ソフトウェア基礎論 29−3
(1989. 5. 18)

制約指向論理型言語に基づく対話的問題解決環境

Interactive Problem Solving Environment based on Constraint Logic Programming Language

清水 広之　和気 朝臣　細野 善久　山岡 孝行　竹内 彰一
Hiroyuki SHIMIZU　Tomoomi WAKE　Yoshihisa HOSONO　Takayuki YAMAOKA　Akikazu TAKEUCHI

三菱電機 (株) 中央研究所
Central Research Lab. MITSUBISHI Electric Corp.

あらまし　知識メディアステーションは多様な知識を扱う能動的メディア「知識メディア」を目指したものである．知識メディアステーションでは，知識表現，問題解決用言語として制約指向論理型言語を採用し，その実行環境として Incremental Query, Interactive Query Revisionを拡張した対話的問題解決環境を導入した．本論文では，知識メディアステーションの概要について述べ，制約指向論理型言語τ，及びそれに基づく対話的問題解決環境について述べる．制約指向論理型言語は問題に対する高い宣言的記述能力を持ち，柔軟な知識表現が可能である．この制約指向論理型言語と対話的問合せ機構とは極めて親和性が高く，Incremental Query などの prolog をベースとした問合せモデル以上に柔軟な対話的問題解決環境を提供できる．

Abstract　Knowledge Media Station aims at the active knowledge medium not only to store and apply knowledge represented by knowledge representation language, but also to generate, integrate and propagate various types of knowledge. In knowledge Media Station, we adopt constraint logic programming language "τ" for knowledge representation and problem solving language. And we introduce interactive problem solving environment that has extended Incremental Query and Interactive Query Revision. In this paper, we describe the overview of Knowledge Media Station at first, and then explain this constraint logic programming language "τ", and interactive problem solving environment based on this language "τ". Constraint logic programming language has a high ability to describe problems declaratively and enables to represent knowledge flexibly. The combination between constraint logic programming language and interactive query mechanism offers more flexible and harmonious interactive problem solving environment than interactive query models based on Prolog like Incremental Query and Interactive Query Revision.

1．はじめに

人間にとって知識を表現する方法としては論理式やルールなどのような形式以外に，文書，表，図形，グラフといった様々な形態のものが考えられる．知識を保存するメディアとして，従来から，本，書類，フィルム，テープなどのメディアがある．これらの**伝統的メディア**は知識を保存し，その内容を忠実に再現するだけの受動的なメディアである．

今日では，計算機を多形態の知識を蓄積するメディアとして利用しようという大きなトレンドがあり，ワードプロセッサやパーソナルコンピュータ上の文書処理，ＤＴＰ，表計算，データベースソフトなどに具体例を見ることができる．これらのメディアは知識の記録，再現のほか，それを対話的に加工する能力があり，伝統的メディアよりも能動的であるといえる．しかし，これらのメディアは蓄積しているものの内容を理解しておらず，それゆえ知識を扱うメディアとはいえず，むしろ**情報メディア**と呼ぶべきである．

知識は相互作用により新たな結論や矛盾を導き出すという意味で生きている．知識処理とは蓄積された知識を活用し，計算機が自動的にあるいは人間を支援し

て，新たな知識を生み出すことを可能とするものである．**知識メディア**は，このような知識処理技術を用いて，まさに知識を扱えるよう情報メディアの能動性を高めたものである．知識メディアは知的作業における人間の記憶能力や問題解決能力を増幅するメディアであり，究極的には人間と機械の高度な知的協同の実現を目指している．[1],[2],[3],[4]

このような知識メディアを実現する上で重要となるのは以下のような点であると考えられる．

①開かれた知識表現

計算機が解釈可能な言語による表現以外にも文書，表，図形，イメージなどの人間が通常用いる表現も許されなければならない．

②知識の活用

知識を生かすためには，多様な表現の知識を単に蓄積するだけでなく，質問や検索要求に対して蓄積した知識を総動員して推論し，答えを導くというように知識を十二分に活用できなければならない．

③知識の整理，体系化

多様な知識を様々な角度から検討し，関連付け，整理，体系化していくことは極めて本質的な知識処理作業であり，従来の階層的ファイルシステムを越えた知識の整理，体系化の機構が必要である．

④対話的メディア

人間の記憶能力，問題解決能力を増幅させるためには，内部の知識や動作は人間に対して透過であり，欲するときに随時関与できるように開かれていなければならない．

知識メディアステーションは，上で述べたような知識を扱う能動的メディア「知識メディア」の実現を目指したものである．知識メディアステーションは，ルールなど知識表現言語で記述された知識に限らず文書，表，図形，イメージなど様々な形態で表現されたものを知識として扱い，それらの知識の整理，体系化を促進し，また蓄積された知識に基づく問題解決を行うことにより，人間の行う知的作業を多角的に支援することを目的とするものである．[1]

