(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4039484号
(P4039484)
(45)発行日　平成20年1月30日(2008.1.30)　(24)登録日　平成19年11月16日(2007.11.16)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 546Z
G06F 12/00 547Z

請求項の数 24　(全 42 頁)

(21)出願番号　特願2002-267625(P2002-267625)
(22)出願日　平成14年9月13日(2002.9.13)
(65)公開番号　特開2003-323332(P2003-323332A)
(43)公開日　平成15年11月14日(2003.11.14)
審査請求日　平成15年1月8日(2003.1.8)
(31)優先権主張番号　特願2002-53265(P2002-53265)
(32)優先日　平成14年2月28日(2002.2.28)
(33)優先権主張国　日本国(JP)

前置審査

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】ＸＰａｔｈ評価方法、これを用いたＸＭＬ文書処理システム及びプログラム

(57)【特許請求の範囲】
【請求項１】
コンピュータを用いて、所定のデータファイルを対象として当該データファイルがＸＰａｔｈで指定される特定部分を持つか否かを評価するＸＰａｔｈ評価方法において、
メモリから評価を行うべき複数のＸＰａｔｈ式を読み出し、当該複数のＸＰａｔｈ式から、当該複数のＸＰａｔｈ式の共通部分を共通のノードとし、当該複数のＸＰａｔｈ式の他の部分を個別のノードとする木構造のデータ構造を生成する工程と、
メモリから処理対象のデータファイルを入力し、前記データ構造のノードごとに、当該ノードに対応する前記ＸＰａｔｈ式の部分で指定される前記特定部分を当該処理対象のデータファイルが持つか否かを評価し、評価結果として得られた当該特定部分を結合する工程と、
前記複数のＸＰａｔｈ式の各ＸＰａｔｈ式に関して、個々の当該ＸＰａｔｈ式を構成する部分に対応する前記ノードごとの評価結果に基づいて各ＸＰａｔｈ式の評価結果を得、得られた評価結果をメモリに格納する工程と
を含むことを特徴とするＸＰａｔｈ評価方法。
【請求項２】
前記データ構造を生成する工程は、
個々の前記ＸＰａｔｈ式をロケーションステップごとに分解する工程と、
分解して得られた各ロケーションステップにノードを対応させ、複数のＸＰａｔｈ式に共通するロケーションステップを１つのノードに対応させたデータ構造を生成する工程と

を含むことを特徴とする請求項１に記載のＸＰａｔｈ評価方法。
【請求項３】
前記データ構造を生成する工程では、
複数の前記ロケーションステップが共通の特定の形で記述されている場合に、当該ロケーションステップにおける述部の式の評価結果をキーとして、ＸＰａｔｈにおける残りのロケーションステップを検索するためのハッシュテーブルまたは二分探索木を生成し、
前記ノードごとの評価結果を結合して各ＸＰａｔｈ式の評価結果を得る工程では、
前記ハッシュテーブルまたは前記二分探索木を用いた検索により得られたＸＰａｔｈの前記残りのロケーションステップに関して、当該残りのロケーションステップで指定される前記特定部分を前記処理対象のデータファイルが持つか否かを評価し、その評価結果と前記ノードごとの評価結果とに基づいて各ＸＰａｔｈ式の評価結果を得ることを特徴とする請求項２に記載のＸＰａｔｈ評価方法。
【請求項４】
前記データ構造を生成する工程は、
前記複数のＸＰａｔｈ式が演算式または関数を含む場合に、当該演算式または関数を部分式に分解する工程と、
分解して得られた前記部分式のうち、共通する部分式を１つにまとめて有向非循環グラフを生成する工程とを含み、
前記ノードごとの評価結果に基づいて各ＸＰａｔｈ式の評価結果を得る工程では、
前記有向非循環グラフを用い、複数のＸＰａｔｈ式に共通する部分式については評価結果を当該複数のＸＰａｔｈ式で共用して各ＸＰａｔｈ式を評価し、その評価結果と前記ノードごとの評価結果とに基づいて各ＸＰａｔｈ式の評価結果を得ることを特徴とする請求項１に記載のＸＰａｔｈ評価方法。
【請求項５】
コンピュータを用いて、所定のデータファイルを対象として当該データファイルがＸＰａｔｈで指定される特定部分を持つか否かを評価するＸＰａｔｈ評価方法において、
メモリから評価を行うべき複数のＸＰａｔｈ式を含むＸＰａｔｈセットを読み出し、当該複数のＸＰａｔｈセットに共通して含まれるＸＰａｔｈ式と所定のＸＰａｔｈセットに固有のＸＰａｔｈ式とを分類し、当該ＸＰａｔｈ式の冗長な要素を省略したデータ構造を生成する工程と、
前記ＸＰａｔｈセットに固有のＸＰａｔｈ式について評価する工程と、
処理対象のデータファイルが前記固有のＸＰａｔｈ式で指定される特定部分を持つと評価される場合に、当該ＸＰａｔｈセット中の他のＸＰａｔｈセットと共通するＸＰａｔｈ式について評価する工程と
を含むことを特徴とするＸＰａｔｈ評価方法。
【請求項６】
前記データ構造を生成する工程では、前記ＸＰａｔｈセットに含まれる連結ＸＰａｔｈ式に関して、当該連結ＸＰａｔｈ式を構成する各ＸＰａｔｈ式の間の依存関係に基づき、評価が不要なＸＰａｔｈ式を評価対象から除くことにより、当該連結ＸＰａｔｈ式を簡略化することを特徴とする請求項５に記載のＸＰａｔｈ評価方法。
【請求項７】
前記データ構造を生成する工程では、複数のＸＰａｔｈ式の間の依存関係に基づき、評価結果が他のＸＰａｔｈ式に依存するＸＰａｔｈ式を評価対象から除くことを特徴とする請求項５に記載のＸＰａｔｈ評価方法。
【請求項８】
前記データ構造を生成する工程では、複数のＸＰａｔｈ式の間で共通する部分と各ＸＰａｔｈ式に固有の部分とを分割し、各部分を個別のＸＰａｔｈ式として評価対象とすることを特徴とする請求項５に記載のＸＰａｔｈ評価方法。
【請求項９】
複数のＸＰａｔｈ式を格納したＸＰａｔｈ格納部と、

