

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4912678号

(P4912678)

(45) 発行日 平成24年4月11日(2012.4.11)

(24) 登録日 平成24年1月27日(2012.1.27)

(51) Int.Cl.

F I

G 0 6 F 9/45 (2006.01)

G 0 6 F 9/44 3 2 0 A

請求項の数 12 (全 19 頁)

(21) 出願番号	特願2005-370931 (P2005-370931)	(73) 特許権者	500046438
(22) 出願日	平成17年12月22日(2005.12.22)		マイクロソフト コーポレーション
(65) 公開番号	特開2006-209748 (P2006-209748A)		アメリカ合衆国 ワシントン州 9805
(43) 公開日	平成18年8月10日(2006.8.10)		2-6399 レッドモンド ワン マイ
審査請求日	平成20年12月8日(2008.12.8)		クロソフト ウェイ
(31) 優先権主張番号	11/044,100	(74) 代理人	100077481
(32) 優先日	平成17年1月27日(2005.1.27)		弁理士 谷 義一
(33) 優先権主張国	米国 (US)	(74) 代理人	100088915
			弁理士 阿部 和夫
		(72) 発明者	ヘンリクス ヨハネス マリア メイヤー
			アメリカ合衆国 98052 ワシントン
			州 レッドモンド ワン マイクロソフト
			ウェイ マイクロソフト コーポレーシ
			ョン内

最終頁に続く

(54) 【発明の名称】 実行時型推論による効率的なデータ・アクセス

(57) 【特許請求の範囲】

【請求項 1】

メモリに結合されたプロセッサを含むコンピューティング環境において、実行時型スキーマ推論を使用してXML入力データからクラスおよび前記クラスのインスタンスを動的に生成し、それにより実行時の動的データ型への効率的で便利な遅延束縛アクセスを提供するコンピュータで実行する方法であって、

XML文書からXML入力データの第1の入力ストリームを受け取るステップであって、前記XML入力データを受け取る際、前記XML入力データの型スキーマは未知であるステップと、

前記第1の入力ストリームの前記入力データの実際の値に基いて、前記入力データの型スキーマを推論するために、実行時に、前記XML入力データを処理するステップであって、前記型スキーマは、推論される前に未知である前記第1の入力ストリームの前記入力データの編成または構造を表すステップと、

前記型スキーマからソース・コードを生成し、前記ソース・コードをコンパイルして、前記第1の入力ストリームの前記XML入力データを表す1または複数のクラスを生成するステップと、

前記生成された1または複数のクラスをメモリに読み込むステップと、

前記生成された1または複数のクラスの各々のインスタンスを生成し、前記第1の入力ストリームの前記XML入力データを前記1または複数のインスタンスにデシリアライズし、それにより前記1または複数のインスタンスを通じて前記第1の入力ストリームの前

10

20

記XML入力データに対する遅延束縛アクセスを有効にするステップと、

前記実行時プロセスの間に前記型スキーマの生成のためのキャッシュ動作を実行するステップと、

XML文書からのXML入力データの第2の入力ストリームを受け取るステップと、

前記第2の入力ストリームの前記XML入力データを実行時に処理して、前記型スキーマに対する前記第2の入力ストリームの前記XML入力データの妥当性を確認するステップであって、前記型スキーマが前記第2の入力ストリームの前記XML入力データに関して有効であるか否かを判断し、

前記型スキーマが前記第2の入力ストリームの前記XML入力データに関して有効である場合、前記第2の入力ストリームの前記XML入力データに対して前記型スキーマを適用し、該適用は、前記第2の入力ストリームの前記XML入力データを前記生成された1または複数のクラスのインスタンスにデシリアライズして、それにより前記1または複数のインスタンスを通じて前記第2の入力ストリームの前記XML入力データに対する遅延束縛アクセスを提供し、

前記型スキーマが前記第2の入力ストリームの前記XML入力データに関して有効でない場合、

前記第2の入力ストリームの前記XML入力データの実際の実行時の値に基づいて新しい型スキーマを推論し、

前記新しい型スキーマからソース・コードを生成し、前記ソース・コードをアセンブリ・コードにコンパイルし、かつ前記アセンブリ・コードを処理してメモリに読み込まれる1または複数の新しいクラスを生成し、前記1または複数の新しいクラスは前記第2の入力ストリームの前記XML入力データを表し、

前記1または複数の新しいクラスの各々のインスタンスを生成し、かつ

前記第2の入力ストリームの前記XML入力データを前記1または複数の新しいクラスの前記1または複数のインスタンスにデシリアライズし、それにより前記1または複数の新しいクラスの前記1または複数のインスタンスを通じて前記第2の入力ストリームの前記XML入力データに対する遅延束縛アクセスを有効にする

ことを含む、妥当性を確認するステップと

を含むことを特徴とするコンピュータで実行する方法。

【請求項2】

データアクセスを提供するコンピュータに実装されたシステムであって、前記システムはメモリに結合されたプロセッサを備え、前記プロセッサにより実行されると請求項1に記載の方法を実行するコンピュータ実行可能命令を有する1または複数のコンピュータ読取可能な記録媒体を含むことを特徴とするコンピュータに実装されたシステム。

【請求項3】

確率のおよび／または統計に基づく分析を使用して、ユーザが自動的に実行されることを希望する動作を予知または推論する人工知能コンポーネントをさらに備えることを特徴とする請求項2に記載のコンピュータに実装されたシステム。

【請求項4】

請求項2に記載のコンピュータに実装されたシステムを使用することを特徴とするサーバ。

【請求項5】

前記XML入力データ、前記型スキーマ、前記生成されたソース・コード、前記ソース・コードをコンパイルして生成されたアセンブリ・コード、または実行中における前記実行時プロセスの間のデシリアライゼーション情報のうちの少なくとも1つをキャッシュするキャッシングコンポーネントをさらに備えることを特徴とする請求項2に記載のコンピュータに実装されたシステム。

【請求項6】

XML入力データの少なくとも1つの追加のストリームに基づいて前記型スキーマを更新するステップを更に含むことを特徴とする請求項1に記載の方法。

【請求項 7】

新しいXML入力データに対し前記型スキーマを有効にするステップを更に含むことを特徴とする請求項 1 に記載の方法。

【請求項 8】

新しいXML入力データに関する予め定められた基準における変更に基づいて前記型スキーマを更新するステップを更に含むことを特徴とする請求項 1 に記載の方法。

【請求項 9】

前記XML入力データ、前記型スキーマ、前記生成されたソース・コード、前記ソース・コードをコンパイルして生成されたアセンブリ・コード、または前記実行時プロセスの間のデシリアライゼーション情報のうちの少なくとも1つをキャッシュするステップをさらに含むことを特徴とする請求項 1 に記載の方法。

10

【請求項 10】

前記実行時プロセスの間に新しいXML入力データに対して前記型スキーマを有効にするステップを更に含むことを特徴とする請求項 1 に記載の方法。

【請求項 11】

メモリに結合されたプロセッサを含むコンピューティングシステムにおける、XMLデータの型スキーマを推論すること、前記型スキーマを具体化する動的なクラスを生成すること、および前記XMLデータにより前記クラスのオブジェクト表現をインスタンス化し、それにより推論の前に未知であったデータタイプへの静的アクセスを有効にすることを提供するコンピュータ実行可能命令を含む1または複数のコンピュータ記録媒体であって、前記コンピュータ実行可能命令は、実行されたときに、前記コンピューティングシステムに対し、

