(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4703099号
(P4703099)
(45)発行日　平成23年6月15日(2011.6.15)　(24)登録日　平成23年3月18日(2011.3.18)

(51)Int.Cl.　　F I
G06F 17/27 (2006.01)　　G06F 17/27　Z
G06F 17/28 (2006.01)　　G06F 17/28　B

請求項の数 9　外国語出願　(全 45 頁)

(21)出願番号　特願2003-6415(P2003-6415)
(22)出願日　平成15年1月14日(2003.1.14)
(65)公開番号　特開2003-233606(P2003-233606A)
(43)公開日　平成15年8月22日(2003.8.22)
審査請求日　平成17年12月1日(2005.12.1)
審判番号　不服2008-864(P2008-864/J1)
審判請求日　平成20年1月10日(2008.1.10)
(31)優先権主張番号　10/047,472
(32)優先日　平成14年1月14日(2002.1.14)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(74)復代理人　100115624
弁理士　濱中　淳宏
(74)復代理人　100115635
弁理士　窪田　郁大

最終頁に続く

(54)【発明の名称】談話表現構造を正規化する方法および正規化されたデータ構造

(57)【特許請求の範囲】
【請求項１】
ボックス要素と、入力ストリング内のエンティティを特定するマーカおよびデータ値を有するボックス要素引数と、を有するボックスを含む談話表示構造(DRS)を正規化する、プロセッサ、メモリ、および出力機構を有するコンピュータにより実施される方法であって、前記DRSは入力ストリングに関する情報を表現し、前記方法は、
前記プロセッサにより、前記メモリ内のDRSの中でボックスおよびボックス要素が表現される形式を正規化するステップであって、
前記DRSから、該DRSへのポインタを有しない未使用のボックスと、識別子を有しない未使用のボックス要素とを削除するステップと、
前記ボックスに連続するインデックスを付けるステップと、
前記ボックス要素に連続する新たな識別子を付けるステップと
を含む、前記ステップと、
前記プロセッサにより、前記ボックスおよび前記ボックス要素の正規化された形式に基づき、前記マーカに関わらず、前記ボックスおよび前記ボックス要素をソートして、前記メモリ内の予備の順序付けを獲得するステップと、
前記プロセッサにより、前記マーカに連続するマーカ値を付けて、前記DRSの中でマーカが表現される形式を正規化するステップと、
前記プロセッサにより、前記予備の順序付けに基づき、かつ前記マーカの正規化された形式に基づき、前記ボックスおよび前記ボックス要素を更にソートして、前記DRSを正規

化するステップと、
前記正規化されたDRSを前記出力機構から出力するステップと
を含み、
前記ボックスのソートを、関連する前記ボックス要素に基づいて行い、
前記ボックス要素のソートを、該ボックス要素および関連する前記ボックス要素引数に基づいて行うことを特徴とする方法。
【請求項２】
前記ボックスおよび前記ボックス要素の正規化された形式に基づき、前記マーカに関わらず前記ボックスおよび前記ボックス要素をソートするステップは、
前記プロセッサにより、前記インデックスおよび前記識別子に基づいて前記ボックスおよび前記ボックス要素を辞書式順序に並べて辞書式に順序付けられたボックスおよびボックス要素を得ることを含むことを特徴とする請求項１に記載の方法。
【請求項３】
前記ボックスおよび前記ボックス要素の正規化された形式に基づき、前記マーカに関わらず前記ボックスおよび前記ボックス要素をソートするステップは、
前記プロセッサにより、連続する識別子が付けられたボックス要素を参照するように前記ボックスを更新することを含むことを特徴とする請求項２に記載の方法。
【請求項４】
前記ボックスおよび前記ボックス要素の正規化された形式に基づき、前記マーカに関わらず前記ボックスおよび前記ボックス要素をソートするステップは、
前記プロセッサにより、連続するインデックスが付けられたボックスを参照するように前記ボックス要素を更新することを含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記DRSの中でマーカが表現される前記形式を正規化するステップは、
前記プロセッサにより、各マーカと、前記マーカを含むボックスおよびボックス要素を特定するリストとの間のマッピングを前記メモリ内に生成することを含むことを特徴とする請求項４に記載の方法。
【請求項６】
前記DRSの中でマーカが表現される前記形式を正規化するステップは、
前記プロセッサにより、マーカを含むボックスおよびボックス要素を特定する前記リストと、各マーカの間の逆マッピングを前記メモリ内に生成することを含むことを特徴とする請求項５に記載の方法。
【請求項７】
前記DRSの中でマーカが表現される前記形式を正規化するステップは、
前記プロセッサにより、連続するマーカ値が付けられた前記マーカを参照するように前記ボックス要素を更新することを含むことを特徴とする請求項６に記載の方法。
【請求項８】
前記予備の順序付けに基づき、かつ前記マーカの正規化された形式に基づき、前記ボックスおよび前記ボックス要素をソートするステップは、
前記プロセッサにより、連続するマーカ値が付けられた前記マーカに基づいて前記辞書式に順序付けられたボックスおよびボックス要素をソートして、正規化されたDRSを得ることを含むことを特徴とする請求項７に記載の方法。
【請求項９】
前記プロセッサにより、前記正規化されたDRSを表すストリングを前記メモリ内に生成することをさらに備えることを特徴とする請求項８に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、自然言語資料の意味解析に関する。より具体的には、本発明は、自然言語処理システムによって提供される言語構造の正規化に関する。

【０００２】
【従来の技術】
ユーザからの自然言語入力を受け入れて正確に処理するのに適した自然言語インターフェースが、多くのアプリケーションに有益である。そのような自然言語システムは、言語および概念のばらつきに対して堅牢でなければならない。したがって、そのような自然言語システムは、使用するのが容易でなければならない。
【０００３】
例えば、自然言語システムは、修飾詞付加の曖昧さ、数量詞範囲の曖昧さ、接続詞および選語記号の曖昧さ、複合名詞形の曖昧さ、照応、省略文等に対応することができなければならない。
【０００４】
従来のシステムでは、自然言語入力における曖昧さを解決するのに限られた量の意味論的知識またはヒューリスティックを使用した。しかし、この手法は、不都合に複雑であり、あるいは非効率的になる可能性がある。
【０００５】
自然言語表現が自然言語処理を受けるようにすることができ、また自然言語入力の意味を表す言語構造を開発することができると想定しても、特定の構造を有する入力を予期するアプリケーションでその言語構造を使用するためには、大きな障害を克服しなければならない。例えば、多くの従来のアプリケーションは、事前定義された構造で情報が入力されることを予期するが、その構造は、しばしば、アプリケーションごとに異なる。多くの自然言語処理システムの出力は、そのようなアプリケーションに入力するのに適した構造ではない。
【０００６】
したがって、自然言語入力の意味を表現する言語構造を開発することができると想定しても、そのことは、その言語構造が、自然言語入力によって表現される情報を使用することを望むアプリケーションに入力するのに適していることを意味しない。
【０００７】
【特許文献１】
米国特許第5,995,922号明細書
【０００８】
【非特許文献１】
C.A.R. Hoare, Quicksort. Computer Journal, 5(1):10-15, 1962
【０００９】
【非特許文献２】
R. Bayer, Symmetric binary B-trees:Data structure and maintenance algorithms. Acta Informatica, 1:290-306, 1972
【００１０】
【非特許文献３】
D.E. Knuth, Sorting and Searching, Vol. 3 of The Art of Computer Programming. Addison-Wesley, 1973)
【００１１】
【発明が解決しようとする課題】
本発明は、談話表現構造(DRS)を正規化するためのシステムおよび方法である。異なるように見える可能性があるが、それでも等価である構造を同一の正規化された表現に関連付けることができるように、構造の要素を書き換え、記憶する。
【００１２】
また、本発明は、DRSのためのデータ構造も含むことが可能である。DRSは、1組の要素をそれぞれが有するボックスのアレイで表現され、各アレイは、多種多様な言語情報を表現するのに適した事前定義された構造を有する。
【００１３】

【課題を解決するための手段】
A.概要
本発明は、談話表現構造(DRS)を正規化するためのシステムおよび方法である。本発明は、自然言語解析システムによって出力された言語構造を解釈する意味解析のシステムおよび方法のコンテキストで説明する。意味解析システムからの出力は、意味談話表現構造(SemDRS)と呼ばれるデータ構造モデルである。本発明は、いくつかのサブプロセスおよびデータ構造に加えて、全体的なプロセスおよびアーキテクチャも含む。本発明をよりよく理解するため、まず、本発明を使用することができる一例としての環境について述べる。もちろん、本発明は、多種多様なその他のシステムおよび環境においても使用できることが理解されよう。

【００１４】
【発明の実施の形態】
B.例示的動作環境
図1は、本発明を実施することができる適切なコンピューティングシステム環境100の例を示している。コンピューティングシステム環境100は、適切なコンピューティング環境の一例に過ぎず、本発明の使用または機能の範囲に関する何らかの限定を示唆するものではない。また、コンピューティング環境100は、例示の動作環境100で例示する構成要素のいずれかに関する、またはその組合せに関する何らかの依存性または要件を有するものと解釈すべきでもない。

【００１５】
本発明は、多数の他の汎用または専用のコンピューティングシステムの環境または構成で動作できる。本発明で使用するのに適する可能性がある周知のコンピューティングシステム、コンピューティング環境、および/またはコンピューティング構成の例には、パーソナルコンピュータ、サーバコンピュータ、ハンドヘルドデバイスまたはラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラマブル家庭用電子機器、ネットワークPC、ミニコンピュータ、メインフレームコンピュータ、前述したシステムまたはデバイスのいずれかを含む分散コンピューティング環境等が含まれるが、以上には限定されない。

【００１６】
本発明は、コンピュータによって実行可能なプログラムモジュールなどのコンピュータ実行可能命令の一般的コンテキストで説明することができる。一般に、プログラムモジュールには、特定のタスクを行う、または特定の抽象データタイプを実施するルーチン、プログラム、オブジェクト、コンポーネント、データ構造等が含まれる。また、本発明は、タスクが、通信網を介してリンクされたリモートの処理デバイスによって行われる分散コンピューティング環境において実施してもよい。分散コンピューティング環境では、プログラムモジュールは、メモリ記憶デバイスを含むローカルのコンピュータ記憶媒体とリモートのコンピュータ記憶媒体の両方に配置することができる。

【００１７】
図1を参照すると、本発明を実施するための例示のシステムが、コンピュータ110の形態で汎用コンピューティングデバイスを含む。コンピュータ110の構成要素には、処理ユニット120、システムメモリ130、およびシステムメモリから処理ユニット120までを含む様々なシステム構成要素を結合するシステムバス121が含まれることが可能であるが、以上には限定されない。システムバス121は、様々なバスアーキテクチャのいずれかを使用する、メモリバスまたはメモリコントローラ、周辺バス、およびローカルバスを含むいくつかのタイプのバス構造の任意のものにすることができる。限定としてではなく例として、そのようなアーキテクチャには、ISA(Industry Standard Architecture)バス、MCA(Micro Channel Architecture)バス、EISA(Enhanced ISA)バス、VESA(Video Electronics Standards Association)ローカルバス、およびメザニン(Mezzanine)バスとしても知られるPCI(Peripheral Component Interconnect)バスが含まれる。

【００１８】

コンピュータ110は、通常、様々なコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータ110がアクセスすることのできる任意の利用可能な媒体であることが可能であり、揮発性媒体および不揮発性媒体、ならびに取外し可能な媒体および取外し不可能な媒体をともに含む。限定としてではなく例として、コンピュータ可読媒体は、コンピュータ記憶媒体および通信媒体を含むことが可能である。コンピュータ記憶媒体は、コンピュータ可読命令、データ構造、プログラムモジュール、またはその他のデータなどの情報を記憶するための任意の方法または任意の技術で実装される揮発性媒体および不揮発性媒体、ならびに取外し可能な媒体および取外し不可能な媒体をともに含む。コンピュータ記憶媒体には、RAM、ROM、EEPROM、フラッシュメモリまたはその他のメモリ技術、CD-ROM、デジタル多用途ディスク(DVD)または他の光ディスクストーレッジ、磁気カセット、磁気テープ、磁気ディスクストーレッジまたは他の磁気記憶デバイス、あるいは所望の情報を記憶するのに使用することができ、コンピュータ110がアクセスすることのできる任意の他の媒体が含まれるが、以上には限定されない。通信媒体は、通常、搬送波または他のトランスポート機構などの変調されたデータ信号でコンピュータ可読命令、データ構造、プログラムモジュール、またはその他のデータを実現し、あらゆる情報送達媒体が含まれる。「変調されたデータ信号」という用語は、特性の1つまたは複数が、情報を信号内で符号化するように設定された、または変更された信号を意味する。限定としてではなく例として、通信媒体には、有線網または直接有線接続などの有線媒体、ならびに音響、RF、赤外線、その他の無線媒体などの無線媒体が含まれる。また、前述した媒体のいずれかの組合せも、コンピュータ可読媒体の範囲に含まれるべきものである。

【００１９】

システムメモリ130は、読取り専用メモリ(ROM)131およびランダムアクセスメモリ(RAM)132などの揮発性、および/または不揮発性の形態のコンピュータ記憶媒体を含む。始動中など、コンピュータ110内部の要素間で情報を転送するのを助ける基本ルーチンを含む基本入力/出力システム133(BIOS)が、通常、ROM131の中に記憶されている。RAM132は、通常、処理ユニット120が即時にアクセスすることができ、かつ/または現在、処理ユニット120が操作しているデータおよび/またはプログラムモジュールを含む。限定としてではなく例として、図1は、オペレーティングシステム134、アプリケーションプログラム135、その他のプログラムモジュール136、およびプログラムデータ137を示している。

【００２０】

また、コンピュータ110は、その他の取外し可能/取外し不可能な揮発性/不揮発性のコンピュータ記憶媒体も含むことが可能である。単に例として、図1は、取外し不可能な不揮発性の磁気媒体に対して読取りまたは書込みを行うハードディスクドライブ141、取外し可能な不揮発性の磁気ディスク152に対して読取りまたは書込みを行う磁気ディスクドライブ151、およびCD-ROMまたはその他の光媒体などの取外し可能な不揮発性の光ディスク156に対して読取りまたは書込みを行う光ディスクドライブ155を示している。例示の動作環境で使用することができるその他の取外し可能/取外し不可能な揮発性/不揮発性のコンピュータ記憶媒体には、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、半導体RAM、半導体ROM等が含まれるが、以上には限定されない。ハードディスクドライブ141は、通常、インターフェース140のような取外し不可能なメモリインターフェースを介してシステムバス121に接続され、また磁気ディスクドライブ151および光ディスクドライブ155は、通常、インターフェース150のような取外し可能なメモリインターフェースでシステムバス121に接続される。

【００２１】

前述し、図1で示したドライブおよび関連コンピュータ記憶媒体は、コンピュータ可読命令、データ構造、プログラムモジュール、およびコンピュータ110のための他のデータのストーレッジを提供する。図1では、例えば、ハードディスクドライブ141が、オペレーティングシステム144、アプリケーションプログラム145、その他のプログラムモジュール146、およびプログラムデータ147を記憶しているものとして示している。以上の構成要素は、オペレーティングシステム134、アプリケーションプログラム135、その他のプログラム

モジュール136、およびプログラムデータ137と同じであることも、異なることも可能であることに留意されたい。オペレーティングシステム144、アプリケーションプログラム145、その他のプログラムモジュール146、およびプログラムデータ147には、最低限、それらが異なるコピーであることを示すために、ここでは異なる符号を付けている。

【００２２】

ユーザは、キーボード162、マイクロホン163、ならびにマウス、トラックボール、またはタッチパッドなどのポインティングデバイス161などの入力デバイスを介してコンピュータ110にコマンドおよび情報を入力することができる。その他の入力デバイス(図示せず)には、ジョイスティック、ゲームパッド、サテライトディッシュ、スキャナ等が含まれることが可能である。以上、およびその他の入力デバイスは、しばしば、システムバスに結合されたユーザ入力インターフェース160を介して処理ユニット120に接続されるが、パラレルポート、ゲームポート、またはユニバーサルシリアルバス(USB)などの他のインターフェースおよびバス構造で接続してもよい。また、モニタ191またはその他のタイプの表示デバイスも、ビデオインターフェース190などのインターフェースを介してシステムバス121に接続される。コンピュータは、モニタに加えて、出力周辺インターフェース195を介して接続することができるスピーカ197やプリンタ196などの他の出力周辺デバイスも含むことが可能である。