前記ＸＰａｔｈ格納部に格納されている前記複数のＸＰａｔｈ式を読み出し、当該複数のＸＰａｔｈ式から、当該複数のＸＰａｔｈ式の共通部分を共通のノードとし、当該複数のＸＰａｔｈ式の他の部分を個別のノードとする木構造のデータ構造を生成するデータ構造生成部と、

処理対象のデータファイルを入力し、前記データ構造のノードごとに、当該ノードに対応する前記ＸＰａｔｈ式の部分で指定される特定部分を当該処理対象のデータファイルが持つか否かを評価し、当該ノードごとの評価結果として得られた当該特定部分を結合し、前記複数のＸＰａｔｈ式の各ＸＰａｔｈ式に関して、個々の当該ＸＰａｔｈ式を構成する部分に対応する前記ノードごとの評価結果に基づいて各ＸＰａｔｈ式の評価結果を得る評価実行部と

を備えることを特徴とする文書処理システム。

【請求項１０】

前記データ構造生成部は、

個々の前記ＸＰａｔｈ式をロケーションステップごとに分解するステップ分解手段と、

分解して得られた各ロケーションステップにノードを対応させ、複数のＸＰａｔｈ式に共通するロケーションステップを１つのノードに対応させたデータ構造を生成する木生成手段と

を備えることを特徴とする請求項９に記載の文書処理システム。

【請求項１１】

前記データ構造生成部は、前記ＸＰａｔｈ格納部に格納されているＸＰａｔｈ式が変更された場合に、新たに追加されたＸＰａｔｈ式のうちで前記データ構造に対応するノードがない部分について新たなノードを生成して当該データ構造に加え、削除されたＸＰａｔｈ式の部分に対応する前記データ構造のノードのうちで当該ＸＰａｔｈ式に固有の部分に対応するノードのみを除去することにより、当該データ構造を更新することを特徴とする請求項９記載の文書処理システム。

【請求項１２】

前記データ構造生成部は、他のＸＰａｔｈ式と重複するＸＰａｔｈ式または評価結果が他のＸＰａｔｈ式に依存するＸＰａｔｈ式を評価対象から除くことを特徴とする請求項９に記載の文書処理システム。

【請求項１３】

前記データ構造生成部にて生成されたデータ構造を格納し保存するデータ構造格納部をさらに備え、

前記評価実行部は、前記データ構造格納部に格納されている前記データ構造を用いて前記複数のＸＰａｔｈ式の評価を行うことを特徴とする請求項９に記載の文書処理システム。

【請求項１４】

前記評価実行部は、前記データ構造におけるノードごとに対応する部分の評価を行い、この部分的な評価結果を結合して個々のＸＰａｔｈ式全体の評価結果を得ると共に、複数のＸＰａｔｈ式間で共通するノードに対応する部分の評価結果を当該複数のＸＰａｔｈ式の評価において共用することを特徴とする請求項９に記載の文書処理システム。

