JAIST
JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY
1990

北陸先端科学技術大学院大学
21世紀COEプログラム　検証進化可能電子社会
Verifiable and Evolvable e-Society

COE Research Monograph Series, Vol. 2

法令工学の提案

片山　卓也 編

2007年9月

法令工学の提案

片山卓也 編

まえがき

法令工学は，法令文書の作成や変更，法令実働化情報システムの構築を系統的に行うため，人工知能，言語処理，ソフトウェア工学の研究成果を使おうとする工学的アプローチであり，21 世紀 COE プログラム「検証進化可能電子社会」の主要研究課題の一つである．安心な電子社会の実現には，電子社会の仕様書である法令を適切に作成し，それを施行する情報システムを正しく構築しなければならない．また，法令の改定に対しては，関係法令への変更伝播を整合的に行い，それを情報システムへの変更に矛盾なくつなげる必要がある．法令工学は，このような問題を工学的に解決することめざして，本 COE で世界で初めて提案されたものである．

現在，企業活動における法令尊守などが大きな社会問題として取り上げられているが，組織における規則の作成や尊守機構の設計なども法令工学の範疇に入ると考えられ，今後訪れる本格的な電子社会時代において，安心で公正な社会や組織の設計や実現に，法令工学はその基本技術を提供するものであると考えている．

本書は，本 COE における法令工学の研究活動を紹介するために書かれたものであるが，各章の著者は以下のようである．

第 1 章 片山卓也

第 2 章 島津　明

第 3 章 東条　敏

第 4 章 二木厚吉，緒方和博，有本泰仁

第 5 章 落水浩一郎，早坂　良

21 世紀 COE「検証進化可能電子社会」
拠点リーダ　片山卓也

目 次

図 目 次

表 目 次

第1章　法令工学のめざすもの

1.1　社会の仕様としての法律

我々の社会は，非常に多数の相互に関連した法律や法令により規定されている．それらは社会の組織や構造，目標や目的を記述すると同時に，組織における活動や手続きを定めており，われわれはその法令に従って社会活動を行っている．したがって，法令が適切に作られており，それが社会のなかで正しく運用されていることは，社会の安心性の基本である．これは，社会というシステムを考えたとき，法令がこのシステムの仕様の役割を果たしていることを意味している．したがって，その仕様が適切に書かれており，それが実際に正しく運用されていることは，我々が安心で質の高い社会生活を送る上の基本的要件ということになる．

特に，今後本格化する電子社会では，政治，経済，司法，行政，医療，教育など社会生活の基幹部分が電子化され，電子社会情報システムとして実現されることになる．この情報システムは，社会の仕様としての法令にもとづいて実現されたものであり，我々はそれにより利便性を享受する一方，仕様や情報システムのもつ欠陥や不完全さなどから，不利益をこうむり，場合によっては，生命や財産の危機に直面する可能性もある，電子社会時代における法令は，電子社会情報システムの仕様書としての側面も加わり，一層重要な役割を持つことになる．

1.2　法令工学の目的

法令工学は，法令がその制定目的にそって適切に作られ，論理的矛盾や文書的問題がなく，関連法令との整合性がとれていることを検査・検証し，法令の改定に対しては，矛盾なく変更や追加削除が行われることを情報科学的手法によって支援することを目的とする学問分野である．それと同時に，法令によって定められた内容を，情報処理システム－法令実動化システム－として実現する際のシステム設計を，法令の構造にもとづいて系統的に行う方法論の展開もその目的としている．

社会の変化に対応して新しい法令が日々作られていると同時に，既存の法令に対する変更や修正が頻繁に行われており，これに対応して，法令実動化情報シス

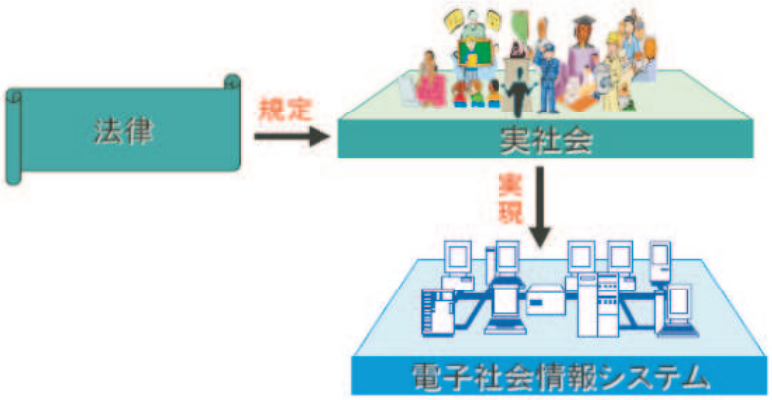

図 1.1: 電子社会の仕様としての法律

テムの開発と保守作業が多大なコストをかけて行われている．現在，それらは法務実務者や情報システム開発者の努力と献身によって支えられているが，法令工学は彼らの活動を情報処理技術によって支援することを目指している．

1.3　法令工学の概要

法令工学では，主に，

- 法令文書の作成，解析，保守などを系統的に行う方法論
- 法令を実働化する情報システムの開発を系統的に行う方法論

の 2 つの問題を扱う．

1.3.1　法令文書の系統的作成・解析・保守方法論

法令工学のねらいの一つは,「法令は社会を動かすソフトウェアである」と言う立場に立って，ソフトウェア工学的手法を法令に対して適用することを試みることである．プログラムやソフトウェアの作成においては，問題の分析，設計，プログラミングなどは，それらを支援するための開発境環を利用して行われるのが

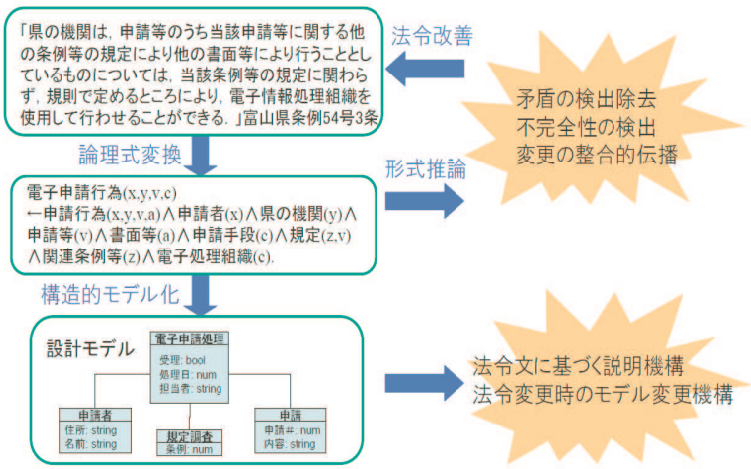

図 1.2: 法令工学のめざすもの

普通である．また，構築されたプログラムに対しては，テストや，場合によっては，定理証明やモデル検査などによる形式検証を行い，構築されるシステムの信頼性を確保している．さらに，構成管理ツールや版管理ツールなどの備わったレポジトリを使って，生成物管理を行い，開発過程において生成られる文書やプログラムの整合的管理が行われるのが通常である．

このような方法論を法令に適用し，その作成や解析，変更作業を支援する計算機支援環境を構築することが，法令工学の大きな目的である．勿論，プログラムやソフトウェアは，その構造や動作が明確に定義され，それによりこのよう方法論が可能であるのに対し，人間が理解し運用するように作られている法令においては，必ずしも計算機を利用した解析が可能なものとは限らない．しかしながら，法令文自身は用語の定義が注意深くなされており，文章の構造なども十分な推敲を受けている場合が多い．また，情報システムとして実現されているような行政手続きなどでは，その内容は十分な明確さを持っており，論理式などによる形式表現が可能であるであると考えられている．したがって，最近の言語処理や人工知能の技術を用いることにより，法令文から自動的あるいは半自動的に論理式や形式表現を導き出し，それに対してソフトウェア工学的手法を適用することにより，質の高い法令文書の作成や維持のために計算機を利用できる可能性が高いと思われる．

1.3.2 法令実働化情報システム開発方法論

法令の内容を実働化するシステムは，たとえば税金や年金の計算や徴収システム，運転免許管理システムなどさまざまなシステムが作成され，運用されている．これらのシステムを，法令実動化情報システム（Law Enforcing Information System, LEIS）とよぶことにするが，これらのシステムは，基本的には，その根拠となる法令によってその処理内容が定められる．法令工学はこれらのシステムの開発に，例えば，システムのアーキテクチャなどに，明確な設計指針をあたえるものでなければならない．また，システムの処理内容に対する疑問や質問に対して，関連する法令に即した説明を可能にするものであることが望まれる．たとえば,「なぜ，年金額がこのように計算されるのか？」という質問に対しては，対応する年金法に即して回答することが可能であれば，情報処理システムにアカウンタビリティを具備することができるようになる．また，法令の変更に対応して行われる情報システムの変更は，一般に大きなコストを要するが，この変更作業を系統的に行う方法に道を拓く可能性がある．

1.4 法令工学と法情報科学

法律に関する情報科学的研究は，従来から活発に行われてきたが，その中心は，高度に法的な推論や法解釈の人工知能的研究が主であった．たとえば，弁護士と検事の間の論理の衝突に代表される，立場の異なる当事者間の論争の論理学的あるいは人工知能的定式化などである．

これに対して，法令工学は，法令文書にもとめられる無矛盾性や完全性などの計算機による支援，たとえば，法令で定められるべきことが明確に規定されているか，場合の漏れはないかなどを検査し，法令文書の作成や保守を科学的に行い，また，法令を実働化している情報システムを設計する技術を研究，開発するためのものである．社会の中での我々の社会活動を計算機の上で動作するプログラムの振る舞いに例えると，これは，プログラムやソフトウェアを正しく作ることための技術に対応させることができる．

1.5 法令工学の研究課題

法令工学の目指すものについては，これまでに述べてきたが，以下では，その研究課題について考える．次の５つが代表的な研究課題であろう．

1. 法令文書の作成管理環境

2. 法令文書の言語解析
3. 法令文書の論理推論技術
4. 法令対象ドメインの形式記述と検証
5. 法令実働化情報システム

1.5.1 法令文書の作成管理環境

現在，法令文書の作成には一般の文書作成ソフトウェア（ワープロ）や法令文書データベースが利用されている．しかしながら，法令の構造や法令特有な言葉づかいなど適切に利用することにより，より高度な法令文の作成の計算機支援が可能になると思われる．また，法令の変更時に行わなければならいない関連法令への変更伝播は，非常に手間と経験を要する作業であるが，用語や概念辞書などを備えることにより，その一部を計算機に行わせることも可能であろう．法令変更時の，いわゆる，溶け込ませなどの自動化の研究はすでに行われているが，任意の時点での法令を機械的に構成する機能をサポートする構成管理や版管理，多数の利用者が法令データベースを同時に操作する際のロック機能など，ソフトウェアレポジトリで培われた技術が利用できる局面は多い．

- 法令文書の構造や法律用語の意味に即した文書処理が可能な法令文書作成エディタ
- 用語の定義参照関係や法令文の参照関係の検査，解析機能
- 用語の概念階層関係（オントロジー）などを利用した関連法令の検索機能
- 法令の変更や部品化を容易とする版管理，構成管理

1.5.2 法令文書の言語解析

法令文書の言語処理の目的は，法令文書からその法令構造を抽出することである．法令構造の表現としては，論理式や適切にタグ付された XML 文書など種々のものが考えられる．抽出された法令構造は，推論や法令実動化情報システム開発の基礎となるものである．法令工学は，現実の法令を対象としており，言語処理・解析のスケーラビリティは必須の条件であり，この意味で自動的あるいは半自動的処理はきわめて重要である．自然言語解析や翻訳に関するこれまでの長い研究にもかかわらず，一般の文書の自動処理はいまだ困難であるのが現状であるが，法令文書に関しては，自動あるいは半自動解析が成功する可能性は高い．それは，法

令文が吟味された用語と構文を用いて注意深く書かれているからである．法令文書の言語解析で抽出された法令構造は，形式推論などの基礎になるものであるが，一方，これをもとにしてより可読性の高い，自然言語文を作り出すことも可能であろう．また，完全な言語解析が不可能な場合でも，用語の出現順序などを利用した言語推論を行える可能性もある．

- 法令文中の法令要素，法令構造の特徴付けと同定
- 法令文の論理・形式表現への変換方式
- 可読性の高い法令記述言語の設計と既存法令の翻訳
- 法令文書ベースからの言語推論

1.5.3 法令文書の論理推論技術

形式論理によって表現された法令文書から，定理証明などの推論技術を使って法令文書の解析を行う．法令文書の持つべき性質としては，法令的な矛盾や不完全さがないことや，他の法令や上位法令との整合性がとれていることなどあげられるが，これらを形式論理に対する推論技術によって解析し，矛盾や不完全の除去する技術が研究課題である．法令で記述される事柄を最も自然に表現できる論理体系とそこでの推論アルゴリズムが基本であるが，現実規模の法令文に対応するための技術の開発が特に重要である．定理証明やプログラムの形式検証で培われた推論技術が利用可能であるが，現実の法令文に適用するには，法令文では陽に表現されていない事実や未定義・一般用語の扱いや，多量の論理式を処理するためのメカニズムなどの解決が必要である．

- 法令表現のための形式論理体系と形式推論
- 法的整合性や完全性の論理学的定式化
- 大規模法令論理式ベースの効率的整合性・完全性解析アルゴリズム
- 法令論理式推論のためのオントロジー

1.5.4 法令対象ドメインの形式記述と検証

法令が対象とする社会活動や社会システムを，その本来的意味において捉えたものを法令対象ドメインという.（たとえば，電子商取引ドメイン，病院ドメイン，

企業活動コンプライアンスドメイン.）これは，ソフトウェア工学におけるドメインの概念を法令に適用したのものである．法令で記述されるべき対象を，その法令とは独立に把握し，記述することにより，法令記述の適切性をドメイン記述にもとづいて判断や検証することが可能となる．あるいは，ドメイン上で抽象的な実行を行うことにより，そのドメインに関する性質やふるまいを予め理解した上で，適切は法令を作ることが可能になる．ドメイン記述の方法としてはいろいろあるが，高い数学的解析性や実行可能性を持つ形式記述や，ソフトウェア開発で標準的なオブジェクト指向概念に基づくものなどが考えられる．

- 法令の記述対象であるドメインの数学的記述法，そのための形式体系
- 法令対象ドメインの性質の形式検証
- ドメインの形式的記述を利用した法令の正しさや適切さの形式検証
- オブジェクト指向概念によるドメイン記述

1.5.5 法令実働化情報システム

法令実動化情報システムは，多くの場合，高信頼性が要求される基幹的なシステムであり，長期にわたって利用され，その間，法令の改定などによる進化が必要なシステムである．これまで，このようなシステムを設計，開発，保守するための系統的な方法論は存在しなかったが，法令実動化システムという考えは，法令の構造に基づいたアーキテクチャのシステムを採用することにより，高信頼かつ変更に強いシステムを開発可能であるということを提唱するものである．法令構造からシステムを開発することの利点は，高信頼，高進化性だけでなく，情報システムのアカウンタビリティの実現に決定的である．これは，情報システムの動作や振る舞いを，それが由来する法令文，法令構造にもとづいて解釈することが可能となるからである．

- 法令論理表現，法令構造からの法令実働化情報システムのアーキテクチャの導出
- 法令実動化情報システムのアカウンタビリティ機構の設計
- 法令の変更に伴い法令実動化情報システムを系統的に進化させる方法論
- ゴール指向要求獲得と法令構造の関連

第2章　法令文書の言語処理

2.1　法令工学における自然言語処理の役割

自然言語処理は法令工学の目的 (図 2.1) に用いることができる．自然言語処理の様々な技術を用いて，法令文に論理的矛盾がないか，文書的問題がないか，関連法令と整合性がとれているか，法令の変更，追加，削除に矛盾がないか等を調べることができる．これらの検査・検証は機械的に行うレベルから人間を支援するレベルまであり，それに応じて，自然言語処理の利用は次の場合に分けられる．

(1) 法令の文や文章の論理的矛盾の機械的な検査への適用

(2) 論理式から自然言語への変換

(3) 言語的な推論による文書的問題の検査への適用

(4) 法令理解の支援

(5) 法令作成の支援

以下，本節では，(1)，(2)，(3)，(4), (5) を補足する．次節で，法文を論理表現で表すということについて論じ，その次の節で，法文の言語解析について述べる．

(1) 法令の文や文章の論理的矛盾の機械的な検査への適用

論理的矛盾は論理式により調べることができる．そのためには，法令の文や文章の意味を論理式で表現する必要がある．論理式は，3 章で説明するように，推論

> 法令 (契約書，社内規定等を含む) がその制定目的にそって適切に作られ，論理的矛盾や文書的問題がなく，関連法令との整合性がとれていることを検査・検証し，法律の改定に対しては，矛盾なく変更や追加削除が行われることを情報科学の手法を用いて支援する．

図 2.1: 法令工学の第一の目的 [7]

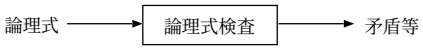

図 2.2: 論理式の検査による矛盾等の発見

図 2.3: 自然言語処理による法令文書の論理式への変換

手続き (定理証明器) によりその矛盾が調べられる (図 2.2)．法令の文や文章の意味を論理式で表現することは，自然言語処理により機械的に求めることができる (図 2.3)．ただし，現状の技術では，どんな文でも論理式に自動的に変換できるわけではない．一般に完全な変換ができないので，文や文章を前編集して機械的に変換しやすくしたり，あるいは，出力の論理式を後編集して正しいものにする必要がある．これは，機械翻訳技術を利用するときに，前編集や後編集しているのと同様の発想である [21]．

論理的矛盾は，論理式で調べるのが厳密な方法と言えるが，近似として，自然言語処理による疑似的な推論で調べることも考えられる．これは文書的整合性の検査の一環して位置付けられる．(3) の項はこの観点からのものである．

(2) 論理式から自然言語への変換

法令の意味を表す論理式の検査の結果，矛盾や誤りがあると分かれば，あるいは，相反する事項があるとか，必要な事項が足りないとか，論理に誤りがあるとかが分かれば，一般のユーザには，そのような結果は自然言語により説明する必要がある．論理式の検査により分かった問題点の説明には，厳密には関連する論理式を用いる必要があるが，一般の人に説明するにあたっては，自然言語により説明することが望まれる．このために，論理式を自然言語に変換することが必要となる (図 2.4)．これは言語生成と呼ばれる技術により行われる [21]．

(3) 言語的な推論による文書的問題の検査への適用

考慮すべき文書的問題としては，以下の場合等が考えられる．論理式を介在せず，言語的に推論して文書の整合性を調べる (図 2.5)．

- 法令に盛り込むべき内容が必要にして十分に記述されているか．
  - 用語の定義があるか．

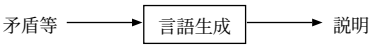

図 2.4: 論理式の検査結果の説明の生成

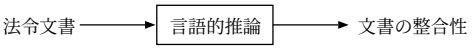

図 2.5: 自然言語処理による法令文書の整合性の検査

  - 対象概念の定義や分類にもれがないか.
  - 義務規定に対して罰則条項があるか.

- 語句や条文の参照関係が明確か.
- 語句の用い方に関連条文とづれがないか.
- 語の意味が曖昧でないか.
- 法令文の構文が曖昧でないか.
- 法令文の意味が明確か.

法令に盛り込むべき内容が必要にして十分に記述されているかというのは，例えば，年金の法律において，被保険者の種類が何種類かあるとき，それらがもれなく定義されているかとか，何かの義務が規定されているときに，その義務が守られなかっときのことが書かれているかといったものである.

語の意味の曖昧性には，多義語がどの意味で使われているのかが不明確な場合とか，語句が指示する対象が曖昧な場合などがある．語を同じような意味で用いていても，文脈によりそれが指し示すものが異なっていると，必ずしも読み手の理解は明確とはならない．例えば,「年金」という語も，福祉の制度を指す場合もあれば，具体的に支給される給付そのものを指す場合もある.「保険料納付期間」という語における「期間」は単純な理解では，開始時から終了時までの一定の時間と思うが，国民年金法では，そのような場合だけでなく，飛び飛びの期間の集まりも含んでいる場合がある．一般に，読み手の理解を明確かつ容易にするためには，一つの法令内では，一貫した語の使用が望まれる.

構文構造 (語句の係り受け) が曖昧であると，その結果，一般に文全体の意味も曖昧になりうるから，この検査は基本的なものである．接続詞など，法令特有の読み方があり，それらの考慮も必要である.

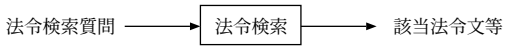

図 2.6: 法令文書等の検索支援

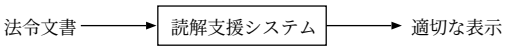

図 2.7: 法令文書等の読解支援

語句や条文の参照関係が不明確というのは，例えば，参照先の条文がないとか直接関係がないという場合である．

(4) 法令理解の支援

法令作成に携わる人，法令を利用する人，法令を学ぶ人にとって，必要とする法令が効率よく的確に探せる検索システム (図 2.6)，法令文や文章自体の理解を容易にする読解支援システムがあれば大いに便利と考えられる (図 2.7)．このようなことは法令だけが対象でなく，組織の規則や契約書も対象になる．関連法令や過去の履歴にもアクセスでき，それらとの関連の理解も容易になることが望まれる．このようなシステムの構築に自然言語処理技術は必要不可欠のものである．

(5) 法令作成の支援

法令作成に携わる人，組織で規則の作成に携わる人，契約書を作成する人にとって，法令，規則，契約書が効率よく的確に作成できるよう支援するシステムがあれば，大いに便利と考えられる (図 2.8)．このようなシステムは関連法令等や過去の履歴にもアクセスでき，それらとの関連も扱えることが望まれる．このようなシステムの構築には自然言語処理技術は必要不可欠のものである．

2.2　法文の論理表現

法令の文や文書の解析は，論理式への変換，文書的問題の検査，法令理解の支援等のために行われる．本節では，主に法文の論理式への変換の前提として法文の意味を論理表現で表すことについて論ずる．

論理表現への変換は一般に一文だけを対象にするのでは不十分であるが，ここでは基本となる一文の論理表現への変換を念頭におく．なお，文書的問題の検査

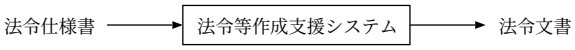

図 2.8: 法令文書等の作成支援

にも複数文の処理が必要である．複数文の扱いについては，本稿の後半で触れる．

2.2.1 自然言語を論理表現で表すこと

法令の論理的整合性を論理表現のレベルでみるため，法文の意味を自然言語処理技術により解析し論理表現を求める．論理表現は論理式により表す．論理式を用いるのは，厳格性のためであり，論理式の矛盾を機械的に調べる方法があるからである．そこで，自然言語の文と論理式とを対応付けるわけであるが，対応は必ずしも単純でない．

何故，対応が単純でないかというと，次のような問題があるからである．

(1) 自然言語の語は多義である，文は一般に曖昧である，文には省略がある，同じ意味でも多様な言い方がある等のため，自然言語文と論理式との対応は単純でない．

(2) そもそも自然言語の意味が論理表現で捉えきれるのかという疑問がある．

以下，(1)，(2) の問題に対して，法文を論理式で表現することの可能性を述べる．

(1) 言語の曖昧性，多様性，省略などに関する面

自然言語の文が表す意味は，一般に世界モデルや文脈を考慮することで明確となる．そこで，明確な意味は形式的に表現できるとすると，世界モデルや文脈に基づいて法文の意味を的確に解析できれば，自然言語の文が表す意味は論理表現できると言える．

この点に関して，法文については以下のことが言えるであろう．

- 法文の意味は，法令が特定の領域について言及することから，法令が対象とする領域の世界モデルを与えることにより，明確な解釈が可能になる．
- 法令は社会のルールを規定するためのものであり，社会の構成員に明確に理解されるべきものであることから，法令は文章として一定の構造を持ち，文としても一定の構造を持ち，機械的な解析が可能になる．

(2) 自然言語の意味を論理表現で捉えられるかという面

この点については，次のことが言えるであろう．

> 法令が人間の社会活動を規定するものであることから，法文の意味は明確であるはずであり，従って，論理式に対応させることができると考えられる．

自然言語は，感情や感覚など，様々な微妙なことも表現し，これまでに開発されてきた論理表現でそれらが表現しきれるのかという問題はあるが，我々が対象とするのは，人間や組織などの活動を明確に規定する法文であるので，論理表現により表現できるはずと考える．

上記の点で，法文の意味は明確であり，論理式に対応させることができると仮定する，ただし，法令にも憲法のような理念を述べているものから，所得税法のように具体的な税金の計算法を述べているものまで幅があるが，それらの中で特に以下の法令を対象にする．

> 国民年金，所得税法，道路交通法等，それが規定する対象や内容が情報処理の面で明確なもの

明確なというのは，所得税，年金等については，それら及び関連する法令に従って情報処理システムが作られているという意味である．そういう点では，法令は対応する情報処理システムの一種の仕様書とも言える [6, 7].

2.2.2　法文の言語表現

法文が記す内容はおおよそ以下のようなものである．

> 法令が対象とする概念の定義・関連・分類，人や組織の権利や義務，禁止事項，罰則，法令の施行に関すること等．

このような内容を持つ法文の言語表現は論理的には次の要素を持つとされる [19, 20].

> 要件と効果

法文は，上記の内容が，一般に,「○○の条件が成立すれば，□□のことができる」,「○○の条件が成立すれば，してはいけない」,「○○の条件が成立すれば，罰になる」等の形で書かれている．すなわち，条件と帰結という要素からなる論理的形式で書かれている．このように法文の言語表現は，条件とそれが満されたときのことを表す形式になっていて，要件効果構造と呼ばれている (図 2.9)[19]．例

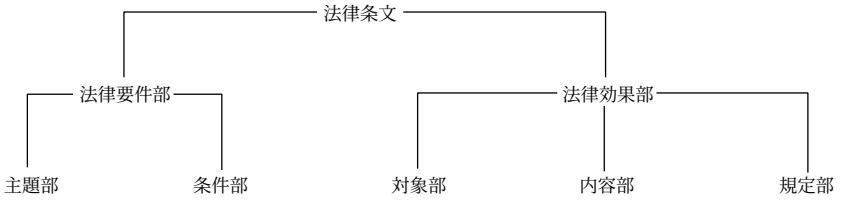

図 2.9: 要件効果構造 [19]

年金給付は，その支給を停止すべき事由が生じたときは，その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する．ただし，これらの日が同じ月に属する場合は，支給を停止しない．

図 2.10: 国民年金法第 18 条 2 項

えば，国民年金法第 18 条 2 項 (図 2.10) の要件部と効果部は図 2.11 のようになる[1]．

法文は要件と効果の部分からなるが，これは論理的にそのような内容を持つのであって，文が要件部分と効果部分とに常にきれいに分割されるわけではない．自然言語文には一般に主題を表す部分があり，法文も例外ではない．そのような場合，主題部は法文の主題を表すだけでなく，要件部や効果部の主語，目的語，あるいは要件部や効果部中の語の修飾語としても機能する．例えば，国民年金法第 18 条 2 項 (図 2.10) では，文頭の「年金給付」は，この法文の主題を表すとともに，要件部「その支給を停止すべき事由が生じたときは,」の「その」に対する被指示語であり，効果部「その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する.」の「支給」の修飾語である．

このような要件効果構造の表現形態には以下の場合がある．

(1) 1 文に一つの要件効果が記される．

(2) 1 文に複数の要件効果が記される．

(3) 号の列挙と合せて要件効果が記される．

(4) 上記 (2) と (3) を合せた形式で要件効果が記される．

(5) 用語の定義として記される．

[1]国民年金法などの法律は官報に載る縦書きのものが正式のもので，余白や句読点等，書き方に決まりがある [3] が，ここでは横書きにして，句読点は「.」と「,」を用いる．

要件部: 年金給付は，その支給を停止すべき事由が生じたときは，
効果部: その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する．

図 2.11: 要件効果構造の例

(端数処理)
第十七条　年金たる給付（以下「年金給付」という.）を受ける権利を裁定する場合又は年金給付の額を改定する場合において，年金給付の額に五十円未満の端数が生じたときは，これを切り捨て，五十円以上百円未満の端数が生じたときは，これを百円に切り上げるものとする．

図 2.12: 国民年金法第 17 条

以下，各場合について，補足する．

(1) 1 文に一つの要件効果

基本的な場合で，図 2.10 の 18 条 2 項の第 1 文でみると，要件部は「年金給付は，その支給を停止すべき事由が生じたときは,」，効果部は「その事由が生じた日の属する月の翌月からその事由が消滅した日の属する月までの分の支給を停止する.」である (図 2.11)．正確には,「年金給付は,」は効果部の構成素でもある．

(2) 1 文に複数の要件効果

一つの法文は一対の要件効果を述べているとは限らない．複数の要件効果を記している場合もある．例えば，図 2.12 に示す国民年金法第 17 条では,「年金たる給付を受ける権利を裁定する場合又は年金給付の額を改定する場合において，年金給付の額に五十円未満の端数が生じたときは，切り捨てる」という要件効果構造と「年金たる給付を受ける権利を裁定する場合又は年金給付の額を改定する場合において，五十円以上百円未満の端数が生じたときは，これを百円に切り上げるものとする.」という要件効果構造が記されている．この例では，用言連用形 (「切捨て」) により，並列表記されているが，しばしば見られるのは，括弧内に別条件が示され，その条件の場合も含めて，複数の要件効果が表される場合である．例を図 2.13 に示す．この例では，括弧内には複数の要件が示してあり，しかも後段

(資格喪失の時期)
第九条　第七条の規定による被保険者は，次の各号のいずれかに該当するに至つた日の翌日 (第二号に該当するに至つた日に更に第七条第一項第二号若しくは第三号に該当するに至つたとき又は第三号から第五号までのいずれかに該当するに至つたときは，その日) に，被保険者の資格を喪失する．

図 2.13: 国民年金法第 9 条

(被保険者の資格)
第七条　次の各号のいずれかに該当する者は，国民年金の被保険者とする．
一　日本国内に住所を有する二十歳以上六十歳未満の者であつて次号及び第三号のいずれにも該当しないもの (被用者年金各法に基づく老齢又は退職を支給事由とする年金たる給付その他の老齢又は退職を支給事由とする給付であつて政令で定めるもの (以下「被用者年金各法に基づく老齢給付等」という．) を受けることができる者を除く．以下「第一号被保険者」という．)
二　被用者年金各法の被保険者，組合員又は加入者 (以下「第二号被保険者」という．)
三　第二号被保険者の配偶者であつて主として第二号被保険者の収入により生計を維持するもの (第二号被保険者である者を除く．以下「被扶養配偶者」という．) のうち二十歳以上六十歳未満のもの (以下「第三号被保険者」という．)

図 2.14: 国民年金法第 7 条

は括弧の外と違う条件を示しており，分かりにくい．なお，この例は，正確には，号に示す事項と合わせて，要件効果が表されている．

(3) 号の列挙と合せて表現される要件効果

国民年金法には混み入った内容を表す場合が多いが，そのような場合，要件効果構造が一つの文で述べられているとは限らず，対象や条件等，複数の事項を号として列挙して，それらと合せて要件効果構造を表していることも多い．例えば，図 2.14 の国民年金法第 7 条では,「次の各号のいずれかに該当する者」が条件を示し，具体的な内容は号に列挙されている．

(4) 上記 (2) と (3) を合せた形式で表現される要件効果

(資格取得の時期)
第八条　前条の規定による被保険者は，同条第一項第二号及び第三号のいずれにも該当しない者については第一号から第三号までのいずれかに該当するに至つた日に，二十歳未満の者又は六十歳以上の者については第四号に該当するに至つた日に，その他の者については同号又は第五号のいずれかに該当するに至つた日に，それぞれ被保険者の資格を取得する．
一　二十歳に達したとき．
二　日本国内に住所を有するに至つたとき．
三　被用者年金各法に基づく老齢給付等を受けることができる者でなくなつたとき．
四　被用者年金各法の被保険者、組合員又は加入者の資格を取得したとき．
五　被扶養配偶者となつたとき．

図 2.15: 国民年金法第 8 条

国民年金法などでは，図 2.15 の国民年金法第 8 条のように，上記 (2) と (3) を合せた形式で要件効果が記される場合もある．図 2.15 の例では，対象者とその条件日の対が併記され，条件日の内容が号に示されている．

(2) や (3) のように，要件効果構造が複数ある場合，人間にとって必ずしも読みやすくない．さらには，(4) のように，(2) と (3) が複合した場合は，さらに読みづらいことになる．このような形式の法文では，並列解析や文脈解析が必要となるが，どちらも簡単ではない．実際，並列構造に強い従来の構文解析プログラムも誤る場合がある．このプログラムは特に法文を考慮したものでない．法文における，法文の特徴を考慮した解析をすることにより解析精度は上げられると考えられる．例えば，法文においては，並列語句を接続する「又は」「若しくは」「かつ」「及び」等については，そのスコープの解釈の仕方が仮定されており [3, 11, 23]，これを利用することができる．

(5) 用語の定義

用語の定義も,「○○とは，〜である□□を言う」という形式で書かれており，要件効果構造になっていると言える．この文では,「○○」が要件部,「〜である□□」が効果部とみなせる．また，逆方向のことも表明されており,「□□について〜が成立するば，○○である」という要件効果構造にもなっている．すなわち，定義文では等価関係が成立している．国民年金法第 5 条 3 項の例を図 2.16 に示す．

上記のような定義文は定義条文に見られるが，一般の条文中に括弧内には,「以

(保険料免除期間)
3　この法律において,「保険料免除期間」とは，保険料全額免除期間，保険料四分の三免除期間，保険料半額免除期間及び保険料四分の一免除期間を合算した期間をいう.

