修 士 論 文

不完全知識下での類推解釈を支援する法的推論システムの研究

指導教官　國藤 進 教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

森沢 浩造

1998 年 2 月 13 日

目 次

第 1 章

序論

1.1　研究の背景

論理法学では、法的推論を法的正当化の推論と法的発見の推論に区別して分析する。法的正当化の推論は、目的とする結論を法令文と事実を用いて導いていく推論プロセスであり、法的発見の推論は、目的とする結論から諸前提を発見する推論プロセスである。最終的な法的判断を下すときには、必要に応じてこれらの推論プロセスを用いるというのが論理法学における法的推論の構造である。判決は、法令文と事実だけでは直接に演繹できず、法令文の解釈によって追加された法令文を用いることによってはじめて演繹できるものであるとされているからである。

知識処理型のシステムにおける代表的な推論方式は、基本的推論と拡張的推論に分類することができる。論理法学における法的推論モデルにおいて、法的正当化の推論は基本的推論である演繹推論を用いて実現することができるが、法的発見の推論は、不完全知識を扱う必要があることから演繹推論だけでは実現できず、拡張的推論を用いる必要がある。拡張的推論に分類される代表的な推論方式には帰納推論、アブダクション、類推があり、現在、これらの推論方式を基にして法的発見の推論を実現するための研究が、理論とシステム実装の両側面でなされている。

1.2　研究の目的

本研究の最終目的は、法的推論を実現するために必要な法的発見の推論を推論機構として実現することにあり、そのために必要となることは拡張的推論機構を実現することである。法的発見の推論を実現するための方法論についての研究は、知識獲得や類推などの方法を用いて既に行われている [4, 30, 35, 36, 10] が、知識の不完全性の問題に対処しきれていないという問題点があった。そこで本研究では、法的発見の推論を実現するのに障害となる知識の不完全性に着目し、不完全知識を補完するための方法論を展開し、そのための機構の実現を試みることにより、方法論の有用性とその問題点を検証する。

1.3　研究の概要

本研究では、法的推論機構を実現するために既存の理論を利用する。それらの理論はそれぞれ推論、知識獲得、知識表現の方法論として用いる。

推論を実現するための理論的基盤としてアブダクション [40] を用いる。推論機構の実装にはアブダクティブ論理プログラミング (Abductive Logic Programming,ALP) の処理系を用いる。アブダクティブ論理プログラミングは、論理プログラミングをアブダクションに拡張した拡張論理プログラミングであり、Kowalski らによってその法的推論への適用可能性が指摘されている [37] 。

知識の獲得は不完全知識の補完を行うために必要である。そのための方法として、類推を実現するために最近よく用いられている抽象化による類推モデル [29] を用いる。知識の獲得は、このモデルを用いて知識の階層 (節階層) を生成することにより行う。

知識表現には順序ソート論理 (Order-sorted Logic, OSL)[44] の表現を用いる。順序ソート論理は、通常の一階述語論理の表現における項にソートを付けたソート付き論理の一種であり、ソート集合が部分的に半順序関係を形成する論理である。

本研究では、これらの理論を用いて法的推論機構を作成し、不完全知識の獲得 (節階層集合の生成) とその補完 (類推解釈) について実験を行う。法令文データベースとして国際物品売買法に関する国際条約 (CISG) の 2,3 部を用い、事実知識として、CISG における実験用の設例を用いている。そして、これらの実験の結果から、実現できたこととできなかったことを分析し、システムの有用性を評価する。

1.4　本論文の構成

本論文は、本章も含めて 6 章編成になっている。本章は 1 章であり、2 章では論理法学における法的推論モデルについて述べる。3 章では本研究で用いる理論のアブダクション、抽象化による類推、順序ソート論理について述べる。4 章では本研究において作成した法的推論システムの概要とその方法論について述べる。5 章では本研究のシステムを用いて行った実験について述べ、その結果を考察する。最後に、6 章で本研究の概要をまとめ、今後の課題を示す。

第 2 章

論理法学における法的推論モデル

本研究が中心に扱うのは、法的推論における法的発見の推論である。論理法学 (Logical Jurisprudence) では、法的推論を法的正当化の推論と法的発見の推論とに区別して分析を行う。本研究における法的発見の推論とは、論理法学上の推論形式の区別に基づくものである。

法的推論における法的発見の推論の位置づけを示すことを目的として、論理法学の基本概念、法的推論の構造について以下に示す。

2.1 論理法学における基本概念

論理法学における基本概念は以下に示す 3 つである。論理法学における課題の 1 つは法文の論理構造を解明することにあるが、その説明のためにこれらの基本概念を用いる。

- 法文

  ルール文 (例) $p(X) \rightarrow q(X)$: (Xが)p であるならば、(Xは)qである。
  ファクト文 (例) $p(a)$: a は p である

- 推論規則

  「A ならば Bである」が真で、A が真であるなら、Bも真であることが帰結される

- 真理値

  真 (true) か偽 (false)
  有効 (valid) か無効 (invalid)

2.2　法的推論の構造

論理法学では、法的推論を法的正当化の推論と法的発見の推論に区別して分析する。古典的法的三段論法では、図 2.1 に示すように、法規と事実を用いて法的判断を導く構造をしていた。そこで用いられている推論は、論理法学の立場から見ると、法的正当化の推論のみである。

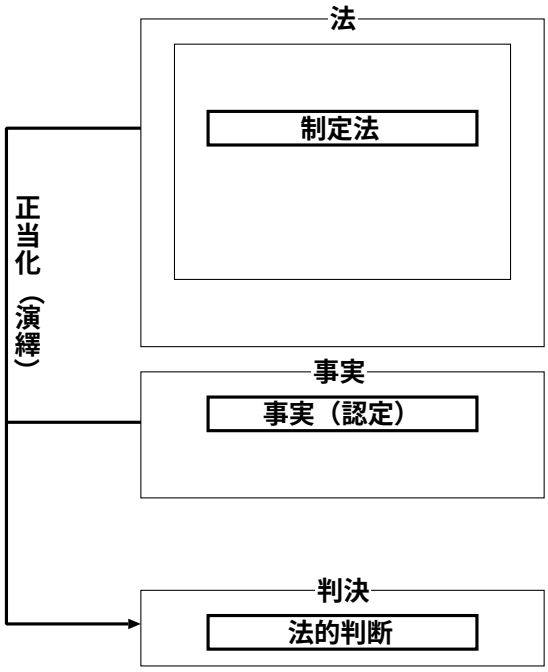


図 2.1: 古典的法的三段論法

この表現形式で推論を行う場合、法的判断を導くために法規と事実が同一レベルで記述されていなければならない。しかし、実際の事例における事実はより具体的であることが多く、この表現形式における事実は、それらの具体的事実がもとになって、あるいは法解釈を行うことにより導かれるものである。古典的法的三段論法における表現形式は、こうした事実を適切に表現していないという点で不十分であるといえる。

こうした問題点を踏まえ、法的推論の構造を改良したのが図 2.2 である。古典的法的三段論法と大きく異なるのは、知識を発見するプロセスが新たに加わっていることである。これが法的発見の推論であり、正当化の推論の際に用いられる前提そのものの発見や選択を行う推論である。図 2.2 に示されているように、法的判断は、法規の解釈などにより諸

法文を定立し追加することによって、法規や事実や付加された法文から演繹されることとして提示することができる。

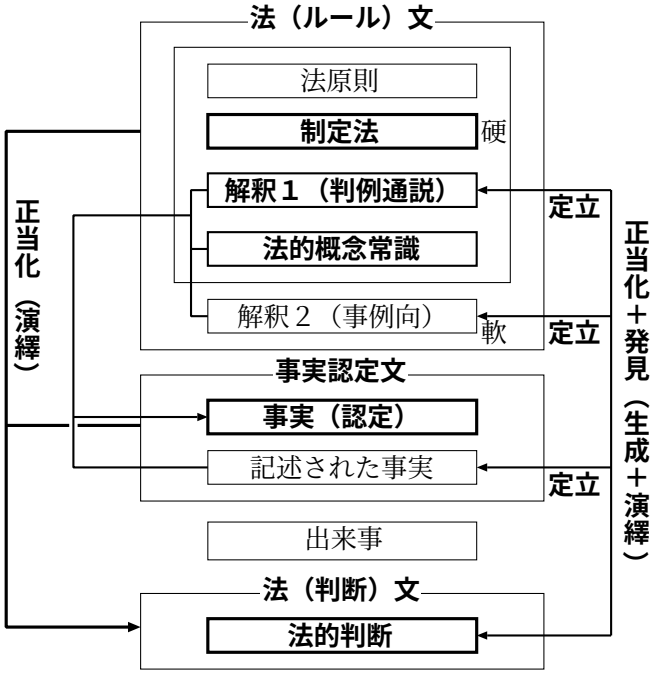


図 2.2: 正しく理解された法的推論の構造

第 3 章

本研究で用いる理論

本研究では、拡張的推論を実現するための推論方式としてアブダクションを用い、類推解釈を実現するために必要な知識の獲得を行う際の知識階層の生成方式として抽象化による類推モデルを用い、法令文データベースと判例知識を表現するための知識表現として順序ソート論理 (Order-sorted Logic) を用いる。本章ではこれらについて説明する。

3.1 アブダクションと ALP

アブダクション (Abduction) という用語を最初に用いたのは Peirce[40] であると言われているが、その概念はアリストテレスが著書 *Prior Analytics* において三段論法形式で示したものが起源であるとしている。しかし、アブダクションを「説明を求める推論」と定義し、演繹 (Deduction) や帰納 (Induction) と明確に区別したのは Peirce 自身である。

Peirce のアブダクションの論理は以下のように与えられる。

1. ある驚くべき事実として、Cが観察されている。
2. しかし、もし A が真であれば、Cであることは当然の事柄である。
3. それゆえ、A は真でないかと考える理由が存在する。

Perice によると、アブダクションは仮説あるいは説明の生成過程のことを指していて、仮説の生成、選択、利用を含めた総合的な問題解決のことである仮説推論とは少し意味が異なる。

アブダクションを帰納や演繹と区別したのは Peirce が最初であると述べたが、アブダクションと演繹との違いは、アブダクションが伝統的論理学における「後件肯定の虚偽」と呼ばれる妥当でない推論であるところからきている。また、アブダクションの帰納との違いは、帰納が観測された事例からそれらが属するクラス全体について一般化を行う推論であり、「事例の中に見出される規則性と同様の現象の存在を推論する」のに対し、アブダクションは観測事象とは違う種類の原因を追求し、「事例の中からは直接的に観測できない現象の可能性についても推論を行う」ことにある。

また、Kakas ら [34] は、Peirce のアブダクションで明らかにされていなかった無矛盾性の条件を追加した定式化を行っている。その定式化は次のようであり、無矛盾性の条件にあたるのは式 (3) である。

理論を T、観測された事実を G、G の説明をΔとすると、

(1) $T \not\models G$

(2) $T \cup \Delta \models G$

(3) $T \cup \Delta$ は無矛盾である。

アブダクションを実現した時の一般的な特徴は、説明Δが複数生成されることである。複数の説明からもっともらしい説明を選択するための方法にはさまざまなものがあるが、その 1 つに統合性制約 (Integrity Constraints) がある。統合性制約の目的は、意図しない知識ベースの更新を防ぐことにあり、知識ベースが常に統合性制約を満たすように更新を制御する。

統合性制約を Iとすると、Kakas らの定式化による無矛盾性の条件 (3) は

(3') $T \cup \Delta$ が I を満足する

と言い替えることができる。理論を T、可能な説明の集合 A、統合性制約 I として、アブダクションの枠組は 3 つ組$< T, A, I >$で表すことができる。

アブダクティブ論理プログラミング (Abductive Logic Programming, ALP) は、通常の論理プログラミングに加えて、アブダクションを扱えるようにした拡張論理プログラミングである。

アブダクションの論理プログラミングへの実装は、SLD 導出の拡張によって可能である。導出の過程で、あるサブゴールへの単一化に失敗した場合、証明を失敗とする代わり

に、サブゴールを仮説とみなして導出を続行することにより、アブダクションのメカニズムを実現する。そして、導出が終了した時には、単一化に失敗した節の集合が残ることになり、それがアブダクションにおけるゴールの説明に対応する。

ALP におけるアブダクションの枠組は、理論を T、観測された事実を G、可能な G の説明を A とし、その部分集合Δとして

(1') $T \nvdash G$

(2') $T \cup \Delta \vdash G$

(3') $T \cup \Delta$ が I を満足する

で表すことができる。

アブダクションの応用にはさまざまなものがある。故障診断や医療診断に代表される診断型のシステムは、観察結果からその原因を説明するプロセスを伴うが、アブダクションを用いることによりその実現が可能である。他の応用としては、知識同化があり、また、法的推論への適用可能性が Kowalski によって示唆されている [37] 。

3.2　抽象化による類推

類推とは、すでによく知っている事柄を、その事柄に類似したある知りたい事柄にあてはめる推論方式であり、人間が普通に行っている認知活動の 1 つである。専門的には、すでによく知っている事柄をベースドメイン (base domain) あるいは単にベースと呼び、知りたい事柄をターゲットドメイン (target domain) あるいは単にターゲットと呼ぶ。そして、あてはめることとは、ベースドメインの要素をターゲットドメインの要素に写像することである。

類推をモデルとして考えるときの重要なポイントは、ベースドメインの要素とターゲットドメインの要素の間に生ずる写像の対応付けの方法にある。類推の定義から、類推モデルにおけるベースドメインの要素とターゲットドメインの要素の間に起こる写像は、何らかの類似性を持っている必要がある。

Thagard らの提案した類推モデルである ARCS(Analogical Retrieval by Constraint Satisfaction)[46] では、対象レベル、関係レベル、プラグマティックなレベルの類似性を満たしていることが、ベースの検索の制約になるという仮定をもとにしている。ここで、

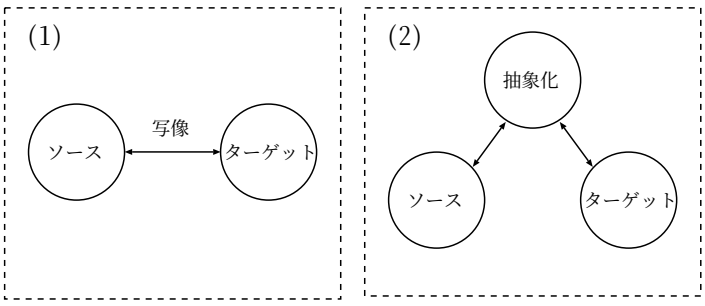
対象レベル、関係レベル、プラグマティックなレベルの類似性は、類推研究を行う認知科学者らが類推モデルを考える際に区別するべきであると言われているものである。同様の類推モデルとして、Holyoak らの提案した ACME(Analogical Constraint Mapping Engine)[33] がある。

また、Gentner の構造写像理論 (Structure mapping)[28] では、属性の非写像、構造的一貫性、システム性原理の 3 つの原理を満たす写像を適切な写像とすることによって、ベース検索の制約を実現している。

しかし、これらの理論に共通する問題点として、

1. ベースドメインの要素とターゲットドメインの要素が記憶の階層性を無視している。
2. 観点の設定に基づく意味、同一性を全く考慮していないため、類推が当てはめのようになっている。

というのが挙げられている。1. は、人間の経験の間には比較的明確な形での階層構造が存在するという Schank の主張 [42] によるものであり、2. は、人間は観点を設定しながら類推を行っているという実験結果に基づくものである。



(1) ソースからターゲットに直接写像する方式
(2) 抽象化知識を介する方式

図 3.1: 類推モデル

これらの問題点に対処するために、類推に関して新しい見方を試みたのが Greiner の**抽象化による類推モデル** [29] である。図 3.1 に示すように、従来のモデルでは、ベースとターゲットの間での写像を仮定していたのだが、抽象化による類推モデルでは、ベースとターゲットの間にカテゴリーと呼ばれる知識を仮定する。カテゴリーというのは、ベース

ドメインとターゲットドメインにおいて、特定の観点で同一視された知識のことを指している。また、特定の観点で同一視するプロセスのことをカテゴリー化という。

また、Greiner は論文 [29] において、抽象化による類推システムを生成検査法により実現し、さまざまなヒューリスティックスを導入することにより、効率的な問題解決が可能になることを示している。

また、ソース知識が不要になる抽象化による類推モデルとして、Plaisted のモデル [41] と Tenenberg のモデル [45] がある。これらのモデルは、ターゲット領域の抽象化により抽象的な領域を自ら形成する。

3.3　順序ソート論理 (Order-sorted Logic)

論理にソートの概念を用いたのは Herbrand[32] が最初である。しかしこの定式化の証明が間違っていることを指摘し、多ソート論理 (Many-sorted Logic) の完全性を証明したのは Schmidt[43] である。

この多ソート論理は、自動推論 (Automated Deduction) の分野でも着目されていた。自動推論の分野では、探索空間を縮小する方法を考えることは重要である。従来、自動推論の分野では、知識を表現するのに一階述語論理が用いられていた。しかし、通常の一階述語論理にはオブジェクトに関する制限がなく、不必要な探索を行ってしまうという問題点があった。そこで考えられたのが、知識表現に多ソート論理を用いることである。異なるクラスに属するオブジェクトをソートによって区別し、探索の際にその情報を利用することによって、探索空間の縮小を実現しようとしたのである。

順序ソート論理 (Order-sorted Logic) は多ソート論理の一種である。順序ソート論理の特徴は、ソート集合が部分的に半順序関係を持ち、下位ソート (subsort) の概念を含んでいることである[1]。

ソート論理に下位ソートの概念を付加したのは、自動推論の分野では Walther が最初であり [47] 、Cohn によりそれが拡張されている [25] 。

順序ソート論理における項は、アトムを c、ソートを s とすると、

$$c : s$$

[1] 論文の中には、下位ソートの概念を含むソート付き論理を多ソート論理としているものもあるが、本論文では、Schmidt-Schauß[44] の使い方に従い、順序ソート論理という表現を用いることにする。

で表すことができる。n 個の引数を持つ関数は、関数 f、引数のソート $s_1 \ldots, s_n$、関数の出力のソートを s とすると、

$$f : s_1, \ldots, s_n \rightarrow s$$

と表せ、関数 fそのもののソートは s である。関数もアトムも項となり得るため、項を t、ソート s とした時には関数もアトムも

$$t : s$$

で表すことができる。n 個の引数を持つ述語は、述語 p、引数のソート $s_1 \ldots, s_n$とすると、

$$p : s_1, \ldots, s_n$$

で表すことができる。また、順序ソート論理では、ソート間に部分的な上位・下位関係を定めている。ソート集合を $\mathcal{S}$とすると、それらは半順序集合 $(\mathcal{S}, \prec)$ を形成し、これをソート階層と呼ぶ。このソート階層は、単一化を行う際の型チェックにおいて意味を持つ。例えば、変数 Xが持つソートを s とすると、変数に代入できる項はソートが s もしくはその下位ソートに限られる。

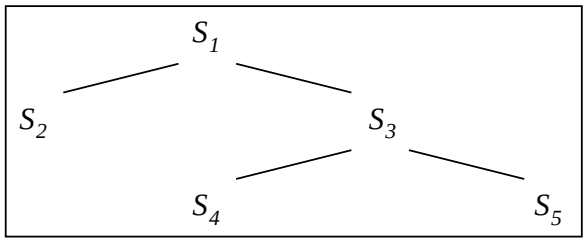


図 3.2: ソート階層の例

例として、図 3.2 のようなソート階層があったとすると、$X_1 : s_2$の場合、変数 X_1に代入できる項は項のソートが s_2である場合に限られ、$X_2 : s_3$の場合、変数 X_2に代入できる項は項のソートが s_3, s_4, s_5のどれかである場合に限られる。

第 4 章

不完全知識下での類推解釈を支援する法的推論システムの実現

4.1 従来の研究における問題点

法的推論に必要な法的発見の推論を実現するためには、演繹的推論を拡張する必要がある。そのための要素技術に関する研究は以前から行われていたが、これらの研究には、法的発見の推論を実現する上での問題点があった。以下に、その問題点を示す。

1. アブダクションのメカニズムを実現するアブダクティブ論理プログラミング (Abductive Logic Programming) の処理系の開発 [8] とその法的発見の推論への応用 [16, 7]