【００２３】

コンピュータ110は、リモートコンピュータ180のような1つまたは複数のリモートコンピュータに対する論理接続を使用するネットワーク化された環境で動作することができる。リモートコンピュータ180は、パーソナルコンピュータ、ハンドヘルドデバイス、サーバ、ルータ、ネットワークPC、ピアデバイス、またはその他の一般的なネットワークノードであることが可能であり、通常、コンピュータ110に関連して前述した要素の多く、またはすべてを含む。図1で描いた論理接続は、ローカルエリアネットワーク(LAN)171およびワイドエリアネットワーク(WAN)173を含むが、その他のネットワークも含むことが可能である。そのようなネットワーキング環境は、オフィス、企業全体のコンピュータ網、イントラネット、およびインターネットで一般的である。

【００２４】

LANネットワーキング環境で使用される場合、コンピュータ110は、ネットワークインターフェースまたはネットワークアダプタ170を介してLAN171に接続される。WANネットワーキング環境で使用される場合、コンピュータ110は、通常、インターネットなどのWAN173を介して通信を確立するためのモデム172またはその他の手段を含む。内部にあることも、外部にあることも可能なモデム172は、ユーザ入力インターフェース160、または他の適切な機構を介してシステムバス121に接続することができる。ネットワーク化された環境では、コンピュータ110に関連して描いたプログラムモジュール、またはモジュールの部分は、リモートのメモリ記憶装置の中に記憶することができる。限定としてではなく例として、図1は、リモートのアプリケーションプログラム185が、リモートコンピュータ180上に常駐しているものとして示している。図示したネットワーク接続は、例示のものであり、コンピュータ間で通信リンクを確立する他の手段も使用できることが理解されよう。

【００２５】

C.テキスト処理システムの概要

図2は、本発明のための環境の一実施形態によるテキスト処理システム200を示すブロック図である。テキスト処理システム200は、言語解析構成要素202、および意味解析構成要素204へのDRS入力の正規化を含む。本発明は、意味解析構成要素204へのDRS入力の正規化、および正規化されたDRSのデータ構造を指向するが、例として、システム200のコンテキストで示している。

【００２６】

システム200では、テキストストリング206が、言語解析構成要素202に入力される。ストリング206は、例として発話であり、あるいはテキストに変換されている任意の他のタイプの自然言語入力である。言語解析構成要素202は、入力ストリング206を解析して、一例

としての実施形態では、UDRS、構文解析ツリー、論理形式、トークン化されたストリング、および1組の名前付きエンティティを含む構文解析を生成する。以上のデータ構造のそれぞれは、周知であり、したがって、簡単にだけ述べる。言語解析構成要素202は、例として、最良優先(best-first)順にランク付けされた、任意の所与のテキスト入力ストリング206に関する複数の異なる構文解析を出力することができる。

【００２７】

UDRSは、言語解析構成要素202によって出力される言語構造であり、図5に関連してさらに詳細に述べる。また、構文解析ツリーおよび論理形式グラフもそれぞれ、言語解析構成要素202における自然言語処理によって生成される従来の依存関係、およびグラフ構造である。これらの構文解析ツリーおよび論理形式は、1999年11月30日、ペンテルダキス(Penteroudakis)他に発行された米国特許により詳細に説明されている(特許文献1参照)。また、トークン化されたストリングも、従来のものであり、入力ストリング206から生成される。入力ストリング206を構成要素のトークンに分解し、そのトークンにラベル付けをするように構成された任意の周知のトークナイザ(tokenizer)を使用して、トークン化されたストリングを生成することができる。名前付きエンティティは、単一のユニットとして認識されるべき固有名などのエンティティである。名前付きエンティティは、異なる領域において大幅に異なる可能性があるが、一般に、名詞句に相当する。

【００２８】

構文解析の以上の要素のいくつかだけを意味解析構成要素204に提供する必要がある可能性があるが、一例としての実施形態では、以上の要素がすべて、言語解析構成要素202によって生成されて(または獲得されて)、意味解析構成要素204に提供される(ストリング206の構文解析の部分として)。

【００２９】

意味解析構成要素204は、入力として、構文解析構成要素202からの構文解析、アプリケーションスキーマ、および1組の意味マッピング規則を受け取る。この入力に基づき、意味解析構成要素204は、出力として、入力ストリング206によって示される発話を表す1つまたは複数のSemDRSを非言語領域のエンティティーリレーションモデル(entity-and-relation model)として(例えば、アプリケーションスキーマとして)提供する。

【００３０】

アプリケーションスキーマおよび意味マッピング規則は、本出願において以下でより詳細に説明している。ただし、概略を述べると、アプリケーションスキーマは、例として、アプリケーション開発者によって作成されることが可能である。アプリケーションスキーマは、関連するタイプ階層を伴う、エンティティーリレーションモデルによるアプリケーションの能力および挙動のモデルである。また、意味マッピング規則も、例として、アプリケーション開発者によって作成されることが可能であり、入力のUDRSと1組のSemDRSフラグメントとの間の関係を示す。意味マッピング規則の左側は、特定の形態のUDRSにマッチし、一方、右側は、アプリケーションスキーマのある部分に直接に対応するSemDRSフラグメントを特定する。意味マッピング規則をUDRSに適用することにより、また複数のマッピングおよび他のデータ構造を維持することにより、意味解析構成要素204は、アプリケーションスキーマに正確に対応し、また入力ストリング206によって示される発話、および意味解析構成要素204に入力されたUDRSを表す、所望のボックス構造を有する総合SemDRSを生成することができる。

【００３１】

D.意味解析層

1.アーキテクチャ

図3は、意味解析構成要素204をより詳細に表すブロック図である。図3は、意味解析構成要素204が、例として、意味解釈コントローラ(SIC)210、解釈フラグメント生成器(IFG)212、および解釈アセンブラ(IA)214を含むことを示している。SIC210は、サービスインボーカ(invoker)216、初期探索状態(ISS)生成構成要素218、DRS構造復元生成器220、およびIFG212によって生成されるすべての解釈フラグメントおよび関連情報を含むデータ記憶装置

215を含む。IFG212は、スキーマエンジン222、Blurb UDRS処理エンジン224、および堅牢性戦略処理エンジン226を含む。IA214は、解決パターン生成構成要素228を含む。

【００３２】

SIC210は、言語解析層(言語解析器202)によって提供される証拠から出力SemDRS解釈を生成するプロセスを制御し、また意味解析器204の様々なサブ構成要素に供与される必要がある情報を管理することも行う。

【００３３】

次に、SIC210に入力される関心を引く主な構成要素をより詳細に説明する。そのアイテムは、入力ストリング206(図2で示す)が、次のとおり読み取られる例(例1)に関連して説明する。すなわち、「KDからでないすべての電子メールを削除する。」

【００３４】

2.入力DRS-構造および正規化

図5Aは、入力ストリングの言語解析を表す入力DRS(またはUDRS)250を示している。UDRS250は、入力ストリングにおいて話題となっているエンティティ、および入力ストリングにおいてそれらのエンティティが有する言語関係の表現である。例示の入力ストリングでは、話題となっている3つのモデル化されたエンティティ、つまり事柄が存在する。すなわち、削除されるもの、指名された個人であるKD、および削除するイベントである。以上の論理要素のすべてが、図5Aで示したボックス構造250の中に存在する。このボックス構造は、各サブボックスが、アレイにおけるインデックスに対応する動的アレイとして表現することができる。DRSアレイへのインデックスは、ボックス構造250の各ボックスの右下隅に表示されていることに留意されたい。また、定量化されている変数(すなわち、入力ストリングの中の「メール」に関連する変数)は、論理構造の一部として含まれることにも留意されたい。

【００３５】

図5Aで示したボックス構造250では、変数yが、KDを表し、変数xが、1通の電子メールを表し(すなわち、Eメール)、(x,y)からという語句が、電子メール(x)が、エンティティ(y)からのものであることを表す。変数eが、削除する動作を表し、また「(e,x)を削除する」という語句が、eを削除する動作が、電子メール(x)に対して行われるべきことを示す。ボックス構造250の中の矢印、およびボックス構造250の中の内部ボックスは、電子メールという条件、および「からでない」という条件（否定は、ボックス5の外側の左上隅の波形符号で表されていることに留意されたい)を満たすすべてのx(電子メール)に対して、その動作が行われるべきことを表している。言い換えれば、この特定の例では、ボックス構造および矢印は、KDからでないすべての電子メールに対して削除動作が行われる(すなわち、削除される)ことを示している。

【００３６】

DRS表現

DRSを実現するデータ構造、およびそのような構造を正規化するための方法のより一般的で完全な説明が、役立つ可能性がある。もちろん、この説明は、一例としての表現および正規化のスキームに過ぎない。

【００３７】

DRSは、1つまたは複数のボックスのコンテナであると考えることができる。kをDRSの中のボックスの数とする。各ボックスは、ボックスラベルと呼ばれる0からk-1の間の数である固有のインデックスを有する。

【００３８】

ボックスには、ゼロまたは1つ以上のボックス要素が関連している。nが、ボックスの要素の数である場合には、各要素は、0からn-1の間の数である固有のインデックスを有する。

【００３９】

ボックス要素は、マーカ要素、制約要素、または論理要素であることが可能である。マーカ要素は、談話におけるエンティティの表現である。制約要素は、ゼロまたは1つ以上の引数、およびその制約が何であるかに関する何らかの情報を有する。一実施形態では、こ

の情報は、当面の制約の種類を示すタグ、ストリング値および意味値を含む。同様に、論理要素も、ゼロまたは1つ以上の引数およびその引数に適用される論理演算子が何であるかに関する情報を有する。

【００４０】

ボックス要素引数は、マーカ、ボックスのインデックス、またはデータ値である。

【００４１】

一実施形態では、DRSは、自らのすべてのボックスのベクトル、および自らのすべてのボックス要素のベクトルをDRSに関するレコードの中に記憶する。各ボックスは、その特定のボックスの一部である要素のインデックスのベクトルを含む。これにより、ボックスの要素をソートすることがコストのかからないものになり、要素がどのボックスに属するかに関わらず、DRSのすべてのボックス要素をトラバースすることが容易になる。各ボックス要素は、自らのボックス要素引数のベクトルを含む。

【００４２】

正規化されたDRSに関して、維持される追加の情報も存在する。すなわち、DRSの中の別々のマーカの数、各マーカをボックスのインデックスとそのボックス内部のボックス要素のインデックスのペアのリストに関連付けるベクトル、およびDRSの正規化されたストリング表現であるストリングである。

【００４３】

ヘルパタイプ

マーカは、(非負)整数である。DRSのグラフ式表現では、その整数を英数字ストリングにマップすることが役立つが、この内部表現は、例として整数である。

【００４４】

ボックスのインデックスまたはボックスラベルは、(非負)整数である。

【００４５】

BoxArgumentKind値は、null_arg、arg_marker、arg_label、およびarg_valueのうちの1つである。

【００４６】

BoxElementKindは、null_elem、elem_marker、elem_constraint、およびelem_logicalのうちの1つである。

【００４７】

BoxElementSemKindは、いくつかの別々の値のうちの1つである。本開示の中で、以上の値がどのようなものであるかに依存するものは全く存在しないが、一実施形態は、null_sem、sem_none、sem_entity、sem_relation、sem_value、およびsem_dummyを含む。

【００４８】

データ値は、不特定の合併タイプである。本開示の中で、データ値の許可された範囲がどのようなものであるかに依存するものは全く存在しないが、一実施形態は、空値、ブーリアン、整数、およびストリングを含む。例として、データ値全体にわたり、総合順序付け関係<が存在する。一実施形態では、空値は、他のすべての値に先行し、ブーリアンは、整数およびストリングに先行し、整数は、ストリングに先行し、偽は、真に先行し、整数は、従来の順序で並べられ、ストリングは、辞書式順序で並べられる。

【００４９】

SemanticNode値は、整数である。SemanticNode値は、意味領域を表す外部データ構造へのインデックスとなるように意図されている。

【００５０】

ボックス要素引数の表現

ボックス要素引数は、以下のフィールドを有するレコードによって表現される。

・BoxArgumentKind値、m_kind
・整数、m_id
・データ値、m_data

【００５１】

ボックス要素引数を生成する際、m_kindを設定する。ボックス要素引数がarg_markerまたはarg_labelである場合には、m_idも設定する。ボックス要素引数が、arg_valueである場合には、m_dataも設定する。

【００５２】

効率的でオプションの実施形態は、m_idフィールドおよびm_dataフィールドをオーバーレーすることができる。というのは、m_idフィールドおよびm_dataフィールドが同時に使用されることは決してないからである。

【００５３】

ボックス要素引数に対する演算

xをボックス要素引数とする。

・x.kind()はx.m_kindフィールドを参照する。

・x.maker()はx.m_idフィールドを参照するものであり、x.m_kindがarg_makerである場合に使用することができる。

・x.label()はx.m_idフィールドを参照するものであり、x.m_kindがarg_labelである場合に使用することができる。

・x.data()はx.m_dataフィールドを参照するものであり、x.m_kindがarg_valueである場合に使用することができる。

・また、比較演算子$<_a$および$<_b$も使用することができ、以下で説明する。

【００５４】

ボックス要素の表現

ボックス要素は、以下のフィールドを有するレコードによって表現される。

・BoxElementKind値、m_kind

・BoxElementSemKind値、m_sem_kind

・ストリング、m_word

・SemanticNode値、m_hnode

・整数、m_number_of_args

・長さが、m_number_of_argsで与えられるボックス要素引数のベクトル、m_args

【００５５】

ボックス要素引数を生成する際、m_kindを常に設定しなければならない。

・m_kindがelem_markerである場合には、m_number_of_argsが1であり、かつm_args[0]が引数であり、m_args[0].kind()がarg_markerである。

・m_kindがelem_constraintである場合には、m_sem_kindおよびm_number_of_argsを常に設定しなければならなず、またm_argsをm_number_of_args引数で初期設定しなければならない。一例としての実施形態では、m_sem_kind、m_wordおよびm_hnodeの間の関係も存在する。

・m_sem_kindがsem_noneまたはsem_valueである場合には、m_wordをストリングで初期設定しなければならない。

・m_sem_kindがsem_entityまたはsem_relationである場合には、m_hnodeをSemanticNode値で初期設定しなければならない。

【００５６】

m_number_of_argsも設定しなければならず、また$0 \leqq i \leqq$m_number_of_argsである各iに関して、m_args[i]をボックス要素引数に初期設定しなければならない。

【００５７】

効率的でオプションの実施形態は、m_wordフィールドとm_hnodeフィールドが、同時に使用されることは決してないので、これらのフィールドをオーバーレイすることができる。

【００５８】

一例としての実施形態は、正規化の演算に関わらない追加の2つのフィールドを有する。これらのフィールドの中で表現される情報は、文の意味を確立する際に使用する。これらのフィールドは、以下を含む。

・整数、m_tokenのリスト。このリストは、この特定のボックス要素によってカバーされ

る入力文の中のトークン(の位置)を記録する。
・整数、m_token_spanのリスト。このリストが先頭である構文の句の全体によってカバーされる入力文の中のトークン(の位置)を記録する。
【００５９】
ボックス要素に対する演算
xをボックス要素とする。
・x.kind()はx.m_kindフィールドを参照する。
・x.sem_kind()はx.m_sem_kindフィールドを参照する。
・x.word()はx.m_wordフィールドを参照するものであり、x.m_kindがelem_constraintであり、かつx.m_sem_kindがsem_noneまたはsem_valueである場合に使用することができる。
・x.schema_node()はx.m_hnodeフィールドを参照するものであり、x.m_kindがelem_constraintであり、かつx.m_sem_kindがsem_entityまたはsem_relationである場合に使用することができる。
・x.operator()はx.m_wordフィールドを参照するものであり、x.m_kindがelem_logicalである場合に使用することができる。
・x.number_of_args()がx.m_number_of_argsフィールドを参照する。
・x.arg(i)がx.m_args[i]を参照し、ここで0≦i<x.m_number_of_argsである。 ・x.add_arg(y):iをx.m_number_of_argsとする;x.m_args[i]をボックス要素引数yに設定する;1をx.m_number_of_argsに加える;iを戻す。
・x.covered_tokens()がx.m_tokensフィールドを参照する。
・x.phrase_token_span()がx.m_token_spanフィールドを参照する。
・以下で述べる比較演算子$<_a$および$<_b$も使用することができる。
【００６０】
ボックスの表現
ボックスは、以下のフィールドを有するレコードによって表現される。
・DRSに対するポインタであるフィールドm_ptr_containing_drs
・整数、m_number_of_elements
・m_number_of_elementsで長さが与えられる整数ベクトル、m_element_indices
【００６１】
ボックスを生成するとき、m_ptr_containing_drsを、そのボックスが中に入っている、または入っているべきDRSに対するポインタで初期設定する。また、m_number_of_elementsも、初期設定しなければならず、0≦i<m_number_of_elementsである各iに関して、m_args[i]をボックス要素引数に初期設定しなければならない。
【００６２】
ボックスに対する演算
xをボックスとする。
・x.number_of_elements()が、x.m_number_of_elementsフィールドを参照する。
・x.element_index(i)が、x.m_element_indices[i]を参照し、ここで0≦i<x.m_number_of_elementsである。
・x.element(i)が、x.m_ptr_containing_drs→m_elements[x.m_element_indices[i]]を参照し、ここで0≦i<x.m_number_of_elementsである。
・x.add_element(y):jをx.m_ptr_containing_drs→m_number_of_elementsとする;x.m_ptr_containing_drs→m_elements[j]をボックス要素yに設定する;1をx.m_ptr_containing_drs→m_number_of_elementsに加える;iをx.m_number_of_elementsとする;x.m_element_indices[i]をjに設定する;1をx.m_number_of_elementsに加える;iを戻す。
・0≦j<x.m_number_of_elementsである何らかのjに関して、x.element(j).kind()が、elem_logicalまたはelem_constraintに等しく、かつ0≦k<x.element(j).number_of_args()である何らかのkに関して、x.element(j).arg(k).kind()=arg_labelかつx.element(j).arg(k).label()=iである場合、x.dominates(i)は真を戻し、それ以外の場合、偽を戻す。これ