図 2.16: 国民年金法第 5 条 3 項における用語の定義

障害基礎年金は，疾病にかかり，又は負傷し，かつ，その疾病又は負傷及びこれらに起因する疾病 (以下「傷病」という．) について初めて医師又は歯科医師の診察を受けた日 .....

図 2.17: 国民年金法第 30 条における用語の定義

下，○○という.」という形で書かれる定義もある．例えば，一部を図 2.17 に示す国民年金法第 30 条の「以下「傷病」という.」，図 2.12 に示した国民年金法第 17 条の「以下「年金給付」という.」がそのような例である．このような例では，括弧の前の表現中から「〜である□□」に相当するところを推論しなければならない.

なお，括弧内において「〜□□という」表現があっても，必ずしも上記の意味での定義になっていない場合もある．例えば，図 2.18 に示す国民年金法第 12 条 2 項の「世帯主」は，論理式における定義というよりも,「被保険者の属する世帯の世帯主」に対する省略表現の定義とみるべきであろう．プログラミング言語におけるマクロ表現のようなものとも言える.

2.2.3 論理表現の考慮点

言語の意味を表現する言語はこれまで種々研究されてきたが，我々は広く用いられている 1 階述語論理をベースにする．法文を論理表現するに際し，以下の点を考える.

(1) 自然言語の論理表現における問題点と対処

(2) 実際的な論理表現

(3) 述語の表現

2　被保険者の属する世帯の世帯主 (以下単に「世帯主」という．) は，.....

図 2.18: 国民年金法第 12 条 2 項における用語の定義

「県の機関」
県の議会、執行機関、公営企業管理者、警察本部（警察署を含む.）若しくはこれらに置かれる機関又はこれらの機関の職員であって法律上独立に権限を行使することを認められたものをいう。

図 2.19: 富山県条例 54 第 2 条 2 項における事象参照表現

自然言語の論理表現における問題点を考慮し，特に，Davidsonian style [16, 25] と呼ぶ形式を参考にするが，以下，この形式を説明し，次に，この表現形式をベースに，実際上の問題も考慮した表現を示し，個々の法文を表現するにあたって考えなければならない述語の表現について述べる．

(1) 論理表現の形式

法文の論理表現を考えるにあたり考慮しなければならない自然言語文の特徴および日本語文の特徴として以下のものがある．

(a) 事象を参照する表現

(b) ゼロ代名詞と格要素

(c) 様相表現

(a) および (b) の問題を考慮して，自然言語文を論理表現をするのに，Davidsonian style と呼ぶ形式がとる考え方が参考になる．(c) については，様相論理の表現が参考になる．以下に，上記特徴と表現を説明する．

(a) 事象参照

自然言語文の特徴に，文が複数の節から構成される場合，節が表す事象を後続の節が参照することがある．例えば，富山県条例 54 第 2 条 2 項 (図 2.19) の「県の機関」の定義では,「権限を行使することを認められた」という表現があるが,「行使する」という行為 (事象) が「認める」という行為 (事象) の格となっている.「認める」は「x_1 が x_2 に x_3 を認める」と言うことから，その述語表現を

認める (x_1, x_2, x_3)

と表現すると，x3 に対応するものは「行使する」となる.「行使する」は「y1 が y2 を行使する」と言うことから，その述語表現を

行使する (y_1, y_2)

と表現すると,「権限を行使することを認められた」は，条文の意味を考慮すると，

認める $(x_1, x_2,$ 行使する $(x_2, y_2)) \wedge$ 権限 (y_2)

となるが，これは高階述語である．高階述語の証明は一般に扱い難いため，このような高階表現は問題となる．このような問題については，法文の意味表現に関する過去の研究においても指摘されていて [15, 24]，それらの研究においても，事象参照の表現を問題にし，独自の表現が提案されている [15, 24]．このような問題に対して，我々は，新たに表現言語を考えるのでなく，定評があり広く用いられている 1 階述語論理の範囲で事象参照を表現することとする．

Davidsonian style の形式では，事象に対して事象変数を導入するが，これにより事象参照の表現も高階にならない．上記の例は，次のようになる．

行使する $(e_1, y_1, y_2) \wedge$ 権限 $(y_2) \wedge$ 認める (e_2, x_1, y_1, e_1)

「行使する」を事象変数 e1 に対応付け,「認める」ではそれを参照する形で表現する．

(b) ゼロ代名詞と格要素

法文の論理表現を求めるにあたり考慮するもう一点は動詞の格要素を論理的にどのように表現するかという点である．

日本語の特徴は，主語や目的語が文脈から分かるときは，表現されないことである．英語等の言語の場合，そのような主語や目的語は代名詞で表される．このような点から，日本語の表現されない主語や目的語を言語学ではゼロ代名詞と呼んでいる．当然であるが，法文にもゼロ代名詞は同様に出現する．なお，文脈から分かるものと言ったが，文によっては必ずしも明らかとは限らない場合もある．

問題は，人間にとってはゼロ代名詞であっても，機械には必ずしも自明でないことである．ゼロ代名詞が何を指しているか 100%復元できるアルゴリズムは今のところないが，機械処理上は，ゼロ代名詞があっても，何らかの論理表現を出力する必要がある．

もう一つの問題は，同じ動詞でも，意味によって，それが取る格要素の個数や種類が同じとは限らないことである．どこまでを必須の格とし，どこまでを任意の格とするか，明確でないところもある．たぶんに応用に依存するところもある．そんな状況を考慮すると，辞書が改編されたとしても，論理表現を出力する意味

(保険料の納期限) 第 91 条 毎月の保険料は，翌月末日までに納付しなければならない．
$\forall\, x_1, t_1\ \exists\, x_2, e_1, t_2, t_3$ 被保険者 $(x_1) \wedge$ 年金 $(x_3) \wedge$ 月 $(t_1) \wedge$ 金 $(x_2) \wedge$ 保険料 (x_2, x_1, x_3, t_1) $\Rightarrow \Box$(納付 $(e_1, x_1, x_2) \wedge ttime(e_1, t_2) \wedge$ 翌月 $(t_3, t_1) \wedge$ 末日 (t_2, t_3))

図 2.20: 国民年金法第 91 条

解析システムは影響を受けないことが期待される．

Davidsonian style を用いると，これらの問題に対処できる．すなわち，事象変数と格関係を表す述語とを用いて，動詞の格要素をそれぞれ別個の述語として表現することができる．例えば，上記の富山県条例の例は次のようになる．

行使する $(e_1) \wedge agent(e_1, y_1) \wedge$ もの $(y_1) \wedge object(e_1, y_2) \wedge$
権限 $(y_2) \wedge$ 認める $(e_2) \wedge recipient(e_2, y_1) \wedge object(e_2, e_1)$

この例では,「認める」の主語が省略して表現されている．

(c) 様相表現

「〜しなければならない」「〜することができる」などの様相表現は法文を特徴付けるものである [3, 23]．これらの表現は次の形をしている．

〜する + < 様相表現 >

このような言語表現の論理表現は様相表現の扱いが問題になる．様相表現を述語にすると，上記で述べたように，高階の論理式になるため，様相表現に対するものは一般に様相演算子として表現されている．例えば，国民年金法第 91 条の表現は図 2.20 のようになる[2]．この表現では,「しなければならない」を様相演算子 □ により表現している．様相表現を述語として扱う考え方もある [17]．この考え方では,「保険料を翌月末日までに納付する」という命題が述語の引数になるが，これが述語だと高階になるので，この命題を reify という操作により object 化して引数とする．

どちらの表現方法を用いようと，そのための推論規則が必要であり，証明手続きも必要となる．そこで，近似として様相表現を外して扱うか，対象領域におい

[2]図の表現は，次項 (2) の議論に基づいた表現をしている．また，条文にはない「被保険者」を補完している．'$ttime(e_1, t_2)$' は「e_1 が t_2 までに起きる」を表す．

て様相表現が表す実際上の意味を考慮して，法文の論理表現を表すことが考えれる．国民年金法や所得税法などを扱う場合，実際上の意味を考慮することが処理上も有効であると考えられるが，これについては別稿で議論する．

(2) 実際的な論理表現

上記のように表現すると，事象参照，ゼロ代名詞，格要素数の問題に対応できるが，細かく分割して表現されているために，必ずしも見やすくない．このような式は，また，機械的に矛盾の検査をする場合，照合操作が増え，非効率と考えられる．

論理表現は，実際上，特定の領域の法令毎に用いるから，動詞や名詞の意味は固定して考えることができる．そうすると，格要素毎に述語を分けることもない．本質的なところは，事象変数を用いる点であろう．そのような考え方からは，上記の論理表現は以下のようになる．

$$\text{行使する}\,(e_1,\, y_1,\, y_2) \wedge \text{もの}\,(y_1) \wedge \text{権限}\,(y_2) \wedge \text{認める}\,(e_2,\, x_1,\, y_1,\, e_1)$$

このような表現だと，今度は動詞述語の引数の役割が分かりづらくなる．そこで，役割を書くと次のようになる．

$$\begin{aligned}&\text{行使する}\,(e_1,\, \mathit{agent}\colon\, y_1,\, \mathit{object}\colon\, y_2) \wedge \text{もの}\,(y_1) \wedge \text{権限}\,(y_2) \wedge \\ &\text{認める}\,(e_2,\, \mathit{agent}\colon\, x_1,\, \mathit{recipient}\colon\, y_1,\, \mathit{object}\colon\, e_1)\end{aligned}$$

さらに論理式の表現の一貫性を持たせ，意味が明確になるように，引数の意味クラスあるいはタイプを書き加えると次のようになる．

$$\begin{aligned}&\text{行使する}\,(e_1,\, \mathit{agent}{:}\mathit{human/organization}\colon\, y_1,\, \mathit{object}{:}\mathit{competence}\colon\, y_2) \wedge \\ &\text{もの}\,(\mathit{thing}\colon\, y_1) \wedge \text{権限}\,(\mathit{competence}\colon\, y_2) \wedge \\ &\text{認める}\,(e_2,\, \mathit{agent}{:}\mathit{human/organization}\colon\, x_1, \\ &\qquad\qquad \mathit{recipient}{:}\mathit{human/organization}\colon\, y_1,\, \mathit{object}{:}\mathit{act}\; e_1)\end{aligned}$$

このような表現により，動詞がとる格のタイプと名詞のタイプのミスマッチが見付かるとすると，論理表現が不十分でなければ，もとの自然言語文の不適切さが分かることになる．なお，以下の議論では簡単のため，タイプは用いないで表現する．

(3) 論理式の要素

実際に法文を適切に論理表現するにあたっては，法令が対象とする領域の概念 (オントロジー) を定めることが重要である．ここでいうオントロジーとは，概念の列挙やその関連だけをいうのではなく，概念を表すのは述語なのか項なのか，述語の場合，引数の個数は幾つか，引数のタイプは何なのか等を問題にする [17]．以

下では，これまでに県の条例，所得税法，国民年金法などの条文の論理表現を分析してきた経験に基づいた述語表現について論ずる．

言語が記述するのは,「ものごと」である．すなわち，記述対象の物 (object)，それに関する事象 (event) である[3]．事象は様態により形容される[4]．物には無生物と生物がある．生物の一つとして人間がある．事象の一つとして人間の行為がある．人が集まって組織や社会が作られ，そこには規則や慣習がある．それらに基づいて，抽象的な人や行為が定義される[5]．

以上のように言語が記述するものをみて，論理式の主な要素として，(a) 事象の表現，(b) 物，(c) 関係，(d) 属性を取り上げ，それらについて以下に述べる．また，(e) 同形語の動詞と名詞の扱い，(f) 副詞の表現について補足する．

(a) 事象の表現

行為を含む事象は，動詞，サ変名詞等で表され，論理表現上は述語 (事象述語と呼ぶ) により表現される．論理表現の考慮点のところで示したように，事象述語は引数の一つに事象変数をとる．さらに事象述語は事象を構成する役割 (格) を引数にとる．引数は定項か変数である．それらは，事象の役割を担っている対象を表す．対象はそのカテゴリを表す述語で表現される．事象の論理表現として，例えば，所得税法の条文にある「居所を有する個人」は次のようになる．

$$\text{有す}\,(e_1,\, agent\colon x_1,\, object\colon x_2) \wedge \text{個人}\,(x_1) \wedge \text{居所}\,(x_2)$$

この例では，‘有す’ の引数として，事象変数 e_1，物に対する変数 x_1, x_2 があり，x_1, x_2 が事象 e_1 を構成する．x_1, x_2 はそれぞれ ‘個人’, ‘居所’ というカテゴリである．

(b) 物

物 (object) は，抽象物，具体物，無生物，有性物，人等のカテゴリに分類される．物に関する推論は一般にカテゴリのレベルでなされる [17]．カテゴリは述語として表現される．例えば，物 a がカテゴリ ‘被保険者’ なら，次のように表現される[6]．

$$\text{被保険者}\,(a)$$

(c) 関係

語には物と物の関係を表すものがある.「親」「父」「母」「兄」「弟」等がそのような語であり，例えば,「親」の論理表現は次のようになる．

[3]物も事象のような性質を持っている [17]．例えば,「日本」は，物の一つとみれるが，長い時間軸上でみると，広さが地理的に変化するという意味で，事象のようなところがある

[4]様態というのは，例えば,「激しく川が流れる」という場合の「激しく」である．

[5]抽象的な人や行為というのは，法人，名前を付ける，契約する，入学するといったものである．

[6]カテゴリをオブジェクト化して，‘$Member(a,$ 被保険者$)$’ と表す考え方もある [17]．

人 $(x_1) \wedge$ 人 $(x_2) \wedge$ 親 (x_1, x_2)

x_1 は x_2 の親であることを表し，‘親’という述語により x_1 と x_2 の関係が表される．国民年金法にも「配偶者」「夫」「妻」などの語があり，関係述語により表現される．

この他，国民年金法では，例えば,「保険料」という語は，料金あるいは金のカテゴリに分類されるが，国民年金法の領域を考慮すると,「金」「被保険者」「年金の種類」「納付月」などの概念により次のように関係付けられるものと分析される．

保険料 $(x_1, x_2, x_3, x_4) \wedge$ 金 $(x_1) \wedge$ 被保険者 $(x_2) \wedge$ 年金 $(x_3) \wedge$ 月 (x_4)

ちなみに，保険料については納付済期間という概念もあり，それに対応する保険料もあるため，検討が必要である．

(d) 属性

語には対象の属性あるいは性質を現す語もある．例えば，物の「重さ」「大きさ」「色」等がそのような語である．これらは一般に物を変数とする関数により表現される．関数は変数とその値の対のリストである．見方を変えて，変数値と対応する関数値に対して真となる述語としても表現できる．国民年金法では，例えば，被保険者の「住所」とか「年齢」が属性の例である．

(e) 同形語の動詞と名詞の表現

論理表現するにあたっては，まず，語に対する論理表現を上記のどれにするかを検討するが，同じ語であってもタイプが異なる場合があることに注意しなければならない．

例えば，国民年金法第 30 条に以下の文がある．

> ... 疾病にかかり，又は負傷し，かつ，その疾病又は負傷及びこれらに起因する疾病 (以下「傷病」という．) について ...

この例では，最初の「疾病」は一般概念を表しているが，次の「その疾病」は「疾病にかかる」という事象の結果の特定の「疾病」を指しており，同じ「疾病」であるが，表しているものが異なる．従って,「疾病にかかり」とその結果の「疾病」に関する部分の論理式は次のようになる．

かかる $(e_1, agent{:}\ x_1, loc{:}\ x_2) \wedge$ 疾病 $(x_2) \wedge$ 結果 $(e_1, x_3) \wedge$ 疾病 (x_3)

これを踏まえ,「傷病」の定義を解釈すると，その定義の論理表現は図 2.21 のようなる．

(f) 様態表現

$\forall\ x_1\ \exists\ e_1, x_2$ 傷病 $(x_1) \Leftrightarrow$
　　疾病 $(x_1) \vee$ 負傷 $(x_1) \vee$
　　(疾病 $(x_1) \wedge$ 起因 $(e_1, object{:}\ x_1, cause{:}\ x_2) \wedge$ (疾病 $(x_2) \vee$ 負傷 (x_2)))

図 2.21: 国民年金法第 30 条における用語の論理表現

上記では，動詞，名詞を対象に論理表現を論じたが，副詞等，事象の様態を形容する表現もある．例えば，国民年金法の第 30 条に,「初めて ... 診察を受けた日」という表現がある．この表現は，被保険者が当該の傷病について診察を受けた日の中で最初のものという意味である．厳密にはそのように論理式で定義するのであろう．例えば，次のように表現できる．

$\forall\ x\ \exists\ e\ e'\ y\ d\ t\ t'$ 被保険者 $(x) \wedge$ 医師 $(y) \wedge$ 傷病 $(d) \wedge$
診察する $(e, agent{:}\ y, object{:}\ x) \wedge$ 理由 $(e, d) \wedge$ 時 $(e, t) \wedge$ 初めて $(e) \wedge$
診察する $(e', agent{:}\ y, object{:}\ x) \wedge$ 理由 $(e', d) \wedge$ 時 $(e', t') \Rightarrow \leq(t, t')$

これは，ちょうど，人間の足が 2 足と定義するのに，x と yが足であるとき，すなわち，足 (x)，足 (y)であるとき，zが足なら，z は x あるいは yであると表現する [17] のと似ている．

現実的には，単に事象 eが「初めて」と以下のように形容するだけの表現をし，必要に応じ,「初めて」に関する推論をしたり，あるいは情報処理システムで実際に初めての日を計算するのが妥当かもしれない．

$\forall\ x\ \exists\ e\ y\ d\ t$ 被保険者 $(x) \wedge$ 医師 $(y) \wedge$ 傷病 $(d) \wedge$
診察する $(e, agent{:}\ y, object{:}\ x) \wedge$ 理由 $(e, d) \wedge$ 時 $(e, t) \wedge$ 初めて (e)

2.2.4　論理表現の例

上記に述べた観点に基づいた論理表現の例を示す．

(1) 基本的な要件効果構造は次の形の論理式である．

$C \Rightarrow P$

この例として，国民年金法第 48 条を図 2.22 に示す．

(2) 用語の定義等は「W とは P をいう」といった形式で記される．その論理式は以下の形になる．

(失権) 第 48 条 付加年金の受給権は，受給権者が死亡したときは，消滅する．
$\forall\, x_1\ \exists\, x_2, e_1, e_2, t_1$ 受給権者 $(x_1) \wedge$ 死亡 $(e_1, object\colon x_1) \wedge time(e_1, t_1)$ $\Rightarrow$ 付加年金の受給権 $(x_2) \wedge$ 消滅 $(e_2, object\colon x_2) \wedge time(e_2, t_1)$

図 2.22: 国民年金法第 48 条とその論理式

この法律において,「年金保険者たる共済組合等」とは，国家公務員共済組合連合会，地方公務員共済組合連合会又は日本私立学校振興・共済事業団をいう．
$\forall\, x_1$ 年金保険者たる共済組合等 $(x_1) \Leftrightarrow$ 国家公務員共済組合連合会 $(x_1) \vee$ 地方公務員共済組合連合会 $(x_1) \vee$ 日本私立学校振興・共済事業団 (x_1)

図 2.23: 国民年金法第 5 条 10 項とその論理式

$$W \Leftrightarrow P$$

国民年金法第 5 条 10 項の表現を図 2.23 に示す.

2.3 法文の解析

自然言語処理技術の解析は，形態素解析，構文解析，意味解析などからなり，文の構文 (統語) 構造や意味表現を求めることができる [21]．文が曖昧であれば，解析処理により複数の解釈が出力される．複数の解釈が可能な場合は，複数の中から世界モデルや文脈情報により妥当な解釈が求められる．

以下では，我々が開発している解析システム [1, 8, 13, 14] の概略を説明する．システムの処理は次のように進む.

(1) 文の構文構造を解析する.

(2) 骨格的な論理構造を求める

(3) 構成素の論理表現を生成する.

(4) 文全体の論理表現を生成する.

表 2.1: 主題部，条件部を示す手掛かり語の例

主題部	～は，～が，～も
条件部	～するときは，～については，～においては，～する場合には

以下，各ステップの概要を説明し，システムについて述べる．

2.3.1　文の構文構造の解析

法文の文字列を形態素解析し，次に構文解析して，構文構造を求める．形態素解析は文字列を語に分割し，語の品詞を求める．これらの情報に基づいて，構文解析は，文を構成する名詞句や動詞句を求める．日本語の解析では名詞句や動詞句を解析するよりも，どの語がどの語を修飾するかという係り受けを求めることが多いが，ここでも構文解析は係り受け解析である．

2.3.2　骨格的な論理構造の生成

文の要件部，効果部等，全体的な論理構造を解析する．以下の処理がある．

(1) 文の要件部と効果部への分割

(2) 様相部分の同定

(3) 標準的表現への言い替え

以下，各項目について説明する．

(1) 文の要件部と効果部への分割

構文解析により求められる構文 (係り受け) 構造を要件部と効果部に分割する．この分割は法文の言語特徴に基づいて行うことができる．例えば,「～する場合,」という表現があるところは，条件部とそれ以降との切れ目となりうる．このような特徴の主なものを表 2.1 に示す．国民年金法の始めの方の 50 文についてみると，27 文が含意関係 (⇒)，20 文が定義 (⇔) である．そのうちの 30 文について，条件部を示す特徴語がある．主題部については,「～は,」「～が,」「～も,」を手掛かりとする．これらの語も含め，国民年金法全 148 条中 100 条，富山県条例第 54 号全 10 条，千代田区条例第 53 号全 28 条の計 501 文を分析し，84 種類の手掛かり語を抽出している．

表 2.2: 標準的な表現への言い替え

表現	言い替え	頻度
NP_1 の SN による NP_2	NP_1 に SN する NP_2	16
NP の SN を行う.	NP を SN する	8
NP の SN をする場合	NP を SN するとき	6
NP にあっては	NP の場合	5
NP の SN 上	NP を SN するとき	4
NP の SN に当たっては	NP を SN するとき	3
NP をもって	NP により	3
NP の SN のために	NP を SN するため	3
NP への SN がされた	NP へ SN された	2

(2) 様相部の同定

文末の述部に様相表現がある場合，係り受け構造上で，様相表現が支配する部分を同定する．この部分は一般に効果部であり，出力のための暫定表現に様相演算子，O, M, P を加える．それぞれ，義務，可能，許可に対応する．

(3) 標準的な表現への言い替え

法文は一定の表現により書かれているが，それでも,「〜N する場合」「〜N するとき」等，表現は異なっていても，それぞれが同じ意味と捉えられる表現がある．このような表現を一方の表現に統一して，論理式への変換の一貫性を保つために，言語表現の言い替えをする．上例の場合,「〜N する場合」を「〜N するとき」に言い替える．ここで，N はサ変名詞とする．富山県条例第 54 号 (全 10 条 34 項), 千代田区条例第 53 号 (全 28 条 81 項)，所得税法 (全 244 条中 100 条 255 項 247 号) 全 530 文を分析した結果を表 2.2 に示す．表では，名詞句を NP，サ変名詞を SN としている．

2.3.3 構成素の論理表現の生成

動詞句 (節)，名詞句の意味解析を行い，その論理表現を生成する．以下，それぞれについて説明する．

(1) 格解析

動詞句の意味解析としては，格構造を解析する．格解析について，格構造，格フレーム，解析手続きについて概略を説明する．

(a) 格構造

格構造は，動詞とそれを修飾する名詞句が作る構造で，行為とその役割を表す．例えば,「県の機関が権限を行使する」の格構造は,「行使する」という行為，その動作主「県の機関」，対象「権限」からなる．格構造の構成素となる名詞句と後置する助詞等とを格要素と呼ぶが，以下では簡単のため，これも名詞句と呼ぶ．

(b) 格フレーム

格構造の解析は，格フレームと呼ぶ知識を用いて行う．格フレームは動詞の一つの語義に対応する．従って，一つの動詞は一般に複数の格フレームを持つ．

V: $\{CF_1\ CF_2\ ...\ CF_n\}$

格フレームは格スロットのリストである．

CF_i: $\{CS_1\ CS_2\ ...\ CS_m\}$

各格スロットは動作主や対象などの役割を示す名詞を規定するものである．すなわち，格スロットは役割の名前，構文制約，意味制約，用例といった情報からなる．

CS_i: (格名, 構文制約, 意味制約, 用例)

構文制約はどのような格助詞が名詞に後置するか，意味制約は名詞句の主名詞の意味カテゴリ，用例は名詞の例である．

(c) 解析処理

格解析は，文の格要素の並びがどの格フレームともっともマッチするかを調べる処理である．すなわち，どの格要素とどの格スロットがマッチするかを調べる．日本語は語順の自由度があることから，格フレームの格スロットが 2 個，文の格要素が 2 個としても，4 通りの場合がある．これを格フレームの数だけ行う．4 通りのうちの一つについて，ここでは説明のために，格フレームをラムダ表現を使って書くことにし [5]，次の文を例に説明する．

県の機関は権限を行使する．

動詞「行使する」は格として，動作主 (agent) と対象 (object) をとると仮定する．

行使する: $\lambda x_2 \lambda x_1$ 行使する ($agent$:org: x_1, $object$:$power$: x_2)

「県の機関」「権限」の論理表現は以下とする．

県の機関: 県の機関 (y_1)
権限: 権限 (y_2)

「県の機関が権限を行使する」の係り受け関係が次のように解析されたとする．

(NP: 県の機関が (NP: 権限を VP: 行使する))

名詞句と動詞句が結びついて動詞句となるが，これは次の規則で捉えられる．

$VP \rightarrow NP\ \ VP$

この規則に意味を合わせたものを書くと次のようになる．

$VP(sem_2(sem_1)) \rightarrow NP(sem_1)\ VP(sem_2)$

そうすると，まず，(NP: 権限を VP:行使する) について上式の意味を適用すると次のようなる．

$\lambda x_2 \lambda x_1$ 行使する ($agent$:org: x_1, $object$:$power$: x_2) (権限 (y2))
$= \lambda x_1$ 行使する ($agent$:org: x_1, $object$:$power$: 権限 (y_2))

同様に，(NP:県の機関が VP:権限を行使する) に上式の意味規則を適用すると次のようになる．

λx_1 行使する ($agent$:org: x_1, $object$:$power$: 権限 (y2)) (県の機関 (y1))
$=$ 行使する ($agent$:org: 県の機関 (y_1), $object$:$power$: 権限 (y_2))

さらに，この式から，県の機関 (y1) と権限 (y2) を取り出すと，次式が得られる．

県の機関 (y_1) $\wedge$ 権限 (y_2) $\wedge$ 行使する ($agent$:org: y_1, $object$:$power$: y_2)

なお，実際の文では，ゼロ代名詞を考慮して，名詞を補完する解析が必要である．

(2) 名詞句の意味解析

名詞句は日本語では，(a) 用言連体修飾句，(b)「A の B」，(c) 複合名詞がある．

(a) 用言連体修飾句

動詞句＋名詞句という構造をしており，動詞句の格解析結果と名詞句の意味解析とを合成する．合成には大きく 2 つの場合がある．一つは，名詞句の主名詞が動詞句の主動詞の格要素である場合である．この場合は格解析を行う．もう一つは，動詞句と名詞句との関係が格関係以外の関係である．それには，同格 (「〜すること」)，事象と結果 (「魚の焼けるにおい」)，名詞句が動詞句の格要素の名詞を修飾する関係 (「枝が折れた木」) という場合がある．ここで,「枝が折れた木」は「木の枝」が「折れた」という意味で，名詞「木」が動詞句の格要素の名詞「枝」を修飾している．法文では一般に格関係，同格が見られる．

(b) A の B

日本語に数多く出現する形である．A と B を結ぶ何ならかの動詞が省略さらたものと解釈されるものである．そういう意味で，格解析のような考え方で解析される．一般に，A あるいは B の意味に応じて関係を表すフレーム型知識を用意して格解析のように解析する [18].

