問題解決オントロジーの構成と利用に関する研究

1998年1月

瀬田和久

内容梗概

本論文は，筆者が大阪大学大学院工学研究科電子工学専攻博士後期課程在学中に行った問題解決オントロジーの構成と利用に関する研究をまとめたものであり，以下の6章をもって構成されている。

1章は序論であり，問題解決システムの開発に関わる本研究の背景，目的および工学上の意義について述べる。

第2章では，本研究で設定した問題解決オントロジーの基本構成と問題解決オントロジーが定める公理について述べる。

問題解決オントロジーを構成するにあたって本研究では，オントロジーとモデルの関係を重視している。オントロジーは対象をモデル化する際の規約であり，モデルはオントロジーが定める公理によってその整合性が保証される。問題解決過程を表現するモデルには二つのモデルがある。すなわち(I)問題解決過程の記述のモデルと(II)問題解決過程の意味を表すモデルである。本研究では，この二つのモデルに対する規約として，問題解決オントロジーが定める三つの公理((I)統語論的公理,(II)概念レベル公理,(III)語用論的公理)を明らかにした。この公理に基づいて，問題解決過程を記述/実行するにあたっての規約が明確になり，問題解決オントロジーに整合した問題解決モデルの記述/実行が可能になる。

第3章では，問題解決オントロジーを基盤としたシステム，概念レベルプログラミング環境CLEPE(Conceptual LEvel Programming Environment)について述べる。

CLEPEにはオントロジー構築支援環境としての側面と，問題解決モデル記述環境としての側面がある。まず，オントロジーを構築するオントロジーオーサの側面から，オントロジーとモデルの関係について考察しメタモデルとしてのオントロジーの役割を明確にする。そのことによって，オントロジーの整合性を検証する枠組みを実現することが可能になる。そして，問題解決モデル記述環境の側面から，本システムの基本思想について述べ，問題解決オントロジーに基づいて実現される問題解決モデル記述環境について概観する。

第4章では，3章で述べたCLEPEの問題解決モデル記述環境の側面についてより詳細に，問題解決モデルの再構成について述べる。

まず，本研究で対象とする問題解決モデルの構成と，それぞれのモデルが持つ能力を整理する。次に，問題解決オントロジーに基づいて問題解決モデルがいかにして構成されるかということを主にシステムの内部的な観点から説明する。最後に，本研究で開発したプロトタイプシステムの動作について述べる。

第5章では，関連研究と比較し本研究の位置づけを行う。

第6章では，本研究で得られた主な成果をまとめ，今後に残された課題について述べる。

関連発表論文

A. 学会誌掲載論文

[A1] 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎: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, (採録決定).

[A2] 瀬田 和久, 池田 満, 島 輝行, 角所 収, 溝口 理一郎: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, (投稿中).

B. 国際会議発表

[B1] Kazuhisa SETA, Mitsuru IKEDA, Osamu KAKUSHO, Riichiro MIZOGUCHI: Design of a Conceptual Level Programming Environment Based on Task Ontology, Proc. of International Conference on Success and Pitfall of Knowledge-Based Systems in Real-World Applications, pp.11-20, (1996).

[B2] Kazuhisa SETA, Mitsuru IKEDA, Osamu KAKUSHO, Riichiro MIZOGUCHI: Capturing a Conceptual Model for End-User Programming: Task Ontology As a Static-User Model, Proc. of the Sixth International Conference on User Modeling, pp.203-214, (1997).

[B3] Riichiro MIZOGUCHI, Mitsuru IKEDA, Kazuhisa SETA, Johan VANWELKENHUYSEN: Ontology for Modeling the World from Problem Solving Perspectives, Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, (1995).

[B4] Mitsuru IKEDA, Kazuhisa SETA, Osamu KAKUSHO, Riichiro MIZOGUCHI: An Environment for Building Conceptual Model of Problem Solving , Proc. of Pacific Rim Knowledge Acquisition Workshop, pp.210-225, (1996).

[B5] Mitsuru IKEDA, Kazuhisa SETA,Riichiro MIZOGUCHI: Task Ontology Makes It Easier To Use Authoring Tools, Proc of IJCAI-97, pp.342-347, (1997).

C. 研究会発表

[C1] 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:タスクオントロジーに基づく概念レベルプログラミング環境の設計, SIG-KBS-9503-9, pp.51-58, (1995).

[C2] 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:ドメインオントロジーのタスクコンテクストへの統合 -CLEPE におけるタスクドメインオントロジーの表現と利用 -, SIG-J-9601, pp.89-96, (1996).

[C3] 島 輝行, 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:問題解決環境：CLEPE におけるエンドユーザ支援機能, 信学技法 AI97-14, pp.17-24, (1997-07).

D. 全国大会発表

[D1] 瀬田 和久, 池田 満, Johan Vanwelkenhuysen, 角所 収, 溝口 理一郎:タスクオントロジーの表現とその利用 -Ontolingua による記述を通して -, 人工知能学会全国大会(第九回)論文集, pp.379-382, (1995).

[D2] 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:概念レベルプログラミング環境へ向けての基礎的検討, 情報処理学会第 51 回全国大会, pp.3-177-3-178, (1995).

[D3] 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:概念レベルプログラミング環境のためのオントロジー記述言語の開発, 人工知能学会全国大会(第十回)論文集, pp.207-210, (1996).

[D4] 瀬田 和久, 島 輝行, 池田 満, 角所 収, 溝口 理一郎:タスクオントロジーに基づくエンドユーザプログラミング支援機能 - 問題解決過程における因果関係の明示化 -, 情報処理学会第 55 回全国大会, pp1-383-1-384 (1997)

[D5] 島 輝行, 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:概念レベルプログラミング環境 CLEPE における概念レベル実行環境の開発, 人工知能学会全国大会(第十回)論文集, pp.211-213, (1996).

[D6] 島 輝行, 瀬田 和久, 池田 満, 角所 収, 溝口 理一郎:概念レベルプログラミング環境におけるオブジェクトの状態変化のモデル化, 人工知能学会全国大会(第十一回)論文集, pp.249-252, (1997).

目次

図目次

表目次

第1章

序論

技術革新に対する人類のあくなき挑戦が今日の繁栄をもたらしている。その歴史において先人の見いだした創意と工夫の蓄積が，高度技術社会を支える根本的な礎を担っている。この創意と工夫の蓄積/継承といったプロセスはよりミクロな形で，工業製品の開発サイクルに見ることができる。一般に，工業製品の開発は，製品に対するユーザの要求の分析，要求に適合する製品の設計，設計に基づく製品の製造/テスト，メンテナンスといったライフサイクルを繰り返すことによって行われている。そして，それぞれの工程に関与した人々の経験が継承/蓄積され，新しい製品のライフサイクルに活かされている。

もちろん全ての工業製品のライフサイクルでそのような理想的な基盤が整えられているわけでは必ずしもない。しかし，JIS規格に代表されるように，製品を構成する部品の標準化を基礎として合理的なライフサイクルが確立されている領域は少なくない。そのような領域では，標準化規格が提供する用語を用いることでライフサイクルに関わるエージェント間で知識の伝達/フィードバックさらには，知識の継承/蓄積が容易になっている。

一方ソフトウェアの開発の場合には，ライフサイクル全般にわたって各工程に関与するエージェント間で，知識の伝達/フィードバックをするための基盤が充分に整備されていないのが現状である。このため，同様のライフサイクルを持つ工業製品の開発の場合と比べて，開発に関与するエージェント間での意志疎通が弱くソフトウェアの品質がそれぞれの工程に関与する人の能力に強く依存してしまっている。

という問題がある。ソフトウェア工学の分野ではOMT(Object Modeling Technique)に代表されるようなアプローチでこの問題を解決しようという取り組みがなされている。しかし，ソフトウェアの特徴として部品の概念に明確な定義を与えることが難しいため，製品化のライフサイクルをきめ細かく支えるまでには至っていない。

この問題の解決に必要なこととして，人間を系に含むということを真摯に受けとめ，その認識の主体が認知する概念的な部品を体系的に捉えることが本質的である。その基礎を支えるのが「オントロジー」である。オントロジーは，

- ライフサイクル全般にわたって意志疎通を図るための概念の体系を提供する
- ソフトウェアの設計意図を明確にする
- 対象世界をモデル化するための基盤を与える

といった役割を担っている。この概念体系を用いて計算機内のモデルを操作することも可能になる。

本研究の目的は，

- 問題解決システムを作成するライフサイクルを支える概念として問題解決オントロジーの働きを明確にし，
- 問題解決モデルの作成を支援する環境を実現するための技術的問題を明らかにし，解を与えることにある。

現状の知識ベースの問題解決システムでは，一般のソフトウェアの開発工程の場合と同様，システムに対する様々な要求と，それを満たすために問題解決システムに格納されている個々の知識の間の関係，すなわち設計意図が明確になっていないという問題があり，また，計算機内部に格納する対象世界のモデルが現実対象の構成と必ずしも一致しないため，現実世界とシステムの内部的な世界の間にギャップが生じてしまっている。このために，要求分析，設計，コーディングに関与するエージェントの間で概念ギャップが埋められず，知識ベースの問題解決システムの開発サイクル全般にわたっての意志疎通が難しいものとなっている。

これに対して，対象世界におけるオントロジーを事前に準備し，オントロジーに基づいて問題解決システムを構築することで，問題解決システムの設計原理が明確になり，また，現実世界とシステムの内部的な世界の間のギャップが相対的に減少することが期待されるため，問題解決システムの開発サイクル全般にわたって各工程に関与する人々の間での意志疎通を円滑に行うことが可能になると考えられる。

本研究では，人間が行っている知的な問題解決をモデル化するための基盤としての問題解決オントロジーを研究の対象とし，

- 問題解決オントロジーの構成原理を明らかにする。さらに，
- 問題解決オントロジーに基づいてエンドユーザによる問題解決システムの開発サイクルを支える概念レベルプログラミング環境CLEPEを提案する。

CLEPEは問題解決オントロジーの構築から利用にいたる過程を支援するシステムであり，

● 問題解決オントロジー構築支援環境

● 問題解決モデル記述環境

の二つの側面を持っている。前者が問題解決システムの設計，開発/テストのための基礎となるオントロジーを開発するためのシステムであり，後者がそのオントロジーの上で問題解決システムの設計，開発/テストを支援するシステムである。問題解決システムを設計，開発/テストするための基盤を整えるフェーズと，その基盤の上で問題解決システムを設計，開発/テストするフェーズを明確に分離し，それぞれのフェーズにおける作業を支援するための環境を提供している点が本研究の特徴である。

以下，第2章では，本研究で扱う問題解決オントロジーについての理解を深めるために，まずオントロジー一般に関する議論を行い，知識工学においてオントロジーが担う役割を整理する。計算機内部に格納する問題解決過程のモデルと，人間の頭の中にある問題解決過程に関する知識を対応づけるためには，

- 人間の頭の中にある問題解決の過程を，表現としてモデル化するための規約を明確にすること。
- 表現から，人間が意図する問題解決の過程をモデル化するための規約を明確にすること。

が重要である。このような要求を満足する問題解決オントロジーの基本構成について述べる。そして，問題解決オントロジーが定めるモデル化の規約を，本研究で作成した問題解決オントロジー記述言語TOLを用いて例示する。

第3章では，CLEPEの全体像について述べる。

CLEPEの二つの側面の内，問題解決オントロジー構築支援環境では，問題解決オントロジーを構築するためのモデル化の基盤が与えられ，対象世界の複雑な概念構造を捉え，矛盾のない問題解決オントロジーを構築することが可能になっている。問題解決モデル

記述環境では，問題解決過程をモデル化するための基盤が与えられ，このモデル化の基盤に基づいて問題解決モデルを記述するための環境が実現されている。エンドユーザは，CLEPEが提供する様々な支援を受けながら，問題解決モデルを容易に記述することが可能になっている。

第4章では，エンドユーザが記述した問題解決モデルを再構成する枠組みについて述べる。エンドユーザが行う問題解決モデルのモデル化を支援するためには，エンドユーザが記述した問題解決モデルが自分が意図したものであるかどうかを確認することができるような枠組みが不可欠である。CLEPEは，エンドユーザが記述した問題解決モデルから，エンドユーザが意図した問題解決の過程を問題解決オントロジーに基づいて読みとり，その過程を表現する概念レベル実行モデルを構成することができる。本研究では，CLEPEによるこの作業を「問題解決モデルの再構成」と呼んでいる。計算機の内部的なモデルである概念レベル実行モデルは，問題解決オントロジーに基づいてそれを表現する対象世界の語彙と対応づけられるおかげで，エンドユーザは対象(ドメイン)の言葉によって，このモデルの振る舞いを確認することができる。

第5章では，他の関連研究と比較し本研究を位置づける。

第6章は結論であり，本研究のまとめと今後の展望を示す。

第２章

問題解決オントロジー

2.1 緒言

オントロジーは本来,「世界を構成する物や概念の間の関係を体系化する学問」という哲学の分野の存在論を表す言葉である。人工知能の分野では,「概念化の明示的な仕様」という意味で用いられている[Gruber 92]。知識ベースシステムの分野では,「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概念/語彙の体系とその理論」という意味で用いられる[Mizoguchi 93]。

知識工学におけるオントロジー研究の目的は，対象世界に関する原理/原則を明確にし，モデル構築のための基盤を与えることにある[溝口 97]。問題解決オントロジー研究の目的は，人間が行う問題解決に関わる原理/原則を明確にし，問題解決過程をモデル化するための基盤を与えることにある。

このような，モデル化の基盤としてのオントロジーが適切に構成されれば，モデルを利用するエージェント間で対象を捉える概念構成に関する明示的な合意の下でのモデルの共有が可能になると考えられる。

本章では，問題解決に関するモデルの共有を可能にするための問題解決オントロジーの構成について述べる。まず，そのための準備として，オントロジー一般について概観し，オントロジーの定義，分類，知識工学における役割について述べる。次に，モデル化の基盤としてのオントロジーとそれに基づいて構成されるモデルの関係を位置づける。そして，問題解決オントロジーを適切に構成するための基本思想について

述べ，問題解決オントロジーの基本構成を明らかにする。そして，問題解決をモデル化するために問題解決オントロジーが定める公理を明らかにし，これを例示する。

2.2 オントロジー

2.2.1 オントロジーの定義

オントロジーと類似する概念であるTaxonomy，TerminologyとVocabularyに関する研究と比較しながらオントロジーを定義する。そしてオントロジー記述の重要な概念である公理と準公理をとりあげて相互の違いを明確にする[溝口 97]。

Terminology（用語論）

対象世界から概念を切り出した後で，その概念に付与すべきラベルを決定する必要がある。Terminologyは，概念に対するラベルを何にするかを議論することが中心的な課題である。

Vocabulary (語彙)

Vocabularyは 自然言語処理などで用いられる処理が対象とする単語の集合を指す。必然的に単語は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。しかしながら，語彙は特定の自然言語に依存しており，普遍性に欠ける。さらに，語彙（単語）が表象する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とVocabularyの違いは，Terminologyが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ことにある。

Taxonomy(分類学)

Taxonomyは一般には分類全般を指す。これを特化して，ここでは概念分類を指すと考えると，オントロジーに最も近い概念である。概念の分類は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。直感的にはTaxonomyにおいて各概念の意味定義と概念間のさらに詳細な制約を明確に記述したものがオントロジーに対応

する。

Ontology（存在論）

哲学で議論されるオントロジーをそれ以外のオントロジーと区別するために，哲学における存在論を表すオントロジーを，初めの"o"を大文字"O"にした"Ontology"という英語で表すことにする[Guarino 95]．Ontologyは，存在とは何か，存在を説明するために必要な概念は何か，そしてその概念はいかにして存在を体系的に説明できるか，などの質問に応えるべき理論である。哲学者はそれぞれ独自のOntologyを提案してきた。

ontology（オントロジー）

基本的な方法論は哲学のOntologyに従うが，存在一般という基本的なものだけを考察の対象とするのではなく，人工物を含めた具体的なもの，例えば熱力学，企業活動，問題解決構造などを考察の対象とする。そしてその結果として，そこに現われる概念と関係を明示的に示し，明確な意味定義を与えるものである。従って，原理的にオントロジーは考察対象の数だけ存在するため，英語では"ontologies"のように複数形を取りえる。必ずしも論理を使った定式化を行う必要はない．

Formal ontology（形式的オントロジー）

オントロジーの記述を述語論理などを用いて公理化したもの。公理化することによって，オントロジーが持つ能力に関する質問に答えることができる．

Axiom（公理）

宣言的かつ厳密に表現された知識でその正しさを証明なしで受け入れるべきもの。論理の世界では推論の前提となる知識を表すので本質的な意味を持っている。一般に，「公理」と言う言葉は定理を導くための数少ない法則という意味合いが強い。しかし，オントロジー研究において用いる「公理」という言葉には，定理に対して相対的に上位であるという意味合いに対する意識が強い。数学的な公理とは異なり，広い範囲ではたらく少数の公理で基盤を整えるのは難しく，様々な問題に対してどのくらいの公理が必要なのかを明らかにしながら積み上げていくことが重要である。

オントロジー記述における公理には次の二つの役割がある。

(1) 語彙／概念の意味定義を厳密に表現すること。

(2) (宣言的に書かれた知識の範囲で) オントロジーに含まれる語彙／概念を使って記述されたものがもつ能力。すなわち, オントロジーの能力に関する質問に対して解答すること。

オントロジーの能力に関する質問は, オントロジーの評価に関して重要な役割を担っている。これには, 大きく分けて次の二つに分類することができる.

(1) オントロジーを用いて記述できるもの全体に関する抽象的な性質に関する質問。

(2) オントロジーを用いて記述したものが実行された際の振る舞いに関する質問。

(1)はCompetenceに関する質問[Gruninger 94]であり, (2)はPerformanceに関する質問である。(1)に関しては述語論理などで形式化された公理を用いた記述が適している。しかし, (2)に関しては, 振る舞いを生み出す手続き的なプログラム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, 必要となる概念が次の準公理である。

Axiom equivalent（準公理）

述語論理の様な完全で普遍性のある証明系を前提として形式化された公理ではなく, Performanceに関する質問に答えるために必要な手続き的なエンジンの存在を前提として, 本質的な情報は宣言的な公理として記述したもの。Performanceに関する質問には人間が解釈することによって答えを推論することができれば良しとし, 必ずしも形式化されている必要はない。

2.2.2 オントロジーの分類

本研究では, 溝口の定義に習ってオントロジーを「知識ベースを構成する基礎概念/部品についての体系的記述」という意味で用いる。ここでは, 知識ベースを構成する基礎概念(内容)を整理する試みであるContentオントロジーに関する研究を分類する。

Contentオントロジーは大きく, 利用の範囲を限定しないオントロジーと, 利用の範囲を特定の問題解決に限定したオントロジーがある[溝口 97][Heust 97]。それらはさらに,

(I) 実体やプロセス, 時間などのいわゆる「常識」と呼ばれる一般的な概念を定義

するオントロジー [Lenat 90][Sowa 95]。

(II) 問題解決などが対象とする領域(ドメイン)に関する概念を明らかにするオントロジー [Gruber 94][Gruninger 95][来村 97][笹島 96][高岡 95]。

(III) 問題解決が行われる環境が問題解決に与える影響を規定するオントロジー [Vanwelkenhuysen 95]。

(IV) 問題解決の構造をドメイン独立に記述するオントロジー[Hori 94][Ikeda 97][Steels 90][Wielinga 93]

に分けられる。主に(I)(II)のオントロジーが利用方法を限定しないオントロジーでありそれぞれ,(I)一般/共通オントロジー,(II)ドメインオントロジーと呼ばれる。(III)(IV)のオントロジーは利用の範囲を特定の問題解決に限定するオントロジーであり,それぞれ(III)Workplace オントロジー,(IV)問題解決オントロジーと呼ばれる。既に述べたように,本研究ではこのうち問題解決オントロジーを対象として議論を行う。

2.2.3 オントロジーの役割

知識工学におけるオントロジーの役割は大きく以下の三つにまとめられる。

(1) モデル化の規約としてのオントロジー

現実世界での現象,行為を計算機で捉えるためには,対象をモデルとして計算機内部に格納しなくてはならない。オントロジーは,対象をモデル化する際に必要となる概念や,概念間の関係を明示的に規定する。オントロジーが提供する概念と制約の下で,オントロジーに整合した「モデル」が構成される。この関係を,オブジェクト指向言語とのアナロジーで言えば,モデルオーサ(プログラマ)に対してあらかじめ用意された組み込みのクラス定義がオントロジーに対応し,そのクラスを利用して作るインスタンスがモデルに対応する。このようなオントロジーとモデルの関係を明らかにし,オントロジーを計算機可読な形で定義しておくことによって,「モデル」のオントロジーに対する整合性を検証することが可能になる。

本研究では、オントロジーが定める規約に従ってオーサが記述した問題解決知識をモデルと呼び,モデルが「オントロジーに従っている」あるいは「整合している」といった表現を用いることにする。

(2) 設計意図としてのオントロジー

オントロジーの役割の一つとして忘れてならないのが,設計意図(DR:Design

Rationale)の明示化である。つまり，従来暗黙的であった概念構造を明示化することによって，システム設計者が持っている対象世界に対する理解，システム設計の意図が明確になるということである。

一般に，設計に関する行為は，数多くの決定プロセスの集積であり，その全てに設計者の意図と合理的な理由が存在している。知識ベースにおけるオントロジーは，知識ベース構築の決定プロセスに関するDRを明確にする働きを担っている。他人が設計/構築した知識ベースを理解するためには，前提としている条件や環境，解くべき問題が要求する仮定などの暗黙的な情報，さらにそれらを反映した対象世界の概念化に関わる根本的な情報を知ることが重要である。しかしながら，これまで数多くの知識ベースが設計/構築されてきたにも関わらず，その様な情報が明示的に示される事はほとんどなかった。オントロジーはその様なDR情報を明確にし，知識ベース構築を支えるバックボーンとしての役割を担っている。

(3) 標準化

オントロジーのもう一つの役割は，標準化にある。標準化の問題は知識処理の高度化，知識ベース構築の効率化を考える際には避けて通ることのできない問題である。知識ベース構築に必要な基本部品の標準化は知識ベースの生産性の高度化に貢献する。これは，工業製品の生産性の高さを見れば明らかである。ボルトとナットに対応するような規格品を知識処理の分野においても作る必要がある。オントロジーは，標準概念の意味を規定するものであり，知識の内容を厳密な形式を通して積み上げていくことで知識の標準化に貢献する。

2.3 問題解決オントロジーの基本構成

本研究では，問題解決オントロジーを「人間が行っている問題解決に関する操作や操作対象の物の間の関係の体系的記述」という意味で用いる。直感的に，欧州の知識コミュニティで使われている(タスク)モデルに，米国で使われているメソッドとタスクを総称したものに，本研究の問題解決オントロジーが対応している。

ここでは，2.2で行ったオントロジー一般に関する議論を踏まえた上で，問題解決オントロジーの基本構成について考察する。

2.3.1 オントロジーとモデルの関係

問題解決オントロジーの研究の目的を端的に言えば，問題解決知識のオーサと、その知識を利用する計算機システムの間のコミュニケーションをより円滑にすることにある。問題解決オントロジーはそのための基礎として「問題解決オントロジーに整合した問題解決モデル」を規定している。

厳密には，モデルとオントロジーの関係は絶対的ではなく、オーサがモデル化する対象に依存する。これを，オブジェクト指向とのアナロジーで説明すると，例えばクラスを定義するオーサにとっては、そのクラス定義自体がモデルに対応し，クラス定義に際して参照するメタクラスがオントロジーに対応する。

モデル化の規約として適切なオントロジーを設定するためには，モデルオーサの視点とオントロジーの利用目的を注意深く検討しなければならない。すなわち，(モデルオーサを含む)どのようなエージェントが存在し，それぞれのエージェントはオントロジーを使ってどのようなモデルを操作し，どういった目的を達成したいのか？その目的を達成するための規約として何が必要か？ということが重要となる。

このことを本研究で開発した概念レベルプログラミング環境 CLEPE(Conceptual LevelProgramming Environment)の構成(図 2.1)と対応づけて説明すると以下のようになる。CLEPEはオントロジーの構築から利用にいたる過程を支援することを目的としており，大きく分けて

(1) 図 2.1 の右側面の問題解決モデル記述環境と

(2) 図 2.1 の上面のオントロジー構築支援環境

の二つの側面を持っている。

(1)問題解決モデル記述環境では，モデルオーサ(エンドユーザ)とシステム(モデル処理系)がエージェントとして存在する。計算機に馴染みのないエンドユーザは，(問題解決オントロジーとして定義された)日頃使っている平易な言葉を使って問題解決知識(以降，問題解決モデルと呼ぶ)を記述して、実行内容を確認する。また、システムは、エンドユーザが記述した問題解決モデルをオントロジーを参照して実行する(表 2.1)。