   (a) アブダクションによって生成された仮説が複数になる場合、複数の仮説の中から最も適切な仮説を選択することができない。

   (b) 完全に単一化できる知識の組しか導出の対象にしない。すなわち、類似知識であっても完全に単一化できなければ導出の対象にはしないので、仮説集合の要素の数が必要以上に大きくなってしまう。

2. 知識表現変更支援システムの構成に関する研究 [10]

   (a) 生成された知識の階層が不完全であり、法的発見の推論を適切に行なうことができない。

3. ゴールに依存した抽象化 (Goal-Dependent Abstraction, GDA) を用いた法的類推の実現 [4, 36, 39]

   (a) 抽象化を述語名あるいは項のソートに限っているため、類推できる範囲が制限されてしまう。

この中で、(1a) については、判例を用いた仮説選択機構の実現 [16] により、既に問題点の解決が試みられているが、その他の問題点については、解決を試みた例はまだない。

4.2　推論システムの実現

本研究では、3 章で示した理論をもとにして、推論システムとして、類推解釈を実現するために必要な知識階層を生成する**知識階層生成機構**と、類推解釈を実際に実現する**類推解釈機構**を作成した。これらの機構を、理論とのつながりを示した上で以下に説明する。

4.2.1　推論システムへの理論の適用方法

3 章で、本研究における推論システムを実現するための理論について述べたが、これらがどのようにして推論システムに適用されるかを以下に示す。

ALP の処理系による推論

本研究の推論システムにおいて、不完全知識下での推論を実現するために、論理プログラミングをアブダクションに拡張したアブダクティブ論理プログラミング (Abductive Logic Programming, ALP) の処理系を用いる。本研究で用いた ALP の処理系は、論文 [34] にある Kakas-Mancarella 手続きを用いてアブダクションを実現している。

論理プログラミングの基本になるのは事実、ルール、質問 (ゴール) であるが、本研究の推論システムにおいては、法令文知識をルールの形式で、事実知識 (事例) を事実の形式で記述する。そして法的問題を推論システムに解かせる時には、法的問題をゴールとして与えることになる。

ALP の処理系が通常の論理プログラミングの処理系と異なる点は、ゴールを満たすための知識が不足していてゴールが導けない場合も、不足した知識を仮説として出力することによって解を導く点にある。

例として、法令文が「B_1、B_2および B_3であれば、G である」、事実知識が「B_1であり、B_2である」とすると、論理プログラミングにおいて、これらは

$G \leftarrow B_1, B_2, B_3.$

$B_1.$

$B_2.$

といった表現で表される。ここで、法的問題を G として問題を解かせた時、ALP の処理系の場合、B_3を仮説として出力する[1]。

抽象化による類推モデルを用いた知識獲得と類推解釈

本研究では、類推解釈に必要な知識の獲得を行う際の知識階層の生成方式として、抽象化による類推モデルを用いる。本研究で用いるモデルは、Greiner の類推モデル [29] に基づいている。

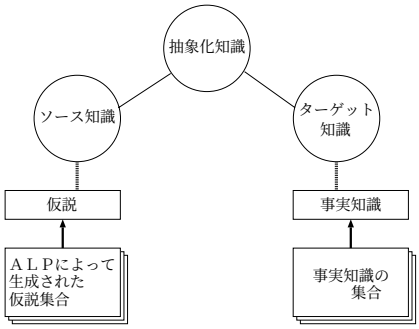


図 4.1: 本研究における類推モデルの利用方法

Greiner の類推モデルでは、ソース知識とターゲット知識の間に抽象化知識を生成するが、本研究では、図 4.1 に示すように、ソース知識として ALP の処理系で生成した仮説を、ターゲット知識として事実知識を用いる。これらの知識の選択はユーザが行い、抽象化知識の生成はアルゴリズムによってシステムが行う。

また、抽象化知識の生成によって階層を形成するが (本研究ではこれを節階層という)、類推解釈を実現する時は、抽象化知識 (本研究では、これを抽象化節と呼ぶことにする) を介してたどることのできる知識のみを類推解釈の対象とする。

[1]実際には、統合性制約 (Inregrity Constraints) を満たしているかどうかをチェックするプロセスがあり、本研究で用いた ALP の処理系においてもその手続きを行う。

順序ソート論理を用いた知識表現

また、法的知識の表現として順序ソート論理 (Order-sorted Logic) を用いる。順序ソート論理は、通常の一階述語論理にソートの概念を用いたソート論理の一つであり、ソート間に部分的に半順序関係を形成するためにこの名がある。本研究で用いる順序ソート論理は、項のアトムのみにソートの概念を用いる Walther のモデル [47] に基づいている。

例えば本システムでは、

> 「4 月 1 日、申込者 a が披申込者 b に農業機械を 10000 ドルで売る申込の手紙を出し、その通知が 4 月 8 日に届いた」

という事実知識を、順序ソート論理を用いて

> 申込 ((4 月 1 日:日時, 4 月 8 日:日時), (a:申込者, b:披申込者), [手紙:通信手段, 農業機械:商品, 10000 ドル:明確な値段, 1:数量], 申込:法律行為).

と表現している。

4.2.2　知識階層生成機構

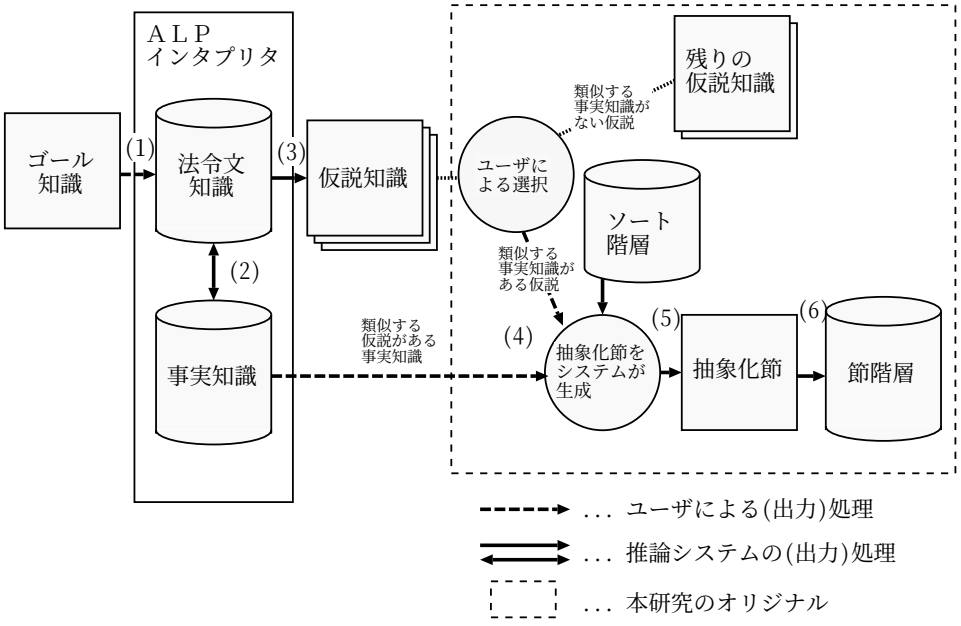


図 4.2: 知識階層生成機構

図 4.2 に示されているように、知識階層生成機構では、大きく分けて次の 3 つの処理を行う。

1. 法令文知識ベースと事実知識を ALP インタプリタ上に読み込ませ、ゴールを満たすために必要な知識 (これを仮説集合という) をシステムが生成する。

2. ユーザによって、関連のある仮説と事実の対を選択する。そして、選択された対から抽象化節を生成する。このとき、既存のソート階層を利用する。

3. 生成された知識を節階層に付加するために、節階層を再構成する。

この中で、2. における抽象化節の生成と、3. における節階層の再構成については、本研究独自のアルゴリズムを用いて行う。それを以下に示す。

ソート階層を用いた抽象化節の生成

本研究では、法的知識を順序ソート論理によって表現する。本研究で用いるシステムにおいては、階層をソート階層と節階層に分けて区別する。図 4.3 にその例を示す。

ソート階層: ソート階層は抽象化節の生成の際に利用されるが、このソート階層はあらかじめシステムに与えられている。

節階層: 節階層は、生成される抽象化節を含む節を単位とした知識の階層である。この節階層は、ユーザが選択する知識によって、また、節階層の再構成の仕方によって変動する。

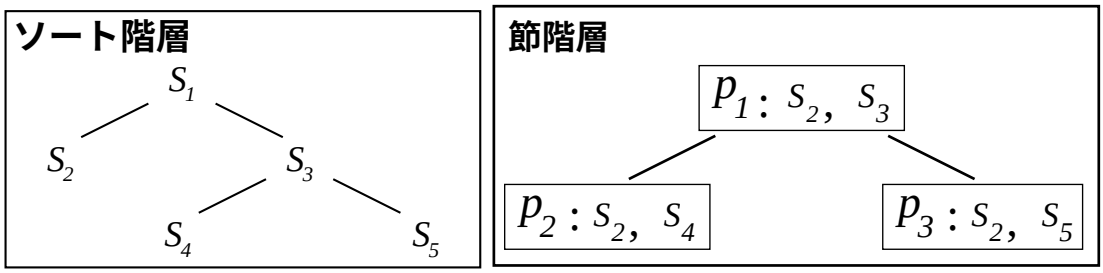


図 4.3: ソート階層と節階層

本研究における抽象化節の生成は、述語の抽象化と項に含まれるソートを同時に抽象化することによって行われる。ここで、本研究における「抽象化」とは、ソート階層あるいは節階層の上位ソートへの写像を行うことを指す。ここで、そのための関数として抽象化関数 φ を定義する。

ソート集合を $\mathcal{S}$とすると、本研究で用いる順序ソート論理において、$\mathcal{S}$は半順序集合 $\mathcal{S}_{osl} = (\mathcal{S}, \prec)$ を形成する。$s_1 \prec s_2 \in \mathcal{S}_{osl}$において、抽象化関数 φ を用いると、$\varphi(s_1) = s_2$ となる。

抽象化節の生成手順を以下に示す。

ALP の処理系によって生成された仮説集合を $\mathcal{H}$、事実集合を $\mathcal{F}$とする。そして、ユーザによって選択された仮説 $C_h(C_h \in \mathcal{H})$ の述語を p_h、述語が持つ n 引数のソートを ${s_h}^1, \ldots, {s_h}^n$、事実 $C_f(C_f \in \mathcal{F})$ の述語を p_f、述語が持つ n 引数のソートを ${s_f}^1, \ldots, {s_f}^n$とする。そして、C_hと C_fから生成される抽象化節を C_aとする。$1 \leq i \leq k$ $(k = \min(m, n))$ において、${s_h}^i$ と ${s_f}^i$に対応する C_aのソートを ${s_a}^i$とし、述語を p_aとする。

ここで、C_hと C_fにおける項のソートの抽象化規則は次のようになる。

アルゴリズム 1.　(ソートの抽象化規則)

1. ${s_h}^i = {s_f}^i$であるとき、${s_a}^i = {s_h}^i = {s_f}^i$

2. ${s_h}^i \neq {s_f}^i$で、$\varphi({s_h}^i) = \varphi({s_f}^i) = s$ であるとき、${s_a}^i = s$

3. ${s_h}^i \neq {s_f}^i$で、$\varphi({s_h}^i) = \varphi({s_f}^i) = s$ となるような s が存在しないとき、${s_a}^i = \phi$

ここで、規則 3. を適用した C_aの項は存在しないことになるが、これは、ある観点においては似ているが、別の観点においては似ていないという、「観点に着目した」抽象化による類推をモデル化したものである。

また、述語の抽象化規則は次のようになる。

アルゴリズム 2.　(述語の抽象化規則)

1. $p_h = p_f$の時、$p_a = p_h = p_f$

2. $p_h \neq p_f$の時、$p_a = p$　(p はユーザが指定する)

ソートの抽象化規則と述語の抽象化規則を適用することにより、抽象化節を生成した例を以下に示す。仮説 C_hと事実 C_fが述語とその引数の型

$$p_h : s_1, s_2, s_4$$

$$p_f : s_1, s_3, s_5$$

を持ち、ソート階層の順序関係として

$$s_4 \prec s_6, s_5 \prec s_6$$

が成り立つとすると、$\varphi(s_4) = \varphi(s_5) = s_6$であるから、抽象化節 C_aとして

$$p_a : s_1, s_6$$

が生成され、図 4.4 のように、異なる述語記号間で、項のソートを抽象化した形の節階層が生成される。

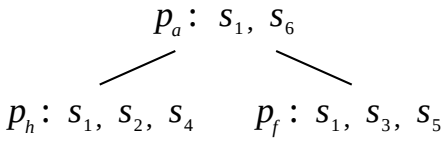


図 4.4: 生成される節階層

節階層の再構成

本研究のシステムでは、生成された節階層を既存の節階層に付加する際、節階層の再構成を行う。その方法について以下に示す。

アルゴリズム 3. (**節階層の再構成**)
仮説 C_hと事実 C_{f1}からその抽象化節 $C_a(= \varphi(C_h) = \varphi(C_{f1}))$[2]が生成されているとする。ここで、

(i) 仮説 C_hと事実 $C_{f2}(\neq C_{f1})$ から抽象化節 C_aが生成される場合、$C_{f1} \simeq C_{f2}$であるかどうかをユーザに質問する[3]。ここで、

1) $C_{f1} \simeq C_{f2}$であれば、C_h, C_{f1}, C_{f2}の抽象化節が全て C_aとなる。

[2]記号φ は、17 ページにおける**アルゴリズム 1.** で用いた抽象化関数と同じようなはたらきを持つ。ここではそれを節階層に適用している。

[3]ここで、$A \simeq B, A \not\simeq B$を、「$AB$間に関連がある」「$AB$間に関連がない」という意味で用いることにする。

2) $C_{f1} \not\simeq C_{f2}$であれば、C_hと C_{f2}から生成される抽象化節は $C_a{}'(\neq C_a)$ となる。

(ii) **(i)** における C_hと C_{f2}から抽象化節 $C_a{}'(\neq C_a)$ が生成される場合、C_hと C_{f2}から生成される抽象化節は $C_a{}'(\neq C_a)$ となる。

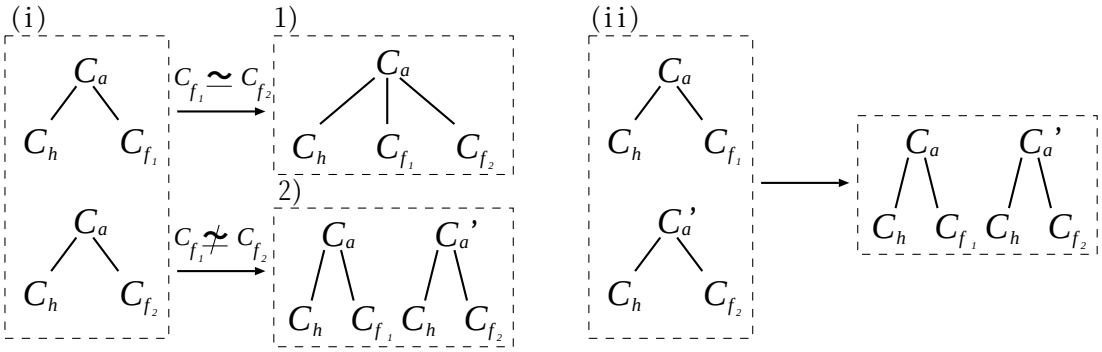


図 4.5: 生成される節階層の相互作用

4.2.3　類推解釈機構

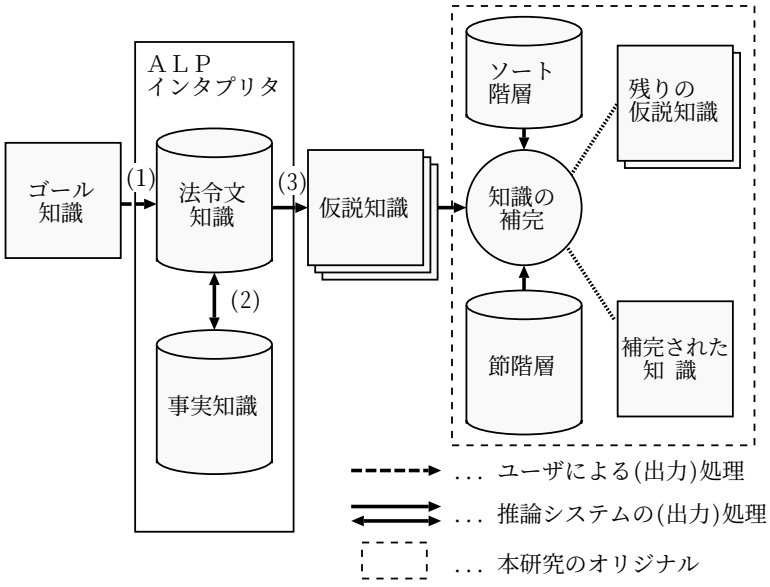


図 4.6: 類推解釈機構

図 4.6 に示されているように、類推解釈機構では、大きく分けて次の 2 つの処理を行う。この中で、1. については、知識階層生成機構と全く同じである。

1. 法令文知識ベースと事実知識を ALP インタプリタ上に読み込ませ、ゴールを満たすために必要な知識 (これを仮説集合という) をシステムが生成する。

2. ソート階層と節階層を用いて、類推解釈を行うことができる知識を、仮説集合の中からシステムが選択し、次に事実集合の知識の 1 つを用いることによって類推解釈を行う。

ここで、2. における類推解釈を行うためのアルゴリズムについて以下に示す。

ソート階層と節階層を用いた類推解釈

本研究では、類推解釈を実現するために、節階層とソート階層を利用する。類推解釈の対象にされる知識は、ALP によって生成された仮説集合である。その中で節階層にマッチするものが類推解釈の知識として選択される。そのためのアルゴリズムについて、以下に示す。

アルゴリズム 4.　(節階層利用アルゴリズム)

ALP によって生成された仮説集合を $\mathcal{H}$、部分的に半順序集合を形成する既存の節階層の要素の集合を $\mathcal{C}$とすると、節階層 $\mathcal{C}_{hier}$は $\mathcal{C}_{hier} = (\mathcal{C}, \prec)$ と表すことができる。$\mathcal{C}$の中で、葉にあたる要素の集合を $\mathcal{C}_{leaf}(\mathcal{C}_{leaf} \subseteq \mathcal{C})$ とする。

ここで、ある仮説集合の要素 $C_h(C_h \in \mathcal{H})$ が、ある節階層の葉の要素 $C_l(C_l \in \mathcal{C}_{leaf})$ と関連がある、すなわち $C_h \simeq C_l$となる時に節階層が利用され、類推解釈のための手続きを行うことになる。

C_hの述語を p_h、その n 項のソートを ${s_h}^1, \ldots, {s_h}^n$、C_lの述語を p_l、その n 項のソートを ${s_l}^1, \ldots, {s_l}^n$とすると、

1. $p_h = p_l$である。

2. $1 \leq i \leq n$ において、以下の (a)(b) のどちらかであること。

   (a)　${s_h}^i = {s_l}^i$

   (b)　$\varphi({s_h}^i) = {s_l}^i$

が、$C_h \simeq C_l$と判断されるための条件である。ただし、記号φ は、17 ページにおける**アルゴリズム 1.** で用いた抽象化関数のことである。

例えば、23 ページの図 4.3 のソート階層があり、類推解釈に用いる仮説の 1 つ $C_{h1}(C_{h1} \in \mathcal{H})$ が $C_{h1} = p_1 : s_2, s_5$で、節階層の葉の集合 $\mathcal{C}_{leaf}$が $\mathcal{C} = \{C_{l1}, C_{l2}, C_{l3}\}$ で、

$$C_{l1} = p_1 : s_2, s_3 \quad C_{l2} = p_1 : s_2, s_4 \quad C_{l3} = p_2 : s_2, s_5 \ (p_2 \neq p_1)$$

である時、仮説 C_{h1}に対する節階層利用の候補となり得る知識は C_{l1} だけである。なぜなら、C_{l2}においては、C_{h1}のソート s_5と C_{l2}のソート s_4が利用条件を満たさなく、C_{l3}においては、C_{h1}の述語 p_1と C_{l3}の述語 p_2が異なり、利用条件を満たさないからである。

次に、類推解釈のための手続きについて説明する。本研究において、ソース知識となる仮説 C_hから類推解釈される知識 (ターゲット知識) の対象になるのは、抽象化節を共有する節であり、その集合を $\mathcal{C}_{tar}(C_h)$ とすると、

$$\mathcal{C}_{tar}(C_h) = \{C \mid C \in \varphi^{-1}(\varphi(C_h)) \text{ かつ } C \neq C_h\}$$