(c) 複合名詞

「A の B」のように名詞の間に関係を求めて解析することができるが，法令のように特定の領域に対する文書の場合，複合名詞が一つの語として扱えるなら，その方が簡潔であろう．

2.3.4　文全体の論理表現の生成

最初に求めた全体の論理構造の骨格に前節で求めた各構成素の論理構造をあてはめて，文全体の論理構造を求める．これまで触れてこなかったが，1 階述語論理では変数があるので，その量化子の表現が必要である．英語では “a” や “the” の冠詞に対して，$\forall$ や $\exists$ の量化子の表現を対応させる方法が知られているが，実際の英文の解析システムでは，英文の特徴に基づく意味関数やアルゴリズム [2] が工夫されている．法文の論理表現に対する量化子については別の機会に論じる．

2.3.5　法文の解析システム

上述の考え方に基づくシステムを実装し，改良中である．最初の版 (2005) は perl を用いて作成された [1, 8]．主に，富山県条例第 54 号，千代田区条例第 53 号の分析に基づいて作成された．形態素解析は JUMAN[10] と呼ばれるプログラムを用いている．構文解析は KNP[9] と呼ばれるプログラムを用いている．システムは金沢市の条例，広島市の条例等に適用して，動作することを確かめたが，辞書が不十分なためもあり，解析できる文は半数程度であった．

その後，所得税法の一部の分析も行い，上記システムを ACL Lisp により再実装した [13, 14]．形態素解析と構文解析は前と同様のものを用いている．このシステムでは，格フレームの量を増やし，日本語彙体系 [4] と呼ぶ大規模なシソーラスも利用するようにし，例えば，格解析の精度は 78%程度になっている．

ここでは，形態素解析，構文解析，意味解析と自然言語処理技術を順に用いる方法について示したが，文とその論理式からなるコーパスを利用して，文の論理式を求める変換器を機械学習する方法についても研究を進めている [12]．近年，形態素解析，構文解析等は，大量のコーパスと機械学習による方法により，よい精度の解析が可能となっており，そのようなアプローチを論理式への変換に試みて

いる.

2.4 言い替え–複数文にわたる文等に関する処理–

先に法文の言語表現の節において，法文は一般に要件効果構造を表すが，国民年金法などをみると，条文によっては以下の場合があると述べた.

- 1 文に複数の要件効果が記される場合,
- 号の列挙と合せて要件効果が記される場合,
- 前記の二つの場合が混合した形式で要件効果が記される場合

2 番目と 3 番目の場合，他の文を参照する表現があるが，例えば,「以下の号に該当するものを」とあるとき，これをそのまま論理式で表しても，論理式が文脈参照するわけではないので，そもそも証明に用いることができない．このような表現の場合，元の法文を解析し論理表現に変換しても意味がなく，参照表現は実体を表す内容に置き換える必要がある.

号の列挙や他条文の参照等により，1 文に複数の要件効果が記される法文は，論理式による表現でも問題があるが，人間にとっても，繁雑で読みにくい表現であると言える．このような文は，要件効果毎に分けて，論旨や内容を分かりやすくする方がよいように思われる.

また，所得税法や国民年金法などの条文の中には，算術計算を表すものがあるが，これらはむしろ数式を用いた方が厳密で分かりやすいとも言える.

以下では，複数の要件効果の扱いに関することや読みやすさについて論じる.

2.4.1 複数の要件効果の同定

要件効果毎に意味のある式を求めるため，以下のような処理を考える.

- 要件効果毎に分割する.
- 参照表現を号の内容で置き換える.

(1) 各要件効果の同定

先に例に上げた国民年金法第 8 条 (2.2.2 節，図 2.15) は複数の要件効果が号の列挙とともに記されていたが，これを例に説明する．この条文では，条件部に次の 3 つの場合が記されている.

前条の規定による被保険者は，第7条第1項第2号及び第3号のいずれにも該当しない者については第1号から第3号までのいずれかに該当するに至つた日に，被保険者の資格を取得する． 一　20歳に達したとき． 二　日本国内に住所を有するに至つたとき． 三　被用者年金各法に基づく老齢給付等を受けることができる者でなくなつたとき．
前条の規定による被保険者は，20歳未満の者又は60歳以上の者については第4号に該当するに至つた日に，被保険者の資格を取得する． 四　被用者年金各法の被保険者、組合員又は加入者の資格を取得したとき．
前条の規定による被保険者は，第7条第1項第2号及び第3号のいずれにも該当しない者でも20歳未満の者又は60歳以上の者でもない者については同号又は第5号のいずれかに該当するに至つた日に，被保険者の資格を取得する． 四　被用者年金各法の被保険者、組合員又は加入者の資格を取得したとき． 五　被扶養配偶者となつたとき．

図 2.24: 国民年金法第8条の条件毎に条文を分割したもの

- 同条第1項第2号及び第3号のいずれにも該当しない者については第1号から第3号までのいずれかに該当するに至つた日に，
- 20歳未満の者又は60歳以上の者については第4号に該当するに至つた日に，
- その他の者については同号又は第5号のいずれかに該当するに至つた日に，

上記に，主題部および対応する号等を加える[7]と図2.24となる．さらに，次に述べるようにして，号の内容を埋め込むと全部で3つ要件効果表現が得られる．

(2) 参照表現を号の内容で置換

国民年金法第30条では，図2.25に示すように，要件に示される条件の内容が号に示される．図2.25の下線が号を参照している箇所で，そこを号の内容で置き換えることにより実際の条件が得られる．このとき2つの表現が可能である．下線

[7]図では，さらに,「同条」を指示対象の「7条」に置き換え，効果部の「それぞれ」は条件毎に文を分けたので削除し,「その他の」はその前の二つの条件に該当しないとして，前の条件を具体的に書き出している．

(支給要件)
第三十条　障害基礎年金は，疾病にかかり，又は負傷し，かつ，その ... 疾病 (...) について ... 診療を受けた日 (...) において 次の各号のいずれかに該当した者が，... 一年六月を経過した日 (...) において，その傷病により ... 障害の状態にあるときに，その者に支給する．
一　被保険者であること．
二　被保険者であった者であって，日本国内に住所を有し，かつ，六十歳以上六十五歳未満であること．

図 2.25: 条件の内容が号に記される例 (国民年金法第 30 条)

障害基礎年金は，疾病にかかり，又は負傷し，かつ，その ... 疾病 (...) について ... 診療を受けた日 (.....) において 被保険者である 者が，... 一年六月を経過した日 (.....) において，その傷病により ..... 障害の状態にあるときに，その者に支給する．

障害基礎年金は，疾病にかかり，又は負傷し，かつ，その ... 疾病 (...) について ... 診療を受けた日 (...) において 被保険者であった者であって，日本国内に住所を有し，かつ，六十歳以上六十五歳未満である 者が，... 一年六月を経過した日 (...) において，その傷病により ... 障害の状態にあるときに，その者に支給する．

図 2.26: 条件を号を埋め込んだ文

の箇所を選言で置き換えて 1 つの式にする場合と，2 つの号を別々に置き換えて 2 つの式にする場合である．これを簡単な論理式でみてみる．元の式が次の式であるとしよう．

$$A \wedge D \Rightarrow E$$

今，A が $B \vee C$ だとする．これで A を置き換えると次の式になる．

$$(B \vee C) \wedge D \Rightarrow E$$

元の文をこのように言い替えるのが一つの方法である．上例であれば，下線のところを次の文で言い替えることになる．

(併給の調整)
第二十条　遺族基礎年金又は寡婦年金は，その受給権者が他の年金給付（付加年金を除く.）又は被用者年金各法による年金たる給付（当該年金給付と同一の支給事由に基づいて支給されるものを除く．以下この条において同じ.）を受けることができるときは，その間，その支給を停止する．老齢基礎年金の受給権者が他の年金給付（付加年金を除く.）又は被用者年金各法による年金たる給付（遺族厚生年金並びに退職共済年金及び遺族共済年金を除く.）を受けることができる場合における当該老齢基礎年金及び障害基礎年金の受給権者が他の年金給付（付加年金を除く.）を受けることができる場合における当該障害基礎年金についても，同様とする．

図 2.27: 国民年金法第 20 条 1 項

被保険者であるか，又は被保険者であった者であって，日本国内に住所を有し，かつ，六十歳以上六十五歳未満である

実際，これを元の文に埋め込んだら，大変，読みづらい文になるであろうから，実際の条文は号を用いて表現されていると考えられるが，これがなくても，本文自体，大変読みづらい．

もう一つの表現は，$(B \vee C) \wedge D \Rightarrow E$ という式に等価な次の 2 式に基づいて言い替えをする．

$$B \wedge D \Rightarrow E$$
$$C \wedge D \Rightarrow E$$

この式に基づいて言い替えた場合は，図 2.26 のようになる．

なお，上記は，条文に複数の要件効果が記されている場合の対処であるが，一文に要件効果が何個あろうとも，他の条文を参照している表現はそのまま論理式に表現しても意味がない．上例の第 8 条であれば,「前条の規定による」「同条第一項第二号第三号の」がそのような表現である．論理表現としては，実質的内容の式で表現する必要がある．

2.4.2　読みやすさのための言い替え

上述は，論理式に変換するための処理であるが，要件効果を一つづつ分ける，論理の骨子を見やすくする等の処理により原文を言い替えて，原文を読みやすくす

A の受給権者が B または C の年金給付を受けることができる場合，A の支給を停止する．
A: 遺族基礎年金または寡婦年金
B: A 以外の年金給付で，付加年金でないもの
C: 被用者年金各法による年金給付 (A の受給権と同一の支給事由に基づいて支給されるものを除く．以下この条において同じ．)
D の受給権者が E または F の年金給付を受けることができる場合，D の支給を停止する．
D: 老齢基礎年金
E: D 以外の年金給付で，付加年金でないもの
F: 被用者年金各法による年金給付（遺族厚生年金・退職共済年金・遺族共済年金を除く．)
G の受給権者が H の年金給付を受けることができる場合，G の支給を停止する．
G: 障害基礎年金
H: G 以外の年金給付で，付加年金でないもの

図 2.28: 国民年金法第 20 条 1 項の言い替え

ることが期待される．読みやすいことは，正しい理解につながるので，法令利用者，学習者，作成者にとってよいことであろう．

例を国民年金法第 20 条 1 項についてみてみよう．この条文を図 2.27 に示す．この条文は 3 つの要件効果を記している．括弧内にも条件が書いてあり，大変分かりにくい条文の例と言える．このような場合，要件効果を個別に分けるとともに，括弧内の条件も含め，それらを分かりやすく別表示すると見やすくなる．その例を図 2.28 に示す．

法令は様々のことを文章で表しているが，情報によってはそれに適した表現がある．所得税法や国民年金法では，数式で表現した法が厳密で理解しやすい場合もある．そのような例として，国民年金法第二十七条七号とその数式表現を図 2.29 に示す．図 2.29 において「前号に規定する保険料 4 分の 3 免除期間の月数」は第二十七条六号に規定されるもので，それが “保険料 4 分の 3 免除期間 × 5/8” である．なお，第二十七条自体は，このような各号の計算に基づいて計算されるもので，これも数式で表現すると分かりやすい．

法文の可読性を高めるには様々なことを考慮する必要がある．例えば，法文では接続詞が使い分けられており [3, 11, 23]，それにより文構造が明確になり，これ

保険料 4 分の 3 免除期間の月数から前号に規定する保険料 4 分の 3 免除期間の月数を控除して得た月数の 8 分の 1 に相当する月数
{(保険料 4 分の 3 免除期間 − 保険料 4 分の 3 免除期間 × 5/8) × 1/8} の月数

図 2.29: 国民年金法第 27 条 7 号とその数式表現

を手掛かりにした法文の表示も可能である [22]．条文中にある他の条文を参照している表現についても，人間が読む場合においては，参照先が何を書いてあるかは専門家でない限り，すぐには分からない．参照する他条文の該当部分の内容がある程度分かるような提示も同様に考える必要がある．

第3章　法律の論理推論

3.1　法令に求められる無矛盾性や完全性の論理学的定式化

個々の法令文を命題あるいは述語論理式と捉え，それら法令文の集合を知識ベースと捉えたときに，この知識ベースには命題集合としての論理学的性質を論じることができる．従来的な法的推論で代表されるものは個々の法律規則を IF-THEN 型の推論規則と捉え，その規則間に連鎖を行って所定の推断結果を得るものである．これは 1980 年代に主に研究されたエキスパートシステムという考え方に依拠する．このとき，以下のような論理学とのアナロジーを考える．

1. 知識ベースが無矛盾であるとは，知識ベースを理論 (theory) と考え，代入と Modus Ponens について飽和した命題集合を作るとき，φ と $\neg\varphi$ の両方を同時に含まないこと．

   例 1) 集合 $\{A, A \to B, B \to C, A \to \neg C\}$ は Modus Ponens によって $\{A, A \to B, B \to C, A \to \neg C, B, c, \neg C\}$ と拡張され，C と $\neg C$ を同時に含む矛盾した命題集合である．

   例 2) 「この公園には車馬の通行を禁ずる」,「この公園にはうば車の通行を許可する」に対しては「うば車は車である」という常識の補完と「禁止」と「許可」が肯定・否定と同様に対立する概念であるという知識を持ち込むことにより「うば車は禁止され，かつ許可される」ことが結論される．

2. 知識ベースが完全であるとは，法律推論において必要な場合がすべて列挙され，抜けがない状態であること．

しかしながら実際法律の体系に記載されていることは IF-THEN 型の推論規則ではない．規則制定の精神，用語の定義，法律適用における場合分け，その他の付帯条件など必ずし も論理規則の形式になるものではない．さらに例 2 で見たように法令文を Modus Ponens のように連鎖させる場合においても，解釈の幅によっ

てその連鎖の可能性が変化する．法令文に対してはさまざまか解釈が可能である．例えば「乗り物は立ち入り禁止」に対しては，

- 拡張解釈 「馬も立ち入り禁止」
- 縮小解釈 「うば車は入ってもよい」
- 限定解釈 「人間は入ってもよい」
- 類推解釈 「鹿も入ってはいけない」

の四種類の解釈が可能である．よって解釈の幅を固定しないことには推論規則の連鎖は不可能である．また推論規則の連鎖を行おうとすれば，各推論規則は節形式 (clausal form) である必要がある．このことは後件部に ‘or’ を排除する必要があるなど，非常に窮屈な表現を要求することとなる．このことを次に論じる．

3.2　法令表現のための形式論理体系と形式推論

3.2.1　論理和標準形による導出

論理式のうち，一番上位の構造が論理和 (logical or) すなわち ‘$\vee$’ によって連結された式

$$A_1 \vee A_2 \vee (B_1 \wedge B_2)$$

は論理和標準形 (disjunctive normal form) と呼ぶ．(i) 論理和標準形は $A \to B$(論理的含意; logical implication) を $\neg A \vee B$ に置換し，(ii) 論理和が論理積 (logical conjunction) の内側にある式は，ド＝モルガンの法則

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$$

によって置換することにより得ることができる．

さらに論理式を命題論理から一階述語論理に拡張し，存在量化子 ($\exists$) はスコーレム定数およびスコーレム関数を導入して排除した以下の形式:

$$\forall x_1 \forall x_2 \ldots [C_1 \vee C_2 \vee \cdots \vee C_n]$$

をスコーレム標準形と呼ぶ．この形式では変数はいずれも汎化量化子 ($\forall$) によって束縛されており，かつ量化子はすべて式の前に集められ (冠頭形)，C_i は内部に論理積を含んでもよい．導出 (resolution) とは，$\neg A \vee B$ と $\neg B \vee C$ のようにある式

(ここではB) の肯定・否定のペアを見つけて，そこから $\neg A \vee C$ という式を作る操作を指す．これは '$\vee$' と '$\rightarrow$' に関する書き換え規則により $A \rightarrow B$ と $B \rightarrow C$ という二つの式を見つけて $A \rightarrow C$ という規則を生成する操作 (三段論法: syllogisms) に相当する．さていますべての変数は汎化量化子 ($\forall$) によって束縛されているためにこの量化子は省略して書くこととし，残りの論理和標準形の部分がすべて基礎論理式 (述語一つからなら論理式) で構成され，かつ否定辞 ($\neg$) を含まない基礎論理式が高々一つのものをホーン節 (Horn clause) と呼ぶ．例えば

$$\neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_n \vee B$$

はホーン節であり，これは '$\vee$' と '$\rightarrow$' に関する書き換え規則により

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n \rightarrow B$$

という推論規則であるとみなすことができる．実際，左辺が空の形，右辺が空の形をそれぞれ事実 (fact)，質問 (query) と呼び，ホーン説は次の三種類に分類される．以下では '$\rightarrow$' に代えて左右を反対にし ':-' を左向き矢印として用いる．また '$\wedge$' に代えて ','(コンマ) を用いる．

推論規則: `B :- A1, A2,...,An`
事実: `A1 :-`
質問: `:- B1,B2x`

法令文を記述する規則がすべてホーン節で記述されれば，推断は論理プログラミング (logic programming) によって自動化されるが，先に述べたように以下のようなものはホーン節化不可能である．

- 用語の概念定義，法適用の精神など IF-THEN 構造にはならないもの．
- 後件部に '$\vee$' を含むもの

3.2.2 付帯条件の記述

法令文のうち，概念定義やその他の付帯条件部は推論式の前件部とは別に以下のような記述が考えられる．

```
B :- A1, A2, A3 || C1, C2.
```

このうち `C1`，`C2` などの部分は概念階層など，本来一階述語論理ではない条件記述を書くために用いる．ただし実際は '`C1= (P1 < P2)`'(P2 は P1 を包摂 (subsume) する) などの表現はプログラム実行時には

$$P1(x) \rightarrow P2(x) (= \neg P1(x) \vee P2(x))$$

などのように一階述語論理に翻訳されている必要がある．また上記の式の意味は

$$B \leftarrow A_1 \wedge A_2 \wedge A_3 \wedge C_1 \wedge C_2.$$

であるとしても，$C_1 \wedge C_2$ を評価するタイミングを $A_1 \wedge A_2 \wedge A_3$ と違えるなど (committed-choice 型)，オペレーショナル・セマンティクス (実行時の意味づけ) で付帯状況を表す工夫が必要になる．

3.2.3　イベントとプロパティの区別

本来，一階述語論理の述語は静的なプロパティ (property) 記述を表すものである．例えば「X が『飛ぶ (fly)』という属性を持つならばその下位概念として『湖上を飛ぶ (fly_over_the_lake)』属性も持つ」はずである．

$$\forall x[fly(x) \rightarrow fly\_over\_the\_lake(x)]$$

一方，述語を一回性のイベント (event) であるとすると,『湖上を飛んだ』なら『飛んだ』ことになり，

$$\forall x[fly\_over\_the\_lake(x) \rightarrow fly(x)]$$

となる．これは述語の用法におけるイベントとプロパティの混同である．このような恣意性を免れるためには Davidsonian semantics を用い，イベント変数・イベント定数を導入し，

$$fly\_over\_the\_lake(e) \wedge agent(e, a)$$

「『fly_over_the_lake』というイベント e におけるエージェント (動作主) が a である」

という記述をする必要がある．また，この例のように各イベントにおける動作主・対動作主・目的などを明示するために agent, co-agent, object などのロール (役割) 記述のための述語のセットを用意する必要がある．また定数 e がイベントのソートに属し，定数 a が個体のソートに属することを明示するためにソート宣言を ‘:’(コロン) によって行い，

$$fly\_over\_the\_lake(e : event) \wedge agent(e : event, a : ind)$$

などのように表現を拡張することとする．Davidsonian semnatics は近年の状況理論 (situation semantics) を用いると

$$e\colon \langle\!\langle fly\_over\_the\_lake, (a : ind)^{agent} \rangle\!\rangle$$

のようによりコンパクトに書くことができ，一階述語論理のシンタックス・シュガーとして活用できる可能性がある．

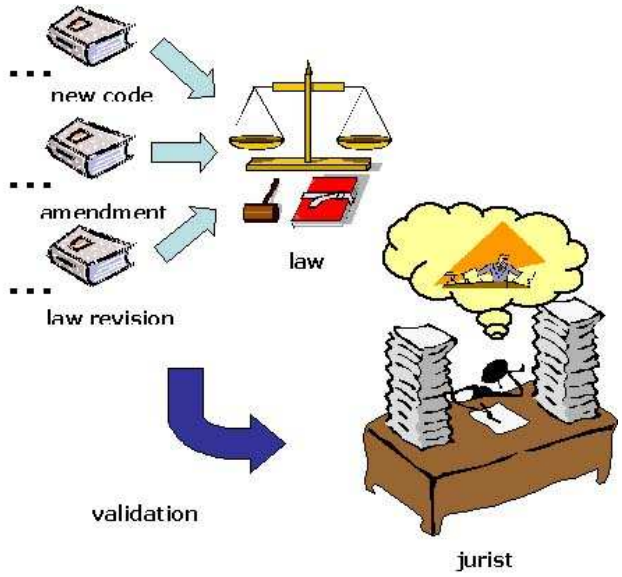

図 3.1: 法律改正の問題

3.3 大規模法令論理式ベースの解析

本節では具体的な問題として，大規模な法知識ベースの維持の問題を考える．図 3.1 は大規模な知識ベースに対して法改正に苦しむ法律家のようすを描いたものである．現実社会の要求に応えるためには法の改正は必要であり，そのつど法律家はその影響範囲を同定するとともに無矛盾性の検出を行わなければならない． このような要求に応えるために，これまでの法的推論で主に扱われてきた IF-THEN 型の推論規則に加え，概念階層と定義矛盾に関する問題を扱うこととする．すなわち，

1. 推論規則の集合としての妥当性の問題
2. 概念階層の構築と定義矛盾，重複，循環の問題

を扱う．特にここで扱う不具合 (discordance) は，論理的矛盾 $\varphi \wedge \neg\varphi$ だけではない．‘¬’ というシンボルに表されるように，法律文書に中に陽に命題の肯定と否定が共存するようなケースは稀であるし，否定の概念を形式的に記述するのに論理記号の ¬ を用いるかどうかは主観による．以下の例，

防衛行為は正当である．

$\forall x[\text{行為}\,(x) \wedge \text{防衛}\,(x) \rightarrow \text{正当}\,(x)]$

防衛行為は罪にはあたらない.
$\forall x[\text{行為}\,(x) \wedge \text{防衛}\,(x) \rightarrow \neg\text{罪}\,(x)]$

においては同じ概念を記述するのに否定記号を用いるかどうかに恣意性がある．本研究で対象とする不具合には以下のようなものを含む.

1. 論理的矛盾
2. 反対語および両立不可能 (incompatible) な概念などを含んだ広義の対立概念.
3. 定義の循環.

まず論理的矛盾とは異なる対立概念の例として，ある物体 x が同時に car であり ship であることは不可能である.

$$\forall x[car(x) \wedge ship(x) \rightarrow \bot]$$

car と ship は概念として両立不可能であることを，

$$car \wedge ship \vdash \bot$$

のように宣言を行い，この二つが否定に準ずる対立関係であるとする．さらに vehicle(乗り物) が car(車) と ship(船) を下位範疇化しており，その共通集合は空 $\bot$ であることを示すと，

$$\begin{cases} vehicle \sqsubseteq \top \\ car \sqsubseteq vehicle \\ ship \sqsubseteq vehicle \\ \bot \sqsubseteq car \\ \bot \sqsubseteq ship \end{cases}$$

は概念階層上のラティス (束: lattice) をなす．ここでラティスとは元 a, b に対して $a \sqcap b$(共通の最大下位概念)，$a \sqcup b$(共通の最小上位概念) が一意に定まるような集合をさし，

- $a \sqcap a = a, a \sqcup a = a$ (べき等則)
- $a \sqcap (b \sqcap c) = (a \sqcap b) \sqcap c,\ a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c$ (結合則)
- $a \sqcap b = b \sqcap a,\ a \sqcup b = b \sqcup a$ (交換則)
- $a \sqcap (a \sqcup b) = a,\ a \sqcup (a \sqcap b) = a$ (吸収則)

を充たす．さらに分配則，有界のもの

- $a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c),\ a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$ (分配則)
- $\bot$ と $\top$ が存在する．(有界 (bounded) ラティス)

を有界分配ラティスという．ちなみに上記の構造に補元が存在するとブール代数となる．

不具合検出を行うには，すべての規則を同時に起動させてみる必要がある．例えば，

$$\forall x[vehicle(x) \rightarrow prohibied(x)]$$
$$\forall x[vehicle(x) \rightarrow admissible(x)]$$

という二つの規則に対して，それらを同時に起動する vehicle(x) という知識を人工的に追加し，無矛盾検出を行うことができる．

$$\{vehicle(x)\} \cup \left\{ \begin{array}{l} vehicle(x) \rightarrow prohibited(x) \\ vehicle(x) \rightarrow admissible(x) \end{array} \right\} \vdash \bot$$

where

$$\{prohibited(x) \wedge admissible(x)\} \vdash \bot.$$

ここで prohibited と admissible が対立概念であれば，$\bot$ を産出する．このような人工的に加える知識を assumptive facts と呼ぶ．図 3.2 はこのような assumptive facts を一斉に加えて推論規則の対立を発見するためのアルゴリズムである．図中，assumptive facts は $P_{fact0}(a)$ および $P_{fact1}(b)$ である．また，

```
P2(x,y) :- P3(x), P4(x)
P1(x,y) :- P2(x,y)
P0(x,y) :- P1(x,y)
P5(x) :- P4(x)
P6(x) :- P5(x)
```

などの規則を連鎖させたものである．ここに $P0(x, y) \wedge P6(x) \rightarrow \bot$ であるとすると，二つの assumptive facts からこれらの規則群が矛盾を含んでいることがわかる．

さて，このようなアルゴリズムを実装するにはいくつかの問題がある．まず assumptive facts 全体の発見であるが，すべての規則を exhaustive に検索し，その前提部を集め，かつ重複を取り除くのに効率の良いアルゴリズムが望まれる．また，すべての assumptive facts が同時に必要かというとそうでもない．

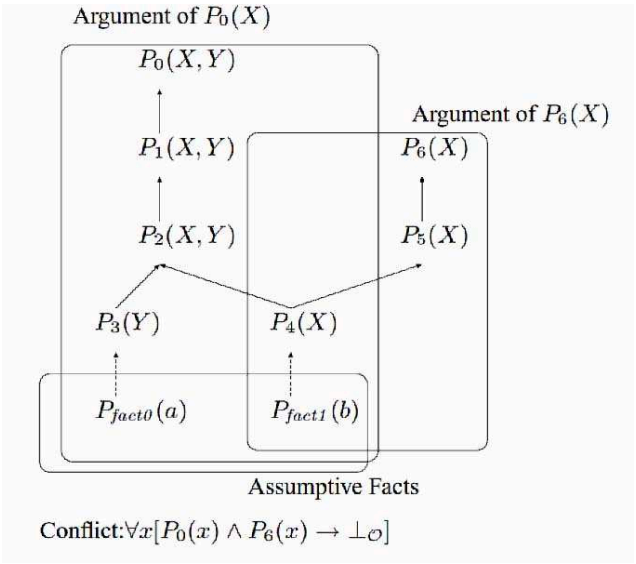

図 3.2: Assumptive facts からの矛盾

```
Q(x) :- P1(x), P2(x).
-Q(x) :- P2(x), P3(x).
```

のような規則のペアに対して $P_1(x) \wedge P_2(x) \wedge P_3(x)$ は確かに $Q(x) \wedge \neg Q(x)$ という矛盾を起こす．しかし $P_1(x) \wedge P_2(x)$ や $P_2(x) \wedge P_3(x)$ が現実的に可能な組み合わせでも P1,P2,P3 がすべて同時に成り立つような組み合わせは現実的でない可能性がある．これらがオントロジーからの対立概念によってすべて発見されていればよいが，そうとは限らない．したがって集められた assumptive facts の集合自身の中に既に対立の概念が含まれていないかどうかを精査する必要がある．また先に述べたように帰結部に ‘or’ を含むような構造を含め，すべての規則がホーン節として不適当な形態をしている場合の式変形，あるいは連鎖を行わせるにあたって引数の数を一致させる式変形など，法推論とは本質的ではないところで恣意的な操作を行う必要がある．図 3.3 は大規模知識ベースに対して assumptive facts を加え，知識ベースに含まれる法令文を

