オブジェクト指向知識表現 auk を用いた知的システム構築用シェル AUK†

藤村 茂†† 富田昭司††
飯間 昇†† 鈴木 明††

本稿では，知的システム構築用シェル AUK について報告する．AUK は，効率のよい知的システムの開発を目的とし，知識ベースの構築，および開発されたシステムの利用において，優れた操作性を実現するという観点から設計を行ったシェルである．このシェルの開発にあたって，まず，オブジェクト指向知識表現 auk (autonomic knowledge unit: 自律的知識単位) を提案した．この知識表現では，知識をオブジェクトとして扱い，そのオブジェクトの外部インタフェースを定め，知識のモジュール性，再利用性，統合性を高めた．また，種々の表現に対応すべく，拡張が容易であるように設計した．このような知識表現をベースに，知的システム構築用シェル AUK を，Smalltalk-80 上に開発した．AUK における知識ベースの構築は，この知識表現の特徴を生かし，グラフィカルな知識ベース開発環境内で，インタラクティブに進めることができ，効率的な開発が可能である．

1. はじめに

最近，第二世代のエキスパート・システム構築用のシェルを用いた，エキスパート・システムの開発が，数多く行われている．これらのシェルにおいては，広範な領域の知識を表現するために，多種の知識表現が用意されている．さらに，マルチ・ウィンドウ環境内での，グラフィックス機能を駆使した知識の開発環境が提供されているのも，特徴の一つとして考えられる．しかし，マルチ・パラダイムの知識表現ということから，知識の管理，有効利用が難しい．知識の記述においては，部分的にあいまい性を持たせたかったり，ときには，マトリックスとして表現された表から，値を取り出せる形で知識を表現したかったり，いろいろな形態で知識を表現したい場合が存在する．しかし，第二世代シェルを用いて知識を表現する場合には，そのシェルの枠組みの中でどのように表現するかを考えていかなければならない．この枠組みの制約のために，かなり窮屈な表現をしなければならない．このように考えると，自由な知識の表現を行うためには，問題に適切な知識表現の枠組みが，シェルの機能として備わっている必要がある．この問題を解決する方法としては，まず，ドメイン・シェルがあげられる．また，別の方法としては，シェルに拡張性を持たせ，問題に適切な表現がとれるような枠組みを与えることが考えられる．

一方，マン・マシン・インタフェースについて考えてみると，われわれが，コンピュータを利用して，知的な作業を行っていく場合，本当の意味でのインタラクティブな情報の交換が必要である．インタラクティブとは，コンピュータが単にわれわれに情報を示し，入力を促すだけではなく，われわれから，コンピュータに情報の提供をいつでも催促できなければならない．当然，その催促に対して，状況を判断し，適切な情報をわれわれに示してくれなければならない．そのようなマン・マシン・インタフェースを実現するためには，このようなことに対処できる推論機能はもとより，柔軟な入出力が行えるグラフィックス機能が必要と考えられる．

本稿では，このような観点のもとで，知的なシステムを実現するためのツールとして作成した，知的システム構築用シェル AUK について報告する．そこで，まず，AUK の設計目標について述べ，その目標における機能を実現するために提案する，オブジェクト指向知識表現 auk (autonomic knowledge unit: 自律的知識単位) について説明する．そして，この知識表現を用いた知的システム構築用シェル AUK について機能を概説し，このシェルを用いた適用例を示すことにする．

2. 構想

本章では，まず，第二世代シェルの問題点について示し，次に，知識表現方法，マン・マシン・インタ

† AUK : Shell for Building Intelligent System Based on Object-oriented knowledge Representation auk by SHIGERU FUJIMURA, SHOJI TOMITA, NOBORU IIMA and AKIRA SUZUKI (Section 2, Corporate R & D Department IV, Yokogawa Electric Corporation).

†† 横河電機(株)研究開発四部第二研究室

フェースに関する，われわれの基本的な考え方について述べる．そして，知的システム構築用シェル AUK の設計目標を述べることにする．

2.1 第二世代シェルの問題点

1) 多種の知識表現が混在している．

多種の知識表現を用意することにより，ある程度広範な知識の表現が可能となった．しかし，一方では，知識間の統合性，既存の知識の再利用性，保守性において問題が生じている．

2) 知識表現の拡張が難しい．

エキスパート・システム開発の流れより，現在，いろいろな問題を表現するためには，適切な知識表現を用いる必要があることが認識されつつある．第二世代シェルにおいては，いくつかの知識表現が提供されているが，その適応範囲は限られている．適応できない場合，汎用言語により記述しなければならない．その際，知識表現の同一枠組み内の多少の拡張ならば，汎用言語を用いてモディファイ可能であるが，それ以上の拡張は難しい．

3) グラフィックス・インタフェースとの柔軟な結合が難しい．

第二世代シェルにおいては，ある程度柔軟なグラフィックス・インタフェース用の部品が用意されており，作成されたシステムにおいては，グラフィカルな操作が可能である．しかし，形状や，入出力制御の柔軟なモディファイは容易ではない．

4) 扱いにくい．

第一世代シェルに較べると機能は増強され，広範な知識の表現がとれるようになった．しかし，そのため，一般ユーザにとっての操作性は悪くなった．また，扱いにくい理由としては，1)，2)の問題点に加え，構築の方法が明確にされていない点にもあると考えられる．

2.2 知識表現方法

知識を表現するためには，われわれの思考の過程を素直に表現できることが必要である．その知識の表現方法のモデルとして，次のようなアプローチをとるものとする．

まず，知識を表現する場合，全体の知識を部分的な知識に分割し，それらの知識を，それらに適した方法を用いて表現するものとする．分割された知識は，さらに分割されるかもしれない．これらの知識は，プロダクション・ルール，あるいは，フレームで表現したかったりする．また，あるときは，あいまい性を持たせたかったり，単に表計算のみを行わせたかったりする．これらの知識が互いに協調しあい，問題を解決する．