20

XML入力データの第1のストリームを受け取ること、

XML入力データの前記第1のストリームを処理して、XML入力データの前記第1のストリームの実際の実行時の値に基づいて型スキーマを推論することであって、前記型スキーマは、推論される前に未知である前記第1のストリームの前記XML入力データの編成または構造を表すこと、

前記型スキーマからソース・コードを生成すること、

前記ソース・コードをアセンブリ・コードにコンパイルすることであって、前記アセンブリ・コードは実行可能コードであること、

30

前記アセンブリ・コードを処理してメモリに読み込まれる1または複数のクラスを生成することであって、前記1または複数のクラスは前記第1のストリームの前記XML入力データを表すこと、

前記生成された1または複数のクラスの各々のインスタンスを生成すること、

前記第1のストリームの前記XML入力データを前記1または複数のインスタンスにデシリアライズし、それにより前記1または複数のインスタンスを通じて前記第1のストリームの前記XML入力データに対する遅延束縛アクセスを有効にすること、

前記実行時プロセスの間に前記型スキーマの生成のためのキャッシング動作を実行すること、

XML入力データの第2のストリームを受け取ること、

40

前記XML入力データの前記第2のストリームを処理して、前記型スキーマに対する前記第2のストリームの前記XML入力データの妥当性を確認することであって、前記型スキーマが前記第2のストリームの前記XML入力データに関して有効であるか否かを判断し、

前記型スキーマが前記第2のストリームの前記XML入力データに関して有効である場合、前記XML入力データの前記第2のストリームに対して前記型スキーマを適用し、該適用は、前記第2のストリームの前記XML入力データを前記生成された1または複数のクラスのインスタンスにデシリアライズして、それにより前記1または複数のインスタンスを通じて前記第2のストリームの前記XML入力データに対する遅延束縛アクセスを有効にし、

50

前記型スキーマが前記第2のストリームの前記XML入力データに関して有効でない場合、

XML入力データの前記第2のストリームの実際の実行時の値に基いて新しい型スキーマを推論し、

前記新しい型スキーマからソース・コードを生成し、前記ソース・コードをアセンブリ・コードにコンパイルし、かつ前記アセンブリ・コードを処理してメモリに読み込まれる1または複数の新しいクラスを生成し、前記1または複数の新しいクラスは前記第2のストリームの前記XML入力データを表し、

前記1または複数の新しいクラスの各々のインスタンスを生成し、かつ

前記第2のストリームの前記XML入力データを前記1または複数の新しいクラスの前記1または複数のインスタンスにデシリアライズし、それにより前記1または複数の新しいクラスの前記1または複数のインスタンスを通じて前記第2のストリームの前記XML入力データに対する遅延束縛アクセスを有効にする

ことを含む、妥当性を確認すること

を実行させることを特徴とするコンピュータ記録媒体。

【請求項12】

前記キャッシュする動作は前記第1のストリームの前記XML入力データ、前記型スキーマ、前記生成されたソース・コード、前記アセンブリ・コード、または前記実行時プロセスの間のデシリアライゼーション情報のうちの1または複数のキャッシュすることを含むことを特徴とする請求項11に記載の1または複数の記録媒体。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、データ処理技術に関し、より具体的には、入力データの組に基づく型推論に関する。

【背景技術】

【0002】

型推論は、完全な型情報の不足が避けられない場合に、コンパイラが式および変数の型を決定しようとする処理である。コンパイラは、入力データ・ストリームの文脈情報に基づいて、欠落した型情報を推論する。例えば、整数算術演算および浮動小数点算術演算について同じ構文が用いられる場合には、算術演算子に難しさがある場合がある。このことは、 $f\ n\ n \Rightarrow n + n$ といった関数についての型情報を曖昧さを伴わずに再構成することが不可能であるため、型推論についての問題を生じさせる。この極めて単純な関数は、加算演算が整数加算であるか浮動小数点加算であるかを見分ける方法がないため、曖昧である。例えば、この式は、型 $int \rightarrow int$ を持つ短縮形の $f\ n\ n : int \Rightarrow n + n$ か、または型 $real \rightarrow real$ を持つ $f\ n\ n : real \Rightarrow n + n$ として、解釈することができる。

【0003】

いくつかの場合においては、周囲の文脈を用いて何を意味するのかを決定する。関連する難しさの原因は、記録についての「明瞭な (sharp)」注釈である。文脈からの情報がない限り、これらの注釈を用いる関数の型を決定することはできない。したがって、関数のドメイン型を決定するのに十分な情報がないため、この関数は曖昧なものとして拒否されることになる。

【0004】

上記の例は、曖昧さが難しさにつながる場合を示すものであるが、欠落している型情報を一意的に決定することができなければ型推論が働かないものと断定することはできない。多くの場合において、省略された型情報を推論する唯一の方法は存在しないが、最良の方法は存在することがある。

【0005】

XML (eXtensible Markup Language) の主な利点の1つ

10

20

30

40

50

は、文書を、その正確なスキーマをアプリアリに知ることなく処理できることである。しかしながら、C #またはJ a v a（登録商標）などの言語を扱うことにより、プログラマは、冗長で説明的な計算上効率の悪いプログラミング・モデルを用いて、以下のコードで表されるような型無し文書にアクセスすることを強いられる。

【0006】

```
XmlDocument b=new XmlDocument();b.Load(...);  
string s=b.GetElementsByTagName("Title")[0].Value;
```

【0007】

値のスキーマまたは型が分かっている場合には、その値の部分により効率的にアクセスすることが望ましい、すなわち、型情報を推測するアクセス・パターンをコンパイルすることが望ましいであろう。スキーマ情報を推測することがなければ、何らかの形式の一般的な表現を用いて値を表さなければならず、アクセスは必然的に説明的なものとなる。コンパイル時に文書のスキーマが分かっているときは、以下のコードで表されるように、該スキーマに対応し、該文書をオブジェクト・グラフにデシリアライズすることができ、簡潔で強く型付けされた計算上効率の良い方法で該文書に対してプログラムされる、クラスの組を生成することができる。

【0008】

```
Book b=new XmlSerializer(typeof(Book)).Deserialize(...);  
string s=b.Title;
```

【発明の開示】

【発明が解決しようとする課題】

【0009】

しかしながら、これは、多くの場合においてコンパイル時に使用可能なスキーマが存在せず、説明的な手法を用いることを強いられるという点で欠点を有する。データベースならびにC #およびJ a v a（登録商標）などの今日のプログラミング言語は、非静的型の扱いが極めて不十分である。P e r l、P y t h o n、R u b y、P H P、G r o o v yといったスクリプト言語は、動的型の扱いは十分であるが、堅牢ではなく、大きなソフトウェア・システムに合わせて十分に規模を拡大することができないという点を犠牲にしている。このように、従来技術においては、動的に型付けされた言語から静的に型付けされた言語までにわたって、改良されたデータ・アクセスを提供する機構について、まだ満た

【課題を解決するための手段】

【0010】

本発明のいくつかの態様に関する基本的な理解を提供するために、以下に本発明の簡単な要約を示す。この要約は、本発明を広範囲に外観するものではない。この要約は、本発明の主要な／重要な要素を特定すること、または本発明の範囲を記述することを意図するものではない。その唯一の目的は、本発明のいくつかの概念を、後述のより詳細な説明に対する前置きとして簡略化した形式で提示することである。