- 安定な知識
- 半安定な知識 (矛盾に寄与する可能性がある知識)

に切り出し，極小の矛盾領域を同定するようすである．

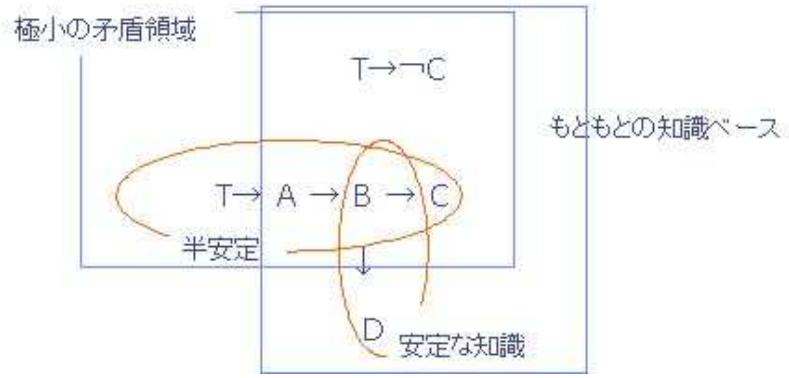

図 3.3: 安定領域の検出

また法令文の中に含まれる定義の循環も広い意味で法令文の不具合とみなすべきである．いま複数の IF-THEN 型の推論規則に対し，帰結部にあるような情報が互いの条件部にある場合は規則の連鎖が循環してしまう．法律文書の無矛盾性解析においては論理的矛盾に加えてこのような定義の循環も検出される必要がある．

3.4 法令論理式推論のためのオントロジー

3.4.1 階層構築の指針の問題

オントロジーにおいて中心的な役割を果たすのは概念階層である．すなわち各概念間で上位・下位の関係があるときにそれらに包摂関係 (subsumption relation) を宣言したグラフ構造である．しかしながら上位・下位の関係は一意ではない．

- 集合とその部分集合の関係
- 集合とそのメンバーの関係
- クラスとインスタンスの関係

などなど多様なものが上位・下位であると呼ばれる．この研究分野では以前より，次のようなものに対して包摂関係が議論されてきた．

- type(型) の関係
- sort(ソート) の関係

- class/instance の関係
- role や property の関係

このうち例えば sort の関係に関しては個体の集合によってモデル化できるもののみを対象とし，その個体の集合間で包含関係によって上位・下位を定めるものである．たとえば「鉛筆」と「筆記具」に対して [[鉛筆]] をすべての鉛筆からなる集合，[[筆記具]] を筆記を集めてきた集合としたとき，[[鉛筆]] $\subseteq$ [[筆記具]] となることより，この二つの概念に包摂関係を定義することができる．しかしながら抽象的な概念や数え上げができない液体のような概念に対してこの方法は適用できない．先に述べたように法律知識の無矛盾性を検証する上で対立概念をオントロジーから探す手法は有用であるが，オントロジーがどのように構築されているかを検討することなしには，この対立概念も定義できないことになる．

3.4.2　オントロジー・マッチング

法律用語のオントロジーは多数のサイトでそれぞれ独自に定義されてしまう運命にあり，前述のように概念階層の意味も異なる．しかしながら複数の概念階層を結びつけてより大規模なオントロジーが構築できれば，より客観性を増すとともに扱う語彙も豊富にできる．するとオントロジー間において対応する概念を見つけ，概念階層のグラフ構造を融合する試みは大変有用である．しかしながら，一致のプロセスでは概念の対応，上位下位概念の対応とともに一般にグラフ融合のアルゴリズムの効率化が求められるなど，研究課題が多く残されている．

図 3.4 はソートが IC(Identity condition) と呼ばれる固有の特徴を持つとして，これら IC を持つソートの間から二つの概念階層を関係づけようとする試み [27, 28] を表す．図中 w, w' は二つのオントロジーを表し，IC は上位概念より継承されるために S と S' に同じ属性が見つかった場合，S に対応する概念は w' 中グレーのノードで表示された領域に存在することが予想される．

3.5　非古典論理による法令工学の発展

3.5.1　否定概念の拡張

前節までにオントロジー上の対立，すなわち共通に下位範疇化できる概念が空 ($\bot$) であるときをもって対立する概念とする考え方を導入した．しかしながら否定の概念はこれだけではない．ここに非古典論理の典型的な否定概念として直観主

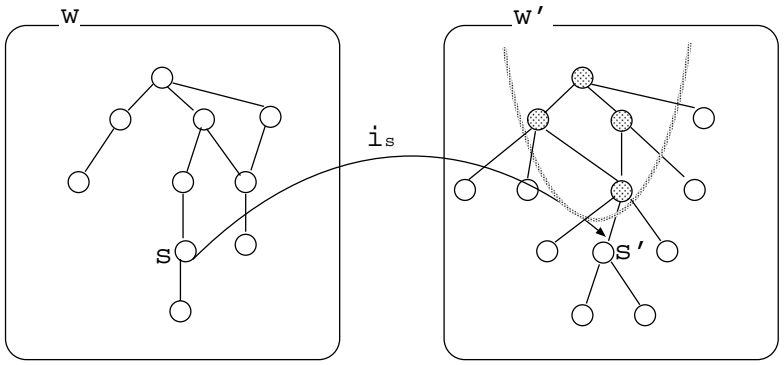

図 3.4: オントロジーのマッチング

義論理による否定の考え方を述べ，さらに段階的否定 (graded negation)・極小否定 (minimal negation) などの拡張を述べる．

直観主義論理

直観主義論理 (intuitionistic logic) は二重否定の消去

$$\neg\neg\varphi \rightarrow \varphi$$

を認めない論理である．これは自然言語の否定の概念をより忠実に反映した考え方である.「好きでないわけではない」と言ってもそれは「好きである」ということにはならない．古典論理は否定辞 $\neg$ をともなった概念と肯定概念とによって，世界を二分割するが，直観主義論理では，$\neg\neg\varphi$，$\neg\varphi$，φ によって三分割される．ここで φ は $\neg\neg\varphi$ の中に含まれ，

$$\varphi \rightarrow \neg\neg\varphi$$

は認めることとする．これにより，以下の性質はいずれも成り立たなくなる．

- 排中律 $\varphi \vee \neg\varphi$
- 対偶の一方向 $(\neg\beta \rightarrow \neg\alpha) \rightarrow (\alpha \rightarrow \beta)$
- ド＝モルガンの法則の一つ $\neg(\alpha \wedge \beta) \rightarrow (\neg\alpha \vee \neg\beta)$
- 含意に関わる書き換え $(\alpha \rightarrow \beta) \rightarrow (\neg\alpha \vee \beta)$

- 背理法 $(\neg\alpha \to \bot) \to \alpha$

これらはいずれも互いから互いを導くことができる同等な概念である．

段階的否定

否定の概念を相対化にすることで，より現実的な否定概念を表すことができる．

$$\Delta \vdash \neg_{\varphi}\psi$$

という式は段階化否定 (graded negation) と呼ばれ，Δ が実際に φ を導く

$$\Delta \vdash \varphi$$

のとき，

$$\varphi \wedge \psi \vdash \bot$$

によって定義される．これは ψ の否定を φ に相対化させたものであり，φ の存在がない限り ψ は否定されない．またこれにより否定の強弱を比較できる．いま

$$\delta = \neg(\alpha \vee \beta \vee \gamma) = \neg\alpha \wedge \neg\beta \wedge \neg\gamma$$

とすると，

$$\begin{aligned}
&\alpha \wedge (\neg\alpha \wedge \neg\beta \wedge \neg\gamma) = \bot \\
&(\alpha \wedge \beta) \wedge (\neg\alpha \wedge \neg\beta \wedge \neg\gamma) = \bot \\
&(\alpha \wedge \beta \wedge \gamma) \wedge (\neg\alpha \wedge \neg\beta \wedge \neg\gamma) = \bot
\end{aligned}$$

であるが，このうち $(\alpha \wedge \beta \wedge \gamma)$ は δ を最も強く否定する．この考え方により，最も弱い否定，極小否定 (minimal negation)

$$\Delta \vdash \ominus_{\alpha}\beta$$

は次のように定義される．

$\Delta \vdash \neg_{\alpha}\beta$
かつ，任意の $\Delta \vdash \neg_{\varphi}\beta$ であるような φ に対して
$\vdash \varphi \to \alpha$ であるならば $\vdash \alpha \equiv \varphi$.

法的推論，特に刑法の推論には刑に至る最小限の理由を持ちうるかどうかを判定する必要があるが，この最小限の理由を同定するときにこの極小否定の考え方が応用できる可能性がある [29].

3.5.2 含意の拡張

論理学の含意 '$A \to B$' は '$\neg A \vee B$' で意味づけされる．これは，含意の意味として「Aが真である状況においてBが偽であるのは不条理である」とする考え方

$$\neg(A \wedge \neg B)$$

に基づく．すなわち，上式を式変形していくと，

$$\begin{aligned} & \neg(A \wedge \neg B) \\ = \ & \neg A \wedge \neg\neg B \quad (\text{ド}=\text{モルガンの法則}) \\ = \ & \neg A \wedge B \quad (\text{二重否定の消去})x \end{aligned}$$

という過程を経るが，このうちのド モルガンの法則や二重否定消去は直観主義論理で述べたとおり，必ずしもいつも受け入れられる考え方ではない．これにより，含意は自然言語の「ならば」とは相容れない意味になる．例えば，

前提が偽なら帰結は何であっても真？
「太陽が西から昇るなら地球は平らだ」

帰結が真なら前提は何であっても真？
「太陽が西なら昇るなら地球は丸い」

など，現実の意味とは相容れない含意の関係を導くことになる．このことを改善するために，関連性の論理 (relevant logic) の適用が考えられる．関連性の論理とは，

- 前提と帰結の間に命題変数の共有

$$\Delta \to \Gamma \text{ のとき } \Delta \cap \Gamma \neq \varphi.$$

- 前提はすべて帰結に寄与する

$$\Delta \to \alpha \text{ だからと言って } \Delta, \beta \to \alpha \text{ としない.}$$

のように '$\to$' の両側で条件を付与したものである．

3.5.3 論理和の拡張

日常言語に現れる 'or' は論理和とするのは不適であり，場合により排他的or(exclusive or) であるケースがほとんどである．よって

$$\alpha \cdot \beta = (\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)$$

によって意味づけされる or の記号を用いた法令文の論理化が考えられる．

3.6　様相論理を用いた知識と信念の記述

3.6.1　知識と信念の様相オペレータ

極小否定 (minimal negation) の項で述べたように，責任を追及する原理としてある人の行為がある帰結の原因となりうるかどうかを問うことができる．しかしそれが刑事責任を負うかどうかは別問題である．例えば通常は薬であっても 100mg を超えると致死量となる劇薬を仮定する．A, B それぞれが 60mg の薬を C のコーヒーカップに入れて C がそれを飲んでしまった場合，A の行為は明らかに C の致死に責任がある．ところが，A は B が後に 60mg の毒を入れることを知らずに 60mg の薬を入れたならば刑事責任はない．この例のように各エージェントの知識状態を記述できる手法として知識と信念の様相オペレータを用いることが考えられる．

いまエージェント i が情報 φ を知っているということを K_i と書くことにすると，このオペレータは通常次のような要件を満たすことが要求される．

(K)　$K_i(\alpha \to \beta) \to (K_i\alpha \to K_i\beta)$
(T)　$K_i\alpha \to \alpha$
　　(α を知っているならば，α は事実である)
(4)　$\neg K_i\alpha \to K_i\neg K_i\alpha$
　　(α を知らないならば，知らないことを自覚している)
(5)　$K_i\alpha \to K_iK_i\alpha$
　　(α を知っているならば，知っていることを自覚している)

このようにオペレータ K_i は様相論理の $S5(= KT5 = KT45)$ を満たす．これに対してエージェント i が情報 φ を信じているという場合，$B_i\varphi$ と書いて，次の要件を満たすことが要求される．

(K)　$B_i(\alpha \to \beta) \to (B_i\alpha \to B_i\beta\ )$
(D)　$B_i\alpha \to \neg B_i\neg\alpha$
　　(α を信じているならば，$\neg\alpha$ は信じていない)
(4)　$\neg B_i\alpha \to B_i\neg B_i\alpha$
　　(信じていないならば，信じていないことを信じている)
(5)　$B_i\alpha \to B_iB_i\alpha$
　　(α を信じているならば，信じていることを自覚している)

K_i との一番の違いは $B_i\alpha$ であるからといって α であるとは限らないことであり，すなわち虚偽の情報を信じる可能性を残すことである．B_i オペレータは様相論理の KD45 となる．

3.6.2 通知の論理

通知 (communication) とはエージェント間で情報を伝えることにより，各エージェントあるいはエージェントのグループで知識と信念の状態が書き換わることを指す．通知が可能であるためには自分 i が情報 φ を自覚し ($B_i\varphi$)，相手 j がその情報を知っているかどうかに対して確度がなく ($\neg B_i(B_j\varphi \vee \neg B_j\varphi)$)，かつ通知の結果として相手 j がその情報を信じる ($B_j\varphi$) とともに，その情報を i も知っていると信じる ($B_jB_i\varphi$) ことになる．これは知識状態の動的な書き換えであり，かつ時間の推移を示すインデックスも必要となるため，知識の論理 (epistemic logic) に加えて，動的論理 (dynamic logic) もしくはモデルの書き換え (model updating)，時間の論理 (computational temporal logic) が必要となる．

もし通知を受けたエージェントが，もらった情報とそれまでの自分の知識と矛盾する場合，知識状態の修正 (belief revision) を行う必要がある．またこのような知識状態の修正が複数のエージェントに波及する場合に集団での修正を考える必要がある．さらに通知に限らず命令や依頼と言った行為も同様な枠組みで扱うとすると，信念 (belief)・欲求 (desire)・意図 (intention) をそれぞれ別オペレータとした BDI 論理が必要であるとともに，エージェントの行為に対するコミットを記述するための義務の論理 (deontic logic) が必要となる．

3.7 議論 (Argumentation)

近年の法的推論の会議 (JURIX や ICAIL など) においてもっとも論文数が多いのが論争に関わる話題である．これはもともと論駁可能な推論 (defeasible reasoning) として研究されていたものである．ここでは推論規則は次の二種類の意味で論駁可能である．

- 反駁 (rebuttal) 同じ前提から規則において異なる結論を導くこと．例えば $A_1, A_2, A_3 \to B$ に対して $A_1, A_2, A_3 \to \neg B$ など．
- アンダーカット (undercutting) 推論規則のうちの前提なる $A_1, A_2, A_3 \to B$ に対して A_2, A_3 は不成立とする．

議論とは前提から結論に至る複数の推論規則の連鎖を指すが，議論の強弱を決定するにはさまざまな要因が考えられてきている．例えば，解釈の項で述べたように前提と帰結を連鎖させる際の妥当性，および連鎖の長さなどが比較の対象となる．また複数の議論から別の議論を導き双方が受け入れ可能となるような「妥協」の概念も提案されている．

法令工学においては法廷における論争そのものが対象とはならないが，法令の妥当性を検討する際に論駁可能な推論規則によって議論を比較することも研究の視野に入ってくるものと思われる．

3.8　【ケース・スタディ】富山県の条例変更にともなう知識ベースの改編

富山県は業務効率改善のため，諸手続きを電子的な方法で代替できるよう規則を定めた．

> 富山県行政手続等における情報通信技術の利用に関する条例 (富山県条例第 54 号原文)
>
> 富山県行政手続等における情報通信の技術の利用に関する条例
> (目的)
> 第 1 条この条例は県の機関に係る申請届出その他の手続等に関し電子情報処理組織を使用する方法その他の情報通信の技術を利用する方法により行うことができるようにするための共通する事項を定めることにより県民の利便性の向上を図るとともに行政運営の簡素化及び効率化に資することを目的とする

この改正により影響を受ける条例は次の三種類に分けられる．

- これまでの申請書類がそのまま電子的手段で代替できるようになるもの
- これまでの条例の記述を変更しない限り，この改正が適用できないもの
- この改正が適用できないもの

これは改正範囲が条例全体に及ぶ大規模知識ベース改編の例であり，

1. 影響の及ぶ範囲を同定し，
2. 影響の及んだ範囲内で無矛盾であることを検証する

ことは容易ならざる作業となる．実際富山県は手作業によってこの膨大な処理を完了したが，われわれは計算機上のシミュレーションによってこの問題の解析を試みた．

法令文の論理化

本文 (富山県条例 54 条 3 項 1 号)

> 本文 1
> 県の機関は申請等のうち当該申請等に関する他の条例等の規定により書面等により行うこととしているものについては当該条例等の規定にかかわらず規則で定めるところにより電子情報処理組織 (県の機関の使用に係る電子計算機 (入出力装置を含む。以下同じ。) と申請等をする者の使用に係る電子計算機とを電気通信回線で接続した電子情報処理組織をいう。) を使用して行わせることができる。

本文 1 の括弧内の注意書きを抜き出して，本文 2 とする．

> 本文 2
> 電子情報処理組織⇒県の機関の使用に係る電子計算機と申請等を受ける者の使用に係る電子計算機とを電気通信回線で接続した電子情報処理組織をいう。

本文 1 を条例ロジックに分解し，本文より単語をあてはめ，述語論理の変数を用いると以下の条例ロジックが得られる．

> 本文 1 のロジック
> 電子申請行為 (x, y, v, c) ←申請行為 (x, y, v, a) ∧申請者 (x) ∧県の機関 (y) ∧申請等 (v) ∧書面等 (a) ∧申請手段 (c) ∧規定 (z, v) ∧関連条例等 (z) ∧電子情報処理組織 (c) 付帯状況：かかわらず (v,z)

同様に本文 2 の条文ロジックは以下のようになる．

> 本文 2 のロジック
> 電子情報処理組織 (o) ←接続 (a, b, c) ∧ (電子計算機 (a) ∨入出力装置 (a)) ∧申請者 (s) ∧使用者 (s, a) ∧電子計算機 (b)

XML 化

このように論理化された条文は XML 形式によって保存される．以下はその最初のほんの一部分である．

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.webgen.co.jp/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns="http://www.webgen.co.jp/">
<xsd:element type="OrdinanceType" name="ordinance"/>
<xsd:complexType name="OrdinanceType">
<xsd:sequence>
<xsd:element ref="section" minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element type="SectionType" name="section"/>
<xsd:complexType name="SectionType">
<xsd:sequence>
<xsd:element ref="subject"/>
<xsd:element ref="paragraph" minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element type="xsd:string" name="subject"/>
<xsd:element type="ParagraphType" name="paragraph"/>
<xsd:complexType name="ParagraphType">
<xsd:sequence>
<xsd:element ref="logic"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element type="LogicType" name="logic"/>
<xsd:complexType name="LogicType">
<xsd:choice minOccurs="0" maxOccurs="unbounded">
<xsd:element ref="implies"/>
<xsd:element ref="clause"/>
<xsd:element ref="and"/>
<xsd:element ref="or"/>
<xsd:element ref="not"/>
<xsd:element ref="unp"/>
</xsd:choice>
<xsd:attribute name="lang" type="xsd:string" use="required"/>
</xsd:complexType>
```

⋮

無矛盾性検出

われわれは以上のデータに対して，以下のような方法を試みた．

1. XML データから一階述語論理式を復元し，引数の数を合わせるように Prolog

のリテラルに変換し，Prologプログラムとする．
2. 法令文中のリテラルを洗い出し，assumptive factを生成する．
3. オントロジーから概念的に共存不可能なペアを洗い出す．
4. 以下のような不具合の検出を行う．
(a) 論理的な矛盾
(b) オントロジー上共存不可能な概念による矛盾
(c) 定義の循環

われわれが試みたのは第2,4,8,7,33,187条に限る．この範囲では修正後の条例集には上記のような不具合は検出されなかった．しかしながらこの手法が実際に有効であることをテストする目的で，われわれはXMLデータの一部を作為的に改編し，不具合の検出を試みた．以下は旅行命令権者の定義に関わる部分である．

> 第4条 次の各号に掲げる旅行は、当該各号に掲げる区分により任命権者(県費負担教職員にあつては、市町村教育委員会)又はその委任を受けた者(以下「旅行命令権者」という。)の発する旅行命令又は旅行依頼(以下「旅行命令等」という。)によつて行わなければならない。
> (1) 前条第1項の規定に該当する旅行 旅行命令
> (2) 前条第4項の規定に該当する旅行 旅行依頼
> 2 旅行命令権者は、電信、電話、郵便等の通信による連絡手段によつては公務の円滑な遂行を図ることができない場合で、かつ、予算上旅費の支出が可能である場合に限り、旅行命令等を発することができる。
> 3 旅行命令権者は、既に発した旅行命令等を変更する必要があると認める場合で、前項の規定に該当する場合には、自ら又は第5条第1項若しくは第2項の規定による旅行者の申請に基き、これを変更することができる。
> 4 旅行命令権者は、旅行命令等を発し、又はこれを変更するには、旅行命令簿又は旅行依頼簿(以下「旅行命令簿等」という。)に当該旅行に関し必要な事項の記載又は記録をし、これを当該旅行者に提示して行わなければならない。ただし、これを提示するいとまがない場合には、この限りでない。この場合において、旅行命令権者は、できるだけ速やかに旅行命令簿等に当該旅行に関し必要な事項の記載又は記録をし、これを当該旅行者に提示しなければならない。
> 5 前項の旅行命令簿等の提示については、富山県行政手続等における情報通信の技術の利用に関する条例(平成15年富山県条例第54号)第4条の規定は、適用しない。
> 6 旅行命令簿等の記載事項又は記録事項、様式その他の必要な事項は、人事委員会規則で定める。

(平 15 条例 55・一部改正)

この部分に関し，追加条項

> ＜原文＞任命権者 (県費負担教職員にあつては、市町村教育委員会) 又はその委任を受けた 者 (以下「旅行命令権者」という。) の発する旅行命令又は旅行依頼 (以下「旅行命令等」という。) によつて行わなければならない。（富山県条例第 54 号 4 乗第 1 項）

を組み合わせると定義のループが起こる [26]．まず，

```
pv_sub(Root, 任命権者 (x)):-
        pv(Root, 市町村教育委員会 (x)),
        pv(Root, 県費負担職員 (y)),
        pv(Root, 所属 (y,x)),
        pv(Root,acceptable(富山県職員等の旅費に関する条例:第 4 条:
第 1 項)).
```

の中に現れる市町村教育委員会は次のように定義される．

```
pv_sub(Root, 市町村教育委員会 (Var_0)):-
        pv(Root, 旅行命令権者 (Var_0)).
```

この中の旅行命令権者は次の規則で定義されている．

```
pv_sub(Root, 旅行命令権者 (z)):-
        pv(Root, 任命権者 (z)),
        pv(Root,acceptable(富山県職員等の旅費に関する条例:第 4 条:
第 1 項)).
```

ところがこの中に再び任命権者が現れ,「任命権者」→「市町村教育委員会」→「旅行命令権者」→「任命権者」の間で循環が起きている．

第4章　法令対象ドメインの形式記述と検証

法令の系統的な作成・解析・検証・保守のためには、法令文に記述された情報のみでなく，その法令が運用され意味を持つ領域（ドメイン）にたいする情報が必要である．ソフトウェアシステムにおいても，そのシステムが運用され意味のある機能を果たすべきドメインを特徴付けるドメイン記述の重要性が認識されている．

ドメイン記述の重要性は、法令など、そのドメインである意図を持って定められた規則や方針（ポリシー）、の整合性や正当性などの解析・検証においてより顕著となる．したがって，法令対象領域に対するドメイン記述は客観的で厳密な解析や推論が可能である形式記述 (formal description) であることが望まれる．

以下では北陸先端科学技術大学院大学 (JAIST) におけるドメインの形式記述と検証の研究を紹介する．

4.1　ドメインの形式記述法

ドメインの形式記述法としては様々なものが考えられるが，本研究では Dines Bjorner のドメインの記述法 [32] や CafeOBJ 言語による振舞仕様の形式記述法 [36, 33, 30] で採用されている以下の立場に基づくものを採用する．すなわちドメインの記述は「ドメインでの観測可能な現象の記述」であり,「観測可能な現象」とは，実体 (entity)，関数 (function)，事象 (event)，そして振舞 (behavior) のことであるとする。ここで，実体，関数，事象，振舞は以下のように定義される．

実体： 実体とは対象とするドメインを構成する「もの」である．実体は型 (type または sort) により分類される．

関数： 関数はドメインに存在する種々の機能 (function) を抽象化したものであり，実体を入力すると実体を出力する．関数を働かせて入力から出力を得ることを関数適用という．

事象： 事象はドメインに変化を引き起こす (つまりある時点の状態を入力として

次の状態を出力とする) 関数適用である.

振舞： 振舞は事象の系列である.

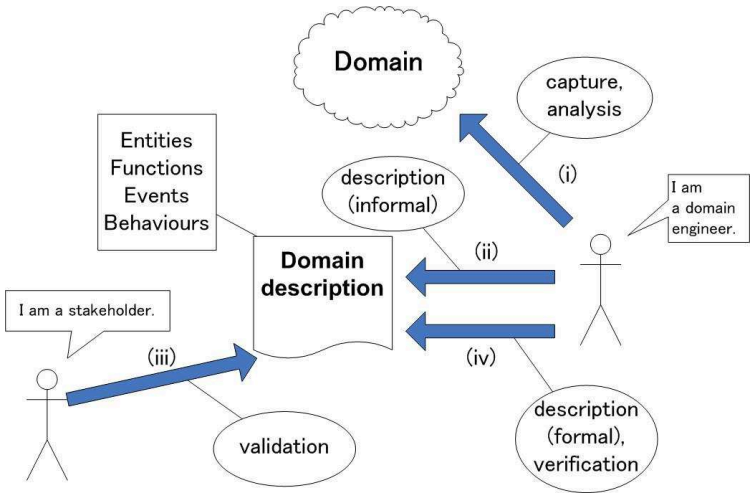

図 4.1: ドメインの形式記述の流れ

ドメインの形式記述はおおよそ次の手順で行われる図 4.1.

1. ドメイン情報の収集 (capture)，解析 (analyze) を行う (図 4.1 の (i) の部分).
2. 自然言語，図などを使ってドメインのモデル図を記述する (図 4.1 の (ii) の部分).
3. 利害関係者 (stakeholder) による確認 (validation) が行われる (図 4.1 の (iii) の部分).
4. (ii) と (iii) の繰り返し.
5. ドメイン記述の形式化，検証 (verification) を行う (図 4.1 の (iv) の部分).

確認とは，ドメイン記述が利害関係者 (stake holder) の意図と一致しているか確かめる作業のことであり，検証とは，ドメイン記述が記述者の意図するものと一致していることを証明することである．確認により「正しいドメイン記述」(the right domain description) を得ることができ，検証により「正しくドメイン記述」(the domain description right) を行うことができる．確認は図や自然言語などで記述された非形式ドメイン記述を利害関係者に提示することによって行われ，検証は形式記述の表しているモデルがそのドメインが持つべき性質を満たしていることを形式的に証明することによって行われる．確認と検証は補完的なものであり，

確認により形式記述が改良され検証が促進され，検証により非形式記述が改良され確認が促進される．

4.2 ドメインの形式記述と検証の事例

ドメインの形式記述と検証の重要性は認識されつつあるが，その事例は少ない．ここでは，JAIST における研究事例を紹介する．

4.2.1 安全プロトコルの形式記述と検証 [37, 30]

通信プロトコル (communication protocol) や安全プロトコル (secure protocol) のモデル化と解析・検証の研究は，情報通信システムや電子商取引システムの発展に伴い進展してきた．各種プロトコルは情報ネッタワークの基盤をなす規約であり，人間社会における法令に対応する．

安全プロトコルのモデル化と解析・検証においては,「安全性」をどのように定義するか，すなわちドメインをどのようにモデル化するか，がモデル化の最大のポイントになる．最近では，確率などを導入した精密なモデル化も試みられているが，解析・検証が複雑になる傾向があり，解析・検証がより容易な古典的なモデル化で十分であるとする立場もある．現状は,「どのモデル化が妥当であるかは状況に応じて判断する」である，といえる．

我々は，Dorev-Yao の古典的な安全性のモデル化に基づき，2 で示したドメインの記述法を採用し，安全プロトコルの安全性の検証の事例をいくつか開発してきた．これらの事例を開発した時点ではドメインの形式記述という意識は強くは無かったが，結果的に興味深いドメインの形式記述と検証の事例となっている．

モデル化

まず，2 種類の主体の存在を仮定する．1 つはプロトコルに則った動作のみを行い，もう 1 つはプロトコルに則った動作に加え規定外のことも行う．規定外の動作とは，ネットワークを流れるメッセージを盗聴し，盗聴した情報を蓄え，蓄えた情報を基にメッセージを偽造することである．後者の協調した動作を (汎用の) 侵入者としてモデル化する．ただし，プロトコルで使われる暗号システムは充分に頑強であると仮定する．つまり，侵入者は Dolev-Yao の汎用の侵入者モデルに則っているとする．

つづいて，暗号文やメッセージ等のプロトコルで用いられているデータ型を定義する．プロトコルでやりとりする各メッセージのデータ構成子は，4 個以上の引数

を取る．1番目の引数は，対応するメッセージを実際に送信した (あるいは生成した) 主体を表し，2番目の引数は，対応するメッセージの見せかけの送信者を表し，3番目の引数は，対応するメッセージの受信者を表す．4番目以降の引数は，対応するメッセージの本体を表す．1番目の引数と2番目の引数が異なっていれば，そのメッセージは1番目の引数で表される主体により偽造されたことを意味しておる．この場合，1番目の主体は侵入者である．メッセージを表すデータ構成子の1番目の引数は，メタ情報で，プロトコル外部の観察者 (検証者) によってのみ利用することができ，侵入者は1番目の引数を偽造することができない．一方，2番目以降の引数は，侵入者により偽造することができる．

ネットワークは，プロトコルでやりとりするメッセージの多重集合として表現する．ネットワークは，侵入者の一部，あるいは侵入者が自由に使用できる記憶装置と見做すことができるので，メッセージが一旦送信されたら，そのメッセージは (送信者により何度も再送されることができるので) ネットワークに留まることができる．このため，空のネットワークは，メッセージが1個も送信されていないことを意味する．また，メッセージがネットワークに存在することは，その1番目の引数で表される主体が，そのメッセージを送信したことを意味し，主体がメッセージを受信することは，そのメッセージがネットワークに存在していることを意味する．このような事実を用いて，検証したい性質を定式化する．

ネットワークをメッセージの多重集合として定式化すると，ネットワーク (そこに含まれるメッセージ) から侵入者が利用できるデータを定義できる．

次に，観察可能なプロトコル内部の値 (たとえば，ネットワーク) を決め，プロトコルの振舞 (プロトコルに則ったメッセージの送信および侵入者によるメッセージの偽造) を遷移規則としてモデル化し，観察可能な値がどのように変化するかで定義する．プロトコルのモデル (観測遷移システム:Observational Transition System) を CafeOBJ で記述する．

検証

検証したい性質を CafeOBJ の項として記述し，CafeOBJ で証明譜を記述し，CafeOBJ 処理系で証明譜を実行させることで検証を行う．

証明譜の記述方法の概略を示す．帰納法に基づき検証を行う場合に記述する証明譜について概説する．

プロトコルが $P_1(W, X_1) \wedge \ldots \wedge P_n(W, X_n)$ で表される性質を有することを検証すると仮定する．ただし，W はプロトコルの状態を表し，$X_i\,(i = 1, \ldots, n)$ はそれ以外のパラメタを表している．まず，その性質を記述するモジュール (たとえば `PRED`) を宣言する．そのモジュールで，$X_i\,(i = 1, \ldots, n)$ に対応する任意の値を表

す定数 $\mathtt{x}_i$ を宣言する．また，議論中の性質を表す演算子 (たとえば $\mathtt{p}_1, \ldots, \mathtt{p}_n$) およびそれらを定義する等式を次のように宣言する：