は、ボックスxが、ラベルiを有するボックスを(直に)含むことを意味する。
・xが、何らかの引数においてボックスラベルjを有する要素を含み(前述の「dominates」を参照)、i=jであるか、またはx.m_ptr_containing_drs→box(j).dominates_star(i)である場合、x.dominates_star(i)が真を戻し、それ以外の場合、偽を戻す。これは、ボックスxが、ラベルiを有するボックスを、場合により間接的に含むことを意味する。
・x.retire()が、x.m_ptr_containing_drsを空のポインタに設定する。これは、そのボックスがもはや使用されていないものと解釈され、解析中にdrsを変更する際、役立つ演算である。
・x.m_ptr_containing_drsが空のポインタでない場合、x.in_use()が真を戻し、空のポインタである場合、偽を戻す。
・また、以下の比較演算子$<_a$および$<_b$も参照されたい。

【００６３】

DRSの表現
DRSは、以下のフィールドを有するレコードとして表現される。
・整数、m_number_of_boxes
・m_number_of_boxesで長さが与えられるボックスベクトル、m_boxes
・整数、m_number_of_elements
・m_number_of_elementsで長さが与えられるボックス要素のベクトル、m_elements
・整数、m_number_of_markers

【００６４】

また、正規化されたDRSは、以下のフィールドも含む。
・m_number_of_markersで長さが与えられるペアの整数のリストのベクトル、m_marker_map
・ストリング、m_key

【００６５】

DRSを生成するとき、m_number_of_boxesを1に初期設定し、単一の空のボックスをm_boxesの中に入れなければならない。さらに、m_number_of_elementsおよびm_number_of_markersを0に設定しなければならない。

【００６６】

DRSに対する演算
xをDRSとする。
・x.number_of_boxes()が、x.m_number_of_boxesフィールドを参照する。
・x.box(i)が、x.m_boxes[i]を参照し、ここで0≦i<x.m_number_of_boxesである。
・x.number_of_elements()がx.m_number_of_elementsフィールドを参照する。
・x.element(i)がm_elements[i]を参照し、ここで0≦i<x.m_number_of_elementsである。
・x.add_empty_box():iをx.m_number_of_boxesとする;x.m_boxes[i]を空のボックスに設定する;1をx.m_number_of_boxesに加える;iを戻す。
・x.add_box(y):y.m_ptr_containing_drsが、xのアドレスであることを確実にする;iをx.m_number_of_boxesとする;x.m_boxes[i]をyに設定する;1をx.m_number_of_boxesに加える;iを戻す。
・x.add_next_marker(i):kをx.m_number_of_markersとする;マーカkに関するマーカ要素をx.box(i)の最後の要素として加える、前述した「add_element」を参照;1をx.m_number_of_markersに加える;kを戻す。
・x.add_next_marker(i,j):kをx.m_number_of_markersとする;マーカkに関するマーカ引数をx.box(i).element(j)の右端の引数として加える、前述した「add_arg」参照;1をx.m_number_of_markersに加える;kを戻す。
・x.dominates(i,j):x.box(i).dominates(j)を戻す。
・x.dominates_star(i,j):x.box(i).dominates_star(j)を戻す。
・x.finalize():以下の正規化アルゴリズムを呼び出す;xのボックス、要素、および引数をトラバースして、ストリングキーを生成し、x.m_keyの中に記憶する;xのボックス、要

素、および引数をトラバースし、マーカが出現するたびに毎回、x.m_marker_mapの中のそのマーカに関するリストにペア<i,j>を加え、ここでiはボックスのインデックス、jはマーカが出現したボックスの中の要素のインデックスである。

【００６７】

DRS正規化アルゴリズム

正規化アルゴリズムは、所与のDRSの内部表現を変更して、同じ情報を含みながらも、マーカ、ならびにボックス間およびボックス要素間の順序が、正規化されるようにする。例として、このアルゴリズムへの入力は、回収されたボックスおよび参照されないボックス要素を含むことが可能なDRSであり、使用されたマーカIDのシーケンスは、ギャップを含むことが可能である。出力は、入力DRSが変形されたものであり、ボックスの順序、およびボックスの中の要素の順序が、正規化されているように変更されており、これは、DRSのボックス、ならびに各ボックスの要素が、以下で定義する$<_b$の順序付けを満たすことを意味する。さらに、未使用のボックスまたはボックス要素は存在せず、DRSのマーカは、0で開始する途切れないシーケンスを形成する。

【００６８】

次に、いくつかの想定および用語について述べる。どのように順序付けを決定するかに関する以下の説明で、「...の場合、x$<_a$yは、偽である」という言い回し、またはそれに類する言い回しを使用するとき、これは、比較の結果が、明言されたものになるはずであり、さらなるステップが行われないことを意味する。「ベクトル[a...b]を定位置でソートして$<_z$を満たす」という言い回しを使用するとき、これは、効率的なソートアルゴリズム(Quicksortアルゴリズムなどの(例えば、非特許文献1参照))を要素ベクトルvector[a]、vector[a+1]...vector[b-1]、vector[b]に適用した後、u<vである場合には、vector[v]$<_z$vector[u]ではないはずであることを意味する。ソートアルゴリズムは、例として、結果が、所与の順序付けに違反しないことを確実にするが、順序付けが全体的ではない場合、複数の許可された結果が存在する可能性がある。

【００６９】

ベクトルには、0から開始してインデックス付けが行われ、したがって、サイズkのベクトルは、0からk-1までの範囲の有効なインデックスを有するものと想定する。ベクトルが導入される際、特に明記しない限り、ベクトルの要素の初期設定は行われないものと想定する。同様に、リストの要素は、0から開始して列挙され、したがって、k個の要素を有するリストは、その要素を位置0ないしk-1に有する。

【００７０】

マーカ出現三成分 (marker occurrence triple) は、整数の三成分<i,j,k>であり、ここでiはボックス番号、jはそのボックス内の要素番号、kはその要素内の引数の位置である。マーカ出現三成分は、その引数におけるマーカの出現を表し、またはjおよびkが、0になる場合ボックスのマーカ要素の出現を表す。

【００７１】

$<i_1,j_1,k_1>$および$<i_2,j_2,k_2>$をマーカ出現三成分とする。$<i_1,j_1,k_1><<i_2,j_2,k_2>$($<i_1,j_1,k_1>$が、辞書式順序で$<i_2,j_2,k_2>$に先行すると読む)が真であることは、以下のとおり定義される。

・$i_1<i_2$である場合、$<i_1,j_1,k_1><<i_2,j_2,k_2>$は、真である。

・$i_1=i_2$であり、かつ$j_1<j_2$である場合、$<i_1,j_1,k_1><<i_2,j_2,k_2>$は、真である。

・$i_1=i_2$であり、$j_1=j_2$であり、かつ$k_1<k_2$である場合、$<i_1,j_1,k_1><<i_2,j_2,k_2>$は、真である。

・以上3つの条件のどれも成立しない場合、$<i_1,j_1,k_1><<i_2,j_2,k_2>$は、偽である。

【００７２】

L_1およびL_2をマーカ出現三成分のリストとする。$L_1<L_2$(L_1が、辞書式順序でL_2に先行すると読む)が真であることは、以下のとおり定義される。

【００７３】

kを最大数とし、0≦i<kであるすべてのiに関して、t<uも、u<tも成立しないようにし、こ

ここでtおよびuはそれぞれL_1およびL_2のi番目の要素である。(L_1およびL_2はともに少なくともk個の要素を有する。)
・L_1の中の要素の数が、kである場合、$L_1<L_2$は、偽である。
・そうではなく、L_2の中の要素の数が、kである場合、$L_1<L_2$は、真である。
・それ以外の場合、$L_1<L_2$は、t<uと等価であり、ここでtおよびuはそれぞれL_1およびL_2の要素kである。

【００７４】

タイプTからタイプUへのマップは、タイプTの値の効率的なルックアップを可能にし、タイプUの対応する値を戻すデータ構造である。このマップは、関連付け(association) t→uを含むという言い方がされ、ここでtはタイプTの値、uはタイプUの値である。タイプTの各tに関して、少なくとも1つの関連付けt→uが存在する。

【００７５】

mが、TからUへのマップである場合、eがタイプTのものである表現m[e]は、タイプUのものである。mが、関連付けt→uを含み、eの値が、tである場合、m[e]の値は、uである。mが、tに関する関連付けを全く含まない場合、m[e]の値は、未定義である。マップを効率的に実装するための多くの周知のデータ構造、例えば、レッド-ブラックツリー(例えば、非特許文献2参照)およびハッシュテーブル(例えば、非特許文献3参照)が存在する。

【００７６】

タイプTからタイプUへのマルチマップは、Tの中の各tに関して複数の関連付けt→uを含む可能性があることを除けば、TからUへのマップと同様である。また、TからUへのマルチマップは、Tからタイプの値のセットへのマップとも同様であるが、空のセットを値tに関連付けるマップと、セットをtに全く関連付けないマップとの間の区別は、存在しない。

【００７７】

ボックス間の次の2つの順序関係$<_a$および$<_b$が、定義される。これらは、ボックス間およびボックス要素引数間の対応する順序関係に関しても再帰的に定義され、便宜のため、その順序関係も、$<_a$および$<_b$と呼ぶ。コンテキストから、順序付けが、ボックスに関するものか、ボックス要素に関するものか、またボックス要素引数に関するものかが、常に明らかとなる。2つのボックス、2つのボックス要素、または2つのボックス要素引数が比較されるとき、その2つは、常に同じDRSに属する。最後に、DRSの中のボックスのインデックスに関する順序付け$<_a$および$<_b$は、ボックス自体に関する順序付けで定義される。

【００７８】

ボックス要素引数に関する$<_a$
DRSの中のボックス要素引数xとyとの間の関係$<_a$は、テーブル1のマトリックスに従って定義される。

【００７９】

【表１】

ボックス要素引数に関する$<_a$		y.kind()			
		null_arg	arg_marker	arg_label	arg_value
x.kind()	Null_arg	偽	真	真	真
	Arg_marker	偽	偽	真	真
	Arg_label	偽	偽	以下の(a)を参照	真
	Arg_value	偽	偽	偽	以下の(b)を参照

表１

【００８０】
(a)x.kind()およびy.kind()が、arg_labelである場合、$x <_a y$は、drs.box(x.label())$<_a$drs.box(y.label())と等価である。
(b)x.kind()およびy.kind()が、arg_valueである場合、$x <_a y$は、x.data()<y.data()と等価であり、ここで<はデータ値に関する全体的順序付けである。
【００８１】
ボックス要素引数に関する$<_b$
DRSの中のボックス要素引数xとyとの間の関係$<_b$は、テーブル2のマトリックスに従って定義される。
【００８２】
【表２】

ボックス要素引数に関する$<_b$		y.kind()			
		Null_arg	arg_marker	arg_label	arg_value
x.kind()	Null_arg	偽	真	真	真
	Arg_marker	偽	以下の(c)を参照	真	真
	Arg_label	偽	偽	以下の(d)を参照	真
	Arg_value	偽	偽	偽	上の(b)を参照

表２

【００８３】
(c)x.kind()およびy.kind()が、arg_markerである場合、$x <_b y$は、x.marker()<y.marker()と等価である。
(d)x.kind()およびy.kind()が、arg_labelである場合、$x <_a y$は、x.label()$<_a$y.label()と等価である。drs.box(x.label())$<_a$drs.box(y.label())を(a)の場合と同様に使用することもできたが、$<_b$が適用されるときはいつでも、これらの試験は等価であることが知られており、また整数比較x.label()$<_a$y.label()の方が効率的であることに留意されたい。
【００８４】
ボックス要素に関する$<_a$
DRSの中のボックス要素xとyとの間の関係$<_a$は、テーブル3のマトリックスに従って定義される。
【００８５】
【表３】

ボックス要素 に関する$<_a$		y.kind()			
		null_elem	elem_marker	elem_constraint	Elem_logical
x.kind()	Null_elem	偽	真	真	真
	Elem_marker	偽	偽	真	真
	Elem_constraint	偽	偽	以下の(e)を参照	真
	Elem_logical	偽	偽	偽	以下の(e)を参照

表３

【００８６】
(e)x.king()およびy.kind()がともに、elem_constraintであるか、elem_logicalである場合、$x<_a y$は、以下のステップによって判定される。
【００８７】
ステップ1.n_x=x.number_of_args()とし、またn_y=y.number_of_args()とする。$n_x<n_y$である場合、$x<_a y$は、真である。$n_x>n_y$である場合、$x<_a y$は、偽である。それ以外の場合、次のステップに進む。
【００８８】
ステップ2.s_x=x.sem_kind()とし、またs_y=y.sem_kind()とする。$s_x<s_y$である場合、$x<_a y$は、真である。$s_x>s_y$である場合、$x<_a y$は、偽である。それ以外の場合、次のステップに進む。
【００８９】
ステップ3.s_xおよびs_yが、null_sem、no_sem、または値ではない場合、次のステップに進む。それ以外の場合、x.word()とy.word()を辞書式順序で比較する。x.word()が、y.word()に先行する場合、$x<_a y$は、真である。y.word()が、x.word()に先行する場合、$x<_a y$は、偽である。どちらでもない場合、ステップ5に進む。
【００９０】
ステップ4.s_xおよびs_yがエンティティまたは関係ではない場合、次のステップに進む。それ以外の場合、x.schema_node()とy.schema_node()を比較する。x.schema_node()が、y.schema_node()に先行する場合、$x<_a y$は、真である。y.schema_node()が、x.schema_node()に先行する場合、$x<_a y$は、偽である。どちらでもない場合、次のステップに進む。
【００９１】
ステップ5.ボックス要素xおよびyは、同数の引数(前述したn_xおよびn_y)を有することが分かっている。整数の変数iを0に設定する。
【００９２】
ステップ6.$i=n_x$である場合、$x<_a y$は、偽である。それ以外の場合、aおよびbをそれぞれ、ボックス要素xおよびyの位置iにおける引数とする。$a<_a b$である場合、$x<_a y$は、真である。そうではなく、$b<_a a$である場合、$x<_a y$は、偽である。それ以外の場合、iを1だけ増加させ、このステップを繰り返す。
【００９３】
ボックス要素IDに関する$<_a$
DRSにおけるボックス要素ID、iとjとの間の関係$<_a$は、DRSのそれぞれID、iおよびjを有するボックス要素間の関係$<_a$を決定することによって決定される。
【００９４】

ボックス要素に関する$<_b$
DRSの中のボックス要素xとyとの間の関係$<_b$は、テーブル4のマトリックスに従って定義される。

【００９５】

【表４】

ボックス要素に関する$<_b$		y.kind()			
		null_elem	Elem_marker	elem_constraint	elem_logical
x.kind()	null_elem	偽	真	真	真
	elem_marker	偽	以下の(f)を参照	真	真
	elem_constraint	偽	偽	以下の(g)を参照	真
	elem_logical	偽	偽	偽	以下の(g)を参照

表４

【００９６】

(f)x.kind()およびy.kind()が、elem_markerである場合、$x<_b y$は、x.marker()<y.maker()と等価である。
(g)x.kind()とy.kind()がともに、elem_constraintであるか、またはelem_logicalである場合、ステップ6では、ボックス要素引数aとbとの間の比較が、$<_b$を使用して行われることを除けば、前述した$x<_a y$に関する(e)の場合と全く同じように決定される。

