Concept Explorer*の利用の手引き†

ConExp documentation team　　黒田 航 (Kuroda, Kow)

First Japanese version: 2017/03/30; Revisions: 2017/04/01

1　序

1.1　これは何？

これは “Concept Explorer” (ConExp) と言うツールのバージョン 1.3 で、形式概念分析 (FCA) の研究と調査に必要な基本機能を実装しています。

形式概念分析は束理論の応用例の一つであり、Darmstadt の Rudolf Wille のグループの何人かのメンバーによって 1980 年代初めから開発されて来たものです。これは (FCA で「文脈」と呼ばれる) 属性と対象の単純なカケ合せ表の分析と属性間に存在する依存関係の探索的調査のために利用できます。形式概念分析の詳細については `http://www.math.tudresden.de/~ganter/fba.html` と `http://www.fcahome.org.uk/` を参照して下さい。

ConExp は BSD 風のライセンスの下で公開されています。配布物の中にある license.txt ファイルをお読みください。ConExp で使用されているライブラリは “X.jar” ファイルで、その “X.license” ファイルと “X.readme” ファイルが読めます。

科学的研究で ConExp を利用する場合は、論文 [1] を引用して下さい。

1.2　これで何ができるの？

ConExp は次の機能を提供します:

*`http://conexp.sourceforge.net/index.html` で公開.

†本文書は “Concept Explorer. The User Guide. (September 12, 2006)” (`http://www.comp.dit.ie/pbrowne/compfund2/UserGuide.pdf`) の黒田 航 (Kuroda, Kow) による日本語訳です．Google Translation Toolkit を使って翻訳を進めました．原典の英語が間違っていたり，部分的に破綻していたので，本文書は原典の忠実な日本語訳ではありません．[...] は黒田が補った部分を表わします．いずれ Web ページ `http://conexp.sourceforge.net/users/documentation/` の解説と整合させる予定です.

- 文脈 (context) を編集する
- 文脈から概念束 (concept lattices) を構築する
- 文脈中で真である含意関係の基本要素 (base) を見つける
- 文脈中で真である連想規則の基本要素 (base) を見つける
- 属性探査 (attribution exploration) を [対話的に] 実行する

1.3 歴史を少し

ConExp は最初、2000 年にウクライナ国立工科大学の KPI の教授である Tatyana Taran 博士の指導の下で修士論文の一部として開発されました。その後の数年の間に拡張され、現在は SourceForge のオープンソースのプロジェクト [5] になっています。

2 ConExp の導入

2.1 必要なソフトウェア

ConExp の実行には Java Runtime Environment (JRE) のバージョン 1.4 以降が必要です。普通の環境であれば、JRE の最新バージョン (執筆の時点で 1.5.0) の使用をお勧めします。それが手元にない場合は、次の URL で入手可能です: `http://java.sun.com/j2se/downloads.html`

2.2 導入手順

適当な場所に .zip ファイル、ないしは .tar.gz ファイルの内容を解凍するだけです。ただ Java がその場所から呼びだせる事を確認して下さい (選択されたディレクトリー内でコンソールを開き、そこに “`java -version`” と打ち込んで実行すれば、使用しているシステムで Java が利用できるか確認できます。Java がインストールされている場合は、環境情報と Java ソフトウェアのバージョン情報が表示されます).

3 Concept Explorer で作業する

3.1 作業を始める

適切な起動スクリプト (Windows なら “conexp.bat”、Unix なら “conexp.sh”) を実行します。Unix ではそれを行う前に “conexp.sh” の実行可能属性を設定します。[それには Terminal 上で “chmod +X conexp.sh” を実行します。] コマンドラインから実行する場合は、インストールしたディレクトリに自分がいる事を確認して下さい。

これに代わる方法として、全プラットフォームでコマンドラインから “java -jar conexp.jar” を実行しても良いです。Windows で作業していて Java コンソールを表示したくない場合には、“javaw -jar conexp.jar” を使用して下さい。