知識メディアステーションでは，知識表現，問題解決用言語として制約指向論理型言語[5],[6],[7]を採用し，その実行環境として Incremental Query, Interactive Query Revision [8],[9],[10]を拡張した対話的問題解決環境を導入した．

本論文では，まず知識メディアステーションの概要について述べ，制約指向論理型言語τ，及びそれに基づく対話的問題解決環境について述べる．

2．知識メディアステーションの概要

2．1．知識メディアステーションの設計思想

われわれは，次のような設計思想に基づき知識メディアステーションの開発を行った．

（1）多様な表現形式をもつ知識のすべてを計算機が解釈できる必要はないが，解釈可能な形で表現される知識については宣言的な形式で表現する．

これに基づき，知識表現，問題解決用言語として制約指向論理型言語を採用した．論理型言語にはデータベース機能を自然に含むという利点があり，制約指向には実行順序を意識せずに問題記述ができるという利点がある．

（2）推論ルールなどのプログラムから入出力という概念をなくす．

文書，表，図，イメージなどに対応する専用の標準入出力シートを用意し，プログラムに対する入出力と各種シートの内容とを随時結合することにより，好みの表示形態がプログラムと独立に得られるようにした．これらのシートはプログラムによる処理結果の出力表示だけでなく，プログラムへの入力も行える双方向インタフェースを備えている．

（3）知識の整理，体系化の支援機能として，ハイパーメディアを提供する．[11]

これにより，知識ベースのハイパーメディア化が可能となった．マニュアルや教科書などの知識体系を知識ベースに格納でき，さらに未整理のメモや書類などの多様な知識を整理，体系化し，知識ベースに格納できる．

（4）推論とハイパーメディアの融合により，多様な知識を問題解決に活用する．

推論の中でハイパーメディア化された知識を参照することが可能であり，推論の途中あるいは結果の提示時においてハイパーメディア化されたマニュアルや教科書中の適切な記述を引用して表示することができる．

（5）柔軟な対話的問題解決環境を提供する．

Incremental Query, Interactive Query Revisionを拡張した問合せモデルに基づき対話的に問題解決のための条件を与えることができ，問合せ，知識の追加，修正に対し推論結果の変化を即座に分析できる．また，必要に応じて様々な推論機構の内部情報を提供したり，対話的にハイパーメディアを用いて関連知識を参照，活用できる．

（6）デスクトップ思想に基づき作業環境を管理する．

ある作業において利用される知識，問合せな

どを作業環境として保存，再現でき，デスクトップ思想に基づき複数の作業環境を管理する．従って，複数の作業を同時に行ったり，作業の中断，再開を容易に行える．

2．2．知識メディアステーションの構成

知識メディアステーションは制約指向論理型言語による推論を行う推論機構，ユーザとのインタフェースとなる知識プロセッサ，知識を整理，体系化するハイパーメディアの3つの部分から構成される．

知識メディアステーションでは，制約指向論理型言語による知識の他に，文書，表，図形，イメージ，グラフ，メニューの各種知識を扱うことができる．

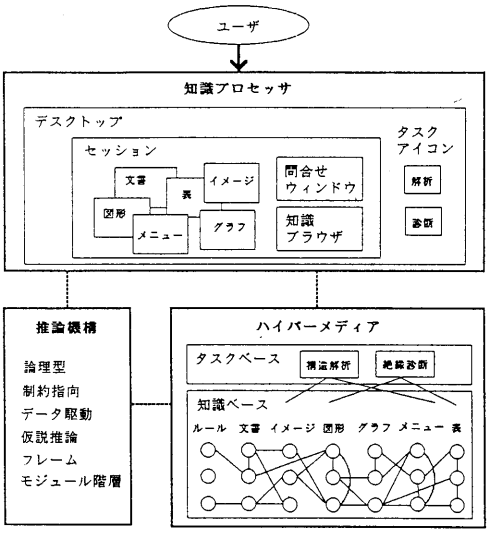

図1．知識メディアステーションの構成

（1）推論機構

制約指向論理型言語τにより記述された知識に基づき推論を行う．

（2）知識プロセッサ

知識プロセッサはハイパーメディアや推論機構の機能を活用し，柔軟な問題解決作業環境を提供するユーザインタフェースであり，セッション，タスク，デスクトップに大別される．

・セッションは問題解決を行う実際の作業場で，各種知識に対する標準入出力シートにより知識の作成，編集を行ったり，それらの知識内容を推論機構に渡し推論を行ったり，あるいは推論結果を受け取り各知識に反映したりする．また，ハイパーメディア機能を利用するためのインタフェースでもあり，簡単なマウス操作により，知識を関係付け，整理，体系化したり，関連知識を検索，参照できる．

・タスクはセッションの任意の時点での状態を保存したものであり，いわゆるアプリケーションに相当するものである．タスクを起動することにより，セッションが開かれ登録時の作業環境を再現でき，アプリケーションの実行を行うことができる．従って，アプリケーションとしてのタスクを起動し，知識や知識間の関係付けの編集を行い，タスクとして登録することにより，ユーザは簡単にアプリケーションをカスタマイズすることができる．