【００９７】

ボックスに関する$<_a$
DRSの中のボックスxとyとの間の関係$<_a$は、以下のステップを使用して計算される。
ステップ1.yが、drs.box(0)である場合、$x<_a y$は、偽である。そうではなく、xが、drs.box(0)である場合、$x<_a y$は、真である。そうではなく、DominatesStar(x,y)である場合、$x<_a y$は、真である。そうではなく、DominatesStar(y,x)である場合、$x<_a y$は、偽である。それ以外の場合、n_xおよびn_yをそれぞれ、ボックスxおよびyの中の要素の数として、整数の変数iを0に設定し、次のステップに進む。
ステップ2.$i=n_x$である場合、$x<_a y$は、偽である。そうではなく、$i=n_y$である場合、$x<_a y$は、真である。それ以外の場合、aおよびbをそれぞれ、ボックスxおよびyの要素iとする。$a<_a b$である場合、$x<_a y$は、真である。そうではなく、$b<_a a$である場合、$x<_a y$は、偽である。それ以外の場合、iを1だけ増加させ、このステップを繰り返す。

【００９８】

IDに関する$<_a$
DRSの中のボックスID、iとjとの間の関係$<_a$は、DRSのそれぞれID、iおよびjを有するボックス間の関係$<_a$を決定することによって決定される。

【００９９】

ボックスに関する$<_b$
DRSの中のボックス、xとyとの間の関係$<_b$は、ステップ6では、ボックス要素引数aおよびbの比較が、$<_b$を使用して行われることを除けば、前述した$x<_a y$に関するのと全く同じように判定される。

【０１００】

ボックスIDに関する$<_b$

DRSの中のボックスID、iとjとの間の関係$<_b$は、DRSのそれぞれID、iおよびjを有するボックス間の関係$<_b$を決定することによって決定される。

【０１０１】

以上の考察から、$<_a$の順序付けは、マーカに関係なくソートを行い、一方、$<_b$の順序付けは、マーカに関してもソートを行うことがわかる。

【０１０２】

正規化アルゴリズム

正規化アルゴリズムは、いくつかのステージを有するものとして理解することができる。第1のステージでは、あらゆる未使用のボックスおよびボックス要素が、DRSからパージされる。第2のステージでは、マーカを無視する予備の順序付けが、ボックスおよびボックス要素に課される。(マーカだけが異なっているボックスおよびボックス要素は、あたかも等価であるかのように扱われる。)この予備の順序付けに基づき、マーカに関する順序付けが確立され、マーカには、連続シーケンスで番号が付けなおされる。最後に、予備の順序付け、およびマーカの順序付けに基づき、決定的な順序付けが、ボックスおよびボックス要素に課される。マーカに関する順序付けは、一意的には決定されず、どの可能なマーカの順序付けが選択されるかに関わらず、最終的な再番号付けの後、結果は同じであるという好ましいプロパティが得られる。

【０１０３】

図5Bは、生成された様々なDRS構造が、どのように後の処理のために正規化されるかを示す流れ図である。入力DRSの正規化は、構文解析構成要素202、または意味解析構成要素204、または何らかの他の外部構成要素によって行われるのが可能であることに留意されたい。ただし、入力DRSの正規化は、例として、意味解析構成要素204の中のSIC210によって行われることが可能である。また、図5Bのアルゴリズムによって示されるプロセスに関する擬似コードは、本明細書に付録Aとして含められており、図5Bの考察全体にわたって付録Aへの何らかの参照が行われることにも留意されたい。

【０１０４】

まず、正規化構成要素(どこに存在していても)が、入力DRS(すなわち、ボックスのアレイ)を受け取る。これが、ブロック251で示される。

【０１０５】

ときとして、入力DRSの生成中、使用されないボックスまたはボックス要素が含められる可能性がある。したがって、DRSの中の未使用のボックスおよびボックス要素が、特定され、ブロック253で示されるとおり、削除される。次に、255で示されるとおり、残っているボックスに番号が付け直される。未使用のボックスを削除すること、およびそのボックスに番号を付け直すことが、付録Aにおけるステップ1で示されている。

【０１０６】

次に、ボックスの予備のソートが、ボックスのインデックスに基づいて行われる。これは、図5Bのブロック257で示されている。

【０１０７】

ボックスが、そのインデックスに基づいて予備的にソートされた後、ボックス要素の予備のソートが、そのボックス要素のマーカを無視して行われる。この予備のソートは、付録Aにおけるステップ3-7で行われ、図5Bのブロック259で示されている。より具体的には、ステップ3で、アルゴリズムが、ボックス要素IDに関して$<_a$ソートを行う。次に、ボックス要素は、そのソートに従って並べ替えられ、ボックスまたは未使用のボックスによって前に占有されていた記憶スペースが、開放される。これが、付録Aのステップ4および5で行われる。ボックス要素がソートされ、並べ替えられた後、ボックス自体が、更新されて、再び正しいボックス要素を有するようになる。これは、付録Aのアルゴリズムのステップ6で行われる。同様に、ボックス要素自体も更新され、再び正しいボックスを参照するようになる。これは、アルゴリズムのステップ7で行われる。

【０１０８】

付録Aのアルゴリズムの最初の7つのステップの後、およびブロック257で示されるボック

スのインデックスに基づくボックスの予備のソート、および図5Bのブロック259で示されるボックス要素の予備のソート(マーカを無視する)の後、ボックスおよび要素が、マーカを無視して完全にソートされていることがわかる。これは、ボックスおよびボックス要素のそれぞれが、前述した辞書式順序のソートおよび比較のアルゴリズムに従ってソートされていることを意味する。完全にソートが済んでいないボックスおよびボックス要素は、マーカに関してだけ異なるものだけである。そのようなボックスおよびボックス要素は、アルゴリズムのこの時点では等価としてランク付けされる。

【０１０９】

次に、最終ソートを行うのに必要な情報およびデータ構造を得るためにDRSをパススルーすることによってステップ8-11が行われる。最終ソートでは、ボックスおよびボックス要素がインデックスに従ってソートされるだけでなく、マーカによってもソートされる。したがって、アルゴリズムは、マーカに固有の連続値を与えることで開始する。ただし、このプロセス中、マーカと要素またはボックスとの間の同一性が全く失われないようにすることが望ましい。したがって、ステップ8および9が、この同一性を保持するマップを記録する。

【０１１０】

ステップ8で、マーカ要素、制約要素、および論理要素に関して、マーカからマーカ出現三成分のリストへのマップが生成される。マップの中のリストにマーカ出現三成分が追加される順序により、各リストが、三成分上で辞書式順序に順序付けられるのが確実になることがわかる。

【０１１１】

ステップ9は、マーカ出現三成分のリストからマーカへの逆のマップを生成してマーカに関する順序付けを確立する。マップを生成することは、図5Bのブロック261で示される。

【０１１２】

同一性を保持するマップが生成された後、マーカに新しいマーカ値が割り当てられ、その新しいマーカ値で要素が更新される。これは、図5Bのブロック263で示される。マーカに対する新しいマーカ値の割当ては、付録Aのステップ7で行われ、新しいマーカ値で要素を更新すること(したがって、要素に連続シーケンスで番号を付け直すこと)は、付録Aのステップ11で行われる。

【０１１３】

最後に、予備の順序付け、およびマーカの順序付けに基づいて最終ソートが行われる。これが、図5Bのブロック265で示され、付録Aのステップ12でさらに示される。出力は、ボックスおよびボックス要素の順序が正規化されているように変更されたDRSである。これは、DRSのボックス、および各ボックスの要素が、上記で定義した$<_b$の順序付けを満たすことを意味する。さらに、未使用のボックスまたはボックス要素が存在せず、DRSのマーカがゼロから始まる途切れのないシーケンスを形成する。

【０１１４】

キーストリング

この表現構造においてDRSが与えられると、DRSと1対1関係にあるストリングを出力する手続きを実施することができる。正規化されたDRSの場合、そのストリングも、それに対応して正規化されている。DRSの正規のストリング表現が与えられると、元のDRSと区別することのできない現在の表現構造におけるDRSをもたらす逆の手続きも実施することができる。そのようなシリアル化された正規の表現を有することのいくつかの利点が存在し、例えば、

・ストリングをファイルもしくはデータベースの中で存続すること、またはシリアルストリームを介して伝送することができる。そのようなファイルもしくはデータベースの読取り側、またはそのようなストリームの受取り側は、元の正規化されたDRSを再構成することができる。

・ストリングをデータベースシステムにおけるキーとして使用することができる。

・標準のストリングユーティリティを使用してストリングを等価関係(およびその他の適

切な関係)に関して効率的に比較することができる。
【０１１５】
そのような手続きは、DRSの再帰的トラバース、および遭遇したボックス、ボックス要素、およびボックス要素引数を簡潔に曖昧さなしに記述するストリングの生成を含む。
【０１１６】
逆の手続きは、そのようなストリングを構文解析し、DRSをボックスごとに、ボックス要素ごとに、およびボックス要素引数ごとに再構成することを含む。
【０１１７】
3.アプリケーションスキーマ
図6は、本例に適したスキーマ300を示している。スキーマ300は、エンティティが、円形(または長円形)の中で示され、また関係が、ボックスの中で示されるエンティティおよび関係のグラフである。例えば、スキーマ300は、アプリケーションが、様々な特定の電子メールメッセージを送信すること、および削除することをサポートするのを示している。これは、電子メールアイテムが、「DeleteAct」または「InitiateEmailAct」の目標となる可能性があるため、示されている。
【０１１８】
さらに、電子メールメッセージは、文字ストリングで示される「Name」を有する「Person」で指定される送信側または受信側を有することが可能である。また、電子メールアイテムは、送信された時刻、および件名（subject）によって特定されることも可能であり、時刻および件名も、文字ストリングで表される。
【０１１９】
本発明の意味解析構成要素204のジョブは、構文解析(図5で示したUDRSボックス構造を含む)を受け取り、図6で示したアプリケーションスキーマ300に従ってその構文解析を正確に解釈することである。次に、この解釈が、アプリケーションに送られ、容易に理解されることが可能である。これを行うため、意味解析構成要素204はまず、意味マッピング規則をUDRSに適用する。
【０１２０】
4.意味マッピング
意味マッピングは、1組のUDRSと1組のSemDRSフラグメントとの間の関係である。意味マッピング規則は、特定の形態のUDRSの下位部分にマッチする左側と、あるSemDRSフラグメントを指定する右側を有する表現である。さらに、この規則は、左側の各位置に関して、マーカにマッチする対応関係を含み、変数が単に落とされるべきものでない限り、対応する右側の位置が特定される。構文上、この対応関係は、単にマッピング規則の両側で同じマーカを使用することによって表すことができる。したがって、意味マッピング規則は、UDRSを1組のSemDRSフラグメントに書き換えるために使用することができる。
【０１２１】
1組の意味マッピング規則が、意味マッピングを示し、そのセットの中の何らかの規則に関して、その左側がUDRSの下位部分にマッチし、またその右側がUDRSから置き換えられた対応する変数を有するSemDRSフラグメントである場合、意味マッピングの構成要素が、左側の入力UDRSと右側のSemDRSフラグメントとの間の関係であるようになっている。もちろん、セットの中の複数の規則が、UDRSに適用される場合、その規則セットによって表される意味マッピングを適用することの複数の結果が存在する。
【０１２２】
一例としての実施形態では、意味マッピング規則の多くが、以下のように[|word(x)]→[|type(x)]の形式である。
[|message(x)]→[|MailItem(x)]
[|email(x)]→[|MailItem(x)]
[|email(e,x,y)]→[|InitiateEmailAction(e)]
【０１２３】
一例としての実施形態では、意味マッピング規則の一般的ケースが、任意のUDRSを左側で

マッチさせることができる。例えば、一般的ケースは、例として、ボックスおよび論理演算子のある構成を探す。また、右側も、やはり例として、(場合により、入れ子型の)ボックスおよび論理演算子を含め、任意のSemDRSを指定することができる。最後に、変数対応関係は、例として、マーカの出現の数が、左側から右側に変化するのを可能にするだけ十分に一般的である。より一般的な規則の例は、以下のとおりである。

[|make(e,x,[font(y),bold(y)])→[|AlterFont(e),Alters(e,z),HasNewProperty(z,y),Boldface(y)];かつ

[|～[save(e,x,y)],mail(y)]→[|DeleteMailAction(e),HasTarget(e,y),MailItem(y)]

【０１２４】

以上の例が示すとおり、ときとして、任意の所与の意味マッピング規則において複数の語をマッチさせようと試みることが望ましい。例として、以下の幾分異なった仕方で扱われる2つのケースが存在する。

【０１２５】

第1に、意味マッピング規則を語の特定のシーケンスに関して、場合により、ステミング(stemming)等に関して正規化された語の特定のシーケンスに関して生成することができる。これは、「ホットドッグ」、またはファーストネームおよびラストネームを含む個人の固有名などの語に適用される。

【０１２６】

第2に、意味マッピング規則をいま前述した2つの例におけるように、ある構文の構成で現れる語に関して生成することができる。第1のケースでは、語のシーケンスが、用語集の中のマルチワードエンティティとして現れる場合、このシーケンスは、単一の語彙単位（lexical unit）であり、このシーケンスの意味マッピングは、単一の語と同じ仕方で扱われる。また、語のシーケンスが、名前付きエンティティである場合も、意味規則は、そのシーケンスを単一の単位として扱う。

【０１２７】

意味マッピング規則は、例として、アプリケーション開発者によって開発されることが可能であり、あるいは、アプリケーション開発者からの十分な入力に基づき、またスキーマから帰納することによって生成することができる。

【０１２８】

E.意味解析層の全体的動作

SIC210によって受け取られた情報はすべて、IFG212に提供され、意味マッピングおよびアプリケーションスキーマ、ならびにデータ記憶装置215の中のデータが、IA214に提供される。IFG212(およびその様々な下位構成要素)およびIA214(およびその様々な下位構成要素)のサービスを呼び出すため、SIC210は、システムの中の様々な構成要素の間でデータを整理するサービスインボーカ216を含む。この情報に基づき、IFG212は、SIC210に提供され、次にデータ記憶装置215の中に記憶される解釈フラグメントを生成する。UDRSに意味マッピング規則を適用することによって生成されるもの、以降、Blurbと呼ぶ、入力の中の要素をストリングとして解釈することによって生成されるもの、および堅牢性戦略を介して生成されるものを含む、3つの異なる基本タイプの解釈フラグメントが存在する。第1のタイプの解釈フラグメントが生成される機構は、例として、主に言語独立であることが可能であり、一方、残りの2つは、例として、トークン化されたストリング、構文解析ツリー、論理形式、および名前付きエンティティに関する言語依存の推論によって生成することができる。解釈フラグメントは、SIC210がIFG212に提供するインターフェースを介してデータ記憶装置215に書き戻される。

【０１２９】

解釈フラグメントに基づき、SIC210は、ISS生成構成要素218においてパターンおよび初期探索状態を生成する。初期探索状態は、論理構造およびボックス構造が除去された部分的に完成したSemDRS構造に対応する。初期探索状態が、初期探索状態で実現される解釈フラグメント(またはパターン)をどのようにアセンブルするかに関する選好（preference）とともに、IA214に提供される。