【0011】

本発明は、実行時に実際の値に関してスキーマ推論を使用し、このスキーマを用いて効率的かつ簡便にアクセスできるデータの特種表現を作成することによって、純粋に動的に型付けされた言語と純粋に静的に型付けされた言語との間のギャップを埋める。（例えば、V i s u a l B a s i c、ならびに、E C M A S c r i p t、P y t h o n、およびP e r lなどのスクリプト言語における）オブジェクトに対する遅延束縛アクセスを仮定すると、本発明は、オブジェクトのシリアライズによる早期束縛アクセスの効率性と型無し文書アクセスの融通性とを組み合わせる。

【0012】

ここで開示され請求される本発明は、その1つの態様においては、入力として入力データ・ストリームを受信する推論コンポーネントを含むシステムを備える。推論コンポーネントは、実行時に入力データ・ストリームを処理し、その実際の値に基づいてスキーマを

開発する。次いで、推論されたスキーマを用いて、入力ストリーム・データの特種表現である推論された出力データを作成し、その結果、その出力データに効率的かつ簡便にアクセスすることができる。

【0013】

本発明の別の態様においては、処理のすべての時点または選択された時点においてキャッシングを利用することによって、実行時推論処理を最適化する方法が提供される。この方法は、入力データおよび／または中間データおよび／またはコードをキャッシュ・メモリにキャッシュすることによって、実行時処理の際のすべての時点または選択された時点においてキャッシングを提供するキャッシュ・マネージャを含む。

【0014】

本発明のさらに別の態様においては、本発明に従ってXML文書の実行時型推論を処理する方法が提供される。

【0015】

本発明のさらに別の態様においては、実行時に類似する入力データの処理を最適化する方法が提供される。いくつかの文書が連続して読み込まれる場合には、この方法は、以前に推論されたスキーマに対する次の文書の妥当性確認を行い、妥当性確認が成功した場合には、新たなスキーマを推論する必要はない。例えば、第1のスキーマが実行時に推論され、第2のデータの組を受信した後に、該第2のデータの組に対して該スキーマの成功（または失敗）を比較するメトリックが利用される。成功が基準を下回る場合には、新たなスキーマが第2のデータの組から推論され、以下同様である。そうでなければ、第1のスキーマが、第2のデータの組に対して適用され続ける。

【0016】

本発明の別の態様においては、本発明に従って経時的に推論されたスキーマを改良することによって実行時推論処理を最適化する方法が提供される。推論されたスキーマは、別の文書が読み込まれる度に、経時的に改良することができる。より正確な型を持つことによって、より効率的なアクセスが可能になる。

【0017】

本発明のさらに別の態様においては、確率的な分析および／または統計に基づく分析を利用して、ユーザが自動的に実行されることを望む動作を予測または推論する人工知能コンポーネントが提供される。

【0018】

前述のおよび関連する結果を達成するために、以下の説明および添付図面に関連して、本発明の特定の例示的な態様がここで説明される。しかしながら、これらの態様は、本発明の原理を利用することができる様々な方法のほんの一部を示すものであり、本発明は、こうした態様のすべておよびそれらの等価物を含むように意図される。本発明の他の利点および新規な特徴は、図面と併せて検討したときに本発明の以下の詳細な説明から明らかとなるであろう。

【発明を実施するための最良の形態】

【0019】

ここで図面を参照して本発明を説明するが、図面においては、全体を通して、同じ参照番号は同じ要素を示すように用いられる。以下の記述においては、説明の目的で、本発明の完全な理解をもたらすように多くの特定の詳細が記載される。しかしながら、本発明は、これらの特定の詳細なしに実施できることが明らかであろう。他の例においては、本発明の説明を容易にするために、周知の構造および装置がブロック図で示される。

【0020】

本出願で用いられる場合には、「コンポーネント」および「システム」という用語は、コンピュータ関連エンティティ、すなわち、ハードウェア、ハードウェアおよびソフトウェアの組み合わせ、ソフトウェア、または実行中のソフトウェアのいずれかを指すことを意図している。例えば、コンポーネントは、プロセッサ上で動作している処理、プロセッサ、オブジェクト、実行可能ファイル、実行のスレッド、プログラム、および／または、

10

20

30

40

50

コンピュータとすることができるが、これらに限定されるものではない。一例として、サーバ上で動作しているアプリケーションおよび該サーバの両方をコンポーネントとすることができる。1つまたは複数のコンポーネントは、実行の処理および／またはスレッド内に常駐することができ、1つのコンポーネントは、1つのコンピュータ上に配置するか、および／または、2つまたはそれ以上のコンピュータ間に分散させることができる。

【0021】

ここで用いられる場合には、「推論する」または「推論」という用語は、一般に、イベントおよび／またはデータを介して取り込まれる観測結果の組から、システム、環境、および／またはユーザの状態を推理または推論する処理を指す。推論は、例えば、特定の文脈または動作を特定するのに利用するか、または状態に関する確率分布を生成することができる。推論は、確率的なもの、すなわち、データおよびイベントの検討に基づく対象の状態に関する確率分布の計算とすることができる。推論はまた、イベントおよび／またはデータの組からより高いレベルのイベントを構成するために利用される技術を指す場合がある。こうした推論は、イベント同士が時間的に近接して関連付けられるかどうか、ならびに、イベントおよびデータが1つまたはいくつかのイベントおよびデータソースに由来するものであるかどうかに関わらず、観測されたイベントおよび／または格納されたイベント・データの組から新たなイベントまたは動作を構築することにつながる。

10

【0022】

最初に図面を参照すると、図1は、本発明に従って実行時にスキーマを推論する型推論システム100を示す。本発明の主な態様は、実行時に実際の値に対してスキーマ推論を用いること、および、そのスキーマを用いて効率的かつ簡便にアクセスすることができるデータの特殊表現を作成することである。

20

【0023】

上記の例を続けて、(例えばVisual Basicにおける)型オブジェクトの値に対する遅延束縛アクセスを仮定すると、以下の擬似コードを用いて新規な機構を表現することができる。

【0024】

```
XmlDocument b=new XmlDocument();b.Load(...);  
Type T=InferSchemaAndGenerateCode(b);  
object b=new XmlSerializer(T).Deserialize(...);  
object s=b.Title;
```

30

【0025】

実際の文書インスタンスdを用いて実行時型Tを生成し、次いで実行時型Tを用いて、文書dを動的型Tのオブジェクト・グラフにデシリアライズすることができる。Tは静的に未知であるため、(文書が静的型objectを持つものと仮定すると)遅延束縛方式でのみ文書に対してプログラムすることができる。ここで、通常のオブジェクト処理を行う能力のため、文書(XmlDocument b)をオブジェクト表現に読み込み、次いでタイトルにアクセスすることがより一層効率的である。object b=new XmlSerializer(T).Deserialize(...);、およびobject s=b.Title;というコードは、該コードが書かれたときにはbがどのような型であるかが静的に未知であるため、オブジェクトに対する遅延束縛アクセスを定める。しかしながら、依然としてアクセスされることが望まれている。

40

【0026】

背景技術における第2の例を参照すると、本発明によれば、実行時までBookの型を定める必要はない。

【0027】

この処理の効率を高めるいくつかの方法が存在する。第1に、InferSchemaAndGenerateCodeアルゴリズムにおいて、種々のキャッシング戦略を実行することができる。例えば、いくつかの文書が連続して読み込まれる場合には、以前に推論されたスキーマに対する次の文書の妥当性を最初に確認することが望ましく、妥当性確

