09-R 004

大規模知識ベースに関する調査研究

— オントロジー工学に関する調査研究 —

報 告 書

平成１０年３月

JIPDEC

財団法人　日本情報処理開発協会

KEIRIN 00 この事業は、競輪の補助金を受けて実施したものです。

まえがき

近年、情報処理システムは、高度化の１つとして人工知能（AI：Artificial Intelligence）技術などの発展を背景として、知識処理の方向に進んでおります。これとともに大量の知識情報を体系的に蓄積し、共有する大規模知識ベース（VLKB：Very Large-Scale Knowledge Bases）は、より高度な知識処理を実現する情報基盤として期待されております。

しかしながら、大規模知識ベースは、様々な分野・領域（学術、技術、産業など）の知識情報が多様なメディア（文字、言語、画像・映像、音声・音響など）により表現されることから、各分野・領域、各メディアで蓄積・表現された知識は共通の概念体系のもとに関連づけられることが求めらます。この課題に対し、「オントロジー（概念の体系）」が貢献するものとして期待されています。

このような状況のもと、当協会では、オントロジーの概念整理および工学的活用方法を探るため、平成８年度より２年計画で「オントロジー工学に関する調査研究」を進めてまいりました。

今年度は、この調査研究の終了年度として、オントロジーの定義、その使い方と有用性を工学的活用の視点から整理するとともに、オントロジーの研究・開発に必要とされる基礎理論、記述言語・環境、および具体的な活用例について、内外の研究動向を調査いたしました。

実施にあたっては、大学、企業などの研究者、専門家からなる「オントロジー工学調査委員会（委員長　溝口理一郎　大阪大学産業科学研究所教授）」を設置して、審議・検討ならびに委員各位による調査を行ったほか、外部専門家からヒアリングを実施しました。

本報告書は、平成９年度に実施した調査研究の成果を取りまとめたものです。４章から成り、１章では、オントロジー（工学）を必要とする背景、２章では、オントロジーの性質や機能、３章では、オントロジーに関する研究・開発の動向（理論、記述言語・環境、辞書、応用の具体例）、４章では、オントロジー工学の意義と目標など、についてまとめています。

本書が今後の大規模知識ベース関連のプロジェクト計画の立案ひいては高度情報化の発展に寄与することを念願する次第です。

最後に、本調査研究の実施にあたり、ご指導ご協力いただいた委員各位ならびに関係各位に深甚なる謝意を表する次第であります。

平成１０年３月

財団法人　日本情報処理開発協会

委員会名簿

オントロジー工学調査委員会（敬称略、50音順）

委員長：	溝口理一郎	大阪大学 産業科学研究所 知能システム科学研究部門　教授
委　員：	伊藤　公俊	埼玉大学 工学部 機械工学科　助教授
	小山　照夫	文部省 学術情報センター 研究開発部　教授
	鶴丸　弘昭	長崎大学 工学部 電気情報工学科　助教授
	寺野　隆雄	筑波大学 大学院 経営・政策科学研究科 経営システム科学専攻　教授
	外池　俊幸	名古屋大学 言語文化部　助教授
	冨山　哲男	東京大学 大学院 工学系研究科 精密機械工学専攻　助教授
	西田　豊明	奈良先端科学技術大学院大学 情報科学研究科　教授
	堀　雅洋	日本アイ・ビー・エム (株) 東京基礎研究所　主任研究員
	増永　良文	図書館情報大学 図書館情報学部 図書館情報学科　教授
	山口　高平	静岡大学 情報学部 情報科学科　教授
	横井　俊夫	東京工科大学 工学部 情報工学科　教授
	横田　一正	岡山県立大学 情報工学部 情報通信工学科　教授
オブザーバ：	武田　英明	奈良先端科学技術大学院大学 情報科学研究科　助教授
	山本　雅亮	通商産業省 機械情報産業局 電子政策課　課長補佐
	根津　正志	通商産業省 機械情報産業局 電子政策課　国際係長
	市川　隆	(財)日本情報処理開発協会　常務理事
事務局：	片岡　幸一	(財)日本情報処理開発協会 技術企画部 技術課長

以　上

目　次

1．　はじめに

1. はじめに

1.1 背景・状況認識

情報、材料、生命、環境は科学技術の４大課題であると認識されているが、来世紀には「情報」を越えた「知識」がますますその重要性を増すものと考えられている。

高度情報化社会の到来が実感できる動きが米国において、「情報による企業活動の統合」という形で活発化している。電子商取引、電子データ交換、そして企業活動リエンジニアリングである。これらは一見コンピュータネットワークと情報処理技術が基盤となる課題であるように思われるが、実は、その根底には交換される情報の「内容」に関わる処理とそれを支える「オントロジー」という知識処理の根本課題が本質的であることを見過ごしてはならない。

一方、日常の生活においても新しい動きを見ることができる。小型化が極限にまで進化しつつあるコンピュータが家庭に浸透し、発達したコンピュータネットワークと相俟って、家庭に居ながらにして世界中の情報にアクセスし、また情報を世界中へ発信することができるようになりつつある。しかしながら、このような地球規模の大量情報の効果的な流通と利用は、実際には容易に実現できるものではない。２１世紀の高度情報化社会における国民生活を考えたとき、その恩恵を多くの人が容易に享受することを実現しなければならない。そこで重要となる概念が、「大量情報／知識の知的処理」である。情報洪水の中から必要な情報を入手する方法、大部な情報の要約、個人の要求に応じた知識ベースの半自動的構築技法などの確立が望まれる。言うまでもなく、これらの技術の実現には何らかの範囲での意味の共有は不可欠であり、その意味においてもオントロジーが重要となる。そのような共有オントロジーによって根底から支えられた「高度知識処理」は国民の生活に密着したところにおいても大きく貢献するものと期待されている。

今後急速に情報インフラの整備が加速され、それにつれて社会全体が高度情報化社会へと移行する。しかし、その巨大システムとそこを流れる大量情報を有効に運用し、企業と国民がそれぞれの目的に利用することを目指すには、大量の知識を扱える、すなわち内容指向に根ざした高度な知識処理技術が不可欠なのである。クリントン大統領の就任演説にもあるように、米国は今後４年間の最重要政策に「情報化社会」を挙げている。２１世紀は「高度情報化社会」における「知識」の時代、そして、内容指向研究の成果が本質的な貢献をする時代となるものと考えられる[溝口96a] [溝口96b]。

近年、知識処理／AIの分野で大きなパラダイムシフトが起こりつつある。それは次の３つ、(1)処理中心から情報中心へ、(2)コンピュータ中心から人間中心へ、(3)形式中心から内容中心へ、に集約される。

初めの２つの動向の重要さもさることながら、上で述べた２１世紀に到来する「知

識の時代」に最も深く関係する「形式中心から内容中心へ」の傾向は本委員会の中心課題である。これまでのAI研究を概観するとフォーマリズム指向が強く、形式的で、一般性が高い研究が中心的な位置を占めている。言い換えれば、「入れ物」を作ることに関わる基礎研究は精力的に行って来たが、何を入れるかという議論が脱落している。現実の問題を扱うには、対象に関する深い理解とそれに適した取り扱いを工夫する必要があるが、その議論が欠落しているのである。これら３つの流れは、いずれもAI研究を良い意味で役に立つ工学としての側面を育てることに貢献することを目指しているという点で共通している。

もう１つの重要な課題は大規模知識ベースである。これまでの知識ベース構築に関する努力は主にエキスパートシステム開発においてなされてきた。それはコンピュータによる自動問題解決のための知識ベース開発であったと言える。しかし、上述のパラダイムシフトからも分かるように人間とコンピュータとの共生という命題を真摯に受け止めれば、自動問題解決のための大規模知識ベースではなく、人間と協調して問題解決にあたる知的パートナーとしての大規模知識ベースが必要であることが分かる。

このような大規模知識ベースの構築には、コンピュータと人間とによる意味の共有という本質的な課題があることが分かる。そこで貢献するのがオントロジーである。

知識ベース構築には対象とする世界の概念化の明示化が不可欠である。そして、多くの人々の協調による「知識の積み上げ」による開発が必要となる。これはオントロジーの問題である。オントロジーは大規模知識ベースの構築の基盤を提供する。

参考文献

[溝口96a] 形式と内容—内容指向人工知能研究の勧め—、人工知能学会誌、Vol.11, No.1, pp.50-59, 1996.

[溝口96b] 「AIマップ—形式と内容—内容指向人工知能研究の勧め—」へのコメントと回答、人工知能学会誌、Vol.11, No.4, pp.550-565, 1996.

1.2　調査研究の目的

以上の考察から明らかなように、オントロジーは２１世紀の知識処理技術の根幹をなすと期待される。しかし、これまでオントロジーに関する研究は一部の理論家を除いて体系的な研究あるいは工学の観点からの研究は行われていない。内容指向AI研究[溝口96a] [溝口96b]、そして大規模知識ベースの構築の基礎を与えるオントロジーを工学の視点から考察を深め、オントロジーに関する基礎から具体的な応用までの幅広い研究課題を研究する学問、すなわちオントロジー工学の確立が、今強く望まれている。

そこで、本委員会ではオントロジーに関連する活動を行っている関連分野、知識処理、自然言語処理、エキスパートシステム、機械設計、データベースなどの専門家の参加を得て、オントロジーに関する包括的な検討を行うことによってオントロジー工学の可能性を検討することを目的として活動を行った。時間の関係もあり、今回の委員会活動では、「オントロジーとはなにか」という基本的な問題から考察を深めるとともに、我が国を含む世界におけるオントロジー開発の現状、オントロジーの理論、オントロジーの利用の角度から現状を調査し、今後のあるべき方向を検討した。

1.3　活動概要

本調査研究の２年間の活動の概要は以下のとおりである。

平成８年度では、のべ６回の会合をとおして、オントロジーの工学的活用への課題等を探るため、知識工学、言語処理、機械可読辞書、データベース、またスケジューリング問題、法律、医療などの応用分野からのオントロジーに対する期待、問題意識等について、委員各位の発表とそれに基づく検討を中心に行った。

また、1994 年にスタートしたISO（国際標準化機構）の概念スキーマモデル機能（CSMF: Conceputual Schema Modeling Facility）標準化プロジェクトの動向に関し、外部専門家からヒアリングを行うとともに、自然言語処理の観点からオントロジーに関連する研究開発動向を探るため、米国（Stanford 大学、Zerox PARC、SRA International）に調査員を派遣し、会合での議論の参考とした。これらの内容は、平成８年度報告書にまとめている。

平成９年度では、のべ８回の会合を開き、８年度の議論を踏まえ、工学的な観点からのオントロジーの定義づけの議論を重点的に行った。併せて、オントロジーに関する研究・開発動向（理論、記述言語・環境、機械化読辞書、活用事例など）について、インターネットを通じた情報収集を主体に、内外の動向を調査した。

また、異企業間でのデータ交換のため具体的な標準化が進んでいる、電子データ交換（EDI：Electronic Data Interchange）における商品属性の標準化動向について、外部専門家からヒアリングを行い、会合での議論の参考とした。

2. オントロジーとは

2．オントロジーとは

まず、オントロジーの定義から始めよう。複数の定義があることからわかるように、委員会として合意に達した定義はまだ得られていない。

2.1　いくつかの定義

(1)本来、オントロジーは哲学の用語であり、「存在に関する体系的な理論（存在論）」という意味を持つ。世の中に存在する全てのものを系統立てて説明することを目指している。例としては、もの／生物／動物／ほ乳類．．．などの概念の階層構造を構築し、なにが存在として本質的で、なにが存在物の区別をするのか？存在物の間の関係は？などと言うことを考え、その結果としての理論と体系をオントロジー（存在論）と呼ぶ。

(2)人工知能の立場からは、「概念化の明示的な記述」by Tom Gruber という定義がなされる。ここで概念化とは、対象（世界）に関して、興味を持つ概念とそれらの間の関係とを指す。我々が世界を認識し、そのモデルをコンピュータ内に作ろうとするときには、必ず、世界を概念化する。例えば、ビジネスプロセスのモデルを構築する際には、ビジネス自体を構成する概念を洗いだし、それらの間の関係を整理し、概念を特徴付け、他と区別する属性を明確にする。そして、それらの概念を記述する。概念のなかには、オブジェクト（名詞に対応）だけではなく、活動を表す動詞的なものもある。そして、活動が対象とするオブジェクトとそれらを関係付ける制約など、これら全てを体系化したものをオントロジーと呼ぶ。

(3)知識ベースの立場からは、「人工システムを構築する際のビルディングブロックとして用いられる基本概念／語彙の体系（理論）」by Mizoguchi という定義がなされる。知識ベースは対象とする世界における問題解決モデルを構築するときに不可欠となるものである。問題解決を対象とするので、オントロジーは問題解決過程に固有の概念化であるタスクに関するオントロジーであるタスクオントロジー、そしてタスクが実行される領域（ドメイン）に関わるドメインオントロジーの大きく２種類に分かれる。

(4)緩い定義

「ある目的のための世界の認識に関する共通の合意」 by Takeda。ここでは、定義の一般性を増すことに留意して意識的に合意内容を規定することは避けている。もう一つの特徴は、オントロジー目的依存性を明示していることである。

(5) 少し丁寧な定義 by Tom Gruber

オントロジーは複数の人の間で共有される合意内容である。共有される合意内容としては、ドメイン知識のモデル化の概念フレームワーク、相互に交信し合うエージェントの会話の内容に関わる通信規約、ドメインの知識を記述するときの合意事項、などが含まれる。知識共有の考えのなかでは、オントロジーは表現のための語彙の定義の形で記述される。最も簡単な場合としては、カテゴリーとそれらの間の上下関係（包含関係）を規定したもの（概念階層）であり、関係データベースにおける概念スキーマがある。

関連する語彙の定義

(a) Terminology（用語論）

概念に関する合意を得た後に、その概念に付与すべきラベルを決定する必要があるが、ラベルを何にするかを論じることが中心課題となる。言い方を変えれば、概念の呼び名を議論の対象としている。

(b) Vocabulary（語彙）

自然言語処理などで用いられる処理の対象とする単語の集合を指す。単語は必然的に概念を指示するためオントロジーと非常に近い関係にある。しかし、語彙は自然言語依存であり、普遍性に欠ける。しかも、語彙（単語）が指す概念の意味と単語間の関係の記述に関する配慮もオントロジーに比べて弱い。Terminologyでは概念が先にあるのに対して、Vocabularyは名前が先にあり、その意味の明示的な記述に興味の中心がある。

(c) Taxonomy（分類学）

一般には分類全般を指すが、ここでは概念分類を指すことにすると、オントロジーと最も近い概念である。概念分類はis-aリンクやpart-ofリンクを用いて階層的になされる場合が多い。概略的にはTaxonomyにおいて各概念の意味定義とその他興味ある関係を明確に記述したものがオントロジーといえる。

2.2　オントロジー定義に関する視点

オントロジーの定義に関しては未だに研究者の間でも議論が続いており、現時点では全員が合意する定義はないと言ってよい[Guarino 97]。実際、本委員会において半年ほどの時間を使って各委員が「私はオントロジーをこう考える」という意見を提案し、それを元にオントロジーの定義に関する議論を行ったが、やはり完全合意には至らなかった。そこで、ここではその議論に基づいて、オントロジーを考える際に必要と思われるいくつかの観点を整理して、それについて考察を加える。

2.2.1　オントロジーが持つ性質

(1) 目的依存性

哲学のオントロジー（存在論）は「分かる」という中立的なことを目的にして存在を考える。しかし、我々が対象のモデルを記述する際には、必ず何らかの個別の「目的」を持っている。目的は対象を眺める「視点」を明確にして、その視点に基づいてこそ初めて有効なモデルの構築が可能となる。実は、この視点が人間の認識によるものであり、後に論じる認識論と存在論との相互依存性の議論を生む。

(2) 一般性

世界に一つしかない対象、あるいは一人の人しか興味を持たない領域などに関するオントロジーは意味がないことは明らかであろう。しかし、万人が興味を持つものしかオントロジー構築の対象にしてはいけないと言うのも強すぎる制限である。例えば「変電所のオントロジー」はどうであろう？一見狭すぎる例であるように思える。しかし、日本にある変電所の数は万のオーダーであり、変電所オントロジーはそこでの問題解決の機械化に貢献する十分な一般性を持っている。

(3) 共通性・合意

次に重要な視点は共通性である。構築されたオントロジーは多くの人の合意を得ているべきであり、それらの人々にとって共通であることが必要である。一般にオントロジーの合意には次の4つのレベルがある（by Dough Skuce）。

(a) 構築の方法論
(b) 概念の意味
(c) 概念の名前
(d) 公理

通常は、対象世界の概念化のレベル、即ち、(b) の段階が最も骨の折れる作業となる。情報科学的には、最後の(d) も極めて重要であり、かつ骨の折れる仕事である。

(4) 安定性

オントロジーは作成者、作成時間などによらずある程度安定したものでなくてはならない。言い換えれば、存在論という概念が示唆されるように、そうでないものはオントロジーとは言えないということもできよう。しかし、実際には作成者によって揺らぎが生じることは無視できず、それを吸収すること、細部における相違を越えて合意に達することの難しさが存在する。

(5) 形式性（コンピュータ理解可能性）

オントロジーの記述には自然言語による記述と形式論理による記述の大きく二つある。いずれでなくてはならないと言うことはない。自然言語は表現力が豊かであり、意味を表現できる唯一の表現形式である。しかし、解釈の曖昧性は免れない。論理による記述は解釈の一貫性は高いが、記述力は自然言語より劣る。形式化によって多くの意味が欠落する。一長一短である。しかしここで重要なことは、論理による表現は計算機が理解可能であることである。コンピュータ理解可能性を追求することによって、概念の厳密さが増すと同時に人間の理解のレベルである知識レベルとコンピュータ理解レベル、即ち記号レベル（実行レベル）までの異なったレベルの間の（ある程度の）連続性が保たれる。言い換えると、人間の概念操作がコンピュータの実行で裏付けられる訳である。このことは人間とコンピュータとの意味共有ということを意味しており、オントロジーで構成されたモデルに基づく知識ベース（問題解決システム）が容易に構築できる。

(6) 部分性

目的依存性、そして形式化による意味の欠落、などの要因からオントロジーは部分的にならざるを得ない。すなわち、オントロジーが表す意味、対象とする範囲、合意の程度、共通性などは全て部分的である。しかし、これはあらゆる「記述」の宿命でもある。

(7) 一貫性

記述されたもの全てに共通なものではあるが、オントロジーは一貫していなくてはならない。これは特に形式化されたオントロジーでは必須である。逆に言えば、オントロジーの形式化は一貫性の維持に貢献するということもできる。

(8)明示性

全ての宣言的記述は明示性を持つが、特にオントロジーは「世界（対象）に関する合意」という元来暗黙的であったものを明示化したものであり、明示性という性質は特徴的である。

(9)部品性

オントロジーが「概念化の明示的記述」であることから、そこに含まれる概念は対象とする世界を記述する際の部品としての性質を持っている。

2.2.2 オントロジーの構成的定義

ここでは、「オントロジーとはどういうものであるのか」ということを記述された結果としてのオントロジーの観点から論じる。

(1)概念の切り出し

オントロジー構築にはArticulation（分節化）操作が伴われる。従って、オントロジーには対象世界に存在する「概念」が主要な構成要素として含まれる。

(2)Taxonomy（概念分類、階層的表示）

抽出された概念はis-aやpart-of関係に基づいて階層的に組織化されることが多く、それはTaxonomyと呼ばれる。

(3)概念間の関係記述

Taxonomyにおける概念間の関係にはis-aとpart-of関係しか記述されないが、オントロジーにはその利用目的に応じて必要となる様々な関係が記述される。

(4)形式的定義

以上の記述は多くの場合述語論理などを用いて、公理の形で形式的に記述される。

2.2.3 オントロジーの機能

オントロジーの目的依存性から、オントロジーはその利用目的に合致した様々な機能を持っている。ここでは、それらの機能を数え上げてみよう。

(1)語彙の提供

対象とする世界を記述する際には厳密に定義された、関係者の合意に基づく「語彙」が必要であるが、オントロジーはそのような標準的な語彙を提供する。

(2)メタモデル

モデルは現実の対象を抽象化してコンピュータ内部に作られる。そして、オントロジーはある対象をモデル化するときに必要となる概念とそれらの間に成立する関係を明示的に規定し、そのモデルはオントロジーが提供する概念と制約の下で作られる。この意味で、オントロジーはメタモデルということができる。

(3)暗黙情報を明示化

ある領域において仕事をしている人々にとって無意識に仮定しているものが多くある。その典型が日頃頻繁に用いている用語（概念）の定義であり、それらの概念間に存在する基本的な関係や制約である。また、知識ベースは何らかの概念化に基づいているが、その概念化に関する情報は多くの場合暗黙的である。オントロジーはまさにこのような暗黙知識を記述したものであり、それらを明示化する役割を持っている。

(4)概念定義

厳密な定義をすることなく日常的に用いている概念の明確な定義を提供する。

(5)データ構造

データベースの概念スキーマはデータベースのオントロジーであるが、その意味においてオントロジーは対象とする世界に存在する概念や情報を記述するデータ構造を提供する。

(6)知識の体系化

知識を体系化するにはまず厳密に定義された概念（用語）が必要である。そしてそれらを用いて様々な現象、観測事象、興味ある対象を説明する理論が記述され、組織化される。オントロジーは知識を体系化する際の拠り所となるバックボーンとしての機能を持っている。

(7)標準化

今後の知識処理の高度化、高能率化を考えるとき、標準化の問題を避けて通ることはできない。工業製品の生産性の高さを考えれば明らかなように、必要な基本部品の標準化は生産性の高度化に貢献するところが大である。ボルトとナットに対応するような規格品を知識処理の世界でも作る必要がある。オントロジーは標準概念の意味を

規定するものであり、オントロジーの一般性、共通性、形式性は標準化に貢献する。

(8) 設計意図

オントロジーはシステム設計者が持っている対象に関する理解、システム設計の意図、すなわちdesign rationaleを明らかにする。オントロジーは前提とされている条件や環境、解くべき問題が要求する仮定などの暗黙的な情報、そしてそれを反映した対象の世界の概念化に関わる根本的な情報を明示し、知識ベースの構築を支えるバックボーンとして機能する。

(9) Theory of content（内容の理論）

以上のことを総合して、オントロジーはともすれば個別の状況に依存したアドホックな議論になりがちで、形式理論の様に積み重ねが効くような方法論を持たなかった「内容指向研究」に対して、内容を扱うために必要な新しい「理論」、Theory of Content を提供する。

2.2.4 オントロジーベースのシステムのメリット

オントロジーの実用的な効果を見るためにオントロジーに基づいて開発されたシステムが持つと期待される利点を考察する。

(1) 再利用性

システム、各モジュールが基づいている世界モデルの明示的な記述はシステムの仮定や前提を明らかにし、その再利用性を向上させる。しかし、再利用にはモジュールの検索、検証、修正などの操作が必要であり、システムの透明性だけでは不十分である。

(2) 相互運用性

上述の前提知識の明示化に加えて、システムの機能や通信プロトコルの記述などが容易になるとともに、オントロジーの形式性が検証を容易にするので相互運用性が向上する。

(3) 実装容易性

対象世界の概念化とその形式的記述はデータの論理構造や概念（オブジェクト）の機能の形式的仕様記述としての役割を果たすため、概念設計結果が詳細かつ厳密に記述され実装が容易となる。

(4) 知的ツール

ツールはそれが構築するシステムが持つべきモデルに関するモデル（メタモデル）を持っていなければならないが、オントロジーが持つメタモデルとしての性質はまさしくそのためにあるようなものである。言い換えれば、オントロジーは何を生成するためのツールであるかということを知っている知的なツールの構築を可能にする。

2.2.5 オントロジーとエピステモロジー（存在論と認識論）

存在論は「存在」を人間の認識から独立に論じることを目的とする。一方、認識論では、存在は人間の「認識」なしに説明できないということを前提として人間の「（知識）認識とその能力」を考察するものであり、基本的に相容れない要素がある。しかし、これは哲学の分野での話である。我々には、「もの」は認識に無関係に「存在」する事は素直に受け入れらるし、認識なしには「存在」を説明できないことも頷ける。しかも、いずれの用語もその学問分野全体、あるいは考察の試み全体を指すものであり、概念の階層構造を指すものではない。

情報科学が興味を持つオントロジーは目的に依存するものであることから、人間の認識を通した後の事物を語っていることは明らかである。しかし、上で述べたように哲学における用語そのものの語義とは異なった意味で情報分野の人が使ってしまって定着してしまったためもはや修正は不可能な状態にある。かといって我々は「認識論」を議論しているのでもない。溝口の個人的な見解であるが、情報科学の人間がいうオントロジーは、「認識を通して見える存在を説明するための理論」であり、コンピュータ内部に構築可能な（表現可能な）世界における存在を議論していると見なしてよいのではないかと思われる。

2.2.6 オントロジーの３つのレベル

オントロジーという言葉が意味するものが多様になり、不要な混乱が生じていると思われるので、溝口の個人的な見解であるが、ここでオントロジーに関する３レベルのレベル分けを行おう。

(1) レベル１

オントロジーの基本的な機能は、対象世界に存在する概念の切り出し（選択）とそれらの関係の記述である。最も一般的で簡単な記述が階層関係の記述であり、そこには概念のラベルと階層記述だけが存在する。このレベルのオントロジーを

最もプリミティブなものであるという意味で、レベル１オントロジーと呼ぼう。

(2) レベル２

次に各概念の意味定義（制約）や関係の記述（公理的記述）が加わることによって、オントロジーを利用したモデル構築において種々の適切なガイドや示唆を与えると共に、オントロジーを用いて記述できるもの全体の性質（Competence）に関する質問に答えることができる。このオントロジーをレベル２オントロジーと呼ぼう。

(3) レベル３

このレベルのオントロジーはオントロジーを用いて構築されたモデルの、ある問題解決における実行のパフォーマンス（Performance）、即ち、オントロジーを用いて記述したものが実行されたときの振る舞いに関する質問に回答する。このようなパフォーマンスに関する質問に対しては、本質的に手続き的な記述が必要な場合があり、形式的な公理と証明系では答えることができないことが多いのが現実である。そこで準公理が必要となる。

参考文献

[Guarino 97] Understanding, building and using ontologies, Int. J.Human-Computer Studies, Vol.46, pp.293-310, 1997.

2.3 FAQ：Frequently Asked Questions

ここではよくなされる質問とその解答（ある物は解答というより検討のレベルのものもあるが）を示す。

(1)オントロジーと知識ベースは何が異なるのか？

以下の解答は Ontolingua mailing list において Stanford 大学の Adam Farquhar <axf@HPP.Stanford.EDU>によって、Wed, 26 Feb 1997に掲載された意見の翻訳である。

(a) それはその分野の人々（エージェント）の合意した知識を表現したものか？
(b) 人々はそれを正確に定義された用語として参照しているか？
(c) その表現に用いられている言語は人々が言いたいことを表現するのに十分な表現力を持っているか？
(d) それは複数の異なった問題解決に再利用可能か？
(e) それは安定しているか？
(f) それは、新しい知識ベース、データベーススキーマ、そしてオブジェクト指向プログラムなどの複数のアプリケーションプログラムを開発するための出発点として利用可能か？

Yesの解答が多ければ（強ければ）多いほど、それだけオントロジーらしくなる。

このように、「オントロジーと知識ベースには明確な境界はない」というのも一つの立場である。それは、オントロジーも言うまでもなく、知識の一種であることから当然の帰結であるとも言える。しかし、「知識ベース」の意味がきわめて曖昧であるのでその定義をすることも必要であろう。例えば、Cycのような知識ベースを意味するのであればこの問はまさに的を射た問であるといえる。しかし、専門家の経験則を蓄積したエキスパートシステムのルールベースを想定すれば、オントロジーとの比較は問題外となろう。なぜならそのようなルールベースは開発者が持つ（暗黙的な）オントロジーに基づいて構築されているからであり、決してオントロジーを表現していないからである。

(2)オブジェクト指向方法論、あるいはクラス階層と何が違うのか？

まず、オブジェクト指向言語のクラス階層とオントロジーの概念階層との類似性は高い。また、オブジェクト指向方法論はその上流における分析ではオントロジー開発

方法論との共通点は多い。しかし、実装に近いフェーズでは前者は実行性能を重視し、後者は宣言的記述に基づき明示性、形式性を重視する。決定的な相違点は公理の「宣言的」かつ「明示的」記述性の有無にある。すなわち、オブジェクト指向言語は本質的に手続き的である。そのためクラスの意味、クラス間の関係、そしてメソッドの持つ意味はその手続き的記述の中に埋もれており暗黙的である。一方、オントロジーでは各概念の意味定義と概念間の関係記述は宣言的、かつ明示的である。手続きに関してもその仕様は宣言的に記述され、形式性、明示性は維持される。

(3)何が新しいのか？ Taxonomyと何が違うのか？

オントロジーは「Taxonomy（概念階層）」という意味を含意するが、それ以外の意味も含んだ概念である。一般に、新しい用語・概念は既存の概念を発展させたものが多く、全く新しい概念であることはまれである。多くはそれまでに存在していた概念を拡張したもの、あるいは新しい観点から再評価したもの、などである場合が多い。オントロジーもその例外ではない。すなわち、従来から存在していた「概念階層」、「標準語彙」、「Upper model」などの概念を含むとともに、知識ベースの開発に際してなされている暗黙の仮定、前提となっている、対象世界の概念化（利用目的に依存した概念とそこで興味のある概念間の関係）を述語論理などを用いた明示的かつ形式的な定義という新しい概念が付加されており、概念間の関係がリッチになる。

(4)具体的にどのようなメリットがあるのか？

ある意味で最も解答困難な問である。オントロジー研究の歴史は短く、その成果の具体的な提示をするに至っていないからである。従って、メリットの主張は抽象的にならざるを得ない。以下にメリットをリストアップしておく。

共通語彙の提供
暗黙知識、前提知識の明示化
システム構築のビルディングブロックの提供
システム構築時にガイドラインを提供
領域知識に基づく類推を支援
再利用性
相互運用性
実装容易性
知識の共有・再利用を促進
Theory of contentの提供

(5) 誰が作るのか？

エキスパートシステムの知識ベースの開発においてもこの種の質問はしばしばなされた。知識ベースを構築するのは知識工学者かドメインエキスパートか、という質問である。どちらでなくてはならないということはない。いずれの場合においても両方の知識が必要であることには変わりはない。

Upperオントロジーは本質的には哲学者の領分であるが、コンピュータ科学者の協力によってコンピュータという「武器」を駆使してこれまでにない、コンピュータ科学のニーズに合致したオントロジーが構築される可能性がある。

(6) 汎用性の高い上位レベルオントロジーとタスク依存のオントロジーとは両立するか？

工学的にはオントロジーはその利用目的に依存するものであって、汎用のオントロジーは使いものにならないという考えが支配的である。その代表的な考えがタスクオントロジーであり、問題解決に用いられるオントロジーはタスクが要求するように組織化されているべきであるというものである。しかし、一方ではUpperモデルと呼ばれる汎用のオントロジーが存在する。両者の距離は遠い。まず思想の相違は一見決定的であるように見える。しかし、両者の用途を考えると、Upperモデルは領域やタスクを越えた多くの人々の間で共有することを目指した概念化であり、タスクオントロジーに現れるタスク固有の概念においても仮定される基本的なものである。実際、両者の遠い距離を埋めた例は少ないが、法律オントロジーでの成功に見られるように不可能であるとは思われない。タスクオントロジーは合意可能なUpperモデルを前提としてその上に積み上げる形で設計すべきなのであろう。

(7) 共有とはなにか？どの範囲でどの深さで共有することを目指すのか？

オントロジーの共有は可能か？万人が共有するオントロジーがあり得るとは思えない。哲学者が数千年議論してきて未だに上位のオントロジーでさえ合意が得られていないことがその困難さを示している。また、細部にわたった詳細な概念全てに合意をしてそれを共有することが可能であるとも思われない。オントロジーの共有に悲観的な考えが生まれる根本的な理由はここにある。しかし、これはこれまでの哲学者のオントロジー（存在論）に関する評価である。

我々コンピュータ科学者が必要なオントロジーは目的に依存するオントロジーであり、存在一般を対象とするものではない。そして、最大の相違点は我々にはコンピュータという強力な武器があることである。数１０万件の概念を記述してその中の矛盾を

瞬時にして検出することができる。形式的に記述された二つのオントロジーを比較し、その相違を吸収する支援システムの利用もいずれ可能になる。万人が深いレベルのオントロジーを共有することは不可能であると考える方が健全であることは言うまでもないが、悲観的になることもまた慎むべきであろう。工学的な立場ではむしろ、どの範囲であればどの程度の深さのオントロジーを共有が可能であるかを、コンピュータという強力なツールを前提にして積極的に考えることが重要である。

(8) 共有と再利用とはどこが異なるのか？

たしかに非常に紛らわしい。用語の問題として考えると、「再利用」には生産性の向上という経済上の効果をもたらす行為を意味するところがあり、「共有」は状態を意味することが主である。行為と状態という対比がみられる。一方、機械設計の分野で考えると実際の部品が「再利用」されることはなく、重要なのは「標準化」された情報を「共有する」ことであることを考えれば分野によって意味が異なることが分かる。しかし、我々のオントロジーはソフトウェア分野の問題であるので、ここでは「再利用」「共有」という言葉はソフトウェアの分野で用いられる意味であるとする。いずれにしても、「標準化」されていることが本質的に重要である。別の見方として、用語の問題としてではなく、それと関係の深い行為を直接考察することが挙げられる。知識ベースの中の知識を再利用する際には知識ベースをホワイトボックスとして扱う立場とブラックボックスとして扱う立場とがある。前者は各知識を詳細に分析し、理解した後に再利用できそうな知識を選定する。後者は、知識ベースに問い合わせをして対応する解答を得ることで満足する。その際、知識ベースの中でどのような処理が行われたかは問わない。この二つを指す適切な用語は見あたらないが、強いて言えば前者を「再利用」、後者を「共有」と呼べそうに思われる。

3．　オントロジーに関する研究・開発の動向

3．オントロジーに関する研究・開発の動向

3.1　基礎理論

3.1.1　Guarinoの形式的オントロジー

Guarinoは哲学での議論を元に数理論理学の手法を用いてformal ontology の問題に取り組んでいる[Guarino93a] [Guarino95a] [Guarino97a]。ここでのformal ontologyとは実際の現実とは独立に領域の要素の形式的な区別を可能とする理論のことである[Guarino97a]。そこは知識表現との関わり合いでは以下のようになる。

知識表現における表現レベルとしてのオントロジーを認識的レベルでの知識の表現と概念レベルの知識の表現の中間に位置づけた[Guarino95a] [Guarino93a]。ここでの概念レベルの表現とは人間が理解可能な形での概念やその関係の表現であり、認識的レベルとは概念要素やその関係の形式的表現である。ここで問題なのは認識的レベルで与えられている形式的表現が我々のもっている概念レベルでの表現に含まれている仮定を含んでいないことにあると考え、オントロジー・レベルにおいてこの仮定を表現するレベルとして位置づけている。

具体的には概念や属性といったものをどう形式的に分類するかが目標である。彼の問題提起はこれまでのオントロジーの定義はad hocであり、本当の意味での形式的定義になっていないということである。彼のアプローチとしては哲学における事物の区別に関する帰結を数理論理学の手法を使って定式化するものである。

以下では具体的な成果として、(1) top-level ontology の構成原理、(2) ontological commitment について説明する。

(1)　top-level ontology の構成原理

まず、top-level ontology の構成原理についての議論がある[Guarino97a]。これは一連の彼の研究の集大成的研究である。

まず、形式オントロジーformal ontology 構成のための必要となる背景理論としては次のものがある。

◎ 2つの先験的分別

- 特殊particulars：実世界の実体の認識。物理的な物体、事象、空間領域、量
- 普遍universals：我々の議論領域に含まれる実体について語るときに使うカテゴリー。概念、属性、質、状態、関係。

◎ 部分の理論

例えば以下の質問に対応するための理論である。

・ 何を実体の部分としてみなすか。
・ 部分性 parthoodのもつ属性とは。
・ 部分には違いがあるか。

部分性の理論の例としては外延的 mereology、あるいは内包的 mereology がある。

◎ 全体性の理論（完全性の理論）

全体性の理論とはどのように要素を接続したらひとつの全体とみなされうるのか、についての議論。例えば以下の質問を答えるような理論である。

・ 何を全体とみなすか。何が全体をつくるか。
・ 接続されているとはどういうことか。接続関係に属性はあるか。
・ 背景から切り分けられるか。境界とは。
・ 全体にとって部分はどんな役目を果たすか。

◎ 同一性の理論

同一性の理論は異なる属性を持つ実体がいかなるときに同一になりうるかの理論であり、部分性の理論と全体性の理論のもとに成り立つ。以下のような質問に答える理論である。

・ 同一性を保ちつつ実体は変化できるか。
・ 実体に本質的属性はあるか。
・ どういう状況で同一性を失うか。
・ 部分の変化は同一性に影響を与えるか。
・ トポロジーあるいは形態的変化は同一性に影響を与えるか。
・ 視点の変化は同一性の条件を変えるか。

◎ 依存性の理論

特定の個体が異なるクラスに含まれるという外延的な依存の形態に関する研究。次のような質問に答えるような理論である。

・ 個体の具体的存在は他の特定の個体の具体的存在を必然的に含意するか。
・ 個体の具体的存在は特定のクラスに属する他の個体の具体的存在を必然的に含意するか。
・ ある個体の具体的存在が特定のクラスに属することは他の個体の具体的存在が、別のクラスに属することを必然的に含意するか。

Top-level ontology はad hocではよくなく、構成原理に基づく必要があり、その原理の

主要な点は以下の通りである。

・特殊 particular と普遍 universal の２つのオントロジーのカテゴリー。
・普遍 universals のontology の全ての１引数関係が特殊 particulars のontology の一部が明示的な概念に対応する。それをtaxonと呼び、適切な構成基準によって選べれるものである。
・taxon はStrawson のいうsortal なカテゴリーであり、そのインスタンスに特定の同一性基準を含意するものである（例：物理的物体 physical body、事象 event）。非sortal なカテゴリーとは特定の同一性基準を持たないもので、分解可能なものや赤いもの red thingといったものをさす。
・全ての taxon はプリミティブである。必要条件といった形で、形式的なオントロジー的属性はそれらの意味を特徴づけるに役立つ。
・taxon は主にその同一性基準に従って構成される。異なる同一性基準は別々のtaxon に対応する。同一性基準はグループ化され、層をなす。このオントロジーの層は依存性関係がある。

以上のような方針の元に top-level ontology の構成を図３．１．１-１のように提案した。以下、その概要について説明する。なお、普遍 universalsのオントロジーは後述の ontological commitment で説明する。

Substrate
  Location
    Space
    Time
  Matter
    Iron
    Wood
    ....
Object
  Concrete object
    Continuant
      Singular -
        Configuration
        Singular body
          Physical body
            Topological body
              Topological whole
                (a planet)
              Topological piece
                (a piece of wood)
            Morphological body
              Morphological whole
                (a cube of wood)
              Morphological piece
                (a mountain)
            Functional body

Functional whole
Artifact
...
Functional piece
Artifact component
Organic structure
...
Singular feature
Physical feature
Topological feature
(the boundary of a blob)
Morphological feature
(the top of a mountain)
(a hole in a piece of cheese)
Functional feature
(the head of a bolt)
(the profile of a wing)
Plural -
Plurality (Plural body)
Physical plurality
Mereological plurality
Topological plurality
(a lump of coal)
Morphological plurality
(a line of trees)
(a pattern of stars)
Functional plurality
(a disassembled artifact)
Plural feature
(a hole in a lump of coal)
(the sword of Orion)
Occurrent
Situation
Physical occurrent
Topological occurrent
Morphological occurrent
Functional occurrent
Biological occurrent
Intentional occurrent
Action
Mental event
...
Social occurrent
Communicative event
Abstract object
Quality
(the color of this rose)
(the style of this talk)

図３．１．１-１　top-level ontology

◎ 特殊 particulars のオントロジー

基質 Substrate は物体が依存する究極の実体であり、その特徴はそれらが外延的同一

性基準をもつということである。物体 object は非外延的同一性基準をもつ点で基質と異なる。すなわち、部分を喪失あるいは追加されても同一性が不変になりうる。具体的物体 concrete objectと抽象的物体 abstract object は空間的・時間的位置の有無で区別される。

具体的物体には継続体 continuant とoccurent に分けられる。

Particular
- Substrate
  - Location
    - Space
    - Time
  - Matter
- Object
  - Concrete object
    - Continuant
    - Occurent
  - Abstract object
- Qulality

さらに具体的物体を以下のような層strata によって分類する。これらはそれぞれ異なる内包的同一性基準を与えるものである。すなわち、ひとつの物理的存在に対して、これらの層それぞれ異なる同一性を与える可能性、例えばそれが一つの対象と受け取られたり、複数と受け取られたりなどの違いがありうるということである。下の層は上の層に依存して、層間に論理的関係が与えられる[Borgo96]。

Static	(a situation)
Mereological	(an amount of matter)
Physical	
Topological	(a piece of matter)
Morphological	(a cubic block)
Functional	(an artifact)
Biological	(a human body)
Intentional	(a person or a robot)
Social	(a company)

(2) Ontological commitment の形式化

知識表現で用いられているクラス、属性などの用語は概念レベルでの我々の直観的表現を借用することでつくられているが、それらに課せられた制約、役割は明らかではない。この制約が ontological commitmentであり、この論理的性質を明らかにするということが必要である。そこで、ここでは1引数述語の意味的なカテゴリーを論理的に定義している[Guarino95a]。

Strawsonによると1引数述語の種類は図3.1.1-2のように分けられる。Sortalとは

その指し示す個体を識別する原理を提供するものであり、非sortalとは他の原理によって識別された個体をさらに識別するような原理を提供するものである。例えば「りんごは赤い（Apple is red.）」というときに、「りんご」はsortalであるが、「赤い」は非sortalである。Sortalはさらに「りんご」や「人」といった「実質的（substantial）」なものと「食物」や「学生」といた「非実質的（non-substantial）」に別れる。また非sortalも「もの」といった「一般的述語（generic predicate）」と「赤い」といった「特徴づけ述語（characterizing predicate）」に分けられる。

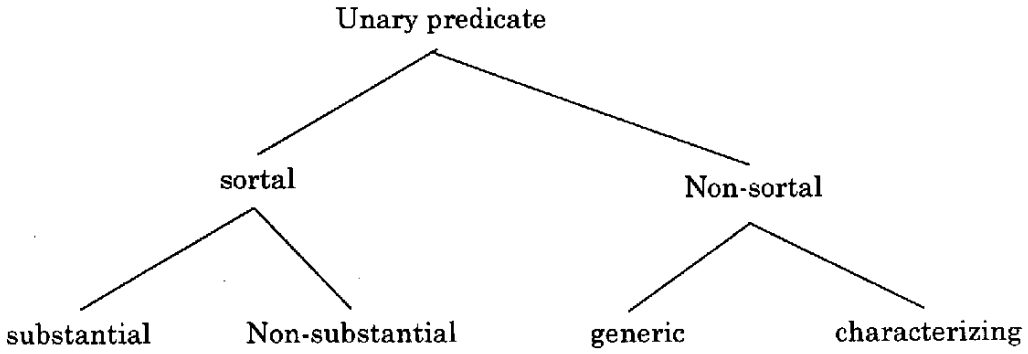


図３．１．１-２　引数述語の基本的オントロジー

Guarinoはこれらの分類をS5の様相論理を用いて形式化した。S5の様相論理では、すべての可能世界がお互いにみることができる。

まず、ある述語が「自然（natural）」であるとは、それをどこかの世界で満たす要素が存在すること（C | =∃x◇Px）である。

ある述語が「オントロジー的に固定されている（ontologically rigid）」とは、その述語が自然でありかつ、もしそれを満たせばどの世界でも満たされる（C | =∀x(Px⊃□Px)）ときである。

ある「自然」な述語が満たす要素が部分を持つことが可能であり、そのような部分の要素でその述語を満たすものが常に存在するならば「分割可能（divisive）」であるという。すなわち、今、部分を示す「<」を導入し、C | =∃x◇(Px ∧∃y. y<x) ∧∀x□(Px⊃(∃y.(y<x⊃Py)))となる。

ある「オントロジー的に固定的」な述語が「分割可能」であれば「一般的述語」であり、そうでなければ「実質的にsortal」である。

ある「オントロジー的に固定的」でない「自然」な述語が「実質的にsortal」な述語に包含されるならば「非実質的にsortral」であり、そうでなれけば「特徴づけ述語」である。すなわち、C | =∀x□(Px⊃Sx) である。

ontological commitmentがwell-foundedとは、すべての要素が「実質的にsortal」な述語に属しかつ、どの「実質的にsortal」な述語同士も包含関係であるか排他的であるかで

ある。

図３．１．１-３に単語 red の異なる利用法での異なるオントロジーにおける役割を示す。また、知識表現での形式化における用語とここでの用語の対応を図３．１．１-４に示す。すなわち、知識表現での形式化はこのような ontological commitment があるということである。

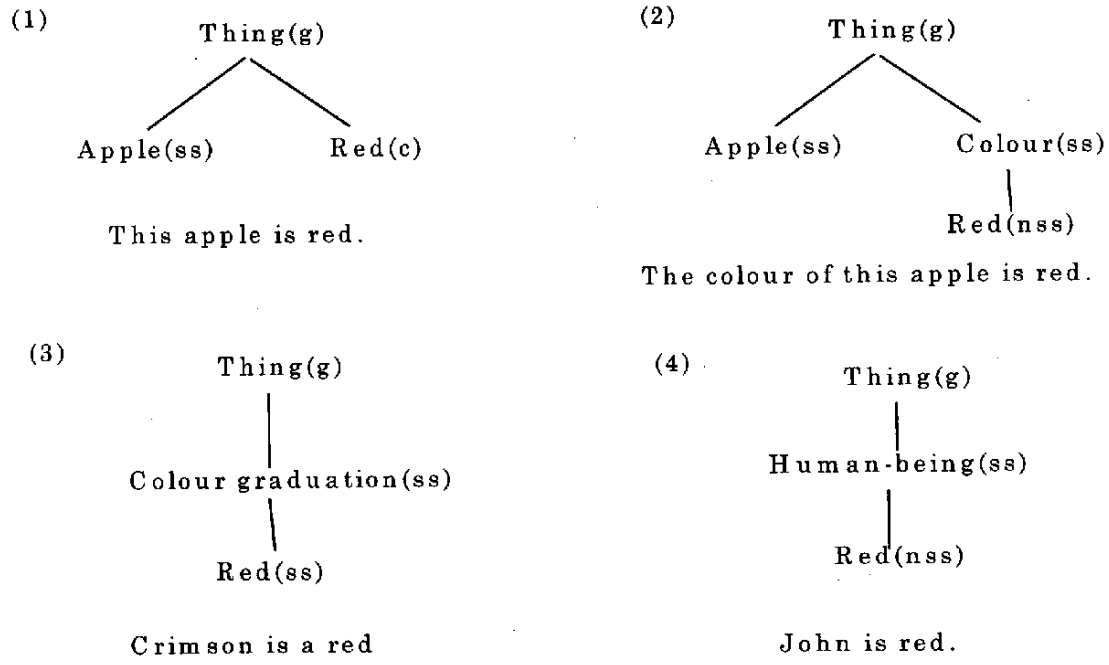


図３．１．１-３　叙述によって異なるontological commitment を持つ例

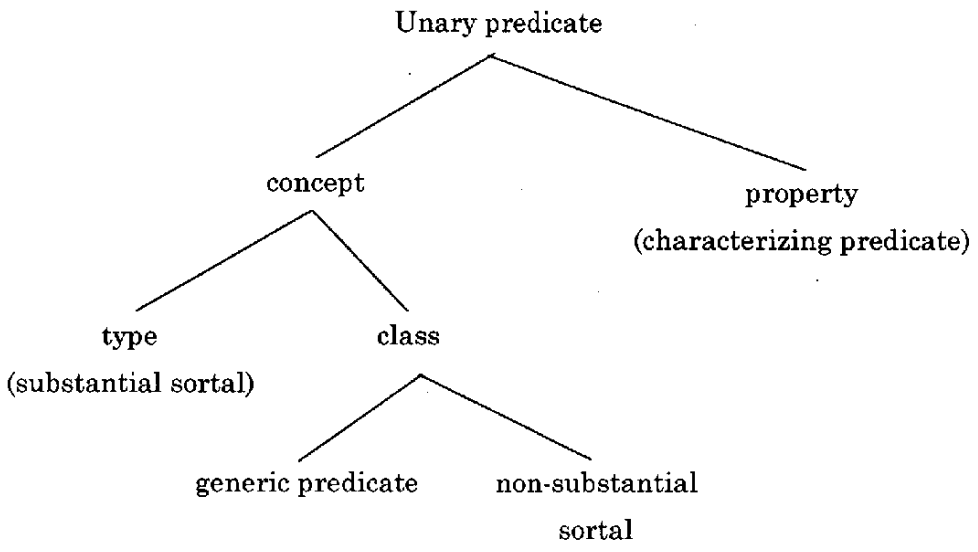


図３．１．１-４　知識表現の形式化との対応

参考文献

[Guarino97a] Guarino N.,Understanding, Building and Using Ontologies. A Commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga. International Journal of Human and Computer Studies vol.46 n. 2/3, 1997, pp. 293-310.

[Guarino95a] Guarino N., Formal Ontology, Conceptual Analysis and Knowledge Representation, International Journal of Human and Computer Studies, special issue on The Role of Formal Ontology in the Information Technology edited by N. Guarino and R. Poli, vol 43 no. 5/6, 1995

[Guarino93a] Guarino N. The Ontological Level. Invited paper Presented at IV Wittgenstein Symposium, Kirchberg, Austria, 1993. In R. Casati, B. Smith and G. White (eds.), Philosophy and the Cognitive Sciences, Vienna, Holder-Pichler-Tempsky 1994.

3.1.2 Sowaの理論とConceptual Graph

(1) Sowaの理論

Conceptual graph (CG) はJohn Sowaによって開発されたグラフ形式の知識表現システムである。PeirceのExistential graphという論理体系に概念的な階層構造を組み入れたもので述語論理と同等の一般性をもっている。

述語論理との比較においてCGの意義は、「オントロジー」を明確に位置づけようという考え方が定式化の根底にある点に見いだせる。数理論理学の理論的体系において無視されていた区別を明確にしている点が重要である。Sowaはこのことを次のようにまとめている。

○論理は純粋に形式であり、オントロジーは内容を提供する。
○オントロジーがなければ、論理は何についても意味のあることを言えない。

オントロジーという用語がCG研究の前面に現れたのは1970年代に始まるCGの歴史から言えば最近のことである。しかし、その重要性はCGの最初の教科書[Sowa 84]のp.264で概念分析の説明において次のように指摘されている。

> 概念分析は分野ごとに違った名前で呼ばれている。例えば、計算機システムの分野ではシステム分析、企業分析、知識工学という名前で呼ばれている。名前は異なっていても、その本質は概念、関係、事実、第一原理のカタログを正確かつ明示的に捉えることにある。...[中略] この分析の結果はオントロ

ジーである。オントロジーは世界を構成する「もの」のカタログであり、その「もの」をどう組み上げるか、そしてそれがどのように働くのかを明らかにするものである。

また、p.361ではより明示的に「概念タイプのカタログは存在の様式を捉えるものとしてオントロジーである。」と述べ、その構築にあたっては「抽象度が高い概念を定式化を考えると哲学的な問題になり困難さが極まる」と、オントロジー研究の現状を予言しているような記述も見られる。

ここで触れられている「概念タイプのカタログ」が一般の論理体系では明確に区別されていない部分である。直感的に概念タイプは述語論理での述語にあたり、カタログはその間に成り立つ包摂関係を表すラティスである。

もちろん、包摂関係は述語論理の体系で普通に議論されている関係で決して目新しいものではない。しかし、概念化の公理の一部として重要なことは明らかであるのに、その関係に特別な注意がはらわれていない点にオントロジー工学の基礎理論として問題が感じられる。また、オブジェクト指向やフレーム言語を考えれば、包摂関係はクラス間のis-a関係を記述するメタレベルの記述の枠組みに相当しており、そこでもやはり目新しくは感じられない。しかし、その理論的な側面は多くの場合に処理系に手続き的に埋め込まれておりオントロジー工学の基礎理論となり得ない。CGでは「概念タイプ」を表現として区別して扱い、その扱いを重視して意味論を明確にしながら理論を展開している。この点でCGはオントロジー工学の基礎理論として評価する価値がある。

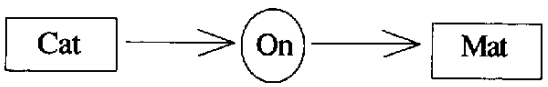
このような理論的な差異はCGの表現の枠組みを一瞥しただけではなかなか認識されないことが、SowaのCG研究に関して「意味ネットワーク」と同等であるという評価が下されがちな要因になっているように思われる。出版予定のSowaの本[Sowa]ではオントロジー工学の基礎理論として有益な示唆がみられる。

以下では、CGの基本的な定義を紹介する。

(2) Conceptual Graphの概要

CGは有限の大きさを持つ連結グラフであり、概念（Concept）を表すノードと概念的関係（Conceptual Relation）を表すノードの2種類のノードと、型を伴わないアークで構成される。

例えば、英文の "A cat is on a mat" に対応するCGは次のようになる。



このようなCGの図的表現はDisplay formと呼ばれ、概念をボックスで、概念的関係を円で表現しており、我々にとって読みやすい形式になっている。このDisplay formと等価でよりコンパクトな表現としてLinear formという表現形式があり、上の例は以下のように表記される。

[Cat]<-(On)<-[Mat].

ここでは概念は[]で、概念的関係は()で表現されている。Display formとLinear formは主に人間と計算機の両者にとって受け入れやすいように設計された表現である。計算機どうしのコミュニケーションのためにはConceptual Graph Interchange Format（CGIF）という、より簡単な文法と少ない文字セットで表現される枠組みが用意されている。

例えば、上の例は、

[Cat:*x] [Mat:*y] (On ?x ?y)

と表現される。この表現は次のようなKIFによる表現に容易に変換できる。

(exists ((?x cat) (?y mat)) (On ?x ?y))

①概念タイプ

概念タイプは概念を表象するラベルである。全てのCGシステムには基本となる概念タイプの集合と、概念間の包摂関係（<）による階層構造が既定のものとして与えられていることが仮定されている。後述するSowaのオントロジーはこの概念タイプの階層の内の最上位の構造を表しており、具体的な知識ベースシステムや自然言語処理システムが対象とする概念階層を組織化するための基本的な枠組みとして位置づけられている。

図３．１．２-１は概念タイプの階層構造の一例を示しており、例えば、概念タイプPHYSICAL-OBJECT、ANIMAL、MAMMAL、CATの間には、

PHYSICAL-OBJECT < ANIMAL < MAMMAL < CAT

という関係が設定されている。

新しい概念タイプの定義は以下のような形式で記述される。

MaineFarmer = [Farmer:&lambda] <- (Loc) <- [State:Maine]

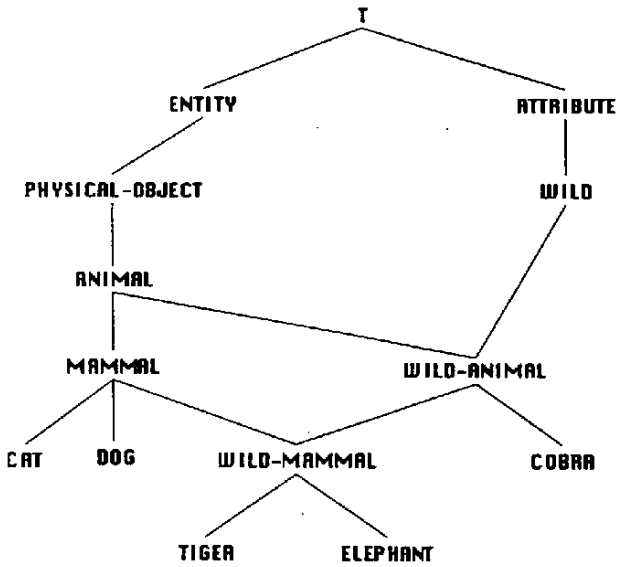


図３．１．２-１　概念タイプの階層構造

この表現によってMaine州に住むFarmerを表す概念タイプMaineFarmerが新たに定義され、階層構造中のFarmerの下位にMaineFarmerが新たに組み入れられる（MaineFarmer < Farmer）。

CGでは「関係」についても同様に関係タイプの階層構造が定義されているが、ここでは省略する。

②概　念

概念は概念タイプのインスタンスである。概念は概念タイプと参照子（Referent）によって[概念タイプ: 参照子]の形式で表現される。参照子によって同じ概念タイプについて次に示すような様々な概念を表現することができる。

[Cat] , [Cat:*x] ：ある猫。(exist x Cat(x))

[Cat:All] ：全ての猫。

[Cat:#920229] , [Cat:Bat]：特定の猫。

[Cat:{*} ?] ：猫の実体を得るための問い合わせ。

[Cat:{*}@2]：２匹の猫。

③概念的関係

概念的関係は概念と同様に基本的にタイプによって分類されている。新しい関係タイプも概念タイプと同様に、例えば、

GoingTo = [Person:&lambda-1] <-(AGNT)-[Go]-(Dest)->[City:&lambda-2]

というように定義することができる。この定義は「関係GoingToは、概念[Go]の(AGNT)のある人を、(Dest)である都市に関係づける」ことを表している。この関係によって「Johanはボストンに行こうとしている」という文は、

[Person:John] -> (GoingTo) -> [City:Boston]

というCGで表現される。

このグラフは関係タイプGoingToを定義に基づいて展開することによって、

[Person:John]<-(AGNT)-[Go]-(Dest)->(City:Boston)

詳細化することができる。この展開の操作はJoinと呼ばれており、次に述べるCGの理論における４つの生成オペレータの一つである。

④生成オペレータ

CGにおける推論は４種類の生成オペレータの適用によって行われる。ここではその詳細は割愛し、概要を述べることにする。

Copy　：CGのコピーを生成する。
Restrict　：CGを、概念タイプを下位階層のものと置換するか、あるいは参照子の限定を強めることによって詳細化する。
Join　：２つのCGの同一のノードをマージし１つのCGを生成する。
Simplify　：同じ概念に複数の同一の関係が付加されている場合、一つを残して削除する。

これらのオペレータの適用を繰り返し、グラフに変形を加えていくことがCGにおける推論である。直感的には、Join操作が述語論理の単一化と考えれば推論の全体像を想像できる。CGのツール群を集めたホームページには、一定の制約のもとでこのような操作を実現したものが公開されている。

(3)　Sowaによる最上位のオントロジー

Sowaによる最上位のオントロジーを図３.１.２-２に示している。このオントロジーは①で述べた概念タイプの階層であり、ここでのSowaの関心はあらゆる概念化において基本となる概念タイプは何かということになっている。図３.１.２-２には概念タイプの最上位の分類として３つの指標が現れている。

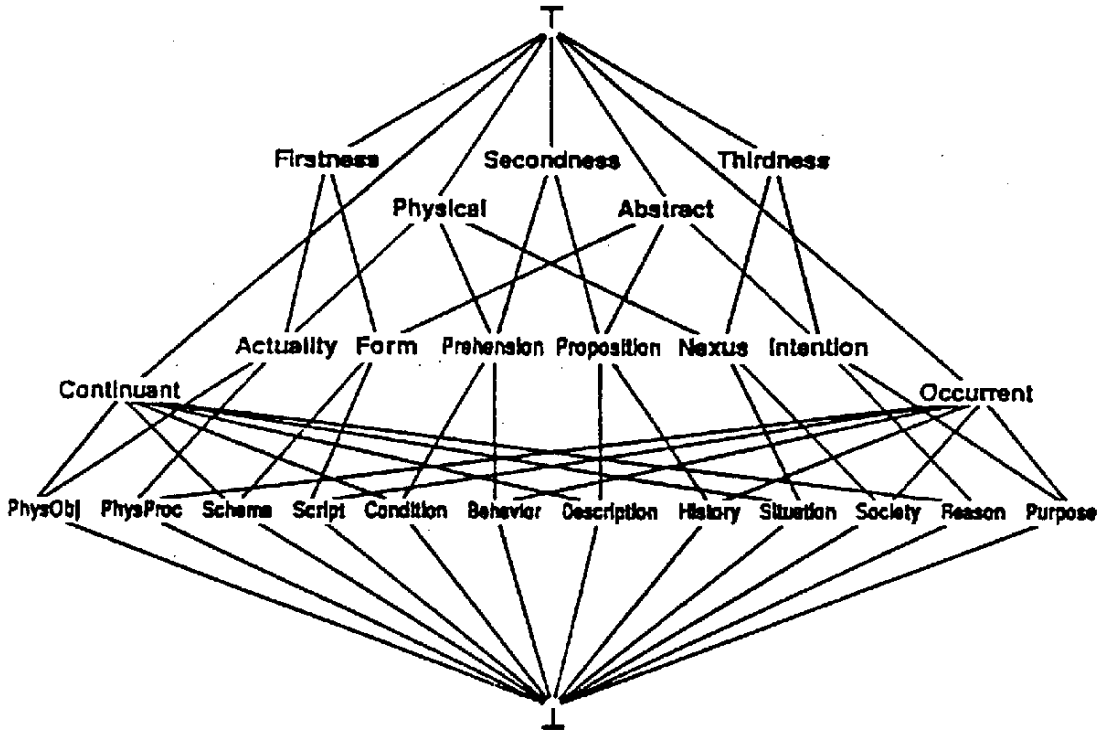


図３．１．２-２　Sowaの最上位オントロジー

最初のものはCharles Sander Peirceによるもので、Firstness、Scecondness、Thirdness（1,2,3）という３種類の概念タイプである。Peirceは徹底的に概念を分類し、概念タイプの根元的な分類は、その概念タイプを定義するために必要最小限の変項（Argument）の数であると結論した。Firstnessの概念タイプはWomanのように他の概念とは独立に定義できるものである。Secondnessの概念タイプはMother、Wifeのように、その定義にChildやHusbandといった他の概念タイプとの関係が必要とされるものである。Thirdnessは少し難解で、ある概念タイプを他の概念タイプと関連づける環境（mediating circumstance）を表す概念タイプである。例えば、Motherhood（母性）は母が子供に命を与えて育む行為を意味し、MotherをChildに関連づけるThirdness概念である。

SowaはPeirceとAlfrred North Whiteheadを初めとする数多くの哲学者、の思索の結果に目を通したうえで、残りの指標をPhysicalとAbstract(P,O)、Continuant/Occurent(C,O)の二つに定め、図３．１．２-２に示されたオントロジーを構成した。これらの３つの指標の組み合わせによって概念タイプは大きく１２に分類されオントロジーとしてより詳細な概念を体系化する基礎になる。

(4) おわりに

CGはオントロジー工学のための基礎理論として魅力的である。現在、我々が抱えているオントロジーに関する様々な疑問に対して、直接的な答えこそはないが、それについて系統的に考察する土台が与えられているように思われる。

もちろん、その内容について理論的に過度に複雑で、実用的には冗長な面を切り捨ててみたときにオントロジー工学の基礎としてとして何が残りうるかということはまだ明らかではない。KIFのような他のオントロジー記述言語と同等のものしか残らない可能性は十分にあるにせよ、CGに関するSowaの著作（一般に難解で長く本質を読みとりにくいが）には様々な疑問に対する答えが隠れているような感触があり、オントロジー工学の基礎を固める出発点として検討する価値は十分にあるように思われる。

CGに関する国際会議が毎年開催されており、エディターや推論エンジンのツール群を紹介したホームページ[Cgtools]が公開されている。しかし、Sowaのオントロジーとツールの概念レベルが一致しておらず、汎用のCG言語処理系の開発事例が先行している。ここにもCG研究の本質を捉えにくくする元凶があるように思われる。Sowaのオントロジーをベースにしたツールの開発が進み、アイデアとツールが早く一致することが望まれる。

参考文献

[Sowa 84] John F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA (1984)

[Sowa] John F. Sowa, Knowledge Representation, Logical, Philosophical, and Computational Foundation, PWS Publishing Company, Boston(in press?)

