Prologプログラミングの協調学習環境の設計と運用 - 行き詰まりの分類とヒントの共有

中京大学情報科学部　　土屋孝文　谷志穂　岡崎加奈
(tsuchiya@sccs.chukyo-u.ac.jp)

1．はじめに

Prolog プログラミングの初学者が、例題と解説を基に、課題を作成しながら学習していく活動について、協調学習支援システムを含む活動の設計と実践を報告する。参加者は手続き型言語の学習経験のある理系大学 3 年次 (平均参加者数 49.3 人)である。本活動では、課題の完成時に仲間へ向けた課題解決ヒントの提供を求め、ヒント集の共有を行った。

2．Prolog プログラミングの協調学習環境

土屋ほか (2003)は、C言語に関する様々な学習資源の利用支援を目的に設計された、ネット上の学習環境を報告した。そこには、(A)学習者が振り返って例題中の知識を確認できるよう、例題プログラム中の各構文要素に、学習者集団に依存した小さな解説が関係付けされており、(B)課題作成がきっかけになって起こる、プログラム解釈に関するつまずきを学習者同士で支えあうことができるよう、例題-課題セッションごとの電子掲示板が運用されている。また、これらの情報を利用した学習者に評価を求め、それらを集約したページ（いわば課題ごとの急所にあたるページ）を提供している。本実践は、この環境の対象言語をPrologに変更した。図1は、例題プログラムmember/2 (リストの要素であることの判定)に関するページである。

土屋ほか (2002)は、学習者に応じた例題-課題セッションの組み合わせと、上のような環境を用いた協調活動の活性化には、課題の達成に関する効果が見られることを報告した。しかし、与えられたプログラムの解釈ではなく、プログラムを生成する過程におこるつまずきや行き詰まりは、支援環境外（ネットの外）の活発な相互作用の中で解消されることが多く、また、課題が達成されたとしても、必ずしも対象知識に関してより深い理解をともなっているわけではないことも指摘した。Cの場合は協調的な状況をうまく利用した課題解決がみられたが、Prologの場合は「例題プログラムが正しく解を出力することはわかるが、正しく解を出力するプログラムを書くのは難しい」という初学者が多く、課題プログラムの生成過程に「ここから先、どうしたらよいかわからない」と表現されてしまう行き詰まりが予想される。そのため、上の環境に加えて、行き詰まりへの対処に関する支援が必要と考えられる。

そこで本実践では、Prologの構文や実行戦略などの対象知識のほかに、Prologプログラミング初期の行き詰まりの要因と考えられる再帰的プログラミングに関する教授を行った。これは初学者の弱い問題解決を支えるプランニングに関する知識にあたると考えられる。また、同程度の知識に基づく問題解決過程を経験的に共有した仲間同士に起こる情報交換が、問題解決の支援になっていることに注目し、先行して課題を解決した学習者に仲間へ向けた課題解決ヒントの提供を求めた。ヒント集には参加者からの有用性評価に基づく社会的フィルタリングを運用し、共有を行った。図2は、課題perm/2（順列組み合わせの生成)に関するヒント集の一部である。

本稿では、member/2を例題とする課題last/2（リストの最後尾要素の判定）、append/3（2つのリストの結合）を例題とする課題del/3（リスト要素の削除）とperm/2に関する実践を報告する。

図1　例題ページ（member/2)

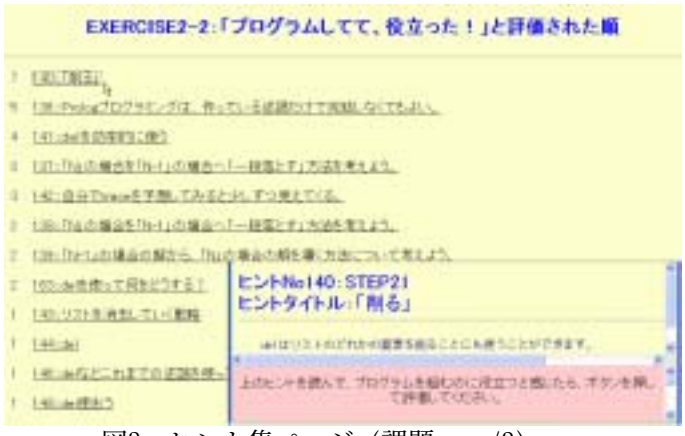

図2　ヒント集ページ（課題perm/2)

3．再帰スキーマ

Prologのリスト処理には、複数の節からなる再帰的定義と、変数の単一化を利用したPrologに特有なコーディング法の適用が要求される。だが、代表的なテキストには、プログラム例と、正しく解を生成する実行過程を示しているだけで、いわば「Prologによる再帰的なプログラムの作り方」にあたる発見

的な知識を明示的していないものが多い。その理由は、他の言語の場合でも同様に、このタイプの知識が、例題や解説を利用しながら正解に至るまでの自分の弱い問題解決過程を通して、学習者自身によって経験的に学習されるものと期待されているからと考えられる。しかし、少なくとも本実践に関する限り、この期待は楽観的すぎると思われる。