このようなアプローチをとることにより，全体の知識ベースの開発を，トップダウンな設計により行うことが可能となる．また，知識ベースの開発に際しては，こうしたトップダウンな全体の知識の部分知識への展開とともに，そのような部分知識を，部分的に動作を確認しながら，ボトムアップに構築していく必要もあると考えられる．そのためには，エディット/実行/デバッグという，開発のフェーズを意識せず，部分的に開発を進める方法がとれることも重要であると考えられる[1]．

2.3 マン・マシン・インタフェース

プログラムの開発においては，Smalltalk-80 に代表されるグラフィカルな開発環境の有用性が，現在，認識されつつある[2]．また，知的システムにおいても，グラフィカルなマン・マシン・インタフェースが重要であると考えられ，KEE 3.0 においては，アクティブ・イメージ (active image) と呼ばれる部品が用意されている[3]．そこで，マン・マシン・インタフェースについて，以下では，シェルを用いて知的なシステムを構築する開発者にとっての開発環境としてのものと，その開発されたシステムの利用者にとってのものに対して考えてみることにする．知的システムの開発者にとっては，まず，前記の知識表現方法に合致したグラフィカルなものでなければならない．また，経験の浅い開発者のためには，マウスを用いたスプレッドシート風な知識の入力や自然言語風なルールの記述などが行える開発環境を，シェルの拡張として用意できるような枠組みが必要である．そのためには，インタフェースのモディファイが容易に行える必要があると考えられる．知的システムの利用者にとっては，得たい情報が，必要な時に，簡単に取り出せなければならない．そのためには，ユーザの要求に対していつでも対応できる，操作性のよいグラフィカルなインタフェースを実現する必要がある．そのために，シェルは，柔軟な入出力制御を行える構造が必要である．また，シェルの推論部では，このような入力に対して，推論の局面に応じた推論制御の切り替えを行える枠組みが与えられている必要がある．

2.4 設計目標

2.1 節で述べた第二世代シェルの問題点に対応し，2.2 節，2.3 節の考察をもとに以下の点を知的システ

ム構築用シェル AUK の設計目標とした.

1) 知識のモジュール性がよく，再利用性，統合性に優れていること.
2) 表現の拡張が可能なこと，および，容易なこと.
3) 視覚的でインタラクティブな操作が行えるシステムを作成できること.
4) 知識ベース構築方法を明確にし，その開発環境を提供すること.また，システムの開発者のレベルに応じた知識ベース開発環境が用意できること.

3. オブジェクト指向知識表現 auk

3.1 オブジェクト指向知識表現 auk とは

知的システム構築用シェルを作成するにあたって，その知識表現として，オブジェクト指向知識表現 auk を提案する[4]．本知識表現では，知識の単位を，その知識が持つ静的な情報と，それらの知識を扱う方法を表す動的な情報のかたまりとしてのオブジェクトと考えることにする．そして，これらの知識を表すオブジェクトに対して，統一した外部インタフェース仕様を定め，この仕様のもとで，メッセージ・パスによって，知識間の情報交換が行われるものとする．そして，目的に応じるような，いろいろな推論機能を持つ知識の集まりとして，全体の知識を表現する．このように考えることにより，内部的には，どのような推論が行われようと，外部的には，すべての知識を同じように扱うことができる．このように，外部からは，情報隠蔽がなされており，モジュール性のよい知識ベースの構築を行うことができる．そのため，モジュールごとに，内部仕様の異なる推論系で，全体の推論系を構成することが可能となる．これは，オブジェクト指向の概念を取り入れたためで，目的に応じるように，推論系を拡張することができるのも，それによるものである．なお，本知識表現は，グラフィックス機能を有するオブジェクト指向言語上での実現であることを前提としている．そのため，オブジェクト指向言語のレベルでグラフィックス機能との結合を可能とする．

3.2 基本要素

知識を auk というオブジェクトで表現する．その知識 (auk) は，原子知識 (aukAtom) と，知識集合 (aukSet) から成るものとする．

1) aukAtom

auk の基本単位．スロット名に対するスロット値を持つデータ構造をしており，知識の表現としては，事実や状態の記述などに用いられる．このような静的な表現のほかに，デーモン記述によって動的な表現にも用いられる．外部インタフェースとしては，以下のようなメソッドを用意する．

slot : スロット名
　スロットの読み取り
slot : スロット名 value : スロット値
　スロットの書き込み
　(** : は一引数を伴うメソッド,
　** : ** : は，二引数を伴うメソッド.)

単純には，スロット値そのものの読み取り，書き込み，デーモンの起動を行う．この基本的な aukAtom のことを基本 aukAtom と呼ぶことにする．この基本 aukAtom に対して，スロット値のデータ構造を変更し，そのアクセス・メソッドをモディファイしたり，書き込み制限を加えるような拡張を行っていく．

2) aukSet

要素として auk を持つ auk の集合．すなわち，知識の集合として考え，これも知識の基本要素として考える．外部インタフェースとしては，以下のようなメソッドを用意する．

add : anAuk	要素に anAuk を追加.
remove : anAuk	要素から anAuk を削除.
get : anAuk	anAuk とマッチング可能な auk を取り出す.
redo	get : の後に用いられ，他のマッチング可能である auk を取り出す.

単純には，aukSet は，ある知識 (auk) を加えたり，取り除いたり，あるいは，ある知識を取り出したりすることができる集合として考えられる．この基本的な aukSet のことを，基本 aukSet と呼ぶことにする．そして，この aukSet を，いろいろな推論系に対応するように，この外部インタフェース仕様のもとで拡張し，これらを組み合わせて用いることにより全体の推論を行う（図 1）．

オブジェクト指向言語上での実現にあたっては，基

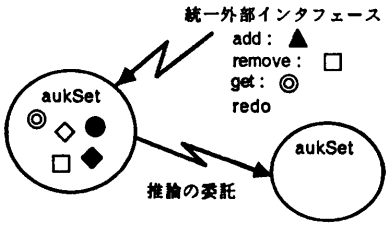

図 1 auk の概念
Fig. 1 Concept of auk.