・デスクトップは複数のセッションやタスクを管理するものである．複数のセッションを並行して実行でき，また1つのタスクからは複数の独立したセッションを展開することができ，それらはもとのタスクに影響を及ぼさない．

（3）ハイパーメディア

従来のファイルシステムは情報を階層的な木構造の関係でしか保存できなかった．ハイパーメディアは，文書，表，図，イメージなど多形態の知識を対象とし，ネットワーク構造で保存できる．従って，知識をリンクにより関係付けし，整理，体系化して保存し，効率的に検索，活用できる．

3．制約指向論理型言語τ

知識メディアステーションにおける知識表現，問題解決用言語である制約指向論理型言語τには次のような特徴がある．

①属性リストによるフレーム型知識表現

②制約条件による自然な問題記述

線形方程式，線形不等式，領域制約，等価，不等価，起動条件付き述語

③オブジェクト指向に基づくモジュール階層

④柔軟な推論機構

Prologと同様な後向き推論機構

制約条件に関する前向き推論機構

データ駆動推論機構

⑤付加知識機構による動的な知識の追加

3．1．フレーム表現

フレーム表現は知識の有効なクラスタリングを可能にする．τではＣＩＬ[12]と同様に属性リストを用いてフレーム表現を行う．属性リストは属性名とそれに対応する属性値の対の集合であり，記述対象を多角的にかつまとまった形で表現できる．

以下，属性リストの特徴について述べる．

①属性リストは様々な属性を持ったある一つの対象を表す．

例えば，年齢が３０才の男性は次のような属性リストで表される．

Ｘ＝｛@，年齢！30，性別！男｝

ここで，年齢，性別は属性名，30，男性は属性値である．

②属性リストは常に開かれた記述である．

属性リストによる記述は対象の持つ属性を限定するものではなく，動的に属性を追加することができる．

例えば，上記の男性がａ社に就職した場合，

Ｘ！勤務先　＝　ａ社

とすることで，勤務先に関する属性を追加できる．このユニフィケーションの実行後，属性リストＸの内容は，

｛@，年齢！30，性別！男，勤務先！ａ社｝

となる．

③属性値は必ずしも具体的な値を持つ必要はなく，属性値として変数を与えることもできる．

例えば，上記の男性の勤務先での所属が未定の場合には，

Ｘ！所属　＝　Ａ

としておき，所属が決定したとき変数Ａを具体化すればよい．

④属性リストのアイデンティティはその変数名にあり，属性記述の集合にあるわけではない．

従って，次の２つの属性リストは現時点で同じ属性記述を持つが，同一のものを指しているわけではない．　Ｘ＝｛@，年齢！30｝

Ｙ＝｛@，年齢！30｝

これらが必ずしも同一のものでないことは，この後，

Ｘ！性別＝男，Ｙ！性別＝女

とできることでも明かである．

⑤２つの属性リストを等価にする．

異なったアイデンティティをもつ２つの属性リストを等価にする方法はユニフィケーションである．２つの属性リストのユニフィケーションでは，双方の属性名，属性値対の和集合をとることにより属性記述がまとめられ，それが両者の新しい属性記述となる．このとき，両者が同一属性名の属性をもてば，それらの属性値のユニフィケーションが行われ，これが失敗すれば属性リストのユニフィケーションも失敗する．

３．２．制約条件

制約指向プログラミングの基本の一つは，その実行が通常の実行順序とは切り離されて管理される制約条件（Constrains）の導入である．制約条件の実行のタイミングはそれが表す条件の判定が可能になったときである．データが不十分なために判定ができないときは，その評価はデータが揃うまで保留（サスペンド）される．従って，制約指向プログラミングでは，実行順序を特に意識せずに問題の記述を行えるという利点がある．

制約指向論理型言語τでは，非等価，不等式，否定，方程式，有限領域制約の各組み込み述語が制約条件として処理される．また，起動条件を付加することにより，任意のユーザ定義述語を制約条件とすることができる．以下，これらの制約条件について述べる．

①非等価（＝／＝）

Ｘ＝／＝Ｙは，ＸとＹが等しくないという制約を与える．Ｘ，Ｙに変数が含まれていて「等しくない」ということが判定できないときにサスペンドする．

②不等式（＜，＞，＝＜，＞＝）

数及び変数と演算子　＋，－，＊，／，ｍｏｄ，ｄｉｖで構成される数式Ｘ，Ｙを不等号で結び大小関係の制約条件を与える．Ｘ，Ｙに変数が含まれるときサスペンドする．変数を一つだけ含む不等式は，その変数に関する区間制約として扱われる．