50

認が成功した場合には、新たなスキーマを推論する必要はなく、新たな型が生成される。代替的には、別の文書が読み込まれる度に、最初に推論されたスキーマを経時的に改良することができる。より正確な型を持つことによって、より効率的なアクセスが可能になる。これらの最適化は、後述される。

【0028】

これに対応して、入力としてデータ・ストリームを受信する推論コンポーネント102を含むシステム100が提供される。推論コンポーネント102は、入力データ・ストリームを処理し、入力データの実際の値に基づいて実行時にスキーマ104を推論する。次いで、スキーマ104を用いて、入力ストリーム・データの特種表現である推論された出力データを生成し、その結果、その出力データに効率的かつ簡便にアクセスすることができる。

10

【0029】

図2は、本発明に従って実行時型推論を提供する方法を示す。説明を簡単にするために、例えばフローチャートの形式でここに示される1つまたは複数の方法は、一連の動作として示され説明されるが、いくつかの動作は、本発明によれば、ここに示され説明される動作とは異なる順序でおおよそ／またはそれらの動作と同時に行われる場合があるため、本発明は、動作の順序によって制限されるものではないことを理解し、認識すべきである。例えば、当業者であれば、方法が、状態図の場合のように一連の相関状態またはイベントとして代替的に表すことができることを理解し、認識するであろう。さらに、本発明に係る方法を実行するために、示されたすべての動作が必要であるとは限らない。

20

【0030】

200において、システムは入力データ・ストリームを受信する。この入力データは、ほんの数例を挙げると、XML (eXtensible Markup Language) 文書形式のもの、文脈自由文法 (context-free grammars) によって記述される言語と互換性のあるデータ、および、リレーショナル・スキーマによって記述されるリレーショナル・テーブル形式のデータとすることができる。202において、推論コンポーネントが入力データ・ストリームを処理し、実行時にそこからスキーマを推論する。204においては、このスキーマを利用して、入力データ・ストリームの実際の値に基づいて推論された出力データを生成する。

【0031】

ここで図3を参照すると、本発明に係る実行時型推論の一般化した方法が示される。データの入力ストリーム300が、実行時処理への入力として受信される。302において、実行時に入力データについてスキーマが推論される。304において、推論されたスキーマからソース・コードが生成される。306において、ソース・コードは、実行可能コードとすることができるアセンブリ・コードにコンパイルされる。308において、アセンブリ・コードが入力データ300の型に対して処理され、生成されたクラスがメモリに読み込まれる。310において、そのクラスが元のデータ入力ストリーム300のデシリアライゼーションに用いられ、新たに作成され読み込まれたクラスのインスタンス312を生成する。その結果、インスタンス312に、効率的かつ簡便にアクセスすることができる。

30

40

【0032】

図4は、本発明に従ってXML文書402の実行時型推論を処理する方法400を示す。XMLおよびオブジェクトのシリアライゼーションの場合には、スキーマを用いて、例えばMicrosoft社による、NETブランドのフレームワークにおける既存のxsd.exeツールを用いてソース・コードを生成し、該ソース・コードをC#コンパイラを用いてコンパイルし、アセンブリをメモリに読み込み、作成され読み込まれたばかりのクラスのインスタンスとして元の文書をデシリアライズすることができる。しかしながら、Reflection.Emitなどの推論されたスキーマ、またはより軽い形式の実行時コード生成から、クラスを生成する他の多くの代替方法が存在する。NETブランドのフレームワークのReflection名前空間は、アセンブリ・メタデータを表示

50

し、実行時にアセンブリを作成することを容易にする。例えば、Reflectionは、コードが、型発見を行い、アセンブリ・メタデータを表示し、実行時にアセンブリ・コードを動的に呼び出すことを可能にする。

【0033】

したがって、XML文書402を実行時処理400への入力データとして受信する方法が提供される。404において、スキーマは、XML文書402の実際の値に基づいて実行時に推論される。406において、推論されたスキーマからソース・コードが生成される。408において、そのソース・コードは、例えばC#コンパイラを用いて、実行可能コードとすることができるアセンブリ・コードにコンパイルされる。410において、そのアセンブリ・コードを用いてXML文書402の型が処理され、メモリに読み込まれるクラスを生成する。412において、そのクラスが元のデータ入力ストリームのデシリアライゼーションに用いられ、新たに作成され読み込まれたクラスのインスタンス414を作成する。その結果、インスタンス414に、効率的かつ簡便にアクセスすることができる。

10

【0034】

図1の擬似コード例を再び参照すると、スキーマが推論されたときには、XML文書をオブジェクト・グラフにデシリアライズするために一般的なXMLシリアライザを用いる必要がないことに注目されたい。その代わり、例えばISerializableインターフェースのカスタム実装を提供することによって、同様にシリアライゼーションおよびデシリアライゼーションのために、型固有のコードを生成することができる。

20

【0035】

この例は、XML文書およびスキーマに関するものであるが、この機構は、文脈自由文法によって記述される言語、リレーショナル・スキーマによって記述されるリレーショナル・テーブルなどといった他のドメインに対しても、等しく適用可能である。さらに、本発明において説明される機構は、オブジェクトにデシリアライズする代わりに、スキーマ情報（例えば、スキーマ固有のDOM（文書オブジェクト・モデル）または解析ツリー）を用いて適合されたいずれかの表現を生成することにも等しく適用される。例えば、XML DOMは、XML文書のためのプログラミング・インターフェースであり、XML文書にアクセスして操作することができる方法を定義する。XMLパーサ（parser）と呼ばれるプログラムは、XML文書をコンピュータのメモリに読み込むのに用いることができる。文書が読み込まれるときに、DOMにアクセスすることによって、文書情報を取り出して、操作することができる。DOMは、XML文書のツリー表示を表す。

30

【0036】

図5は、本発明に従って、処理500のすべての時点または選択された時点においてキャッシングを利用することによって、実行時推論処理500を最適化する方法を示す。この方法は、入力データ504および／または中間データおよび／またはコードをキャッシュ・メモリ506にキャッシュすることによって、実行時処理500の際のすべての時点または選択された時点においてキャッシングを提供するキャッシュ・マネージャ502を含む。508において、実行時スキーマを生成するためにより高速にアクセスできるように、データの入力ストリーム504を、実行時処理への入力として受信されたときにキャッシュすることができる。510において、ソース・コードを生成するために、より高速のアクセスおよび処理を目的としてスキーマをキャッシュすることができる。512において、ソース・コードは、実行可能コードとすることができるアセンブリ・コードにコンパイルされる。ソース・コードは、より高速のアクセスのためにキャッシュすることができる。514において、入力データ型に対するより高速のアクセスおよび処理のために実行可能アセンブリ・コードをキャッシュし、処理メモリに読み込まれるクラスを生成することができる。516において、元のデータ入力ストリームに対するデシリアライゼーション処理に備えて、クラス型をキャッシュすることができる。518において、新たに作成され読み込まれたクラスのインスタンス520であるデシリアライゼーション処理の出力をキャッシュすることができる。これにより、インスタンス520に対して、より効率

40

50

的かつ簡便にアクセスすることができる。

【0037】

ここで図6を参照すると、本発明に従って実行時における類似する入力データの処理を最適化する方法が示される。例えばいくつかの文書が連続して読み込まれる場合には、以前に推論されたスキーマに対する次の文書の妥当性を最初に確認することが望ましく、妥当性確認が成功した場合には、新たなスキーマを推論する必要はない。例えば、第1のスキーマが実行時に推論され、第2のデータの組を受信した後に、第2のデータの組に対してそのスキーマの成功（または失敗）を比較するメトリックが利用される。成功が基準を下回る場合には、新たなスキーマが第2のデータの組から推論され、以下同様である。そうでなければ、第1のスキーマが、第2のデータの組に対して適用され続ける。

