人　工　知　能 88－2
（1993．6．23）

汎タスクレベルにおける推論オペレータの導出

中村　祐一　　堀　雅洋　　濱　利行
日本アイ・ビー・エム(株) 東京基礎研究所

本研究は，探索制御知識としての推論手続きを，推論対象に関するモデルに対する一連の操作によって表すことを前提として，ライブラリー化された推論オペレータに基づいて知識ベースシステム構築方法を確立することを目指している．本稿では，推論操作の記述レベルとして，タスクレベル/論理的レベル/汎タスクレベルの３つのものを設定し，このようなレベル分けに基づいて推論オペレータの性質について議論する．この中では，推論オペレータを各レベルにおける推論対象モデルに対する操作として記述し，各レベルのオペレータを相互に関連付けることによってオペレータ導出手順を提案する．

Reformulation of Inference Operators at Task-General Level

Y.Nakamura　　M.Hori　　T.Hama
IBM Research, Tokyo Research Laboratry
e-mail: nakamury@trl.ibm.co.jp

Development of a library of inference operators is an important issue to establish a methodology for building knowledge-based systems. This paper describes reformulation of inference operators, taking account of their description levels. In order to find appropriate description levels, models that are manipulated by operators are analyzed first. At each level, inference operators are then defined with regard to name, input/output, and inference specification. Relations between inference operators at different levels are revealed, and a method of refining inference operators is proposed.

1 はじめに

筆者らは CAKE (Computer-Aided Knowledge Engineering) 環境の構築を目指してタスクに特化した部品合成の枠組について検討を進めてきた [堀 他, 1991]. 特に, 解くべき問題のモデルと解法のモデルを区別し, 基本的な制御の流れを部品として整理してきた [中村 他, 1992]. ここでの部品は具体的な推論操作を伴うことによって実行可能になるものであることから, そのような推論操作を整理することは部品合成の枠組を実現していく上で不可欠である. しかしながら, 推論操作は, 問題解決過程の記述に用いられるものであり, 問題の構造や問題解決の状況に応じて様々なものが存在する.

一方, 現在進められている知識ベースシステム (KBS) 構築方法論の研究においても, 様々な観点から推論操作の整理が進められている [Tijerino *et al.*, 1991; Wielinga *et al.*, 1992]. さらに, 診断型のタスクを対象に推論操作を導出する方法についても議論されている [Clancey, 1992]. このように, 知識集約的な問題解決に用いられる基本的な推論操作を整理することは KBS 構築法を体系化していく上で, 重要な研究課題の1つとなっている.

本研究は, 探索制御知識としての推論手続きを, 推論対象に関するモデルに対する一連の操作によって表すことを前提として, ライブラリー化された推論オペレータに基づいて知識ベースシステム構築方法を確立することを目指している. 本稿では, 推論オペレータの記述レベルについて考察し, その結果得られたレベル分けに基づいて推論オペレータを導出する手順を提案する. まず, 推論オペレータを推論対象のモデルに対する操作と見なす立場から, 推論対象モデルの記述レベルについて考察する (2 節). そこでは, ジョブ割付問題というタスクに特化した推論対象モデルを抽象化することによって, 3つの記述レベル (タスクレベル, 汎タスクレベル, 論理的レベル) を見い出している. 次に, 推論オペレータは推論対象モデルのプリミティブで記述されることから, 各レベルごとにオペレータを記述し, それぞれのオペレータを関連付けることによりオペレータ導出手順を導き出す (3 節). そして, 本手法をジョブ割付問題に適用した例を示し (4 節), 最後に関連研究と比較する (5 節).

2 推論対象モデル

推論対象とは KBS が対象とする知識集約的な問題において中心的な役割を果たす存在物であり [Clancey, 1985], 医療診断における患者, 機械設計における機械の構造, 計画問題におけるスケジュール表等が例として挙げられる. 推論対象は一般にノードとリンクからなるグラフとして表わされるが [Clancey, 1992], ノードやリンクといった記述プリミティブの抽象度や, 記述プリミティブが予め組み合わされるべきマクロ的な構造を仮定しているかどうかによって, 様々なタイプのものが考えられる. 本節では, まず筆者らがこれまで検討してきたジョブ割付問題を例として推論対象モデルを具体的に示す. そして, そのようなモデルをより抽象的なレベルで捉え, 特に知識表現におけるレベル分けとして Brachman によって提示された表現レベル [Brachman, 1979]を踏まえて, 推論対象の記述レベルについて考察する.