人間が行う様々な問題解決の内，この環境では，専門家が日常的に行っているスケジューリングや診断などの処理の手順を中心的に考える問題解決を対象としている。

(2)オントロジー構築支援環境では、オントロジーオーサ(オントロジーを構築する

オーサ)とシステム(言語処理系)がエージェントとして存在する。オントロジーオーサは、エンドユーザが作成する問題解決モデルに対して規約としてはたらく問題解決オントロジーを構築することが目的である。また，システムはオントロジーオーサの作業をガイドし，構築されたオントロジーに矛盾がないかどうか(整合性)を検証する。二つのエージェントが協調して整合したオントロジーを構築するために，オントロジーオーサとシステムが参照する問題解決オントロジー(メタクラス)が必要である。本研究ではこれをコア問題解決オントロジーと呼んでいる(表2.1)。

このように，(1)問題解決システムを設計，開発/テストするための基盤を整えるフェーズと，その基盤の上で問題解決システムを設計，開発/テストするフェーズを明確に分離したこと。(2)それぞれのフェーズに関わるエージェントと各エージェントの目的を明らかにしたこと。そして，(3)各エージェントの作業を支援するための環境を提供している点が本研究の特徴である。

2.3.2 問題解決オントロジーの基本構成

本研究の主題である問題解決オントロジーの構成を考えるにあたっては，以下に示す要求を満足するような問題解決オントロジーの構成を明らかにする必要がある。

I. 問題解決モデルを容易に記述できること。

II. 問題解決モデルの計算論的な意味が明確になること。

III. スケジューリングや診断などのタスクタイプ固有性に対応できること。

IV. 問題解決全般に広く適用できること。

図2.2に(I)～(IV)の要求を満足する問題解決オントロジーの基本構成を示している。図には，(I)(II)の要求を満足するために，奥行き方向(a軸)にレキシカルレベルオントロジー(TO/K-L)と概念レベルオントロジー(TO/K-C)を配置し，(III)(IV)の要求を満足するために，垂直方向(b軸)にタスクタイプ固有(Task-S)オントロジーとコア問題解決(C-Task)オントロジーを配置している。

a軸のレキシカルレベルオントロジーは(I)の要求を満足するために，問題解決モデルを記述する際に使う単語の品詞的な性質を定めており，概念レベルオントロジーは(II)の要求を満足するために，それらの単語の意味を定めている。また，b軸のタスクタイプ固有(Task-S)オントロジーは(III)の要求を満足するために，スケジューリングや診断などのタスクタイプ固有の概念を規定し，コア問題解決(C-Task)オントロジーは問

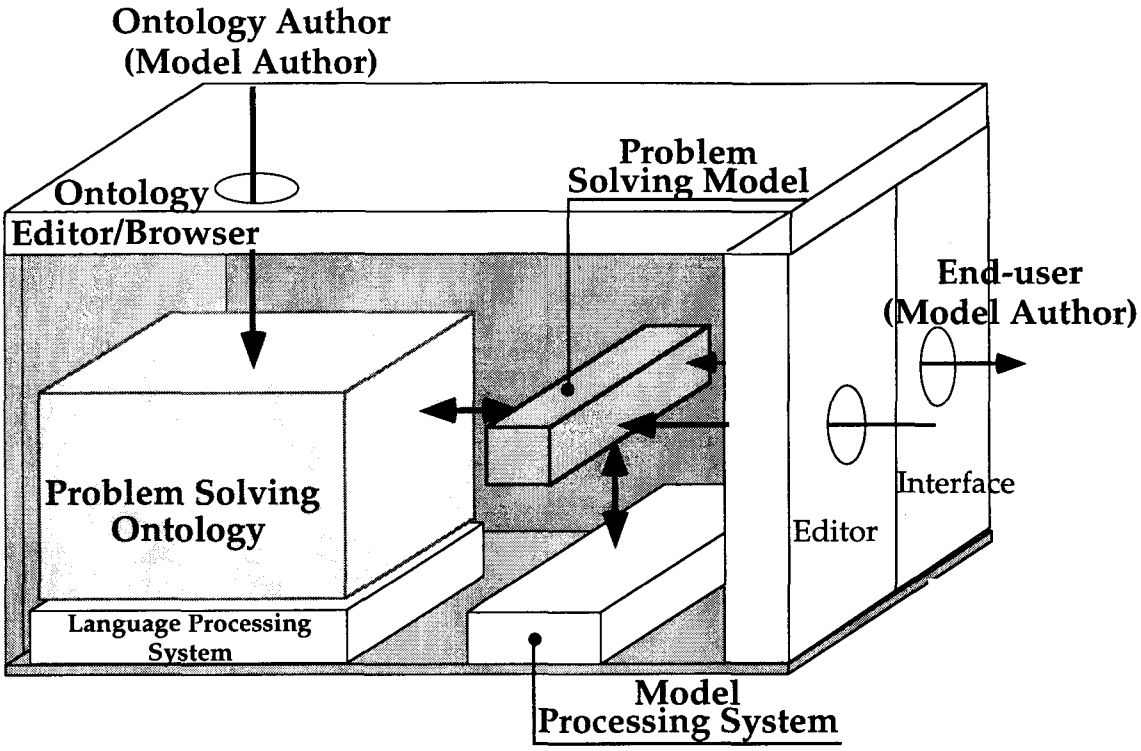


図 2.1: 概念レベルプログラミング環境 CLEPE の全体像

表 2.1: CLEPE におけるオントロジー，モデル，エージェントの関係

	Agent	Ontology	Model	Objective
Environment for describing problem solving model	End-user (Model author)	Problem solving ontology (Class)	Problem solving model	To describe a problem solving model based on problem solving ontology
	Computer			To execute a problem solving model based on problem solving ontology
Environment for building problem solving ontology	Ontology author (Model author)	Core problem solving ontology (Meta class)	Problem solving ontology (Class)	To build a problem solving ontology which plays a role as a specification of problem solving model based on core problem solving ontology
	Computer			To check the consistency of problem solving ontology based on core problem solving ontology

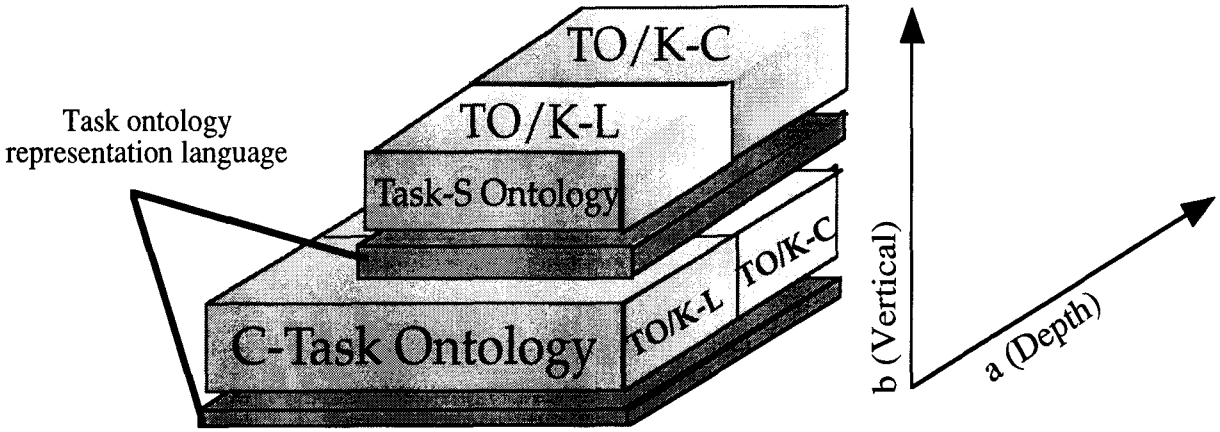


図 2.2: 問題解決オントロジーの基本構成

題解決に一般の概念を捉えている。

2.4節において問題解決オントロジーの奥行き方向(a軸)の構成について，主にタスクタイプ固有オントロジーを例にして述べる。垂直方向の構成と言語処理系の詳細については，3.2節においてオントロジー構築支援環境を例にして述べる。

2.3.3 問題解決モデルの具体例

2.3.2節で述べた要求I, IIの間には，人間にとっての記述の容易性を尊重しようとすると，その記述の意味が曖昧になり，計算機的なオペレーショナリティが明確にならないというジレンマがある。

本研究では，この要求を両立するために，問題解決モデルを，自然言語の記述に近いレキシカルレベルモデルと，そのモデルの意味内容を表す概念レベルモデルに分けている。通常のプログラミング環境との直感的なアナロジーをとれば，レキシカルレベルモデルがソースに相当し，概念レベルモデルがそのターゲット (内部モデル)に相当する。

---

[1] 内部表現を模式的に図示している。

問題解決モデルの例として，図2.3に看護婦の負荷が均一になるようにジョブを割り当てる24時間要員配置問題に対する問題解決モデルを示している。図の左側がレキシカルレベルモデルであり，右側が対応する概念レベルモデル[1]を表している。

モデルの作成にあたって，エンドユーザは処理概念を名詞と動詞の組み合わせ(以降，汎化プロセスと呼ぶ)を基本として表現し，処理順序を有向リンクで結ぶことによって問題解決のフロー(レキシカルレベルモデル)を作成する。直感的に，動詞，名詞に当てはまるターム[2]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。

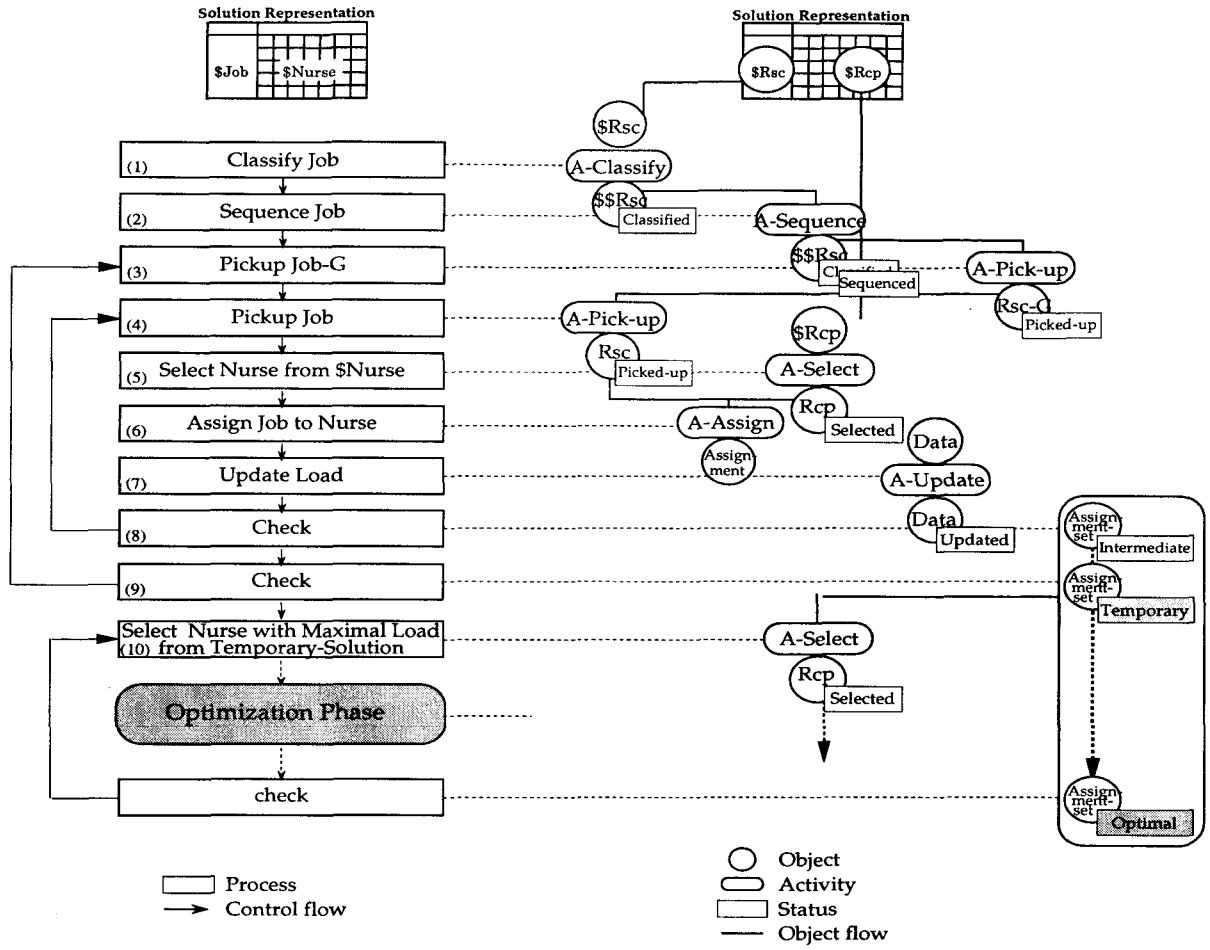
図2.3に示したレキシカルレベルモデルが表している処理の大まかな流れは，まず，(日勤，夜勤などの)勤務単位グループ毎に看護婦に対して勤務を割り当て(図の上部の二重ループの内側のループ)，全ての勤務単位グループについて同じ処理を繰り返す(図の上部の二重ループの外側のループ)。そして，看護婦の負荷が均一になるように割り当てを交換する最適化フェーズ(図の下部のループ)からなっている。

レキシカルレベルモデルの意味内容は対応する概念レベルモデルで明示的に表現される。加えてレキシカルレベルモデルでは表現されていない、オブジェクトフローが明確になっている。

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。それぞれは基本的に，レキシカルレベルモデルの動詞，名詞，形容詞に対応している。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，概念レベルオントロジーで定義される(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. Selected)によって，明らかにされる。例えば，図のassignアクティビティの働きについて，「Assignアクティビティはループの周回の度に，取り出したジョブ(Picked-up Job)と，そのジョブに対して選択した看護婦(Selected Nurse)を割り付け(assign)，一つの割り付け(assignment)を生成し，解(assignment-set)を更新する。」といったことが内部表現として表されている。

---

[2]2.3.4節で後述するように，レキシカルレベルオントロジーでは。jobやnurseなどの対象に固有のタームは定義されていない。タスクに固有の概念のみ定義されている。jobやnurseなどのタームは実際には，2.5節で述べるタスクドメインオントロジーによって，タスク固有の概念の元で組織化される。



(a) A lexical level model (description of problem solving model)

(b) A conceptual level execution model

図2.3:レキシカルレベルモデルと概念レベルモデル(最適化フェーズを省略)

2.3.4 問題解決モデルに対する規約としての問題解決オントロジー

図2.3に示された二つのモデルを比較して注目すべき点は，レキシカルレベルモデルにおいて明らかにされていないオブジェクトの同一性が概念レベルモデルにおいて明示的に表されている事である。

例えば，Pickupプロセスにおいて「取り出されたジョブ」と「Assignプロセスの入力となるジョブ」は，レキシカルレベルモデルにおいては単語としてエンティティが異なっているが，概念レベルモデルとしては同一のオブジェクトであり，同一の「ジョブ」として扱うようになっている。また，図の上部の二重ループの内側のループの出力である，「部分解」と外側のループの出力である「暫定解」，そして，図の問題解決知識全体の出力である「最適解」は，同一のオブジェクトであり，問題解決の過程を通じてその状態が変化していくものと考えられる。このように、レキシカルレベルモデルに対応する概念レベルモデルでは，実行モデルのセマンティクスを定義するために不可欠なオブジェクトフローが明示的に表現されている。

この二つの知識レベルのモデル(レキシカルレベルモデルと概念レベルモデル)を総括するオントロジーを知識レベルオントロジー(TO/K)とよんでおり，各々のモデルに対するオントロジーを(知識レベルの)レキシカルレベルオントロジー(TO/K-L)と(知識レベルの)概念レベルオントロジー(TO/K-C)とよんでいる。

レキシカルレベルオントロジーと概念レベルオントロジーの特性は語彙の階層構造を考えるとわかりやすい。図2.4と図2.5にレキシカルレベルオントロジーと概念レベルオントロジーの階層図を表している。TO/K-Lはモデルオーサ(エンドユーザ)が直接利用するものであり，図2.4に見られるように、単語の品詞を最上位においたオントロジーである。単語の統語的な役割を定義する最下位の階層にはモデル構築者が日頃使っている対象の世界の単語が後述するT-ドメインオントロジー(図2.4のシェイディングされた部分)によって組織化されており，全体として単語の問題解決の観点から見た意味的階層が表現される。一方、TO/K-Cは統語的な制約から離れ、レキシカルレベルモデルの意味内容を表現するためのタスクタイプに固有の概念の構造を組織化している。

タスクタイプに固有の概念と問題解決が対象とする領域固有の概念を明確に分離することは，問題解決モデルの再利用性を高める上で重要である。例えば，スケジュー

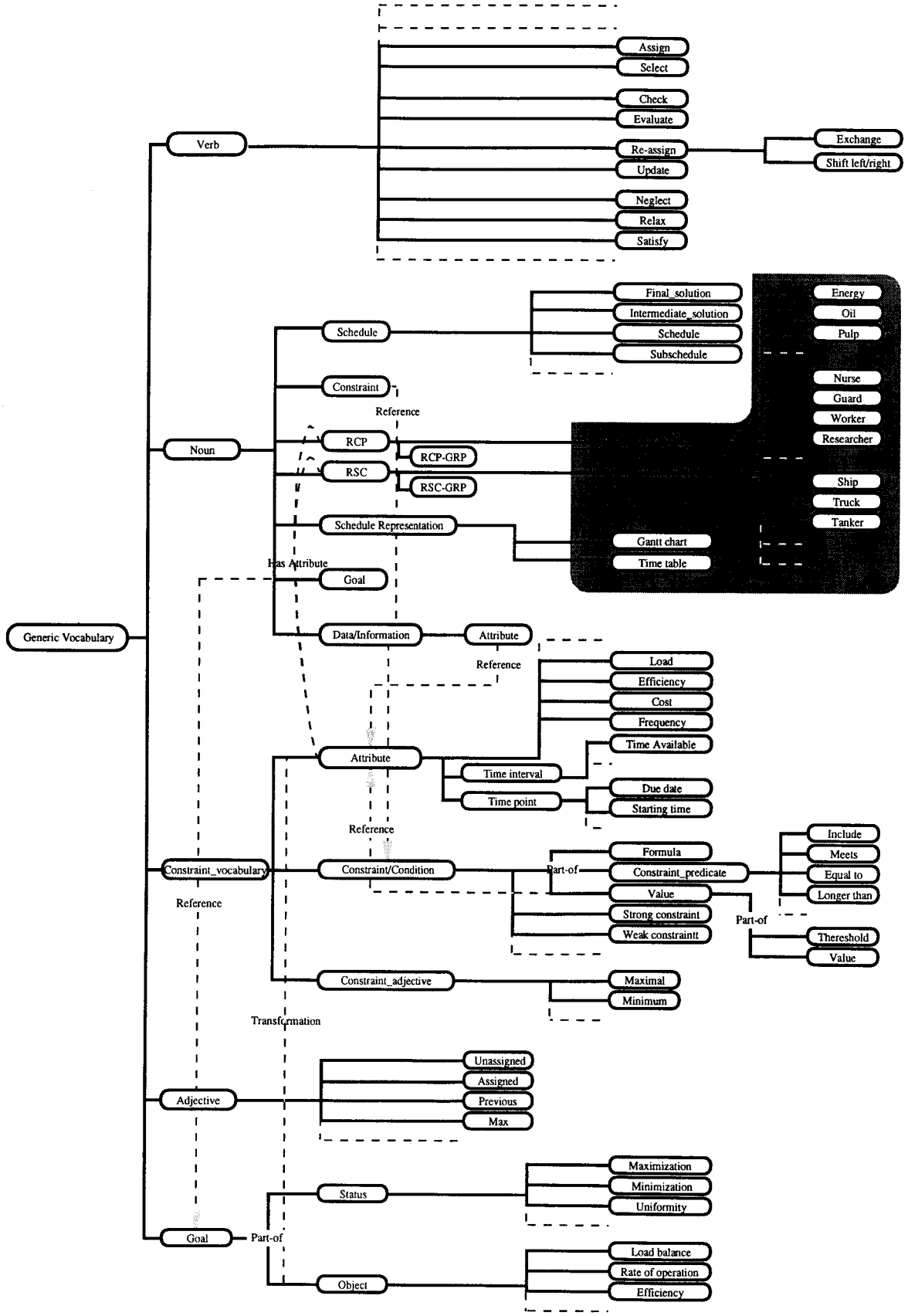


図2.4:レキシカルレベルオントロジー

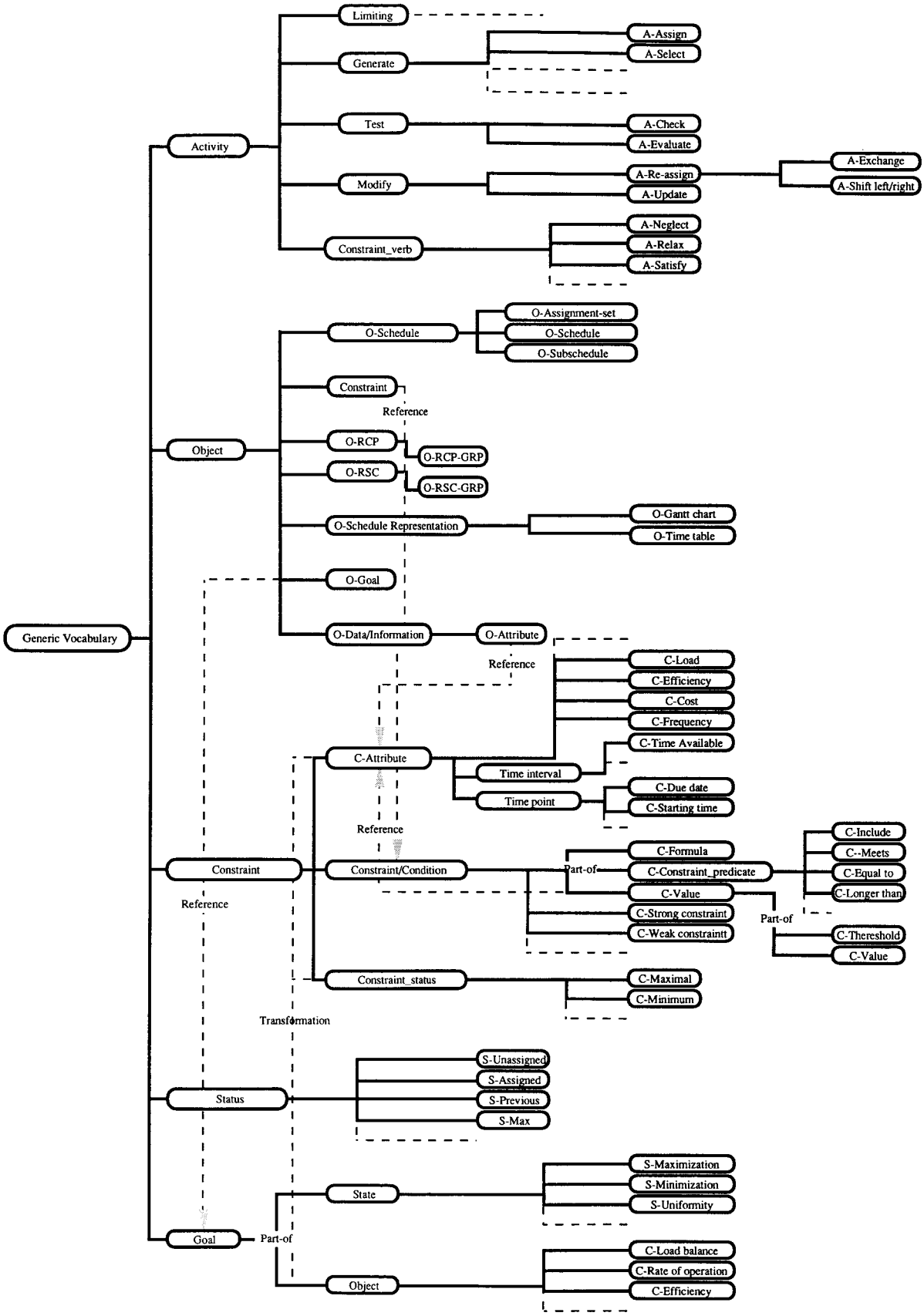


図 2.5:概念レベルオントロジー

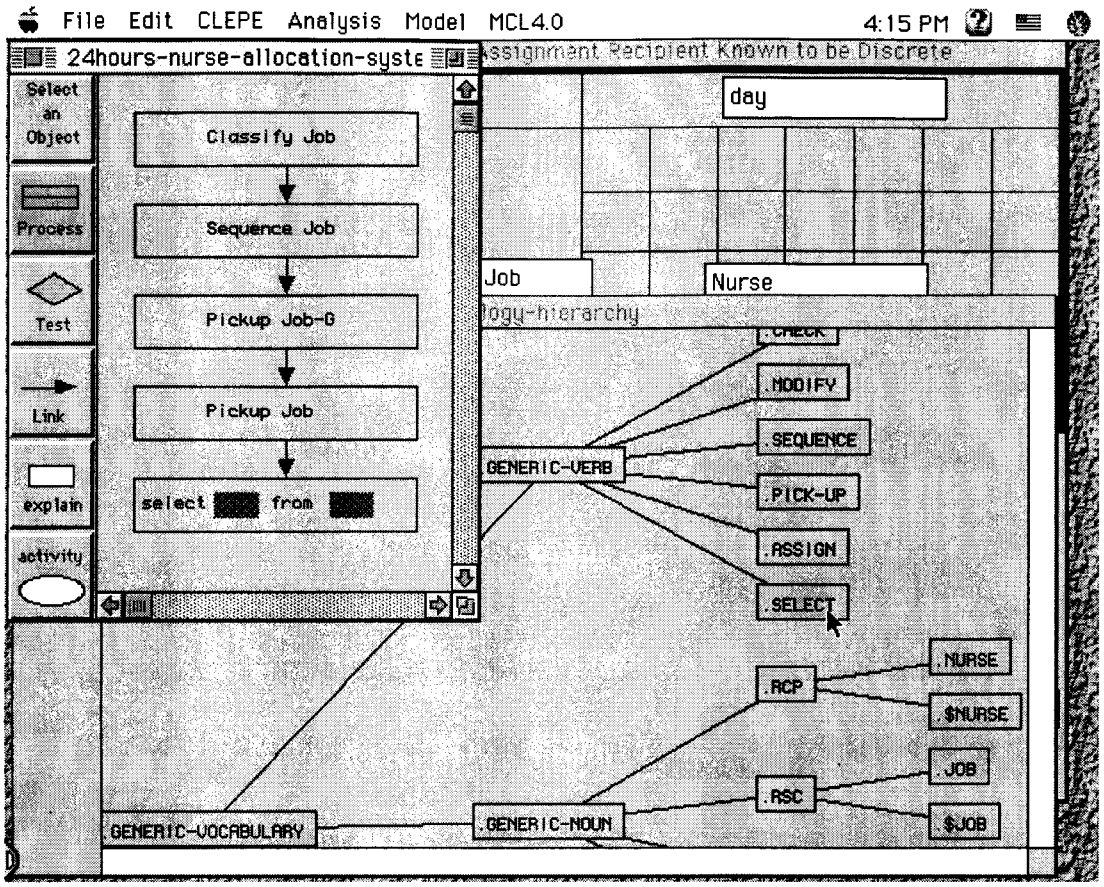


図 2.6:CLEPE におけるワークスペース

```
(Define-Task-S-Verb Select (?select)
  :class-hierarchy (subclass-of select generic-verb)
  :slot-def (
    (in-noun ?In :constraints ((class noun)))
    (out-noun ?Out :constraints ((class noun)))
    (reference-noun ?r-obj :constraints ((class noun)))
    (dk ?dk :constraints ((class constraint)))
    (syntax ?syntax :@constraints ((member ?syntax
                    (or (select ?Out)
                        (select from ?In)
                        (select ?Out from ?In))))
    (cor-activity ?a-select :@constraints
                            ((class A-Select)))
  :axiom (
    (membership (:extensional))
    ))
```

