意味ネットワークによるプログラミングノウハウの表現とプログラミング支援システムへの適用†

芳賀 博英††

知識情報処理技術をプログラム作成に適用するニーズが高まってきた．プログラミングノウハウを利用した知的支援システムは，扱えるプログラムの規模が小さいことや知識ベースの構築が困難であるなどの課題がある．したがって実用的な観点からノウハウの伝授を目的としたコンサルテーションシステムが重要である．従来のコンサルテーションシステムでは，対象とするノウハウがコーディングパターンやそれに類するものに限られていることや，ノウハウの保守性，可読性等についての検討が十分行われていなかった．本論文ではネットワーク型データ表現方式を用いた広範なプログラミングノウハウの表現方式と，それを利用したノウハウの伝授を主としたプログラム作成支援システムの基本構想を述べる．すなわち(a)プログラムの持つ情報をノードとリンクの集合で表現し，(b)プログラミングノウハウをこのリンクで表現するとともに，(c)プログラムの持つ情報をいくつかのカテゴリーに分類し，各カテゴリーごとに関連に従って階層的に管理するようなプログラミングノウハウの表現方式を提案する．論理型言語とC言語を対象として，約240個のノードと250本のリンクを持つプロトタイプシステムをワークステーション上に試作して，本方式の有効性を確認した．

1. はじめに

ソフトウェア危機が言われてから既にかなりの時間が過ぎたが未だに十分な解決策は見出されていない．一方で近年知識情報処理技術の研究が盛んに行われており，既にエキスパートシステムという形で一部が実用化されている．知識情報処理技術を応用したシステムは，人間の持つ知的な機能やノウハウを計算機に組み込み，従来のシステムに比べて柔軟で高度な処理が可能になっている．人間の知的な活動の1つであるプログラミングは，知識情報処理技術の有望な適用分野の1つである．

プログラミングノウハウを利用したプログラミング支援環境としては，プログラミングノウハウの伝授を主目的としたコンサルテーションシステムと，組み込まれたノウハウをプログラムに適用してプログラムを開発する知的支援システムがあり，既にいくつかのシステムが発表されている．まずプログラミングノウハウの伝授を目的としたシステムとしては LISP-PAL[1] がある．LISP-PAL は個々のノウハウをフレームで表現し，自然語による利用者の要求を解析して適切なノウハウを提示する．LISP-PAL のノウハウはある目的（ゴール）を達成するための方法であり，その内容はプログラム例やプログラムスキーマの説明やプログラミングの注意事項から構成されている．またノウハウを適用してプログラムを開発する知的支援システムとしては，Programmer's Apprentice (PA)[2]，PROUST[3]，INTELLITUTOR[4] などがある．PA はプログラミングノウハウをクリシェイというデータで表現している．また PROUST は，典型的なアクション列を表すプログラムの断片であるプランを利用して，プログラミングを支援するシステムである．INTELLITUTOR は，アルゴリズムのレベルから具体的なコーディングのレベルまでの情報をグラフの形式で表現した手続きグラフ[5] を利用してプログラムの理解と診断を目指したものである．

知的支援システムの知識ベースの構築には，複雑なプログラミングに関する知識やプログラムそのものの知識を定められた形式で抽出しなければならず，汎用的な知識ベースを構築するのは困難である．その意味でノウハウの伝授を目的とするコンサルテーションシステムが実用的な観点から重要になってくる．

コンサルテーションシステムが扱うプログラミングノウハウはいろいろなものがある．例えばある目的を実現するために必要な制御構造やデータ構造の選択のしかた，ある抽象的な目的や制御構造を実現するためのより具体的な手段の選択のしかた，ある問題を解くときに利用できる過去の経験の探索法などがある．しかし LISP-PAL ではこのようなノウハウの表現について十分考慮されていなかった．PA，PROUST，INTELLITUTOR は高度な支援機能を目標としているため，知識ベースが複雑になっている．そのためこれらのシステムの知識ベースからプログラミングノウハウを読み取ることは容易ではない．

† Programming Knowhow Representation Method Based on the Semantic Network and Its Application to the Programming Support System by HIROHIDE HAGA (Kansai Systems Laboratory, Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所関西システムラボラトリ

本論文ではプログラミングノウハウの伝授を目的としたコンサルテーションシステムにおいて，上に述べた問題を解決するために，プログラミングノウハウの表現方式とその基礎となるプログラムの持つ情報の表現方式，およびそれを用いたプログラミング環境の概要について述べる．すなわち(a)プログラムの持つ情報をノードとリンクの集合で表現し，(b)各ノードはプログラムの持ついろいろな情報の任意の断片を表現し，(c)情報の関連に従ってそれらの情報を表すノードの間にリンクを定義でき，(d)プログラムの持つ情報をいくつかのカテゴリーに分類し，各カテゴリーごとに関連に従って階層的に管理するようなプログラム情報の表現方式とそれを基礎にしたプログラミングノウハウの表現方式を提案する．本論文ではプログラミングノウハウの表現とはこのリンクの定義にほかならないことを提案する．そしてソースコードに付けられたこれらの情報を利用して，類似プログラムの検索を行うプログラミング環境について述べる．以下第2章でプログラムの持つ情報を，プログラムの意図を表すwhat 情報，プログラミング技法を表す how 情報，ソースコードに分類している．さらにこれらの情報の間の関連がプログラミングノウハウそのものであることを示す．第3章ではこの特徴に基づいて個々のプログラム情報をノードで，ノード間の関連であるプログラミングノウハウをリンクで表現するプログラム情報表現方式を提案するとともに，情報の関連を示すための5種類のリンクを提案する．またこの情報表現方式にもとづいて，類似プログラムの検索を基礎としたプログラミング支援システムについて述べる．第4章ではプログラミング支援システムのプロトタイプの開発と使用例，評価について述べる．