2.1 推論対象の記述レベル

ジョブ割付問題 (*job assignment task*) は「種々の制約条件を満たした上で, 評価基準を可能な限り達成するように, 与えられたすべてのジョブを資源と時区間から構成される割付平面上に配置する」もので, 生産計画や人員配置計画, 乗物のスケジューリング等がこれに含まれる. このタスクにおける特徴的な推論対象はスケジュール表であり, ジョブ (Job), 資源 (Resource), 時区間 (Time-range) という3種類の概念, ならびに資源と時区間へのジョブの割当を示す2種類の関数 (resource-assignment, tr-assignment) が主要な記述プリミティブとなる [Hama *et al.*, 1993]. さらに, スケジュール表という対象物の性質を反映して, 時区間相互の関係 (before, meet, etc.), ジョブ間の関係 (job-order) 等も記述プリミティブとして用いられる. なお, 推論対象をグラフとして表現する場合, 概念はノードに対応し, 関数と関係はリンクに対応する*. 関数とは, 入力に対して出力が一意に定まるものであり, 関係に

*本稿では, ノードに対応するプリミティブは大文字で始まる文字列で表し, リンクに対応するプリミティブは小文字列として表す.

ついてはこのような入出力の区別はない．

スケジュール表のように，対象分野の専門家や担当者に近い概念レベルに着目した知識の表現レベルはタスクレベルと呼ばれ，Brachman のレベル分けでは概念的レベル (*conceptual level*) [Brachman, 1979, p.205] に対応するものとして位置付けられる．しかしながら，タスクレベルのプリミティブを用いて表現された推論操作は前提となるタスクの範囲内でのみ有効であり，推論知識の再利用という観点からはより抽象的なレベルでの推論対象モデルを考えることが必要である．

推論対象がグラフとして表されることから，タスクに固有の性質を極度に抽象化することによって単にノードとリンクからなるグラフとして捉えることもできる．Brachman は，このようなレベルを論理的レベル (*logical level*) と呼んでおり，個々の記述プリミティブがどのような意味を持つかを問わないレベルであるとしている [Brachman, 1979, p.204]．このレベルでの推論対象は，例えば，Item, At-value, function, relation といったプリミティブによって記述することができる．Item と At-value はグラフ中ではノードとして表現されるが，前者は概念，後者は属性値を表している．一方，関数 (function) と関係 (relation) は，前述のようにグラフ中ではノードを関連付けるリンクとして表現される．

対象とする問題クラスの性質を反映したタスクレベルとは対称的に，論理的レベルではそのような具体的な性質を全く反映しないものとなっている．したがって，論理的レベルの推論対象を前提として記述された推論操作には，その知識が類似の状況においても適用可能かどうかを判断する上での拠り所となるもの，すなわち再利用の基盤となるべきものが存在しない．以下では，再利用の基盤を明示的にするレベルとして，論理的レベルとタスクレベル (より広い意味では概念的レベル) の中間に位置する記述レベルについて考察する．

Brachman は概念的レベルのプリミティブが直観的な意味を伴っているために，このレベルで記述されたモデルの背後にある構造が各プリミティブに暗黙の前提として含まれてしまうことを指摘している．そして，このような暗黙的な構造を明示的に表現するためのレベルとして認識論的レベル (*epistemological level*) が存在するとしている [Brachman, 1979, p.205]．ジョブ割付問題を考えた場合，資源列と時区間から構成される 2 次元平面はタスクレベルでは (直観的に) 明示的であるが，論理的レベルではその構造が明示的には表現されていない．このような 2 次元平面は，それぞれの次元上の区間を規定する要素 (Target-object1, 2)，空間上に配置される対象物 (Source-object)，特定の Source-object が Target-object1, 2 を占有することを示すための関数リンク (occupies1, 2)，区間要素の始点や終点を表す属性 (begin-point, end-point) 等によって記述することができる [Hama *et al.*, 1993]．ここで，2 次元平面のように，記述プリミティブがどのように組み合わせられるかを規定するものを，ここではマクロ構造と呼ぶ．

2 次元平面のようなマクロ構造は特定のタスクに固有ではなく，複数のタスクに共通するものとなっている．2 次元平面を表すための記述プリミティブに着目すると，これらはタスクに固有の言葉でなく，構造的な特徴を表すより一般的な言葉によって表現される．このような構造的な特徴は，タスクレベルでは暗黙の前提として含まれ，論理的レベルでは明示的には表現されていない．筆者らは，特定のタスクを前提としないプリミティブによって，推論対象の構造的な特徴を明示的に記述するレベルを汎タスクレベル (*task-general level*) と呼んでいる．