【請求項１５】

複数のＸＰａｔｈ式にて構成されるＸＰａｔｈセットを格納したＸＰａｔｈ格納部と、

前記ＸＰａｔｈ格納部から評価を行うべき複数のＸＰａｔｈ式を含むＸＰａｔｈセットを読み出し、当該複数のＸＰａｔｈセットに共通して含まれるＸＰａｔｈ式と所定のＸＰａｔｈセットに固有のＸＰａｔｈ式とを分類し、当該ＸＰａｔｈ式の冗長な要素を省略したデータ構造を生成するデータ構造生成部と、

処理対象のデータファイルを入力し、前記ＸＰａｔｈセットに固有のＸＰａｔｈ式について当該データファイルが当該ＸＰａｔｈで指定される特定部分を持つか否かを評価し、当該処理対象のデータファイルが当該固有のＸＰａｔｈ式で指定される特定部分を持つと評価される場合に、当該ＸＰａｔｈセット中の他のＸＰａｔｈセットと共通するＸＰａｔ

ｈ式について当該データファイルが当該ＸＰａｔｈで指定される特定部分を持つか否かを評価し、得られた各ＸＰａｔｈ式の評価結果に基づいて前記データファイルに対するＸＰａｔｈセットの評価結果を得る評価実行部と
を備えることを特徴とする文書処理システム。
【請求項１６】
コンピュータを制御して、所定のデータファイルを対象として当該データファイルがＸＰａｔｈで指定される特定部分を持つか否かを評価するプログラムであって、
評価を行うべき複数のＸＰａｔｈ式から、当該複数のＸＰａｔｈ式の共通部分を共通のノードとし、当該複数のＸＰａｔｈ式の他の部分を個別のノードとする木構造のデータ構造を生成する処理と、
前記データ構造のノードごとに、当該ノードに対応する前記ＸＰａｔｈ式の部分で指定される前記特定部分を処理対象のデータファイルが持つか否かを評価し、評価結果を結合する処理と、
前記複数のＸＰａｔｈ式の各ＸＰａｔｈ式に関して、個々の当該ＸＰａｔｈ式を構成する部分に対応する前記ノードごとの評価結果に基づいて各ＸＰａｔｈ式の評価結果を得る処理とを
前記コンピュータに実行させることを特徴とするプログラム。
【請求項１７】
前記プログラムによる前記データ構造を生成する処理は、
個々の前記ＸＰａｔｈ式をロケーションステップごとに分解する処理と、
分解して得られた各ロケーションステップにノードを対応させ、複数のＸＰａｔｈ式に共通するロケーションステップを１つのノードに対応させたデータ構造を生成する処理とを含むことを特徴とする請求項１６に記載のプログラム。
【請求項１８】
前記プログラムによる前記データ構造を生成する処理では、
複数の前記ロケーションステップが共通の特定の形で記述されている場合に、当該ロケーションステップにおける述部の式の評価結果をキーとして、ＸＰａｔｈにおける残りのロケーションステップを検索するためのハッシュテーブルまたは二分探索木を生成し、
前記ノードごとの評価結果に基づいて各ＸＰａｔｈ式の評価結果を得る処理では、
前記ハッシュテーブルまたは前記二分探索木を用いた検索により得られたＸＰａｔｈの前記残りのロケーションステップに関して、当該残りのロケーションステップで指定される前記特定部分を前記処理対象のデータファイルが持つか否かを評価し、その評価結果と前記ノードごとの評価結果とに基づいて各ＸＰａｔｈ式の評価結果を得ることを特徴とする請求項１７に記載のプログラム。
【請求項１９】
前記プログラムによる前記データ構造を生成する処理は、
前記複数のＸＰａｔｈ式が演算式または関数を含む場合に、当該演算式または関数を部分式に分解する処理と、
分解して得られた前記部分式のうち、共通する部分式を１つにまとめて有向非循環グラフを生成する処理とを含み、
前記ノードごとの評価結果に基づいて各ＸＰａｔｈ式の評価結果を得る処理では、
前記有向非循環グラフを用い、複数のＸＰａｔｈ式に共通する部分式については評価結果を当該複数のＸＰａｔｈ式で共用して各ＸＰａｔｈ式を評価し、その評価結果と前記ノードごとの評価結果とに基づいて各ＸＰａｔｈ式の評価結果を得ることを特徴とする請求項１６に記載のプログラム。
【請求項２０】
前記プログラムは、評価を行うべき複数のＸＰａｔｈ式が変更された場合に、新たに追加されたＸＰａｔｈ式のうちで前記データ構造に対応するノードがない部分について新たなノードを生成して当該データ構造に加え、削除されたＸＰａｔｈ式の部分に対応する前記データ構造のノードのうちで当該ＸＰａｔｈ式に固有の部分に対応するノードのみを除