【０１３０】
IA214は、1組の解決パターンを出力する。解決パターンは、完全に解釈されたSemDRSに、可能な場合、論理構造およびボックス構造を復元することによって変換される。これは、DRS構造復元生成器220によって行われ、完全に解釈されたSemDRSが、SIC210によって出力される。
【０１３１】
F.意味解析層のより詳細な動作
図4は、意味解析構成要素204における全体的データフローをよりよく示す流れ図である。このブロックの多くは、詳細な説明の以降の図に関連してより詳細に説明する。
【０１３２】
まず、SIC210が、構文解析、アプリケーションスキーマ、および意味マッピング規則を受け取り、SIC210内部で保持されるデータ記憶装置215を初期設定する。これは、図4のブロック320で示されている。データ記憶装置215は、入力DRSの論理構造およびボックス構造のレコードであるUDRSを含み、意味マッピング規則が、この構造の変更を許可するときはいつでも、更新される。また、データ構造は、以下も含む。
元のシーケンスにおけるトークンのリストであるトークンリスト。
カバーするトークン、関連するコスト、およびDRSの論理構造およびボックス構造を復元する際に中に入れられるべき元のボックスを有する1組のSemDRSフラグメント。このフラグメントはすべて、IFG212の中の異なる下位構成要素によって生成される。このSemDRSフラグメントは、IA214に入力される初期探索状態のコアメンバになる。
初期探索状態の中のパターンをどのように結合して解決パターンを得るかに関する命令である選好。
【０１３３】
データ記憶装置215が、データ記憶装置215の中のデータにアクセスし、追加を行い、あるいは更新を行うシステムの中のその他の構成要素に対する必要なインターフェースを提供する。
【０１３４】
データ記憶装置215が初期設定された後、SIC210の中のサービスインボーカ216が、入力文および意味マッピング規則をIFG212に提供し、IFG212のサービスを呼び出す。これは、図4のブロック322で示されている。
【０１３５】
次に、IFG212の中のUDRS意味マッピング処理エンジン222が、すべての適用可能な意味マッピング規則をUDRSに適用する。これが、ブロック324で示されている。この規則は、コンパイル時に構成することができ、あるいは実行時に他の規則およびスキーマから構成できることに留意されたい。意味マッピング規則を適用した結果は、1組のUDRSボックス要素と1組のSemDRSボックス要素との間のマッピングを指定する1組の解釈フラグメントである。これらの解釈フラグメントのトークンカバレッジは、マッチされたUDRSボックス要素に対応する入力文の中のトークンのセットとして定義される。トークンカバレッジおよび割当てられたコストを有する解釈フラグメントが、データ記憶装置215に送られ、記憶される。これは、図4のブロック326で示されている。また、もちろん、UDRSの同じ部分に複数の規則を適用するのが可能であることにも留意されたい。
【０１３６】
また、UDRSの中の情報は、「Blurb」として解釈することもできる。Blurbの存在は、入力DRS(つまりUDRS)を解釈した結果によって示唆され、したがって、SIC210が、BlurbがIFG212によって示唆されているかどうかを判定する。これが、ブロック332で示されている。示唆されている場合、サービスインボーカ216が、Blurb UDRS処理エンジン224を呼び出し、このエンジン224は、構文解析ツリーまたは論理形式に基づいてBlurb処理を行い、そのBlurb処理に基づいて解釈フラグメントを生成する。結果の解釈フラグメントが、ブロック334で示されるとおり、IFS215に提供されるトークンカバレッジおよびコストである。
【０１３７】

次に、SIC210が、堅牢性戦略を使用すべきかどうかを判定する。堅牢性戦略は、図11に関連してより詳細に説明しており、基本的に、UDRSボックス要素のセットに対してではなく、ストリングごとに書換え規則を適用し、さらなるBlurb処理を行うことを含む。堅牢性戦略を使用すべきかどうかを判定することは、ブロック340で示されている。

【０１３８】

堅牢性戦略を使用すべき場合、サービスインボーカ216が、IFG212の中の堅牢性戦略処理エンジン226を呼び出し、このエンジン226が、堅牢性戦略処理に基づいて解釈フラグメントを生成する。この処理は、構文解析ツリー、論理形式、および名前付きエンティティを検査して、UDRSの中にないアイテムを特定し、そのアイテムから解釈フラグメントを生成することを含む。ブロック342で示されるとおり、これらの解釈フラグメントが生成され、そのトークンカバレッジおよびコストが、データ記憶装置215に送られる。

【０１３９】

SIC210が、ブロック346で示されるとおり、データ記憶装置215の中に記憶されたデータを更新し、解釈フラグメントを追加のパターンに変換する。

【０１４０】

このように生成されたパターンが、SIC210のISS生成構成要素218によって初期探索状態にパッケージ化される。これが、ブロック348で示されている。初期探索状態は、入力文のすべてのトークンが、1つ限りのパターンによってカバーされるようにパッケージ化される。形式上、初期探索状態は、1組のパターン、およびパターンがどのようにアセンブルすべきかを制約する選好を含む。ISS生成構成要素は、例として、この入力文に関するすべての可能な初期探索状態を形成する。

【０１４１】

もちろん、初期探索状態は、グループで生成することもできる。例えば、意味マッピング規則をまず、UDRSのボックス構造に適用することからもたらされたパターンだけに関して初期探索状態を生成することができる。次に、Blurb処理から生成されたパターンを初期探索状態を生成するプロセスに追加することができる。同様に、網羅的に、または時間制限などの何らかの他の制約に達するまで堅牢性戦略を使用して生成されたものを含むパターンに関して、初期探索状態を生成することができる。あるいは、すべての可能な探索状態を即時に生成することができる。

【０１４２】

いずれにしても、生成された初期探索状態は、スキーマとともに、SIC210からIA214に提供される。これが、ブロック350で示される。IA214は、初期探索状態の一部としてIA214にも提供されるローカルの選好に従って初期探索状態の中のパターンをアセンブルするか、または結合することによって解決パターンを生成する。解決パターンは、スキーマおよびローカルの選好が与えられると、全体的に最も好ましい総合的解釈を表す。解決パターンの生成は、ブロック352で示されている。

【０１４３】

IA214は、解決パターン生成構成要素228を使用して解決パターンを生成する。解決パターンは、SIC210に戻され、SIC210は、ブロック354で示されるとおり、その解決パターンを受け取り、解決パターンにボックス構造および論理構造を復元する。これにより、完全に解釈されたSemDRSが生成され、このことは、図4のブロック356で示されている。ボックス構造および論理構造は、可能な場合、解決パターンの中の保存された構造情報および継承情報(heritage infromation)、ならびにSIC210によって保持されるその他のデータ構造を使用して復元される。

【０１４４】

図7は、初期設定ブロック320をより詳細に示す流れ図である。図5に関連して述べたとおり、各入力UDRSは、例として、論理要素の動的アレイとして実装される。論理要素は、ボックスによって表され、各ボックスは、そのアレイへのインデックスを表す。したがって、初期設定中、データ記憶装置215がまず、図7のブロック400で示されるとおり、入力UDRSに関するボックス構造を記録する。これが、UDRSデータ構造である。

【０１４５】
次に、データ記憶装置215は、入力ストリング206に対応するトークンのリストを記録する。これは、図7のブロック402で表されている。例1、「KDからではないすべての電子メールを削除する (Delete all emails not from KD) 」における入力ストリングに関する1つのトークンリストは、以下のとおりである。

Delete all emails not from KD

1 2 3 4 5 6

【０１４６】
次に、データ記憶装置215は、図7のブロック404で示されるとおり、IFGによって生成される意味マッピング結果を記憶するのに使用するデータ構造を初期設定する。このデータ構造は、最初、空であるが、最終的には、IFG212における処理の結果によるパターンで埋められる。

【０１４７】
例として、何がパターンを構成するかについてもう少し詳細に見ることが役立つ可能性がある。何らかのパターンPの要素であるエンティティ変数が、T(X)で示され、ここでTはエンティティタイプ名、XはPにおいて固有の記号である。同様に、パターンPの要素である関係が、T(X,S,D)で表され、ここでTは関係タイプ、XはPにおいて固有の記号、Sはソースエンティティに関する記号、Dは宛先エンティティに関する記号である。パターンは、[EmailItem(X)]のような単一のエンティティ変数からなることが可能である。また、パターンは、以下のように、関係で分けられた2つ、またはそれより多くのエンティティ変数で表すこともできる。
EmailItem(X),HasSender(Y,X,Z),Person(Z)
PatternListデータ構造を初期設定することが、ブロック404で示されている。

【０１４８】
次に、データ記憶装置215が、入力UDRSの中のDRS要素と入力ストリング206の中のトークンとの間のマッピングであるデータ構造を初期設定し、このデータ構造が、IFG212の中の下位構成要素(例えば、図8のブロック418)によって使用されて、適用された意味マッピング規則のトークンカバレッジが判定される。これが、ブロック406で示されている。一部のトークンは、ボックス要素に対応しないが、それでも、ボックス構造で表されることに留意されたい。

【０１４９】
例1では、トークンに対するDRS要素の初期マッピング(drselems2t)が、以下のとおりである。
KD(y)が、トークン6をカバーする
email(x)が、トークン3をカバーする
from(x,y)が、トークン5をカバーする
delete(e,x)が、トークン1をカバーする

【０１５０】
トークン4および2は、どのボックス要素にも対応していないことに留意されたい。ただし、トークン4および2の意味の寄与分は、ボックス構造の中に記録されているため、失われていない。

【０１５１】
初期設定が完了した後、IFG212によるデータのアクセスが可能にされ、またIFG212の中のスキーマエンジン222が、すべての適用可能な意味マッピング規則をUDRSに適用する。このプロセスは、図8で示した流れ図でより詳細に示している。

【０１５２】
前述した例1では、以下のセットの意味マッピング規則が与えられているものと想定する。
email(x)→EmailItem(x)

from(x,y)→HasSender(x,y)
from(x,y)→HasSentTime(x,y)
delete(e,x)→DeleteAct(e)HasTarget(e,x)

【０１５３】

以上の規則を適用する際、UDRS意味マッピング処理エンジン222がまず、図8のブロック410で示されるとおり、意味マッピング規則を選択する。次に、UDRS意味マッピング処理エンジン222は、意味マッピング規則の左側を記録された入力UDRSボックス構造の中のサブセットのボックスのすべてと比較する。UDRS意味マッピング処理エンジン222は、ブロック412で示されるとおり、意味マッピング規則の左側が、ボックス構造の中の何らかのサブセットのボックスに合致するかどうかを判定する。

【０１５４】

対応しない場合、UDRS意味マッピング処理エンジン222は、単に次の意味マッピング規則に進み、検査されるべき意味マッピング規則が存在しなくなるまで処理を続ける。これが、ブロック414および416で示されている。

【０１５５】

しかし、ブロック412で、意味マッピング規則の左側が、入力UDRSのボックス構造のサブセットにマッチした場合、そのマッピングのトークンカバレッジは、選択された規則の左側にマッチしたUDRSボックス要素のトークンカバレッジの組み合わせ（union）であると判定され、またボックス番号は、すべてのマッチしたUDRSボックス要素を含むものである。これが、ブロック418で示されている。次に、選択された意味マッピング規則の右側のSemDRSフラグメントが、トークンカバレッジ、ボックス番号、およびそのSemDRSフラグメントに関連するスコアまたはコストとともにIFS215に送られる。これが、ブロック420で示されている。コストは、マッチの中の語数や、SemDRSフラグメントの中でオブジェクトが現れる確率をコストの基準とするなどの任意のいくつかの周知の方法を使用して計算することができ、あるいは、単にすべての意味マッピング規則に関して固定値(例えば、1.0)を有することが可能である。

【０１５６】

次に、UDRS意味マッピング処理エンジン222が、ブロック420で示されるとおり、ルースニング(loosening)を適用し、選好を得る。ルースニングを適用するため、パターンのリスト(PatternList)の中の他のパターンに対する明示的な結合が、抽象エンティティで置き換えられ、選好が生成される。ルースニングについては、図9に関連して以下でより詳細に述べる。

【０１５７】

ルースニングを適用した後、パターンとDRS要素との間のマッピング(pat2drselemsデータ構造)が更新される。これを行うため、UDRS意味マッピング処理エンジン222が、PatternListに追加されたばかりのパターンに関連するpat2drselemsマッピングの中のペアを、そのパターンの生成をトリガした入力UDRSボックス構造の中のボックスに追加する。言い換えれば、マッピング規則の左側にマッチしたボックスと、そのマッチによって生成されたPatternListの中のパターン(すなわち、マッピング規則の右側)との間でマッピングが生成される。これが、ブロック422で示されている。

【０１５８】

次に、パターンを許可したSemDRSフラグメントのトークンカバレッジに基づくトークンへのパターンのマッピング、pat2tが、ブロック424で示されるとおり、更新される。スキーマエンジン222が、drselem2tマッピングにアクセスして、出力パターンに関連するpat2tマッピングの中のペアを入力ストリングの位置に追加する。これが完了した後、スキーマエンジン222が、ブロック414で示されるとおり、追加の意味マッピング規則を処理する必要があるかどうかを再び調べる。

【０１５９】

例1においてマッピング規則のすべてが適用された後、SIC210の中のデータ構造は、以下のとおりとなる。

UDRSボックス構造は、変更されていない。
TokenList構造は、変更されていない。
【０１６０】
PatternList構造は、以下のパターンを含む。この場合、各パターンの中のエンティティおよび関係には、最終的に再構成されることになるボックス構造の中のボックスでラベルを付ける。また、マッピング、pat2tも以下のとおり指定される。
P_1:EmailItem[3] トークン3をカバーする
P_2:E_2[3] HasSender[5] E_1[1] トークン5をカバーする
P_3:DeleteAct [4] HasTarget [4] E_3[3] トークン1をカバーする
P_7:E_4[3] HasSentTime [5] E_5[1] トークン5をカバーする
【０１６１】
以上のパターンP_1...P_nの中のエンティティE_1...E_nは、ルースニングプロセスの結果として導入される。次に、このプロセスを図9に関連してより詳細に説明する。
【０１６２】
第1に、パターンリストデータ構造からの関係パターンが選択される。これが、図9のブロック480で示される。次に、選択された関係パターンの中のソースエンティティおよび宛先エンティティが、抽象エンティティ(E_1-E_nで表されるもののような)で置き換えられる。これが、ブロック482で示されている。
【０１６３】
次に、抽象エンティティを有するパターンが、PatternListデータ構造の中の他のパターンにどのように関連するかに関する選好が、生成される。これが、ブロック484で示されている。
【０１６４】
例1に関連して、入力DRSの中のエンティティxとyとの間の関係が、(x,y)からの関係の解釈、ならびにDRSの中のxおよびyの残りの出現において緩められていることがわかる。これを行うことにより、例えば、入力UDRSの中のKD(y)に対応するエンティティとスキーマの中のHasSender関係の指定との間の関係が、直接でなくてもよいことに留意されたい。この場合も、図6で示したスキーマ300を見ると、これは望ましいルースニングである。例えば、KD(y)は、ストリングとして解釈される可能性が高く(以下で説明するBlurb構成要素により)、HasSenderが、EmailItemsをタイプPersonのエンティティに結合する。このように関係を緩めることにより、IA構成要素214が、personエンティティを導入し、HasName関係を介してストリングをそのエンティティに関連付けることによってKD(y)のストリング解釈をHasSenderの宛先に関連付けることができる。
【０１６５】
また、入力UDRSの中のエンティティeとHasTarget関係のソースとの間の関係は、緩められていないこともわかる。これは、EとHasTarget関係との間の結合が、単一の意味マッピング規則の中で明示的に行われていることで、前述したケースとは異なっている。したがって、意味マッピング規則の右側は、スキーマにおける正当な結合を表し、したがって、ルースニングの必要性が回避されるものと想定される。
【０１６６】
例1において、意味マッピング規則およびルースニングプロセスを適用することで(図8のブロック420で示される)、以下の選好が生じる(以下で、Blurb処理に関連してパターンP4について述べている)。
H_1:P_1のEmailItem P_2のE_2に近い
H_2:P_2のE_1 P_4のBlurb(「KD」)に近い
H_3:P_3のE_3 P_1のEmailItemに近い
H_4:P_7のE_4 P_1のEmailItemに近い
H_5:P_7のE_5 P_4のBlurb(「KD」)に近い
【０１６７】
前述したパターンとそのパターンを許可するDRS要素との間のマッピングが指定された後(

すなわち、図8のブロック422で示されるとおり、pat2drselemsデータ構造が更新された後)、このマッピングは、次のとおりとなる。

P_1:EmailItem[3] email(x)にマップされる
P_2:E_2[3] HasSender [5] E_1[1] from(x,y)にマップされる
P_3:DeleteAct [4] HasTarget [4] E_3[3] delete(e,y)にマップされる
P_7:E_4 [3] HasSentTime [5] E_5[1] from(x,y)にマップされる

【０１６８】

このマッピングは、実際に、パターンの中の[]という略記を形式化する。例えば、email(x)が、元のUDRS構造のボックス3の中で出現したことがわかる。したがって、パターンP_1に対応するUDRSは、そのボックスに再構成するべきであることが分かる。

【０１６９】

この時点で、IFG212の中のスキーマエンジン222の動作が完了する。この時点で結果を部分的に解釈されたSemDRSにパッケージ化することを選択する場合、十分な情報が存在していることに留意されたい。例えば、「from」が、HasSenderとして解釈されるものと、この要素がHasSentTimeとして解釈されるものの2つのSemDRSがもたらされる。

【０１７０】

意味層の中で生成されたすべてのSemDRSは、使用される可能性が全くない解釈が生成されないことを確実にする様々な適格性制約を受ける。そのような制約は、例えば、タイプの不整合を除外すること、および使用される特定のスキーマによく適合しない解釈を退けることを含む。そのような適格性条件により、解釈プロセスのあらゆるステージにおいてありそうもない解釈を退けることが可能になる。

【０１７１】

図10は、本発明の一実施形態によるBlurb処理のより詳細なブロック図である。Blurb処理は、UDRSの中でまだ解釈されていないものを取り上げ、それをBlurbとして解釈して、その情報に関して解釈フラグメントを(また、したがって、パターンを)生成することができるようにする。さらに、SemDRSフラグメントを実際に生じさせたある資料に関する代替の解釈も提供される。例えば、発話が、サーチエンジンに適したストリングとしての解釈を有する可能性があると信じる理由がある場合、その発話全体に関するBlurb解釈を生成するのを選択することができる。

