エージェント指向言語 Flage(5) — メタ知識を用いた自動証明*

3B－5

粂野文洋† 大須賀昭彦‡ 本位田真一‡
新ソフトウェア構造化モデル研究本部
情報処理振興事業協会（ＩＰＡ）

1 はじめに

実際の数学問題やプログラム生成／検証に自動証明を適用した場合、形式的論理体系のみに基づいたものでは、公理や補題などの知識の増大に対して、その探索空間が指数関数的に広がってしまい、組合せ論的爆発を招く場合が多い。この問題に対して、ヒューリスティックスや問題領域上の構造定理、類推をメタ知識として利用し、一種のプランニングを行うことによって探索空間を削減する研究が行われている [1, 2, 3]。

一方、プランニングの問題として、問題領域が大規模／複雑になった場合、プランニングの適用可能性チェックのコストや必要な知識（オペレータ）を事前にすべて与えることの困難性があげられる。これは自動証明のプランニングの場合にもあてはまる場合が多いと考えられる。この問題の解決策としては、マルチエージェントによるプランニング方式が提案されている [4]。

そこで我々は、両者の技術の特長を利用し、メタ知識／マルチエージェントによる自動証明の探索空間の削減と分散化を行なう試みを行っている。本稿では、構成的プログラミング [5] に自動証明を応用したプログラム合成を類推／マルチエージェントによって実現する試みについて、概要を示す。また、こうした試みがエージェント指向言語 Flage によって自然にモデル化できることを示す。

2 マルチエージェントによる自動証明／プログラム生成

まずプログラミングを行うエージェントが複数存在する系を考える。この系はメッセージ(中身はプログラムの仕様である。処理を行うデータを含む場合もある)を受信し、指示された処理を行い、結果を送信元に返す。

各エージェントはプルーバ、整数やリスト等の基本データおよび基本操作に関する知識を持ち、さらに証明を行う際にプルーバを補助するプランナ、過去に自エージェントが生成したプログラム、仕様およびその証明、証明プラン等の情報を知識として持っている。また、証明木からプログラムを導出するためのルーチンおよび知識も持っている。

本アプローチによるメッセージ処理の流れを以下に示す。

1. メッセージが系に送られたとする。系にはいくつかの場が存在し、メッセージがどの場に送られるかの情報はメッセージが持っているものとする。場はデータ定義領域や仕様の意図等によって規定されているものとする。メッセージは、場に存在するエージェントにブロードキャストされ、同じ仕様のプログラムを持っているエージェントが存在すれば、同じ場のエージェントにその旨をアナウンスし、そのエージェントがプログラムをメッセージの送信元に返す。メッセージに処理するデータが含まれている場合はその計算を行い、結果も合わせて送信元に返す。

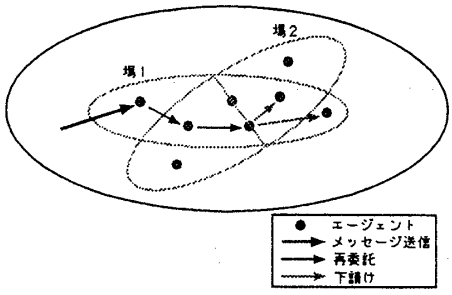

図 1: プログラミングエージェントの系

2. 同じ仕様のプログラムを持つエージェントがいなかった場合は、場内の任意の１エージェントにメッセージが送られる。メッセージを受けとったエージェントはメッセージが持つ仕様と自分が持っている仕様との類似点を探す。類似性の規定については3章を参照のこと。メッセージの仕様とエージェントが持つプログラムの仕様に類似性が見つけられない場合は、そのエージェントが他のエージェントへ再委託を行う。図1中の下請けについては4章を参照のこと。

3. 2で分かった類似性から証明のプランニングを立て、プルーバによって、対話的に証明を行う。証明が失敗した場合は、他エージェントへの再委託が行なわれる。証明が成功した場合は、その証明木からプログラム（これは正当性が保証されている）を抽出する。そして1と同様に結果を送信元に返す。

ユーザがこの系をどのように利用するかという点から考えると、この系は、たとえば、一種の知的なプログラム部品が格納されたライブラリと見ることができる。場はプログラムを整理して格納し、検索を容易にするための分類や検索キーによって規定される。エージェントは仕様、証明 etc を含んだプログラムの格納単位である。場の規定は基本的には入出力データ型に基づくが、ライブラリの構築方針にも依存する。つまり、ライブラリの構築者が自分の構築方針に基づいた場も規定することができる。ユーザはこの系に対し、コマンドまたはプログラムからの呼びだしという形で、欲しいプログラムの仕様（または、エージェント内にあるプログラム仕様を変更したもの、すなわちもとの仕様とその変更点を指示したもの）やデータをメッセージ送信する。するとこの系内に存在するエージェント（プログラム生成能力を有したプログラム部品の集まり）の協調によってプログラムが生成され、プログラム自身やそのプログラムで実行したデータをユーザに返す。

3 仕様の類似性

仕様の類似性を考察する前に仕様の記述を規定しておく。仕様は入出力のデータの種類を規定するソートと入

*Agent oriented language Flage(5) – Automated Deduction by meta knowledge: Fumihiro Kumeno,Akihiko Ohsuga,Shinichi Honiden

†(株) 三菱総合研究所より出向
‡(株) 東芝より出向

力と出力の関係で記述される。ソートは定数と構成子およびそのデータ型を引数とする関数（およびその仕様）から構成される。入出力関係は一階述語論理の論理式とする。現在、仕様の類似性として以下の２種類の対応関係の有無による組合せで規定できるものを類似性の条件として考察の対象とし、対応関係を探すマッチャを開発中である。もちろん仕様の変更点が陽に指示されている場合、このマッチャを利用する必要はない。