2. プログラミングノウハウについての考察

本論文で考察するプログラミングノウハウとは，例えば「○○という問題は××という技法を使えば解決できる」とか，「この問題は以前解いた△△という問題に似ているから，それを利用する」あるいは「□□という技法の具体的なコーディング例は◇◇というソースコードである」というものである．つまり，ある問題が与えられたときに，その問題を解くのに必要な技法を知っているとか，抽象的な問題を与えられたときにそれをより具体的な問題に分割／詳細化する過程，あるいは以前に類似の問題を解いたときの経験を活かす，さらに個々のプログラミング言語に特有のコーディングテクニックを知っているというものである．

このようなプログラミングノウハウは，プログラムの持つ各種の情報の間の関連によって表現できる．つまり「どのような問題を解いたのか」を示すwhat 情報と「問題をどのようにして解いたのか」を示すhow 情報とプログラムを実現しているソースコードの間の関連で表現できる．例えばソーティングという処理を考えると，「論理型言語でソーティングを実現するためには，リスト表現された入力列に対してパターンとのユニフィケーションを行い，条件再帰を利用し空リストになったら処理を終了する」というノウハウが，「ソーティング」という what 情報と「パターンとのユニフィケーション」「条件再帰」「取り尽くし的条件判定」という how 情報の間のリンクで表現できる．また「条件判定を含む再帰構造の具体的なソースコードの例はリストを平坦化するプログラムである」というノウハウが，「条件再帰」という how 情報と flatt というソースコードの間のリンクで表現できる．

一般的に言って，how 情報とはプログラムの対象領域（ドメイン）には依存せず，処理内容（タスク）の依存度の高い情報であり，what 情報とは逆にドメインへの依存度が高く，タスクにあまり依存しない情報のことである．しかし必ずしも常に how 情報と what 情報を明確に区別できるとは限らず，例えば“楕円の描画方法”という情報は，その情報が使われる文脈に応じて how 情報とも what 情報とも考えられる．本システムではこのような情報に対して，あえて無理に分類するのではなく，どちらに分類しても良い，あるいは両方に所属する情報であると解釈する立場を取ることとした．そして文脈に応じて適切な方から情報を参照できる機構を用意することで情報を利用できるようにすることとした．したがって見かけ上は同じ情報が how 情報にも what 情報にも含まれているという場合があるが，それらの情報の内容は異なっている．

抽象的な仕様や技法をより具体的なものに分割／詳細化してゆく過程や，ある目的の実現のための代替案の表現というプログラミングノウハウは，what 情報と how 情報の階層構造で表現できる．what の階層とは次のようなものである．例えば先ほどのソーティングという処理を考える．ソーティングは要素の順序の変更という観点から見ると，「順序変更」というより抽象的な what を持っている．同じ what を持つ処

理としては，例えばリストの要素の順序を逆転する処理がある．さらに「順序変更」という what は，より抽象的な「リスト操作」という what を持っている．このように what 情報は IS_A 関係[6]に基づく階層的な構造になっている．また例えば論理型言語では，場合分けを行うには述語の頭部の単一化で行う方法と，判定型述語で行う方法がある．この場合，「場合分け」という how 情報と「頭部単一化」，「判定型述語の利用」という how 情報が IS_A の階層構造になっている．このような関係を利用することにより，how 情報と what 情報の階層構造を構築できる．how 情報と what 情報の階層構造を用いると，例えば「ツリー処理は構造データ処理の一種であり，他にはリスト処理，集合処理などがある」とか「場合分けを実現するためには，論理型言語では頭部単一化と判定型述語の利用という2つの方法がある」といったノウハウが，「リスト処理」「構造データ処理」「ツリー処理」「集合処理」「場合分け」「頭部単一化」「判定型述語の利用」という what 情報や how 情報の間の階層構造で表現できる．したがって what 情報と how 情報とソースコードの関連を表現することと階層構造を定義することが，プログラミングノウハウを表現することになる．ベテランプログラマほどこのようなノウハウを多く持っており，必要に応じて有効に活用している．

3. ネットワークによるプログラム情報とプログラミングノウハウの表現方式の提案[7]

3.1 基本方針

第2章で述べたような特徴を持つプログラム情報とプログラミングノウハウを表現するには，what 情報，how 情報，ソースコードという個別の情報と，それらの情報の関連を自由に表現できなければならない．そこでプログラム情報の表現方式の設計の基本方針を以下のとおりとした．

（1） 個々の情報の断片（fragment）をノードで，ノード相互の関連をリンクで表現する．1つの断片は任意の情報の任意の大きさの塊である．各ノードにはそのノードを特定するための名前と情報の種類を示す属性をつける．各ノードは，利用者の考え方を反映した任意の情報を表現する．

（2） 情報の断片の間の関連を示すためにリンクを定義することができるようにする．リンクは任意の2つのノードを結ぶことができる．リンクにはそのリンクの意味を示すラベルが付けられる．リンクによって結ばれたノードは，リンクの意味に対応したなんらかの関係があることを示している．