【０１７２】

この例では、まだ解釈されていないボックス要素は、入力ストリングの中のトークン6に対応するKD(y)だけである。この情報に対してBlurb処理を行うため、Blurb UDRS処理エンジン224がまず、入力UDRSの中で未解釈の資料を特定する。これが、図10のブロック490で示されている。次に、特定された資料が、関連するコストを有するBlurbパターンとして解釈される。これが、ブロック492で示されている。コストは、任意の所望のコストであることが可能である。例えば、Blurb処理を使用して生成されたパターンのコストは、2.0に1語当り1.75を足したものにすることができる。もちろん、任意の他の適切なコストの数値も使用することができる。

【０１７３】

パターンが生成された後、ブロック494で示されるとおり、pat2tマッピングデータ構造が更新される。次に、ブロック496で示されるとおり、ルースニングが適用され、選好が獲得される。ブロック498で示されるとおり、pat2drselemsマッピング構造も更新される。

【０１７４】

この更新が行われた後、PatternListおよびpat2tマッピング構造は、次のとおりとなる。

P_1:EmailItem [3] トークン3をカバーする
P_2:E_2 [3] HasSender [5] E_1 [1] トークン5をカバーする
P_3:DeleteAct [4] HasTarget [4] E_3 [3] トークン1をカバーする
P_4:Blurb(「KD」)[0] トークン6をカバーする
P_7:E_4 [3] HasSentTime [5] E_5 [1] トークン5をカバーする

【０１７５】

セットの選好は、以下のとおりとなる。
H_1:P_1のEmailItem　P_2のE_2に近い
H_2:P_2のE_1　P_4のBlurb(「KD」)に近い
H_3:P_3のE_3　P_1のEmailItemに近い
H_4:P_7のE_4　P_1のEmailItemに近い
H_5:P_7のE_5　P_4のBlurb(「KD」)に近い

【０１７６】
最後に、以下のペアが、pat2drselemsマッピングデータ構造に追加される。
P6:Blurb(「KD」)KD(y)にマップされる

【０１７７】
この時点で、初期探索状態を生成することができる。ただし、一例としての実施形態によれば、初期探索状態を生成するのに先立って堅牢性戦略が実施される。ただし、この戦略は、初期探索状態の初期セットが生成される前でも、後でも実施することができるが、完璧を期して以下に説明する。

【０１７８】
図11は、堅牢性戦略の実施をよりよく示す流れ図である。堅牢性戦略は、所与の文の誤った構文解析または定型の構文解析を補償するために、または適切な構文の解析が全く存在しない入力を扱うために実施される。例えば、アプリケーションが、情報検索サーチエンジンである場合、ユーザは、不完全な文を探索クエリとして使用することができる。その場合、入力ストリングの中の一部の件名は、UDRS構造の中で完全に無視することができる。したがって、堅牢性戦略処理エンジン226は、UDRSの中で構造上の制約が反映されることなしに、構文解析の中で得られたすべての言語証拠を列挙する。

【０１７９】
例えば、この堅牢性戦略を適用して、入力UDRSから初期探索状態を構成するプロセスでカバーされない言語証拠に従ってパターンを追加し、次に、IA214において解決を構成する代替の仕方としてそのパターンに関する追加の探索状態を生成することができる。堅牢性戦略の実施中に生成されたパターンは、例として、意味マッピング規則をUDRS構造に適用することによって生成されたパターンより低くランク付けされ、かつBlurb処理を介して生成されたパターンより低くランク付けされる。したがって、そのパターンから生成された探索状態は、例として、IA214が、入力UDRSおよびBlurbによって許可されたパターンに従って適合する解決をもたらすのに失敗した場合だけ、IA214によって考慮される。

【０１８０】
堅牢性戦略の実施をよりよく示すため、別の例(例2)を使用する。以下の入力ストリングおよびトークンリストを考慮されたい。

Email to Harry about Majel

1 2 3 4 5

【０１８１】
入力UDRSが、図11-1で与えられている。

【０１８２】
図6で示したのと同じスキーマを使用すると、この入力UDRSに関連する1組の意味マッピング規則が、以下のとおり推論される。
email(x)→EmailItem(x)
email(u,w)→InitEmailAct(u),HasTarget(u,w)
you(y)→(除去されるべき)
about(e,z)→HasSubject(e',z')

【０１８３】
意味マッピング規則が、IFG212によって入力UDRSに適用された後、またUDRS構造によって示唆されるとおりBlurbが生成された後、以下のパターンが現れる。
P_1:EmailItem [1]　トークン1をカバーする

P_2:E_1 [1] HasSubject [1] E_2 [1]　トークン4をカバーする
P_3:Blurb(「Majel」)[1]　トークン5をカバーする
P_4:無視可能　トークン2をカバーする

【０１８４】

また、以下の選好も得られる。

H_1:P_2のE_2　P_3のBlurb(「Majel」)に近い

【０１８５】

動詞の意味での「email」に関する意味マッピング規則は、構文解析が誤っていたことに起因して、この入力UDRSには適用されないことがわかる。つまり、その語が、入力UDRSの中で、動詞ではなく名詞として構文解析されている。また、入力文における語「Harry」が、ボックス構造に基づいて動詞として認識され、入力UDRSの中で関係として表されているのも見て取ることができる。適用可能な意味マッピング規則は全く存在せず、さらに、UDRS構造は、この資料に関するBlurb解釈を示唆する可能性があるが、そのような解釈フラグメントは、意図される最終解釈には適切でない。以上の効果は、この資料が、所望の最終SemDRSには全く寄与しないことである。

【０１８６】

ただし、この堅牢性戦略は、比較的良好な解決を探し出すのに役立つ可能性がある。この戦略を実施するため、図11のブロック500で示されるとおり、まず、無視されたトークンが選択される。次に、エンジン226が、より単純なストリングベースの書換え規則を選択されたトークンに適用して、さらなるパターンおよび関連するコストを生成する。この書換え規則は、開発者によって作成されること、またはスキーマから推論されることが可能である。これが、ブロック502で示されている。結果のパターンは、例として、意味マッピング規則を入力UDRSに適用することによって生成されたパターンより高い初期コストに関連付けられる。

【０１８７】

このパターンの中で生成されたエンティティおよび関係には、そのエンティティおよび関係が、入力UDRSの中のボックスに属さないことを示すラベルが付けられる。これが、ブロック504で示されている。この例では、以下のパターンが追加される。

P_5:InitEmailAct [0]　トークン1をカバーする
P_6:E_3 [0] HasEmailAddr [0] E_4 [0]　トークン1をカバーする
P_7:E_5 [0] HasRecipient [0] E_6 [0]　トークン2をカバーする
P_8:E_7 [0] HasTarget [0] E_8 [0]　トークン2をカバーする

【０１８８】

次に、エンジン222が、さらなるトークンを処理する必要があるかどうかを判定する。これが、ブロック506で示されている。処理する必要がある場合、処理は、ブロック500に戻る。しかし、処理する必要がない場合は、エンジン226は、無視された名前付きエンティティが存在するかどうかを判定する。これが、ブロック508で示されている。入力UDRSの中で名前付きエンティティが無視される可能性がある2つのケースが、実質的に存在する。第1に、名前付きエンティティが構文解析に整合するが、別の名前付きエンティティ(同じ語のスパンの)が構文解析によって選択された場合、第1の名前付きエンティティは、無視される。第2に、名前付きエンティティが構文解析と全く整合しない場合、そのエンティティは、無視される。

【０１８９】

以上2つのケースで、名前付きエンティティが無視された場合、比較的低いランクを有する、無視された名前付きエンティティを伴うパターンが生成される。これが、ブロック510で示されている。この例は、そのような名前付きエンティティは、全く有さない。

【０１９０】

次に、あるBlurbが、入力UDRS構造に違反する可能性がある(入力UDRS構造の中の無視された語を適切にカバーするBlurbを含めて)。構文解析ツリーまたは論理形式、およびUDRS構造によって生成されたBlurbより低いランクを提供するヒューリスティックに基づき、そ

のようなBlurbに関するBlurbパターンが生成される。これが、ブロック512で示されている。例2におけるそのようなBlurbは、以下のとおりである。
P_9:Blurb(「harry」)[0]　　トークン3をカバーする
P_{10}:Blurb(「to harry」)[0]　　トークン2、3をカバーする
【０１９１】
構文解析ツリーまたは論理形式に基づき、さらなる選好も生成することができる。この選好は、元の入力文における、このパターンによってカバーされる語の互いに対する位置に基づく。この選好の生成が、ブロック514で示されており、この例に関するそのような選好は、以下のとおりである。
H_2:P_5のInitEmailAct　P_7のE_5またはE_6に近い
H_3:P_5のInitEmailAct　P_8のE_7またはE_8に近い
H_4:P_7のE_5またはE_6　P_9のBlurb(「harry」)に近い
H_5:P_8のE_7またはE_8　P_9のBlurb(「harry」)に近い
【０１９２】
次に、このパターンを使用して初期探索状態を生成することができる。
【０１９３】
図12は、SIC210の中のISS生成構成要素218による初期探索状態の生成を示す流れ図である。初期探索状態の生成は、前述した例1に関連して説明する。初期探索状態は、堅牢性戦略を実施するのに先立って生成することができ、また堅牢性戦略を必要な場合だけ実施してもよいことに留意されたい。あるいは、初期探索状態は、堅牢性戦略を実施した後に生成することもできる。いずれにしても、初期探索状態は、3種類の情報を含む。
【０１９４】
1.入力ストリングをカバーする1組のパターン
2.パターンから入力ストリングの中のトークンへのマップ(pat2tデータ構造)、および
3.パターンに関連する1組の選好
【０１９５】
堅牢性戦略を実施する前でさえ、この情報は、利用可能であり、何らかのパターンから、意味マッピング規則およびBlurbのいずれによっても解釈されなかった語(前述した例1における「not」や「all」などの、意味内容がボックス構造の中で反映される語などの)へのマッピングが、例外である可能性がある。したがって、堅牢性戦略に先立って探索状態が生成された場合、その探索状態は、ブロック520で示されるとおり、まず、そのトークンを特定し、ブロック522で示されるとおり、無視することができるパターンをPatternListデータ構造に追加して特定されたトークンをカバーすることによって完成される。例1では、これは、以下の2つのパターンをPatternListに追加することになる。
P_5:無視可能　トークン2をカバーする
P_6:無視可能　トークン4をカバーする
【０１９６】
次に、可能な初期探索状態が、パターンリストおよびマッピングから形成されて、入力UDRSのすべての部分が何らかのパターンに対応し、複数のパターンによってカバーされる部分が全く存在しなくなる。これが、ブロック524で示されている。
【０１９７】
初期探索状態が、堅牢性戦略の実施に先立って生成される場合、すべての可能な初期探索状態は、PatternListデータ構造の中のパターン、およびマッピングに基づいて形成される。ただし、堅牢性戦略が既に実施されており、初期探索状態が、そのパターンに関しても生成される場合は、生成される可能性がある初期探索状態の数は、指数的になる。したがって、堅牢性戦略において生成されるパターンに基づく初期探索状態の生成は、時間などの何らかの他の基準によって制約される可能性がある。
【０１９８】
ISS生成制約が満たされると(例えば、入力UDRSおよびBlurbによって生成されたパターンに基づいてすべての可能な探索状態が生成され、また堅牢性戦略からのパターンに基づい

て生成された初期探索状態に関して最大時間に達すると)、初期探索状態の生成は、完了であり、その探索状態が、IA214に提供される。これが、ブロック526および528で示されている。この場合も、堅牢性戦略によって生成されたパターンに基づく初期探索状態は、入力UDRSの外部にある証拠によって許可されている。したがって、この初期探索状態は、一般に、その他の初期探索状態よりも低くランク付けされる。

【０１９９】

図13は、IA214の中の解決パターン生成構成要素228の全体的動作をよりよく示す流れ図である。解決パターン生成構成要素228はまず、SIC210からスキーマおよび初期探索状態を受け取る。これが、図13のブロック600および602で示されている。この場合も、初期探索状態は、コストに関連するSemDRSフラグメント、加えてSemDRSフラグメントの特定の組合せのコストを低減することができる選好、加えてパターンから入力ストリングの中のトークンへのマップ(すなわち、pat2t)である。

【０２００】

次に、解決パターン生成器228は、初期探索状態のヒューリスティックな探索を行って解決パターンを得る。これは、生成器228が、提供される選好に従い、最低のコストでパターンを解決パターンにアセンブルしようと試みる最適化問題である。この探索が、図13のブロック604で示されている。

【０２０１】

探索の結果として、コストの低い順にランク付けされた解決パターンのシーケンスが生成される。コストは、フラグメントの選択、スキーマの中におけるそのフラグメントの互いとの近接性、ならびに提供される選好の満足度との関係にあるものである。解決パターンの生成が、ブロック606で示されている。

【０２０２】

次に、IA214が、SIC210の中のDRS構造復元生成器220を呼び出す。生成器220は、ボックス構造を解決パターンに復元して完全に解釈されたSemDRSを生成する。これが、ブロック608で示されている。

【０２０３】

解決パターンに関するボックス構造は、解決パターンの中で使用されるパターンの入力UDRSボックス構造へのマッピングにアクセスすることによって復元される。これらのパターンは、可能な場合、入力UDRSの中で見られるのと同じボックス構造の中に入れられる。

【０２０４】

図14は、ヒューリスティック-解決パターン生成器228の実装をさらに示す流れ図である。一例としての実施形態では、ヒューリスティックは、周知のA^*アルゴリズムの変形である。概要を述べると、生成器228はまず、ブロック610で示されるとおり、1組の初期探索状態を選択する。セットの中の各初期探索状態に関して、最小コストGが、初期探索状態の中のパターンに割り当てられたコストに基づき、IFG212によって計算される。これが、ブロック612で示されている。次に、セットの中の各初期探索状態に関して、探索状態を使用する解決の生成に関する追加のコストHが、推定される。これが、ブロック614で示されている。次に、探索状態が、合計コストG+Hが低い値の順でソートされる。これが、ブロック616で示されている。

【０２０５】

探索状態をこのように初期にソートした後、現ステップでまだ処理されていない最低の合計コストを有する探索状態が、選択される。これが、ブロック618で示されている。選択された探索状態から、その選択された探索状態の中のフラグメントを異なった仕方で組み合わせることにより、すべての可能な新しい探索状態が生成される。これが、ブロック620で示されており、これについては、図15に関連してさらに詳細に述べる。ブロック620で生成された新しい探索状態のそれぞれに関して、新しい最小コストG'および新たに推定されるコストH'が計算される。これが、ブロック622で示されている。フラグメントの組合せに基づき、ブロック624で示されるとおり、適用可能な選好のそれぞれに関連する割引きが適用される。

【０２０６】
選好は、ある解釈フラグメントが、ある構成(選好によって指定される)で生じるあらゆる可能な解決に割引きを提供する。選好の存在は、言語証拠の存在を反映しており、その言語証拠に適合している解決が、その他の解決よりも高くランク付けされるのを確実にする。割引きは、解決パターンが生成された後に解決パターン全体に対してではなく、解決パターンの生成の最中に適用されることに留意されたい。使用することのできる選好の例には、以下が含まれる。
【０２０７】
1.強いソース/宛先選好。この例示の選好では、言語証拠およびスキーマがともに、フラグメントが、別のフラグメントのソース側または宛先側で現れるべきことを示し、実際に現れた場合、この選好が適用されており、割引き(例えば、0.75の)が、その解決パターンに施される。
【０２０８】
2.弱いソース/宛先選好。言語証拠だけが、1つのフラグメントが他のフラグメントのソース側または宛先側で現れるべきことを示し、実際に現れた場合、この選好が適用され、割引き(例えば、0.5の)を与える。
【０２０９】
3.近接性選好。スキーマまたは言語証拠が、2つのフラグメントが互いに可能な限り近くなるべきことを示し、実際にそうなる。この選好が適用され、割引き(例えば、0.25の)を与える。
【０２１０】
選好が適用された後、解決パターン生成器228が、完成した解決パターンが存在するかどうかを判定する。探索状態の中のすべてのフラグメントが、単一のフラグメントに結合され、または統合されている場合、その探索状態が、その関連するコストG'とともに、解決パターンとして出力される。これが、ブロック626で示されている。
【０２１１】
次に、処理は、ブロック628に進み、生成器228が、現ステップで処理されるべきさらなる探索状態が存在するかどうかを判定する。存在する場合、処理は、ブロック618に戻り、現ステップでまだ処理されていない2番目に低い合計コストを有する探索状態が選択される。
【０２１２】
ブロック628で、現ステップで処理されるべきさらなる探索状態が存在しない場合は、生成器628が、処理される必要のある探索状態がまだ残っているかどうか(すなわち、1つのフラグメントに完全に結合されて解決パターンを形成していない探索状態がまだ、存在するかどうか)を判定する。これが、ブロック630で示されている。そのような探索状態が存在する場合、処理は、ブロック616に戻り、残っている探索状態が、合計コストG'+H'の低い値の順にソートされ、最低コストを先にして再び処理される。探索状態のすべてが、完全に処理されると、ブロック632で示されるとおり、解決パターンの生成が完了する。
【０２１３】
また、長い文および曖昧な意味マッピングの場合、初期探索状態の数が、望ましくないほど大きくなる可能性があることにも留意されたい。これは、その探索状態をセットアップするのに法外に長い処理時間をもたらす可能性があり、その探索状態の比較的少数だけしか、実際に探ることができない。したがって、探索状態は、アプリケーションスキーマの中で比較的緊密な近接性を示す入力UDRS要素の解釈に対応するものだけを生成することにより、さらに制限することができる。この初期探索状態から達せられる解決は、必ずしも、探し出すことのできる最も安価な解決ではない可能性があるが、最も安価である可能性が高い。さらに、この制限は、例として、入力文が望ましくないほど長い場合だけに適用することができる。したがって、以上の制約は、例として、妥当な長さの文に関しては、実施されることさえない。
【０２１４】