```
op p1 : Sys SortX1 -> Bool
...
op pn : Sys SortXn -> Bool
eq p1(W, X1) = P1(W, X1) .
...
eq pn(W, Xn) = Pn(W, Xn) .
```

ただし，$\mathtt{SortX}_i\,(i = 1, \ldots, n)$ は X_i に対応する値のための可視ソートで，W と $\mathtt{X}_i$ はソート Sys および $\mathtt{SortX}_i$ の CafeOBJ 変数である．

任意の初期状態で，議論中の性質が成り立つことを示すには，各 $i \in \{1, \ldots, n\}$ に対し，以下の証明譜を記述すればよい：

```
open PRED
  red pi(init, xi) .
close
```

CafeOBJ コマンド open で指定されたモジュール (およびそこで輸入されているモジュール) に宣言されてるソート，演算子および等式を利用することを宣言し，CafeOBJ コマンド close でその利用を終了することを宣言する．CafeOBJ コマンド red は，等式を左から右への書き換え規則と見做し，与えられた項を簡約する (書き換える)．

次に，各帰納段階で証明すべき述語を記述するモジュール (たとえば ISTEP) を宣言する．そのモジュールで，2 つの定数 s と s' を宣言する．s は検証中の性質が成り立つ任意の (到達可能) 状態を，s' は状態 s で遷移規則が適用された事後状態を表す．各帰納段階で証明すべき述語およびそれを定義する等式を以下のように宣言する：

```
op istep1 : SortX1 -> Bool
...
op istepn : SortXn -> Bool
eq istep1(X1) = p1(s, X1) implies p1(s', X1) .
...
eq istepn(Xn) = pn(s, Xn) implies pn(s', Xn) .
```

各帰納段階において (CafeOBJ の作用演算 a で表される遷移規則が議論中の性質を保存することを示すとき)，通常，状態空間を複数に分割する (場合分けを行う)．分割した各空間 (各場合) ごとに以下のような証明譜を記述する：

```
open ISTEP
  任意の値を表す定数の宣言.
  場合を表す等式の宣言.
  eq s' = a(s, ...) .
  red istepi(xi) .
close
```

まず，作用演算 a で使われる任意の値を表す定数ならびに議論中の場合を表す等式を宣言する．また，必要であれば，補題等からの事実を等式として宣言する．続いて宣言する等式は，定数 s' が定数 s で表される状態で作用演算 a に対応する遷移規則が適用された事後状態を表すことを意味する．最後に，項 $\mathtt{istep}_i(\mathtt{x_i})$ を簡約する．簡約の結果が期待どおりであれば (この場合は true)，この場合の証明が成功したことを意味する．そうでなければ，この場合に対応する空間をさらに分割しなければならないか，この場合の証明にはさらなる補題を必要とするか，を判断しなければならない．あるいは，議論中のプロトコルは議論中の性質を有していないことを示唆しているのかもしれない．

帰納段階において，帰納法の仮定 (ここでは $\mathtt{p}_i(\mathtt{s}, \mathtt{x}_i)\,(i = 1, \ldots, n)$ で表されている) が弱いため証明が成功しないこともあり得る．この場合，何らかの方法で帰納法の仮定を強化する必要がある．

NSLPK 認証プロトコル

1978 年に Needham と Schroeder により公開鍵暗号方式を用いた認証プロトコルが提案された．提案の目的は，大規模ネットワークにおける認証の問題点を明らかにすることであった．このプロトコルを NSPK 認証プロトコルと呼ぶ．NSPK 認証プロトコルが提案された 17 年後の 1995 年に Lowe により，侵入者 (intruder) が他の主体 (principal) になり済まし別の主体との認証を確立できる，という欠陥が発見された．この欠陥の発見と同時に，Lowe は修正案を提案している．この修正案を，本稿では，NSLPK 認証プロトコルと呼ぶ．

NSLPK 認証プロトコルは，NSPK 認証プロトコルと同様，公開鍵暗号方式を用いる．各主体に対し公開鍵対 (公開鍵；public key と私用鍵；private key) が与えられる．公開鍵は鍵サーバ等を用いて広く世間に公開され，他の主体はこの公開鍵を自由に取得することができる．一方，私用鍵は他の主体に知られないように安全に保管される．通常，公開鍵は (セッション鍵等の長くない) メッセージの暗号化ならびに電子署名の確認に用いられ，私用鍵は暗号文の復号化ならびに電子署名の作成に用いられる．NSLPK 認証プロトコルでは，公開鍵対をメッセージの暗号化と復号化にのみ用いる．主体 p の公開鍵を $k(p)$ で表し，対応する私用鍵を $k^{-1}(p)$ で表す．また，公開鍵 k で暗号化したメッセージ m を $\{m\}_k$ で表す．どの主体も主体 p の公開鍵 $k(p)$ を用いてメッセージ m を暗号化し，暗号文 $\{m\}_{k(p)}$ を生成できるが，暗号文 $\{m\}_{k(p)}$ を復号できるのは私用鍵 $k^{-1}(p)$ を保持している主体 p だけである．

各主体が他の主体の公開鍵を取得していると仮定すると，NSLPK 認証プロトコルは以下のように記述できる：

$$\begin{array}{lllll} \text{Message}\,1 & p \to q & : & p.q.\{n_p.p\}_{k(q)} \\ \text{Message}\,2 & q \to p & : & q.p.\{n_p.n_q.q\}_{k(p)} \\ \text{Message}\,3 & p \to q & : & p.q.\{n_q\}_{k(q)} \end{array}$$

小点 "." は組 (tuples) のデータ構成子である．

主体 p が主体 q との間で相互に認証したい場合，主体 p はノンス n_p を生成し，識別子 p と共に主体 q の公開鍵 $k(q)$ で暗号化した暗号文 $\{n_p.p\}_{k(q)}$ を主体 q に送る．このメッセージを Message 1 と呼ぶ．上記のプロトコルの記述に現れる各メッセージの最初の 2 つの値は，それぞれメッセージの送信者と受信者の識別子である．

Message 1 を受理した主体 q は，暗号文を自身の私用鍵 $k^{-1}(q)$ で復号し，ノンス n_p と識別子 p を取得する．まず，得られた識別子が送信者の識別子と同じであることを確認する．つづいて，ノンス n_q を生成し，受け取ったノンス n_p と自身の識別子 q と共に主体 p の公開鍵 $k(p)$ で暗号化した暗号文 $\{n_p.n_q.q\}_{k(p)}$ を主体 p に送る．このメッセージを Message 2 と呼ぶ．NSPK 認証プロトコルでは，Message 2 の暗号文中に送信者の識別子を含めておらず，これを含めることが Lowe により提案された修正案である．

Message 2 を受理した主体 p は，暗号文を自身の私用鍵 $k^{-1}(p)$ で復号し，2 つのノンス n_p, n_q と識別子 q を得る．まず，得られた識別子が送信者の識別子と同じであることを確認する．つづいて，ノンス n_p が主体 q との認証のために生成したものであることを確認する．このことにより，通信相手が確かに主体 q であることを確証する．というのは，ノンス n_p を取得できるのは，私用鍵 $k^{-1}(q)$ を保持している主体 q だけだからである．これらの確認の後，もう一方のノンス n_q を主体 q の公開鍵 $k(q)$ で暗号化した暗号文 $\{n_q\}_{k(q)}$ を主体 q に送る．

Message 3 を受理した主体 q は，暗号文を自身の私用鍵 $k^{-1}(q)$ で復号し，ノンス n_q を取得し，それが主体 p との認証のために生成したノンスと一致することを確認する．このことにより，通信相手が確かに主体 p であることを確証する．というのは，ノンス n_q を取得できるのは，私用鍵 $k^{-1}(p)$ を保持している主体 p だけだからである．

以上により主体 p と q の間で相互に認証できたことになる．

認証プロトコルが満たすべき重要な性質の一つにノンス秘匿性 (Nonce Secrecy Property) がある．ノンス秘匿性は以下のように記述できる．

> 侵入者は，侵入者とは異なる主体が侵入者とは異なる別の主体との認証のために生成したノンスを，取得することはない．

プロトコルをモデル化し，CafeOBJ で記述し，プロトコルがノンス秘匿性を満たすことを示す証明譜を記述し，CafeOBJ システムで証明譜を実行することで証明の正しさを示すことが出来る．

NSLPK の CafeOBJ 仕様

以下にプロトコルのモデルである観測遷移機械の CafeOBJ 仕様を示す．この仕様に対する証明譜 (proof score) による検証法の詳細については [30] を参照されたい．

```
-- 主体 (principle)
mod* PRIN {
  pr(EQL)
  [Prin]
  op intr : -> Prin
}

-- 乱数 (random number)
mod! RAND {
  pr(EQL)
  [Rand]
  op seed : -> Rand
  op next : Rand -> Rand
  vars R1 R2 : Rand
  eq (seed = next(R1)) = false .
  eq (R1 = next(R1)) = false .
  eq (next(R1) = next(R2)) = (R1 = R2) .
}

-- ノンス (nonce)：あるセッションをユニークに特定するもの
mod! NONCE {
  pr(PRIN + RAND)
  [Nonce]
  op n : Prin Prin Rand -> Nonce
  op p1 : Nonce -> Prin
  op p2 : Nonce -> Prin
  op r  : Nonce -> Rand
  vars P1 P2 P12 P22 : Prin
  vars R1 R2 : Rand
  eq p1(n(P1,P2,R1)) = P1 .
  eq p2(n(P1,P2,R1)) = P2 .
  eq r(n(P1,P2,R1))  = R1 .
  eq (n(P1,P2,R1) = n(P12,P22,R2))
     = (P1 = P12 and P2 = P22 and R1 = R2) .
}

-- ノンスの多重集合
```

```
mod! NONCE-SET {
  pr(NONCE)
  [Nonce < NonceSet]
  op empty : -> NonceSet
  op _,_ : NonceSet NonceSet -> NonceSet {assoc comm}
  op _\in_ : Nonce NonceSet -> Bool
  vars N1 N2 : Nonce
  vars S1 S2 : NonceSet
  eq (empty , S1) = S1 .
  eq (N1 , N1 , S1) = (N1 , S1) .
  eq N1 \in empty = false .
  eq N1 \in N2 = (N1 = N2) .
  eq N1 \in (N2 , S2) = (N1 = N2) or (N1 \in S2) .
  eq (empty = N2) = false .
  eq (empty = N2 , S2) = false .
  eq (N1 , S1 = N2) = (N1 = N2) and (S1 = empty) .
  ceq (N1 , S1 = N2 , S2) = (S1 = S2) if N1 = N2 .
  ceq (N1 , S1 = S2) = false if not(N1 \in S2) .
}

-- 暗号文(cipher)
mod! CIPHER {
  pr(NONCE)
  [Cipher]
  op enc1 : Prin Nonce Prin -> Cipher
  op enc2 : Prin Nonce Nonce Prin -> Cipher
  op enc3 : Prin Nonce -> Cipher
  op k    : Cipher -> Prin
  op n1   : Cipher -> Nonce
  op n2   : Cipher -> Nonce
  op p    : Cipher -> Prin
  vars K1 K2 : Prin
  vars N1 N2 N11 N21 N12 N22 : Nonce
  vars P1 P2 : Prin
  var C : Cipher
  eq k(enc1(K1,N1,P1)) = K1 .
  eq k(enc2(K1,N1,N2,P1)) = K1 .
  eq k(enc3(K1,N1)) = K1 .
  eq n1(enc1(K1,N1,P1)) = N1 .
  eq n1(enc2(K1,N1,N2,P1)) = N1 .
  eq n1(enc3(K1,N1)) = N1 .
  eq n2(enc2(K1,N1,N2,P1)) = N2 .
  eq p(enc1(K1,N1,P1)) = P1 .
  eq p(enc2(K1,N1,N2,P1)) = P1 .
```

```
  eq (enc1(K1,N11,P1) = enc1(K2,N12,P2))
      = (K1 = K2 and N11 = N12 and P1 = P2) .
  eq (enc1(K1,N11,P1) = enc2(K2,N12,N22,P2)) = false .
  eq (enc1(K1,N11,P1) = enc3(K2,N12)) = false .
  eq (enc2(K1,N11,N21,P1) = enc2(K2,N12,N22,P2))
     = (K1 = K2 and N11 = N12 and N21 = N22 and P1 = P2) .
  eq (enc2(K1,N11,N21,P1) = enc3(K2,N12)) = false .
  eq (enc3(K1,N11) = enc3(K2,N12)) = (K1 = K2 and N11 = N12) .
}

-- 暗号文の多重集合としてのネットワーク
mod! NETWORK {
  pr(CIPHER)
  [Cipher < Network]
  op empty : -> Network
  op _,_ : Network Network -> Network {assoc comm}
  op _\in_ : Cipher Network -> Bool
  vars NW1 NW2 : Network
  vars C1 C2 : Cipher
  eq (empty , NW1) = NW1 .
  eq (C1 , C1 , NW1) = NW1 .
  eq C1 \in empty = false .
  eq C1 \in C2 = (C1 = C2) .
  eq C1 \in (C2 , NW2) = (C1 = C2) or C1 \in NW2 .
  eq (empty = C2) = false .
  eq (empty = C2 , NW2) = false .
  eq (C1 , NW1 = C2) = (C1 = C2) and (NW1 = empty) .
  ceq (C1 , NW1 = C2 , NW2) = (NW1 = NW2) if C1 = C2 .
  ceq (C1 , NW1 =  NW2) = false if not(C1 \in NW2) .
}

-- NSLPKプロトコル
mod* NSLPK {
  pr(NONCE-SET + NETWORK)
  *[Sys]*
  -- an arbitrary initial state
  op init : -> Sys
  -- observers
  bop rand : Sys -> Rand  bop nw : Sys -> Network
  bop nonces : Sys -> NonceSet
  -- actions
  bop send1 : Sys Prin Prin -> Sys
  bop send2 : Sys Prin Prin Nonce -> Sys
  bop send3 : Sys Prin Prin Nonce Nonce -> Sys
```

```
bop fake1 : Sys Prin Prin Nonce -> Sys
bop fake2 : Sys Prin Prin Nonce Nonce -> Sys
bop fake3 : Sys Prin Nonce -> Sys
-- CafeOBJ variables
var S : Sys  vars P1 P2 P3 : Prin  vars N1 N2 : Nonce
-- init
eq rand(init) = seed .
eq nw(init) = empty .
eq nonces(init) = empty .
-- send1
eq rand(send1(S,P1,P2)) = next(rand(S)) .
eq nw(send1(S,P1,P2))   = (enc1(P2,n(P1,P2,rand(S)),P1) , nw(S)) .
eq nonces(send1(S,P1,P2))
   = (if P2 = intr then (n(P1,P2,rand(S)) , nonces(S))
                   else nonces(S) fi) .
-- send2
op c-send2 : Sys Prin Prin Nonce -> Bool
eq c-send2(S,P1,P2,N1) = enc1(P1,N1,P2) \in nw(S) .
--
ceq rand(send2(S,P1,P2,N1)) = next(rand(S)) if c-send2(S,P1,P2,N1) .
ceq nw(send2(S,P1,P2,N1))
    = (enc2(P2,N1,n(P1,P2,rand(S)),P1) , nw(S))
      if c-send2(S,P1,P2,N1) .
ceq nonces(send2(S,P1,P2,N1))
    = (if P2 = intr then (N1 , n(P1,P2,rand(S)) , nonces(S))
                    else nonces(S) fi)
       if c-send2(S,P1,P2,N1) .
ceq send2(S,P1,P2,N1)         = S if not c-send2(S,P1,P2,N1) .
-- send3
op c-send3 : Sys Prin Prin Nonce Nonce -> Bool
eq c-send3(S,P1,P2,N1,N2) =
          enc2(P1,N1,N2,P2) \in nw(S) and enc1(P2,N1,P1) \in nw(S) .
--
eq  rand(send3(S,P1,P2,N1,N2))   = rand(S) .
ceq nw(send3(S,P1,P2,N1,N2))     =
                   (enc3(P2,N2) , nw(S)) if c-send3(S,P1,P2,N1,N2) .
ceq nonces(send3(S,P1,P2,N1,N2))
    = (if P2 = intr then (N2 , nonces(S)) else nonces(S) fi)
                                          if c-send3(S,P1,P2,N1,N2) .
ceq send3(S,P1,P2,N1,N2)         = S if not c-send3(S,P1,P2,N1,N2) .
-- fake1
op c-fake1 : Sys Prin Prin Nonce -> Bool
eq c-fake1(S,P1,P2,N1) = N1 \in nonces(S) .
--
```

```
  eq  rand(fake1(S,P1,P2,N1))   = rand(S) .
  ceq nw(fake1(S,P1,P2,N1))     = (enc1(P2,N1,P1) , nw(S))
                                  if c-fake1(S,P1,P2,N1) .
  eq  nonces(fake1(S,P1,P2,N1)) = nonces(S) .
  ceq fake1(S,P1,P2,N1)         = S if not c-fake1(S,P1,P2,N1) .
  -- fake2
  op c-fake2 : Sys Prin Prin Nonce Nonce -> Bool
  eq c-fake2(S,P1,P2,N1,N2) = N1 \in nonces(S) and N2 \in nonces(S) .
  --
  eq  rand(fake2(S,P1,P2,N1,N2))   = rand(S) .
  ceq nw(fake2(S,P1,P2,N1,N2))
      = (enc2(P1,N1,N2,P2) , nw(S)) if c-fake2(S,P1,P2,N1,N2) .
  eq  nonces(fake2(S,P1,P2,N1,N2)) = nonces(S) .
  ceq fake2(S,P1,P2,N1,N2)         = S if not c-fake2(S,P1,P2,N1,N2) .
  -- fake3
  op c-fake3 : Sys Prin Nonce -> Bool
  eq c-fake3(S,P1,N1) = N1 \in nonces(S) .
  --
  eq  rand(fake3(S,P1,N1))   = rand(S) .
  ceq nw(fake3(S,P1,N1))
      = (enc3(P1,N1) , nw(S)) if c-fake3(S,P1,N1) .
  eq  nonces(fake3(S,P1,N1)) = nonces(S) .
  ceq fake3(S,P1,N1)         = S if not c-fake3(S,P1,N1) .
}
```

4.2.2　病院ドメインの記述 [31]

ドメインの形式記述法を具体的に研究開発することを目的として，病院ドメインを取り上げ，形式記述と検証の事例研究を行っている．医療事故は社会的に大きな問題であり，その客観的な解析・検証には病院ドメインの体系的な形式記述が有効であると期待される．また，病院ドメインは比較的多くの人が共通の理解をもっている対象であり，形式記述の例として適当である．

病院ドメインの実体

病院ドメインにおいて，実体は病院 (hospital)，手術室 (operating room)，患者 (patient)，医療スタッフ (medical staff)，病棟 (ward)，経営スタッフ，薬局 (pharmacy) などがある．図 4.2 はこれらの実体がそれぞれどのように構成されているのかを示したものである．

太い線のボックスは合成実体を表している．[32] で示されている非形式ドメイン

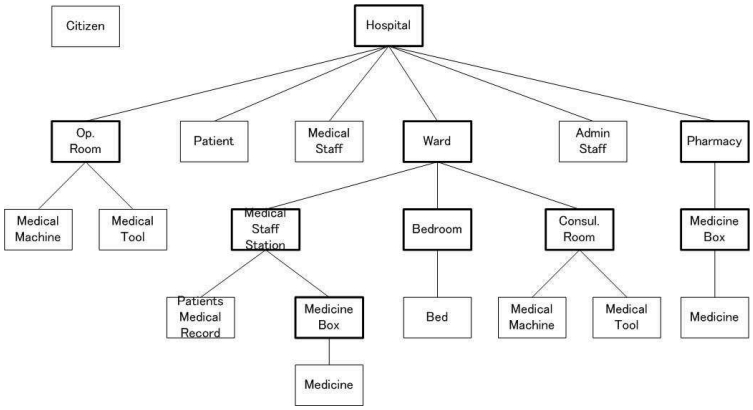

図 4.2: 病院ドメインの実体

記述のスタイルでは各実体の次の情報を記述していく.

(a) その実体がアトミックな実体か，合成実体かの情報.

(b) その実体の属性.

(c) その実体が合成実体なら，下位実体の情報 (どの実体が下位実体であるか).

(d) その実体が合成実体なら，その実体のメレオロジー (構成).

例えば，病院は (a) 合成実体で，(b) 属性は病院の名前と住所，(c) 下位実体は手術室，患者，医療スタッフ，病棟，経営スタッフ，薬局で，(d) メレオロジーは”病院は 1 棟以上の病棟，1 室以上の手術室，1 局以上の薬局，1 人以上の医療スタッフ，0 人以上の患者，1 人以上の経営スタッフから構成されている”という実体の構成を記述していく.

病院ドメインの関数

病院ドメインにおいて，市民を患者として受けいれる (admit), 問診 (interview), 検査計画を立てる (plan analysis), 検査をする (analyse)，診断をする (diagnose), 治療計画を立てる (plan treatment)，治療を行う (treat)，患者を他の病棟に移す (transfer)，患者を退院させる (release) などの行為が関数としてみなされる．例えば，関数 admit のシグネチャは次の通りである.

admit : Citizen × Ward ID × Hospital → Hospital

Citizen は市民，Ward ID は病棟の ID，Hospital は病院を表す型である．admit が適用されると Citizen C は病院に患者 P として受け入れられる．つまり，病院 H に患者 P が入り，P のためのカルテが生成される．

4.2.3　病院ドメインの事象

病院ドメイン上の事象としては”市民が病院にやって来る”，”患者の容態が回復する”，”患者の容態が悪くなる”，”患者が死ぬ”，”患者に新しい症状が現れる”などが考えられる．例えば，”市民が病院にやってくる”という事象は関数 admit の引き金となる．また，事象”患者の容態が回復する”は関数 treat が適用された後に起こりうる．そしてこの事象によって関数 release の引き金となりうる．

4.2.4　病院ドメインの振舞い

先に述べたように振舞いとは事象の系列である．例えば，

- 市民が病院にやって来る，
- 患者を登録する
- 患者を診察する
- 患者の治療プランを作る
- 患者を治療する

といった事象の系列は 1 つの（意味のある）振舞いとしてみなすことができる．このような振舞いは一般には無数にある．

OTS によるドメインのモデル化と形式記述

観測遷移システム (OTS:Observational Transition System) によるドメイン記述の形式化について述べる．OTS はシステムの中の観測可能な値の変化を規定することでシステムの振舞いをモデル化するための状態機械で，ここでは，病院ドメインの振舞いのモデルとなる OTS を代数仕様言語 CafeOBJ で記述する [30]．病院ドメインの状態空間を OTS でモデル化することによって検証可能な病院のドメインの振舞いの記述を行うことができる．

OTS はシステムの中の観測可能な値の変化を規定することでシステムの振舞いをモデル化するための状態遷移機械で $\langle \mathcal{O}, \mathcal{I}, \mathcal{T} \rangle$ の 3 組で表される．それぞれの意味は以下の通りである．

$\mathcal{O}$ ： 観測関数 (observations) の有限集合．

$\mathcal{I}$ ： 初期状態 (initial states) の集合．

$\mathcal{T}$ ： 遷移演算 (transitions) の集合．

OTS によるドメインのモデル化は以下のようにして行う．

- 実体は観測関数によって観測する．
- 関数と事象は遷移演算としてみなす．
- 振舞いは遷移演算の適用の繰り返しによって表される．

観測関数で観測するのは市民と病院の下位実体である病棟，手術室，薬局，医療スタッフ，患者，経営スタッフである．また，病院自身も実体であり，属性を持つので，病院の属性を観測するための観測関数も用意する．

ドメインの状態が変化するのは事象が起こったときである．状態変化を引き起こす関数は自然に遷移演算（つまり事象）として定義することができるが，一般の事象は非形式記述で述べたままの状態では定義できない．そこで事象にも引数と返り値を与えることで遷移演算として定義する．例えば，”市民が病院にやってくる”という事象のシグネチャは次のように記述することができる．

cch ： Citizen × Hospital → Hospital

`cch` は”a citizen comes to a hospital”の省略形である．

実体の CafeOBJ 仕様

データ型として実体を CafeOBJ で記述する方法を述べる．実体は属性の組と下位実体の集合の組の対で表すことができる．つまり，ある実体の型を E とし，E の属性の組の型を $EAttri$，下位実体の集合の組の型を $ESub$ とすると E は

- $E = EAttri \times ESub$
- $EAttri = a_0 \times a_1 \times ... \times a_n$ (n は自然数)
- $ESub = \mathrm{NoSub} \mid s_0\mathrm{set} \times s_1\mathrm{set} \times ... \times s_m\mathrm{set}$ (m は自然数)

と表すことができる．NoSub は下位実体を持たないことを表す定数である．

アトミックな実体

アトミックな実体の例として薬の CafeOBJ 仕様を示す.

```
mod! MEDICINE principal-sort Med {
  pr(MID + NOSUB + EQL)
  [ MAttri Med ]
  op med : MAttri NoSub -> Med
  op mAtt : Mid -> MAttri
  op mid : Med -> Mid
  var M : Mid| \verb|vars ME1 ME2 : Med
  eq (ME1 = ME2) = (mid(ME1) = mid(ME2)) .
  eq mid(med(mAtt(M), nosub)) = M . }
```

pr(MID + NOSUB + EQL)ではモジュール MID, NOSUB, EQL を輸入している．ソート Mid と NoSub はモジュール MID，NOSUB の中で定義されており，モジュール EQL の中では等価性を判定する述語_=_が定義されている．med は薬を表すソート Med の構成子で引数として MAttri と NoSub をとる．MAttri は薬の属性の組を表すソートで，NoSub は空の下位実体の集合の組を表すソートである．mAtt は MAttri の構成子である．ここでは薬の属性は薬の ID であるので，引数として Mid をとる．

合成実体

合成実体の仕様を記述する前に下位実体の集合を現すソートを定義する必要がある．上で定義した薬の集合を現すソートはパラメータ化モジュールを使い，以下のように記述することができる.

```
mod! MSET {
  pr (SET(MEDICINE)
     * {sort Set -> MSet,
        op nil -> nilM })}
```

このモジュールを使って合成実体である薬箱の記述は以下のように行える.

```
mod! MEDBOX principal-sort MeB{
  pr(MBID + MSET)
  [ MBAttri MBSub MeB ]
  op meb : MBAttri MBSub -> MeB
  op mbAtt : MBid -> MBAttri
  op mbid : MeB -> MBid
  op mbSub : MSet -> MBSub
```

```
op mset : MeB -> MSet
op mere : MeB -> Bool
vars MB : MeB  var MBA : MBAttri  var MBS : MBSub
var M : MBid  var Ms : MSet
eq (MB1 = MB2) = (mbid(MB1) = mbid(MB2)) .
eq mbid(meb(mbAtt(M), MBS)) = M .
eq mset(meb(MBA, mbSub(Ms))) = Ms .
eq mere(MB) = card(mset(MB)) > 0 . }
```

MeBは薬箱を表すソートである．mbSubは薬箱の下位実体の集合の組の構成子である．薬箱の下位実体は薬なので引数として薬の集合を現すソート MSet を取る．また，メレオロジーを満たしているかチェックするための述語 mere を定義した．この述語を適用して真を返す MeB の要素はメレオロジーを満たしている．

病院ドメインの CafeOBJ 仕様

病院ドメインの振舞いはモジュール HOSPITAL として記述する．シグネチャの部分は以下のように記述できる．

```
mod* Hospital{
  pr(WSET + ORSET + PHSET + ASSET + PASET + MSSET)
  *[ Hos ]*

  -- observations
  bop h-id : Hos -> HID
  bop h-loc : Hos -> Loc
  bop wset : Hos -> WSet
  bop orset : Hos -> ORSet
  bop phset : Hos -> PhSet
  bop asset : Hos -> ASSet
  bop paset : Hos -> PaSet
  bop msset : Hos -> MSSet
  bop cset : Hos -> CSet

  -- transitions
  bop admit : Cit Wid Hos -> Hos
  bop interview : PMRid Wid Hos -> Hos
  bop planAnal : PMRid Wid Hos -> Hos
  bop doAnal : PMRid Wid Hos -> Hos
  ...
  bop cch : Cit Hos -> Hos -- a citizen comes to a hospital.
  bop pgc : Pid Hos -> Hos -- a patient gets cured.
  bop pgw : Pid Hos -> Hos -- a patient gets worse.
```

```
bop pd : Pid Hos -> Hos -- a patient dies.
...