で表すことができる。これらの節に関して類推解釈を行うかどうかはシステムが判断する。その条件は、ターゲット知識の対象となるある知識 $C_l(C_l \in \mathcal{C}_{tar}(C_h))$ が、事実集合 $\mathcal{F}$のある知識 $C_f(C_f \in \mathcal{F})$ と関連がある、すなわち $C_l \simeq C_f$となることである。

C_lの述語を p_l、その n 項のソートを ${s_l}^1, \ldots, {s_l}^n$、C_fの述語を p_f、その n 項のソートを ${s_f}^1, \ldots, {s_f}^n$とすると、

1. $p_l = p_f$である。
2. $1 \leq i \leq n$ において、以下の (a)(b) のどちらかであること。
   (a) ${s_l}^i = {s_f}^i$
   (b) $\varphi({s_l}^i) = {s_f}^i$

が、$C_l \simeq C_f$と判断されるための条件である。ただし、記号φ は、17 ページにおける**アルゴリズム 1.** で用いた抽象化関数のことである。

例えば、図 4.7 に示すような節階層があるとする。図 4.7 においては、

$$\mathcal{C}_{leaf} = \{C_{a2}, C_{a3}, C_{a4}, C_{b2}, C_{b3}, C_{b4}\}$$

である。

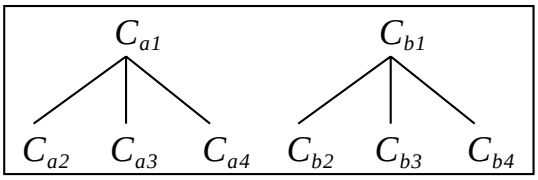


図 4.7: 類推解釈に用いた節階層の例

ここで、仮にある仮説 C_{h1}が C_{a2}にマッチした場合、$\mathcal{C}_{tar}(C_{h1}) = \{C_{a3}, C_{a4}\}$ であり、これらをターゲット知識として探索する。一方、C_{h1}が C_{b3}にマッチした場合、$\mathcal{C}_{tar}(C_{h1}) = \{C_{b2}, C_{b4}\}$ であり、これらをターゲット知識として探索する。そして、事実集合 $\mathcal{F}$に含まれるある事実 C_fがターゲット知識にマッチした場合に類推解釈を実現する。

4.3　従来の研究との相違点

本研究の方法における、従来の研究との相違点は以下のようである。

1. 異なる述語間で項のソートをの抽象化を実現することにより、節単位の知識の抽象化を実現する。
   法的類推に関する従来の研究 [4, 30, 35, 36] では、知識の抽象化を述語か項のソートかのどちらかに限って行っている。本研究の方法では、異なる述語間で項のソートを抽象化することにより、節単位の抽象化を実現している。

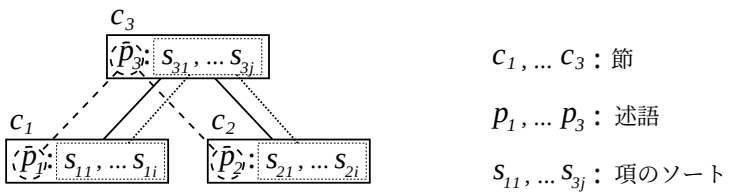


図 4.8: 抽象化節の生成

2. 不正な更新に関するチェックを行う。
   法的知識獲得に関する従来の研究 [10] では、知識を更新する際の不正な更新 (知識

ベース間の矛盾など) に関するチェックは行なっていない。本研究の方法では、ALP による仮説生成、知識の再構成の際にチェックを行うことによって、不正な更新を回避している。

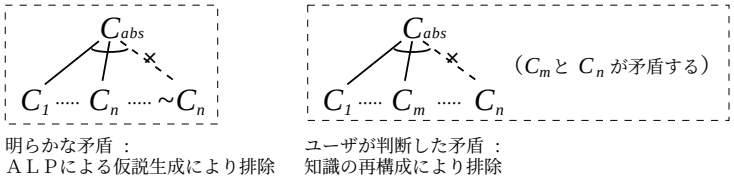


図 4.9: 不正な更新の回避

第 5 章

実装システムによる実験と考察

本研究では、不完全知識下での類推解釈を支援する法的推論システムを実現するために、知識階層生成機構と類推解釈機構を Sicstus-Prolog で実装した[1]。そして、これらのシステムを評価するためにその評価実験を行った。評価実験は、知識階層生成機構の評価を行うための**知識階層の生成に関する実験**と、類推解釈機構の評価を行うための**類推解釈に関する実験**の 2 つである。システムのソースが付録 C に、システムによる実行結果についてはその一部が付録 D に記載されているので、参照されたい。

5.1 知識階層の生成に関する実験

知識階層の生成に関する実験では、1) 推論の際に、不完全知識として生成された仮説を、関連する事実を用いて節階層を生成できる度合、2) アルゴリズムを用いて生成された節階層、から知識階層生成機構の有用性を評価する。

5.1.1 実験方法

契約の成立に関する設例を用いた知識階層生成実験

評価実験のために、法令文知識として、国際物品売買契約に関する国際条約 (CISG) の 2 部 (付録 A 参照) を用い、事実知識として、契約の成立に関する設例 (付録 B.1 節参照)

[1] ALP の処理系はメタインタプリタだが、これも Sicstus-Prolog を用いて記述されていて、インタプリタ間の値の引渡しが可能になっている。

を用いた。評価実験においては、契約が成立するかどうかをゴール知識 (質問) として与え、ALP の処理系を用いて推論を実行した時に、不完全知識として仮説が生成された 11 の設例について評価を行った。

節階層を生成する際、仮説と事実の選択はユーザが行うことになるが、この選択は、契約の成立に関する設例 (付録 B.1 節参照) において、法律家の手によって行われた推論 (付録 B.1 節参照) を参考にし、システムの作成者が行った。また、抽象化節を生成する際に、選択された仮説と事実の述語名が異なる場合、述語名を全て 'abs' にしている。

また、知識階層生成機構では節階層の生成のためにソート階層を用いる。評価実験においては、図 5.1に示すソート階層を用いた。

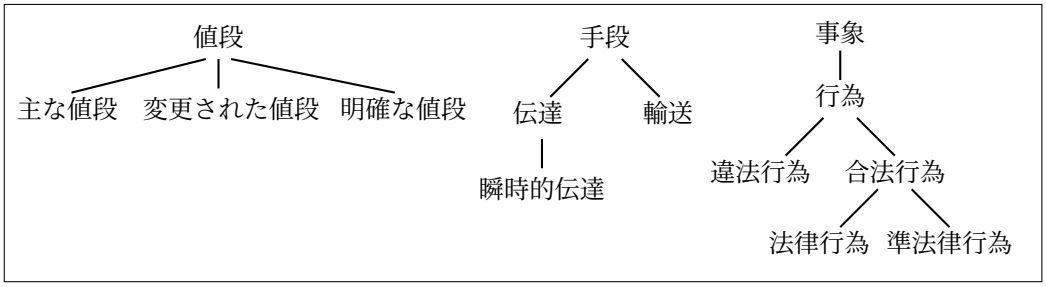


図 5.1: 実験に用いたソート階層

契約の解除に関する設例を用いた知識階層付加実験

契約の成立に関する設例を用いた知識階層生成実験と同様の手法によって、契約の解除に関する設例を用いた知識階層付加実験を行った。ただし、契約の成立に関する設例での実験とは、1) 法令文知識として、CISG の 2 部に加えて 3 部の一部を用いる、2) 新たに節階層を生成する場合は、契約の成立に関する実験で生成された節階層に付加する、という点で異なる。評価実験においては、契約の解除に関する設例 7f,7g(付録 B.2 節参照) を用いた。

5.1.2　実験結果、考察

契約の成立に関する設例を用いた知識階層生成実験

契約の成立に関する設例を用いた知識階層生成実験に関する結果について、設例別に表 5.1 に示す。

表 5.1: 知識階層生成実験の実験結果

設例	仮説数	節生成可能数	節生成不能数
6	●		●
8	●	●	
9	●●	●	●
13	●	●	
14	●●●	●	●●
14a	●●●	●	●●
15	●		●
16	●●	●	●
18	●●	●	●
18a	●●	●	●
19	●●	●●	

表 5.1 において、**仮説数**は、ゴールを与えてそれぞれの設例を解かせた場合に生成された仮説集合の数を示す。**節生成可能数**は、生成された仮説集合の中で、仮説に関連する事実知識が設例中に含まれていて節階層を生成できた仮説の数を表し、**節生成不能数**は、生成された仮説集合の中で、仮説に関連する事実知識が設例中に含まれず節階層を生成できなかった仮説の数を表す。

本実験における、仮説集合が生成された 11 設例において、仮説数は 20 であり、節生成可能数は 10 であった。よって、生成された仮説の半数が設例中の事実知識に関連付けられたことがわかる。設例中の事実知識に関連付けられなかった仮説に見られた特徴としては、仮説が成立するかどうかを判断する材料が法的知識の範囲を超えており、人間の常識によって判断しなければいけない仮説が多かったことである。

次に、生成された階層の図を図 5.2 に示す。ただし、節階層の中で、点線で囲まれた (1) の部分は、述語名「銀行振込」の節と述語名「小切手の郵送」の節との間に関連があるとユーザが指定した場合の階層である[2]。

仮に、この 2 つの節の間に関連がないとした場合、(1) の部分は図 5.3 に示される (1') のようになる。

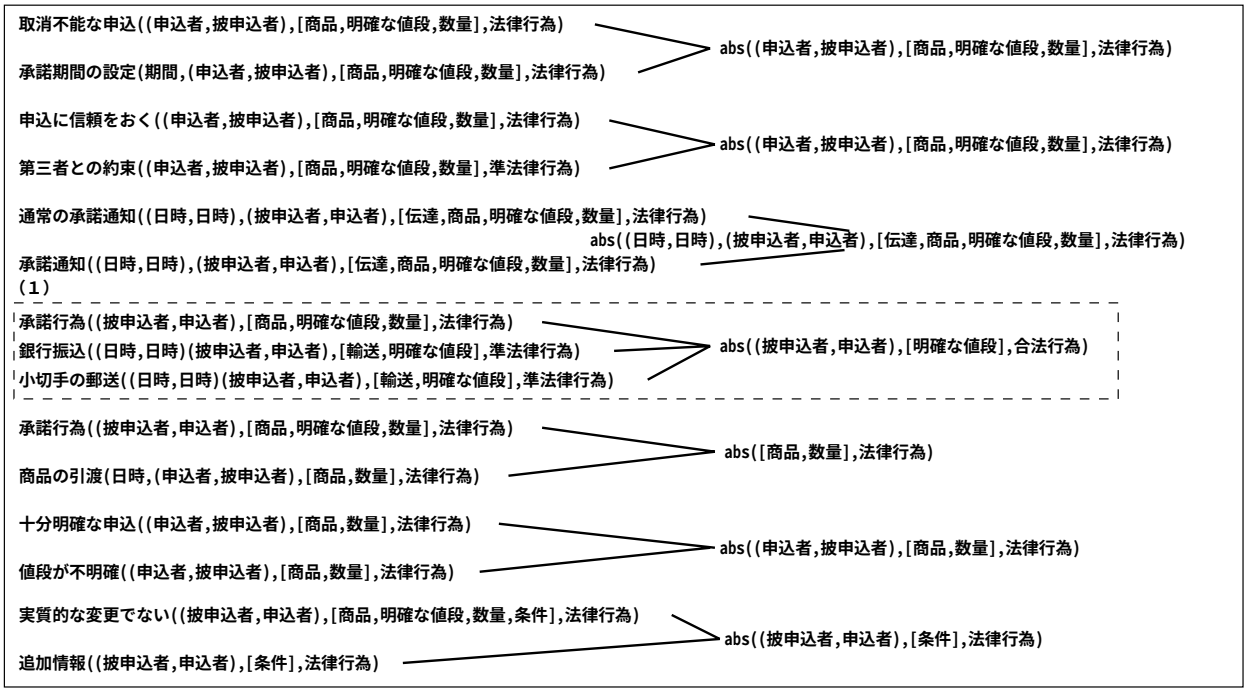


図 5.2: 知識階層生成実験により生成された節階層

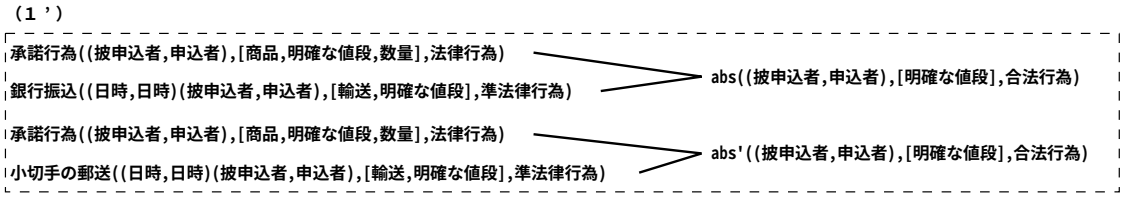


図 5.3: ユーザが解釈を替えた場合の節階層

[2]図 5.2, 5.3, 5.4, 5.5, 5.6 全てにおいていえることであるが、生成された節階層の項にはソートのみがあり、アトムの部分は取り除かれている。これは、本研究で実装したシステムにおいて、アトムを取り除く処理をしているからである。

契約の解除に関する設例を用いた知識階層付加実験

契約の解除に関する設例を用いた知識階層付加実験に関する結果について、設例 7f に関しては表 5.2 に、設例 7g に関しては表 5.3 に示す。

それぞれの表において、黒丸 (●) と二重丸 (◎) が混在しているが、黒丸は、契約が成立するために必要な仮説であり、二重丸は、契約が解除できるために必要な仮説である。

表 5.2: 知識階層付加実験の実験結果 (設例 7f)

仮説集合番号	仮説数	節生成可能数	節生成不能数
1	●●◎	●	●◎
2	●●◎	●◎	●
3	●●◎	●◎	●
4	●●◎	●	●◎
5	●●◎	●◎	●
6	●●◎	●◎	●

表 5.3: 知識階層付加実験の実験結果 (設例 7g)

仮説集合番号	仮説数	節生成可能数	節生成不能数
1	●●●●◎	●●	●●◎
2	●●●●◎	●●◎	●●
3	●●●●◎	●●◎	●●
4	●●●●◎	●●	●●◎
5	●●●●◎	●●◎	●●
6	●●●●◎	●●◎	●●

次に、追加された節階層を図 5.4 に示す。知識階層生成実験においては、契約が成立するために必要な仮説のみを扱い、節階層を生成していた。よって、知識階層付加実験においては、契約が解除できるための仮説を用いた節階層が、最初あった節階層に新たに付加される結果となった。図 5.4 において、(2) で示された部分が、本実験において新たに生成された部分である。

図 5.4: 知識階層付加実験によって付加された節階層

5.2　類推解釈に関する実験

類推解釈に関する実験では、推論の際に、知識階層生成機構を用いて生成された節階層と既存のソート階層を用いて類推解釈を実現できる度合から、類推解釈機構の有用性を評価する。

5.2.1　実験方法

評価実験のために、法令文知識として、国際物品売買契約に関する国際条約 (CISG) の 2 部、3 部の一部 (付録 A 参照) を用い、事実知識として、契約の解除に関する設例の 7f,7g(付録 B.2 節参照) を用いた。評価実験においては、契約が解除できるかどうかをゴール知識 (質問) として与え、ALP の処理系を用いて推論を実行した時に生成された仮説について評価を行った。

また、類推解釈を実現するために節階層とソート階層を必要とする。節階層は、5.1 節の知識階層の生成に関する実験で生成された図 5.2 の節階層をそのまま使用した[3]。ソート階層は、図 5.1 に示されている知識階層生成機構で用いたものと同じものを使用した。

5.2.2　実験結果、考察

設例 7f を用いた実験結果

表 5.4: 類推解釈実験の実験結果 (設例 7f)

仮説集合番号	仮説数	類推解釈可能数	類推解釈不能数
1	●●◎	●	●◎
2	●●◎	●	●◎
3	●●◎	●	●◎
4	●●◎	●	●◎
5	●●◎	●	●◎
6	●●◎	●	●◎

[3] ここで、図 5.4 でなく図 5.2 を用いた理由は、知識階層生成実験のために用いる設例と類推解釈実験のために用いる設例を異なるものにするためである。

まず、ALP の処理系を用いて設例 7f を解かせて生成された仮説集合に関する結果について、表 5.4 に示す。設例 7f を解かせた時、解を満たす仮説集合は 6 つ生成されたので、それぞれの仮説集合について示している。

表 5.4 において、**仮説数**は、ゴールを与えてそれぞれの設例を解かせた場合に生成された仮説集合の数を示す。**類推解釈可能数**は、生成された仮説集合の中で、図 5.2 の節階層を利用して類推解釈が実現できた仮説数を示し、**類推解釈不能数**は、図 5.2 の節階層を利用できず、類推解釈が実現できなかった仮説数を示す。

表 5.4 において、黒丸 (●) と二重丸 (◎) が混在しているが、黒丸は、契約が成立するために必要な仮説であり、二重丸は、契約が解除できるために必要な仮説である。

次に、類推解釈実験で用いた節階層を図 5.5 に示す。図 5.5 において、(3) で示された部分が類推解釈のために利用された節階層である。知識階層実験で生成された図 5.2 の節階層は契約の成立に関するものであり、類推解釈に用いられた仮説も契約の成立に関するものであった。

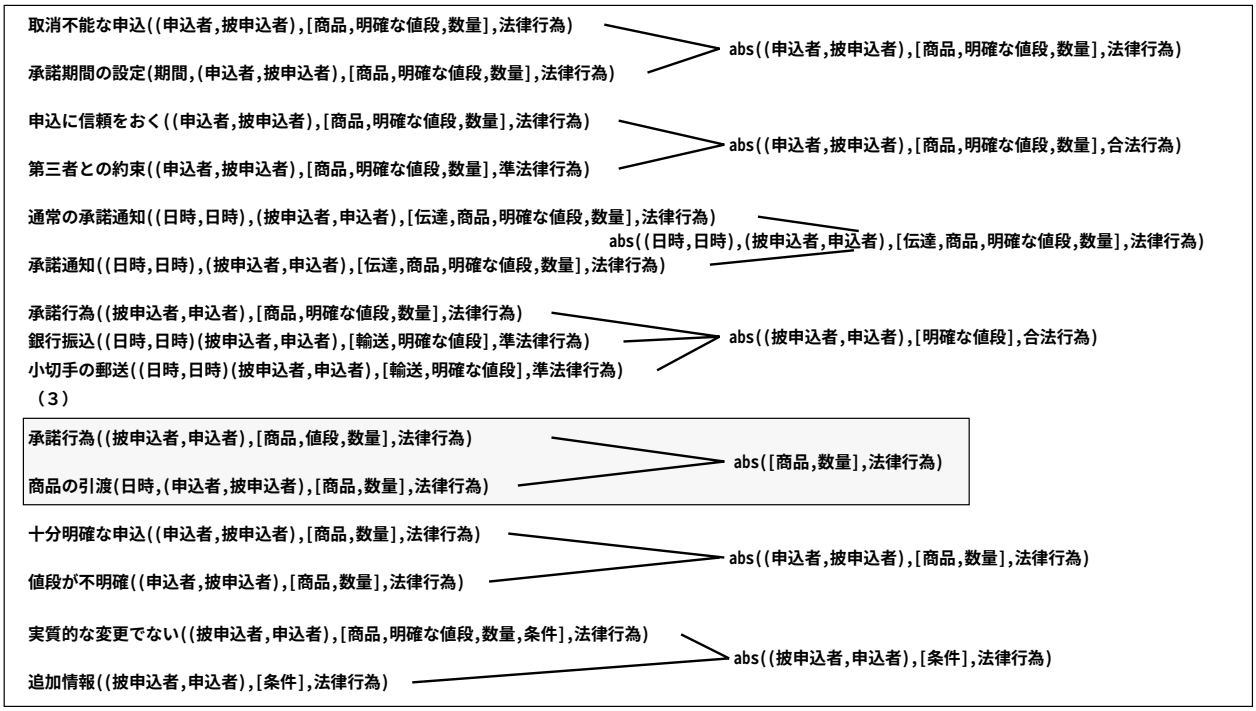


図 5.5: 類推解釈に利用された節階層 (設例 7f)