去することにより、当該データ構造を更新する処理を、前記コンピュータにさらに実行させることを特徴とする請求項１６に記載のプログラム。
【請求項２１】
コンピュータを制御して、所定のデータファイルを対象として当該データファイルがＸＰａｔｈで指定される特定部分を持つか否かを評価するプログラムであって、
複数のＸＰａｔｈ式を含んで構成された評価を行うべきＸＰａｔｈセットから当該複数のＸＰａｔｈセットに共通して含まれるＸＰａｔｈ式と所定のＸＰａｔｈセットに固有のＸＰａｔｈ式とを分類し、当該ＸＰａｔｈ式の冗長な要素を省略したデータ構造を生成する処理と、
前記ＸＰａｔｈセットに固有のＸＰａｔｈ式について評価する処理と、
処理対象のデータファイルが前記固有のＸＰａｔｈ式で指定される特定部分を持つと評価される場合に、当該ＸＰａｔｈセット中の他のＸＰａｔｈセットと共通するＸＰａｔｈ式について評価する処理と
を前記コンピュータに実行させることを特徴とするプログラム。
【請求項２２】
前記プログラムによる前記データ構造を生成する処理では、前記ＸＰａｔｈセットに含まれる連結ＸＰａｔｈ式に関して、当該連結ＸＰａｔｈ式を構成する各ＸＰａｔｈ式の間の依存関係に基づき、評価が不要なＸＰａｔｈ式を評価対象から除くことにより、当該連結ＸＰａｔｈ式を簡略化することを特徴とする請求項２１に記載のプログラム。
【請求項２３】
前記プログラムによる前記データ構造を生成する処理では、複数のＸＰａｔｈ式の間の依存関係に基づき、評価結果が他のＸＰａｔｈ式に依存するＸＰａｔｈ式を評価対象から除くことを特徴とする請求項２１に記載のプログラム。
【請求項２４】
前記プログラムによる前記データ構造を生成する処理では、複数のＸＰａｔｈ式の間で共通する部分と各ＸＰａｔｈ式に固有の部分とを分割し、各部分を個別のＸＰａｔｈ式として評価対象とすることを特徴とする請求項２１に記載のプログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、ＸＰａｔｈ（XML Path Language）を用いてＸＭＬ文書やＨＴＭＬ文書における特定部分を指し示す場合に、ＸＰａｔｈ式を効率よく評価する技術に関する。
【０００２】
【従来の技術】
ＸＭＬ文書の特定部分を指し示すためのパス言語として、Ｗ３Ｃ（World Wide Web Consortium）から勧告として公開されているＸＰａｔｈがある。ＸＰａｔｈは、例えば、ＸＰｏｉｎｔｅｒ、ＸＳＬＴ、ＸＱｕｅｒｙなどの構成要素として使用され、また所定のアプリケーション・プログラムの中でＸＭＬ文書のＤＯＭ（Document Object Model）ツリーをアクセスするのにも使用される。
【０００３】
ところで、ＸＭＬ文書を用いた実際の情報処理においては、１つのＸＭＬ文書に対して複数のＸＰａｔｈ式を評価することがよく行われる。例えば、ＸＳＬＴスタイルシートでは、テンプレート・ルールごとにＸＰａｔｈ式がパターンとして指定される。したがって、複雑なＸＳＬＴスタイルシートには多くのＸＰａｔｈ式が含まれ、処理対象であるＸＭＬ文書に対して、かかる複数のＸＰａｔｈ式を評価することが必要となる。
【０００４】
また、インターネット上でのデータ交換等に用いられるウェブページに対してアノテーションを付加することにより、所定のウェブページを様々な形に再利用することが可能であり、新たなアプリケーションを開発可能であることが広く知られている。ウェブページの記述に用いられるＨＴＭＬ文書は、ＸＭＬ文書と同様にＸＰａｔｈ式にて所定の部分を指