-- initial states
op init : -> Hos
...  }
```

病院の状態空間を隠蔽ソート (hidden sort) Hos として宣言する．隠蔽ソートの宣言は CafeOBJ では*[]*で囲って記述する．HID と Loc はそれぞれ病院の名前，住所を表すソートで，WSet，ORSet，PhSet，ASSet，PaSet，MSSet はそれぞれ病棟の集合，手術室の集合，薬局の集合，経営スタッフの集合，患者の集合，医療スタッフの集合を表すソートである．これらのソートは他のモジュールで定義し，pr(モジュール名) でそれらのモジュールを輸入することで使うことができる．

初期状態は，病院ドメインの場合，病院のメレオロジーが満たされている状態とする．これは CafeOBJ で以下のように記述できる．

```
op init : -> Hos
eq (card(wset(init)) > 0) = true .
eq (card(orset(init)) > 0) = true .
eq (card(phset(init)) = 1) = true .
eq (card(asset(init)) > 0) = true .
eq (card(paset(init)) >= 0) = true .
eq (card(msset(init)) > 0) = true .
```

init はソート Hos の要素である．病院のメレオロジーは”病院は 1 棟以上の病棟，病院は 1 室以上の手術室，1 局の薬局，1 人以上の経営スタッフ，0 人以上の患者，1 人以上の医療スタッフから構成されている”であるので上記のように等式で表すことができる．

遷移演算の定義はこのモジュールの中の公理として記述される．ここでは，コードの詳細には立ち入らず，疑似コードとして記述する．例として”市民を患者として受け入れる”ための遷移演算 admit は以下のように記述される．

```
op c-admit : Cit Wid Hos -> Bool
var C : Cit var W : W var H : Hos
eq c-admit(C, W, H) = 事前条件 C
eq h-id(admit(C, W, H)) = h-id(H) .
eq h-loc(admit(C, W, H)) = h-loc(H) .
ceq wset(admit(C, W, H)) = 後の病棟の集合
                           if c-admit(C, W, H) .
eq orset(admit(C, W, H)) = orset(H) .
eq phset(admit(C, W, H)) = phset(H) .
```

```
eq asset(admit(C, W, H)) = asset(H) .
ceq paset(admit(C, W, H)) = 後の患者の集合
                            if c-admit(C, W, H) .
ceq msset(admit(C, W, H)) = 後の医療スタッフの集合
                            if c-admit(C, W, H) .
ceq admit(C, W, H) = H if not(c-admit(C, W, H)) .
```

`c-admit : Cit Wid Hos -> Bool`は`admit`が適用され，病院の状態を変化させるための事前条件を満たしているかをチェックするための述語である．この述語によってチェックされる事前条件 C は $C = C_0 \wedge C_1 \wedge C_2 \wedge C_3 \wedge C_4$ の形をしており，$C_n(0 \leq n \leq 4)$ はそれぞれ，以下の条件である．

C_0: これから受け入れられる市民`C`は病院の敷地内にいるか．

C_1: 患者が治療を受け入れられたいと考えている病棟`W`はその病院に存在しているか．

C_2: その患者はすでに受け入れられている患者ではないか．

C_3: 患者が受け入れられたいと考えている病棟のベッドは充分にあるか．

C_4: その病院の中の患者の数が 0 人以上か (人数が負の数など起こりえない数になっていないか)．

上記の条件が満たされているとき (`c-admit(C,W,H)`が true を返すとき) に`admit`が適用されると次の観測関数が返す値が変化する．

`wset`：病棟の集合を観測するための観測関数．この集合の要素である病棟`W`が次のように変化する．

- 病棟`W`の属性である受け入れられている患者の ID の集合に新たな患者の ID が加えられる．
- 病棟`W`の下位実体の 1 つに医療スタッフステーションがある．その医療スタッフステーションのさらに下位実体にはカルテがある．医療スタッフステーションの下位実体の集合の 1 つであるカルテの集合に新たな患者のカルテが加わる．

`paset`：患者の集合を観測するための観測関数．新たな患者がこの集合に加えられる．

msset：医療スタッフの集合を観測するための観測関数．医療スタッフの属性には受け持っている患者の ID の集合がある．新たな患者を受け持つことになる医療スタッフの属性が変化する (患者の ID の集合に新たな患者の ID が加わる)．

その他の観測関数が返す値は admit が適用される前とされた後での違いはない．また，c-admit が false を返すときは全ての観測関数が返す値は変化しない．各遷移演算子による病院の状態の変化を公理として記述していく．

4.3　その他の事例

デジタル著作権ドメインの形式記述 [38]

音楽や映像のネットワークを通じての販売が大きなビジネスに成長したことから，デジタル著作権管理 (DRM: Digital Right Management) の社会的な重要性はますます大きくなっている．DRM は，著作権法などの法律システムとライセンス制度などの社会制度システム，さらには利益追求を目標とする企業の論理が勝る市場システムなど，技術論で整理しきれない多くの要素を含み，そのドメイン全体を体系的かつ形式的に記述することは困難である．我々は，作品の利用権 (演奏，複製，譲渡などの権利) を厳密に規定するライセンス言語に焦点を当て，その形式記述法を研究開発してきた．　DRM ドメインにおけるライセンス言語で書かれたライセンス規約は，一般的に

```
License i : licensor a grants licence b on work w
             with permitted actions {a1,a2,a3,⋯,am}
              and obligated actions {b1,b2,b3,⋯,bn}
```

のように記述される．こうしたライセンス言語の厳密な意味を，ドメインの形式記述を用いて与え，ドメインにおける振舞のより正確な記述と解析を行っている．　DRM ドメインにおいてはいくつかのライセンス言語が設計されているが，ライセンス言語の考え方は，他のドメインにおいても有効であると期待される．これに付いては 4 で述べる．

公的文書ドメインの形式記述と検証 [34, 35]

ドメインの形式記述が，行政システムや会社システムなどの社会システムの解析・検証の新しい方法論となり得ることを目指して，公式文書ドメインの形式記述と検証の事例研究を行っている．

公式文書ドメインとは，図 4.3 のように，法治国家における法令の制定，改訂，運用を，法令文書に対する生成 (制定)，改訂，閲覧，複写，破棄などの操作と，それらの操作を誰が行いうるかの権利関係としてモデル化したドメインである．この公式文書ドメインは，電子政府の機能をネットワーク上に流通する電子文書に施される操作とそれらの権利関係の視点から捉えたものでもある．また行政組織や会社組織における統治や管理の機能を，それらの組織内を流通する文書に施される操作とそれらの権利関係の視点から捉えたものでもある．

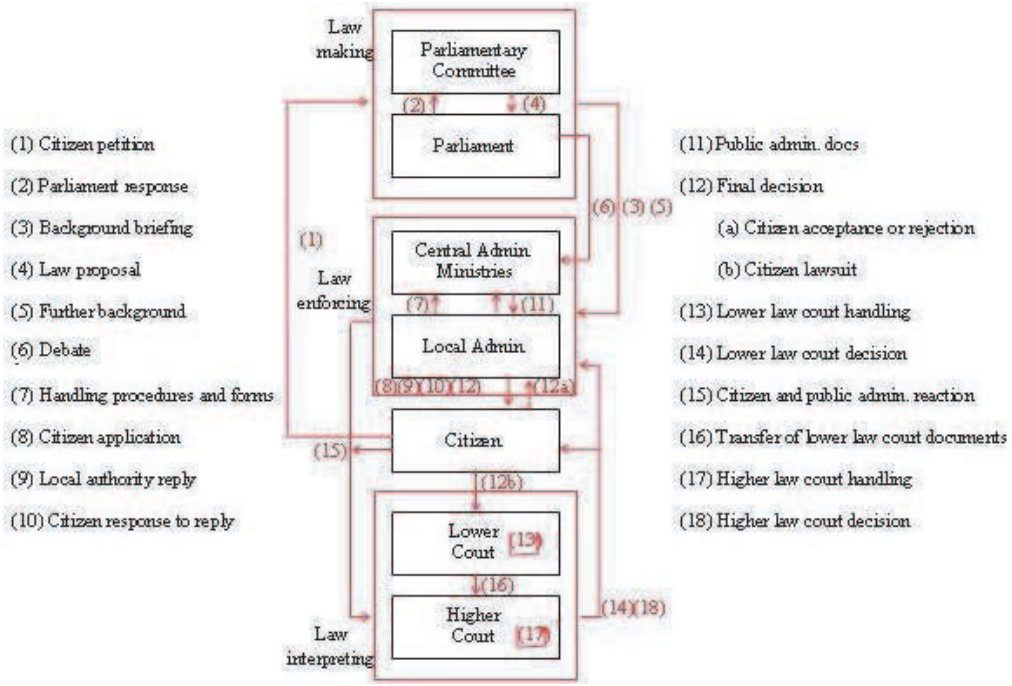

図 4.3: 公文書ドメインのモデル化

我々は法治社会における法令文書ドメインを以下のようにモデル化し，そのモデルにおいて「制定された法令が十分に利用者に解放されている」という意味の透明度 (transparency) の概念を定義し，それが成り立つか否かの解析を行っている．

4.4　振舞の記述・検証とライセンス言語

ドメインの形式記述は，問題としたいドメインを構成する実体とそれら実体に適用される関数を特定することで，そのドメインの基本構造をモデル化する．実体は種類により分類することで型付けされ, 関数も実体の型を用いて型付けされる. こうして得られた実体と関数の（名前の) 集合は，問題のドメインに付いて語るための言語となる．ドメインの様々な性質は，この言語を用いて記述・定義・定式化・形式化される.

ドメインに存在する実体が有する機能や, 実体の間の関係などは, 実体や関数を用いて直接的に記述・定義されることが多い. しかし，ドメインが示す振舞を記述・定義するためには, ドメインの状態を入力として新たな状態を出力する (特殊な) 関数をいくつか特定し, ドメインにおける事象（event）をこれらの関数の適用として定義し, それら事象の系列として振舞を記述・定義することが必要になる.

以上が，我々のドメインの形式記述の基本スキーマである．この基本スキーマにより，たとえば通信プロトコルや安全プロトコルで規定される，様々な振舞を記述・定義し検証することが出来る．安全プロトコルのドメインでは, 意味のある振舞（つまり事象の系列）を定義することがドメインの形式記述の最重要の要件であることが多い. この傾向は, ドメインにおける振舞を規定・規制するための，規則, 法令，ポリシーなどの解析・検証が目的であるときに顕著である.

ドメインの実体が動作（関数）を実行し得る条件を明確にすることで，ドメインの振舞を規定することが良く行われる．ライセンス言語は，先に DRM ドメインで説明したように，権利を有する実体が権利を行使する実体にその権利の行使権（ライセンス）を保証することを記述する．ライセンス言語の考え方は，DRM ドメインだけでなく，他のドメインにおいても振舞を規定する際に有効でありかつ重要である．特に法令，規則，プロトコルなどで規定される振舞の性質を解析することが主要な関心であるようなドメインにおいては，ライセンス言語（的なもの）で規定される振舞の記述と解析・検証が重要となる.

```
License Description
   1) study1-License
       L =  study1-license  actor G grants
               operations {Create Edit Read Copy Grant Calculate Shred }
               on documents { Feedback SR } to actor E
   2) study2-License
       L =  study2-License actor G grants
                operations { Read }
                on documents { Archives} to actor E
   3) study3-License
       L =  study3-License actor Company grants
```