本 aukAtom，基本 aukSet のクラスを種々の auk のクラスのスーパ・クラスとして，あらかじめ用意しておくことにする．それにより，目的に応じた種々の auk のクラスを，差分プログラミングにより容易に作成することができる．

3.3 基本 aukSet の拡張

基本 aukSet の外部インタフェース仕様上における拡張の方法について以下に示す．

aukSet への add：メソッドによる auk の追加は，知識のモジュールへのある知識の追加と考えられる．その知識の追加により，その知識モジュールの状態が変化すると考えることができる．すなわち，aukSet へある auk を追加することにより，その副作用として，推論が行われ，aukSet の状態が変化すると考えることができる．remove：についても，同様のことがいえる．例えば，aukSet の推論方法がルールで規定されている場合には，**図 2** a)のように実行される．

一方，get：，redo メソッドによる auk の取り出しは，知識のモジュールからの知識の取り出しとして考えられる．その知識の取り出しは，直接そういう情報が取り出せる場合もあるが，推論によって行われなければならない場合もある．そのようなことに対応するために，get：，redo の処理を拡張することが考えられる（図 2 b)）．

このように，統一した外部インタフェースのもとで内部仕様を拡張して種々の推論系に対応できる aukSet を作成するものとする．

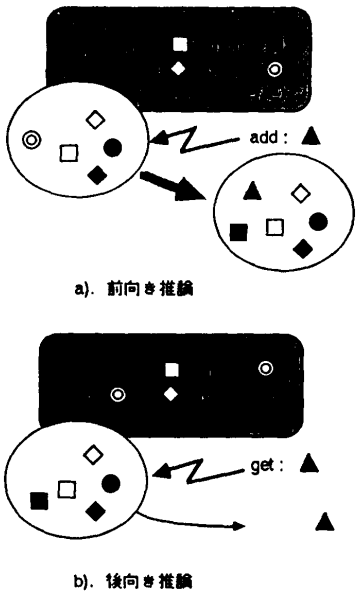

図 2 aukSet の拡張
Fig. 2 Extension of aukSet.

3.4 他のシェルにおけるオブジェクト指向導入方法との比較

第二世代シェルにおいては，オブジェクト指向の概念を導入しモジュール性の向上を図っているものが多い．例えば KEE 3.0 においては，知識の単位であるユニットに宣言的な属性のみならず，メソッドと呼ばれる手続き的な属性を定義できるようにしている．この枠組みの中にフレームやルールを統一的に組み込んでいる．他のシェルにおいても，このように，フレームの機能を拡張しオブジェクト指向化したものが多い．

一方，本知識表現におけるオブジェクト指向の導入の方法はこれらとは異なる．本知識表現では，種々の推論機能によって表現される知識を個々の auk として考えている．そして，これらの auk は統一外部インタフェース仕様で表現されているので，モジュール性，統合性よく多種の知識を表現できる．さらに，種々の推論機能の実現は，オブジェクト指向言語であらかじめ用意された既存の auk を差分プログラミングによって拡張することで容易に行われる．従来のシェルでは，このような観点からオブジェクト指向の概念を導入していないため，多種の推論機能を統一的に結び付けることは難しく，推論方式の拡張も容易ではない．

4. 知的システム構築用シェル AUK

本知的システム構築用シェル AUK は，オブジェクト指向言語 Smalltalk-80[5] を用いて開発した．シェルの構成を**図 3** に示す．基本知識表現は，知識表現 auk の基本要素（基本 aukAtom，基本 aukSet）と，これらの拡張として実現した，ルールの前向き推論 aukSet，後向き推論 aukSet，フレーム表現用の aukSet を含む．さらに，これらを用いて知識ベースを構築するための開発環境を提供している．これらの部分が，本知的システム構築用シェルの基本構成である．ユーザは，この基本構成のもとで，必要に応じて知識表現，開発環境の拡張を行い，知的システムを開発していくものとする．以下では，まず，基本構成の

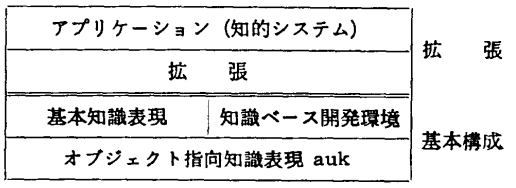

図 3 AUK の構成
Fig. 3 System configuration of AUK.

```
Object
  Auk
    AukSet
      RuleAukSet
        ForwardAukSet
        BackwardAukSet
      FrameAukSet
  AukBrowser
    AukSetBrowser
      RuleAukSetBrowser
      RuleSetBrowser
    AukAtomBrowser
```

図 4 Smalltalk-80 におけるクラス階層
Fig. 4 Class hierarchy on Smalltalk-80.

Smalltalk-80 上でのインプリメント方法を簡単に示す．そして，基本知識表現を用いた推論方法について，および，知識ベース開発環境について説明する．さらに，本シェルにおける知識ベースの構築方法について説明し，知的システム構築用シェル AUK の概観を示すことにする．

4.1 Smalltalk-80 上でのクラス

図 4 に，作成したクラスの階層を示す．

1) auk の基本要素のクラス

● Auk クラス

基本 aukAtom の機能を提供し，すべての aukAtom，aukSet のスーパ・クラスとなりうるクラス．スロット名に対するスロット値，デーモン手続きなどの情報を保持している．インスタンス・メソッドとしては，スロット値の設定，スロット値の参照，デーモンの登録，auk どうしのマッチング用のメソッドを用意している．

● AukSet クラス

基本 aukSet の機能を提供し，すべての aukSet のスーパ・クラスとなりうるクラス．aukSet の基本外部インタフェースである，単純な要素の追加/削除/取り出しのメソッドを用意している．