し示すことができるため、アノテーションをウェブページ中のエレメントに対応付けるためにもＸＰａｔｈが用いられる。ところで、ウェブページに対するアノテーションの付加を効率的に行うために、特定のアノテーションを多数のウェブページに対して適用し、使い回すことが考えられる。この場合、所定のアノテーションが所定のウェブページに対して適用可能かどうかを判断するために、アノテーション内の複数のＸＰａｔｈ式が対象のウェブページにおける所定のエレメントを正しく指しているかどうかを評価することが必要となる。

【０００５】

ＸＰａｔｈによるＸＭＬ文書の特定部分を指し示す機能は、対象となるＸＭＬ文書がＸＰａｔｈにて指定される特定の部分を持つかどうかをチェックする条件とみなすことができる。例えば、米国ＢＥＡ社のサーバシステムであるWebLogic Collaborate（http://www.bea.com/index.html）は、ＸＭＬで表現されるメッセージのルーティングとフィルタリングの条件を記述するためにＸＰａｔｈを用いる。このような用途では、１つのＸＭＬ文書が到着するたび多数のＸＰａｔｈ式を評価することとなる。

【０００６】

このように、１つのＸＭＬ文書に対して複数のＸＰａｔｈ式を評価する必要がある場合、ＸＰａｔｈ式の評価を効率的に行うことが要求される。従来、この種の技術としては、ＸＭＬで記述された文書に対する各利用者の購読条件をＸＰａｔｈで記述しておき、ＸＭＬ文書が到着するたびにＸＰａｔｈ式とのマッチングを検査して、検査を通過する文書を、当該ＸＰａｔｈを購読条件とする利用者に配信するものがある（例えば、非特許文献１参照。）。このＸＰａｔｈ式の評価方法は、ロケーション・パスの各ステップの評価をテーブルの検索を使って行うことにより、ＸＰａｔｈ式１つあたりの実行時間を改善している。

【０００７】

【非特許文献１】

Altinel M., Franklin, M., "Efficient Filtering of XML Documents for Selective Dissemination of Information", International Conference on Very Large Data Bases, 2000.

【０００８】

【発明が解決しようとする課題】

上述したように、ＸＭＬ文書やＨＴＭＬ文書に対する処理において、１つのデータファイル（文書）に対して複数のＸＰａｔｈ式を評価する場合、効率的にＸＰａｔｈ式の評価を行うことが要求される。

しかし、上記文献に開示された方法のように、ＸＰａｔｈ式１つあたりの評価の実行時間を改善する方法は知られているが、ＸＰａｔｈ式の数の増加に比例して、全体の評価に要する時間が直線的に増加してしまうため、全体的な実行時間の短縮には限界があった。

これは、従来のＸＰａｔｈの評価方法が、複数のＸＰａｔｈ式を評価する場合に、各ＸＰａｔｈ式を互いに独立なものとして扱っていることに起因する。

【０００９】

ところが、１つのデータファイルに対して評価されることが想定されているＸＰａｔｈ式が複数ある場合、それらのＸＰａｔｈ式は対象となるデータファイルの構造や要素の値のバリエーションなどから制約を受けるので、複数のＸＰａｔｈ式の中に類似なものが含まれる。したがって、類似するＸＰａｔｈ式から共通の部分を取り出して評価し、類似するＸＰａｔｈ式の間で共通部分の評価結果を共用することにより、複数のＸＰａｔｈ式を個別に評価するよりも高速に実行できると考えられる。

【００１０】

また、評価されるべき複数のＸＰａｔｈ式の間に依存関係がある場合、この依存関係を利用して、ＸＰａｔｈ式の評価に要する処理を簡単化することができると考えられる。ここで、ＸＰａｔｈ式の間の依存関係とは、例えば、所定のＸＰａｔｈ式が指し示すウェブページ中に２つのテーブルコンテンツ（テーブル[１]、テーブル[２]）がある場合、テーブ

ル[１]が存在しなければテーブル[２]は存在し得ないというような関係、すなわち、複数のＸＰａｔｈの１つまたはその一部を評価することにより、他のＸＰａｔｈの評価結果が自明となる関係である。

【００１１】

そこで、本発明は、ＸＭＬ文書やＨＴＭＬ文書等のデータファイルに対して複数のＸＰａｔｈ式を評価する場合に、かかるＸＰａｔｈ式の間の類似性や依存関係を利用して、効率よく複数のＸＰａｔｈ式の評価結果を求める方法を提供することを目的とする。

