわちプログラム）を記述し，コンピュータ上で実行することが可能となる．一方で，対象分野で扱われる問題領域を専用言語として適切にモデル化し，さらに，その言語のための処理系を実装するプロセスには様々な困難が伴う．そこで，本研究では，ドメイン特化型言語を効率よく開発するための定式化およびフレームワークを明らかにすることを目標とする

３．研究の方法

(1) 本研究で想定するドメイン特化型言語開発フレームワークの核となる定式化手法について明らかにする．定式化手法を明らかにする際には，具体的なケーススタディとして，既存の社会科学シミュレーションシステムの事例を調査する．また，ドメイン特化型言語開発を効果的に行うことのできる定式化の手法として，言語を定式化する一つの手法である属性文法による方法を応用する．この定式化手法は，言語処理系の作成と同時に，その言語処理系を運用するためのツール群，すなわち言語処理系の開発環境の構築に応用可能である．

(2) この定式化手法を適用し，言語処理系を含む一連のツール群の要素を仕様記述から自動生成する手法を用いて，ドメイン特化型言語と開発環境の効率的な実現を目指す．提案する定式化手法を実現するために，独自の自動生成システムを実装し，ドメイン特化型言語開発環境のためのフレームワークの試作を行う．さらに，本フレームワークを用いて，具体的なドメイン特化型言語の開発を行い，本研究の有用性を明らかにする．

４．研究成果

(1)研究成果の概要

本課題では上記の研究方法に基づいて研究を行った．主な研究成果は次の①②③である．

① ドメイン特化型言語開発環境のための定式化とフレームワークの設計：

ドメイン特化型言語のフレームワークの核となる定式化手法について研究を行った．既存のドメイン特化型言語の調査と具体的な利用手法に関しては主に研究成果[１]，[３]で示した．また，定式化手法は，木文法および属性文法を核としたものであるが，これに関しては研究成果[２]，[４]で示した．

② ドメイン特化型言語のエディタの設計手法：

ドメイン特化型言語では専用のエディタの作成が欠かせないが，その実装は簡単ではない．本フレームワークの手法に基づいたエディタの設計手法について研究成果[２]，[４]で示している．

③ フレームワークに基づくシミュレーションシステムの設計と実装：

本フレームワークの考え方に基づいたシミュレーションシステムの試作を行った．本課題では，既存の社会科学シミュレーションシステムを作成した．トランスレータシステム，ドキュメント生成システム，エディタの一部の試作を行った．またシミュレーションシステムをドメイン特化する考え方と試作システムについては，研究成果[３]で示した．

(2) ドメイン特化型言語処理系の定式化手法と言語指向エディタの自動生成

本節では主に研究成果[２]および[４]に基づき，ドメイン特化型言語処理系の定式化手法とこの方法を応用した言語指向エディタの定式化手法についての研究結果を中心に述べる．この定式化手法は抽象構文木によるＤＳＬ言語の定義および，属性文法を応用した処理系の動作記述に基づくものであり，本課題におけるドメイン特化型言語フレームワークの定式化の核となる手法である．

① 概要

「ドメイン特化型言語」は，特定の分野で使われることに目的を特化したプログラミング言語である．一方でドメイン特化型言語の利用者は，通常のプログラミングの知識を持っているとは限らない．従ってドメイン特化型言語を実際に運用する際には，言語処理系の他，開発環境を提供することが需要となる．中でもプログラムエディタは重要な要素となる．通常プログラムを作成，編集する際には汎用的なテキストエディタを用いるが，ここで想定するＤＳＬに対するエディタは，利用者から見れば通常のＧＵＩアプリケーションとしてとらえられるものを考え，ここではビジュアルエディタと呼ぶ．典型例としてウィザード形式によるＧＵＩエディタが挙げられるが，このような単純なものであっても，開発には言語仕様を満たすための緻密性が要求され，定式化や自動生成による開発コスト削減，保守性の向上が望まれる．このようなエディタは，一般的には言語指向エディタあるいは構造化エディタの一種と考えられる．