ConExp 1.3 で日本語を使用するための工夫 [1)]

ConExp 1.3 のデフォールト設定では .cex ファイルで日本語が使えません (入力時に日本語が表示されていても，保存して開き直すと文字化けています)．この問題を解決するには次の二つの解決法があります:

(1) ConExp の起動を次のコマンドで行う:

```
java -Dfile.encoding=utf-8 -jar conexp.jar
```

(2) OS X なら ConExp アプリの property list を編集して上の情報を追加する．

3.2 Concept Explorer のユーザーインターフェイスの概要

ConExp のユーザーインターフェイスは次の部分で構成されています:

Manu (メニュー)

Main toolbar (主なツールのバー): 常時実行可能な次のアプリケーションの操作ためのボタンをまとめたものです: “Create new document” (文書の新規作成)、“Open (文書を開く)”、“Save (変更を保存)”、“Compute the

[1)] この節の内容は翻訳者の黒田が追加したものです.

number of concepts (概念の数を数える)”、“Compute concept lattice (概念束の計算)”、“Perform attribute exploration (属性探査を実行)”、“Calculate Duquenne-Guigues set of implications (含意の Duquenne-Guigues 集合を計算)”、“Calculate association rules (連想規則を計算)”、文脈から計算される文書の部品 (概念束、含意と連想の規則集合) の更新を選択するための小窓です。

今の所、二種類の更新モードがサポートされています: 変更を受ける部品の除去と変更を受けるコンポーネントの再計算です。一つ目のモードは、文脈に一度に数多くの変更をする場合や文脈の規模が大きい場合に使う場合に向いていますと良いです。このモードでは、変更される部品は消去され、画面に対応する計算操作を呼び出して再計算できます。二つ目のモードでは、変更を受ける部品は、文脈が変更される毎に再計算されます。この方法だと文脈の規模が大きいか濃度が密である場合には計算が長くかかる可能性があります。

Main pane (主な仕切り扉): Main pane には、文書の分類図 (Document tree)、配置オプション (Layout options) の仕切り、それから表示画面の仕切り (View pane) が含まれています。

Document tree (文書の分類図): 使用中の文書の構造を表示し、異なる表示画面 (文脈編集用の画面、束構造の可視化のための (幾つかの) 画面、含意用の画面，連想規則用の画面) の間の行き来を可能にします。

Option pane: ここで表示中の画面に関係するオプションを色々編集できます。

View pane この中に使用中の画面が表示されます。それぞれの処理画面に対応して実行可能な操作の入った道具箱があります。

Status bar: [記載なし]

3.3 文書の新規作成

通常、新しい文書が ConExp で作業を開始した時に作成されます。あるいは、メインツールバーの「新しい文脈」ボタンを押すか、「ファイル」メニューの「新規作成」メニュー項目を選択して、新しい文書を作成することができます。別の文書の新規作成する際、別の文書が先に開かれて変更済みのある場合、ユーザーはそれを保存する (か、新規作成を取り消す) ように指示されます。

3.3.1 既存の文書を開く

ConExp は何種類かのデータ形式で作業できます。ConImp [3] で作成された文脈で作業できます。現在、以下の書式がサポートされています:

.cex: これは ConExp 独自の書式です。これは XML ベースの書式です。これは文脈と束図に関する情報、それに加えて含意規則や連想規則が算出されたかどうかの情報を保持しています。作業の記録が目的なら、この書式の使用をお勧めします。

.cxt: ConImp 文脈データだけ: この形式で保存可能なのは文脈だけです。

.csv: コンマ区切り値の列: 今のところ、この書式で実行できるのは文脈の取り込みだけです。実際の区切りのキーは [コンマでなく] セミコロン (;) です。ファイルの先頭行に属性名の列があり、その行の最初のセルは空です (つまり 属性に attr1 と attr2 を持つ文脈ファイルの先頭行は “;attr1;attr2” になります)。その後の行の行はぞれぞれ、対象名で始まり、0 と 1 の連鎖が [属性の個数分] 続きます。取り込んだ文脈では，値が 1 に設定されているセル [=フィールド] には X 印が付きます。