本節で述べた 3 つのレベルでの記述プリミティブをまとめたものを表 1 に示す．この表には，プリミティブのレベル間での対応関係も示されているが，このような対応関係については 2.2 で述べる．

2.2 記述レベル間の関係

本節では，各記述レベルの推論対象を比較することによりレベル間の関連について述べる．論理的レベルでは，ノードならびにリンクの間に見い出される特徴的な構造は表現されていない．これに対して，汎タスクレベルでは，推論対象の構造的な特徴を明示的にするようにノードとリンクにラベル付けしてある．論理的レベルの推論対象モデルにこのようなラベル付けをしたものが汎タスクレベルのモデルで

表 1. 各レベルの記述プリミティブ

Logical Primitives	Task-Specific Primitives	Task-General Primitives
Item	Job Resource Time-range	Source-object Target-object1 Target-object2
At-value	At-value	At-value
function	resource-assignment tr-assignment ——— ——— ——— ——— etc.	occupies1 occupies2 length (for Source-object) begin-point (for Target-object2) end-point (for Target-object2) length (for Target-object2) etc.
relation	before meet overlap job-order etc.	———— ———— ———— ———— etc.

あり，このような意味でこれらは構造的に等価である．

一方，タスクレベルでは，推論対象のある部分構造がタスク固有のプリミティブとして定義される場合があるからことから，このレベルの推論対象モデルの構造は，他のレベルのものに比較して簡略化されたものとなっている．図 1 にタスクレベルと汎タスクレベルの推論対象モデルの例を示す．この図はタスクレベルで暗黙的な前提となっている部分が汎タスクレベルでは明示的に表現されることを示している．このような表現上の違いは，タスクレベルのプリミティブが，より細粒度のプリミティブへと分解されることに基づいている．例えば，タスクレベルのプリミティブである before と job-order は，汎タスクレベルでは以下のように定義される．

$$\text{before}(t_1, t_2) \stackrel{\text{def}}{=} \text{end-point}(t_1) < \text{begin-point}(t_2)$$
$$\text{job-order}(j_1, j_2) \stackrel{\text{def}}{=} \text{end-point} \circ \text{occupies2}(j_1) < \text{begin-point} \circ \text{occupies2}(j_2)$$

ここで，before は時区間 t_1 が t_2 よりも前にあることを意味しているが，汎タスクレベルでは，t_1 の終点 (end-point) が t_2 の始点 (begin-point) よりも小さいという形で表される．一方，job-order はジョブ j_1 が j_2 よりも早い時間帯にあることを意味している．job-order は，j_1 が占有する時区間の終点が j_2 が占有する時区間の始点よりも小さいという形で定義されている．ただし，記号 $\circ$ は関数の合成を表す．図 1 (a) では job-order によってジョブ間の関係が直接的に表されているが，(b) では Source-object 間の関係を occupies2, begin-point, end-point, < 等の複数のリンクによって表している．

3 推論オペレータ

3.1 推論操作の記述

推論オペレータは推論対象モデルに対する操作を規定するものである．ここでは，推論操作を特徴付ける名前，その操作の入出力，さらに入出力関係を表す推論スペックといった 3 つの要素に基づいて推論オペレータを定義している．このような定義はオペレータの処理手続きそのものは含んでいないが，推論スペックが入出力関係を規定することから，オペレータの処理内容を宣言的に記述したものと見なすことができる．以下では，各レベルごとに推論オペレータの記述法について説明する．