③否定（ｎｏｔ）

ｎｏｔ（Ｇｓ）は，ゴール列Ｇｓの計算が失敗したときに成功し，成功したときに失敗する．Ｇｓに変数が含まれるときサスペンドする．

④方程式（＝）

２つの数式を等号（＝）で結び方程式制約を与える．Ｘ＝Ｙ（Ｘ，Ｙは数または変数）の形に簡約化できないときにサスペンドする．

⑤有限領域（ｄｏｍａｉｎ）

有限領域制約は次の形式で与える．

ｄｏｍａｉｎ（Ｘ，＜領域リスト＞）

＜領域リスト＞はアトム，数，ストリングからなるリストである．有限領域制約は，Ｘの値を領域リスト中の要素のいずれかに限定するものである．

有限領域制約は，Ｘが既にある値に具体化されているときには，その値が領域リストの含まれていれば成功し，それ以外は失敗する．Ｘが未定義変数のときは，領域リストが１要素であればＸをその要素とユニファイし，領域リストが２つ以上の要素を含むときサスペンドする．

この有限領域制約に対応して，領域中から一つ要素を取り出して変数とユニファイする組み込み述語として　ｉｎｄｏｍａｉｎ（Ｘ）　を用意した．これは，バックトラックが起きた場合には，別の要素を取り出し変数とのユニファイを行う．取り出すべき

要素がなければ失敗する．

⑥起動条件付き述語

ユーザ定義述語には，その述語の引数がどの様な条件を満たせば評価してよいかを規定する起動条件を付加することができる．起動条件としては，各引数について，

・グラウンドである（＋＋）
・少なくともファンクタが決まっている（＋）
・未定義でもよい（＿）

のいずれかが指定できる．この起動条件を満たさない述語呼び出しはサスペンドする．

例えば，次のような形で起動条件を与える．

```
：－　mode　p（＋＋，＋，＿）；
```

この条件により，

p（a（A），B，C）　はサスペンドし，

p（a（a），b（B），C）　は直ちに評価される．（A，B，Cは未定義変数）

3．3．制約条件に関する前向き推論

実行中にたまった複数のサスペンドした制約条件に関しては前向き推論が行われ，解や矛盾やより強い制約条件が導き出される．このような制約条件は，active constraints [7] と呼ばれる．このような処理は有限領域制約関連，区間制約関連，方程式制約関連の３つに分類される．

①有限領域制約関連

一つの変数に対して，

・複数の有限領域制約がある場合：
　積集合が新たな有限領域となる．
・有限領域制約と非等価がある場合：
　非等価にされた値を除いた領域を新たな有限領域とする．
・有限領域制約と区間制約がある場合：
　領域のうち区間制約の範囲内にある部分集合を新たな有限領域とする．

以上の処理で，新たな有限領域が

・空になるときは，失敗する．
・１要素になるときは，変数をその要素に具体化する．
・それら以外のときは，導出するために用いられた制約を新たな有限領域制約で置き換える．

②区間制約関連

一つの変数に対して，複数の区間制約がある場合には，それらの区間の共通部分を求める．

共通部分が

・空になるときは，失敗する．
・１点になるときは，変数をその値に具体化する．
・それ以外のときは，導出するために用いられた制約を求められた共通部分を示す新たな区間制約で置き換える．

③方程式制約関連

線形方程式はすべてまとめられ掃き出し法によって解が求められる．非線形方程式は，それが線形になったときに線形方程式系に加えられる．線形方程式が解を持たない（不能となる）とき失敗する．

3．4．モジュール階層

τでは，ＥＳＰなどと同様にオブジェクト指向に基づき知識のモジュール化を行え，その継承機能により知識の階層化を行える．モジュール定義の単位はワールドと呼ばれ，ＥＳＰのクラスに相当するものである．制約指向を導入したのもそうであるが，できる限りプログラムを簡易化するという観点から，τではＥＳＰにおけるようなインスタンスという概念をなくしている．従って，基本的なワールド定義は，継承定義，パブリック述語定義，ローカル述語定義からなる．パブリック述語定義はＥＳＰのクラスメソッド定義に相当するものである．

```
world　＜ワールド名＞　has
  nature　＜継承ワールド名＞　；
  ＜パブリック述語＞
  local
  ＜ローカル述語＞
end.
```

3．5．付加知識機構

一般に，人間が計算機に与える知識は完全なものではなく，推論を行うためには多くの必要な知識が欠けていると考えられる．このような考えに基づき，計算機が人間に問合せを発し，人間から不足している知識を得ながら推論を行うものとしてQuery-The-User[13]がある．付加知識機構とは，このように推論を行うために不足する知識をユーザから動的に得るための機構である．

τでは，ワールド定義に動的にユーザの入力した節定義を追加するという形で知識の付加が行われる．ある述語を付加知識の対象とするためには，その述語に対して次のようにしてopen-knowledge節を定義しておく．

```
＜ヘッド＞　:-　ask_to　user ;
```

open-knowledge節は，述語の選択肢の一つとして扱われる．実行時にopen-knowledge節が選択されると，

そのボディ部の定義をユーザに問い合わせる．得られたボディ部の定義は，通常の節定義と同様に評価される．

3．6．τによる知識表現の特徴

τによる知識表現の特徴をまとめると次のようになる．

（1）Prologと同様なデータベース機能
（2）継承機能を備えた知識の階層的モジュール化機能
（3）属性リストによるフレーム表現
（4）制約指向による宣言的な知識の記述
（5）制約条件についてのデータ駆動推論
（6）制約条件による仮説推論
（7）付加知識機構による動的な知識の追加