[Cgtools] http://cs.une.edu.au/~cgtools/

3.1.3 多重オントロジーとオントロジー変換

共有オントロジーを研究する場合、主に単一オントロジーによって行うものが多い。しかし、実際のシステムを構築する場合、対象は同一でも複数の概念化が行なわれていたり、逆に同一の概念に見えても、別のものを指していることも多い。そこで、武田らはアスペクトという単位を考え、アスペクトを単位としてオントロジーを構築することで、異なる概念化からの概念の共存や、異なる概念の同一化を可能とすることを試みている[Takeda95]。

(1) アスペクト

オントロジーは対象世界を記述する体系であるが、それは対象世界の概念化（con-ceptualization）の結果である。しかし、単一の概念化によって、記述の詳細度の幅をある記述や広範な対象世界の記述を行なうことは困難であり、実際的ではない。例えば図３.１.３-１は「寺」という事象を概念化しようと思った時にどのような概念

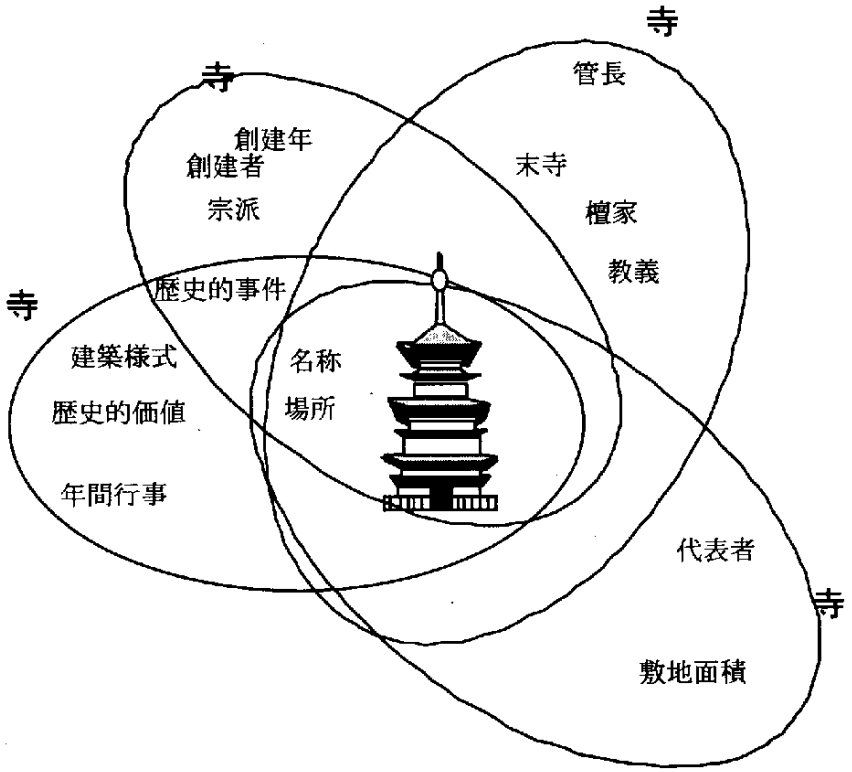


図３.１.３-１　複数の視点からの概念化

化が可能かを摸式的に示したものである。寺には宗教施設の側面もあれば、歴史的施設の側面もあり、また観光施設の側面があることもある。それぞれの場合、住所や名称のように同じ概念を共有することもあれば、敷地面積のようにある側面しか必要でない概念もある。また、同じ概念にみえても、名称といった場合のように、異なる意味をもつことがある。

実際には、我々は複数の概念化を適宜組み合わせることによって、対象世界を、広い範囲を詳細度の幅がある形で、とらえている。オントロジーの構築のおいても、この複数の概念化の利用を反映した方法をとる必要がある。本研究では、一つの概念化による概念の体系をアスペクトと呼び、オントロジーをアスペクトの組み合わせによって構成する。

ここではアスペクトを基本（atomic）アスペクトと複合（composite）アスペクトに分けて定義する。基本アスペクトは、他のアスペクトとは独立に定義されるもので、アスペクトの名前と用いられる概念の定義および概念間の関係からなる。これに対して、複合アスペクトはアスペクトの組み合わせによる新たなアスペクトの構成を行なうもので、他のアスペクトを利用して定義される。複合アスペクトはさらに2種類あり、異なる対象領域のアスペクトを合わせることによって定義する組み合わせ（combination）アスペクト、対象領域が重なり合うアスペクト間の関係によって定義されるカテゴリー（category）アスペクトからなる。前者は基本的には二つのアスペクトの和を構成するものであるに対して、後者は異なる概念化間の関係を記述している。例えば、旅行に関するオントロジー「旅行アスペクト」は「観光ホテル・アスペクト」と「行程アスペクト」などからなる組合せアスペクト、「ホテル・アスペクト」は「観光ホテル・アスペクト」、「ビジネスホテル・アスペクト」などからなるカテゴリ・アスペクトとして定義できる（図3.1.3-2参照）。

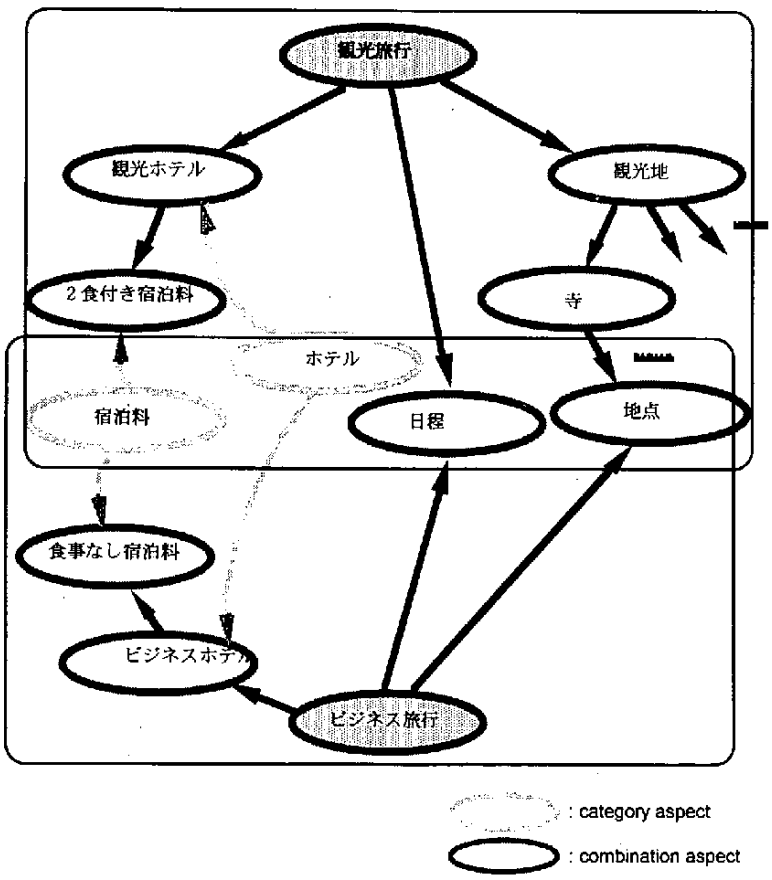


図3.1.3-2　共有可能な複数のオントロジー

(2) アスペクトの論理的定義

①アスペクトの定義

まず、他にアスペクトに依存しない基本アスペクトを定義する。

定義1 基本アスペクトAの一階述語論理における理論をT(A)とするとき、そのアスペクトの名前 aspect (A) は次の式を満たす。

$$\text{aspect}(A) \longleftrightarrow T(A) \square$$

次に、組合せアスペクトを定義する。

定義2 アスペクト$A_1, \ldots, A_n$の組合せアスペクトの理論 $T(A_{COM}(A_1, \ldots, A_n))$は次式を満たす無矛盾な理論である。

$$T(A_{COM}(A_1, \ldots, A_n)) = \text{aspect}(A_1) \wedge \ldots \wedge \text{aspect}(A_n) \wedge I(A_1, \ldots, A_n)$$

ただし、$I(A_1, \ldots, A_n)$はアスペクト$A_1, \ldots, A_n$のアスペクト間理論である。□

この場合、組み合わされるアスペクトの中に同名の述語があった場合、予期せぬ結果をもたらすことになる。ここでは同名の述語がある場合は、同一の意味を持つと仮定している。そうでないときとは、以下で示すカテゴリアスペクトで表現するべき場合だと考える。

次にカテゴリアスペクトを定義する。この場合、要素となるアスペクトの理論が常に正しいのではなく、状況に依存する。このため、様相論理を導入して形式化する。以下では様相論理S4を用い、直感的にはLは必然を、Mは可能を示す様相記号である。

定義3 アスペクト$A_1, \ldots, A_n$のカテゴリアスペクトの理論$T(A_{CAT}(A_1, \ldots, A_n))$は次式を満たす無矛盾な理論である。

$$T(A_{CAT}(A_1, \ldots, A_n)) = M\text{aspect}(A_1) \wedge \ldots \wedge M\text{aspect}(A_n) \wedge I(A_1, \ldots, A_n)$$

ここで、$I(A_1, \ldots, A_n)$はアスペクト$A_1, \ldots, A_n$のアスペクト間理論である。□

組合せアスペクトやカテゴリアスペクトの要素として、また組合せアスペクトやカテゴリアスペクトを使うことができるので、アスペクトは階層的に定義することがで

きる。いま、あるアスペクトAをA = f (A_1, ..., A_n) と表現したとき、fはA_{COM}とA_{CAT}の組合せで記述できる。

② アスペクト間の関係

アスペクト間の関係として、まず包含（inclusion）と真包含（strict inclusion）を定義する。

定義4 アスペクトAはアスペクトBに包含されるときは、aspect (B) ⊢ *M* aspect (A)が満たされるときである。□

定義5 アスペクトAはアスペクトBに真包含されるときは、aspect (B) ⊢ aspect (A)が満たされるときである。□

ここで包含関係とはアスペクト間の包含を意味しており、真包含関係とは論理的な包含関係を意味している。同様に論理式に対しても関係を定義できる。

定義6 式fはアスペクトBに包含されるときは、aspect (A) ⊢ *M*fが満たされるときである。□

定義7 式fはアスペクトBに真包含されるときは、aspect (A) ⊢ f.□

この定義の意味するところは、アスペクト理論を解釈するには二つの方法があるということである。すなわち、真包含関係とは通常の論理的解釈を意味し、包含関係とは、理論の競合をまで含めた解釈を意味している。

定理1 もしアスペクトAがアスペクトBに真包含されるならば、AはBに包含される。

もう一つの関係は、競合（compatible）関係で、二つのアスペクトが関係しあっていることを示す。□

定義8 アスペクトAとBが競合関係にある時は、以下の条件のどれかが満たされる時である。

1. AとBが同一アスペクトである。
2. AとBを要素として持つアスペクトが存在する。
3. Aの要素A_0とBの要素B_0が競合関係にある。□

定義9 式fがアスペクトAにおいて競合関係であるとは、fを含むアスペクトBが存在して、AとBが競合関係であるときである。□

競合関係は無矛盾性や、後で述べる変換可能性を意味するわけではなく、その前提としてアスペクト間に共通部分があることを指している。

③アスペクト間理論(1):アスペクトレベル関係

カテゴリアスペクトの特徴はアスペクト間理論にどうようなものがはいるかに依存する。以下ではその特徴としてコンパクトというものを定義する。

定義10 もしカテゴリアスペクトのアスペクト間理論 $I(A_1, ..., A_n)$ が以下の式を満たす時、このアスペクトをコンパクトであると呼ぶ。

$$I(A_1, ..., A_n) \vdash_L (aspect(A_1) \vee ... \vee aspect(A_n))$$

直感的には、コンパクトなカテゴリアスペクトは、その要素のアスペクト$A_1, ..., A_n$で十分であることを意味している。□

定理2 コンパクトなカテゴリアスペクトが2つの要素アスペクトA_1、A_2からなり、A_1がA_2に真包含されるとき、A_1はAに真包含される。□

実際にはこのAとA_1の関係はもっと強く、$aspect(A) \vdash_L aspect(A_1)$である。これを固定的(rigid)関係と定義する。

定義11 式fがアスペクトAにおいて固定的である時は、$aspect(A) \vdash_L f$のときである。□

もつひとつの特徴は排他的(exclusive)である。

定義12 もしカテゴリアスペクトのアスペクト間理論 $I(A_1, ..., A_n)$ が次式を満たすとき、そのアスペクトは排他的であるという。

$$I(A_1, ..., A_n) \vdash \bigwedge_{k=1, ..., n-1} \bigwedge_{l=k+1, ..., n} (aspect(A_k) \wedge aspect(A_l))$$

④アスペクト間理論(2):対象レベル関係

また、個々の述語に対してアスペクト間理論を定義することができる。これを対象

レベル関係と呼ぶ。例えば、アスペクトAの式pがアスペクトBのqを含意するとは、

$$M(\text{aspect(A)} \wedge \text{p}) \rightarrow L(\text{aspect(B)} \rightarrow \text{q})$$

と書くことができる。もっと一般的に"もし アスペクトA_1の式f_1が真ならばアスペクトA_2の式f_2が真である" とは次式のように書ける。

$$M(\text{aspect}(A_1) \wedge f_1) \rightarrow L(\text{aspect}(B_2) \rightarrow f_2)$$

もし、アスペクトAの式pとアスペクトBのqが同値ならば、次のようにかける。

$$M(\text{aspect(A)} \wedge \text{p}) \rightarrow (\text{aspect(B)} \rightarrow \text{q}) \wedge M(\text{aspect(B)} \wedge \text{q}) \rightarrow L(\text{aspec(A)} \rightarrow \text{p})$$

定理3 もし命題pがあるコンパクトなカテゴリアスペクトAの要素全てにおいて同値ならば、pはAにおいて固定的である。□

参考文献

[Takeda95] Hideaki Takeda, Kenji Iino, and Toyoaki Nishida. Agent organization and communication with multiple ontologies. International Journal of Cooperative Information Systems, 4(4): 321-337, December 1995.

3.2　記述言語・構築環境（ツール）

3.2.1　ARPAの知識共有の枠組み

ARPAの知識共有活動（Knowledge Sharing Effort）ワーキンググループでは、ソフトウエアモジュールの入出力を共通言語を用いて統一する枠組をつくり、インターオペラビリティ（相互運用性）を向上させる試みをしてきた[Neches91a][Patil92a]。共通のプロトコルと情報表現言語を用いたメッセージ交換によって相互作用するソフトウェアモジュールは、仮想知識ベースまたはエージェントと呼ばれる。仮想知識ベース間で共通言語・プロトコルを用いた知識共有が行なわれる。各仮想知識ベースは、知識ベースシステム、データベースシステム、ソフトウェアツールなどのツールにメッセージ処理をするモジュールをかぶせてインプリメンテーションの詳細を隠蔽（カプセル化）したものである[Genesereth94a]。

エージェント間のメッセージ交換は、原則としてファシリテータ（facilitator）と呼ばれる管理用エージェントを介して行なわれる。ファシリテータは、

(1) エージェントから発信されたメッセージ内容を解析して宛先を決定する。
(2) エージェントの取り扱うツール依存のメッセージ表現と共通言語による
　メッセージ表現の間の相互変換を行う。
(3) エージェントの初期化や実行のモニタリングを行う。

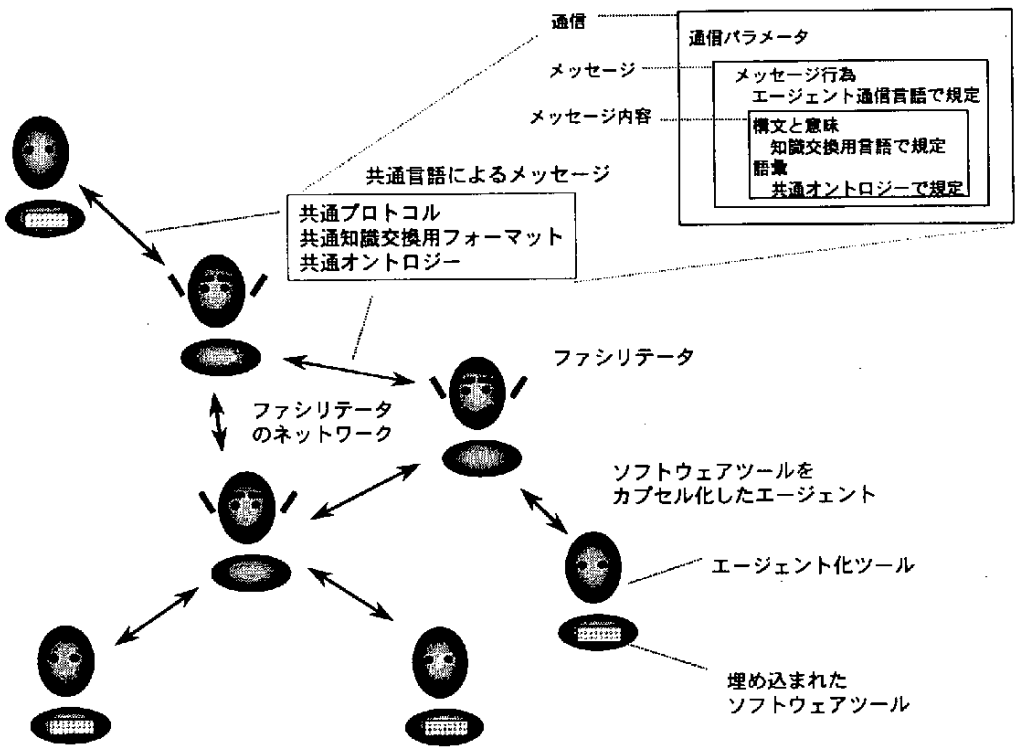


図3.2.1-1　ARPAの知識共有の枠組み

などの役割を果たす。このアーキテクチャは**連邦アーキテクチャ**（federation architecture）と呼ばれる（図3.2.1-1）。エージェント間でやり取りされるメッセージは3層に分けられる。最も外側の層は通信層であり、メッセージを実際に伝達するために必要となる低レベルの通信に関する情報が記述される。その内側の層はメッセージ層であり、メッセージによる行為を**エージェント通信言語**を用いて記述する。最も内側の層であるメッセージ内容は、メッセージにおいて参照される処理対象の記述である。構文と意味は**知識交換用言語**で規定され、語彙は**共通オントロジー**で規定されたものが用いられる。

ARPAの知識共有活動では、エージェント通信言語として、KQML（Knowledge Query and Manipulation Language）[Finin92a]を、知識交換用言語としてKIF（Knowledge Interchange Format）[Genesereth90a]を、共通オントロジー記述用言語としてOntolingua [Gruber92d]を、それぞれ提案している。KQMLは人間どうしの対話を分析した発話行為理論に基づいている。KIFは1階述語論理を拡張したものである。Ontolinguaは KL-ONE系のいくつかの知識表現言語への移植性を考慮して設計した言語で、トランスレータが開発されている。

(1) KQML

KQMLによるメッセージを理解し、然るべき応答を返すエージェントは、KQMLエージェントと呼ばれる。KQMLエージェントを外部からみると、仮想知識ベース（VKB：Virtual Knowledge Base）に基づく応答をしているとみなすことができる。

KQMLのメッセージは、

（〈メッセージ行為タイプ〉 ｛ :〈キーワード〉 〈LispのS式〉｝ ）

という形式をしている。例えば、次のKQML式では、

```
(tell   :language KIF
        :ontology motors
        :in-reply-to q1
        :content (fastens frame3 motor5))
```

メッセージ行為はtell、内容記述言語はKIF、オントロジーはmotors、内容は（fastens frame3 motor5）（「台 frame3にモーターmotor5が取り付けられている」）である。また、in-reply-toというキーワードによってこのメッセージがq1というタグのつけられた質問への回答であることが示されている。

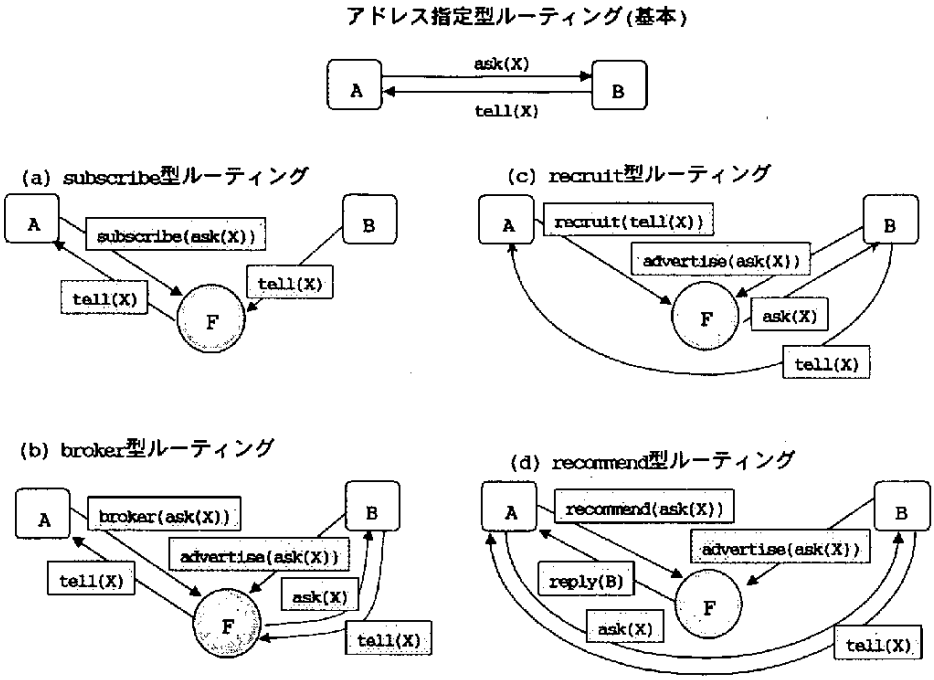
KQMLには30種類余りのタイプのメッセージ行為が定義されている（図３．２．１-２）。それらは、情報伝達（tell, deny, untell）、質問（ask-if, ask-all, stream-all, reply, sorryなど）、依頼（achieve, unachieveなど）、生成器関係（standby, ready, next, discardなど）、通知（subscribe, monitor）、ネットワーク処理（register, unregister など）、仲介処理（broker-one, recommend-one, recruit-oneなど） などに分類できる。ユーザは新しいメッセージ行為を所定の形式で定義することによって与えられたメッセージ行為集合を拡張することができる。KQMLのメッセージは、表３．２．１-１に示すように、電子メールと同様のキーワードを与えられる。

- Basic information performatives
  - tell, deny, untell,
- Database performatives
  - insert, delete, delete-one, delete-all
- Basic responses
  - error, sorry
- Basic query performatives
  - evaluate, reply, ask-if, ask-about, ask-one, ask-all, sorry
- Multi-response query performatives
  - stream-about, stream-all, eos
- Basic effector performatives
  - achieve, unachieve
- Generator performatives
  - standby,ready, next, rest, discard, generator
- Capability-definition performatives
  - advertise
- Notification performatives
  - subscribe, monitor
- Networking performatives
  - register, unregister, forward, broadcast, pipe, break, transport-address
- Facilitation performatives
  - broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

図３．２．１-２　KQMLのメッセージ行為一覧

表３．２．１-１　KQMLのメッセージのキーワード

:content	メッセージ行為の対象となっている情報
:force	Sender がそのメッセージ行為の意味を取り消すことがあるかどうか
:in-reply-to	Reply につけることを期待されるラベル
:language	:content パラメータの表現言語の名前
:ontology	:content パラメータで使われるオントロジー(ターム定義の集まり)の名前
:receiver	メッセージ行為の実際の receiver
:reply-with	Sender が reply を期待するかどうか、その場合は reply につけるべきラベル
:sender	メッセージ行為の実際の sender



A: 情報利用エージェント、B: 情報提供エージェント、F: ファシリテータ

図３．２．１-３　KQMLによるメッセージの知的ルーティング方式

内容によるメッセージ知的ルーティングに関するものは特徴的である。KQMLがサポートしている仲介のパターンを図３．２．１-３に示す。subscribeを用いた仲介では、クライアントがはじめにsubscribeメッセージを用いてメッセージ受信申し込みをファシリテータに通知しておくと、その後ファシリテータがそれにしたがってメッセージを配送する。Broker型とrecruit型のルーティングでは、仲介は、(1) 情報提供側は自分の提供できる機能を協調促進器にadvertiseする、(2) 情報利用側は自分の求めている機能を協調促進器にsubscribeする、(3) 協調促進器は両者を照合して情報提供側と情報利用側を取り持つ（matchmaking）、の３ステップで行なわれる。Recommend型のルーティングでは、情報受信側のエージェントに対して質問への回答ではなく、情報提供エージェントの名前が送付される。情報利用エージェントは、通知された名前を手がかりに情報提供エージェントに直接情報要求を行う。

streamを用いたメッセージのやりとりのパターンを図３．２．１-４と図３．２．１-５に示す。subscribeを用いたメッセージのやりとりのパターンを図３．２．１-６に示す。KQMLにおけるメッセージ交換の時系列のパターンを図３．２．１-７に示す。

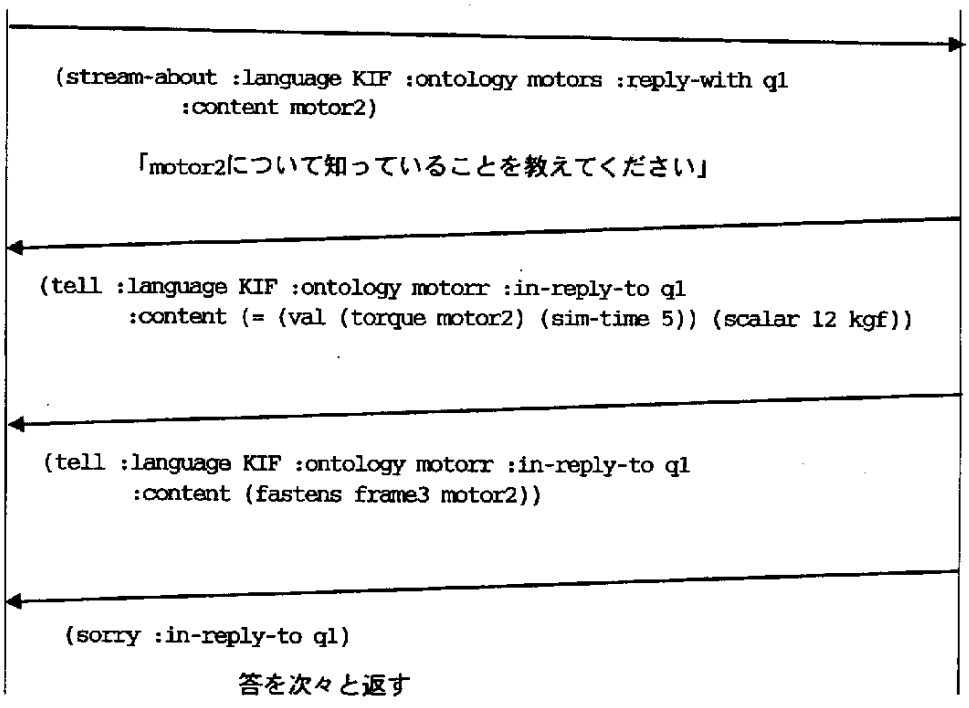


図３．２．１-４　KQML --- streamを用いたやり取りの例（１）

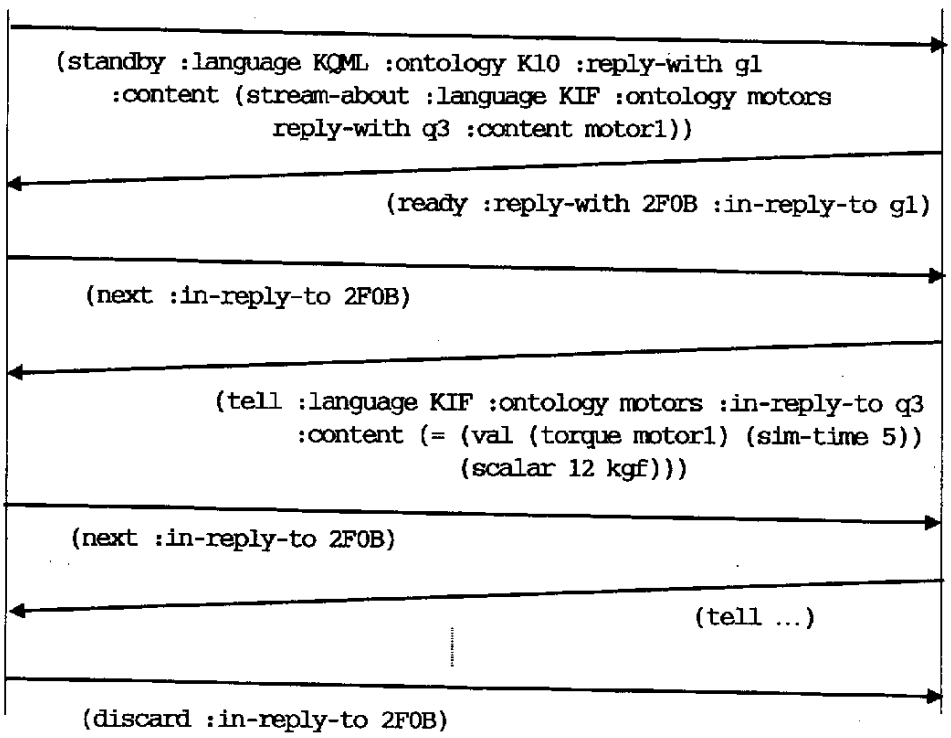


図３．２．１-５　KQML --- streamを用いたやり取りの例（２）

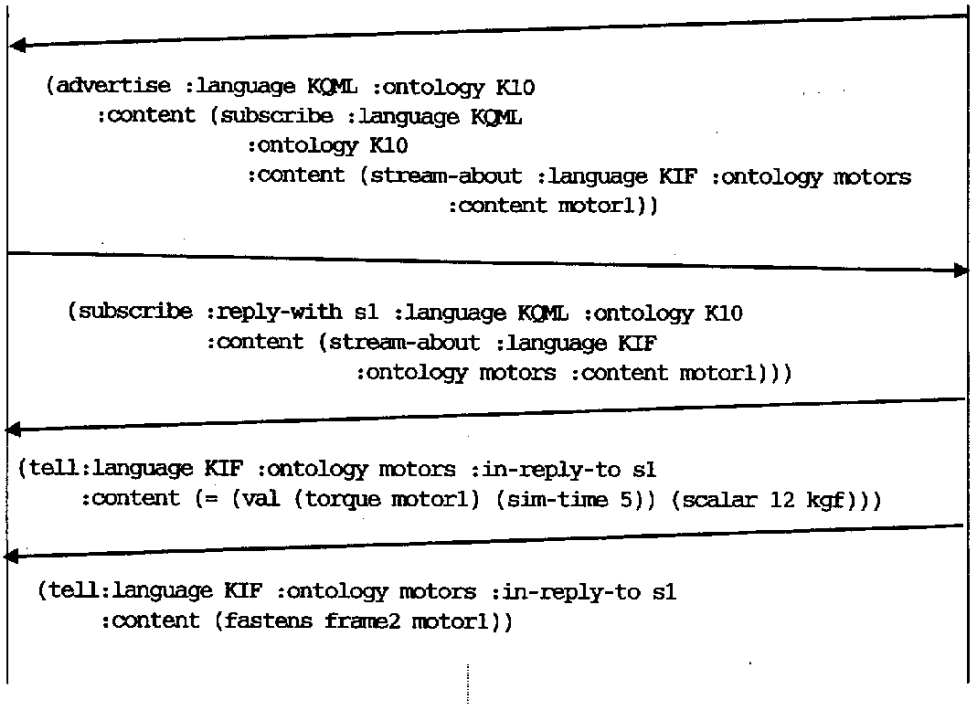


図３．２．１－６　KQML --- subscribeの使用例

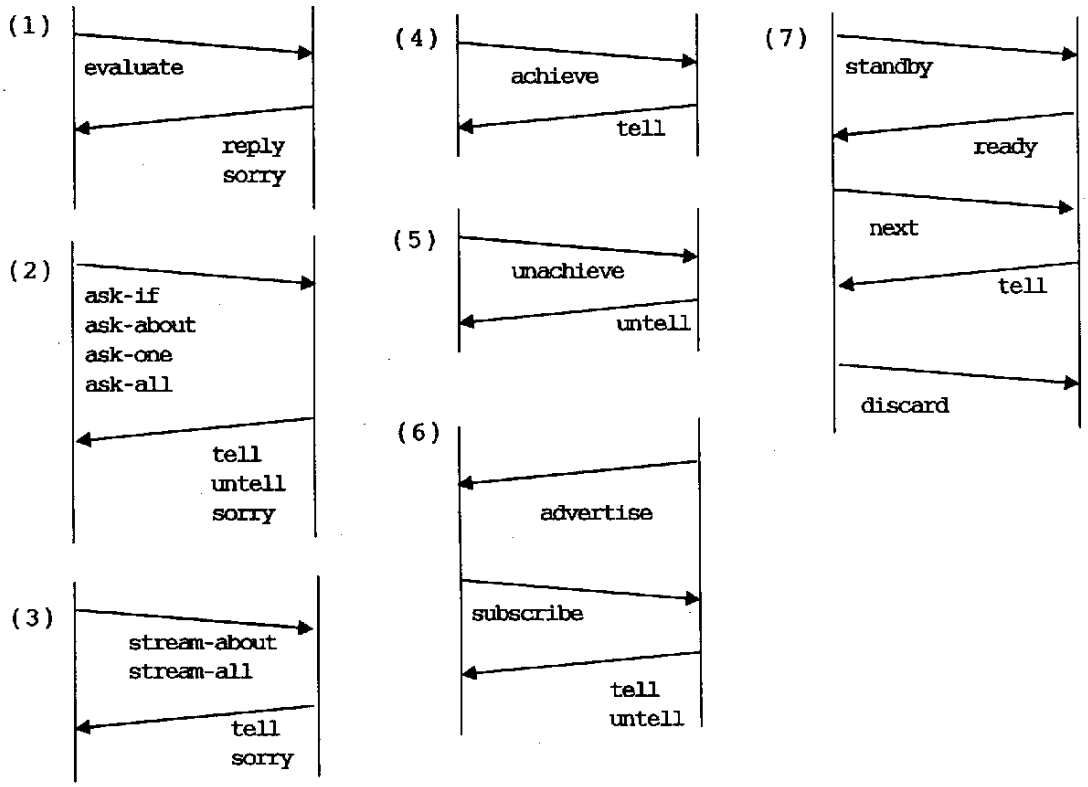


図３．２．１－７　典型的なメッセージ行為対のパターン

(2) KIF

KIF（Knowledge Interchange Format）は、1階述語論理を拡張して、項の定義、メタ知識（知識に関する知識）、集合、非単調理論などを記述できるようにした知識交換フォーマットである。KIFの表現は、表3.2.1-2のように、1階述語論理の表現をLisp風のCambridge記法で表したものである。

表3.2.1-2　KIFの表現の例

述語論理 / 数学の表記	KIF
(1) 項	
$x + y$	(+ ?x ?y)
{a, b, c, d}	(setof a b c d)
{x | wheel(x)∧ made-in-japan(x)} ;「日本製の車輪の集まり」	(setofall ?x (and (wheel ?x) (made-in-japan x)))
ιx [wheel(x) ∧ made-in-japan(x)] ;「その日本製の車輪」	(the ?x (and (wheel ?x) (made-in-japan x)))
κx [wheel(x) ∧ made-in-japan(x)] ;「…は日本製の車輪である」	(kappa ?x (and (wheel ?x) (made-in-japan x)))
λx [founded-year(vendor(x))] ;「xの製造者の設立年」	(lambda ?x (founded-year (maker ?x)))
if number(x) ∧ $x \neq 0$ then $1/x$;「x が 0 でない数のとき、$1/x$（を与える関数）」	(if (and (number ?x) (not (= ?x 0))) (/ 1 x))
(2) 命題	
p (a, b)	(p a b)
∀x ∃y[product(x)→vendor(y)∧makes(y,x)]	(forall ?x (exists ?y (=> (product ?x) (and (vendor ?y) (makes ?y ?x)))))

1階述語論理の最小限の表現に加えて、setofやlistofのように集合・リストを外延的に表現する記述子や、theやsetofallのように内包的な表現を作り出すための記述子や、lambdaやkappaのように関数や関係を作り出すための記述子や、ifやcondによる条件記述子などが用意されている。

さらに、次のような形式のdefobject、deffunction、defrelationを使ってオブジェクト、関数、関係を定義することができる。

(defobject <objconst> := <term>)

(deffunction <funconst> (<indvar>* [<seqvar>]) := <term>)

(defrelation <relconst> (<indvar>* [<seqvar>]) := <sentence>)

次のKIF式は、deffunctionを使って共通集合を計算する関数（union）を定義している。

```
(deffunction union (@sets) :=
  (if (forall (?s) (=> (item ?s (listof @sets)) (set ?s)))
    (setofall ?x (exists (?s) (and (item ?s (listof @sets))
                                   (member ?x ?s))))))
```

unionは、与えられたアイテムの集まり@setsに対して、@setsの全ての要素 ?sが集合であるとき定義される。@setsのunionは、次のような全ての要素 ?xの集合であると規定される：「?xに対して@setsの要素となっている集合 ?sが存在し、?xは ?sの要素になっている」。

```
(defrelation binop (?f ?s) :=
      (and (binary-function ?f)
           (subset (universe ?f) ?s)))
(defrelation associative (?f ?s) :=
     (forall (?x ?y ?z)
          (=> (and (member ?x ?s) (member ?y ?s) (member ?z ?s))
              (= (value ?f ?x (value ?f ?y ?z))
                 (value ?f (value ?f ?x ?y) ?z)))))
(defrelation commutative (?f ?s) :=
      (forall (?x ?y)
          (=> (and (member ?x ?s) (member ?y ?s))
                   (= (value ?f ?x ?y) (value ?f ?y ?x)))))
(defrelation invertible (?f ?o ?s) :=
      (forall (?x)
          (=> (memberp ?x ?s)
              (exists (?y)
                  (and (member ?y ?s)
                       (= (value ?x ?y) ?o) (= (value ?y ?x) ?o))))))
(defrelation distributes (?f ?g ?s) :=
       (and (binop ?f ?s) (binop ?g ?s)
            (forall (?x ?y ?z)
              (=> (and (member ?x ?s) (member ?y ?s) (member ?z ?s))
                  (= (value ?f (value ?g ?x ?y) ?z)
                     (value ?g (value ?f ?x ?z) (value ?f ?y ?z)))))))
```

図３．２．１-８　KIFによる２項関数の定義例

図３．２．１-８にKIFによる２項関数の定義の例を示す。

従来の知識表現言語では、表現力と推論能力の間のトレードオフ（許容する表現力の豊かさに伴って推論に要する計算量が増大する）が問題となったが、KIFは情報伝

達だけに目的を限定している。KIFによって豊かな情報の表現が可能であるが、エージェントはすべてのKIF表現を理解できないことが前提とされる。

(3) Ontolingua

Ontolinguaは、対象領域の概念的なモデリングとその表現のしかたに関する合意を明示的に表示するためのオントロジー記述言語である。KIFのなかの定義に関する表現を用いて、対象領域をオブジェクト、関数、関係の集まり（オントロジー）として形式化して定義することができる。オントロジーOに定義された用語の集合（語彙）を用いた記述を認識し、操作するプログラムは、Oにコミットしたプログラムと呼ばれる。

与えられたプログラムがコミットしているオントロジーには、そのプログラムが操作するオブジェクトや関係の定義が与えられているので、それを理解することによってプログラムの共有・再利用が可能になる。また、Ontolinguaにはオントロジー定義をいくつかの知識表現言語の定義に変換するトランスレーターが用意されているので、共通オントロジーを実際に自分の知識ベースシステムの定義のなかに組み込んで使用することができる。共通するオントロジーにコミットしたプログラムの間では、内容レベルで整合性が保証されたメッセージ交換を行うことができる。

Ontolinguaによるオントロジーの典型的な記述例を図3.2.1-9に示す。これらの記述のベースになっているのはフレーム形式の表現である。フレーム形式の表現によってオントロジーを記述するための語彙はフレームオントロジーと呼ばれている。

Gruberによると、オントロジー記述が適切であるためには、次のような点についての考慮を払うことが必要である。

①明確さ：オントロジーはたとえその背景が種々の便宜上の都合に由来したものであっても、形式的な表現を用いて客観的かつできるかぎり完全に記述されていなければならない。
②結束性：定義から自然に誘導される推論と整合性が取れたものでなければならない。定義からの帰結がオントロジーの別の定義に矛盾するようなことがあればそのオントロジーは結束性がないことになる。
③拡張可能性：オントロジーのユーザが与えられたオントロジーを単調に（与えられた定義を変更せずに、新しい定義を追加するだけで）拡張できるようになっていなければならない。
④表記上のバイアスが最小限であること：オントロジーは特定の表記体系（例えば、年月の表記）からできる限り独立していなければならない。
⑤オントロジーのコミットを最小限にとどめること：一つのオントロジーは特定の観

点から世界を捉え、そのための語彙を規定したものである。オントロジーの有用性・汎用性を高めるためには、その前提として設定された条件を最小限にとどめることが望ましい。

```
(define-class AUTHOR (?author)
:def (and (person ?author)
          (= (value-cardinality ?author AUTHOR.NAME) 1)
          (value-type ?author AUTHOR.NAME biblio-name)
          (>= (value-cardinality ?author AUTHOR.DOCUMENTS) 1)
          (<=> (author.name ?author ?name)
               (person.name ?author ?name))))
```

「?authorがauthorであるとは、?authorがpersonであり、単一値を取り、値のタイプがbiblio-nameであるAUTHOR.NAME属性と、1個以上の値を取るAUTHOR.DOCUMENTS属性をもち、?authorのauthor.nameとperson.nameが同一値となることを意味する。」

```
(defrelation REFERENCE
    (=> (REFERENCE ?ref)
        (and (defined (ref.document ?ref))
        (defined (ref.title ?ref)))))
```

「REFERENCE(参照)という関係について、?refがREFERENCEであるならば、?refについて(ref.document ?ref)と(ref.title ?ref)が定義されている」

```
(deffunction REF.TITLE
  (=> (and (defined (REF.TITLE ?ref))
           (= (REF.TITLE ?ref) ?title))
      (and (reference ?ref)
           (title-name ?title))))
```

「REF.TITLEという関数について、?refについて(REF.TITLE ?ref)が定義されていて、その値が?titleであれば、?refはreference、?titleはtitle-name、という述語を満足する」

図３．２．１-９　Ontolinguaによるオントロジー記述例

```
(defrelation PHYSICAL-QUANTITY
  (=> (PHYSICAL-QUANTITY ?q)
    (and (defined (quantity.magnitude ?q))
         (double-float (quantity.magnitude ?q))
         (defined (quantity.unit ?q))
         (member (quantity.unit ?q)
             (setof meter second kilogram
                 ampere kelvin mole candela))))))
(deffunction THE-QUANTITY
 (<=> (and (defined (THE-QUANTITY ?m ?u))
        (= (THE-QUANTITY ?m ?u) ?q))
   (and (physical-quantity ?q)
      (= (quantity.magnitude ?q) ?m)
      (= (quantity.unit ?q) ?u))))
```

コーディングバイアス

特定の単位系への過度のコミット
拡張可能性の低さ

図３．２．１-１０　不適切なオントロジー記述の例

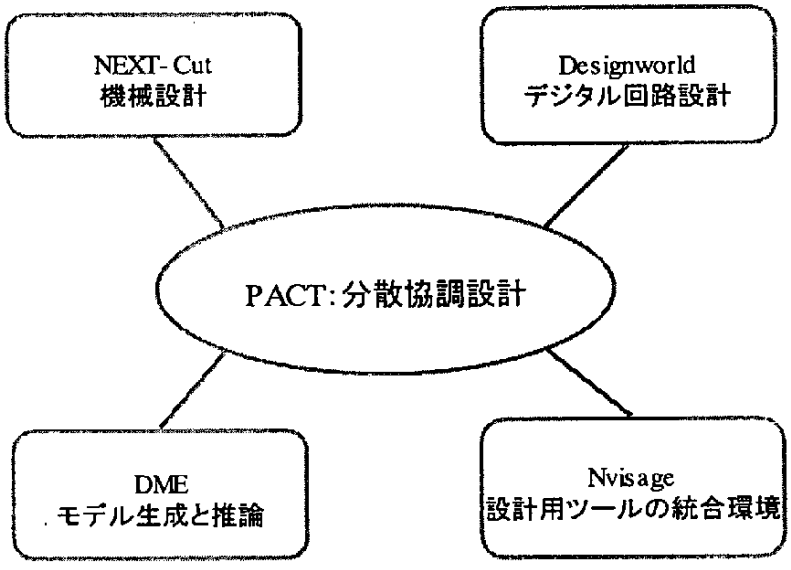


図３．２．１-１１　PACT、知識共有技術のコンカレントエンジニアリングへの適用

例えば、図３．２．１-１０の例は物理量の概念を表現するための概念定義を示しているが、ここには物理量の大きさを倍精度浮動小数点数（double-float）として表現することを前提としていたり、どの単位系を使用するかが物理量の定義の中に予め盛り込まれたりしている、など上に述べた条件に反する記述が含まれている。

このアプローチの適用例としてPACT（Palo Alto Collaborative Testbed）プロジェクト

[Cutkosky93]がよく知られている。これは、アメリカのベイエリアにあるスタンフォード大学、ロッキード人工知能研究センターなど異なる研究機関にまたがるNEXT-Cut（機械設計）、Designworld（デジタル回路設計）、DME（モデル生成とモデルベース推論）、Nvisage（設計用ツールの統合環境）の４つの異なるシステムを接続してロボットマニピュレータを分散協調設計するコンカレントエンジニアリングのテストベッドである（図３.２.１-１１）。また、デジタル図書館（図３.２.１-１２）や電子商取引（図３.２.１-１３）の枠組みとして適用する試みなども行われている。

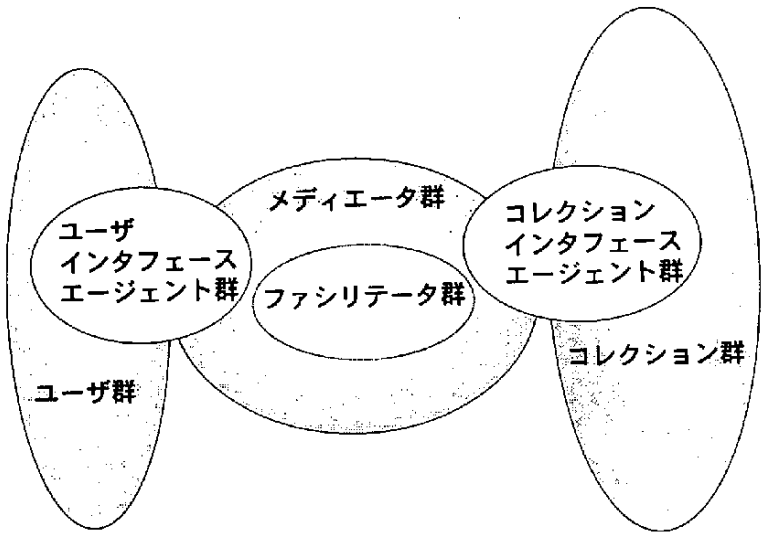
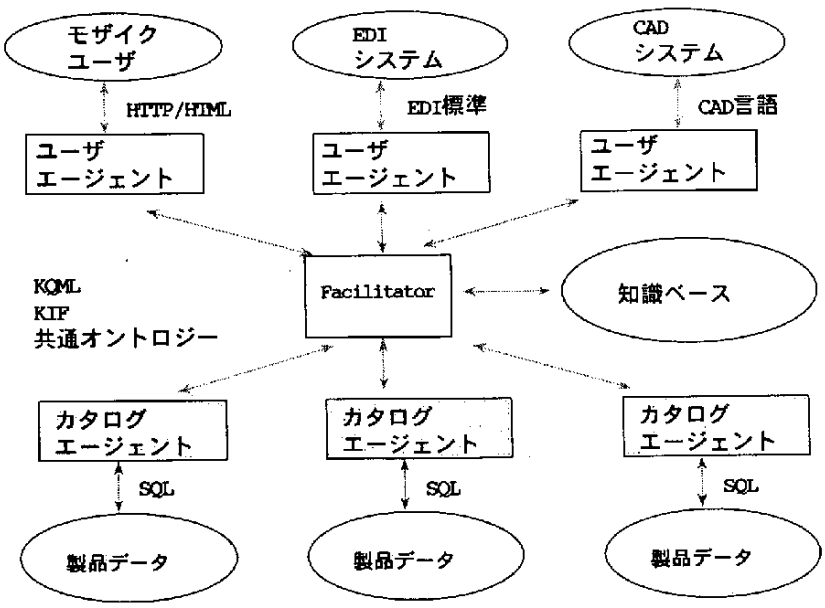


図３.２.１-１２　ミシガン大学におけるデジタル図書館への適用



スマートカタログとブローカー機構

図３.２.１-１３　電子商取引における仲介機構の適用例

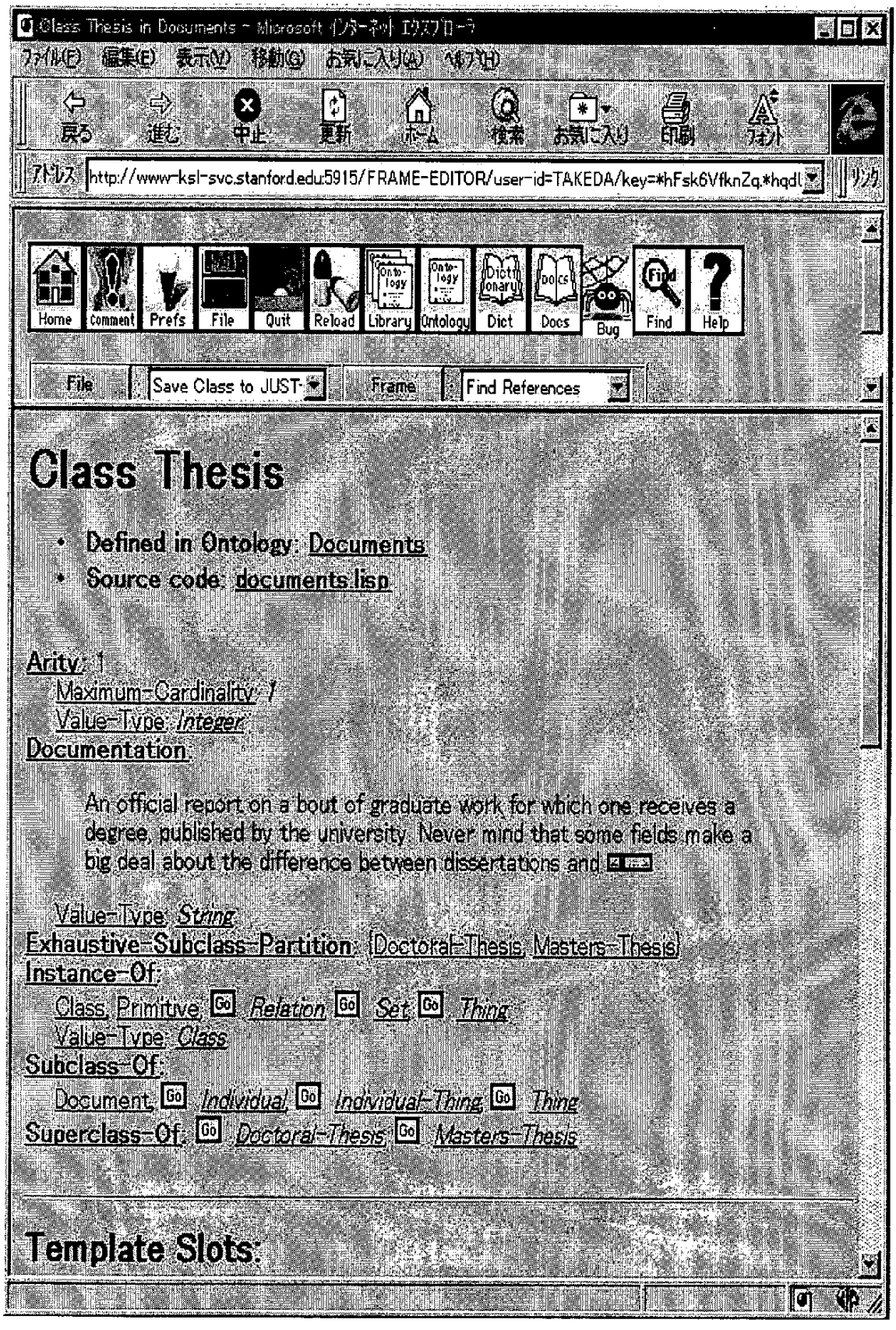


図３．２．１－１４　Ontlogy Editorの表示画面

さらに、こうしたオントロジーを共有する試みも行われている。Stanford大学のKSL（Knowledge Systems Laboratory）ではオントロジーをネットワークで共同authoring・共有を図るシステムOntology Editorを稼動させている[Fikes97] [Farquhar96]。このOntology EditorはWWWのブラウザーから利用可能なシステムで、Ontolinguaで書かれたオントロジーの閲覧と作成ができる。ユーザはこのシステムに入り、オントロジーを作成し、保存・あるいは公開することができる（図３.２.１-１４）。作成する際にすでに公開されたオントロジー・ライブラリを拡張する形で利用することもできるし、必要なクラスだけを利用してつくることもできる。オントロジーの編集の際には、クラスや関係、関数といった単位を表示・編集ができ、関連項目にはhyperlinkで移動することができる。また、Ontolinguaの定義をKIF、Epikit、Loomに変化することができる。システムの内部動作としてはオントロジー間の関係とシンボルの一貫性の維持が行われている。オントロジー間の関係としては、包含inclusion、多様化polymorphic refinement、制約restrictionなどがある。１番目はあるオントロジーが別のオントロジー全体を利用する関係、２番目は別のオントロジーで定義されている定義の一部を拡張・変更する関係、３番目は別のオントロジーで定義された定義の一部をより強めて使う関係である。シンボルの一貫性とはあるオントロジーで利用されるシンボル集合を適切に維持することで、オントロジー間の包含関係、シンボルのrenaming、シンボルの隠蔽shadowingなどの情報を用いて計算される。

参考文献

[Cutkosky93] Mark R. Cutkosky, Robert S. Engelmore, Richard E. Fikes, Michael R. Genesereth, Thoas R. Gruber, William S. Mark, Jay M. Tenenbaum, and Jay C. Weber. PACT: An experiment in integrating concurrent engineering systems. IEEE Computer, January 1993:28-38, 1993.

[Farquhar96] A. Farquhar, R. Fikes, & J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Knowledge Systems Laboratory, KSL-96-26, September 1996.

[Fikes97] R. Fikes, A. Farquhar, & J. Rice. Tools for Assembling Modular Ontologies in Ontolingua. Knowledge Systems Laboratory, KSL-97-03, April 1997.

[Finin92a] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, P. Pelavin, S. Shapiro, and C. Beck. Specification of the KQML agent communication language. Technical Report EIT TR 92-04, Enterprise Integration Technologies, 1992. (Updated July 1993).

[Genesereth90a] Michael R. Genesereth and Richard Fikes. Knowledge Interchange Format version 3.0 reference manual. Technical Report Logic-90-4, Computer Science Department, Stanford University, 1990.

[Genesereth94a] Michael R. Genesereth and Steven P. Ketchpel. Software agents. Communications of ACM, 37(7):48-53, 1994.

[Gruber92d] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992.

[Neches91a] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout. Enabling technology for knowledge sharing. AI Magazine, 12(3):36-56, 1991.

[Patil92a] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, and R. Neches. The DARPA knowledge sharing effort: Progress report. In Charles Rich, Bernhard Nebel, and William Swartout, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference. Morgan Kaufmann, 1992.

3.2.2 エージェントのコミュニケーションにおけるオントロジーの利用

武田らはエージェント間のコミュニケーションにおいてオントロジーを積極的に使う方法を提案している[Nishida94][Takeda95]。ひとつはオントロジーをメッセージの配送先決定に用いる方法で、エージェントとオントロジーのマッピングを用いて行っている。二つ目は、メッセージの変換で異なるオントロジーのエージェントにおいてオントロジー間の関係を用いて、メッセージを変換する。

(1) オントロジーを用いたメッセージの仲介

① アスペクト記述言語

3.1.3項では論理的性質について述べたが、ここでは実際にアスペクトが記述可能な言語ASPECTOL を定義する。ASPECTOL はOntolingua [Gruber91] の拡張として定義される（図3.2.2-1参照）。基本アスペクトはアスペクト理論に相当するOntoligua の定義文によって定義される。組み合わせアスペクトは利用するアスペクトの宣言をさらに含む（図3.2.2-2(b) 参照）。カテゴリー・アスペクトは利用するアスペクトの宣言とアスペクトに含まれる概念間の関係が論理式として定義される（図3.2.2-2(a) 参照）。

② アスペクト間の変換

知識コミュニティ[Nishida94] にようなエージェントに基づく分散協調システムでは、エージェントによって必ずしも同一のアスペクトに基づいているとは限らず、この場合異なるアスペクト間で知識（ここではエージェント間のメッセージ）を変換する必

基本アスペクト

(**define-aspect** <アスペクト名>
Ontolingua definitions are here
...
)

組み合わせアスペクト

(**define-aspect** <アスペクト名>
(**:include** <*要素アスペクト名*>*)
(**:rename** (<*要素アスペクト名*>*!*<*述語名*> . <*述語名*>)
*)
Ontolingua definitions are here
...
)

カテゴリ・アスペクト

(**define-category-aspect** <アスペクト名>
(**:include** <*要素アスペクト名*>*)
(**:category-type** <カテゴリ・タイプ>)
(**:rename** (<*要素アスペクト名*>*!*<*述語名*> . <*述語名*>)*)
(**:default-aspect** <*デフォルト・アスペクト名*>)
(**:translation**
(=> <*変換前概念記述*> <変換後概念記述>)
([**:query-precedence** <*質問文変換時条件*>]
[**:inform-precedence** <*伝達文変換時条件*]
(-> <*変換前概念付帯記述*> <*変換後概念付帯記述*>))+)*
)

太字は予約語、[]は省略可能、＋は一回以上の繰り返し、
＊は0回以上の繰り返しを示す。

図3.2.2-1　ASPECTOLの文法

要がある。 競合するアスペクトにおいては、一方のアスペクトに基づく知識を他方に変換することが可能な場合がある。可能である場合とは、知識に含まれているアスペクトに関連するカテゴリー・アスペクトに適用可能な論理式が含まれている時である。

ASPECTOLでは以下のようアルゴリズムで変換を行なうことができる。なお、ここでは単純文の連言のみについての変換を対象としている。

[1] KIFによるメッセージ文から次の手順によりメッセージ・データベースを作成する。
(a) 関数項を全て述語に置き換える。
(b) 引数項の概念定義述語をもし含まれていなければ、オントロジーを参照して追加する。

[2] オントロジーを参照して、変換記述のリストからデータベースと以下の意味でマッチする変換規則を求める。

(a) 質問文の場合：

i.データベース中の概念定義述語または述語の上位概念の概念定義述語が変換規則の変換後概念記述に一致する。

(b) 情報伝達文の場合：

i.データベース中の概念定義述語または述語の上位概念の概念定義述語が変換規則の変換前概念記述に一致する。

ii.データベース中の述語が変換規則の変換前概念付帯記述に一致する。このとき、文置換対（規則を適応した時にデータベースから除かれる述語と足される述語の対）、項置換対（文置換に伴って入れ換えるべき変数の対）を求める。

[3] 優先順位に従って、データベースに対して文置換対と項置換対を適用する。

[4] データベース中の文について、目的のアスペクトに適合しているかどうか調べる。存在しない述語は削除。

[5] メッセージを送る。情報伝達文ならば終り。

[6] （質問文の場合）応答を受け取り、変数への束縛リストを得る。もし、有用な応答でなければ、[2] へ行き、別の変換記述を用いる。代替変換記述がなければ、もとの質問に対してsorry を返す。

```
(define-category-aspect FEE
  (:use fee/a fee/b)
  (:category-type nil)
  (define-translation FEE
      (=> (fee/A!fee ?fee) (fee/B!fee ?fee))
    ((:query-precedence nil
      :inform-precedence nil
      (-> (fee-value ?fee ?value)
          (and (adult ?fee ?fee1)
               (student ?fee ?fee2)
               (fee-value ?fee1 ?value)
               (fee-value ?fee2 ?value))))))
    ...
 )
```

(a) Category Aspect **fee**

```
(define-aspect temple/A (TEMPLE)
  (:use fee/A)
  (define-class temple (?x)
    :def (and (has-one ?x name)
              (has-one ?x fee)))
  (define-function name (?x) :-> ?n
    :def (and (temple ?x) (string ?n)))
  (define-function temple-fee (?x) :-> ?f
    :def (and (temple ?x) (fee ?f))))
```

(b) Combination Aspect **temple/A**

図３.２.２-２　アスペクト定義の例

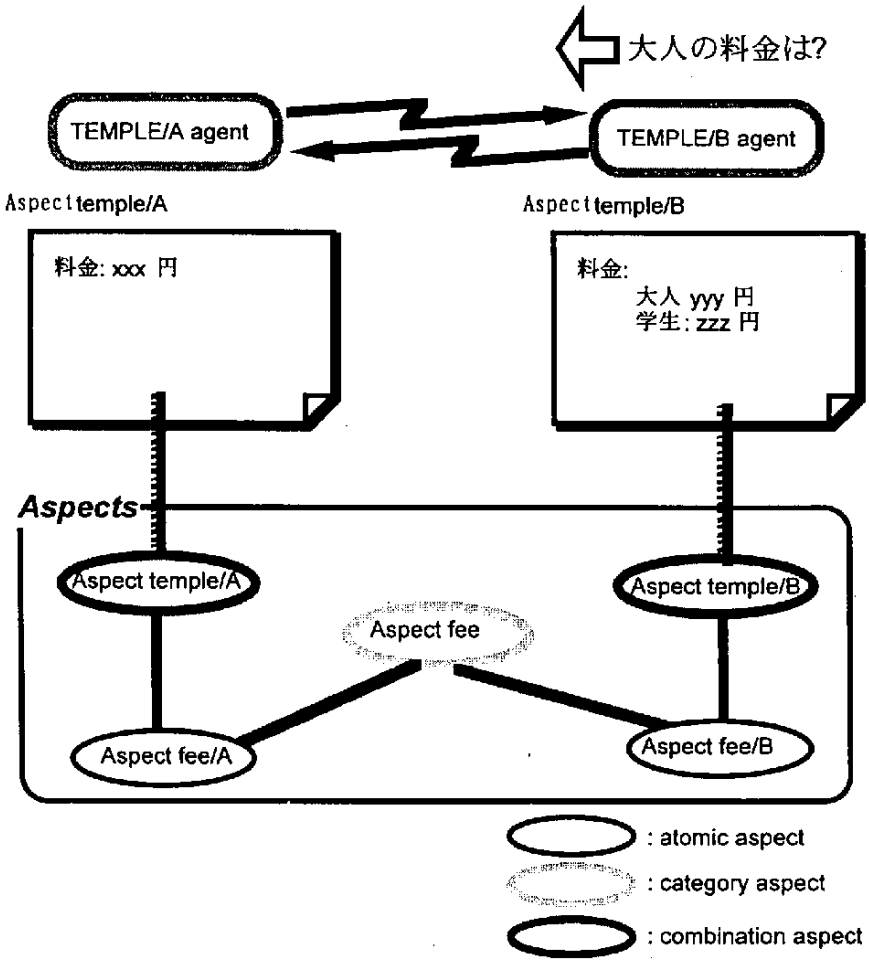


図3.2.2-3　エージェント間コミュニケーションの例

[7] 応答メッセージを元の質問メッセージの応答になるよう、[2] の項置換対、文置換対を利用してメッセージの述語や変数を復元し、質問したエージェントに返答する。

例えば、図3.2.2-3に示すような、料金の概念化が異なる二つの「寺」エージェント間のコミュニケーションを考える。今、料金区分という概念があるエージェントが料金区分のないエージェントに質問をした場合、図3.2.2-4に示すような手順により、料金区分のないメッセージに直される。返答はこの逆の変換が行なわれる。

③ 知識コミュニティKC0における実装

知識コミュニティKC0ではアスペクト間の変換機構を導入することで、メッセージ仲介機構 [Takeda94]をより効果的に利用することができる。仲介とアスペクト変換を組み合わせた場合のメッセージのやり取りを以下に示す。ここでは、前項で述べた変換機構はtranslatorというエージェントによって行なわれる。また、オントロジーはアスペクトを単位としてontology serverと呼ばれるエージェントで管理され、概念の定義、そのアスペクト、それを用いるエージェントなどの情報を提供する。

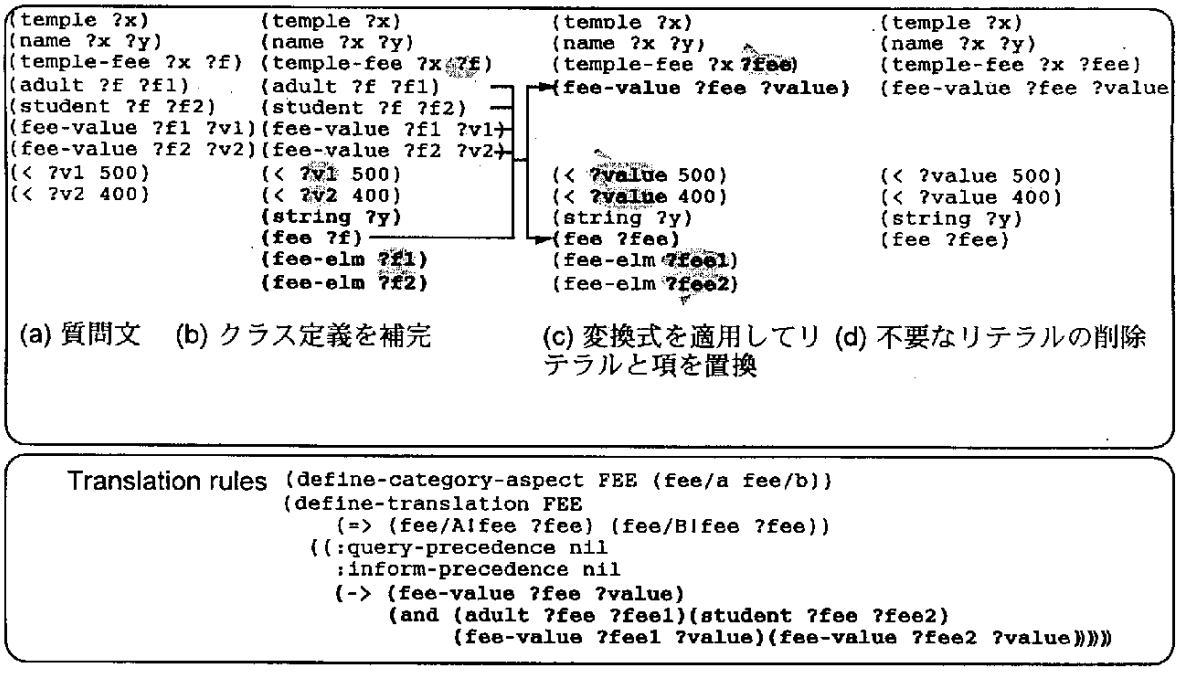


図３．２．２-４　変換例

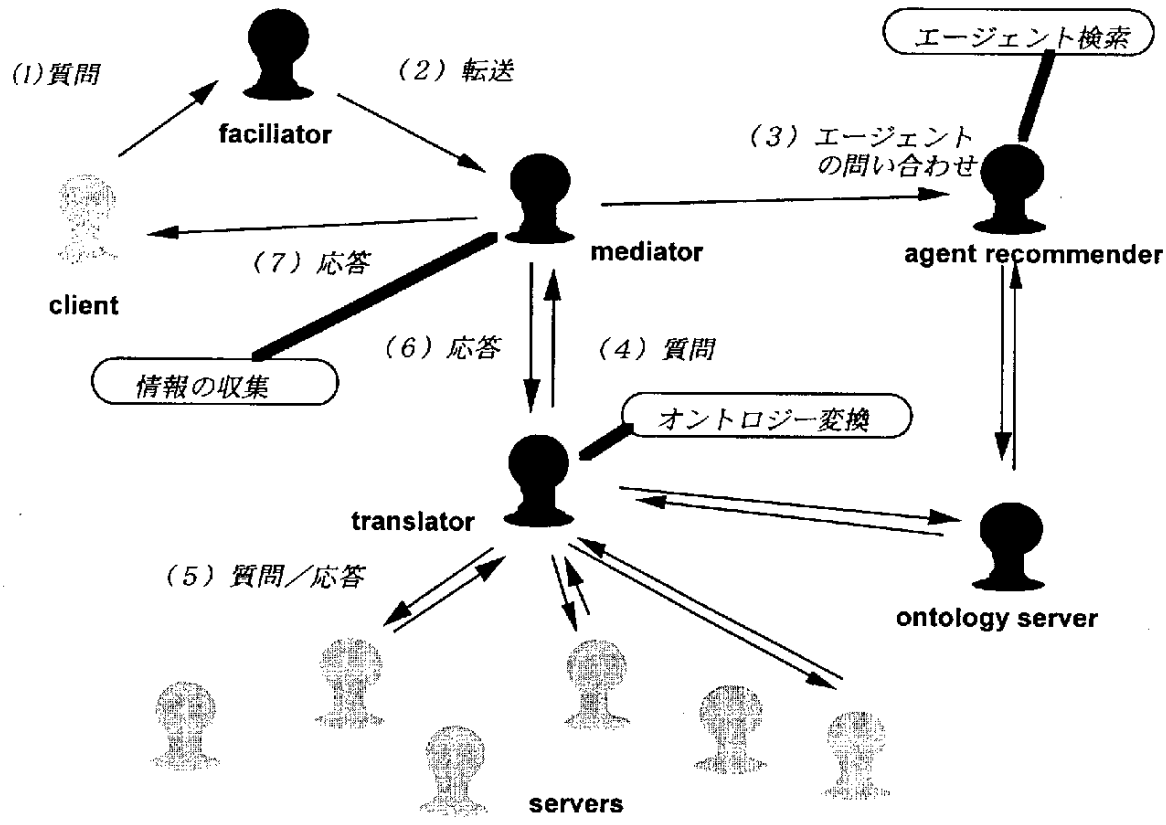


図３．２．２-５　質問の場合の仲介の流れ

1. 宛先の指定されていないメッセージはfacilitatorによってmediatorへと送られる（図3.2.2-5 (1), (2)）。
2. mediatorはagent recommenderへそのメッセージに答えられると期待さるエージェントのリストを質問し、候補となるエージェントのリストを得る（図3.2.2-5 (3)）。
3. mediatorは得られた候補に対して順時メッセージを送信する。この際に、オントロジーの異なるエージェントへ質問する場合は、translatorを介して質問を送る（図3.2.2-5 (4), (5)）。
4. translatorはontology serverに問い合わせることで、メッセージの属するアスペクトやカテゴリ・アスペクトにある変換規則を得る。これを適用することでメッセージを変換する。
5. mediatorは応答のうちで有意なものの中から必要なものを集め、もとの質問者へ返答する（図3.2.2-55 (6), (7)）。

④結論と展望

本研究では柔軟なオントロジー構築を目指して、アスペクトという単位によるオントロジーの構築とその利用について述べた。アスペクトによるオントロジー構築は(i)範囲を明示的に限定したオントロジー構築による構築の容易性と信頼性の向上、(ii)構成的にオントロジーが構築可能、(iii)例えば概略的定義と詳細定義の混在など、概念の多重性を許したオントロジーも可能、などの利点がある。しかし、今回の研究では、(i)アスペクト変換記述および変換方法が必ずしも一般的ではない、(ii)変換記述の作成が容易ではない、などの問題がある。今後はより一般的な変換記述やその自動獲得などを研究していく必要がある。

参考文献

[Gruber91] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory, 1992.