図 2.7:動詞「Select」の定義

リングタスクタイプにおいて「看護婦」や「ガードマン」といった対象世界に固有の概念の間には，問題解決の観点からは意味の違いに重要性はなく，むしろ共通の意味としてスケジュールの受け手(Rcp)であることが重要である。この共通性を捉え，問題解決モデルの再利用性を高めるためには，領域固有の概念を用いて記述された問題解決モデルから，領域固有の概念を捨象し，このような枠組みを実現するための鍵となる考え方が，T-ドメインオントロジーの概念である。本研究で構築した問題解決オントロジーは，再利用性を高めるために領域に固有の概念が問題解決における役割に沿って問題解決に一般的な概念へと汎化した形で体系化している。これを，本研究では汎化語彙(レキシカルレベルオントロジー)，汎化概念(概念レベルオントロジー)と呼んでいる。図2.4の階層図にあるように，T-ドメインオントロジーは，対象世界に固有の概念をタスクタイプに固有の概念の下で位置づける概念である。これは，対象世界の概念を用いて記述された問題解決モデルから，タスクタイプに固有の意味を取り出す働きを担っているが，詳細については2.5節で述べる。

2.4 問題解決オントロジーの構成 - モデルを支配する三つの公理 -

2.2節で述べたオントロジーの定義に従って本研究では，オントロジーが規定する，モデルが満たすべき制約を「公理」とよぶことにする。ここではオントロジーが定める公理の内容をタスクタイプ固有オントロジー(図2.2b軸の上側)を例に取り，本研究で開発した問題解決オントロジー記述言語TOL(Task Ontology representation Language)を用いてそれを例示する。

2.4.1 レキシカルレベルオントロジー:TO/K-L- 統語論的公理 -

CLEPEにおいてユーザから見たレキシカルレベルモデルの実体は図2.6に示すワークスペースである。

ワークスペースはタスクビューとGPN(Generic Process Network[ティヘリノ 93])ビューからなっている。タスクビュー(図2.6右上)は対象タスクの問題空間(ゴール、解表現など)を表現し、GPNビュー(図2.6左上)は問題解決の処理構造を表現する。ビューはフィールドから構成されフィールドにTO/K-Lの構成要素(TO/K-Lで定義される語彙:

以下汎化タームと呼ぶ)を埋め込む。CLEPEにおけるTO/K-Lの役割は、このワークスペース上にエンドユーザが問題解決を表現する際の語とその配列を定める文法を提供することにある。

TO/K-Lで定義する語彙の品詞は，汎化動詞，汎化名詞，汎化形容詞，汎化制約に分類されており，GPN上に配置される汎化タームは各フィールドに与えられた品詞に関する制約を受けることになる。

問題解決モデルの記述において「文」としての役割を担うのが汎化プロセスである。

汎化プロセスは動詞と名詞を基本構成要素とし，一つの処理概念を表している。例えば，図2.3に表された，一番上のプロセス「Classify Job」は「ジョブを分類する」という処理を表している。このプロセスは「Classify Rsc」に汎化され汎化プロセスと呼ばれる。レキシカルレベルオントロジーは汎化プロセスに含まれる語の数，配列，それぞれの語が持つ統語的な役割を汎化動詞の定義において規定する。

図2.7に汎化動詞selectの定義を示している。問題解決オントロジーの記述は，クラス間の関係を定義する:class-hierarchyパート,構成要素や属性に関する制約を記述する:slot-defパート,あるインスタンスがそのクラスに所属するかどうかを判定する方法や，構成要素の間に成立する関係を定義するための:axiomパートから構成される。例えば、selectの場合は次の統語規則(:syntax)をもつことが必須であることを@により表しており

select (- [N1]) (-from - [N2])

という統語形態をとり得ることを定義している。ここで()内の文構成要素はいずれかが省略可能であることを表している．フィールドN1,N2には汎化動詞の対応アクティビティの作用対象を表す汎化名詞が入りうることを表している。さらにN1の実体は出力,N2の実体は入力オブジェクトであることを規定している。入力オブジェクト，出力オブジェクトに対する意味制約はTO/K-C(汎化動詞selectに対しては対応するアクティビティ(cor-activity)A-select(後述))を通して規定する。また，あるインスタンスが動詞「select」のインスタンスであるかどうかを判定する所属性定義(membership)において外延的判定(extensional)を指定した場合には,内包的な制約の検査は行わず,instance-of関係に基づいて判定することを表している。

エンドユーザによる問題解決モデルの記述は動詞が定めるこのような統語則に従ってなされる。例えば図2.6の画面で、エンドユーザがTO/K-Lの語の階層からマウス操

作により動詞selectをクリックした際には、選択可能な統語則がブラウザ上で表示される。この中からいずれかを選択することによって、統語則に適合したフィールドの列が汎化プロセス内に表示される。そしてユーザによってフィールド内に埋め込まれた単語がオントロジー(統語論的公理)に対して整合するかどうか確認された後で, TO/K-Cの概念レベルモデルと対応づけられる。

2.4.2 概念レベルオントロジー:TO/K-C- 概念レベル公理 -

TO/K-Cでは問題解決の過程を概念レベルで捉えるために必要な公理が定義される。TO/K-Cに定義される概念は，オブジェクト概念，叙述概念、アクティビティ概念，制約，に分類される。直感的にはそれぞれ、汎化名詞、汎化形容詞，汎化動詞，制約語彙の意味内容に相当する。

オブジェクト概念定義は，汎化名詞の対応概念として，それがタスクにおいてどういう性質をもった「もの」であるかということを表現する。

図2.8にオブジェクト概念である「解」の定義を示している。オブジェクト概念は，:slot-defパートにおいて，構成要素(component-of)，普遍制約(perm-spec)，遷移状態制約(status-spec-temp)によって特徴づけられる。それぞれのスロットについて，構成要素に関する制約，タスクの実行によるコンテクストとは独立の普遍的な性質と，タスクコンテクストのもとで付加され得る状態制約を記述する。図2.8の「解」オブジェクトの定義では，それが「割り付けの集合(O-assignment-set)」であること，問題解決の過程を通じて「中間(割り付けが完了していない)の」，「部分の」，「暫定の」，「最終(ゴールを最適に満足した)の」という状態を取り得ることを定義している。

叙述概念は，オブジェクトが取る状態の意味を定義する。図2.9の「暫定の」の定義では，「暫定の」クラスが叙述概念のサブクラスであり，割り付け集合に対して状態「暫定の」を付加しうる(modification-obj-of)ことを定義している。叙述概念「暫定の」の意味内容(status-spec)については，「全てのスケジュールの受け手(?rcp)の割り当てが終了しているが(:S-assigned)，最適(:S-optimal) ではない」ことを定義している。

また，図2.10に解オブジェクトがとり得る状態の順序に関する公理として，解オブジェクトが問題解決の過程を通じて，「部分の」，「中間の」，「暫定の」，「最適の」という順序で変化することを定義している。

アクティビティ概念は問題解決における行為の意味内容を表し，入力オブジェクトや出力オブジェクトに関するクラス制約や，オブジェクトの生滅に関わる作用を定義する。例えば，入力オブジェクトの集合から一つのオブジェクトを取り出すアクティビティ概念「A-Select」の定義では，入力オブジェクトが集合であり，出力オブジェクトは個体でなくてはならないことを定義している。また，オブジェクトの構成や生滅に関わる作用として，「A-Select」は入力オブジェクトからオブジェクトを限定する作用を持つことが定義されている。作用定義の具体例については，4.2.2節で述べる。

2.4.3 TO/Kの公理：語用論的公理

統語論的公理が汎化プロセス内部の語の統語的な役割を規定するのに対して，語用論的公理は，汎化プロセス間の整合したオブジェクトの参照関係を定める。

例えば，2.3.4節で述べたような問題解決コンテクストに依存したオブジェクトの同一性について，図2.3の(10)のプロセス「Select Nurse with Maximal Load from Temporary-Solution」プロセスにおけるTemporary-Solutionが二重ループの外側のループの出力である暫定解と参照関係にあるものとして矛盾がないことが語用論的公理によって定められている。

例として，図2.11に汎化名詞「暫定解」の定義を示している。汎化名詞のクラス定義には，クラス階層(:class-hierarchy)と対応オブジェクトに対する制約を記述する。この定義において語用論的公理は，対応オブジェクトスロット(cor-object)に記述される制約の内容である。語用論的公理としてここでは，汎化プロセスに埋め込まれた名詞，「暫定解」が前の汎化プロセスの出力である「暫定状態(S-temporary)にある解(O-assignment-set)」と参照関係になりうることを定義している。

2.5 タスクドメインオントロジーの公理

問題解決オントロジーが定める三つの公理は，(1) 統語論的公理が問題解決モデル記述のための基本的枠組みを定め，(2) 概念レベル公理が問題解決モデルの計算論的なオペレーショナリティを明確にしている。そして，(3) 語用論的公理が問題解決モデルの記述と意味を対応づける役割を担っており，全体としてタスクの意味論を明確にし，記述の容易性とオペレーショナリティの明確化を両立するための基礎的枠組みを与え

```
(define-task-s-object O-assignment-set (?$ass)
  :class-hierarchy (subclass-of O-assignment-set $object)
  :slot-def (
     (component-spec ?ass :@constraints
                          ((class O-assignment)))
     (status-spec-perm ?s-perm :constraints
                               ((class status)))
     (status-spec-temp ?s-temp :constraints ((or
                               (:s-temporary :s-partial
                                :not-s-completed :s-optimal))))
  :axiom (
     (membership (:extensional))))
```

図2.8:オブジェクト概念「解」の定義

```
(define-task-s-status S-temporary (?s-temporary)
  :class-hierarchy (and
    (substatus-of S-temporary Status)
    (modification-obj-of S-temporary O-assignment-set
                                 :instance ?$O-assignment))
  :slot-def (
    (status-spec ?s-spec :@constraints (and
         (forall ?rcp
              (=> (member ?rcp (problem.$rcp PROBLEM))
                  (S-assigned ?rcp)))
         (not (S-optimal ?$O-assignment)))))
  :axiom (
    (membership (:extensional))))
```

図2.9:叙述概念「暫定の」の定義

```
(define-task-s-axiom-status-order $O-ass%Status (?s-order)
   :slot-def (
    (Object ?object :@constraints ((class O-assignment-set)))
    (status-order ?status-order
                  :@constraints (
                   (S-Partial S-Intermediate
                    S-Temporary S-Optimal)))
    ))
```

図2.10 割り付け集合がとり得る状態の順序の定義

```
(define-task-s-noun temporary-solution (?t-sol)
 :class-hierarchy (subclass-of temporary-solutoin noun)
 :slot-def (
    (cor-object ?cor-object
                :@constraints (
                          (and (class O-assignment-set)
                               (status s-temporary)))))
  :axiom (
     (membership (:extensional))
          ))
```

図2.11:名詞「暫定解」の定義

ている。

本研究では，問題解決オントロジーが提供する枠組を基盤として，図2.3に示したように，「看護婦」「ジョブ」といったエンドユーザが日常的に意識しているドメインの概念を指向した記述の枠組みを設定している。このような枠組みにおいて，エンドユーザが記述する問題解決モデルから計算論的なオペレーショナリティまでの連続性を実現するためには，エンドユーザが記述するドメインの概念に対して，問題解決オントロジーで定義される問題解決の観点からの意味付けを与えるための枠組みが必要となる。このための枠組みとして，タスクドメインオントロジー(以下T-ドメインオントロジーと呼ぶ)という概念を導入する。T-ドメインオントロジーは対象概念であるドメインオントロジーを，問題解決における意味論を定義する問題解決オントロジーと対応づけるための概念の枠組みである。

このT-ドメインオントロジーには，大きく以下の2つの事について明らかにする能力が求められる。

I. 問題解決モデルに現れるドメイン概念の統語的な役割を明らかにする。

II. 問題解決モデルに現れるドメイン概念のタスクコンテクストでの意味内容を明らかにする。

この2つの能力をT-ドメインオントロジーが備えることで，エンドユーザによる問題解決知識の記述の観点からは，自然言語の統合則に準じた問題解決知識記述のための枠組みの中で，エンドユーザが日頃用いているドメイン概念を用いて自身の問題解決モデルを記述することが可能になる。また，システムによる問題解決知識の解釈の

観点からは，タスク世界の概念とドメインの概念が複雑に絡み合った問題解決モデルについて，タスク世界の整合した概念と対応づけて解釈することによって，ドメインの概念を用いて記述した問題解決モデルの整合性を，タスクにおける意味論のもとで検証することができる。

このように，対象世界の概念に対してタスクの意味論を与えるT-ドメインオントロジーを導入することで，問題解決モデルの記述から問題解決オントロジーに整合した意味論までの連続性が提供される。

図2.12に対象世界の概念である「nurse」に対して，タスクの観点からの「rcp(スケジュールの受け手」という意味付けを与えるT-ドメインオントロジーの定義を示している。T-ドメインオントロジーの定義は大きく四つのスロットから構成されている。task-conceptスロットは対象概念のタスクレベルの意味を同定するためのスロットである。ここでは，対象概念として問題解決オントロジーとして定義されたRcpオブジェクトクラスと結びつけられている。domain-conceptスロットは対象概念のドメインの世界の意味内容を取り出すためのスロットであり，ここでは，ドメインの意味内容としてドメインオントロジーとして定義された，Nurseクラスと結びつけられている。attr-mappingスロットではドメイン世界の属性とタスク世界の属性との対応関係が定義される。ここでは，ドメイン世界の看護婦の負荷という属性がタスク世界の負荷と対応づけられている。syntactic-propertyスロットでは，ドメイン概念の統語的な性質が定義され，レキシカルレベルモデル上に配置されたドメイン語彙の統語的性質が明示的に表現される。ここでは，ドメイン概念の看護婦がレキシカルレベルオントロジーで定義された名詞と明示的に対応づけられている。

この定義において上述したT-ドメインオントロジーに求められる二つの能力はsyn-

```
(Define-TD-Binding rcp&nurse (?td)
  :slot-def (
    (task-concept :@constraints O-Rcp)
    (domain-concept :@constraints Nurse)
    (attr-mapping ((of load :domain-concept) <->
                   (of load :task-concept)))
    (syntax-property :@constraints (noun))
))
```

図2.12:T-ドメインオントロジー「nurse&rcp」の定義

tactic-propertyスロットおよび，domain-conceptスロットとtask-conceptスロットを結びつけるT-ドメインオントロジーの公理によって実現される。このT-ドメインオントロジーがどの様にして生成されるかということについては，3.3.3節で述べる。

このように，問題解決コンテクストに依存したドメイン概念の意味内容を捉えるタスクドメインオントロジーを用いることで，問題解決知識の記述をユーザが日常的に捉えているドメインの世界の用語を用いて行うことができ，システムはこれをタスクコンテクストの下で解釈することができるため，ユーザにとっての記述の容易性と，システムにとっての計算論的意味の明示性を両立することが可能となる。

2.6 結言

本章では，問題解決オントロジーの構成と問題解決オントロジーが定める三つの公理について述べた。本研究で構成した問題解決オントロジーは，以下の4つの特徴を備えた枠組みを実現するための基盤的役割を担っている。

I. **問題解決モデルを容易に記述できる。**

II. **問題解決モデルの計算論的な意味を明確にする。**

III. **スケジューリングや診断などのタスクタイプ固有性に対応できる。**

IV. **問題解決全般に広く適用できる。**

問題解決オントロジーが定める基盤の下で，さらに，エンドユーザが指向するドメインの概念を指向した記述の枠組みを実現するための概念としてT-ドメインオントロジーの概念を導入した。これにより，ドメインの概念を指向した問題解決モデルから，問題解決オントロジーとして定義された計算論的な意味論までの連続性を実現するための基盤が整えられたことになる。

このような基盤の下で，問題解決のモデル化を支援するための環境がCLEPEの問題解決モデル記述環境としての側面である。問題解決モデル記述環境では，エンドユーザによる問題解決知識のモデル化の過程が問題解決オントロジーに基づいて適切に支援される。本研究では，１３のスケジューリングタスクを対象として，エンドユーザが問題解決モデルを記述することができること，その表現から計算機での実行概念までの連続性を実現できることを確認している。これは主に本研究で設定した問題解決オントロジーの奥行き方向の構成によって可能になる能力である。

また，問題解決知識をモデル化するための規約として，問題解決オントロジーを適

切に構築する際には，問題解決オントロジーを構築するための規約と支援環境が必要である。問題解決オントロジーの構築を支援するための環境がCLEPEのオントロジー構築支援環境である。オントロジー構築支援環境では，オントロジーオーサによる問題解決オントロジーの構築の作業を支援することができる。これは，主に本研究で設定したタスオントロジーの垂直方向の構成によって可能になる能力である。

第3章

概念レベルプログラミング環境CLEPEの全体像

3.1 緒言

問題解決オントロジーをシステムの基盤とすることで，問題解決オントロジーの構築から利用に至る過程を支援する概念レベルプログラミング環境CLEPEが実現される。本章では，CLEPEの二つの側面，オントロジー構築支援環境と問題解決モデル記述環境について，それぞれの環境を利用するオントロジーオーサとエンドユーザの視点で述べる。

3.2 問題解決オントロジー構築支援環境

オントロジーオーサの観点から，オントロジーとモデルの関係を考えるにあたっては，既に述べたようにタスクタイプ固有(Task-S)オントロジーとコアタスク(C-Task)オントロジーの構成(図2.2のb軸として表された垂直方向の構成)が重要となる。

本研究では，タスクタイプ固有オントロジー，コア問題解決オントロジーを構築するオントロジーオーサを，それぞれタスクタイプ固有オントロジーオーサ(OA(Task-S))，コア問題解決オントロジーオーサ(OA(C-Task))と呼んでいる。

3.2.1 オントロジー言語処理系

タスクタイプ固有オントロジーオーサにとって，タスクタイプ固有オントロジーがモデルになり，そのモデル記述に際して概念化の規約としてコア問題解決オントロジーが働く。このようなオーサの定義対象の階層性を捉えるためにメタの概念を導入する。直感的にメタクラス(コア問題解決オントロジー)のインスタンスが相対的に下位の概念(タスクタイプ固有オントロジー)に対応する。メタクラスはクラス定義を行うタスクタイプ固有オントロジーオーサが，定義を行う際に従うべき概念化の規約，すなわち，名詞，動詞，アクティビティなどの問題解決一般の概念について，各々のクラスが持つべき必須のスロットやスロットの役割などのメタな構成を規定する。タスクタイプ固有オントロジーではそのインスタンスとして，Selectなどの概念が定義される。本研究では，この作業を行うコア問題解決オントロジーオーサを，問題解決オントロジーの構築に際して重要な役割を担うオーサとして，オントロジー工学者を想定している。

図3.1に言語処理系を組み込んだCLEPEの全体像を示している。言語処理系は，モデルのオントロジーに対する整合性を検証する役割を持っている。問題解決オントロジーはその役割に応じて複数のレイヤに分けられている。これらの間の関係は，図3.1の下層から上層と進むにつれて，相対的に下層のオントロジーで定義される語彙が，オントロジー記述言語を段階的に拡張し，相対的に上層のオントロジーの記述を支える形態になっている。例えば，コア問題解決オントロジーはタスクタイプ固有オントロジーの意味を支える役割を担っている。

二つのタイプのオントロジーオーサと言語処理系の関係は次のようになっている(図3.1)。コア問題解決オントロジーオーサはTOL/0を用いてコア問題解決オントロジーを定義する。言語処理系は，コア問題解決オントロジーに基づいて言語を拡張し，タスクタイプ固有オントロジーを定義するための言語TOL/Sをタスクタイプ固有オントロジーオーサに対して提供する。タスクタイプ固有オントロジーオーサは，TOL/Sが提供する概念プリミティブを用いてモデルとしてのタスクタイプ固有オントロジーを定義する。