論理型プログラミングには知識の宣言的記述を可能にするという特徴がある．しかし，一般には制御構造を意識して手続き的に書かないと，組み込み述語実行時にエラーが発生したり，またプログラムは正しくても効率の悪いものになったりする．既に述べたように，制約指向プログラミングでは述語の実行順序を特に意識する必要はない．また，制約条件として処理される組み込み述語や起動条件付き述語については，条件判定が可能となるまで評価を保留し，データが揃ったときにその述語を実行するというように，データ駆動推論が行われる．このように制約指向プログラミングとは論理型プログラミングの延長にあり，言語の宣言的記述能力と推論効率を同時に高めるものである．

また，推論の各時点における解は，そのときサスペンドしている制約条件がすべて満たされたときに限って真であるという意味で，条件付きの解[14]といえる．制約条件をサスペンドさせて推論を進めることは，仮説推論を行うことに等しいと考えられる．

4．対話的問題解決環境

知識メディアステーションにおいて実際に問題解決作業を行うのは，知識プロセッサのセッションである．セッションでは，制約指向論理型言語τの実行環境として，Incremental Query，Interactive Query Revision の問合せモデルを拡張した対話的問題解決環境を導入した．[5],[6],[7]

また，セッションは，保留されている制約条件や領域制約，区間制約を受けた変数の取りうる値の範囲など，様々な推論機構の内部情報を提供する機能も備えている．

4．1．論理型言語に基づく対話的問合せモデル

一般的に，論理型言語を用いた問題解決は，問題領域を論理式で記述し，解くべき条件を問合せとして与え，その解を推論により求めることである．代表的な論理型言語であるPrologの場合，問合せ，つまり解の満足すべき条件は論理積の関係で結ばれたゴール列の形で与えられる．そして，条件を満足する解はゴール列中の変数の具体値として得られ，それぞれのゴールは変数の共有関係で関連付けられる．

通常のPrologのインタプリタでは，複数の問合せがあった場合，それぞれの問合せは互いに独立したものであり，変数は問合せごとに局所的に扱われ，問合せにまたがったバックトラックは行われない．従って，このようなPrologでの問題解決の方法は，一つのゴール列に問題を解くための条件を完全に記述しなければならない点で，人間の行う試行錯誤的な問題解決には向かない．例えば，図2において，2つの問合せ（1），（2）に共通の変数名である _太郎の親 に対する変数は別々のものとして扱われる．この変数を同一のものとし，（1），（2）を同時に満足するような解を得たい場合には，（3）のような問合せを与えなければならない．

```
?- 親（太郎,_太郎の親）.                          ･･･(1)
?- 兄弟（_太郎の親,_太郎の親の兄弟）.  ･･･(2)
                ↓  (1),(2)を同時に満たす解
?- 親（太郎,_太郎の親）,
   兄弟（_太郎の親,_太郎の親の兄弟）.  ･･･(3)
```

図2．Prologの実行環境

この問題を解決するために，Incremental Query と呼ばれる新しい問合せモデルが提案された．[5] これは，複数の問合せを論理積の関係で結合されたものと考え，変数を複数の問合せ間で大域的に扱い，問合せにまたがったバックトラックを行うというようにProlog の問合せモデルを拡張したものである．従って，Incremental Queryでは問合せは段階的に入力可能で，その解を見ながら次に必要となる問合せを入力していくことになる．

```
???-  A1,A2,A3.    ･･･ (1)
   S1              S1：問合せ(1)の解
???-  B1,B2,B3.    ･･･ (2)
   S2              S2：S1と問合せ(2)の解の合成
```

図3．Incremental Queryの実行環境

図3において，（2）は実行上，

```
???-  A1,A2,A3,B1,B2,B3
```

と解釈される．人間は（１）と（２）の問合せによる解を見ながら，段階的に不足した条件を補う問合せを与えたり，提示された結果を否定するような問合せを与えたりして，対話的により詳細な解や別解を求めることができる．

また，Incremental Query では入力した問合せの部分的な削除を行うことができ，問題解決のための条件を一部変更したら，解がどの様に変化するかを容易に見ることができる．

このようなIncremental Query の考えをさらに発展させ，問合せの編集機能などを強化したものが Interactive Query Revision である．[6] Interactive Query Revisionでは，問合せの追加，削除，置き換えといった問合せの編集を行うことができる．さらに，実行時に動的な問合せの追加，あるいは推論ルール，つまり節の動的な追加，削除を行うことができる．