設例 7g を用いた実験結果

表 5.5: 類推解釈実験の実験結果 (設例 7g)

仮説集合番号	仮説数	類推解釈可能数	類推解釈不能数
1	●●●●◎	●●	●●◎
2	●●●●◎	●●	●●◎
3	●●●●◎	●●	●●◎
4	●●●●◎	●●	●●◎
5	●●●●◎	●●	●●◎
6	●●●●◎	●●	●●◎

節例 7f と同様にして、設例 7g を解かせて生成された仮説集合に関する結果について、表 5.5 に示す。設例 7g を解かせた時、解を満たす仮説集合は 6 つ生成されたので、それぞれの仮説集合について示している。

表 5.5 における項目と 2 種類の丸の見方は、設例 7f の実験で示した表 5.4 と同じである。

次に、類推解釈実験で用いた節階層を図 5.6 に示す。図 5.6 において、(4)(5) で示された部分が類推解釈のために利用された節階層である。設例 7f の実験と同様に、契約の成立に関する仮説が類推解釈された結果となった。

また、設例 7f より設例 7g の方が、生成された仮説数や、類推解釈実験によって類推解釈された仮説数が多かったが、これは、設例 7g の方が設例 7f より曖昧な記述が多く (付録 C 参照)、結果として不完全知識としての仮説集合が増えることになり、類推解釈を実現できる仮説の数も増えたものと考えられる。設例 7f と 7g の結果を見る限り、不完全知識が多いほど、類推解釈実験による効果がより得られるといえる。

図 5.6: 類推解釈で利用された節階層 (設例 7g)

第 6 章

結論

6.1 研究内容のまとめ

本研究では、法的推論システムを実現するのに必要な法的発見の推論機構を実現することを目的として、そのための調査研究とシステムの実装・評価を行った。

調査研究の中で、法的発見の推論機構を実現するためには不完全知識を扱う必要があることが指摘され、それを実現するには演繹推論だけでは不十分で、推論を拡張する必要があることが明らかになった。また、法的発見の推論を実現するための研究は既に一部の研究室で行われていたが、推論機構の実現範囲の制限、知識の整理の不十分さから、推論が適切に行われないという問題点が残されていた。

そこで本研究では、拡張的推論を実現する手法として、アブダクションと類推に注目した。これらの推論を実現するための基盤として、論理プログラミングをアブダクションに拡張したアブダクティブ論理プログラミング (Abductive Logic Programming, ALP) の処理系を用いた。知識獲得における抽象化を実現するために、知識表現として順序ソート論理を用いた。そして、ALP の処理系によって生成される不完全知識を利用する形で、類推解釈を実現するために必要な知識を獲得するための機構と、類推解釈を実際に行う機構を作成した。

本研究におけるシステムが他研究と違う最も重要な点は、知識の階層を節単位で構成する点である。知識獲得の際は、類推解釈を実現する際に必要な抽象化知識を生成するが、その際に異なる述語間で項のソートを抽象化することにより、節単位での抽象化を実現している。

そして、本研究において作成したシステムによって 2 つの評価実験を行った。知識階層の生成に関する実験では、本研究において提案したアルゴリズムによってユーザの手により関連知識を選択することにより、実際に節単位での階層を生成することができた。類推解釈に関する実験では、本研究において提案したアルゴリズムによって、生成された節階層を利用して、新たな設例を用いて推論を実行させた時に、一部の不完全知識を類推解釈することができた。

6.2　本研究における成果

本研究における最も重要な成果は、アブダクションと類推の枠組を統合する枠組を提供し、法的推論に広く応用できるための一つの足掛かりを作ったことである。

6.3　今後の課題

本研究において、今後の課題として残された点を以下に示す。

- 抽象化節から抽象化節を生成するための手法の提案
  本研究では、事実と仮説から抽象化節を生成する手法を提案したが、生成された抽象化節から新たに抽象化節を生成するといった、生成された抽象化節そのものの関連についてはまだ扱っていない。そのためには本研究で用いたアルゴリズムにさらに手を加える必要があるといえる。

- 類推解釈に関するユーザ依存部分を縮小する手法の提案
  本研究では、節階層の生成における関連知識の選択などをユーザの判断に任せているので、類推の完全自動化を行っているわけではない。知識同士に関連性を認めるかどうかの判断は、状況あるいは判断する人により異なると考えられ、完全自動化は非常に困難であるといえる。しかし、ユーザに依存する部分を縮小するための枠組を考える余地は残されているといえる。

- 不適切な類推解釈を回避する手法の提案
  本研究の手法では、異なる述語間の類推が可能であり、類推解釈を実現できる範囲が拡張されている。推論できる範囲が拡張されるということは、推論の可能性を広

げるという利点をもつ反面、不適切な推論をする危険性を高めるという欠点ももつ。本研究のシステムにおいては、類推解釈を不適切なものにしないように、節階層生成の際に無矛盾性の処理を行っているが、類推解釈を行う際の制約が十分ではない。今回の実験においては不適切な類推解釈は生じていないが、生じる可能性はあるといえるので、この点についても検討する必要があるといえる。

謝辞

本研究を進めるにあたって、お世話になった多くの方々に対し、この場で一言お礼を申し上げたいと思います。

主指導教員の國藤進教授には、研究指導をしていただいただけでなく、研究発表を含め、社会に出る上で役に立つであろう貴重な機会を与えてくださったことに感謝いたします。

自然言語処理学講座のタナラック・ティラマヌコン助手、知識工学講座の鳥居鉱太郎助手には、重大な場面で大変お世話になりました。このことに感謝いたします。

知識工学講座の金井貴氏には、研究をするための基盤を与えてくださった上、適切な助言をいただくことができました。これらのことに感謝いたします。

また、研究内外のことで助言をいただいた知識工学講座の先輩方、精神の支えになってくれていた同期の友に感謝いたします。

本研究は、文部省科学研究費補助金重点領域研究「法律エキスパートシステムの開発研究（研究代表: 吉野 一)」(課題番号 05208102) の補助を受けて行ったものです。この場を借りて感謝の意を示します。

参考文献

[1] 石塚 満 : 知識の表現と高速推論, 丸善(1996).

[2] 井上 克己 : アブダクションの原理, 人工知能学会誌, Vol. 7, No. 1, pp. 48–59 (1991).

[3] 大久保 好章, 原口 誠 : ゴールに依存した抽象化を利用した知識修正法, コンピュータソフトウェア, Vol. 14, No. 5, pp. 60–66 (1997).

[4] 角田 篤泰, 原口 誠, 大久保 好章 : ゴールに依存した抽象化による法的推論, 1996 年度人工知能学会全国大会 (第 10 回) 論文集, pp. 91–94, 早稲田大学(1996).

[5] 角田 篤泰, 原口 誠, 大久保 好章 : ゴールに依存した語彙的構造類推, 日本ソフトウェア科学会第 13 回大会論文集, 筑波研究学園都市 科学技術庁 研究交流センター(1996).

[6] 角田 篤泰, 原口 誠, 大久保 好章 : ゴールに依存した抽象化における包摂関係の保存, 1997 年度人工知能学会全国大会 (第 11 回) 論文集, pp. 166–168, 早稲田大学(1997).

[7] 金井 貴, 森沢 浩造, 松永 佳丈, 國藤 進 : ALP を用いた仮説生成・選択機構の実現, 第１５回人工知能国際会議 (IJCAI-97) ＡＩ学術展示「法律エキスパートシステムの開発研究」発表資料, 名古屋国際会議場(1997).

[8] 金井 貴 : アブダクションを用いた帰納論理プログラミングの研究, 北陸先端科学技術大学院大学情報科学研究科修士論文(1996).

[9] 機械システム振興協会 編: 法律エキスパートシステムに関する調査研究報告書, システム技術開発調査研究 61–R–4(10) (1987).

[10] 樽松 理樹 : 知識表現変更支援システムの構成に関する研究, 静岡大学大学院電子科学研究科博士論文(1996).

[11] 薦田 憲久, 大川 剛直, 安信 千津子： エキスパートシステムの設計と開発, 昭晃堂 (1997).

[12] 桜井 成一朗, 脇園 竜次, 原尾 政輝： 抽象化に基づく類推, 情報処理, Vol. 34, No. 5, pp. 558–565 (1993).

[13] 鈴木 宏昭： 類似と思考, 共立出版(1996).

[14] 曽野 和明, 山手 正史： 国際売買法, 青林書院(1993).

[15] 夏井 高人： 裁判実務とコンピュータ, 日本評論社(1993).

[16] 松永 佳丈： ALP に基づく仮説選択機構の国際統一売買法への適用に関する研究, 北陸先端科学技術大学院大学情報科学研究科修士論文(1997).

[17] 溝口 理一郎： 知識の共有と再利用研究の現状と動向, 人工知能学会誌, Vol. 9, No. 1, pp. 3–9 (1994).

[18] 森沢 浩造, 國藤 進： ALP に基づき不完全知識下での類推解釈を実現する法的推論機構, 第 8 回 AI シンポジウム 研究会資料, pp. 65–70, (1997).

[19] 吉野 一 編: 平成 8 年度文部省科学研究費重点領域研究「法律エキスパートシステムの開発研究」研究報告書(1996).

[20] 吉野 一 編: 平成 9 年度文部省科学研究費重点領域研究「法律エキスパートシステムの開発研究」研究報告書(1997).

[21] 吉野 一 編: 法律知識ベースの構築： 文部省科学研究費重点領域研究「法律エキスパート」平成 8 年度第 2 回全体研究集会資料, 明治学院大学国際会館会議室, (1996).

[22] `http://aurum.cs.inf.shizuoka.ac.jp/les/`

[23] Beierle, C., Hedtstück, U., Pletat, U., Schmitt, P. H. and Siekmann, J. : *An order-sorted logic for knowledge representation systems*, Artificial Intelligence, Vol. 55, pp. 149–191 (1992).

[24] Bollinger, T., Pletat, U. : *An Order-Sorted Logic with Sort Literals and Disjointness Constraints*, Proceedings if the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR92), pp. 413–424, Cambridge, MA (1992).

[25] Cohn, A. G. : *Improving the Expressiveness of Many-Sorted Logic*, AAAI(American Association for Artificial Intelligence)-83, pp. 84–87, Washington (1983).

[26] Gabbay, Dov M., Hogger, C. J., Robinson, J. A. eds.: *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 1 Logical Foundations, Oxford University Press, New York, (1993).

[27] Gabbay, Dov M., Hogger, C. J., Robinson, J. A. eds.: *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 2 Deduction Methodologies, Oxford University Press, New York, (1993).

[28] Gentner, D. : *Structure mapping: A theoretrical framework for analogy*, Cognitive Science, Vol. 7, pp. 155–170 (1983).

[29] Greiner, R. : *Learning by Understanding Analogies*, Artificial Intelligence, Vol. 35, pp. 81–125 (1988).

[30] Haraguchi, M. : *A Reasoning System for Legal Analogy*, Machine Intelligence, Vol. 14, pp. 323–346 (1995).

[31] Haraguchi, M., Kakuta, T. : *A Sorted Generalization for Legal Reasoning*, Proceedings of Workshop on Application of Logic Programming to Legal Reasoning, pp. 61–72 (1994).

[32] Herbrand, J. : *Recherches sur la Théory de la Démonstration*, Travaux de la Soc. des Sciences et des Lettre de Varsovie, Nr. 33,128 (1930).

[33] Holyoak, K. J., Thagard, P. (1989): *Annlogical mapping by constraint satisfaction*, Cognitive Science, Vol. 13, pp. 295–355 (1989).

[34] Kakas, A. C., Kowalski, R. A. and Toni, F. : *Abductive Logic Programming*, Journal of Logic and Computation, Vol. 2, No. 6, pp. 719–770 (1992).

[35] Kakuta, T., Haraguchi, M. and Okubo, Y. : *Legal reasoning by structural analogy based on goal dependent abstraction*, Proceedings of the Ninth International Conference on Legal Knowledge-Based Systems(JURIX-96), pp. 111–122, Tilburg, the Netherlands (1996).

[36] Kakuta, T., Haraguchi, M. and Okubo, Y. : *A Goal-Dependent Abstraction for Legal Reasoning by Analogy*, Artificial Intelligence and Law, Vol. 5, pp. 97–118 (1997).

[37] Kowalski, R. A. : *Legislation as Logic Programs*, Proceedings of the Eighth International Conference on Logic Programming, p. 910, Paris, France, MIT Press (1991).

[38] Nitta, K., Haragichi, M. and Sakurai, S. : *Representation of Legal Knowledge*, Journal of Information Processing, Vol. 15, No. 3 (1992).

[39] Okubo, Y., Haraguchi, M. : *Constructing Predicate Mappings for Goal-Dependent Abstraction*, Algorithmic Learning Theory '94, Lecture Notes of Artificial Intelligence, Vol. 872, pp. 516–531, Springer-Verlag (1994).

[40] Peirce, C. S. : *Elements of Logic*, in C. Hartshorne and P. Weiss eds. *Collected Papers of Charles Sanders Peirce*, Vol. II, Harvard University Press, Cambridge, MA (1932).

[41] Plaisted, D. A. : *Theorem Proving with Abstraction*, Artificial Intelligence, Vol. 16, pp. 47–108 (1981).

[42] Schank, R. C. : *Dynamic memory : A theory of learning in computers and people*, Cambridge, MA, Cambridge University Press (1982).

[43] Schmidt, A. : *Über deduktive Theorien mit mehreren Sorten von Grunddingen*, Math. Annalen 115, pp. 485–506, (1938).

[44] Schmidt-Schauß, M. : *Computational Aspects of an Order-sorted Logic with Term Declarations*, Lecture Notes in Artificial Intelligence, Vol. 395, Springer-Verlag (1989).

[45] Tenenberg, J. D. : *Inheritance in Automated Planning*, In Proceedings of 1st International Conference on Principles of Knowledge Representation and Reasoning, pp. 475–485 (1989).

[46] Thagard, P., Holyoak, K. J., Nelson, G. and Gochfeld, D. : *Analog retrieval by constraint satisfaction*, Artificial Intelligence, Vol. 46, pp. 259–310 (1990).

[47] Walther C. : *A Many-sorted Calculus Based on Resolution an Paramodulation*, Proceedings of the 8th IJCAI, Karlsruhe (1983).

[48] Walther, C. : *Many-Sorted Unification*, Journal of the Association for computing Machinery, Vol. 35, No.1, pp. 1–17 (1988).

[49] Yoshino, H., Haraguchi, M., Sakurai, S. and Kagayama, S. : *Towards a Legal Analogical Reasoning System: Knowledge Representation and Reasoning Methods*, Proceedings of the Fourth International Conference on Artificial Intelligence and Law, pp. 110–116, Association for Computing Machinery, New York (1993).

付録 A

国際物品売買契約に関する国際条約 (CISG)

本研究では、推論システムによる実験として、国際物品売買契約に関する国際条約 (CISG) の 2,3 部を用いた。2 部は契約の成立に関するものであり、3 部は物品売買の義務に関するものである。以下にそれらについて示す。実験において多くの法令文を用いた 2 部については全て示すが、法令文の一部だけを使用した 3 部については、使用した部分だけを示すことにする。

第 II 部　契約の成立

第 14 条 〔「申込」の間接的定義〕

(1) 一又は複数の特定の者に向けられた契約締結の申入れは、それが十分明確であり、かつ、承諾があった場合には拘束されるとの申込者の意志が示されているときは、申込となる。申入れは、物品を示し、かつ、明示又は黙示に数量及び代金を定め又はその決定方法を規定している場合には、十分明確なものとする。

(2) 不特定の者に向けられた申入れは、申込の単なる誘引として扱う。ただし、申入れをした者が異なった意向を明瞭に示している場合はこの限りでない。

第 15 条 〔申込の効力発生期間〕

(1) 申込は、披申込者に到達した時にその効力を生ずる。

(2) 申込は、たとえ取消不能のものであっても、申込の撤回通知が申込の到達前又はそれと同時に披申込者に到達する場合には、撤回し得る。

第 16 条〔申込の取消可能性とその期限〕

(1) 契約が締結されるまで、申込は取消すことができる。ただし、この場合には、披申込者が承諾の通知を発する前に取消の通知が披申込者に到達しなければならない。

(2) しかしながら、申込は、次のいずれかの場合には、取消すことができない。

(a) 申込が、承諾期間の設定その他の方法により、取消不能のものであることを示している場合。

(b) 披申込者が、申込を取消不能のものであると了解したのが合理的であり、かつ、披申込者がその申込に信頼をおいて行動している場合。

第 17 条〔拒絶による申込の失効〕

申込は、たとえそれが取消不能であっても、その拒絶通知が申込者に到達した時は、その効力を失う。

第 18 条〔承諾、その効力発生時期、申込の承諾期間〕

(1) 申込に同意する旨を示す披申込者の陳述その他の行為は、承諾とする。沈黙又は反応のないことは、それだけでは承諾と見なされることはない。

(2) 申込に対する承諾は、同意の意志表示が申込者に到達した時にその効力を生ずる。同意の意志表示が、申込者の定めた期間内に申込者に到達しないとき、また期間の定めがない場合においては、申込者が用いた通信手段の迅速性を含め取引の状況を十分に勘案した合理的な期間内に到達しないとき、承諾は効力を生じない。口頭による申込は、特段の事情がある場合を除き直ちに承諾されなければならない。

(3)

第 19 条〔申込の条件付承諾〕

(1) 承諾の形をとっているが、付加、制限その他の変更を含んでいる申込に対する回答は、申込の拒絶であり、反対申込となる。

(2) しかしながら、承諾の形をとった申込に対する回答が、付加的条件や異なった条件を含んでいても、申込の内容を実質的に変更するものでない場合には、申込者が不当に遅滞することなくその相違に口頭で異義を述べ又はその旨の通知を発しない限り承諾となる。申込者が異義を述べない場合には、契約の内容は申込の内容に承諾中に含まれた修正を加えたものとする。

(3) 付加的条件又は異なった条件であって、特に代金、支払、物品の品質及び数量、引渡の場所及び時期、一方当事者の相手方に対する責任の限度、又は紛争の解決方法に関するものは、申込の内容を実質的に変更するものとして扱う。

第 20 条〔申込の承諾期間の計算方法〕

(1) 申込者が電報又は書簡中で定めた承諾期間は、電報の発信を依頼した時点又は書簡に示された日付、またかかる日付が示されていない場合においては封筒に示された日付から起算する。申込者が電話、テレックスその他の瞬時的通信手段によって承諾期間を定めたときは、その期間は、申込が披申込者に到達した時点から起算する。

(2) 承諾期間中の公の休日又は非取引日も期間の計算に算入する。ただし、期間の末日が、申込者の営業所所在地の公の休日又は非取引日にあたるため、承諾の通知が期間の末日に申込者に配達され得ない場合には、期間はこれに次ぐ第一の取引日まで延長される。

第 21 条〔遅延した承諾〕

(1) 遅延した承諾といえども、申込者が有効な承諾として扱う旨を遅滞なく披申込者に口頭で通告し又はその旨の通知を発した場合には、承諾としての効力を有する。

(2) 遅延した承諾を含む書簡その他の書面が、通常の通信状況であれば適切な時期に申込者に到達したであろう状況の下で発送されたことを示しているときは、申込者が遅滞なく披申込者に対して申込が既に失効したものとして扱う旨を口

頭で通告するか、又はその旨の通知を発しない限り、遅延した承諾であっても承諾としての効力を有する。

第 22 条〔承諾の撤回〕

承諾は、その撤回通知が、承諾の効力が生じたであろう時よりも前又はそれと同時に申込者に到達すれば、撤回できる。

第 23 条〔契約の成立時期〕

契約は、申込に対する承諾がこの条約の規定に従って効力を生じた時に成立する。

第 24 条〔意志表示等の「到達」の定義〕

この条約第 II 部の適用上、申込、承諾の宣言、その他の意思の表示が相手方に「到達」した時とは、相手方にそれが口頭で伝えられた時、又はその他の方法で相手方に個人的に若しくは相手方の営業所又は郵便送付先に、また相手方が営業所も郵便送付先をも有しない場合においては相手方の常居所に配達された時とする。