図15は、IA214の解決パターン生成器228においてどのようにパターンが組み合わせられるかを一般的に示す流れ図である。初期探索状態のパターンまたはフラグメントを組み合わせるため、解決パターン生成器228はまず、初期探索状態のための2つのパターンおよび関連する選好を、スキーマとともに受け取る。これが、図15のブロック650で示されている。次に、生成器228は、そのパターンの中のエンティティまたは関係を統合することができるかどうかを判定する。これが、ブロック652で示されている。例えば、パターンは、以下のとおりであると想定する。

P1=エンティティ1、関係1、エンティティ2;および

P2=エンティティ3、関係2、エンティティ4

【０２１５】

エンティティ2および3は、エンティティ2および3が同一である場合、またはどちらかが、もう一方の推移的な（transitive）サブタイプである場合、統合可能である。

【０２１６】

そうである場合、以下のとおり、結合パターンがもたらされる。

エンティティ1、関係1、統合されたエンティティ2/3、関係2、エンティティ4

【０２１７】

関係は、同じタイプを有する場合、ソースを統合することができる場合、および宛先を統合することができる場合、統合可能である。

【０２１８】

いずれにしても、ブロック652で示すとおり、2つの選択されたパターンの中のエンティティまたは関係を統合することができる場合、そのパターンは、統合され、所望する場合、コストを適用することができる。例示の実施形態では、パターンまたは関係を統合するためのコストは0である。これが、ブロック654で示されている。

【０２１９】

ブロック652で、受け取られたパターンの中のエンティティまたは関係を統合することができない場合は、解決パターン生成器228が、そのパターンのそれぞれからのエンティティを結合するスキーマを介する最も安価な（least expensive）パスを探し出す。これが、ブロック656で示されている。スキーマの中のパスのそれぞれのコストは、事前計算することができる。もちろん、このコストは、実行時に計算することもできる。2つのパスに関するコストが同じである場合は、例として、短い方のパスが、最も安価なパスとして選択される。

【０２２０】

ブロック656で最も安価なパスが特定された後、生成器228は、スキーマに従って2つのパターンを結合するのに必要な関係およびエンティティを追加し、この新しい結合されたパターンに関連するコストを適用する。これが、ブロック658で示されている。

【０２２１】

図14に関連して示すとおり、次に、新たに結合されたパターンは、スコアを付けられ、スコア順にソートされ、探索状態が単一のフラグメントで表されるようになるまで、結合および統合が続けられる。探索状態は、単一のフラグメントで表されるようになった場合、解決パターンである。スコア付けおよびソートが常に行われるため、IA214は、最高のランクが付けられた(最も安価な)解決パターンを先にして提供するように確実にされている。

【０２２２】

したがって、本発明は、DRSを正規化して、意味解析システムにおいて入力として使用することができる正規化されたデータ構造にするのを理解することができる。

【０２２３】

本発明は、特定の実施形態に関連して説明してきたが、当分野の技術者には、本発明の趣旨および範囲を逸脱することなく、形態および詳細において変更を加えるのが可能であることが認められよう。

【０２２４】

【表５】

APPENDIX A

```
Step 1.   [Remember what boxes are in use.]
          Let old_number_of_boxes =
          drs.number_of_boxes().
          Let used_box_ids be an integer vector of
          size old_number_of_boxes and let used_boxes
          be a box vector of the same size.
          i ← 0. j ← 0.
          while i < old_number_of_boxes
             if drs.box(i).in_use()
                used_box_ids[i] ← j.
                used_boxes[i] ← drs.box(j).
                j ← j + 1.
             i ← i + 1.
          Let new_number_of_boxes = j.
Step 2.   [Sort the box indices preliminarily.]
          Sort used_box_ids[0 .. new_number_of_boxes
          - 1] in place to satisfy <a (cf "<a over box
          IDs" above).
Step 3.   [Move the boxes and remember what box
          elements are in use.]
          Let forward_boxes be an integer vector of
          size old_number_of_boxes.
          Let old_number_of_elements =
          drs.number_of_elements().
          Let elements_in_use be a Boolean vector of
          size old_number_of_elements, all
          initialized to false.
          i ← 0. k ← 0.
          while i < new_number_of_boxes
             forward_boxes[used_box_ids[i]] ← i.
             drs.box(i) ← used_boxes[used_box_ids[i]].
             sort the element IDs of drs.box(i) in
             place to satisfy <a (cf "<a over box
             element IDs" above).
             let z = drs.box(i).number_of_elements().
             j = 0.
             while j < z
                elements_in_use[drs.box(i).element_id(
                j)] ← true.
                j ← j + 1. k ← k + 1.
             i ← i + 1.
```

【０２２５】

【表６】

```
        Let new_number_of_elements be the current
        value of k.
Step 4. [Move box elements.]
        Let forward_elements be an integer element
        vector of size old_number_of_elements.
        i ← 0. k ← 0.
        while i < old_number_of_elements
          if elements_in_use[i]
            if k < i
              drs.element(k) ← drs.element(i).
            forward_elements[i] ← k.
            k ← k + 1.
          i ← i + 1.
Step 5. Any storage occupied by boxes
        old_number_of_boxes to new_number_of_boxes
        - 1 of drs, and by box elements
        old_number_of_elements to
        new_number_of_elements - 1 of drs can now
        be released.
Step 6. [Update boxes to have the correct box
        elements again.]
        i ← 0.
        while i < new_number_of_boxes
          let z = drs.box(i).number_of_elements().
          j ← 0.
          while j < z
            drs.box(i).element_id(j) ←
              forward_elements[drs.box(i).element_
              id(j)]
            j ← j + 1.
          i ← i + 1.
Step 7. [Update elements to refer to the correct
        boxes again.]
        i ← 0.
        while i < new_number_of_elements
          if drs.element(i).kind() =
          elem_constraint or elem_logical
            let z =
            drs.element(i).number_of_args().
            j ← 0.
            while j < z
              if drs.element(i).arg(j).kind() =
              arg_label
                drs.element(i).arg(j).label() =
```

【0226】

【表7】

```
                    forward_boxes[drs.element(i).ar
                    g(j).label()].
                j ← j + 1.
            i ← i + 1.
Step 8.   [Collect information about marker
          occurrences.]
          Let marker_occurrences_map be a map from
          markers to lists of marker occurrence
          triples (see above for a description),
          initially empty.
          i ← 0.
          while i < new_number_of_boxes
            let z = drs.box(i).number_of_elements().
            j ← 0.
            while j < z
              if drs.box(i).element(j).kind() =
              elem_marker
                let marker be
                drs.box(i).element(j).marker().
                if marker_occurrences_map doesn't
                have an association for marker
                  add an association from marker to
                  an empty list.
                Add the triple ⟨i,0,0⟩ to the end of
                the list associated with marker in
                marker_occurrences_map.
              if drs.box(i).element(j).kind() =
              elem_constraint or elem_logical
                let w be
                drs.box(i).element(j).number_of_args
                ().
                k ← 0.
                while k < w
                  if
                  drs.box(i).element(j).arg(k).kind
                  () = arg_marker
                    let marker be
                    drs.box(i).element(j).
                    arg(k).marker().
                    if marker_occurrences_map
                    doesn't have an association for
                    marker
                      add an association from
                      marker to an empty list.
```

【0227】

【表8】

Add the triple $\langle i,j,k \rangle$ to the
end of the list associated with
marker in
marker_occurrences_map.
$k \leftarrow k + 1$.
$j \leftarrow j + 1$.
$i \leftarrow i + 1$.
(Note that the order in which we add marker occurrence triples to the lists in the map ensures that each list is ordered lexicographically on the triples.)

Step 9. [Create an inverse map of marker occurrences.]
Let *marker_occurrences_inverse_map* be a multimap from lists of marker occurrence triples to markers, initially empty. All lists used as keys in the association are ordered in the same way (see the note at the end of the previous step).
for each association *marker* → *list_of_triples* in *marker_occurrences_map*
(the order of traversal is irrelevant)
add the association *list_of_triples* → *marker* to *marker_occurrences_inverse_map*.

Step 10. [Assign new marker values.]
Let *forward_markers* be a map from integers to integers, initially empty.
$k \leftarrow 0$.
for each association *list_of_triples* → *marker* in *marker_occurrences_inverse_map*
(if L_1 precedes L_2 lexicographically, then any association for L_1 should be visited before any association for L_2)
if *forward_markers* doesn't contain an association for *marker*
add the association *marker* → k to *forward_markers*.
$k \leftarrow k + 1$.
drs.number_of_markers() $\leftarrow k$.
(*forward_markers* now contains an association for every marker occurring in *drs*.)

Step 11. [Update elements with new markers.]
$i \leftarrow 0$.

【0228】
【表9】

```
while i < new_number_of_elements
   if drs.element(i).kind() = elem_marker
      drs.element(i).marker() =
         forward_markers[drs.element(i).marke
         r()].
   if drs.element(i).kind() =
   elem_constraint or elem_logical
      let z =
      drs.element(i).number_of_args().
      j ← 0.
      while j < z
         if drs.element(i).arg(j).kind() =
         arg_marker
            drs.element(i).arg(j).marker() =
               forward_markers[drs.element(i).
               arg(j).marker()].
            j ← j + 1.
      i ← i + 1.
```

Step 12. [Final sort. This mimics Steps 1 – 3 and Step 7.]

```
We will reuse vectors used_box_ids,
used_boxes, and forward_boxes. (If memory
is scarce and new_number_of_boxes is
significantly larger than
old_number_of_boxes, then we could instead
allocate new vectors of length
new_number_of_boxes.)
i ← 0.
while i < new_number_of_boxes
   used_box_ids[i] ← i.
   used_boxes[i] ← drs.box(i).
   i ← i + 1.
Sort used_box_ids[0 .. new_number_of_boxes
- 1] in place to satisfy <b (cf "<b over box
IDs" above).
i ← 0.
while i < new_number_of_boxes
   forward_boxes[used_box_ids[i]] ← i.
   drs.box(i) ← used_boxes[used_box_ids[i]].
   sort the element IDs of drs.box(i) in
   place to satisfy <b (cf "<b over box
   element IDs" above).
   i ← i + 1.
i ← 0.
```

【0229】

【表10】

```
while i < new_number_of_elements
   if drs.element(i).kind() =
   elem_constraint or elem_logical
      let z =
      drs.element(i).number_of_args().
      j ← 0.
      while j < z
         if drs.element(i).arg(j).kind() =
         arg_label
            drs.element(i).arg(j).label() =
               forward_boxes[drs.element(i).ar
               g(j).label()].
         j ← j + 1.
   i ← i + 1.
```

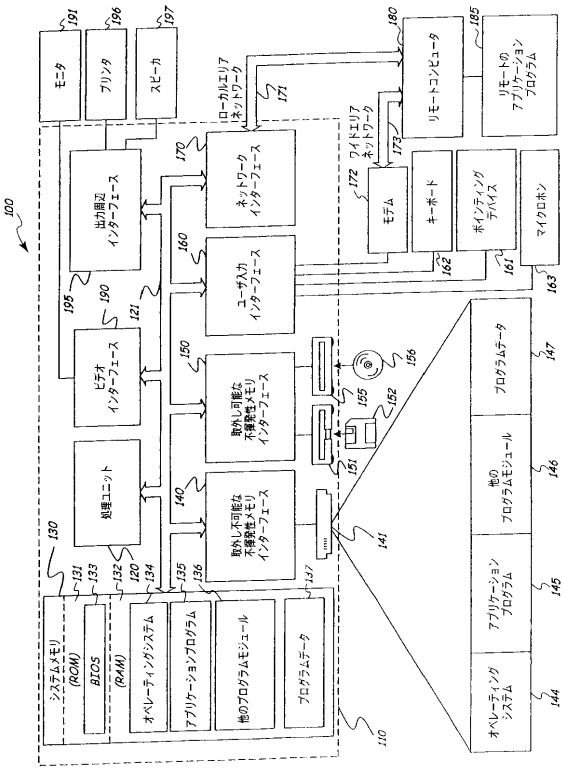
【０２３０】
注
・$<_a$(ステップ3)または$<_b$(ステップ12)に従ってボックスの要素がソートされると、この事実をボックス実施形態の追加のフィールドの中に記録して、同じボックスに複数回、遭遇した場合、ソートアルゴリズムを再び適用しなくてもよいようにすることができる。
【０２３１】
・このアルゴリズムは、入力DRSを変更して正規化されるようにする。このアルゴリズムが、入力DRSの正規化されたコピーを代わりに出力するように変更されたとすると、used_boxesベクトルをなくすことができる。used_boxesベクトルの目的は、ボックスがコピーされる前に上書きされるのを回避するため、ボックスがコピーされる間、そのボックスを記憶することだけである。ボックス要素がコピーされる順序により、上書きされるあらゆるボックス要素が、既にコピーさているか、または廃棄されるべきものであることが保証される。
【図面の簡単な説明】
【図１】本発明を使用することができるコンピュータの一例としての実施形態を示すブロック図である。
【図２】本発明を使用することができるテキスト解析システムを示すブロック図である。
【図３】本発明の一実施形態による意味解析システムをより詳細に示すブロック図である。
【図４】本発明の一実施形態による構文解析(UDRSも含め)の処理を一般的に示す流れ図である。
【図５Ａ】　UDRSボックス構造を示す図である。
【図５Ｂ】　UDRSの正規化を示す流れ図である。
【図６】一例としてのアプリケーションスキーマを示す図である。
【図７】どのようにシステムを初期設定するかをより詳細に示す流れ図である。
【図８】意味マッピング規則のUDRSのボックス構造への適用を示す流れ図である。
【図９】本発明の一実施形態によるルースニングプロセスを示す流れ図である。
【図１０】本発明の一実施形態によるBlurb処理を示す流れ図である。
【図１１】本発明の一実施形態による堅牢性戦略の実施を示す流れ図である。
【図１１－１】第2の例としてのUDRSブロック構造を示す図である。
【図１２】本発明の一実施形態による初期探索状態の生成を示す流れ図である。
【図１３】本発明の一実施形態による初期探索状態を探索して解決パターンおよびSemDRSを得るプロセスを示す高レベルの流れ図である。
【図１４】解決パターンを得るための初期探索状態の探索をより詳細に示す流れ図である。

【図１５】どのようにパターンを結合して、または統合して解決パターンを得るかを示す高レベルの流れ図である。

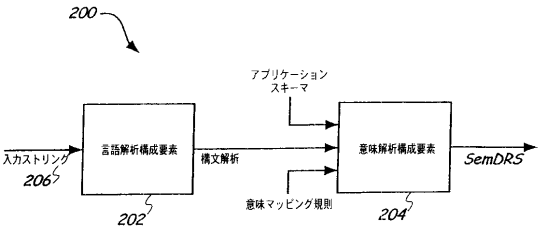
【符号の説明】

200　テキスト処理システム
202　言語解析構成要素
204　意味解析構成要素
206　入力ストリング
210　意味解釈コントローラ
212　解釈フラグメント生成器
214　解釈アセンブラ
215　データ記憶装置
216　サービスインボーカ
218　初期探索状態(ISS)生成構成要素
220　DRS構造復元生成器
222　スキーマエンジン
224　Blurb UDRS処理エンジン
226　堅牢性戦略処理エンジン