【００１２】

【課題を解決するための手段】

上記の目的を達成する本発明は、コンピュータを用いて所定のデータファイル（ＸＭＬ文書やＨＴＭＬ文書）を対象として複数のＸＰａｔｈを評価する、次のようなＸＰａｔｈ評価方法として実現される。すなわち、このＸＰａｔｈ評価方法は、評価を行うべき複数のＸＰａｔｈ式における共通部分を木構造における共通のノードとして持つデータ構造を生成するステップと、処理対象のデータファイルに対し、生成されたデータ構造のノードごとに評価を行うステップと、このノードごとの評価結果を結合して個々のＸＰａｔｈ式の評価結果を得るステップとを含むことを特徴とする。

【００１３】

ここで、このデータ構造を生成するステップは、詳しくは、個々のＸＰａｔｈ式をステップごとに分解するステップと、分解して得られた各ステップにノードを対応させ、複数のＸＰａｔｈ式に共通するステップを１つのノードに対応させたデータ構造を生成するステップとを含む。

さらに好ましくは、このデータ構造を生成するステップは、ＸＰａｔｈ式の複数のステップが共通の特定の形で記述されている場合、具体的には、「[(式) (比較演算子) (定数)]」の形になる述部が存在する場合に、特殊なデータ構造を用いることができる。例えば、比較演算子が「＝」や「！＝」の場合、ステップにおける述部の式の評価結果をキーとしてＸＰａｔｈにおける残りのステップを検索するためのハッシュテーブルを生成し、生成されたハッシュテーブルを用いた検索により得られたＸＰａｔｈの残りのステップを評価することができる。また、比較演算子が「＞」や「＞＝」、「＜」、「＜＝」の場合、ステップにおける述部の式の評価結果をキーとしてＸＰａｔｈにおける残りのステップを検索するための二分探索木を生成し、生成された二分探索木を用いた検索により得られたＸＰａｔｈの残りのステップを評価することができる。

【００１４】

また、このデータ構造を生成するステップは、複数のＸＰａｔｈ式が演算式または関数を含む場合に、この演算式または関数を部分式に分解するステップと、分解して得られた部分式のうち、共通する部分式を１つにまとめて有向非循環グラフを生成するステップとを含み、ＸＰａｔｈ式の評価を行うステップは、生成された有向非循環グラフを用い、複数のＸＰａｔｈ式に共通する部分式については評価結果を複数のＸＰａｔｈ式で共用して、各ＸＰａｔｈ式を評価するステップを含む。

このとき、個々の部分式については、上述したように共通部分を木構造における共通ノードとして持つデータ構造を生成することができ、これを用いた評価を行うことができる。

【００１５】

本発明による他のＸＰａｔｈ評価方法は、評価を行うべき複数のＸＰａｔｈ式の中から共通部分を抽出するステップと、処理対象であるデータファイルに対して、ＸＰａｔｈ式における共通部分と他の部分とを分けて評価し、評価結果を結合して個々のＸＰａｔｈ式に対する評価結果を得るステップとを含み、このＸＰａｔｈ式の評価を得るステップは、所定のＸＰａｔｈ式を評価する際、共通部分に関して、他のＸＰａｔｈを評価した際に得られた評価結果がある場合に、この評価結果を当該所定のＸＰａｔｈ式における共通部分の評価結果として用いるステップを含むことを特徴とする。

【００１６】

本発明による、さらに他のＸＰａｔｈ評価方法は、コンピュータを用いて、所定のデータ

ファイルを対象として複数のＸＰａｔｈを評価するＸＰａｔｈ評価方法において、メモリから評価を行うべき複数のＸＰａｔｈ式を含むＸＰａｔｈセットを読み出し、このＸＰａｔｈセットに含まれるＸＰａｔｈ式の冗長な要素を省略したデータ構造を生成するステップと、処理対象のデータファイルを入力し、当該データファイルに対し、このデータ構造を用いてＸＰａｔｈセットの評価を行い、得られた評価結果をメモリに格納するステップとを含むことを特徴とする。