４．２．対話的問合せ機構

ここでは，まず制約指向論理型言語と対話的問合せモデルとの親和性について述べ，次いで知識メディアステーションで採用した対話的問合せ機構について述べる．

４．２．１．制約指向論理型言語と対話的問合せモデルとの親和性

既に前章でも述べたように，論理型言語においてはゴールの実行順序を考慮しなければならない．従って，前述のIncremental Query などの論理型言語による対話的問合せモデルでも，問題解決の手順を考慮して問合せの記述を行う必要がある．例えば，図４の算術演算では，Ｃを求めるためには（１）の計算を先に行わなければならない．

```
A = B + 1.    ･･･（１）
C = A * A.    ･･･（２）
```

図４．算術計算における順序

このように実行順序を考慮しながら問合せを与えることは，Incremental Query などを用いた柔軟な問題解決を妨げる要因となり得る．

制約指向論理型言語では，制約条件の導入により言語処理系が自動的に条件の保留，実行を行う．従って，制約指向論理型言語に基づく対話的問合せモデルでは，実行順序を特に意識せずに問合せを記述することができる．この意味で，論理型言語に比べ，制約指向論理型言語は Incremental Queryなどの対話的問合せモデルとの親和性が高いと言える．

４．２．２．対話的問合せ機構

知識メディアステーションでは，対話的問合せ機構としてIncremental Query などに次のような拡張，修正を行ったものを導入した．

（１）問合せの編集環境としてスクリーンエディタを採用した．これにより，与えた問合せをスクリーンエディタで見ながら，問合せを一度に複数編集可能となり，効率的な編集を行えるようにした．

このスクリーンエディタは，問合せ実行要求に応じて，編集前後の問合せを比較してどの問合せが修正されたかを判定し，修正箇所をインタプリタへ送信する機能を備えている．

（２）問合せ間のバックトラックと問合せ内のゴール間のバックトラックとを統一的に扱っている．すなわち，言語処理系の持つバックトラック機能をそのまま利用して，問合せ間のバックトラックを実現している．従って，問合せ間のバックトラックを効率的に行える．

Incremental Query では，各問合せについて，問合せとその問合せを実行する時点での変数の具体化の状況を保存したスタックを使って，問合せ間のバックトラックをシミュレートする．

（３）問合せ修正時にどの問合せから実行するかを決定するための情報を保存するテーブル（再実行問合せ番地テーブル），および各問合せを実行するときのスタック上のレベルを保存するテーブル（問合せレベルテーブル）を導入した．

この情報とは，各問合せについて，その問合せが原因でどの問合せまでバックトラックしたかを示すもの，つまりその問合せが原因でバックトラックが起き別の選択肢が取られた問合せのうちの先頭の問合せの番地である．これらのテーブルを用いて，次のようにして修正に伴う問合せの再実行を行っている．

- 修正された問合せ以降の問合せについて再実行問合せ番地テーブルを検索し，それらが原因でバックトラックが起き，別の選択肢が取られた問合せのうちの先頭の問合せの番地を求める．
- 問合せレベルテーブルを検索し，求めた番地からその問合せの実行レベルを取り出す．
- 取り出したレベルまで戻り，その問合せから再実行を行う．

これにより，問合せ修正に伴う効率的な問合せの再実行を行うことができる．

問合せ修正に伴う再実行では，必ずしも修正された問合せから再実行すればよいとは限らない．例えば，図５で（４）の問合せをＺ＝２．と修正

した場合，実際にはX＝2，Y＝1，Z＝2という解を持つが，（4）の問合せから再実行しても既にY＝2であるため失敗する．また，最初から解き直しをすれば，（1），（2）というむだな問合せの実行を行ってしまう．この場合には，（4），（5）が原因でバックトラックした先頭の問合せ（3）から再実行を行うべきである．

```
( X = 1 ; X = 2 ).  ···(1)
  X =/= 1.          ···(2)
( Y = 1 ; Y = 2 ).  ···(3)
  Z = 1.            ···(4) → Z = 2.
  Y =/= Z.          ···(5)
```

図5．問合せ修正に伴う再実行

（4）現在の問題解決状況を表示するモニタを備えることによって，制約指向論理型言語による問題解決の可能性を高めた．

このために，各問合せとそれに含まれる変数を保存したテーブルや変数とその変数が含まれる問合せを保存するテーブルなどを用意した．

このような制約指向論理型言語に基づく対話的問合せ機構により，計算機と人間の能力を互いに補いながら問題の解決策を考えていく対話的問題解決が可能になった．

4．3．対話的問題解決環境

制約指向論理型言語による問題解決では，問題解決の途中において問合せの実行状態を把握できることが望ましい．これは，制約指向論理型言語の実行環境においては，問合せと問合せ結果が同次元のものではなく，また制約指向特有の保留条件などが存在するためである．