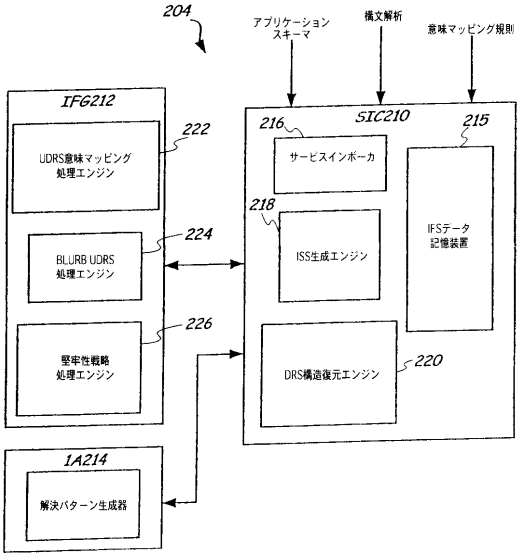
【図１】



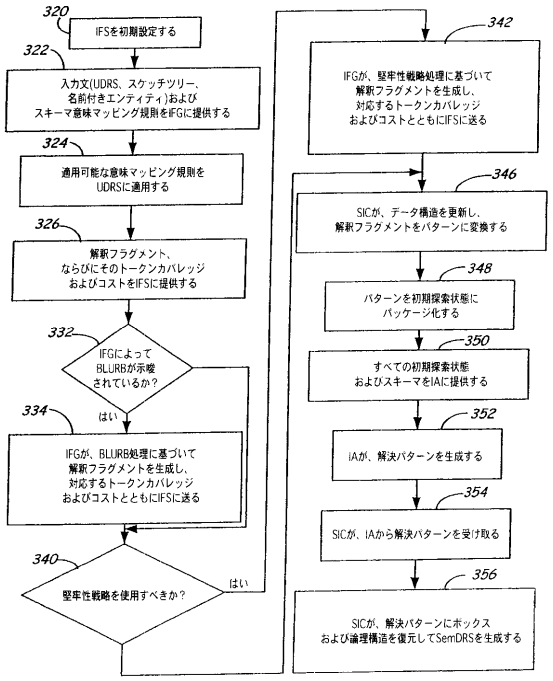
【図２】



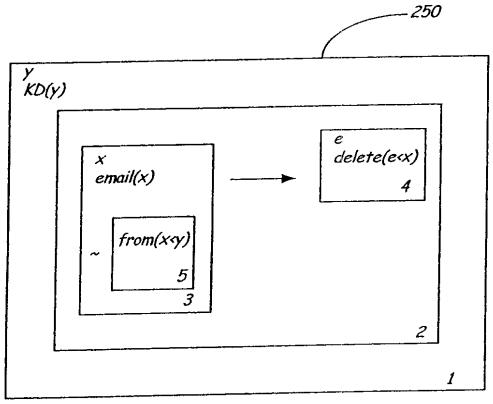
【図３】



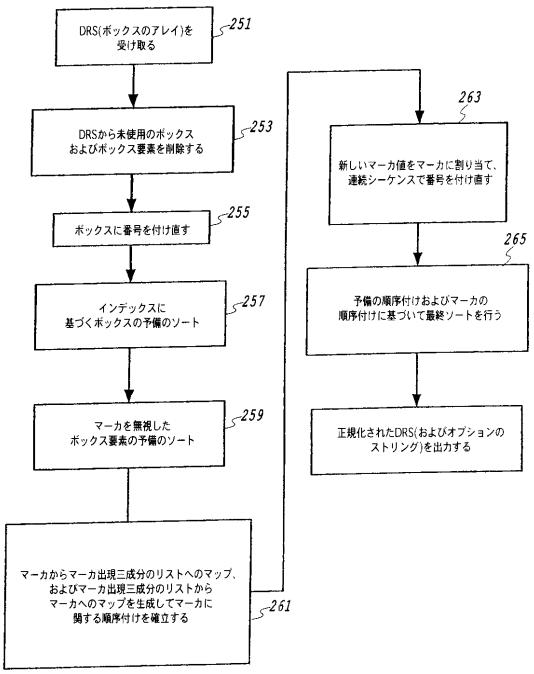
【図４】


320
IFSを初期設定する
322
入力文(UDRS、スケッチツリー、名前付きエンティティ)およびスキーマ意味マッピング規則をIFGに提供する
324
適用可能な意味マッピング規則をUDRSに適用する
326
解釈フラグメント、ならびにそのトークンカバレッジおよびコストをIFSに提供する
332
IFGによってBLURBが示唆されているか?
はい
334
IFGが、BLURB処理に基づいて解釈フラグメントを生成し、対応するトークンカバレッジおよびコストとともにIFSに送る
340
堅牢性戦略を使用すべきか?
はい
342
IFGが、堅牢性戦略処理に基づいて解釈フラグメントを生成し、対応するトークンカバレッジおよびコストとともにIFSに送る
346
SICが、データ構造を更新し、解釈フラグメントをパターンに変換する
348
パターンを初期探索状態にパッケージ化する
350
すべての初期探索状態およびスキーマをIAに提供する
352
IAが、解決パターンを生成する
354
SICが、IAから解決パターンを受け取る
356
SICが、解決パターンにボックスおよび論理構造を復元してSemDRSを生成する

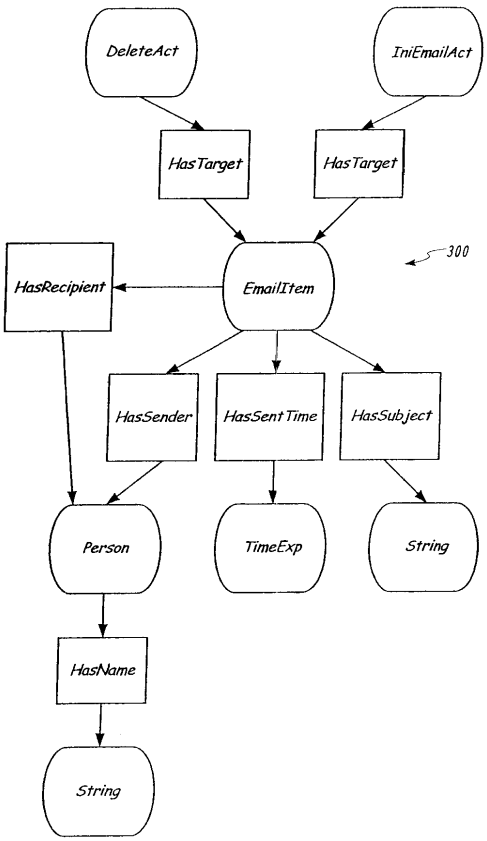
【図５Ａ】


250
y
KD(y)
x
email(x)
e
delete(e<x)
4
~
from(x<y)
5
3
2
1

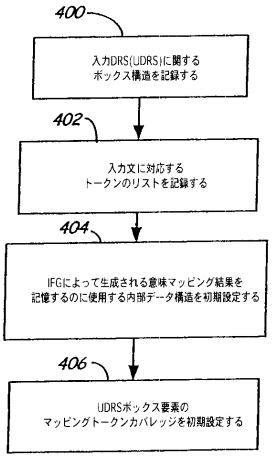
【図５Ｂ】


DRS(ボックスのアレイ)を受け取る
251
DRSから未使用のボックスおよびボックス要素を削除する
253
ボックスに番号を付け直す
255
インデックスに基づくボックスの予備のソート
257
マーカを無視したボックス要素の予備のソート
259
マーカからマーカ出現三成分のリストへのマップ、およびマーカ出現三成分のリストからマーカへのマップを生成してマーカに関する順序付けを確立する
261
新しいマーカ値をマーカに割り当て、連続シーケンスで番号を付け直す
263
予備の順序付けおよびマーカの順序付けに基づいて最終ソートを行う
265
正規化されたDRS(およびオプションのストリング)を出力する

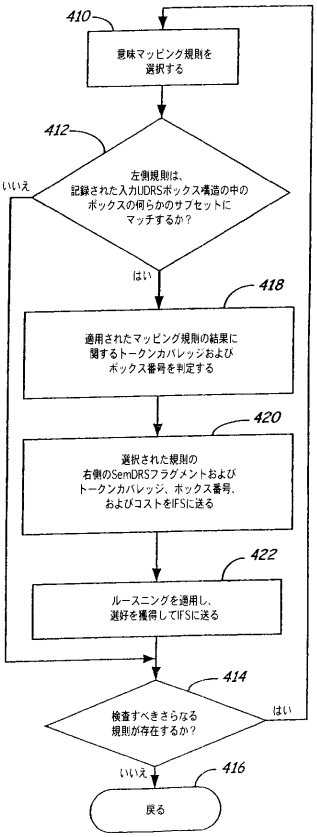
【図６】


DeleteAct
IniEmailAct
HasTarget
HasTarget
HasRecipient
EmailItem
300
HasSender
HasSentTime
HasSubject
Person
TimeExp
String
HasName
String

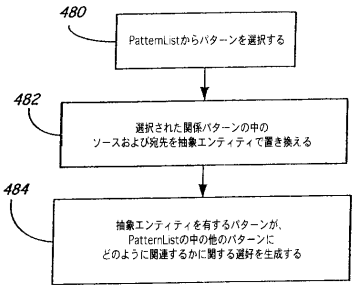
【図７】



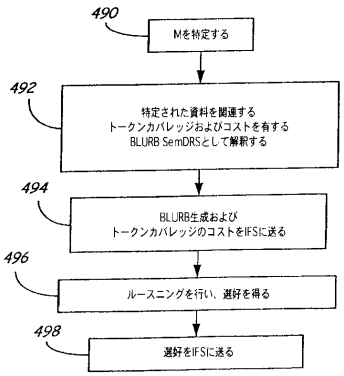
【図８】



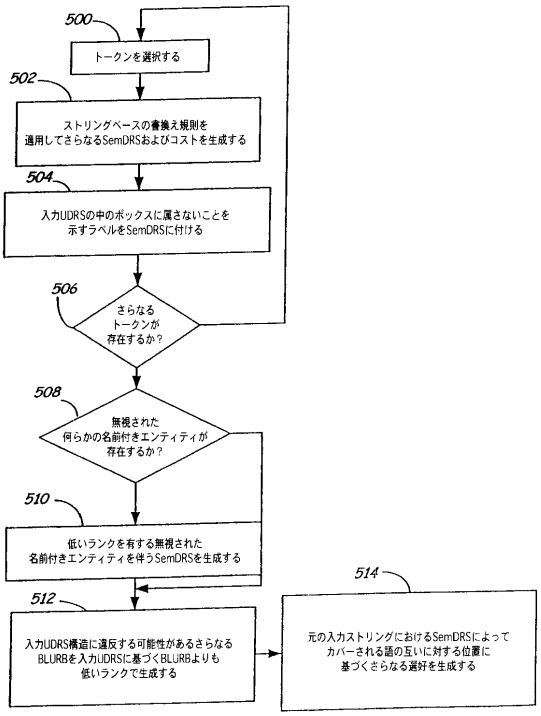
【図９】



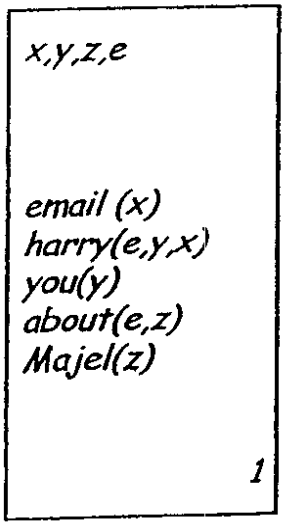
【図１０】



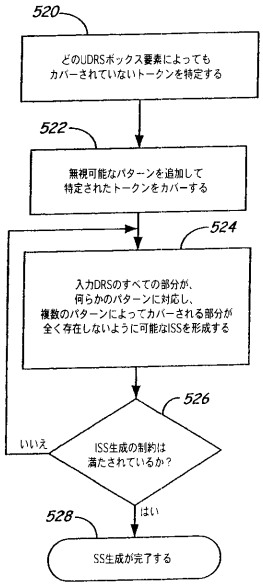
【図１１】



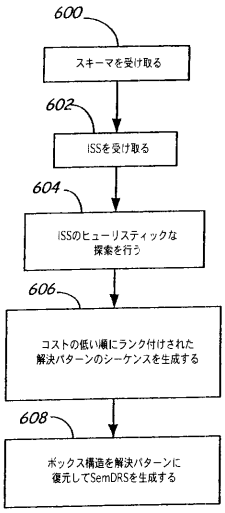
【図１１－１】



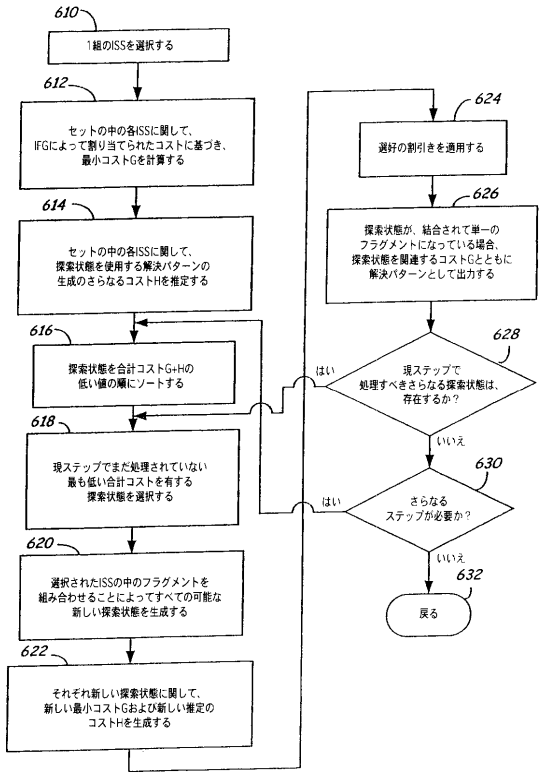
【図１２】



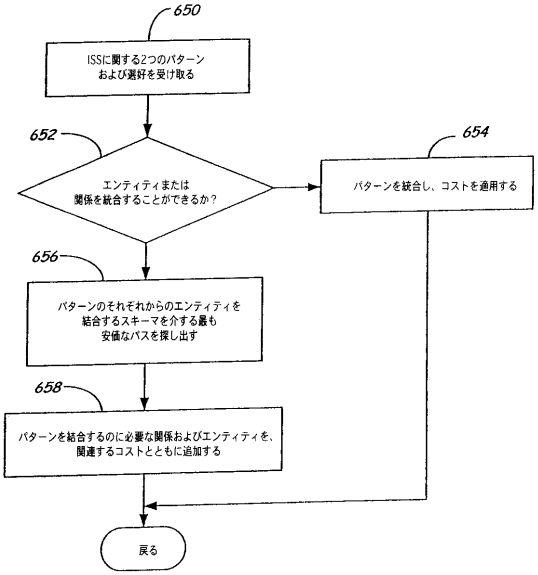
【図１３】



【図１４】



【図１５】


650
ISSに関する2つのパターン
および選好を受け取る
652
エンティティまたは
関係を統合することができるか？
654
パターンを統合し、コストを適用する
656
パターンのそれぞれからのエンティティを
結合するスキーマを介する最も
安価なパスを探し出す
658
パターンを結合するのに必要な関係およびエンティティを、
関連するコストとともに追加する
戻る

フロントページの続き

(72)発明者 パール ヨーナス バークランド
アメリカ合衆国 ９８０３３－７２４０ ワシントン州 カークランド １０８ アベニュー ノースイースト ６０２０
(72)発明者 マイケル ブイ. カルカーニョ
アメリカ合衆国 ９８０３３ ワシントン州 カークランド レイク ワシントン ブールバード ５２１０ ナンバー２０２

合議体
審判長 小曳 満昭
審判官 久保 正典
審判官 長島 孝志

(56)参考文献 米国特許第５７９４０５０（ＵＳ，Ａ）
米国特許第５８７３０５６（ＵＳ，Ａ）
Ｓｃｈｉｅｈｌｅｎ，Ｍ．Ｄｉｓａｍｂｉｇｕａｔｉｏｎ ｏｆ Ｕｎｄｅｒｓｐｅｃｉｆｉｅｄ Ｄｉｓｃｏｕｒｓｅ Ｒｅｐｒｅｓｅｎｔａｔｉｏｎ Ｓｔｒｕｃｔｕｒｅｓ ｕｎｄｅｒ Ｓｎａｐｈｏｒｉｃ Ｃｏｎｓｔｒａｉｎｔｓ，Ｐｒｏｃｅｅｄｉｎｇｓ Ｏｆ Ｔｈｅ Ｓｅｃｏｎｄ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｗｏｒｋｓｈｏｐ Ｏｎ Ｃｏｍｐｕｔａｔｉｏｎａｌ Ｓｅｍａｎｔｉｃｓ，０８－０１－１９９７

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ１７/２７
Ｇ０６Ｆ１７/２８