（3） 複数のノードの集合であるワールドという概念を導入する．あるワールドに含まれているノードは，ある見方から見て同じ種類のノードである． ■

以上の原則の元でプログラム情報とプログラミングノウハウを表現するためには，what 情報と how 情報，ソースコードのそれぞれをノードに割り当て，それらのノードの間の関連を示すリンクによってプログラミングノウハウを表現すれば良い．各ノードには名前をつけることができるため，what 情報と how 情報については「ソーティング」「リスト処理」「場合分け」「再帰」などの情報の名前をノードの名前とする．一方ソースコードは，手続き名，関数名，述語名などをノードの名前とすることができるが，それだけでは文字列そのもので表現されたソースコードを表現することができない．そこでノードの「内容」という概念を導入する．ノードの内容とはノードの名前の具体化された情報である．これによってソースコードでは手続き名，関数名，述語名などで示された抽象的な情報の具体的な内容を表現することが可能になる．また what 情報と how 情報のノードでは，ノードの内容はノードの名前で表現された情報の具体的な説明となる．各ノードの内容を表すテキストの任意の部分には，他のプログラム情報へのリンクを定義することができる．つまり各ノードの内容はハイパーテキスト[12]的な機能を持っている．

リンクの種類としてシステムがあらかじめ用意するシステム定義リンクと，ユーザが自由に定義することのできるユーザ定義リンクの2種類を設ける．システム定義リンクとしては，個々のワールドの中の階層構造を示すリンクと，ワールドの要素の間の関連を示すリンクがある．ユーザ定義リンクは使用するプログラミング言語や動作環境に依存するようなノウハウの表現に用いる．例えばあるプログラムにおいて，特定の形式をした外部データベースを利用している場合などは，ソースコードあるいは目的を表すノードから，「データベースの使用」あるいは「××型データベースの使用」というノードに対して，「使用データベース」というラベルのリンクを定義することによって，そのデータベースの使用に関するプログラミングノウハウを表現できる．これらのリンクによって広範なノウハ

ウが表現できる.

複数のノードの集合であるワールドを導入することによって, what 情報, how 情報等のプログラム情報を体系的に理解することができる. 例えば how 情報のワールドには, ある言語のプログラミング技法の体系が抽象的なレベルから具体的なレベルまで体系的に表現されている. また what 情報のワールドでは, システムが支援することができるプログラミングの目的がやはり体系的に整理されている. ソースコードのワールドには, 利用できるライブラリが示されている. これらを利用することによって, 目的とするプログラム情報に迅速にアクセスすることができる.

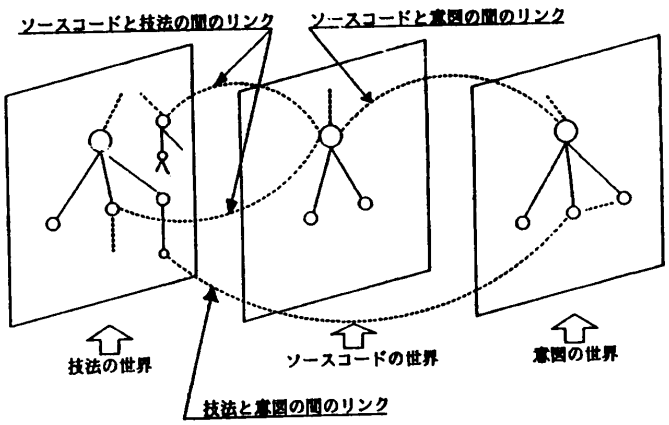


図 1 プログラム情報管理方式の概念図
Fig. 1 Conceptual structure of the program information management method.

過去の事例との類似性による経験の利用については, 個々のソースコードのノードに定義されている how 情報と what 情報へのリンクを用いて, 同じ技法を使っていたり意図を持っているプログラムは類似していると考えることによって実現できる. また what 情報と how 情報の階層はそれぞれ IS_A 階層で関連付けられているので, what 情報や how 情報の類似性が自然に表現できる. 例えば論理型言語の技法を表す階層構造では, IS_A の関係にある「場合分け」というノードと「頭部単一化」というノードは類似していると考えることができる. この類似性を利用して類似のプログラムを検索することができる. 例えばあるソースコードが "頭部単一化" という how 情報とリンクで結ばれていて, ユーザは "場合分け" という技法を使ったプログラムを検索したい時に, 2つの情報が IS_A の関係にある事を利用して, "場合分け" というノードと結ばれているソースコードがなくても, 下位概念である "頭部単一化" というノードと結ばれているソースコードを類似のソースコードとして検索してくることができる.

以上のように, この基本方針を実現することによって, 第2章で述べたプログラム情報とプログラミングノウハウを表現することができる. **図1**にプログラム情報管理方式の概念を示す. 図1で小さな丸で示したものが個々のプログラム情報を表すノード, ノードの集合の平面がワールドを示している. また個々のノードを結ぶリンクが, ノード間の関係を表現している.

3.2 システム定義リンクの種類

本プログラム情報表現方式で用いるシステム定義リンクには次の5種類がある.

(a) 抽象具象リンク (AC-Link): how 情報と what 情報のノードにおける抽象-具象の関係を表現するリンク.

(b) 部分全体リンク (PW-Link): ソースコードのノードにおける全体-部分の関係を表現するリンク.

(c) ソースリンク (S-Link): ソースコードを表すノード以外のノードから, そのノードに関連するソースコードへの関連を表現するリンク.

(d) how リンク (H-Link): how 情報を表すノード以外のノードから, そのノードが関連する how 情報への関連を表現するリンク.

(e) what リンク (W-Link): what 情報を表すノード以外のノードから, そのノードが関連する what 情報への関連を表現するリンク.

what 情報と how 情報の階層構造は AC-Link で, ソースコードの階層構造は PW-Link で表現することができる. またプログラミングノウハウは, S-Link, W-Link, H-Link でそれぞれのソースコードノード, what 情報ノード, how 情報ノードを結ぶことによって表現することができる. この5種類のリンクによって第2章で述べたプログラミングノウハウを表現できる.