そこで，知識メディアステーションでは，セッションにおいて，以下に述べるような環境を提供することにより，問題解決過程の可視化を行っている．（図6）

①問合せウィンドウ

問合せウィンドウは問合せの入力，編集を行うウィンドウであり，前述のようにスクリーンエディタになっている．

②保留条件ウィンドウ

保留条件ウィンドウは前章で述べたような保留中の制約条件（方程式，不等式など）を表示するためのウィンドウである．

③変数ウィンドウ

変数ウィンドウは問合せに用いられている変数名を表示するウィンドウである．

④問合せ参照ウィンドウ

問合せ参照ウィンドウは変数名を指定することにより，その変数の値（具体化の情報）及びその変数を含む全ての問合せを表示する．

⑤変数領域ウィンドウ

変数領域ウィンドウは，領域制約，区間制約，方程式制約などの制約を受けた変数に対し，その制約や変数の取りうる値の範囲を表示する．

⑥様々な形態の知識に応じた標準入出力シート

人間との親和性の高い問題解決を行うために，文書，表，図形，イメージ，グラフ，メニューなどとの入出力を行うための機構を備えている．

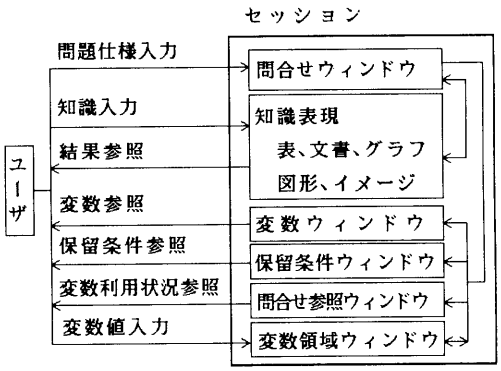

図6．セッションのインタフェース構成

4．4．多形態の知識を対象とする入出力

制約指向論理型言語τは入出力のための特別な機構を持たない．これは，プログラムとその入出力インタフェース（各種知識に応じたシート）との独立性を高める．このことは解の表示方法を考えずに問題解決の方法を記述できると共に，後で適切なシートを適宜用意することによって，分かりやすい形態での解の表示および条件の入力を行えることに結び付く．また，簡単に複数のシートを同一問合せ中で用いることが可能となる．

セッションは文書，表，図形，イメージ，メニューを取り扱うための環境を備えている．推論機構とこれらの知識は，エントリ変数という特別な変数でつながっている．例えば，シート1の表のA列2行目の欄は”S1A2”という変数名で参照できる．また，文書の場合は文書中の任意の領域を参照する変数を定義でき，グラフや図形の場合は任意の作図パラメータに対応する変数を定義できる．このような変数を問合せ中で用いることにより，表や文書などの中から変数の具体値を取ったり，表や文書などへ具体化された変数の値を書き込んだり，その値によって図やグラフを描い

たりすることが自動的に行われる．これはセッションが問合せで用いられている変数名を管理し，表や文書などを参照している変数（エントリ変数）とそうでないものを区別することによって実現されている．

このような変数の値の入出力機構は，表や文書などに値を書き込むことにより推論を起動できること，多彩な形式での解の表示が可能であること，さらに表や文書に表示したものを棒グラフに書き換えるといった表現形式の変更が容易にできるといった利点を持つ．

4．5．問題解決例

図7にブリッジ回路の設計問題の例を示す．この例は，回路に関する制約条件を入力するための表（表シート），各抵抗の抵抗値やそれらのに流れる電流値を表示する回路図（図形シート），各抵抗に流れる電流のグラフ（グラフシート），および問合せウィンドウからなる．この場合，各シートに対する入出力を行うためのエントリ変数を用いて，回路方程式，抵抗の種類，各抵抗を流れる電流の最大値，最小値などの制約条件を問合せとして記述するだけで問題を解くことができる．制約条件入力シートから制約条件が入力されると，それに応じた問合せが問合せウィンドウに送られ，問合せが実行される．実行の結果，制約条件入力を行った表や回路図に抵抗値，電流値が表示され，また各抵抗を流れる電流値がグラフの棒の長さとして表示される．

以下，①から⑥までのそれぞれの問合せについて説明する．

①各シートに対する入出力を行う問合せ

各シートにタイトルや抵抗値，電流値などを表示したり，入力を受け付けるために，エントリ変数の具体化，およびエントリ変数と他の変数またはエントリ変数どうしの関連付けを行う問合せである．