第 III 部　物品売買

第 30 条〔売主の一般的義務〕

売主は、契約及びこの条約の定めるところに従い物品を引き渡し、それに関する書類を交付し、かつ、物品上の権源を移転しなければならない。

第 31 条〔引渡の場所〕

売主が物品を他の特定の場所で引き渡すことを要しない場合には、売主の引渡義務は、次の通りとする。

(a) 売買契約が物品の運送を予定する場合には、買主に送付のための物品を第一の運送人に交付すること。

(b) 前号が該当しない場合において、契約が、特定物、又は、特定の在庫品中から抽出されるべき又は製造若しくは生産されるべき不特定物に関するものであり、かつ、契約締結時に、両当事者が、物品が特定の場所に存在し又はそこで製造あるいは生産されることを知っていた場合には、その場所で物品を買主の処分に委ねること。

(c) その他の場合には、契約締結時において売主が営業所を持っていた場所で、物品を買主の処分に委ねること。

第 46 条〔特定履行、代替品引渡又は修理の要求〕

(1) 買主は、売主に対してその義務の履行を要す求ることができる。ただし、買主がこの要求と両立し得ない救済を求めている場合はこの限りでない。

(2) 物品が契約に適合していない場合には、買主は代替品の引渡を要求することができる。ただし、その不適合が重大な契約違反を構成し。かつ、その要求が、第 39 条の下での通知の際又はその後合理的な期間内になされたときに限る。

(3) 物品が契約に適合していない場合において、全ての状況から見て不合理でないときは、買主は売主に対してその不適合を修理によって治癒することを要求できる。修理の要求は、第 39 条の下での通知の際又はその後合理的な期間内になされなければならない。

第 47 条〔履行のための付加期間の付与〕

(1) 買主は、売主による義務の履行のために、合理的な長さの付加期間を定めることができる。

(2) その期間内に履行しない旨の通知を売主から受け取った場合でない限り、買主はその期間中契約違反についてのいかなる救済をも求めることができない。ただし、これにより買主は履行の遅滞について損害賠償を請求する権利を失うことはない。

第 49 条〔買主による契約の解除〕

(1) 買主は、次のいずれかの場合には、契約の解除を宣言することができる。

(a) 契約又はこの条約に基づく売主の義務のいずれかの不履行が、重大な契約違反となる場合。

(b) 引渡の不履行の場合であって、第 47 項 (1) 項に基づき買主が定めた付加期間内に、売主が物品を引き渡さない場合、又は売主がその期間内に引渡をしない旨を宣言した場合。

(2) しかしながら、売主が物品を既に引き渡している場合においては、次に掲げる時期に契約の会場を宣言しない限り、買主は解除を宣言する権利を失う。

(a) 遅延した引渡については、買主が引渡のなされたことを知った時以後の合理的期間内。

(b) 遅延した引渡以外の違反については、次に掲げるいずれかの時以後の合理的期間内。

(i) 買主がその違反を知り又は知るべきであった時。

(ii) 第 47 条 (1) 項に基づき買主が定めた付加期間が経過した時、又は売主がその付加期間内に義務の履行をしない旨を宣言した時。

(iii) 第 48 条 (2) 項に基づき売主が示した付加期間が経過した時、又は買主が履行を受け入れない旨を宣言した時。

付録 B

契約の成立と解除に関する設例

B.1　契約の成立に関する設例

本研究では、知識階層生成実験の設例として、契約の成立に関する設例を用いた。それらについて以下に示す。

設例 4 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙は、4 月 8 日に B に到達した。その前日の 4 月 7 日に、A は B に電話をして、「建設機械を 1 万ドルで販売する旨の申込の手紙を出したが、申込はなかったことにしてほしい」と述べた。B は、即座に、「その申込を承諾する」と述べた。契約は、成立したことになるか。

推論 4 申込は、申込の撤回通知が申込の到達前または到達と同時に到達したときは、撤回することができ、申込はなかったことになる (15 条 2 項)。意思表示は、相手方に口頭で伝えられた時や郵便配達時に、相手方に「到達」したことになる (24 条)。設例では、申込の到達前に、撤回の通知が到達しているので、B が承諾しても契約は成立しない。

設例 5 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。その前日の 4 月 7 日に、A は B に電話をして、「建設機械を 1 万ドルで販売する旨の申込の手紙を出したが、

申込はなかったことにしてほしい」と述べた。B は、「いや、その申込を承諾する」と述べた。契約は、成立したことになるか。

推論 5　「4 月末日までは取り消さないので、その日までに返答されたい」旨の記載により、4 月末日までは取消不能であることを A が示しているのであるが、取消不能の申し込みであっても、撤回は可能である (15 条 2 項)。したがって、B の承諾によっても、契約は成立しない。

説例 6　4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙は、4 月 8 日に B に到達した。B が返事を出す前の 4 月 9 日に、A は B に電話をして、「申込はなかったことにしてほしい」と述べた。B は、「いや、その申込を承諾する」と述べた。契約は、成立したことになるか。

推論 6　申込の効力は、手紙が B に到達した 4 月 8 日に発生している (15 条 1 項)。申込の効力発生後でも、B が承諾の通知を発する前であれば、A の申込の取消しの通知が B に到達することを条件に A は申込を取消すことができる (16 条 1 項)。設例では、電話による B の承諾の通知は、電話による A の取り消しの通知の到達より後で発せられているので、A による取り消しが有効となり、その後の B の承諾によっても契約は成立しない。ただし、申込が取消不能のものであれば、A は取消できない。手紙の文言が実際はどうであったのかは、設例では触れられていない。この点 (取消不能の有無) については、設例に書かれていなければの書かれていないことは存在しないものとして扱うというデフォールト処理をしておけばよかろう。

設例 7　4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。その翌日の 4 月 9 日に、A は B に電話をして、「建設機械を 1 万ドルで販売する旨の申込の手紙を出したが、申込はなかったことにしてほしい」と述べた。B は、「いや、その申込を承諾する」と述べた。契約は、成立したことになるか。

推論 7　「4 月末日までは取り消さないので、その日までに返答されたい」旨の記載により、4 月末日までは取消不能であることを A が示している。その期間は、A は申込を取消すことができない (16 条 2 項 a 号)。したがって、B の承諾によって、契約は成立する。

設例 8 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B が返事を出す前の 4 月 9 日に、A は B に電話をして、「申込はなかったことにしてほしい」と述べた。B は、「いや、その申込を承諾する」と述べた。契約は、成立したことになるか。

推論 8 「4 月末日までに返答されたい」との記載は、承諾期間の設定を意味する。承諾期間の設定が、取消不能を示すものであれば、この期間は A として申込を取消すことができない (16 条 2 項 a 号)。しかし、承諾期間は、それだけでただちに取消不能を意味するのではなく、最終的には、当事者間の交渉過程、慣習、その後の行為等の一切の状況を考慮に入れて判断されるが (8 条 3 項)、取消不能の意味であると主張する側の重要な根拠になる。

設例 9 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、承諾期間については特に明示されていない。手紙は、4 月 8 日に B に到達した。B が返事を出す前の 4 月 9 日に、A は B に電話をして、「申込はなかったことにしてほしい」と述べた。B は、A からの手紙を受け取った直後に、C にその機械を 1 万 2000 ドルで販売する契約を結んでいたので、「申込を承諾する」と述べた。契約は、成立したことになるか。

推論 9 申込に取消不能が示されていなくても、申し込みを受けた B が申込を取消不能のものであると了解したことに合理性があり、かつ B がその申込に信頼を置いて行動している場合には、A は申込を取消すことができない (16 条 2 項 b 号)。B が C と機械の転売契約を既に締結していることは、B が A の申込に信頼を置いて行為していることを意味するが、取消不能のものと B が了解したことに合理性があるかどうかは、取引の特徴、緊急性、その他の事情から判断される。設例では、B はまず A に承諾の通知をしておくべきであり、そうすることなしに転売契約を結んでしまった点に、信頼の合理性がないといえる。

設例 10 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは申込を取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B は、4 月 10 日に、申込を拒絶するとの返事を出し、その返事は 4 月 17 日に A に到達した。4 月

20 日、B は A は電話をして、「拒絶の通知をしたが、気が変わったので、承諾することにする」と述べた。A は、「それには応じられない」と述べた。契約は、成立したことになるか。

推論 10 申込は、たとえ、一定期間取消不能のものであっても、拒絶通知が申込者 A に到達した時点以降は、申込としての効力を失う (17 条)。したがって、その時点以降、B が承諾しても、契約は成立しない。

設例 11 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙は、4 月 8 日に B に到達した。B は、4 月 10 日に、申込を承諾するとの返事を出し、その返事は 4 月 17 日に A に到達した。4 月 20 日、B は A は電話をして、「承諾の通知をしたが、気が変わったので、承諾を取り消したい」と述べた。B は、「それには応じられない」と述べた。契約は、成立したことになるか。

推論 11 契約は、申込に対する承諾が効力を生じた時点で、成立する (23 条)。承諾は、その通知が申込者 A に到達した時に効力を生ずるので (18 条 2 項 1 文)、設例では契約が成立している。

設例 12 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙は、4 月 8 日に B に到達した。B は、4 月 10 日に、申込を承諾するとの返事を出し、その返事は 4 月 17 日に A に到達した。返事の到達する前日の 4 月 16 日に、B は A は電話をして、「承諾の通知をしたが、気が変わったので、承諾を撤回したい」と述べた。B は、「それには応じられない」と述べた。契約は、成立したことになるか。

推論 12 承諾は、承諾の効力が生じたであろう時よりも前、またはそれと同時に申込者 A に、承諾の撤回の通知が到達したときは、撤回できる (22 条)。設例では、承諾の通知の到達前に、撤回の通知が到達しているので、撤回は有効であり、契約は成立しない。

設例 13 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは申込を取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B は、4 月 15 日に承諾の返事を出したが、その返事が A に到達したのは、5 月 1 日であった。A と

して、期間終了後の承諾として放置していたところ、B が契約の履行を請求してきた。契約は成立しているか。

推論 13 承諾の返事が承諾期間内に申込者に到達しないときは、承諾は効力を生じない (18 条 2 項 2 文)。しかし、B の承諾の返事の日付から、通常の通信状況であれば、4 月末日までに到達しているはずであることが明らかな場合は、A が申し込みが既に失効していることを遅滞なく B に通告しない限り、遅延した承諾であっても承諾としての効力が認められるので (21 条 2 項)。通常、AB 間の手紙が 1 週間で到達するケースである場合には、放置していた A は契約の不成立を主張できない。

設例 14 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、承諾期間については特に明示されていない。手紙は、4 月 8 日に B に到達した。4 月 15 日、B は購入するつもりで代金 1 万ドルを銀行を通じて送金し、4 月 16 日に A の口座に入金された。契約は成立しているか。

推論 14 申込に対する承諾は、申込に同意する旨の陳述 (明示の意思表示) のみならず、同意の旨を示すその他の行為 (黙示の意思表示) によってもなすことができる。したがって、B による 1 万ドルの銀行送金が同意の旨を示す黙示の意思表示であると判断されるならば、それが A に到達した時点で、契約が成立する。しかし、A が B の多数の取引を行っている場合であれば、B からの入金だけではどの取引の分であるのかがわからないし、その他の者とも多数の取引をしている場合にも、常時入金の状況とその趣旨をチェックするのは負担である。したがって、従来からのその種のやり方での取引が行われているような場合を別にすれば、承諾とは認められないことが多いと思われる。その後に、A からの問い合わせや、B からの問い合わせがあって、承諾の趣旨であることが明確になった時点で、実際の承諾があったことになろう。さらに、18 条 3 項によれば、申込に、代金の発送をもって承諾としての同意を示すことができるとされている場合や、当事者間でそのような確立された慣行または慣習がある場合は、その行為の時点で承諾としての効力が生じ、契約が成立する。したがって、A の手紙にその趣旨が書いてあったり、その趣旨の慣行・慣習があれば、銀行送金の時点で契約が成立していることになる。しかし、このような慣行・慣習は希である。

設例 14 のバリエーション A　設例 14 で、B が購入するつもりで代金 1 万ドルの小切手を A に郵送した場合はどうか。

推論 14 のバリエーション A　小切手は手紙で配達されてくるので、承諾の黙示の意思表示ありとされる可能性が大きい。

設例 15　4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、承諾期間については特に明示されていない。手紙は、4 月 8 日に B に到達した。1 月後の 5 月 8 日、B は承諾の返事を A に発送し、その返事は 5 月 15 日に A に到達した。契約は成立するか。

推論 15　承諾期間の定めのない申込に対する承諾は、取引の状況を考慮して判断される「合理的期間内」に申込者に到達しない場合には、効力が生じない (18 条 2 項 2 文)。A が手紙を出してから、返事が到達するまでの 1 月半が合理的期間内といえるかどうかによる。したがって、結論は不明。

説例 16　4 月 1 日、A は、B に対して、建設機械を販売する旨の申込みの手紙を出したが、価格は記載していなかった。手紙には、「4 月末日までは申込を取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B は、4 月 10 日に承諾の返事を出し、その返事は 4 月 17 日に A に到達した。契約は成立しているか。

推論 16　有効な申込であるためには、特定の者に向けられており (14 条 1 項 1 文)、承諾があれば拘束されるとの意思が示されており (同)、十分明確であること (同) 次の場合は、十分明確である (14 条 1 項 2 文) 物品を示し明示または黙示に数量を定めまたはその決定方法を定め明示または黙示に代金を定めまたはその決定方法を定めている設例では、明示の代金の定めはない。しかし、黙示に代金が定められているか、または代金の決定方法が定められていると認められるような場合は、申込として有効である。明示または黙示に代金を定めておらず、その決定方法をも定めていない場合であっても、契約が有効に締結されておれば、特段の事情のない限り、両当事者は、契約締結時における同種物品の市場価格に黙示に言及しているものとみなすとの 55 条との関係で、両当事者に契約意思があれば、契約は有効に成立し、代金については 55 条の定めるところによるとの説 (曽野＝山手説) と、55 条は条約第 2 部 (14 条

を含む) に加入せず、第 3 部 (55 条を含む) に加入している締約国に当事者の一方が営業所を有する場合で、当該締約国の法が、代金の定めがまったくなくても契約が有効に締結されることを認めている場合にのみ適用されるとする説 (UUNCITRAL Secretary Commentary 説) とに分れている。前者の説は、別々に起草されたために矛盾した内容を含んでいる第 2 部と第 3 部とを整合的に解釈し、ＣＩＳＧの適用される国際的契約をなるべく広く認めようとの立場にたっている。

設例 17 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは申込を取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B は、4 月 9 日に、「承諾する。ただし、代金は 9000 ドルのこと」との返事を出し、その返事は 4 月 16 日に A に到達した。A がそのまま放置していたところ、5 月に入って、B が A に建設機械の引渡を請求してきた。契約は成立しているか。

推論 17 承諾の形をとっていても、付加、制限、変更を含んでいるときは、申込の拒絶であり、反対申込になる (19 条 1 項)。承諾における価格の変更は、申込内容を「実質的に変更」するものに当たるので (19 条 3 項)、A として放置しておいても、契約は成立しない。

設例 18 4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは申込を取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B は、4 月 9 日に、「承諾する。ただし、代金は 9000 ドルのこと」との返事を出し、その返事は 4 月 16 日に A に到達した。5 月 1 日に建設機械を B に引き渡され、5 月 5 日に、A は B に代金 1 万ドルを請求した。契約は成立しているか。成立している場合、B は代金をいくら支払わなければならないか。

推論 18 B の返事は、A の申込を実質的に変更するものであり、承諾ではなく、反対申込である。これに対して、A は承諾の返事をしていない。しかし、目的物を B に引き渡したことは、同意を意味し、承諾と認められる (18 条 1 項)。B は 9000 ドルを支払えばよい。

設例 18 のバリエーション A 設例 18 で、A から B に、1 万ドルの請求書を付けて建設機械が配達された場合はどうか。

推論 18 のバリエーション A　B からの 9000 ドルでとの反対申込に対して、A が拒絶し、1 万ドルでとのサイドの反対申込を行うとともに、現物を一方的に送り付けたものと見ることができる。B は、1 万ドルで購入する気がないのなら、その旨を連絡すれば、拒絶がされたことになり、契約は成立しない。しかし、だまっていると、黙示の承諾があったとみなされる場合もある (18 条 1 項は、沈黙又は反応のないことはそれだけでは承諾とみなされることはないとするが、他の条件も加わればみなされることもある)。

設例 19　4 月 1 日、A は、B に対して、建設機械を 1 万ドルで販売する旨の申込みの手紙を出した。手紙には、「4 月末日までは申込を取り消さないので、その日までに返答されたい」と記載されていた。手紙は、4 月 8 日に B に到達した。B は、4 月 9 日に承諾の返事を出したが、そこには、A からの申込の手紙には記載のない、AB 間の紛争は仲裁によって解決する旨の約定が付加されていた。返事を受け取った A は、特に異議を述べることもなく、建設機械を発送した。契約は成立するか。

推論 19　B の返事は、仲裁条項が付加されているので、承諾ではなく、反対申込である (19 条 1 項)。しかし、変更を含んでいても、申込の内容を「実質的に変更」するものでなく、かつ申込者が不当に遅滞することなく異議を述べないときは、承諾となり (19 条 2 項 1 文)、変更を加えられた承諾の内容で契約が成立する (19 条 2 項 2 文)。「実質的に変更」するものである場合は、放置しておいても契約は成立しない。仲裁条項は、「紛争の解決方法」に関するものであり、申込を「実質的に変更」するものにあたるとされる (19 条 3 項) ので、A として遅滞なく異議を述べなくても、契約が成立することはない。ただし、別段の合意がない限り仲裁が慣行となっている取引分野では、仲裁条項の付加は申込に実質的を実質的に変更することにならないから (曽野＝山手 98 頁)、設例の取引がこれに該当すれば、仲裁条項の入った B の反対申込を内容とした契約が成立する。

B.2　契約の解除に関する設例

本研究では、知識階層追加実験と類推解釈実験の設例として、契約の解除に関する設例 7f,7g を用いた。それらについて以下に示す。

設例 7f

1) 1996 年 4 月 1 日、ニューヨークの農業機械メーカー A が日本商社 B のハンブルク支店に対して、申込みの手紙を発信した。手紙の内容は、A が B に農業耕作機械一式を代金 5 万ドルで売るというもので、A は当該機械を B に対して 5 月 10 日までに引き渡す、B は代金を A に対して 5 月 10 日までに支払う、機械はアメリカの貨物船で運ぶこととあった。

2) 4 月 8 日、その手紙は B の郵便受けに届いた。

3) 4 月 9 日、B は A に電話をした。「申込みは承諾、但し日本のコンテナ船で運ばれたし」

4) 5 月 1 日、A は農業機械をニューヨーク港において日本のコンテナ船に引き渡した。

5) 5 月 10 日、B は代金 5 万ドルを A に対して支払った。

6) 5 月 31 日、機械は B のハンブルク支店に届けられた。

7) 6 月 5 日、B は機械を検査した。

8) 8 月 10 日、機械は動作異常、原因は接続ギアの不良であると判明。

9) 同日、B は A に事実を告げた。

10) 9 月 1 日、B は A に物品の契約不適合を一ヵ月以内に修理するよう要求した。

11) 10 月 1 日までに、A は不適合の修理を行わなかった。

12) 10 月 10 日、B は契約を解除すると宣言した。

設例 7g

1) 1996 年 4 月 1 日、ニューヨークの農業機械メーカー A が日本商社 B のハンブルク支店に対して、手紙を発信した。手紙の内容は、A が B に農業耕作機械一式 (それはトラクターとそれに付属するレーキからなる) を売るというもので、(主要部分である) トラクターの代金は 5 万ドルである。A は当該機械を B に対して 5 月 10 日までに引き渡す、B は代金を A に対して引渡後 20 日以内に支払う、機械はアメリカの貨物船で運ぶこととあった。