2) ルールを扱う aukSet のクラス

● RuleAukSet クラス

ルールを扱うすべての aukSet のスーパ・クラスとなるクラス．インスタンス変数として，この aukSet で扱うルール群を保持している．

● ForwardAukSet クラス

ルールの前向き推論用のクラスで，aukSet への aukAtom の追加，削除により推論が起動されるように，add : remove : のメソッドを差分プログラミングにより，モディファイしている．ルールは，Rete ネットワーク[6]に展開され，推論は，このネットワークを用いて行われる．競合解消規則は，容易に変更可能である．

● BackwardAukSet クラス

ルールの後向き推論用のクラスで，aukSet からの知識の取り出しの際，推論が起動されるように，get : redo のメソッドをモディファイしている．

3) フレーム用クラス

● FrameAukSet クラス

aukSet からの知識の取り出しの際，フレーム間の関係をたどるように，get : redo のメソッドをモディファイしている．

4) 知識ベース開発環境用クラス

auk を，必要に応じて表示/操作できるように，MVC (Smalltalk-80 におけるオブジェクト表示の概念)[1] にもとづいて作成したクラス．

● AukBrowser クラス

すべての auk のブラウザのスーパ・クラス

● AukSetBrowser クラス

aukSet のブラウザのクラス

● RuleSetBrowser クラス

ルール群のブラウザのクラス

● RuleAukSetBrowser クラス

ルールを扱う aukSet のブラウザのクラス

● AukAtomBrowser クラス

aukAtom のブラウザのクラス

4.2 基本知識表現

基本知識表現には，auk の基本要素のほかに，ルールの前向き推論 aukSet，後向き推論 aukSet，フレーム表現用 aukSet を含む．これらは，aukSet の外部インタフェース仕様上で拡張されたもので，add : remove : get : redo によるメッセージ・パスにより推論が起動される．

以下では，ルールの前向き推論 aukSet，および，後向き推論 aukSet で用いられるルールの簡単な例を示し，さらに，各種の推論を行う aukSet における推論起動方法について示す．

ルールのシンタックスは，前向き，後向きで共通である．例を以下に示す．

```
moveMv                                  (1)
  |mode|                                (2)
if                                      (3)
  (name : #flowIsBig)                   (4)
  (name : #f101 mode : mode)            (5)
  [mode~=#man]                          (6)
```

```
then                                                    (7)
  (name : # emergency aspect : # lowAlarm)
                                                        (8)
after                                                   (9)
  [Transcript show :                                    (10)
    'f 101 の流量が高いようです.
    f 101 のモードを man にして MV 値を下げて
    下さい. '; cr]
```

1行目では，ルール名を表している．このルール名は，ひとつの aukSet において唯一に定義される．2行目は，変数の宣言を行っている（リテラル，シンボル，ストリングのシンタックスは，Smalltalk-80 に準ずる）．ここで宣言された変数は，ルール内で有効である．また，この変数宣言は，後述する RuleSet-Browser によって自動的に行われる．3行目は，前提部を表すキーワードである．4, 5行目は，マッチングすべき auk の属性を表している．すなわち，4行目は，name というスロット名のスロット値が # flow-IsBig というシンボルである auk とマッチングすることを，5行目は，name というスロット名に対し，# f 101 というシンボルにマッチングする auk で，mode というスロット名に対応するスロット値を，mode という変数に代入することを表している．6行目は評価式で，[ ] で囲まれている部分を評価し，false 以外の場合はこのルールが適用されることを表す．[ ] 内では，Smalltalk-80 の式が評価される．7行目は，結論部を表すキーワードである．8行目は，結論部の aukAtom を表す．9行目は，実行部のキーワードで，この場合 after であり，結論部の評価の後に 10 行目の評価式が実行されることを表している．このキーワードが before の場合は，結論部の評価の前に実行される．

次に，基本知識表現の推論起動方法について，説明する．

1) ルール前向き推論 aukSet

ルール前向き推論 aukSet においては，ある auk を引数として add : のメッセージ・パスで，まずその auk が aukSet の要素として追加される．そして，その追加に伴って，その aukSet に登録されたルール群の中で前提部分とのマッチングが行われる．あるルールの前提部分のすべてのマッチングが成立すると，まず，実行部の before が実行される．そして，結論部に示されたスロットの仕様に従って，aukAtom が作成され，その aukAtom を引数として，add:，(結論部の ' (' の前に '—' が付けられている場合は remove:) のメッセージ・パスが行われる．さらに，その後，after の実行部が実行される．

2) ルール後向き推論 aukSet

ルール後向き推論 aukSet においては，ある auk を引数として get:) のメッセージ・パスで，推論が起動され，情報が取り出せる．その引数の auk は，スロット値を指定したい場合には，その値をスロット値として与え，既知でなく知りたい場合には，nil をスロット値として与えておく．そのような auk が渡されると，まず，aukSet の要素とのマッチングが行われる．そして，そのマッチングが失敗した場合には，その aukSet に登録されたルール群の中で結論部分とのマッチングが行われる．あるルールの結論部分のどれか一つの項のマッチングが成立すると，まず，実行部の before が実行される．そして，前提部に示されたスロットの仕様が成立するかどうか評価する．それがすべて成立すれば，その後，after の実行部が実行される．

3) フレーム用 aukSet

フレーム用 aukSet においては，ある auk を引数として get: のメッセージ・パスで推論を起動する．引数の与え方は，2) と同様である．まず，マッチング可能な auk がその aukSet 内にあるかどうかをみる．そして，存在すれば，その auk のスロット値について，与えた auk と比較する．スロット定義がない場合，isA スロットを参照し，is_a 階層をたどる．

4.3 知識ベース開発環境[8)]

他のエキスパート・システム構築用シェルにおいては，知識の表現の違いにより，種々のツールが用意されている．例えば，知識として表現されたルールのエディット，フレーム構造のエディット用のツールが用意されている．また，知識ベースのロード/セーブや，デバッグ，実行環境など，高級なツールになればなるほど複雑さが増していく．

それに対して本シェルでは，一つのウィンドウが一つの auk を表現しており，auk に対する操作はすべて，これらのウィンドウ中でポップアップメニューを開いて行うことができる．そのため，知識の単位である auk を，ウィンドウを介して直接操作でき，エディット/実行/デバッグの開発フェーズを意識しない知識ベースの構築が可能である．