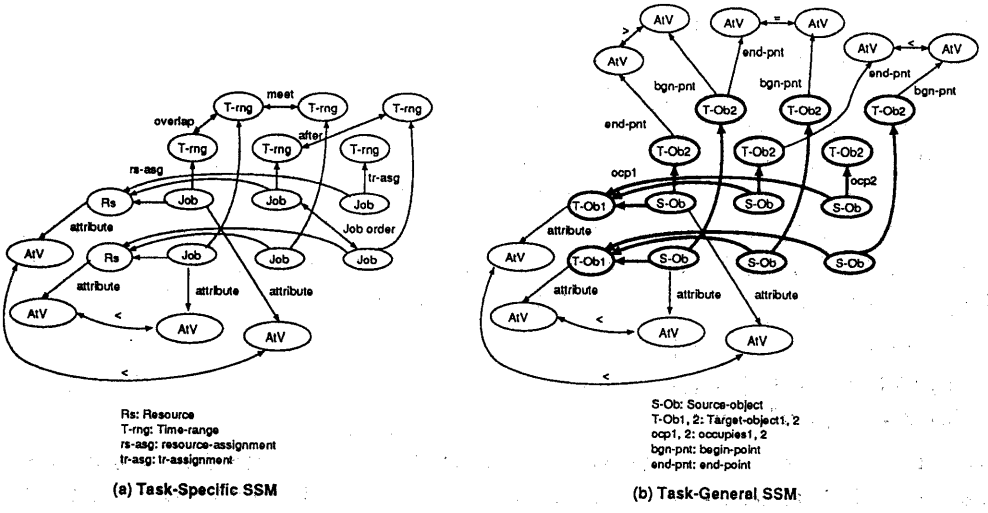

図 1. タスクレベルと汎タスクレベルの推論対象モデル

(1) 論理的レベル

論理的レベルのオペレータは，推論対象モデルの記述プリミティブである Item, At-value, function, relation によって表現されることから，特定の問題に依存しない汎用の推論操作として定義される．そして，このような性質を反映して，このオペレータは入出力と推論スペックにより定義され，特に意味のある名前は与えられない．例えば，要素集合 $\{Item\}$ から 1 つの要素 $Item$ を選択する際に，各要素の属性と与えられた別の要素 $Item_0$ の属性を比較する操作は以下のように定義される．

Name: ————

Input to output: $\{Item\}, Item_0 \longrightarrow Item$

Inference specification: $relation(function1(Item), function2(Item_0))$

推論スペックには $function1$ で指定される $item$ の属性値と，$function2$ で指定される $item_0$ の属性値の間に $relation$ という関係が成り立たなければならないことが記述されている*.

(2) 汎タスクレベル

汎タスクレベルのオペレータはマクロ構造に対する操作として記述されるが，論理的レベルのオペレータと同様にその操作を特徴付ける名前は与えられない．例えば，2 次元平面に対する推論操作は Source-object，Target-object1, 2，occupies1, 2 等を用いて記述され，ある特定の区間要素 Target-objcet2 に対して対象物集合 {Source-object} の中から 1 つを選択する推論操作は以下のように定義される．

Name: ————

Input to output: {Source-object}, Target-object2 $\longrightarrow$ Source-object

Inference specification:

relation(end-point o occupies2(Source-object), begin-point(Target-object2))

推論スペックは occupies2, end-point によって指定される Source-object の属性値と，begin-point によって指定される Target-object2 の属性値の間に $relation$ が成り立たなければならないことを意味し

*推論オペレータの定義では，記述プリミティブが具体的なものを指す場合それをローマン体で表し，任意のものを指す場合イタリック体で表すという記法を用いる．ここでは，*Item, function, relation* 等は任意のものを指していることからイタリック体で表されている．

ている．

(3) タスクレベル

タスクレベルのオペレータは名前と入出力によって定義され，推論スペックは省略される．推論スペックが明記されないことにより，推論対象の部分構造は暗黙的なものになるので，その処理内容が容易に想起できるようにオペレータの名前は非常に注意深く与えなければならない．例えばジョブ割付問題の場合，既に割り付けられた特定のジョブに着目し，そのジョブの次に連続的に割り付けられるべきジョブを選択するといった操作が行なわれる．このような推論操作には select-consecutive-job といった名前が与えられる．すなわち，タスクレベルのオペレータは，その処理内容に関わる性質の多くを注意深く選ばれた名前によって表現していると見なすことができる．select-consecutive-job という名前は，ジョブが同一資源に連続的に割付いている状態を consecutive と呼ぶことに由来しており，タスクに固有の意味を反映したものとなっている．以下に該当する操作のタスクレベルでの記述を示す．

Name: select-consecutive-job
Input to output: {Job}, $\text{Job}_0 \longrightarrow$ Job
Inference specification: ————————

このレベルのオペレータのもう一つの特徴は，他のレベルのオペレータに比べ粒度の大きいものが存在することである．例えば，select-consecutive-job は汎タスクレベルでは個別の推論操作として与えられた2つのオペレータを合成したものとなっている．select-consecutive-job を汎タスクレベルのプリミティブによって記述したものを以下に示す．

Input to output: $\{\text{Source-object}_a\}$, $\text{Source-object}_b \longrightarrow \text{Source-object}_a$
Inference specification:
$\text{occupies1}(\text{Source-object}_b) = \text{occupies1}(\text{Source-object}_a) \wedge$
$\text{end-point} \circ \text{occupies2}(\text{Source-object}_b) = \text{begin-point} \circ \text{occupies2}(\text{Source-object}_a)$