10

【0038】

したがって、600において、多数の類似データの組に対して推論されたスキーマを適用するために、基準を決定して利用する。602において、第1の入力データの組が受信される。604において、第1のスキーマが、実行時に第1のデータの組から推論される。606において、第2のデータの組が受信される。608においては、第1の推論されたスキーマを用いて、第2のデータの組に対して実行時に妥当性確認処理が行われる。610において検査されたときに妥当性確認が失敗した場合には、フローは612に向かい、新たなスキーマが第2のデータの組から推論される。614において、第2の推論されたスキーマに従って、および前述の処理の開発ソース・コード、コンパイルング、デシリアライゼーションなどに従って、第2のデータの組が処理される。あるいは、610において妥当性確認が成功した場合には、616に示されるように、第1の推論されたスキーマが第2のデータの組に対して適用される。次いで、600に戻って次のデータの組を処理する。

20

【0039】

この処理は2回の繰り返しに限定されるものではない。例えば、最初の2つのスキーマが用いられない場合には、第3の推論されたスキーマを導出することができ、以下同様である。しかしながら、実質的に類似する入力データを使用するときにスキーマを推論する数または深さに対して限界が存在すべきであり、そうでなければ、システム性能が影響を受ける場合がある。

【0040】

30

図7は、本発明に従って、推論されたスキーマを経時的に改良することによって実行時推論処理を最適化する方法を示す。推論されたスキーマは、別の文書が読み込まれる度に経時的に改良することができる。より正確な型を持つことによって、より効率的なアクセスが可能になる。したがって、700において、入力データ型からクラスを導出するために、推論されたスキーマの適用について基準が設定される。例えば、基準が比較的高く設定され、推論されたスキーマが、入力データに対して処理されたときにその基準を下回る出力を発生させた場合には、出力された推論されたスキーマが最終的に所望の基準を達成するように、推論アルゴリズムを改良することができる。その時点では、それ以上の改良は処理されない。

【0041】

40

702において、実行時の推論処理のために第1の入力データの組が受信される。704において、入力データに基づく推論処理によってスキーマが導出される。706において、処理のために次の入力データの組が受信される。708において、推論されたスキーマが所定の基準に従って動作しているかどうかを判断するために、検査が行われる。動作している場合には、706に戻って次のデータの組が受信される。動作が所望の基準を満たしていない場合には、フローは708から710に進み、実行時に既存のスキーマを更新または改良する。次いでフローは706に戻り、処理のために次の入力データの組が受信される。代替的には、実行時に最適な出力を利用することを保証する更新されたスキーマを用いて、推論されたスキーマが基準を満たさない原因となった同じ入力データの組を、再び処理することができる。

50

【0042】

図8は、本発明に従って、1つまたは複数の機能を自動化することを容易にする人工知能(AI)コンポーネント802を利用するシステム800を示す。本発明は(例えば、推論に関連して)、本発明の様々な態様を実施するための様々なAIベースのスキーマを利用することができる。例えば、自動分類システムおよび処理によって、推論されたスキーマを更新することに関する所定の基準または推論されたスキーマのいずれかを改良する時期を判断するための処理を容易に行うことができる。

【0043】

分類子は、入力属性ベクトル $x = (x_1, x_2, x_3, x_4, x_n)$ を、入力がクラスに属する $confidence$ にマップする関数、すなわち $f(x) = confidence(class)$ である。こうした分類は、ユーザが自動的に実行されることを望む動作を予測または推論するために、(例えば、分析の有用性およびコストを考慮に入れる)確率的なおよび/または統計に基づく分析を利用することができる。

【0044】

サポートベクタマシン(SVM)は、利用することができる分類子の一例である。SVMは、可能な入力空間において、非トリガ・イベントからトリガ基準を分離しようとする超曲面を発見することによって動作する。直観的には、これにより、訓練データに近いが同一ではない試験データについて、分類が正しいものになる。他の有向および非有向モデル分類手法として、例えば、ナイーブ・ベイズ、ベイジアン・ネットワーク、決定木、ニューラル・ネットワーク、ファジー論理モデルが挙げられ、独立の異なるパターンを提供する確率的分類モデルを利用することができる。ここで用いられる場合には、分類は、優先順位のモデルを開発するのに利用される統計的回帰を含む。

【0045】

本明細書から容易に理解されるように、本発明は、(例えば、ユーザのふるまいを観測すること、外部情報を受信することによって)黙示的に訓練された分類子だけでなく、(例えば、一般的な訓練データによって)明示的に訓練された分類子を利用することができる。例えば、SMVは、分類子コンストラクタおよび機能選択モジュール内の学習または訓練フェーズによって設定される。このように、1つまたは複数の分類子は、以前に推論されたスキーマを更新または改良する時期を所定の基準に従って判断すること、処理中のデータの種類(例えば、金融対非金融、個人対非個人、...)に基づいて推論アルゴリズム上の基準を厳しくすること、および、一日のいずれかの時間に(例えば、システム性能がそれほど影響を受けない夕方に)より厳しい基準制御を実行することを含むがこれらに限定されるものではない多くの関数を、自動的に学習して実行するために用いることができる。

【0046】

ここで図9を参照すると、開示されたアーキテクチャを実行するように動作可能なコンピュータのブロック図が示される。本発明の様々な態様について付加的な文脈を与えるために、図9および以下の記述は、本発明の様々な態様を実装することができる適切なコンピュータ環境900に関する簡潔で一般的な説明を提供するように意図されている。本発明は、1つまたは複数のコンピュータ上で動作することができるコンピュータ実行可能命令を一般的な背景として上述されたが、当業者であれば、本発明はまた、他のプログラム・モジュールと組み合わせて、および/または、ハードウェアとソフトウェアの組み合わせとして、実装できることが分かるであろう。

【0047】

一般に、プログラム・ジュールは、特定のタスクを実行するか、または特定の抽象データ型を実装する、ルーチン、プログラム、コンポーネント、データ構造などを含む。さらに、当業者であれば、本発明の方法は、単一プロセッサ・コンピュータ・システムまたはマルチプロセッサ・コンピュータ・システム、ミニコンピュータ、メインフレーム・コンピュータ、ならびに、パーソナル・コンピュータ、ハンドヘルド・コンピュータ装置、マイクロプロセッサ・ベース家庭用電化製品またはプログラム可能な家庭用電化製品などを

含む他のコンピュータ・システム構成を用いて実施することができ、その各々は1つまたは複数の関連装置に動作可能に結合できることが分かるであろう。

【0048】

本発明の例示された態様はまた、通信ネットワークを介してリンクされるリモートの処理装置によって特定のタスクが実行される分散コンピューティング環境において実施することができる。分散コンピューティング環境においては、プログラム・モジュールを局所およびリモートのメモリ記憶装置の両方に配置することができる。