図5に，この知識ベース開発環境の画面を示す．各ウィンドウは，次のような機能を有する．

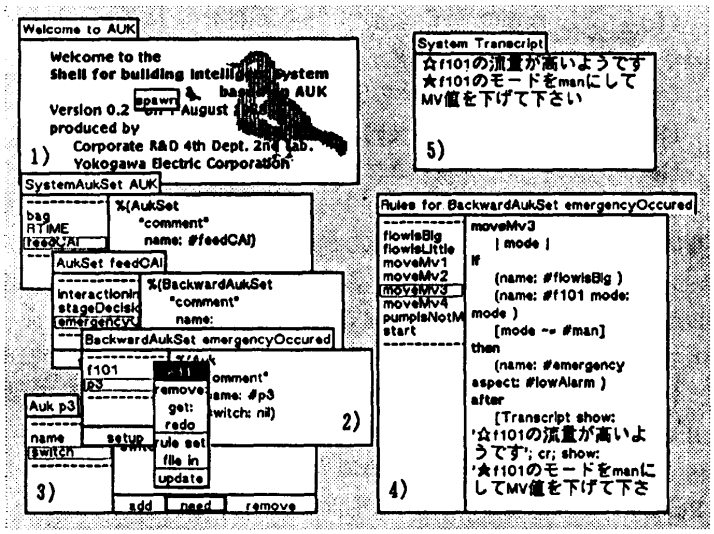

図 5 知識ベース開発環境
Fig. 5 Environment for knowledge base construction.

1) WelcomeView

システムのトップレベル．このウィンドウからポップアップメニューでトップレベルの AukSetBrowser を開くことができる．

2) AukSetBrowser

aukSet のブラウザ．このウィンドウを通じて，aukSet の要素のエディット，部分的な実行によるデバッグ，この aukSet 以下の知識ベースのロード/セーブが行える．

3) AukBrowser

aukAtom のブラウザ．このウィンドウを通じて，デーモンの記述，値の設定が行える．値の設定は，デバッグの際に有効である．

4) RuleSetBrowser

ルールのエディット用のブラウザ．基本操作は，AukSetBrowser とほぼ同様である．

5) TranscriptView

文字出力用のウィンドウ．

以下では，ルールの後向き推論 aukSet を用いた故障診断を例に，これらのウィンドウを用いた知識ベースの構築のイメージを示す．

まず，後向き推論 aukSet を作成する．**図 6** は，本シェル中のすべての知識を含むトップレベルの基本 aukSet (SystemAukSet) を表す AukSetBrowser であり，この aukSet の要素として，後向き推論 aukSet を追加する．新しい要素の追加は，図のように，右のサブウィンドウでエディットし，ポップアップメニューで [accept] を選択することにより行われる．追加されると，その auk の名前が，左のサブウィンドウに表示される．その要素を選択し，ポップアップメニューで，[spawn] を選択する (**図 7**) ことにより，その auk のブラウザが開かれる．AukSetBrowser では，ポップアップメニューによって，aukSet の外部インタフェース操作 [add:] [remove:] [get:] [redo], 要素の消去/コピー [remove it] [copy] などをすることができる．外部インタフェース操作の引数は，[copy] によってコピーされた要素である．さらに，下位の aukSet のブラウザを開く操作 [spawn] や，ファイル入出力 [file in] [file out] などもこのポップアップメニューを用いて行うことができる．次に，この後向き推論 aukSet の要素として，aukAtom をエディットする (**図 8**)．このように付加された auk-

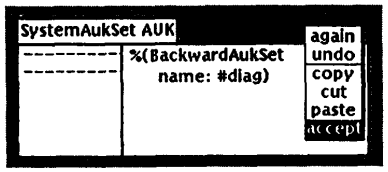

図 6 AukSetBrowser
Fig. 6 AukSetBrowser.

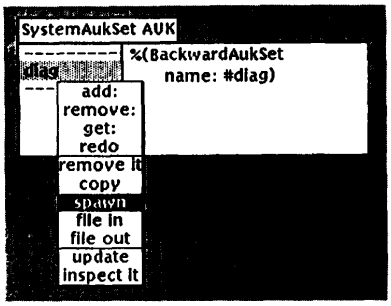

図 7 AukSetBrowser における auk の展開
Fig. 7 Expand an auk on AukSetBrowser.

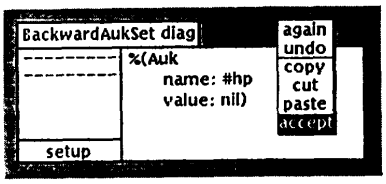

図 8 AukSetBrowser における aukAtom の生成
Fig. 8 Generate an aukAtom on AukSetBrowser.

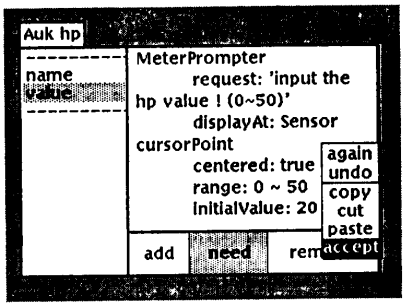

図 9 AukAtomBrowser
Fig. 9 AukAtomBrowser.

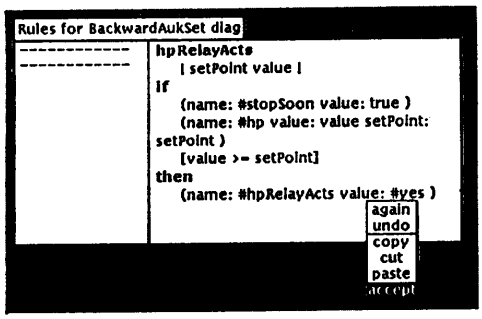

図 10 RuleSetBrowser
Fig. 10 RuleSetBrowser.