3.3 ネットワーク型プログラム情報表現方式を基礎としたプログラミング支援システムの概要

3.1 節, 3.2 節で述べたプログラム情報表現方式によって, 次のようなプログラミング支援システムが実現できる.

利用者は自分の作成したプログラムを表すキーワードを用いて, プログラミングノウハウもしくはプログラム作成の参考となるソースコードを検索する. ノウハウの検索は, そのノウハウに含まれるプログラム

情報のノードに定義されているリンクをたどることによって行う．類似のプログラムを検索する場合には，システムは入力されたキーワードの集合に最も近い特徴を持つプログラムを検索してくる．検索されたプログラムは，通常は目的とするプログラムとは異なっている．利用者は検索されたプログラムに付けられたwhat 情報や how 情報などを利用して，目的とするプログラムと検索されてきたプログラムの違いを判断し，検索されてきたプログラムを目的プログラムに改造する．このとき改造のための what 情報や how 情報，プログラミングノウハウは，それぞれの情報を表現するノードに付けられたコメントや S-Link，H-Link，W-Link などのシステム定義リンクやユーザ定義リンクを参照することによって得られる．このようにして既存のプログラムのコードだけでなく，技法やプログラムの意図，プログラミングノウハウなどのより広い情報を再利用できるプログラム作成支援環境を構築できる．

入力されたキーワードと最も近い特徴を持つプログラムを検索するときの，キーワードの集合と個々のプログラムの間の類似度は次のように定義する．

IK_i $(i=1,\cdots,m)$: 入力されたキーワード

GK_j^l $(j=0,\cdots,n;\ l=1,\cdots,L)$: プログラム P_l に付けられたキーワード

ここで

$$\delta_j^l=\begin{cases}\alpha^n: IK_i\equiv GK_j^l \text{ となるキーワードのペアがある}\\ 0: \text{それ以外}\end{cases}$$

$$\xi_j^l=\begin{cases}1: IK_i\equiv GK_j^l \text{ となるキーワードのペアがない}\\ 0: \text{それ以外}\end{cases}$$

とする．ここで $A\equiv B$ は2つのキーワード A と B が AC-Link で結ばれていることを示し，α は $0\leqq\alpha\leqq1$ の実数である．n は2つのキーワードをマッチングするために AC-Link を上下した回数である．このときキーワード同士の一致度と不一致度を表す値 $simm_l$ と $diff_l$ を次のように定義する．

$$simm_l=\sum_j \delta_j^l \qquad diff_l=\sum_j \xi_j^l$$

このとき入力されたキーワードの集合とプログラム P_l の間の類似度 S_l を次のように定義する．

$$S_l=simm_l/(simm_l+diff_l)$$

この定義によれば，入力キーワードと完全に一致するプログラムでは $S_l=1$ となり，全く一致しないプログラムでは $S_l=0$ となる．それ以外の場合は S_l は0から1の間の値をとる．

4. プロトタイプシステムの開発[8]と評価

4.1 機能とインプリメントの概要

第3章で述べたプログラム情報管理方式を基礎としたプロトタイプをワークステーション上に開発した．対象言語は論理型言語[9]とC言語[13]である．本システムは次のような機能を提供している．

（1） 1つのノードの情報をカード型データベース[10]における1枚のカードのイメージで表示する．このカードの上には，ノードの名称，ノードの種類，ノードの内容，そのノードに定義されているリンクの名称などが表示される．

（2） ワールドは AC-Link もしくは PW-Link を基とした階層構造によって表示する．

（3） ノードを検索する手段としては，名前で直接指定して検索する手段と，リンク情報を指定して検索する方法を提供している．後者は例えば「○○という意図を持ったソースコード」あるいは「××というソースコードに使われている技法」という形で検索する手段である．

（4） what 情報と how 情報の階層性を利用して，与えられたキーワードから階層を上下することによって，類似の情報を持つプログラムを検索することができる．

（5） プログラムや説明文の任意の部分をノードとして定義でき，そこから他のノードへのリンクつけることができる．

現在のプロトタイプシステムのデータベースには約100個の再利用可能なライブラリプログラムと60個の how 情報，80個の what 情報が組み込んである．how 情報としては逐次実行や繰返し，判定処理などの基本的なプログラミング手法の実現法などのプログラミング技法が組み込んである．what 情報としては，対象をリスト処理とグラフィックス処理に限定し，リスト処理の基本的な処理やグラフィックス処理におけるマウスやマルチウインドウの処理についての情報を組み込んである．またライブラリプログラムとしては，基本的なプログラミング技法を説明するためのユーティリティ的なプログラムと，リスト処理，グラフィックス処理を行うプログラムが登録してある．これらの how 情報，what 情報，ライブラリプログラムの間に約250本のリンクを定義し，各種のプログラミングノウハウを表している．これらのデータは，各言

語のマニュアルや解説書などを参考にし，各マニュアルなどの記述体系を“抽象一具象”の立場から再構築することによって作成した．またライブラリとしてはマニュアルなどに記載されているプログラム例を利用した．**図2**はノードの一例である．この図の①はソースコードを表すノードを開いた状態を示している．また①の(*)の部分は，このソースコードのノードに定義されているリンクの一覧表を示しているエリアである．このノードにはプログラムの目的を表す“意図”，そのプログラムに使われているプログラミング技法を表す“技法”というリンクが定義されている．②は how 情報のノードの1つである‘単純再帰’のノードを開いた状態である．このノードにはこの技法の上位概念へのリンクを表す“上位”というリンクとこの技法を使ったプログラム例へのリンクを表す“ソースコード”というリンクが定義されている．また**図3**は本システムに定義されている how 情報のワールドの一部を階層構造で示したものである．類似の特徴を持ったノードを検索するときには，この階層構造を利用する．本システムのプログラム情報とプログラミングノ