2) 4 月 8 日、その手紙は B の郵便受けに届いた。

3) 4 月 9 日、B は A に電話をした。「申込みは承諾、但し日本のコンテナ船で運ばれたし」

4) 5 月 1 日、A は農業機械をニューヨーク港において日本のコンテナ船に積んだ。

5) 5 月 10 日、B は代金 5 万ドルを A に対して支払った。

6) 5 月 31 日、機械は B のハンブルク支店に届けられた。

7) 6 月 5 日、B は機械を検査した。

8) 8 月 10 日、機械は動作異常、原因は接続ギアの不良であると判明。

9) 同日、B は A に事実を告げた。

10) 9 月 1 日、B は A に物品の契約不適合を一ヵ月以内に修理するよう要求した。

11) 10 月 1 日までに、A は不適合の修理を行わなかった。

12) 10 月 10 日、B は契約を解除すると宣言した。

付録 C

実装システムのソースリスト

C.1　システムのモジュール構成

本研究のシステムにおけるモジュール構成は以下の図 C.1 のようになっている。

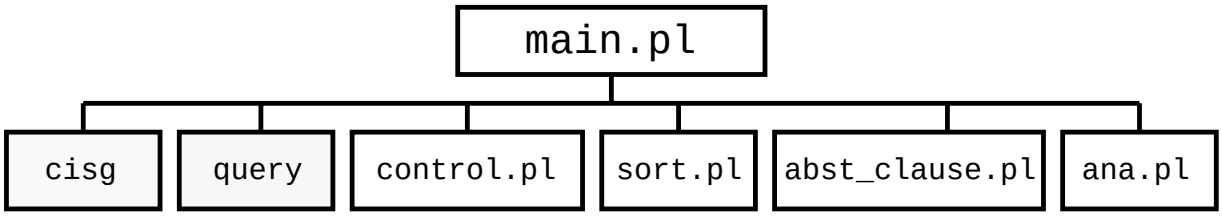


図 C.1　本システムのモジュール構成

この中で、モジュール ‘main.pl’ が行うことは、下位モジュールを読み込むことのみである。また、モジュール ‘cisg’ と ‘query’ は、通常の Prolog インタプリタではなく、ALP インタプリタ上に読み込まれる。

次に、それぞれのモジュールについて説明する。

- main.pl
  システムで必要となるモジュールを全て読み込むためのモジュール。

- cisg
  ルール化された法令文知識の集合。

- query
  法的問題を解くための質問が繁雑になるため、それを簡略化するために質問ルールとして定義したもの。

- control.pl
  主として、ユーザに対する質問部分や、表示部分を扱うモジュール。知識階層生成機構と類推解釈機構の切り替えも行う。

- sort.pl
  ソート階層の集合。

- abst_clause.pl
  知識階層生成機構の中枢部。抽象化節の生成、それに伴う節階層の生成、再構成を行う。

- ana.pl
  類推解釈機構の中枢部。節階層を用いて類推解釈を実現する。

また、事実集合モジュール 'case??' については、システム起動中に ALP インタプリタに読み込む仕組みになっている。節階層集合モジュール 'hier' は、システム起動中に Prolog インタプリタ上に読み込む。

C.2　ソースリスト

以下に、それぞれのモジュールのソースリストを示す。

C.2.1　モジュール 'main.pl'

以下のソースリストの中で、述語 'abd_call/3' は、ALP インタプリタを呼び出す述語である。

```
* ―――――――――――――――――――― from here ―――――――――――――――――――― *

:- abd_call([cisg,query],[],_).

:- [control].
:- [sort].
```

```
:- [abst_clause].
:- [ana].
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.2　モジュール ‘cisg’

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
% operator

:- op(500, fx, not).

/* Article 14 */

proposal((Offeror,Offeree),OObject,O:juristic_act) :-
   t(offer((_,_),(Offeror,Offeree),[Com|OObject],O:juristic_act)),!,
   t(sufficiently_definite_offer((Offeror,Offeree),OObject,O:juristic_act)),
   t(intension_bound((Offeror,Offeree),OObject,O:juristic_act)).

sufficiently_definite_offer((Offeror,Offeree),OObject,O:juristic_act) :-
   t(member(_:goods,OObject)),
   t(member(_:quantity,OObject)),
   t(member(_:price,OObject)),!.

/* Article 15 */

effective_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   t(proposal((Offeror,Offeree),OObject,O:juristic_act)),
   t(offer((_,ODate:date),(Offeror,Offeree),[Com|OObject],O:juristic_act)),
   greater_date(ODate,Date).

effective_withdrawal_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   t(offer((_,ODate:date),(Offeror,Offeree),[OCom|OObject],O:juristic_act)),
   t(withdrawal_offer((_,WDate:date),(Offeror,Offeree),[WCom|OObject],O:juristic_act)),
   greater_date(WDate,ODate),
   greater_date(WDate,Date).

-effective_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   effective_withdrawal_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act).
```

```
/* Article 16 */

effective_revocation_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   t(accept((ADate:date,_),(Offeree,Offeror),[ACom|AObject],A:juristic_act)),
   t(revocation_offer((_,RDate:date),(Offeror,Offeree),[RCom|OObject],O:juristic_act)),
   greater_date(RDate,ADate),
   greater_date(RDate,Date).

-effective_revocation_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   t(irrevocable_offer((Offeror,Offeree),OObject,O:juristic_act)).

irrevocable_offer((Offeror,Offeree),OObject,O:juristic_act) :-
   t(revocation_offer((_,_),(Offeror,Offeree),[RCom|OObject],O:juristic_act)),
   t(fixed_accept_period(Pdate:period,(Offeror,Offeree),OObject,O:juristic_act)),
   t(other_cause_accept_period((Offeror,Offeree),OObject,O:juristic_act)).

irrevocable_offer((Offeror,Offeree),OObject,O:juristic_act) :-
   t(revocation_offer((_,_),(Offeror,Offeree),[RCom|OObject],O:juristic_act)),
   t(reasonable_irrevocable((Offeror,Offeree),OObject,O:juristic_act)),
   t(rely_on_offer((Offeror,Offeree),OObject,O:juristic_act)).

-effective_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   effective_revocation_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act).

/* Article 17 */

-effective_offer(Date:period,(Offeror,Offeree),OObject,O:juristic_act) :-
   t(irrevocable_offer((Offeror,Offeree),OObject,O:juristic_act)),
   t(rejection_offer((_,RDate:date),(Offeree,Offeror),OObject,_:juristic_act)),
   greater_date(RDate,Date).

/* Article 21 */

effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(acceptance((Offeree,Offeror),AObject,A:juristic_act)),
   t(fixed_accept_period(PDate:period,(Offeror,Offeree),AObject,O:juristic_act)),
   t(accept((A1Date:date,A2Date:date),(Offeree,Offeror),[Com|AObject],A:juristic_act)),
   greater_date(PDate,A2Date),
   t(admit_delay_accept((A1Date:date,A2Date:date),(Offeror,Offeree),AObject,D:juristic_
   act)),
```

```
   greater_date(A2Date,Date),!.

effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(acceptance((Offeree,Offeror),AObject,A:juristic_act)),
   t(fixed_accept_period(PDate:period,(Offeror,Offeree),AObject,O:juristic_act)),
   t(accept((A1Date:date,A2Date:date),(Offeree,Offeror),[Com|AObject],A:juristic_act)),
   greater_date(PDate,A2Date),
   t(normal_accept_deliver((A1Date:date,A2Date:date),(Offeree,Offeror),[Com|AObject],A:
   juristic_act)),
   greater_date(A2Date,Date),!.

/* Article 18 */

acceptance((Offeree,Offeror),AObject,A:juristic_act) :-
   t(accept((_,_),(Offeree,Offeror),[Com|AObject],A:juristic_act)),!,
   t(indicate_accept_conduct((Offeree,Offeror),AObject,A:juristic_act)),
   t(not materially_alter((Offeree,Offeror),AObject,A:juristic_act)).

effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(acceptance((Offeree,Offeror),AObject,A:juristic_act)),
   t(accept((_,ADate:date),(Offeree,Offeror),[Com|AObject],A:juristic_act)),
   greater_date(ADate,Date).

-effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(fixed_accept_period(PDate:period,(Offeror,Offeree),AObject,O:juristic_act)),
   (t(accept((_,ADate:date),(Offeree,Offeror),[Com|AObject],A:juristic_act)) ->
   greater_date(PDate,ADate),
   greater_date(PDate,Date) ; fail),
   (t(normal_accept_deliver((_,ADate:date),(Offeree,Offeror),[Com|AObject],A:juristic_a
   ct)) ->
   fail;true).

-effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(accept((_,ADate:date),(Offeree,Offeror),[Com|AObject],A:juristic_act)),
   t(irrational_deliver_accept_period(ADate:period,(Offeree,Offeror),AObject,A:juristic
   _act)),
   greater_date(ADate,Date).

effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(acceptance((Offeree,Offeror),AObject,A:juristic_act)),
```

```
   t(custom((Offeror,Offeree),AObject,custom:juristic_act)),
   t(indicate_accept_conduct((Offeree,Offeror),AObject,A:juristic_act)),
   t(reasonable_deliver_accept_period(Date:period,(Offeree,Offeror),AObject,A:juristic_
   act)).

/* Article 19 */

-effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   effective_rejection_offer(Date:period,(Offeree,Offeror),AObject,A:juristic_act).

effective_rejection_offer(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(counter_offer((Offeree,Offeror),AObject,A:juristic_act)).

counter_offer((Offeree,Offeror),AObject,A:juristic_act) :-
   t(alter((Offeree,Offeror),AObject,A:juristic_act)),
   t(materially_alter((Offeree,Offeror),AObject,A:juristic_act)).

alter((Offeree,Offeror),AObject,A:juristic_act) :-
   t(offer((_,_),(Offeror,Offeree),[OCom|OObject],_)),
   t(member(X,OObject)),
   t(not member(X,AObject)),!.

-acceptance((Offeree,Offeror),AObject,A:juristic_act) :-
   t(discrepancy_no_alter((Offeror,Offeree),AObject,D:juristic_act)).

/* Article 22 */

effective_withdrawal_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   t(accept((_,ADate:date),(Offeree,Offeror),[ACom|AObject],A:juristic_act)),
   t(withdrawal_accept((_,WDate:date),(Offeree,Offeror),[WCom|AObject],A:juristic_act)),
   greater_date(WDate,ADate),
   greater_date(WDate,Date).

-effective_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act) :-
   effective_withdrawal_acceptance(Date:period,(Offeree,Offeror),AObject,A:juristic_act
   ).

/* Article 23 */

effective_contract(Date:period,(Offeror,Offeree),AObject,contract:juristic_act) :-
```

```
   effective_offer(Date:period,(Offeror,Offeree),OObject,offer:juristic_act),
   effective_acceptance(Date:period,(Offeree,Offeror),AObject,accept:juristic_act).

/* Article 30 */

effective_satisfy_obligatory(Date:period,(Offeror,Offeree),DObject,D:juristic_act) :-
   t(deliver_goods(DDate:date,(Offeror,Offeree),DObject,D:juristic_act)),
   t(deliver_documents(DDate:date,(Offeror,Offeree),DObject,D:juristic_act)),
   t(satisfy_condition_of_contract((Offeror,Offeree),DObject,D:juristic_act)),
   greater_date(DDate,Date),
   t(suitable_goods_of_contract((Offeror,Offeree),DObject,D:juristic_act)).

/* Article 46 */

% 買主側が代替品の引渡を要求する ⇔ 売主側が代替品を引渡す義務がある %
effective_obligatory_deliver_substitute_goods(Date:period,(Offeror,Offeree),DObject,D:j
uristic_act) :-
   t(deliver_goods(DDate:date,(Offeror,Offeree),DObject,D:juristic_act)),
   greater_date(DDate,Date),
   t(not suitable_goods_of_contract((Offeror,Offeree),DObject,D:juristic_act)),
   t(important_breach_of_contract((Offeror,Offeree),DObject,D:juristic_act)),
   (t(demand_deliver((_,DRDate:date),(Offeree,Offeror),GObject,DD:juristic_act)) ->
   greater_date(DRDate,Date);
   t(reasonable_demand_period(Date:date,(Offeree,Offeror),DObject,D:juristic_act))).

% 買主側が修理を要求する ⇔ 売主側が修理する義務がある %
effective_obligatory_repair_goods(Date:period,(Offeror,Offeree),DObject,D:juristic_act)
 :-
   t(deliver_goods(DDate:date,(Offeror,Offeree),DObject,D:juristic_act)),
   greater_date(DDate,Date),
   t(not suitable_goods_of_contract((Offeror,Offeree),DObject,D:juristic_act)),
   t(reasonable_situation_goods((Offeror,Offeree),DObject,D:juristic_act)),
   (t(demand_repair((_,DRDate:date),(Offeree,Offeror),GObject,DR:juristic_act)) ->
   greater_date(DRDate,Date);
   t(reasonable_demand_period(Date:date,(Offeree,Offeror),DObject,D:juristic_act))).

/* Article 49 */
effective_avoid_contract(Date:period,(Offeror,Offeree),DObject,AC:juristic_act) :-
   effective_contract(Date:period,(Offeror,Offeree),AObject,contract:juristic_act),
   t(avoid_contract((_,CDate:date),(Offeree,Offeror),[H|DObject],AC:juristic_act)),
```

```
   t(not effective_satisfy_obligatory(Date:period,(Offeror,Offeree),DObject,_:juristic_
   act)),
   greater_date(CDate,Date).

effective_avoid_contract(Date:period,(Offeror,Offeree),DObject,AC:juristic_act) :-
   effective_contract(Date:period,(Offeror,Offeree),AObject,contract:juristic_act),
   t(avoid_contract((_,CDate:date),(Offeree,Offeror),[H|DObject],AC:juristic_act)),
   t(effective_obligatory_deliver_substitute_goods(Date:period,(Offeror,Offeree),DObjec
   t,D:juristic_act)),
   greater_date(CDate,Date).

effective_avoid_contract(Date:period,(Offeror,Offeree),DObject,AC:juristic_act) :-
   effective_contract(Date:period,(Offeror,Offeree),AObject,contract:juristic_act),
   t(avoid_contract((_,CDate:date),(Offeree,Offeror),[H|DObject],AC:juristic_act)),
   t(effective_obligatory_repair_goods(Date:period,(Offeror,Offeree),DObject,D:juristic
   _act)),
   greater_date(CDate,Date).

effective_avoid_contract(Date:period,(Offeror,Offeree),DObject,AC:juristic_act) :-
   effective_contract(CDate:period,(Offeror,Offeree),AObject,contract:juristic_act),
   t(avoid_contract((_,CDate:date),(Offeree,Offeror),[H|DObject],AC:juristic_act)),
   greater_date(CDate,Date),
   t(obligation_added_period(DDate:period,(Offeree,Offeror),DObject,obligation_added_pe
   riod:juristic_act)),
   t(not deliver_goods(DDate:period,(Offeror,Offeree),DObject,deliver:juristic_act)),
   greater_date(DDate,Date).

/* Utility */

greater_date(Date1,Date2) :-
   dates(Dateall),
   member(Date1,Dateall),
   member(Date2,Dateall),
   Date1 =< Date2.

not A :-
   facts(Flist),
   member(A,Flist),!,fail.

not A :-
```

```
   \+ A.

t(X) :-
   facts(Flist),
   member(X,Flist).

t(X) :-
   X.
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.3　モジュール ‘query’

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
/* query */
qc6 :-
   effective_contract(19960409:period,(a:offeror,b:offeree),[construction_machine:goods
   ,10000:definite_price,1:quantity],contract:juristic_act).
qc7f :-
   effective_contract(19960409:period,(a:offeror,b:offeree),[farm_machine:goods,50000:d
   efinite_price,1:quantity,japanese_cargo_ship:change_means],contract:juristic_act).
qc7g :-
   effective_contract(19960409:period,(a:offeror,b:offeree),[farm_machine:goods,50000:m
   ain_price,1:quantity,japanese_cargo_ship:change_means],contract:juristic_act).
qc8 :- qc6.
qc9 :- qc6.
qc13 :-
   effective_contract(19960501:period,(a:offeror,b:offeree),[construction_machine:goods
   ,10000:definite_price,1:quantity],contract:juristic_act).
qc14 :-
   effective_contract(19960416:period,(a:offeror,b:offeree),[construction_machine:goods
   ,10000:definite_price,1:quantity],contract:juristic_act).
qc15 :-
   effective_contract(19960515:period,(a:offeror,b:offeree),[construction_machine:goods
   ,10000:definite_price,1:quantity],contract:juristic_act).
qc16 :-
   effective_contract(19960417:period,(a:offeror,b:offeree),[construction_machine:goods
   ,1:quantity],contract:juristic_act).
qc18 :-
   effective_contract(19960505:period,(a:offeror,b:offeree),[construction_machine:goods
```

```
   ,9000:changed_price,1:quantity],contract:juristic_act).
qc19 :-
   effective_contract(19960501:period,(a:offeror,b:offeree),[construction_machine:goods
   ,10000:definite_price,1:quantity,solve_dispute:agreement],contract:juristic_act).

qv7f :-
   effective_avoid_contract(19961010:period,(a:offeror,b:offeree),[farm_machine:goods,5
   0000:definite_price,1:quantity],avoid_contract:juristic_act).
qv7g :-
   effective_avoid_contract(19961010:period,(a:offeror,b:offeree),[farm_machine:goods,5
   0000:main_price,1:quantity],avoid_contract:juristic_act).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.4　モジュール ‘control.pl’

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
qana(Goal) :-
    getfact(Facts),
    gethypo(Goal,Hyposs),ttynl,
    disphf(Hyposs,Facts),
    readhiers,
    (anamode ->
    anas(Hyposs,Facts,NHyposs),
    disphf(NHyposs,Facts);
    choicehf(Hyposs,Facts,Hypo,Fact),
    abst(Hypo,Fact,Clause)).

anamode :-
    write('Do you enter analogical process mode? :'),
    read(X),
    agree(X),ttynl.

getfact(Facts) :-
    write('Select Facts: '),
    read(File),
    abd_call([File],[],[]),!,
    abd_call(facts(Facts),[],[]).

gethypo(Goal,Hypo) :-
```

```
    findall(A1,abd_call(Goal,[],A1),Hypo1),
    delterms(Hypo1,Hypo).

disphf(Hss,Fs) :-
    write('Generated Hypotheses:'),ttynl,ttynl,
    wlists(Hss,1),
    write('Facts:'),ttynl,ttynl,
    wlist(Fs,1).

choicehf(Hss,Fs,H,F) :-
    write('Select Hypotheses containing Analogical Hypothesis: '),
    read(HN1),
    select(Hss,Hs,HN1),
    ttynl,write('No. '),write(HN1),write('.:'),ttynl,
    wlist(Hs,1),
    write('Select Analogical Hypothesis: '),
    read(HN2),
    select(Hs,H,HN2),
    ttynl,write('Selected Hypothesis :'),ttynl,write(H),ttynl,
    ttynl,write('Select Analogical Fact: '),
    read(FN),
    select(Fs,F,FN),
    ttynl,write('Selected Fact :'),ttynl,write(F),ttynl,ttynl.

select([Hlist|Tlist],Hlist,1) :- !.

select([Hlist|Tlist],List,N) :-
    N1 is N - 1,
    select(Tlist,List,N1).

wlists([],_) :- ttynl.

wlists([Hlist|Tlist],Num) :-
    write('No. '),write(Num),write('.'),ttynl,
    wlist(Hlist,1),
    Num1 is Num + 1,
    wlists(Tlist,Num1).

wlist([],_) :- ttynl.
```

```
wlist([H|T],Num) :-
    write(Num),write(': '),
    write(H),ttynl,
    Num1 is Num + 1,
    wlist(T,Num1).

delterms([],[]).

delterms([Hlist|Tlist],Nlist) :-
    delterm(Hlist,[]),!,
    delterms(Tlist,Nlist).

delterms([Hlist|Tlist],[NHlist|NTlist]) :-
    delterm(Hlist,NHlist),
    delterms(Tlist,NTlist).

delterm([],[]).

delterm([\+ -H|T1],T2) :-
    !,delterm(T1,T2).

delterm([\+ H|T1],[not H|T2]) :-
    !,delterm(T1,T2).

delterm([H|T1],[H|T2]) :-
    delterm(T1,T2).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.5　モジュール 'sort.pl'

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
:- op(500, xfx, is_a_s).

juristic_act       is_a_s  lawful_act.
quasi_juristic_act is_a_s  lawful_act.

lawful_act         is_a_s  act.
illegal_act        is_a_s  act.
```

```
act                is_a_s  event.

instantaneous_communication is_a_s communication.

communication      is_a_s  means.
transportation     is_a_s  means.

definite_price     is_a_s  price.
main_price         is_a_s  price.
changed_price      is_a_s  price.
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.6　モジュール ‘abst_clause.pl’

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
:- unknown(_,fail).

:- op(500, fx, not).
:- op(500, xfx, is_a_c).

abst(H,F,NSortclause) :-
    getclause(H,HNum,HSortlist,HSortclause),
    getclause(F,FNum,FSortlist,FSortclause),
    abst_slist(HNum,FNum,HSortlist,FSortlist,NSortlist),
    getnewclause(NSortlist,NSortclause),
    gethierarchy(HSortclause,FSortclause,NSortclause),
    modifyhier(HSortclause,FSortclause,NSortclause),
    printhiers.

getclause(C,CNum,CSL,CSC) :-
    gtermlist(C,CPred,CTL,CNum),
    delvalue(CTL,CSL),
    gclause(C,CPred,CSL,CSC).

gtermlist(C,CPred,CTerm,CNum) :-
    (C = not C2 -> C2 =.. [CPred|CTerm];
     C =.. [CPred|CTerm],C2 = C),
    functor(C2,CPred,CNum).
```

```
delvalue([],[]).

delvalue([(X:XS,Y:YS)|T],[(XS,YS)|NT]) :-
    !,delvalue(T,NT).

delvalue([X:XS|T],[XS|NT]) :-
    !,delvalue(T,NT).

delvalue([L|T],[NL|NT]) :-
    !,delvalue(L,NL),
    delvalue(T,NT).

gclause(C,CPred,CSL,CSC2) :-
    CSC =.. [CPred|CSL],
    (functor(C,not,1) -> CSC2 = not CSC;CSC2 = CSC).

abst_slist(HN,FN,HSL,FSL,NSL) :-
    (HN = FN -> abst_slist(HSL,FSL,NSL);
     (HN > FN -> [_|HSL1] = HSL,abst_slist(HSL1,FSL,NSL);
      [_|FSL1] = FSL,abst_slist(HSL,FSL1,NSL))).

abst_slist([],[],[]).

abst_slist([(XS1,XS2)|HT],[(YS1,YS2)|FT],[(S1,S2)|NT]) :-
    abst_slist([XS1],[YS1],[S1]),
    abst_slist([XS2],[YS2],[S2]),
    !,abst_slist(HT,FT,NT).

abst_slist([HS|HT],[FS|FT],[S|NT]) :-
    abst_sort(HS,S),
    abst_sort(FS,S),
    !,abst_slist(HT,FT,NT).

abst_slist([XL|HT],[YL|FT],[NL|NT]) :-
    abst_list(XL,YL,NL),
    !,abst_slist(HT,FT,NT).

abst_slist([X|HT],[Y|FT],NT) :-
    abst_slist(HT,FT,NT).
```

```
abst_list([],_,[]).

abst_list([HS|XL],YL,[S|NL]) :-
    abst_sort(HS,S),
    member(FS,YL),
    abst_sort(FS,S),!,
    abst_list(XL,YL,NL).

abst_list([X|T],YL,NL) :-
    abst_list(T,YL,NL).

abst_sort(S,S).

abst_sort(S,OS) :-
    S is_a_s AS,
    abst_sort(AS,OS).

member(X,[X|_]).

member(X,[_|T]) :-
    member(X,T).

getnewclause(NSL,C) :-
    write('Abstract Predicate Name? : '),
    read(Pred),
    PC =.. [Pred|NSL],
    write('Predicate Pole ? :'),
    read(Pole),
    (n_pole(Pole) -> C = not PC ; C = PC),
    ttynl,write('Abstract Clause : '),ttynl,
    write(C),ttynl,ttynl.

n_pole(not).
n_pole(minus).
n_pole('-').

gethierarchy(HSL,FSL,NSL) :-
    write('Generated Hierarchy : '),nl,
    write(HSL),write(','),ttynl,
    write(FSL),ttynl,
```

```
    tab(5),write('--->'),tab(2),write(NSL),ttynl,ttynl,
    abd_call(retract(facts(_)),[],[]),
    abd_call(retract(dates(_)),[],[]).

modifyhier(HSL,FSL,NSL) :-
    [HSL|FSLs] is_a_c NSL,
    member(FSL,FSLs),!.

modifyhier(HSL,FSL,NSL) :-
    (functor(HSL,not,_) -> HSL = not HSLz;HSL = HSLz),
    HSLz  =.. [HPred|HTL],
    [HSL2|FSLs] is_a_c NSL,
    (functor(HSL2,not,_) ->
     functor(HSL,not,_),HSL2 = not HSL2z;
     functor(HSL,FH,_),FH \== not,HSL2 = HSL2z),
    HSL2z =.. [HPred|HTL2],
    checkana(FSL,FSLs),!,
    abst_slist(HTL,HTL2,NHTL),
    NHSL =.. [HPred|NHTL],
    (functor(HSL,not,_) -> NHSLz = not NHSL;NHSLz = NHSL),
    (member(FSL,FSLs) -> FSLss = FSLs;append(FSLs,[FSL],FSLss)),
    retract([HSL2|FSLs] is_a_c NSL),
    assert([NHSLz|FSLss] is_a_c NSL).

modifyhier(HSL,FSL,NSL) :-
    (functor(HSL,not,_) -> HSL = not HSLz;HSL = HSLz),
    HSLz  =.. [HPred|HTL],
    [HSL2|FSLs] is_a_c ONSL,
    (functor(HSL2,not,_) ->
     functor(HSL,not,_),HSL2 = not HSL2z;
     functor(HSL,FH,_),FH \== not,HSL2 = HSL2z),
    HSL2z =.. [HPred|HTL2],
    NSL \== ONSL,
    checkana(FSL,FSLs),!,
    abst_slist(HTL,HTL2,NHTL),
    NHSL =.. [HPred|NHTL],
    (functor(HSL,not,_) -> NHSLz = not NHSL;NHSLz = NHSL),
    choiceabs(ONSL,NSL,NNSL),
    (member(FSL,FSLs) -> FSLss = FSLs;append(FSLs,[FSL],FSLss)),
    retract([HSL2|FSLs] is_a_c ONSL),
```

```
    assert([NHSLz|FSLss] is_a_c NNSL).

modifyhier(HSL,FSL,NSL) :-
    assert([HSL,FSL] is_a_c NSL).

checkana(FSL,[]).

checkana(FSL,[FSL|FSLT]) :-
    !,checkana(FSL,FSLT).

checkana(FSL,[FSLH|FSLT]) :-
    write('"'),write(FSL),write('"'),tab(2),ttynl,
    write('"'),write(FSLH),write('"'),tab(2),ttynl,
    write('These facts are related? '),
    read(Agree),
    ttynl,
    agree(Agree),
    checkana(FSL,FSLT).

agree('yes').
agree('y').
agree('ok').

choiceabs(ONSL,NSL,NNSL) :-
    write('1. "'),write(ONSL),write('"'),tab(2),ttynl,
    write('2. "'),write(NSL),write('"'),tab(2),ttynl,
    write('Which abstract is suitable? '),
    read(Choice),
    ttynl,
    (choice(Choice) -> NNSL = ONSL ; NNSL = NSL).

choice(1).
choice('one').
choice('first').

printhiers :-
    findall(X is_a_c Y,X is_a_c Y,List),
    write('Analogical Clause Hierarchy :'),ttynl,
    printhiers(List),
    tell('hier'),
```

```
    write(List),write('.'),nl,
    told.

printhiers([]).

printhiers([X is_a_c Y|T]) :-
    printhier(X),
    tab(5),write('--->'),tab(2),write(Y),nl,nl,
    printhiers(T).

printhier([]).

printhier([H|T]) :-
    write(H),nl,
    printhier(T).

append([],L,L).

append([H|L1],L2,[H|L3]) :-
    append(L1,L2,L3).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.7　モジュール ‘ana.pl’

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
readhiers :-
     see('hier'),
     read(List),
     readhiers(List),
     seen.

readhiers([]).

readhiers([H|T]) :-
    assert(H),
    readhiers(T).

anas([],Facts,[]).
```

```
anas([Hs|Ts],Facts,NTs) :-
    ana(Hs,Facts,[]),!,
    anas(Ts,Facts,NTs).

anas([Hs|Ts],Facts,[NHs|NTs]) :-
    ana(Hs,Facts,NHs),
    anas(Ts,Facts,NTs).

ana([],Facts,[]).

ana([H|T],Facts,T2) :-
    (functor(H,not,_) -> H = not Hz;H = Hz),
    Hz =.. [HPred|HTerm],
    [HC|FCs] is_a_c AC,
    (functor(HC,not,_) ->
     functor(H,not,_),HC = not HCz;
     functor(H,FH,_),FH \== not,HC = HCz),
    HCz =.. [HPred|HCTerm],
    member(FC,FCs),
    (functor(FC,not,_) -> FC = not FCz;FC = FCz),
    FCz =.. [FPred|FCTerm],
    member(F,Facts),
    (functor(F,not,_) ->
     functor(FC,not,_),F = not Fz;
     functor(FC,FFC,_),FFC \== not,F = Fz),
    Fz =.. [FPred|FTerm],
    delvalue(HTerm,HTerm2),
    H2z =.. [HPred|HTerm2],
    (functor(H,not,_) -> H2 = not H2z;H2 = H2z),
    anah(H2,HC),
    delvalue(FTerm,FTerm2),
    F2z =.. [FPred|FTerm2],
    (functor(F,not,_) -> F2 = not F2z;F2 = F2z),
    anaf(F2,FC),!,
    ana(T,Facts,T2).

ana([H|T],Facts,[H|NT]) :-
    ana(T,Facts,NT).

anah(H,H) :- !.
```

```
anah(H,HC) :-
    write('"'),write(H),write('"'),ttynl,
    write('"'),write(HC),write('"'),ttynl,
    write('These Hypotheses are analogical? '),
    read(Agree),
    ttynl,
    agree(Agree).

anaf(F,F) :- !.

anaf(F,FC) :-
    write('"'),write(F),write('"'),ttynl,
    write('"'),write(FC),write('"'),ttynl,
    write('These Facts are analogical? '),
    read(Agree),
    ttynl,
    agree(Agree).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

C.2.8　事実集合モジュール

本研究のシステムでは、事実集合モジュールは、設例 (事例) ごとに別々のファイルとして用意していて、ユーザが法的問題を解くための質問を与えた時に、ユーザが指定することによって読み込むようになっている。以下に、モジュールをそれぞれの設例に対して示す。以下に出てくる宣言 'pred' は、アブダクション可能な述語の定義である。

最初に、契約の成立に関する設例のモジュールについて示す。

設例 6

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960409]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
```

```
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,construction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
revocation_offer((19960409:date,19960409:date),(a:offeror,b:offeree),[telephone:instantaneous_communication,construction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act),
accept((19960409:date,19960409:date),(b:offeree,a:offeror),[telephone:instantaneous_communication,construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
]).

:- pred irrevocable_offer(+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 8

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960409]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,construction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
fixed_accept_period(19960430:period,(a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
revocation_offer((19960409:date,19960409:date),(a:offeror,b:offeree),[telephone:instantaneous_communication,construction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act),
accept((19960409:date,19960409:date),(b:offeree,a:offeror),[telephone:instantaneous_communication,construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
]).

:- pred irrevocable_offer(+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 9

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960409]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
contract_sale((b:offeror,c:offeree),[construction_machine:goods,12000:definite_price,1:
quantity],contract_sale:quasi_juristic_act),
revocation_offer((19960409:date,19960409:date),(a:offeror,b:offeree),[telephone:instant
aneous_communication,construction_machine:goods,10000:definite_price,1:quantity],offer:
juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:qua
ntity],accept:juristic_act),
accept((19960409:date,19960409:date),(b:offeree,a:offeror),[telephone:instantaneous_com
munication,construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_
act)
]).

:- pred reasonable_irrevocable(+,+,+) is (det,abd).
:- pred rely_on_offer(+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 13

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960415,19960430,19960501]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
fixed_accept_period(19960430:period,(a:offeror,b:offeree),[construction_machine:goods,1
0000:definite_price,1:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:qua
```

```
ntity],accept:juristic_act),
accept((19960415:date,19960501:date),(b:offeree,a:offeror),[letter:communication,constr
uction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
]).

:- pred normal_accept_deliver(+,+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 14

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960415,19960416]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:qua
ntity],accept:juristic_act),
pay_bank_account((19960415:date,19960416:date),(b:offeree,a:offeror),[bank_account:tran
sportation,10000:definite_price],pay:quasi_juristic_act)
]).

:- pred custom(+,+,+) is (det,abd).
:- pred indicate_accept_conduct(+,+,+) is (det,abd).
:- pred reasonable_deliver_accept_period(+,+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 14a

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960415,19960416]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
```

```
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:qua
ntity],accept:juristic_act),
mail_check((19960415:date,19960416:date),(b:offeree,a:offeror),[mail:transportation,100
00:definite_price],pay:quasi_juristic_act)
]).

:- pred custom(+,+,+) is (det,abd).
:- pred indicate_accept_conduct(+,+,+) is (det,abd).
:- pred reasonable_deliver_accept_period(+,+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 15

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960508,19960515]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:qua
ntity],accept:juristic_act),
accept((19960508:date,19960515:date),(b:offeree,a:offeror),[letter:communication,constr
uction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
]).

:- pred irrational_deliver_accept_period(+,+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 16

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960410,19960417]).

facts([
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
```

```
ction_machine:goods,1:quantity],offer:juristic_act),
not definite_price((a:offeror,b:offeree),[construction_machine:goods,1:quantity],offer:
juristic_act),
fixed_accept_period(19960430:period,(a:offeror,b:offeree),[construction_machine:goods,1
:quantity],offer:juristic_act),
acceptance((b:offeree,a:offeror),[construction_machine:goods,1:quantity],accept:juristi
c_act),
accept((19960410:date,19960417:date),(b:offeree,a:offeror),[letter:communication,constr
uction_machine:goods,1:quantity],accept:juristic_act)
]).

:- pred sufficiently_definite_offer(+,+,+) is (det,abd).
:- pred intension_bound(+,+,+) is (det,abd).
```