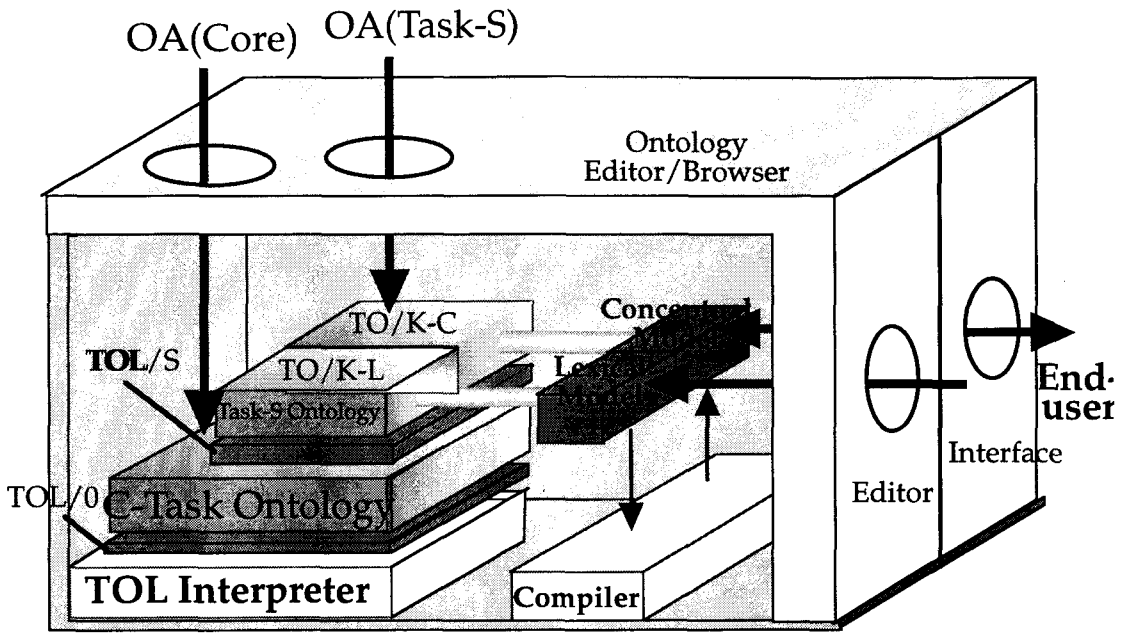


図 3.1:言語処理系を組み込んだ CLEPE の全体像

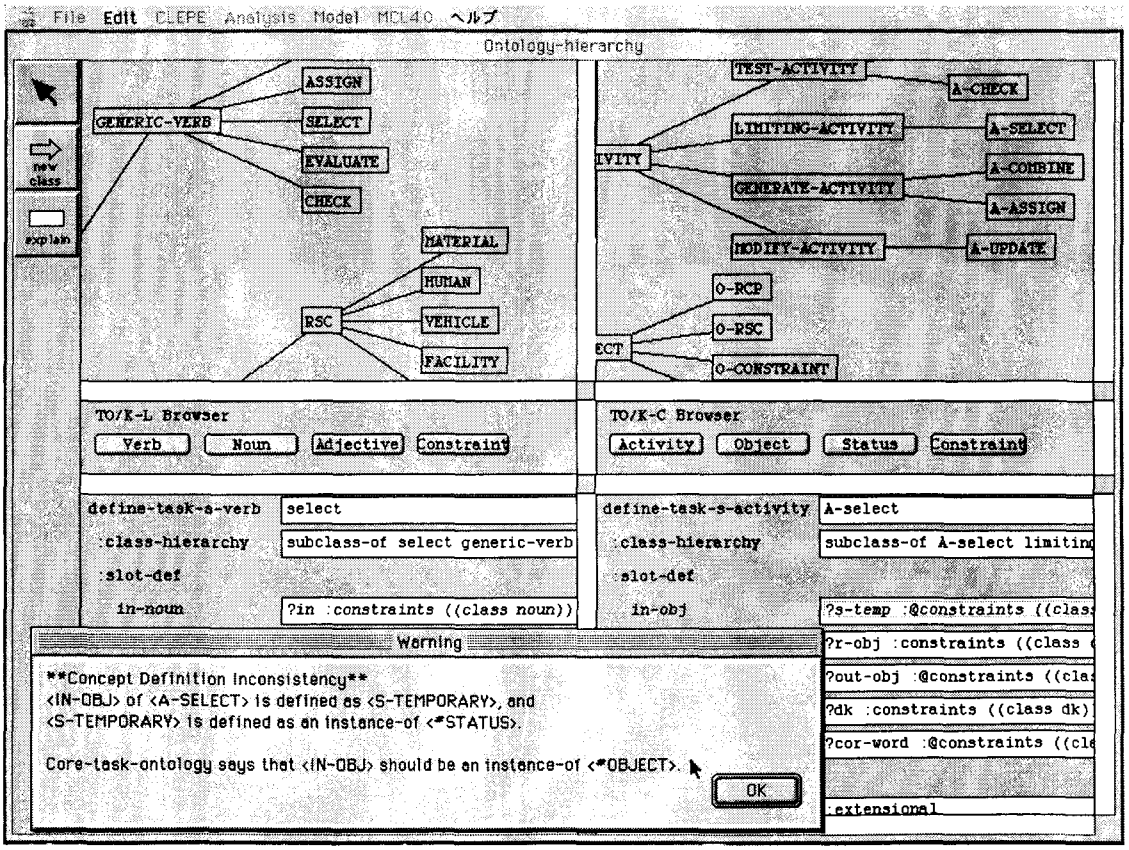


図 3.2:オントロジーエディタの画面ダンプ

3.2.2 オントロジーに基づくモデルの整合性の検証

オントロジー構築支援環境(図3.2)では，定義対象とする，TO/K-LとTO/K-Cをオントロジーオーサが同時に定義できるような環境を提供している。

大まかに図の左側がTO/K-L，右側がTO/K-Cを定義するための画面に対応し，上部に既に定義された概念の階層が表示されている。図の中央に位置する，「verb」，「noun」，「activity」，「object」などのボタンはコア問題解決オントロジーで定義された概念に対応している。このボタンのいずれかをマウスでクリックするかもしくは，上部のブラウザ上でクラスを表す新しいノードを作り，既存のノードと結ぶことで新しいクラスを定義する。その際，下部の画面に概念を定義するための概念プリミティブと各概念定義に応じたスロット，およびスロットの内容を定義するためのフィールドがルートクラス(後述)の定義内容と共に表示される。

タスクタイプ固有オントロジーオーサが定義対象のクラスとしてActivityを選択した際には，アクティビティを定義するための概念プリミティブ(define-task-s-activity)とアクティビティを定義するために必要なスロット，およびアクティビティのルートクラスの定義内容(後述する#Activity，図3.3のベースパートの内容)がフィールド内に示される。タスクタイプ固有オントロジーオーサはこれを参照しながらタスクタイプ固有のアクティビティの意味内容を定義し，プルダウンメニューから「check」を選択することにより，定義した内容のコア問題解決オントロジーが定めるTO/K-L,TO/K-C,TO/Kの三つの公理に対する整合性の検証を行う。例えば，タスクタイプ固有オントロジーオーサがタスクタイプに固有の動詞(e.g. Select)を定義する際に，その意味内容を表す概念として，アクティビティ以外の概念(e.g. オブジェクト概念O-Assignment-set)を指定した際には，コア問題解決オントロジーとして定義されている「動詞(TO/K-L)の意味内容は対応するアクティビティ概念(TO/K-C)によって定義されなけらばならない」というTO/Kの公理に違反しているといったことが，タスクタイプ固有オントロジーオーサに対して示される。図には「A-Select」の定義内容についてのコア問題解決オントロジーに対する整合性を検証した際の様子が示されており，タスクタイプ固有オントロジーオーサが定義した入力オブジェクトスロットのクラス制約がコア問題解決オントロジー(TO/K-C)が定める規約「入力オブジェクトスロットに関するクラス制約はメタクラスで定義される#objectのインスタンス(タスクタイプ固有オントロジーに

おけるオブジェクトクラス)でなければならない。」ということに違反しているというメッセージが示されている。タスクタイプ固有オントロジーオーサはこのメッセージをもとに定義内容を修正し，コア問題解決オントロジーが定める概念化の規約に整合したタスクタイプ固有オントロジーを構築することが出来る。

3.2.3 コア問題解決オントロジーの定義例

タスクタイプ固有オントロジーに対する規約としてのコア問題解決オントロジーについて，#Activity の定義を取り上げ定義内容について具体的に説明する。

コア問題解決オントロジーは，タスクタイプ固有オントロジーオーサがタスクタイプ固有オントロジーを定義する際に従うべき規約を定め，オーサの作業をガイドする役割を持っている。また，オーサが記述したオントロジーの整合性をシステムが検証する際の拠り所となる概念である。

図3.3にコア問題解決オントロジーのアクティビティ概念#Activityの定義を示している。ここで，#はコア問題解決オントロジーを表す表記として用いている。コア問題解決オントロジーの定義は大きくタスクタイプ固有オントロジーが従う規約を定義する:meta-partとこの規約に基づいて作るタスクタイプ固有オントロジーのルートクラスの定義を表す:base-partからなっている。タスクタイプ固有オントロジーオーサがアクティビティのスロットを定義する際の概念化の規約を: meta-partで定義している。整合性を検証する際の規約となる:meta-partの部分を中心に#Activityが定義する内容について説明すると，タスクタイプ固有オントロジーのアクティビティ概念が，入/出力オブジェクト(in-obj, out-obj)スロット，対応ワードスロット(cor-word)を持つことが必須である事が:necessaryキーワードによって定義されている。また，各スロットが全体に対する部分(part-of)あるいは関係(relation)としての役割を担うことが定義されている。スロットの内容を定義する際の規約としては，例えば入力オブジェクトスロットの値に関する制約として，入力オブジェクトに関するクラス制約がコア問題解決オントロジーの#objectのインスタンスすなわち，タスクタイプ固有オントロジーにおけるオブジェクトクラスでなければならず，さらにこの入力オブジェクトに関する内包的制約がタスクタイプ固有オントロジーのアクティビティ概念のクラス階層に沿ってより厳しくならないといけないということが: inherit で定義されている。

```
(Define-Core-Concept #Activity (?#activity)
  :class-hierarchy (#subclass-of #activity #concept)
  :meta-part (
    :base-layer (Task-S :root-class Activity :instance ?activity)
    :slot-def (
      (in-obj :part-of :necessary :class-spec  #object :inherit)
      (out-obj :part-of :necessary :class-spec  #object :inherit)
      (dk :part-of :class-spec  #dk)
      (cor-word :relation :necessary :class-spec #generic-verb)
      (internal-state :part-of)
              )
    :axiom (
      (membership :p-formula :necessary :inherit
         :element (:predicate-among (in-obj out-obj) :membership-p))
      (effect :function :necessary :inherit
              :element (:function-among (in-obj out-obj)))
          )
      )
  :base-part (
    :slot-def (
      (in-obj ?in-obj :@constraints ((class object)))
      (out-obj ?out-obj :@constraints ((class object)))
      (dk ?dk :constraints ((class dk)))
      (cor-word ?cor-word :@constraints ((class generic-verb)))
              )
  :axiom (
      (membership (:extentional))
      (effect ((satisfies ?dk ?in-obj ?out-obj)))
          )
)
```

図3.3:コアタスクオントロジー「#Activity」の定義

3.3 問題解決モデル記述環境

前節で述べた問題解決オントロジーをシステムの基盤とすることで,エンドユーザ自身が問題解決知識をモデル化できるような問題解決モデル記述環境を実現することができる。

ここでは,エンドユーザが容易に問題解決モデルを記述できるような環境を実現するための基本思想について述べ,その基本思想を実現するために問題解決オントロジーが担う役割について述べる。そして,エンドユーザが行うモデル化の作業を支援するためにCLEPEで実現されている機能について述べる。

3.3.1 基本思想

問題解決モデルの構築作業を,計算機システムの利用に不慣れなエンドユーザでも行うことの出来るような環境の実現を目指して,エンドユーザプログラミング環境に関する研究が様々なアプローチで進められている[DeBellis 95][Fischer 96][Cypher 95]。エンドユーザプログラミング環境は,対象世界に関する様々な概念に対応するソフトウェアコンポーネントを準備し,それを用いてエンドユーザが容易に問題解決モデルを記述(概念レベルプログラミング)できるような表現の枠組みと,そのデバッグ及び実行のための環境を提供する。そこでの「表現」としては通常の線形の言語よりも表現力に富んだ二次元(図的)言語が想定されることが一般的である。

本研究で開発を進めている問題解決モデル記述環境の背景にあるアイデアも基本的にこの線上にあり,エンドユーザ自身が自分の頭の中にある問題解決知識をモデル化するための環境を提供している。

ここで重要なことは、自分の頭の中にある「知識」を容易にかつ円滑に「表現」へと変換できるようにするための仕組みを考えることである。この問題を考えるにあたっては、ユーザインタフェイスの表面的な問題として扱うのではなく、エンドユーザが自分の問題解決知識を「モデル」化するのに必要な概念に関する深い洞察が必要である。本研究で提案している問題解決オントロジーは、エンドユーザが問題解決過程をモデル化するにあたって必要となる概念(オブジェクト・基本処理プロセス・制御、及びそれらの関係)を分析し,これを体系的に定義したものである。CLEPEの二つの側面の内,この問題解決オントロジーをモデル化の基盤とすることで,自分の知識を表現へと円滑に変換することができるような仕組みを実現しているのが,問題解決モデル記述環境としての側面である。

ここでは,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。

このような環境を実現するに際しては,2章で述べたように,人間にとっての記述の容易性と計算論的な意味の明示化に関するジレンマがある。この問題を解決し,エンドユーザが問題解決過程を容易にモデル化できるような環境を実現するために考慮すべきことは,以下の四つにまとめられる。

I. 記述容易性:エンドユーザが日頃意識している概念を中心に自分自身の知識を外化しやすい様な平易な枠組みを設定すること。

II. 不完全知識の許容性:エンドユーザが日頃意識していない概念であるが,計算機での実行を考える際に必要な概念について,エンドユーザに対して明示的に記述をすることを求めるのではなく,計算機側が読みとる能力を備えること。

III. 理解容易性:問題解決モデルの振る舞いをエンドユーザにとってわかりやすい形で示すこと。

IV. 計算セマンティクスまでの連続性:具体的な問題解決を行うことができる問題解決システムを生成できること。

このような要求を満足するためにCLEPEは,エンドユーザに対して以下のような機能/枠組みを提供している。

I.に関して,

- エンドユーザが日常的に使っている語彙の提供
- 制御フローを中心とした表現の枠組み

II.に関して,

- 問題解決モデルとして記述されていないオブジェクトフローを読みとり,エンドユーザの意図したモデル(概念レベル実行モデル)を構成する機能

III.に関して,

- II.で作られた概念レベル実行モデルに基づいて,問題解決における処理やそれによる対象世界の変化をエンドユーザの概念レベルで提示する概念レベル実行機能

IV.に関して,

-エンドユーザが書いた問題解決モデルを,具体的な問題を解決する問題解決システム(記号レベルのプログラムコード)へ変換する機能

3.3.2 CLEPEにおける問題解決オントロジーの役割

問題解決オントロジーが定義する三つの公理の役割を,3.3.1で示した要求に対応させてまとめると以下のようになる。

基本的に,統語論的公理は,I.に関して,

- 問題解決モデルを記述する際にエンドユーザが使う語彙(e.g. Pickup, Jobなど)を提供し,

- 語の配列に関する性質(図2.3左側の各ノード内部の文法)を規約として定めている。

さらに,

- レキシカルレベルモデルを構成する各ノードと制御リンクの接続に関する規約を定めている。

レキシカルレベルオントロジーに基づいてレキシカルレベルモデルの文法的な整合性が検証される。

語用論的公理は, II. に関して,

- エンドユーザが書いたレキシカルレベルモデルと概念レベル実行モデルの間の対応関係を定めるための規約を定めている。

また, 概念レベル公理は, III に関して,

- 問題解決モデルの実行を通じた対象世界の変化を表現するために,アクティビティの意味やオブジェクトの変化を表現するための概念を定めている。

3.3.3 問題解決オントロジーに基づくエンドユーザ支援

ここでは, 3.3.1で述べた要求を満足するために, CLEPEにおいてどの様な枠組みが実現されているかという事を, エンドユーザによる問題解決モデル記述のシナリオに沿って具体的に説明する。

問題解決モデル記述の過程は通常のプログラミング言語と同様に大きく,(1)エンドユーザによる問題解決モデルの記述,(2)問題解決モデルの実行 / デバッグの二つのフェーズに分かれている。

(Phase 1) 問題解決モデル記述の支援

図 3.4 に問題解決モデルを記述する際にエンドユーザに対して提供されるインタフェースを示している。エンドユーザがまず初めに行う作業は, 図3.4(a)のウィンドウに現れるフィールドに、対象とする問題の基本的な構成に関する用語を埋め込むことである。この作業を通じて、スケジューリングの受け手は看護婦であり、時間軸は日、スケジューリング資源はジョブであることが特定され、それに加えて問題解決を特徴づける基本的な概念としてゴール(看護婦の付加の均一化)も与えられている。

この準備段階の作業は、以下に述べるように、問題解決概念とドメイン概念を結び

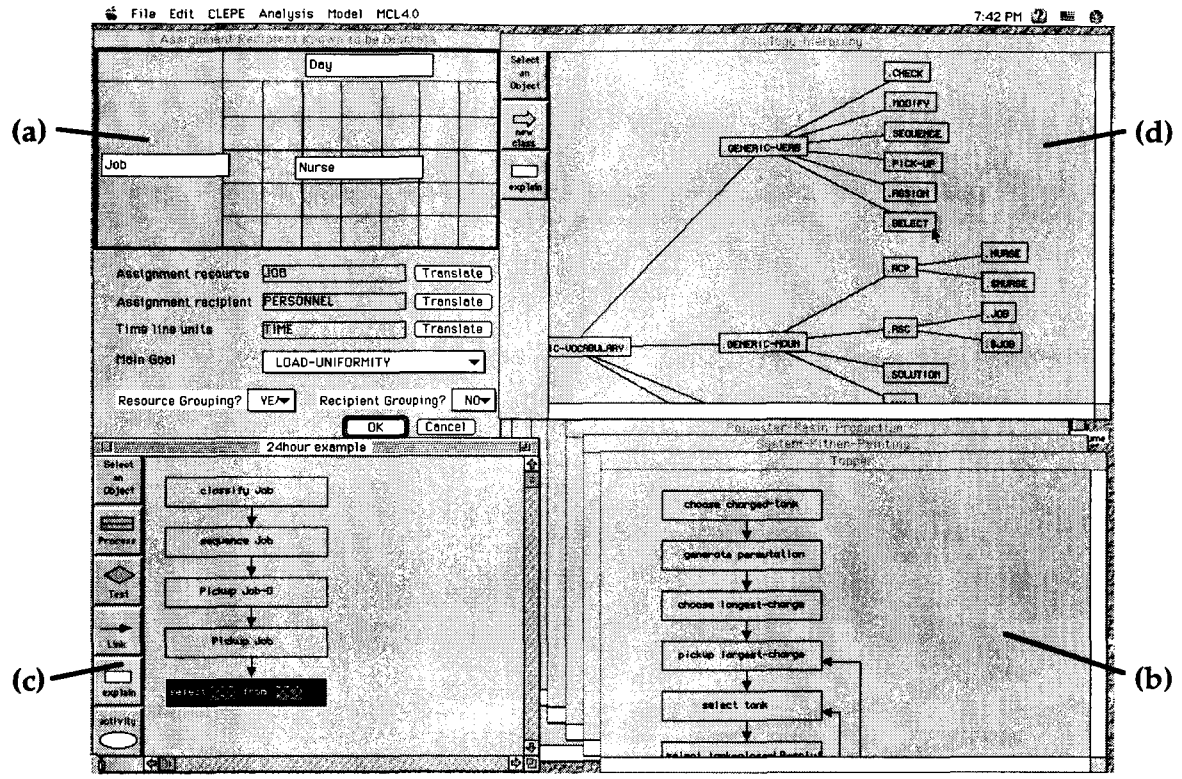


図3.4:CLEPEのインタフェース

つけるという重要な意義を持っている。問題解決オントロジーにおける概念定義は問題解決を規定するために必要な範囲で可能な限り一般的になされており、基本的にはスケジューリングの受け手(RCP)、スケジューリング資源(RSC)といった、問題解決という観点から汎化した語彙(汎化語彙)が階層定義の末端となっている。したがって,看護婦といったドメイン特有の概念については厳密に言えば問題解決オントロジーによる規定が及ばないことになる。しかし、このことは問題解決モデルの記述として見れば問題とはならない。例えば図3.4に示した例においては、看護婦がドメインにおいて持つ詳細な意味内容は、スケジューリング問題を考える場合には必要とはならず、それが問題解決においてスケジューリングの受け手に対応することが示され、問題解決概念として持つべき性質が明確になれば十分である。図3.4(a)に現れているフィールドは、それぞれが汎化語彙に対応しており、そこにエンドユーザが各ドメイン特有の用語を埋め込むことによって、このタスクとドメインの対応関係が与えられることになる(T-ドメインオントロジーを生成する)。タスク語彙とドメイン語彙の対応関係を

与えた後は、エンドユーザは図3.4(c)のウィンドウにおいてドメイン語彙を用いて問題解決モデルを記述することができるようになる。

この記述に先だって、白紙の状態から問題解決モデルを記述する際の負荷を軽減する目的で、類似問題に対する問題解決モデルの事例検索機能がエンドユーザに提供される。事例ベースは汎化語彙によってインデックスづけされており、検索は図3.4(a)で記述されたゴール等の情報に基づいてなされる。検索結果として得られた事例中の語彙は、汎化語彙を介してエンドユーザに適応した語彙に置換して提示されることになる。エンドユーザは必要に応じてこの事例を参照，あるいはコピー&ペーストによって部分的に再利用しながら問題解決モデルを記述する(図3.4(c))ことができる。図3.4(d)のブラウザでは、エンドユーザが利用可能な用語がレキシカルレベルオントロジーに従って階層的に提示される。エンドユーザは、その用語をマウス操作によって選択し、組み合わせることによって、問題解決モデルの処理内容（ノードの内容）を記述する(図 3.4(c))。この様にして記述された問題解決モデル(レキシカルレベルモデル)は，問題解決オントロジーが定義する統語論的公理に基づいて文法的な整合性が検証され，内部的な実行モデルとしての概念レベル実行モデルが構成される。

既に述べた様に，CLEPE では制御フローを中心とした記述の枠組みを提供している。したがって，例えば図2.3のレキシカルレベルモデルでは，「(2)Select」プロセスの出力である「Nurse」と「(3)Assign」プロセスの入力である「Nurse」が「同一の」 Nurse であるといったオブジェクトフローに関する情報が表現として明示的に表されていない。CLEPEでは，このオブジェクトフローを補完する機能(以下，オブジェクトフロー解釈と呼ぶ)を備えている。図3.5にこのオブジェクトフローをシステムが解釈する際に出力されるメッセージを示している。このようなメッセージを介したエンドユーザとシステムの対話的な作業を通じて，無矛盾なオブジェクトフローが同定される。

また，図3.6にエンドユーザが誤ったレキシカルレベルモデルを記述した際に，システムから出力されるエラーメッセージを示している。この例では，図2.3の(7)のプロセスにおいて「**暫定解**から最大の負荷を持つ看護婦を選ぶ」と記述すべき所を，「**初期解**」から選ぶと記述してしまっている。これに対してシステムは，「初期解に対応するものがない。前のループで作られた暫定解ではないか？」というメッセージを出力している。この情報に基づいて，エンドユーザはレキシカルレベルモデルの修正を行うことができる。

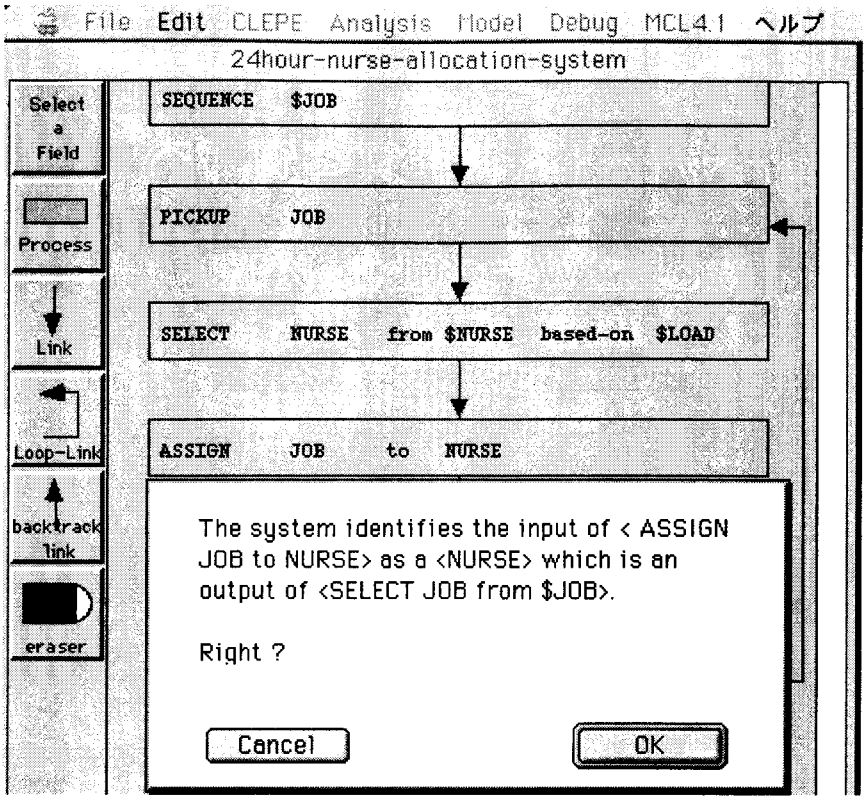


図 3.5: オブジェクトフロー解釈の実行画面

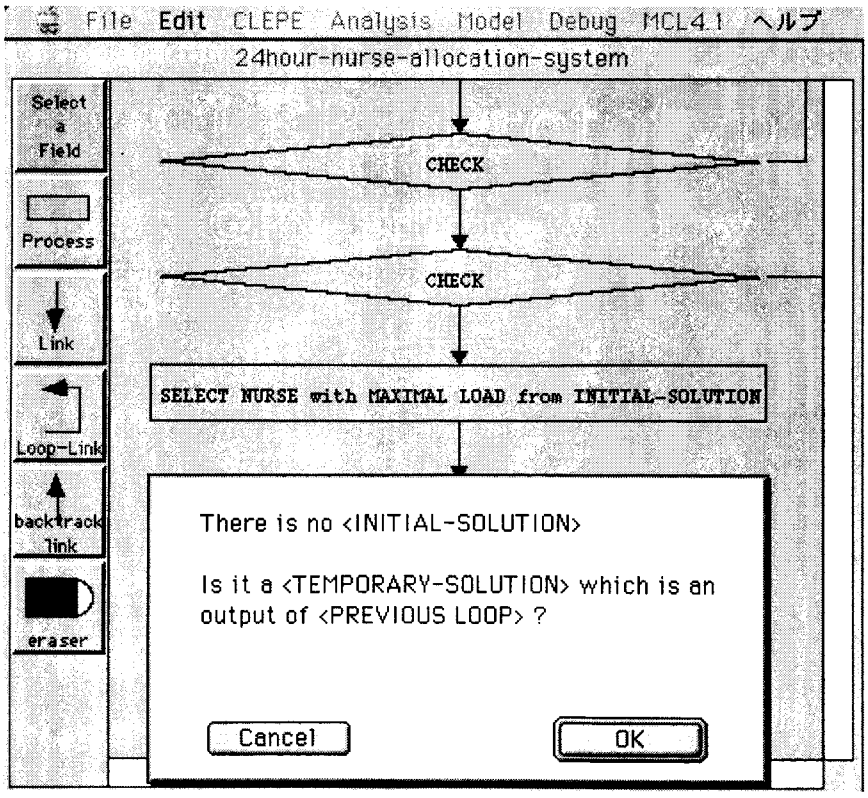


図 3.6: 問題解決オントロジーに基づく問題解決モデルの整合性の検証

この様にCLEPEでは,問題解決モデルのコンテクストの不整合を検出することが可能になっている。これは，問題解決オントロジーをシステムの基盤とすることで実現される能力であり，本研究の特徴の一つである。このオブジェクトフロー解釈の仕組みについては，4章で述べる。