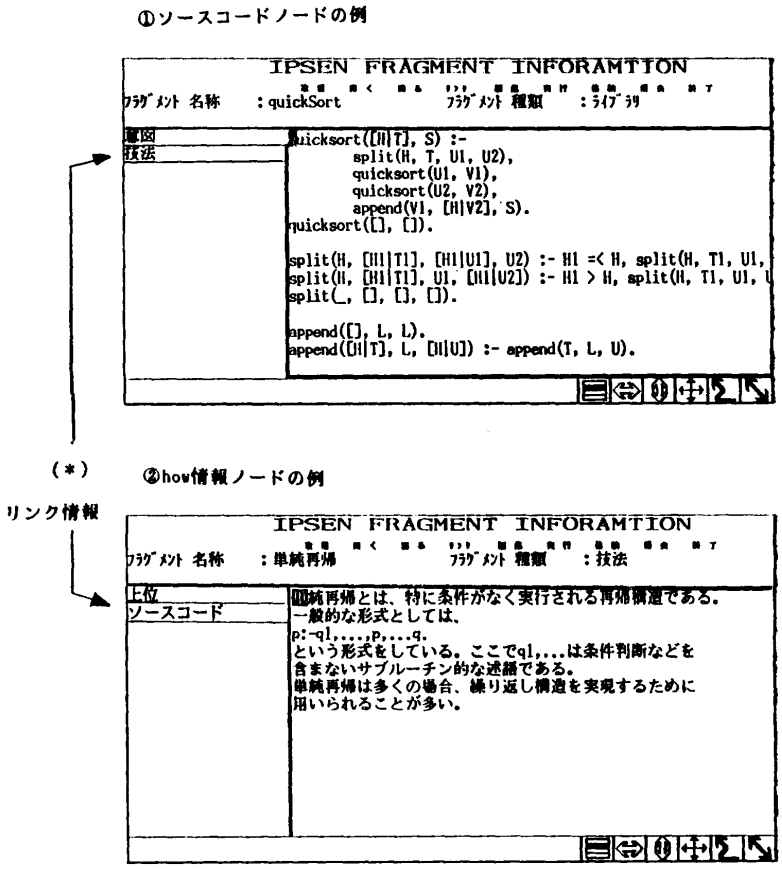


図2 ソースコードノードと how 情報のノードの例
Fig. 2 Examples of source code node and how-information node.

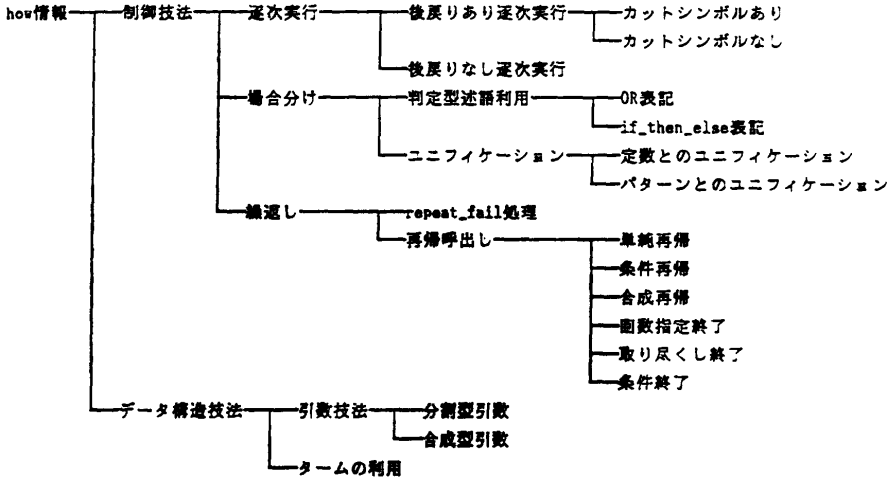


図3 How ワールドの階層構造
Fig. 3 Hierarchy structure of the “How world.”

ウハウの表現方式は LISP-PAL, PA, PROUST, INTELLITUTOR などに比べて,

(a) プログラム情報を3種類に分類したことと同一の種類の情報の集合であるワールドの概念を導入したことにより, 情報を整理しやすい

(b) プログラム情報の関連を5種類のリンクによって表現することにより, 相互の関連を明確に定義できる

(c) プログラム情報自身の表現としてキーワードと自然語によるコメントを用いることにより, 容易に理解できる

(d) 新しいプログラム情報やプログラミングノウハウを追加したり古い情報を更新する際にも, ノードの定義と他のノードとの関連を示すリンクを追加/更新するだけで良いため, 保守性が高いなどの特徴がある.

4.2 システムとの対話例

以下に本プロトタイプシステムを利用したプログラム作成過程におけるシステムとの対話例を示す. ここでは論理型言語で次のようなプログラムを作成することを想定する[11].

> リストの要素がリストである場合には, その要素となっているリストの要素をも反転するプログラム full_reverse を作成せよ.

full_reverse プログラムは次のような処理を行うプログラムである.