[Nishida94] Toyoaki Nishida and Hideaki Takeda. Towards the knowledgeable community. In Kazuhiro Fuchi and Toshio Yokoi, editors, Knowledge Building and Knowledge Sharing, pages 155-164. Ohmsha, IOS Press, 1994.

[Takeda95] Hideaki Takeda, Kenji Iino, and Toyoaki Nishida. Agent organization and communication with multiple ontologies. International Journal of Cooperative Information Systems, 4(4):321-337, December 1995.

[Takeda94] 武田英明, 飯野健二, 西田豊明. 知識コミュニティkc0における知識共有メカニズム. 人工知能学会全国大会(第8回)論文集, pp. 279-282, 1994.

3.2.3 DODDLE

領域オントロジーの構築は、ユーザ（専門家）に委ねられる部分が大きいために開発コストは大きく、領域オントロジー構築支援環境の整備が望まれている。領域オントロジー構築支援環境については、類似した領域オントロジーを再利用・修正していくアプローチ[G.Heijst 95]と自然言語理解の分野で開発済みの計算機可読型辞書MRD（a Machine Readable Dictionary）を再利用・修正していくアプローチ[W.Swartout 96]に大きく分れる。

ここでは、後者のアプローチに含まれる、静岡大学情報学部で開発中のシステムDODDLE（A Domain Ontology rapiD DeveLopment Environment）[T.Yamaguchi 97]について紹介する。

(1) 基本設計

MRDで提供されている概念階層構造は、多数の概念をカバーしており、領域オントロジー概念階層構造を構築するための基盤として利用可能であるが、言語処理の観点から構築された概念階層構造であることから、領域世界を表す概念階層構造と比較すれば、種々の差異があると推定される。この差異は、コンテキスト（主に領域固有性）に依存して概念の意味が変化する Concept Drift（概念変動）と呼ばれるものであり、それを管理する機構が課題となる。

従って、MRDを利用して領域世界の概念階層構造を構築するには、MRDから領域概念に関連する情報を抽出し、その関連情報を分析して、概念変動が発生している箇所を同定し、ユーザが概念階層構造を修正支援するようなツールが考えられる。

以下、「MRDからの関連情報抽出工程」と「領域固有性を考慮した修正工程」に基づく領域オントロジー構築支援環境 DODDLEのシステム構成について述べる。

(2) システム構成

図３．２．３－１にDODDLEのシステムフローを示す。MRDとしてはWordNet[G.A.Miller 95]を用いる。DODDLEの入力は、構築したい領域概念階層のノードとなる領域概念のリストであり、DODDLEの出力は、領域概念階層である。

「MRDからの関連情報抽出工程」では、まず、領域語彙とMRDとの字面レベルでの照合を行い、多義語に対しては、ユーザがWordNetからの情報（テキスト形式の意味記述やルートからその概念までのパス）をみて考え、領域世界から考えて最も妥当な意味を選択する（選択された概念をベストマッチと呼ぶ）。それらベストマッチ群とWordNetのルートまでのパス群を抽出し、ベストマッチ階層とする。次に、ベストマッ

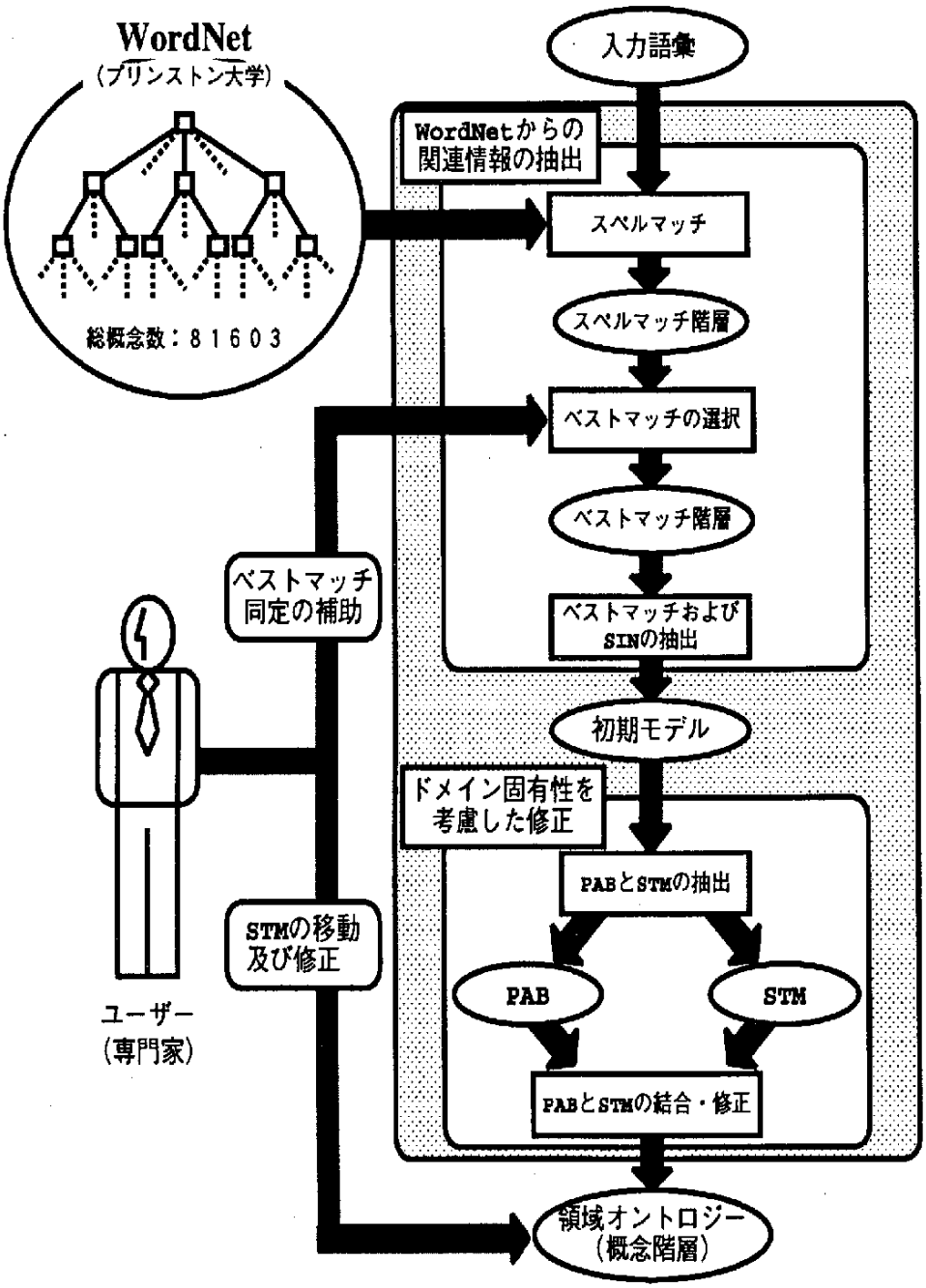


図3.2.3-1　DODDLEのシステムフロー

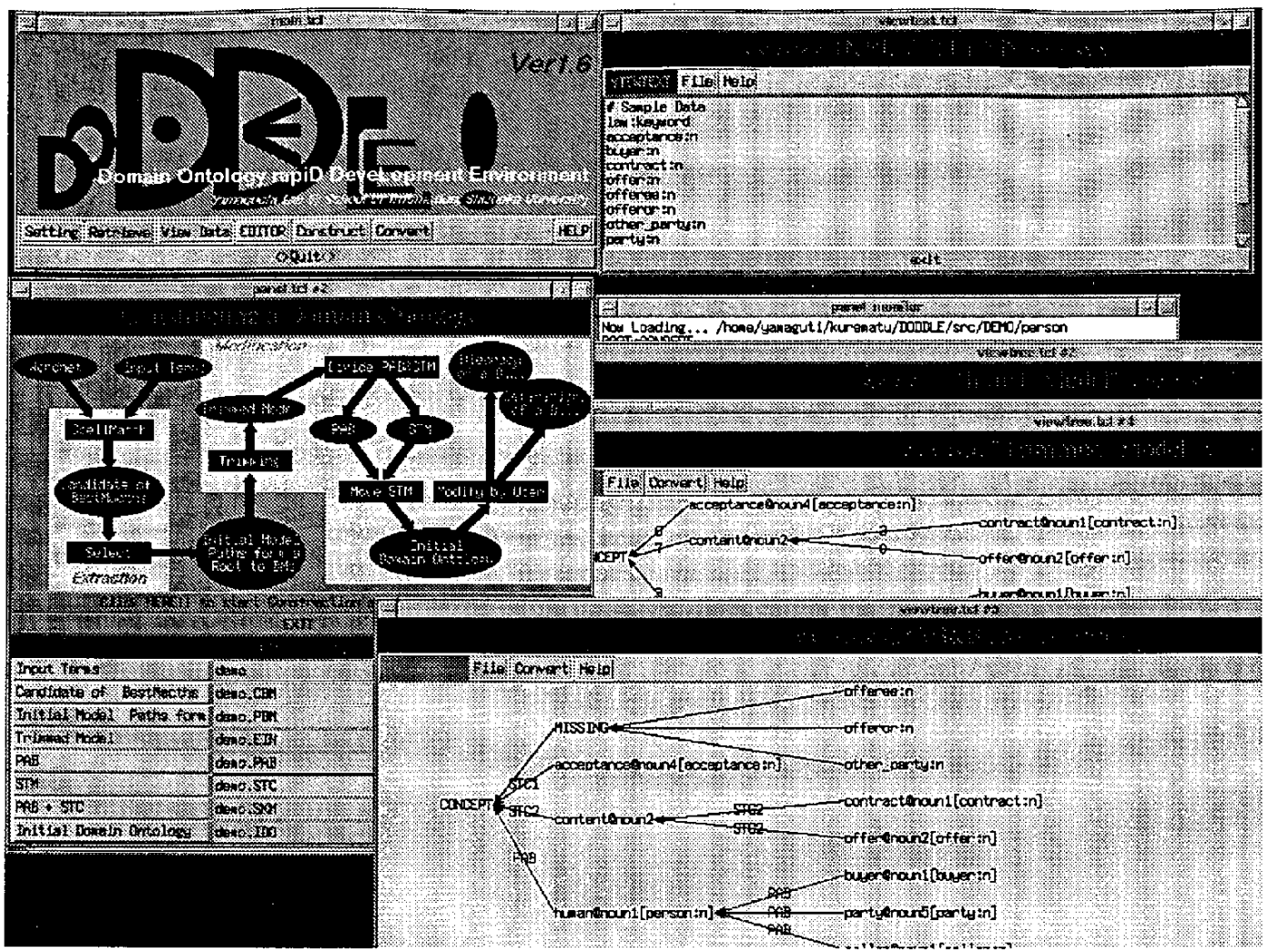


図３．２．３－２　DODDLEの実行画面

チ階層から不用な中間概念群を削除して、初期モデルを作成する。ベストマッチ階層は、ベストマッチ群とルートからベストマッチまでを連結する為の中間概念群を含むが、中間概念には、ベストマッチ間の位相関係（祖先・親子・兄弟関係）を保持することに貢献している中間概念（SIN：Salient Internal Nodes）とそうでない中間概念が存在する。ここでは、ベストマッチ階層から、SIN以外の中間概念群を除去し、ベストマッチとSINだけから構成される（当然、ルートは含まれる）初期モデルを作成する。
「領域固有性を考慮した修正工程」では、初期モデルにおいて再利用可能な領域と、概念変動が発生していると推定されるため修正が必要な領域に分割する。ベストマッチは、問題領域から考えてほぼ妥当と考えられた概念であり、それらが連続するパスは、妥当な概念が集中しているため、再利用可能領域とみなせる。この再利用可能領域をPAB（PAths including only Bestmatches）と呼ぶ。一方、ベストマッチ以外の概念が含まれる領域は、その概念により概念構造の差異（概念変動）が生じている可能性があるため、修正候補領域とみなせる。この領域をSTM（SubTrees manually Moved）とよぶ。ユーザ（専門家）はSTMを移動する事で初期モデルから領域概念階層を構築する。ここで、STMの移動先についてはユーザが決定し、移動する必要がないと判断した場合は移動しない。最終的に、STMの移動のみでは対処できない箇所についてはノード単位でユーザが修正を行い、領域概念階層を構築する。さらに、スペルマッチに失敗した語句がある場合には、それらもユーザが付加する。
現在、DODDLEはPerlとTcl-Tkを用いて、EWSのUnixプラットフォーム上、およびノートパソコンのLinuxプラットフォーム上で実装されている。図３．２．３-２にDODDLEの典型的な実行画面を示す。

（３） 実験と評価

輸送、ソフトウェアプロセス、法律の３つの領域において、DODDLEの評価が進められているが、ここでは、法律分野における評価を示す。具体的な法律としては国際売買法（以下CISGと略記） [曽野 93]を選び、CISGの第２部に含まれる４６個の法律語句をDODDLEに与え、法律概念階層木の構築を試みた。図３．２．３-３に、WordNetからの関連情報抽出後に生成された初期モデルを示す。初期モデルのなかで、矩形ノードはベストマッチノードであり、楕円ノードはSINノードである。一方、図３．２．３-４は、最終的に得られた概念階層木であり、矩形ノードはユーザによって修正が加えられたノードである。表３．２．３-１に評価実験に関連する諸データを示す。結果的に、PABとSTM以外の修正も多々加えられたが、PABとSTMに限定した形で支援率を計算すると、支援率は (1-(X+Y)/(A+B)) * 100 (%)で求められ７６％となる（記号は表３．２．３-１参照）。

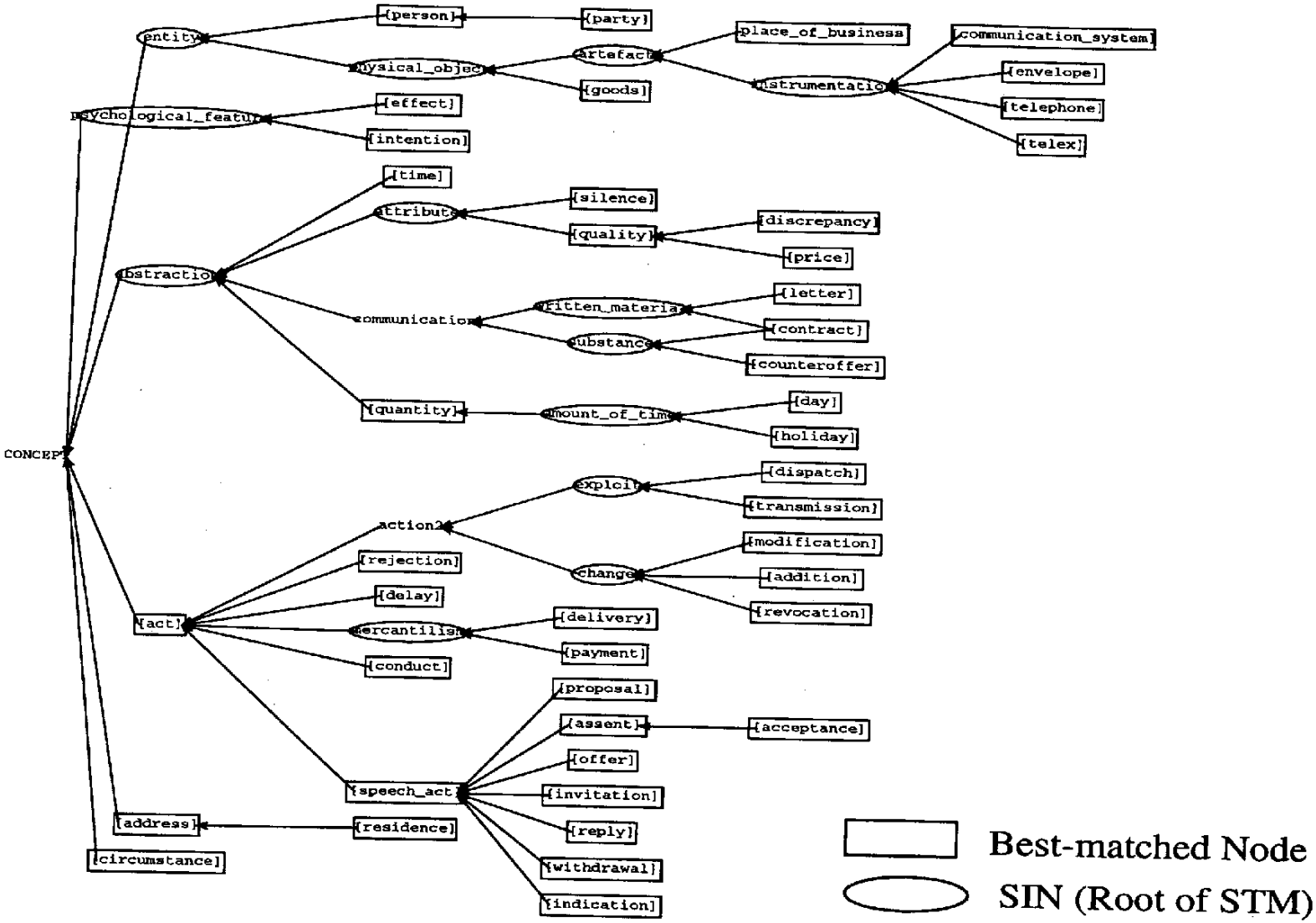

{person}
{party}
entity
place_of_business
communication_system
artefact
{envelope}
{telephone}
{telex}
{goods}
{effect}
{intention}
{time}
{silence}
{discrepancy}
{price}
{letter}
{contract}
{counteroffer}
{day}
{holiday}
{dispatch}
{transmission}
{modification}
{addition}
{revocation}
{rejection}
{delay}
{delivery}
{payment}
{conduct}
{proposal}
{assent}
{acceptance}
{offer}
{invitation}
{reply}
{withdrawal}
{indication}
{address}
{residence}
{circumstance}
Best-matched Node
SIN (Root of STM)

図３．２．３-３　初期モデル

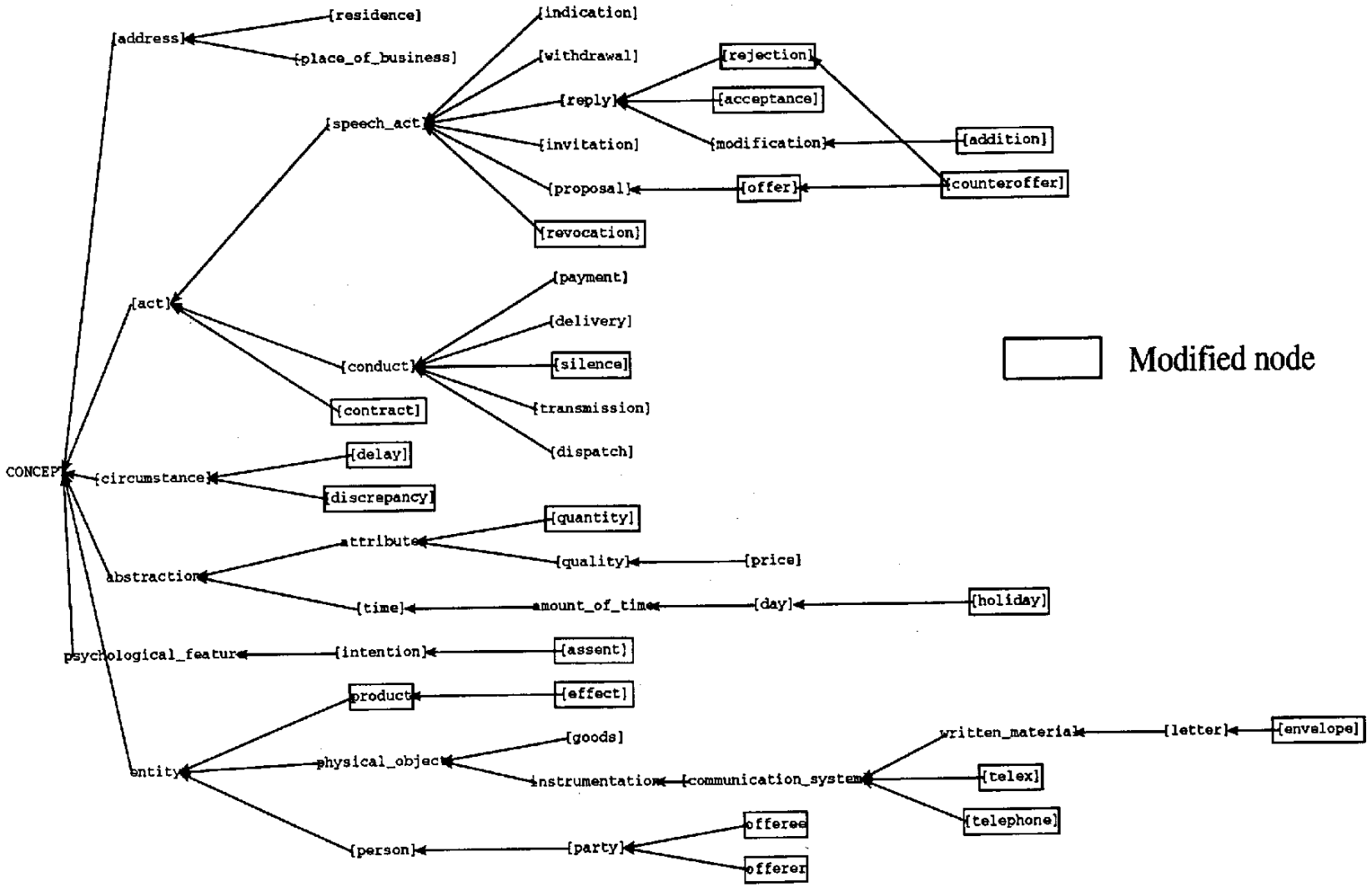


図３．２．３-４　法律オントロジー

表３．２．３-１　法律分野におけるDODDLEの評価実験

入力語句数	46
ベストマッチノード数	44
A:獲得されたPAB数	11
B:獲得されたSTM数	14
X:ユーザが追加修正を加えたPAB数	3
Y:ユーザが追加修正を加えたSTM数	3
支援率((1-(X +Y)/(A+B))*100)%	76%

表３．２．３-１から、PABの抽出とSTMの移動により、比較的高い支援率を達成する事ができたといえる。しかしながら、PABとSTMだけで捉えられない変動も多々あり、意味レベルの支援方法について今後検討していく必要がある。

また、現在のDODDLEは、領域概念階層構造の支援が主であり、領域概念定義の支援については部分的に行っているが十分ではない[関内 97]。法律概念の定義には、多大なコストが必要であり、知識工学だけの技術に留まらず、例えば、Plinius プロジェクト[P.H.Speel 95]のように、専門書テキストから概念定義を半自動的に獲得するような計算機環境を目指して、自然言語理解の研究成果や、語彙知識を表現するための論理である Denotational Logic など、知識表現の研究成果の導入もはかる必要があろう。

参考文献

[G.Heijst 95] Gertjan van Heijst : The Role of Ontologies in Knowledge Engineering, Dr.Thesis, University of Amsterdam (1995)

[G.A.Miller 95] G.A.Miller : WordNet: A Lexical Database for English,ACM, Vol.38, No.11, pp.39-41 (1995) http://www.cogsci.princeton.edu/~wn/obtain/

[関内 97] 関内律恵子、小森聡、青木千鶴、榑松理樹、山口高平： DODDLE:計算機可読型辞書を利用した領域オントロジー構築支援環境(2)－概念定義構築支援－,第55回情報処理学会全国大会, 3AF-7 (1997)

[曽野 93] 曽野,山手 : 国際売買法, 青林書院 (1993)

[P.H.Speel 95] Piet-Hein Speel : Selecting Knowledge Representation Systems, Dr.Thesis, University of Twente (1995)

[W.Swartout 96] W.Swartout, R.Patil, K.Knight and T.Russ ： Toward Distributed Use of Large-Scale Ontologies, KAW96, pp.32-1 - 32-19 (1996)

[T.Yamaguchi 97] T.Yamaguchi and M.Kurematsu: A Legal Ontology Rapid Development Environment Using a Machine-Readable Dictionary, LEGONT'97 pp.69-73 (1997)

3.2.4　タスクオントロジー記述言語・構築環境

(1)　タスクオントロジー記述言語 CLEPE [瀬田98a] [瀬田98b]

CLEPE（Conceptual LEvel Programming Environment）は阪大溝口研究室で開発されたタスクオントロジー記述言語・環境である。CLEPEは人間が行う様々な問題解決の内で、専門家と呼ばれる人々が日常的に行っているスケジューリングや診断などの問題解決を対象としている。CLEPEにおいて、エンドユーザは、(1) 日頃使っている平易な言葉を使って問題解決モデルを記述（モデル化）することができ、(2) 問題解決モデルの実行内容をわかりやすい形で確認／デバッグすることができる。さらに、(3) 平易な言葉を使って記述した問題解決モデルを、記号レベルのプログラムコード（Lispコード）へと変換することができる。

システムのブロック図を図3.2.4-1に示す。上側はオントロジーを開発する者、右側はオントロジーを使って問題解決モデルを記述する者のためのインタフェースである。

タスクオントロジーはその役割によって、問題解決知識の記述上の規則を定めるレキシカルレベルオントロジーと、その記述の意味内容を定義するための概念レベルオントロジーに分割されている。二つのオントロジーは、それぞれレキシカルレベルモデル、概念レベル実行モデルの構成上の規約を与えている。

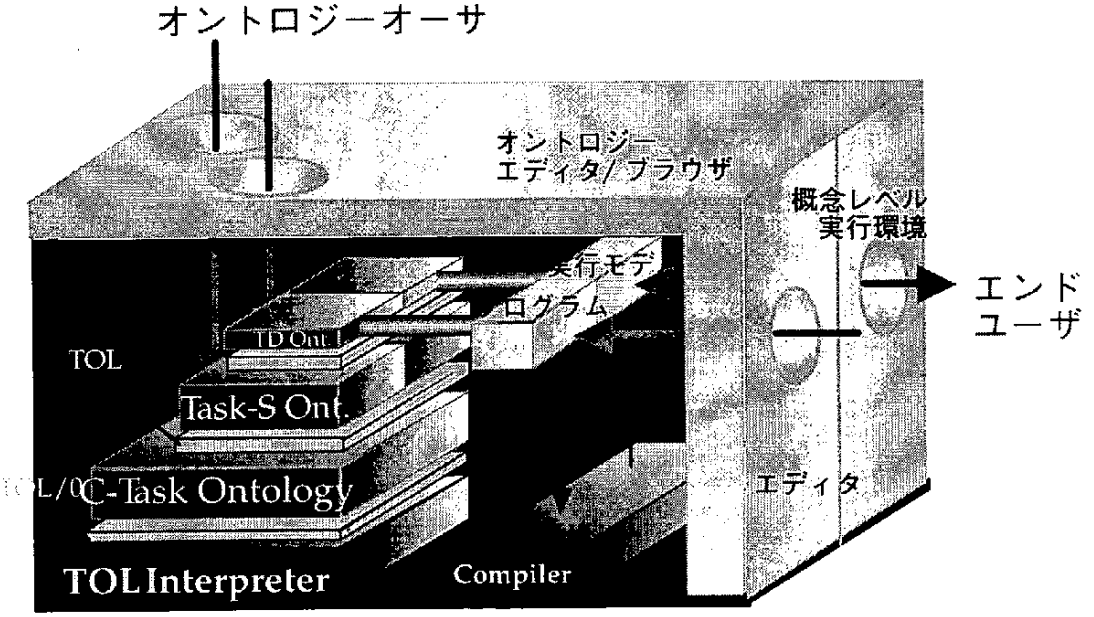


図3.2.4-1　CLEPEのブロック図

CLEPE、エンドユーザ、二つのモデルの関係を直感的に書き下すと次のようになる。

「エンドユーザが自分の問題解決知識をレキシカルレベルモデルとして記述し、
CLEPEはその意味内容を概念レベル実行モデルとして再構成する」

レキシカルレベルモデルを構成するそれぞれのノードは、汎化プロセス（GP：Generic Process）と呼ばれ、「動詞+名詞」を基本構成要素として処理概念を表わす。ノードを結ぶリンクは制御フローを表している。ここで動詞、名詞に当てはまるタームはレキシカルレベルオントロジーとして用意されている。概念レベル実行モデルを構成する基本要素は、アクティビティ、オブジェクト、叙述概念である。そこでは、問題解決の実行を通じて「(1) 何が、(2) どの様な作用を受け、(3) 結果としてどの様な状態になるか」ということが、(1) オブジェクト（e.g. Rcp:スケジュールの受け手）、(2) アクティビティ（e.g. Select）、(3) 叙述概念（e.g. Temporary）によって、明らかにされる。オントロジー開発者が定義するCore-taskオントロジーはタスクタイプに依存しないオントロジーであり、Tolの基盤を構成している。問題解決モデル開発者は、タスクオントロジーが提供するレキシカルレベルの用語を用いて自然言語感覚で利用することができる。

以下では、TOLを用いて具体化されるタスクオントロジーの表現について、ここではスケジューリングタスクタイプを対象に構築されているTask-依存オントロジーについてTO/K-L（Knowledge-Lexical）とTO/K-C（Knowledge-Conceptual）の記述例を示しながら、タスクオントロジーの概念の具体化方法を紹介する。

TO/K-LとTO/K-Cの記述に際しては、TO/K-Cで記述される概念の意味と、それを言語として外化する際の言語的側面を捉えたTO/K-Lの間の対応関係が重要となる。例えば、スケジューリングタスクにおける「assignmentの集合」に対するユーザの認識は、同一のassignmentの集合がタスクの実行を通して、「（割り付けが）完了していない」「（割り付けが）完了したが解制約を完全に満足していない（S-temporal）」、「制約を満足している」というように状態が変化していくものと捉えている。一方その認識を通常のフローチャートに対応するGPN（Generic Process Network）として表現する際にはその状態の変化に対応してそれを表象するタームが「途中解」「暫定解」「最終解」のように異なっている。つまり、人間の認識としての同一オブジェクトがGPN上では異なるエンティティとして記述されることになる。ユーザが表現したい意味（TO/K-C）とターム（TO/K-L）の連続性を提供するために以下に示すように、TO/K-LとTO/K-Cを分離し、相互の対応関係を定義として明確に記述することで言語とそれが指示するオブジェクトとの対応関係が明示的に表現されることになる。

以下に「暫定解」（TO/K-L）および「assignment-set」（TO/K-C）の定義を示している。Task-Sオントロジーの概念である「暫定解」の定義には、「暫定解」が「解」ク

ラスのサブクラス概念であり、対応するオブジェクトとして「暫定状態（S-temporal）にある解オブジェクト（O-assignment-set）が指定されている。上で述べたように、名詞定義に現れる、階層関係（class-hierarchy）、対応するTO/K-Cのオブジェクト概念（cor-object）、オブジェクトの取り得る状態（status-spec）などのスロットは、C-Taskオントロジーにおいてその必要性が明示的に記述されている。

また、TO/K-Cのオブジェクト概念のクラス定義には、クラス階層、タスクの実行による影響を受けない普遍的な性質（object-spec）、タスクコンテクストのもとで付加されうる状態制約（status-spec）が記述されている。assignment-setのクラス定義には、assignment-setがオブジェクトの下位概念であり、タスクの実行を通して普遍的に持つ性質として「assignment-setがassignmentの集合」であること、また状態制約として「中間（割り付けが完了していない）」、「部分の」、「暫定の」、「最適の」という状態を取り得ることが記述されている。

```
(Define-Tol-noun temporal-solution (?t-sol)
 :class-hierachy (subclass-of temporal-solution assignment-set)
 :cor-object (?O-ass-set :constraints
          (instance-of ?O-ass-set O-assignment-set))
 :status-spec ((S-temporal ?O-ass-set)))

(Define-Tol-object O-assignment-set (?$O-ass)
 :class-hierachy (subclass-of O-assignment-set object)
 :object-spec (?$O-ass :constraints
           (forall ?O-ass
            (=> (member ?O-ass ?$O-ass)
              (instance-of ?O-ass O-assignment))))
 :status-spec (?status-spec :constraints
           (member ?status-spec
            ((not (s-temporal ?$O-ass)) (s-partial ?$O-ass)
            (s-temporal ?$O-ass)(s-optimal ?$O-ass)))))
```

（2） オントロジー構築統合環境の開発 [古崎97]

オントロジー構築を支援する為の計算機環境に求められる要求仕様を洗い出し、以下の機能を持つオントロジー構築環境の第一版が開発された。

① 概念間の関係の表示・編集機能

概念間の関係の表示・編集をする画面の構成要素として「ノード」と「リンク」の

2種類を導入している、ノードは概念を表わし、リンクは複数のノード間を結ぶ線分として表示され概念間の関係を表わす。ノードとリンクを用いて、概念と概念間の関係を体系的に表示する。画面に表示されたノードとリンクを編集することにより、概念と概念間の関係の記述が行われる。

② 概念の定義の表示・編集機能

ノード・リンクを用いてグラフ状に表示がなされているオントロジーの中から任意のノード・リンクを選択し、定義内容の表示・編集を行う。定義の記述を効率よく行うために、定義内容は専用の画面に表示される。

③ ネットワークを介した共有機能

ネットワークを用いたデータの共有を行うために、ユーザーネームやパスワードを用いたユーザーの識別・管理が導入されている。また、データの更新履歴も管理される。

④ オントロジー利用の支援

構築したオントロジーを利用して知識を記述するには、オントロジーで定義されたクラスの定義内容を表示・参照しながらインスタンスに当たる概念を作成する機能が必要となる。そのような作業を効率的に記述作業を進めるために、クラス・インスタンスのそれぞれが表示される2つの画面を用い、表示機能については、インスタンスが作成可能なクラス名を一覧表示する機能が提供されている。

このシステムは、システムの動作を管理する「オントロジーサーバ」、オントロジーの表示・編集を行う「オントロジーエディタ」、整合性の確認を行う「オントロジー解釈器」の3つのシステムから構成されている。

これらのシステムのプログラム本体および本システムで構築したオントロジーのデータはサーバマシン上に置かれる。「オントロジーエディタ」の開発には「Java言語」を用いており、クライアントマシン上でWWWブラウザを用いて以下のように実行される。

・クライアントマシンでWWWブラウザを実行し、オントロジーサーバとネットワークを介して接続する。
・ブラウザがオントロジーサーバにオントロジーエディタのアプレット（Javaプログラム）を要求する。
・オントロジーサーバがブラウザにオントロジーエディタのアプレットを返す。
・ブラウザはアプレットを解釈してオントロジーエディタを実行する。

一方、「オントロジー解釈器」は、計算機の負荷を押さえるためサーバマシン上で必要に応じて実行される。ネットワークを介したクライアントマシンとオントロジーサーバ間でのデータの入出力は、「オントロジーサーバ」によってなされる。システムは、現在「オントロジーエディタ」および「オントロジーサーバ」についてはプロトタイプが完成しているが、「オントロジー解釈器」に関しては今後の課題である。図3.2.4-2に実行中の画面を示す。

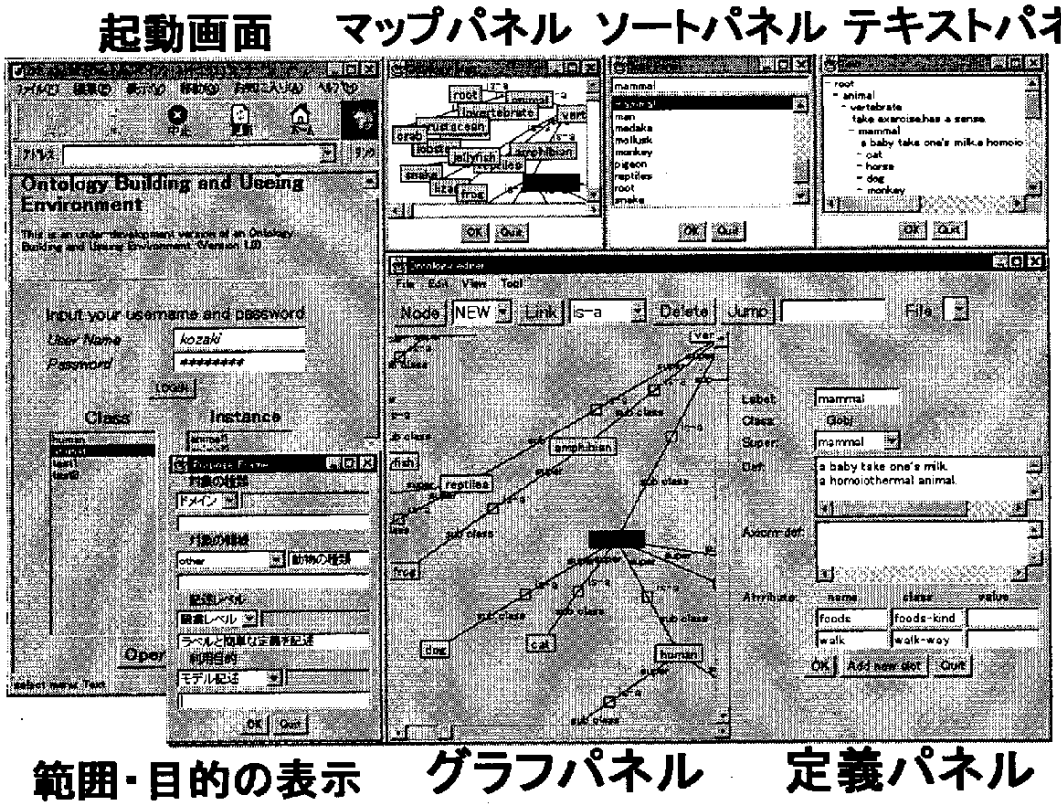


図3.2.4-2　オントロジー構築環境の実行例

参考文献

[瀬田98a] 瀬田、他：問題解決オントロジーの構成—スケジューリングタスクオントロジーを例にして、人工知能学会誌、Vol.13, No.4, 1998.（掲載予定）

[瀬田98b] 瀬田、他：問題解決オントロジーに基づく概念レベルプログラミング環境CLEPE、電子情報通信学会論文誌、1998.（掲載予定）

[古崎97] 古崎、他：分散型オントロジー構築・利用環境の開発、第11回人工知能学会全国大会論文集、pp.245-248, 1997.

3.3　言語知識：テキストのオントロジー

[言語知識の定義]

物理的、観念的存在、あるいは、対象は人間の精神、あるいは、知能に反映され認識、あるいは、知識、あるいは、概念となる。人間は色々なメディアによって表現、あるいは、記述し、認識を外在化（外化）、あるいは、対象化させる。外在化された認識は、存在そのものに代わって、あるいは、存在そのものとして人間の精神への反映機能を持つことになる。すなわち、人間の思考や互いのコミュニケーションは外在化された認識を媒介として行われることになる。（自然）言語は、このような機能をになうメディアの中で最も汎用のメディアである。

世界は多様な分野から成り立つ。それらに対応する多様な（自然）言語による表現、すなわち、言語表現（テキスト）が存在する。言語知識（言語学知識とは異なる）とは、言語表現を存在とした時のその言語表現としての人間精神への反映、すなわち、言語表現すべてが共通に持つ特性、言語表現が言語表現であるべき諸条件に関する知識である。言語知識も（自然）言語とある種の形式言語（ある種の知識表現言語）によって外在化される。

言語知識は、大きく２つの知識に分けられる。表層知識と深層知識である。表層知識は、言語表現の外見で判断される諸条件に関する知識であり、さらに形態的知識と構文的知識に分けられる。形態的知識は、表層上の基本構成要素に関する知識である。文字、形態素、語という基本構成要素についての知識であり、これらの要素にどのようなものがあり、各要素の特性がどのようなものであるのかという知識である。基本構成要素は、文章、文書のレベルで設定されるものもある。構文的知識は、基本構成要素が文、文章、文書を構成する時の構文構造に関する知識である。構文カテゴリにどのようなものがあり、構文カテゴリが構文構造上満たすべき制約条件がどのようなものであるのかという知識である。深層知識は、言語表現の意味的な特性に関する知識である。基本構成要素から文、文章、文書にいたる意味要素と意味構造に関する知識である。どのようなレベルの意味要素にどのようなものを設定するのか、どのような意味構造に対して意味要素がどのような制約条件を満たすのかの知識である。深層知識は、言語表現の意味をどのレベルでとらえ、どのような意味表現形式や意味表現言語で外在化させるかによって、様々なものが設定される。言語の意味処理の立場から、ある程度のコンセンサスが得られているものに概念レベルの深層知識がある。

ここでは、深層知識に重点をおいて各種の言語知識を比較検討する。

[オントロジーの定義]

オントロジー（存在論）とは、字義的には、人間の精神から独立して存在するものの特性、存在そのものとは如何なるものであるかを論究する学である。すなわち、

存在論（ontology）：あらゆる存在者が存在者として持つ共通の規定（存在していること）やその根拠を考察する。観念論的存在論、唯物論的存在論、存在学。
［広辞苑］

対して、認識論がある。すなわち、

認識論（epistemology; theory of knowledge）：認識の起源・本質・妥当範囲を論究する。認識の起源については経験論・理性論、その対象については実在論・観念論などがある。形式論理学（小論理学）に対し内容論理学または大論理学といわれ、真理を求める新しい方法の反省的発見の場である。知識論。［広辞苑］

本来は、認識論が主題である。人間の認識論は、ある意味ではコンピュータの存在論とみなせることからオントロジーという用語が借用されたとも解釈できる。

認識論とは言え、科学や工学の立場では、外在化され、客観的に対象化出来るものを手掛かりにして認識を論究する。あるいは、出来る部分に議論を絞る。そこで、認識や知識に関してもメディアで表現され外在化されたものを論究の対象にすることになる。すなわち、メディアで表現され外在化された認識を存在として論究する、あるいは、メディアで表現され外在化されたものを通して存在そのものを論究する。そこでオントロジーという用語が採用されたとも考えられる。さらに、知識工学からの流れをくむオントロジー工学では、表現するメディアを知識表現言語（形式言語）として、あるいは、それに関連付けて論究を進める。

［オントロジー工学の定義］

知識工学とオントロジー工学の違いは何かである。知識工学は、知識表現言語で対象となる知識を表現し切り、コンピュータが理解できるようにし、コンピュータ上の知識ベースとして知識を外在化しよう、あるいは、出来るという前提でスタートした。色々なエキスパートシステムが作られたが、同じ知識もシステム毎に扱いが異なり、知識ベースの中身も外在化された知識といえるものにはならなかった。知識表現言語も表現能力とコンピュータの処理能力の両面から大方の同意が得られる標準的なものが定められるというところまでに至らなかった。表現のパラダイムも述語論理からファジー、ニューロと変転を続けた。このような状態から、打開策としての知識獲得の議論も確実な成果に結びつくことがなかった。そこで、コンピュータが処理できるようになる段階のもうひとつ手前で知識を表現し、体系付けようという試みを始めたのが、いわゆるオントロジーであり、オントロジー工学である。オントロジー工学では、知識工学が達し得なかった知識の外在化を目指す。したがって、オントロジー工学の課題は次のようになる。

①外在化出来る知識のレベル、領域を明らかにする。
②知識の構造を明らかにし、理論や表現するスキームを明らかにする。
③知識を表現し、外在化させる。大規模になる場合は開発方法を明らかにする。
④外在化された知識の有用性を明らかにし、利用方法、応用分野を明らかにする。

現在、オントロジー工学では、これらの課題に対しておおよそ次のようなコンセンサスのもとで取り組みが行われている。①に関しては、エキスパートシステムの開発経験によって得られた類別に基づく知識領域、電子化辞書開発などによって得られた言語知識の領域などである。②に関しては、知識は大きく幾つかの側面からなるというところから始める。側面は対象領域に特有のものである。各側面に対して知識の基本構成要素となる概念を列挙し、概念の定義を行なう。他の概念との関係を列挙することと、概念の内部構造を記述することによって基本的な定義がなされる。他の概念との関係には、階層関係、全体－部分関係、同値関係などが代表的である。概念の定義には、さらに色々なものが付け加えられる。③に関しては、現在、大規模なものは言語知識に関するものと、エキスパートシステム系から始められたものとしてはCYCがある。CYC自身は、概念階層部分のみをオントロジーと呼んでいるが、本来は、CYC全体がオントロジーである。④に関しては、このように外在化された知識は、高い知識共有性、知識再利用性を持っていることである。また、このような知識を用いることによって、容易にコンピュータ処理可能な知識を作ることが出来る。その時の方法としては、自動的に変換する、人手によって変換する、人手による作成を支援するなど様々である。更に、人間用の知識としても利用される。人間にとっても理解が容易であるという性質は重要な要件である。オントロジー工学では、知識を外在化させることが第一義である。その知識をコンピュータに理解出来るようにするのは次のステップであると考える。外在化された知識は、人間にとっても十分に有用でなければならない。

言語知識は、外在化された言語表現、すなわち、テキストが本来テキストであるべき諸性質、諸条件を外在化した知識としたものである。したがって、言語知識は言語表現のオントロジー、テキストのオントロジーである。

[世界知識と言語知識]

世界知識と言語知識の関係を明らかにする。関係を際立たせるために、まず、世界知識を表現するメディアを（自然）言語として、言語知識との関係をみる。

（自然）言語で表現された知識、テキストを理解するのが、コンピュータであるとするのか人間であるとするのかによって大きく異なった考え方になる。

①コンピュータが理解する：コンピュータがテキストを知識として理解できるため

には、コンピュータが自然言語理解の機能を持たねばならない。そのためには、コンピュータが言語を理解するための十分な世界知識、常識を持つことである。したがって、まず、世界知識をコンピュータに持たせることからアプローチしなければならない。これは、知識処理から言語処理にアプローチする方法で、CYCの考え方が典型である。

② 人間が理解する：人間がテキストを理解するのであるから、コンピュータの自然言語理解機能はごく限られたものでよい。テキストの読み手である人間は、テキストを知識として理解するための十分な世界知識を持っている。この世界知識を前提にして必要な知識のみをコンピュータに持たせればよい。コンピュータに持たせる知識の種類、レベルは、コンピュータに期待する言語処理機能によって異なってくる。コンピュータに持たせる知識は言語知識の範囲内で良い。これは、言うなれば言語処理から知識処理にアプローチする方法である。

閉じた系としてロボットを作るならまだしも、通常、すべての情報表現の最終目的は、人間に情報を伝えること、知識を伝えることである。コンピュータは、情報表現の送り手（書き手）と受け手（読み手）の間にあって、何らかの仲介役を果たすだけである。受け手（読み手）が世界知識を持っていることを前提にすれば、仲介役のコンピュータが持つ知識は限られたものでよい。どのような仲介役に対して、どのような知識を持てばよいのか、言語処理に関しては、かなり明確な基準を示すことが出来る。コンピュータが実現する言語処理機能に対応して言語知識の種類やレベルを確定することが出来る。少なくとも、確定に向けて秩序だった議論をすることが可能である。機能と知識の関係に関する議論をこのように体系的に行えるのは、言語処理の分野だけである。

[深層言語知識と世界知識]

本節では、言語知識も深層の部分に重点を置いて調査結果をまとめるが、世界知識、常識として開発されてきたものもあわせてまとめる。そこで、深層言語知識と世界知識の関係を簡単に整理しておく。

深層言語知識は、言語表現（テキスト）の意味表現に関するオントロジーである。意味の表現形式にしたがって、多様な意味表現が存在する。ここでは、表層に近い表現形式である概念表現形式を取り上げる。概念表現は、単語の概念（意味）を大きく実体概念と関係概念に分け、文の概念（意味）を述語概念（実体概念の一種）に対する実体概念の関係概念（深層格）による修飾として表す。この概念表現に対するオントロジーが概念レベルの言語知識である。概念レベルは、表層に近いことから、表層表現から直接的に概念表現を抽出することが出来る。あるいは、出来ると期待できる。また、他のより深層に渡る意味表現形式に対して、表層表現からの共通の中間ステッ

プとして利用することが出来る。そして、より深層の意味表現形式のあるレベルから、いわゆる世界知識の表現につながっていく。

概念レベルの意味表現と世界知識レベルの意味表現の違いを一例で示しておく。

＜空が青い＞という知識の扱いを例にとると、概念レベルと世界知識レベルでは次のような差がある。

概念レベル：＜空が青い＞を表現する言語表現は多数のものがあるが、それぞれに対応する概念表現を個別のものとして扱い、妥当性を知識化する。すなわち、
「空が青い」
「空の色が青である」
「空が青く見える」
を個別に扱い、文概念表現中での語概念の共起条件を知識化する。表層表現に強く依存した知識である。

世界知識レベル：＜空が青い＞を［＜空＞は＜色の属性＞として＜青色＞を持つ］という知識として扱う。表層表現の違いを捨象し、共通の意味となる一つの知識として扱う。

3.3.1 言語知識の構造

［言語知識への要求仕様］

言語知識全体の基本的な構造を説明する。まず、言語知識の基本的な枠組みを明確にするために、言語知識への要求仕様を要請事項の形式で列挙する。

(1) 十分な有用性を持つこと。

今後の自然言語処理の研究やシステム開発に十分な有用性を発揮できることである。自然言語処理の今後の技術動向に適切に沿うものでなくてはならない。形態素・構文処理に対しては頑健性の達成、意味・文脈処理への本格的な取り組み、コーパスベースなど大量データ解析に基づく言語処理、文書処理という観点からの取り組み等が今後を代表的する技術動向である。これらの技術動向に沿うためには、必要とする言語現象に対し十分な網羅性を達成すること、現実の言語現象を忠実に捕捉すること、表層の情報から深層・意味にかかわる情報までを適切に扱うことができること、大量のコーパスやテキストデータに対して対応できることなどが達成されねばならない。

(2) 十分な柔軟性、汎用性が得られること。

状況の変化に適応できる柔軟性や色々な状況に対応できる汎用性を十分に持つことである。このためには、安定して確保できる部分や共通の合意の得られる部分を土台にして、そうでない部分は巾を持たせて関連付けておくことである。そのようにして、言語理論や処理方式の多様性への対応、他の言語への対応、多様な応用への対応などが可能になる。さらに、十分に予測できない将来の技術変化への対応や客観的な検証を可能にすることへの対応も保証されることになる。

(3) 十分な実現可能性を有すること。

大規模で十分な精度を持つものが低コストで実現できることである。構造がモジュール化されており漸進的な実現が可能であることである。目的とする機能やシステムに対し、それを利用して必要な内容と規模と精度のものが容易に達成できるようにすることである。コンピュータによる十分に強力な開発支援機能が用意できることも重要であり、しかも、規模の拡大や精度の向上にともなってこの支援機能自身の能力が増大していくことである。これによって、規模の拡大が開発効率を落とさぬこと、出来うれば上昇させていくという自己増殖的な仕組みを実現することが可能になる。このような仕組みや基本的な開発素材の蓄積という環境が整えられることによって、あまり訓練を受けていない作業者も安定した効率の良い作業ができるようになる。なお、実際の言語データは開発コストと利用価値のバランスをとりながらそれぞれに必要な部分と量をそれぞれが開発していくことになる。そのような個別の努力が共通に納まるべき枠組みを定めることは重要である。そのような枠組みによって、個々の努力が協調し合うための共通基盤が得られることになる。

[言語知識の構造]

言語知識の全体の骨格となる構造を定める[横井ほか 96] [横井ほか 97] [松本ほか97]。ここでいう構造とは情報・知識が内部に持つ論理的な構造である。格納や検索の仕方についての実際の扱いに関することは含めないようにしたものである。言語知識には、語彙（単語）に関することばかりではなく、シソーラスやコーパスやテキストデータを含む統合的な言語データを対象にした知識が含まれる。

言語知識の全体構造は、記述の単位、記述のレベル、言語の種類の３点で特徴付けられるサブ知識群をもとに組み立てられている。この特徴を座標軸に対応させると言語知識は図３.３.１-１に示すような構造体となる。サブ知識（サブ知識ベース）が基本的なモジュールとなり、これを単位に言語知識は大枠としてのモジュール化がなされることになる。その３つの特徴について説明する。

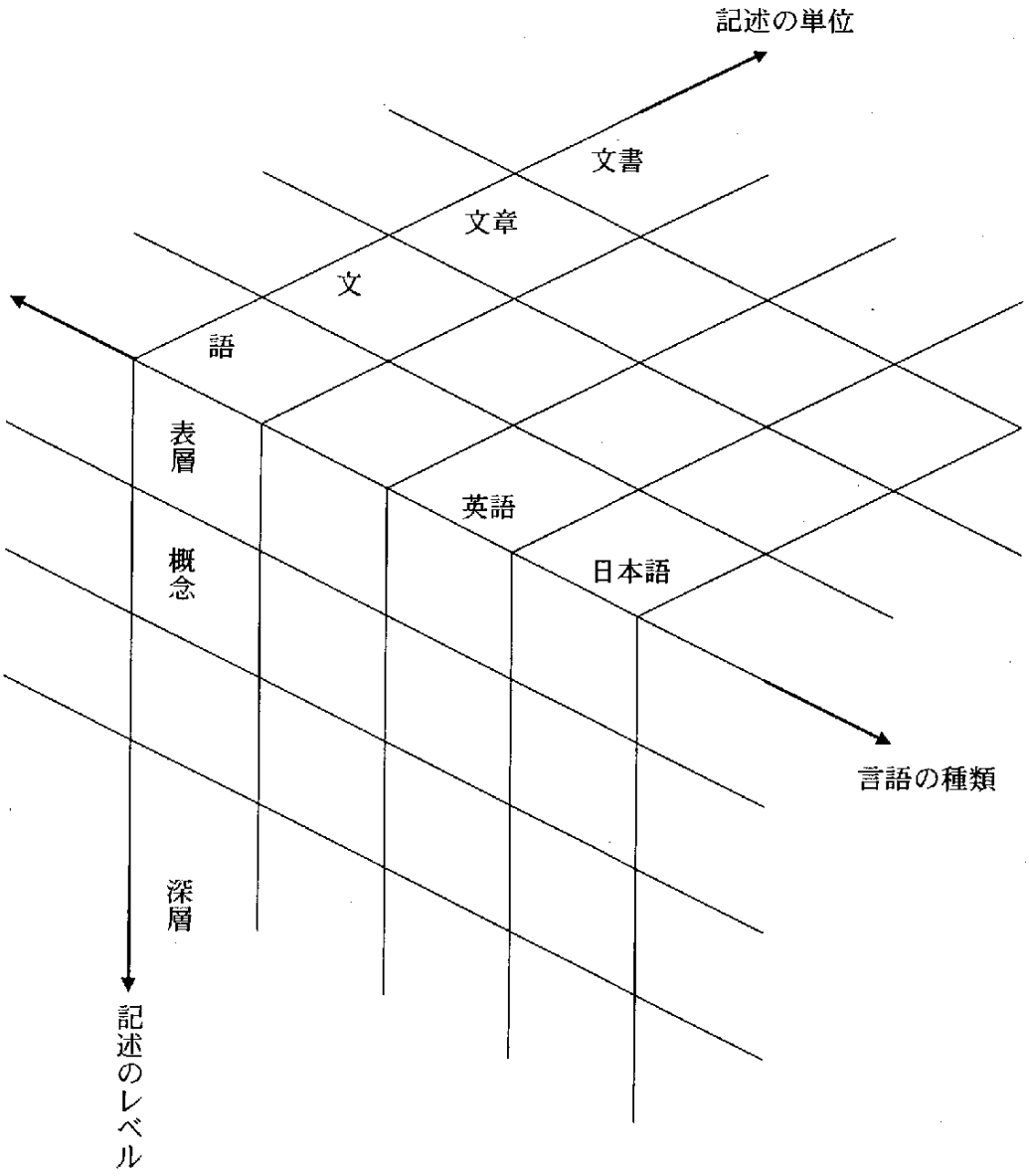


図３.３.１-１　言語知識の全体構造

(1) 記述の単位：言語表現（言語によって表現された情報）のどのような構成単位の知識を記述するのかということである。語、句、文、文章、文書という構成単位のどれについて記述しているかということである。意味の観点からは、このような構成単位の役割は、それほど明確なものではなくなる。本来、自然言語は要素還元的ではないからである。しかし、言語現象を整理していく立場からは外見的に識別できる構成単位という見方は有力な手掛かりとなる。

(2) 記述のレベル：言語表現のどのレベルの知識を記述しているのかということである。表層のレベルから意味記述にかかわる多様な深層のレベルまでがある。表層レベルは

言語表現の外見的なところで判断できる知識を対象にする[国広 98]。形態や構文にかかわる知識は当然であるが、意味、文脈、運用にかかわる知識も表記上で記述できるものは対象にする。これらは深層レベルの知識の判断基準を与えるものである。深層のレベルは、意味の表現形式[郡司ほか 98]によってさらに色々なレベルが設定される。意味表現に対し色々な研究がなされてきたが、実際には部分的な適用であったり理論的な枠組みの提案であったりで、大規模な言語知識に適用できる段階にはない。この中で比較的に安定したレベルとして表層に近い表現形式に基づいた概念レベルというものがひとつの基準になる。

(3) 言語の種類：どの言語の言語表現を記述の対象にしているのかということである。日本語、英語等基本的にはすべての言語を対象候補とする。基本的な構造は言語に共通であるが細部は言語の特性を反映する。

言語知識を表現するための基本となる情報単位が辞書項目である。表層レベルでは表層辞書項目、概念レベルでは概念辞書項目である。サブ知識は辞書項目の集合として構成される。辞書項目の内容の基本部分は共通であるが、詳細は記述の単位ごとに異なる。辞書項目は、入れ子となる多数のサブ項目からなる。サブ項目には辞書項目自身の情報や他の辞書項目との関係情報が含まれる。この関係情報によって辞書項目間の関係付けが表現され言語知識としての情報の構造が形作くられる。関係には、サブ知識間にまたがるサブ知識間関係とサブ知識内のサブ知識内関係がある。サブ知識間関係の主要なものは次の３つである。

(a) 構成関係　：記述の単位の軸にそって同じ記述レベルのサブ辞書間に定義される。上位の構成単位のどの要素に対応しているのかを示す。あるいは、下位の構成単位のどのような文脈となっているのかを示す。

(b) 対意関係　：記述のレベルの軸にそって異なった記述レベルのサブ知識間に定義される。ある記述レベルの辞書項目に対しそれより深層方向の記述レベルのもので意味表現となっているものへの対応関係である。意味表現もある種の言語による意味情報の記述であると考えると、表層言語と深層言語の間の対訳関係と見なすことができる。

(c) 対訳関係　：言語の種類の軸にそって同じ記述レベルのサブ知識間に定義される。異なる言語の間で同義（ほぼ同義）であるという事実が認められる対応関係である。この関係は、原則としてすべての言語対の間に定義される。

[概念、とくに語（語彙）概念の設定の仕方]

概念は実体概念と関係概念に分けられる。実体概念とは、もの、こと、事象、事象列など実体を有するものに対応する概念である。関係概念は実体概念どうしのかかわり方を表す概念である。ひとつの概念を実体概念と見るのか、関係概念と見るのかは、観点によって恣意的となる側面がある。概念レベルの概念は表層表記に直截的に対応付けられる状態で設定されるもので、この観点から概念の種別が決定される。さらに、語、文、文章、文書という表層での記述単位（構成単位）に対応して、それぞれ実体概念と関係概念が定義されることになる。

語（語彙）概念に関しては、名詞、動詞、形容詞、副詞等の概念語（内容語）と通常呼ばれているものによって表されるものが実体概念である。一方、助詞、前置詞等の機能語（関係語）と通常呼ばれているものによって表されるものが関係概念である。関係概念は語の位置等によって表される場合もある。この場合の関係概念は、対応する表層表現を持たぬことになる。実体概念についても、また関係概念についても対応する品詞は言語学的な観点ばかりではなく自然言語処理の観点からの実際的な扱いがなされる。すなわち、通常の語と同じような働きを示す慣用的に用いられる表現などは相当語として扱い、それらの働きに応じてひとつの概念を対応させる。

語は実際の文、文章の中で、ひとつの表層表記のままで実に多種多様な意味を表現する。電子化辞書としては、意味の記述単位である概念として何をもってくるのか、いかなる観点に立つか、いかなる粒度のものとして見るか等についての明確な指針が必要である。指針を以下の３点に整理する。まず、実体概念を中心にしてである。

1) 概念化されたもの

文脈上では語はほとんど個別の対象を表すが、それらのすべてに共通の性質をまとめ概念化した内容を対象とする。いうなれば、クラスを定義するものとして概念を扱う。

2) 慣用化されたもの

文脈上で語はさまざまなレトリカルな意味に対応付けられることがある。この場合、慣用化し定着しているものは、派生義を原義とは独立した概念として扱う。

3) 一体化されたもの

多くの属性、あるいは多くの要素概念を含む一体化されたものとして扱い、属性（要素概念）への分解は別途適切な手段を講ずる。文脈上では、それぞれの属性（要素概念）が個別に対応付けられる場合が多くある。しかし、個別の属性（要素概念）に分解する作業をこの段階では行わない。

関係概念は、実体概念ほどバラエティに富むものではないが、色々な詳細化のレベルが設定できる。しかし、まずは粗い近似のレベルで言語知識データの開発を行い、利用経験を蓄積する中で詳細化を進めるのが妥当である。

次に、以上の指針に従い、概念の定義を実質的なものにするため表層語と語概念との対応付けを行う。表層レベルにおいて、各単語に語義が対応付けられている。多義語には複数の語義が対応している。まずこの語義を上記の指針に従って整理する。単語ごとに整理された語義を、異なる単語の間で比較して、同義とみなされる語義どうしを統合化する。このようにして得られたものを語（語彙）概念とする。

3.3.2 EDR電子化辞書（日本電子化辞書研究所）

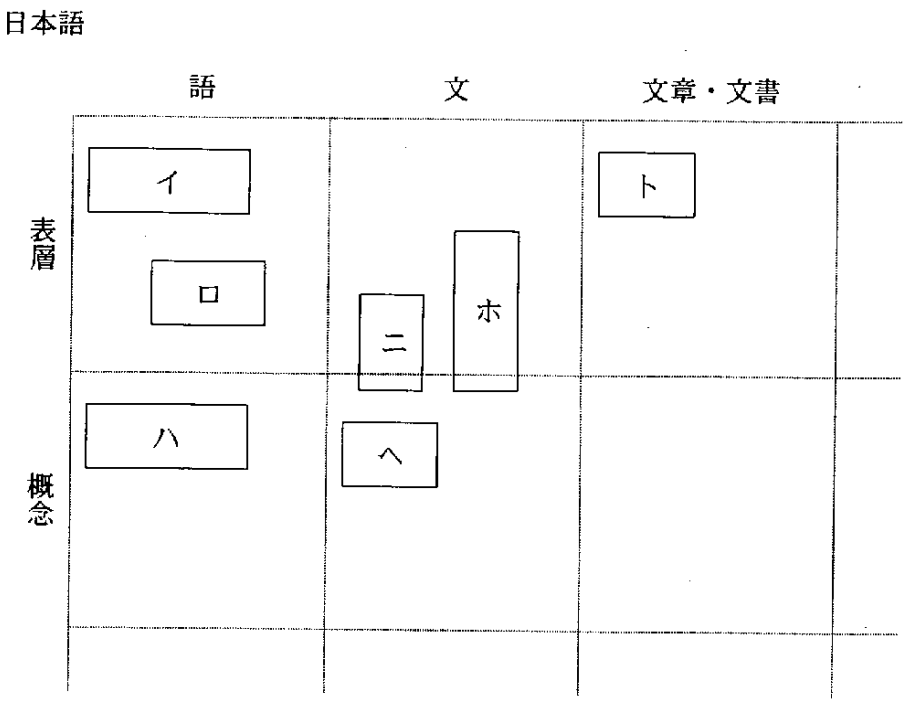
言語知識の構造にEDR電子化辞書[日本電子化辞書研究所 95]を対応付ける。EDR電子化辞書は、言語知識の基本部分の、また当面のニーズに答えるという要請も踏まえた実現である。

EDR電子化辞書は、言語の種類としては、日本語と英語を対象にし、分野としては、日常一般に流通している事務文書、技術文書に対し分野共通となる部分と情報処理分野に対応する部分を対象にしている。日本語単語辞書、英語単語辞書、日英対訳辞書、英日対訳辞書、概念辞書（概念体系辞書、概念記述辞書）、日本語共起辞書、英語共起辞書、日本語コーパス、英語コーパス、日本語テキストベース、英語テキストベース、から構成されている。これらを、図３.３.１-１に示した構造の日本語と英語の部分に対応付けたものが、図３.３.２-１と図３.３.２-２である。基本語対応部分の第一版（1995年４月第１版の基本語対応部分）の規模に関する数値を表３.３.２-１にまとめる。なお、まとまったバージョンアップを1998年９月を目標に行うべく作業が進められている。