(Phase 2) 問題解決モデルの実行 / 修正

エンドユーザが問題解決モデルを記述した後で次に行う作業は，自分が書いた問題解決モデルが自分の意図通りのものであるかを確認し，必要に応じて問題解決モデルの記述を修正することである。自分が書いたモデルが意図通りのものであるかどうかを確認する際の根拠になるのが、モデルの振る舞いに関する情報である。概念レベル実行は、そのような情報を問題解決モデルの記述レベルと同じレベルで提供する機能である。そこでは，問題解決の実行を通じたアクティビティのオブジェクトへの作用，オブジェクトの状態の変化や構成の変化の様子が記述レベルと同一のレベルでエンドユーザに対して提示される。

図3.7に概念レベル実行の様子を示している。図の大きく右側の画面では，処理実行順序とオブジェクトフローが表示され，左側の画面では処理を行うことによって対

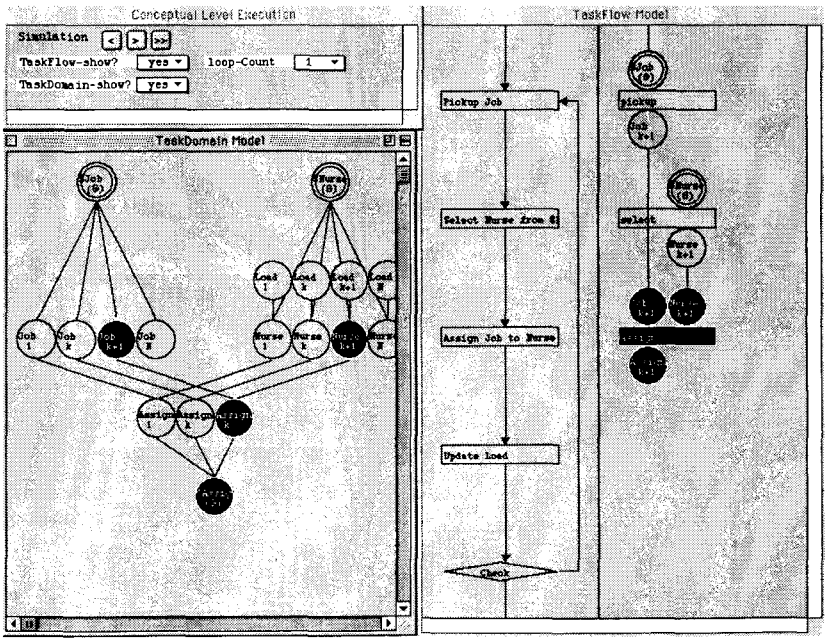


図3.7: 概念レベル実行の実行画面

象世界が変化する様子が示される。

概念レベル実行を支える概念レベル実行モデルの構成については，4章において詳しく述べる。

3.4 結言

本章では，オントロジーとモデルの関係を明確にすることによって実現される概念レベルプログラミング環境CLEPEの全体像について述べた。

本研究の特徴は，オントロジーとモデルの関係を明確に区別した問題解決オントロジーの構成を設定している事にある。「オントロジーとモデル」の関係を明確にすることの利点は，モデルに不具合が生じた際にその不具合を，モデルを操作するエージェントの間で共有するオントロジーに基づいて同定し，その根拠を示すことができる点にある。

CLEPEで実現されている問題解決オントロジーの構築支援，問題解決モデル記述の支援は，どちらもそれぞれの環境を利用するエージェントに応じた，「オントロジーとモデル」の関係を明確にすることによって実現される能力である。

すなわち，オントロジー構築支援環境を利用するタスクタイプ固有オントロジーオーサに対しては，タスクタイプ固有オントロジーを定義するための規約としてのオントロジー(コア問題解決オントロジー)が提供され，この規約に従ってタスクタイプ固有オントロジーオーサは対象とするタスクタイプに固有のモデル(エンドユーザに対してのオントロジー)を定義することが可能になっている。また，システムはオントロジーに基づいてモデルの整合性を検証し，不具合をオントロジーに基づいて指摘できる様になっている。

問題解決モデル記述環境では，エンドユーザが対象とするタスクタイプの固有性を捉えたオントロジーが提供され，このオントロジーに基づいて問題解決知識のモデル化が行われる。システムは，問題解決モデルの整合性をオントロジーに基づいて検証し，問題解決モデルに不具合があった場合には，それをオントロジーに基づいて指摘することが可能になっている。また，エンドユーザが書いた問題解決モデルから，これを問題解決オントロジーに基づいて再構成し，問題解決の意味を表すモデルを構成することが可能になっている。

第 4 章

問題解決モデルの再構成

4.1 緒言

本研究では，エンドユーザが記述したレキシカルレベルモデルから，その記述によって表現されたエンドユーザが意図した問題解決過程を読みとり，これを概念レベル実行モデルとして表現することを,「問題解決モデルを再構成する」と呼ぶことにする。

CLEPEは問題解決オントロジーが定める三つの公理と,T-ドメインオントロジーが定める公理を利用することで，問題解決モデルを再構成することができる。

本章では4.2節でタスクオントロジーをどのように利用して問題解決モデルを再構成することができるかということを,システムの内部的な構成と対応づけて説明する。そして,4.3節で概念レベル実行モデルに基づいて実現される概念レベル実行について具体的に述べる。4.4節では，本研究で作成したプロトタイプシステムCLEPEの動作を示しながら，問題解決過程のモデル化の支援がどの様に行われるかということを具体的に示す。

4.2 問題解決オントロジーに基づく問題解決モデルの構成

4.2.1 オブジェクトフロー解釈

システム実現の観点から機能的詳細に立ち入ってCLEPEの構成を示したものが図4.1である。以下に各機能モジュールの働きを示す。

- *TOL-Parser:* TOLを用いてオントロジーオーサが記述した問題解決オントロジーを読み込みパージングを行う。
- *Ontology Manager:* オントロジーベースを管理し，クラス定義の参照，インスタンスの生成，オブジェクトの同一化要求に応える。
- *Lex-Model-Parser:* レキシカルレベルモデルを読み込み，問題解決オントロジーが定義する統語論的公理に基づいて整合性を検証し，パージング結果を出力する。
- *WM Manager:* オブジェクトフロー解釈に関わるデータを一括管理する。
- *制約Generator:* オブジェクトフロー解釈の際，問題解決オントロジーに基づいて統語論的制約(SC)/語用論的制約(GC)(後述)を生成する。
- *照応解析エンジン:* フォーカス(後述)と制約を参照して，オブジェクトの参照関係を同定する。
- *Focus Manger:* オブジェクトフロー解釈の際に用いられるフォーカスを管理する。
- *Executor:* タスクフローモデル(後述)を読み込み，タスクドメインモデル(後述)を構成し，概念レベル実行を行う。

処理の大まかな流れは次のようになっている。問題解決モデルの記述の際には，まず問題解決オントロジー記述言語TOLで記述されたオントロジーが入力される。これをTol-Parserが解釈し，パージング結果をオントロジーマネージャ(OM)に送る。OMはオントロジーベースを管理し，クラス定義の格納や他のモジュールからのクラス定義の参照，インスタンス生成，オブジェクトの同一化といった要請に応じる。概念レベル実行モデルを構成する際には，図中のシェイディングされたオブジェクトフロー解釈モジュールが大きく関わっている。オブジェクトフロー解釈の際の全てのデータは

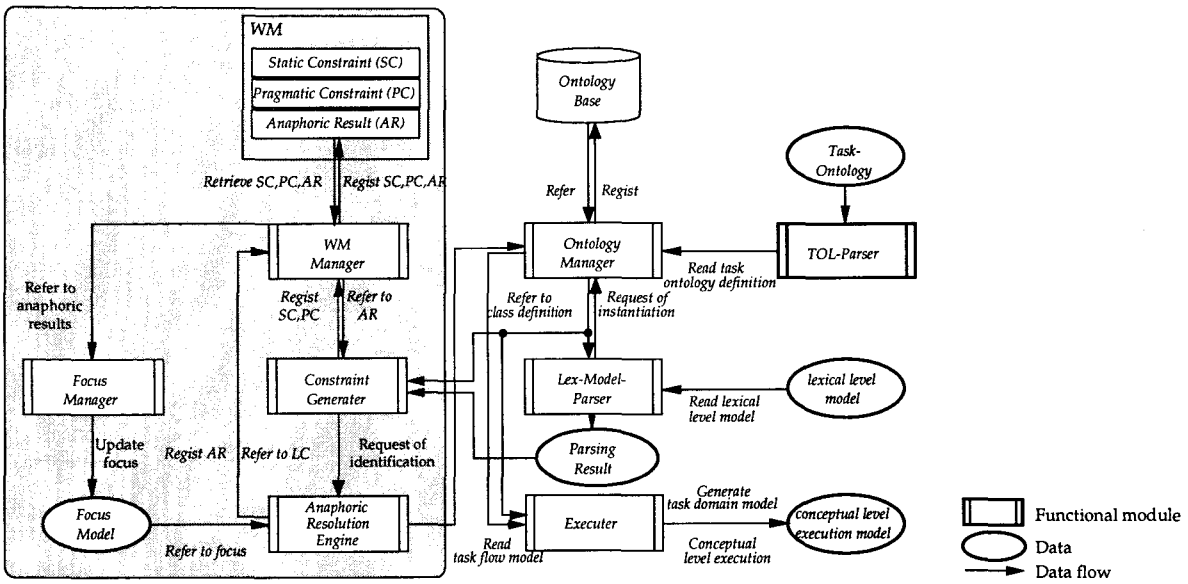


図 4.1: CLEPE のモジュール構成

ワーキングメモリマネージャ(WM Manager)が管理している。図4.2は図2.3のレキシカルレベルモデルの一部とオブジェクトフロー解釈を行う際に用いるフォーカス(後述)及び制約を表している。オブジェクトフロー解釈の過程は大きく,(1)オブジェクトフロー解釈の際に用いる制約の生成,(2)照応オブジェクトの同一化,の二つの段階からなっている。

(1)では，問題解決オントロジーが定めるアクティビティの入/出力に関するクラス制約と汎化プロセスに記述された名詞のクラス所属性に基づいて，各汎化プロセス毎に統語論的な制約(sc:syntactic constraint)が生成される，例えば，図4.2においてGP_2に関するscとして，入力($nurse-1)に関しては$NURSEクラスのインスタンス((instance-of $NURSE))で，出力(nurse-1)はNURSEクラスのインスタンス((instance-of NURSE))であるいう制約が生成される。

(2)次に，レキシカルレベルモデルの開始ノードから制御構造に沿った形で，オブジェクトの照応関係の同定が行われる。このオブジェクトの照応解析を同定する過程で,システムはフォーカスモデルを利用する,図4.2左にフォーカスモデルを示している。フォーカスモデルはタスクの制御構造が設定するオブジェクト参照に関するコンテクストをモデル化したものである。図中のフォーカスF_iでは汎化プロセスGP_{i+1}の入力オブジェクトと照応する可能性のあるオブジェクトがフォーカスされている。

フォーカスF_i内の黒枠で囲われた楕円は，GP_iで出力されたオブジェクトを表し，シャドウされた楕円は，GP_{i+1}の入力オブジェクトと照応したオブジェクトを表している。

照応関係の同定は，フォーカスされたオブジェクトの内,(1)で作られた統語論的制約を満たすオブジェクトを同定する事によって行われる。そして，照応結果(ar: anaphoric result)に基づいて照応関係にあるオブジェクトは同一化され(例えば，GP_3において入力job-2が，GP_1の出力のjob-1と照応することからこれらは同一化される)，語用論的制約(pc:pragmatic constraint)が生成される。語用論的制約は，照応結果に基づいて統語論的制約を更新したものであり，統語論的制約で与えられる制約に加えて，オブジェクトの生滅に関する変化を捉えるための概念も含まれている。これは，照応結果とアクティビティの作用に基づいて生成される。例えば，GP_3に関するpcでは，入力が「job-1」「nurse-1」と照応すること，assignの作用を定めるアクティビティ「A-Assigned(5.1節で後述)」が入力オブジェクトを構成要素とする割り付けオブジェクトを生成することから，「job-1」と「nurse-1」から構成される「assignment-1」が生成される(assigned assignment-1 (job-1 nurse-1))という制約が生成される。

フォーカスF_iの更新はGP_iの照応結果，アクティビティの作用，及び，レキシカルレベルの制御構造に基づいて行われる。例えばフォーカスF_3では，GP_3の照応結果つまり，GP_3の入力オブジェクトがGP_1(「pickup job」プロセス)で出力された「job-1」とGP_2(「select nurse」プロセス)で出力された「nurse-1」と照応すること，及びassignプロセスの作用として入力された二つのオブジェクトからなる「assignment」を生成する作用をもつことに基づいて，フォーカスF_3がF_4へと更新される。さらに，レキシカルレベルモデルの制御構造に基づくフォーカスの更新の例(F_5)として，ループのスコープ外ではループ内で生成されたオブジェクトに対するフォーカスが消滅し，ループ全体の実行を通して生成されたオブジェクトassignment-set-1に対するフォーカスが生成される様子が表されている.

このフォーカスはフォーカスマネージャによって管理され，照応解析エンジンによる照応結果は，WMマネージャによって管理される。フォーカスは，照応データの追加,更新に応じてWMマネージャがフォーカスマネージャを起動することで更新され，データ駆動でフォーカスが更新されるメカニズムが実現されている。

最終的に得られた概念レベル実行モデルを元に，エンドユーザはExecutorを介して4.3節で述べる概念レベル実行を行うことができる。

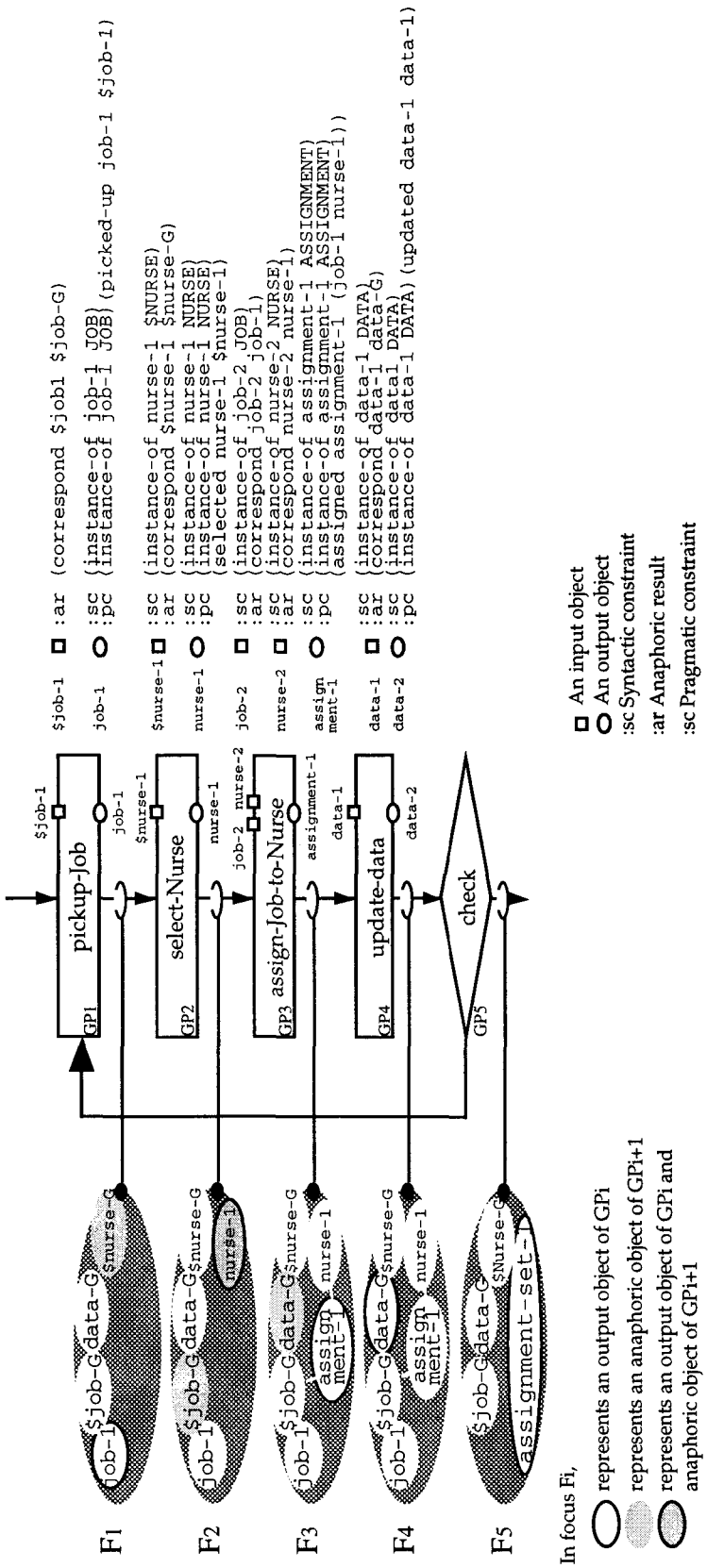


図4.2: レキシカルレベルモデルと対応するフォーカスモデル

4.2.2 概念レベル実行モデルの構成

問題解決知識のモデル化を支援するためには，エンドユーザが記述した問題解決モデルについて，そのモデルが自分の意図したことを表現しているかどうかを，エンドユーザ自身が確認できるような環境が不可欠である。概念レベル実行は，この確認作業を行うためのモデルの振る舞いに関する情報を，エンドユーザが思考する概念レベルで提供する機能である。

概念レベル実行の基盤となる概念レベル実行モデルはタスクフローモデル(TFM: Task-Flow Model)とタスクドメインモデル(TDM:Task-Domain Model)から構成されている。

タスクフローモデルは，アクティビティの作用を受けることによるオブジェクトの生滅，変化をレキシカルレベルモデルの制御構造に沿った形で捉えるモデルであり，アクティビティの実行順序やアクティビティが作用対象とする入出力オブジェクトが明示的に表現される。タスクドメインモデルは，タスクの実行を通じた対象世界での物の成り立ちやその変化を表現するモデルである。この二つのモデル相互の関係，例えば，タスクフローモデルにおける問題解決アクティビティの作用による対象世界の変化を適切に捉えるためには，対象世界の概念を問題解決コンテクストに組み込むことにより，タスク概念とドメイン概念を統合する枠組みを備える必要がある。対象世界の概念と，そのタスクの観点からの意味内容とのバインド関係を表現するT-ドメインオントロジーはこのための概念である。このT-ドメインオントロジーに基づいて，問題解決フローの任意の時点における処理と，それによるドメインにおける変化を対応づけたり，逆にドメインの世界での物の変化をタスクの世界に反映することが可能になる。

図4.3に図2.3のレキシカルレベルモデルの一部((4)～(8)の部分)と，対応するタスクフローモデル，タスクドメインモデルを示している。図の中央がTFMであり右側がTDMに対応している。上下に位置するTFMは，上側(TFM_k)がループのk周目におけるアクティビティの作用，オブジェクトのフローを表している。下側(TFM_{k+1})ではその次の周回k+1周目における同様の情報が表されている。図の左端には問題解決時間を示している。例えば，ループのk周目においてGP_1の処理が終了した時刻は$tp_{k(GP1)}$と表されている。問題解決時間は，レキシカルレベルモデルの制御構造に沿って，処理を基

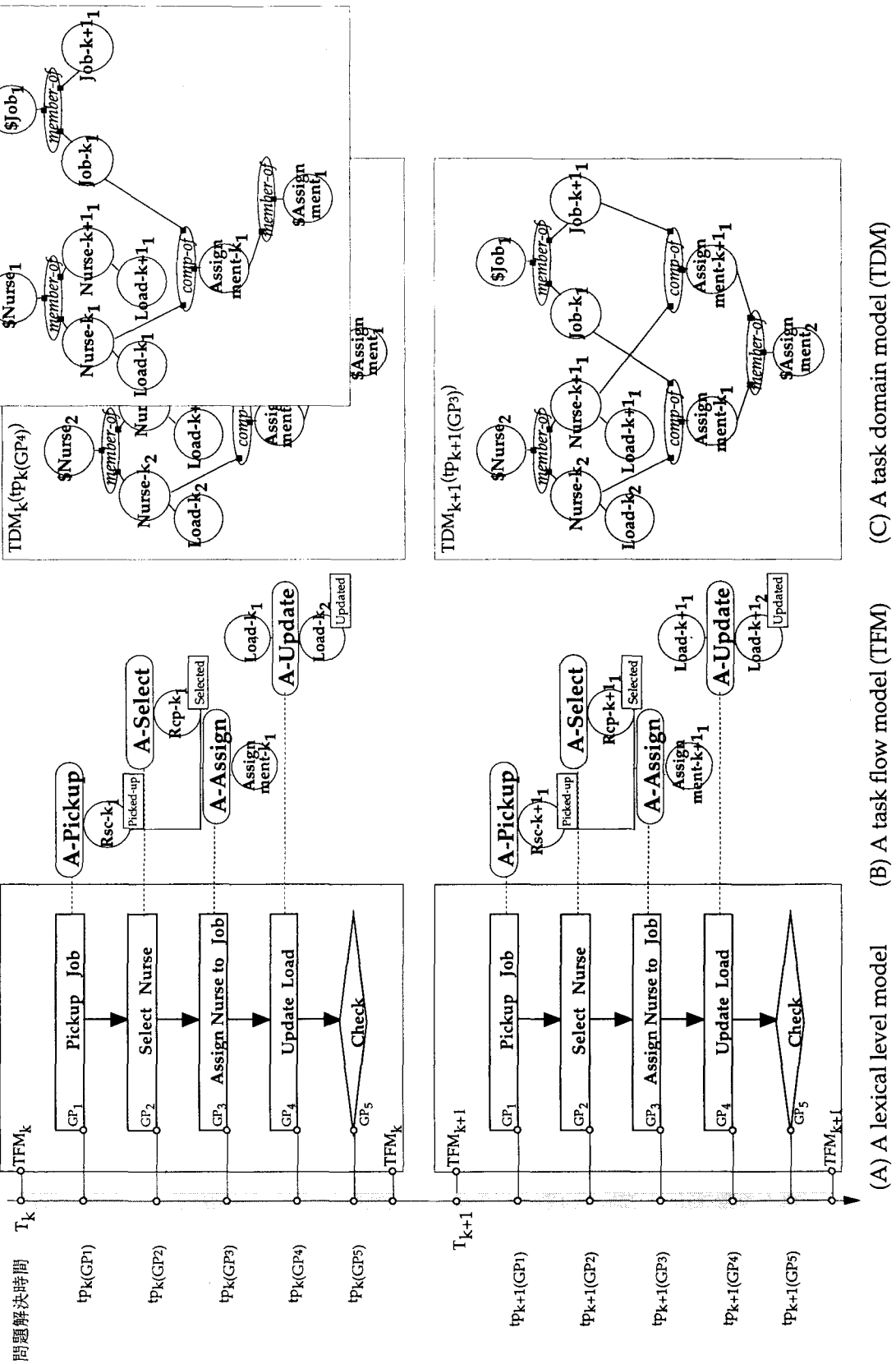


図 4.3: タスクフローモデルとタスクドメインモデル

準に捉えた時間の概念である。これは，概念レベル実行モデルにおけるアクティビティの実行の前後関係を表現する働きを持っている。右側に位置するTDMの内，上部に位置する二つのモデル($TDM_k(tp_{k(GP3)})$), ($TDM_k(tp_{k(GP4)})$)はそれぞれ，ループのk周目でのGP_3, GP_4の処理が終了した際のモデルを表している。下部に位置するモデル($TDM_{k+1}(tp_{k+1(GP3)})$)は，ループのk+1周目でのGP_3プロセスの処理が終了した際のモデルの内容を表している。それぞれのモデルの内容は，例えば，TFM_kでは，ループのk周目においてGP_1で「取り出された(picked-up)ジョブ(スケジュールの資源:Rsc-k_1)」とGP_2で「看護婦の集合から選ばれた(selected)看護婦(スケジュールの受け手:Rcp-k_1)」が入力され，「割り付け(assignment-k_1)」を生成する様子が示されている。ここで，オブジェクトiは$Obji_v$と表されており，vは後述するバージョンを表している。TFMでの処理によるTDMの変化はT-ドメインオントロジーを通じて起こる。例えば，$TDM_{k+1(GP3)}$では，Assignの作用による対象の世界の変化として，ジョブ(Job-k+1_1)と看護婦(Nurse-k+1_1)からなる(component-of)割り付けオブジェクト(Assignment-k+1_1)が新たに生成され解($Assignment)の構成要素(member-of)となる様子が示されている。