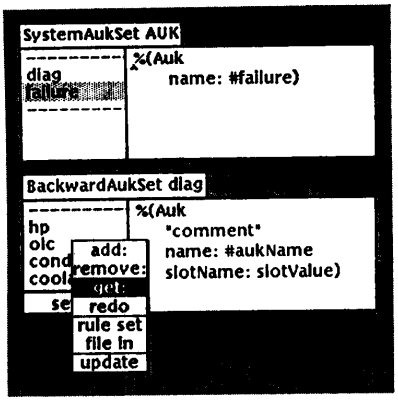

図 11 知識ベース開発環境における部分的実行
Fig. 11 Partial execution on the environment for knowledge base construction.

Atom は，図 9 のような AukAtomBrowser において，スロット名の追加/削除，スロット値の設定，デーモン手続きの記述等が可能である．操作は，ほとんど aukSet のブラウザと同様である．次に，ルールのエディットを行う．そのためには，作成した後向き推論用 aukSet の左のサブウィンドウで，ポップアップメニューの [rule set] を選択する．すると，RuleSetBrowser が開かれ，右のサブウィンドウで，テンプレートに従って，ルールを記述することができる (図 10)．ルールの記述は，このようにルール単位で行うため，部分的にルールを記述し，部分的に実行し動作を確かめることが容易に行える．動作の確認は，図 11 のように，ポップアップメニューを用いて行うことができる．ここでは，failure という auk を get: の引数にすることで推論を起動している．そのためにまず，failure という aukAtom を図のように，この後向き推論 aukSet の外側に作成し，その auk をポップアップメニューで [copy] し，この後向き aukSet のブラウザ内のポップアップメニューで [get :] を実行することにより動作の確認を行っている．

これらのブラウザは，一つの auk に対して，いくつも開くことができ，エディット時の切り貼りも簡単に行うことができる．

4.4 知識ベース構築方法[7]

従来のシェルを用いてシステムを構築する際には，シェルの枠組みを意識し十分な知識の整理を行った上で，知識ベースを記述する方法がとられている．この知識の整理はシェルの環境外で行わなければならないシェルが多い．これは，そのためのツールが用意されていないことのほかに，知識ベースを構築する方法が確立されていないことも起因していると考えられる．そこで，知的システム構築用シェル AUK においては，知識表現 auk の特長を生かして知識ベースの構築方法を規定する．以下ではその概念を示す．

知的システム構築用シェル AUK を用いて知識ベースを構築する場合，“全体の知識を部分的な知識に分割し，それらの知識を，それに適した方法で表現する” というモデルに従って行っていく．その過程においては，次のような二つの方法を，交互に組み合わせることにより，効率的に作業を進めることができる．

1) top-down-planning

ある知識を記述する場合，まず，その知識を部分的な知識に分割する．そして，それらの知識に対して，追加/削除/取り出しの外部インタフェース仕様を定め，機能を規定する．このような部分知識は，その機能を実現するために，さらに，部分知識への展開，あるいは，bottom-up-construction によって，詳細化がなされていくものとする．しかし，この段階では，その方法については考慮しない．これらの知識には，知識利用の制約や，推論状況に応じた推論の切り替えなどの，メタな知識も含まれ，すべての知識が，同じように，部分知識として展開され，外部インタフェースを通じて結び付けられる．

2) bottom-up-construction

機能の規定された知識は，目的に応じた aukSet によって実現される．目的に一致した推論機能を持った aukSet が存在する場合は，その aukSet を用いる．存在しない場合は，既存の aukSet を差分プログラミングによって，機能拡張を行い，目的に応じた aukSet を作成する．

この aukSet が，例えばルールを用いた推論を行う知識の場合は，次に，ルールの記述を行う．この際，基本的なルールをまず記述し，部分的な挙動を確認する．さらに，ルールを詳細化し，規定された機能を実現するように，完成度を増していく．このように，個々の知識内の挙動を実現していくためには，ボトムアップに作業を進めていく．

このような枠組みは，auk が統一外部インタフェースで情報隠蔽されており，差分プログラミングにより推論機能の拡張が容易であるという知識表現の特長によって実現されうる．

現在，このような知識ベース構築方法の支援を，環境として提供していない．開発環境としては，aukSet 単位のルールのステッパ，トレーサなどのデバッグ機能のみを用意している．一貫した支援を提供するためには，例えば，top-down-planning では，全体的なモジュールを設計するためのツールや，メッセージが送られたときインタラクティブに対応できるような未構築の auk を表現するスタッブ，bottom-up-construction では，aukSet 内での知識の整合性のチェックを行う機構などが必要であると考えられる．

次章では，この構築方法に従ってどのように知識を整理し知識ベースを構築していくかを実例を通じて示すことにする．

5. 適 用 例

蒸留プラント立ち上げ時のオペレータ運転訓練用 CAI システムを，Sun-3/260C 上で作成したシェルを用いて評価用として試作した．このシステムは，シミュレータの部分と推論を行う部分から成り，**図 12** のようなインタフェースを介して，オペレータがシミュレータを操作しシステムからのメッセージによって作業を習得していくシステムである．規模としては，