オントロジー工学の立場から関心の集まる概念辞書部分[荻野ほか97]について、現状を説明する。現在、日本電子化辞書研究所は米国ANSI Ad-hoc委員会が推進する“オントロジーの標準化”の一環として、WordNetとEDR概念体系を対応づけを進めている。まずトップレベルの126の概念についてつきあわせが行われ、さらに拡大して2,000あまりの概念についての作業がまとめられつつある。これによって、オントロジーの国際的な共通化を行う試みを始めている[Ogino et al.97] [Utiyama and Hasida97]。EDR概念体系は、40万概念を6,221個の中間ノード、最大の深さ15の体系に分類した状態になっている。

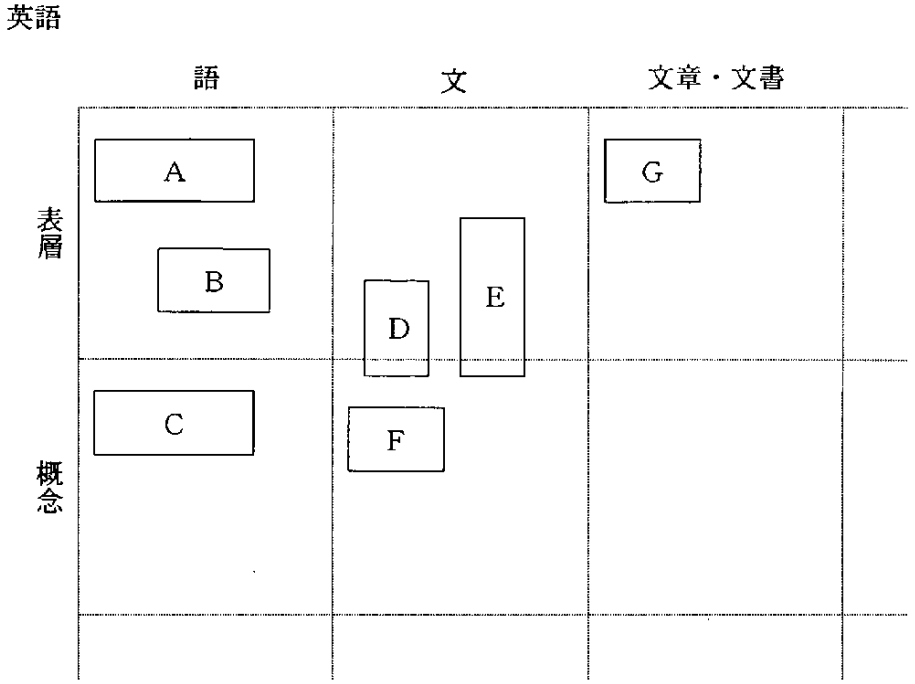
表３．３．２－１　EDR電子化辞書の規模（1995年4月第1版の基本語対応部分）

日本語単語辞書	25万語
英語単語辞書	19万語
概念辞書 概念記述・概念体系	40万 概念
日英対訳辞書	23万語
英日対訳辞書	16万語
日本語共起辞書 （日本語動詞共起パターン辞書）	90万句 （基本語動詞5千語）
日本語コーパス	22万文
英語共起辞書	46万句
英語コーパス	16万文



イ：日本語単語辞書　　ホ：日本語コーパス
ロ：日英対訳辞書　　ヘ：概念記述辞書
ハ：概念体系辞書　　ト：日本語テキストベース
ニ：日本語共起辞書

図３．３．２－１　EDR電子化辞書（日本語）による実現部分



A：英語単語辞書
B：英日対訳辞書
C：概念体系辞書
D：英語共起辞書
E：英語コーパス
F：概念記述辞書
G：英語テキストベース

図3.3.2-2　EDR電子化辞書（英語）による実現部分

3.3.3　日本語語彙大系（NTTコミュニケーション科学研究所）

日本語語彙大系[NTTコミュニケーション科学研究所 97]はNTTの研究所において研究開発されてきた日英機械翻訳システムALT-J/E（Automatic Language Translator-Japanese to English）の「翻訳辞書」の中の「日本語意味辞書」に関する部分を人間用の辞書として編集し直したものである。「日本語意味辞書」は日本語の意味解析を実現するために開発されたもので、日本語単語の「単語意味属性体系」、「単語意味辞書」、「構文意味辞書」からなる。

ALT-J/E「翻訳辞書」

- 「日本語文法辞書」：日本語単語の文法的情報
- 「日本語意味辞書」：日本語単語の意味的情報
  - 「単語意味属性体系」
  - 「単語意味辞書」

・「構文意味辞書」
・「日英対照辞書」：日英単語の語彙の対応関係
・「英語辞書」：英語単語の文法情報

「単語意味属性体系」は、日本語単語の意味的用法を３種類（一般名詞、固有名詞、用言）、合計約3,000種類の単語意味属性として体系化したものである。「単語意味辞書」は日本語単語の意味的用法を単語意味属性で定義したものである。一般名詞と使用頻度の高い固有名詞など、新聞記事の単語をカバーすることを目標に約40万語を対象にしている。「構文意味辞書」は日本語文型を用言中心とする結合価パターン（格パターンに類似）としてまとめたものである。一般の文型と慣用句表現の文型を合わせて約16,000件の日本語文型パターンである。併せて対応する英語文型パターンも収録している。文型パターンは単語意味属性を用いて記述されている。

「日本語語彙大系」は「日本語意味辞書」に対応して「意味体系」、「単語体系」、「構文体系」の３部から成り立っている。「意味体系」は「単語意味属性体系」を収録し、「単語体系」は語彙を一般的なもの約30万語に圧縮した「単語意味辞書」を収録し、「構文体系」は文型数を14,000件に整理した「構文意味辞書」を収録している。また、それぞれに人による利用の便宜のための情報が付加されている。

「日本語意味辞書」は日本語の意味解析に用いるための辞書である。ただし、ここでいう意味解析とは、意味上の多義性の解消を行なうことである。

（１） 日本語語彙大系（日本語意味辞書）の言語知識

① 意味体系（単語意味属性体系）

単語の意味的な用法に着目し、単語の意味属性を「一般名詞意味属性」、「固有名詞意味属性」、「用言意味属性」に分けて木構造に体系化したものである。対象の見方や捉え方は言語によって異なるため、言語ごとに異なった意味属性の体系が必要である。意味属性の分解精度をどの位にするかであるが、一般名詞に関しては日英機械翻訳への利用という目的から英語への訳し分けできることを目安にしてある。複合語等の解析にはより細かな分解精度が必要になるため、固有名詞に関しては、部分的に細分化した体系を用意した。

・一般名詞意味属性体系：一般名詞の意味的用法を表す2,710の意味属性を上位－下位関係（is-a関係）と全体－部分関係（has-a関係）による12段の木構造に体系化したものである。名詞の一般的用法を記述する。上位部分は以下のようになっている。

- 名詞
  - 具体
    - 主体
      - 人
        - 人間
        - 準人間
        - 人(職業・地位・役割)
      - 組織
    - 場所
      - 施設
      - 地域
      - 自然
        - 地勢
        - 宇宙
    - 具体物
      - 生物
        - 動物
          - 動物(個体)
          - 動物(部分)
        - 植物
          - 植物(個体)
          - 植物(部分)
      - 無生物
        - 自然物
          - 物質(部分)
          - 物質(本体)
            - 個体
            - 液体
            - 気体
        - 人工物
          - 物品
          - 資材
          - 薬品
          - 衣料
          - 食料
          - 建造物
          - 道具
          - 機械
          - 乗り物
  - 抽象
    - 抽象物
      - 抽象物(精神)
      - 抽象物(行為)

制度
成敗・業績
習俗
事
人間活動
精神
行為
事象
出来事
変動
自然現象
非生命現象
物象
気象・天象
生命現象
抽象的関係
存在
類・系
関連
性質
状態
形状
数量
場
時間

・固有名詞意味属性体系：固有名詞の意味的用法を表す130の意味属性を上位 - 下位関係によって9段の木構造に体系化したものである。複合語の解析などに利用する。上位部分は以下のようになっている。

固有名詞
地名
地域名
自然名
施設名
人名
姓
名
人物名
組織名
機関名
団体・党派名

学校名
国際組織名
その他の固有名詞
歴史名
文化名
民族・人種名
愛称等
その他の固有名詞(その他)

・用言意味属性体系：用言の意味的用法を表す36の意味属性を上位 - 下位関係によって4段の木構造に体系化したものである。体系は以下の通りである。

事象
状態
抽象的関係
存在
属性
所有
相対関係
因果関係
精神的関係
知覚状態
感情状態
思考状態
心的状態
身体状態
自然現象
行動
物理的行動
物理的移動
所有的移動
属性変化
身体変化
結果
身体動作
利用
結合動作
生成
消滅・破壊
精神的行動
精神的移動
知覚動作

感情動作
思考動作
使役
可能
開始
終了

② 単語体系（単語意味辞書）

一般語12万語、固有名詞20万語、専門用語５万語、その他の時事用語３万語の合計約40万語の日本語単語に単語意味属性を付与したものである。付与は、次のような基準に従って行われている。単語意味属性がその単語の表す概念の抽象度を越えないように、出来るだけ体系の下位にある意味属性を選んでいる。一般名詞には一般名詞意味属性のみを付与し、固有名詞には一般名詞意味属性と固有名詞意味属性を付与している。

③ 構文体系（構文意味辞書）

用言と名詞の結合関係の構造を結合価文法による文型として整理したものである。文型は一般表現文型と慣用表現文型に分けられる。文型パターンには対応する英語文型パターンが対応付けられている。

一般表現文型（一般文型パターン対）は各用言の持つ一般的は文型パターンを用言の字面をキーとする意味的結合価パターンとして表現したものである。パターンは用言の字面と一つ以上の格要素からなっている。格要素は名詞部分と格助詞で構成され、名詞部分には意味属性によって制約条件が記述されている。意味属性は、出来るだけ体系の上位のものを使用する。慣用表現文型（慣用文型パターン対）は一般表現文型と同様のパターン対であるが、一つ以上の格要素が意味属性ではなく、単語の字面で規定されている。

動詞、形容詞、形容動詞、6000語に対して、一般表現文型約13,000件、慣用表現文型約3,000件を収集している。しかし、翻訳実験の結果、かなり文型パターン対が不足であることが判明したため、現在、追加１万件を目標に作業が進められている。

(2) 言語知識の構造

日本語語彙大系の言語知識としての構造の特徴点をまとめる。

意味属性は、意味の要素となるものである。したがって、意味属性を表記する語彙や句は一つの意味要素だけを表し、体系は木構造となり、体系上の上位の意味要素は下位の意味要素の特徴をすべて包含する。意味属性を実体概念に対応付けることが出来る。構文体系における文型パターンは表層格で表現されており、関係概念という考

え方はとっていない。

3.3.4 WordNet

WordNet [Miller et al. 93] [WordNet URL]は米国プリンストン大学認知科学研究所（Cognitive Science Laboratory）において1985年から開発されたものである。もともとは心理言語学の研究のために実現されたもので、語の連想関係を中心に構成された一種のシソーラスである。現在は、自然言語処理への利用が試みられ、そのため表層レベルの辞書（言語知識）であるCOMLEX[Grishman et al.94]との対応付けが行われている。概念レベルが対象であり、語概念辞書の仕様にほぼそのまま対応する構造をもっている。

語彙は、名詞、動詞、形容詞、副詞、機能語（現在は含まれていない）に分け、体系化、ネット化が行われている。最も基本となるのが、同義関係（synonymy）にある語形（word form）の集合であるsynsetである。このsynsetが概念（実体概念）に対応する。Synsetどうしを様々な意味関係で結びネットが構成される。意味的関係には、階層、全体-部分、包含、反義などが含まれる。規模の概数は、文献[Miller et al.93]によると以下のとおりである。開発は続けられており、現在の規模は、さらに大きなものになっていると思われる。

	語数	Synset数
名詞	57,000	48,800
動詞	13,000	8,400
形容詞	19,500	10,000

(1) 名　詞

名詞のsynsetは上位（hypernymy）と下位（hyponymy）の関係によって継承関係を構成する。上位、下位を判定する基準は、以下のような心理言語学的事実に基づいて行われる。

① 上位名詞は下位名詞の照応名詞として理解される。
② 上位名詞と下位名詞が比較構文の中で用いられることはない。

名詞synsetの体系は、一本の木とせずに以下に示す25個のunique beginnerと呼ばれる基本synsetを出発点として構成された。

{act, action, activity}
{animal, fauna}
{artifact}
{attribute, property}
{body, corpus}
{cognition, knowledge}
{communication}
{event, happening}
{feeling, emotion}
{food}
{group, collection}
{location, place}
{motive}
{natural-object}
{natural-phenomenon}
{person, human-being}
{plant, flora}
{possession}
{process}
{quantity, amount}
{relation}
{shape}
{state, condition}
{substance}
{time}

これらのunique beginerは、作業が終わったところで次のような体系にまとめられた。

{thing, entity}
  {living-thing, organism}
    {plant, flora}
    {animal, fauna}
    {person, human-being}
  {non-living-thing, object}
    {natural-object}
    {artifact}
    {substance}
    {food}

概念であるsynsetをさらに定義するためには、示差的特徴が付加されねばならない。特徴としては、

① 属性(主に形容詞によって与えられる)
② 部分(主に名詞によって与えられる)
③ 機能(主に動詞によって与えられる)

などが考えられるが、現在は、②のみが関係として付与されており、他は通常の辞書のように短い自然言語文による注記よって説明されている。
部分（meronymy）、全体（holonymy）を示す全体 - 部分関係には、

① 一部：component-object
② 要素：member-collection
③ 材料：stuff-object

の３つのものが扱われている。また、反義（antonymy）関係も定義されている。

(2) 形容詞

形容詞は次の 4 種類に分類されている。

① 叙述形容詞：descriptive adjective（big, interesting, possible）
② 関係形容詞：relational adjective（presidential, nuclear）
③ 指示修飾形容詞：reference-modifying adjective（former, alleged）
④ 色彩形容詞：color adjective

形容詞は名詞のように上位・下位関係を構成することが出来ないので類義関係（similarity）と反義関係が基本的な関係となる。なお、関係形容詞は反義語が考えにくいので関連する名詞を添付し、意味の補強を行なっている。また、現れる統語的な位置に制限がある場合、その制限をsynset内の語形に付記してある。

(3) 動　詞

動詞は、15の意味分野に分けて構成されている。すなわち、身体動作、変化、認識、伝達、競合、消費、接触、作成、感情、動作、知覚、、所有、社会活動、状態、自然現象の意味分野である。動詞のネットを構成する基本となる関係は、名詞の上位・下位関係にあたる含意関係（entailment）である。含意関係とは以下のような定義である。

“Someone V1がSomeone V2を論理的に含意する時、V1とV2は含意関係にある。”

含意関係は、時間的な含意を含むかどうか、同時であるかどうかなどにより4種類に分けられている。また、統語的な特徴を示すものとして、各synsetに一つ以上の文型が付記されている。

3.3.5 EuroWordNet

EuroWordNet [Vossen et al. 97] [EuroWordNet URL]はECの研究プロジェクトの一つであるTelematics Applications ProgrammeのLanguage Engineering部門のプロジェクトである。1996年3月に3年間のプロジェクトとして始められた。英語、オランダ語、イタリア語、スペイン語に関するワードネットをリンク付けした多言語データベースの開発を目標にしている。英語に関しては、WordNetを採用する。各言語固有の部分を残して、共通となるトップレベルのオントロジーを開発する。University of Amsterdam, NL、Istituto Di Linguistica Computazionale Pisa, IT、Fundacion Universidad Empresa, ES、University of Sheffield, GB、Novell Belgium NV, BEの機関が参加している。開発は、Acquilex、Siftなどのプロジェクトの成果を利用して行われている。

各言語のワードネット内の概念は、ILI（Inter-Lingual-Index）と呼ばれる概念セットの最も近いものにリンクされている。ILIはWordNet1.5のsynsetを改良、拡張したものである。WordNet自身もこのILIにリンクされている。また、ドメインのオントロジーと上層概念のオントロジーがこのILIにリンクされている。ドメインは、スクリプト状に意味（概念）をグループ化したものである。上層概念は、すべてに共通な基礎的な概念である。上層概念（Top-Concept）のオントロジーは以下のようになっている。

- HighOrderEntity
  - Time
  - Static
    - SocialState
    - Relation
      - PossessionRelation
      - MeaningRelation
      - CausalRelation
    - PhysicalState
    - ModalState
    - MentalState
    - MentalObject
    - Measure
    - LocationState
    - ExistentialState
    - Condition

Phenomenon
WeatherPhenomenon
Manner
Dynamic
Stimulus
Sound
Represent
Perception
Operation
Motion
MentalAct
Management
Communication
Change
QuantityChange
PossessionChange
PhysicalChange
ExistentialChange
Causation
Caring
Behavior
Activity
Work
Education
Recreation
Fighting
Art
Aspect

FirstOrderEntity
Origin
Natural
Animate
Plant
Human
Occupation
Creature
Animal
Artifact
Form
Substance
Solid
Liquid

Gas
Object
Composition
Group
Part
Function
Covering
Vehicle
Symbol
MoneySymbol
LanguageSymbol
ImageSymbol
Software
Place
Instrument
Garment
Furniture
Container
Comestible
Building

参考文献

[横井ほか 96] 横井俊夫, 木村和広,小泉敦子, 三吉秀夫: 表層レベルにおける電子化辞書の情報構造, 情報処理学会論文誌, Vol.37, No.3, pp.333-344 (1996).

[横井ほか 97] 横井俊夫, 仲尾由雄, 荻野孝野, 田中裕一: 概念レベルにおける電子化辞書の情報構造, 情報処理学会論文誌, Vol.38, No.1, pp.32-43 (1997).

[松本ほか 97] 松本祐治, 影山太郎, 永田昌明, 斎藤洋典, 徳永健伸: 単語と辞書、岩波講座言語の科学第3巻, 岩波書店 (1997).

[郡司ほか 98] 郡司隆男, 安部泰明,白井賢一郎, 坂原 茂、松本祐治: 意味, 岩波講座言語の科学第4巻, 岩波書店 (1998).

[国広 98] 国広哲弥: 理想の国語辞典, 大修館書店 (1998).

[日本電子化辞書研究所 95] 日本電子化辞書研究所: EDR電子化辞書仕様説明書（第二版）, 日本電子化辞書研究所, TR-045 (1995).

[荻野ほか 97] 荻野孝野, 小林正博: 日本電子化辞書研究所における概念体系, 第 5 回国立国語研究所国際シンポジウム第 1 専門部会発表論文集 (1997).

[Ogino et al.97] Takano Ogino, Hideo Miyoshi, Fumihito Nishino, Masahiro Kobayashi and Junichi Tsujii: An Experiment on Matching EDR Concept Classification Dictionary with WordNet, IJCAI-97 Workshop on Ontologies and Multilingual NLP, pp.23-27 (1997).

[Utiyama and Hasida 97] Masao Utiyama and Koiti Hasida: Bottom-up Alignment of Ontologies, IJCAI-97 Workshop on Ontologies and Multilingual NLP, pp.35-40 (1997).
[NTTコミュニケーション科学研究所 97] NTTコミュニケーション科学研究所監修: 日本語語彙大系 全5巻, 岩波書店 (1997).
[Miller et al. 93] Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K. and Tengi, R.: Five Papers on WordNet, CSL Report 43, Cognitive Science Laboratory, Princeton University (1993).
[WordNet URL] http://www.cogsci.princeton.edu/~wn/
[Grishman et al. 94] Grishman, R., Macleod, C., and Meyers, A.: Comlexsyntax : Building a computational lexicon, Proc. 15th Int'l Conf. Computational Linguistics (COLING94) (1994).
[Vossen et al. 97] Piek Vossen, Pedro Diez-Orzas and Wim Peters: The Multilingual Design of the EuroWordNet Database, IJCAI-97 Workshop on Ontologies and Multilingual NLP, pp.41-47 (1997).
[EuroWordNet URL] http://www.let.uva.nl/~ewn/

3.3.6 SENSUS Project

(1) 概 要 [SENSUS]

機械翻訳やテキスト要約、情報検索などでは、テキストをより深く（意味的に）理解するために推論（reasoning/inference）が必要になるが、このような推論を行うためには、広い範囲に渡って(wide-ranging)のシソーラス（semantic thesaurus）、または、オントロジーを持ったシステムが不可欠となる。

今日、十分な大きさではないが、精巧で、矛盾のないにオントロジーが存在している。SENSUSプロジェクトの目標は、そのようなオントロジーを核に、既存の辞書、テキスト、他のオントロジー資源から得られる情報を利用して、かつ、機械翻訳や要約システムでテストや改訂を繰り返しながら、内容を拡充させる手法によりオントロジーを構築することである。なお、SENSUSプロジェクトは、ISIにおける自然言語処理プロジェクト（NLG-Projects）の一つである。

SENSUSは、約７万のノードからなる用語分類（terminology taxonomy）であり、知識の追加が可能な枠組み（framework）として構築されている。また、オントロジー間整合アルゴリズム（cross-ontology alignment algorithms）が多数開発されている。このアルゴリズムによって、例えば、オントロジーとオントロジー（または、辞書（lexicons））との間での項目（term）の同定ができるようになり、知識を移す（transfer）ことが可能となる。

現在、オントロジーの構築やその利用に関係しているAI/KR communityのメンバーによって研究が続けられている。ここで、注目すべき点は、SENSUS（Pangloss）、CYC、EDR、MIKROKOSMOS、および、他のオントロジーの上位部分（upper regions）を併合することによって、ANSIのための“標準的な（standard）”オントロジーを構築する試みである（IBM Santa Theresa、CYCorp.、Stanford University、EDR Tokyo、および、他の種々の研究機関の研究者の共同研究）。現在のところ、このプロジェクトには財源がない。

以下では、SENSUSの中心となっているPanglossオントロジーについて、その開発を中心に概要を報告する。

（2） Panglossオントロジー[SENSUS][KNIGHT]

PANGLOSSプロジェクトでは、大規模知識ベース機械翻訳システムが構築されている。Panglossオントロジーは、このシステムで意味処理を支援するために開発された大規模概念ネットワークである。このオントロジーは、約5万のシンボル（symbols）からなるタクソノミ（taxonomy）であり、言語中立な（language neutral）インタリンガ（Interlingua）で作成されている。また、これは、Loom、Framkit、および、Prologで書かれており、適当なアクセスルーチン（access routines）を用いて他のPangloss sitesへ分配される。

Panglossオントロジーは，次の3つの領域（region）に大別される。

① upper region：より抽象的な領域で、Ontology Base（OB）と呼ばれている。約400の項目から成り、言語処理に欠くことのできない本質的な（essential）概念（generalization）を含んでいる。OBの中には、意味情報として意味・構文パターン（semantic and syntactic pattern）が保存（capture）されているが、個々の語彙素に特有の処理で要求される構文情報も蓄えられている。なお、OBの機能、および、インターリンガ（Interlingua）との関係は、参考文献[HOVY 92]に記述されている。

② middle region：約5万項目で多くの英単語の語義を表している項目を含んおり、一般の世界モデル（world model）のための枠組を提供する。

③ lower （more specific） region：より専門的な領域で、異なるアプリケーション領域のためのアンカーポイント（anchor points）を提供する。

図3.3.6-1に、OB-Thingの概念階層の一部を示す[SENSUS]。

図３．３．６－１　An exampl of a hierafchy of "OB-THING"

(3) Panglossオントロジーの開発 [KNIGHT]

オントロジーのような大規模知識ベースをすべて手動で構築することは困難である。こんため、既存の資源（例えば、さまざまなオンライン辞書、意味ネットワーク、2カ国語辞書など）を半自動的に合併するいくつかのアルゴリズムが開発されている。また、ここでは、次に述べる5つの言語資源が利用されている。

① 言語資源（Lingusitic Resources）

(a) PENMAN Upper model （USC／ISIから）

Upper Model（Bateman 1990）は、約200のノード（nodes）からなるトップレベルのネットワークであり、システミック文法（Systemic-Functional Linguistics）に基づいている。LOOM知識表現言語（MacGrego 1988）で実現されており、PENMAN英文生成システムで使用されている。

(b) NTOSモデル（カーネギーメロン大学から）

ONTOS（Carlson & Nirenburg 1990）は、PENMAN Upper Modelと同様の規模のもので、機械翻訳を支援するために作られたトップレベルのオントロジーである。その中で、‘こと’概念構造（event structure）は、言語間に渡る動詞の研究に基いている。従って、格の役割と、格に入るものがどういう条件を満たすべきかの制約は、どんな特定の言語とも独立に与えられている。また、ONTOSモデルには、‘もの’概念（object）や概念階層（hierarchies）なども含まれている。

(c) Longman's Dictionary （LDOCE）（ニューメキシコ州立大学から）

LDOCEには、27,758の単語と74,113の語義が含まれている。語義文は約2,000語の‘制限された語彙’で記述されている。また、81の統語コード（syntactic code）、および、名詞に対しては、33の意味コード（semantic code）と124の語用論的コード（pragmatic code）を持っている。なお、LDOCEの意味コードは、PANGLOSSの一部としてニューメキシコ州立大学で手作業で構築されている中規模スペイン語辞書（ULTRA）の意味記述部に使われている。

(d) WordNet

WordNetは、単語の意味データベース（semantic word databese）であり、心理言語学に基礎をおいている。WordNetでは、同義語を“synsets”と呼ばれる1つの単位（single units）にグループ化している。名詞の意味は、深い階層に組織化され、さらに、データベースは、部分全体（part-of-links）や反意語（antonym links）なども含んでいる。約半分のsynsetsには、簡単な略式（informal）定義が与えられている。

(e) Collins Bilingual Dictionary

ハーパーーコリンズの２カ国語辞書（Harper-Collins Billingual Spanish-English dictionary （Collins 1971））は、数万のスペイン語の見出し語とその英訳を含んでいる。単語の英訳には、意味コードはマークされてないが、主題分野コード（例えば、Military [MIL]やCommercial [COM]などのような）がマークされているものがある。

② 言語資源の合併（Merging Resources：PANGLOSSオントロジーの構築方法）

図３．３．６-２に、上記の言語資源の情報を併合して、機械翻訳用オントロジーを構築する手順が示されている[KNIGHT]。

(a) PENMAN Upper ModelとONTOSを合併して、Ontology Base（OB）が作成される[Ⅰ]。これは手作業である。この構造には、インターリンガの格役割（case role）などが付加されるので、随時修正される。

(b) このOBとWordNetを併合して、大規模知識ベース（WordNet under OB）が得られる[Ⅱ]。これも手作業である。これに含まれるほとんどの概念には、WordNetの単語名（name）が付けられているが、中には、ONTOS、Upper Model、および、WordNetのそれぞれの単語名を持っている概念もある。この知識ベースには、PENMAN Upper Modelがそのままの埋め込まれているので、Ontology Base（OB）に基づいた分類（Proper taxonomization）でも、PENMAN本来の仕事は保証される。WordNetの下位の階層（subordination）では、ネットワークを約200の部分（pieces）に分割し、それぞれを手作業でOBと合併している。

(c) LDOCEの語義をWordNetの語義と併合する。この主な理由は、第一に、LDOCEには、WordNetで省かれている辞書情報（例えば、統語コードや主題分野コード、語義文など）が大量に含まれていることであり、第二に、LDOCEの意味コードが、PANGLOSSのインターリンガでは正規のトークン（legal token）であり、かつ、ULTRAでも利用されているからである。この併合のために、WordNetの概念とLDOCEの語義との間で意味の一致する組（一致対）を見つけ出し、結合する半自動化アルゴリズム（semi-automatic algorithm）が開発されている[KNIGHT]。この併合の結果と(b)での知識ベース（WordNet under OB）からオントロジー[Ⅲ]（Ontology & English Lexicon）が得られる。

(d) 最終段階は、大規模スペイン語辞書(オントロジー)[Ⅳ]を構築することである。このために、Collins biringual辞書を利用して、スペイン語の語義と英語の語義との同定を行うアルゴリズムが開発されている[KNIGHT]。これにより、(c)で得られるオントロジーにスペイン語でのインデックス（項目）付けが可能になる。

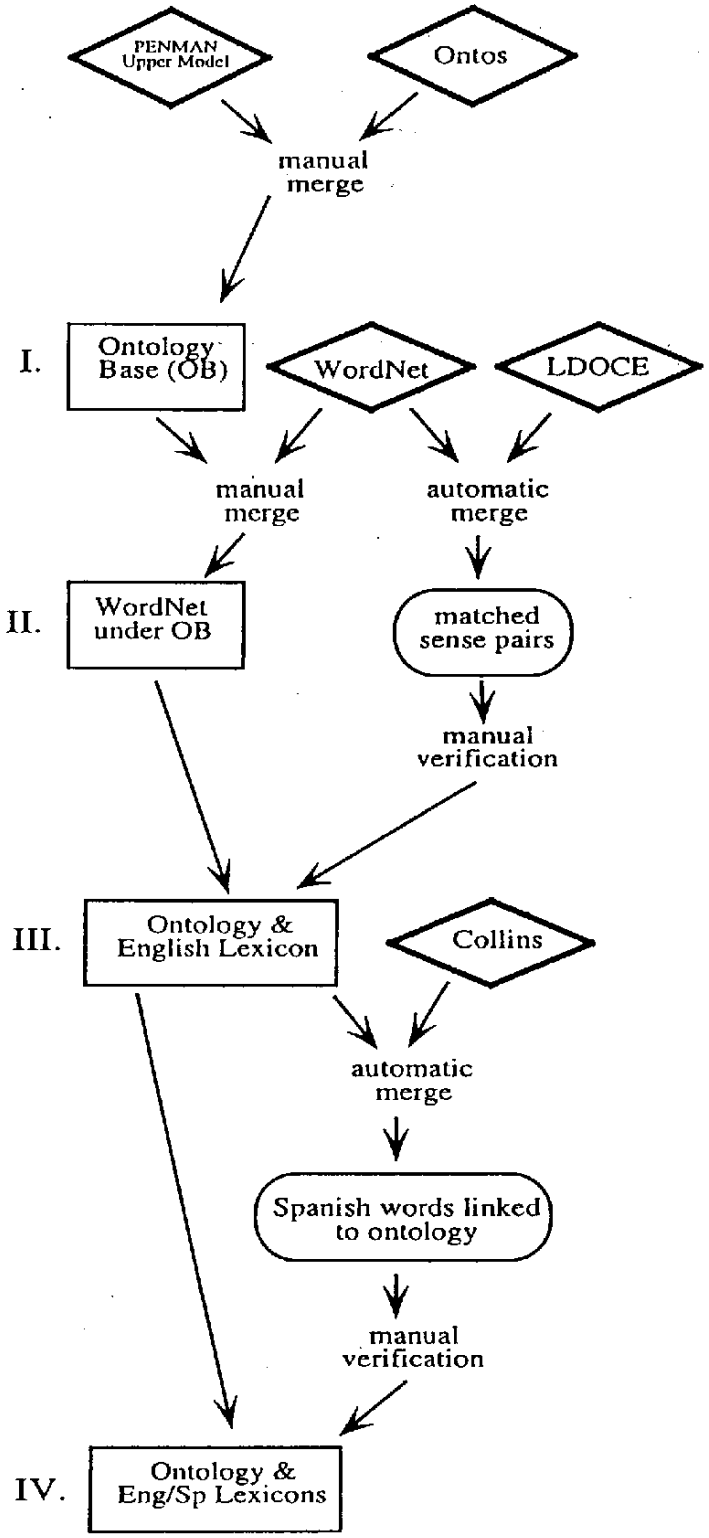


図３．３．６－２　Merging Information in Five Linguistic Resources to Build a Large Scale Ontology for Machine Taranslation

参考文献

[SENSUS] http://www.isi.edu/natural-language/resources/sensus.html

[KNIGHT] Knight,K. & S.Luk:Building a Large Knowledge Base for Machine Translation, Proc. of the American Association of Artificial Intelligence Conference AAAI-94

[HOVY 92] E.Hovy & S.Nirenburg:Approximating an Interlingua in a Principled Way, Proc. of the DARPA Speech and Natural Langusge Workshop, 1992

3.3.7 The Generalized Upper model（GUM）

(1) 概 要

Generalized Upper model（以下、GMDと略記する）は、領域（domain）とタスク（task）に独立な‘言語に動機付けられたオントロジー（linguistically motivated ontology）’である。精巧な自然言語処理を支援するために開発されているが、共有可能な知識資源（sharable knowledge resources）という側面を持っているとみなすことができる。また、このオントロジーは、表層言語表現（surface linguistic realization）と‘概念（conceptual）’（または‘文脈（contextual）’）表現（representaion）との中間の抽象化レベルにあるので、領域特有の知識（domain-specific knowledge）と一般的な言語資源（linguistic resources）との間のインタフェースとしても有用であると思われる。さらに、自然言語に関するところだけでなく、ドメインモデリング（domain modelling）にも根拠のある基礎が与えられることを期待している。

GMDは、テキスト生成に広く利用されている。最初は、英語の単一言語（monolingual）であったが、最近は、GMD／IPSI（KOMET and KPML）で開発された多言語生成環境（Multilingual Generation Environment）KOMET-PENMANで使用されている。このKOMET-PENMANテキスト生成システム（text generation system）では、英語、ドイツ語、オランダ語の首尾一貫したテキストが生成される。他の言語への拡張は開発中である。

概念の定義には、LOOMと呼ばれる知識表現言語（knowledge representaion language）が用いられている（ファイルとして、upper model専用のもの（GUM）とテキスト概念（textual concepts）用のもの（Text Base）とがある）。

(2) 開発の経緯

GUM 2.0は、Beatman et al.(1994&1995)で紹介されている初版のGUM（英語、ドイツ語、イタリア語）を発展させたものである。GUMは、最初、英語とドイツ語に対し

て、Penman Upper Model（Mann, 1985; Mann et al., 1985; Moor and Arens, 1985; Bateman et al., 1990）から始まり、Merged Upper model（Henschel, 1993; Hensch el and Beatman, 1994）と順次、派生して来ている。また、Pennman Upper modelは、PANGROSS OB（Ontlogy Base）の基盤にもなっている（３．３．６項参照）。

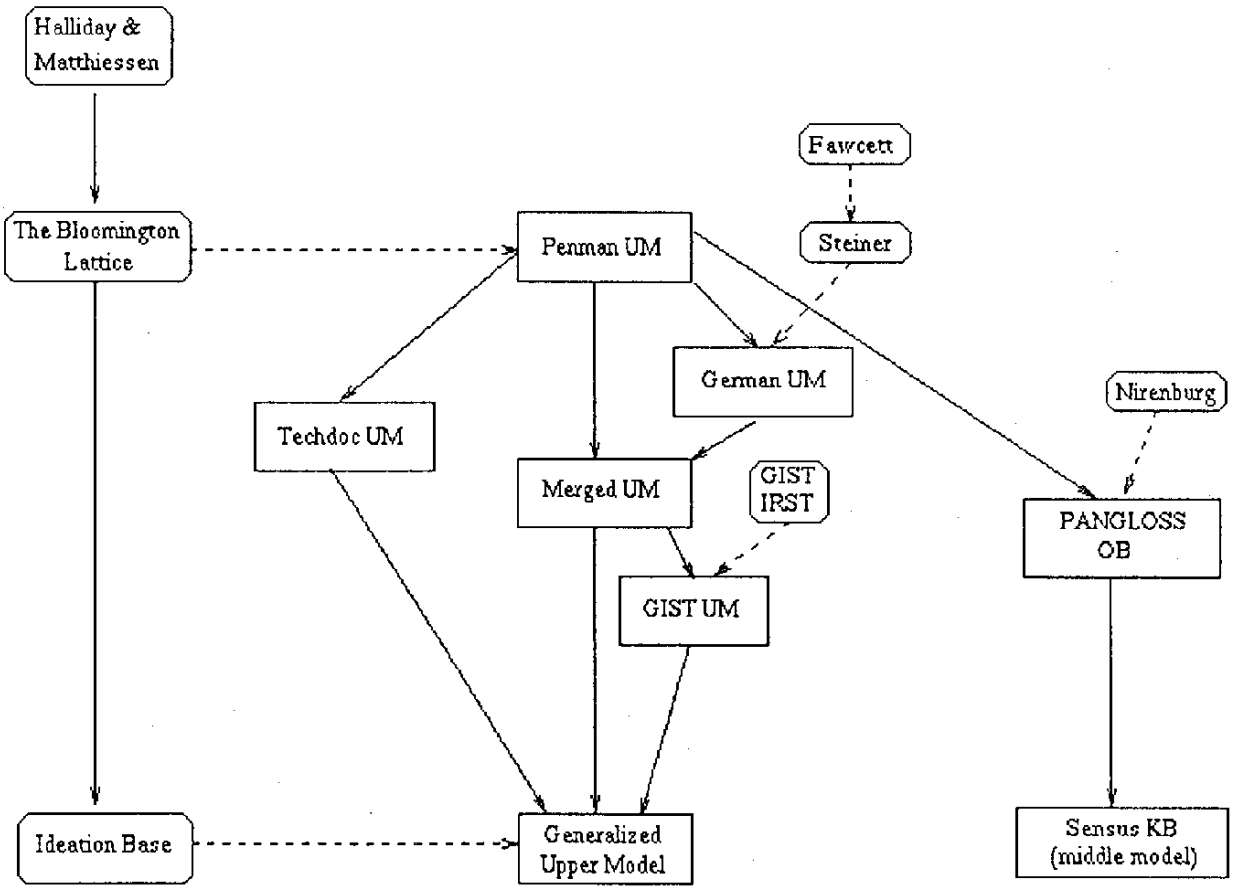


図３．３．７-１　The Development History of the Upper Model
(http://www.darmstadt.gmd.de/publish/komet/gen/node8.html)

最初のUpper Modelの具体化では、HallidayとMatthiessenのBloomington Latticeが主な影響力を与えている。そして、現在でも、理論的な側面からの基礎付けが行われており、Ideation Base(概念形成ベース)の方へ近付いて来ている。GMU開発の経緯と種々のオントロジーとの関係が、図.３．３．７-１に示されている。

最初のPenman Upper Modelは、意味レベルの内容を組織化したもので、grammatical semanticsが理論的な基準になっている。このような組織化に関して、いくつかの方法論が取られて来ているが、一般に、組織化される内容や用いられる基準に従って、いろいろなレベルでの方法論が考えられる。表３．３．７-１に、それらのほとんどが示さ

れている。表には、抽象化の段階に沿って、そこで採用されている方法論によるアプローチの例も与えられている。

なお、言語の組織化が‘知識’の組織化ついての情報を与えると言う考え方が、最近の言語学の中で復活して来ている。文法と意味論／意味（semantice/meaning）は深く関連し合っているので、文法の詳細な理論によって、意味レベルでの組織化に関しての知見が提供されことを期待している。

表３．３．７-１　Sveral methodologies for the organization and contents of ontologies (http://www.darmstadt.gmd.de/publish/komet/gen/node7.html)

nonlinguistic	reality	ontological－ ‘logical’	Weischedel(1989)
	knowledge	congnitive－ ‘psychological’	Langacker(1987)
		situational－ ‘socio/psycho-logical’	Steiner(fc)
linguistic	meaning	grammatical semantics inquiry semantics clause-based lexical semantics	Halliday & Matthiesen(fc) PENMAN UPPER MODEL Jackendoff(1983),LFG
		word senses word-based	Mel'cuk & Zholkovskij(1970)
	form	syntactic realization classes syntax	Steiner et al.(1987) LFG

（３）　the Upper Modelの階層（hierarchy）

Upper Modelの階層は、概念階層（concept hierarchy）と関係階層（relation hierarchy）の２つに分けられている。前者は、すべての概念を含んでおり、トップエントリー（top entry）は概念“um-thing”である。後者は、すべてのロール（roles）を含んでおり、トップエントリーはロール“um-relation”である。ロールは、概念を限定（modify）するために用いられる。これら二つの階層が、図３．３．７-２、図３．３．７-３に示されている。しかし、ここでは、組織化を目的にした概念階層に中心が置かれているので、適当なロールにリンクが張られている。

概念階層のトップノードである“um-thing”は、Upper Modelでは最も抽象的なエントリー（entry）である。これは、Halliday and Mathiessen（to appear）での“Phenomena（現象・事象）”または“Situation（状況）”に対応している。

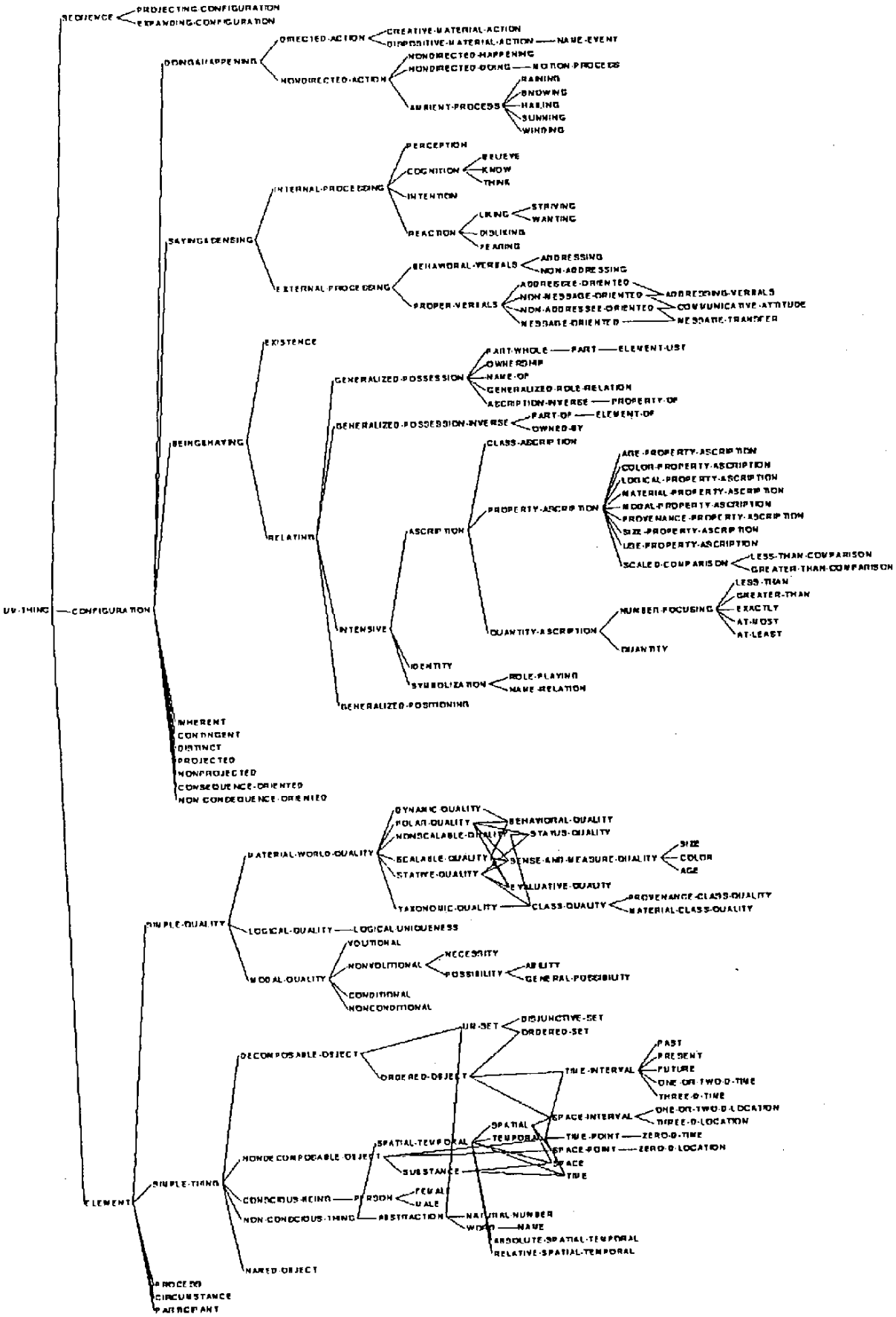


図３．３．７－２　The Concept Hierarcy

(http://www.darmstadt.gmd.de/publish/komet/gen/node10.html)

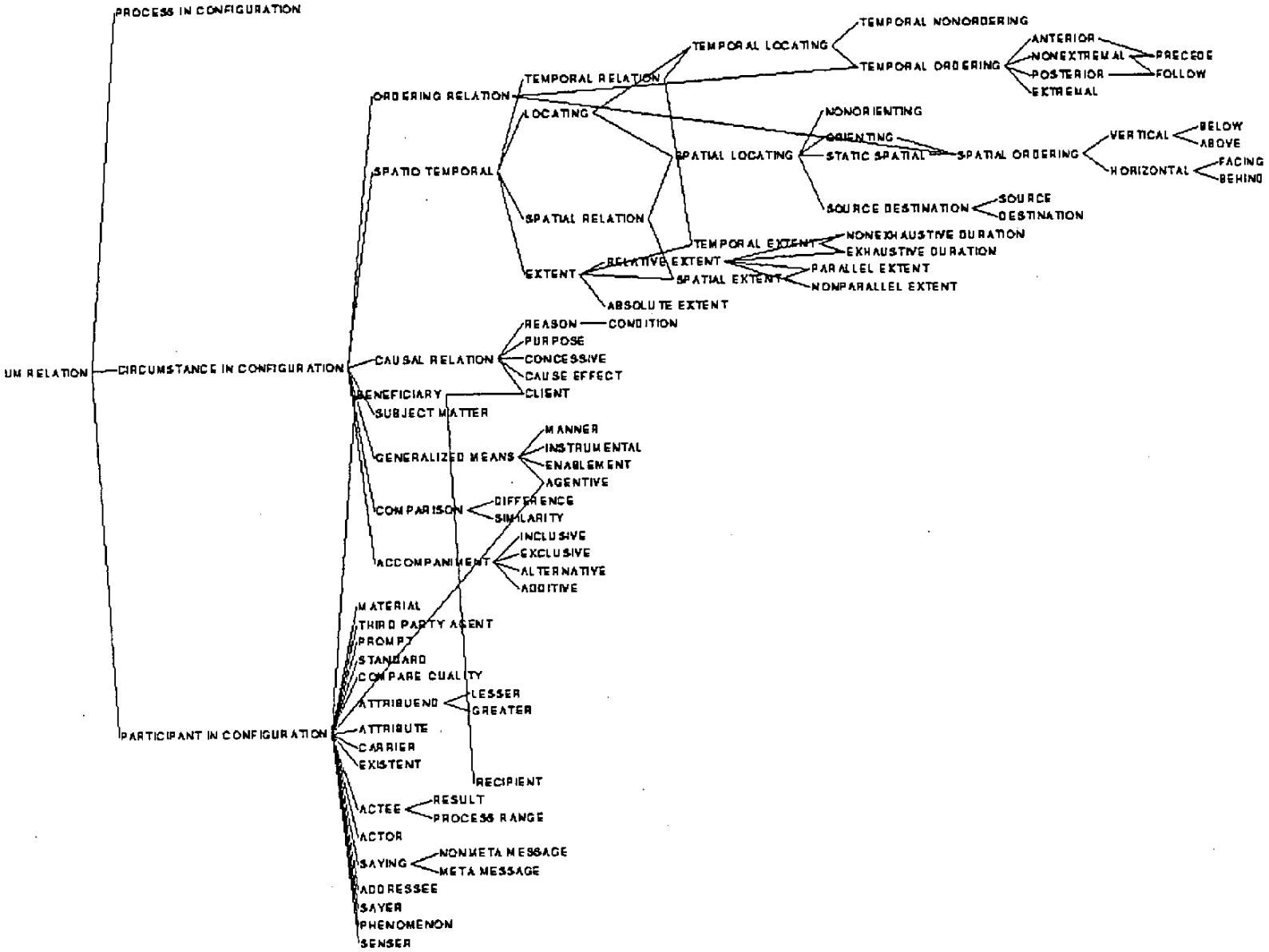


図３．３．７-３　The Relation Hierarcy

(http://www.darmstadt.gmd.de/publish/komet/gen/node11.html)

“um-thing” （“Phenomena”）には、次のような三つのサブタイプ（subtypes）がある。

①“configurayion”概念：活動（activity）または事態（state of affairs）に関係しているすべての要素（elements）のコンフィギュレーションとして。

②“element”概念：単体で、“独立している（stand-alone）”もの（object）または概念項目（conceptual item）として。

③“sequence”概念：種々の活動またはコンフィギュレーションが関係付けられて連鎖（sequence）を形成している複雑な状況（situation）として。

これらのクラスは、オントロジーの提案（ontology proposals）では比較的共通にみられる。ここで、これらに含まれる概念と名詞、動詞、形容詞／副詞のような言語表示との間の類似性がしばしば指摘されている：例えば、“Simple-Things”は名詞で表される‘ものごと(things)’であり、“process”は動詞に、“qualities”は形容詞や副詞に対応している。しかしながら、このことは、実際に与えらる意味分類が、表層的語彙分類（surface linguistic lexical classification scheme）とあまりにも安易に関連づけられるので、誤解される恐れがある。

類似性が役に立つのは、upper modelの中での概念（placement）と言語表現（classes linguistic realization）との間の意味的関係（sense of connection）が必要になった場合である。ここで、名詞、動詞、および、形容詞／副詞が、名詞語句（nominal group）、節（clause：文の一部であり、主語と述語動詞を具えている語群を指す）、および形容詞語句/副詞語句（adjectival/adverbal group）に置き換えられるならば、より確かな根拠に立脚していることになる。しかし、概念と言語表現（realization）との対応を可能にするためには、ほとんどの場合、より詳細な制約と拡張された範囲との両方が必要となる。

参考資料

[UMa] http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html

[UMb] http://www.darmstadt.gmd.de/publish/komet/gen-um/node1.html～node15.html

3.3.8 CYC

(1) CYCの概要

CYCは、大規模知識ベース（multi-contextual knowledge base）と推論エンジン（inference engine）の名称である。その開発は、ダグラス・B・レナート（Douglas B. Lenat）によって1980年代初期にテキサス州オースティンのMicroelectronics and Computer Technology Corporation（MCC）で始められた。現在、Cycorp（1995年に設立）に引き継がれている。

過去十数年にわたり、CYCチームのメンバーは、膨大な量の基本的な知識（human knowledge）、すなわち、日常生活の物（object）や事象（event）について推論するための事実、経験、およびヒューリスティクスなどを、知識ベースに蓄積している。

CYCでの試みは、大規模な知識ベースを記号的に構築しょうとしていることである。従って、統計的な手法に基づくものでもなければ、ニューラルネットワークやファジィ論理に基づくものでもない。CYCでは、すべての知識が論理的な表明（logical assertion）の形式で宣言的（declaratively）に表現されている。現在、40万個以上の表明（あるいは規則）が核として登録されている。それらは、事実についての簡単な言明（statement）、言明が真ならば帰結（conclusions）として何を導出するかについての規則、および、事実と規則のタイプ(type)によって決まる推論方法についての規則などである。新しい帰結は、演繹的な推論（deductive reasoning）を利用している推論エンジンによって得られる。

蓄積される表明が膨大になるに従って、相互に矛盾するものが含まれる可能性が高くなる。それを回避するために、知識ベースを大きく矛盾しない領域に分割する機構（マイクロセオリ）を導入している。

“常識”は、日常生活での基本的な知識であり、用語、規則、関係についての一種の意味的な土台（semantic substratum）となるものである。さまざまな知識集約的な（knowledge-intensive）産物とサービス（products and services）を可能にすると考えられている。また、CYCは、領域特有（domain-specific）のエキスパートシステムからでも自由に利用可能な“深い”層での定義や理解（definitions / understanding）を供給することも意図している。現在まで、CYCは、異質な（heterogeneous）データベースのブラウジングや統合、説明文がついている画像の検索、および自然言語処理などの領域で、草分けとして試験的な応用を可能にしてきた。

また、1996年から、CYCオントロジーの部分的な公開、一般知識ベースの拡張、CYCオントロジーの新しいアプリケーション（例：シソーラスマネージャ）の作成、他のオントロジー（例えば、WordNetなど）に対するリンクの作成などが行われて来ている。

以下では、公開されているCYCオントロジーに関する調査の一部について報告する。

（2） 上位CYCオントロジー（Upper CYC Ontlogy）の定数（用語）

Cycorpは、最も一般的な概念として、約3,000個の用語を公開している。これは、「上位CYCオントロジー」と呼ばれている。この上位オントロジーの下に、CYCの知識ベースが連結されている。この知識ベースには、詳細な事象に関する概念の膨大なデータが含まれている。それぞれの概念は、CYCの定数（constant）として表現されている。CYCでは、定数は用語（term）、あるいは、ユニット（unit）と呼ばれている。CYCの定数の名前（name）は、文字“#$”から始まっている。（例：‘#$Skin’）。定数は、集合（概念）、個々の物（individual object）、自然言語の中の単語、限量子、および、関係、などを表すことができる。

表３．３．８-１　CYCのUpper Ontologyでの乗数（項目）の４３分類
－　CYC Ontology Guide : Table of Cintentos より　－

- Fundamentals
  - Basic
    - #$Thing
    - #$Collection
    - #$isa
    - #$genl
    - ⋮
- Top Level
- Time and Dates
- Types of Predicates
- Spatial Relations
- Quantities
- Mathematics
- Contexts
- Groups
- “Doing”
- Transformations
- Changes Of State
- Transfer Of Possession
- Movement
- Parts of Objects
- Composition of Substances
- Agents
- Organizations
- Actors
- Roles
- Professions
- Emotion
- Propositional Attitudes
- Social
- Biology
- Chemistry
- Physiology
  - Animal Physiology Vocabulary
    - Specific Animal Body Parts
      - #$Skin
      - ⋮
- General Medicine
- Materials
- Waves
- Devices
- Construction
- Financial
- Food
- Clothing
- Weather
- Geography
- Transportation
- Information
- Perception
- Agreements
- Linguistic Terms
- Documentation

CYC知識ベースでは、概念階層の最上位の概念は‘#$Thing’であり、一番下には、例えば、個人を表すようなインスタンス（例えば、‘#$DougLenat’）が来る。上位CYCオントロジーは、その階層の中で比較的上位の概念から構成されている。表３．３．８-１に、これらの概念を４３に分類した項目を紹介する。後で言及される概念（‘#$Skin’など）がどこに分類されているのかも参考のために示している。

（3） CYC定数のエントリー

一般に、CYC定数のエントリー（entry）には、次のような情報が与えられている。

・定数（概念）の名前（name）：見出し語に相当する。
・コメント（comment）：概念の意図的な意味と使用についての英語の説明。
・階層情報（hierarchical information）：その概念を階層的に順序づけ、相互に連結させるために利用される。

例えば、定数の一つである‘#$Skin’のエントリーは、次のようになっている。

#$Skin
A (piece of) skin serves as outer protective and tactile sensory covering for (part of) an animal's body. This is the collection of all pieces of skin.Some examples include #$TheGoldenFleece (representing an entire skin of an animal) and (#$BodyPartFn #$YulBrynner #$Scalp) (representing a small portion of his skin).
isa:#$AnimalBodyPartType
genls:#$BiologicalLivingObject #$AnimalBodyPart #$SheetOfSomeStuff #$VibrationThroughAMediumSensor #$TactileSensor

最初の行に左寄せでエントリーの名前が来ている。コメントは名前の次の行から始まる。上例では、‘#$Skin’が、飛行機の外板や、葉の上皮ではなく、動物の外皮（皮膚）の一部（もちろん、全体の外皮（whole skin）も含まれる）についての概念であることを明らかにしている。さらに、この概念は、外皮のすべての部分の集合であることを表現している。

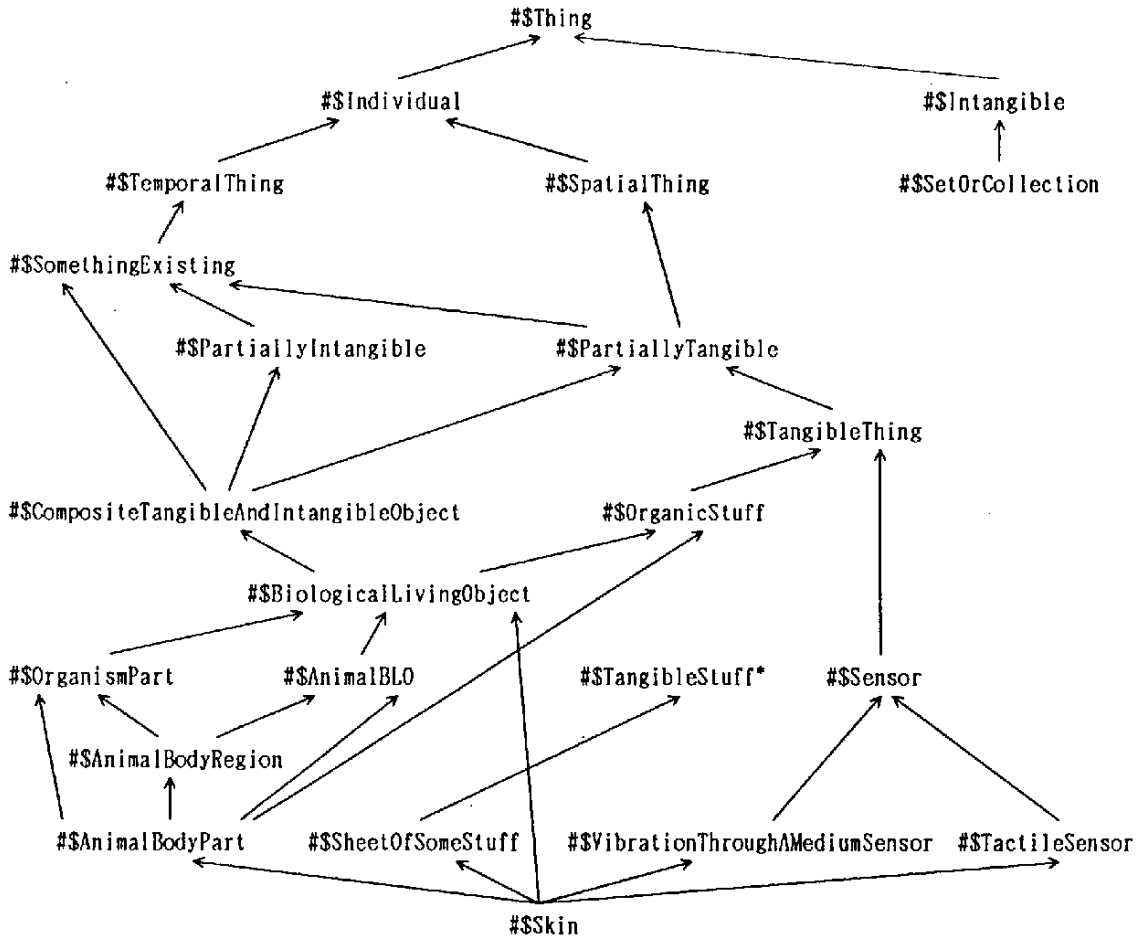
階層情報は、isa:やgenle:の行に書かれる（ただし、定数によっては、some subsets:の行を持っている場合があり、そこにいくつかの下位集合が記載されている）。isa:行には、エントリー概念を要素（element）とする集合が書かれる。例では、‘#$Skin’を要素とする集合として、‘#$AnimalBodyPartType’が一つ書かれている。また、genle:行には、エントリー概念の上位集合（superset）がいくつか与えられている。これらの集合は、CYCの定数であり、それぞれ次の条件を満たしている“z”である。すなわち、

isa:行の集合zに対しての条件；“#$Skin is an element of z.”(#$isa #$Skin z)
genl:行の集合zに対しての条件；“#$Skin is a subset of z.” (#$genle #$Skin z)

それぞれの末尾には参考のために、それぞれの条件に対応するCYC知識ベースの中でのCyclLの表記（prefix notation）が示されている。

一般に、これらの条件を満たす集合は多数存在する。そこで、実際に登録する場合には、集合間で成立する推移律を利用して、冗長性のないように考慮されている。

図３．３．８-１に示されている階層構造は、‘#$Skin’のgenl:行の上位集合（概念）から始めて、それらのgenl:行に現れる上位集合（概念）を順に検索することによって求められたものである。



注）*：この定数のエントリーは公開されてないようである.

図３．３．８-１　‘#$Skin’の上位集合から求められた定数（項目）の買いそう構造

ところで、CYCでは、共通な特性を持った物（things）の集合（set）、あるいはクラス（class）を表す概念として、‘#$Collection’を導入している。定数間の基本的な関係として、要素・集合関係を陽に取り入れているため、この概念は、特に重要と考えられる。なお、この要素・集合関係は、#&isaで定義されている。定数間の部分・全体関係を表す場合にもこの関係が用いられている。概念間の上位・下位関係は、#&genlで定義されている。

(4) CYCの述語 (Predicates)

述語のエントリーには、変項 (arguments：引数とも呼ぶ) が含まれている。各々の変項には、いくつかの制約が付けられている。一つは、それらの順序であり、もう一つは、変項の取る値に対する条件 (すなわち、どのようなクラス (class) の要素でなければならないか) である。なお、CYCでは、述語の名前の最初の文字は小文字で表記するようになっている。

例えば、述語 '#$mother' のエントリーは、次のようになっている。

#$mother:<Animal><FemaleAnimal>
(#$mother ANIM FEM) means that the #$FemaleAnimal FEM is the female biological parent of the #$Animal ANIM.
isa:#$FamilyRelationSlot #$BinaryPredicate

最初の行には、述語 '#$mother' には二つの変項があり、それぞれの変項は順に、集合 '#$Animal' の要素と集合 '#$FemaleAnimal' の要素でなければならない、という制約が示されている。isa:行には、これが類似関係 (kinship relation) で、かつ、2項述語であるという情報が示されている。

述語は、真偽を常に返す関数とも考えることができるので、真または偽として評価される表明を構築するために利用される。なお、'#$isa' も '#$genl' も共にCYCの2項述語である。

(5) CYCの関数 (Functions)

関数は、構文的には述語と同様である。しかし、述語が真か偽のどちらかを結果として返すのに対して、関数はなんらかの値 (すなわち、数、単語、集合、など) を返すようになっている。

例えば、関数 '#$FemaleFn' は、特定のタイプの生物 (organism0に対して、それのすべての雌のクラスを返す関数である。この関数のエントリーは次のようになっている。

#$FemaleFn:<OrganismClassificationType>
(#$FemaleFn TYPE) returns a collection which is the intersection of TYPE and #$FemaleAnimal.For example,(#$FemaleFn #$Person) is #$FemalePerson,and (#$FemaleFn #$Deer) is #$Doe.Notice that this function operates on,and returns as its value,ＣＹＣ collections, not English words.

Note:This function provides an implicit way of referring to
collections such as female platypi which may not have ---or merit---
a permanent atomic constant term being added to Cyc's ontology.
isa:#$CollectionDenotingFunction
arg1Genl:#$Animal
resultIsa:#$ExistingObjectType
resultGenl:#$FemaleAnimal

この関数の変項に集合#$Deerを与えれば、その値として集合‘#$Doe’が返される。このことは、返される集合が実際に命名（‘#$Doe’と）されてなくても、その代わりに、単に（‘#$FemaleFn #$Deer’）で済ますことが可能であることを示唆している。この考えに従って、#&Skinのエントリーの中では、（#$BodyPartFn #$YulBrynner #$Scalp）を使って、‘#$YulBrynnerScalp’の代りにしているわけである。ここで、関数‘#$BodyPartFn’は、２つの変項をもっている関数として定義されている。

なお、関数によっては、返される値が集合である時、それがまた、別の集合の下位集合であることを要求される場合がある。関数の値について、２つのタイプの制約（すなわち、それが何の要素であり、そして、それが何の下位集合であるか）がある場合は、それぞれの制約は、述語‘#$resultIsaと#$resultGenl’を用いて記述される。それによって、ファイルの中で、それぞれ、resultIsa:とresultGenl:で始まる行にリストアップされる。例えば、上記の関数‘#$FemaleFn’では、resultGenlは、関数によって返される値が、‘#$FemaleAnimal’の下位集合でなければならないということを要求している。また、arg1Genlは、変項が、‘#$Animal’の下位集合でなければならないという、必要条件を課している。

(6) 上位CYCオントロジーの応用に関して

上位CYCオントロジーとして公開されている約3,000個の概念は、CYC知識ベースのほんのわずかな割合でしかない。この知識ベースの中には、現在、何千もの異なった述語、スロット(slots)、関係、および、関数などが含まれており、それらによって、上位CYCオントロジーの概念のみならず、それ以外の何万もの概念が互いに連結されている。従って、3,000個程度の概念では少な過ぎると思われるかもしれないが、これらの概念は基本的な語彙であり、しかも詳細なデータが付加されているので、次に示すようなアプリケーションに対して、ある程度の基礎付けが可能であると思われる。

・自然言語理解と生成、
・意味データベースの統合、無矛盾チェック、調査

・意味情報検索
・オントロジー制約の基での（ontology-constrainted）シミュレーション
・ユーザモデルの構築と利用
・独立したグループ作業による知識共有

参考資料

[CYC] http://www.cyc.com/public.html, http://www.cyc.com/documentation.html, http://www.cyc.com/cyc-2-1/ 等々

[Lenat 95] D.B.Lenat: Artificial Intelligence, SCIENTIFIC AMERICAN, pp.62-64, September 1995.