.oal: 対象-属性対のリスト: 今のところ、この書式で実行可能なのは文脈の読み込みだけです。それぞれの行では、どの対象がどの属性を持つかの情報が指定されます。例えば obj1 が attr2 と attr3 を持つと指定するなら，obj1 のための行の指定は “`obj1:attr2;attr3`” です。

また、過去に作業をしたファイルをまた開く事もできます．それには “File” メニューの下位メニューの “Reopen” に候補に挙がっているファイル名を選択します。

3.4 作業の保存

作業結果を保存するには、“File” メニューにある “Save” や “Save as” の項目を使うか、ツールバーにある “Save file” ボタンを使って下さい。推奨する保存形式は ConExp の自前のファイル形式です。

3.5 文脈の作業

操作の取り消し (Undo) /やり直し (Redo) の支援文脈に対して実行される全操作について、undo (取り消し)/ redo (やりな) の実行が実現されています。Context Editor のツールバー上にある “Undo last action” ボタンを押すと実行された操作を取り消す事ができますし、“Redo last action” を押すと操作をやり直すことができます。

3.5.1 文脈の大きさの変更

文脈の大きさを変更するには、表示窓の左手にある properties を使って、“Object count” と “Attribute count” のパラメータに新しい数値を記入します。

Context Editor ツールバーにある “Add object” や “Add attribute” のボタンを押して新規対象や新規属性を追加する事も可能です。部分的に対象や属性を除去するには、Context Editor 上でそれらを選択し、メニューの “Remove object(s)” や “Remove attribute(s)” を実行して下さい。

3.5.2 文脈を圧縮して表示

文脈の全体像をもっと見通したい時には、Context Editor 画面の特性設定用の仕切り窓にある “Compressed” オプションを選択すればできます。そうすると、文脈表示で列の幅が X に合うように決められて、文脈の全体像の見通しが良くなります。

3.5.3 矢印関係の可視化

矢印関係を可視化するには、“Show arrow relation” の parameter で “show arrow relation” の値を選択します。(矢印関係が何だかわからない場合は、おそらく使う必要がないでしょう。矢印関係についての知りたい場合、本 [2] に当って見て下さい。)

3.6 文脈にデータを入力する

3.6.1 高速な文脈編集

それなりに大きな文脈を入力する必要があるなら、いわゆる「文脈の高速編集」法を使用できます。カーソルが関係指定域のセルにいる時に “x” や “.” のキーを使うだけでそのセルにX印か空白が入力され、カーソルが次のセルに移動します。

3.6.2 選択された領域内での一括変換

一定範囲のセルを選択した後、対象と属性の間に成立する関係の変換が幾つか実行できます。次の変換が提供されています:

Fill selection: これは選択範囲の成立関係をX印で埋めます。

Clear selection: これは選択した領域の成立関係の指定を削除します。

Inverse selection: これは選択した範囲の成立関係の値を反転させます。

これらのすべての変換は、Context Editor 上で飛び出しメニューから相応のコマンドを使用して実行できます。

3.6.3 文脈の操作

文脈には以下の操作が実行できます:

object clarification (対象の明確化): 文脈中の属性の値が等しい複数の対象を (文脈中で最初に現われた) 対象に置換して一つにまとめる。この操作は Contect Editor のツールバーにある “Clarify objects” ボタンを押して呼び出します。

attribute clarification (属性の明確化): 上と類似した操作ですが、属性の集合にのみ適用。Context Editor のツールバーの “Clarify attributes” ボタンを押して呼び出します。

object set reduction (対象集合の規模削減=簡略化): 与えられた対象の集合から対象の部分集合の共通集合として表現できる対象をすべて削除します。