```
            operations {Read Copy Restrict Calculate}
            on documents {Law R&R } to actor E
4) inform-License
    L = inform-License actor E grants
            operations { Read Copy Grants Calculate}
            on document {Feedback SR} to actor Company
```

公的文書ドメインにおけるライセンスの記述例

我々は，病院ドメインと公的文書ドメインにおいてもライセンス言語を設計し，ライセンスの形式記述に基づき定義された振舞の解析・検証の研究を行っており，これらに基づきより精密な振舞の記述・解析技術の研究開発を目指している．

第5章　法令実働化情報システムのアカウンタビリティ

5.1　法令実働化情報システムのソフトウェアアカウンタビリティと進化容易性

近年，電子政府・電子自治体への取り組みをはじめとして，社会システムの電子化が急速に押し進められている．行政，金融，医療，交通，教育，企業などの活動の基盤部分が電子システム化され，それらがネットワークを介して相互に接続され巨大な電子社会システムが構築されつつある．

われわれの日常生活は，社会基盤としてのこのような電子社会システムの上に成り立っている．したがって，電子社会システムは，これまでの情報システムのように機能を単に提供するだけでは十分ではなく，われわれが安心して生活できることを保証できるように設計・構築され，かつ運用・保守されている必要がある．

本学21世紀COEプログラム「検証進化可能電子社会」[6]では，安心できる電子社会システムが持っているべき要件として，正当性，アカウンタビリティ，セキュリティ，耐故障性，進化性の5つからなる安心性要件を提案しており，本研究はこのうちソフトウェアアカウンタビリティ（Software Accountability）および進化容易性（Ease-of-Evolution）を対象としている．進化容易性では，ソフトウェアの進化性に加えて，進化のための変更コストの低減も特に重要視する．

われわれの社会には，電子社会システムを含め属するものすべてが守らなければならない法律や条例および組織が定めている各種規則があり，これらを社会規則と呼ぶ．一般に社会規則は，自然言語で記述された文書であり，条・項もしくは節から構成される．

電子社会システムは社会規則の適用を支援し，社会規則を完全に満たすように構築されていなければならない．さらに，社会規則に従って電子社会システムが正しく構築されていることを保証でき，かつ確認できる必要がある．また，社会は常に変化しており，社会規則もそれに応じて改定される．電子社会システムは，社会規則の改定に応じてシステムを迅速に進化させていかなければならない．そのためには低いコストでシステムを変更できる必要がある．こういった特徴を持つ

システムを法令実働化情報システム，Law Enforcing Information System（以下，LEISと略記する）と呼ぶ．

社会規則には，規則の目的，用語定義，事実記述，ワークフロー記述，各種制約および条件の定義，計算式の定義，データ定義などが含まれ，規則適用のドメイン内において有効である．一般にドメインの一部分をシステム化する場合，システムが社会規則に規定されている上記の内容すべてに関係することは少ない．LEISはドメイン内のシステム化する部分に応じて，ワークフロー型と制約型に大きく分けることができる．

5.2 ソフトウェアアカウンタビリティとは

アカウンタビリティとは，文献 [46] によると,「一般に説明責任と訳される．具体的には政府・行政などの国民に対する政策成否の説明責任や，経営者の株主に対する財務状況，経営戦略の展開，見直しとその成果などについての説明責任について用いられている」,「行政機関または公務員個人が行った判断や行為に関して，国民が納得するよう説明しうること」とある．

これに基づいて，まず，ソフトウェアアカウンタビリティを以下のように直感的に定義する [45].

> **「LEIS自体が，行った判断や計算結果に関して，その理由をシステムの利用者に説明できること．言葉をかえれば，LEISが，LEISが行った判断や計算結果に対して利害関係者が疑問を持ったとき，利害関係者の質問に答え得ること．LEISは，利害関係者を満足させる回答をシステムの実行状況を利用して生成できる必要がある．」**

ここで利害関係者を，単なるシステム利用者に限定せず，システムの一般利用者，業務担当者，社会規則整備担当者，システム開発・保守担当者の4種類に拡張する．いくつかのLEISとその利害関係者の例を以下に挙げる．

- 大学の履修規則には，大学の教育理念に基づいて，修了のための資格が定義されており，また，資格を得るために必要な様々の条件とその修得法が示されている．学生の履修登録作業および修了要件の達成状況の確認作業を支援する履修管理システムはLEISであり，学生，事務員，教員，履修管理システム開発者が利害関係者となる．

- 地方自治体には，様々な条例がある．地方自治体システムには，立法担当者，行政担当者，システム開発者，一般市民などの利害関係者が関与する．

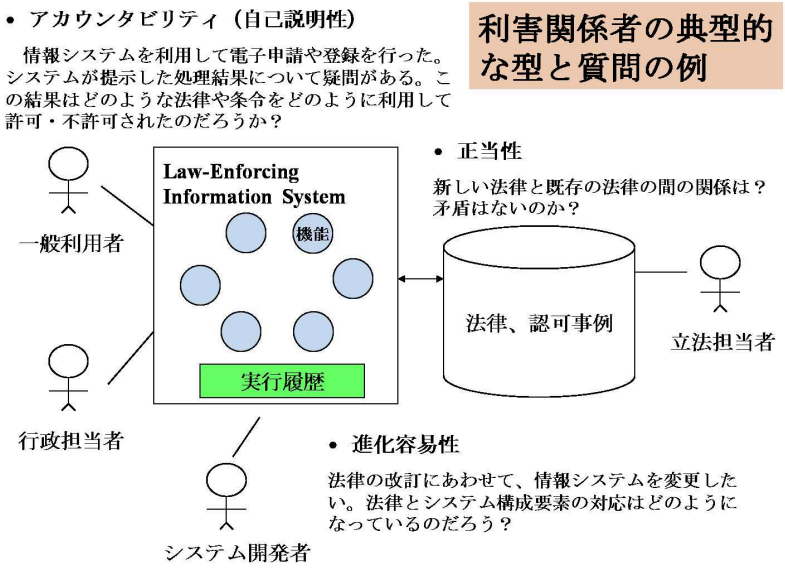

図 5.1: 利害関係者が持つ典型的な質問

- 会社の社内規定には，運営方針に従った様々な決定の基となる規則が定められている．社内システムには，管理者，担当者，従業員などの利害関係者が関与する．

図 5.1 に，地方自治体システムに関与する 4 種類の利害関係者の例と，彼等が持つ典型的な疑問を記す．

- 県や市の立法担当者は，新しい法律の制定をはかる場合，当該法律の内容のみならず，従来の法律との整合性にも関心を持つ（これを安心性要件の正当性の問題とする）.「新しい法律と既存の法律の間の関係は？矛盾はないのか？」などの疑問を持つ

- システム開発者は，初期のシステム開発時には，開発する情報システムに法律内容を正確に反映させることに関心を持ち,「規則を必要十分に実現したか？」という疑問をもつ．また，法律の改定にともなうシステム保守においては,「法律の改定にあわせて，情報システムを変更したい．法律とシステム構成要素の対応はどのようになっているのだろうか？」などの疑問をもつ．すなわち，アカウンタビリティはシステムの進化容易性とも関連する．

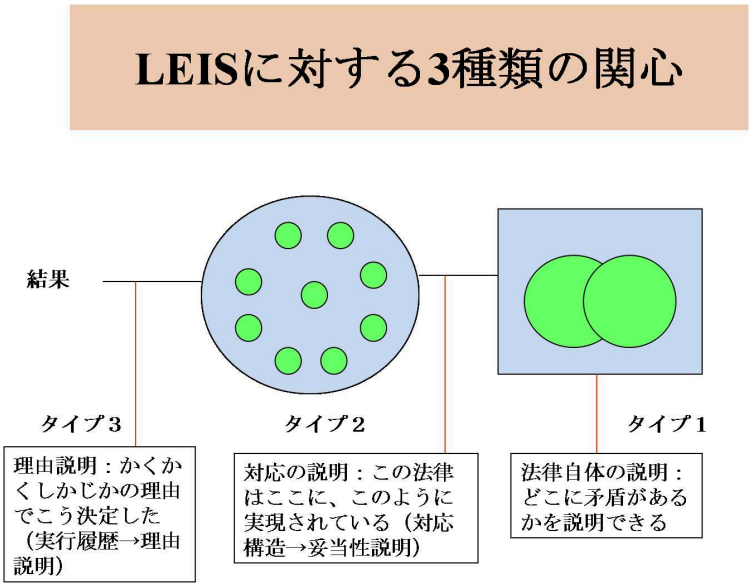

図 5.2: 利害関係者の 3 種類の関心

- 行政担当者やシステムを利用する一般市民は，システムが提供する実行結果に関心を持ち,「情報システムを用いて電子申請や登録を行った．システムが提示した処理結果について疑問がある．この結果はどのような法律や条令をどのように適用して得られたものだろう？」などの疑問を持つ．

各種の利害関係者が持ちうる疑問（質問）を,「システムのどの側面に関係するか」という観点から，図 5.2 に示すように 3 つのタイプに分類する．

図 5.2 において，

- タイプ 1 は，規則そのものに注目した疑問である．
- タイプ 2 は，規則とシステム構成要素の対応構造に注目した疑問である．
- タイプ 3 は，システムの実行結果に注目した疑問である．

以上述べてきた問題設定にしたがい，5.3 節において，ソフトウェアアカウンタビリティの概念を定義するにあたっての基本的立場を，ソフトウェア工学における成果と法理論における研究成果を背景にして示しつつ，ゴール指向木によって編成され，法理論によって型付けされた，アカウンタビリティ木の概念を新たに

導入する．5.4 節において，ソフトウェアアカウンタビリティ機能の実現方式を検討しつつ，ソフトウェアアカウンタビリティモジュールを定義する．5.5 節において，ソフトウェアアカウンタビリティモジュールを LEIS に装着するためのソフトウェアアーキテクチャについて論じる．5.6 節で，大学の履修管理システムを例にとったタイプ 3 のアカウンタビリティ機能の実現例を示す．

5.3 ソフトウェアアカウンタビリティ定義の立場

ソフトウェアアカウンタビリティの定義にあたっての基本的立場を，ソフトウェア工学的側面と法理論的側面からそれぞれ考察する．

5.3.1 ソフトウェア工学的側面からの考察

ソフトウェア工学における要求定義技術に関する研究成果は，ソフトウェアアカウンタビリティの定義にあたって有用である．要求獲得のプロセスを以下のようにとらえる．

> **「種々の利害関係者は，対象領域や LEIS に対して，直接または間接に独自の関心をもつ．各利害関係者は対象世界の学習プロセスを経て，対象世界に関する固有の理解に到達する．理解した結果を彼等自身の言語で表現する．」** [1]

この見解をもとに，ソフトウェアアカウンタビリティ定義の基本的立場を，以下のように定める．

> **「種々の利害関係者は，社会規則が制定され，システムが開発される前に，社会規則やシステムに独自の関心を持ち，その内容を学習し，学習した結果を彼等自身の言語で表現する．さらに，社会規則が制定され，システムが開発された後に，これらの学習結果にふたたび関心を示し，必要に応じて，それらを再獲得しようと試みる．」（図 5.3）**

LEIS がソフトウェアアカウンタビリティの能力を持つとき，それをソフトウェアアカウンタビリティ機能と呼ぶ．ソフトウェアアカウンタビリティ機能を実現するためには，これらの学習結果が，システム中に記録され，必要に応じて検索され得る必要がある．

[1] この見解は東京エレクトロンソフトウェアテクノロジー 熊谷 章 氏との討論によって示唆されたものである．

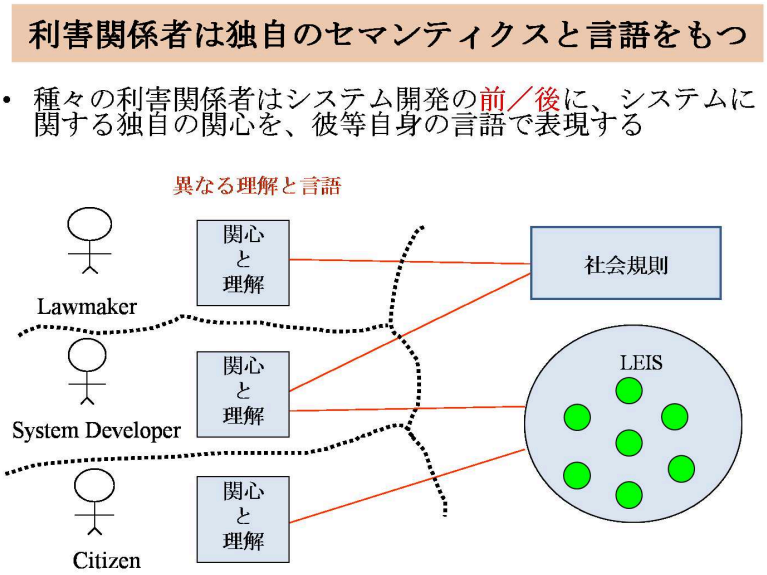

図 5.3: 各利害関係者の関心と理解は（構造化して）記録され共有されるべきである

上記の視点に立つとき，従来の要求定義法（図 5.4）を採用することには問題がある．

従来の要求定義法では，様々な利害関係者からの要求獲得の結果を，単なる機能仕様や非機能仕様として表現する．ソフトウェアアカウンタビリティ機能の基となる大切な情報はシステム分析者の中に埋没し，システム世界と切り離されて存在することになる危険性が高い．

これに関して近年，**ゴール指向要求分析法** [41, 47] に関する研究が行われており，ここで議論する問題と関係が深い [2]．

ゴール指向要求分析とは,「保守が容易である」,「ユーザビリティがよい」などのシステムに対する非機能要求をゴールとして設定し，それを AND-OR 木を利用してサブゴールに展開していく手法である．葉にあたる部分には通常の機能要求がくる．

この分析法の一つの特徴は「ソフトゴール」という概念にある．人工知能分野におけるゴールとは異なり，サブゴールの充足に関して，

[2]筆者によるここまでの考察結果が，ゴール指向分析の研究目標と関連が深いとの指摘は，NTT データ 山本 修一郎 氏によって示唆された．

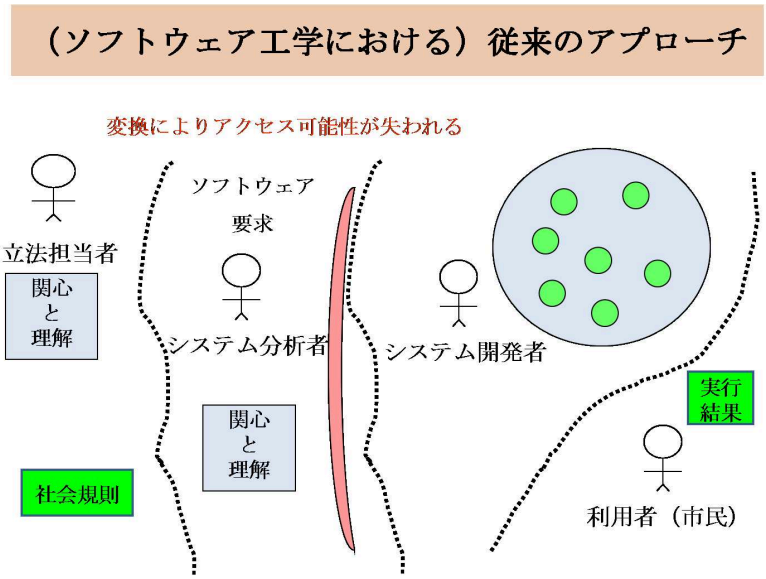

図 5.4: 従来の要求定義法における扱い

- 「肯定的な証拠が十分にあり，否定的な証拠はほとんどない」ときサブゴールは充足され，
- 「否定的な証拠が十分にあり，肯定的な証拠はほとんどない」ときサブゴールは非充足となる．

ゴール指向要求分析を採用することにより，ゴール指向木を図 5.5 に示すような階層で編成すれば，各利害関係者のもつセマンティクスとその相互関係を表現することが可能になるものと考える．

図 5.5 に示した方針に従って，北陸先端科学技術大学院大学情報科学研究科の履修規則制定にあたって，各利害関係者が示した関心と学習結果の構成要素を配置した図を図 5.6 に示す．

これらの情報は，

- 社会規則（この例の場合は履修規則）の改定（進化）を制御するため
- 利害関係者からの，規則制定のいきさつに関する質問に答えるため

に有用である．

ゴール指向木は有用ではあるが，それだけでは十分ではない．

複数の利害関係者が理解した世界に基づく
ゴール木の構成

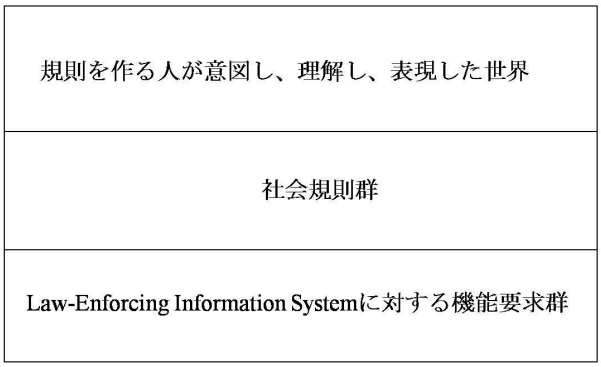

図 5.5: ゴール木の構成方針

- 社会規則はたがいに関連しあっているので，社会規則自体を構造化する必要がある．すなわち，図 5.5 における第 2 層を構造化する必要がある．
- 社会規則を分類しておくほうが関連規則の検索には有用である．

そこで，次節において法理論の成果の適用も試みる．

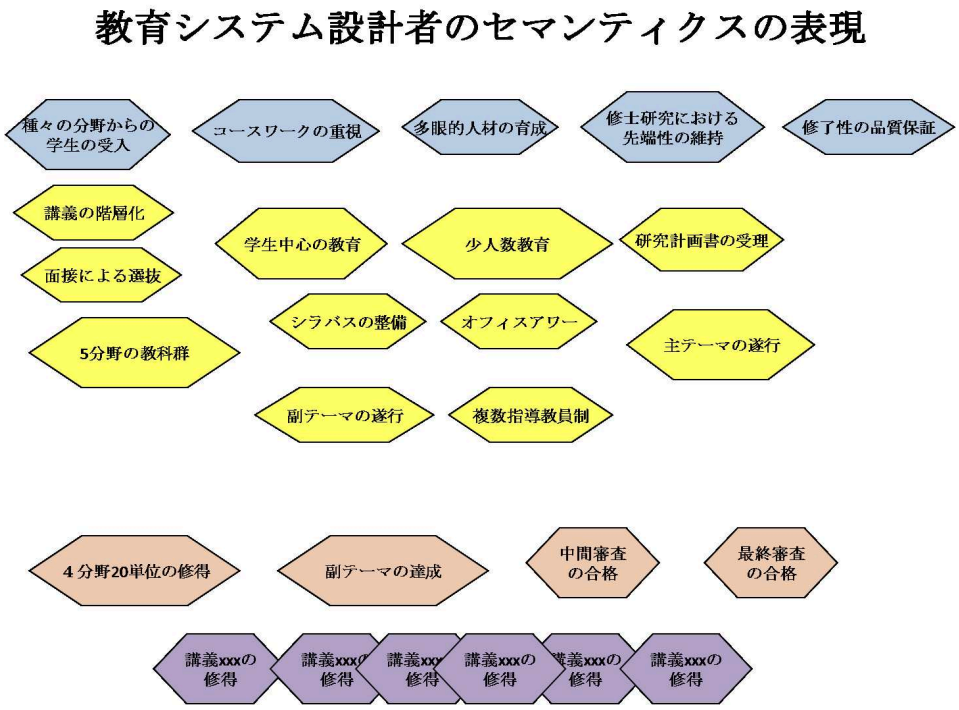

図 5.6: 各利害関係者の関心の階層化例

5.3.2 法理論の立場からの考察

エッフホク [44] によれば，法システムは規範と活動からなり，それらの間には種々の連関が存在する．以下の議論の展開に関係する部分を文献 [44] より要約引用する．

規範とは，準則（則るべき規則），原理，規定，標準，範型，指針，基準などの総称であり，指令，性質決定，授権に分類される．規範内容を言語的に表現したものを規範的言明という．

- **指令**とは，命令，催告，懇願，助言，警告，約束などの総称であり，感化（人に影響を与えて心を変えさせる）の意図を直接的に表現している．
- **性質決定**とは，どのような現象が一定のカテゴリに入れられるのかを示す．数学における演繹ルールに相当する概念である．
- **授権**とは，指令・資格もしくは新しい権限を付与する権能を人に与える言明

である．授権は，法システムにおいて中心的な役割を果たす．立法権と司法権についての基本法の規定，さまざまな決定権限を行政機関に付与する法律などの例がある．

指令もしくは指令の否定を言語的構成要素として含む規範の総称を**義務規範**（または，行動規範）と呼ぶ．義務様式は，命令，禁止，許可，免除の４つの下位グループをもつ．命令された行為は着手する義務がある．禁止された行為は中止すべき義務がある．許可された行為は命令された行為と選択自由な行為（禁止されても命令されてもいない）を含み，免除された行為は，禁止された行為と選択自由な行為を含む．

義務規範には２つのカテゴリの規範主体がある．

１つは規範によって義務を課される人（人々）から構成される．もう１つは，この義務が向けられる準拠集団によって構成される．後者の地位はしばしば権利ということで特徴づけられている．

指針は，どのような観点を選択するかに応じて，義務規範として理解されることもあれば，性質決定規範として理解されることもある．

規範間および規範と活動の間には**連関**がある．規範間の連関には静態的連関と動態的連関がある．

静態的連関（意味連関）とは，法システム内で変化が起こっても，それにより影響を受けない連関である．

- **カップリング連関**　２つ以上の規範的言明（および，その言明の構成要素をなす規範）が，全体で完全な規範的言明を構成するように相互に結びつきあっているという特徴である．法律の１つの定義を，それに対応する定義が現われるいたるところで参照する例がある（例えば,「親族」を定義した規定をいたるところで参照する）．

- **意味の集積**　ひとつの同じ言い回しが複数の言明の担い手になっていること．ある法律が，当局はあれこれのことを「指示することができる」と述べているとすると，このことは，一定の規範を発する権限がその当局にあるということを意味するだけでなく，この権限を用いるかどうかの決定が当局に任されているということをも意味しうる．

- **論理的関係**　等価・包含・矛盾なども一種の意味連関である．

動態的連関とは，現実または想像上の行為の流れにおける諸段階に生じる連関である．規範適用および規範定立から生じる連関である．

- **因果連関**　事象の生起の最初の段階（原因）と第２の段階（結果）との連関．原因が結果をひきおこす．因果的に結びつくことができるのは，事実（状態，行為，事象等）だけである．

- **規範的連関**　事実と事実の連関．ただし，因果連関ではなく，規範に基づく連関である．「Aが盗みをしたため，懲役の判決をうけるべきである」は，事実「Aは盗みをした」と「懲役の判決をうける」が規範によって結びつけられる．２つの事実の間に規範的連関があることは，それらの間に因果連関があることを否定しない．

- **操作的連関**　事象の推移において，ある段階と段階が連続したものであるとき，それは，ある段階で適用される規範が，次の段階で適用される規範を決定するという形で整序される．

規範だけでなく，**一定の活動も法システムの構成要素**である．法的機関が行ういくつかの行動の主な部分は，種々のタイプの決定を準備し，決定を下し，決定を理由づけるという点にある．法的システムにはたえず入力があるが，それは特に，要求と期待という形をとって行われる．ひとつは法的請求を実現するのを手助けせよというような個別的な要求であり，もうひとつは，法システムそのものに変更を加えよという要求である．**法適用**と**法形成**の２つの動態的プロセスがある．

- **法適用**に特徴的なのは，法規範と法的評価が決定の根拠を形作っているという点である．

- **法形成**とは，法規範の作成，変更または廃止を意味する．

法適用と法形成において，**熟慮プロセス**が重要である．熟慮プロセスは，問題とデータを入力し，**立場**と**理由づけ**を出力する（図 5.7）．

- **立場**とは，何かあるものについて，それがどのようでなければならないか，どのようであってはならないのか，どのようになされなければならないかを表明する．

- **熟慮**とは，ある立場に至る心理的プロセスである．

- **理由づけ**とは，ある立場を説明し，拡張し，補足することである．

法理論を用いると社会規則を構造化しかつ型付けすることが可能になり，種々の質問に対して適切な規則と関連する規則を取り出すための基礎を与えうる．

法システムにおける行為

• 法適用と法形成過程における熟慮プロセス
はわれわれの関心に強い関係をもつ

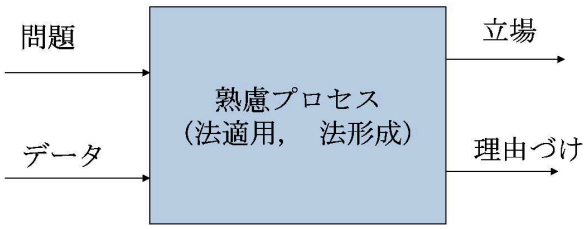

図 5.7: 熟慮プロセス

5.3.3　ソフトウェアアカウンタビリティ機能の実現に関する基本方針の設定

5.3.1 節における分析と 5.3.2 節における理論を関連づけることにより，LEIS のソフトウェアアカウンタビリティ機能を定義・実現するための方針をたてることができる．以下に示す．

- ゴール指向木により，種々の利害関係者が理解し表現した世界を構造化して表現する．
- 図 5.5 における，第 1 層「規則を作る人が意図し，理解し，表現した世界」を，5.3.2 節における，法適用と法制定に関する動態的プロセスの入出力をもとに設計することが可能である．すなわち，熟慮プロセスの入力,「問題とデータ」と「立場と理由づけ」の相互関係または,「立場の理由づけ」そのものをゴールとして設定すればよい．
- 例えば，図 5.6 において，北陸先端科学技術大学院大学の教育システムのゴール,「種々の分野からの学生受入」,「コースワークの重視」,「多眼的人材の育成」,「修士研究における先端性の保持」,「修了生の品質保証」などが設置委員会によって熟慮され，以下のサブゴール（サブゴールの一部は，大学開設

後の教育制度委員会で生み出された）が出力された.「講義の階層化」,「5 分野の教科群」,「学生中心の教育」,「シラバスの整備」,「主テーマの遂行」,「副テーマの遂行」,「少人数教育」,「オフィスアワー」,「複数指導教官制」「研究計画書の受理」など.

- 次に，それらを達成するために，学生に指令する形で履修規則が整備された. 具体的には,「4 分野 20 単位の修得」,「副テーマの達成」,「研究計画書の受理」,「中間審査の合格」,「最終審査の合格」などのサブゴールが学生の活動の流れにそって定められ，履修規則により，細かいルールが規定されたことになる.

- 上記階層に存在する情報は，教育システムを評価し，必要なら改定を行うときに必要となる，基本かつ重要な情報である．すなわち,「熟慮」,「立場」および「理由づけ」に関する情報は，利害関係者の関心と直接的な関係があり，質問に対する回答を生成するために必要な情報の基となる.

- ここまでは，図 5.5 の各階層間の関係を論じてきたが，各階層の情報の構造化もまた必要である.

- 各階層の情報の構造化にあたっては 5.3.2 節における動態的連関に関する基礎考察の結果がとくに関係が深いものと考える．すなわち,「何故私は研究計画書を提出できないのか？」というタイプの質問には，規則間に張られた，因果連関や規範的連関を利用して答えることができる.

5.3.4 アカウンタビリティ木

図 5.6 における第 1 層の各要素を，ゴール指向木で編成し，第 2 層の社会規則を葉に対応させ，かつ，法理論で型付けした木を**アカウンタビリティ木**と呼ぶ. 図 5.8 に例を示す．なお，社会規則間の静的／動的連関はアカウンタビリティ木のクロスリンクとして表現する.

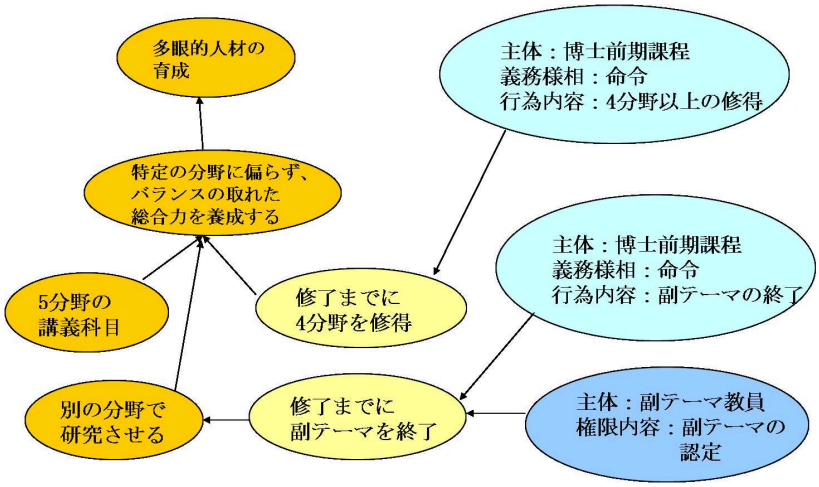

図 5.8: アカウンタビリティ木の一例

5.4　ソフトウェアアカウンタビリティ機能の実現に関する考察

アカウンタビリティ機能を実現するための情報が LEIS 作成過程のどこに存在するのかをまず考察する．

図 5.9 に，われわれが想定している LEIS の開発プロセスを示す．LEIS 開発プロセスの概要を以下に示す．

- 平文で表現された規則（法律）は，自然言語処理により「**論理表現**」に自動変換される．
- 論理表現は法推論機構により自動解析され，**矛盾が検出**される．矛盾解消は人手により行う．ここまでのサイクルを法令デバッギングと呼ぶ．
- 法令デバッギングにおける中間表現は LEIS 設計の入力となる．これに関して 2 つの方式を検討中である．
  - **クラス図を自動生成**する．
  - 利害関係者のワークフローを利用して規則の使い方に関する情報を補足した後，論理表現にあらわれる概念を利用してユースケースを作成し，ユースケース駆動型ソフトウェア開発方法論にしたがって**クラス図を設計**する．

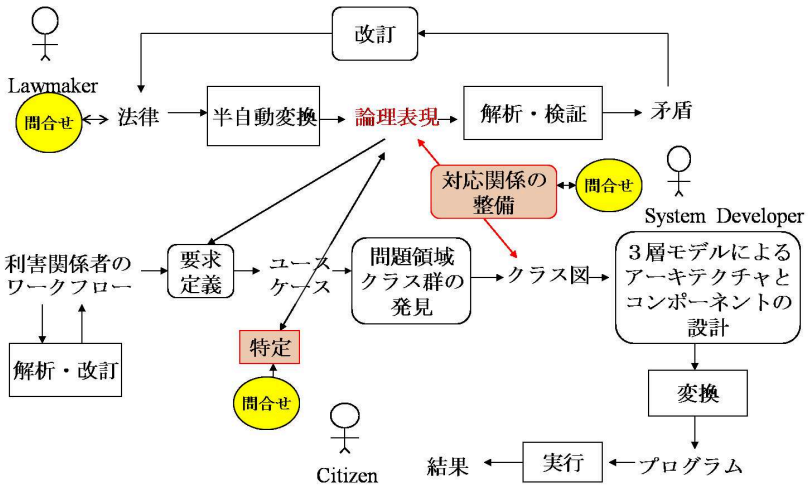

図 5.9: 3 種のアカウンタビリティ機能と LEIS 開発プロセスとの関係

- コンポーネント設計手法に基づき 3 層モデルアーキテクチャを採用してシステムを自動生成する.

上記の処理の流れにおいて，ソフトウェアアカウンタビリティに関与する情報は，図 5.9 の以下の箇所に保持される.

- **論理表現**　タイプ 1 のアカウンタビリティ機能を実現する際の，すなわち，法令デバッギングの一次情報となる.
- **対応関係**　論理表現とクラス図の対応関係を保持する．タイプ 2 のアカウンタビリティ機能を実現する際の一次情報となる.
- **論理表現，対応関係，システムの実行履歴**　タイプ 3 のアカウンタビリティ機能を実現する際の一次情報となる.

上記，論理表現と対応規則を利用して，ソフトウェアアカウンタビリティ機能を以下のように実現する予定である.

- **タイプ 1 のアカウンタビリティ**　法令デバッギングの過程に機能を盛り込む.
- **タイプ 2 のアカウンタビリティ**　対応関係を保持するデータベースと検索システムを開発する.

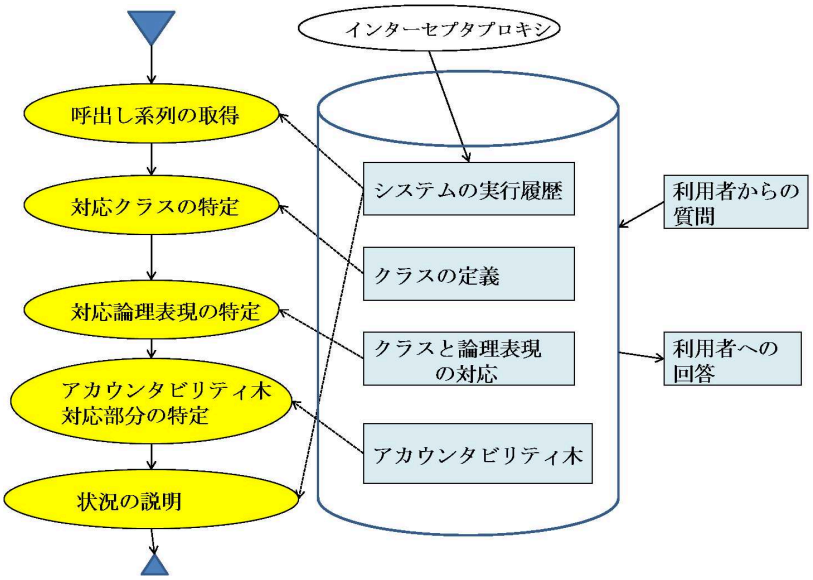

図 5.10: タイプ 3 のアカウンタビリティ機能の実現方式

- **タイプ 3 のアカウンタビリティ**　システムの実行履歴を保持する履歴データベースを設計し，問合せの内容と実行履歴を利用して，対応関係のサブセットを特定する．これにより論理表現のサブセットが特定できる．

これ以降は、タイプ 3 のアカウンタビリティ機能の実現方式に議論を限定する．タイプ 3 のアカウンタビリティ機能は図 5.10 に示す方式に基づき実現する．アカウンタビリティ機能として必要なのは以下の 2 つの機能である．

- 関連する社会規則を特定する機能．
- 特定された社会規則がどのように適用されたのかを説明する機能．

このうち，関連する社会規則を特定する機能は以下の手順で実現する．

- インターセプタプロキシ（次節で導入する）がシステムの実行履歴を記録する．
- 実行履歴のうち，呼出し系列の情報をもとにして実行に関与したクラス群を特定する．

- クラス群と論理表現の対応構造を利用して，論理表現の部分集合を特定する．
- 論理表現はアカウンタビリティ木の葉として表現されている．
- 実行履歴のうち，引数の値等は，質問者の状況を表すデータとして説明に利用する．
- 上記の結果のうち，利用者の質問に関係する部分を特定し，利用者への回答を作る．

以後，図 5.10 に該当する情報構造と機能を持つソフトウェアモジュールを**ソフトウェアアカウンタビリティモジュール**と呼ぶ．

なお，上記の実現法とは独立に，利用者からの質問と，法理論によって型付けされたアカウンタビリティ木との間で，ベクトル空間法を用いて照合を行い，質問に該当する規則と関連規則を取り出す方式も試みている [48]．

5.5 既存システムにアカウンタビリティモジュールを装着する参照アーキテクチャ

参照アーキテクチャ設計にあたっての要件は以下の通りである．

- アカウンタビリティモジュールを既存の情報システムに低コストで結合できる．
- 既存システムを最小の変更コストで再利用する．
- 既存の情報システムのアーキテクチャとして 3 層モデルを仮定する．

LEIS はウェブシステムとして実現されることが多く，一般にウェブシステムはユーザインタフェース層，プロセス管理層，データベース管理層から成る 3 層モデル [43] に基づいて実現される．3 層モデルに基づいたアカウンタビリティモジュールを装着可能な参照アーキテクチャを図 5.11 に示す [50, 51]．

参照アーキテクチャの特徴は，既存システムのユーザインタフェース層とプロセス管理層との間にインターセプタプロキシ [42] を配置し，既存システムに対しアカウンタビリティモジュールを付加できる構成になっている点にある．また，アカウンタビリティ機能はシステムの実行履歴 DB と社会規則 DB を利用して実現する．

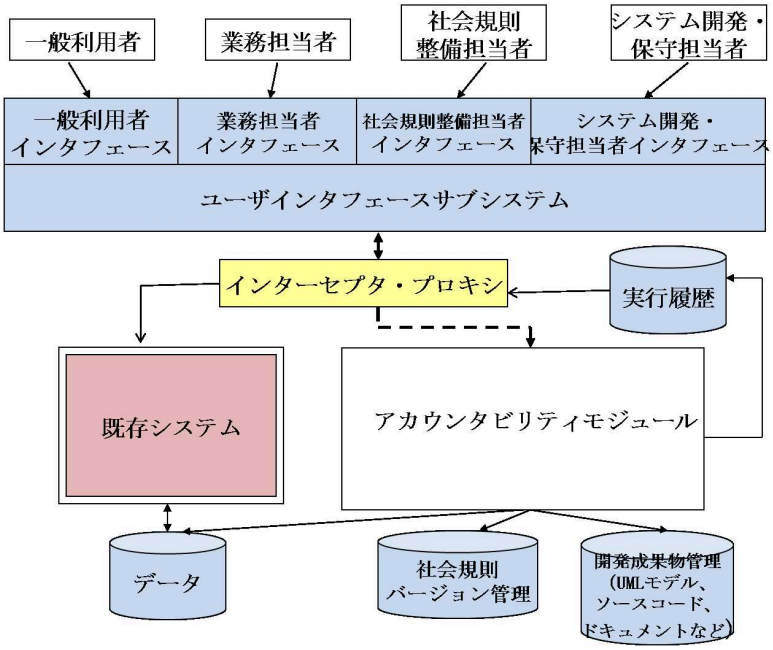

図 5.11: 3 層モデルに基づく参照アーキテクチャ

5.6 タイプ 3 のアカウンタビリティ機能を対象とした実証実験

本節では，タイプ 3 のアカウンタビリティ機能を対象として実施した実証実験の成果をまとめる．

本学の履修規則に則った履修管理システムを対象として事例研究を行った．

- まず，履修管理システムをユースケース駆動オブジェクト指向開発方法論にしたがって開発した．これは，Java EE (Java Platform, Enterprise Edition 5) プラットフォームを用いたクライアント・サーバ方式のウェブアプリケーションである．
- 次に，これに追加するかたちで，参照アーキテクチャに基づいてアカウンタビリティ機能の実現を行う．

本節では，参照アーキテクチャと Java EE プラットフォームとの対応を示し，Java EE の中でも特に JBoss Seam ウェブアプリケーションフレームワーク [40] を

用いた場合の実現方針を示す.

本節の構成は以下のとおりである．5.6.1 節で開発した履修管理システムの概要とシステム構造を説明する．次に 5.6.2 節で，履修管理システムにおけるアカウンタビリティ機能の実現法を論じる．5.6.3 節で実行例を示す.

5.6.1 履修管理システム

ユースケース駆動オブジェクト指向方法論 COMET [39] に基本的にしたがい，履修管理システムを開発した．本システムは本学が定める履修規則に則っている．開発成果物を示しながら，履修管理システムの概要およびシステム構造を説明する.

ユースケース図およびユースケース記述

開発にあたって，まず，システムの主な利用者である本学情報科学研究科の学生全員を対象としたアンケート調査により要求獲得を行った．アンケートの回答を集計し，履修管理システムへの機能要求を整理した．図 5.12 にこれを基に定義したユースケースモデルの一部を示す．なお，システムに関する利害関係者（学生，職員，教員，システム開発者）はすべてアクターとして抽出されるべきであるが，本開発では修士課程の「学生」アクターのみを対象にしている.

学生は，履修登録・修正を行うことができ，登録結果を確認できる．また,「修得状況を確認する」ユースケースでは，修得済みの科目名，取得単位数，評価点の一覧を確認できる.「チェックポイント通過条件をチェックする」ユースケース（図 5.13）は,「修得状況を確認する」ユースケースの中で学生が要求した場合に実行される．本学の定める履修規則には，修士課程を修了するために４つのチェックポイント （「副テーマ開始要件」,「研究計画書提出要件」,「就職推薦要件」,「修了要件」）があり，それぞれの要件を満たしていなければチェックポイントを通過できない.「チェックポイント通過条件をチェックする」ユースケースでは，現在の修得状況における各チェックポイントの通過可否を確認できる.

履修管理システムに対するユースケースモデル（一部分）

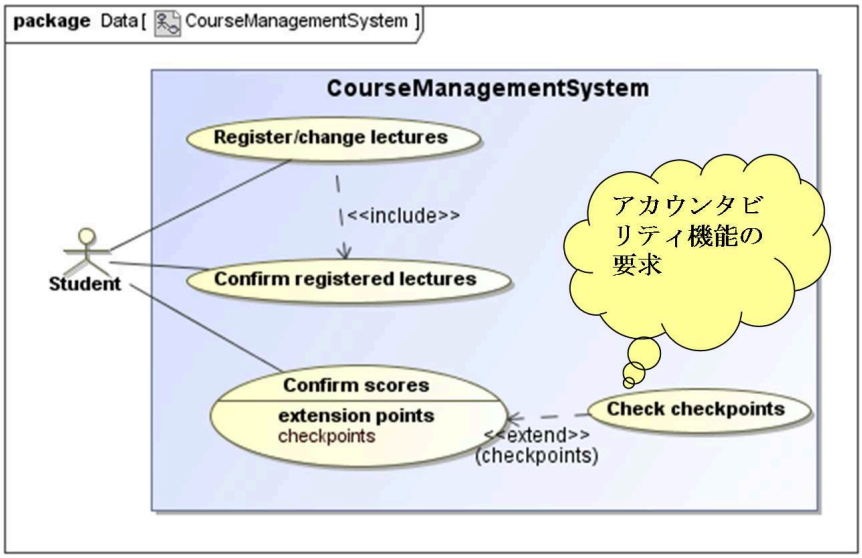

図 5.12: 履修管理システムのユースケース図

- **ユースケース名** チェックポイント通過条件を検査する
- **アクター** 学生
- **事前条件** 学生はログイン状態である。履修規則DBが有効である。
- **記述**
  1. 学生は、チェックポイント一覧からチェックポイントを選択する。
  2. システムは、チェックポイントに対応する条件を履修規則DBから読み出す。
  3. システムは、チェックポイントに対応する期数の(1)科目修得状況および現在作成中の期に対応する(2)スケジュール表を学生履修DBから読み出す。
  4. 履修見込み状況を(1)＋(2)とする。
  5. すべての条件を計算し終わるまで
     1. システムは、履修見込み状況が各条件を満たすか（Bool値）を計算する。
     2. 検査結果を検査結果ベクトルに書き込む。
  6. 検査結果ベクトルと検査結果を出力する。
- **代替**
  ３で必要とする期数の(1)が学生履修DBに入力されていない場合
  a1.1 システムは情報不足エラーを出力する
  ４の代わり//３で(2)がnullの場合に相当する。
  a2.1 履修見込み状況を(1)とする
  ５.1の代わり
  a3.1 システムは、各条件と履修見込み状況の差分を計算する
- **事後条件**

- 結果ベクトル：：＝ （通過可否 不足項目 不足数）＊
- 通過可否：：＝ Bool
- 不足項目：：＝ 分野数｜分野別単位数｜総単位数｜専門性別単位数 ｜ 副テーマ単位数
- 情報不足エラー：：＝ メッセージ 時期＊
- チェックポイント：：＝ 副テーマ｜RP｜就職推薦基準｜修了要件

図 5.13: 「チェックポイント通過条件を検査する」ユースケース記述

画面遷移図

本システムはウェブアプリケーションであるため，画面遷移と画面レイアウトをユースケースモデル定義後に設計した．その結果を図 5.14 に示す．

メニュー画面のメニュー項目が各ユースケースに対応する．ログイン後すべての画面に「メニュー」ボタンが配置してあり，いつでもメニュー画面に戻ることができる．

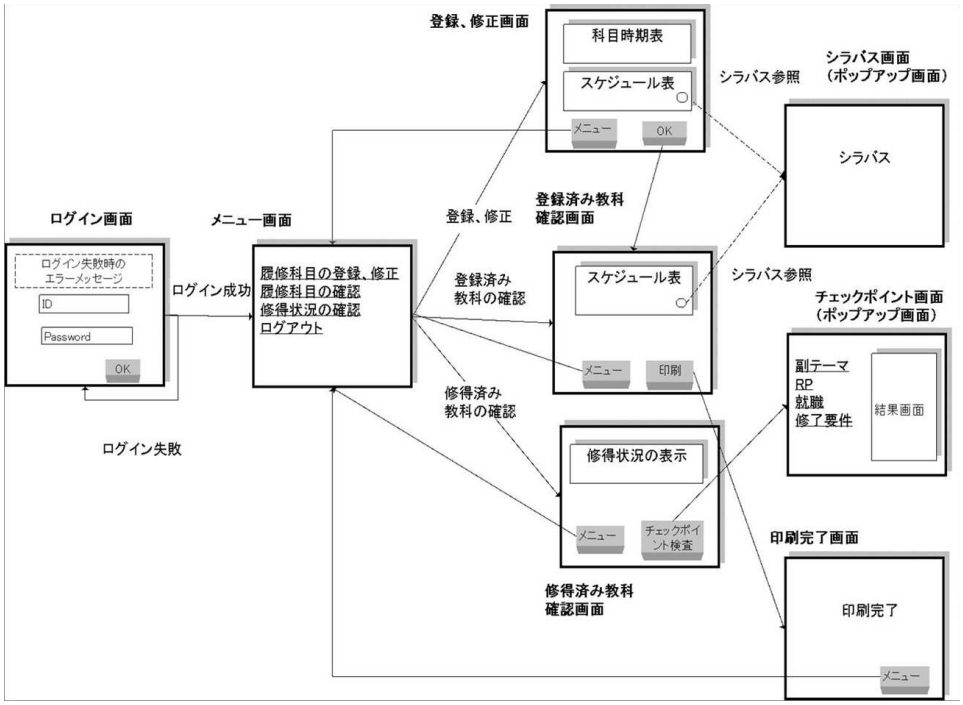

図 5.14: 画面遷移図

プラットフォームおよびウェブアプリケーションフレームワーク

現在，複数のウェブアプリケーションフレームワークが存在し，利用可能である．本開発では，Java EEプラットフォームを選択し，フレームワークとして JBoss Seam を採用した．JBoss Seam は，プレゼンテーション層の技術である JSF (JavaServer Faces) と分散コンポーネント技術 EJB (Enterprise JavaBeans) 3.0 を統合して使用可能であり，ステートフルコンポーネントを扱うことができ，宣言的な方法でアプリケーションの状態管理を行うことが可能であるといった特徴のあるフレームワークである．

クラス図

図 5.15 にクラス図を示す．ただし，EJB Session Bean のローカルインタフェースは省略している．

図 5.15 において,「EJB Entity Bean」ステレオタイプのついたクラスは O/R (Object/Relational) マッピングされたクラスであり，プロパティはリレーショナルデータベースに永続化される.「EJB Session Bean」ステレオタイプのついたクラスはビジネスロジックを実装したクラスであり，画面遷移などのイベントに応じて呼び出されるメソッドを持つ．実際は,「EJB Session Bean」ステレオタイプ付きクラスのプロパティのうちいくつかは JBoss Seam の提供する DI (Dependency Injection) 機構を用いて実装している．

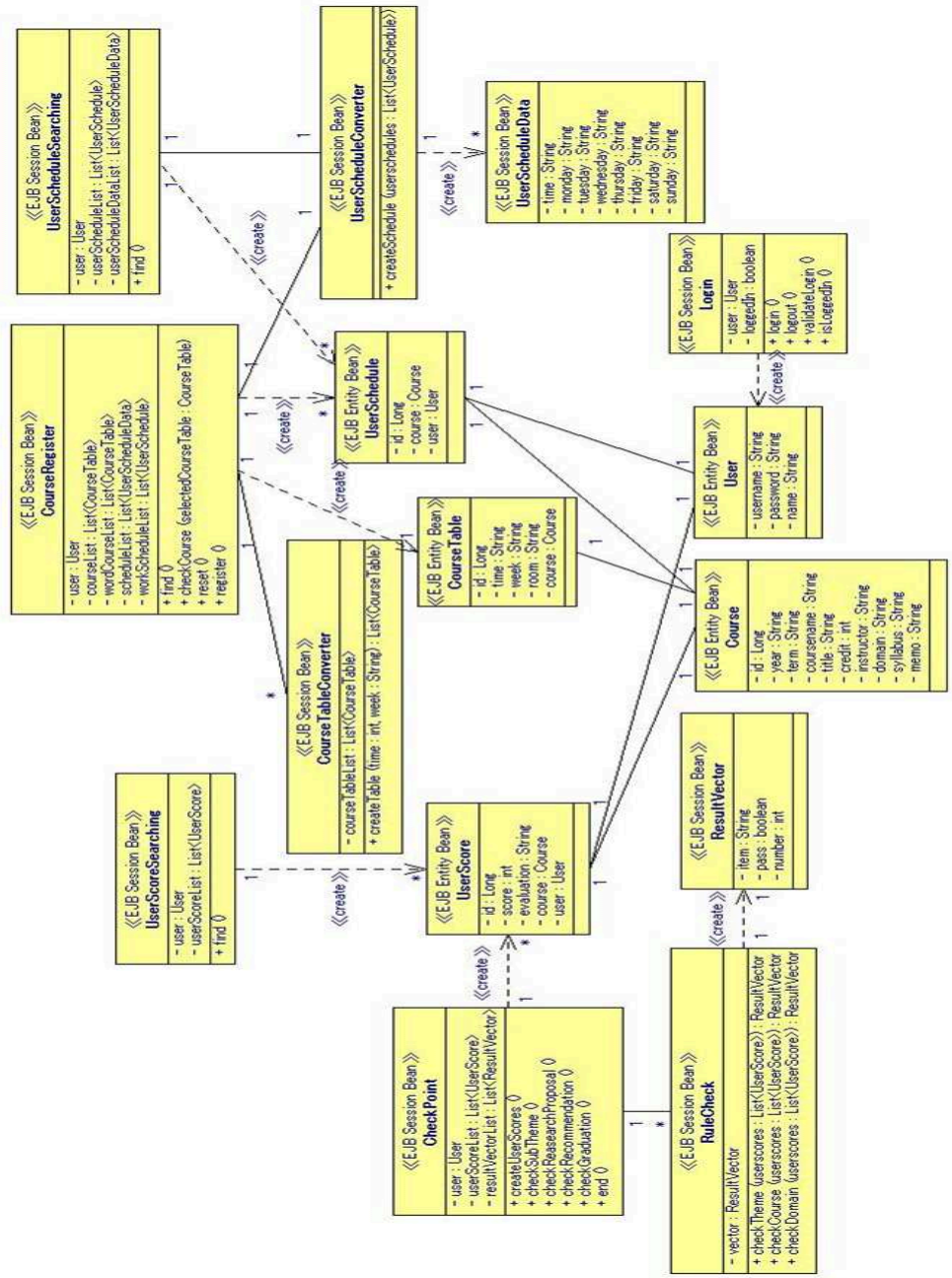

図 5.15: 履修管理システムのクラス図

5.6.2 履修管理システムにおけるアカウンタビリティ機能の実現法

前節で述べた履修管理システムにタイプ３のアカウンタビリティを付加する方法について考察する．

アカウンタビリティ機能と実現アプローチ

図 5.11 に示した参照アーキテクチャにおいて，アカウンタビリティ機能はシステムの実行履歴 DB と社会規則 DB を利用して実現する．ここで，実行履歴の取得方法には大きく２種類考えられ，それぞれ取得できる実行履歴内容が異なるため実現できるアカウンタビリティ機能も異なる．

１つ目の方法は，ユーザインタフェース層とプロセス管理層との間に配置したインターセプタプロキシからのみ実行履歴を取得するものである．インターセプタプロキシは，これらの層の間で行われる機能呼び出しやデータの受け渡しのすべてを実行履歴として取得することができ，実行履歴 DB に格納できる．この方法の場合，既存システムのプロセス管理層には一切変更を加えずにアカウンタビリティ機能を付加できるが，既存システムのプロセス管理層の内部で行われている処理の実行履歴を取得することができない．このため，使用できる実行履歴は層の間のインタフェース，つまりプロセス管理層の機能呼び出し名と引数およびその返り値のみであり，実現できるアカウンタビリティ機能は限られたものとなる．

２つ目の方法は，インターセプタプロキシで取得できる実行履歴に加えて，既存システムのプロセス管理層の内部で行われる処理の実行履歴も取得するものである．例えば，既存システムが Java で実装されている場合，AOP 技術を適用してロギングコードを織り込むことにより既存のソースコードを変更することなくシステムの内で行われている処理の実行履歴を取得可能である．また，通常デバッガにより使用される JDI (Java Debug Interface) を利用すると，既存システムが動作している Java 仮想機械から詳細な実行履歴を取得できる．この方法の場合，実行履歴の内容を意味のあるものにするためには既存システムのソースコードを理解する必要があるが，システムの動作に関する詳細な実行履歴（ビジネスロジックの実行順序や実行パス，変数の値の変化など）を取得できるため，前者の方法より高度なアカウンタビリティ機能を実現できると考えられる．

ここでは，前者の方法によるアカウンタビリティ機能の実現法を対象とする．具体的に，履修管理システムにおいて実現するアカウンタビリティ機能は以下のとおりである．

図 5.14 の「チェックポイント通過条件を検査する」画面では，検査したいチェックポイントをクリックすると，そのチェックポイントの条件を学生が充足しているかどうかに関するシステムによる判断結果が表示される．その判断結果に対し学

生が納得できない場合，システムはアカウンタビリティ機能により下記に示すシステムの判断根拠を表示して学生に説明を行う．

- チェックポイントに関連する履修規則の制定目的
- チェックポイントに関連する履修規則
- 学生の修得状況
- 履修規則に修得状況を適用し，結論を導く

参照アーキテクチャと Java EE アーキテクチャとの対応

図 5.11 に示した参照アーキテクチャは 3 層モデルに基づいており，Java EE アーキテクチャも 3 層モデルに基づいているため対応づけることができる．図 5.16 にアカウンタビリティを実現する Java EE アーキテクチャを示す．

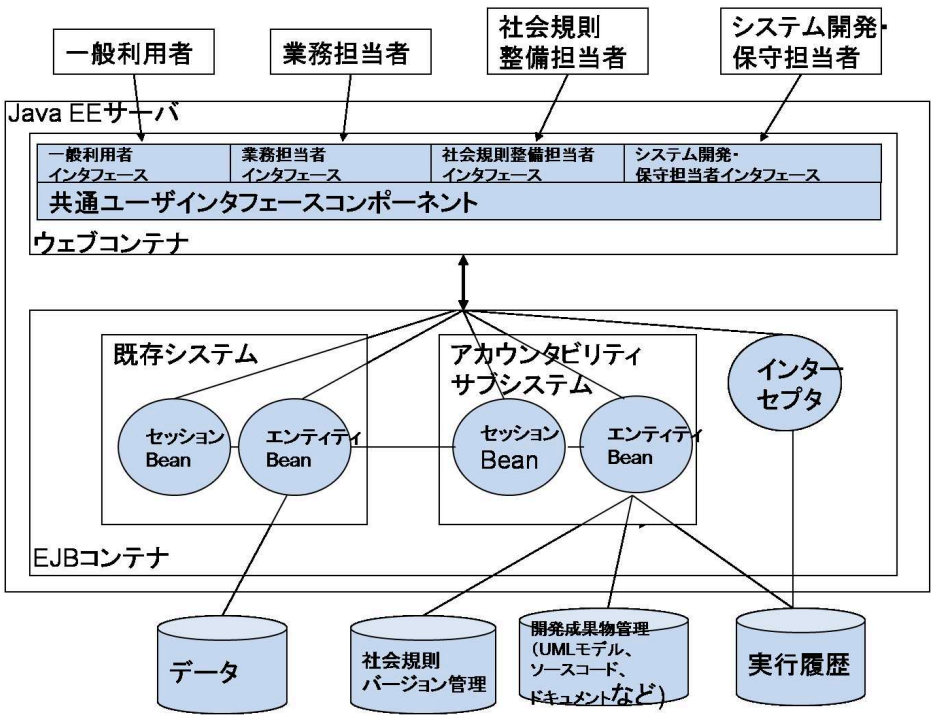

図 5.16: アカウンタビリティを実現する Java EE アーキテクチャ

Java EE はインターセプタ機構を備えているため，インターセプタプロキシは必要なく，実行履歴を取得する機能を持つインターセプタコンポーネントがあればよい．また，Java EE ではユーザインタフェース層からのアカウンタビリティ機能要求とプロセス管理層のアカウンタビリティサブシステムの Session Bean の呼び出しを対にして設定する．

JBoss Seam によるアカウンタビリティ機能の実現法

Java EE プラットフォーム対応のウェブアプリケーションフレームワークである JBoss Seam を用いた，履修管理システムにおけるアカウンタビリティ機能の実現法について述べる．

ユーザインタフェース　アカウンタビリティ機能の呼び出しとその結果である説明の表示を行う必要があるため，履修管理システムのユーザインタフェースの一部に変更を加える．アカウンタビリティ機能のユーザインタフェースはどのようなものが良いのかさまざまな角度から検討する必要があるが，ここでは，システムの実行結果を表示する画面においてアカウンタビリティボタンを追加し，利用者がそのボタンをクリックするとアカウンタビリティ機能が呼び出され，その結果である説明がポップアップされたウィンドウに表示されるという方式を用いる．

具体的には，図 5.14 に示した画面遷移図の「チェックポイント通過条件を検査する」画面内の検査結果表示部にアカウンタビリティ機能を呼び出す「Why?」ボタンを追加する（図 5.17）．実装としては，検査結果表示部のビューを定義している xhtml ファイルに数行のコードを追加するだけでよい．そのコードにはボタンが押されたときに呼び出す Session Bean 名とそのメソッド名を記述する．学生が「Why?」ボタンをクリックするとウィンドウがポップアップし，その中にチェックポイントの通過可否に関する説明が表示される．

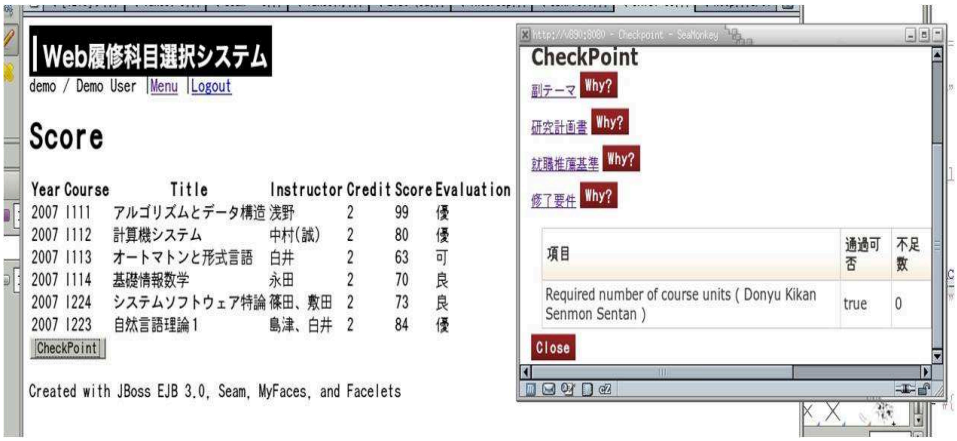

図 5.17: アカウンタビリティ機能に対応するユーザインタフェース

インターセプタ　JBoss Seam はインターセプタ機構をサポートしており，これを利用して実行履歴を取得する．具体的には，まず，EJB 配備記述ファイル ejb-jar.xml ファイルにインターセプタを動作させる対象コンポーネントととして，ユーザインタフェース層から呼び出されるコンポーネントすべてに設定する．下記に示すインターセプタクラスを定義し，前処理や後処理において呼び出し先のコンポーネント名，メソッド名，引数の値，メソッドの返り値を実行履歴 DB に書き出す処理を行う．