【0049】

コンピュータは、典型的には、種々のコンピュータ可読メディアを含む。コンピュータ可読メディアは、コンピュータによってアクセスできるいずれかの利用可能なメディアとすることができ、揮発性メディアおよび不揮発性メディア、ならびに、取り外し可能なメディアおよび取り外し不可能なメディアの両方を含む。例として、コンピュータ可読メディアは、コンピュータ記憶メディアおよび通信メディアを備えるものとすることができるが、これらに限定されるものではない。コンピュータ記憶メディアは、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、または他のデータといった情報を記憶するためのいずれかの方法または技術において実装される、揮発性メディアおよび不揮発性メディア、ならびに、取り外し可能なメディアおよび取り外し不可能なメディアの両方を含む。コンピュータ記憶メディアは、RAM、ROM、EEPROM、フラッシュ・メモリもしくは他のメモリ技術、CD-ROM、デジタル・ビデオ・ディスク（DVD）もしくは他の光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置もしくは他の磁気記憶装置、または、所望の情報を格納するのに用いることができ、コンピュータによってアクセスすることができる他のいずれかのメディアを含むが、これらに限定されるものではない。

【0050】

通信メディアは、典型的には、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、または他のデータを、搬送波または他の搬送機構などの変調されたデータ信号に統合したものであり、あらゆる情報提供メディアを含む。「変調されたデータ信号」という用語は、情報を信号に符号化するような方式で設定または変更された1つまたは複数の特性を有する信号を意味する。例として、通信メディアは、有線ネットワークまたは直接有線接続などの有線メディア、ならびに、音響メディア、RFメディア、赤外線メディア、および他の無線メディアなどの無線メディアを含むが、これらに限定されるものではない。上記のいずれかの組み合わせも、コンピュータ可読メディアの範囲内に含むべきである。

【0051】

再び図9を参照すると、処理ユニット904、システム・メモリ906、およびシステム・バス908を備えるコンピュータ902を含む、本発明の種々の態様を実装するための例示的な環境900が示されている。システム・バス908は、システム・メモリ906を含むが、それに限定されるものではないシステム・コンポーネントを、処理ユニット904に結合する。処理ユニット904は、種々の市販プロセッサのいずれかとすることができる。処理ユニット904として、デュアル・マイクロプロセッサおよび他のマルチプロセッサ・アーキテクチャを利用することもできる。

【0052】

システム・バス908は、（メモリ・コントローラ付きの、またはメモリ・コントローラ無しの）メモリ・バス、周辺バス、および様々な市販のバス・アーキテクチャのいずれかを用いるローカル・バスにさらに相互接続することができる幾つかの種類のバス構造の何れかとすることができる。システム・メモリ906は、読み取り専用メモリ（ROM）910およびランダム・アクセス・メモリ（RAM）912を含む。基本入力／出力システム（BIOS）は、ROM、EPROM、EEPROMなどの不揮発性メモリ910に格納され、起動の際などにコンピュータ902内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。RAM912は、データをキャッシュするために、スタティックRA

Mなどの高速RAMを含むこともできる。

【0053】

コンピュータ902はさらに、外部での使用のために適切な筐体（図示せず）内に構成することができる内部ハード・ディスク・ドライブ（HDD）914（例えば、EIDE、SATA）、（例えば、取り外し可能なディスク918との間で読み書きするための）磁気フロッピー（登録商標）・ディスク・ドライブ（FDD）916、および、（例えば、CD-ROMディスク922から読み出すか、または、DVDなどの他の大容量光メディアとの間で読み書きするための）光ディスク・ドライブ920を含むことができる。ハード・ディスク・ドライブ914、磁気ディスク・ドライブ916、および光ディスク・ドライブ920は、それぞれ、ハード・ディスク・ドライブ・インターフェース924、磁気ディスク・ドライブ・インターフェース926、および光ドライブ・インターフェース928によって、システム・バス908に接続することができる。外部ドライブ実装のためのインターフェース924は、Universal Serial Bus（USB）およびIEEE1394インターフェース技術の少なくとも一方または両方を含む。

10

【0054】

ドライブおよびその関連するコンピュータ可読メディアは、データ、データ構造、コンピュータ実行可能命令などの不揮発性記憶装置となる。コンピュータ902の場合は、ドライブおよびメディアは、適切なデジタル形式のあらゆるデータの記憶装置に対応する。上記のコンピュータ可読メディアの説明は、HDD、取り外し可能な磁気ディスク、および、CDまたはDVDなどの取り外し可能な光メディアについて述べたが、当業者であれば、例示的な動作環境において、ジップ・ドライブ、磁気カセット、フラッシュ・メモリ・カード、カートリッジなどといった、コンピュータによって読み取ることができる他の種類のメディアを用いることができ、さらに、こうしたあらゆるメディアは、本発明の方法を実行するためのコンピュータ実行可能命令を含むことができることを、認識すべきである。

20

【0055】

ドライブおよびRAM912には、オペレーティング・システム930と、1つまたは複数のアプリケーション・プログラム932と、他のプログラム・モジュール934と、プログラム・データ936とを含む多くのプログラム・モジュールを格納することができる。オペレーティング・システム、アプリケーション、モジュール、および／またはデータのすべてまたは一部は、RAM912にキャッシュすることもできる。本発明は、様々な市販のオペレーティング・システムまたはオペレーティング・システムの組み合わせを用いて実装できることが分かる。

30

【0056】

ユーザは、1つまたは複数の有線／無線入力装置、例えばキーボード938およびマウス940などのポインティングデバイスを介して、コマンドおよび情報をコンピュータ902に入力することができる。他の入力装置（図示せず）として、マイクロフォン、IRリモートの制御装置、ジョイスティック、ゲーム・パッド、スタイラス・ペン、タッチ・スクリーンなどを挙げることができる。これらの入力装置および他の入力装置は、多くの場合、システム・バス908に結合される入力装置インターフェース942を介して処理ユニット904に接続されるが、パラレル・ポート、IEEE1394シリアル・ポート、ゲーム・ポート、USBポート、IRインターフェースなどといった他のインターフェースによって接続することもできる。

40

【0057】

モニター944または他の種類のディスプレイ装置もまた、ビデオ・アダプタ946などのインターフェースを介してシステム・バス908に接続される。コンピュータは、モニター944に加えて、典型的には、スピーカ、プリンタなどといった他の周辺出力装置（図示せず）を含む。

【0058】

50

コンピュータ 902 は、リモートコンピュータ 948 などの 1 つまたは複数のリモートコンピュータへの、有線通信および／または無線通信を介した論理接続を用いて、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 948 は、ワークステーション、サーバ・コンピュータ、ルータ、パーソナル・コンピュータ、ポータブル・コンピュータ、マイクロプロセッサ・ベースの娯楽用電気製品、ピア装置、または他の共通ネットワーク・ノードとすることができ、分かり易くするためにメモリ記憶装置 950 のみが示されているが、典型的には、コンピュータ 902 に関連して説明された要素の多くまたはすべてを含む。図示される論理接続は、ローカル・エリア・ネットワーク (LAN) 952、および／または、より大きいネットワーク、例えば広域エリア・ネットワーク (WAN) 954 への有線／無線接続を含む。こうした LAN および WAN ネットワーキング環境は、事務所および企業においては一般的なものであり、イントラネットなどの企業規模のコンピュータ・ネットワークを容易にし、それらのすべては、グローバル通信ネットワーク、例えばインターネットに接続することができる。

10

【0059】

LAN ネットワーキング環境で用いられるときは、コンピュータ 902 は、有線および／または無線通信ネットワーク・インターフェースまたはアダプタ 956 を介して、ローカル・ネットワーク 952 に接続される。アダプタ 956 は、該無線アダプタ 956 と通信するために配置された無線アクセス・ポイントを含むこともできる LAN 952 との有線または無線通信を容易にすることができる。