ここで，オブジェクトの変化を捉えるための概念としてバージョンの概念を導入する。バージョンは問題解決時間を用いて「ある問題解決時間tp_xからある問題解決時間tp_yまで，状態S_{xy}である」という形式で表現される。そして，オブジェクトが変化したことを表す，「バージョンの変化」はそのオブジェクト自体の状態の変化や他のオブジェクトとの関係の変化(e.g. 集合の構成要素の変化など)に基づいて行われる。$TDM_k(tp_{k(GP4)})$では，オブジェクトの構成要素の属性値変化に伴う全体オブジェクトのバージョンの変化が示されている。TFMにおいて看護婦(Rcp-k_1)の負荷(Load-k_1)が(Load-k_2へと)更新された際には，その変化がTDMにおいて伝播し，看護婦(Nurse-k_2)を構成要素とする看護婦の集合($Nurse)のバージョンが($Nurse$_2$へと)更新されている。これによってシステムは，例えばSelectプロセスの働きについて単に「看護婦の集合から看護婦を選ぶ」といった実行ではなく「前回の割り付けに基づいて負荷が更新された看護婦の集合から看護婦を選ぶ」といった問題解決のコンテクストを捉えた適切な情報を与えることが可能になる。

4.2.3 オブジェクトの変化のモデル

問題解決オントロジーが定める様々な概念の内, 4.2.2節で述べた物の変化を捉えるための概念は，アクティビティの作用概念定義である。

図4.4にスケジュールの受け手(Rcp)に対して資源(Rsc)を割り付けるアクティビティ概念「A-Assign」の作用を定義する「A-Assigned」の定義を示している。アクティビティの作用定義は，アクティビティ定義の作用定義フィールドで作用を定義するための述語である。アクティビティが入力に対して処理を行うことによって出す出力と，対象世界における変化が定義される。概念レベル実行でエンドユーザに対して示される処理の内容や対象世界における変化は，この作用定義に基づいて提供される。

図4.5に示した「A-Assigned」は，図2,3，図4.3のレキシカルレベルモデルにおいてエンドユーザが記述した「Assign」の作用を定義する。作用定義は大きく，スロット定義フィールド(:slot-def)と公理定義フィールド(:axiom)からなっている。スロット定義フィールドでは，A-Assignedがスケジュールの受け手(?in-rcp)および資源(?in-rsc)を定義域(:domain)とし，割り付け(?out-assignment)を値域(:range)とする様な作用であることが定義されている。公理定義フィールドは，タスク作用パート(task-effect)とドメイン作用パート(domain-effect)からなっている。タスク作用パートでは作用を受けることによるオブジェクトの生滅や変化に関する公理が定義され，ここでは割り付けオブジェクトを「生成」する作用を持つことを定義している。ドメイン作用パートでは作用を

```
(define-task-s-effect A-Assigned (?a-assigned)
    :slot-def (
      (in-rcp ?in-rcp :domain :@constraints ((class O-rcp)))
      (in-rsc ?in-rsc :domain :@constraints ((class O-rsc)))
      (out-assignment ?out-assignment :range
                      :@constraints ((class O-assignment)))
     )
    :axiom (
      (task-effect ((generate ?out-assignment)))
      (domain-effect ((when (after ?a-assigned)
                    (appear (component-of ?out-assignment
                                          (?in-rcp ?in-rsc)))
                    (appear (member-of %Assignment-set
                                       ?out-assignment)))))
      (membership (:extensional))))
```

図4.4: 作用「A-Assigned」の定義

受けることによる対象世界の物の変化に関する公理が定義され，作用が起きた際に対象の世界において(生成された)割り付け(?out-assignment)と，スケジュールの受け手(?in-rcp)および資源(?in-rsc)の間にcomponent-of関係が現れ(appear)，さらに生成された割り付けと割り付け集合($Assignment-set)の間にmember-of関係が現れることが定義されている。この作用定義によって，問題解決の過程でタスク世界において行われるアクティビティによる対象世界の変化を適切に捉えることが可能になっている。

4.3 概念レベル実行

既に述べたように，CLEPEが提供する概念レベル実行機能によって，エンドユーザは自分が書いたレキシカルレベルモデルが自分が意図したことを正しく表現しているかどうかを確認することができる。ここでは，問題解決オントロジーが提供する知識やモデルについて述べ，それを利用して計算機が提供する実行環境について説明する。

4.3.1 概念レベル実行

CLEPEが提供する概念レベル実行では，問題解決の実行を通じたアクティビティのオブジェクトへの作用，オブジェクトの状態の変化や構成の変化の様子を記述レベルと同一のレベルでエンドユーザに対して提示される。

概念レベルの実行可能性は問題解決オントロジーに基づくシステムが持つ特徴的な機能である。この機能の特徴を捉えるために従来型のプログラミング環境と対比して考えてみる。そこで振る舞いに関する情報として得られる典型的な情報は変数名とその値の変化である。仮に，問題解決モデルに誤りがあったときに，デバッグ中のエンドユーザは、その誤った「値の変化」からそれに関与した、プログラムのオペレーションの実行過程を頭の中で再現し、誤りの原因をプログラミング上の問題と対象世界の問題に自ら区分して解釈しなければならない。概念レベル実行においては、問題解決モデルの記述と意味内容の対応が問題解決オントロジーで規定されているおかげで、処理の進行にともなう対象の変化に関する情報を記述レベルと同一のレベルで提供することが可能になっている。

4.3.2 問題解決コンテクストの同定：問題解決 Causality

問題解決モデルに対するエンドユーザの理解を助け，必要に応じてデバッグできるような環境を実現するためには，エンドユーザが要求する情報に対して，問題解決のコンテクストを捉えた適切な情報を与える能力がシステムに対して求められる。問題解決の動的コンテクストを捉える概念として，本研究では，問題解決Causalityの概念を導入する。

問題解決Causalityは問題解決モデルを構成する部品間の相互の因果関係を捉えるための基本概念である。例えば，ある部品が他の部品に一定の目的の下で，何らかのオブジクトを供給しているときに，それらの部品間にCausalityがあるという。ユーザが記述した問題解決モデルに対してオブジェクトフローが得られ概念レベル実行モデルが再構成された次の段階でCausalityに基づいて，問題解決モデルの因果関係が明らかにされる。因果関係は，一般に，問題解決モデルを構成する個々の部品の役割，例えば,「繰り返し構造」のなかでの「条件判断」の役割などを明確にする役割をもっており，実行情報を提示するために不可欠な基礎知識である。Causalityの概念によって，エンドユーザが作成したドメインに特有の語彙で表現された問題解決モデル(レキシカルレベルモデル)から，ユーザが意図した問題解決構造が明らかになる。

一般に問題解決に現れる様々な概念に対してエンドユーザが認識する様々な因果関係，例えば，問題解決モデルに現れる複数のアクティビティが持つ作用の間の因果関係,アクティビティの作用結果として構成される複数のオブジェクト間の包含関係(全体-部分)の動的な変化は，複雑に絡み合っている。また，エンドユーザにとっては当然であるが，この因果関係に関する認識をモデル記述として明示的に表現することは困難である。さらに，エンドユーザにとっては，問題解決の記述それ自体が問題解決モデルに対する自身の設計意図を表現したものになっている。つまり，単純な例として例えば図2.3のレキシカルレベルモデルの場合において，生成作用(問題解決において新しいオブジェクトを作る)を持つAssignプロセスと参照子生成作用(集合から作用対象とするオブジェクトを取り出す)をもつPickupプロセスの間に操作対象のオブジェクトの授受関係が成立している場合を考える。この場合，例えばPickupプロセスとAssignプロセスを接続することの意図として，Assignプロセスが作用対象とするオブジェクトを限定するためにPickupプロセスをAssignプロセスの前に行うということ

や，それをループ構造に組み入れることで，Pickupの操作対象のオブジェクトに対して網羅的に処理を行うということがエンドユーザの思考の中で暗黙的に捉えられている。ユーザの問題解決モデルに込める設計意図をシステムが解釈し，その結果を概念レベル実行として提示することは，CLEPEが備える能力として重要であるといえる。

問題解決モデルに対してエンドユーザが込める様々な概念間の因果関係，言い換えれば，エンドユーザが問題解決モデルに込めた設計意図を捉えるためのタスク一般に関する知識が問題解決Causalityである。

問題解決オントロジー全体の中で，Causalityを位置づけてみる。

エンドユーザが日常的に行う問題解決に現れる処理の間の因果を捉えるための一般的な知識は問題解決Causalityのコア問題解決オントロジーとして定義される。複数のアクティビティ，例えば参照子生成作用を持つアクティビティと生成作用を持つアクティビティの間に成立する問題解決Causalityの公理が，このレベルで定義される。

一方で，エンドユーザ個々の問題解決の因果に対する認識を捉えるためには，問題解決Causalityをエンドユーザが日常的に意識するドメインの概念と対応づけて定義しなければならない。例えば，「もの」の「成り立ち」についてエンドユーザはドメインの世界で捉えている。問題解決Causalityは問題解決コンテクストでどのようにその「成り立ち」が形成されるかということを一般的に捉え，T-ドメインオントロジーはこの二つを結合する役割を持っている。

問題解決におけるタスクコンテクストを捉えるためには大きく，ドメイン概念のタスクの観点からの意味内容を取り出す能力および，ドメイン概念をタスク概念に統合することにより得られるタスクレベルのモデルにおいて，そこに配置されたタスク概念の間の因果関係を捉える能力が必要となる。この二つの能力はそれぞれT-ドメインオントロジー，および問題解決Causalityにより実現されている。これによって，ヒューマンフレンドリな問題解決モデル記述環境を実現することが可能になっている。

本研究では現在，問題解決を行う際にエンドユーザが認識する因果関係に関するオントロジーとして現在，以下に示す問題解決Causalityを明らかにしている。

I. 成果物の構成に関する成り立ちの因果関係を捉えるプロダクションCausality
II. ループ構造などの制御構造に基づいた，アクティビティ間の因果関係を捉えるコントロールCausality
III. 構成要素と全体の関係について，構成要素が作用を受け変化した際の全体に対す

る変化の伝播を捉えるパートホールCausality

IV. 他のアクティビティの実行条件を整えるなどの,アクティビティ相互の機能的関係を捉えるアクティビティCausality

このうちI,IVのCausalityは,主にオブジェクトフローに基づいてモデルに適用され,IIはレキシカルレベルモデルの制御構造とオブジェクトフローから明らかにされる問題解決Causalityである。IIIについては,対象世界の「もの」の関係に基づいて明らかにされるCausalityである。

次節では,概念レベル実行の例を示しながら,問題解決Causalityの利用法について具体的に述べる。

4.3.3 概念レベル実行の例

エンドユーザが記述したレキシカルレベルモデルに対応する概念レベル実行モデルを構成し,タスクとドメインを統合する枠組みを備えることで,問題解決に際してエンドユーザに対してどのような実行環境を提供できるかについて説明する。

概念レベル実行の際に,エンドユーザに提供されるインタフェースは2つのウインドウから構成され,それぞれのウインドウに,図4.3に示したTFM,TDMが表示される。概念レベル実行の過程は,問題解決時間に沿った概念レベル実行モデルの変化を示すことによって行われる。この概念レベル実行環境では,エンドユーザは,画面上で特定のオブジェクトや,アクティビティ,あるいはループなどをマウスでクリックし,そこで表示されるメニューを選択することによって,システムに対して情報を要求することもできる。これに対するシステムからの回答も図3.7で示した画面上で表示される。

概念レベル実行において,前節で述べた問題解決Causalityは,概念レベル実行モデル中に現れる様々な関係の内,エンドユーザが要求する質問に対して回答すべき重要な部分を取り出す働きを担っている。

ここでは,エンドユーザが書いた図2.3のレキシカルレベルモデルを例にして,エンドユーザに対して提供される情報について述べる。例として,ユーザの興味の対象((I)物の成り立ち,(II)制御,(III)アクティビティの働き)に関する情報の要求と,システムが提示する情報を自然言語で書き下すと以下のようになる。

I. 物の成り立ち

Q.「今作られた割り付けオブジェクトの構成はどのようになっているか？」

A.「Pickupプロセスでジョブが取り出され，Selectプロセスでこのジョブに対して割り付け可能な看護婦が選ばれ，Assignプロセスでジョブと看護婦から成る割り付けが構成されている。そしてこの割り付けオブジェクトが解の構成要素になる。」

この例では，割り付けオブジェクトの構成要素であるジョブと看護婦の関係を問題解決の過程にそって説明している点が特徴的である。そして，「割り付けオブジェクトが解の構成要素になる」という，エンドユーザが問題解決モデルとして明示していない情報を，エンドユーザに対して明示的に示されている点が特徴的である。

この回答を示す際に用いられるCausalityは，プロダクションCausalityである。アクティビティ(Assign)の成果物としての「もの」の構成に対するユーザからの問い合わせに対して，成果物(割り付け)と，それを構成するもの(ジョブと看護婦)に対する処理(Pickup,Select)との間の因果関係を捉えるために用いられている。

II. 制御

Q.「この一連の繰り返し処理はいつ終了するか？」

A.「未割り付けのジョブの集合に対して割り付けが終了したとき，一連の繰り返し処理は終了する。」

この例では，単に(1)Pickupプロセスの入力である「ジョブの集合」を網羅するまでと回答するのではなく，その問題解決コンテクストの下での意味(「まだ割り付けが終わっていない」ジョブの集合)を捉え，問題解決に対してエンドユーザが意図することを明示的に捉えている点が特徴である。

この回答を示す際に用いられるCausalityはコントロールCausalityである。ループの終了条件に関するユーザからの問い合わせに対して，ループを通じて網羅的に操作する対象(Pickupの入力であるジョブの集合)とループ全体の成果物(割り付け集合)の間の因果関係を捉えるために用いられている。つまり，ループの繰り返し処理を通じて操作されたジョブの集合が段階的にループの成果物である割り付け集合を構成していくという因果関係を捉えている。これによって，Pickupの操作対象のジョブの内，操作を受けていないジョブの集合が持つ問題解決コンテクスト，すなわち解の構成要素にまだなっていない(未割り付け)ことが同定されている。

III. アクティビティの働き

Q.「Selectプロセスはどのような処理を行うか？」

A.「取り出されたジョブに対して，割り付け可能な看護婦を前回の割り付けに基づいて負荷の値が更新された看護婦の集合から看護婦を選択する。」

この例では，エンドユーザがレキシカルレベルモデルに明示的に記述した作用だけでなく，暗黙に意識する対象世界の物の変化の伝播を明示している点が特徴的である。

この回答を示す際に用いられるCausalityは，パートホールCausalityである。対象世界においてものが変化した際には，対象世界の関係に基づいて，その変化がモデル全体に伝播する。パートホールCausalityは構成要素(看護婦)が作用((負荷を更新する)を受け変化した際の全体(看護婦の集合)に対する変化の伝播を捉えるために用いられている。これにより，Selectの操作対象の看護婦の集合についての問題解決コンテクスト(負荷が更新された)を捉えることが可能になっている。

4.4 プロトタイプシステムの動作例

ここでは，エンドユーザとCLEPEの対話的な作業を通じた問題解決モデルの再構成の仕方について，本研究で作成したプロトタイプシステムの動作を例にして具体的に示す。エンドユーザとシステムの対話的な作業を通じた問題解決エンジンの生成過程は，大きく(1)問題の基本構成の記述，(2)問題解決モデルの記述，(3)問題解決モデルの概念レベル実行，(4)希望レベルのビルディングブロックとの対応づけの4つのフェーズからなっている。

4.4.1 問題の基本構成の記述

図2.3に示した要員配置問題を例にして述べる。問題解決モデルを記述する際にエンドユーザが行う作業は問題の基本的な構成を記述することである。ここでの作業は，大きく(1)タスクタイプの選択，(2)解表現の選択，(3)ドメイン語彙の入力，(4)問題解決のゴールを選択することである。

(1) タスクタイプの選択

図4.5にCLEPEの起動画面を示している。ここでエンドユーザがまず始めに行う作業は，起動画面左上にあるタスクタイプ選択ボタンをマウスでクリックし，対象とするタスクタイプを選択することである。選択されたタスクタイプに応じて，システム

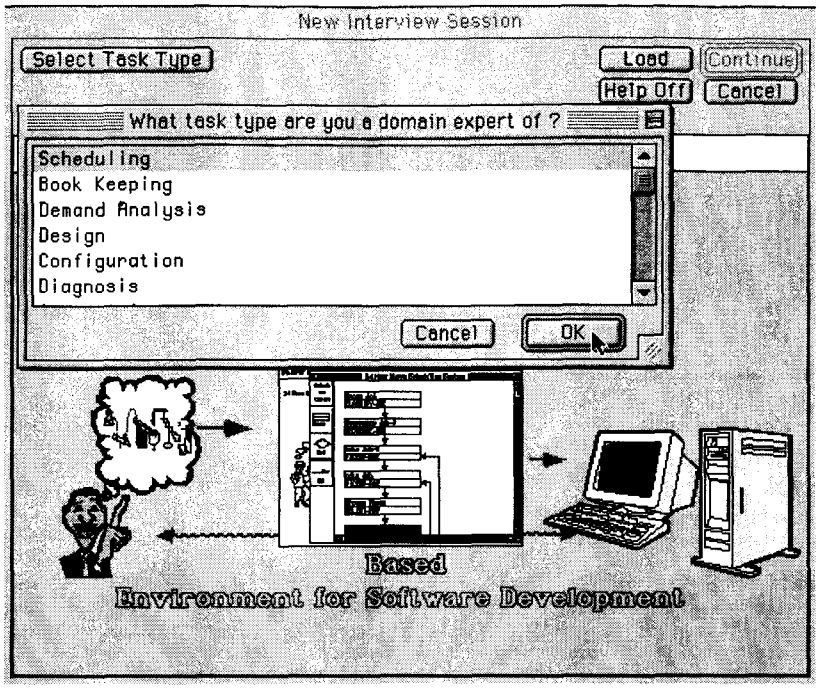


図 4.5: CLEPE の起動画面

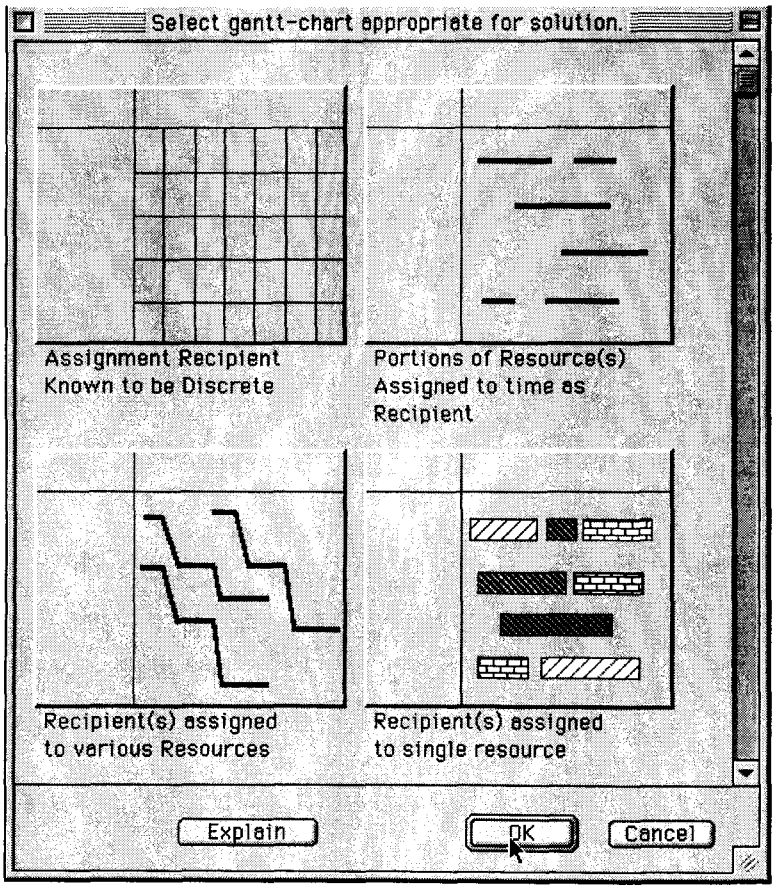


図 4.6: 解表現の選択画面

は対象とするタスクタイプ固有オントロジーをシステム内にロードする。

(2) 解表現の選択

エンドユーザが次に行う作業は，問題解決を行う際に日頃使っている解表現形式を図4.6に示したGUI上で選択することである。ここで，エンドユーザに対して解表現形式を選択してもらう理由は，日頃行っている問題解決を解の構造と対応づけて意識しているためである。

(3) ドメイン語彙の入力

言うまでもなくエンドユーザは日常的に行っている問題解決をドメインの世界の概念で意識している。ここでは，エンドユーザは，問題解決を行う際に用いるドメインの世界の語彙を，(2)で選択した解表現上で入力する(図4.7下のウインドウ参照)。この作業は，エンドユーザにとって無意識のうちにドメイン概念に対してタスクの観点からの意味を与える事になっている。つまり，各フィールドにはタスクの観点からの意味づけが与えられており(図4.7上部のウインドウ)，システムは，エンドユーザがフィールドに埋め込んだドメインの概念に対してタスクの観点からの意味づけを与えるT-ドメインオントロジーを生成する。ここでは，例えば，対象世界の概念であるnurseに対してスケジュールの受け手というタスクの世界の意味づけを与えるT-ドメインオントロジーが生成される。

(4) 問題解決のゴールの入力

対象とする問題解決のゴールを明確にするために，問題解決のゴールを選択する。ここでは，問題解決のゴールとして「看護婦の負荷を均一に保つ」ということを選択する。

(1)～(4)のプロセスが終了した段階で，エンドユーザに対して問題解決モデルを記述するための環境を提供するために必要な情報がシステム内に準備されたことになる。

4.4.2 問題解決モデルの記述

図4.8に，エンドユーザが問題解決モデルの記述を行うための環境を示している。全体は(a)問題解決モデル記述ウインドウ，(b)事例参照ウインドウ，(c)問題解決モデルを記述するための語彙(レキシカルレベルオントロジー)を提供するウインドウから成っている。各ウインドウ相互の関係は以下のようになっている。