ソート間の類似性 ターゲット (証明する仕様) の入出力ソートとベース (証明を真似られる仕様) の入出力ソートとの間に準同型写像等の対応関係が存在している。以下に Flage で記述したソートの類似例を示す。

ベースのソート (リスト)：

```
sort list,
op nil: -> list,
op cons: Alphabet -> list -> list,
op append: list -> list -> list, ...
```

ターゲットのソート (ストリング)：

```
sort string,
op null: -> string,
op scons: Alphabet -> string -> string,
op concat: string -> string -> string, ...
```

入出力関係の類似性 ターゲットの入出力関係を表した論理式を構成する述語に対応するもの（たとえば、アリティが同じでその述語を定義した論理式にも同様の対応関係が存在）すべてが、ベースに存在している。または、ターゲット入出力関係を表した論理式を構成する述語に対応するものの一部が、ベースに存在している。以下に Flage で記述した入出力関係の類似例を示す。

ベースの論理式 (リスト上のソーティング)：

```
all(x,exists(y,perm(x,y) and orderd(y))) = true,
perm(nil,nil) = true,
perm(cons(x,y),z) =
perm(y,z') and in(x,z) and delete(x,z,z'), ...
```

ターゲットの論理式 (ストリング上のソーティング)：

```
all(x,exists(y,perm(x,y) and orderd(y))) = true,
perm(null,null) = true,
perm(scons(x,y),z) =
perm(y,z') and in(x,z) and delete(x,z,z'), ...
```

4 証明のプランニングとその実行

3章で述べた類似性を持つベースの証明木から証明のプランを立てる。証明は与えられた論理式を後向き推論によって true になるまで書き換えることによって行う。書き換え規則には一階述語論理の推論規則の他に入出力データや関数に関する定義、性質などを規則化したものも含まれている。プルーバはこれを半自動的に行う。証明プランは、証明する論理式のどの部分をどの規則でどのような順序で適用するかを規定したものであり、プルーバは失敗しない限り、このプランに沿って証明を自動的に行う。現在採っているプランニング戦略はいずれの類似条件も基本的にはインダクションスキーマとその適用箇所に関するものである。すなわち、ソートの対応関係（定数、構成子）とベースで用いられたインダクションスキーマからターゲットの証明に用いるインダクションスキーマを生成し、論理式の対応関係とベースの証明木から適用する場所（インダクション変数）を特定する。また、3章の例のように密接な対応関係がある場合は、以降の証明もベースの証明と対応をとりながら証明を進める。対応がとれない場合は、rippling-out[1] など一般的な戦略も利用しながら、対話的に証明を行う。証明の過程でサブルーチンにあたる補題を証明する必要が出てきた場合は、自エージェント内に相当する関数があればそれを使い、そうでなければその証明を親の証明と同様の手続きで他エージェントへ依頼する（これが図 1の下請けにあたる）かまたはユーザが直接関与する。証明すべき補題がプログラム生成に直接関係しない Harrop 式の場合は自エージェント内で証明が行われるが、ユーザが正しいと仮定してもプログラム生成は可能である。

5 Flage によるエージェントの記述

エージェントの構成を Flage によってモデル化した図を以下に示す。

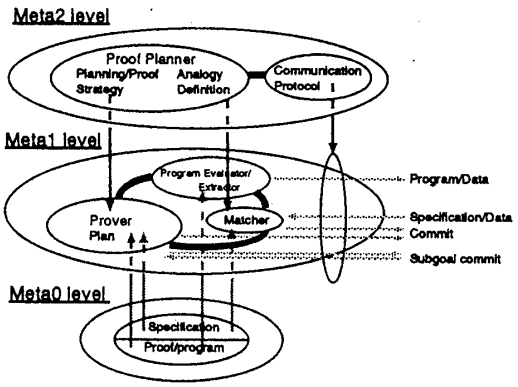

図 2: Flage によるエージェントの構成

6 おわりに

本研究はメタ知識（本稿では類推）とマルチエージェントによって探索空間の削減と分散化を行うことを目標としている。類推による自動証明では、効率的な探索が可能となる一方、隠れた類似性や過剰な類推などの問題点から適用に失敗する場合も多い。これに対しては、事例研究を重ね、仕様記述形式、マッチャ、プランニング方式の研究を進めていく必要がある。また、エージェント間通信については、契約ネットプロトコルを導入することも考えている。こうした試みの実現には、プラン／証明戦略やエージェント間の通信プロトコルを記述する枠組みが必要である。Flage ではメタ階層と場によって協調計算を記述する機能を提供しているため、これらを仕様として自然にモデル化することが可能となった。

謝辞 本研究は、産業科学技術研究開発制度「新ソフトウェア構造化モデルの研究開発」の一環として情報処理振興事業協会（IPA）が新エネルギー・産業技術総合開発機構から委託をうけて実施したものである。

参考文献

[1] Bundy,A et al.: “Experiments with Proof Plans for Induction” ,*Journal of Automated Reasoning* 7,1991,pp.303-324.

[2] 桔梗 宏孝 他: “証明／計算の高速化の一方法”, 日本ソフトウェア科学会第 5 回全国大会,1988,pp.209-212.

[3] Owen,S.:*Analogy for Automated Reasoning*, Perspectives in Artificial Intelligence,1990,Academic Press.

[4] 松本一教 他:“社会構造と協調的プラン生成システムについて”, *MACC '91*, 1991.

[5] Hayashi,S. and Nakano,H. :*PX:A Computational Logic*, 1988, The MIT Press.