* ——————————— end here ——————————— *

設例 18

* ——————————— from here ——————————— *

```
dates([19960401,19960408,19960409,19960416,19960430,19960501,19960505]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
fixed_accept_period(19960430:period,(a:offeror,b:offeree),[construction_machine:goods,1
0000:definite_price,1:quantity],offer:juristic_act),
accept((19960409:date,19960416:date),(b:offeree,a:offeror),[letter:communication,constr
uction_machine:goods,9000:changed_price,1:quantity],accept:juristic_act),
deliver_goods(19960501:date,(a:offeror,b:offeree),[construction_machine:goods,1:quantit
y],deliver_goods:juristic_act),
demand_pay(19960505:date,(a:offeror,b:offeree),[10000:definite_price],depand_pay:quasi_
juristic_act)
]).

:- pred indicate_accept_conduct(+,+,+) is (det,abd).
:- pred materially_alter(+,+,+) is (det,abd).
```

* ——————————— end here ——————————— *

設例 18a

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960409,19960416,19960430,19960501,19960505]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
fixed_accept_period(19960430:period,(a:offeror,b:offeree),[construction_machine:goods,1
0000:definite_price,1:quantity],offer:juristic_act),
accept((19960409:date,19960416:date),(b:offeree,a:offeror),[letter:communication,constr
uction_machine:goods,9000:changed_price,1:quantity],accept:juristic_act),
deliver_goods(19960501:date,(a:offeror,b:offeree),[construction_machine:goods,1:quantit
y],deliver_goods:juristic_act),
bill_pay(19960505:date,(a:offeror,b:offeree),[10000:definite_price],depand_pay:quasi_ju
ristic_act)
]).

:- pred indicate_accept_conduct(+,+,+) is (det,abd).
:- pred materially_alter(+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

設例 19

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960409,19960416,19960430,19960501]).

facts([
proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quant
ity],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,constru
ction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act),
fixed_accept_period(19960430:period,(a:offeror,b:offeree),[construction_machine:goods,1
0000:definite_price,1:quantity],offer:juristic_act),
accept((19960409:date,19960416:date),(b:offeree,a:offeror),[letter:communication,constr
uction_machine:goods,10000:definite_price,1:quantity,solve_dispute:agreement],accept:ju
ristic_act),
```

```
added_mension((b:offeree,a:offeror),[solve_dispute:agreement],accept:juristic_act),
deliver_goods(19960501:date,(a:offeror,b:offeree),[construction_machine:goods,1:quantit
y],deliver_goods:juristic_act)
]).

:- pred indicate_accept_conduct(+,+,+) is (det,abd).
:- pred materially_alter(+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

次に、契約の解除に関する設例のモジュールについて示す。

設例 7f

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
dates([19960401,19960408,19960409,19960501,19960510,19960531,19960605,19960810,19960901
,19961001,19961010]).

facts([
proposal((a:offeror,b:offeree),[farm_machine:goods,50000:definite_price,1:quantity,amer
ican_cargo_ship:means],offer:juristic_act),
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,farm_ma
chine:goods,50000:definite_price,1:quantity,american_cargo_ship:means],offer:juristic_a
ct),
accept((19960409:date,19960409:date),(b:offeree,a:offeror),[telephone:instantaneous_com
munication,farm_machine:goods,50000:definite_price,1:quantity,japanese_cargo_ship:chang
e_means],accept:juristic_act),
alter((b:offeree,a:offeror),[farm_machine:goods,50000:definite_price,1:quantity,japanes
e_cargo_ship:change_means],accept:juristic_act),
deliver_goods(19960531:date,(a:offeror,b:offeree),[farm_machine:goods,50000:definite_pr
ice,1:quantity],deliver:juristic_act),
transport((19960501:date,19960531:date),(a:offeror,b:offeree),[japanese_cargo_ship:tran
sportation,farm_machine:goods,50000:definite_price,1:quantity],transport:juristic_act),
check(19960605:date,b:offeree,[farm_machine:goods],check:quasi_juristic_act),
pay((19960510:date,19960510:date),(b:offeree,a:offeror),[50000:definite_price],pay:quas
i_juristic_act),
out_of_order(19960810:date,farm_machine:thing,[farm_machine:goods],out_of_order:event),
demand_repair((19960901:date,19960901:date),(b:offeree,a:offeror),[farm_machine:goods],
demand_repair:juristic_act),
```

```
obligation_added_period(19961001:period,(b:offeree,a:offeror),[farm_machine:goods,50000
:definite_price,1:quantity],obligation_added_period:juristic_act),
not repair(19961001:period,(a:offeror,b:offeree),[farm_machine:goods],not_repair:quasi_
juristic_act),
avoid_contract((19961010:date,19961010:date),(b:offeree,a:offeror),[oral:communication,
farm_machine:goods,50000:definite_price,1:quantity],avoid_contract:juristic_act)
]).

:- pred materially_alter(+,+,+) is (det,abd).
:- pred indicate_accept_conduct(+,+,+) is (det,abd).

:- pred deliver_documents(+,+,+,+) is (det,abd).
:- pred satisfy_condition_of_contract(+,+,+) is (det,abd).
:- pred suitable_goods_of_contract(+,+,+) is (det,abd).
:- pred important_breach_of_contract(+,+,+) is (det,abd).
:- pred reasonable_demand_period(+,+,+,+) is (det,abd).
:- pred reasonable_situation_goods(+,+,+) is (det,abd).
```

* ——————————— end here ——————————— *

設例 7g

* ——————————— from here ——————————— *

```
dates([19960401,19960408,19960409,19960501,19960510,19960531,19960605,19960810,19960901
,19961001,19961010]).

facts([
offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,farm_ma
chine:goods,50000:main_price,1:quantity,american_cargo_ship:means],offer:juristic_act),
not definite_price((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y],offer:juristic_act),
accept((19960409:date,19960409:date),(b:offeree,a:offeror),[telephone:instantaneous_com
munication,farm_machine:goods,50000:main_price,1:quantity,japanese_cargo_ship:change_me
ans],accept:juristic_act),
alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:quantity,japanese_ca
rgo_ship:change_means],accept:juristic_act),
deliver_goods(19960531:date,(a:offeror,b:offeree),[farm_machine:goods,50000:main_price,
1:quantity],deliver:juristic_act),
transport((19960501:date,19960531:date),(a:offeror,b:offeree),[japanese_cargo_ship:tran
sportation,farm_machine:goods,50000:main_price,1:quantity],transport:juristic_act),
```

```
check(19960605:date,b:offeree,[farm_machine:goods],check:quasi_juristic_act),
pay((19960510:date,19960510:date),(b:offeree,a:offeror),[50000:main_price],pay:quasi_ju
ristic_act),
out_of_order(19960810:date,farm_machine:thing,[farm_machine:goods],out_of_order:event),
demand_repair((19960901:date,19960901:date),(b:offeree,a:offeror),[farm_machine:goods],
demand_repair:juristic_act),
obligation_added_period(19961001:period,(b:offeree,a:offeror),[farm_machine:goods,50000
:main_price,1:quantity],obligation_added_period:juristic_act),
not repair(19961001:period,(a:offeror,b:offeree),[farm_machine:goods],not_repair:quasi_
juristic_act),
avoid_contract((19961010:date,19961010:date),(b:offeree,a:offeror),[oral:communication,
farm_machine:goods,50000:main_price,1:quantity],avoid_contract:juristic_act)
]).