汎用言語に対する言語指向エディタの自動生成は，インクリメンタル構文解析など関連する研究が古くからなされている．一方で，本発表で想定しているエディタでは，ユーザが編集する対象はテキスト表現によるプログラムではなく，テキストの構文などを抽象化した抽象構文木を編集させるという視点をとる．そこで，抽象構文木を定義する木文法(tree grammar) に基づいてエディタの定式化を行う．この方式では，テキストを扱わないので汎用言語における構文エラーは発生しない．本研究では，エディタによるプログラムの編集操作に対する意味的な定式化を行った．さらに，この定式化に対応したエディタの編集操作に必要となるビジュアルエディタのコンポーネントに対する要件を明らかにした．また，提案した定式化を用いたビジュアルエディタの作成を支援するツールの実装を行った．さらに実際にドメイン特化型言語向けのグラフィカルエディタの例として，SOARS シミュレーション言語に対するエディタの試作を行った．

② ドメイン特化型言語の定義と編集モデル

本研究では，対象とするドメイン特化型言語の構造を抽象構文木で定義する．コンパイラ，トランスレータをはじめとする言語処理系は，この抽象構文木を解析，合成することが基本となる．エディタの場合は，抽象構文木を構成する過程が，プログラミングであるという立場をとる．以下は，加算と減算を表せる言語（数式言語）の抽象構文木定義を木文法により記述している．

```
%start input;
input => LINE line;
line => EXP exp;
exp => ADD exp exp | SUB exp exp | NUM num;
```

本研究では，ＤＳＬ言語の構造を上記のような抽象構文木で定義し，ビジュアルエディタをはじめ言語処理系自動生成の際の記述言語とする．

ここで，木文法で扱う記号について次の分類を行う．

［開始記号］　%start の右に現れる文法記号．上の例では input がこれに当たる．

［非終端記号］　=>の左側に現れる文法記号．上の例では input, line, exp がこれに当たる．

［タグ］　=>の右側の 1 番目に現れる文法記号にあたるもの．上の例では LINE, EXP, ADD, SUB, NUM がこれに当たる．

［終端記号］　=>の右側の 2 番目以降に現れる文法記号にあたるもの．上の例では num がこれに当たる．

本研究での抽象構文木の編集のモデルは，木文法の開始記号に対応したルートノードを与え，文法規則に従ってトップダウンでの導出(derivation)を行う，という考え方を基本とする．先の例を用いると，木をＳ式で表現した場合，下記のように表現される．

```
(LINE
  (EXP
    (ADD
      (SUB (NUM "3")
           (NUM "5")
      (NUM "7")))))
```

導出の過程で抽象構文木に現れるノードを以下の 5 種類に分類して定義する．

[ルートノード] 木文法における開始記号に対応するノードで抽象構文木の根にあたる．

[非終端ノード] 木文法における非終端記号に対応するノードで親は必ずタグノードである．もしくは自身がルートノードでもある．

[タグノード]木文法におけるタグに対応するノードで親は必ず非終端ノードである．

[終端ノード]木文法における終端記号に対応するノードで親は必ずタグノードである．もしくは自身がルートノードでもある．

[字句ノード]文字列などを属性（プロパティ）として持つ．作成するプログラムの識別子(変数名)やリテラル等を持たせることを目的とする．

プログラミングの編集過程は，大別して，抽象構文木を導出していく生成の方向と，記述途上のプログラムの一部を書き換えるた

めに部分的な構文木を削除する方向の二つが考えられるが，ここでは，生成の方向に注目して，編集途上にある抽象構文木の状態を表現する．

編集途中の抽象構文木の状態は，木文法をもとに次のように考えられる「ある木文法 G が与えられたとき，これに対応して構築途中も含めた導出可能な全ての抽象構文木の集合」．これは，構文解析のモデル化で用いられる「文形式（sentential form)」に対応するものである．構文解析においては，文形式はテキスト等の構文解析している過程を表現するものであるが，木のプログラミングにおいては，ユーザが編集し得る木の形を与える．