単純化の実行中に clarification (明確化) も同時に実行されます。この操作は概念束の構造を変更しません: 規模を削減された文脈の概念束は、元の文脈の概念束と同型 (isomorphic) です。操作は Context Editor 画面のツールバー上にある “Reduce objects” ボタンを押すと実行されます。

attribute set reduction (属性集合の規模削減**):** 属性集合への、上の対象集合への削減操作と同類の操作。操作は Context Editor 画面のツールバー上の “Reduce attributes” ボタンを押すと実行されます。

context reduction (文脈の規模削減**):** 対象集合と属性集合の規模削減を同時に適用。操作は Context Editor 画面のツールバー上の “Reduce context” ボタンを押すと実行されます。

transposition (文脈表の転地**):** 対象集合と属性集合の交換とこれに対応した集合間の関係の変更。操作は Context Editor 画面のツールバー上の “Transpose context” ボタンを押すと実行されます。

3.7 線図 [=ハッセ図] で色々試す

3.7.1 概念束の構築

概念束を構築するには、Main toolbar の “Build lattice” ボタンを使用します。しばらくすると、(「線図」とも呼ばれる) 束の図が現われますが、所要時間は束の複雑さ次第です。

備考: 束の配置は時間のかかる作業で、これが最初に一つのノードしかない図が描かれ、その後しばらくしてから完全な配置が現われる理由です。

ハッセ図 **(Hasse diagram)** = 線図 [2)]

本利用書には “ハッセ図 (Hasse diagram)” の用語が一切使われていません．その代わりに使われているのが “線図 (line diagram)” です.

3.7.2 束図の解釈

概念束は文脈を曖昧性なしに変換した結果です。概念束の頂点 (top) 要素は、概念束の「単位元」に相当します。概念束の底 (bottom) 要素は、概念束の「ゼ

[2)] この節は翻訳者の黒田が追加しました.

ロ元」を表します。束の各ノードは形式概念分析で言う「(形式的) 概念」に対応します: それは次の条件を満足する対 (O, A) です: O が対象の集合、A が属性の集合で、i) A は O に含まれる全対象が共通して持っている属性の集合で, ii) O は A を真に含む属性を持っていて文脈にある対象の集合です。対象の集合 O は (形式) 概念 (O, A) の「外延」と言い、属性の集合 A は (形式) 概念 (O, A) の「内包」と言います。対象の集合 O は (形式) 概念 (O, A) の「外延」と言い、属性の集合 A は (形式) 概念 (O, A) の「内包」と言います。

「節約したラベルづけ (reduced labeling)」と呼ばれる手法が形式概念の内包と外延を簡潔に表わすために使用されます。属性 A というラベルが概念 C に付随している時、これは A が、C から束のゼロ概念 (底の要素) [=下限] に至る下りの経路上にある全概念の内包中にある事を意味します。対象 O のラベルが概念 C に付随している時、これは O が、C から概念束の単位概念 (頂点要素) に至る上り経路上にある全概念の外延に中にある含事を意味します。

描かれたノードの上半分が青塗りの場合、それは何らかの属性がノードの表わしている概念に付随しているという意味です。描かれたノードの下半分が黒塗りの場合、それは何らかの対象が当ノードが表わしている概念に付属しているという意味です。

線図のノードや辺が赤色で表示される事が時々あります。これは、この辺やノードが別のノードの至近にあるか重なっている事を意味します。配置を改善するには、配置を手で修正するか他の配置法を試してくださ。