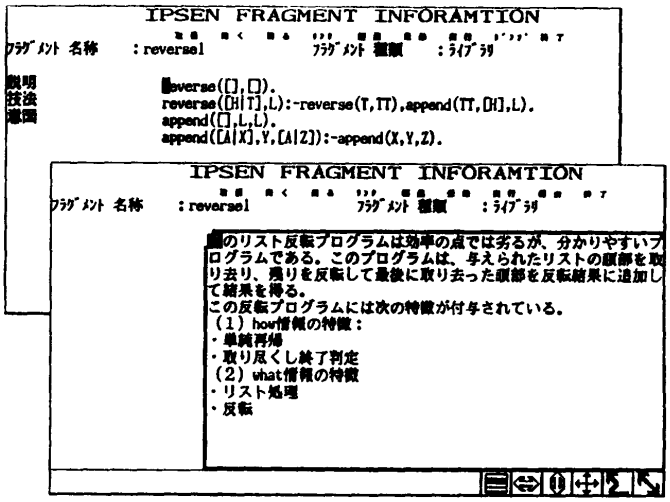
[a, b, [c, d], e]→[e, [d, c], b, a]

[[a, b], [c, d]]→[[d, c], [b, a]]

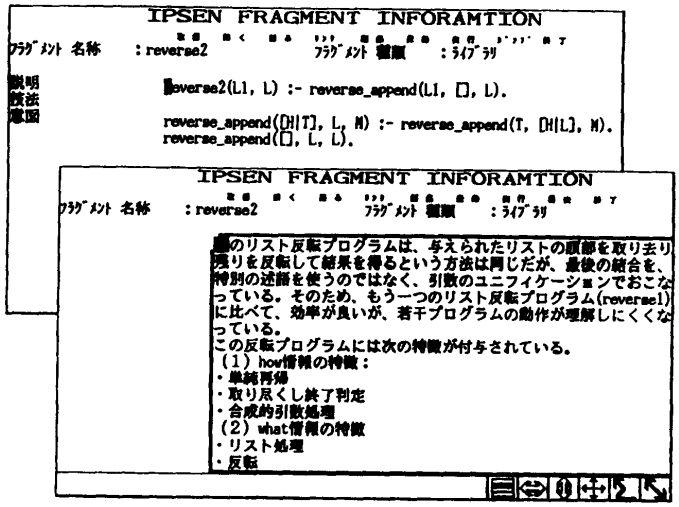
ユーザとシステムは次のような対話を行う.

(1) ユーザは自分の行いたいことを示すキーワードである「リスト処理, 逆転, 並べ替え」などを入力する.

(2) システムはこのキーワードとできるだけ一致するライブラリ要素を検索する. ライブラリには, 要素を単純に反転するプログ



(a) 特徴: 単純再帰, 取り付くし終了判定, リスト処理, 反転



(b) 特徴: 単純再帰, 取り付くし終了判定, 合成的引数処理 リスト処理, 反転

図 4 検索された2つのソースプログラムとその特徴
Fig. 4 Two source program nodes and their features.

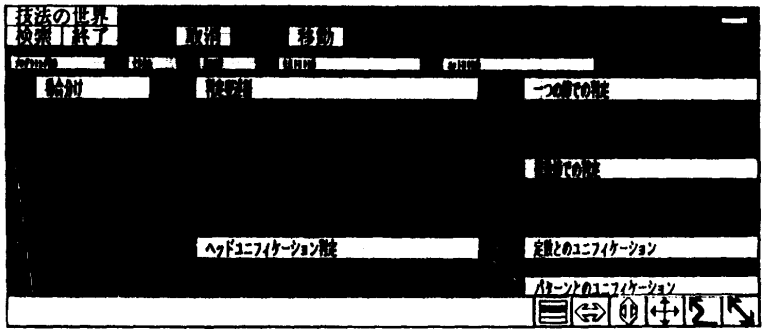


図 5 「場合分け」の部分木
Fig. 5 Subtree of the "Branching."

ラムとして**図4**のような2つのプログラムが登録されている.

(3) このとき3.3節で定義した類似度を計算して,類似性が高い(a)のプログラムが,この検索では適切であるとして表示される.

(4) ユーザはこのプログラムに対して,いくつかの入出力例を与える.すると要素がアトムであるリストに対しては正しく動くが,そうでない場合は正しくないことがわかる.このことからユーザは場合分け(要素がアトムであるか否か)が必要だと判断する.そこでユーザは「場合分け」の技法についての説明を求める.

(5) システムは説明の要求に対して,**図5**のような技法の部分木を表示する.

(6) 要素がアトムか否かを判定するのであるから,ユーザは場合分け技法のうちで「判定型述語の利用」という技法を選択しそのノードを開く.

(7) システムは「判定型述語の利用」のノードの内容を表示する(**図6**).

(8) ユーザは「判定型述語の利用」の具体例を見るために,「ソースコードリンク」を選択する.

(9) システムは S-Link で結ばれたプログラム例を表示する(**図7**).

(10) ユーザはこのプログラムを参考にして,正しいプログラムを作成する.

以上のような対話を行うことにより,**図8**のような目的のプログラムを作成することができる.

以上の対話例からわかるように,本システムを利用することによって,ユーザは既存のいろいろなプログラム情報やプログラミングノウハウ,さらにはライブラリプログラムを参考にして,必要とするプログラムを作成することができる.このとき,表示されたソースコードが理解できなければ,ソースコードの部分に定義されたリンクをたどる.例えばある目的を実現するためのプログラミング言語に特有のコーディングをしている部分には,あらかじめプログラミング技法やプログラムの意図へのリンクを定義しておくことにより,プログラムの理解が促進される.これらの機能を利用することによって,プログラミングノウハウを知らない初心者のプログラマでも,あたかもベテランプログラマがそばにいてアドバイスをしてくれるように,参考となるプログラムを理解でき,目的のプログラムを短期間で完成させることができる.