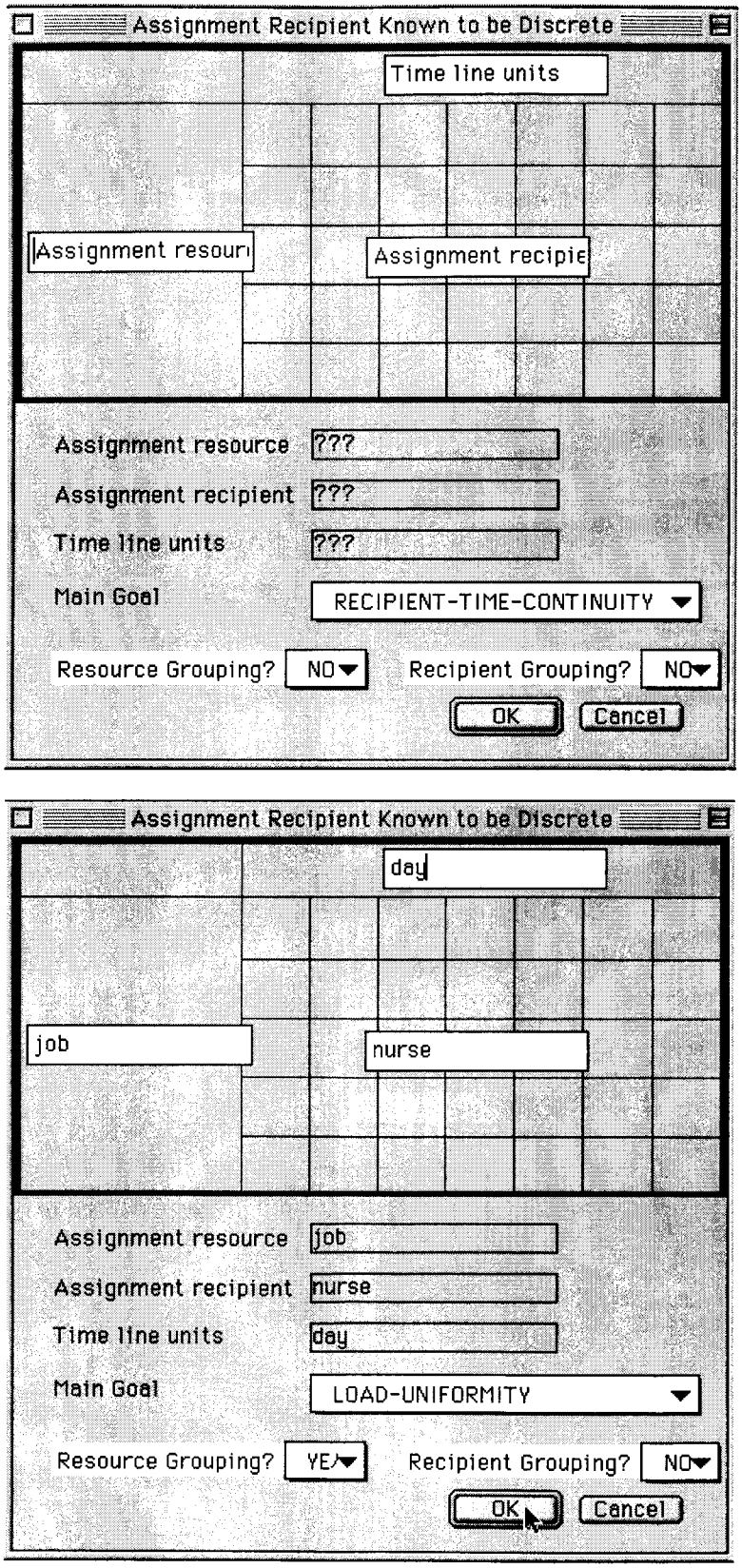


図 4.7: ドメイン語彙と問題解決のゴールの入力(上部のウインドウは初期画面)

問題解決モデルの記述は,(b)の事例参照ウインドウに表示された事例を参照しながら,(a)の問題解決モデル記述ウインドウに表示されたプロセス内に,タームを埋め込むことによって行う。各プロセスにおいて,動詞は必須であることがレキシカルレベルオントロジーの公理として定義されているため,動詞を埋め込むためのフィールドがあらかじめ表示されている。プロセスに埋め込むタームは,(c)のレキシカルオントロジーウインドウで提供される語彙の階層から,マウス操作によってタームを選択することによってなされる。

(b)に表示される事例参照ウインドウでは,白紙の状態から問題解決モデルを記述する際の負荷を軽減するために,エンドユーザが問題解決モデルを記述する際に参考となりうる事例を提供する。この際,事例検索のインデックスとして,問題の基本構成の記述で得られた解表現形式のタイプ,問題解決のゴール等の情報を用いている。一般に,これらの情報は問題解決の構造と強い相関を持っているためである。

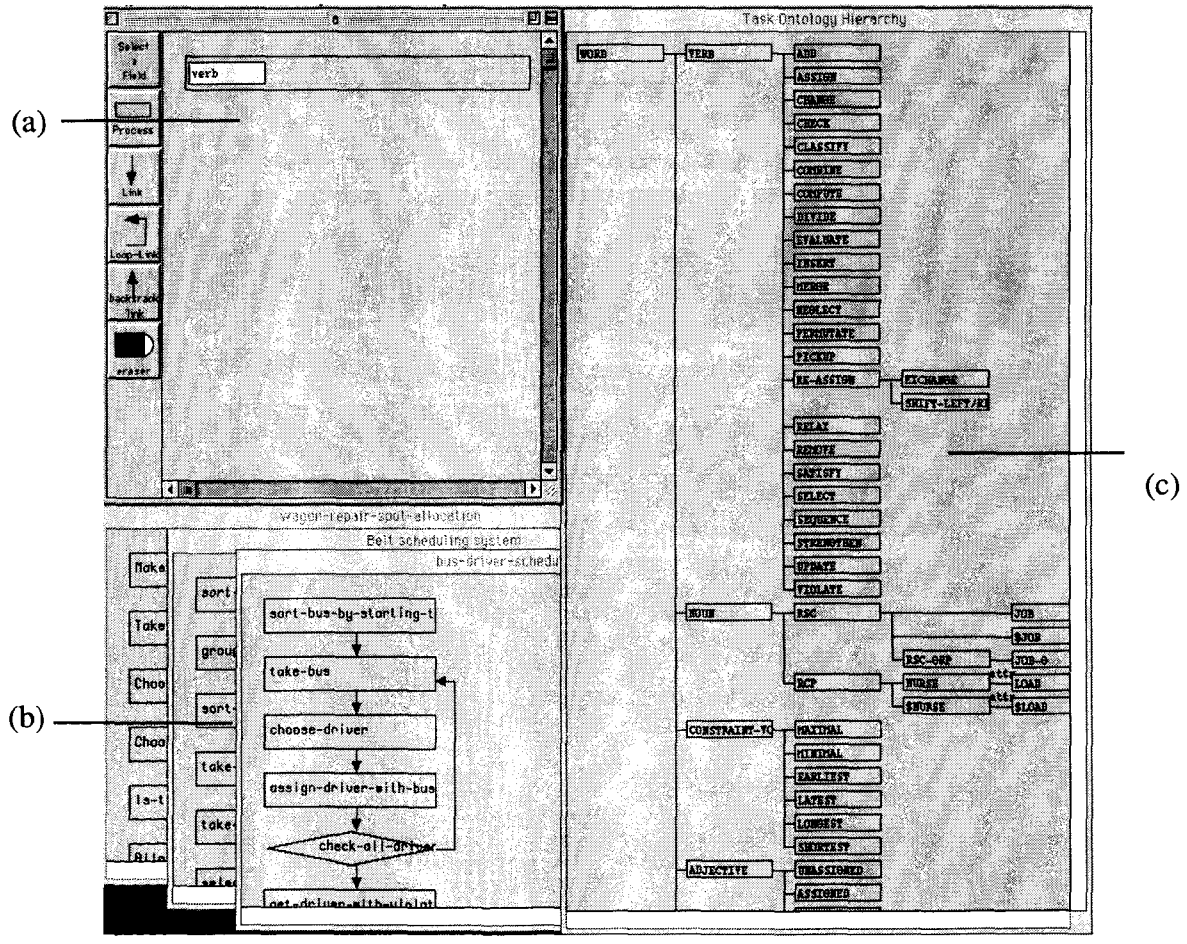


図 4.8: 問題解決モデル記述環境

また,(c)に表示された「nurse」,「job」といった対象世界に固有の概念は，問題の基本構成に基づいて生成されたT-ドメインオントロジーを表している。

問題解決モデルの記述の仕方をより詳しく示すために，図2.3のレキシカルレベルモデルを記述していく様子を順をおって述べることにする。

上述したように，プロセスには動詞フィールドがあらかじめ表示されており，このフィールドをマウスでクリックすると,このフィールドに入力可能な動詞が,(c)のウインドウ上でハイライトされる。エンドユーザはハイライトされた動詞から，適当なタームをマウスで選択しフィールドに埋め込む。ここで埋め込まれた動詞によって，プロセス内の統語則が規定される。

例えば，動詞フィールドに「Assign」を埋め込んだ際には，レキシカルレベルオントロジーのAssignが定義する統語則が図4.9のように表示される。エンドユーザはこの統語則に従ってプロセスを記述し，この記述の過程でシステムはレキシカルレベル

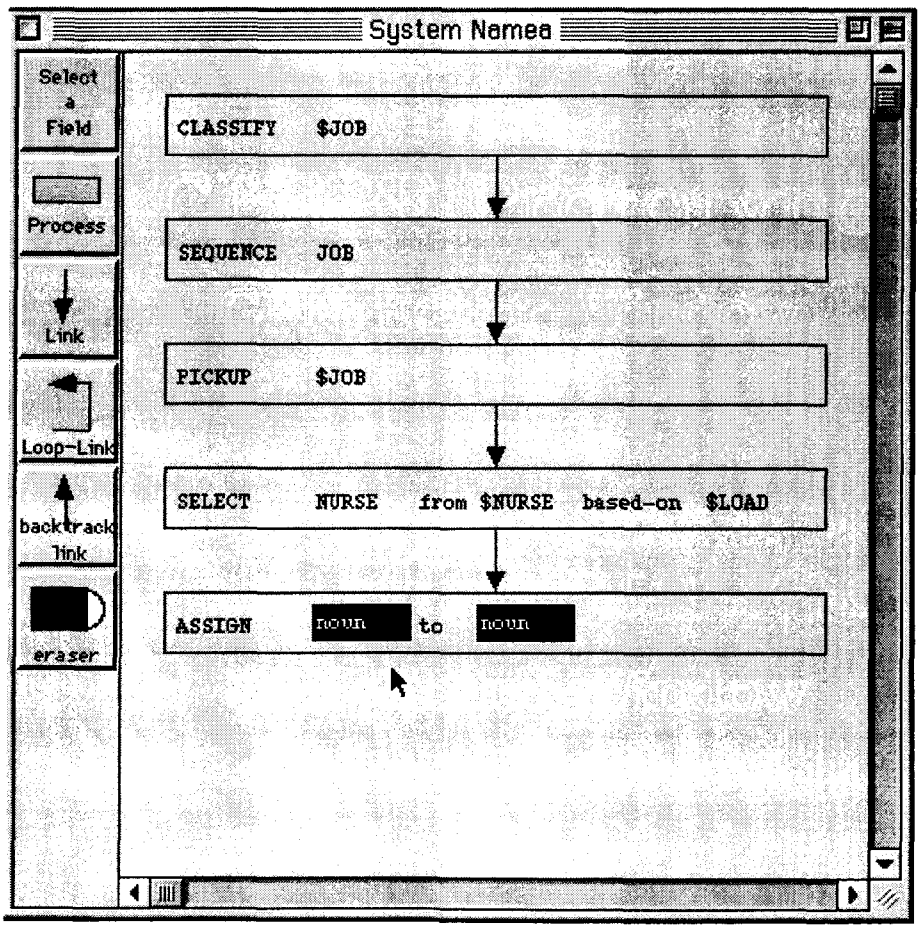


図4.9: 問題解決モデル記述画面

オントロジーに基づいて問題解決モデルの統語的な観点からの整合性を検証する。仮に，統語則への違反があった場合には，システムからエラーメッセージが出力される。

このようにして，記述した問題解決モデルが図4.10である。

図4.10のように，問題解決モデルの記述が終了した段階で，オブジェクトフロー解釈によって問題解決モデルの整合性が検証され問題解決モデルが再構成される。

図4.11にオブジェクトフロー解釈の様子を示している。この例では，「Assignプロセス」の入力として，「Selectプロセスの出力であるジョブ」が照応するというメッセージが出力されている。

問題解決オントロジーに基づいてなされる整合性検証の例を示すために，図4.10の下側のループ内にあるSelectプロセス(図2.3における「(10)のSelectプロセス」)において，「Temporary-Solution」と記述すべき所を「Initial-Solution」と誤って記述した場合を考える。

名詞Initial-Solutionが表象しうるオブジェクトは「初期状態にある解オブジェクト」ということが語用論的公理として定義されている。これに対して，今対象としている問題解決モデルにおいては，上部の二重ループが終了した段階で唯一存在する解オブジェクトは「暫定状態」にあり，Initial-Solutionとは照応しない。システムは，このような問題解決コンテクストの不整合を問題解決オントロジーに基づいて同定し，エンドユーザに対して示すことが可能になっている。図4.12に問題解決コンテクストの不整合を検出した際にシステムから出力されるメッセージが示されている。この例では，「初期解に対応するオブジェクトがない。前のループで作られた暫定解ではないか？」というメッセージが出力されている。そして，エンドユーザはこの情報に基づいて問題解決モデルを修正することができる。

最終的に，問題解決オントロジーに整合した問題解決モデルが再構成され，概念レベル実行を行うための準備が整えられた事になる。

4.4.3 概念レベル実行

概念レベル実行の様子を図4.13に示している。現段階のシステムでは，OMTでなされているビューの概念に対応するような考察は行っておらず，概念レベル実行モデルで表現 されている内容が直接的に示されている。ここでは，概念レベル実行の例と

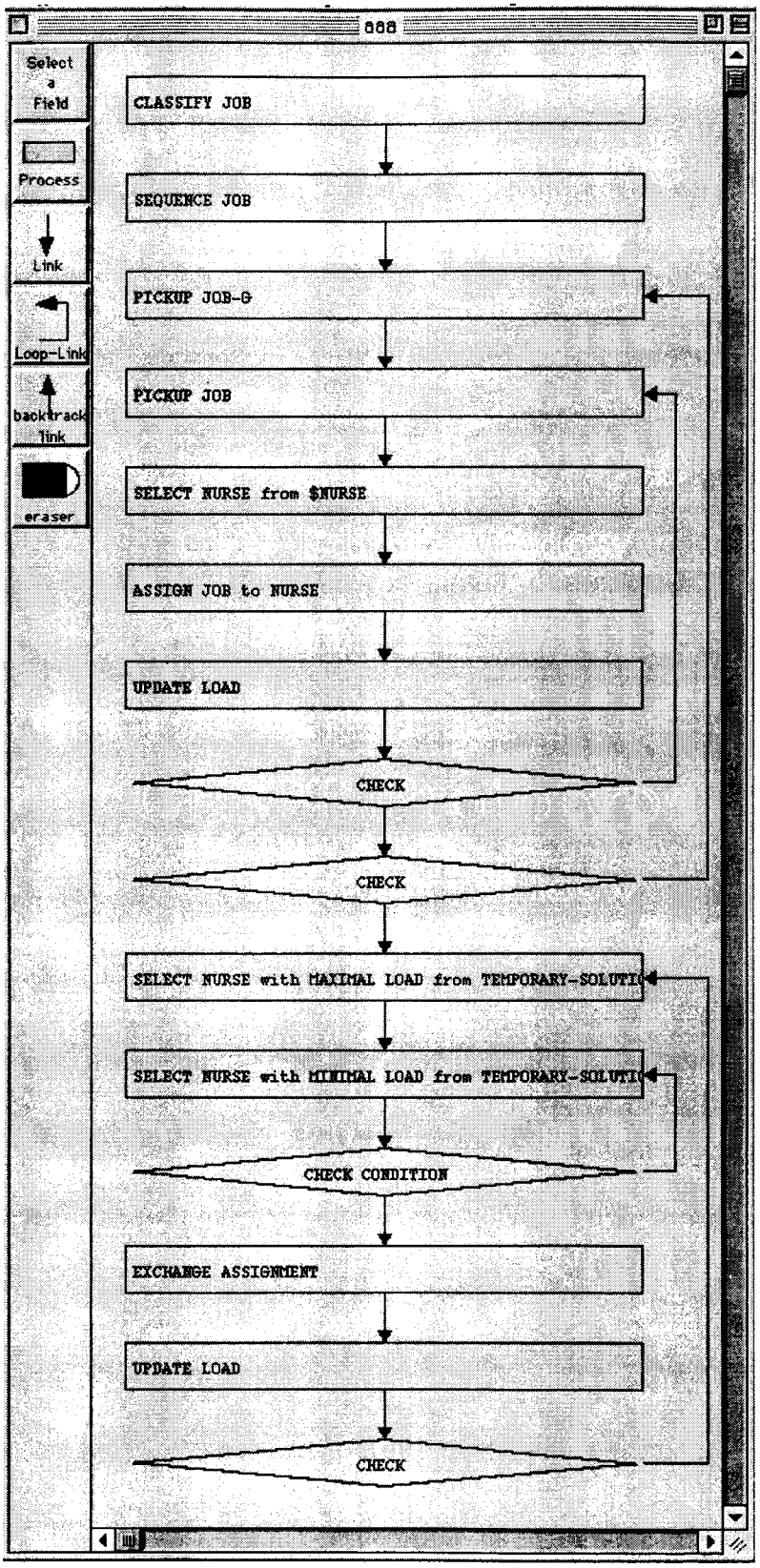


図 4.10: 問題解決モデルの記述

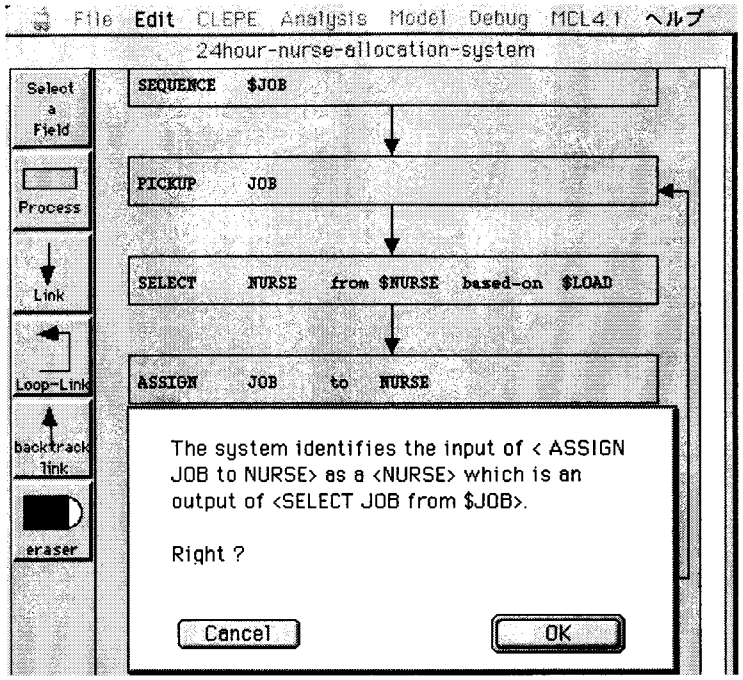


図 4.11: オブジェクトフロー解釈の実行画面(図 3.5 再掲)

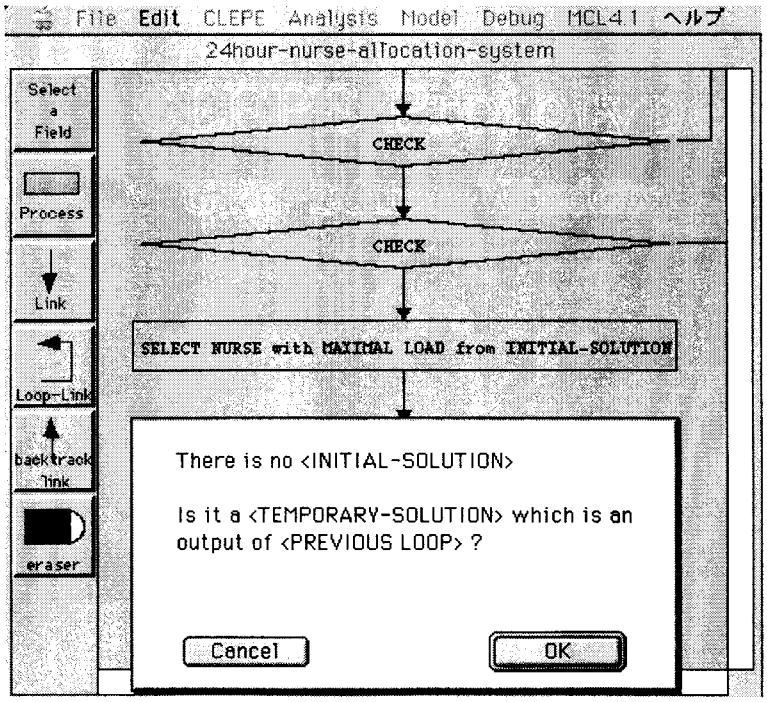


図 4.12: 問題解決オントロジーに基づく問題解決モデルの整合性の検証(図 3.6 再掲)

して，図4.3で示した概念レベル実行モデルの内容が表示されている。上下に位置する三つの画面の内，一番上の画面がループのk周目の「Assignプロセス」が処理を行った際の実行画面を表しており，中央が，同じくループのk周目において「Updateプロセス」が処理を行った際の実行画面を表している。一番下の画面はループのk+1周目において「Assignプロセス」が処理を行った際の実行画面を表している。

各画面において大きく右側にあるウインドウがタスクフローモデルを表示するウインドウであり，左側がタスクドメインモデルを表示するウインドウである。

エンドユーザは左上にあるウインドウにあるボタンをクリックすることで，問題解決のフローに沿ったシミュレーションや，各プロセス単位に実行するステップ実行を行うことができる。

ループのk周目の「Assignプロセス」の処理を表す一番上のウインドウでは，「Assignプロセス」の処理として，Pickupプロセスで「取り出されたジョブ」とSelectプロセスで「選ばれた看護婦」を入力として，それらを構成要素とする「割り付けオブジェクト」を出力するということが示されている。この情報を元にエンドユーザはオブジェクトフローに関する情報を再確認することができる。また，タスクドメインモデルでは，出力された割り付けオブジェクトが「解オブジェクトの構成要素となる」といった情報が示されている。このように，エンドユーザが問題解決モデルとして明確に記述していない暗黙的な情報についても，エンドユーザとシステムの間で明示的な合意が，概念レベル実行を通じてなされることになる。

中央のウインドウでは，「Updateプロセス」が処理を行うことによって，作用対象の負荷が修正され，その変化が対象世界において伝播する様子が示されている。ここでは処理を受けた看護婦の「負荷」の変化が，それを属性に持つ看護婦と，その看護婦を構成要素とする看護婦集合に伝播する様子が，色の変化によって示されている。

一番下のウインドウにおいては，ループのk周目における対象世界の構成に対して，k+1周目の処理を行うことによる対象世界の構成の変化を確認することで，エンドユーザはループの一回の処理における対象世界の変化を確認することができる。この例では，割り付けオブジェクト(assignment-k+1)が生成され，解の構成要素がループの一回の実行を通じて，一つずつ増えるといったことを確認することができる。

このように，概念レベル実行では，対象世界のものの変化を，それを操作するアクティビティと対応づけながら容易に確認することが可能になっている。これは，問題

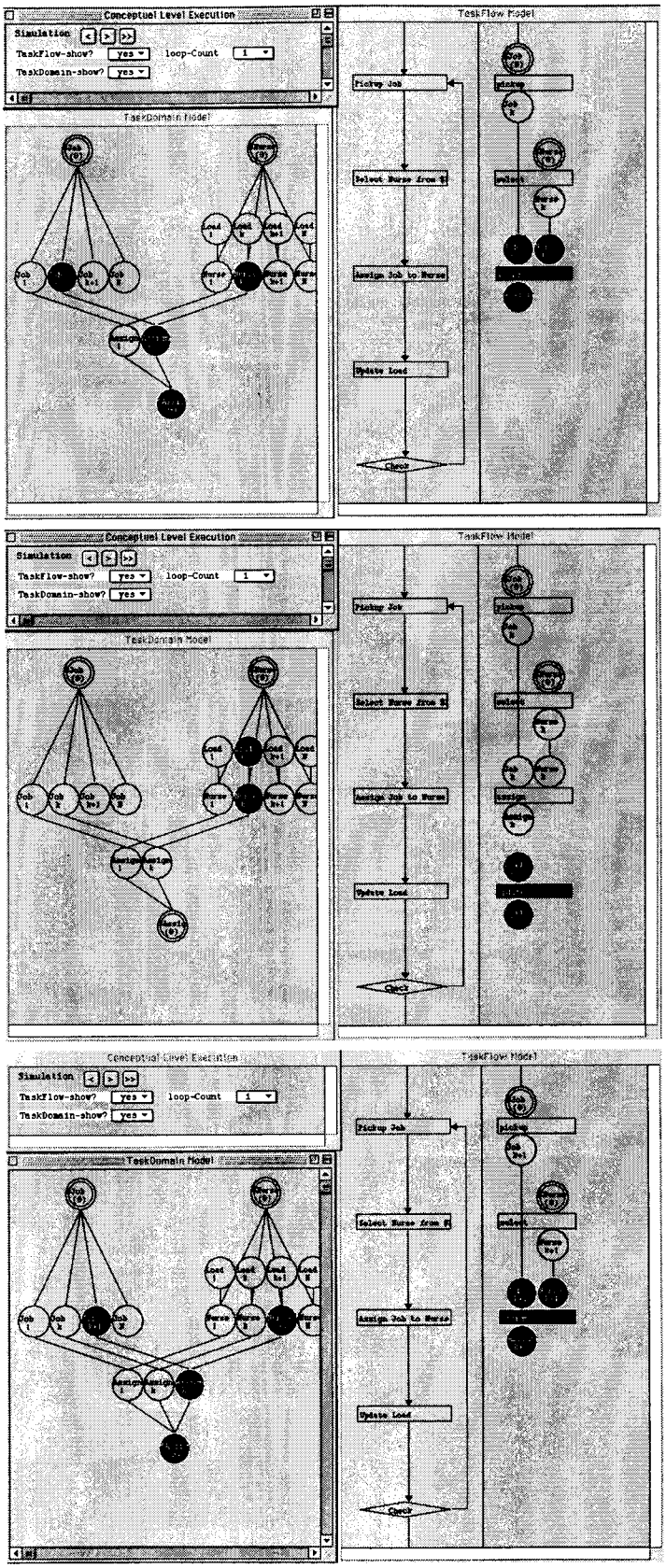


図 4.13: 概念レベル実行の画面

解決オントロジーが提供する概念レベルが，エンドユーザが日頃意識している操作のレベルと一致していることから生まれる効果である。

このような環境の下で，問題解決モデルを適宜修正しながら，エンドユーザの頭の中にある問題解決知識を問題解決オントロジーに整合した形でモデル化する。そして，問題解決モデルの振る舞いの確認が終了した段階で，記号レベルのプログラムコードとの対応づけが，システムとの対話的な作業を通じてなされ，具体的な問題解決を行う問題解決エンジンが生成される。この問題解決エンジンの生成については，本研究の前身である，MULTIS(Multi Task Interview System)プロジェクト[ティヘリノ 93]によって実現されている。本研究では，問題解決モデルの記述から，概念レベル実行モデルまでの連続性を１３のスケジューリングタスクを対象に確認した。