3.7.3 可視化モードの幾つか

基本的には、束構造の可視化法には二種類あって、束が一画面内に収まらない場合に別々の動作をします。具体的には:

scroll mode (並行移動モード**):** 束の描画が一画面内に収まらない場合に仮想的に画面を拡大し、利用者には束の描画の一部だけが見えるようになっています。このモードは初期設定で使えるようになっています。

fit to screen mode: 束の描画の大きさを使用中の画面内に収まるように変更します。

並行移動モードと画面に合わせるモードの切り替えは、束の可視化画面のツールバーにある “Scale picture to fit into the image” ボタンの助けを借りて行われます。このボタン押すと一つ目のモードと二つ目のモードが切り替わります。

次の指令は並行移動モードでのみ有効です:

Grab and Drag (つかんでひっぱる): この指令は、見えている領域の pan を行います。このボタンを押すと、カーソルが X に変わり、利用者は描画を pan できます。このモードを切るには “Grab and Drag” ボタンをもう一度押します。

Zoom in, Zoom out, No zoom: これらの指令は名称に対応した行動を起こします。

3.7.4 可視化用オプションの変更

以下の可視化用オプションが画面の左側の小窓にある “Drawing options” を介して調整可能です:

Attribs: ノードの上側のラベルの可視化モード。取り得る値は次の通りです:

Don’t show: ラベルを表示しない。Show labels: 概念に付属する属性のラベルを個別に表示（節約ラベルづけに関する備考も参照)。Show multi-labels: ノードに付属する属性をまとめて表示。

Objects: ノードの下側のラベルの可視化モード。取り得る値は次の通り:

Don’t show: ラベルを表示しない。

Show labels: 対象ラベルをノードの下側に個別に表示。

Show multi-labels: ノードに帰属する全対象をまとめて表示。

Show own objects: 複数の対象が帰属している (つまり、対象の空でない具有がある) 概念に対して、当概念に厳密に帰属している (つまり、持っている属性集合がノードの内包と等しい) 対象の数とパーセントを表示。

Object count: それぞれのノードで，それが規定する概念の外延に含まれている対象の個数 (割合) を表示。

Stability (安定性): それぞれのノードについて，その内包を持つノードを概念束から消滅させる時に、文脈から削除する必要がある対象を個数の最小値を表示。

Draw node: 本オプションはノードの半径を計算する方法を指定します。設定可能な値は以下の通り:

~to own objects: ノードの半径が偶発の大きさ（正確にこのノードの内包に一致する対象の量）に比例して計算されます

fixed radius: 全ノードが同一の半径を持つ。実際のノード半径は “Node radius” オプションで決まります

~of object extent: ノードの半径は、その外延の大きさに比例して計算されます。

Stability (安定性): ノードの半径は破壊に対してどれだけ安定しているかに比例して計算されます (上記の Stability の説明を参照して下さい)。

Draw edge: 辺をどれぐらい正確に描くかを指定する。設定可能な値は以下の通り:

one pixel: 辺の太さが 1 ピクセルに固定されます

no: 辺を描画しない

~object: 辺の太さをそれを通過する対象の数に比例させる。ノードを描く時の “~of object extent” と等価

~connection: 辺の太さは、接続されている上限概念の外延数と下限概念の外延数の比率に比例します。この値は、辺に対応している近似的な連想規則の信頼度に等しくなります。

Highlight: 選択されている辺を除いて、どのノードが強調して表示されるのか指定。これらのオプションは、束図の探索をやり易くするために作成されました。本オプションが取り得る値は以下の通り:

Filter and ideal: フィルターのノード (選択されたノードから束の上端までの上昇経路で到達可能な全ノード) とイデアルのノード (選択されたノードから束の下端までの下降経路で到達可能な全ノード) が強調して表示されます

Selected: 選択されているノードのみが強調表示されます

Neighbors: 選択されたノードと、その上に下の隣接するノードが強調して表示されます

Ideal: イデアルのノードが強調されます

Filter: フィルターのノードが強調されます

No: どのノードも強調表示されません。このオプションは束の画像を保存する時に有用でしょう。

Label font size: 上下のラベルに使用されるフォントの大きさを指定します。

Grid size x: 描画で同一レベルに並んでいるノード間の x 軸上の距離を好みの値にします。これは要素配置の変数に使われていて、この値を変更すると x 軸の方向の大きさが変更されます。