3.3.9 CoreLex

CoreLexは、生成的辞書（Generative Lexicon）の提唱者の中心人物であるアメリカBrandeis大学のPaul Pustejovsky の指導のもとに、Paul Buitelaarが、仕上げた博士論文CoreLex:Systematic Plysemy and Underspecification (1998)に基づいたシステムである。126個の未指定意味タイプと、317個の意味的多義クラスの体系を立てて、39,937個の実際の名詞に対応させている。オンラインでも公開されている：http://www.cs.brandeis.edu~paulb/CoreLex/corelex.html

(1) CoreLexの特徴

CoreLexは、ある意味で3.3.10で述べるIPALとは全く異なる記述方式、言い替えると対応付けを行っている。IPALでは、辞書を作成するにあたり、当初はは多くの市販の辞書の語義分けも参照しながら作業を進めたが、語義分けに関しては、分けられるものはまず分けておくという方針を取った。また、語義間には、基本的なもの派生的なものなどの相互関係が当然考えられたが、その関係を記述に反映させることは敢えて行わなかった。その作業を全体にわたって安定したものにすることが難しいと考えたからだった。結果として、かなりの余剰性を残した現在のIPALが出来上がった。

CoreLexの方は、各語彙項目が持つ多義性には、法則性があることを前提にして、WordNet1.5で公開されている語彙項目に、仮定した126個の未指定意味タイプと317個の意味的多義クラスの対応付けを試みたものである。言い替えると、IPALにも、126個の未指定意味タイプと317個の意味的多義クラスが、何らかの形で反映されていることが予想されるが、CoreLexでは、そこを捉えることを主目的にしているが、IPAL

ではそこはあからさまには記述されていないということである。

(2) CoreLexに関連する問題

この点は、まだCoreLex自体を詳細に検討できていないので、現時点では確定的なことは言えないが、自然言語の特質を考えると、本来存在してもいい語彙項目が存在しなかったり、ある対応関係が使われていないということがどういうことなのかを検討する必要が言語学的な観点からは出てくるだろう。

3.3.10 日本語計算機用辞書IPAL

筆者（外池）も長年関わった計算機用日本語辞書IPALについて簡単に説明する。説明には、http://galaga.jaist.ac.jp:8000/pub/tools/hipal/manual/を参照させていただいた。IPALに関する詳しい情報は、ホームページに公開されているので、そちらを参照されたい：http://www.ipa.go.jp/STC/NIHONGO/IPAL/ipal.html。

IPALは、特別認可法人情報処理振興事業協会（IPA：Information-technology Promotion Agency）技術センターにおいて、村田賢一氏をプロジェクトリーダーとして昭和５６年に始められた。計算機による日本語処理を目的として、「ドキュメント作成のための日本語辞書の調査」および「計算機理解のための日本語辞書の調査」という名前のプロジェクトで試作された辞書である。IPAL辞書は動詞、形容詞、名詞の３つの辞書に分かれ、動詞、形容詞、名詞の順に完成した。各辞書は、市販の辞書とは異なり、より適切な語義記述を目指すのではなく、見出し語ごとに記述項目として取り上げた点についてどう振る舞うのかを詳細に記述している。

(1) 日本語基本動詞辞書IPAL

IPALのプロジェクトでは、文の中核は動詞的な要素であると考えて、まず日本語の動詞のうちで語彙体系からも使用頻度からも重要であると考えられる基本的な和語動詞861語を選び、それぞれのどのような特徴を持つかを詳細に記述することにした。結果として、意味および統語的特徴に基づいて下位区分し、それを１つの単位として、意味、形態、統語、文法的カテゴリ、慣用表現などに関わる情報をできるだけ詳細に記述したものになった。記述対象となった語の数は少ないが、多くの具体例を基に、記述の体系の検討・整備を進めたので、記述の余剰性が残っているものの、それまで無かった辞書が出来上がった。記述された項目は以下の通りである。

IPAL動詞辞書の記載情報

1. 見出し情報
   エントリ　　：動詞終止形のひらがなによる表記
   同音異義番号：同音異義語を区分するための番号
   サブエントリ：何番目の下位区分かを示す番号

2. 意味情報
   意味記述　　：語彙的意味
   関連語　　：上位語、類義語、反義語
   シソーラス　：既存のシソーラスの分類名称、コード
   意味分類　　：動詞の意味分類

3. 形態情報
   活用　　：語形変化の型
   語幹　　：語幹のローマ字表記
   表記　　：漢字と送り仮名による表記
   自他　　：自動詞、他動詞の区分
   異音同　　：サブエントリの異形態
   派生　　：助動詞・接辞が接続した形
   転成　　：他の品詞への転成語

4. 統語情報
   文型　　：動詞がとる格形式のパターン
   文例　　：その文型の文例
   述語素　　：動詞の統語的特徴
   意味素性　：名詞句の意味分類
   名詞句　　：名詞句の例

5. ヴォイス
   サセ形　　：ヲ使役、ニ使役の有無
   ラレ形　　：ラレ形の有無とその意味
   格形式の交　：受動になる時の格形式の交替
   タイプ　　：ヴォイスによる動詞の分類コード

6. テンス・アスペクト
   ル形　　：ル形の有無とその意味
   テイル形　　：テイル形の有無とその意味
   その他の形式：その他の形式の有無

7. ムード
   命令形　　：命令形の有無とその意味
   意思形　　：意思形の有無とその意味
   その他の形式：その他の形式の有無
   タイプ　　：ムードによる動詞の分類コード

8. その他の情報
　備考１　　：1.～3.に関して特記すべきこと
　備考２　　：4.～7.に関して特記すべきこと

（2）　日本語基本形容詞辞書IPAL

IPAL形容詞辞書は、基本的な形容詞136語を取り上げ、形態、意味、統語、慣用表現などに関する情報を記述したものである。以下に形容詞辞書で見出し語ごとに記載されている情報を示す。

IPAL形容詞辞書の記載情報

1. 見出し情報
　見出し語　：ひらがなによる表記
　語義番号　：語義による下位区分を示す番号
　語義数　　：語義の総数
　区分番号　：文型、述語素による下位区分を示す番号
　区分数　　：文型、述語素による下位区分の総数

2. 形態情報
　表記　　：漢字と送りがなによる表記
　語幹　　：ローマ字による表記
　語尾　　：－い、－な、－だのうち接続可能なもの
　異音同語　：例:さびしい→さみしい
　派生語　　：例:たかさ、さびしげ、いたむ
　複合語　　：例:あまざけ、よしあし

3. 意味情報
　意味記述　：語義の簡単な記述
　関連語　　：同義語、類義語、反義語
　意味の分類

4. 統語情報１（終止用法）
　文型　　：例:ＮＰニＮＰ１ガ
　名詞の意味素性：例:ＨＵＭ、ＡＢＳ
　文型に現れる名詞句
　述語素　　：例:ＰＡ、ＰＶ、ＦＣ
　文例

5. 統語情報２（連体用法）
　制限用法１：例:赤い花（この花が赤い）
　制限用法２：例:近い将来（＊将来が近い）

6. 統語情報3（連用用法）
　－くなる、－くする：例:赤くなる、赤くする
　動詞　　：例:赤く塗る、短く切る
　その他　：例:素晴らしく早い

7. その他の情報
　慣用表現　：例:面の皮が厚い
　備考

(3) 日本語基本名詞辞書IPAL

IPAL名詞辞書は、1081語（第3.1版）の名詞を記述対象としている。見出し語の語義ごとに区分情報、形態情報、項の用法、意味情報を与え、連体・被連体用法、サ変用法、述語用法はこのような用法がある場合にのみ与えられている。

IPAL名詞辞書の記載情報

1. 形態情報：品詞ごとに合成語（複合語、派生語）を記入する

例：見出し語	表記	合成語
麻	あさ	マニラ麻
麻	ま	大麻

2. 項の用法：品詞ごとの共起単語を記入する（共起動詞、共起形容詞、共起名詞）

例：見出し語	格形式	共起単語	品詞
麻（あさ）	&が	枯れる	動詞
麻（あさ）	を	栽培する	動詞
麻（あさ）	が	豊富だ	名詞
麻（あさ）	が	少ない	形容詞

3. 意味情報：関連語を記入する（同義語、類義語、対語）

例：見出し語	同義語	類義語	対語
不景気	不況		
景色		光景、見晴らし	
豊作			凶作

4. 連体・被連体用法：見出し語が「の」を介して修飾語をつくる場合とサ変動詞用法、形容動詞的用法の文型にあるNPがくる場合はその例を記入する。

例：見出し語　用例
　　軽率　　な行動、な言葉、な人
　　安心　　の／な様子、の／な設計

サ変用法：サ変用法のある見出し語はその例を記入する。
例：栽培する

述語用法：見出し語が例１、２の用法がある場合、記入する。
　例１→カキ料理は広島が本場だ
　例２→カキ料理の本場は広島だ

記述対象の名詞1,081語を選んだ基準は、以下の通り。

＜文法的特徴の基づくもの＞
　サ変動詞用法のあるもの
　サ変動詞用法のあるもの
　形容詞転成名詞
　形容動詞語幹といわれるもの
　関係性のあるもの（相対名詞といわれるものなど）
　副詞用法のあるもの・自立性の低い名詞といわれるもの
　関係性のあるもの（相対名詞といわれるものなど）
　「述語用法２」のあるもの
　「述語用法２」のあるもの
　「連体被修飾用法２」のあるもの
　動詞転成名詞

＜意味的分類に基づくもの＞
　植物
　食べ物
　身体
　動物
　場所
　乗り物
　洋装品
　仕事・役職
　自然物

自然現象
組織・団体

(4) 日本語計算機用辞書IPALの統合

IPAL辞書の３つの辞書が、北陸先端科学技術大学院大学の望月源氏、梁慶昇氏、奥村学氏によってハイパーテキスト化されている。http://galaga.jaist.ac.jp:8000/pub/tools/hipal/manual/ に詳しい情報が公開されている。

(5) 日本語計算機用辞書IPALで採用された意味素性

IPALの記述体系の中でも、オントロジーと直接関係して重要なのは、語義の区分など記述の基準として重要な役割を担った意味素性の体系だろう。IPALの動詞、形容詞、名詞の各辞書で採用された名詞の意味素性の数は、それぞれ19、42、58であり、以下に挙げる通りである。数が違うということは、意味素性の不統一性が存在していることを表わしている。単に数が異なるというだけではなく、階層構造も異なるという点が重要である。この不統一が生じたそもそもの原因は、動詞辞書編纂の頃から、形容詞、名詞への拡張を考慮していなかったということにある。

しかし、名詞辞書が完成していく過程で、３つの辞書の意味素性を対応付けようという試みが行われた。その対応表が、『計算機用日本語基本名詞辞書IPAL－解説編－』pp. 297-309にある。

動詞編：（『計算機用日本語基本動詞辞書IPAL－解説編－』p. 30. 表６）

CON: concrete　具体名詞
ANI: animate　動物
HUM: human　人間
ORG: organization　組織・機関
PLA: plant　植物
PAR: parts　生物の部分
NAT: natural　自然物
PRO: products　生産物・道具
PHE: phenomenon　現象名詞（自然／生理）
ABS: abstract　抽象名詞
ACT: action　動作・作用
MEN: mental　精神
LIN: linguistic products　言語作品
CHA: character　性質

REL: relation　関係
LOC: location　空間・方角
TIM: time　時間
QUA: quantity　数量
DIV: diverse　（制限緩やか）

形容詞編：（『計算機用日本語基本形容詞辞書IPAL－解説編－』p. 30. 表6）

●ABS(ABSTRACT)
▼KND(KIND)
REL(RELATIONAL TERMS)
SOC(SOCIAL BONDS)
AFF(AFFECT)
EFF(EFFECT)
▼NOR(NORMS/RULES/SCIENTIFIC SUBFIELDS)
▼INF(INFORMATION)
▼ENT(ENTITY)
▼MEA(MEASURE UNITS/MEASURABLE ENTITIES)
FOR(FORMS/STRUCTURES)
APP(APPEARANCE)
GRA(GRADABLE)
MAN(MANNER)
PER(PERSONALITY)
DUR(DURATION)
DIS(DISTANCE)
QUA(QUANTITY)
●TIM(TIME)
▼PIT(POINT IN TIME)
ORD(ORDINAL)
●PHE(PHENOMENA)
▼NAT(NATURAL ENTITIES)
▼PRC(PROCESS)
ACT(ACTIVITY)
EVE(EVENT)
▼STA(STATE)
●SPA(SPACE)
▼LOC(LOCUS)
▼ORG(ORGANIZATION)
●CON(CONCRETE)
▼AUT(AUTOMATA)
▼EDI(EDIBLE)
▼SOL(SOLID)

▼LIQ(LIQUID)
▼PAS(PASTY)
▼GAS(GAS)
●ANI(ANIMATE)
▼GAT(CONGREGATION)
AML(ANIMAL)
HUM(HUMAN)
CON/ABS 制限なし

名詞編の意味素性に関して詳しくは、『計算機用日本語基本名詞辞書IPAL－解説編－』p. 262-296を参照。全体として以下の階層を与えている。

/ANI/ 動物の領域
/CON/ 具体物の領域
/SAP/ 場所の領域
/PRC/ 出来事および動作／作用の領域
/ABS/ 抽象の領域

さらに詳しくは、動詞編から最後の名詞編まで、意味素性の体系作りを担当された青山文啓氏の解説「意味素性」（『計算機用日本語基本名詞辞書IPAL－解説編－』第1部、第3章、pp. 31-61.）を参照されたい。

(6) 日本語計算機用辞書IPALの問題点

3つの辞書の間の意味素性の不整合という問題以外に、次にあげる点が重要だと考えられる。

① 記述対象語の数が少ない
ありうべき辞書のプロトタイプを示すことを目標に作られたという性格上、また作業量の多さから言って、仕方のないことではあるが、実用にたえるようにするためには語数の拡張が望まれる。

② 離散的な値で記述したことから来る問題
記述項目ごとに、離散的なタイプとしての可能な値を特定して作業を行っている。例えば、動詞で、命令形が命令の意味になるのか、願望の意味になるのかなど。離散的な値を想定することで、品質が安定した面と、可能な値の内のどれかを判断せざるをえなかったので、捉え切れなかったところがある可能性がある。

③静的な記述に終っている

我々の言語使用の実態を考えると、辞書に関する知識は、確定的なものではない。その点が静的な記述では捉えられない。

3.3.11　その他の国内の動向

(1)　言語データに基づく語彙知識ベースの構築に関する基礎的研究（鶴丸研）

①研究目的

自然言語処理への利用を目指したオントロジー（ontology）の開発が進んでいる。日本語に関しては、EDRなどいくつかの電子化辞書の開発についての報告がある。しかし、大規模な知識の収集に関しては、現状では手作業が主体であるため、計算機支援システムの開発が不可欠となっている。我々は、電子化された言語データ、特に辞書データから(半)自動的に語彙知識を獲得するための手法を明らかにし、語彙知識ベースの構築を目指した研究を進めて来ている。

ところで、自然言語処理の分野では、これまで、シソーラス、意味辞書、語彙知識ベースなどの用語が用いられてきており、オントロジーという用語（概念）が使用されることは少なかった。しかし、我々が考えている語彙知識ベースは、基本的には文献[溝口95]で定義されているオントロジーの一つと捉らえることができると思われる。ここで、自然言語で対象とする語彙は膨大であり、しかも一般に単語は多義であるため、機械処理向きにその意味の記述を行うことは、非常に困難な問題を含んでいる。従って、オントロジー工学の確立を切に期待すると共に、我々の研究がその確立へささやかでも貢献できれば幸いである。

②研究経緯

(a) 国語辞典を利用した語彙知識ベースの構成に関する基本的な考え方を提案している。ここでの語彙知識ベースとは、単語の表す概念間の上位・下位関係を基盤に個々の語（概念）の意味の定義を明確にしたものを想定している[Yoshida 82]。ボトムアップによる手法が重要であろう。

(b) 国語辞典の語義文から単語（概念）間の階層関係（上位・下位、部分・全体関係など）を抽出するためのアルゴリズムについて論理的な側面から検討し、その有効性を評価するための計算機実験を試みている。

(c) 前項の成果を利用して得られた関係付けデータに基づいて、シソーラスを試作し、

いくつかの基本的な問題点（極大語の存在、同位語間の不統一、下位語の意味分類など）についての検討を進めて来ている。このために、シソーラス作成・更新支援システムを開発している。

(d) 部分・全体関係を3つのタイプに分けて論理的な定義を与え、これらが自然言語（語義文）では特に区別されずに使用されている場合があることを指摘した。また、上位・下位関係との融合により部分・全体関係を拡張するためのアルゴリズムについて検討した。

③ 研究課題

(a) 語彙知識ベースでは、単語間の階層関係の他に、上位語と下位語がどのような見方からの関係か、また、同じレベルの語（同位語）がお互いにどのような関係になっているか、なども求めておく必要がある。このような語彙情報の一部は、語義文や用例から求めることが可能である。ここでの問題は、語義文や用例は、自然言語で記述してあり、省略も多く、人手に頼らざるを得ない部分が存在することである。しかし、大量のデータを処理する必要があるため、支援システムの開発が不可欠となる。

自然言語文の解析ツール（形態素解析、構文解析など）の開発、計算機上での語彙知識の表現法（データ構造）などについての研究も大切である。なお、国語辞典は使用者のある程度の知識を前提に編集されているわけであるから、それを機械処理に利用する場合には、暗黙の前提とされている知識としてどのようなものがあるかの解明も望まれる。

(b) (a)とも関連した課題であるが、概念間の（意味的）関係についての厳密な検討が必要である。例えば、部分・全体関係についての考察は興味のある課題の一つである[Winston 87]。我々は、部分・全体関係を3つのタイプに分けて論理的な定義を与え、これらが自然言語（語義文）では特に区別されずに使用されている場合があることを指摘した。また、上位・下位関係との融合により部分・全体関係を拡張するためのアルゴリズムを提案しているが[鶴丸 93]、これらについてもさらに調査・検討したい。

(c) シソーラスの試作では、現在のところ、一つの辞書（新明解国語辞典）から得られた情報しか使用していない。このため、一つの閉じた系として種々の考察ができる利点はあるが、情報の欠落などによる不備な点も存在する。例えば、階層構造上での単語間の跳躍や、極大語の存在などである。また、その辞書特有な意味しか記述されてなかったりする場合もあると思われる。従って、複数辞書からの語彙情報の獲得と統合化が重要な課題となる。そのためには、語義文で記述された語義の類似度の評価が重要となろう。

(d) シソーラスの応用に関して、概念階層における上位概念の持つ属性の下位概念への伝播に関した問題などについても検討する必要がある。シソーラスでは一般に、複数上位語を許している。そのため、複数パスが生じた場合に上位概念からの属性の継承に対して冗長性や曖昧性が問題となる。どのようにして最適なパスを優先的に選択したらよいかなどは、推論と関連した重要な課題である。

(e) 動詞、形容詞、抽象語などの語彙知識の獲得と統合化。

(f) 一般分野の概念体系と専門分野の概念体系との有機的な統合システムの開発。

参考文献

[Yoshida 82] S.Yoshida, H.Tsurumaru, T.Hitaka :MAN-ASSISTED MACHINE CONSTRUCTION OF A SEMANTIC DICTIONARY FOR NATURAL LANGUAGE PROCESSING, Proc.of COLING' 82, pp.419-424, 1982.

[Winston 87] Winston, M.E., Herrmann, D.：A Taxonomy of Part-Whole Relation, Cognitive Science, Vol.11, pp.417-444, 1987.

[鶴丸 93] 鶴丸弘昭, 前田英幸, 山本和博, 日高達, 吉田将：国語辞典に基づくシソーラスの構築に関する一考察, 信学技報, NLC93-58, Vol.93, No.367, pp.29-36, 1993.

[溝口 95] 溝口理一郎：オントロジー工学序説, 人工知能学会誌, Vol.12, No.4, pp.559-569, 1997.

(2) 日本語生成的辞書プロジェクト

① 日本語生成的辞書プロジェクトのあらまし

3.3.9で取り上げたCoreLexと関連するプロジェクトを外池らが中心になり進めている。すでに関係する研究者により学会等でワークショップなどを行ってきたが、「WWW上での生成的辞書の構築：辞書を中心とした日本語文法の構造化」（平成10年度-12年度科学研究費基盤研究(B)）[GLEXICON] が認められたので、その内容を紹介する。

IPAL（IPA Lexicon）を利用することを考えており、オントロジーとの関連では、CoreLexで取られた方法が日本語にどの程度当てはまるのかを調べ、意味の体系の検討を行いたいと考えている。

日本語に関する詳細な語彙的情報を記述した辞書として、日本語教育から自然言語処理までの幅広い分野で利用され、その品質を高く評価されているIPAL（特殊認可法人情報処理振興事業協会（IPA）の技術センターで開発されてきた）をベースにして、確定的・静的な記述では捉えることができない言語知識の利用実態を反映した辞書の構造化を目指すものである。

1970年代末以降に生成文法の枠組みの研究の中から、文法のモジュール構成の中での辞書（lexicon）の重要性を見直す動きが出てきた。この方向は現在も続いており、本研究もそのような流れに沿ったもので、日本語の文法全体を辞書中心に捉え直そうというものである。

② 研究目的

本研究の研究目的は次の2つである：

1) 日本語教育にも言語学的な研究のためにも必要な、Pustejovskyの言う「言語の創造的使用（creative use of lingusitic knowledge）」を、文法のモジュール間の関係を明確に仮定して、具体例を基に必要な分類項目とその特徴を明らかにし、その分類に基づいたタグを提案する。そのタグは、言語習得のデータを記述するために提案され、日本語学習者の誤用のデータを記述する拡張も進んでいるCHILDESの拡張版となるようにする。

2) ホームページ上で、JAVAを利用して生成的辞書の環境を構築する。

具体的な研究課題は：

a) 言語習得のデータを記述する規格として多くの研究者が利用し始めているCHILDES（http://jchat.sccs.chukyo-u.ac.jp/childes/index.html）の規格を利用し、言語の創造的使用を実際のデータと辞書情報を参照しながら検討し、記述のために必要になるタグの標準化を行う。

b) 辞書を中心とした構造化を行う。

c) 心理学的な観点から言語の創造的使用がどのようなメカニズムで保証されているのかを検討し、実験を行い、その結果を辞書の構造化に反映させる。

d) 最終的に、構造化した情報をJAVAを使い検索できる環境を開発し、ホームページ上で公開する。

③ 本研究の学術的な特色、独創的・先駆的な点

次の四点である。

a) 従来の辞書の、静的な記述では人間の言語使用の実態を捉えられないという致命的欠点を克服しようとすることが独創的・先駆的であり、日本語に関してこの方向での大きなまとまった研究は未だ行われていない。

b) CHILDESの規格を拡張して記述を行なうので言語習得などのデータと同様に扱え、データの共有が容易になる。

c) 具体的なデータを基に研究を進めるので、心理学的な実験を行うことが容易になる。その結果を辞書の構造化に反映させることで、最終的な構造化が人間の言語使用の実態を反映したものにできる。心理学的な研究を行う点は、この研究の学術的に重要な特色である。

d) CHIDESのフォーマットで生成的辞書の記述を行うので、その規格を海外の研究者と共有することで、各国語のデータの記述を同じ規格で進める環境が整うことの学術的意義は大きい。

b) と関係するが、名古屋大学言語文化部の大曽美恵子を研究代表者とする「日本語学習者の作文コーパス：電子化による共有資源化」（科学研究費基盤研究(B)）[COOKIE] において、誤用に関するタグ付けの検討も同時に進めている。この研究の成果がまとまると、言語習得のデータ、学習者の誤用データと言語の創造的使用のデータすべてがCHILDESの規格で記述され、日本語のデータの共有を大きく促進できる。

④ 社会的貢献度

以下に挙げる点で大きな社会的貢献を期待できる。

a) 従来疎かにされたきた日本語の辞書の致命的とも言える欠陥を克服し、新たな構造を与えることで、日本語の辞書研究を革新することができる。これは、日本語教育で様々な応用が可能で、言語研究に与える影響も大きい。

b) 辞書を中心とした文法の構造化をホームページ上でJAVAを使い実現するので、ホームページを参照できる多くの人が参照できるので、その社会的貢献度は非常に高い。

c) 構築した環境を移植することで、個々の利用者が自ら構造化を実現することができ、データなども自前のものを用意して、それで生成的辞書を個別に作ることができる。

d) 言語習得のデータを記述するために開発されたCHILDESの記述形式を拡張してデータ記述に利用するので、言語習得のデータなどの言語データとの融合が容易であり、具体的なデータをもとにした言語研究に大きな貢献が期待できる。

参考文献

[GLEXICON] http: //lang.nagoya-u.ac.jp/~tonoike/glexicon.html 。

[COOKIE] http://cookie.lang.nagoya-u.ac.jp/

3.4 問題解決（PSM）

3.4.1 COMMET／KresT [Commet-Krest]

(1) 概 要

KresTは、ESPRITのCONSTRUCTプロジェクトの中で開発が始められ、その後ベルギー政府のIUAPプロジェクトの中でVUB（Vrije Universiteit Brussel）により発展させられた。現在COMMET（COMponential METhodology）として開発が進められている。COMMETのツールであるKresT workbenchは、明快な原理に基づく方法論に従う知識ベースアプリケーションシステムの設計と実装ならびに、システム構成要素の再利用可能性を実現することを目的とする。最終的には非プログラマであっても容易にアプリケーションの構築のできる環境の実現を目指している。

他の知識ベースツールはKresT workbenchと比較すれば恐竜にたとえられるであろう。巨大で強力であるが硬直的であり、利用が難しい。KresT workbenchはMacintosh上で手軽に利用可能であり、小規模から中規模のアプリケーション開発に適していると同時に、結果として最終的に大規模なシステム開発にもつながるものである。

COMMETは知識レベルでのモデリングフレームワークに基礎を置いている。実際のシステム記述はmodels、methods、tasksという３つの形式によって行われる。

KresT workbenchは、これらの記述の生成をグラフィカルに支援する環境を併せたツールを提供する。また、知識レベルで記述された定義を機械実行可能なシンボルレベルで記述された実行のためのオブジェクトとリンクすることにより、アプリケーションそのものの開発も支援する。

KresT workbenchの内、application kitはシステム要素再利用の基礎となるものであり、再利用可能な断片要素を一定のまとまりとして提供する。

application kit managerにより、非プログラマでもライブラリの利用・メンテナンスが可能である。

COMMETの知識レベルの枠組みはKADSをはじめとする他の枠組みに類似するがより簡潔で使いやすくなっている。

COMMETのフレームワークは、知識レベルとシンボルレベルとの間の明快で理論的根拠のはっきりした関係を定義した最初のものである。知識レベルは任意の大きさのシンボルレベルのコードともリンクできるため、実際に実用になるシステムの構築も可能である。

現在汎用性（genericity）と再利用性（reusability）が様々な形で追求されているがKresT workbenchは次のような任意の（any possible）観点からの汎用性を実現できる。

1. generic task structures
2. generic model dependency diagrams
3. generic domain models
4. generic methods
5. これらの任意の組あわせ

これら知識レベルの記述は、シンボルレベルのオブジェクトに対するリンクも含んでいる。また、これらの汎用性は任意の粒度レベルで実現されるApplication kit managerは、エンドユーザや開発者の間で再利用可能なライブラリがいかにして共有され、メンテナンスされ得るかを示す最初の明確な実例である。

KresT workbenchは、知識ベースシステムに興味を持ち、明快な原理に基づく方法論に従う設計・実装方法を指向する小・中規模の企業にとって優れたツールである。さらにアプリケーション開発者のグループの間での知識の共有と再利用にも役立つ。

(2) KresT Usergroup

KresT workbenchはKresT usergroupのメンバーが利用することができる。現在約30のヨーロッパのグループがKresTを実際に試験している。ユーザグループの目的は知識システム研究のさまざまな側面を調べ上げることである。これまでにKrest開発チームならびにユーザからのフィードバックに基づいて、3つのバージョンのworkbenchがリリースされている。またこの試験に参加したメンバーの間の定期的会合も開かれている。

(3) KresT Newsgroup

KrestチームとKresTユーザの間の正式な情報交換のためにKresT Newsgroupがある。ユーザグループ会合の報告や新しいバージョンのリリース情報の提供、質問・回答などが行われている。

(4) KresT Experiments

KresT workbenchの試験に関わっている諸機関の実験の進展状況等に関する概要をネットワークを通じて提供する試みも行われている。すべてのユーザに対して、WWWサイトを通じてこのネットワークに参加するよう呼びかけている。最終的にはこのようなネットワークリンクを通じて知識工学に関するノウハウの蓄積と共有を目指している。VUBでは報告書をHTML形式で提供することも開始している。

（5） application toolkit

application toolkitあるいはAppKitは、KresTの機能を拡張する手段の一つで、汎用の設計ツールの実現を目指している。AppKitを用いて次の事が可能となる。

1. 知識表現の拡張
2. インタフェースのカスタマイズ
3. 実行可能なKresT部品の提供
4. 開発試験環境の提供
5. 再利用可能なライブラリの管理

現在までに

1. BaseKit
2. LearnKit
3. MetaKit

が提供されている。以下ではこれらの概要を述べる。

① BaseKit

BaseKitはKresT workbenchにおける知識表現の枠組みとその利用手段を提供するものである。BaseKitでは知識レベルの記述とシンボルレベルの記述は明確に分離されている。また、機能部分と記述部分も分離されている。機能部分はドメイン独立性が高いのに対して記述部分はドメイン依存性が高い。

これらの区分に従いBasekitは全体としては、1. Knowledge level support、2. Knowledge level library、3. Symbol level support、4. Symbol level libraryという４つのモジュールからなる。

1) Feature Structure

BaseKitにおける知識レベルの記述はすべてFS（Feature Structure）と呼ばれる形式に従う。FSは基本的にはattributeとvalueの組の並びであるが、value自体が別のFSであっても構わない「ネストした」構造を持っている。いわばフレームにおける他フレームへのリンクをすべて畳み込んだ構造をしているということができるであろう。FSの例を図３．４．１-１に示す。ネストの深いレベルの情報にアクセスするためにattribute名の並びであるattribute pathが用いられる。例えば図３．４．１-１においてattribute path（address country）に対応するのは"France"である。

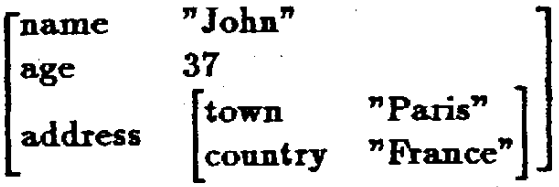


図３．４．１-１　Feature Structure の例

このFSについて、1. FSの加算、2. subsumptionという二つの操作が定義される。加法については、同一レベルのFSについて、attribute名の異なる記述は単純にならべ上げ、attribute名の同一の記述についてはsubsumptionが成り立つかどうかを考慮しながら統合が行われる。一方subsumptionは、簡単に言えば、一方の記述が他方の記述の詳細化になっているとき、単純な記述を詳細な記述で置き換える操作である。subsumeする記述がsubsumeされる記述に現れるすべてのattributeを持ち、それぞれのattributeについてそのvalueが等しいか、subsumeされる側のvalueが未定義であるか、subsumeする側のvalueがsubsumeされる側のvalueの特殊化（要素、部分集合など）になっている、などの時にsubsumptionが成立する。

このFSをブラウズ、編集する手段として、attribute browserが用意されている。これを用いてFSの特定のattributeに値が割り振られている状況を調べたり、また許される場合には値の割付もできる。一般にattributeには取り得る値の制約が（別途）定義されており、制約の中からメニューを通して選択する、あるいは直接値を割り付けた場合制約違反を通知する機能が用意されている。

FSは大きく分けてmodel、task、methodの３種類がある。modelは大雑把に言えばデータ型の指定とattributeの取り得る値の制約を記述するものと言うことができるであろう。データ型は通常はある概念階層の中に位置づけられており、その意味では一種のドメインオントロジーの記述であるということもできる。ここではあるオブジェクトがどのようなグループに属するものか、その属性としてどのようなものが想定されるか、属性値に関する制約はどのようなものか等が指定される。

これに対してtaskは、これも大雑把に言って、タスクの名前を指定するとともに、タスクが適用されるデータに関する制約を記述したものといえる。この制約を記述するためにroleと呼ばれるものが用いられる。

2) Role

role記述は、rolesとconstraintから成り立っており、constraintとしてはsubsumption関係及び／またはequal関係を記述できる。rolesには、タスクに対してどのようなデータ（model FS）が関係し、それはタスクに対してinput、output、または双方向のいずれで

あるかを指定するものである。これに対してsubsumption制約は、それぞれのroleを占めるデータがどのようなFSの特殊化になっていなければならないかという、いわばクラス制約を与えるものであり、equal制約は一つあるいは複数のroleに対応するデータの部分的な等値性が成立していなければならないという制約を与えるものである。equal制約は複数のFSに対してそれぞれattribute pathを指定することにより得られる属性値がすべて等しくなければならないという形で与えられる。このようなroleは、例えば複数の候補の中から一つを選び出すタスクに対して、タスク出力があるmodelに該当するデータであるとき、タスク入力はそのようなmodelのデータの集合であるというような制約を記述するために用いられる。

3) taskとmodel

taskにおけるrole記述は、一般に特定のデータ型（model）を指定するというよりは、データ型の、ある上位のクラスを指定するものである。言い替えれば同一のタスク定義を複数のデータ型（model）に対して適用することができるようになっている。従って実際にシステムの実装に当たってタスクを適用するためには、taskで表現されるタスクをどのような具体的なデータ型（の組）に対して適用するかを指定してやる必要がある。この対応づけはGUI上のアイコン操作（タスクを表すアイコンに対してmodelを表すアイコンを対応づける）によって実現されるが、その結果としてmodel記述の中にはlinked-toと呼ばれるattributeが作られ、どのようなタスクに対してそのモデルがどのようなroleを取るのかが示される。

4) method

タスクは知識レベルの記述であって、どのようなデータにどのような操作を加えるかという論理的な構造を陽に定義するが、しかしそれだけでは実際のシステムに必要な具体的な処理を実行することはできない。methodは、タスクとして定義された操作を実際にシンボルレベルで実行するCLOS（Common Lisp Object System）のgeneric method（関数）に結び付けるものである。シンボルレベルの関数はmethod定義の中のmethod-type属性として指定される。シンボルレベルの関数には、その適用の対象となり得るデータ型に関する制約条件があるが、method定義はこのような制約がtaskレベルの記述とどのような形でリンクできるかに関する記述を含んでいる。実際にコンピュータ上で稼動するシステムを定義するためには、適用対象となるデータ型を与えられたタスクに対して、あらかじめ用意されたライブラリの中からメニューを用いてmethodを割り当ててやる必要がある。これをStampingと呼んでいる。この時methodに関連付けられた制約と、タスクが適用されるデータ型の間の整合性が調べられ、違反のある場合にはエラーメッセージが表示される。また、実際に関連付けが可能な場合には、Methodが適用されるデータに対する制約が実際の具体的なデータ型に対応するものに

Subsumeされる。図３．４．１-２に、StanpingによってMethodに付加される、適用データに関する制約属性の例を示す。同一のタスクに対してstampすることのできるmethodは一般には複数存在する。

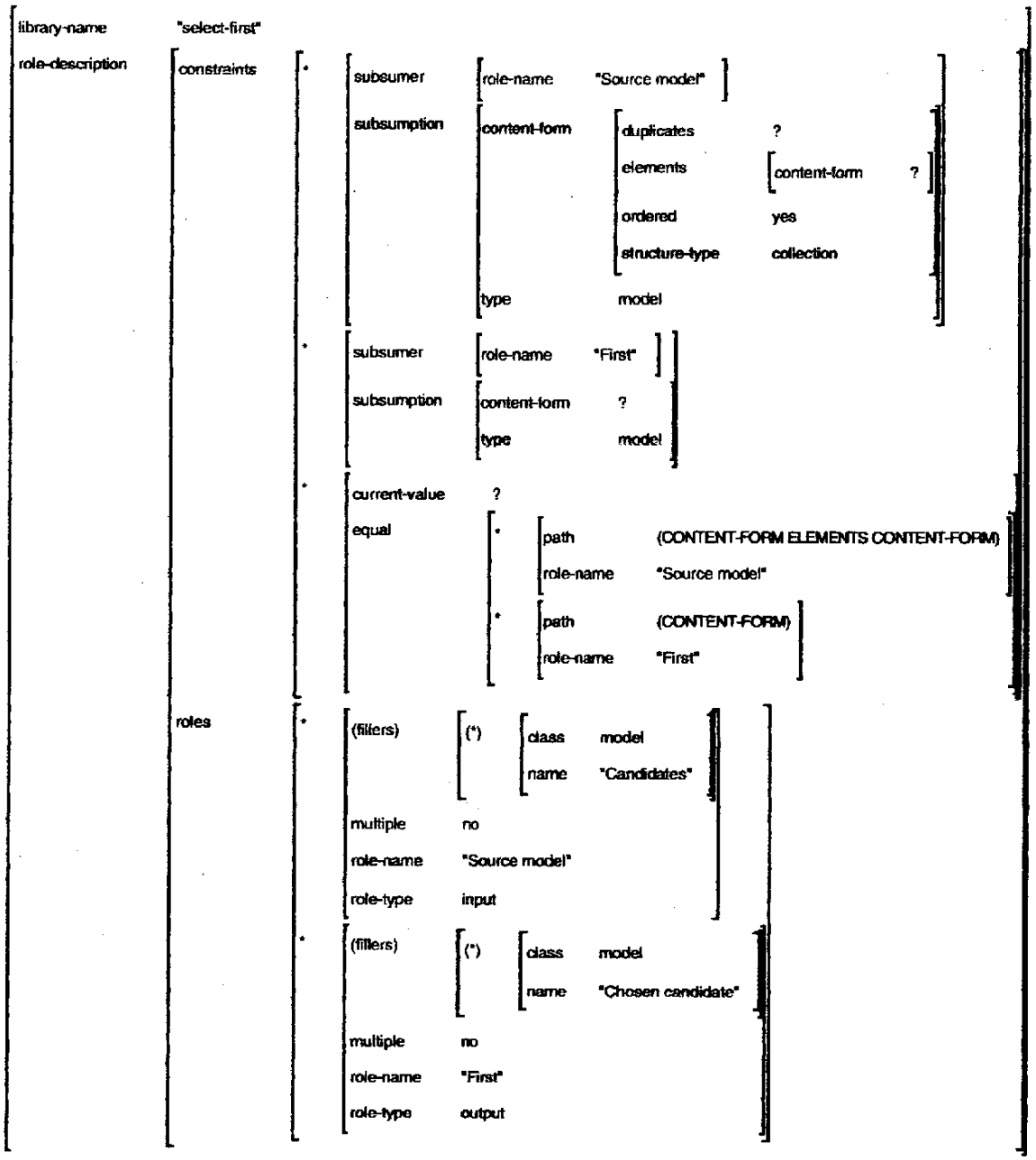


図３．４．１-２　Stampingによって置き換えられたMethodに対する制約例

5) KLlibrary

BaseKitはあらかじめ幾つかの定義をライブラリとして準備している。これらはすべてFSとして定義されるがそれぞれのFSについて、1. どのようなアイコンに対応するか、2. 各attributeに対応する値、3. 各attributeに対応する初期値、4. method-typeとして取り得る値があらかじめ与えられる。

6) Symbol Level

Knowledge levelの記述は、それぞれ対応するsymbol levelの定義を持つ。これらはCLOSのオブジェクトないしはメソッドである。

② LearnKit

知識の再利用と知識ベースシステムの効率的な開発のためには、知識レベルでのシステム設計・開発が有効である。LeranKitはKresT workbenchを用いて知識レベルでのシステム設計と開発を体験するための環境である。LearnKitでは知識レベルの記述の利点を示すための例題と、例題にしたがってシステムを実際に設計・開発するための知識レベルのツール群が用意されている。例題としては分類学習システムであるID4、ID5、IDLを取り上げており、ツールとしてはこれらを実装するために必要なデータ型やタスク、メソッドが、知識レベルのライブラリの拡張という形で用意されている。同時にこれらに対応するシンボルレベルの部品も提供される。この拡張部分は新しい部品として別のアプリケーションに利用することも可能である。

LearnKitの中では、IDLの各バージョンで用いられるデータ型とタスク構造が説明されており、拡張された知識レベルのライブラリを用いて実際のシステムを実装する方法が示されている。

③ MetaKit

BaseKitでは、一つのアプリケーションを実装する際には一般に複数のmodel、task、methodという知識レベルの部品を用いることとなる。実際にはこれらの部品はprojectと呼ばれる上位のオブジェクトによって相互に関連付けられている。実際の問題の中では、このようなProjectを操作することが必要になる場合もある。MetaKitはこのようなProjectに関する問題解決を取り扱うためのメタレベル定義を可能とするBaseKitの拡張である。実際には、

1. 他のprojectの部分構造を表現するmodelのためのcontent form（データ型）
2. 上記のmodelを操作するための幾つかのmethod
3. メタレベル操作のために利用できる知識レベルの部品

が追加されている。

projectを操作するメタレベルは、ある意味でユーザと同じ事ができると考えられる。KresT workbenchを利用するユーザは、1. 知識レベルの記述の作成、2. 記述の検証、3. シンボルレベルの実装、4. アプリケーションの実行、5. 仕様実現の検証などを行うと考えられるが、MetaKitはこれらすべてを行うための知識レベル記述を可能とすることを最終的な目標としている。ただし、現在は記述の検証のみを対象としている。

参考文献

[Commet-Krest] http://arti.vub.ac.be/www/krest/information/commet-krest.html

3.4.2 KACTUS [Kactus]

(1) 概 要

KACTUSプロジェクトは1994年1月から30カ月行われた。協力機関は、Gap Gemini Innovation (F)、LABEIN (sp)、Lloyd's Register (UK)、Statoil (N)、CAP Programmator (S)、University of Amsterdam (NL)、University of Karlsruhe (D)、IBERDROLA (Sp)、DELOS (I)、FINCANTIERI (I)、SINTEF (N)であり、40人年の作業行程で640万ECUのプロジェクトであった。

KACTUSはESPRIT-iiiプロジェクトの一つで、工業システムのライフサイクル全体を通して知識再利用の方法論を開発することを目的としている。ここでは設計、診断、運用、メンテナンス、再設計、訓練等で、同一の知識ベースを利用する。すなわち知識の再利用を行う方法論を明らかにすることが目標とされる。このような知識再利用の基礎は、知識ベースに明示された（explicit）構造を与えること、すなわちオントロジーである。

KACTUSでは、アプリケーションシステムに要求されるオントロジーにはさまざまなものがあることを指摘しており、その一つの現れとして、オントロジーとドメイン知識の関係に、オントロジーがドメイン知識のクラスになる場合とメタ記述になる場合があることを指摘している。単純な制約式で記述される知識（例えばx+y = z）は、知識記述の背景に関する明示的な情報をほとんど持っていないことがある。この記述に対して、'formula'というオントロジー定義は記述された知識に対するメタ記述を与えていると考えられる。一方もう少し背景を陽に記述した、formula (x+y = z)というドメイン知識記述に対しては、このオントロジーは記述された知識のクラスを定義したものということもできる。さらに、'formula ::= constraint | equation'というオントロジー

記述は一方で、formula (x+y = z)に対するメタ記述になっており、また一方で、equation (x+y = z)に対してはクラス記述となっている。

ある知識記述に対してはメタ記述となり、他の知識記述に対してはクラス記述となるオントロジーは、これら二つの知識記述の間での変換を可能とする手がかりとなりうると考えられる。

ここに見られるように、一般にオントロジー記述の詳細さにはさまざまなレベルがあり、どのようなオントロジー記述のレベルが望ましいかはアプリケーションに依存する。また、既にさまざまな分野でのオントロジー定義が現実に相当量存在していることを考えれば、特定のアプリケーションに最適なオントロジー記述を得るためには、既存のオントロジーのライブラリを構成し、必要に応じてここからオントロジーを検索し適切な修正を加える方法を採用することが有効であると考えられる。KACTUSはこのようなオントロジーのライブラリ化と再利用の枠組みを提供しようとするものである。

KACTUSの戦略的目標は次の通りである。

1. 方法論の統合（Method Integration）による効率化
2. オントロジーベースの構築による競争力強化
3. モデリング間の相互運用と情報交換
4. 標準化

また、技術目標は次の通りである

1. 知識再利用の可能性を示す：ドメインオントロジーを構築し、それを異なるアプリケーション間で再利用する。
2. 基本製品データ（basic production data）の記述が、知識表現の定式化（knowledge representation formalism）により強化される結果、記述が知的システムで再利用できる可能性を示す。
3. 製品データ（product data）交換のための既存の枠組みであるSTEPの検証。
4. ツールの整備により、ドメインオントロジーがコントロール可能な形で成長でき、効率的に再利用できることの証明。

開発された方法論の有効性を確認するため次の適用分野が選ばれ、実証検討が行なわれた。

1. 船舶構造評価（ship-structure assessmnt）
2. 操作手順設計（operational procedure design）

3. 電力ネットワーク故障診断ならびに回復計画立案（electorical network fault diagnosis and recovery planning）

（2）KACTUS Toolkit（VOID）

実際に提案するオントロジー共有の手法を実現する手段としてVOIDと呼ばれるKACTUS toolkitが用意されている。これはオントロジー（のライブラリ）を閲覧（brows）、編集（edit）、管理（manage）するための対話型環境であり、同時にC/C++, prologに対応するAPI（application program interface）を提供するものである。

VOID自体はいくつかのソフトウェアモジュールからなる。一般的な使用法では、ユーザはウィンドウベースの環境から対話的利用を行う。KACCTUSの目的の一つは、オントロジーの再利用にあり、ToolkitはCML、Express、Ontolinguaというオントロジー記述形式（ontology formalism）をサポートし、それらの間の（少なくとも部分的な）変換を行うことができる。

VOIDは、内部的にはECO（Express、CML、Ontolingua）というオントロジー記述形式を採用するが、外部的にはCML、Express、Ontolinguaの記述形式をサポートしており、これらの記法に従ってオントロジーライブラリの定義、入力、検索、閲覧ができる。以上の基本的考え方を図３．４．２-１に示す。

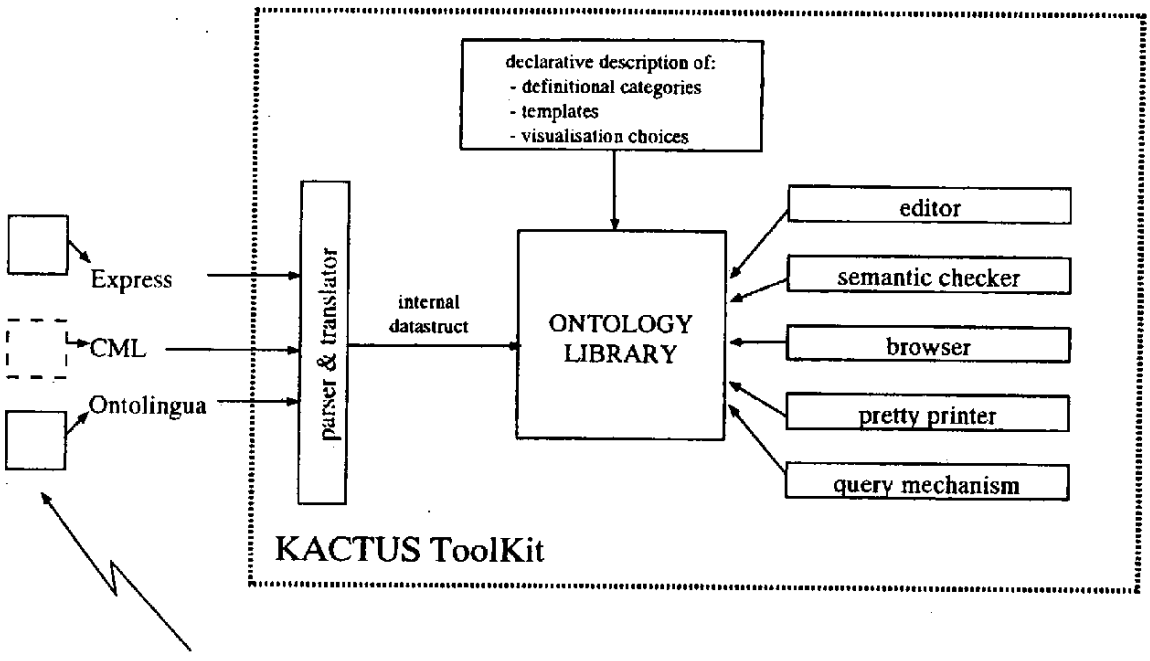


図３．４．２-１　Kactus Toolkitの基本的考え方

複数の記述形式をサポートする理由は、1. プロジェクトの対象が工学分野を含むため、Expressが必要となる、2. タスクや問題解決知識の記述をサポートする記述形式が少ないところからこのような記述を許すCMLを含む、3. 知識工学研究分野に広く普及しており、さまざまなオントロジーがすでに定義されているところからOntolinguaを含むこととした。

CMLはドメイン知識、推論知識、タスク知識、問題解決知識を明瞭に区別するという特徴がある。一方記述はformalityが低く、プログラムで直接に実行することはできない。

CML形式に対してKACTUSは、Import（CMLparserにより、入力ファイルはECO形式に変換され、インポートされる。ECO形式からは、LaTeXおよびHTML形式を生成できる）、Exportおよびprettyprint（ECO形式からCML形式を生成し、ファイル出力ないし印刷ができる）、Brows（CML形式の記述はグラフ形式を含むGUIを用いてブラウズできる、対話的ユーザインタフェースを用いて詳細表示や表示形式の変更を指定できる。CMLPSMについてはグラフィックス機能は現在のところ適用できない）、Ontolingua変換（CMLからOntolinguaへの形式変換ができる。これはinformal->formal変換となるため、完全な変換のためには追加編集が必要である。）、C++変換（CMLから部分的なC++ヘッダファイルが生成できる。現在conceptとbinary-relationに相当する部分がC++クラスに変換される）、および編集機能が提供される。

Express形式に対してはImport（TNOExpress parserにより、prolog clauseが生成され、これを内部形式に変換することにより）、Exportならびにprettyprint、Browse(EXPRESS-Gにより、グラフ形式を含めて）、およびInstances（STEPの物理ファイルを読み込んでデータモデルのインスタンスとのリンクを取る）機能が提供される。

Ontolingua形式に対してはimport、Exportならびにprettyprint、Browse（グラフ形式を含む）、および編集機能が提供される。

VOIDはontology libraryの構築、メンテナンス、利用をサポートする。ユーザから見ると、現在のlibraryはtheories及びその定義のデータベースと見えるlibraryは、各種の外部形式を通じてimport、exportを行ない、また、内部的にはindexingを行なっている。このindexにより、外部形式によるライブラリ検索をサポートする

実際にCMLおよびExpress形式による記述からどのような情報が生成されるかを図3.4.2-2および図3.4.2-3に示す。また、各形式の間でどのような変換が可能であるかを図3.4.2-4に示す。

(3) VOID library manager

VOIDはオントロジーライブラリを中心に各種の機能を提供するToolkitであるが、この中にはライブラリ自体を管理するためのツールとしてのLibrary Managerも用意され

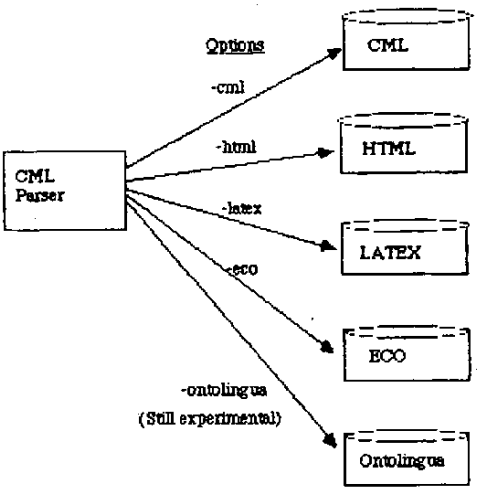


図３．４．２-２　CML からの情報生成

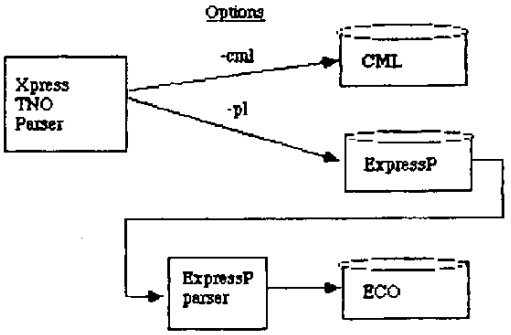


図３．４．２-３　Express からの情報生成

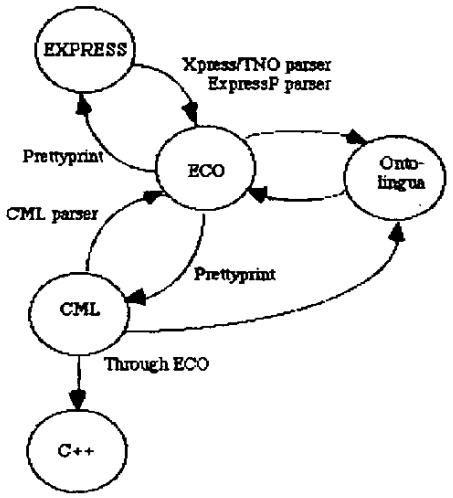


図３．４．２-４　記述形式間の相互変換

ている。このツールを用いることにより、各種のformalismによるontology定義をtheoryとしてlibraryにimportしたり、蓄積されたTheoryを検索したり、ライブラリの内容を調査したりする機能を利用できる。

実際にライブラリにTheoryをInportする際には（手動による）索引づけが行われる。索引の作成・編集のためのダイアログが用意されており、TheoryをInportする際には自動的にこのダイアログが呼び出される。索引づけはtheory type、onto-history、viewpoin、informationという観点から行なわれる。

Theory typeは、Theoryの一般性を示すものであり、generic、inter-domain、domain、applicationのいずれに該当するかを指定する。Genericを指定した場合、さらにPrimaryかsecondaryかを指定することが要求される。また、inter-domainを指定した場合、さらにbasic technical ontologyかあるいはunified domain ontologyであるかを指定することが要求される。

Onto-historyはTheoryが過去にどのような使われ方をしてきたかを示すものであり、利用されたdomain、application、task、problem solving methodを指定することができる。

ViewpointはTheoryがどのような視点から構築されたかを示すもので、例えばstructural、behavioural、physical、functionalを指定できる。

InformationはTheoryに付随する情報を示すもので、作者、作成機関、作成年月日、作成の状況（完成しているか作成の途中かを示す）を指示することができる。

これらの索引を利用してTheoryの検索が可能である。検索にはTheoryによる検索とDefinitionによる検索とがある。Theoryによる検索は上記の索引を利用する検索であり、任意の索引項目に値を指定してやることにより、該当するすべてのTheoryを検索できる。一方Definitionによる検索は、もともとの記述に関連する情報に基づく検索であり、現在は定義が作成された記述形式および定義そのものに与えられた名前を用いた検索が可能である。いずれの検索においても、検索結果をTheory Browserを用いて見ることができる。実際にはこれらの検索方式に加えて、特定の記述形式に特化した検索手段として、CML形式に対するCOQLおよびExpressに対するESQLも用意されている。

(4) Theory Browser

VOIDにはTheoryとして蓄積されたontologyをグラフィカルに表示するためのブラウザが用意されている。実際にはブラウザはDefinitionの記述に用いられた記述形式ごとに用意されている。ブラウザはグラフ表示部分とテキスト表示部分からなっており、グラフ表示部分では、Ontologyに対応するDefinitionが他のDefinitionとどのような関係を持つか（階層関係、相互参照関係など）がグラフ形式で表示される。ここでは表示のレベルをさまざまに変化させるオプションを指定することができる。一方、テキスト表示部分では実際のDefinitionの全体をPretty Print形式で参照することができる。

参考文献

[Kactus] http://www.swi.psy.uva.nl/projects/Kactus/home.html

3.4.3 Euroknowledge [EuroK]

EuroKnowledgeは知識技術の分野の組織化を図り、標準化動向を推進することを目的とするヨーロッパの情報技術施策（initiative）である。EuroKnowledgeはその役割を、知識工学における共通視点の確立とこの分野のユーザ、研究者、ツールベンダーの実践のための最良の焦点となることであると考えている。大学や企業を含む150以上の機関がEuroKnowledgeの活動に興味を持ち、あるいは協力を申し出ている。

現在の主要な目標は、実装レベルとは独立した概念レベルの知識モデリングの標準に関する提案を行うことである。

自発的に提案され予算付けられた幾つかの活動が推進されている。EuroKnowledgeのプロジェクトの一つ（ESPRIT P9806）ではこれらの活動を調整し、資金を強化し、成果の普及を図るとともに、関連する規格団体との連絡も担当している。

EuroKnowledge Associationと呼ばれる国際非営利団体の設立がEuroKnowledgeに興味を持つ団体との連携で設立されることが望まれている。

EuroKnowledgeの具体的な活動についてはあまり資料が見当たらないが、Termi-Knowledgeという形での重要概念を表す用語に関するターミノロジーの決定が企画されている。

参考文献

[EuroK] http://www.aiai.ed.ac.uk/~euroknow/

3.4.4 KEML [Keml]

KEML（Knowledge Engineering Methods and Languages）は、知識モデルを記述する言語（形式）に興味を持つ研究者の間の連携を図るためのMailing Listである。元々はKADSで提唱された専門知識モデルの記述に興味を持つ研究者の連絡機能を提供するものであったが、現在は一般に知識モデルの記述方法に興味を持つ研究者も含む形となっている。

KEMLを通じて提供される情報は、主としてワークショップの案内、新しい論文の紹介などである。KEML自体も毎年のシンポジウムを主催している。

その他、関連情報の蓄積も行っており、FTPでダウンロードが可能になっている。

参考文献

[Keml] ftp://swi.psy.uva.nl/pub/keml/keml.html

3.4.5 CommonKADS

CommonKADSは、欧州のESPRITプロジェクトの一環として1983年より継続的に取り組まれてきた知識工学に関する一連の研究と応用プロジェクトの成果を集約したもので以下の2つの要素からなる[Schreiber 93][Breuker 94][KADS-II]。

1. 知識集約的な問題に対する再利用可能な知識モデルを提供する枠組
2. 知識ベースシステム構築のためのライフサイクル・モデル

CommonKADSはこれまで欧州を中心に産業界も含めて様々なプロトタイプ構築や実用システムの構築に適用されてきた。そして、最近では知識工学に関する素養をもたないソフトウエア開発の実務家にも受け入れられるようにするために、オブジェクト指向方法論の分野で広まりつつある統一表記法であるUML（Unified Modeling Lannguage）[UML 97a]に準拠した表記法も採り入れられている。以下では最新版のCommonKADS [Schreiber 98]に基づき、CommonKADSにおけるモデル間の関係と知識モデルの概要について述べ、最後にUMLについて概説する。

(1) CommonKADSにおけるモデル間の関係

CommonKADSにおけるモデル群は大きく4つに区分されている。以下に各モデルの概要を示す。

組織モデル（Organization model）
開発対象となる知識システムが運用される組織を特徴付けるためのモデル。
知識システム導入の妥当性やその影響を評価するために用いられる。

知識モデル（Knowledge model）
知識集約的な問題において用いられる知識構造を明示的にするモデル。特に、様々な知識要素が果たす役割を特定の実装環境に依存することなく人間にとって理解の容

易なモデルとして記述する。知識モデルは知識システムの問題解決過程について、専門家とシステムの利用者が互いに意見を交換するための手段としても重要である。

コミュニケーション・モデル（Communication model）

組織内での業務プロセスに関わる様々な人間やシステムの間での情報交換あるいは指示連鎖の過程をモデル化するために用いられる。知識モデルと同様に、特定の実装環境に依存しないモデルとして記述される。

設計モデル（Design model）

知識システムの要求仕様を実装形態に依存しないレベルで明らかにする知識モデルとコミュニケーション・モデルに対して、設計モデルは特定の実装環境やプログラム言語を前提としてシステム的な仕様を与えるものとして位置付けられる。

(2) 知識モデルの概要

CommonKADSにおける4つのモデルのうち、問題解決知識のモデル化において中心的役割を果たすのが知識モデルである。知識モデルは、領域知識、推論知識、タスク知識の3つのカテゴリーからなる。図3.4.5-1に知識モデルにおける3つの知識カテゴリーを示す。これらの知識の記述言語としてCML（CommonKADS Knowledge Modelling Language）[Schreiber 98]が定義されている。

図3.4.5-1は知識モデルの簡単な具体例（インスタンス）に相当するものである

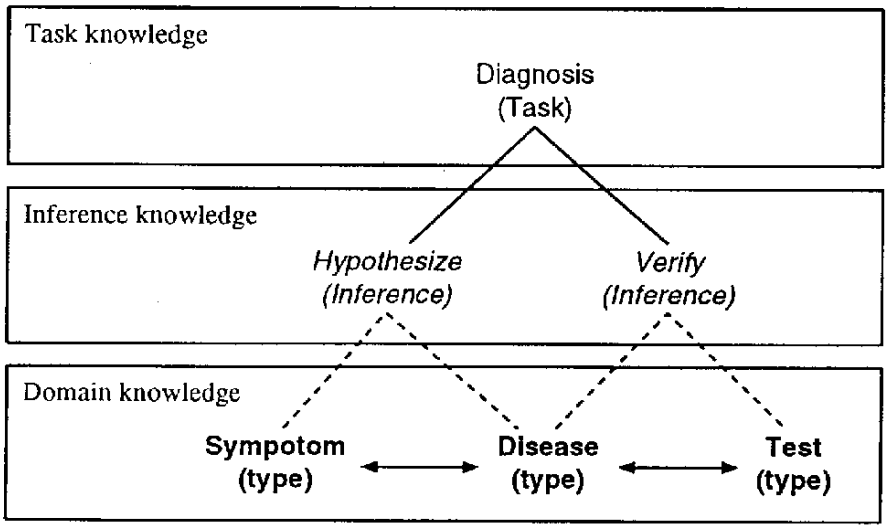


図3.4.5-1　知識モデルにおける3つの知識カテゴリー

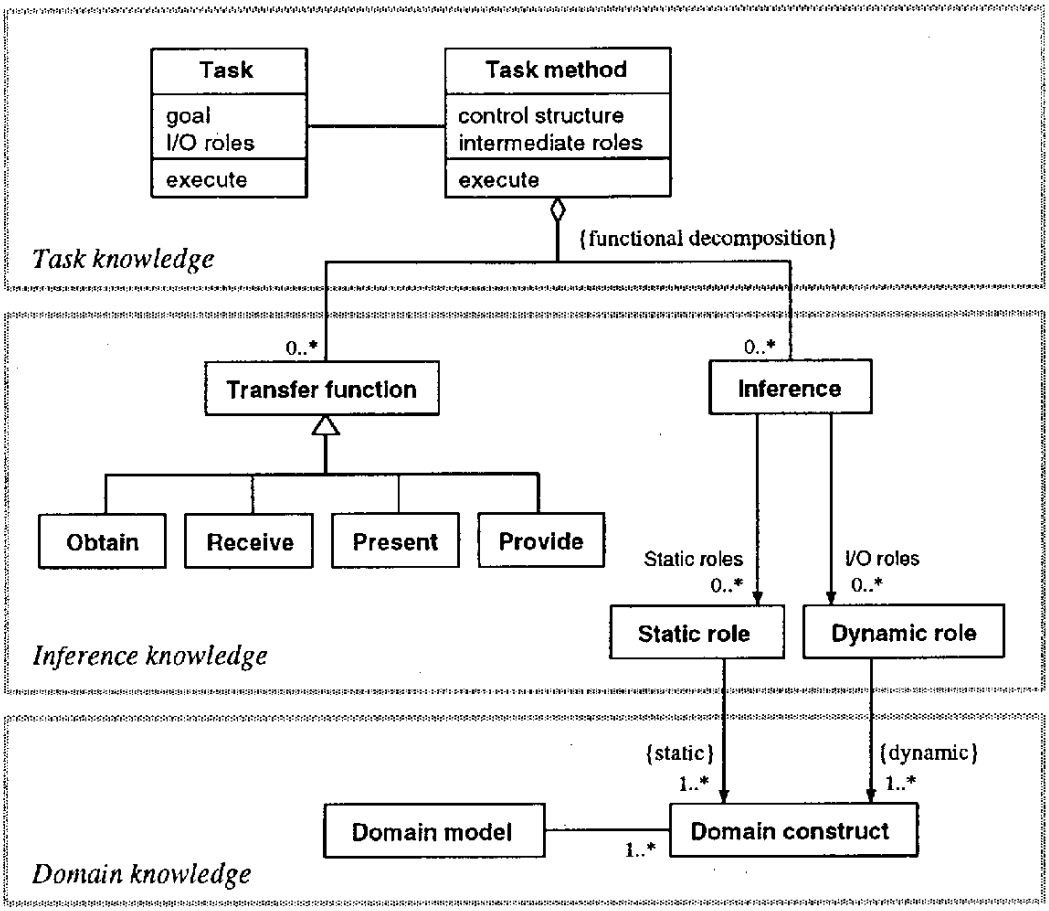


図３．４．５-２　CommonKADSにおける知識モデルの構造

のに対して、知識カテゴリーにおける構成要素とそれらの間の関係に着目することよって知識モデルの構造を図３．４．５-２のように表わすことができる。以下、３階層からなる知識モデルの構造を図３．４．５-２に沿って概説する。

領域知識（Domain knowledge）

適用領域における主要な対象物ならびに知識タイプを与える。個別の問題事例に対して構築される領域モデル（domain model）だけでなく、領域モデルの雛型となる領域知識スキーマ（domain-knowledge schema）を定義することができる。領域知識スキーマには、概念（concept）と関係（relation）の他に概念間の制約条件等をルール形式で記述するためにルール・スキーマ（rule schema）が提供される。

推論知識（Inference knowledge）

領域知識が静的な知識構造を対象とするのに対して、推論知識ではそれらが問題解

決過程においてどのように利用されるかが記述される。推論知識の主要な構成要素には、推論プリミティブ（inferences）、知識ロール（knowledge role)、トランスファー関数（transfer functions）の３種類がある。知識ロールには動的ロール（dynamic role）と静的ロール（static role）があり、前者は推論プリミティブの入出力として領域知識における知識タイプと関連付けられる。トランスファー関数は知識システムと外界の入出力を司るもので、操作主体と情報源がそれぞれシステムと外界のどちらにあるかを区分することによって４種類の関数が提供される。

タスク知識（Task knowledge）

タスク知識にはその目的（What）とそれを達成するための推論戦略（How）が記述されるが、これらはそれぞれタスク（Task）とタスク・メソッド（Task-method）と呼ばれる。タスクは問題解決の目的を入力と出力によって定義される。例えば、診断タスクでは異常兆候を入力として故障タイプとそれを支持する根拠を出力される。一方、タスク・メソッドでは、一連の部分機能とそれらの実行制御を通して、タスクの目的がどのように達成されるかが記述される。

知識集約的問題については、分類と合成の区分を中心にある程度典型的な問題タイプの類型が知られている。CommonKADSでは図３．４．５-３に示したタスクタイプについて、再利用を前提とした知識モデルのカタログが提供されている。知識モデルの具体例として、本報告書の３．５．３項ではCommonKADSにおけるスケジューリング・モデルが示されている。

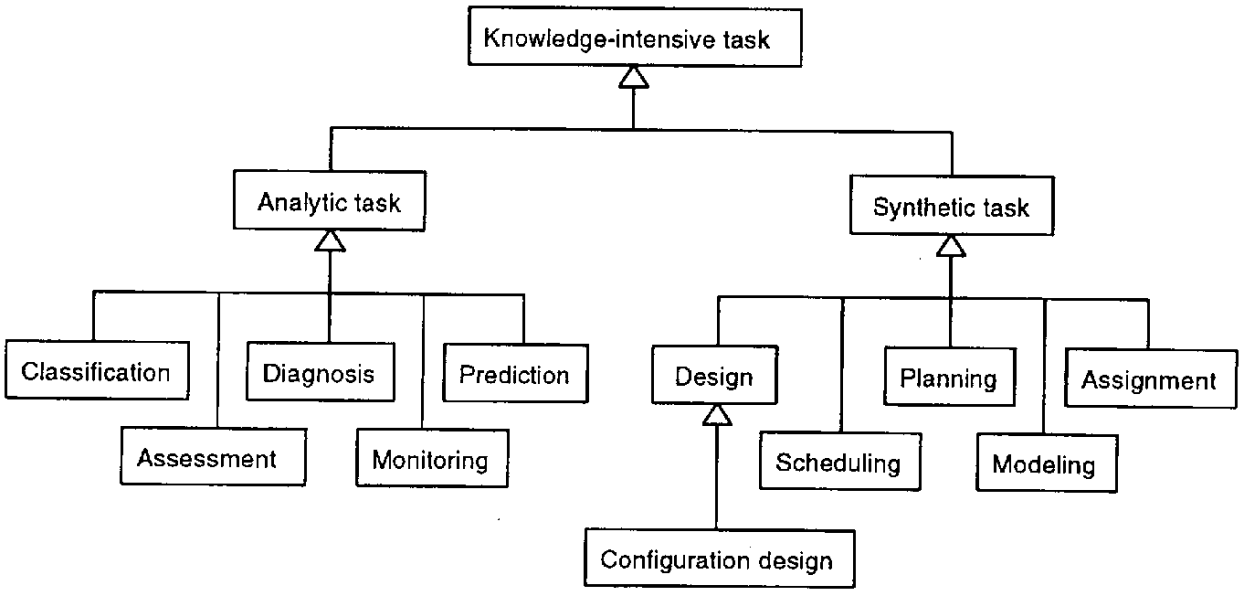


図３．４．５-３　CommonKADSにおけるタスクタイプ階層

（3） UML：Unified Modeling Language

UML（Unified Modeling Language）は、1980年代から1990年代初めにかけて数多く提案されたオブジェクト指向分析／設計方法論を集約するものある。特に、Booch、Rambaugh（OMT）、Jacobsonらの方法論を中心にまとめられているが、UML version 1.1の策定には１７の企業が関わったている[UML 97a]。UMLはオブジェクト指向分析／設計における表記法の標準化案としてOMG（Object Management Group）にも採択されている[OMG 97]。

OMT法やBooch法など従来のオブジェクト指向方法論では、モデリング言語と開発プロセスが必ずしも明確に分離されていなかった。それに対して、UMLは方法論としてではなくモデリング言語として位置付けられている。ソフトウエアの分析／設計方法論はモデリング言語と開発プロセスからなり、前者は設計情報を記述するための（視覚的）表記法を与え、後者は設計過程の諸段階を規定する。実際には、UMLは開発プロセスの中で用いられることになるが、開発プロセスは問題領域や組織ごとに異なる場合が多いためUMLでは共通の表記法とその意味を規定するメタモデルの整備が中心に行なわれた。

このようにモデルの表記法と開発プロセスが明確に分離されたことによって、CommonKADSが前提とする開発プロセスとの整合性を問うことなく、CommonKADSのモデリング言語であるCMLをUMLの拡張言語として位置付けることが容易になった。

UMLでは静的構造を記述するためにクラス図、オブジェクト図、パッケージ図が提供され、動的振舞を記述するためにユースケース図、シーケンス図、コラボレーション図、状態図、アクティビティ図が提供されている。UMLが提供する表現言語の意味は、概念（メタクラス）／関係／制約の３つの側面から規定され、メタモデルとしても定義される[UML 97b]。メタモデルはUMLのクラス図で表記されるが、制約の記述には自然言語文に加えてOCL（Object Constraint Language）が用いられる。メタモデルの利用指針は自然言語文による記述的説明として与えられる。さらに、OCLは集合あるいは集合間の不変的な関係を論理的に表現する手段を与え、これによってモデル要素に対して課せられる制約が明示的に与えられる。

例えば、CommonKADSの表記に基づいて記述されたモデルがその前提となるメタモデルとともに提供されるようになれば、もともとCommonKADSの表記に対応していない開発ツールであっても表記法のメタな定義に基づいてツール自身の機能を適応的に拡張することも可能になると考えられる。

参考文献

[Breuker 94] J. Breuker and W. V. de Velde (Eds.): The CommonKADS Library for Expertise

Modelling. IOS Press, Amsterdam (1994).