4.5 結言

本章では，問題解決オントロジーが定める三つの公理と，T-ドメインオントロジーが定める公理に基づいて，問題解決モデルを再構成する手法について述べた。さらに，概念レベル実行の例を示し，再構成された問題解決モデルを元にして，エンドユーザに対して提供される実行環境の機能を示した。また，本研究で実現したプロトタイプシステムの動作例を示しながら，エンドユーザが問題解決のモデル化をする際に受けることができる支援について具体的に述べた。

第5章

関連研究との比較

5.1 緒言

本章では，本研究と関連深いプロジェクトと比較し，それぞれの特徴やアプローチの違いについて考察する。

5.2 問題解決オントロジーに関する研究

5.2.1 Common KADS

Common KADS[Wielinga 92] は問題解決システムを構成するための一般的な方法論を提供しており，実際にこの方法論に基づいた数多くのシステムが構築されている。これは，KADSが提供する方法論，つまり，知識ベースシステムを構成するために，どのような概念をどういう順序で捉えたらよいかということのガイドラインを知識ベース構築者にとって受け入れられやすい形で明確にしていることから生まれる効果である。本研究では，この点に関して現段階で明確な指針を持っていない。今後，本研究で明らかにした問題解決オントロジーの構成を元にして知識ベース開発のためのガイドラインを与えるような方法論の確立を目指していく予定である。

一方で，元田[元田 94]が指摘しているように，KADS方法論では，(a)モデル化に際して提供されるプリミティブが一般的でありすぎるために，モデルの表現力が不十分，

(b)モデル化の過程を支援する枠組みがない,(c)概念モデルの整合性を検証する枠組みがなく，概念モデルを実行することができない，などの問題点が残されている。

これに対して本研究では(a)の問題を扱うための枠組みとして,問題解決全般を扱えるような枠組みを保ちながら，問題解決タイプに固有の概念を捉えることができるような問題解決オントロジー(コア問題解決オントロジーとタスクタイプ固有オントロジー)の枠組みを示した。(b)の問題解決過程のモデル化に関しては，問題解決過程の記述/実行の各段階において,問題解決オントロジーが与える問題解決タイプ固有の概念化の規約に基づいてモデル化の支援を行うことができる。(c)に関しては,オントロジーとモデルの関係を明確にすることで，問題解決モデルの問題解決オントロジーに対する整合性を検証することが可能になっている。さらに，概念レベル実行の機能によって，問題解決モデルを概念レベルで実行することが可能になっている。

もちろん,それぞれの特徴の違いはアプローチの仕方の違いから生まれる差異であり，ぞれぞれの研究が重視する目的に依存している。KADS方法論は，知識をモデル化するための汎用的な方法論を与えることが目的であり,実際この方法論に基づいて，数多くの分野で実用的なシステムが実現されている。これに対して，本研究のアプローチは一般的な枠組みの実現を視野に入れながら対象の世界の固有の原理/原則を明きらかにし，対象の固有性を捉えるための枠組みや表現を明らかにしながら，それを積み上げていくというアプローチをとっている。将来的には。二つのアプローチがつながり一つの方法論として体系化されることが理想であると考えている。

5.2.2 Protege-II

問題解決モデルのオントロジーに焦点をあてた他の関連研究として，Protege-II, KSM [Molina 96] がある。Protege-II,KSMはKADS方法論が提供する知識ベース構築のためのガイドラインに基づいた知識ベース構築支援システムである。これらのシステムでは，知識工学者が知識ベースの開発を円滑に進めるために，問題解決を知識レベルでモデル化するための環境を実現している。開発者は知識モデルを介したシステムとの対話的な作業を通じて，実行可能なシステムを円滑に開発することが出来るようになっている。

本研究で構成した問題解決オントロジーの新規性は，問題解決モデルの記述/モデルの整合性や振る舞いを問題解決オントロジーに基づいて計算機固有の概念を隠蔽し

た形で検証したり実行したりすることができることにある。この能力は，特に計算機に不慣れなエンドユーザが自らモデル化を行う際に有用である。

Protege-IIやKSMなどの他の問題解決メソッドに関する研究では，問題解決モデルと実行コードの概念的結びつきが弱く，問題解決モデルの妥当性の検証は，実行コードの振る舞いと問題解決モデルを対応づける開発者の洞察力に委ねられている。

5.2.3 TOVE

問題解決のオントロジーを対象とした研究として、本研究と最も関連が強いものは、Mark S. Foxらが進めているTOVE(TOronto Virtual Enterprise)オントロジー[Fox 93] [Gruninger 95]である。企業内のアクティビティを表現する語彙を集積し、その意味を様相論理を用いて公理化する枠組みを考えている。TOVEでは、企業内の様々なアクティビティに対して抽象的な見通しを立てることをオントロジーの能力に求めている。これは、本研究がTO/K-Cに求めている概念レベルでの実行能力に相当している。オントロジーに対する考え方は、TOVEと本研究で概ね一致しているが、問題解決モデルを記述する枠組みとして見たときの本研究の特徴は知識レベルを「レキシカルレベル」と「概念レベル」に分離し，レキシカルレベルオントロジーにエンドユーザとコンピュータの概念ギャップを埋める上で仲介的な役割を設定したところにある。

5.2.4 MULTIS

MULTIS[ティヘリノ 93]は本研究の前身であり，レキシカルレベルモデルから具体的な問題解決を行う記号レベルのプログラムコードへの連続性が実現されており本研究の基盤になっている。本稿で述べた枠組みにおいて，エンドユーザが自分の書いた問題解決知識の動作を概念レベル実行により確認した後，制約などのドメイン固有の知識に関するインタビューが行われ，記号レベルのビルディングブロックと対応づけられる。最終的に，システムは具体的な問題を解くプログラムコードを生成する。MULTISでは明らかになっていなかった知識レベルの三つの公理(統語論的公理，語用論的公理，概念レベル公理)を利用することで，オントロジーに基づくモデルの整合性の検証や，機能的な側面からは，オブジェクトフロー解釈，概念レベル実行といった機能を実現することが可能になっている。

5.3 エンドユーザプログラミングに関する研究

エンドユーザプログラミングの研究として代表的なものにFischerらが進めているDODE[Fischer 96]とCypherらが進めているKidSim[Cypher 95]がある。

DODEsは，ネットワーク設計やキッチン設計などの現実問題を対象として，領域固有の概念を取り入れたエンドユーザプログラミング環境を実現している。対象領域のオブジェクトを部品として準備し，これを用いてソフトウェアを構成するという点で本研究のアプローチと概ね一致している。しかし，問題解決を捉える際の規約となるオントロジーが明確になっておらず，モデルの整合性を検証する枠組みがない。

KidSimは，小学生に対する創発教育を目的としたシステムである。エンドユーザはプロダクションルールの形式で仮想世界を定義し，システムはそれに基づいた仮想世界のシミュレーションを行うことができる。我々のアプローチとKidSimの違いは，KidSimではシミュレーションという一般的な領域の中で対象を限定しておらず，しかも，プロダクションルールという一般的な表現の枠組みを設定しているため，対象世界を定義するための概念や属性をエンドユーザ自身が定義する必要があり，エンドユーザプログラミングを実現しているとは言い難い。一言で言えば，彼らの研究は知識を記述したり，シミュレーションする際のGUI上の表面的な研究に終始している。これに対して，本研究のアプローチでは，知識をモデル化する際のエンドユーザの特性を分析し，オブジェクトフロー解釈や概念レベル実行などのエンドユーザと計算機の間の概念ギャップを埋めるための機能を実現している。また，対象世界を限定することで，問題解決を記述するために必要な概念や属性を問題解決オントロジーとしてあらかじめ準備しておくことができ，問題解決知識を記述するにあたって，エンドユーザが概念を定義するという作業プロセスに煩わされることがない。

5.4 ソフトウェア工学に関する研究

5.4.1 OMT

OMT(Object Modeling Technique)は，要求分析，設計，実装などソフトウェアのライフサイクル全体に渡って連続性のある方法論を提供することを目的としている

[Rumbaugh 91]。実際，これまでに対象世界をモデリングするための汎用の方法論が提供され，数多くのシステムがこの方法論に基づいて実現されている。本研究との直感的なアナロジーをとれば，設計プロセス，実装プロセスで提供される方法論は，それぞれ問題解決オントロジーに基づいてレキシカルレベルモデルを構成する作業と，概念レベル実行モデルを記号レベルのコードと結びつける作業[1]に相当している。

一方で，スケーラビリティのある汎用性の高い方法論を打ち立てている負の側面として，対象の固有性を捉えるための枠組みが本研究と比べて相対的に弱く，モデルの整合性を，対象固有の原理/原則に基づいて検証することが困難になっている。また，本研究で示した概念レベル実行に対応する枠組みがない。

しかしながら，これは両者の目的の違いに起因するものであり，優劣を単純に比較できる性質の議論ではなく，相互に補完する点を模索すべき問題である。

5.4.2 コンポーネントウェア

コンポーネントウェアに関する研究は，再利用可能なソフトウェア部品を準備し，それを組み合わせてソフトウェアを開発しようという壮大なアプローチである。これが実現されれば，ソフトウェア開発の生産性の向上と開発にかかるコストを大幅に削減するものとして期待されている。

本研究とコンポーネントウェアに関する研究の共通性は，ソフトウェアの共有・再利用に貢献する点にある。差異は，本研究が対象世界のモデリングから，最終的なコード生成までの連続性を前提としているのに対して，コンポーネントウェアに関する研究ではその様な前提をおいていない点にある。

もちろん，この問題も対象とする問題の大きさやスケーラビリティを考慮に入れれば，双方の短所・長所といった単純なものではなく研究領域固有の異なる制約に従って研究を進めていることにより必然的に生じた差異である。

---

[1] これは，本研究の前身であるMULTISプロジェクトによって実現されている。

5.5 結言

本章では,(1)問題解決オントロジーに関する研究,(2)エンドユーザプログラミングに関する研究,(3)ソフトウェア工学に関する研究と比較し，それぞれの特徴，研究目的，アプローチの違いについて検討した。

他の関連プロジェクトと比較した本研究の特徴を端的に言えば,(1)問題解決オントロジーに基づいて問題解決モデルの整合性を検証する枠組みを実現していること,(2)問題解決モデルを概念レベルで実行する能力を備えていることといえる。また，オントロジーとモデルの階層関係を明確にし，オントロジーに対するモデルの整合性を検証する枠組みを備えている点も本研究の特徴である。

第6章

結論

本章では，本研究で得られた成果を総括し，今後の展望について述べる。

第2章では，問題解決オントロジーの基本構成と問題解決オントロジーが定める公理を明らかにした。

問題解決オントロジーを構成するにあたり，

I. 問題解決モデルを容易に記述できること。

II. 問題解決モデルの計算論的な意味が明確になること。

III. スケジューリングや診断などのタスクタイプ固有性に対応できること。

IV. 問題解決全般に広く適用できること。

を設計方針とした。その上で，I., II.の要求を満足するためのオントロジーとして，レキシカルレベルオントロジーと概念レベルオントロジーを設定し，III., IV.の要求を満足するためのオントロジーとして，タスクタイプ固有オントロジーとコア問題解決オントロジーを構成した。

さらに，問題解決オントロジーが定める三つの公理を明らかにした。そして，これを本研究で開発した問題解決オントロジー記述言語TOLを用いて例示した。問題解決オントロジーが定める三つの公理は，統語論的公理が問題解決モデルを記述するための基本的枠組みを定め，概念レベル公理が問題解決モデルの計算論的な意味を明確にする役割を担っている。語用論的公理は問題解決モデルの記述と意味を対応づける役割を担っている。この三つの公理に加えて，T-ドメインオントロジーの公理によって，ドメインを指向した問題解決モデルの記述から，問題解決の意味論までの連続性が実現されている。

第3章では，問題解決オントロジーを基盤とした概念レベルプログラミング環境CLEPEの全体像について述べ，CLEPEの二つの環境において問題解決オントロジーが担う役割を明らかにした。

問題解決オントロジー構築支援環境では，コア問題解決オントロジーが，タスクタイプ固有オントロジーオーサに対するモデル化の基盤を与える役割を担っている。システムは，タスクタイプ固有オントロジーのコア問題解決オントロジーに対する整合性を検証することが可能になっている。

問題解決モデル記述環境を実現するにあたっては，

I. 記述容易性:エンドユーザが日頃意識している概念を中心に自分自身の知識を外化しやすいような平易な枠組みを設定すること。
II. 不完全知識の許容性:エンドユーザが日頃意識していない概念であるが，計算機での実行を考える際に必要な概念について，エンドユーザに対して記述をすることを求めるのではなく，計算機側が読みとる能力を備えること。
III. 理解容易性:問題解決モデルの振る舞いをエンドユーザにとってわかりやすい形で示すこと。
IV. 計算セマンティクスまでの連続性:具体的な問題解決を行うことができる問題解決システムを生成できること。

を基本思想とした。そして，(1)問題解決モデルの記述，(2)問題解決モデルの実行/修正の過程で問題解決オントロジーに基づいてなされる支援内容を明らかにした。特に，オブジェクトフロー解釈，問題解決モデルの整合性の検証，概念レベル実行は，問題解決オントロジーを基礎とすることで実現可能な支援であり，問題解決知識のモデル化を支援するに際しての問題解決オントロジーの有用性を明らかにした。

第4章では，問題解決モデルを再構成するための仕組みを示しながら，問題解決オントロジーの利用形態について述べた。そして，概念レベル実行モデルに基づいて実現される概念レベル実行について述べた。CLEPEが提供する概念レベル実行は，問題解決過程のモデル化を支援する上でなくてはならない機能である。問題解決モデルの記述と意味内容の対応関係が明らかにされた問題解決オントロジーに基づいて，問題解決モデルの意味内容がモデルの記述と同一のレベルで示される。エンドユーザは，ここで提供される情報に基づいて，問題解決モデルの振る舞いを確認することが可能になっている。さらに，本研究で作成したプロトタイプシステムの動作を示し，(1)問題解決モデルの記述，

(2)問題解決モデルの実行/修正の各過程で問題解決オントロジーに基づいてなされる支援内容を通じて，3章で述べた基本思想が実現されていることを示した。

以上のように本研究では，問題解決過程をモデル化するための問題解決オントロジーの構成に関する包括的な議論を行い，問題解決オントロジーが定める公理を明らかにした。そして，問題解決オントロジーを基盤とした概念レベルプログラミング環境CLEPEを提案し，そこで実現された機能的な側面を明らかにした。本研究で述べた枠組みは，Macintosh 上で Macintosh Common Lisp を用いて実現されている。

本研究では，問題解決モデルの記述に関して，主に「知識」を「モデル化」するための基盤として問題解決オントロジーを構築した。これによって，エンドユーザが捉えている問題解決の過程と，システムの内部的なモデルとのギャップを埋めることが原理的に可能になっている。しかし，エンドユーザにとってより重要な問題として，問題解決モデルの視覚的な表現の受け入れ易さについてオントロジーは直接的な解を与えていない。現状では，問題解決オントロジーによって「何を」「どう」表現すべきかということの基礎が与えられた段階にある。この基礎の上に，エンドユーザにとって受け入れやすい視覚的な表現(GUI)に関する検討をタスクタイプ毎に行っていく予定である。

問題解決オントロジーの評価に関しては大きく２つの段階が必要になる。第一は、ある特定のタスクタイプに対する有効性であり、第二は、様々なタスクに対する適用可能性・有効性である。本研究では、前者についてスケジューリングタスクタイプを対象として検討した。１３のスケジューリングタスクにおいて、本研究で作成したプロトタイプシステム上で問題解決モデルの記述から概念レベル実行モデルまでの連続性が保証されることを確認した。またスケジューリングタスクの専門家によって，問題解決オントロジーが提供する語彙の十分性と記述容易性がMULTISプロジェクトにおいて確認されている。

第二段階の評価で対象となる適用可能性の問題に対しては、本研究で示した問題解決オントロジーの垂直方向の構成(コア問題解決オントロジーと，タスクタイプ固有オントロジー)が有効に働くと考えられる。本システムを他のタスクタイプに対して適用しながら，問題解決オントロジーの有効性についてより徹底した検証を行う予定である。

本研究では，エンドユーザが問題解決システムを単独で開発する際の作業を支援する環境を実現し，その基礎としての問題解決オントロジーの有効性を示した。しかし，工業製品やソフトウェア開発サイクルでは，異なった役割を持った複数の人々の協調作業

が主流となり，より複雑な形態をしているのが一般的である。そのような製品開発のライフサイクルを支える基盤を整えるためには,開発に関与するエージェント間での異なったオントロジーの共有，様々なレベルでの情報伝達の基盤の実現などの課題がある。今後そのような課題を念頭におきながら，本研究の成果をスケールアップし，ソフトウェア開発における創意と工夫を継承/蓄積するための基盤の確立を目指していきたい。

謝辞

本研究の全過程を通じて，懇切なる御指導，御鞭撻を賜りました大阪大学産業科学研究所 溝口理一郎教授に衷心より深謝致します。

本研究に関して貴重な御教示を頂きました大阪大学工学部電子工学科 西原浩教授，ならびに大阪大学産業科学研究所 元田浩教授に心から感謝致します。

大学院の後期課程において御指導と御教授を賜りました大阪大学工学部電子工学科，濱口智尋教授，吉野勝美教授，尾浦憲治郎教授，春名正光教授，森田清三教授に厚くお礼申し上げます。

本研究の遂行にあたり直接御指導，御討論を頂きました大阪大学産業科学研究所 池田満助教授に深く感謝します。

本研究の全般にわたり貴重な御討論，御助言を頂きました兵庫大学経済情報学部角所収教授，立命館大学理工学部 山下洋一助教授，大阪大学産業科学研究所 来村徳信助手に厚く感謝の意を表します。

貴重な御討論を頂いた溝口研究室の諸氏，そして，様々な面でお世話になった溝口研究室の秘書の方々に感謝致します。

最後に，終始温かく見守り支えてくれた母と妹に感謝申し上げます。

参考文献

[Cypher 95] Cypher, A., and Smith, D. C.,:KidSim: End user programming of simulations, Proc. of Computer Human Interaction'95, pp. 27-34.(1995).

[Debelis 95] DeBellis, M. , :User-centric software engineering,IEEE EXPERT, Vol.10, No.10, p.p.34-41. (1995).

[Fischer 96] Fischer, G.,:Seeding, evolutionary growth and receeding: constructing, capturing and evolving knowledge in domain-oriented environment, Domain Knowledge for Interface System Design, London: Chapman&Hall, pp.1-16 (1996).

[Fox 93] Fox,M.S., Chionglo, J., Fadel, F.,: A Common-Sense Model of the Enterprise, Proc. of the Industrial Engineering Research Conference (1993)

[Gruber 92] Gruber, T.: A translation approach to portable ontology specifications, Proc. of JKAW' 92, pp.89-108, (1992).

[Gruber 94] Gruber, T.: An ontology for engineering mathematics, Proc. of Comparison of implemented ontology, ECAI'94 Workshop, W13, pp.93-104, (1994).

[Gruninger 94] Gruninger, R. and M. Fox: The design and evaluation of ontologies for enterprise engineering, Proc. of Comparison of implemented ontology, ECAI'94 Wrokshop, W13, pp.105-128, 1994

[Gruninger 95] Gruninger, M. and Mark Fox, M.S.,: Methodology for the Design and Evalu-

ation of Ontologies, Proc. of IJCAI-95 Workshop on Basic Ontological Issues in Knwoledge Sharing(1995).

[Guarino 95] Guarino, N. and P. Giaretta: Ontologies and knowledge bases towards a terminological clarification, Proc. of KB¥&KS'95, pp.25-32, (1995).

[Heust 97] Heust, G.V., Schreiber, A.T., Wielinga, B.J.,: Using explicit ontologies in KBS development, Journal of Human-Computer Studies, Vol.45, pp.183-292, (1997).

[Hori 94] Hori,M., and Nakamura, Y.,: Reformulation of problem-solving knowledge via a task-generic level, Proc. of Third Japanese Knowledge Acquisition Workshop for Knowledge-Based Systems: JKAW'94, pp.3-15, (1994).

[Ikeda 97] Ikeda, M, Seta, K, Mizoguchi, R.,: Task Ontology Makes It Easier To Use Authoring Tools, Proc of IJCAI-97, pp.342-347, Nagoya, Japan (1997).

[来村 97] 来村徳信, 笹島宗彦, 池田満, 他：モデルに基づく問題解決のための流体と時間のオントロジーの構築とその評価, 人工知能学会誌, Vol. 12, No. 1, pp.132-143, (1997).

[Lenat 90] Lenat, D. and Guha, R.V.,: Building Large Knowledge-Based Systems,Addison-Wesley, Reading, MA (1990).

[Mizoguchi 93] Mizoguchi, R.: Knowledge acquisition and ontology, Proc. of the KB & KS' 93, Tokyo, pp.121-128, (1993).

[溝口 97] 溝口理一郎, 池田 満: 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).

[Molina 96] Molina, M., Shahar Y., Cuena, J., Musen, M.A.,: Structure of Problem-Solving Methods for Real-time Decision Support Solving Methods: Modeling Approaches Using Protege-II and KSM, Proc. of KAW-96, (1996).

[元田 94] 元田浩: 知識ベースの再利用へのアプローチ, 人工知能学会誌, Vol.9, No.1, pp.10-16 (1994).

[Rumbaugh 91] James Raumbaugh et al: Object-Oriented Moding and Design, Pretice Hall.

Inc., (1991)

[笹島 96] 笹島宗彦, 来村徳信, 池田満, 溝口理一郎：振る舞いと機能のオントロジーに基づく機能モデル表現言語FBRLの開発, 人工知能学会誌, Vol.11, No.3,pp.420-431, (1996).

[Skuce 95] Skuce,D. ed.,: Proc. of IJCAI95 Workshopon Basic Ontological Issues in Knwoledge Sharing (1995).

[Sowa 95] Sowa, J.: Distinctions, combinations, and constraints: Proc. of IJCAIWorkshop on Basic Ontological Issues in Knowledge Sharing, Motreal, (1995).

[高岡 95] 高岡良行, 広部健治, 溝口理一郎：再利用可能知識ベースの構築ー変電所事故復旧問題を例にしてー, 人工知能学会誌, Vol.10, No.5, pp.786-797, 1995.

[ティヘリノ 93] ティヘリノ・ジュリ・A, 池田　満, 北橋忠宏, 溝口理一郎: タスクオントロジーと知識再利用に基づくエキスパートシステム構築方法論ータスク解析インタビューシステム MULTIS の基本思想, 人工知能学会誌, Vol.8, No.4, pp.476-487(1993).

[Vanwelkenhuysen 95] Vanwelkenhuysen, J. and R. Mizoguchi: Workplace-Adapted Behaviors: Lessons Learned for Knowledge Reuse, Proc. of KB¥&KS '95, pp.270-280, (1995).

[Wielinga 92] Wielinga, B., Schereiber, A. T., and Breuker, J. A.,: KADS: A modeling approach to knowledge engineering, Knowledge acquisition, 4(1), pp.5-53 (1992)

[Wielinga 93] Wielinga, B. et al.: Reusable and sharable knowledge bases: A European perspective, Proc. of KB¥&KS'93, Tokyo, pp.103-115, 1993.