Grid size y: 描画の隣接しているレベルにあるノード間の y 軸上の距離を好みの値に指定します。これは要素配置の制御変数に使われていて、この値を変更すると y 軸の方向の大きさが変更されます。

Node radius: この変数は、概念ノードが取り得る最大半径を指定し、ノードを描画するときに使用されます。

3.7.5 束構造の要素配置を変更する

最初に描いた束の図に満足が行かない場合には、異なる配置法を幾つか実行してから、粗い近似として最良のものを見つけるようにすると良いです。手動で描画を調整を始めるのはその後です。

警告: 束の配置変更は（今のところ）不可逆的な操作です。何かを調節してその結果を失いたくないなら配置変更をしないように。

描画のアルゴリズムには何種類かあり、それぞれに異なる選択肢があります。それらは “Properties” パネルの “Layout options” タブで操作できます。

次の配置アルゴリズムが提供されています:

Minimal intersections (共通集合の最小化**):** これは階層グラフの要素配置のためのアルゴリズムを束の要素配置に応用したものです。これはノードを結ぶ辺の交差の数を最小限にしようとします。これにはパラメータはありません。普通はこのアルゴリズムで最良の描画結果が得られますが、ただ大きな束構造を描く時には結構遅いです。

Chain decomposition (連鎖分解**):** M. Skorsky の chain decomposition アルゴリズムを応用したバージョンです。このアルゴリズムは「加法線図 (additive line diagrams)」と呼ばれる図を構築します。このような線図で作業する場合は「イデアルのノードの移動の戦略 (ideal node movement strategy)」の使用をお勧めします。このアルゴリズムでは分配型束構図の描画で非常に良い結果が得られます。連鎖分解アルゴリズムには次のオプションがあります:

Representation i) 概念のために使用される表現の種類、ii) いつ座標が計算されるか。それぞれ属性ベースにも対象ベースにも指定できます。

Placement ベクトル集合に割り当てる値を決定します。i) 指数関数的、ii) 直線的、iii) 角度的という三つの値の一つを取ります。

Rotate left, Rotate right: 属性ベクトルの集合の回転を実行します。幾つかの可能性から最適なものを選択するために使用されます。

次に挙げる数種類のアルゴリズムは、いわゆる「力指向」の配置アルゴリズムに類するものです。具体的には:

Freese layout: Ralph Freese の束構造描画アルゴリズム [4] の適用です。本アルゴリズムには以下の制御変数があります:

Attraction: ノード間の引力を調整します。

Repulsion: ノード間の反発力を調節します。

Angle: 実際にはアルゴリズムの制御変数ではありません。Freese アルゴリズムは、仮想的に 3 次元空間内で配置を行います。この変数は、仮想 3 次元空間内の配置を 2 次元平面に投影する際の角度を制御するのに使われます。

Force layout: 別の力指向アルゴリズムですが、力の計算法が違っています。本アルゴリズムの制御変数は直前のアルゴリズムの制御変数と同様です。

3.7.6 描画の手動調整

残念ながら、束のすべてのタイプで良い結果を生む単一の描画アルゴリズムは今のところ知られていません。ですから、良好な描画を製作するための必要な最後の仕上げは、束の描画に手で調整を加える事です。ConExp では束のノードの移動がノード間の正しい親-子関係 (後継体-先駆体関係) を維持するように制限されています。

ConExp には束の手動調整に役立つ次の道具が存在します:

Ideal node movement mode: ノードを移動する際に、そのノードのイデアル全体が一緒に移動されます。本モードと他のノードの移動戦略の間の行き来は “Toggle node movement mode” ボタンを押して実行します。