下記は，例として挙げた木文法に対する抽象構文木の生成過程を表したものであり，これがプログラム編集のモデルとなる．

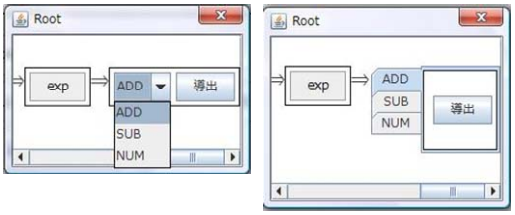
```
(input)  ⇒ (input (LINE))
         ⇒ (input (LINE
                   (line EXP)))

...      ⇒ (input (LINE
              (line (EXP
                 (exp ADD)))))
   ⇒ (input (LINE
          (line (EXP
            (exp (ADD (exp) (exp)))))))
   ⇒ ...
```

③ ビジュアルエディタのコンポーネントの実装例

以上で述べたように，本研究におけるプログラム編集の基礎となる考え方は，導出操作による抽象構文木の構築である．すなわち「文形式」に対応する構築途中の抽象構文木に対する操作として表される．したがって，ビジュアルエディタのコンポーネントの要件は，次のようなユーザインタフェースを備えていることである．

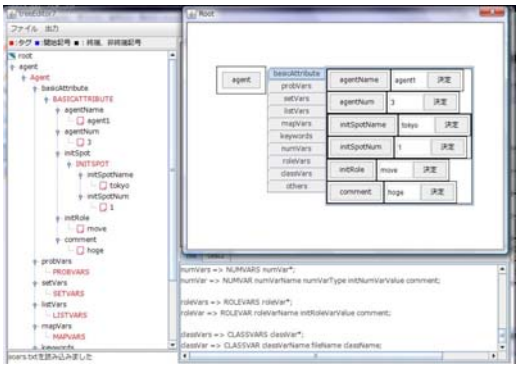
o 構築途中の抽象構文木を適切な表現で視覚化したものであること
o 適切な導出をユーザに促すものであること

導出のパターンは，先に挙げた抽象構文木のノードに対して与えられる．下記では，このパターンに対応するエディタのコンポーネントの例を挙げる．基本的には，ノードの導出は，親ノードを表すコンポーネント上に配置されたボタンやメニュー等に対して，ユーザが適切なアクションを起こすことで行われる．例えば，非終端ノードから選択肢を選ぶインタフェースは次のようにコンボボックスやメニューを使う方法が考えられる．



④ エディタ生成システム

ビジュアルエディタの作成を支援するツールを Java 言語および Swing GUI ライブラリにより実装した．（下図）．エディタ作成者は，木文法の構文定義を与えることで，その言語向けのエディタの基本部分を構成したことになる．ＧＵＩの表現方法は，いくつかのバリエーションがあり，各導出レベルでの表現方法を指定する必要がある．これをカスタマイズすることで，エディタを自動生成することができる．



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１件）

① ［1］ 蒲野茂幸，佐々木晃，移植可能な Superoptimizer による最適な命令パターンの自動生成とそのパターンによる覗き穴最適化，情報科学技術レターズ，査読有，vol. ６，2007 年　pp17--20

〔学会発表〕（計３件）

① [2] 佐々木晃，市川寛，田沼英樹，ドメイン特化型言語のためのビジュアルエディタの定式化，情報処理学会プログラミング研究会，2009 年 3 月 16 日，東京大学

② [3] Akira Sasaki, Manabu Ichikawa, Hideki Tanuma, Hiroshi Deguchi, Constructing Tailored Simulations by Domain Specific Extension Approaches ,2nd. World Congress on Social Simulation (WCSS2008), 2008 年, George Mason 大学, 米国

③ [4] 佐々木晃，須賀康行，ドメイン特化型言語に対するエディタの自動生成手法，情報処理学会プログラミング研究会，2008 年３月１７日，IBM 東京基礎研究所

６．研究組織

(1)研究代表者

佐々木　晃　（SASAKI AKIRA）
法政大学･情報科学部･准教授
研究者番号：90396870