```
@Interceptor(type=InterceptorType.CLIENT)
public class LoggingInterceptor {
    @AroundInvoke
    public Object logging(InvocationContext invocation)
            throws Exception {
        // 前処理
        ...
        Object result = invocation.proceed();
        // 後処理
        ...
    }
}
```

アカウンタビリティサブシステム　アカウンタビリティ機能を提供するアカウンタビリティサブシステムも EJB コンポーネントで実現する．「Why?」ボタンごとに，その説明を行うロジックを実装したアカウンタビリティサブシステムの Session

Bean を対応させるアプローチをとる.

具体的には，学生が「副テーマ開始要件」チェックポイントの結果に疑問を持った場合,「Why?」ボタンをクリックするとアカウンタビリティサブシステムの「副テーマ開始要件」に関する説明ロジックを実装した Session Bean のメソッドが呼び出される.

このメソッドには，5.6.2 節で述べた，システムの判断根拠を説明するロジックが実装されている.「チェックポイントに関連する履修規則の制定目的」および「チェックポイントに関連する履修規則」は社会規則 DB から読み込み,「学生の修得状況」は既存の履修管理システムの使用している DB から読み込み,「履修規則に修得状況を適用し，結論を導く」は，読み込んだデータを基に学生にとってわかりやすく説明した文章を組み立て，それをユーザインタフェース層に返す.

なお，上記とは独立に，学生が自由に入力した質問文にインターセプタプロキシが収集した実行履歴を加えることにより，十分な内容を持つ質問を構成する方式も試みている．質問文と社会規則データベースの内容をもとに，ベクトル空間法による類似度の計算をおこない，質問文に対する回答に関連の深い法文を検索する実験を行った [48]．質問文と法文では暗黙的になっている「行為対象の主体」を補い,「行為対象の主体」と「文章の内容」に対して，それぞれ独立に類似度を求める方法がよいという結論を得ている.

社会規則データベース 5.3.4 節で述べたアカウンタビリティ木を関係データベースとして実現した．本学の履修規則に対するアカウンタビリティ木を定義し，それに基づき図 5.18 に示すスキーマを定義した [49].

以下に，図 5.18 の各テーブルについて説明する.

- 業務目標テーブル　アカウンタビリティ木におけるゴールとサブゴールを保持する．すなわち，図 5.8 における各ノードのその間の関係を保持する.

- 規則要求テーブル　アカウンタビリティ木の葉の情報を保持する．葉では，学生に対する指示が記述されており,「修了までに 4 分野を修得」,「修了までに副テーマを終了」などの例がある.

- 義務規範テーブル　履修規則のうち，義務規範に属するものを保持する．どのカテゴリの人間に，どのような行為が指令されているかを表現する．主体，義務様相，行為内容の属性を持つ.

- 性質決定規範テーブル　履修規則のうち，性質決定規範に属するものを保持する．カテゴリ名，カテゴリ内容の属性を持つ．たとえば,「副テーマ教員」

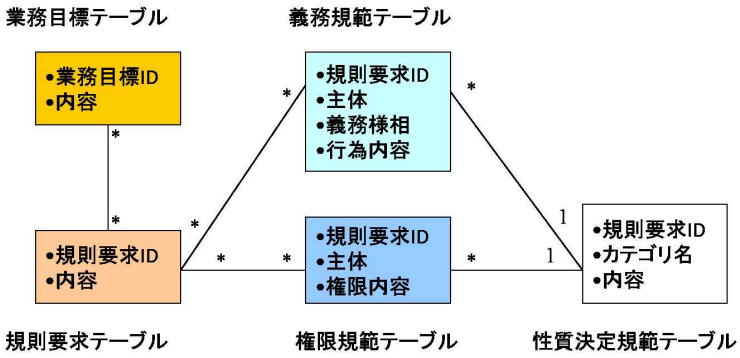

図 5.18: アカウンタビリティ木の関係データベースによる実現

は，性質決定規範テーブルの「カテゴリ名：副テーマ教員，カテゴリ内容：副テーマ開始時に，各学生に研究科によって定められた教員」と定義される．

- 権限規範テーブル　履修規則のうち，権限規範に属するものを保持する．主体，権限内容の属性を持つ．

社会規則の改定をサポートする社会規則 DB のバージョン管理機能に関しては今後の課題である．

5.6.3　アカウンタビリティ機能の実行例

画面遷移図（図 5.14）の「チェックポイント画面」では,「チェックポイント通過条件をチェックする」ユースケースを実装している．学生がチェックポイント項目をクリックすると，チェック結果が表示される．このチェック結果に対するアカウンタビリティ機能の呼び出しを，各チェックポイント項目に対応付けられた「Why?」ボタンによって行う．図 5.19 にアカウンタビリティ機能の実行例を示す．

図 5.19 において,「チェックポイント画面」では，チェックポイント項目のうち副テーマを選択して，チェック結果が表示されている．図中の副テーマのリンクの右に配置されている「Why?」ボタンをクリックすると，このチェック結果に対して質問することができる．

この「Why?」ボタンをクリックすると新たなウィンドウがポップアップしてそこにアカウンタビリティ機能の実行結果である説明（副テーマに関する規則文および規則の制定目的，学生の修得状況，説明文）が表形式で提示される．副テーマに関する規則文および規則の制定目的は社会規則 DB から取り出す．学生の修

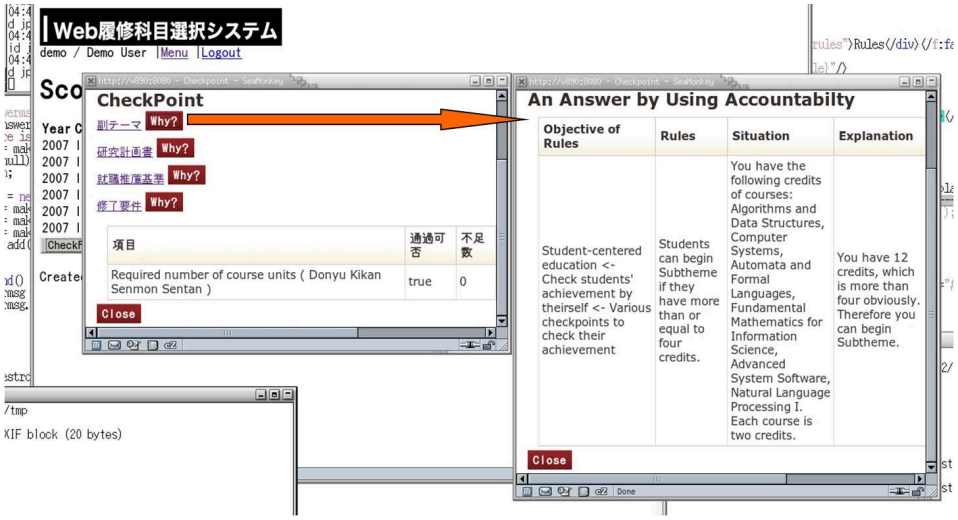

図 5.19: アカウンタビリティ機能の実行例

得状況は実行履歴 DB から取り出す．アカウンタビリティサブシステムにおいて，これらの情報を組み合わせて学生にとって分かりやすい説明文を生成する．

プロトタイプ実装のため，この実行例では表形式で提示しているが，学生にとって分かりやすいように提示する形式を工夫する必要がある．学生は，この説明を読むことにより，副テーマに関するチェックポイントの通過可否の理由を知ることができ，システムによるチェック結果について納得できるようになると期待できる．

5.7 今後の課題

本章では，情報システムにソフトウェアアカウンタビリティ機能を付与する手段について，法理論に基づき，ゴール指向要求定義法の概念を利用して，定義した．また，本学の定める履修規則に則った履修管理システムを対象にアカウンタビリティ機能を付加する方法について，参照アーキテクチャと Java EE アーキテクチャとの対応付けを行い，ウェブアプリケーションフレームワークとして JBoss Seam を用いた場合の実現法を提案した．

現在，各部の理論の精密化を以下の点にわたって進めている．

- 社会規則の改定を支援するための，ゴール指向木の編成法
- インターセプタプロキシにより収集される履歴情報の精密化と収集効率の

向上

また，アカウンタビリティを付加した履修管理システムをより実践的なものにする必要がある．

参考文献

[1] 江尻 暁, 北田安希雄, 島津 明. 法令文の論理式への変換- 論理構造について-. 言語処理学会第 12 回年次大会, pp.624-627, 2006.

[2] Barbara Grosz, Norman Haas, Gary Hendrix, Jerry Hobbs, Paul Martin, Robert Moore, Jane Robinson, and Stanley Rosenschein. DIALOGIC: A Core Natural-Language Processing System. *Proceedings of Computational Linguistics* (Coling 82), pp.95-100, 1982.

[3] 林 修三, 法令作成の常識. 日本評論社, 2003.

[4] 池原 悟, 宮崎正弘, 白井 諭, 横尾昭男, 中岩浩巳, 小倉健太郎, 大山芳史, 林 良彦. 日本語語彙大系. 岩波書店, 1997.

[5] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2000.

[6] 片山卓也. 検証進化可能電子社会 -情報科学による安心な電子社会の実現-. 情報処理, vol.46, No.5, pp.515-521, 2005.

[7] Takuya Katayama. Legal Engineering – An Engineering Approach to Laws in e-Society Age –. *Proceedings of the First International Workshop on Jurisinformatics* (JURISIN 2007), pp.1-5, 2007.

[8] 北田安希雄, 江尻 暁, 島津 明. 法令文の論理式への変換-原始文について-. 言語処理学会第 12 回年次大会, pp.628-631, 2006.

[9] 黒橋禎夫. 日本語構文解析システム KNP version 2.0 b6 使用説明書, 京都大学大学院 情報学研究科, 1998.

[10] 黒橋禎夫, 河原大輔. 日本語形態素解析システム JUMAN version 4.0 使用説明書. 東京大学大学院情報理工学系研究科, 2003.

[11] 前田正道. ワークブック 法制執務. ぎょうせい, 2006.

[12] Minh Le Nguyen, Akira Shimazu and Xuan-Hieu Phan. Semantic Parsing with Structured SVM ensemble Classification Models. *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (ACL-Coling 2006), pp.619-626, 2006.

[13] Makoto Nakamura, Shunsuke Nobuoka, and Akira Shimazu. Towards Translation of Legal Sentences into Logical Forms. *Proceedings of the First International Workshop on Juris-informatics* (JURISIN 2007), pp.6-17, 2007.

[14] 信岡俊祐, 中村 誠, 島津 明. 法令文の論理式への変換. 言語処理学会第 13 回年次大会発表論文集, PD1-5, pp.254-257, 2007.3.20.

[15] 岡田光弘. 法律知識の論理による表現. 法律人工知能, 創成社, pp.169-181, 2000.

[16] Terence Parsons. *Events in the Semantics of English*: *A Study in Subatomic Semantics*. MIT Press, 1990.

[17] Stuart Russell and Peter Norvig. *Artificial Intelligence*: *A Modern Approach*, 2nd Edition. Prentice Hall, 2003.

[18] Akira Shimazu, Shozo Naito, and Hirosato Nomura. Semantic Structure Analysis of Japanese Noun Phrases with Adnominal Particles, *Proceedings of the 25th Annual Meeting of Computational Linguistics* (ACL 1987), pp.123-130, 1987.

[19] 田中規久雄, 川添一郎, 成田一. 法律条文の標準構造 -自然言語による法知識処理をめざして-. 情報処理学会研究報告第 93 巻 97 号, pp.79-86, 1993.

[20] 田中規久雄, 法律効果規定部の意味機能について, 情報処理学会研究報告 98 巻 21 号, pp.1-8, 1998.

[21] 田中穂積 (監修): 自然言語処理 - 基礎と応用 -, 電子情報通信学会, 1999.

[22] 山田大介, 島津 明. 法令文の言語的特徴を利用した可読性向上のための表示. 言語処理学会第 12 回年次大会, pp.196-199, 2006.

[23] 山本庸幸. 実務立法技術. 商事法務, 2006.

[24] 吉野 一．法律知識ベースの構築とは．法律人工知能，創成社，pp.146-168, 2000.

[25] Alessandro Zucchi. *The Language of Propositions and Events*. Studies in Linguistics and Philosophy. Kluwer, 1993.

[26] S. Hagiwara and S. Tojo. Discordance Detection in Regional Ordinance: Ontology-based Validation, JURIX 2006.

[27] N. N Tun and S. Tojo IC-based Ontology Expansion in Devouring Accessibility. Australian Ontology Workshop (AOW), 2005.

[28] N. N. Tun and S. Tojo. Semantic-enrichment in Ontologies for Matching, International Journal of Semantic Web and Information Systems (IJSWIS), vol.2, No.4 (2006).

[29] S. Tojo and K. Satoh. Occam's Razor by Minimal Negation. Jurisin 2007.

[30] 二木厚吉，緒方和博，中村正樹： CafeOBJ 入門 (1)-(6)，コンピュータソフトウェア (投稿中).

[31] Yasuhito Arimoto, Dines Bjorner, and Kokichi Futatsugi: Formal Domain Description and Design of License Language, submitted for publication.

[32] Dines Bjorner: "Software Engineering, Vol 3: Domains, Requirements and Software Design". Texts in Theoretical Computer Science, the EATCS Series. Springer, 2006.

[33] CafeOBJ Home Page: http://www.ldl.jaist.ac.jp/cafeobj/

[34] Xiaoyi Chen, Jianwen Xiang, Dines Bjornor, Kokichi Futatsugi: A License Language Design and Analysis for E-Government, accepted for 2007 International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM2007), Shanghai, China, 21-23 September 2007.

[35] Xiaoyi Chen, Jianwen Xiang, Weiqiang Kong, Kokichi Futatsugi: Formalization and Analysis of Public Administration Domain with the OTS/CafeOBJ Method, accepted for 3rd International Conference on e-Government (ICEG2007), Montreal, Canada, 27-28 September 2007.

[36] Kokichi Futatsugi: Verifying Specifications with Proof Scores in CafeOBJ, Proc. of 21st IEEE International Conference on Automated Software Engineering (ASE'06), pp. 3-10, 2006. (an invited keynote talk)

[37] Kazuhiro Ogata and Kokichi Futatsugi: Rewriting-Based Verification of Authentication Protocols, Proc. of 4th International Workshop on Rewriting Logic and its Applications (WRLA 2002), ENTCS 71, Elsevier, 2002.

[38] Jiansen Xiang, Dines Bjorner, and Kokichi Futatsugi: Formal Digital License Language with OTS/CafeOBJ method, submitted for publication.

[39] Hassan Gomaa. *Designing Concurrent, Distributed, and Real-Time Applications with UML*. object technology series. Addison-Wesley, 2000.

[40] JBoss.org. JBoss Seam. http://labs.jboss.com/jbossseam/.

[41] John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, Vol. 42, No. 1, pp. 31–37, 1999.

[42] Douglas C. Schmidt, Michael Stal, Hans Rohnert and Frank Buschmann. *Pattern-Oriented Software Architecture Volume 2 – Networked and Concurrent Objects"*. John Wiley and Sons, 2000.

[43] George Schussel. Client/Server: Past, Present and Future, 1995. online.

[44] T. エックホフ, N.K. ズンドビー. 法システム:法理論へのアプローチ. ミネルヴァ書房, 1997. 都築廣巳 [ほか] 訳.

[45] 落水 浩一郎. ソフトウェアアカウンタビリティに関する基礎考察. 信学技報 SS2006-33, pp. 49–54. 電子情報通信学会ソフトウェアサイエンス研究会, 2006.

[46] 自由国民社（編）. 現代用語の基礎知識. 自由国民社, 2006 年度版, 2006.

[47] 山本 修一郎. 要求を可視化するための要求定義・要求仕様書の作り方, ソフトリサーチセンター, 2006.

[48] 北山 真太郎. ベクトル空間法に基づく法文書検索のための基礎実験. 北陸先端科学技術大学院大学 情報科学研究科 副テーマ レポート, 2007.

[49] 杉森 隼人, 落水 浩一郎. ソフトウェアアカウンタビリティ実現のための GORA と法理論の利用に関する報告. Technical Report IS-RR-2007-005, 北陸先端科学技術大学院大学, 2007.

[50] 早坂 良, 堀 雅和, 藤枝 和宏, 落水 浩一郎. アカウンタビリティおよび進化容易性を持つソフトウェアアーキテクチャと 3 層モデルとの対応. 情処研報 2005-SE-150, pp 1–8. 情報処理学会ソフトウェア工学研究会, 2005.

[51] 早坂 良, 落水 浩一郎. 履修管理システムにおけるオントロジを用いたアカウンタビリティ設計手法. 情処研報 2006-SE-151, pp. 73–80. 情報処理学会ソフトウェア工学研究会, 2006.

[52] 早坂 良, 秋山 裕俊, 杉森 隼人, 北山 真太郎, 鈴木 正人, 落水 浩一郎. 履修管理システムにおけるソフトウェアアカウンタビリティ機能の実現法. 信学技報 SS2007-14, pp. 29–34. 電子情報通信学会ソフトウェアサイエンス研究会, 2007.

JAIST Press

Publisher: JAIST Press
Nomi, Ishikawa 923-1292, Japan
Tel: +81-761-1980, FAX: +81-761-1199
E-mail: jaistpress@jaist.ac.jp
URL: http://www.jaist.ac.jp/library/press

ISBN4-903092-04-6