Align nodes to grid: ノードの座標を 8 ピクセル x8 ピクセルの目に見えない碁盤目に合わせます。

3.7.7 束の描画のデフォルト設定を保存

線図の配置を実行するオプションは数が多いため、ユーザが設定の組み合わせをデフォルトとして保持して置く事が可能になっています。それを実行するには、Lattice line digram の画面で ”Store preferences as default” ボタンを押します。 この後、こうして保存した好みの値が、新規に計算される概念束の描画のデフォルト値として使用されます。

3.7.8 図画像の保存

ConExp で最も頻繁な用途の一つは、後の利用を見越して束構造の描画の生成して置く事です。この作業は、束をうまく描画してから Lattice line diagram

の画面のツールバー上にある “Save lattice image” ボタンを押せば実現できます。目下、PNG 形式と JPEG 形式で画像を保存する事ができるようになっています。

3.7.9 文脈の一部から束を構築

ConExp は、元になる文脈の部分文脈に対応する束を構築する機能も持っています。この作業は、束の描画のための画面の右側にある属性選択と対象選択で実現できます。対象の名前や属性の名前を選択するか、選択を解除した後に、新しく選択された部分文脈に対応して新しい束が構築されます。文脈に全対象 (や全属性) を取り込むには、対応する作業画面の下にある “Select all attributes” (や “Select all objects”）ボタンを使用します。

警告**:** 部分文脈から束を構築すると、先に描いた図の情報が破壊される事態を招きます。作業した後に有用な結果を得た場合は、ぜひ画像を保存するなり、束の図の snapshot を作成するなどして下さい。

3.7.10 束の **snapshot** の作成

ある程度の時間をかけて束の配置を調整した後には、他の配置法を模索する前に、手にしている図の複製を作成して置くのが非常に有効でしょう。あるいは、異なる部分文脈の束図を比較検討したり、同一の束について異なる描画法を比較検討できたら良いと思う場合あるでしょう。

手にした束図の “snapshot” (すなわち格子の現在の図面の正確なコピー) を作るには、束の可視化用画面のツールバー上の “Store current lattice as a view” ボタンを押して下さい。それに続いて、図の複製が作成され、文書の分類図の所に表示されます。

3.7.11 束の統計情報を表示

計算済みの束では、幾つかの特性について統計情報を表示できます:

Concept count: 使用中の束中の概念の個数を表示。

Edge count: 使用中の束内の辺の個数を表示。

Lattice height: 束の単位要素 [top=上端] からゼロ要素 [bottom=下端] まで下る経路のうちで最大の長さを表示。

Lattice width estimation: 束の幅の上限と下限を表示。下限の推定値は束の層にある要素数の最大値として計算されます (この値は常に束の幅より小さいかそれに等しくなります)。上限の推定値は精度の意味では「劣化」ですが、<概念数 – 束の高さ> に等しい値です。上限の推定値は精度の意味では「劣化」ですが、<概念数 – 束の高さ> に等しい値です。

3.8 含意関係ベースでの作業

3.8.1 Duquenne-Guigues ベースの計算

文脈について成立する、いわゆる含意関係の Duquenne-Guigues ベースを見つけるには、Main toolbar の “Calculate Calculate Duquenne-Guigues base of implications” ボタンを押すようにして下さい。含意関係の Duquenne-Guigues ベースの主な特徴は、文脈について成立する含意ベースの全集合の中で，それが最小の要素数を持つ含意関係ベースになっている事です。

“Implication sets (含意関係の集合)” の作業画面で表示される含意関係は次のような書式に従っています:

No <対象の数> 前提 ==> 帰結 [前提が真なら帰結が真]。

No は単に一覧中での含意の通し番号を意味します。

<対象の数> は含意関係が成立する対象の個数を示します。

前提と帰結は通常、前提と帰結のそれぞれに帰属する属性名の一覧です。前提は “{ }” で表現される場合もあり、これは当該の含意関係の前提が空で、文脈中の全対象に対して成立することを意味します。含意関係は、青か赤のどちらかの色つきで表示される場合があります:

青い色づけは、当該の含意規則を支持する対象が文脈中に存在している事を意味します。

赤い色づけは、当該の含意関係を支持する対象が文脈中に存在せず、そのような含意は通常、その前提に含まれる対象集合が、文脈中に一緒に発生

していない事を意味します。そのような含意関係は更に、それについて成立する属性間の文脈の全属性を含んでいます。

3.8.2 連想関係の検索

連想規則は含意規則と別物で、厳密でない規則も許可されています: 厳密でない規則とは、前提が成立しても帰結が必ずしも成立しない規則、唯前提が覆っている対象全体でなく、その一部についてのみ真である規則の事です。連想規則のベースは二つの部門に分れます: 厳密な規則のベース (Duquenne-Guigues base) と近似的な規則のベース (いわゆる “Luxenburger base”) です。

ConExp は連想規則のベースを求める計算を実行します。これを実行するには、主要ツールバーにある “Calculate association rules” ボタンを押して下さい。連想規則の表示の書式は、含意関係の書式に少し変更を加えたものです。すなわち:

> No <前提が成立する対象の個数> 前提 =[規則の信頼度]=> <前提と帰結が成立する対象の個数>。

含意関係の表示に用いられる赤と青の色づけに加えて、緑の色づけが近似的な、厳密でない規則のために使用されます。

3.8.3 属性探査の実行

特定の文脈で計算された含意集合に生じる問題の一つは、それが与えられた文脈で成立するけれど、調査している対象領域に属する対象全体について一般に成立する訳ではないということです。この欠陥を克服するのに属性の探索の手順が用いられます.

属性探査は対話的な手順で、実行中にプログラムが一定の属性の集合から導かれた依存関係に関する質問をして来ます。あなたが専門家として、このような依存関係が一般的に成立する事が確認できたら、その質問に Yes と答えます。あなたが依存関係の成立を否認する場合には反例を提示しなければなりません。あなたが専門家としてすべての質問に正しく答えたとすると、属性探索の手順が終了した後に、調査している対象領域の異なる属性間に成立する依存関係を記述する含意の全体集合が得られます。

探査手順属性は、属性のみが指定されている空の文脈から始めても、一部の対象の記述が存在している空でない文脈から始めても構いません。属性探査手順を始めるには、主要ツールバーの “Start attribute exploration” ボタンを押して下さい。そうすると最初の質問を受けるので、利用者はそれを確証するか，拒絶するか，属性探索の手順を終了するかのどれかの対応をして下さい。利用者が質問に No と答えると、別の対話窓が表示されて、反例を示すように要求されます。

4 利用者向け Mailing list

本プログラムの使用感を皆さんから聞けるのはいつでもありがたいです。ConExp にフィードバックを与えるのと質問をするのに最適な場所は ConExp の利用者用メーリングリストです: conexp-user@lists.sourceforge.net

5 ConExp のチーム

開発チームの構成は現在、次です:

- Dr. Serhiy Yevtushenko – 当初の，そして主な開発者
- Tim Kaiser
- Julian Tane
- Dr. Sergei Objedkov

解説書チームには次の方々も含まれています:

- Joachim Hereth-Correia
- Heiko Reppe

開発に参加したいのでしたら、歓迎します。

参照文献

[1] Serhiy A. Yevtushenko. System of data analysis “Concept Explorer”. (In Russian). *Proceedings of the 7th national conference on Artificial Intelligence KII-2000*, p. 127-134, Russia, 2000.

[2] B. Ganter and R. Wille. “Formal Concept Analysis:Mathematical Foundations”, Springer-Verlag, 1999

[3] ComImp: http://www.mathematik.tu-darmstadt.de/~burmeister/

[4] http://www.math.hawaii.edu/~ralph/LatDraw/

[5] http://conexp.sourceforge.net/