[KADS-II] KADS-II ESPRIT Project P5248. http://www.swi.psy.uva.nl/projects/Common KADS /home.html

[OMG 97] OMG CORBA News: Object Management Group adopts Unified Modeling Language and Meta Object Facility Specifications. http://www.omg.org/news/pr97/umlpr.htm (1997)。

[Schreiber 93] A. Th. Schreiber, B. J. Wielinga, and J. A. Breuker (Eds.): KADS: A Principled Approach to Knowledge-Based Systems Development. Academic Press (1993).

[Schreiber 98] A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, W. Van de Velde and B. J. Wielinga: Engineering of Knowledge: The CommonKADS Methodology (Version 0.5), Department of Social Science Informatics, University of Amsterdam (1998).

[UML 97a] UML Summary, version 1.1 (1997). http://www.rational.com/uml/html/summary/

[UML 97b] UML Semantics, version 1.1 (1997). http://www.rational.com/uml/html/semantics/

3.5 プランニング・オントロジー

本節では、プランニングならびにスケジューリングに関わる基本概念を定義し、オントロジーとして体系化を目指して行なわれている活動を紹介する。以下、ARPAとRome Laboratoryによって1991年に開始されたプロジェクトの一環として策定されたプラン記述言語（3.5.1）とプラン・オントロジー（3.5.2）について概説する。次に、知識システム構築方法の確立を目指して欧州で取り組まれてきたCommonKADS方法論において提案されている形式に沿ってスケジューリングに関する知識モデルを紹介する（3.5.3）。さらに、プランニング／スケジューリングが実施されるより実用的な環境に関わるものとして、生産スケジューリング・オントロジー（3.5.4）を取り上げる。

3.5.1 ARPI Plan Language

ARPAが推進している知識の共有／再利用に関する活動であるKIF、Ontolingua、KRSS はいずれも特定の対象領域に依存しない汎用の表現形式の策定を目指すものであるのに対して、ARPA/Rome Laboratory Planning Initiative（ARPI）[Fowler 95][KRSL]はより具体的な応用分野を見据えた活動として位置付けられる。

Knowledge Representation Specification Language（KRSL）[Lehrer 93]は、ARPIプロジェクトにおける初期の成果として策定されたプラン記述言語である。

KRSLはプランニングに固有の基本概念を表現するための記述言語で、それまで個別に扱われてきた時間／資源／プランといった概念を扱う共通の枠組を提供することを目指している。KRSL 2.0.2 [Lehrer 93]では、プランニングに関わるプリミティブとして以下のものが定義されている。

TIME, PLANS, RESOURCES, MEASUREMENT, SETS, NUMBER,
SENTENCES, OBJECTS, FRAMES, CONCEPTS, RELATIONS

なお、KRSLはLOOM[MacGregor 88]と類似の構文をもつものとして定義されている。以下では、特に資源とプランのプリミティブについて具体例とともに概説する。

(1) 資源プリミティブ

資源（RESOURCES）は様々なアクティビティを実施する能力を提供する抽象的な構造である。例えば、輸送用の飛行機では荷物の収納スペースが資源として提供され、

輸送中にその収納スペースが消費される（図３．５．１-１）。資源プリミティブに定義されている属性の一部を図３．５．１-２に示す。

資源によって供与される能力の測定単位や、資源が消費され再利用できるか否か、あるいは資源が共有可能かどうかといった一般的な性質は、資源タイプ（resource-type）

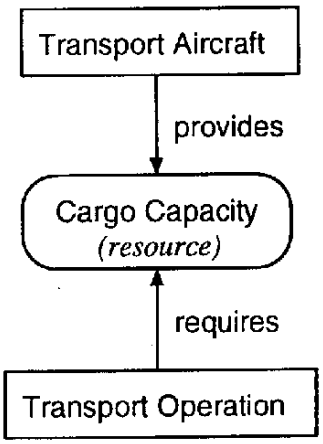


図３．５．１-１　資源の供与と消費

```
RESOURCES
        resource-type
                consumable
                reusable-sharable
                reusable/synchronized-sharable
                reusable/independently-sharable
        resource-records
                provided-resources
                required-resources
                quantity
                max-quantity
                min-quantity
                unit-type-of
                measured-by
                allocation-event
                deallocation-event
                production-event
        resource-usage-records
                provider-usage
                consumer-usage
                producer-usage
```

図３．５．１-２　KRSLにおける資源プリミティブの属性（一部）

```
(define (event produce-resource)
  :parameters ((?r :type-restriction resource-type)
               (?q :type-restriction quantity)
               (?producer :type-restriction object))
  :time (?te)
  :context nil
  :events
   (:when t
    :event (:exists (?t2)
            (:and (interval-before ?te ?t2 :inf :eps)
                  (:holds ?t2
                    (quantity ?producer
                              ?r
                              (+ (quantity ?provider ?r) ?q)))))
    :likely 1 ))
```

(a) 資源生成イベント

```
(define (event allocate-consumable-resource)
  :parameters ((?r :type-restriction resource-type)
               (?q :type-restriction quantity)
               (?consumer :type-restriction object)
               (?provider :type-restriction object))
  :time (?te)
  :context nil
  :events
   (:when (:holds ?te (>= (quantity ?provider ?) ?q))
    :event (:exists (?t2)
             (:and (interval-before ?te ?t2 :inf :eps)
                   (:holds ?t2
                     (quantity ?provider
                               ?r
                               (- (quantity ?provider ?r) ?q)))
                   (:holds ?t2
                     (quantity ?consumer
                               ?r
                               (+ (quantity ?consumer ?r) ?q)))))
    :likely 1 ))
```

(b) 資源消費イベント

```
(define (resource-type consumable-resource)
  :allocation-events allocate-consumable-resource
  :deallocation-events nil
  :production-events produce-resource)
```

(c) 消費型の資源

```
(define (resource-type fuel)
  :defaults-form consumable-resource
  :unit-type barrel)
```

(d) 消費型資源としての燃料

図3.5.1-3　消費型資源の定義

によって規定される。例えば、再利用できない消費型の資源（consumable-resource）は以下のように定義される。まず、資源提供者から供与された資源によって一定時間経過後に資源量が増加することを表現する資源生成イベント（produce-resource）を定義する（図３．５．１-３(a)）。次に、消費型資源を使用することによって提供者からは資源が減少し消費者側でそれに相当する資源量が増加することを表すイベント（allocate-consumable-resource）を定義する（図３．５．１-３(b)）。

これらのイベントを前提として、消費型の資源は図３．５．１-３(c)のように定義される。消費型資源では、一度消費された資源を再度利用できないことを表すために、:deallocation-eventsの値がnilとなっている。燃料をバーレルを単位として量る消費型資源として定義すると図３．５．１-３(d)のようになる。

(２)　プラン・プリミティブ

プラン（PLANS）は適用領域における妥当な振舞を規定するために用いられ、公理（axiom）、因果規則（causal rule）、選好関係（preference relation）、プラン（plan）、イベント（event）といった要素から構成される（図３．５．１-４）。

公理と因果規則は、それぞれ対象世界における事実と状態変化に課せられる制約条件を領域知識として記述するために用いられる。選好関係は複数の状態記述に対して優先順位を規定する。プラン（plan）は所与の状態変化を達成するための内包的手順を定義するものである。

```
PLANS
        axiom
        causal-rule
        preference-relation
        plan
                inheritance-list
                execution-interval
                goal
                parameter-list
                constraints
                preferred-constraints
                effects
                decomposition
                        nodes
                        graph
                expansion
                results
        event
```

図３．５．１-４　KRSLにおけるプラン・プリミティブの属性（一部）

```
(define (plan stack)
  :documentation "Stack ?block on ?obj"
  :parameters ((?block :type-restriction block)
               (?obj :type-restriction object))
  :goal (:holds-after (time-of Stack) (on ?block ?obj))
  :effects
   ((:when t
     :effect (:holds-after (time-of Stack) (on ?block ?obj))
     :likely 1.0))
  :decomposition
   (:nodes
     ((:label clear-block
       :achives (:holds-after (time-of clear-block) (clear ?block))
       :use :any)
      (:label clear-obj
       :achives (:holds-after (time-of clear-obj) (clear ?obj))
       :use :any)
      (:label puton
       :achives (:holds-starting-with (time-of puton) (on ?block ?obj))
       :use :primitive))
    :graph
     (:and (interval-before (time-of clear-block) (time-of puton))
           (interval-before (time-of clear-obj)   (time-of puton)))
    :duration
     (:min :epsilon :max (:minutes 1))))
```

(a) 積み上げプラン

```
(define (causal-rule deduce-clear)
  :parameters ((?block1 :type-restriction block)
               (?block2 :type-restriction block))
  :trigger (:hold-starting-with (time-of deduce-clear) (on ?block1 ?block2))
  :effects
   (:for-all ((block ?other-block) (time-interval ?t))
    :when
     (:and (:holds ?t (on ?block1 ?other-block))
           (interval-before ?t (time-of deduce-clear))
           (interval-before-bounds ?t (time-of deduce-clear)
             (:the (duration-bounds :epsilon :epsilon))))
    :effect (:holds-starting-with (time-of deduce-clear) (clear ?other-block))))
```

(b) deduce-clear 因果規則

図3.5.1-5　プラン定義の例

図3.5.1-5にブロックを目標物の上に積み上げるプラン定義を示す。このプランでは以下の3つの状態経てブロックが目標物の上に積まれた状態に到達する手順が示される。

1. (clear ?block)　：ブロックの上面から障害物を取り除く
2. (clear ?obj)　　：目標物の上面から障害物を取り除く
3. (on ?block ?obj)：ブロックを目標物の上に載せる

これらの状態は :node として定義され、状態間の順序関係および操作全体が１分以内に終了すべきことはそれぞれ :graph と :duration に明記されている。

KRSLはプランニング・アプリケーションにおける要素システム間の情報交換に利用されたが、当初期待されたほどの成果はみられなかった。KRSLの仕様自身も1993年（Version 2.0.2）以降、改訂作業が行なわれていない。その理由としては、KRSLが必要以上に厳密（rigid）であることに加えて、現状の技術で実現可能なプランニングの基本機能の多くが仕様化の対象になっていなかった点が指摘されている[Tate 96a]。

参考文献

[Fowler 95] N. Fowler, S. E. Cross, and C. Owens: The ARPA-Rome knowledge-based planning and scheduling initiative. IEEE Expert, Vol.10, No.1, pp.4-9 (1995).

[KRSL] Knowledge Representation Specification Language (KRSL) effort. (http://isx.com/pub/ARPI/ARPI-pub/krsl/krsl-info.html).

[Lehrer 93] N. Lehrer (Ed.): ARPI KRSL Reference Manual 2.0.2, ISX Corp. (1993).

[MacGregor 88] R. MacGregor: The evoluving technology of classification-based knowledge representation systems. In Principles of Semantic Networks: Explorations in the Representation of Knowledge. J. F. Sowa (Ed.), pp. 385-400, Morgan Kaufmann (1991).

[Tate 96a] A. Tate: Towards a plan ontology. Journal of the Italian AI Association, January (1996).

3.5.2 ARPI Plan Ontology

KRSLの仕様が策定された後、ARPIプロジェクトは分散協調環境におけるプランニング・システムの構築を中心に進められ、知識共有のための知識サーバや共通の開発環境の整備が進められた[Burstein 95]。また、それと並行して、プラン・オントロジーの構築を目指したグループ（POCG：Plan Ontology Construction Group）がKRSL仕様策定の活動から派生した。POCGは活動を開始するにあたって、オントロジーの主要な役割として以下の２つがあることを最初のミーティングで確認した[Doyle 94]。

1.知識伝達のための媒体 (media for communication)
2.情報表現の手段 (means for representing information)

以下では、1996年9月に公開されたドラフト[Tate 96b]に基づいてARPIプロジェクトにおけるプラン・オントロジーの内容を概説する。プラン・オントロジーの主要な構

成概念は以下のとおりである（図３．５．２-１）。

プラン、行動仕様、アクティビティ、状態、エージェント、目的プラン（Plan）は何らかの目的（Objective）を達成するための行動仕様である。行動仕様（Activity Specification）には１つあるいは複数のアクティビティ（Activity）が伴われる。アクティビティの遂行や目的の達成にあたっては、エージェント（Agent）が活動主体として関与する。アクティビティにはその開始と終了の時刻が付随する。また、アクティビティは状態（State of Affairs）の変化を伴うが、その適否については目的に付随する評価基準（Evaluation Criterion）に準じて判断される。

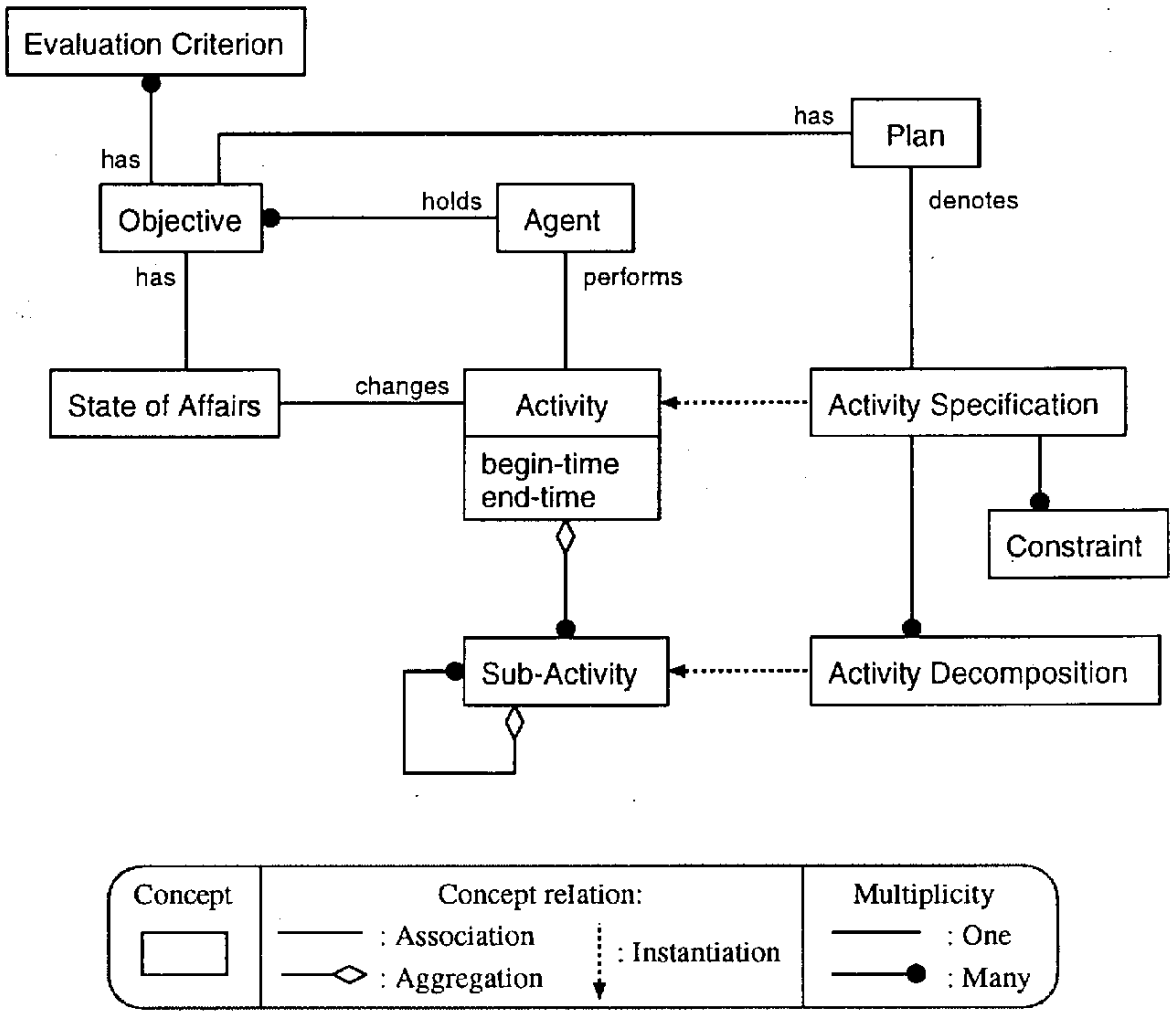


図３．５．２-１　ARPIプラン・オントロジー

行動仕様はそれ自身あるいは対応するアクティビティの時区間に対して、制約（Constraint）を課す。また、行動仕様は複数の部分行動仕様（Activity Decomposition）に分割することができる。部分行動仕様はあるアクティビティがどのようにして複数のサブ・アクティビティ（Sub-Activity）に分割されるかを規定する。この分割によっ

て、各サブ・アクティビティにも制約が課されることがある。

POCGのプラン・オントロジーは、対象領域のモデル化、プランの生成／解析、プランに基づく行動のモデル化など様々な目的に利用されることを前提としている。したがって、図3.5.2-1に示したプラン・オントロジーはこのような目的に沿ってワーキング・グループのメンバーの合意に基づいて得られたものである。以下の各項ではスケジューリングに関するいくつかのオントロジーを、プラン・オントロジーとのモデル化の観点の相違を踏まえて概説する。

参考文献

[Burstein 95] M. H. Burstein, R. Schantz, M. A. Bienkowski, M. E. des Jardins, and S. F. Smith: The common prototyping environment: A framework for software technology integration, evaluation, and transition. IEEE Expert, Vol.10, No.1, pp.17-26 (1995).

[Doyle 94] J. Doyle: Report on the KRSL/Ontology kickoff meeting. http://isx.com/pub/ARPI/ARPI-pub/krsl/workshop-summary.txt (1994).

[Tate 96b] A. Tate: KRSL Plans working group: Plans and activities. http://www.aiai.ac.uk/~bat/krsl-plans.html (1996).

3.5.3 CommonKADSにおけるスケジューリング・モデル

CommonKADSは欧州におけるESPRITプロジェクトの成果として体系化された知識システム構築方法論である[Schreiber 98]（3.4.5参照）。本項では、CommonKADSにおける知識モデルの具体例としてスケジューリング・モデル[Hori 98a]について述べる。

(1) タスク知識

スケジューリング問題では、時間的順序関係が予め規定された複数のアクティビティが与えられる。そして、それぞれのアクティビィティが要求する作業を実施するために占有すべき資源をその開始／終了時刻とともに決定しなければならない。

与えられたすべてのユニットに対して、その開始／終了時刻を確定しながら資源に割り付ける典型的な手順を図3.5.3-1に示す。この手順では以下の基本操作系列が割付候補であるユニットがなくなるまで繰り返される。まず、割付候補となるユニットを選択し、それを割り付けることができる資源を選択した後、資源へのユニットの割付が行なわれる。さらに、その結果を評価し、評価基準が満たされない場合にはス

ケジュールの修正が行なわれる。

```
task scheduling;
  roles:
    input: jobs;
    output: schedule;
end task scheduling;

task-method temporal-dispatching;
  realize: scheduling;
  decomposition:
    inferences: specify, select, assign, modify, evaluate;
  roles:
    intermediate: candidate-unit, target-resource, truth-value;
  control-structure:
    specify( jobs → schedule );
    while has-solution select( schedule → candidate-unit ) do
      select( candidate-unit + schedule → target-resource );
      assign( candidate-unit + target-resource → schedule );
      evaluate( schedule → truth-value );
      if truth-value = false then
        modify( schedule → schedule );
    end while
end task-method temporal-dispatching;
```

図３．５．３-１　CommonKADSにおけるスケジューリング・タスク知識

(２)　領域知識スキーマ

個別の問題事例に対して構築される領域モデルに対して、領域知識スキーマは領域モデルの雛型となるモデルを定義するものである。図３．５．３-１に示したタスク知識におけるタスク・メソッドが前提とする領域モデルの構造は、図３．５．３-２の領域知識スキーマとして表すことができる。すなわち、スケジューリング問題の性質は、資源（resource）とユニット（unit）間の一対多関係が特に動的に設定されることによって特徴付けることができる。資源の容量に関する制約はこの一対多の関係に課せられていると見なすことができる。

一方、複数ユニットは時間的順序関係を前提として１つのジョブ（job）として集約される。このジョブとユニット間の集約関係はスケジューリング問題の入力として予め与えられる。ジョブとユニット間のこのような関係は、ARPIのプラン・オントロジーにおけるアクティビティ（Activity）とサブアクティビィティ（Sub-Activity）の関係に対応している（図３．５．２-１）。そして、アクティビィティの階層構造は、プランニングにおける問題解決の結果が反映される部分である。

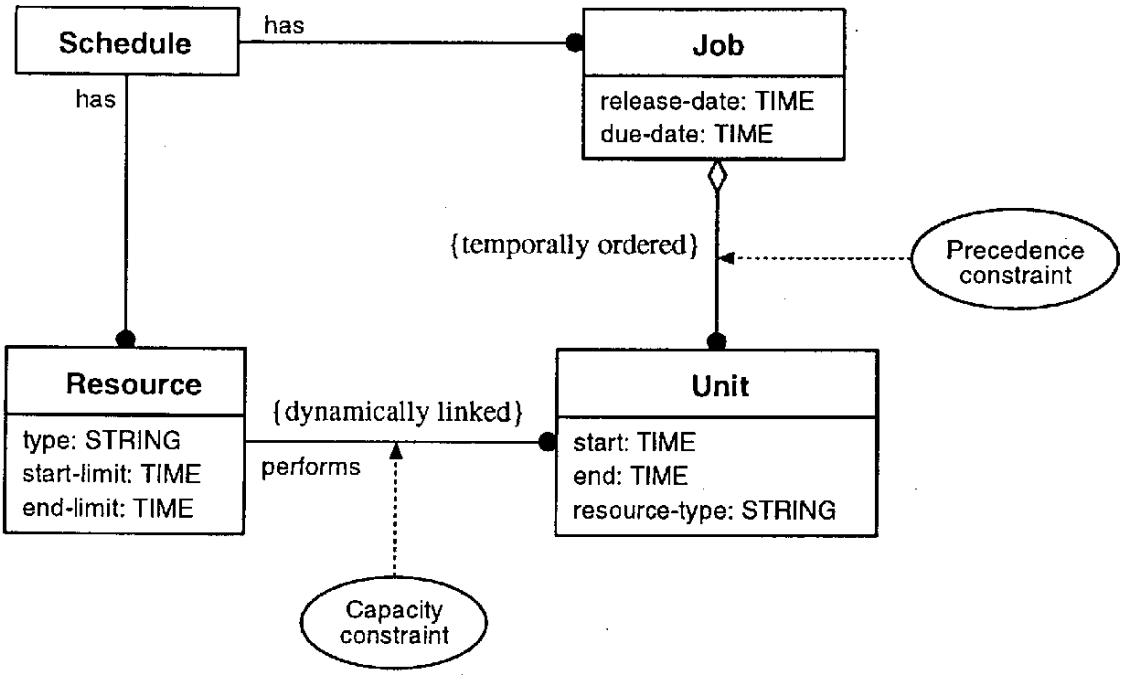


図３．５．３-２　スケジューリングにおける領域知識スキーマ

したがって、ジョブおよびアクティビィティを中心とする２つの部分構造はいずれも基本動作の時間的順序関係を規定するという意味では共通している。しかしながら、プランニングではその出力として複雑な振舞を表現する必要があり、アクティビィティの階層構造を何段階にもネストさせることができるようにモデル化されている。図３．５．２-１において、サブアクティビィティが自分自身に対して集約関係を定義されている部分がこれに相当する。

一方、スケジューリングではジョブとユニットの一対多の集約関係は確定済みの入力データとして与えられるため、特にネストした階層構造を表現できるようにはモデル化されていない。しかしながら、スケジューリングでは作業としてのユニットを資源に対して割り付けた結果が出力となる。そのためユニットと資源の間の一対多の関係まで含めてモデル化しておく必要がある（図３．５．３-２）。

なお、プランニングにおけるエージェントはあくまでアクティビティの動作主体であり（図３．５．２-１）、アクティビィティ（あるいは基本作業であるユニット）の実行を可能にする環境としての資源とは異なるものである。

図３．５．３-２の領域知識スキーマ、図３．５．２-１のプラン・オントロジーは、いずれもプリミティブな推論操作が問題解決において参照する概念とそれらの間の関係を明示的にするもので、単一タスクを拠り所とした機能的観点からその前提をモデル化している。次項では、ドメイン・オントロジーに対してより具体的な対象分野の性質

をモデル化するものとして生産スケジューリング・オントロジーを取り上げる。

参考文献

[Schreiber 98] A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, W. Van de Velde and B. J. Wielinga: Engineering of Knowledge: The CommonKADS Methodology, Department of Social Science Informatics, University of Amsterdam (1998).

[Hori 98a] M. Hori: Scheduling knowledge model in CommonKADS. Research Report RT-0233, Tokyo Research Laboratory, IBM Japan Ltd. (1998).

3.5.4 生産スケジューリング・オントロジー

生産スケジューリング・オントロジーは、生産スケジューリング・システム構築のための部品ライブラリSCOOP（Scheduling COmponents fOr Production control systems）として実現されている[Hori 98b]。SCOOPは、生産設備や生産工程といった製造現場に特徴的な構造を表すスケジュール・モデル、計画立案の機能を担うスケジューリング・エンジン、さらにユーザインタフェイスに関するモジュールの３つのモジュールすなわちサブシステムから構成される。

スケジュール・モデルはドメイン・オントロジーに相当するものである。そして、スケジューリング・エンジンにおける基本操作が参照する概念だけでなく、エンド・ユーザに対するグラフィカルな表示に必要な概念をモデル化している。したがって、スケジュール・モデルは複数の観点からの前提をモデル化するものとなっている。

以下では、各サブシステムごとにその概要を述べる。なお、SCOOPは小型ハードディスク製造現場における工程管理システムの開発に用いられ、開発されたシステムはタイとハンガリーで操業中の合計３ヶ所の工場で実稼働している。

(1) スケジュール・モデル

スケジュール・モデルはスケジューリングの観点から生産現場をモデル化するもので、図３．５．４-１に示すように資源（Resource）、生産工程（Process）、製品（Product）に関する３つのサブシステムから構成される。資源サブシステムは工作機械などの生産設備に関するもので、同一タイプの資源集合である資源グループ（ScRscGrp）、個々の資源（ScRsc）、および資源の休止時間（ScRest）からなる。一連の休止時間はその雛型である休止パタン（ScRestPattern）に基づいて生成される。一方、生産工程はあるタイプの資源において実施される最少の作業単位であるユニッ

ト（ScUnit）、ある製品を完成させるために順序付けられた一連のユニット全体に相当するジョブ（ScJob）からなる。ユニットは個別のロットに対応付けられた具体的な作業系列であり、製品の種類ごとに定められた工程経路情報（ScRoute）と工程ステップ情報（ScProc）に基づいて生成される。さらに、製品については製品グループ（ScPrdGrp）、各製品（ScPrd）、および納期や数量の定められた受注単位であるロット（ScLot）からなる。

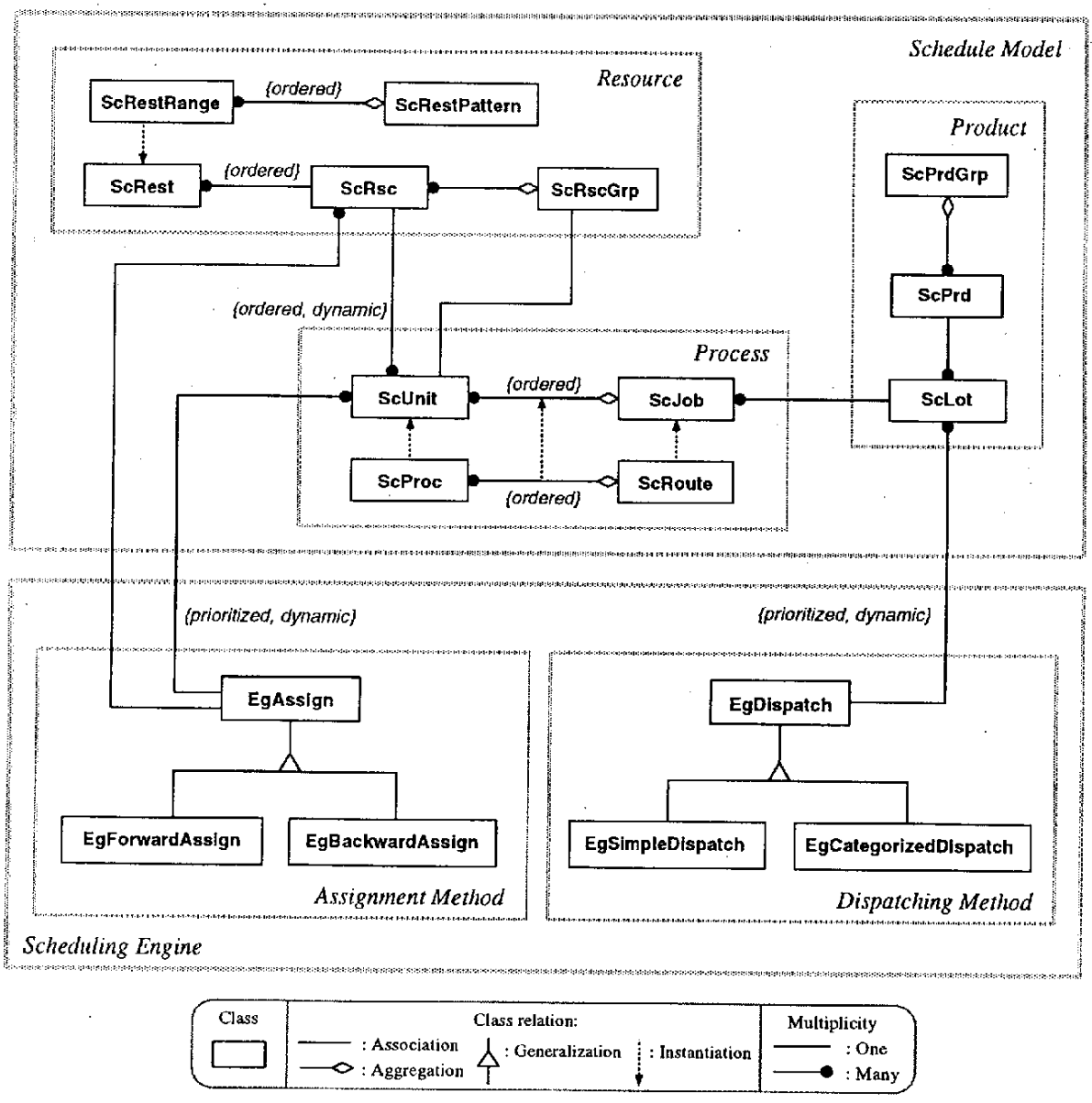


図３．５．４-１スケジュール・モデルとスケジューリング・エンジンの構成

一般にスケジューリング問題は、割付対象としてのユニットと割付先である資源と

の間に多対一関係を動的に設定するものとして特徴付けられる（図３．５．３-２）。この関係は多くの割付問題ならびにスケジューリング問題において共通に見られるものである[Tijerino 93][Hori 94][Sundin 94]。そして、こらの割付問題はすべて供給対象（supplies）に需要者（demands）を割り付けると言う点において共通している[Poek 92]。しかしながら、スケジュール・モデルではユニットと資源の多対一関係に加えて生産工程をはじめとする製造分野に特徴的な問題領域の性質がモデル化されており、より具体的な適用分野を想定したドメイン・オントロジーとして位置付けられる。そして、そこに伴われる概念やそれらの関係は、問題解決（スケジューリング・エンジン）や内部状態の視覚化（ユーザインタフェイス）における前提を明示的にするものとなっている。

(2) ユーザインタフェイス

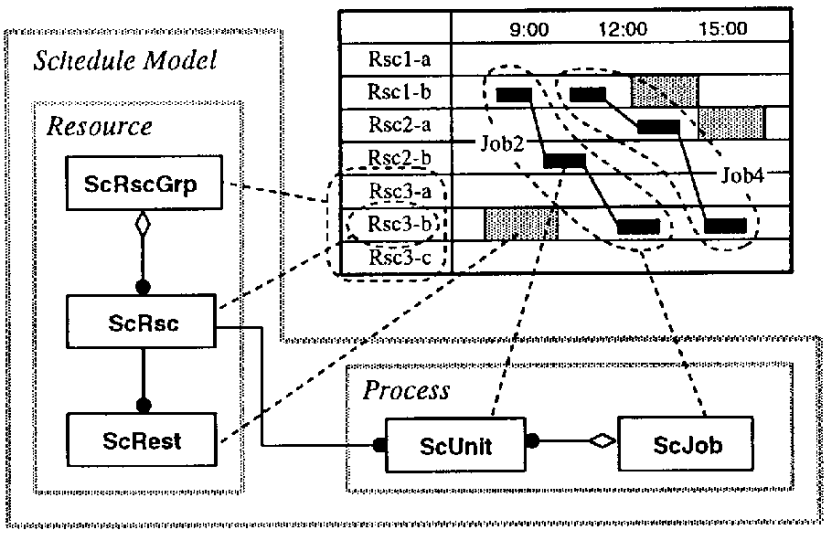
生産スケジューリングでは、作業の基本単位であるユニットが特定の資源に割り付けられるとともに、作業の開始／終了時刻が設定される。このような資源とユニット間の割付関係は、資源中心とジョブ中心の２つの観点から視覚化することができる。資源中心の視覚化は各資源（ScRsc）ごとにその資源と関連付けられた複数ユニット（ScUnit）が同一時間軸上に表示される（図３．５．４-２(a)）。一方、ジョブ中心の視覚化ではジョブ（ScJob）ごとにそのジョブと関連付けられた複数ユニットが同一時間軸上に表示される（図３．５．４-２(b)）。

このようなスケジュール・チャートについてはスケジューリングの分野で従来から良く知られている[Mulvehill 88]。ここで重要な点は、ドメイン・オントロジーとしてのスケジュール・モデルを前提とすることによって、スケジュール・チャートとしてのタイプが網羅されていることが明らかとなることである。

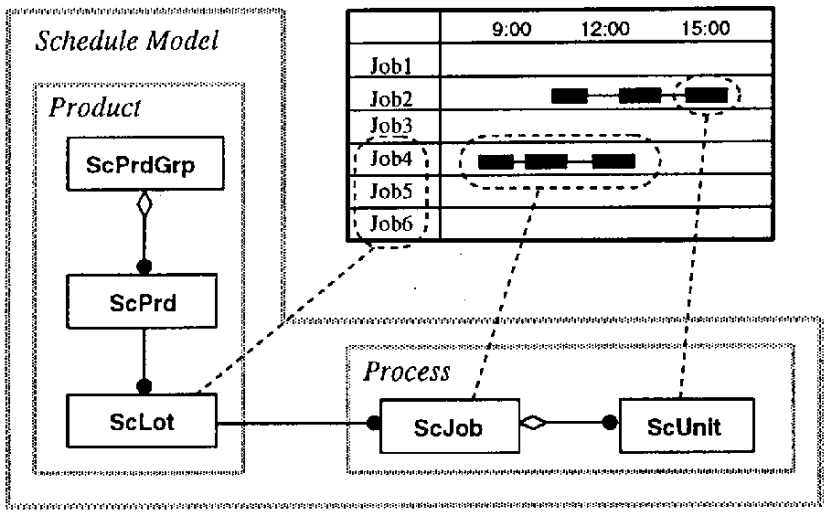
すなわち、図３．５．４-１のスケジュール・モデルの構造は複数のユニットを直接取りまとめる単位となり得る概念が資源（ScRsc）とジョブ（ScJob）のいずれかしかあり得ないことを明示的に表現している。逆に、資源あるいはジョブ以外の概念を介して複数ユニットを配置するスケジュール・チャートが必要ならば、その概念はこの分野において本質的役割を果たすものであり、スケジュール・モデルに追加されなければならない。

さらに、スケジュール・チャートに関するこのような区分は、生産管理における業務区分にも対応している。生産管理おもに生産（production）と製造（manufacturing）の２つの部門によって役割分担がなされる。生産部門はロットの適正投入や顧客に対する納期の遵守が主要な責務である。スケジュール・チャートにおけるジョブ中心の観点は、一連の製造工程を経てロットと関連付けられたジョブが最終作業を完了するまでの経緯を視覚化するものとなっている。それに対して、製造部門は各工程におけ

る個別の作業（ScUnit）の進捗を管理し、設備の稼働率を向上させることに対して責任を負っている。資源中心のスケジュール・チャートは、各工程での作業者に対して設備の稼働状況や前工程における作業の進み具合などを監視する上で自然な観点を提供している。



(a) Resource-centered view



(b) Job-centered view

図３．５．４-２　異なる観点からのスケジュール・モデルの視覚化

(3)　スケジューリング・エンジン

スケジューリング・エンジンのサブシステムには、割付（図３．５．４-３）とディス

パッチング（図３．５．４-４）の２種類のスケジューリング手順が伴われる。図３．５．４-１に示したようにすべてのロット（ScLot）は製品が完成するまでの一連の作業であるジョブ（ScJob）を介して生産現場での作業単位であるユニット（ScUnit）と関連付けられる。上記の２種類のスケジューリング手順が担う役割は、このユニットとロット間の部分全体の関係に着目することによって明確にすることができる。すなわち、割付手順は各作業（ScUnit）の開始／終了時刻を使用する生産設備（ScRsc）とともに決定するもので、現場での詳細な作業の実施に関わるものである。それに対して、ロットの優先順位を決定するディスパッチング手順は作業の実施手順に関わらずより大域的な状況での作業指針を策定するものとなっている。このような前提条件は、割付あるいはディスパッチングのスケジューリング手順それぞれにおいて明示的にできるものではなく、ドメイン・オントロジーであるスケジュール・モデルを参照することによって可能となる。

1. **reset** : Initialize a unit (**ScUnit**) queue.
2. **isDone** : If there exists any unit in the queue.
3. **doNext** : Assign a unit to an appropriate resource (**ScRsc**);
   3.1) Select the first unit in the queue
   3.2) Check the unit if it is allowed to modify
   3.3) Select a resource for the unit
   3.4) Assign the unit to the resource
   3.5) Modify dependent time
   3.6) Detect violation
   3.7) Reprioritize the queue.

図３．５．４-３　割付の処理手順

1. **reset** : Initialize a lot (**ScLot**) queue.
2. **isEmpty** : If there exists any lot in the queue.
3. **removeTop** : Provide the lot with the highest priority;
   3.1) Select the first lot in the queue
   3.2) Remove the first lot from the queue
   3.3) Reprioritize the queue.

図３．５．４-４　ディスパッチングの処理手順

このように、図３．５．４-１に示したスケジュール・モデルとスケジューリング・エンジンは、領域固有の概念が問題解決手順における推論操作によってどのように用いられるかを明示的にするもので、全体としてアプリケーション・オントロジー

[Gennari 94]を構成しているとみなすことができる。Gennariらのアプローチでは、領域に依存しない問題解決手順が特定の対象領域において再利用されカスタマイズされる段階で、推論操作の領域概念への対応関係（domain mapping）が決定される。それに対して、SCOOPではそのような対応関係は再利用の対象として提供される部品ライブラリの設計として予め規定される。これは再利用知識を設計する上での有用性と一般性のトレードオフに関わる問題で、包括的に解決されるものではない。しかしながら、そのようなトレードオフに対して有用な原則や指針を提示するには、再利用知識の背後にある知識構造をオントロジーとして明示的にしていかなければならない。

参考文献

[Gennari 94] J. H. Gennari, S. W. Tu, T. E. .Rothenfluh, and M. A. Musen: Mapping domains to methods in support of reuse. International Journal of Human-Computer Studies, Vol.41, pp.399-424 (1994).

[Hori 94] M. Hori, Y. Nakamura, and T. Hama: Configuring problem-solving methods: a CAKE perspective. Knowledge Acquisition, Vol.6, No.4, pp.461-488 (1994).

[Hori 97] M. Hori: SCOOP: Scheduling Components for Production Control Systems. Research Report RT-0216, Tokyo Research Laboratory, IBM Japan Ltd. (1997).

[Hori 98b] M. Hori and T. Yoshida: Domain-oriented library of scheduling methods: Design principle and real-life application. International Journal of Human-Computer Studies (to appear).

[Mulvehill 88] A. Mulvehill: A user interface for a knowledge-based planning and scheduling system. IEEE Trans. on Systems, Man and Cybernetics, Vol.18, No.4, pp.514-521 (1988).

[Poek 92] K. Poek and F. Puppe: COKE: Efficient solving of complex assignment problems with the propose-and-revise method. Proceedings of Fourth International Conference on Tools with Artificial Intelligence, pp.136-143, Arlington, VA (1992).

[Sundin 94] U. Sundin: Assignment and scheduling. In J. Breuker and W. V. de Velde, Eds. The CommonKADS Library for Expertise Modelling. pp.231-264, IOS Press, Amsterdam (1994).

[Tijerino 93] Y. Tijerino and R. Mizoguchi: MULTIS II: Enabling end-users to design problem-solving engines via two-level task ontologies. In N. Aussenac, G. Boy, B. Gaines, M. Linster, J.-G. Ganascia, and Y. Kondratoff, Eds. Knowledge Acquisition for Knowledge-Based Systems, pp.340-359, Lecture Notes in Artificial Intelligence 723, Berlin: Springer-Verlag.

3.6　エンタープライズ・モデル

近年、ビジネスオブジェクトとか業務統合（Enterprise Resource Planning）などの言葉がよく聞かれるようになり、また、それに関連したワークフローツールや業務統合パッケージソフトが普及してきた。ビジネス活動を快適に支援していく計算機環境を実現するには、様々な要素技術を統合していくことが重要であるが、その中でも、業務活動のモデリング技術は、中心的な役割を担うと考えられ、業務活動に現れる様々なビジネス概念の体系化をはかったエンタープライズオントロジーの研究開発プロジェクトが活発化してきた。

以上の背景から、本節では、エジンバラ大学のエンタープライズプロジェクト、ビジネスプロセス記述用に広く使われているIDEFファミリーの中で、オントロジーに関連するIDEF 5、さらに、国際産学共同で推進されているPIFプロジェクトについて述べる。

3.6.1　エジンバラ大学のエンタープライズプロジェクト[Enterprise URL]

エジンバラ大学では、エンタープライズをモデリングするための方法論とツール群を開発するためのプロジェクトが進められている。エンタープライズオントロジーの開発は、そのプロジェクトの一環として成され、人と人、あるいは、人とシステム、さらには、システムとシステム間のコミュニケーションをはかるためのメディア（a communication medium）として大きな役割を担うとされている。また、企業活動に関わる様々な知識の獲得・表現・操作・構造化・組織化等を支援する役割も担うとされている。

エンタープライズオントロジーは、基本的には、企業活動に関わる様々な概念の体系であるが、エジンバラ大学では、下記の手順により開発が進められた。

<u>エンタープライズオントロジー開発手順</u>

(1) Scope：ブレーンストーミングにより、重要なビジネス概念（単語と語句）群を構造化されない形で洗い出した後、意味が似ている（あるいは、参照関係にある）概念群をwork areaと呼ばれるグループにまとめあげていく作業を通しながら、ビジネス概念群が取捨選択され、エンタープライズオントロジーの記述範囲が明らかになっていった。結果的に、Activity、Organization、Strategy、Marketing、Timeといった５つのwork areaが整理され、エンタープライズオントロジーの主要な構造となった。また、概念の意味は、広く一般的に受け入れられている意味というよりは、企業活動に関連する特別な意味になる事が多かった。

(2) Choosing Terms：エンタープライズオントロジーで使用する語句は、通常の意味とはかなり異なることがあっても気にかけずに、企業活動においてよく使われ、あいまいさが除去されることを重視して、決定されていった。
(3) Definitions：各work area において、少数の基本的概念を選出して定義を与えた後、その基本的概念を利用して他の概念を定義した。概念定義に使われるプリミティブ（Entity、Relationship、State of Affairs、Role など）はMeat-Ontologyと呼ばれるが、プリミティブ数を少なくすることによって、形式性が高められた。

以下、work area ごとに、整理された主要な概念の意味を示す。

（1） Activity etc.

Activity	：あらゆる行為
Doer	：行為者（Personか Organizational Unitか Machine）
Capability	：Doerになるためのスキルや性能
Resource	：Activityで使用される資源
Effects	：Activityの結果生じるもの
Plan	：Activityを実行計画（より詳細なActivity）
Purpose	：Activityは、Strategyの概念Purposeと関連づけられる。
Process Specification	：Planの仕様
Authority	：Doerを認める権利（Activityのコントロールにつながる）

（2） Organization

Legal Entity	：法的権利と責任をもつ実体（Person、Corporation）
Organizational Unit (OU)	：組織内で単に認められた実体
Machine	：人でもLegal Entity でもないもの
Ownership	：法的観点からの権利と責任の所有権
Management Link	：組織における管理構造
Manage	：OUに対してPurposeを割り当てる。
Organizational Structure	：OU間のManagement Linkのパターン

（3） Strategy

Purpose	：Planの実行によって達成されるものか、OUが責任をとれるもの（Vision、Mission、Goal、Objective などに分かれる）
Strategy	：ハイレベルのPurposeを達成するためのPlan
Strategic Planning	：Decision、Assumption、Risk などのファクターを使って表現される。

（4） Marketing

Sale	：VendorとCustomer間でProductをある価格（Sale Price）で交換するための合意事項（なお、既に合意されたSaleと予想されている将来のPotential Saleに分かれる）
Market	：すべてのSaleとPotential Sale
Market Segment	：Product、Vendor、Customer、Sale Price などのSale に関連した属性（Segmentation Variableと呼ぶ）によってMarketを分割したもの
Market Research	：Marker Researchによって、ProductのFeature、CustomerのNeeds、BrandやProductやVendorのImageを分析する。
Promotion	：PurposeとImageを関連づけるActivity

（5） Time

Time Line	：時間軸上の線区分であり、Duration やTime Intervalや Calendar Date などに分かれる。
Time Point	：時間軸上の点。
Before/After	：Time Point 間の関係
Disjoint/During/Overlaps	：Time Interval間の関係。

表３．６．１-１に、上記の主要概念を含む、エンタープライズオントロジーに含まれるすべての概念のリストを示す。なお、このエンタープライズオントロジーは、最終的にスタンフォード大学知識システム研究所で開発されたオントロジー記述言語Ontolinguaによってコーディングされている。

３．６．２　ＩＤＥＦファミリーにおけるオントロジー[Idef URL]

IDEF（Integration DEFinition Language）は、元来、米国空軍において、航空機の発注仕様を標準化するために開発された仕様記述言語であり、構造化分析手法SADT（Structured Analysis and Design Technique）をベースにして開発されたものである。IDEFは、当初、IDEF０（機能モデル）、IDEF１（情報モデル）、IDEF２（動的モデル）の３つのモデルが規定されたが、その後拡張され、現在は、１６のモデルが提案され、総称してIDEFファミリーと呼ばれている。現在、商用化されよく使われているのは、IDEF０とIDEF 1X（データモデル）であるが、近年、オントロジー記述用言語としてIDEF 5が提案され、試作されている。IDEFは、企業の様々な業務活動のモデリングに利用されるため、IDEF 5は、３．６．１のようにトップレベルの組織活動をモデル化す

表3.6.1-1　エンタープライズオントロジーに含まれる諸概念

ACTIVITY etc.	ORGANIZATION	STRATEGY	MARKETING	TIME
Activity	Person	Purpose	Sale	Time Line
T-Begin	Machine	Hold-Purpose	Potential Sale	Time Point
T-End	Corporation	Purpose-Holder	For Sale	Calendar Date
Pre-Condition	Partnership	Objective	Vender	Relative Time Point
Effect	Partner	Vision	Actual Customer	Duration
Doer	Legal Entity	Mission	Potential Customer	Duration Bounds
Sub-Activity	Organizational Unit	Goal	Customer	Time Interval
Activity Decomposition	Manage	Achieve	Reseller	Before
Authority	Delegate	Help Achieve	Product	Same or Before
Activity Owner	Management Link	Strategy	Asking Price	After
Event	Organizational Structure	Strategic Planning	Sale Price	Same or After
Plan	(Non-)Legal Ownership	Strategic Action	Market	Distance
Execute	Ownership	Decision	Segmentation Variable	Earliest Start Time
Sub-Plan	Owner	Assumption	Market Segment	Latest Start Time
Planning	Asset	(Non-)Critical Assumption	Market Research	Earliest End Time
Process Specification	Stakeholder	Influence Factor	Brand	Latest End Time
Capability	Contract of Employment	(Non-)Critical Influence Factor	Image	Interval Before
Skill	Share	Critical Success Factor	Feature	Interval During
Resource	Shareholder	Risk	Need	Interval Overlaps
Resource Allocation			Market Need	Interval Disjoint
Resource Substitute			Promotion	
			Competitor	

ることというよりは、もっと一般的なレベルで整理されている。以下、IDEF 5の概要を述べる。

まず、IDEF 5で提案しているオントロジー開発手順を示す。

(1) Organizing and Scoping：オントロジー開発プロジェクトの目的などを明確にし、プロジェクト構成員に役割を割り当てる。
(2) Data Collection：オントロジー開発のために必要な諸データを収集する。
(3) Data Analysis：オントロジー構築に向けて、諸データを分析する。
(4) Initial Ontology Development：初期オントロジーを構築する。
(5) Ontology Refinement and Validation：初期オントロジーをチェックし精錬する。

IDEF 5では、上記の開発プロセスを支援するために、IDEF 5 Schematic 言語とIDFE5 Elaboration 言語という２つの言語を提供している。前者は、概念間の関係を図的に書け、ユーザにとって把握しやすいが、表現能力が限定されている。一方、後者は、テキスト形式で定義を与えていく言語であり、読解性は劣るが、厳密に定義する時に効果があり、表現能力は高く、一階論理で表現できるものはすべて表現可能である。また、IDEF Elaboration 言語では、IDEF 5 関係子ライブラリー（Relation Library）という、概念関係子を記述するためのライブラリーが利用可能であり、オントロジーを効率よく開発できることが大きな特徴となっている。

IDEF 5 関係子ライブラリーは、下記のように７つのカテゴリーに分類され、さらに、これらのカテゴリーは、６８個の関係子に細分化される。また、関係子について成立する諸性質を１３０個の公理群によって記述されている。以下、代表的な関係子について、説明を加える。

・分類関係子（Classification Relations, Class Inclusion Relations）
・メロニミック関係子（Meronymic Relations）
・時間関係子（Temporal Relations）
・空間関係子（Spatial Relations）
・影響関係子（Influence Relations）
・依存関係子（Dependency Relations）
・事例関係子（Case Relations）

（1） 分類関係子

分類関係子は、直感的には IS-A として捉えられるが、下記のようなより詳細な包含関係子をもつ（：以下で例を示す）。

・Function-Inclusion relation（機能に関する包含関係子）：A hammer is a tool.
・State-Inclusion relation（状態・状況に関する包含関係子）：Hate is an emotion.
・Activity-Inclusion relation（活動に関する包含関係子）：Tennis is a sport.
・Action-Inclusion relation（行為に関する包含関係子）：Lecturing is a form of talking.
・Perceptual-Inclusion relation（知覚対象に関する包含関係子）：A cat is a mammal.

(2) メロニミック関係子

メロニミック関係子は、基本的には、全体部分（Part-Of）関係を記述するためのものである。まず、メロニミック関係子は、物理的全体部分関係子（Physical Part-Of対して適用される場合）と概念的全体部分関係子（Conceptual Part-Of）に分けることができる。物理的全体部分関係子は、地理的な対象（地名など）に対して適用される場合（Place-Within）、物とその構成物に対して適用される場合（Component-Of）、物とその原料に適用される場合（Stuff-Of）、類似した物に適用される場合（Portion-Of）という4つの関係子に細分化される。一方、概念的全体部分関係子は、集合とその要素に対して適用される（Member-Of）と、活動とそれを構成する活動に対して適用される（Activity-Within）という4つの関係子に細分化される。以下、これら6つの関係子について例を挙げる。

・Place-Within：The Alps are part of Europe.
・Component-Of：A wheel is a part of a bicycle.
・Stuff-Of：A chair is partly wood.
・Portion-Of：A yard is part of a mile.
・Member-Of：A tree is part of a forest.
・Activity-Within：Paying is part of shopping.

(3) 時間関係子

時間関係子は、様々な時間関係を記述するが、time-point間の関係子とtime-interval間の関係子に大きく分かれる。前者については、time-pointが順序づけられ、before関係子によって明示的に記述される。一方、後者については、2つのtime-interval（beginning（開始時点）、end（終了時点）、duration（持続時間）という3つの属性をもつ）間の関係が、John Allenの時間論理の研究を基礎にして、equals、before、during、overlaps、meat、start、finishという7つの時間関係子が、下記のように整理されている（図的に示す。なお、XXXとYYYが2つのtime-intervalである）。

X equals Y	XXX YYY	X before Y (or Y after X)	XXX YYY
X during Y (Y contains X)	XXX YYYYYYY	X overlaps Y (Y overlapped-by X)	XXX YYYY
X meats Y (Y mets-by X)	XXXYYY	X starts Y (Y startrd-by X)	XXX YYYYYY
X finishes Y (Y finished-by X)	XXX YYYYYY		

(4) 空間関係子

2つの物の空間的関係を示すために、17個の空間関係子が提供されている。ここでは、それらを提示するに留める。

Left-of, right-of, above, below, behind, in-front-of, inside, outside, between, far, near, touching, beside, adjacent, disjoint, intersect, coincident

(5) 影響関係子

影響関係子は、ある物の変化が別の物の変化を引き起こすことを記述する関係子であり、どのような影響が起こるか不明の時、influenceという影響関係子を使う。また、変化の方向が明らかな時、(influences-pp X,Y)（Xが増加すればYも増加する）、(influences-pm X,Y)（Xが増加すればYは減少する）、(influences-mp X,Y)（Xが減少すればYは増加する）、(influences-mm X,Y)（Xが減少すればYも減少する）という4つの影響関係子が提供されている。

(6) 依存関係子

依存関係子は、2つのものに存在する依存関係を記述する関係子であるが、XがYに何かしら依存している場合に、(depends-on X Y)という影響関係子を使う。また、依存関係が以下の場合に当てはまる時、すなわち、Xの存在がYの存在に依存している（Yが消滅すればXも消滅する）場合は、(depends-on-existentially X Y)という影響関係子を使う。さらに、Yが存在して、その結果、Xが存在する（Yが消滅してもXは消滅しない）場合は、(depends-on-causally X Y)という影響関係子を使う。

(7) 事例関係子

事例関係子は、今までのように2つのものの間に成立している関係を記述するので

はなく、イベントを記述するための構造を与えるものであり、agent-action、agent-instrument、agent-object、agent-recipient、action-instrument という５つの関係子に分かれる。以下、これら５つの関係子について例を挙げる。

・Agent-Action（行為者と行為）：dog-bark
・Agent-Instrument（行為者と行為者が使用する道具）：A skier-skis
・Agent-Object（行為者とそのイベントに現れる物）：baker-flour
・Action-Recipient（行為とその行為を受けるもの）：lay down-bed
・Action-Instrument（行為とその行為に利用される道具）：paint-brush

3.6.3　ＰＩＦプロジェクト[Pif URL]

1993年頃までに、欧米各地ではBPR（Business Process Reengineering）に関するプロジェクトが開始されていた。すなわち、MIT におけるプロセスハンドブックプロジェクト（種々のプロセスモデルを参照・比較できる電子ハンドブックの開発）、DECにおけるSpark プロジェクト（BPR のライブラリーを整備していくためのツールの開発）、スタンフォード大学における VDT（Virtual Design Team）プロジェクト（ビジネスプロセスをモデル化し評価する環境）、トロント大学におけるエンタープライズモデリングプロジェクト（プロセス、時間、資源、成果物、品質、組織などを表現するための形式的定義の提供）などが進行していた。これらのプロジェクトには、ビジネスプロセスの設計方針やモデリングや表現方法などに差異があったが、情報を交換することは大変有用であることから、異なった形式で表現されたプロセス記述を変換して、プロセス記述を共有できることを目標としたPIF（Process Interchange Format）プロジェクトが、これらの組織間で開始された（後に、エジンバラ大学も参加）。

PIFプロジェクトでは、基本的なビジネスプロセスに関わる諸概念を記述したPIF-COREと呼ばれるPIFオントロジー（クラス階層として提供）、プロセス記述をPIF形式に相互変換するトランスレータ、PIFクラスを拡張するためのPIFフレームワークなどが開発された。PIFオントロジーは、以下の手順を通して開発された。

・計算効率よりは一般性を重視して、ビジネスプロセスの体系化を計った。
・PIF 構成子は、流布しているプロセス記述言語であるIDEFやペトリネットの構成子を表現できるようにした。
・PIFオントロジーは、必要最小限のクラス群から構成して、必要に応じて拡張された。
・誰でもPIFクラスの追加を提案できるが、その提案はPIF-WGによって判断される。

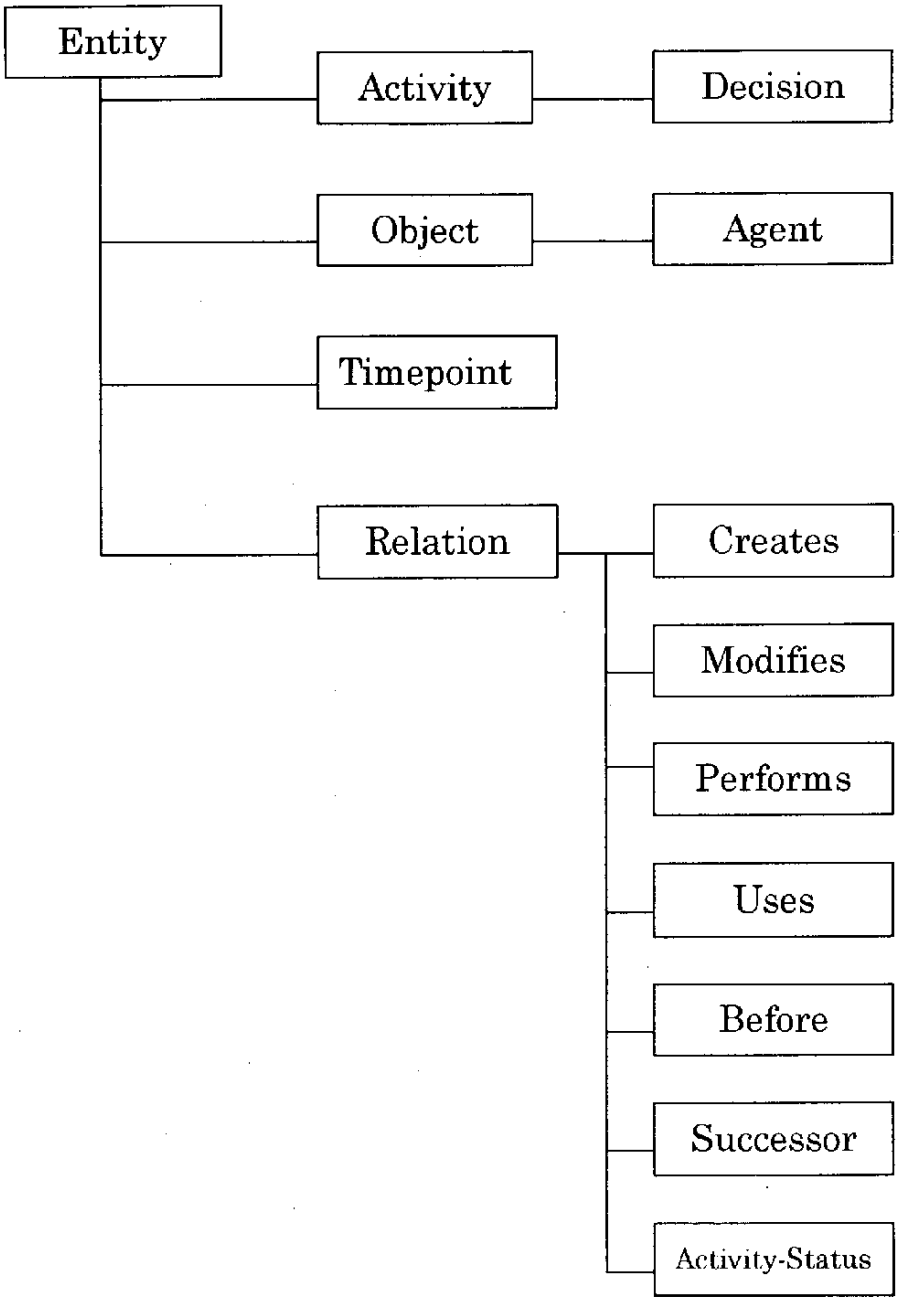


図３．６．３－１　PIF-COREクラス階層

図３．６．３－１にPIFオントロジー（PIF-COREクラス階層）を示す。PIFにおいては、あらゆるものをEntity とみなし、クラスEntity は、Activity、Object、Timepoint、Relation という４つのサブクラスに分けられる。各クラスは、フレーム形式で定義が与えられている。

Activity は、時間経過を伴うものであり、Begin とEndによって開始時点と終了時点を示す。Componentによりサブactivityを示す。また、Preconditionは、開始時刻 Beginにおける Activity の開始条件であり、Postconditionは、終了時刻 Endにおける Activityの終了条件であり、それぞれ PIF-sentences（論理式）によって記述される。

Activity

Attribute	Value Type	Multiple Values Alowed
Component	Activity	Yes
Precondition	PIF-sentence	No
Postcondition	PIF-sentence	No
Begin	Timepoint	No
End	Timepoint	No

Objectは、Activityで生成・使用・修正される任意のものであり、Agentやプロダクトを生成するための材料などを含む。Objectの定義は、最終的に定まっていない。

Timepointは、時間の一点を示すが、PIF-COREでは、Activity の開始時刻と終了時刻などに使用される。Timepoint は、構造的定義を持たない。

Relationは、PIF構成子間の関係を示すものであり、属性を持たないEntityである。Relationは、Creates/Uses/Modifies（Activityで参照されるObject間の関係を表現する）、Performs（Agent とActivityの関係）、Before（Timepoint 間の時間的関係）、Successor（Activityの順序関係）、Activity-Status（Activityの実行状態）に分かれる。

PIFは、異なったプロセス記述の共通仕様を与えるものであるが、組織に固有のビジネスプロセスの表現も存在するので、PIFのクラス記述をローカルに拡張できることも必要である。この機能を提供するのが、PSV（Partially Shared Views）である。PSVは、異なったクラス階層を使って、プロセスを変換する組織ペアが異なれば、変換方法も変えることができるようにする。例えば、組織 Aはクラス Agent だけをもち、組織 Bはサブクラス Employee までもつとする。この場合、組織 Bのプロセス記述を組織 Aに変換する時、EmployeeはAgentに変換される。また、Employeeが特殊な属性 Pをもつ時、ユーザが組織 A上でPの定義をシステムに与える。このように、変換時に、情報が欠落す場合は、ユーザがサポートすることになる。