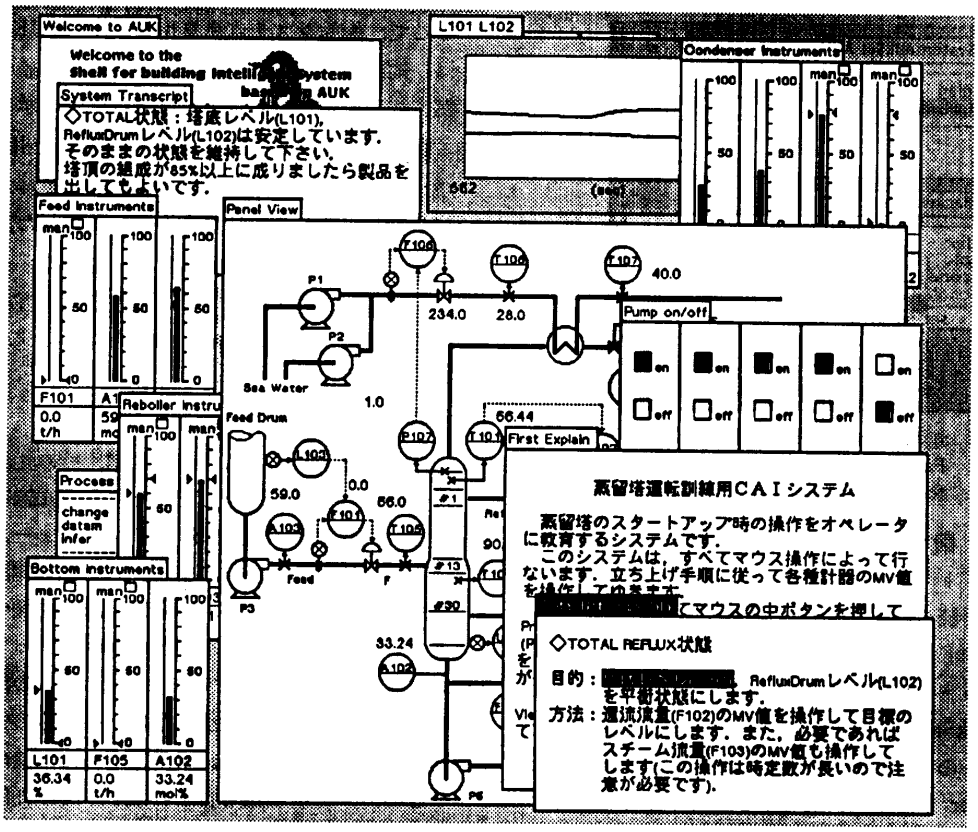

図 12 蒸留塔運転訓練用 CAI システム
Fig. 12 A CAI system for operator's training of distillation tower.

aukSet 数 22 個，全ルール数約 250 個程度のシステムである．この章ではまず，4 章での知識ベース構築方法がどのように適用されたかを示す．そして，いくつかの種類の推論とグラフィカルな操作環境を有機的に結び付けるために用いた推論切り替え制御の方法について示す．

5.1 知識ベース構築過程

まず，システムの開発に先駆けて，提供されるべき機能の整理を行った．

1) 異常時の診断
2) オペレータの誤動作に対する指示
3) オペレータの問い合わせに対する指示
4) 定期的な監視

このような機能を提供するために，知識の分割を検討した．まず，これら別個の機能を実現するためには，緊急度に応じた推論の切り替えが必要であると考えた．そして，プラントの時系列データを保持する auk，プラントの状態を認識する auk，個々の機能を実現するための auk に分割した．とりあえず，これらの auk のインタフェースを規定し，いくつかの auk の詳細化を行っていった．時系列データを保持する auk は，フレーム aukSet を拡張して用い，プラント状態を認識する auk は，オペレータの動作をモデルにし，ルールの前向き推論 aukSet で実現した．これらの auk は，いくつかの auk から呼び出されるものである．そして，推論切り替えを行う auk においては，まず，異常診断用 auk のみに作用するようなルールを用意し，部分的にデバッグを行った．そして，異常診断用 auk について，さらに部分知識への分割を行い，それらの詳細化を行っていった．

このように，top-down-planning と bottom-up-construction を交互に組み合わせ，知識ベースの構築を行うことにより，効率的に作業を進めることができた．

5.2 推論切り替え制御

知識の分割を行っていく過程で，推論の切り替えを行う auk が必要であると考えた．

例えば，異常時の推論を行う場合は，この推論を最優先して開始するべきである．しかし，定期的な監視推論を行っているときにオペレータの問い合わせなどがあった場合，推論が切り替えられるべきである．このように，現在の推論を終了して他の推論を開始したり，中断して他の推論が終了したら再開したりするようなことが自由に表現できる必要がある．このシステムでは，このような推論の切り替えを行うようなメタな知識も図 13 のようなルールで表現している．このルールは，定期的な監視を行わせようとするメッセージが送られてきた場合，前提部で現在推論中のモジュールが存在しないことを確認し，存在していなければ結論部で新しい推論のプロセスを生成しそれを実行するということを意味している．

さらに本システムでは，推論実行中でもシミュレータの操作は行えなければならず，その操作によって推論が起動されるべき場合もある．このようなインタラクティブな操作を可能とするためには，割り込み的な入力を受け付ける機能が必要である．これについては，Smalltalk-80 の複数のプロセスを管理するスケジューラの機能を利用している．

5.3 評　　価

以上のように，推論中でもオペレータの適時の操作

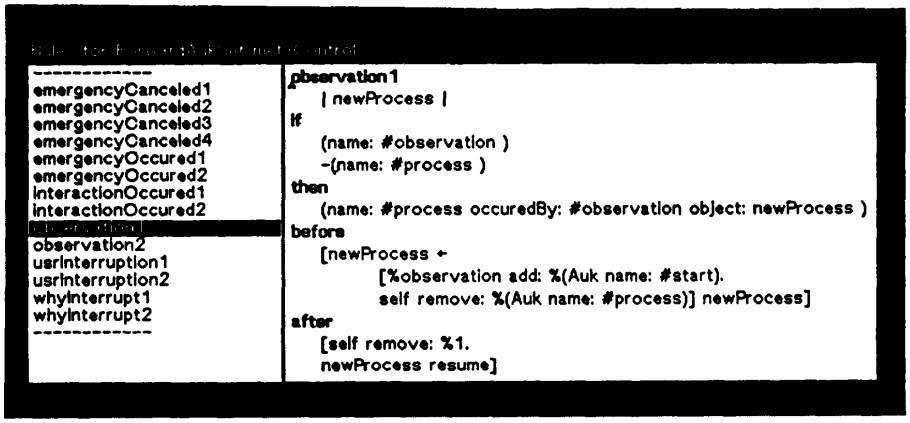

図 13 ルールの記述例
Fig. 13 Example of a rule description.