ここでは，Source-object_b は既に割り付けられた特定のジョブに対応し，Source-object_a は選択対象のジョブに対応している．そして，これらのジョブが同一の資源に割り付けられ，かつ前者の終了時刻と後者の開始時刻が一致しなければならないことが記述されている．このように，select-consecutive-job の推論スペックは汎タスクレベルでは独立に与えられる2つの推論スペックの連言として表される．

3.2 推論オペレータの導出

本節では，各レベルの推論オペレータを関連付けることによって，論理的レベルのオペレータから汎タスクレベルを経てタスクレベルのオペレータが導出できることを示す．図2は，各レベルのオペレータがどのように関連付けられるかを前節で述べたオペレータに基づいて示したものである．図では，各レベルのオペレータは名前/入出力/推論スペックのうち該当するものを明記している．推論スペックについて，例えば，L-op1 は，それぞれの要素 (*Item, Item0*) からいくつかの関数リンク (*function*) を介して指定された属性値 (*At-val*) の間に関係リンク (*relation*) を張ったものとして表現されている．

論理的レベルと汎タスクレベルのオペレータはそれぞれ推論ステップの記述パターンが同じで記述プリミティブが異なったものとなっている．例えば，図 2 において L-op1 の記述プリミティブが *Item, Item0, function* であるのに対して，TG-op1 の記述プリミティブは Source-objcet, Target-objcet2, occupies2, end-point となっているが，記述パターンは同一のものである．なお，記述プリミティブの違いを反映して，L-Op は汎用の操作，TG-Op1 は特定のマクロ構造 (2次元平面) に対する操作になっている．

一方，汎タスクレベルとタスクレベルのオペレータの違いは単に記述プリミティブの違いとして捉えることはできない．すなわち，汎タスクレベルのオペレータの推論スペックはタスクレベルではオペレータの名前の中に埋め込まれているという点が本質的な差となっている．例えば，図 2 において TS-op1 は TG-op1 の入出力を Job, Time-Range で置き換え，さらに，TG-op1 の推論スペックをタスクレベルで

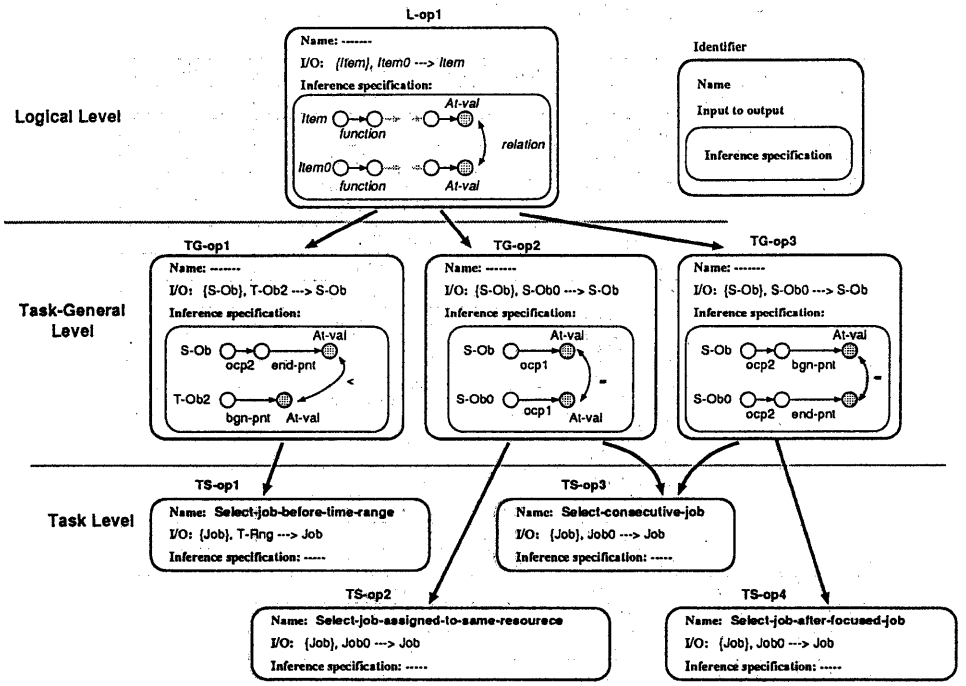

図 2. 推論オペレータの関連付け

「ある時刻より前のジョブを選択する」と解釈することによって，Select-job-before-time-range という名前が与えられている．TS-op1, TS-op2 等はこのようにして得られたオペレータであるが，特に TS-op3 は合成によって得られたものである．