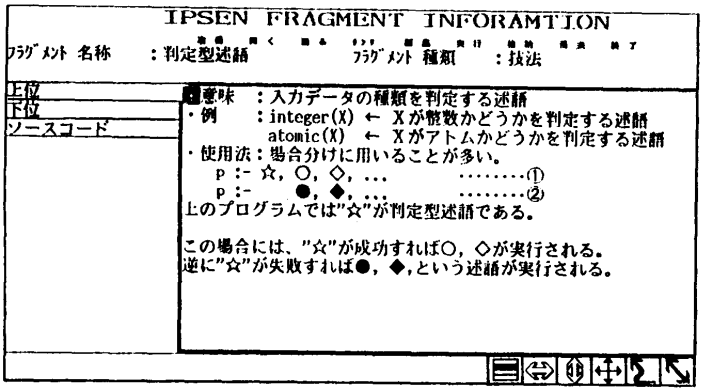


図6 「判定型述語利用」のノードの内容
Fig. 6 Contents of the node "Using Judge-type Predicate."

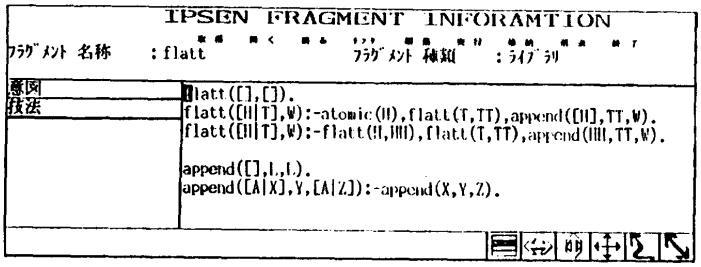


図7 「判定型述語利用」のソースコードの例
Fig. 7 Source code example of node "Using Judge-type Predicate."

```
full_reverse([],[]).
full_reverse([H|T],W):-atomic(H),!,
        full_reverse(T,TT),append(TT,[H],W).
full_reverse([H|T],W):-
        full_reverse(H,HH),full_reverse(T,TT),append(TT,[HH],W).
```

図8 作成した full_reverse のソースコード
Fig. 8 Source code of "full_reverse."

4.3 評　価

本システム開発の第1の目的であるプログラム情報およびプログラミングノウハウの表現については,プログラムの情報を3種類に分類することと,各情報間のリンクによってノウハウを表現することによって,広範なノウハウをわかりやすく表現できた.プログラミングノウハウについては,「○○ という意図を実現するには,どのような技法を使えば良いか」とか,「△△ という技法を使うためのソースコードの具体例はどのようなものか」,あるいは逆に「×× というソースコードの意図あるいは技法は何か」といったノウハウが,H-

Link, W-Link, S-Link などのリンクを利用することによって表現できる．これによってベテランプログラマの持っている各種のプログラミングノウハウが表現できる．同時にソースコードに付加した H-Link, W-Link によって，プログラムの可読性が向上し，既存のプログラムの理解，ひいてはプログラムの再利用が促進できる．

what 情報と how 情報の AC-Link を利用した類推的な検索機能を利用することによって，必要なプログラムに類似したプログラムを検索することができるため，ライブラリ内に完全に一致するプログラムがない場合でも，類似のプログラムを改良することによって，プログラム作成に利用することができ，システムの利用の範囲が拡がった．またライブラリプログラムにはそれぞれ what 情報や how 情報が付加されているため，自分の目的とするプログラムとどこが違うのかを把握することが容易になり，プログラムの改良がスムースに行える．また what 情報と how 情報の階層構造は，一種の設計ガイダンスとして利用できる．つまり例えばある意図をより具体化する場合，あるいはある技法のより具体的な方法を知りたい場合には，what および how のワールドの階層構造を上下に移動することによって，知ることができる．

今後の研究課題としては，what 情報および how 情報の付与の機械化，より柔軟なユーザインタフェースの提供，ソースコードだけではなく過去の設計過程そのものを有効に活用するための方式の研究などがある．またテスト・デバッグについても検討が必要である．これについてはプログラム作成時に利用したノードにあらかじめテスト・デバッグのための情報，例えばライブラリであればそのテストケースの例，how 情報や what 情報のノードであればそれらのノードの内容に特有のデバッグ手順の情報を，“テスト・デバッグ”というリンクを定義することによって利用者に示す方法などがある．

5. まとめ

プログラミングノウハウを利用したプログラム作成を支援するためのノウハウの表現方式とその基礎となるプログラムの持つ情報の整理方式，およびそれを利用した知的プログラミング環境の基本構想について報告した．本論文でのべたプログラム情報管理方式の特徴は，次のとおりである．

（1） プログラムの持つ中心的な情報を，そのプログラムが何をするものかを表す what 情報，どのように実現しているかを表す how 情報，ソースコードの3つに分類した．

（2） プログラムは1つのノードで表す．またある一定の種類のノードの集合であるワールドを導入した．

（3） 個々の情報の関連を表現するために，2つのノードを結ぶリンクを定義する．リンクには中心的なプログラム情報の関連を示すための5種類のシステム定義リンクと，それ以外の情報の関連を示すためのユーザ定義リングがある．