に対処すべく入出力制御，推論の切り替えが，スムーズに行えるシステムが作成でき，知識もモジュール性よく構築できた．また，このシステム作成にあたって，時系列データを扱うことが可能な auk への拡張も容易に行うことができ，拡張性も確認できた．

6. おわりに

6.1 結　論

以下では，設計目標がどのように実現されたかをまとめる．

1) 知識のモジュール性がよく，再利用性，統合性に優れていること．

知識を，推論系を含み持つオブジェクトで表し，それらの間に統一した外部インタフェースを規定することにより，知識の推論形態や静的な情報を外部から情報隠蔽した．そのため，モジュール性のよい知識の表現が行える．また，これらのすべての部分知識の扱いかたを統一することができ，その意味で単一パラダイムが実現でき，統合性のよい表現がとれるようになった．

2) 表現の拡張が可能なこと，および，容易なこと．

上記のように外部インタフェースが規定されているため，本シェルでは，この仕様のもとで拡張した種々の推論形態を有する表現を自然に取り入れることができる．また，オブジェクト指向言語上での実現を仮定しているので，差分プログラミングにより容易に拡張していくことが可能である．

3) 視覚的でインタラクティブな操作が行えるシステムを作成できること．

5.2 節で示したように，本シェルでは，Smalltalk-80 のスケジューラを用いることで割り込み入力の受付けが行え，グラフィックス環境のもとでインタラクティブな操作が可能である．さらに，推論のプロセスを aukSet で管理し状況に応じて推論を切り替えるような知識をルールで記述することにより，複合的な機能を提供するシステムを構築できる．

4) 知識ベース構築方法を明確にし，その開発環境を提供する．また，システムの開発者のレベルに応じた知識ベース開発環境が用意できること．

現在，知識ベース開発環境としては，基本知識表現の auk 単位のブラウザ，デバッガを用意しているが，知識ベース構築方法を直接支援するような環境はまだ提供していない．しかし，4.4 節で述べたような環境を実現することを今後の課題として考えている．また，システムの開発者のレベルに応じた知識ベース開発環境についても，よりユーザ指向なものが必要であると考えている．これも，オブジェクト指向言語上のインプリメントであるということから容易に拡張されうると考えられる．また，このようなツールは，外部インタフェース仕様が統一された auk 単位で作成されるため，全体的なシェルとしての概念は損なわれない．

6.2 課　題

より多くのアプリケーションを通じて，推論系の拡張を行っていき，このような推論系によって，問題領域に応じたドメイン・シェルの構築を考えている．

また，本シェルは，現在，Smalltalk-80 を用いて記述しているが，より柔軟な知識の記述を可能とするため，われわれが，現在作成中である，手続き型言語/論理型言語/フレーム表現を融合した知識情報処理言語 KPL (Knowledge Processing Language)[9] 上へのインプリメントを考えている．

謝辞 本研究の機会を与えてくださった研究開発四部 福井康裕部長，同第二研究室 久保哲也室長に深謝いたします．

参考文献

1) 上谷：統合化プログラミング環境 Smalltalk-80 と Interlisp-D，丸善 (1987)．
2) Goldberg, A.: *Smalltalk-80: The Interactive Programming Environment*, Addison-Wesley (1984).
3) IntelliCorp KEE: Software Development System User's Manual, KEE Version 3.0 (1986).
4) 藤村ほか：オブジェクト指向知識表現を用いた知的システム構築用シェル，第37回情報処理学会全国大会論文集II，pp. 1228-1229 (1988)．
5) Goldberg, A.: *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley (1983).
6) Forgy, C.L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Patten Match Problem, *Artif. Intell.*, 19, pp. 17-37, North-Holland (1982).
7) 藤村ほか：知的システム構築用シェル AUK —知識構築方法—，第38回情報処理学会全国大会論文集II，pp. 889-890 (1989)．
8) 飯間ほか：知的システム構築用シェル AUK —知識構築環境—，第38回情報処理学会全国大会論文集II，pp. 891-892 (1989)．
9) 鈴木ほか：オブジェクト指向知識情報処理言語の試作と評価，*WOOC '86* (1986)．
10) OPS 5 User's Manual, Department of Computer Science, Carnegie Mellon University (1979).
11) Melle, W.: A Domain-Independent System that Aids in Constructing Knowledge-Based Consultation Programs, Department of Computer Science, STAN-CS-80-820 (1980).

(平成元年 1 月 30 日受付)
(平成元年 10 月 11 日採録)

藤村 茂（正会員）

1960年生．1983年早稲田大学理工学部電気工学科卒業．1985年同大学大学院理工学研究科電気工学専攻修士課程修了．同年横河北辰電機(株)入社．現在，横河電機(株)研究開発四部第一研究室にてオブジェクト指向知識情報処理系の研究開発に従事．プログラミング環境，ユーザインタフェース，ロジックプログラミングに興味を持つ．日本ソフトウェア科学会会員．

富田 昭司

昭和37年8月8日生．昭和60年東京理科大学理工学部電気工学科卒業．62年同大学大学院理工学研究科電気工学専攻修士課程修了．同年横河電機(株)入社．研究開発四部第一研究室所属．現在，知識情報処理システムに関する研究に従事．高次推論，プロセス制御に興味を持つ．計測自動制御学会，電気学会各会員．

飯間 昇（正会員）

1962年生．1985年豊橋技術科学大学生産システム工学課程卒業．1987年同大学生産システム工学専攻修士課程修了．同年横河電機(株)入社．現在，横河電機(株)研究開発四部第一研究室にて，ウィンドウシステムおよび日本語処理の研究に従事．プログラミング言語，プログラミング環境に興味を持つ．

鈴木 明（正会員）

昭和48年東京大学工学部計数工学科卒業．昭和51年同大学大学院計数工学専攻修士課程修了．同年北辰電機(株)入社．現在，横河電機(株)技術開発部門スーパテックプロジェクト担当係長．ACM，日本ソフトウェア科学会，計測自動制御学会各会員．