以上のことから，各レベルの推論オペレータは以下のように関連付けられる．すなわち，汎タスクレベルのオペレータは論理的レベルのオペレータの入出力ならびに推論スペックにおける記述プリミティブを汎タスクレベルのプリミティブに置き換えたのみであり，記述パターンは同一である．一方，タスクレベルのオペレータは，汎タスクレベルのオペレータの入出力の記述プリミティブをタスクレベルのもので置き換えると共に，推論スペックをタスクレベルで解釈することによって名前を与えたものとなっている．

さらに，推論オペレータ間の関連を示した図 2 は推論オペレータが導出される過程と見なすこともできる．すなわち，論理的レベルのオペレータを具体化する段階と，汎タスクレベルのオペレータを具体化する段階からなるオペレータ導出手順へとまとめることができる．最初のステップでは，論理的レベルの記述プリミティブを汎タスクレベルのものへと単純に置き換えるのみである．4 節でも述べるように，特定のマクロ構造を前提とすることにより，生成されるオペレータを絞り込みながらの具体化が可能である．一方，次の段階では，記述プリミティブの置換えに加え，対象タスクにおける意味付けを考慮しながらタスクオペレータに名前を与える．さらに，このステップには，汎タスクレベルにおける複数のオペレータをタスクレベルにおいて単一のタスクオペレータへとまとめる操作も含まれる．このような導出手順は，推論オペレータを収集しライブラリー化していくための手段として位置付けられる．

4 ジョブ割付問題を対象としたオペレータ導出

前節で示したように，汎タスクレベルのオペレータは論理的レベルのオペレータの記述プリミティブを置き換えることによって得られる．L-op1 (図 2 参照) の具体化によって導出される汎タスクレベルの

表 2. L-op1 から導出される汎タスクレベルのオペレータ

I/O		Inference Specification		Possibilities
Item	$Item_0$	*function*1	*function*2	
S-Ob	S-Ob	occupies1 ...	occupies ...	——
S-Ob	T-Ob1	occupies1 ...	*function*	——
S-Ob	T-Ob2	occupies1	begin-point	×
		occupies1	end-point	×
		occupies1	length	×
		occupies1	*function*	
		occupies2	begin-point	×
		occupies2	end-point	×
		occupies2	length	×
		occupies2	*function*	
		function	begin-point	
		function	end-point	
		function	length	
		function	*function*	
		function ∘ occupies1	begin-point	
		begin-point ∘ occupies2	end-point	
		end-point ∘ occupies2	begin-point	
T-Ob1	S-Ob	*function*	occupies1 ...	——
T-Ob1	T-Ob1	*function*	*function*	——
T-Ob1	T-Ob2	*function*	begin-point ...	——
T-Ob2	S-Ob	begin-point ...	occupies1 ...	——
T-Ob2	T-Ob1	begin-point ...	*function*	——
T-Ob2	T-Ob2	begin-point ...	begin-point ...	——

オペレータの一覧を表 2 に示す．この表において，最初の 2 列は，論理的レベルの入出力 ($Item$, $Item_0$) がタスクレベルでどのようなプリミティブに対応付けられるかを表している．この場合，$Item$ ならびに $Item_0$ は，それぞれ Source-object, Target-object1, Target-object2 の 3 種類のもので置き換え可能であることから，入出力には全部で 9 種類の具体化が可能である．表の第 3，4 列は，推論スペックの関数が汎タスクレベルにおいてどのように具体化されるかを示している．ここでは，$Item$，$Item_0$ がそれぞれ S-Ob, T-Ob2 に対応付けられる場合について詳細に示している．この場合，$Item$ が Source-object (S-Ob) であることから，*function*1 に関しては，occupies1, occupies2，それ以外のもの (表の中では *function* と記述されている) による置き換えが可能である．さらに，複数の関数を介して属性値を指す場合の例も示されている (begin-point ∘ occupies2 等)．一方，*function*2 の置き換えは，$item_0$ が T-Ob2 (tagert-object2) であることに基づいて行なわれている．なお，ここでの具体化は 2 次元平面に対応するマクロ構造に特化した部分のみを前提として行なわれている．そのため，個々の問題に固有の部分は *function* のように具体化されずに残されている．

オペレータの導出結果を表 2 のようにまとめることは，オペレータ生成における絞り込みを考える上で有益である．例えば，S-Ob に 関数 occupies1 を適用した結果は T-Ob1 であり，T-Ob1 に対して begin-point や end-point 等の属性を考えることは意味がない．S-Ob に対して begin-point ∘ occupies1 等の意味のないリンクを張るような具体化はこの表には示されていない．また，表 2 の右端の列は，要素から関数リンクを介して指示された 2 つの属性値を関連づける関係が存在するかどうかを示している．例えば，S-Ob に occupies1 を適用した結果は T-Ob1 という概念であり，T-Ob2 の始点 (begin-point) は次元上での位置を示す数値となることから，これらを関連づける関係リンクは存在しない．したがって，表の第 3 行目の右端の列には × 印が付けられている．

