4M-8

プログラム合成における対話型知識獲得支援と知識デバッギング

中村 孝　上原 邦昭　豊田 順一
(大阪大学産業科学研究所)

1. はじめに

筆者らはユーザとシステムとが共同して問題を解決する「協調的問題解決システム」についての研究を行っている[1]. 現在, 対象領域として「プログラム合成における対話型知識獲得支援と知識デバッギング」を取り上げ, 知識のデバッギングを効率的に支援するシステムの構築を進めている. 具体的には, 既に開発済みの「自然言語仕様からのプログラム合成システム」を土台として, システムの特徴を生かした知識の獲得とデバッギングを支援する機能を取り入れていくという形で構築を行っている. 本稿では, 知識のデバッギングに有効な支援機能について報告する.

2. 自然言語仕様からのプログラム合成システム[2]

ここでは, 土台となる「自然言語仕様からのプログラム合成システム」, すなわち自然言語による仕様文から Prolog プログラムを合成するシステムの特徴について述べる.

動作の概要

本プログラム合成システムでは, ひとつの仕様文からひとつのプロダクションルール型の Prolog プログラムが合成される(図1). まず, 仕様文を解析した結果からユーザの意図を推論する. 次に, そのゴールを達成するためのプランを推論し, さらに得られたプランからより詳細なプラン列を推論する. このようにして段階的にプラン列を精密化し, プリミティブなプラン列を得る. 最後に個々のプリミティブなプランをそれぞれ対応する Prolog の節に置き換え, 実行可能なプログラムを生成する. これらの推論に必要な, ゴール, プラン, プラン精密化やプランの補足などについての知識(変換のためのルール)をまとめて「知識」と呼ぶ(図2).

「隠れた仕様」の存在

本システムでは, 入力される「仕様文」だけで合成されるプログラムの仕様全体が決まるのではない. 「仕様文」入力以前に既に定められている仕様(「隠れた仕様」と呼ぶ)が存在している. 「隠れた仕様」のうち最も大きなものとして, 合成されるプログラムのタスクがあらかじめ限定されていることがあげられる. プリミティブなプランの種類や, 合成されたプログラムの実行形式なども「仕様文」レベルでは現れない「隠れた仕様」である. 「隠れた仕様」は「知識」レベルにも陽には現れない.

仕様文:
辞書項目 中 に ある 最初 の ボールド体 は 見出し語 を 示す .

Prolog プログラム:

```
rule(辞書項目(_42537),_50460,_50461):-
  not exist_um(flag(ボールド体),_50460),
  pick_up(_42537,[手,_41368,脚],_41603),
  delete_um(辞書項目(_42537),[flag(ボールド体)|_50460],_50516),
  make_pointer(_40977),
  goal([top=見出し語(_41368,_40977),辞書項目(_41603)|_50516],_50461).
```

図1 プログラム合成の実行例

ゴール推論知識:
Xの中のYはZを示す -> XからYをZとして抽出する.

ゴール・プラン知識:
XからYをZとして抽出する -> XからYを取り出す,
YをZとして登録する.

プラン精密化知識:
XをYとして登録する -> Yの内容をXとする,
YをYの付加情報と共に付加する,
::Yは構造名である.

プラン補足知識:
XからYを取り出す -> Xは得られている.

図2 「知識」の例

「知識」の特徴

プログラム合成システムの「知識」を一般のエキスパートシステムなどの知識(規則)と比べてみると, 「知識」の特徴として次の3つがあげられる.

(1) 一般性に欠ける.

「知識」の記述は「専門家からの知識の移行」というよりも「プログラミング」に近く, 個々の「知識」は「分野固有の知識を理解可能な形で記述したもの」にはなっていない. これは, プログラム合成という手続き的な処理をすべて宣言的に記述することが困難であること, および一般的なプログラミング(プログラム合成)についての知識と対象領域に固有な知識とを明確に分離することが困難であることによる. したがって「知識」の正しさはそれ自身だけからではわかりにくい.

(2) 書き換え型のルールである.

プログラム合成システムは一種の設計型エキスパートシステムであり, 診断型エキスパートシステム(MYCINなど)のルールのように結論を付加していくタイプのル

Interactive Knowledge Debugging on Program Synthesis

Takashi NAKAMURA, Kuniaki UEHARA, Jun'ichi TOYODA
The Institute of Scientific and Industrial Research, OSAKA Univ.