:- pred intension_bound(+,+,+) is (det,abd).
:- pred sufficiently_definite_offer(+,+,+) is (det,abd).

:- pred materially_alter(+,+,+) is (det,abd).
:- pred indicate_accept_conduct(+,+,+) is (det,abd).

:- pred deliver_documents(+,+,+,+) is (det,abd).
:- pred satisfy_condition_of_contract(+,+,+) is (det,abd).
:- pred suitable_goods_of_contract(+,+,+) is (det,abd).
:- pred important_breach_of_contract(+,+,+) is (det,abd).
:- pred reasonable_demand_period(+,+,+,+) is (det,abd).
:- pred reasonable_situation_goods(+,+,+) is (det,abd).
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

付録 D

実装システムの出力例

本研究では、**知識階層生成に関する実験**と**類推解釈実験**を行った。その実行結果について以下に示す。これらの実行結果に出力されているコメント行は、後から付けたものである。/* ... */で示したコメントはユーザによる処理に関するものであり、% ... %で示したコメントはシステムによる処理に関するものである。

D.1 知識階層生成に関する実験

知識階層生成に関する実験では、**知識階層生成実験**と**知識階層付加実験**を行ったが、双方とも知識階層生成機構を用いて実験を行っており、その手続きも同じである。ここでは、知識階層生成実験の一部をとりあげることにする。節階層生成の際に生成される抽象化節の述語名は$'abs'$に統一し、述語の前には否定記号をつけないことにする。

D.1.1 節階層の生成例

以下に示すのは、契約の成立に関する設例 14(B.1 節を参照) を用いて節階層を生成した例である。ここでは、節階層が全く生成されていない状態で実験を行っている。

* ——————————————— from here ——————————————— *

```
| ?- qana(qc14).                          /* 契約の成立に関する質問 */
Select Facts: case14.                  /* 事例 (事実集合) として設例 14 を選択 */
loading done.
```

```
Generated Hypotheses:                                          % 仮説集合 %

No. 1.
1: reasonable_deliver_accept_period(19960416:period,(b:offeree,a:offeror),[construction
_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
2: indicate_accept_conduct((b:offeree,a:offeror),[construction_machine:goods,10000:defi
nite_price,1:quantity],accept:juristic_act)
3: custom((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quan
tity],custom:juristic_act)


Facts:                                                         % 事実集合 %

1: proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:qu
antity],offer:juristic_act)
2: offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,cons
truction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act)
3: acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:
quantity],accept:juristic_act)
4: pay_bank_account((19960415:date,19960416:date),(b:offeree,a:offeror),[bank_account:t
ransportation,10000:definite_price],pay:quasi_juristic_act)

Select Hypotheses containing Analogical Hypothesis: 1.        /* 仮説集合の選択 */

No. 1.:
1: reasonable_deliver_accept_period(19960416:period,(b:offeree,a:offeror),[construction
_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
2: indicate_accept_conduct((b:offeree,a:offeror),[construction_machine:goods,10000:defi
nite_price,1:quantity],accept:juristic_act)
3: custom((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quan
tity],custom:juristic_act)

Select Analogical Hypothesis: 2.                                /* 仮説の選択 */

Selected Hypothesis :
indicate_accept_conduct((b:offeree,a:offeror),[construction_machine:goods,10000:definit
e_price,1:quantity],accept:juristic_act)

Select Analogical Fact: 4.
```

```
Selected Fact :
pay_bank_account((19960415:date,19960416:date),(b:offeree,a:offeror),[bank_account:tran
sportation,10000:definite_price],pay:quasi_juristic_act)

Abstract Predicate Name? : abs.                          /* 述語名は abs にする */
Predicate Pole ? :plus.                                  /* 否定記号はつけない */

Abstract Clause :                                         % 生成された抽象化節 %
abs((offeree,offeror),[definite_price],lawful_act)

Generated Hierarchy :                                      % 生成された節階層 %
indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act),
pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_ju
ristic_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)

Analogical Clause Hierarchy :                              % 更新後の節階層集合 %
indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act),
pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_ju
ristic_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)


yes
```

* ——————————— end here ——————————— *

D.1.2　節階層同士の相互作用例

次に示すのは、契約の成立に関する設例 14(B.1 節を参照) を用いて節階層を生成した後に、契約の成立に関する設例 14a(B.1 節を参照) を用いて節階層を生成した例である。ここでは、生成される節階層が、既存の節階層と関連がありそうだとシステムが判断し、ユーザにその旨についての質問をしてくる。ここでは、システムからの質問に対して、ユーザが「関連がある」と判断していて、システムは既存の節階層集合の再構成を行っている。

```
* ____________________________ from here ____________________________ *

| ?- qana(qc14).                                  /* 契約の成立に関する質問 */
Select Facts: case14a.                        /* 事例 (事実集合) として設例 14a を選択 */
loading done.

Generated Hypotheses:                                          % 仮説集合 %

No. 1.
1: reasonable_deliver_accept_period(19960416:period,(b:offeree,a:offeror),[construction
_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
2: indicate_accept_conduct((b:offeree,a:offeror),[construction_machine:goods,10000:defi
nite_price,1:quantity],accept:juristic_act)
3: custom((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quan
tity],custom:juristic_act)


Facts:                                                         % 事実集合 %

1: proposal((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:qu
antity],offer:juristic_act)
2: offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,cons
truction_machine:goods,10000:definite_price,1:quantity],offer:juristic_act)
3: acceptance((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:
quantity],accept:juristic_act)
4: mail_check((19960415:date,19960416:date),(b:offeree,a:offeror),[mail:transportation,
10000:definite_price],pay:quasi_juristic_act)

Select Hypotheses containing Analogical Hypothesis: 1.        /* 仮説集合の選択 */

No. 1.:
1: reasonable_deliver_accept_period(19960416:period,(b:offeree,a:offeror),[construction
_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
```

```
2: indicate_accept_conduct((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)
3: custom((a:offeror,b:offeree),[construction_machine:goods,10000:definite_price,1:quantity],custom:juristic_act)

Select Analogical Hypothesis: 2.                                    /* 仮説の選択 */

Selected Hypothesis :
indicate_accept_conduct((b:offeree,a:offeror),[construction_machine:goods,10000:definite_price,1:quantity],accept:juristic_act)

Select Analogical Fact: 4.                                          /* 事実の選択 */

Selected Fact :
mail_check((19960415:date,19960416:date),(b:offeree,a:offeror),[mail:transportation,10000:definite_price],pay:quasi_juristic_act)

Abstract Predicate Name? : abs.                                     /* 述語名は abs にする */
Predicate Pole ? :plus.                                            /* 否定記号はつけない */

Abstract Clause :                                                    % 生成された抽象化節 %
abs((offeree,offeror),[definite_price],lawful_act)

Generated Hierarchy :                                                % 生成された節階層 %
indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act),
mail_check((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)

"mail_check((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic_act)"
"pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic_act)"
These facts are related? y.                          /* ユーザが節階層同士の関連を認める */

Analogical Clause Hierarchy :                                        % 更新後の節階層集合 %
indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act)
pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic_act)
mail_check((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic
```

```
_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)


yes
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

また、システムからの質問に対して、ユーザが「関連がない」と判断した場合、節階層集合は次のように生成される。

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
"mail_check((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristi
c_act)"
"pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_j
uristic_act)"
These facts are related? n.                /* ユーザが節階層同士の関連を認めない */

Analogical Clause Hierarchy :                                    % 更新後の節階層集合 %
indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act)
pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_ju
ristic_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)

indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act)
mail_check((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic
_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)


yes
```

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ end here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

D.1.3　知識階層生成実験で生成された節階層

知識階層生成実験では、契約の成立に関する設例のうちで、不完全知識が含まれる 11 の設例について実験を行っている。その結果生成された節階層集合を以下に示す。

* ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ from here ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ *

```
Analogical Clause Hierarchy :                                    % 更新後の節階層集合 %
irrevocable_offer((offeror,offeree),[goods,definite_price,quantity],juristic_act),
fixed_accept_period(period,(offeror,offeree),[goods,definite_price,quantity],juristic_a
ct)
     --->  abs((offeror,offeree),[goods,definite_price,quantity],juristic_act)

rely_on_offer((offeror,offeree),[goods,definite_price,quantity],juristic_act),
contract_sale((offeror,offeree),[goods,definite_price,quantity],quasi_juristic_act)
     --->  abs((offeror,offeree),[goods,definite_price,quantity],lawful_act)

normal_accept_deliver((date,date),(offeree,offeror),[communication,goods,definite_price
,quantity],juristic_act),
accept((date,date),(offeree,offeror),[communication,goods,definite_price,quantity],juri
stic_act)
     --->  abs((date,date),(offeree,offeror),[communication,goods,definite_price,quanti
           ty],juristic_act)

indicate_accept_conduct((offeree,offeror),[goods,definite_price,quantity],juristic_act),
pay_bank_account((date,date),(offeree,offeror),[transportation,definite_price],quasi_ju
ristic_act)
mail_check((date,date),(offeree,offeror),[transportation,definite_price],quasi_juristic
_act)
     --->  abs((offeree,offeror),[definite_price],lawful_act)

sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act),
not definite_price((offeror,offeree),[goods,quantity],juristic_act)
     --->  abs((offeror,offeree),[goods,quantity],juristic_act)

indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act),
deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)
     --->  abs([goods,quantity],juristic_act)

not materially_alter((offeree,offeror),[goods,definite_price,quantity,agreement],jurist
ic_act),
added_mension((offeree,offeror),[agreement],juristic_act)
     --->  abs((offeree,offeror),[agreement],juristic_act)


yes
```

* ———————————— end here ———————————— *

D.2　類推解釈に関する実験

類推解釈に関する実験では、類推解釈機構を用いて実験を行った。実験で用いた設例は、契約の解除に関する設例である 7f と 7g(B.2 節を参照) である。ここでは、設例 7g を用いて類推解釈を行った例を示す。この実験においては、知識階層生成実験によって生成された節階層があらかじめ生成されている状態で実験を行っている。

また、本来の類推解釈機構はユーザによる選択を必要としないが、どの節階層が類推解釈に用いられたかを確認するために、ユーザによる選択部分を設けている。

* ———————————— from here ———————————— *

```
| ?- qana(qv7g).                                    /* 契約の解除に関する質問 */
Select Facts: case7g.                          /* 事例 (事実集合) として設例 7g を選択 */
loading done.

Generated Hypotheses:                                                  % 仮説集合 %

No. 1.
1: not deliver_documents(19960531:date,(a:offeror,b:offeree),[farm_machine:goods,50000:
main_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: indicate_accept_conduct((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1
:quantity,japanese_cargo_ship:change_means],accept:juristic_act)
4: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
5: sufficiently_definite_offer((a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity,american_cargo_ship:means],offer:juristic_act)

No. 2.
1: not satisfy_condition_of_contract((a:offeror,b:offeree),[farm_machine:goods,50000:ma
in_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: indicate_accept_conduct((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1
:quantity,japanese_cargo_ship:change_means],accept:juristic_act)
4: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
```

y,american_cargo_ship:means],offer:juristic_act)
5: sufficiently_definite_offer((a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity,american_cargo_ship:means],offer:juristic_act)

No. 3.
1: not suitable_goods_of_contract((a:offeror,b:offeree),[farm_machine:goods,50000:main_
price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: indicate_accept_conduct((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1
:quantity,japanese_cargo_ship:change_means],accept:juristic_act)
4: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
5: sufficiently_definite_offer((a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity,american_cargo_ship:means],offer:juristic_act)

No. 4.
1: reasonable_demand_period(19961010:date,(b:offeree,a:offeror),[farm_machine:goods,500
00:main_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: indicate_accept_conduct((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1
:quantity,japanese_cargo_ship:change_means],accept:juristic_act)
4: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
5: sufficiently_definite_offer((a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity,american_cargo_ship:means],offer:juristic_act)

No. 5.
1: reasonable_situation_goods((a:offeror,b:offeree),[farm_machine:goods,50000:main_pric
e,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: indicate_accept_conduct((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1
:quantity,japanese_cargo_ship:change_means],accept:juristic_act)
4: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
5: sufficiently_definite_offer((a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity,american_cargo_ship:means],offer:juristic_act)

```
No. 6.
1: not deliver_goods(19961001:period,(a:offeror,b:offeree),[farm_machine:goods,50000:ma
in_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: indicate_accept_conduct((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1
:quantity,japanese_cargo_ship:change_means],accept:juristic_act)
4: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
5: sufficiently_definite_offer((a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity,american_cargo_ship:means],offer:juristic_act)


Facts:                                                    % 事実集合 %

1: offer((19960401:date,19960408:date),(a:offeror,b:offeree),[letter:communication,farm
_machine:goods,50000:main_price,1:quantity,american_cargo_ship:means],offer:juristic_ac
t)
2: not definite_price((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quan
tity],offer:juristic_act)
3: accept((19960409:date,19960409:date),(b:offeree,a:offeror),[telephone:instantaneous_
communication,farm_machine:goods,50000:main_price,1:quantity,japanese_cargo_ship:change
_means],accept:juristic_act)
4: alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:quantity,japanese
_cargo_ship:change_means],accept:juristic_act)
5: deliver_goods(19960531:date,(a:offeror,b:offeree),[farm_machine:goods,50000:main_pri
ce,1:quantity],deliver:juristic_act)
6: transport((19960501:date,19960531:date),(a:offeror,b:offeree),[japanese_cargo_ship:t
ransportation,farm_machine:goods,50000:main_price,1:quantity],transport:juristic_act)
7: check(19960605:date,b:offeree,[farm_machine:goods],check:quasi_juristic_act)
8: pay((19960510:date,19960510:date),(b:offeree,a:offeror),[50000:main_price],pay:quasi
_juristic_act)
9: out_of_order(19960810:date,farm_machine:thing,[farm_machine:goods],out_of_order:even
t)
10: demand_repair((19960901:date,19960901:date),(b:offeree,a:offeror),[farm_machine:goo
ds],demand_repair:juristic_act)
11: obligation_added_period(19961001:period,(b:offeree,a:offeror),[farm_machine:goods,5
0000:main_price,1:quantity],obligation_added_period:juristic_act)
12: not repair(19961001:period,(a:offeror,b:offeree),[farm_machine:goods],not_repair:qu
asi_juristic_act)
```

```
13: avoid_contract((19961010:date,19961010:date),(b:offeree,a:offeror),[oral:communicat
ion,farm_machine:goods,50000:main_price,1:quantity],avoid_contract:juristic_act)

"indicate_accept_conduct((offeree,offeror),[goods,main_price,quantity,change_means],jur
istic_act)"
"indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act)"
These Hypotheses are analogical? y.                    /* 仮説「承諾意図」がマッチ */

"deliver_goods(date,(offeror,offeree),[goods,main_price,quantity],juristic_act)"
"deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.                         /* 事実「物品引渡」がマッチ */

"sufficiently_definite_offer((offeror,offeree),[goods,main_price,quantity,means],jurist
ic_act)"
"sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act)"
These Hypotheses are analogical? y.              /* 仮説「十分明確な申込」がマッチ */

"not definite_price((offeror,offeree),[goods,main_price,quantity],juristic_act)"
"not definite_price((offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.                  /* 事実「明確な申込でない」がマッチ */

"indicate_accept_conduct((offeree,offeror),[goods,main_price,quantity,change_means],jur
istic_act)"
"indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act)"
These Hypotheses are analogical? y.

"deliver_goods(date,(offeror,offeree),[goods,main_price,quantity],juristic_act)"
"deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"sufficiently_definite_offer((offeror,offeree),[goods,main_price,quantity,means],jurist
ic_act)"
"sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act)"
These Hypotheses are analogical? y.

"not definite_price((offeror,offeree),[goods,main_price,quantity],juristic_act)"
"not definite_price((offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"indicate_accept_conduct((offeree,offeror),[goods,main_price,quantity,change_means],jur
```

```
istic_act)"
"indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act)"
These Hypotheses are analogical? y.

"deliver_goods(date,(offeror,offeree),[goods,main_price,quantity],juristic_act)"
"deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"sufficiently_definite_offer((offeror,offeree),[goods,main_price,quantity,means],jurist
ic_act)"
"sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act)"
These Hypotheses are analogical? y.

"not definite_price((offeror,offeree),[goods,main_price,quantity],juristic_act)"
"not definite_price((offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"indicate_accept_conduct((offeree,offeror),[goods,main_price,quantity,change_means],jur
istic_act)"
"indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act)"
These Hypotheses are analogical? y.

"deliver_goods(date,(offeror,offeree),[goods,main_price,quantity],juristic_act)"
"deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"sufficiently_definite_offer((offeror,offeree),[goods,main_price,quantity,means],jurist
ic_act)"
"sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act)"
These Hypotheses are analogical? y.

"not definite_price((offeror,offeree),[goods,main_price,quantity],juristic_act)"
"not definite_price((offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"indicate_accept_conduct((offeree,offeror),[goods,main_price,quantity,change_means],jur
istic_act)"
"indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act)"
These Hypotheses are analogical? y.
```

```
"deliver_goods(date,(offeror,offeree),[goods,main_price,quantity],juristic_act)"
"deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"sufficiently_definite_offer((offeror,offeree),[goods,main_price,quantity,means],jurist
ic_act)"
"sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act)"
These Hypotheses are analogical? y.

"not definite_price((offeror,offeree),[goods,main_price,quantity],juristic_act)"
"not definite_price((offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"indicate_accept_conduct((offeree,offeror),[goods,main_price,quantity,change_means],jur
istic_act)"
"indicate_accept_conduct((offeree,offeror),[goods,price,quantity],juristic_act)"
These Hypotheses are analogical? y.

"deliver_goods(date,(offeror,offeree),[goods,main_price,quantity],juristic_act)"
"deliver_goods(date,(offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

"sufficiently_definite_offer((offeror,offeree),[goods,main_price,quantity,means],jurist
ic_act)"
"sufficiently_definite_offer((offeror,offeree),[goods,quantity],juristic_act)"
These Hypotheses are analogical? y.

"not definite_price((offeror,offeree),[goods,main_price,quantity],juristic_act)"
"not definite_price((offeror,offeree),[goods,quantity],juristic_act)"
These Facts are analogical? y.

Generated Hypotheses:                    % 類推解釈されなかった残りの仮説集合 %

No. 1.
1: not deliver_documents(19960531:date,(a:offeror,b:offeree),[farm_machine:goods,50000:
main_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
```

```
No. 2.
1: not satisfy_condition_of_contract((a:offeror,b:offeree),[farm_machine:goods,50000:ma
in_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)

No. 3.
1: not suitable_goods_of_contract((a:offeror,b:offeree),[farm_machine:goods,50000:main_
price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)

No. 4.
1: reasonable_demand_period(19961010:date,(b:offeree,a:offeror),[farm_machine:goods,500
00:main_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)

No. 5.
1: reasonable_situation_goods((a:offeror,b:offeree),[farm_machine:goods,50000:main_pric
e,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)

No. 6.
1: not deliver_goods(19961001:period,(a:offeror,b:offeree),[farm_machine:goods,50000:ma
in_price,1:quantity],deliver:juristic_act)
2: not materially_alter((b:offeree,a:offeror),[farm_machine:goods,50000:main_price,1:qu
antity,japanese_cargo_ship:change_means],accept:juristic_act)
3: intension_bound((a:offeror,b:offeree),[farm_machine:goods,50000:main_price,1:quantit
y,american_cargo_ship:means],offer:juristic_act)
```

```
yes

* ____________________________________ end here ____________________________________ *
```