（4） プログラミングノウハウを what 情報，how 情報の階層構造およびこれとソースコードの間の関連で表現し，この表現のためにシステム定義リンクを用いる．

この情報管理方式を用いて，類似のプログラムを参考にして，目的のプログラムを作成することを支援するプログラミング支援環境の提案を行った．類似プログラムの検索のときの類似度としては，what 情報と how 情報の階層性と不一致であるという情報を利用する方式を提案した．以上の提案に基づき，論理型言語を対象としたプロトタイプシステムを開発した．このシステムでは1つのノードを一枚のカードのイメージで表現し，このカードの上に各種の情報を表現する．個々のカードにはいくつかのリンクが定義されており，ユーザはそのリンクをたどって必要な情報やプログラミングノウハウにアクセスできる．プロトタイプシステムは論理型言語とC言語を対象として，リスト処理とグラフィックス処理の領域の約100個のライブラリプログラム，60個の how 情報，80個の what 情報と約250本のリンクが定義され，プログラミングノウハウを表現している．

謝辞 本研究の機会を与えていただいた(株)日立製作所システム開発研究所所長堂免信義氏，主管研究員石原孝一郎博士，第5部部長森文彦博士，関西システムラボラトリ長増位庄一氏，有益な討論をしていただいた主任研究員中所武司博士，プロトタイプシステムのインプリメントにご協力をいただいた藤波努氏に深謝いたします．

参考文献

1) 上原ほか：LISP-PAL：プログラミング支援のための自然語による質問応答システム，情報処理学会論文誌，Vol. 30, No. 11, pp. 1413-1423 (1989).

2) Waters, R. C.: The Programmer's Apprentice: A Session with KBEmacs, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 11, pp. 1296-1320 (1985).
3) Johnson, W. et al.: PROUST: Knowledge-based Program Understanding, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 3, pp. 11-19 (1985).
4) 上野：知的プログラミング支援システム INTELLITUTOR について—背景と開発思想—，情報処理学会知識工学と人工知能研究会資料，37-5 (1984).
5) 上野ほか：プログラム理解による論理チェックシステム，昭和 62 年度人工知能学会全国大会論文集，pp. 449-452 (1987).
6) Barr, A. et al. (田中，淵訳)：人工知能ハンドブック第Ⅰ巻，p. 221，共立出版，東京 (1983).
7) 芳賀ほか：プログラム作成支援システムにおけるプログラム情報管理方式，日本ソフトウェア科学会第 5 回全国大会論文集，B 1-1，pp. 33-36 (1988).
8) 藤波ほか：意図・技法を備えたプログラム作成支援システム，第 38 回情報処理学会全国大会論文集，6 L-8，pp. 1195-1196 (1989).
9) Sterling, L. et al.: *The Art of Prolog*, The MIT Press, Cambridge, Mass. (1986).
10) Goodman, D.: *The Complete HyperCard Handbook*, Bantam Books, New York (1987).
11) 有川：類推の理論，知識の獲得と学習（大須賀，佐伯編），オーム社，東京 (1987).
12) Conklin, J.: Hypertext: An Introduction and Survey, *Computer*, Vol. 20, No. 9, pp. 17-41 (1987).
13) カーニハンほか（石田訳）：プログラミング言語 C，共立出版，東京 (1981).

付録 類似度の解釈

本論文で提案した類似度の意味について述べる．K_i を事例 i に付与されたキーワードの個数，G を検索キーワードと一致したキーワードの個数，N を検索キーワードと一致しないキーワードの個数，S_i を単純な数だけで定義する類似度，M_i を本論文で提案した類似度とすると，$G+N=K_i$ であるから，結局 $S_i=M$，$M_i=G/K_i$ となる．

一般に情報に与えられているキーワードの数にはばらつきがある．そのばらつきはその情報の曖昧度あるいは一般性の度合いに対応すると考えて良い．つまり曖昧な，あるいは一般性が高い情報ほどキーワードの数が少ないと考えても良い．したがってキーワードの数は，その事例の一般化，あるいは曖昧さの程度を表していると考えられる．単純方式では，この情報を十分活用していない．本論文で主張する類似度の定義は，事例の付与されたキーワードの数に反比例して，類似度が下がってくる．つまり，ある事例に多くのキーワードが与えられているにもかかわらず，検索に利用したキーワードと一致するキーワードが少ないということは，よりその事例が好ましくない，あるいはその事例が詳細な情報であり，利用者が望む情報と一致する確率が低いという風に解釈できる．別の見方をすれば，利用者が曖昧な要求しか入力しない場合には，まずできるだけ一般的な情報を検索してきて，その後でより詳細な情報を検索する，という使い方が自然であるが，本論文で提案する類似度の定義はこの要求を満たすものである．4.2 節で取り上げた例について述べる．利用者の入力したキーワードと類似していると判定した 2 つのプログラム reverse 1 と reverse 2 について，それぞれの類似度を計算すると，$\alpha=0.6$ として reverse 1 の類似度は 0.65，reverse 2 の類似度は 0.49 となる．この例では利用者の入力したキーワードの中に reverse 2 の特徴である“合成的引数処理”というキーワードが含まれていないため，利用者の要求する情報のうちで最も一般的なプログラムである reverse 1 が検索されることになる．このようにこの類似度は「利用者が入力した情報に類似した最も一般的なプログラム」を検索することが可能である．

(平成元年 10 月 11 日受付)
(平成 2 年 5 月 8 日採録)

芳賀 博英（正会員）

1954 年生．1978 年同志社大学工学部電子工学科卒業．1980 年同大学院工学研究科修士課程修了．同年(株)日立製作所入社．現在同社システム開発研究所関西システムラボラトリに勤務．知的プログラミングシステム，プログラミング用エキスパートシステム等の研究に従事．ハイパーテキストシステムとその応用，CAI 等にも興味を持っている．日本ソフトウェア科学会，人工知能学会，IEEE Computer Society, The Association for Logic Programming 各会員．