【００１７】

より詳しくは、このデータ構造を生成するステップでは、複数のＸＰａｔｈセットに共通して含まれるＸＰａｔｈ式と所定のＸＰａｔｈセットに固有のＸＰａｔｈ式とを分類し、ＸＰａｔｈセットの評価を行う際の評価対象となるＸＰａｔｈ式を絞り込む。また、ＸＰａｔｈセットに含まれる連結ＸＰａｔｈ式に関して、この連結ＸＰａｔｈ式を構成する各ＸＰａｔｈ式の間の依存関係に基づき、評価が不要なＸＰａｔｈ式を評価対象から除くことにより、当該連結ＸＰａｔｈ式を簡略化する。さらに、複数のＸＰａｔｈ式の間の依存関係に基づき、評価が不要なＸＰａｔｈ式を評価対象から除く。さらにまた、複数のＸＰａｔｈ式の間で共通する部分と各ＸＰａｔｈ式に固有の部分とを分割し、各部分を個別のＸＰａｔｈ式として評価対象とする。

また、ＸＰａｔｈセットの評価は、まず、各ＸＰａｔｈセットに固有のＸＰａｔｈ式を評価し、固有のＸＰａｔｈ式が全て処理対象のデータファイルにマッチする場合に、このＸＰａｔｈセット中の他のＸＰａｔｈセットと共通するＸＰａｔｈ式を評価する。

【００１８】

また、上記の目的を達成する本発明は、次のように構成された文書処理システムとしても実現される。すなわち、この文書処理システムは、複数のＸＰａｔｈ式を格納したＸＰａｔｈ格納部と、このＸＰａｔｈ格納部に格納されている複数のＸＰａｔｈ式の冗長な要素を省略したデータ構造を生成するデータ構造生成部と、このデータ構造生成部にて生成されたデータ構造を用いて、処理対象であるデータファイルに対し、複数のＸＰａｔｈ式の評価を行う評価実行部とを備えることを特徴とする。

さらに、この文書処理システムは、データ構造生成部にて生成されたデータ構造を格納し保存するデータ構造格納部を備える構成とすることができる。この場合、評価実行部は、このデータ構造格納部に格納されているデータ構造を繰り返し用いて複数のデータファイルに対してＸＰａｔｈ式の評価を行うことができる。

【００１９】

ここで、このデータ構造生成部は、詳しくは、個々のＸＰａｔｈ式をステップごとに分解するステップ分解手段と、分解して得られた各ステップにノードを対応させ、複数のＸＰａｔｈ式に共通するステップを１つのノードに対応させたデータ構造を生成する木生成手段とを備える。

また、このデータ構造生成部は、ＸＰａｔｈ格納部に格納されているＸＰａｔｈ式が変更された場合に、新たに追加されたＸＰａｔｈ式のうちでデータ構造に対応するノードがない部分について新たなノードを生成して既存のデータ構造に加える。一方、削除されたＸＰａｔｈ式の部分に対応するノードのうちでＸＰａｔｈ式に固有の部分に対応するノードのみを除去することにより、データ構造を更新する。

さらに、このデータ構造生成部は、他のＸＰａｔｈ式と重複するＸＰａｔｈ式または評価結果が他のＸＰａｔｈ式に依存するＸＰａｔｈ式を評価対象から除く。

さらにまた、この評価実行部は、データ構造におけるノードごとに対応する部分の評価を行い、この部分的な評価結果を結合して個々のＸＰａｔｈ式全体の評価結果を得ると共に、複数のＸＰａｔｈ式間で共通するノードに対応する部分の評価結果を、複数のＸＰａｔｈ式の評価において共用する。

【００２０】

また、上記の目的を達成する本発明は、上述したＸＰａｔｈ評価方法の各ステップに対応する処理をコンピュータに実行させ、またはコンピュータを制御して上述した文書処理システムとしての各機能を実現するプログラムとしても実現することができる。このプログ

ラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記録媒体に格納して配布したり、ネットワークを介して配信したりすることにより提供することができる。

【００２１】

【発明の実施の形態】

以下、添付図面に示す実施の形態に基づいて、この発明を詳細に説明する。

なお、本実施の形態では、ＸＭＬ文書やウェブページの記述に用いられるＨＴＭＬ文書など、ＸＰａｔｈ式で所定の部分を指し示すことができるデータファイルを対象とする。したがって、以下の説明においてＸＭＬ文書を対象として記載している部分をＨＴＭＬ文書に対して適用することも、反対にＨＴＭＬ文書を対象として記載している部分をＸＭＬ文書に対して適用することも可能である。