②抵抗値に関する制約条件

各抵抗値の取り得る値を有限領域制約として与えている．

③各抵抗を流れる電流に関する制約条件

各抵抗に流れる電流の最大値，最小値を不等式制約として与えている．

④回路方程式の記述

⑤制約条件の入力

制約条件シートから値が入力されると，その入力

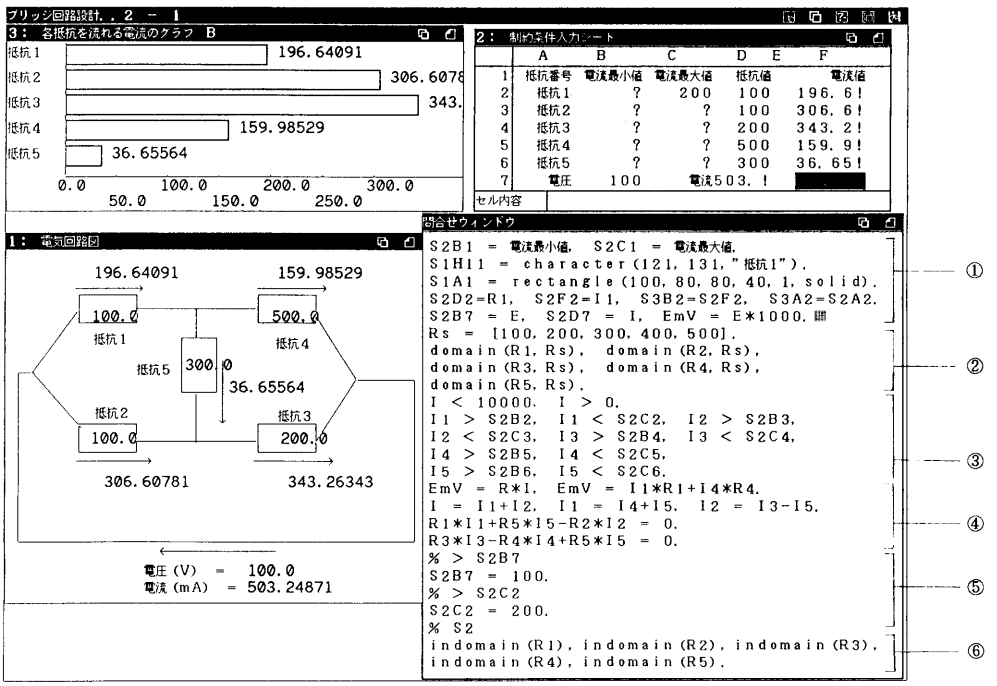

図7．ブリッジ回路の設計

に対応する問合せがこの位置に挿入される.

⑥値が未定義の抵抗に対する抵抗値の具体化

各抵抗値は有限領域制約を受けているが，この問合せを実行する時点で特に抵抗値の入力がなければ，組み込み述語　ｉｎｄｏｍａｉｎ　により領域中から値を一つ取り出し抵抗値を具体化する.

図７は回路の両端の電圧を１００（Ｖ）とし，抵抗１を流れる電流の最大値を２００（ｍＡ）という制約を与えているが，さらに電流に関する制約を付加したり，ある抵抗の値を入力したり，あるいは一度与えた制約を外したりというように，提示された実行結果を見ながら対話的に回路設計を行っていくことができる.

５．おわりに

本論文では，知識メディアステーションの概要，および知識メディアステーションにおける知識表現，問題解決用言語τ，その実行環境である対話的問題解決環境について述べた．制約指向論理型言語τの採用により，問題の宣言的記述能力が向上し，より柔軟な知識表現が可能となった．また，セッションの提供する対話的問題解決環境は，制約指向論理型言語との親和性も高く，通常人間が行うような試行錯誤的問題解決を柔軟に支援する.

今後も，より柔軟な推論機構，より洗練された知識表現方法などについて検討を行い，さらに推論とハイパーメディアとの融合性を高めるなどして，知識メディアの実現を目指していく.

参考文献

[1]竹内他:知識メディアステーション(1)-(6),情報処理学会第37回全国大会,pp.1246-1257(1988)

[2]竹内彰一:"オブジェクト指向の指向するもの",情報処理,Vol.29,No.4(1988)

[3]M.Stefik:"The Next Knowledge Medium", THE AI MAGAZINE,Vol.7,No.1,pp.34-46(1986)

[4]A.Kay,A.Goldberg:"Personal Dynamic Media",COMPUTER,pp.31-41(March,1977)

[5]C.Lessez:"Constraint Logic Programming",Byte,pp.171-176(August,1987)

[6]A.Colmerauer:"Opening the PrologⅢ Universe",Byte,pp.177-182(August,1987)

[7]M.Dincbas,P.van Hentenryck et al:"The Constraint Logic Programming Language CHIP",Proc. of the Int. Conf. on FGCS,pp.693-702(1988)

[8]M.H.van Emden:"Logic as an Interaction Language",Proc. of 5th Conf. Canadian Soc. for Computational Studies in Intelligence(1984)

[9]M.Ohki,A.Takeuchi,K.Furukawa:"A Framework for Interactive Problem Solving based on Interactive Query Revision",Proc. of Logic Programming Conf.,pp.167-176(1986)

[10]淵一博監修,古川康一,溝口文雄共編："インタフェースの科学",共立出版,pp.77-110(1987)

[11]J.Conklin:"Hypertext:An Introduction and Survey",IEEE Computer(September,1987)

[12]K.Mukai:"Unification over Complex Indeterminate in Prolog",Proc. the Logic Programming Conf.,pp.271-278(1985)

[13]M.Sergot:"A Query-The-User for Logic Programming",Proc. of the European Conf. on Integrated Computing Systems,North Holland(1983)

[14]P.Vasey:"Qualified Answers and Their Application to Transformation",Proc. 3rd Int. Conf. on Logic Programming,pp.425-432(1986)