参考文献

[Enterprise URL] http://www.aiai.ac.uk/~entprise/enterprise/ontology.html

[Idef URL] http://www.idef.com/idef5.html

[PifURL] http://ccs.mit.edu/pif/index.html

3.7　設計（CAD 関連）

3.7.1　TRADEプロジェクト

TRADEは、アムステルダム自由大学において行われている研究プロジェクトであり、その名称は"Toolkit for Requirements and Design Engineering"のacronymに由来している。TRADEの開発プロジェクトの目的は、要求工学のためのテクニックとヒューリスティクスを集めた道具を提供することにある。このプロジェクトでは、ソフトウエア要求工学を、開発したいソフトウエア製品自身、その動作環境および対象とするドメインについての概念モデルを発見し、分析し検証する過程と捉えている。したがって、ここでの概念モデルは、製品ソフトウエア、その動作環境およびドメインを表現する。TRADEプロジェクトは、２つの構成要素、LCMとTCMから成っている。

LCMは、概念モデル用言語（Language for Conceptual Modeling）、TCMは、概念をモデル化するためのツール（Toolkit for Conceptual Modeling）である。LCMは、多層動的論理に基づく形式言語である。TCMは、概念モデルを構築するための記法とヒューリスティクスの集成であり、ワークベンチとして実装されている。現在のところ、ワークベンチ上では、ERダイアグラム、データフローダイアグラム、状態遷移図、ジャクソン構造化図式およびクラス階層図式など、あるいは機能展開表のような様々な種類の対応表を利用できるようになっているとともに、EMACSエディタにLCMモードを追加してLCM記述を直接構文検査できるようになっている。TCMのツールを利用することによって、幾つかのダイヤグラムや構造化文書によって構成される製品要求の準形式的概念モデルを生成することができる。このとき、形式定義言語LCMによる形式的仕様を準形式的概念モデルと明確に対応づけることができる。

現在のところ、TRADEプロジェクトにおける具体的適用例は報告されていない。

3.7.2　ESPRITプロジェクトにおけるドイツ語圏の関連活動

ESPRITプロジェクトでは、様々の活動が行なわれているのでOntology関連プロジェクトは他にもあると予想されるが、本項では主にドイツ語圏で行われている機械系製品データ共有のための知識共有の動き（ESPRIT9049）を、ワークショップ報告から概観する。このワークショップは、"Workshop on Product Knowledge Sharing for Integrated Enterprises"（ProKSI'96）と呼ばれるもので、1996年10月末にスイスのバーゼルで開催された。主な目的は、人工知能とSTEP規格開発を結合することにあった。会議の結論として、統合化された製品情報モデルの構築におけるOntologyの重要性が認識され、製品知識表現におけるターミノロジーの論理表現の適用可能性検討の必要性が指摘さ

れた。

ここで、STEPとは、Standard for Exchange of Product dataと呼ばれ、製品定義情報の交換を目的としてISO 10303として規格化作業が進められているものである。製品定義情報交換は、並行設計（Concurrent Design）および世界規模の分散製造などの目的に有効であるとされている。当初、機械部品などのユークリッド幾何形状情報の交換を目指して作業が始まり、既に第1版が一応リリースされているが普及しているとは言えない。近年は、これに加えてAP（Application Protocol）と呼ばれるドメイン依存の部分規格の制定作業が進められている。現在、具体的になっている分野は、化学プラント、造船、自動車である。なお、本項でいうSTEPにはCompanion StandardであるISO13584（Plib）が含まれている。Plibについては後述するが、標準部品の諸元情報の配布を規格化対象としており、非標準的製品情報の交換を規格化対象とするSTEPとは相補的関係にある。以下に会議録の中からOntologyに関連する発表を概説する。

（1） Ontology-based development of integrated product models;

H.Grabowski, E. Meis (University Karlsruhe)

ドイツの基礎研究プロジェクトSFB346（Sonderforschungsbereich 346: Rechnerintegrierte Konstruction und Fertigung von Bauteilen）の成果報告である。ここでは、統合化製品モデル開発の品質と効率向上ためにOntology Engineeringの手法を適用しようとしている。統合化製品モデル開発において非常に重要な問題は、既開発の製品モデルに新しい観点のデータ項目を追加、すなわち新しい概念の統合化を行なう必要が生じることである（注：この問題の解決は、製品ライフサイクルのスコープの拡大を社会的に要請されている現在、さらに重要性を増している）。Ontology Engineeringの基本アイデアは、概念的統合化を実際の統合モデルを構築する前工程と捉えること、および概念的統合化の工程を明示化すること、出来れば形式化することである。SFB346では、一つのオントロジーは基礎概念（Ontological Concept）の集合として表わされる。それぞれの基礎概念は、それぞれの統合化製品モデルの構成要素の意味を表現するために用いられる。概念的データの統合化とはデータ分類の特殊ケースと考えられるので、一つのオントロジーを表現するための言語は、分類（Classification）言語ということになり、Ontolingua言語を利用している。これによって、目的の一つであった概念的統合化の工程の明示化が実現された。

（2） Modelling of reusable product knowledge in terminological logics ? a case study;

T. Liebig, D. Roesner (University Magdeburg).

TechDocと呼ばれるプロジェクトの成果の一部の報告である。このプロジェクトの目的は、その名前から推測されるように技術文書を対象としたもので、製品保守などの製造活動に利用するための多言語文書を自動的に生成することである。多言語化の要

請は多言語経済圏であるEC全域に共通的に利用される必要性から生じており、また製品責任などの強化によって組織的文書生成の必要性が以前にも増して高まっていることが、プロジェクトの背景にある。文書生成に必要な知識は、特定製品固有なものと製品領域共通なものに分けて構造化されるが、後者は、製品機能、形状およびその他に関する再利用可能な知識として用語知識システムに表現される。この知識は、文書化の際に行なわれる定性シミュレーションにも利用可能なように設計されているという。このプロジェクトでは、知識表現ツールとしてKL-ONEの一族であるLoomを用いて、Common LispとCLOSによって実装されている。ただし、用語知識システムを利用した知識共有を行なおうとしているが、オントロジーについてはそれ以上の報告はされていない。

3.7.3 IEC61360-2およびISO13584（P-Lib）におけるBSU（Basic Semantic Unit）

ISO13584は、P-Libとも呼ばれ、機械系CADシステムのための標準化活動であるISO TC184 SC4の中のWG2（Standard for the Neutral Representation of Standard Parts）において作業が進められている標準部品の中立表現に関する標準である。ISO TC184 SC4は前述の3.7.2においても述べたSTEP（ISO 10303）と呼ばれる製品モデル情報交換の規格を主に制定するものであり、P-Libは、それを補うものであるが、IECと共同でBSUを用いた共通辞書方式を採用してやや異なる立場にある。BSUは、ターミノロジー学を基礎に持ち、それを利用するBSR（Basic Semantic Repository）とともに機械系CADシステム間の情報交換に重要な要素になりつつあるが、EC（Electric Commerce）においても利用されると見られる。本項では、主にISO13584の規定に基づいてBSUの詳細を示す。

ISO13584-42（Part42: Methodology for structuring part families）では、部品の分類と個々の部品を諸元項目により記述するための枠組みが規定されている。ISO13584-42はIEC TC3 SC3Dとの共同開発によるもので、実際は国際規格として IEC61360-2: 1997（Standard data element types with associated classification scheme for electric components-Part2: EXPRESS dictionary schema）のみが存在し、ISO13584 Part42はこれを引用する形を取っている。

（1） ISO13584_IEC61360_DICTIONARY_SCHEMA

ISO13584-42 （IEC61360-2） は、ISO13584_IEC61360_DICTIONARY_SCHEMA とISO13584_ IEC61360_LANGUAGE_RESOURCE_SCHEMAから構成されており、中心となるのはISO13584_IEC61360_DICTIONARY_SCHEMAである。

ISO13584_IEC61360_DICTIONARY_SCHEMAは、 Basic Semantic Unit （BSU）、Dictionary Element、 Content Itemと呼ばれる3つの抽象データ型を中心として構成されている。BSUは、ある概念に割り当てられる言語に依存しないラベルであり、このBSUを介して概念の定義（Dictionary Element）、あるいはその具体化（Content Item）を参照することができる。Basic Semantic UnitおよびDictionary Elementはそれぞれ特殊化されて、供給者（Supplier BSU / Supplier Element）、製品定義（Class BSU / Class）、諸元項目（Property BSU / Property DET）、データ型（Data Type BSU/Data Type Element）に関する辞書を構成する。

各々のBSUには定義されるスコープがあり、Supplier BSUは全世界で一意的となるように、Class BSUはSupplier BSUのスコープで、Property BSU、Data Type BSUはClass BSUのスコープで定義される。従って、ある供給者（Supplier BSU ='A'）の製品（Class BSU = 'B'）の諸元（Property BSU = 'C'）は、'A. B. C'の形で一意的に特定することができる。具体的な例として、諸元項目に関するBasic Semantic UnitとDictionary Elementの構成を次に示す。まず、Property BSUは、表3.7.3-1のようなフィールドを持つデータ型である。また、Property DETは、表3.7.3-2のようなフィールドを持つデータ型である。このように、人間が判読することのできる名称などは、Dictionary Elementに収められており、識別子としてのBSUのCode（文字列）は電子計算機による処理を念頭に置いたものである。

表3.7.3-1　property BSU のデータ構造

フィールド名	データ型	備考
Code	文字列	Basic Semantic Unit から継承
Version	文字列	Basic Semantic Unit から継承
Name scope	Class BSU の参照	この Property を定義した Class

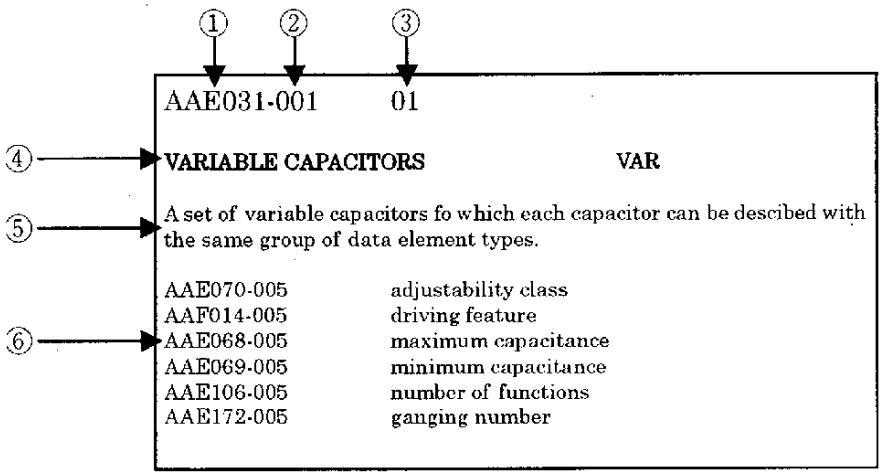
表3.7.3-2　property DET のデータ構造

フィールド名	データ型	備考
Identified by	対応 Property BSU の参照	Dictionary Element から継承
Time stamp	文字列	Dictionary Element から継承
Revision	文字列	Dictionary Element から継承
Names	Item Names 型	一般名称, 同義名称, 短縮名称
Definition	Definition type 型	文字列
Source doc of definition	Document 型	文字列による出典の引用など
Note	Note type 型	文字列
Remark	Remark type 型	文字列
Preferred symbol	Mathematical string 型	文字列或いは SGML
Synonymous symbols	Mathematical string 型	文字列或いは SGML
Figure	Graphics 型	
Det classification	文字列	
Domain	Data type 型	データ型と単位系
Formula	Mathematical string 型	文字列或いは SGML

(2) 標準辞書

ISO13584-42は枠組みであって、これを用いて記述するコンテンツを規定するものではないが、オントロジーの標準化と同様に、辞書には広く合意された内容が記述されることが望ましい。実際、ネジ、ベアリング、歯車などのようにJIS、ISO等で定められた規格に基づく製品は多い。これらの製品記述に、共通の標準辞書（コンテンツが標準化された辞書をここでは標準辞書と呼ぶ）が使えるならば製品情報の相互運用性が飛躍的に高まる。

ISO13584-42（IEC61360-2）に基づく標準辞書として、電子部品関連でIEC61360-4（Part4: IEC reference collection of standard data element types, component classes and terms）を挙げることができる。IEC61360-4では、製品の記述に必要なData element type（諸元項目）辞書、階層的に分類されたElectric/Electronic Component class辞書と諸元項目による各々のComponentの定義、およびData element typeの定義に用いられる用語の定義等から構成されている。図３．７．３-１、図３．７．３-２、図３．７．３-３にそれぞれData element type定義、Component Class定義、Component classificationの例を示す。



Class BSU 関連項目

① Code

② Version

Class (Dictionary Element)関連項目

③ Revision

④ Preferred name / Coded name

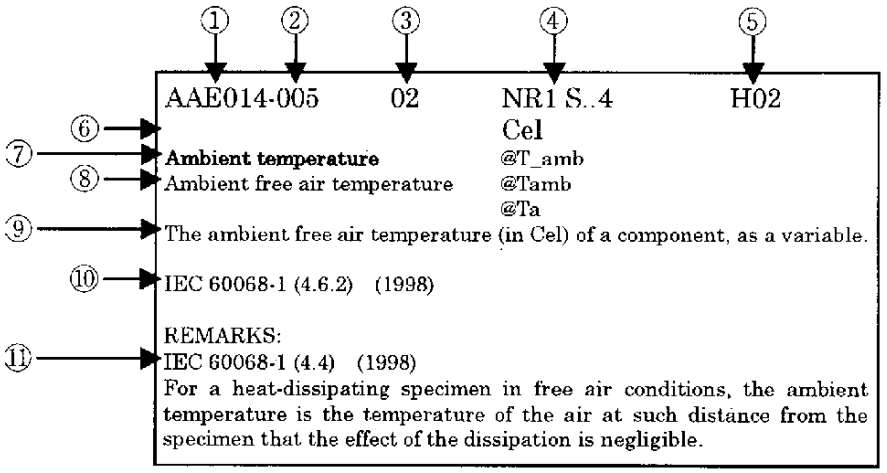
⑤ Definition

⑥ Class を記述する Data Element Type のリスト

図３．７．３-１　IEC61360-4によるComponent Class の定義例

なおISO13584-42では、部品の分類階層を作成する時の規則について次のように規定している。

1) 標準化された分類階層は、ISOまたはIECで規格化されている全ての部品をカバーしなければならない。
2) 部品供給者の分類階層は、部品供給者が提供する全ての部品をカバーしなければならない。
3) 標準化された分類階層は、International Classification of Standards（ICS）を基本として作成されなければならない。
4) ある階層で定義された諸元項目は、その階層に属する部品すべてに適用できなければならない。
5) 部品の見方に依存して異なる分類階層がありうる場合には、上位階層で定義された諸元項目の適用範囲が最も広くなる構造を採用しなければならない。



Property BSU 関連項目

① Code

② Version

Property DET(Dictionary Element)関連項目

③ Revision

④ データ型(符号付き整数 4 桁)

⑤ DET Classification(H02: Celsius temperature)

⑥ 単位系

⑦ Preferred Name / Short name

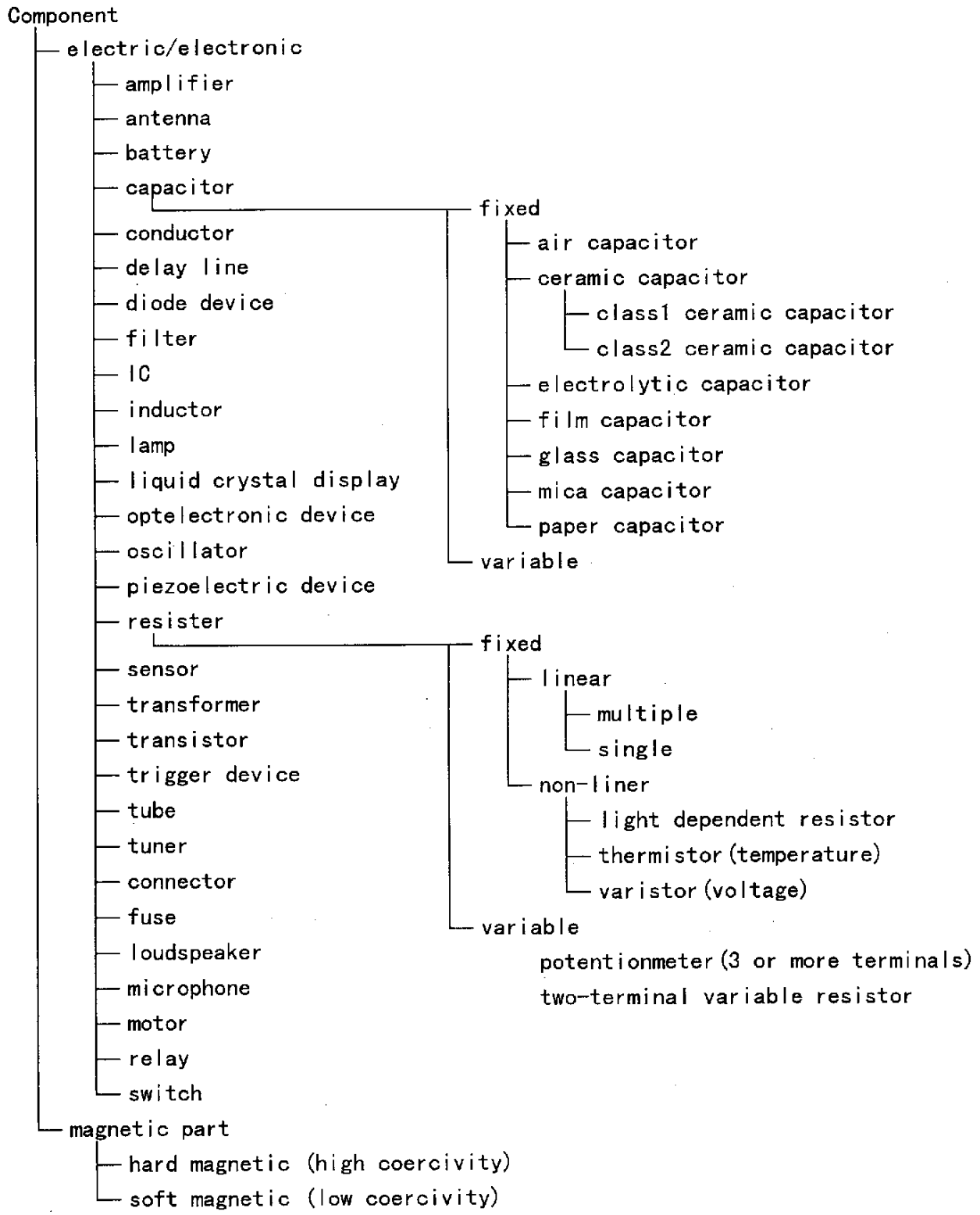
⑧ Synonymous name / Preferred letter symbol, Synonymous letter symbol

⑨ Definition

⑩ Source doc of definition

⑪ Remark

図３．７．３-２　IEC61360-4によるData element type の定義例



（一部分のみ分類階層を展開してある）

図３．７．３-３　IEC61360-4によるComponent Class の定義例

3.7.4 CONMOTOオントロジー

(1) 機械工学におけるオントロジー工学の必要性

機械工学は、その成立から力学とは深く関係してきたが、それゆえもあって理学優位の意識の強い分野である。機械技術は、原理的には物理学や化学などの理学として解決済みであるからその枠組みで探求を進めればよいという立場である。しかし、近年になって機械工学が解決できていない問題の存在が強く認識されるようになってきた。これらは大きく二つに分けられると考えられる。

一つは、技術的な問題で、磨耗、疲労、機構学あるいは切削などが挙げられる。これらは、確かに易しい問題ではないが、工業的には何とか克服されていることの多い問題でもある。すなわち、たとえば摩耗あるいは疲労の問題に全く打つ手が施されていない機械製品は存在しないのであって、そのことは取りも直さず問題の解決が全く不可能というわけではないことを示している。それは、「理学的に」解決できないということであり、「解決」ということの定義、および解決手法自身の問題であると考えることができる。

もう一つは、人間も対象に含めた技術問題、主に技能と呼ばれる問題の解明であり、従来は理学的にも工学的にも探求の対象とされることが少なかった。しかし、少なくとも機械産業は、いわゆる技能によって支えられており、技能によって機械の性能に歴然とした差異が生ずる。

以上のような問題群の解決には、理学的手法の直接応用だけでなく、考え方や姿勢の変更が必要であると考えられ、現象あるいは対象物を視る観点が理学からの継承であってもよいのかを検討する必要がある。工学に適した対象視点を確立する必要があるという意味で「工学的オントロジー」の必要性を強調しておきたい。

幾何形状個差は、筆者らが提唱している、機械部品表面性状の設計者からの指示法の一つであるが[伊藤ほか92]、工学的オントロジーの一例となっていると考えられる。すなわち、幾何形状個差は、部品表面相互が接触する状態を制御するために、部品表面のある特定領域について、それが全体として微小に中凸あるいは中凹などのような傾向を持つべきことを指示するものである。このような傾向を与えることの有効性は古くから製造現場では知られていたことであったが、定量的厳密性を重視する（科学的）姿勢とは相容れない考え方である。従来の幾何学的かつ直接的に形状を指示するISO幾何公差方式では、理想的な平面あるいは円筒面などからの偏差のみを問い、理想形状に近づけることを要請していた。精度が良いとは、ユークリッド幾何学的理想形状に近いことを言っていた。しかし、光学的な鏡面を作る場合を除くと、ユークリッド的理想形状の機械部品は、もしそれができたとしても性能が良くない。

たとえば、理想的平面の金属表面同士は通常固着してしまうのである。幾何形状個

差は、製造時に部品表面に微少な傾向を与えることが可能であるというより、むしろ厳密に形状指示することに比べて格段に容易かつ低コストで実現可能であることも工学的な特徴である。機械技術関連の現場的知識全般と同様の事情であるが、幾何形状個差をどのように指示すればどのように機械の性能が変わるのかが定量的に判明してはいない。しかし、幾何形状個差が外在化されていなければ、それを指示して性能向上を図ることなども不可能であるし、それを語彙とした設計知識の記述も不可能である。幾何形状個差と同様な非定量的で曖昧な観点あるいは概念は、工学の多くの分野に存在することが予想され、それらを外在化して工学的オントロジーを拡充してゆく必要があると考えられる。

(2) CONMOTOオントロジー

機械は、部品同士の接触によって機能を担う。したがって、機械製品の機能は、その「形」によって発現する。このことは、機械工学関連ではよく知られている。そのため、部品などの設計対象を計算機表現する（前述のISO 10303 STEPではプロダクトモデリングと呼ばれる）に当たって、三次元ユークリッド幾何学に基づくSolid Modelとして実現することが試みられてきている。言い換えると、プロダクトモデルのためのオントロジーとしてユークリッド幾何学を採用している。しかし、機械部品においてその「形」が本質的であるからといって、オントロジーとしてユークリッド幾何学を採用することを直接意味する訳ではない。ユークリッド幾何学の枠組みでは、頂点－稜線－面－立体という階層構造が常に無矛盾に管理される必要がある。

機械技術は、近年肉眼では見えないほど微少なサイズにまで対象を広げようとしているが、古くから考えられてきた機械のサイズにおいてさえも、そこには頂点や稜線は存在しない。金属表面には、微少な凹凸が存在して、ユークリッド幾何学の定義を満たす幾何実体は一切存在しない。設計者のメンタルモデルにおいても、機械が頂点および稜線で構成されているとは考えていないと予想される。しかし、近似的には、あるいは機械設計用語を用いれば称呼値（呼び値）としては、「表面」は考えており、頂点や稜線は表面の交わりとして抽象的に捉えていると考えられる。機械加工で作れるのは切削面であり、切削面の交わりとして結果的に似たものができる以外に、頂点や稜線のみを直接加工することはできない。

CONMOTOシステムは、筆者らが設計対象物のメンタルモデルを表現するものとして開発した機械設計用の対象モデリングシステムである[伊藤ほか 91]。CONMOTOシステムには、矛盾はしないがユークリッド幾何とは異なる、設計に適したと筆者が考える形状表現法が採用されており、「追い寸」あるいは基準の概念によって部品表面を空間配置することによって、「三次元形状」を表現している。ここでは前述の幾何形状個差を含めてCONMOTOシステムの対象表現を、CONMOTOオントロジーと呼ん

でおく。

CONMOTOシステムでは、部品の構成要素を「部品素」と呼ぶ部品表面の一部とし、部品素同士の相対位置を構成要素間の依存関係とするオントロジーを採用している。そして、機械部品については全体の基準が唯一であるため、依存関係が木構造をなすことが証明されている。

形式的になっていないが、まとめれば下記のようになる。ただし、Solid Modelについては、たとえば「面」をプリミティブとして、面の方程式係数を保持する方法などがあり、ここに記したことが唯一の方法ではないが、実体の左右に記した定義の論理関係は常に成立する。

Solid Model（多面体に限定）オントロジー

(x,y,z座標)	頂点	(稜線・稜線)
(頂点 x 頂点)	稜線	(面・面)
(稜線 x 稜線)	面	(立体・立体)
(面 x 面)	立体	——

CONMOTOオントロジー

Entity　：部品素（平面分、円筒面分、円錐面分）+ 幾何形状個差

Relation：基準となる部品素に対する相対位置（直交座標系、円筒座標系）

次に、属性モデリングは、CONMOTOの例などから敷延させて提唱している設計対象モデリングの方法論であり、機械以外の分野についても適合性を調査したが、現在のところ反例は見つかっていないものである。ここから導かれた仮説は以下のようなものである。—すべての分野で、設計対象物は、分野固有の構成要素と、それらの間の存在の依存関係によって表現でき、さらにこの依存関係は設計の最終段階では木構造をなす。—

この命題の証明はできていないが、加工あるいは施工が基準を設けて行なわなければならない過程であることによっていると考えられる。したがって、設計対象物のオントロジーは、IDEF5の形式によって記述可能であり、そこで用いられる関係は、（存在の）依存関係のみで良いということになる。

CONMOTOモデルおよび属性モデルは、IDEF5によって形式的に記述でき、図式に基づきインプリメントも可能であろうと予想される。したがって、工学的オントロジーを適切に採用すれば、アドホックに行われ勝ちであったCADシステムの開発が組織化されその信頼性の向上が期待できる。ただ、内容の領域意味論の問題が重要である。共通辞書方式などを利用すれば、専門用語の一部は定義されるが、それでもSymbol Groundingの問題[Hamad 90]は、依然として残る。この問題は、古くから努力の払われ

てきたCAD/CAM統合化において最大の障害となっていたと筆者は考えている。オントロジー工学の進展によって、この最後の問題がさらに明確になることは明らかである。

参考文献

[伊藤ほか92] 伊藤公俊、弘末太郎、塚田忠夫：設計知識の計算機処理を目的とする幾何形状個差の提案－接触を考慮した表面性状指示の方式－；日本設計工学会誌、27、6、249～255(1992).

[伊藤ほか91] 伊藤公俊（分担）：インテリジェントＣＡＤ(下)－テクノロジーと展望－(吉川、冨山・共編著)；朝倉書店(1991)、または、伊藤公俊：設計対象物のメンタルモデル；人工知能学会誌、7、2、203～211(1992).

[Hamad 90] Harnad,S., The Symbol Grounding Problem, PHYSICA D 42,1990.

3.7.5　PHYSSYSオントロジー

物理的な人工物は物理法則を示す数式があれば記述可能であるわけではない。というのは、物理法則は数式そのものではないからである。例えば、F=maという式自体が「力」と「質量」「加速度」の関係を示すわけではない。「力」や「質量」といった概念を使ってこの式を解釈するとき、はじめてその関係が現れるのである。人間はこのような解釈を自明のものとして扱っているが、計算機に可能にさせるには、これらの概念を明示的に用意する必要がある。Borstらはこのような工学的な問題解決に必要な工学的オントロジーEngineering ontologyについて研究している[Borst96][Borst97]。物理的な人工物を記述する際には(1)システム構成、(2)振る舞いを実現する物理プロセス、(3)数学的記述、の３つの視点がある。ここでは、これらそれぞれをオントロジーとする３つのオントロジーで物理システムを記述する。３つのオントロジー間の相互依存関係はprojectionとして定義される。さらにそれらはより基本的なオントロジーを利用して定義される（図３.７.５-１参照）。なお、PhySysでは、全ての定義はontolinguaを用いて定義されている。

物理的人工物をみるひとつの視点は、それをシステムとしてみることである。すなわち、「境界」により「環境」から切り離され、「内部構造」をもつものであるとみることである。その基本オントロジーとしてmereologyとtopologyがある。

Mereological ontologyとは全体－部分の関係のオントロジーである。このオントロジーにおいて包含関係の公理が定義される。

Topological ontologyでは接続という関係が定義される。物理システムにおける接続とはエネルギーを交換しうるものである。これらに基づいて、システム理論のオントロ

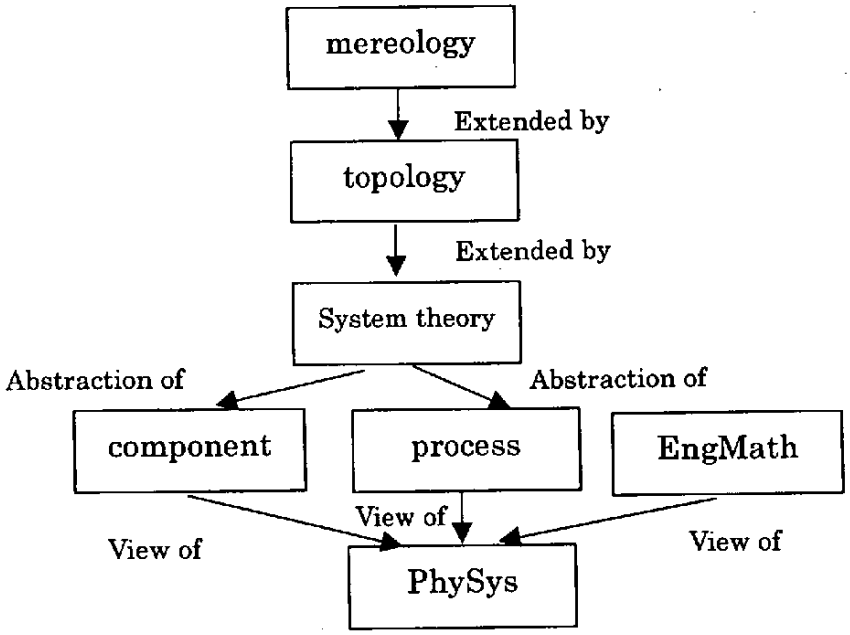


図３．７．５-１　PhySyS オントロジーの包含関係

ジーが定義される。ここでは、システム、副システム、システム境界、環境などが定義される。Componentとはこれらを用いて物理システムを構造的なものとして表現する視点である。ここでは、コンポーネント、副コンポーネント、ターミナルなどが定義される。

プロセス・オントロジーとは物理システムの振る舞いをみる視点である。ここではシステムダイナミックス理論を用いて、振る舞いを記述する。この理論においては、システムのダイナミックスは、なにかstuffの変化としてみる。そしてstuffの変化はflowとして捉えられる。そして、flowに必要なものをeffortと呼ぶ。例えば、電気分野において、stuffは電荷であり、flowは電流であり、effortは電圧である。同様に、機械では場所、速度、力となる。ここで、flowとeffortの積はエネルギー／時間という単位になり、これをenergy flowと呼ぶ。プロセスオントロジーにおいては、物理法則をenergy flowに当てはめたものとして、物理的な機構を記述する。この利点は様々な分野間の類推が適用可能になることである。

数学オントロジーはOntolingua libraryで定義されたものを用いている。ここでは物理量、物理単位とそれらを用いた数学的関係が定義される。

オントロジー・プロジェクションとは、オントロジー同士の関係付けである。基本的な関係とは、「コンポーネントは物理量と数学的関係で記述される物理プロセスの担体である」と解釈されている。

これらのオントロジーを概念レベルの記述して、実際に物理システムのシミュレーションを可能とするライブラリOLMECOを開発している。ただし、ここでは数学オントロジーは数式として表現される。

参考文献

[Borst96] Pim Borst, Hans Akkermans, and Jan Top. Engineering ontologies (short version), In proccedings of KAW96, 1996

[Borst97] Pim Borst, and Hans Akkermans. Engineering ontologies, International Journal of Human-Computer Studies, 365-406, 1997

3.8　法律オントロジー

法律分野においては、法令、判例、法理論、学説、暗黙知など多数の法律知識が存在するので、法律概念体系を記述した法律オントロジーがあれば、一貫性を保持しながら法律知識ベース全体の開発が可能になり、また、保守の面でも大きく貢献するであろうと期待されている。以上の背景から、1997年７月にメルボルンにおいて、第６回人工知能と法律に関する国際会議に併設する形で、法律オントロジーに関する国際ワークショップ（LEGONT '97：First International Workshop on Legal Ontologies）が初めて開催され、法学者と計算機科学者が一同に集まり、法律オントロジーに関して種々の議論がなされた。

本節では、LEGONT '97において話題を集めた、法体系の機能分析に基づく法律オントロジー構築方法論[J.Breuker 97]とフレームに基づく法律オントロジー構築方法論[R.Kralingen 97]について説明する。

3.8.1　法体系の機能分析に基づく法律オントロジー構築方法論

Valente [A.Valente 95][J.Breuker 97]らは、法体系（制度）の機能を分析し、分析された機能に基づいて法律知識を分化する事により、法律オントロジーの構築を進めている。すなわち、法体系は、社会の様々な振る舞いに反応しながら社会を変革していく機能を有していると考え、その機能を６つのサブ機能に分け、その分化に応じて、法律知識も６種類のカテゴリーに分かれるとしている。以下、６種類の法律知識について概説した後、それらの知識を利用したシステム構成の概要について説明する。

(1)　法律知識の分類

① 規範知識（Normative Knowledge）

規範知識は、人や社会の振舞いの拠り所となる基準を定義した法律知識であり、「～すべきである」といった形で個々の法律規範において叙述される。

② 世界知識（World Knowledge）

現実世界において起こりうるあらゆる行為を記述するための知識が世界知識である。規範知識は、理想的な振舞いを記述することから、世界知識は、そのような規範に含まれない振舞いを記述する知識ともいえる。

また、世界知識は、規範知識や他の法律知識に基づいて、現実世界のモデル（概念定義や概念間関係）を構築していると捉えることができ、人々のもつ常識知識と規範

知識間の仲介の役割を果たす法律抽象モデルとしてみなすこともできる。

なお、世界知識は、現実世界の出来事を法的に解釈するための静的な知識である定義知識（Definitional Knowledge）と定義知識において因果関係を推論するための因果知識（Causal Knowledge）に分かれる。

③ 責任知識（Responsibility Knowledge）

責任知識は、行為者の行為の結果により負わされる義務の割当てについて言及した法律知識である。また、その義務を拡大か制限する内容になることもある。義務の割当ては、規範違反とその違反の責任を取るべき人との間にリンクを張ることにより処理される。

④ 反応知識（Reactive Knowledge）

反応知識は、行為者が規範に違反した時に、社会から受けるべき行為（反応）を定義した法律知識である。通常、反応は法的制裁となるが、報酬の場合もある。

⑤ メタ法律知識（Meta-Legal Knowledge）

メタ知識とは、知識を何らかの形で処理するための知識であることから、メタ法律知識は、法律知識を何らかの形で処理するための法律知識といえる。通常、メタ法律知識は、1) 法体系の発生・改訂・消滅に関する規定（法改正、規範からの法制定の方法の規定など）と、2) どちらの法律知識を適用すべきかといった競合解消、といった形で利用される。

⑥ 創造知識（Creative Knowledge）

創造知識は、以前存在していなかった法律概念（実体）を新たに創世するための法律知識である。

(2) システム構成

(1)で分類した6種類の法律知識は、４１種類のクラス、１７種類の関係、１０種類の関数、6種類のインスタンスを用いて、Ontolingua [T.R.Gruber 92]により記述されている。４１種類のクラスは、図３.８.１-１のようなクラス階層をもち、6つのカテゴリーは、クラス階層の２番目の上位クラスに割り当てられている。また、これらの定義とOntolingua自身が持つ定義（フレームオントロジー）を利用して、法律知識ベース中の各概念が記述される。

このように記述された6種類の法律知識を利用して、法的推論がどのように実行されるかを図３.８.１-２に示す。まず、法律システムは、世界知識（定義知識）を利用

して、現実世界の出来事を解釈して、その出来事に対応する法的に資格付けられた振る舞い（legally qualified behavior）を生成する。次に、規範知識を利用して、その振る舞いを規範に違反しているかどうかを検証し、許可される振る舞いか違反している振る舞いか（classified behavior）に分類する。また、その処理と並行して、世界知識（因果知識）を利用して、legally qualified behaviorに関連するすべての行為者を決定した後に、責任知識を利用して、責任を負わされるべき行為者を決定する。このようにして

- 法律知識
  - 規範知識
    - 第一規範
      - 規範関数
    - 規範ステータス
  - 世界知識
    - 法的概念
    - 法的関係
      - 静的関係
      - 振舞関係
    - 事例
    - 状況
    - 汎用事例
    - 状態
  - 責任知識
    - 責任制限
    - 責任譲渡
  - リアクティブ知識
    - リアクション詳細
    - リアクション
  - 創造知識
    - 創造
  - メタ法律知識
    - 規範データ
      - 規範メタクラス
        - 範囲制限
      - 規範メタ関係
    - 規範制限
    - 適用順規規範
    - 法的権威規範
    - 法的権威滅損規範

図３．８．１-１　法体系の機能のクラス階層

得られたclassified behaviorと責任を負わされるべき行為者を反応知識に与え、その行為者に対する制裁を決定する。以上の処理が基本的な法的推論の流れであるが、必要ならば、創造知識により新しい抽象的な法律概念が生成されて、世界知識（定義知識）に付加され、法律の実行制御などが必要であれば、メタ法律知識が適用されることになる。

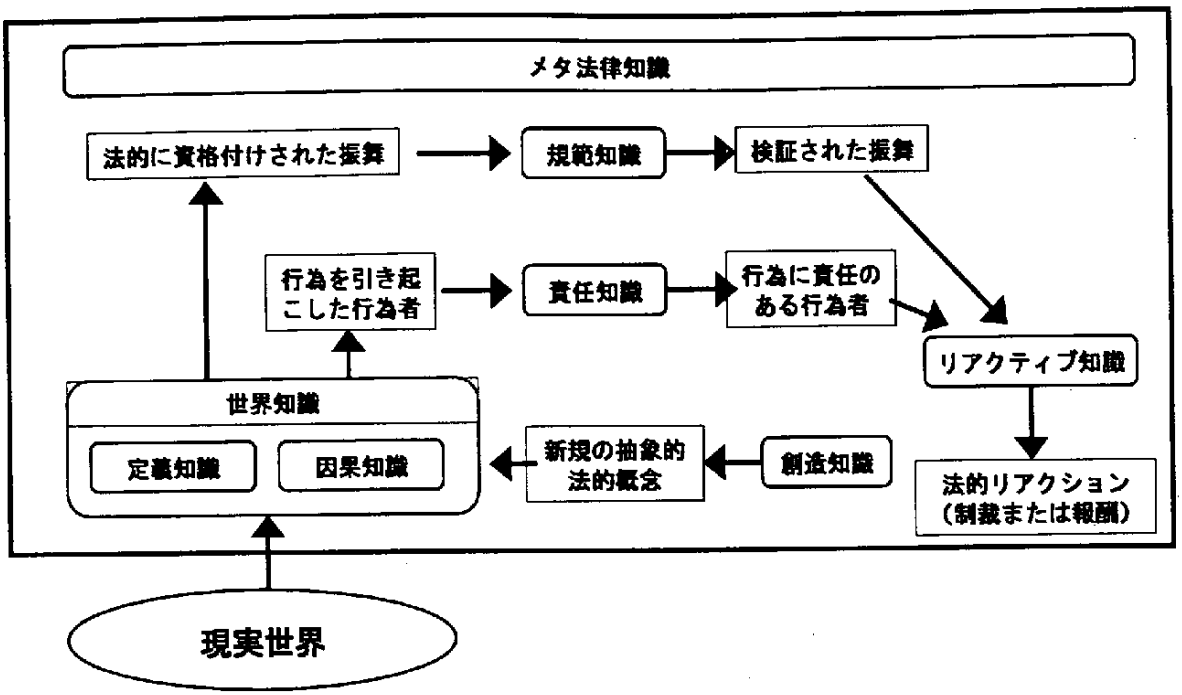


図３．８．１-２　法体系の機能のクラス階層を利用した法的推論

３．８．２　フレームに基づく法律オントロジー構築方法論

３．８．１では、法体系の機能分析に基づく法律オントロジー構築方法論について述べたが、分類された法律知識が持つべき構造など、法律知識の詳細な定義については触れていなかった。一方、Kralingen [R.Kranlingen 95]とVisser [P.R.S.Visser 95]は、様々な法律世界をモデル化するための概念プリミティブ群を詳細に整理し、法律オントロジーの詳細仕様まで開発する方法論を検討している。彼らは、独立に研究を進めていたが、事前に準備したフレーム群を具象化する形である特定の法律オントロジーを構築する方法論をともに採用しており、ここでは、Kralingenの法律オントロジー構築方法論を紹介することを通して、フレームに基づく法律オントロジーについて概説する。

Kralingenは法律家であり、法律を深く分析することにより、３種類のフレームを提案している。ある特定の法律オントロジーを構築する場合は、その法律（制定法）に

含まれる語彙を整理し、これら３種類のフレームのスロット群をそれらの語彙により埋めることにより、法律オントロジーを構築していく。

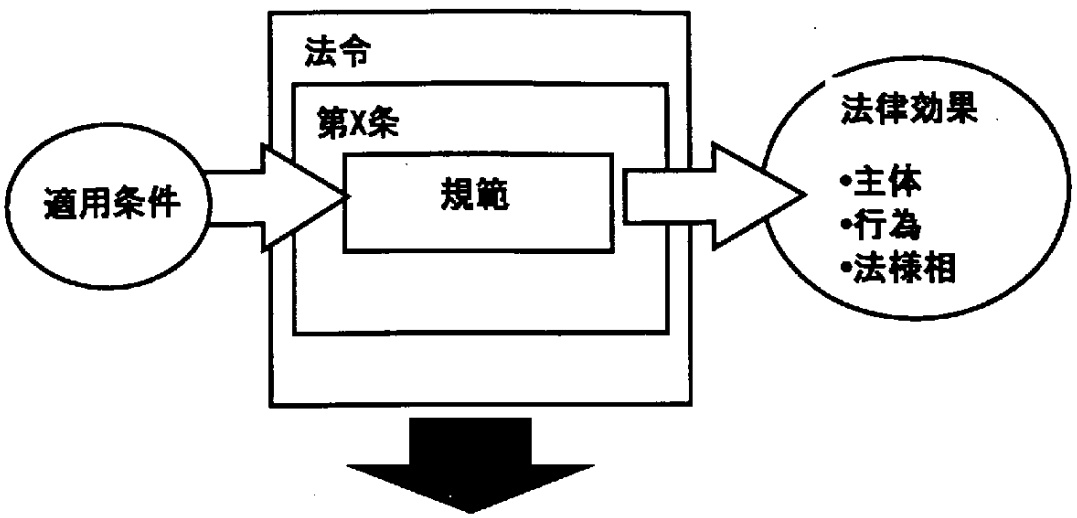
以下においては、フレーム、語彙、開発手順について説明する。

(1) フレーム

Kralingenは、法律を深く分析した結果、規範・行為・概念という３種類のフレームを提案した。これらは、ある特定の法律に依存するものではなく、様々な法律オントロジーの基本的なスキーマ構造として利用できる。

①規範フレーム

規範フレームは、図３.８.２-１に示されるように、５種類の主要スロットと３種類の補助スロットから構成される。規範フレームでは、現実世界で発生した状況が入力となり、その入力が規範の適用条件（第４スロット）を満たすかどうかを決定し、適



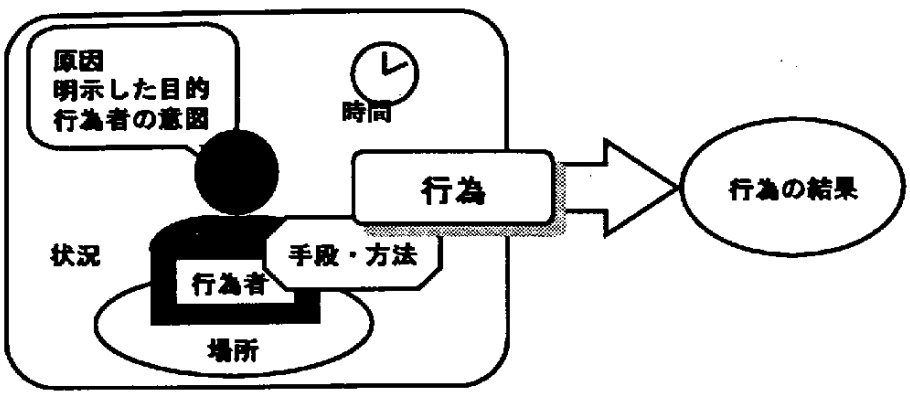
	スロット名	内容
1	規範識別子	規範を参照するときのポインタ
2	規範タイプ	行為規定、または、権利能力
3	公布法令	規範が含まれている法令
4	成定法	その規範が含まれている成定法
5	適用条件	規範が適用される状況
6	主体	規範により導かれる法律効果の主体
7	法様相	法律効果の法様相 （すべき、すべきではない、してもよい、できるの４種類）
8	行為識別子	法律効果の行為を参照するポインタ

図３.８.２-１　規範フレーム

用可能ならば、どのような法律効果が発生するか（第６スロット：法律効果の主体、第７スロット：法様相、第８スロット：行為）を決定する。以上のスロットに加えて、規範タイプが行為規範（義務として発生する行為など）と権利規範のどちらであるかを示す第２スロットが、規範の内容に関わる主要スロットである。さらに、規範識別子（第１スロット）、規範に対応する法令（第３スロット）、規範が含まれる成定法（第４スロット）が補助スロットとして利用されている。

②行為フレーム

行為は、現実世界の状態を変える様々な動的な側面をもつが、図３.８.２-２に示すように、１１種類の主要スロットと３種類の補助スロットから構成された行為フレームにより、現実世界の様々な行為を表現する。主要スロットとしては、行為者（第４スロット）、行為のタイプ（第５スロット）、行為の様相として、その行為に使った手段（第６スロット）と方法（第７スロット）、行為のコンテキスト情報として、行為を行った時間（第８スロット）、場所（第９スロット）、その他の環境（第１０スロット）、行為の合理性を言及する情報として、行為の原因（第１１スロット）、目的（第１２スロット）、意図（第１３スロット）、最後に行為を行った結果（第１４



	スロット名	内容
1	行為識別子	行為を参照するときのポインタ
2	公布法令	行為が含まれる法令
3	成定法	行為が含まれる成定法
4	行為者	行為者
5	行為タイプ	法令固有、または汎用。記述される行為で識別
6	手段	行為を行う為に利用した手段
7	方法	行為を行った方法
8	時間	行為を行った絶対時間
9	場所	行為を行った場所
10	状況	行為を行ったその他の環境
11	原因	行為を実行する原因
12	目的	行為者によって明示された行為の目的
13	意図	行為者の意図
14	行為の結果	行為が引き起こす結果

図３.８.２-２　行為フレーム

スロット）の１１スロットがある。一方、補助スロットとしては、行為識別子（第１スロット）、その行為が含まれる法令（第２スロット）、その行為が含まれる成定法（第３スロット）の３スロットがある。

③概念フレーム

概念は、規範や行為以外の諸々の概念をさすが、図３.８.２-３に示すように、７種類のスロットから構成された概念フレームにより、法令中で定義されている概念のみではなく、法令中の概念から（容易に）導きだされる概念、法律要件になる可能性があると考えられる要因、法令の実行により導かれるメタ概念なども表現される。要因概念に関しては、要件になりうる度合い（第３スロット）を与えている。また、概念の例示（第７スロット）も与えている。

	スロット名	内容
1	概念	記述される概念
2	概念タイプ	定義された概念、定義された概念より導かれる概念、要因概念、メタ概念の４種類をとる
3	度合	法律要件になりうる度合い
4	公布法令	概念を含む法令
5	成定法	概念を含む成定法
6	適用条件	概念の適用条件
7	例示	概念のインスタンスの列挙

図３.８.２-３　概念フレーム

(2) 語　彙

与えられた問題領域に含まれる語句を利用して、上述の３種類のモデルを具象化することにより、法律モデルを構築する。フレーム構造を考慮して、1)行為、2)行為者、3)行為に関連する物、4)人と物の間に成立している関係、5)人や物がもつ特性、6)時間、7)場所、8)関連する成定法、9)テキスト構造、10)算術、11)法様相、という１１種類のカテゴリーに関連する語句が、法令固有語句か汎用語句かを識別されて抽出されるが、現在、この作業はユーザに委ねられている。語句が抽出された後、法様相のように、スロット名とカテゴリーが一致する場合は、カテゴリーに含

まれる語彙によりスロット値が定義される。しかしながら、規範フレームの適用条件のように、カテゴリーとスロット名が一致しない場合は、抽出された語句が組み合わされて、スロット値が定義される。

(3) 法律オントロジー開発手順

以下に示すガイドラインと経験則を利用して、法令から語句を抽出して、各フレームのスロット値を定義することにより、法律オントロジーを開発する。

「法律オントロジー開発のためのガイドライン」

1. 可能な限り、法律知識源の元々の意味を保持する。
2. 可能な限り、規範、行為、概念を単一のフレームで表現する。
3. 具象フレーム数を少なくするために、可能な限り、規範、行為、概念を結合する。
4. モデルの保守にコストを要する複雑なフレームは、複数のフレームに分割する。

「法律オントロジー開発のための経験則」

1. 規範、行為、概念の核となる条文から構築を開始する
2. 可能な限り、一つのフレームで多くの条文を記述する。
3. 規範や概念が異なるタイプに対しては、別のフレームを構築する。
4. 法令固有の特徴的な用語は、独立したフレームで表現する。
5. 局所的なテキスト構造を削除する
6. 大域的なテキスト構造を保存する
7. 同じ情報を含む異なった条文を一つのフレームに統合する。

フレームのスロット値をすべて定義する必要はなく、規範フレームでは、規範主体・法様相・行為識別子スロット、行為フレームでは、行為者・行為タイプスロット、概念フレームでは、概念・概念タイプ・適用条件・インスタンススロットのいずれかを定義すればよい。

Kralingenは、以上のような検討をした後、実際に、Prolog言語を用いてDUBA（オランダの失業者給付法）オントロジーを実際に構築し、フレームに基づく法律オントロジー構築方法論の適用可能性を示している。DUBAオントロジーの一部を図3.8.2-4に示す。また、この他に、オランダの刑法や民事訴訟法、大英帝国大学図書館規定などを題材として、法律オントロジーの構築を試みている。

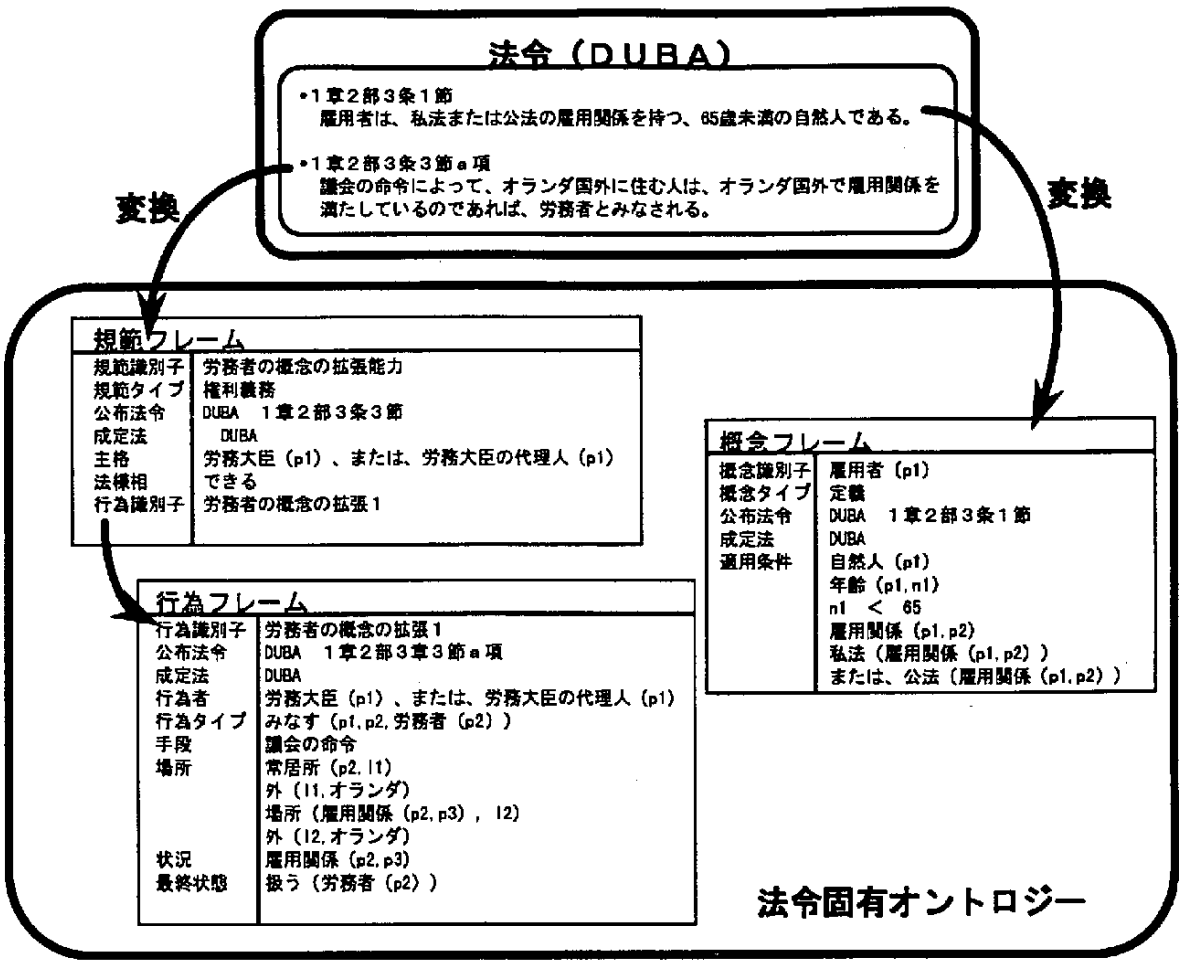


図3.8.2-4　DUBAオントロジー

参考文献

[Breuker 97] J.Breuker, A.Valente and R.Winkels : Legal Ontologies: A Functional View, LEGONT'97, pp.23-36 (1997)

[T.R.Gruber 92] T.R.Gruber : ONTOLINGUA : A Mechanism to Support Portable Ontologies, TR, KSL, Stanford Univ. (1992)

[R.Kralingen 95] Kralingen,R.W : Frame-based Conceptual Models of Statute Law, Kluwer Law International (1995)

[R.Kralingen 97] R.Kralingen : A Conceptual Frame-Based Ontology for the Law, LEGONT'97, pp.15-22 (1997)

[Legont 97] Proceedings of the First International Workshop on Legal Ontology, (1997)

[A.Valante 95] Valante, A : Legal Knowledge Engineering A Modeling Approach, IOS press (1995)

[P.R.S.Visser 95] P.R.S.Visser : Knowledge Specification for Multiple Legal Tasks; A Case

Study of the Interaction Problem in the Legal Domain, Computer/Law Series, No.17, Kluwer Law International (1995)

3.9　インターネット上の情報検索とオントロジー

3.9.1　はじめに

インターネットの普及とともに非定型的かつ地球規模で分散化された情報を検索するための研究開発が人工知能の分野において、最近、非常に活発になってきている。たとえば名古屋で開催されたIJCAI97 [IJCAI 97]においても数多くの発表がなされている。これには、インターネット上のYAHOO [Yahoo 98]、AltaVista [Altavista 98]、MataCrawler [Metacrawler 98]、Ahoy [Ahoy 98]などのさまざまな検索エンジンの開発・普及、ネットワークエージェントやソフトウェアエージェントの技術 [JIPS 97]、必要な情報を探索するデータマイニングの技術 [JSAI 97]などが含まれる。本節では、これらに共通する基盤技術・概念としてのオントロジーについて、事例調査を中心に報告する。

インターネット上の情報検索にオントロジーの概念を適用する方法は基本的には次のとおりである。インターネット上の記述言語であるHTMLをいろいろな観点から拡張し、高度な検索が容易にできるようにする。検索対象となる情報のタグづけにオントロジー情報を利用する。このような方法においては、共通のタグ言語の開発と普及が重要であり、あまり複雑なオントロジーは望まれない。したがって情報検索に使われるオントロジー情報は、現在のところ比較的単純な構造をしており、2.2節でいうレベル1オントロジーに相当すると考えられる。

以下では、代表的な研究開発例として　Meta-Content Format（MCF）、eXtensible Markup Language（XML）、Information Manifold、SHOE/Knowledge Annotator について、この順に、オントロジーの適用という視点から詳しく述べ、その他のシステムについても簡単に触れる。

3.9.2　Meta-Content Format（MCF）と eXtensible Markup Language（XML）

(1)　XMLの概要

Meta-Content Format（MCF）は、従来のタグ言語では実行が難しかった複雑な方法でウェブサイトの情報を表現するための手段を提供する、たとえば、MCFを利用すると3次元のオブジェクトのブラウズが簡単にできるようになる。拡張可能なマークアップ言語（XML）は、MCFを実現するための標準言語として、W3C（World Wide Web Consortium）で規格化が検討されているものである[MCF 98]。

XMLの検討が開始された背景としては以下があげられる。

・HTMLでは、小規模なタグセットのみが定義されているだけで、複雑な構造のドキュメントの交換ができない。
・HTMLではデータをパラメータ化したりデータの意味を明示するために利用者がタグを作ったり、属性を指定したりする機能がない。
・HTMLでは、データベーススキーマやオブジェクトなどの複雑な構造を含む情報が表現できない。
・一方、マークアップ言語の元になったSGMLドキュメントの規格は複雑であり、HTMLドキュメントより作成が難しい。
・SGMLで提供可能な機能だけではウェブ情報に対応しきれない。

これらに対応するためにXMLではSGMLの複雑な機能を一部取り除くとともに、ドキュメント構造を正確に記述でき、インターネット上でのハイパーリンク機能を追加した形で設計が行われた。XMLは次の点でHTMLを異なっている。

1. 利用者は、新しいタグや属性名を自由に定義することができる。
2. ドキュメント構造の複雑な入れ子を作成することができる。
3. XMLドキュメント中に文法の記述をオプションとして埋め込むことができ、この文法を用いてアプリケーションに構造の正当性検証をさせることができる。

XML設計の基本方針は以下のとおりである。

1. XMLはインタネット上で直接的に使用可能である。
2. XMLはさまざまなアプリケーションでの利用を支援する。
3. XMLはSGMLと互換性を持たなくてはならない。
4. XMLドキュメントを処理するシステムは簡単に作成できなければならない。
5. XMLのオプション機構はできるだけ少なくする。
6. XML文書は人間にとっても理解しやすいものでなければならない。
7. XMLの設計作業は迅速に行う必要がある。
8. XMLの設計は形式的かつ簡潔でなければならない。
9. XMLドキュメントは簡単に作成できなければならない。
10. XMLのマークアップ機能は複雑になってもかまわない。

(2) XMLのWebアプリケーション [Bosak 97]

XMLの利用が適切なアプリケーションは、以下の４つのタイプがある：

1. 二つ以上の異種データベースの間を仲介するWebクライアントを必要とするシステム：

この例としては、家庭の健康管理に関する情報提供システムがある。このシステムでは、患者情報のデータベース、薬品に関するデータベース、病気や治療法に関するデータベースなど非定型的で複雑なマークアップを必要とする情報資源を統合して検索しなければならない。

2. 処理の負荷をWebサーバからWebクライアントに分散させたいシステム：

この例としては、半導体産業におけるIC技術データ提供システムがある。このようなシステムでは、マークアップされた情報資源ごとに特定のアプリケーション（たとえばJAVAアプレット）が起動するようなしくみが重要となる。そこで、業界専用のSGMLマークアップ言語を作り、豊富なパラメータを持ったデータ・ストリームとして半導体データを表現すれば、判読可能な文書としてデータを単に表示するだけではなく、設計プロセスを高度化するインテリジェントなアプリケーションが可能になる。

3. 同一のデータを、利用者ごとに別の視点で、操作するシステム：

この例としては、動的に生成される目次情報提供システムがある。Webサーバ上で動的な目次の処理を行うかわりに、構造化された全目次をクライアント上へダウンロードすることで、実行時に文書の階層構造から直接生成させることができる。

4. 情報検索をカスタマイズするWebエージェントシステム：

電子商取引や検索エンジンの利用時に、利用者のプロファイル情報を利用して知的な処理を行うエージェントシステムの考え方は広く知られている。このようなシステムでは、すべてのプロバイダの情報を対象とし、このユーザの好みやその他の特性（教育水準、興味、職業、年齢、視覚の鋭敏さ）が、ベンダーに依存しない標準的な形式で指定されていることが必要となる。XMLはこの種のシステムの開発手順を容易化、標準化するものとなる可能性が高い。

(3) XMLのリンク付けとスタイルシート

1. リンク付け

HTMLのリンクは、ハイパーテキスト・システムの概念と結び付けられてきた機能のうち、すなわち文書中に直接書き込まれたロケーションへの単一方向のリンクに対応しているのみの単純なものである。それに対してXMLにおけるリンクでは以下がサポートされている：

・ロケーションに依存しない名前付け
・双方向のリンク
・ドキュメントの外部に指定され管理されるリンク
・N組のハイパーリンク（例えば、リング、複数ウィンドウ）
・集合リンク（複数のソース）
・リンク先の文書が、リンク元の文書の一部分であるように見える機能（Tranclusion）
・リンク属性の定義

2. スタイルシート

XMLでは、次のようなスタイルシート・プログラム言語が提案されている：

・利用者が自由にタグを拡張することができる。
・スタイルシートは通常のプログラムと同等の能力をもち、任意の手続きを記述できる。
・習得を容易にするために標準シンタックスをベースにしている。
・XMLドキュメントの木構造のどの部分でも、構造的な用語で参照することができ、文書の要素間の前後関係について複雑な指定も表現することができる。
・国際化されており、左から右、右から左、上から下へ書かれたスクリプトを取り扱うことができ、１つの文書の中にこれらが混在していても処理が可能である。
・複数カラム、テキストの回転、フロート領域といった、複雑なページ・レイアウトの指定が可能な高度なレンダリング・モデルを備えている。

Web上で文書が効果的に配布できるようにするため、部分的なレンダリングが可能なように定義されている。

3.9.3 The Information Manifold

Information Manifold System [IM 98][Levy 96]はWWW上のデータベースやフォームなどの資源に代表される複数の構造的な情報資源に対して一様なアクセス方法を提供するシステムである。そのため複数資源の情報を組み合わせなければ解答が選られないような複雑な質問処理を行うことが可能となる。Information Manifoldによって、与えられた質問に該当する情報源の発見、情報資源の内容に対する直截的な記述によるアクセス、複数資源情報を人手で組み合わせるなどの作業はすべて不要になる。

The Information Manifold Movie Query Engine

This site uses information from The Internet Movie Database, MovieLink, Movie Review Query Engine at Telerama, Palo Alto Weekly, and VideoFlicks.

Specify any attributes of the movies you are looking for. In the bottom of the form you can specify additional information to receive about the movies.
Click on an attribute name for more details about the attribute.