ここで述べたような絞り込みは，2次元平面のようなマクロ構造を前提としたことによって可能になっている．すなわち，ここでのプリミティブは組み合わせに関してある種の制限が与えられているが，このような制限はマクロ構造を前提とすることにより得られたものである．そして，絞り込みはこのような制限によって可能になっている．

5 考察

5.1 推論オペレータの導出法

Clancey は，集合操作を対象にしたオペレータ導出法を示している [Clancey, 1992]．この手法では，$Item \rightarrow Item$, $\{Item\} \rightarrow Item$, $Item \rightarrow \{Item\}$, $\{Item\} \rightarrow \{Item\}$ といった4種類の推論操作をタスク固有の言葉を用いて具体化している．すなわち，$Item$ を診断型のタスクに特徴的な概念である仮説 (hypothesis) と 兆候 (finding) 等に置き換えることによって，$hypothesis \rightarrow \{finding\}$, $finding \rightarrow \{finding\}$ 等を導出している．これは論理レベルのオペレータをタスクレベルへと直接具体化していると見なすことができる．すなわち，推論対象の部分構造について考慮していないが，分類問題では特に仮説集合を絞り込むといった操作が中心的であり，マクロ構造を明示的に扱う汎タスクレベルを前提とする必要がないと考えられる．しかしながら，スケジューリング等の合成問題では推論対象の構造を前提とした多様な推論操作が存在することが予想されるため，この手法では不十分である．また，Clancey が用いた論理的レベルでのオペレータは入出力のみを規定していることから，導出結果も具体化された入出力のみとなっている．これに対して，筆者らの手法では，推論スペックを導入することによって，オペレータの処理内容も含めた導出手順を提供している．

5.2 推論オペレータの記述レベル

本節では，Brachman のレベル分け [Brachman, 1979] に基づいて関連研究との比較を行なう．ここでは，KBS 構築方法論の研究として代表的な MULTIS [Tijerino *et al.*, 1991] ならびに KADS-II [Wielinga *et al.*, 1992] を対象として，それぞれの試みで扱われる推論操作の記述レベルについて考察する．

MULTIS では，入出力と名前によって定義される推論操作として汎化動詞と呼ばれるものを用意し，その入出力を具体化することによって汎化プロセスを収集している．汎化プロセスは専門家の概念レベルに近い推論操作であり，ここでは概念的レベルに位置付けている (図 3 参照)．一方，汎化動詞については問題構造を強く反映したものからあまり反映していないものまで様々であるが，KADS の知識源と非常に近い概念であることから認識論的レベルに位置付けている．また，汎化プロセスに対応するプログラム部品は Building Block (BB) と呼ばれ，生成検査法やルールインタープリタとして実現されている．MULTIS の試みは汎化プロセスと BB を明示的に区別した点では先駆的な研究である．しかしながら，いずれのレベルにおいても推論スペックに対応するものが与えられていないために，汎化プロセスに暗黙的に仮定されている処理内容を顕在化させながら BB を構築しなければならない．

KADS では，プリミティブな推論操作を集合，構造，属性に対する操作と見なすことにより，知識源 (*Knowledge Source*) として整理している．知識源は汎化動詞と同様に推論スペックを含まない操作であるが，操作対象の構造を反映した名前付けがされているという意味で認識論的レベルであるとしている．推論スペックに対応するものは，$(ML)^2$ [van Harmelen and Balder, 1992] を用いて記述される．$(ML)^2$ は，特定の概念や関係，構造を想定していないので，これによって記述される内容は論理的レベルに位置付けられる．さらに，その記述はプログラムとしての実装も容易であることから，知識源の整理と共に実装レベルでのプログラムライブラリを構築することも可能である．また，概念的レベルの推論操作に相当するものは，知識源が前提としている構造を対象領域のモデルと結びつけることによって (Domain View) 表される．例えば，知識源 Abstract は is-a 階層を前提としていることから，医療診断における病気の階層等に結びつけられる．KADS では知識源の処理内容が認識論的レベルで定義されていないので，is-a 階層のような構造は論理的レベルにおいて個別に定義されている．