20

【0060】

WAN ネットワーキング環境で用いられるときは、コンピュータ 902 は、モデム 958 を含むか、WAN 954 上の通信サーバに接続されるか、または、例えばインターネットを介して WAN 954 上で通信を確立するための他の手段を有することができる。内部装置または外部装置および有線装置または無線装置とすることができるモデム 958 は、シリアル・ポート・インターフェース 942 を介してシステム・バス 908 に接続される。ネットワーク化された環境においては、コンピュータ 902 に関連して図示されたプログラム・モジュールまたはその一部は、リモートのメモリ／記憶装置 950 に格納することができる。示されるネットワーク接続は例示的なものであり、コンピュータ間の通信リンクを確立する他の手段を使用できることが分かるであろう。

30

【0061】

コンピュータ 902 は、例えば、プリンタ、スキャナ、デスクトップおよび／またはポータブル・コンピュータ、携帯データ端末、通信衛星、無線検出可能タグと関連付けられたあらゆる機器または場所 (例えば、キオスク、新聞売り場、化粧室)、および電話といった、無線通信内に動作可能に配置されるいずれかの無線装置またはエンティティと通信するように動作することができる。これは、少なくとも Wi-Fi および Bluetooth (商標) 無線技術を含む。したがって、通信は、従来のネットワークまたは少なくとも 2 つの装置間の単なるアドホック通信と同様の所定の構成とすることができる。

【0062】

Wi-Fi すなわち Wireless Fidelity は、無線で、家のソファ、ホテルの部屋のベッド、または職場の会議室からインターネットへの接続を可能にする。Wi-Fi は、携帯電話に用いられるものと同様の無線技術であり、こうした装置例えばコンピュータが、屋内および屋外すなわち基地局の範囲内のどこでもデータを送受信することを可能にする。Wi-Fi ネットワークは、IEEE 802.11 (a、b、g など) と呼ばれる無線技術を用いて、安全で信頼性が高い高速無線接続を提供する。Wi-Fi ネットワークを用いて、コンピュータを、相互に、インターネットに、および (IEEE 802.3 またはイーサネット (登録商標) を用いる) 有線ネットワークに、接続することができる。Wi-Fi ネットワークは、例えば、データ速度 11 Mbps (802.11a) または 54 Mbps (802.11b) で、ライセンスのいらぬ 2.4 GHz および 5 GHz 無線帯域において動作するか、または両方の帯域 (デュアル・バンド) を含む製品で動作し、したがって、該ネットワークは、多くの事務所で用いられる基本的な 1

40

50

OBASET有線イーサネット（登録商標）・ネットワークと同様の実性能を提供することができる。

【0063】

ここで図10を参照すると、本発明に係る例示的なコンピューティング環境1000の概略的なブロック図が示されている。システム1000は、1つまたは複数のクライアント1002を含む。クライアント1002は、ハードウェアおよび／またはソフトウェア（例えば、スレッド、処理、コンピューティング装置）とすることができる。クライアント1002は、例えば、本発明を利用することによって、1つまたは複数のクッキーおよび／または関連する文脈情報を保管することができる。

【0064】

システム1000はまた、1つまたは複数のサーバ1004を含む。サーバ1004もまた、ハードウェアおよび／またはソフトウェア（例えば、スレッド、処理、コンピューティング装置）とすることができる。サーバ1004は、例えば、本発明を利用することによって変換を実行するスレッドを保管することができる。クライアント1002とサーバ1004との間の1つの可能な通信は、2つまたはそれ以上のコンピュータ処理間で伝送されるのに適したデータ・パケット形式とすることができる。データ・パケットは、例えば、クッキーおよび／または関連する文脈情報を含むことができる。システム1000は、1つまたは複数のクライアント1002と1つまたは複数のサーバ1004との間の通信を容易に行うのに利用することができる通信フレームワーク1006（例えば、インターネットなどのグローバル通信ネットワーク）を含む。

【0065】

通信は、有線（光ファイバーを含む）および／または無線技術を介して容易に行うことができる。1つまたは複数のクライアント1002は、該クライアント1002内での情報（例えば、1つまたは複数のクッキーおよび／または関連する文脈情報）を格納するのに利用することができる1つまたは複数のクライアント・データ格納装置1008に動作可能に接続される。同様に、1つまたは複数のサーバ1004は、該サーバ1004内での情報を格納するのに利用することができる1つまたは複数のサーバ・データ格納装置1010に動作可能に接続される。

【0066】

上述されたものは、本発明の例を含む。当然のことながら、本発明を説明するために、コンポーネントまたは方法について考え得るあらゆる組み合わせを記述することは不可能であるが、当業者であれば、本発明のさらに多くの組み合わせおよび置換が可能であることが分かるであろう。したがって、本発明は、特許請求の範囲の趣旨および範囲内にあるこうしたすべての代替、修正、および変更を包含するように意図されている。さらに、詳細な説明または特許請求の範囲のいずれかにおいて「含む」という用語が用いられる限りにおいて、こうした用語は、「備える、含む」という用語が特許請求の範囲において遷移語（transitional word）として使用されるときに解釈される場合と同様に、包括的なものであるように意図されている。

【図面の簡単な説明】

【0067】

【図1】本発明に従って実行時にスキーマを推論する型推論システムを示す図である。

【図2】本発明に係る実行時型推論を提供する方法を示す図である。

【図3】本発明に係る実行時型推論の一般化した方法を示す図である。

【図4】本発明に従ってXML文書の実行時型推論を処理する方法を示す図である。

【図5】本発明に従って、処理のすべての時点または選択された時点においてキャッシングを利用することによって、実行時推論処理を最適化する方法を示す図である。

【図6】本発明に従って実行時に類似する入力データの処理を最適化する方法を示す図である。

【図7】本発明に従って経時的に推論スキーマを改良することによって実行時推論処理を最適化する方法を示す図である。

【図 8】本発明に係る 1 つまたは複数の機能を自動化することを容易にする人工知能コンポーネントを利用するシステムを示す図である。

【図 9】開示されたアーキテクチャを実行するように動作可能なコンピュータのブロック図である。

【図 10】本発明に係る例示的なコンピューティング環境の概略的なブロック図である。

【符号の説明】

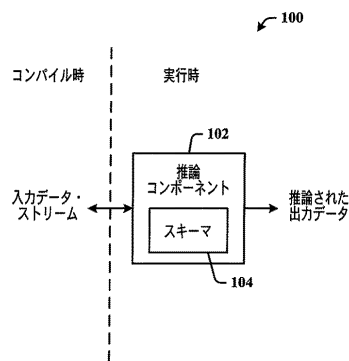
【0068】

102 推論コンポーネント
 104 スキーマ
 300 データ入力ストリーム
 310 デシリアライズ
 312 インスタンス
 402 XML 文書
 414 インスタンス
 504 データ入力ストリーム
 520 インスタンス
 102 推論コンポーネント
 104 スキーマ
 802 AI コンポーネント
 1002 クライアント
 1006 通信フレームワーク
 1004 サーバ
 1008 クライアント・データ格納装置
 1010 サーバ・データ格納装置