ールではなく，中間結果（ここではプラン列）の一部分を書き換えていくタイプのルールを「知識」として持っている．したがって，ルールの適用だけでなくルール適用による中間結果の変化もシステムの動作を説明する重要な要素となる．
(3) 協調型プロダクションシステムにより知識源ごとに管理される．

本システムの特徴として，各処理で必要な知識をまとめて知識源としてグループ化し，協調型プロダクションシステムによりグループごとに知識の管理を行っている．

3. 知識デバッギングの場面

ここでは，プログラム合成システムの知識デバッギングの場面として，次のような場面を仮定する．
・ひとつの仕様文から合成したプログラムに誤りがあった場合を考える．他の仕様文で同様の誤りがあるかどうかは考えない．
・デバッギングの対象は「知識」であり，ゴールからプリミティブなプラン列を推論していく過程で，「知識」の不足・誤りによりプラン列の誤りが生じたものとする．
・知識デバッギングの場面で，一度にひとつの「知識」の付加もしくは修正を行うものとする．複数の知識の付加・修正を一度に行うことは考えない．
・デバッギングを行うユーザは「隠れた仕様」を理解しており，個々の「知識」や各段階のプラン列の正誤を評価できる．
・知識のデバッギングを行うのはあくまでユーザであり，システムはユーザに対し各種の支援を行うという立場をとる．

4. 知識デバッギング支援機能

一般的な知識デバッギングではなく，プログラム合成システムの「知識」の特徴を生かしたデバッギングを支援する機能について考える．

知識デバッギング支援のための諸機能

・基本はトレース．
ルール（「知識」）適用とそれによる中間結果（プラン列）の変化を表示する．「知識」の特徴(1)(2)から，ルール適用だけでなく，プラン列変化の表示が必要となる．なお表示はできるだけ自然言語化して行う．前向き・後向きのトレースやスキップなど，トレースはユーザの要求どおりに行えることが望ましい．
・有効な関連情報の提示．
ルールや中間結果の他にも，有用な情報は必要に応じてユーザに提示する．仕様文，「知識」の条件部・結論部にあらわれる述語・語彙の集合，プリミティブなプランの集合，問題となっている「知識」に関連する「知識」の集合（同じ知識源中の類似ルール，そのルールの直前・直後に発火する可能性のあるルールなど）などが考えられる．これらの提示は「隠れた仕様」の確認・顕在化のために有効である．
・ユーザの推論の誘導．
デバッギング作業を効率よく行わせるためには，ユーザの推論を効果的に導いてやることが必要になる．その時点で問題になっていることがらやユーザの意図などの状況を把握して，ユーザの誤解を招かないように，かつ問題の焦点が明らかになるように，ユーザに対して質問や情報の提示を行う．

誤り（バグ）の種類

正しいプラン列Pが「知識」Rの適用により誤ったプラン列P'に書き換えられたものとする．この場合誤りの種類として，R自体に誤りがある「知識の誤り」と，R自体は正しいがここで適用すべきではないという「知識の適用の誤り」が考えられる．「知識の適用の誤り」は，さらに適用すべき「知識」がルールベース中に存在しない場合と，適用すべき「知識」は存在するが条件部の誤りなどにより適用されない場合に分けられる．

誤り発見の過程

ユーザが誤りを発見する過程を
(1) 誤り候補の発見
(2) 誤りの同定
の２つの段階に分けて考える．
(1) の「誤り候補の発見」では，プラン列の正誤を評価することにより誤りの生じる部分の候補を選び出す．トレースをたどっていき，「正しいプラン列Pが「知識」Rの適用により誤ったプラン列P'に書き換えられている」場所を見つけ出す．システムはトレースのたどり方やプラン列の評価についてユーザに質問し，プラン列の評価に有効な関連情報を提示する．プラン列の評価に有効な関連情報として，仕様文，「知識」の語彙集合，プリミティブなプランの集合などがある．システムがトレースを支援できる機能として，ユーザが誤りのある部分プラン列を指定し，その部分プラン列が生成される時点までトレースをスキップする機能などが考えられる．
(2) の「誤りの同定」では(1)で選び出した「知識」Rの評価を行い，誤りの同定を行う．ユーザは，提示された関連情報などを参考にして誤りの種類を同定し，修正すべき「知識」とその修正方法を見つけ出す．システムは「知識」の評価についてユーザに質問し，「知識」の評価に有効な関連情報を提示する．「知識」の評価に有用な関連情報として，同じ知識源中の類似ルール，Rの直前・直後に発火する可能性のあるルールなどRに関係のある「知識」があげられる．

必ずしもすべてのユーザが(1)(2)の過程をたどるとは限らないが，システム側がこの過程にそってユーザを誘導してやることにより誤りの発見が円滑に行えると期待できる．

5. まとめ

本稿ではプログラム合成システムにおける，知識のデバッギング，特に誤りの発見を支援するに有効ないくつかの機能について報告した．ここで検討した知識デバッギングの諸機能を実際のシステム上に実現し，諸機能の有効性を確認することが課題として残されている．特に協調的問題解決システムの構築にあたっては，ユーザとシステムとの対話の機能に重点をおいてシステム構築を進めていきたいと考えている．

参考文献

[1] 中村，上原，豊田：“問題解決型対話を扱う知的インタフェースの実現“，情報処理学会第32回全国大会講演論文集7M-2，pp.1301-1302，(1986)
[2] 藤井，上原，豊田：“知識工学的手法を用いた自然言語からのプログラム合成システム“，情報処理学会「プロトタイピングと要求定義」シンポジウム資料，pp.131-140，(1986)