CAKE における KADS との最も大きな相違点は，認識論的レベル (汎タスクレベル) での推論操作

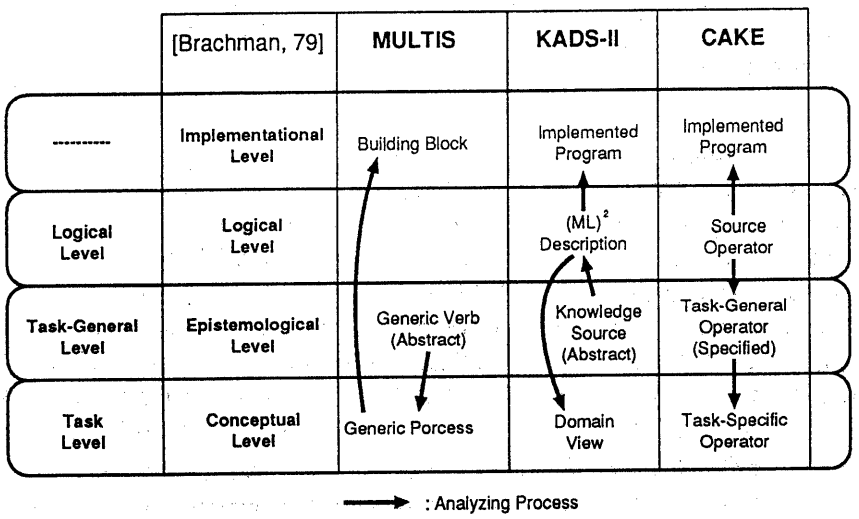

図 3. 他の研究との比較

に処理内容を反映した推論スペックが含まれている点である．このような推論スペックはマクロ構造を前提としたプリミティブによって記述されるものであり，論理的レベルでの記述とは明確に区別される．さらに，概念的レベル (タスクレベル) でのオペレータには処理内容を反映した名前と，入出力パラメータのタイプが与えられる．このように，CAKE では推論操作を名前/入出力/推論スペックによって表現し，各レベルのオペレータを関連付けている．なお，2 次元平面以外のマクロ構造，ならびにマクロ構造を前提とした推論オペレータの再利用性については別途報告する予定である [中村 他, 1993].

参考文献

[堀 他, 1991] 堀, 中村. タスクモデルに基づく問題解決エンジンの合成 – 知識工学支援のための cake 環境を目指して –. 人工知能学会「AI シンポジウム '91」論文集, SIG-FAI-HICG-KBS-9101, pp. 31–40, 1991.

[中村 他, 1992] 中村, 堀. 知識ベースシステム構築のための問題解決部品の同定ータスクの性質を反映した部品空間についてー. 人工知能学会研究会資料, KBS-9104, pp. 31–40, 1992.

[Tijerino *et al.*, 1991] Y. A. Tijerino, T. Kitahashi, and R. Mizoguchi. MULTIS: A knowledge acquisition system based on problem-solving primitives. In *Sixth Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1991. Banff, Canada.

[Wielinga *et al.*, 1992] B. J. Wielinga, T. Schreiber, and J. A. Breuker. KADS: a modelling approach to knowledge engineering. *Knowledge Acquisition*, 4:5–53, 1992.

[Clancey, 1992] W. J. Clancey. Model construction operators. *Artificial Intelligence*, 53:1–115, 1992.

[Clancey, 1985] W. J. Clancey. Heuristic classification. *Artificial Intelligence*, 27:289–350, 1985.

[Brachman, 1979] R. J. Brachman. On the epistemological status of semantic networks. In N.V.Finder, editor, *Associative networks: representation and use of knowledge by computers*, pages 3–50. New York: Academic Press, 1979.

[Hama *et al.*, 1993] T. Hama, M. Hori, and Y. Nakamura. Task-specific language constructs for describing constraints in job assignment problems. Technical Report RT-0084, IBM, 1993.

[Tijerino *et al.*, 1991] Y. A. Tijerino, T. Kitahashi, and R. Mizoguchi. MULTIS: A knowledge acquisition system based on problem-solving primitives. In *Sixth Knowledge Acquisition for Knowledge-Based Systems Workshop*, 1991. Banff, Canada.

[van Harmelen and Balder, 1992] F. van Harmelen and J. Balder. $(ML)^2$: a formal language for KADS models of expertise. *Knowledge Acquisition*, 4:127–161, 1992.

[中村 他, 1993] 中村, 堀, and 濱. 推論オペレータ導出のための問題構造のモデル化 – 汎タスクレベルのマクロ構造 –. 1993 年度人工知能学会論文集, 1993. (発表予定).