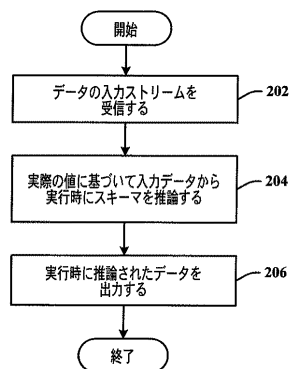
10

20

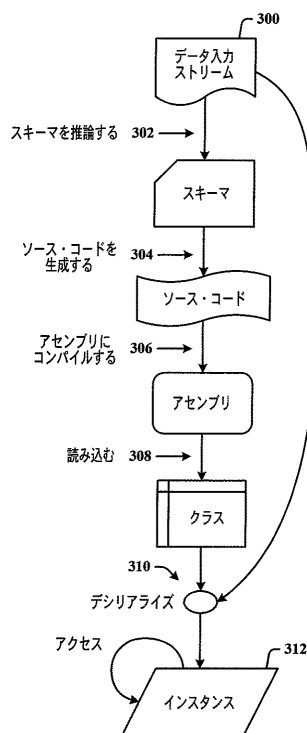
【図 1】



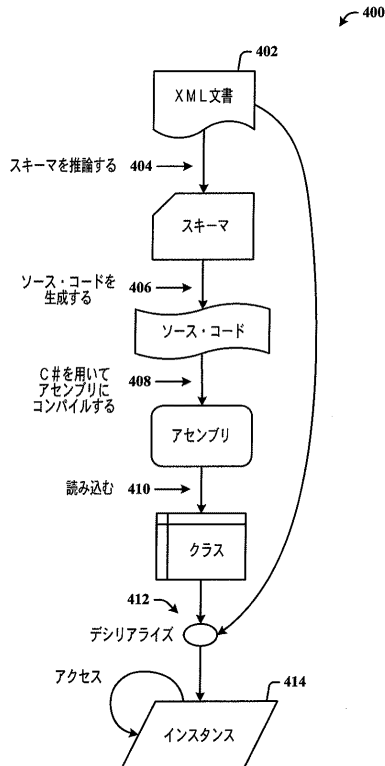
【図 2】



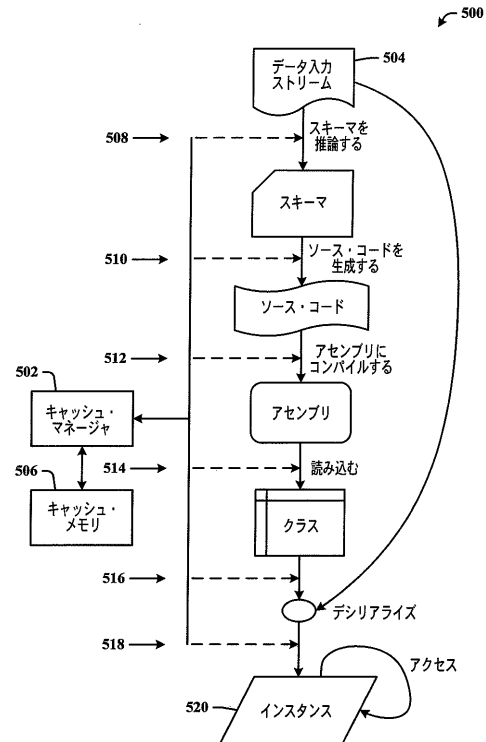
【図 3】



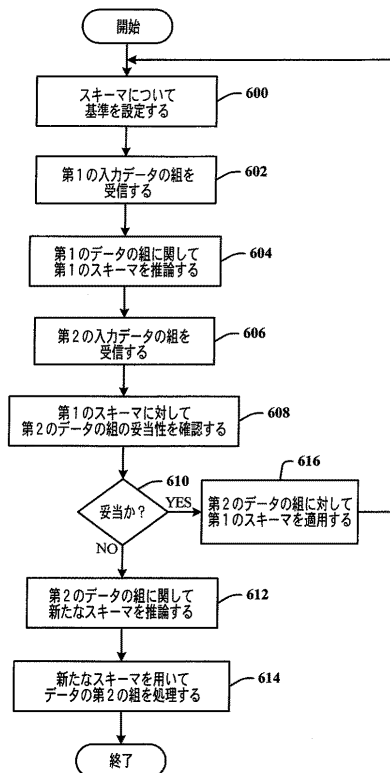
【図 4】



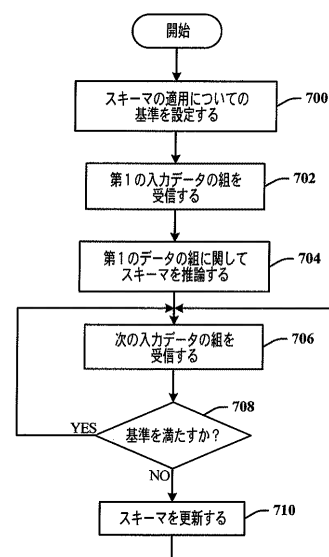
【図 5】



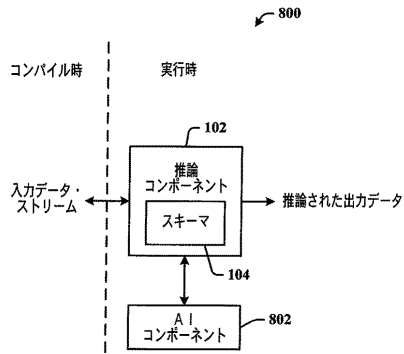
【図 6】



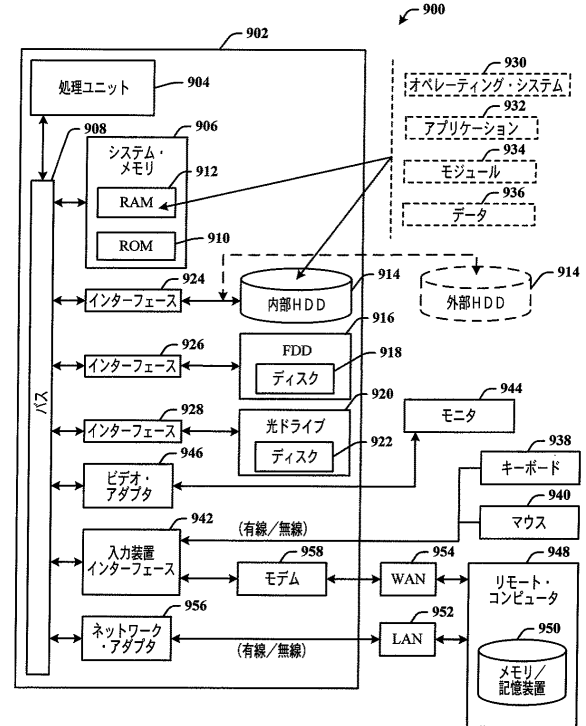
【図 7】



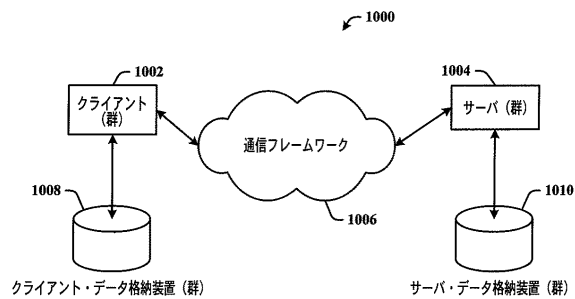
【図 8】



【図 9】



【図 10】



フロントページの続き

(72)発明者 オルデア ビクター オバサンジョ
アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ マ
イクロソフト コーポレーション内

審査官 川崎 優

(56)参考文献 特開2003-085016 (JP, A)
Nithya Sampathkumar, Using the XSD Inference Utility, MSDN library, 2003年 2月1
2日, URL, <http://msdn.microsoft.com/en-us/library/aa302302.aspx>

(58)調査した分野(Int.Cl., DB名)
G06F 9/44, 45、17/21-28