Movie Title: (Substring)
Movie Year: between and
Movie Genre: All Genres
Movie Director: Last name: First name:
Movie Actor: Last name: First name:
AND
Last name: First name:

Type in your zip code or city, if you are looking for movies showing in your town.
Zip Code: (5 digit) (eg. 07974)
OR
Location: (City, State) (eg. Murray Hill, NJ)

Check for additional information about selected movies.
- Movie Review
- Purchase Information (for Movies on Video)
- Related Movies

Submit

[Home | Examples | Demo | Related Projects]

Back to Top

図3.9.3-1　IMを使用する例題

すなわち、Information Manifold（IM）は一様でないネットワーク上で分散した情報源を対象とする情報のナビゲーション、検索、および組織化の手段を提供するツールである。IMは、インターネットのような膨大かつ構造化が不十分な情報空間を効率的に管理するために、演繹的な知識表現システムを採用している。このアプローチは、利用者の生業下にあるクライアント側の情報世界を知的なものにしようという点に特長がある。IMは通常のウェブブラウザと同様のクライアントシステムであり、インターネット上で分散し、単純なハイパーテキストデータモデルで表現されるような情報資

源を操作するためのマルチメディアインタフェースを提供する。このような立場からは、ブラウジングは情報検索の基本的な機能であり、さまざまな索引機能をもつ探索サービスを伴っている。IMのインタフェースは、情報源、情報コンテンツ、それらの間の関係の詳細な記述を含む知識ベースとそのブラウザを結合することによって、ハイパーテキストのナビゲーション機能を高度化している。ブラウザと知識ベースの間の密な結合によって、IMの利用者は、ハイパーテキストのナビゲーションと、知識ベースを利用した構造化情報のブラウズと検索を同時に実行することが可能となる。

図3.9.3-1に映画情報を検索するための例題画面を示す。検索画面に指定された情報から、必要な情報資源を同定し、その資源に対して適切な部分検索コマンドを発行する。さらに複数の情報資源から得られた情報を組み合わせて最終結果とする。

IMの知識表現は、Classicという演繹的な知識表現システムで記述されている。Classicは、計算論的に高速処理が可能な、記述論理に基づくシステムであり、構造化された、オブジェクト指向の概念とインスタンス記述を可能とする。Classicは、すべての適切な概念の下で、個々オブジェクトを分類するとともに、自動的に一般化分類に概念を組織化するために包含関係と分類化に関する推論処理を実行する。

IMの知識ベースは、仮想的な情報リポジトリであり、加工した情報を格納することにより、むしろ情報資源の記述を保持する仕組みとなっている。ここで、加工した情報記述というのは、意味内容の記述、フォーマットされたテキストや検索可能な索引などのデータ型、情報検索の手がかりや方法、情報の変更やアクセスの記録、関連するドキュメントなどを意味する。この知識ベースはIMを利用する間は保持されている。

以下に、IMの知識ベース、およびその使用方法に関する重要な特徴をまとめる:

・知識ベーススキーマは拡張できる。IMインタフェースは、利用者が新しい情報を発見し、情報空間の視野が発展するにつれて、知識ベースを逐次的に改良し、拡張することを支援する。

・与えられたIM知識ベースは情報空間の視点を一意に定めるので、この知識ベースを個人やグループごとの特定の興味を表現するようにカスタマイズすることができる。

・知識ベースへのドキュメントの追加は単純であり、マウスでドキュメントを知識ベースブラウザに投入するだけで可能となる。さらに特化した知識ベースエディタが動的生成され、より複雑な入力が可能となる。

・知識ベースサーバは情報資源に関するデータのリポジトリの働きと同時に、情報資源をブラウズしたり検索したりするためのメディアの働きもする。

・ブラウジング操作で表現できないような知識の操作については、質問言語によってより複雑なドキュメント集合を記述することができる。

・しばしば利用され慣用句のようになった質問は自動的にClassic概念記述へと変換され、後の利用に備えて知識ベースへ分類・保存される。

IMにおいては、このようなClassicによる知識ベースの記述がオントロジーの役割を果たしている。

3.9.4 SHOEとKnowledgeAnnotator

(1) SHOEの概要

SHOE [SHOE 98]はHTMLを拡張したもので、ウェブページドキュメントに計算機で処理できるような知識つきの注釈をつけることができるようにしたタグ記述言語である。HTMLがドキュメントの人間による理解を目的にしているのに対し、SHOEはソフトウェアエージェントにドキュメントを理解させることを目的とする。

SHOEはウェブ上での応用を念頭に設計されており、探索や知識の収集など、大量のデータの操作が可能となるような限られたセマンティックスをもつ。これは知識ベースセマンティックスとして与えられ、大域的な制御なしで分散されたデータを操作するさまざまなメカニズムを提供できるようになっている。

SHOEの特長は次の3点である：

・豊富なセマンティックスをもつので、ウェブ設計者はドキュメントの内容に関する情報のみではなく、任意の情報をドキュメントに持たせることが可能となる。そしてソフトウェアエージェントが、学習し推論することを可能とする。

・SHOEは大量データの処理を容易にするように計算量の意味で軽く設計されているので、KIFほどは表現力は豊富でない。

・SHOEにはあらかじめ定義したオントロジー、カテゴリー、関係、推論といった仕組みは存在しない。これらはSHOEオントロジーを使って定義できるようになっている。

SHOEのタグは２つに分類される。１つ目は、オントロジーを構成するのに使われる。SHOEオントロジーはSHOEドキュメントが生成できるアサーションの種別とその意味とを定義するようなルールの集合である。たとえば、SHOEドキュメントに「犬」というエンティティを宣言したら、「犬」にはどれに付随する「名前」というデータエンティティを持たせることができる。

２つ目は、ウェブドキュメントに注釈を追加するのに使われる。これによってウェブドキュメントに１つ以上のほかのオントロジーを加えたり、データエンティティを宣言したり、特定のオントロジーで提供されたルールのもとで、そのエンティティに関するアサーションを作ったりすることができる。たとえば、SHOEオントロジーに付け加えたSHOEドキュメントにはFIDOという名前の犬に関する情報をすべて持たせることができる。このため高度な検索エンジンをブラウザ上で開発することがきわめて容易となる。

(2) SHOEのオントロジーとその形式

SHOEのオントロジーは、クラスとカテゴリーのISA階層、これらのカテゴリーの間の原子関係の集合、および単純化したホーン節で記述した推論規則を意味する。各カテゴリーは親カテゴリーで定義された関係を継承する。この仕様に従うかぎり、ユーザエージェントはHTMLページの主張を事実としてではなく、クレイムとして扱う。

SHOEオントロジーでは、データエンティティの分類、データエンティティ間の関係、ホーン節による推論、他オントロジーからの継承、およびバージョンづけの機能がサポートされる。これは、HTMLのタグをオントロジーの宣言ができるように以下の面で拡張したものになっている。

ONTOLOGY, /ONTOLOGY, USE-ONTOLOGY, DEF-CATEGORY, DEF-RELATION, /DEF-RELATION, DEF-ARG, DEF-RENAME, DEF-CONSTANT, DEF-TYPE, DEF-INFERENCE, /DEF-INFERENCE, INF-IF, /INF-IF, INF-THEN, /INF-THEN, COMPARISON, /COMPARISON, CATEGORY, RELATION, /RELATION, ARG。HTMLと同様に / ではじまる宣言はその宣言の終了を表わす。たとえば、オントロジーは、<ONTOLOGY>と</ONTOLOGY>の間で定義される。

SHOEドキュメントには次のヘッダがつく：

```
<META HTTP-EQUIV="SHOE"
        CONTENT="VERSION=1.0">
```

HTMLで定義されていない項には以下のものがある：

・Category（Class）

カテゴリーはHTMLページのインスタンス階層を分類・定義するのに使われる。多重継承の機能をもつ。

・Data

データはインスタンスかオントロジーで定義された型を表わす。基本オントロジー(base ontology)には、Strings、Numbers、Dates、Booleansの4種類の型がある。基本オントロジー以外では、データ型が任意に定義できる。しかし1つのオントロジーが他のオントロジーを参照する場合にはPrefix Chainを使う必要がある。すべてのSHOEオントロジーは、基本オントロジーを拡張・変更して作られる。

・Element

要素は、カテゴリーが、関係、または型であり、1つのオントロジー中では要素は同じnamespaceを共有する。

・Instance

ゼロ以上のカテゴリーの下で分類されて、関係に含まれることもあるデータである。

・Instance Key

インスタンスを特定するためのストリングである。SHOE-Conformantドキュメントの基本キーはそのドキュメントに対する単一の絶対URLである。たとえば、http://cs.umd.edu はこの例である。ドキュメントは複数のインスタンスを持つことが許されるが、それらには、http://www.cs.umd.edu#MyDog のように一意にキーがついていなくてはならない。

・Ontology

HTMLインスタンスの有効な分類と例と要素との有効な関係について記述する。すべてのオントロジーは直接的または間接的に基本オントロジーと継承関係にある。

・Ontology Identifier

オントロジーを一意に定義するストリングである。

・Relation （Relationship）

インスタンスと他の例やデータとの関係を定義する要素である。関係名は要素名であり、前に置かれるかもしれない。アーギュメント（arguments）と呼ばれるゼロ個

以上の要素の間には、関係が定義される。

・Rule
妥当な分類。妥当な関係を定義するための形式的なルールである。

オントロジーの宣言の例を以下に示す：

・オントロジー定義：

```
<ONTOLOGY ID="ontology-identifier"
          VERSION="version"
          [BACKWARD-COMPATIBLE-WITH="version list"]
          [DESCRIPTION="text"]
          [DECLARATORS="list-of-declaring-instances"]>
</ONTOLOGY>
```

・既存オントロジーの拡張：

```
<USE-ONTOLOGY ID="ontology-identifier"
          VERSION="version"
          PREFIX="prefix"
          [URL="URL"]>
```

・分類ルールの定義：

```
<DEF-CATEGORY NAME="category-name"
          [ISA="parent-category-list"]
          [DESCRIPTION="text"]
          [SHORT="text"]>
```

・関係ルールの定義：

```
<DEF-RELATION NAME="relation-name"
          [DESCRIPTION="text"]>
          [SHORT="text"]>
  argument-list
  </DEF-RELATION>
```

・アーギュメント：

```
<DEF-ARG POS=(positive-integer ¦ FROM ¦ TO)
          TYPE="data-type"
          [SHORT="text"]>
```

・名前代えルールの定義：

```
<DEF-RENAME FROM="element-name"
          TO="new-element-name">
```

・推論ルールの定義：

```
<DEF-INFERENCE
            [DESCRIPTION="text"]>
  <INF-IF>
      body-subclauses
  </INF-IF>
  <INF-THEN>
      head-subclauses
  </INF-THEN>
</DEF-INFERENCE>
```

・推論ボディ節の定義：

```
<CATEGORY NAME="prefixed.category"
            FOR="instance"
            [[USAGE=]("VAR" ¦ "CONST")]>

<RELATION NAME="prefixed.relationship">
  argument-list
</RELATION>
<ARG POS=(positive-integer ¦ FROM ¦ TO)
            VALUE="key"
            [[USAGE=]("VAR" ¦ "CONST")]>
<COMPARISON OP="special.declaration">
  argument-1
  argument-2
</COMPARISON>
```

・定数インスタンスの定義：

```
<DEF-CONSTANT NAME="constant-instance-name"
            [CATEGORY="prefixed-category-name"]>
```

・データ型の定義：

```
<DEF-TYPE NAME="type-name"
            [DESCRIPTION="text"]
            [SHORT="text"]>
```

オントロジーを用いたHTMLドキュメントのマークアップは次のようにして行われる。

・インスタンスの宣言：

<INSTANCE KEY="key-value"
[DELEGATE-TO="instance-list]>
</INSTANCE>

・オントロジーの使用の宣言：

<USE-ONTOLOGY ID="ontology-identifier"
VERSION="version"
PREFIX="prefix"
[URL="URL"]>

・カテゴリーの宣言：

<CATEGORY NAME="prefixed.category"
[FOR="key"]>

・関係の宣言：

<RELATION NAME="prefixed.relationship">
argument-list
</RELATION>
argument-list
<ARG POS=(positive-integer ¦ FROM ¦ TO)
VALUE="key">

(3) KnowledgeAnnotator

KnowledgeAnnotator[KA 98]はSHOEオントロジーによってウェブドキュメントに注釈をつけることを容易化するために設計されたJAVAアプレットである。Annotatorのグラフィック表示にしたがって、ウェブドキュメントを編集することによって、適切なSHOEタグを文書に付加していくことができる。

Annotatorは以下の3つの画面にわかれる：

・Entity Instances画面：
HTMLファイル中で宣言されたすべてのインスタンスを表示し、編集する機能を提供する。

・Claims表示画面：
特定のインスタンスで作成されたすべてのクレイムが表示される。インスタンスを指定すると関連するクレイムが表示される。

・Inspector画面：
特定のインスタンスに関する情報を表示する。データ、オントロジー、インスタンスの属性、関係、カテゴリーごとに異なる情報が表示される。

KnowledgeAnnotatorは以下の手順で利用される：
・注釈づけの対象となる（HTMLの誤りのない）ウェブページの準備
・新しいウェブページの初期化
・オントロジーの追加
・カテゴリーの追加
・関係の追加
・クレイム、オントロジー、エンティティの参照と変更
・属性の参照
・結果のHTMLコードの表示
・HTMLファイルの再ロード

これらの作業をグラフィカルインタフェース上で容易に実行できるのがKnowledgeAnnotatorの特長である。

3.9.5　その他のシステム

インターネット上の情報検索を対象とするシステムの研究開発例として、上に述べたものの他に以下のプロジェクトがオントロジー研究の立場からは注目される。

(1)　LiveTopics [LT 98]

AltaVistaを利用している際に、検索結果が多くなりすぎた場合などに、その検索内容を改善するためのシステムである。この欠点を解消するために、AltaVista側で、検索結果を動的に分類し、関連語句のネットワーク構造を表示する。そして、利用者が指定する範囲の検索を実行できるようにしてある。

(2)　Untangle [Untangle 98]

Common Lispで記述したClassic　HTMLのインタフェースである。システムそのものを公開していないので、詳細な情報は不明である。

(3) FERMI [Fermi 98]

マルチメディア情報検索のモデルを開発するために、形式論理、不確実推論からの接近を試みているプロジェクトである。

3.9.6 まとめ

本節では、インターネット上の情報検索とオントロジー技術の関連について報告した。HTMLの拡張としてのマークアップ言語を設計し、それに人工知能技術を導入することで、高度な情報処理システムを開発するための新たな道具がそろいつつある。XMLを複雑な非定型データを扱うためのシステム開発の中心的なツールとして用いるというアイデアも試行の段階にきている。これらの技術をより確実なものとするためには、単純かつ強力なオントロジーの確立がきわめて重要であると考えられる。

3.9.7 オントロジー工学の課題

(1) オントロジーの捉えかた

我々の研究室では、さまざまな分野において知識システムを開発するという立場、その中から共通の方法論を確立するという立場から研究を行ってきた[小林 1990][寺野 1993]。その中で課題となったものは、(1) 知識をどう記述するかという知識表現の問題、(2) 知識をどのように抽出し定式化するかという知識獲得の問題、(3) 開発したシステムをどう保守し、普及させていくかという知識管理の問題、(4) 開発した知識ベースや方法論をどう再利用していくかという知識の共有と再利用の問題にわけることができる。1980年代はじめにAIブームが起こったときから、この順に課題が発生している。

オントロジー工学という立場から我々のシステム開発について考察するようになったのはごく最近である。これは、知識の共有と再利用の立場といってよい。この点から我々の知識システム開発を振り返ると以下が特徴的である。

- 知識システム開発の視点からのこれまでの研究経験のまとめ：
  タスク・オントロジーよりもドメイン・オントロジーを重視する立場で開発を行ってきたこと。これは、推論機構や開発ツールが与えられた状況のもとでの開発が

中心であったために当然の面がある。この意味で我々は「知識は力である」という知識工学の原理を非常に重視してきたことになる。

- オントロジーのメリット：
知識システム開発で得られた知識をオントロジーの視点で考察すると、これは、共通に扱える問題解決プリミティブを提供してきたということで非常に有用であったと考えられる。

上の立場から我々のシステム開発経験をとりまとめると以下のようになる。

(2) これまでの研究から

1: 知識システム開発のプロジェクト管理 ---
システム評価ガイドラインとプロセス管理手法の構築

ここで注目していたのは、システム評価のためのV&V（Validation & Verification）技術、ライフサイクルモデルの提案と評価、知識獲得作業の分類、エキスパートシステムのタイプなどさまざまな観点から利用が可能な、チェックリストベースのエキスパートシステム評価ガイドラインを提案することであった[Terano 1994]。そして、これを実際の知識システム開発プロセス管理に適用するために、品質機能展開（QFD）の手法と統合し、プロトタイプ手法に基づく知識システム開発の方法論の開発と評価を実施した[Abe 1997]。

この際、開発・利用したオントロジーは、チェックリストに表現される用語集ならびに品質機能展開表ということができる。これは、知識の共有を目的とする階層的知識（タクソノミー）を整理していたことになる。

2: 知識集約型の多戦略学習型知識システム ---
アミノ酸配列データからのタンパク質機能予測

この研究では、帰納的な学習手法をデータベース情報に適用し適切な仮説の候補を導出し、さらに、類推学習の機能を用いて仮説の候補を拡張する。最後に演繹的な学習手法により仮説の検証を試みるという多戦略学習の枠組を用いたシステム開発経験である[Ishikawa 1996]。

ここで、オントロジーとして重要な概念は、類推学習の基礎となる概念階層であった。これもドメインオントロジーとして性格が強い。ただし、類推計算への利用を目的とした知識であったために、単なる階層構造であるだけではなく、推論への利用を

強く意識した整理が必要であった。

3: 知識システムへの接近に創発的計算を適用 ---
アンケートデータ分析; プリント基板CAD

目的関数を明示的にもたないような複雑な問題に対して、遺伝的アルゴリズム（GA）を中心とする創発的計算手法を適用したシステム開発例である[石野 1997][Takadama 1997]。問題解決の中心には機械学習による問題解決器が存在するが、解候補生成の部分では創発的手法が重要である。これらのシステムでは、コンピュータのもつ高速な計算機能ならびに、生成検査のもととなる基本的な概念集合の同定が非常に重要であった。

この意味において、オントロジーとして重要なのは、比較的単純な対象分野の概念のプリミティブな要素であり、用語集的な意味で問題解決に必要な情報を収集することとなった。

（3） オントロジー工学の課題

1: システムニーズはどこにあるか

ほかの多くのプロジェクトと同様に、オントロジー工学においても技術的な課題に興味が集中しており、どのようなニーズが存在するかという分析が不十分である。私は、ニーズ分析が必要な対象分野として、まず、技術的に成熟しつつある分野の技術保存・技術移転を上げたい。現在、我国で高度な知識システム開発の担い手となっている大企業は、今後も、それぞれの能力に応じて、技術保存・技術移転を行っていくことができよう。しかし、個々の努力ではそれができないような分野--中小の機械企業や農業--も存在する。そのような分野においてこそ、潜在的な知識の共有や再利用のニーズは高く、また発展途上国に対する貢献度も大きい。これらの知識システム開発にあたっては、まず、データベースの整備・標準化など、研究になりにくい部分の検討が不可欠になる。

2: 運用管理の高度化にむけて

それぞれの対象分野／タスクにおいてオントロジーが整備されたとして、その開発・運用・管理は、当然、分散環境においてなされるものと期待される。それに対する技術的な検討課題として、運用管理を自動化する仕組の研究開発が重要である。

参考文献

[IJCAI 97] Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997.

[Yahoo 98] http://www.yahoo.com

[Altavista 98] http://www.altavista.digital.com

[Metacrawler 98] http://www.metacrawler.com

[Ahoy 98] http://ahoy.cs.washington.edu

[JIPS 97] 情報処理：特集：ネットワーク社会を支援する新しい知能メディア技術. Vol. 38, No. 1, 1997.

[JSAI 97] 人工知能学会誌：特集：大規模データベースからの知識獲得. Vol. 12, No. 4, 1997.

[Shoe 98] http://www.cs.umd.edu/projects/plus/SHOE/

[KA 98] http://carmi.cs.umd.edu/KnowledgeAnnotator.html

[IM 98] http://www.research.att.com/~levy/imhome.html

[Levy 96] Levy, A. L., Rajaraman, A., Ordille, J. J.: Query-Answering Algorithms for Information Agents. Proc. AAAI96, pp. 40-47, 1996.

[MCF 98] http://www.w3.org

[LT 98] http://www.altavista.digital.com/av/lt/help.html

[Untangle 98] http://www.cs.vassar.edu/faculty/welty/classic-dixt.html

[Fermi 98] http://www.dcs.gla.ac.uk/fermi/

[Bosak 97] Bosak, J.(富士ゼロックス情報システム訳)：XML,Java,そしてWebの将来. http://www.fxis.co.jp/DMS/sgml/xml/xmlapps.html.

[小林 1990] 小林重信，寺野隆雄（編・著）：知識システムハンドブック．オーム社，1990年11月．

[寺野 1993] 寺野隆雄：知識システム開発方法論．朝倉書店，1993年４月．

[Terano 1994] Terano, T.: The JIPDEC Checklist-Based Guideline for Expert System Evaluation. International Journal of Intelligent Systems, Vol.9, No.9, pp. 893-925, 1994.

[Abe 1997] A. Abe, T. Terano, T. Yoshizawa: QFD Approach to Managing Prototyping Processes for Knowledge-Based Scheduling Systems IJCAI-97 Workshop on VV&T, pp. 61-64, 1997.

[Ishikawa 1996] Ishikawa, T., Terano, T.: How to Predict it: Inductive Prediction by Analogy Using Taxonomic Information. Proceedings of the Third International Workshop on Multistrategy Learning (MSL-96), pp. 295-303, 1996.

[石野 1997] 石野洋子, 寺野隆雄：模擬育種法と帰納学習を適用したマーケティング情報分析. 人工知能学会誌, Vol. 12, No. 1, pp. 121-131, 1997.

[Takadama 1997] Takadama, Keiki, Terano, Takao: Good Solutions will Emerge without a

Global Objective Function: Applying Organizational-Learning Oriented Classifier System to Printed Circuit Board Design Proc. IEEE SMC'97, pp. 3355-3360, 1997.

3.10　医療関連　GAMES-II [GAMES-II]

3.10.1　概　要

GAMES-IIは、医療知識ベースシステム構築のための一般的方法論の確立を目指したEuropian AIM projectの一つである。AIMはここではAdvanced Informatics in Medicineを示しておりTelematics programmeのサブプログラムとなっている。GAMES-IIはAIM Workplanの中で、1992年から1995年までの3年間実施され、主としてTask 242、「医療知識ベースシステムのアーキテクチャ」に関連している。協力機関はSAGO（Florence）、University of Pavia（Italy）、University College of London、University of Amsterdam、University of Ulm、University Hospital of Geneve、Forth（Crete）であった。プロジェクト総額は5百万ECUである。

3.10.2　知識ベース構築の考え方

GAMES-IIは医療、特に治療を中心とした医療のための知識ベースシステムを構築するための方法論を明らかにすることを目標にしている。

GAMES-IIにおける知識ベースシステム構築方法論の考え方は、[Wielinga et al, 1989]に基づいている。すなわち、World view、Theory、Methods、Tools、Useという基本的な5つの側面から検討を加えることができる。

① World view

GAMES-IIにおけるWorld viewは知識レベルのモデリング、再利用可能性、統合化という3つの目標を達成することである。これらの目標を達成するために、Theoryは、認識論的モデル（epistemological model）及び計算モデル（computational model）という二つの観点から構成されなければならない。認識論的モデルではさらに、推論モデル（inference model）、オントロジー、汎化医療タスク（generic medical tasks）を考慮する必要がある。また、計算モデルでは、黒板アーキテクチャ（blackboard architecture）と問題解決器（problem solver）が問題となる。

② Theory

1) 認識論的モデル

最初に認識論的モデルについて説明しよう。このモデルは、対象分野（すなわち医療）の専門家がシステムのモデリングを行うための方法論を提供するものである。ここでは単一の推論モデルとしてのSTModel（Select and Test Model、選出・検査モデル）

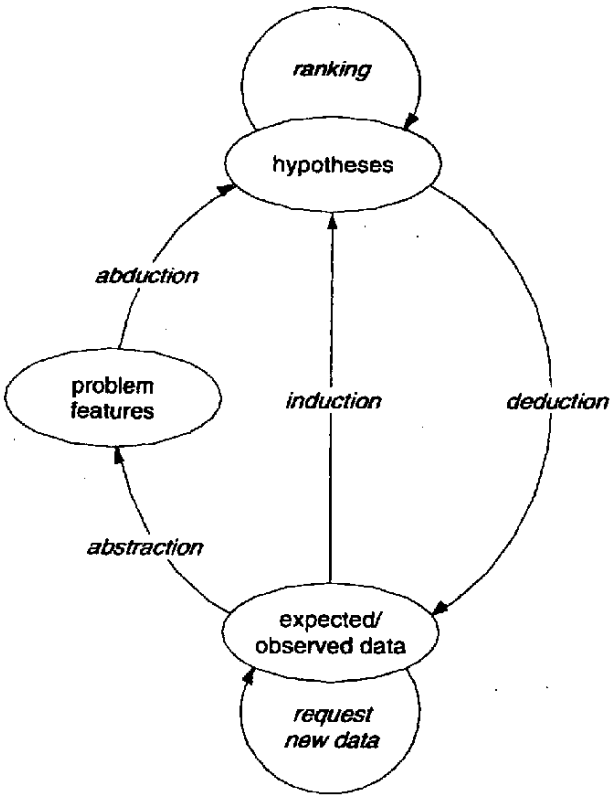


図３．１０．２-１　STModel（Select and Test Model）

が採用されている。その概要を図３．１０．２-１に示す。このモデルでは、すべての医療推論は、観測結果あるいはそれに適切な抽象化を行った結果から仮説を形成し、更に実データとの対照によって仮説を検証していく過程として捉えることができると考えている。このSTModelに関連して、推論のタイプとしてdeduction、abduction、inductionが存在することが指摘される。

実際の応用分野の問題をこのような推論モデルと結び付けるためには、対象分野に関わるオントロジー（application ontology、アプリケーションオントロジー）が確立されていなければならない。医療分野において取り扱うべき概念の膨大さやレベルの多様さを考えると、全体を一つのオントロジーで統一するような汎用的なオントロジーの構築は困難であり、オントロジーライブラリを用意し、必要に応じてその内容を検索し、適切な修正を加えてアプリケーションオントロジーを構築することが適切である。多様なオントロジーをライブラリとして提供するため、その全体量は膨大なものとなる。GAMES-IIは医療分野で高頻度に使用されるいくつかの概念セットをあらかじめ提供するとともに、新しい概念セットを定義したり、概念セットを組みあわせる際の指針を示している。これらのオントロジー記述のためにはoutolinguaを用いている。

アプリケーションオントロジーは、医療知識ベースシステムが実際の医療の現場で

利用されるものとなるためにも重要な役割を果たす。医療知識ベースシステムは単独で用いられるというよりは、むしろ医療情報システム全体の中に組み込まれて利用されることが要求される。これは、医療知識ベースシステムが外部のさまざまな情報システムとの間で情報交換が可能でなければならないことを示している。アプリケーションオントロジーは、このような情報交換を円滑に行なうことを助ける働きを持つ。

オントロジーライブラリの構築に当たって、その中核には医療分野に共通に用いられる基本的カテゴリが設定されているが、これはいくつかの基本的な領域について定義されたTheoryの集合であり、Theoryの間に参照関係が定義されている。この中核的定義の周辺にさまざまなアプリケーションに対応するオントロジーが定義されるが、それらは基本的には実際のシステム実装方法を検討することによって得られたものであり、1. 領域特殊性、2. タスクを実現するメソッドの特殊性という二つの観点から索引づけられている。これらの関係を図３．１０．２-２に示す。

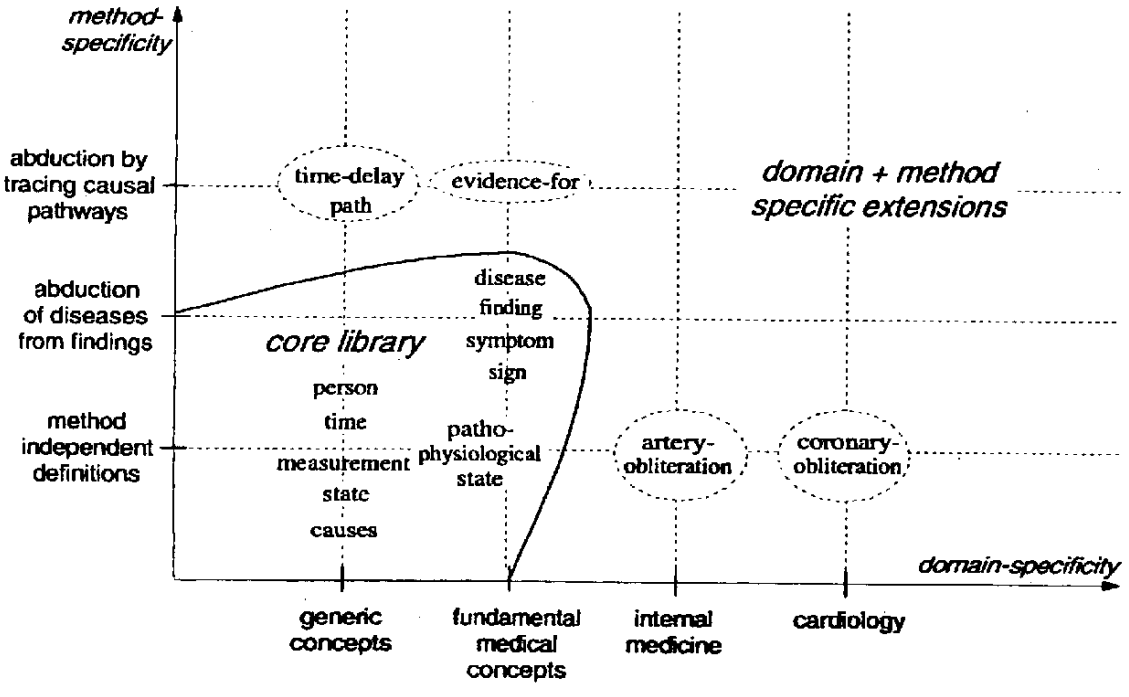


図３．１０．２-２　医療オントロジーの特殊化系列例

認識論的モデルの最後の要素としては、実際の推論を行うための、汎化医療タスクが必要である。タスクは一般に、1. その目的、2. 採用される推論モデル、3. 推論制御方法という３つの側面を持つ。

GAMES-IIで取り扱われるシステムでは医療における代表的なタスクである、診断、治療方法選択、患者状態監視という３種類を想定する。これらのタスク実行のために

は、幾つかのサブタスクが必要となるが、個々のサブタスクはいずれもSTModel（あるいはその一部）と考えられる推論モデルに該当する。医師はこれらのサブタスクを適切な推論制御の下に実行していると考えることができる。

必要とされるタスクを（認識レベルで）決定するためには、推論モデルとアプリケーションオントロジーの間の対応づけが必要である。診断問題の場合はこの対応づけは比較的単純で、仮説（hypotheses）は疾患や病態に対応し、データや問題構造（problem feature）は所見（findings）に対応する。この意味で診断問題ではこの対応づけは単純な語彙間の対応付けであるといえるであろう。

これに対して治療計画立案の場合、データや問題構造（problem feature）のマッピングはより複雑である。多くの場合これらに該当するのは（おそらく）診断の結果として得られた疾患・病態であるが、患者の状態やその立場、あるいは最終的に患者をどのような状態に導きたいかなどを考慮しなければならない場合もある。また、患者監視の問題では、患者状態の予測が必要となるし、さらに診断のやり直しや治療計画の変更（再立案）が必要となる場合もある。

2） 計算モデル

構築された認識論的モデルは実際に計算機で取り扱うことのできる形に変換されなければならない。このために計算モデルが必要となる。GAMES-IIでは黒板アーキテクチャと問題解決器が計算モデルを構築する主要な要素となる。

黒板アーキテクチャはサブタスク間の情報交換媒体として、また、推論制御の目的で利用される。元々黒板モデルでは各知識源（knowledge source）はルールの集合として定義されていたが、ここでは内容を関知しない（black box）知識源の存在も認める。競合する知識源の中から実行されるものを選ぶ推論制御のためにはメタルールを用いる。

黒板は実際には分割されており、各STModelの知識ロール（knowledge role、hypothesesやproblem featureなど）が異なる場合、異なる空間に配置されることとなる。さらに、推論モデルに対する役割に応じてそれぞれの空間に書き込み禁止などのアクセス制限を設定することもできる。

実際の推論を実行するための知識源として幾つかの問題解決器が用意されている。先に延べたようにこれらの知識源はルール集合として実現されることも、また、別の手続きの集まりとして実現されることもある。

問題解決器は、認識論的モデルで設定されたサブタスクを忠実に実行できるものでなければならない。ある場合には出来合いの問題解決器が利用できるであろうが、一方で要求されるサブタスクに合わせて問題解決器を新規に作らなければならない場合もある。また、一般に認識論的モデルにおけるアプリケーションオントロジーは、そのままの形では問題解決器が直接に取り扱えるデータ型に直接対応するものになって

いるとは限らず、何らかの変換を必要とする場合もある。

あらかじめ用意されている問題解決器としては、例えば因果確率ネットワーク(causal probabilistic network)、定性シミュレーション(qualitative simulation)、連想推論(associational reasoning)等がある。

実際の問題解決器が前提とするデータ型が、解決器に依存するということから、サブタスクが協調できるためには黒板上のデータ構造と、解決器が前提とするデータ構造の間の変換ができなければならない。この変換のために中間的なデータモデルとし表現能力モデル(representational competence model)というデータ構造が(複数)定義されている。アプリケーションオントロジーは自動的にこの構造の上にマッピング可能となっているから、解決器の実装にあたっては、解決器の要求するデータ型に最も変換しやすいモデルを選び、そのモデルと解決器の前提とするデータ型の間の変換を定義してやることによって要求されるデータ形式変換が可能となる。

③ Methods

次に必要となるのは実際にシステムとして実装を進めていくための方法論(method)である。ここでは認識論的モデルの構築と、それに基づく計算モデルの実装を規定する方法論が提供されなければならない。この方法論は、ドメインエキスパートと知識工学者の分業を達成できることが望ましい。つまり、計算モデルの詳細を理解する必要なく、これとは独立に認識論的モデルをドメインエキスパートが構築できることが望ましいし、一旦認識論的モデルが構築されたならば、知識工学者だけでそれを計算モデルとして実装できることが望ましい。

認識論的モデル構築の手順としては、

1. アプリケーションのタスクモデルの決定
2. タスクに関連するオントロジーの選択と洗練(必要なら)、または作成(configure)
3. アプリケーションオントロジーのタスクの知識ロール(hypotheses、problem feature 等)上へのマッピング
4. アプリケーションオントロジーのインスタンス化(アプリケーション知識化)

を行うこととなる。一方、計算モデルを構築するためには、

1. メタルールの選択または定義
2. 問題解決器の選択
3. データ変換方法の決定

が必要である。

④ Tools

実際にシステムを設計・構築するための上記の作業を支援するためのツールもまた重要である。GAMES-IIではCUEと呼ぶworkbenchを用意している。CUEはいくつかのツールを提供するがその主要なものとして、アプリケーションオントロジーの選択、作成のためにQUOTE、定義されたオントロジーに基づくドメイン知識入力のためにQUAKEというグラフィカルな編集ツールを用意している。さらに、タスク構造を構築・編集するためのツールとしてQUITEを用意している。

また、一般的なツールでは定義しにくい種類の問題を定義するため、幾つかの知識モデルに関しては専用のツールを用意している。たとえば、因果確率ネットワーク構築のためのグラフカルツールGAMEESや、生理学的定性モデル構築のためのグラフィカルツールQCMFなどがある。

実際のシステムとしての実装を行なうためには、これらのツールを用いて構築された認識論的モデルを計算モデルに対応づける必要がある。M-KATはこのような対応づけを可能とするための環境であり、黒板モデルの構築、問題解決器の選択、メタルールによる推論制御構造の定義・編集を支援するものである。

⑤ Use

GAMES-IIは医療分野の知識ベースシステムを構築するための汎用の方法論を提供するものである。このような方法論の有効性は実際に方法論にしたがってシステムを構築することを通して検証する必要がある。幾つかの分野を実際に選んでシステム実装を行っている。GAMES-II最終報告書には、GAMESの方法論に従っていくつかのシステムを実際に構築した例が紹介されている。

参考文献

[GAMES-II] http://www.swi.psy.uva.nl/projects/GAMES-II/home.html

[Wielinga et al, 1989] Wielinga, B.J., Schreiber, A.T., & de Greef, P. (1989). Synthesis report. ESPRIT Project P1098. Deriverable Y3, University of Amsterdam.

3.11　データベースとオントロジー

3.11.1　はじめに

データベースという分野には、関係データベースから演繹データベースのように推論機能をもったものまで幅広く含まれている。とくに演繹データベースの研究では、論理プログラミングや高次推論などの分野に大きな貢献を行っており、単なるデータの管理だけを意味していないので、文脈に応じて、知識データベースと呼んだり、単に知識ベースと呼ぶこともある。もちろんこれは知識工学の立場での「知識ベース」とは大きく異なっている。データベースのもっとも重要な概念のひとつは、スキーマを始めとするデータ抽象概念であるが、これはさまざまな応用に対応するために、より柔軟なものにと変化している。たとえば関係データベースの場合、関係スキーマと一貫性制約によって定義できるが、これは実世界の一部データを切り出したに過ぎない。応用が高度化するにしたがって従来切り捨てた部分をもモデル化する必要が出てきた。より広範なデータを扱うためには、必ずしも構造化されていないデータをも対象にする必要があり、そのためにはラベル付きグラフのようなより柔軟なデータ概念を使用せざるをえなくなっており、さらに意味情報の扱いが議論されている。

動作環境としては、単体のデータベースだけでなく、ネットワークの急速な発達によって、ネットワークを意識した分散データベースやマルチデータベースがますます重要となってきている。マルチデータベースは異種データベースを統合することを目的にしており、関係データモデルやオブジェクト指向データモデルなど異なったモデルをいかに統合するか、またスキーマ間に意味的異種性があった場合、それらをいかに解消するかという研究が行われてきた。これらのアプローチのひとつにメタデータをいかに充実させるかがある。たとえばMetadata'96（http://www.computer.org/conferen/meta96/meta_home.html）やMetadata'97（http://www.llnl.gov/liv_comp/metadata/md97.html）ではドメイン固有の情報をメタデータとして持たせ、それによって相互運用性を高めたり統合を容易にする研究等が多く述べられている。これらに関連したものとしてはhttp://www.llnl.gov/liv_comp/metadata/other_efforts.html を参照されたい。またマルチデータベースの部分として連邦データベース（federated databases）という考えがあるが、ここでは意味的異種性はあまり検討されていない（たとえばhttp://wwwiti.cs.uni-magdeburg.de/~conrad/EFDBS97/）。

一方で、インターネットを始めとするネットワークの急速な発達によって、ネットワーク上のさまざまな情報源をアクセスし、利用することが可能になっている。これらをデータベースの観点から利用しようとすると、各情報源のモデルをいかに構成するか、情報源間の矛盾をいかに解消するか、それらをいかに融合するか、等の研究が行われている。このためにはマルチデータベースをより拡張した、データベースを越

えたアプローチが必要となる。Webそのものをターゲットにしたもの（たとえばhttp://poincare.inf.uniroma3.it:8080/webdb98）と、より一般的に異種情報源を統合しようというアプローチ（たとえばメディエータ）がある。

データベース分野で、異種データベースあるいは異種情報源を統合しようとする研究はさまざま行われており、スキーマ変換、意味的異種性の解消、メタデータ記述の拡張、データモデルの柔軟化などさまざまなアプローチがあるが、それらをオントロジーという立場で研究しようという研究はまだ端緒に付いたばかりである（Journal of Parallel and Distributed Databasesで「データベースとオントロジー」の特集号が初めて組まれることになっている（http://www.cs.auc.dk/~busatto/db-events/dbworld/dpd-si））。

本報告では、まず異種情報源の統合の先駆的研究を行い、またメディエータシステムの提案者として知られているスタンフォード大学のGio Wiederhold教授のSKCプロジェクトを紹介し、次にメディエータシステムの代表例としてこれもスタンフォード大学のTSIMMISおよびIBMのGarlicを紹介する。そして最後にその他の議論を簡単に紹介する。

3.11.2 SKCプロジェクト

メディエータ概念の提唱者であるスタンフォード大学のGio Wiederhond教授の関わっているプロジェクトは総称してLarge-Scale Interoperation and Composition（LIC）(http://www-db.stanford.edu/LIC/)と呼ばれており、以下のプロジェクトから構成されている。

- Managing ontologies, and developing an Ontology Algebra to assure Scalability (SKC).
- Compiling High-level Access Interfaces for Multi-site Software (CHAIMS).
- Protection of Privacy in Collaborative Settings, specifically Trusted Interoperation of Health care Information (TIHI) on the Internet and creating Secure Access Wrappers (SAW) for manufacturing data. Both projects are subcontracts through SRI International.
- Mediators for access to Health Information (PDQ, Cancerlit) at NLM, CAMIS, through Lexical Technology Corporation (LTI).
- Mediators for fusion of data for Environmental Restoration at the Idaho National Engineering Labs, (INEL-ERIS).
- Mediators for Feedback in Training (MIFT) (old MIFT).
- A Simulation Access Language (SimQL) enabling integration and reuse of simulations within information systems.
- Processing of Image data for Digital Libraries and Medicine. Also initiating research with

ICBM at UCLA and LRI at UCSF.

・A meta issue for all these efforts is improving the transition of technology into practice, which I hope to accomplish through early engagement and support to companies who can help in development and then serve as demonstration and transfer sites to end-users.

この中でSKCがオントロジーに関係しているのでそれを紹介しよう。

SKCはScalable Knowledge Compositionの略で、DARPAのHigh-Performance Knowledge Base（HPKB）プログラムと共に、AFOSRのサポートを得ている（http://www-db.stanford.edu/SKC/）。プロジェクトは2000年まで続けられる。SKCは情報システム間（分散した大規模知識ベース）の意味的異種性を解消し、無矛盾性を保持することを目的としている。

SKCの問題意識としては以下のものがある。

・どんな単一の知識表現言語も支配的にはなっていない。Ontolinguaは広く配布されているが、応用システムで広く使用されているわけではない。
・もっとも解消が容易でよく現れるものが、ドメインによる名前付けの違い、つまり同意語の問題である。
・メンバーシップ関連のスコープの差異もよく現れる問題だが、これもドメインによって決定されることがほとんどなので、ドメイン内部の変数を参照するルールによって差異を解消する必要がある。
・セットされる値のコーディング法の差異もよくある問題だが、これもテーブルで比較的簡単にできる。
・抽象レベルの違いも大きな問題で、追求しなければならない問題である。

SKCではこれらすべての問題の解決を目指しているが、新規な解決法を追求するより、大規模知識ベースのスケーラビリティに力をいれている。

SKCでの基本的な考え方は、大規模知識ベースに無矛盾性を要求するのはたとえ強権を使っても不可能であるため、個々に無矛盾性を保守できる知識ベースに分割し、それらを接合(articulation)するためのルールを別に準備することである。分割された個々の知識ベースは互いに共有部分を持たない自律的な無矛盾なオントロジーを持っており、その保守に互いの同意は必要としない。この意味で、（分散した）大規模知識ベースの保守には、オントロジーの保守を小さく、共通部分のない、特殊なドメインに分割することが、最上のもので意味的相互運用性をスケーラブルにする唯一の方法と考えている。

分割された知識ベース間の関連付けを行うために、異なった自律的なドメインの語彙を表現するオントロジーに関する代数を提案している。これは関係代数用の演算を

オントロジー間に定義したものである。たとえば共通部分演算はドメイン間にリンクを張ることによって2つのオントロジーの接合演算と同一視できる。各オントロジーは自律的に保守されるが、接合ルールはそれらを共有することに利益のあるグループが保守する。SKC プロジェクトはオントロジーが相互運用可能な接合ルールを使う方法を発達させることを目指している。

ここでオントロジーというのは、用語（term）の集合とそれらの間の関連（relationship）と定義している。用語は実世界や抽象的オブジェクトの集合に対応している。抽象的オブジェクトとは他の抽象的オブジェクトや実世界オブジェクトの集合に対応する概念である。実世界オブジェクトの集まりは定義の基礎であり、用語の意味の正しさの検証のために用いられる。用語からオブジェクトへの写像はドメインごとに異なっている。関連は意味的かつ構造的に重要である。Is-a、所有、Part-of、参照、… のような構造的関連型を多く共有されている。SKC の関連をさまざまなオントロジーで採用されている関連に写像することは、これも各ドメインごとに異なっている。この意味では、SKC で考えているオントロジーは比較的浅いところを対象にしているいえる。

SKC 代数が考えているオントロジー型の例として以下のものがある。

* オブジェクト指向クラス階層
* ER モデルや構造モデルで議論されるようなデータベーススキーマ
* 半構造化データベース
* 加算集合の型仕様
* 定義可能なシソーラス

SKC ではオントロジーの定義として、内包的（スキーマ）知識による外延（オブジェクト）へのアクセスの処理をサポートするため、内包的仕様の中に実世界オブジェクトへアクセスする述語あるいはメソッドを含んでいる。オントロジーがドメインに関する完全な記述となっていることを要求するのではなく、オントロジーの使用がそのオントロジーに関して完全な結果を提供することを要求している。

オントロジーで互いの概念間の照合関係は以下のように記述される。

SHOE (store) = SHOE (factory)
SINKER(carpentry) in NAIL (householder)
{PUMPS, WEDGIES, LOAFERS, SANDALS}(store) ISA SHOE(factory)
PUMPS (store) = SHOE(factory) if HEEL(factory) > 5cm.

これらは store と factory の間の概念間の対応関係を示している。すべての用語間に

共通部分がないという前提に立っているのでSHOEに関しても上記のような指定を行なうことになる。このルールで表現されていない概念については、他の知識ベースからアクセスすることはできない。付加的なテーブルを使用する場合には以下のように指定する。

```
if (LOCATION (factory) ='EUROPE')
        then SIZE (factory) = SIZETABLE (SIZE (store));
        else SIZE (factory) = SIZE (store);
COLORCODE (factory)= colortable (COLOR (store)),
```

これらオントロジーに対する演算（オントロジー代数）は以下のように集合操作に基づいている。

1. Union(A,B) または A U B:
   A と B におけるすべての一意的なエントリの集まり
2. Intersection(A,B) または A ^ B:
   A と B で共有されるすべてのエントリの集まり
3. Difference(A,B) または A - B:
   B とは共有されない A のエントリの集まり

これら操作を各知識ベースで可能にするためには、各知識ベースでこれら演算に対応する機能を持たせなければならないが、それは使用されている知識表現言語に依存している。

SKC のアプローチは、スケーラブルな知識ベースという観点からは、現実的で堅実なものだが、

・各ドメインがすべて一意的な用語をもっているということで「冗長な」定義が多く含まれること
・オントロジー間の矛盾の解消を考えていないこと
・オントロジー自体の大規模化を対処できないこと

などに問題がある。

3.11.3 メディエータシステム

メディエータ（mediator）という用語も Gio Wiederhold によって用いられたのが最初であるが、現在では一般的に、複数情報源（データベース、知識ベース、など）からの情報を融合あるいは統合するソフトウェアモジュールを指している。各情報源はラッパー（wrapper）と呼ばれるソフトウェアモジュールによってカプセル化され共通モデルに変換される。メディエータは、メディエータ仕様記述言語での指定に従い、ラップされた情報源から情報を収集し必要とされる形に情報を加工する。メディエータが外部に見せる情報が共通モデルであれば（あるいはさらにラップされ他の共通モデルに変換されれば）それは他のメディエータから参照することが可能となる。つまり階層構造とすることができる。このようなシステムを総称して、メディエータシステムと呼んでいる。

メディエータシステムとしては、スタンフォード大学と IBM Almaden 研究所の共同プロジェクトである TSIMMIS（http://www-db.stanford.edu/tsimmis/tsimmis.html）、AT&T の Information Manifold が有名である。Information Manifold については 3.9 節で紹介されるのでここでは省略する。また IBM Almaden 研究所が独自で立ち上げた、より具体的なシステムを目指す子プロジェクト Garlic（http://www.almaden.ibm.com/cs/garlic/homepage.html）についても本節で紹介することにする。

TSIMMIS は The Stanford-IBM Manager of Multiple Information Sources の略であるが、それはまた、「異種の」果物と野菜を統合し全体としてはすばらしい味に仕立てたシチューを意味するユダヤ語でもある。このプロジェクトは DARPA によってサポートされている。このプロジェクトの目的は、構造化、半構造化、あるいは両方のデータを含む情報源を高速に統合することを容易にするツールを開発することである。以下の構成要素からなっている。

・ラッパー
・メディエータ
・Web ページからの情報の抽出
・Web 上のデータ源のブラウジング

このプロジェクトでは、情報源間の意味的な異種性はラッパーあるいはメディエータで解消する方針をとっており、「オントロジー」という独立したものは考慮しておらず、また用語は使用していない。SKC に比べ、この TSIMMIS のアプローチがこれまでのデータベース分野ではより一般的であったと考えられる。

共通モデルとして OEM（Object Exchange Model）を使用するが、以下のように定義している。

```
<&r1,report,set,{&r1n,&r1a,&r1t,&r1r}>
  <&r1n,rn,string,'AB-123'>
  <&r1a,authors,set,{&r1a1}>
    <&r1a1,author,string,'John Patriot'>
<&r1t,title,string,'UN Conspiracies'>
<&r1r,rel,set,{&r2}>
  <&r2,report,set,{&r2n,&r2a,&r2t,&r1r}>
```

ここで第1引数の &rxx はオブジェクト識別子であり、第2引数は属性名、第3引数は属性値の型、第4引数は属性値である。OEM は情報源に対する一種のビューであり、半構造化データを含む情報源に対する共通モデルを定義している。データ構造としてはオブジェクト識別子をもった複合オブジェクトといえる。

メディエータは、複数の情報源のそれぞれの共通モデルを用いて、統合された一種のビューを定義する。たとえば上に関連した例をメディエータ仕様記述言語 MSL（Mediator Specification Language）で示すと、ルールで以下のように表現できる。

```
<trep(RN) tr {<title T>}@m :-
  <report {<rn RN> <title T>}>@s1
<trep(RN) tr {<postscript T>}@m :-
  <report {<rn RN> <postscript T>}>@s2
```

ここでヘッドの "@m" はその前の情報がメディエータで得られることを示し、ボディ部の "@s1" "@s2" は対応する情報が情報源 s1, s2 からそれぞれ得られることを示している。

この TSIMMIS の能力を考えるために、表現を Datalog に限定し、上で述べた Information Manifold と比較してみるとわかりやすい。たとえば情報源 s1, s2, s3 が以下のように定義されていたとする。

```
s1(E,P,M)←employee(E),phone(E,P),manager(E,M)
s2(E,O,D)←employee(E),office(E,O),department(E,D)
s3(E,P)←employee(E),phone(E,P),department(E,toy)
```

このとき、従業員 sally の電話番号とオフィスを問い合わせる質問を考えよう。以下のように表現できる。

```
q1(P,O)←phone(sally,P),office(sally,O)
```

しかしこれを上の情報源から得ようとしても、通常の論理プログラミングの導出では単一化できる述語がないため、たちまち失敗してしまう。Information Manifold では description logics を元にしているが、問合せは各サブゴールをすべての情報源に投げ、解の可能性があるものの組合せを生成している。たとえばこの例では問合せは

```
answer(P,O)←s1(sally,P,M),s2(sally,O,D).
answer(P,O)←s3(sally,P),s2(sally,O,D).
```

のように２種類が生成される。それぞれをさらに展開し単純化すると、

```
answer(P,O)←employee(sally),phone(sally,P),manager(sally,M),
    office(sally,O),department(sally,D).
answer(P,O)←employee(sally),phone(sally,P),department(sally,toy),
    office(sally,O),department(sally,D).
```

となる。ここで注意すべきなのは、通常の論理プログラミングの観点からすると、問合せ q1 との意味が変わってしまっていることである。Information Manifold ではメディエータはとくに指定せず、存在するかどうかは不明であっても、すべてのサブゴールをすべての情報源に投げることによって問合せを処理している。オントロジーという観点がここでは欠如している。

一方 TSIMMIS では上記のままでは何も起こらず、以下のようなメディエータを事前に指定する必要がある。

```
epo(E,P,O)←s1(E,P,M),s2(E,O,D)
epo(E,P,O)←s3(E,P),s2(E,O,D)
edm(E,D,M)←s1(E,P,M),s2(E,O,D)
```

つまり、epo は情報源 s1 と s2, s3 と s2 を統合することによって得られることを定義するのである。edm は別のメディエータ仕様となっている。これが与えられることによって問合せは、

```
answer(P,O)←epo(sally,P,O)
```

となり、メディエータ仕様によって、

```
answer(P,O)←s1(sally,P,M),s2(sally,O,D)
answer(P,O)←s3(sally,P),s2(sally,O,D)
```

のように展開される。この例ではたまたま結果は同じになったが、論理的な意味が大きく異なっていることに注意されたい。TSIMMIS は、情報の探索としては静的で面白みに欠けるが、オントロジーの観点からすれば、現在はまったく考慮されていないが、メディエータ仕様と結合すると、面白い結果が得られそうである。

TSIMMIS の子プロジェクトと位置付けられている Garlic では目的をもって限定することによって、よりシステム色の強いものを目指している。対象分野は大規模マルチメディア情報システムであり、分散／分割されたデータのそれぞれは特定の型を持ち、それに応じて充分な管理がなされていることを前提にしている。結果としてラッパーのアーキテクチャ、問合せの最適化など伝統的なデータベース技術の適用が可能となっている。具体的な応用分野としては、医学、インテリア設計、広告代理店などが挙げられている。

たとえば医学の分野では、病院はさまざまな情報は部局毎にさまざまな形で管理されており、それらをいかに統合するかが紹介されている。放射線科では MRI のスキャンなどの画像データがあり、心臓科では EKG データなどがあり、研究室では報告書があり、事務部門では患者の記録があり、それぞれのデータに応じて適した管理システムが採用されている。それぞれのシステムに対応してラッパーが作成され（たとえば画像ラッパー、文書ラッパー、関係ラッパーなど）、それらがミドルウェアとしての Garlic スキーマに統合される。ここで採用されているのは ODMG ベースのオブジェクト指向モデルである。この統合のためにメタデータが使われるが、これが（浅い）オントロジーに対応している。特定の患者のデータを矛盾なく管理し効率的に取り出すためには、このように分散管理されている異種データを統合することが有効であろう。

3.11.4　その他の議論

SKC と TSIMMIS を中心に紹介したが、この周辺ではさまざまなトピックが議論されている。ひとつの大きな流れは、TSIMMIS は静的でかつ構文的な側面しかとらえてないということから、それを動的にさらには意味的側面をもとらえたものにしようという研究である。

現在岡山県立大で研究開発している（分散）知識表現言語 QUIK は TSIMMIS に比べて以下の特徴を持っている。

・表現モデルが演繹オブジェクト指向パラダイムに基づいているため、包摂関係／制約を含む豊富な表現能力をもっている。
・オントロジーは拡張データディクショナリとして保持し、情報源間のオブジェクト

の同定に使用される。
・情報源を統合するためには局所的情報の大域化が必要となるが、その際、束生成などを始めとするチェック環境によって、動的なオブジェクト同定を可能にしている。
・情報の探索として、アブダクションを利用した情報源の自動探索を含む導出を可能にしている。

目標は単なる情報源の統合だけでなく、知識ベースを含む問題解決器の統合に置いているために、TSIMMIS や SKC に比べ高度な能力を追求している。

また意味情報の表現としては、description logics の採用、それに対応したが多くの言語が提案されている。上の QUIK を含め詳細は http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/ を参照されたい。とくにこの中の KRDB'97 は SIGMOD Record でも紹介されている（http://www.cs.umd.edu/areas/db/record/）。

参考文献

[SKC 関係]

1. Gio Wiederhold and Michael Genesereth: "The Conceptual Basis for Mediation Services"; to appear in IEEE Expert, 1997; presented earlier as "Basis for Mediation" in the Proc. COOPIS'95, Vienna Austria, May 1995.
2. Maluf, David A.: "Implementing Articulation Rules for Object Request Brokers as an Extension to Production Systems "; submited to the IEEE Knowledge and Data Engineering Exchange Workshop, Newport Beach, California, 1997.
3. David Maluf and Gio Wiederhold: Abstraction of Representation for Interoperation; To appear in the 10th Int. Symp. on Methodologies for Intelligent Systems, Lecture Notes in Computer Science, Springer Verlag , 1997.
4. Wiederhold, Gio: "Objects and Domains for Managing Medical Knowledge"; Methods of Information in Medicine, Schattauer Verlag, Vol.34, No.1, pp.1-7, March 1995; presented earlier in "Proc. IMIA WG6 meeting, 1994.
5. Wiederhold, Gio: Interoperation, Mediation, and Ontologies"; Proc. Int. Symp. on Fifth Generation Computer Systems (FGCS94), Workshop on Heterogeneous Cooperative Knowledge-Bases, Vol.W3, pp.33-48, ICOT, Tokyo, Japan, Dec. 1994.
6. Wiederhold, Gio: "An Algebra for Ontology Composition"; Proc. of 1994 Monterey Workshop on Formal Methods, Sept 1994, U.S. Naval Postgraduate School, Monterey CA, pp.56-61.
7. Wiederhold, Gio: "The Roles of Artificial Intelligence in Information Systems"; J. of Intelligent Information Systems, Vol.11, No.1, 1992, pp.35-56.

[TSIMMIS 関係]

1. S. Chawathe, H. Garcia-Molina and J. Widom. "Flexible Constraint Management for Autonomous Distributed Databases." IEEE Data Engineering Bulletin, Vol. 17, No. 2, June 1994, pp. 23-27.
2. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J.D. Ullman, and J. Widom. "The Tsimmis Project: Integration of Heterogeneous Information Sources." Proc. 100th Anniversary Meeting, IPSJ, Oct., 1994, Tokyo, Japan, pp. 7-18.
3. S. Chawathe, H. Garcia-Molina and J. Widom. "A Toolkit for Constraint Management in Heterogeneous Information Systems". Proc. IEEE Int. Conf. on Data Engineering, pp. 56-65, New Orleans, Feb. 1996.
4. H. Garcia-Molina , Y. Papakonstantinou , D. Quass , A. Rajaraman , Y.Sagiv , J. Ullman , V. Vassalos, and J. Widom. "The TSIMMIS approach to mediation: Data models and Languages". Journal of Intelligent Information Systems, 1997.
5. J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo."Extracting Semistructured Information from the Web". Proc. of the Workshop on Management of Semistructured Data. Tucson, Arizona, May 1997.
6. J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantinou, J.D. Ullman, and J. Widom. "Information Translation, Mediation, and Mosaic-Based Browsing in the TSIMMIS System." Proc. ACM SIGMOD Conf., June 1995, page 483.
7. J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, R.Yerneni. "Template-Based Wrappers in the TSIMMIS System". Proc. ACM SIGMOD Conf., Arizona, May, 1997.
8. Y. Papakonstantinou, H. Garcia-Molina and J. Widom. "Object Exchange Across Heterogeneous Information Sources." Proc. IEEE Int. Conf. on Data Engineering, Mar. 1995, pp. 251-260.
9. Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, J. Ullman. "A Query Translation Scheme for Rapid Implementation of Wrappers". Proc. Int. Conf. on Deductive and Object-Oriented Databases, 1995.
10. Y. Papakonstantinou, H. Garcia-Molina, J. Ullman. "Medmaker: A Mediation System Based on Declarative Specifications". Proc. Int. Conf. on Data Engineering, pp.132-141, New Orleans, Feb., 1996.
11. Y. Papakonstantinou, S. Abiteboul, H. Garcia-Molina. "Object Fusion in Mediator Systems".Proc. VLDB, Bombay, India, September 1996.
12. D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, and J. Widom. "Querying Semistructured Heterogeneous Information". In Int. Conf. on Deductive and Object-Oriented Databases, 1995.
13. A. Rajaraman, Y. Sagiv, and J.D. Ullman. "Answering Queries Using Templates with

Binding Patterns". Proc. the 14th ACM PODS, pp. 105-112, San Jose, California, May 1995.

14. V. Vassalos , Y. Papakonstantinou. "Describing and Using Query Capabilities of Heterogeneous Sources". Proc. VLDB, Athens, Greece, August 1997.

4．オントロジー工学のまとめ

4．オントロジー工学のまとめ [溝口97]

4.1　オントロジー再考

ここまで我が国における活動も含めて、欧米におけるオントロジーに関する研究開発活動を概観してきた。ここでは重要と思われる課題を少し論じてみたい。

(1)オントロジーの標準化

ANSI ad hoc 委員会にオントロジーの標準化を行う委員会ができており、すでに2年ほど活動が続けられている（URL:http://ksl-web.stanford.edu/onto-std/）。実現するかどうかは別として、Cyc、WordNet、 EDRなどの知識ベースや機械可読辞書の Upper modelを取り出してそれらの共通性、相違点を洗い出して概念間の対応をとりながらReference Ontology（RO）と呼べるようなものを作るべく努力を重ねている。今のところ、相違点の方が共通性を圧倒しておりROが無事構築される可能性は小さいが、とにかく彼らは果敢にもオントロジーの標準化という困難な問題に取り組んでいることは注目に値する。

標準化に関わるもう一つの動きはDublin CoreやMeta Content Framework に代表されるInternet上の情報を対象にした Metadata 記述に関する標準化活動である。これらは溝口のいう、インデキシングオントロジーであり、内容の深いところまで関わってこないが、やはり気になる存在ではある。特に XML との関わりは重要であろう。

(2)オントロジーに関する共通の理解

2章において述べたように、未だにオントロジーの定義については研究者の間で合意は得られていない。というより、むしろ"オントロジーとは何かという議論はやめよう"という動きがでてきている状況にある。少なくとも、自然言語処理に携わるもの、知識ベースの立場をとるもの、哲学的な立場をとるものの3者では合意を得ることは困難であるように見受けられる。自然言語研究者は"語彙"あるいは"辞書"という観点を重視し、知識ベース研究者は"実行可能性"や知識ベース構築の"部品"という考えを重視し、哲学者は"共通性、汎用性の高い上位モデル"の立場を固執する。いずれも十分強固な正当性の根拠を持っており議論を収束する方向に持っていくことは至難であるように思われる。

(3) オントロジー工学への道

このような状況ではあるが、オントロジーが重要であるという点では研究者の合意は強固である。厳密な定義にはふれないようにして、各自が必要と考えるオントロジー理論、システム、そしてオントロジー自身を開発する事に専念することで今は十分であるようにも思う。現在はオントロジー研究は幼児期にあり、はじめの一歩で躓かないことの方が重要であろう。

そこで重要なことはオントロジーの定義ではなく、オントロジー工学の旗印の下で何をなすべきかということを明確にすることであると考える。

4.2　オントロジー工学とその意義

オントロジーは「内容指向」研究を支える理論的基盤を提供すると述べた。John Sowaも述べているように、論理はオントロジーなしでは何も具体的に意味のあることは表現できない。その意味で、オントロジーは論理に"命"を吹き込む役割を持っているといえる。論理に関する研究をみれば明らかであるように、これまでの人工知能研究では形式指向の研究が重要視され、内容指向の研究は軽んじられてきた。これにはいくつかの理由があるが、最も重要なことは内容指向の研究が以下のような問題を抱えていることであろう。

(1)個別の状況に依存したアドホックな議論になりがちであること。従って、
(2) 形式理論の様に積み重ねが効くような方法論がないこと。
(3)しっかりした基盤研究、あるいは技術がないこと

この様な問題を解決するのが、オントロジー工学である。オントロジー工学は、知識ベース構築の元になる対象世界の概念化、基本概念の意味の厳密な定義などを与えるだけでなく、実世界の情報モデル構築のために不可欠な、知識を「積み上げる」技術と理論を提供する。

以下にオントロジーの明示的な記述で得られるメリット、即ちオントロジー工学の意義を明確にする。

(1)究極の目的

オントロジー工学の究極の目的は「実世界、厳密には情報科学が対象とし得る全ての対象のモデル構築の基盤を与えること」にある。そしてそれは共有／再利用が可能で、人間とコンピュータの双方に理解可能でなければならない。

(2) Design Rationaleとしてのオントロジー

設計におけるDesign rationale（設計意図：以降、DRと呼ぶ）と同様に、知識ベースのオントロジーは知識ベース再利用において重要な働きをする。他人が構築した知識ベースを理解するには、前提とされている条件や環境、解くべき問題が要求する仮定などの暗黙的な情報、そしてそれらを反映した対象の世界の概念化に関わる根本的な情報を知ることが必要である。しかしながら、設計と同様、これまで実に多くのシステム（知識ベース）が構築されてきたにもかかわらず、その様な情報が明示的に示されることはほとんどなかった。オントロジーはそのようなDR情報を提供し、知識ベー

スの構築を支えるバックボーンとしての役割を持っている。今後の知識処理は明示的に示されたオントロジーに基づいて行われることが望ましい。

(3) 知識を積み上げる

設計意図や暗黙の仮定や基本的な概念構造などを明示的に記述し、意味を共有可能にすることによって、知識を「積み上げる」ことを可能にするような理論を提供する。一口でいえば、オントロジー工学は内容に関する理論を提供する。

4.3　オントロジー工学の領域

この様なオントロジー工学ではどのような研究が行われなければならないのであろうか？　以下にオントロジー工学がカバーすべきと思われる研究領域を示す。

基幹研究

・哲学（存在論）

アリストテレス以来の哲学者が論じてきたOntologyの議論と基礎理論の提供。

・科学哲学

物理の世界における時間、空間、因果、プロセス等の考察と物理学の視点からの貢献。

・知識表現

知識表現に関する基礎的な研究とオントロジー表現のための言語の設計開発。更に、計算機科学の立場からのオントロジーに対する考え方、例えば人間とコンピュータとの意味の共有、手続き的、宣言的表現等に関する基礎的検討。

オントロジー設計

・一般オントロジー

時間、空間、プロセス、因果関係、部分／全体関系、上位／下位などの普遍性の高い基礎概念に関するオントロジーの研究。

意味の内容に関する深い検討と共に、厳密な公理化の問題の検討。

・タスクオントロジー

人間の問題解決過程を記述するためのオントロジーの開発。問題解決のコンテキストをドメイン独立に規定する役割を持ち、ドメインオントロジーの再利用性の向上に貢献する。

・T-ドメインオントロジー

タスクオントロジーが決定するコンテキストの下で考察したドメインオントロジーの開発。

タスクオントロジーとドメインオントロジーとを接続する機能を持つ。

・ドメインオントロジー

プラント（鉄鋼、化学）、電力、土木、建築、企業活動などのドメイン毎のオントロジー開発。現実の問題との接点としての機能を果たす。

常識ベース研究

一般オントロジーの開発と並行した、常識の収集と知識ベース化。

オントロジー構築方法論

・方法論

オントロジー開発の方法論の研究

・構築支援環境

オントロジー開発方法論を支援する環境の開発。オントロジー言語との関連も大切である。

評価方式

・オントロジーの能力と有用性等の実問題を通した評価尺度と方式の開発。

標準化問題

・EDIとデータ意味素

完全自動EDIを目指したデータの意味のプリミティブに関する標準化[Hawes 94]。

・基本意味素レポジトリ

知識共有実現のためには避けて通れない、基本意味素の標準化の検討。

・語彙の標準化

・モデル部品の標準化／規格化

知識処理における"ボルトやナット"の開発を目指した、機械系、流体系等のドメインなどにおける対象モデル表現のための機能部品の標準化。

データ／知識変換

・オントロジー比較

同一（類似）対象に関する異なるオントロジーの相互比較を行ない、共通点と相違点の明示的な記述を支援する方式の開発。

・データベース変換

異なる概念スキーマを持つデータベース間でのデータ交換。

・知識ベース変換

異なるオントロジーに基づいて構築された知識ベース間の変換。

知識共有／再利用方式

・再利用方式

タスクオントロジーとドメインオントロジーに基づいた実際に知識を再利用する方式の開発。

・通信プロトコル

協調的問題解決のためのエージェント間の通信を規定するオントロジーの開発。

・協調タスクオントロジー

エージェント間の協調をタスクと見たオントロジーの開発。

・情報統合

オントロジーに基づいて異種の知識／情報を統合する方式の開発。

メディア

・メディアオントロジー

ドキュメント、静止画像、動画像等の各メディアに固有の構造オントロジーの開発。

・知識メディア共通オントロジー

一般オントロジーに基づいた人の振る舞いの基本単位の確立と行動／動作オントロジー、物語オントロジーなどの開発、及び知識メディアに共通するオントロジーの開発。

・メディア統合

オントロジーに基づくメディアの意味表現言語の開発と意味内容を考慮したメディア統合方式の開発。

オントロジー工学はまだ実体もほとんどないといっても過言ではない。また、容易に完成するとも思われない。しかし、知識処理のさらなる発展のためにはなくてはならないものであることは間違いない。本報告書がオントロジーに関する理解に何らかの貢献をして、オントロジー工学の確立へ役立てば幸いである。

参考文献

[溝口97]溝口、池田：オントロジー工学序説、人工知能学会誌、Vol.12, No.4, pp.559-569, 1997.

大規模知識ベースに関する調査研究報告書
－オントロジー工学に関する調査研究　－

発　行　平成１０年３月
発行所　財団法人　日本情報処理開発協会
東京都港区芝公園３丁目５番８号
機械振興会館内
電話（03）3432-9390

09-R 004