本実践では、再帰的プログラムの解釈と生成を支えるために、以下の再帰スキーマを教授した。

(A) 境界条件
(B) 再帰条件
  (B-1) N次の問題をN-1次の問題へ還元する記述
  (B-2) N-1次の問題の解をN次の問題の解と関係づける記述

再帰スキーマの獲得や運用という観点からみると、Prologプログラミング初期の課題解決は、再帰スキーマをプログラム生成へと手続き的に対応づける知識や、具体的なコードを生成するコーディングレパートリを獲得、運用していく過程と考えられる。

講義場面では、例題プログラムを、再帰スキーマがPrologのコーディング法に依存して適用された事例として解説した。適切なコードの生成には、利用可能なコーディングレパートリの組み合わせや変形操作などを行う発見的な適用過程が含まれていることを確認しながら、プログラムを部分的に構成していく過程を示した。

4．運用結果

プログラムが未完成のままでも課題提出(3日ほど後、1週間後)を求めた。表1に各時期の正答提出の割合を示す。特にperm/2の課題解決が少ない。これらの例題や課題が、代表的な再帰プログラムであることと、同じ学習者による手続き的プログラミング言語(たとえばCやRuby)におけるパフォーマンスを考慮すると、Prologの課題プログラム作成は、期待されていたほど易しくはないということだろう。

[ヒント共有]

ヒントは最初の課題提出後から公開された。表1に課題ごとのヒントの利用状況を示す。ヒントは、公開以前にプログラムを完成できなかった参加者のうち平均47.5%に参照された。またヒントを参照した参加者が、課題提出時に特定のヒントを自分のプログラム作成に有用と評価した割合は平均46.9%だった。自分の課題解決にヒントを利用しなかった参加者が、課題解決後にヒントを参照し、有用性を評価した事例もみられる。適切な対照がないが、ヒントの共有は、学習者の課題解決あるいは振り返りや比較について、一定の支援になったと考えられる。

[評価されたヒントの変化]

ヒントを、その内容に関して、実行過程に関するもの（例：トレース結果の解釈）、構文知識の解釈（例：リスト記法）、再帰スキーマ（例：例題を用いた対応の確認）、コーディング（例：節内の変数共有)、問題分析（例：節の意味やコードを示す直接的なヒント)に分類した。表 1 に、分類別に評価をうけたヒント数を示す。繰り返して利用される知識に関するヒント（実行過程や構文知識）は次第に評価されなくなり、課題の問題構造に依存したヒントが評価されていくと考えられる。

[行き詰まりとヒントとの関係に関する事例の考察]

特に難易度が高かった perm/2 の未完成提出 46 のうち、38 例には境界条件(perm(L,L)あるいはperm([],[]))の記述があった。しかし適切な再帰条件コードを生成し正答に至った例はなかった。未完プログラム中の再帰条件に、他の述語を利用している例が見られなかったことから、再帰条件が生成されなかった理由には、対象となる述語のみで定義しようとしたことや、再帰スキーマ中(B-1)(B-2)にあたるコードの探索が単一化を用いたリスト操作に限定されたことが推測される。これらは、この時点までのプログラム例から学習された、いわば「学習者自身による限定」が引き起こした行き詰まりとみられる。この課題で評価をうけたヒント(図 2)は、このような制約の緩和に関係していると考えられる。

5．今後の課題

プロダクションシステムに基づき、Lispプログラミングの学習を分析したAndersonら (1984)は、例題からの構造類推が、初学者の問題解決をガイドすると仮定している。本実践の参加者にみられた多様な行き詰まり事例を問題解決過程と対応づけ、そこから学習支援を試みるためには、行き詰まりを引き起こす、もっと弱い問題解決過程について考察する必要があると考えられる（土屋ら，2004)。

文献

Anderson, J. R. & Farrell, R. & Sauers, R. (1984). Learning to program in LISP, *Cognitive Science, 8,* 87-129.

土屋孝文，市川ひと美，鈴木健志 (2002). 協調的なプログラミング学習環境の設計と実践，2002PC カンファレンス論文集，430-431.

土屋孝文，岡崎加奈，谷志穂 (2004). Prolog プログラミングの協調的学習環境の設計と運用 - 学習初期の問題解決過程について，日本認知科学会第 21 回大会発表論文集.

表１　課題ごとの活動結果

	正答提出 (%)		ヒントの利用 (%)		評価をうけたヒントの種類					
	ヒント公開前	最終〆切	参照	評価	実行過程	構文知識	再帰スキーマ	コーディング	問題分析	その他
last/2	62.2	76.9	45.8	54.6	5	3	2	0	2	1
del/3	80.0	82.7	45.7	38.1	4	0	5	0	2	0
perm/2	0	57.1	51.0	48.0	1	0	4	3	5	0