〔第１の実施の形態〕

図１は、第１の実施の形態によるＸＭＬ文書処理システムを実現するのに好適なコンピュータ装置のハードウェア構成の例を模式的に示した図である。

図１に示すコンピュータ装置は、演算手段であるＣＰＵ（Central Processing Unit：中央処理装置）１０１と、Ｍ／Ｂ（マザーボード）チップセット１０２及びＣＰＵバスを介してＣＰＵ１０１に接続されたメインメモリ１０３と、同じくＭ／Ｂチップセット１０２及びＡＧＰ（Accelerated Graphics Port）を介してＣＰＵ１０１に接続されたビデオカード１０４と、ビデオカード１０４にて生成されたグラフィックデータを表示する表示装置１１０と、ＰＣＩ（Peripheral Component Interconnect）バスを介してＭ／Ｂチップセット１０２に接続されたハードディスク１０５及びネットワークインターフェイス１０６と、さらにこのＰＣＩバスからブリッジ回路１０７及びＩＳＡ（Industry Standard Architecture）バスなどの低速なバスを介してＭ／Ｂチップセット１０２に接続されたフロッピーディスクドライブ１０８及びキーボード／マウス１０９とを備える。表示装置１１０としては、例えば液晶ディスプレイ（ＬＣＤ）やＣＲＴディスプレイを用いることができる。

なお、図１は本実施の形態を実現するコンピュータ装置のハードウェア構成を例示するに過ぎず、本実施の形態を適用可能であれば、他の種々の構成を取ることができる。例えば、ビデオカード１０４を設ける代わりに、ビデオメモリのみを搭載し、ＣＰＵ１０１にてイメージデータを処理する構成としても良いし、音声による入出力を行うためのサウンド機構を設けたり、ＡＴＡ（AT Attachment）などのインターフェイスを介してＣＤ－ＲＯＭ（Compact Disc Read Only Memory）やＤＶＤ－ＲＯＭ（Digital Versatile Disc Read Only Memory）のドライブを設けたりしても良い。

【００２２】

図２は、第１の実施の形態によるＸＭＬ文書処理システムの機能を説明するブロック図である。

図２を参照すると、本実施の形態は、処理対象であるＸＭＬ文書の特定部分を指し示す複数のＸＰａｔｈ式を格納したＸＰａｔｈ格納部１０と、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈ式から本実施の形態による効率的な評価を行うためのデータ構造を生成するデータ構造生成部２０と、データ構造生成部２０にて生成されたデータ構造を格納するデータ構造格納部３０と、データ構造格納部３０に格納されているデータ構造を用いて処理対象であるＸＭＬ文書に対してＸＰａｔｈ式の評価を行う評価実行部４０とを備え、また処理対象であるＸＭＬ文書を格納するＸＭＬ文書格納部５０と、評価実行部４０による評価を経たＸＭＬ文書を対象として所定の情報処理を実行するＸＭＬ文書処理部６０とを備える。

【００２３】

図２に示したシステム構成において、データ構造生成部２０、評価実行部４０及びＸＭＬ文書処理部６０は、図１に示したプログラム制御されたＣＰＵ１０１にて実現される仮想的なソフトウェアブロックである。ＣＰＵ１０１を制御してこれらの機能を実現するプログラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記録媒体に格納して配布したり、ネットワークを介して配信したりすることにより提供され、メインメモリ１０３

に読み込まれる。また、ＸＰａｔｈ格納部１０、データ構造格納部３０及びＸＭＬ文書格納部５０は、メインメモリ１０３にて実現される。なお、メインメモリ１０３に保持されるデータやプログラムは、必要に応じてハードディスク１０５などの記憶装置に退避させることができる。

【００２４】

図２に示したシステム構成において、ＸＰａｔｈ格納部１０には、ＸＭＬ文書に対して評価を行うべき、予め用意された複数のＸＰａｔｈ式が格納されている。データ構造格納部３０には、データ構造生成部２０にて生成され、評価実行部４０にて用いられるデータ構造が格納される。ＸＭＬ文書格納部５０には、図１に示したネットワークインターフェイス１０６や、キーボード／マウス１０９等の入力デバイス、処理対象であるＸＭＬ文書を格納した記録媒体から当該ＸＭＬ文書を読み出すドライブ装置などを介して入力されたＸＭＬ文書が格納されている。

【００２５】

データ構造生成部２０は、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈ式から、評価実行部４０による評価に用いるデータ構造を生成する。また、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈが変更（追加または削除）されたならば、この変更に応じてデータ構造を更新する。このデータ構造を用いることにより、ＸＭＬ文書に対してＸＰａｔｈ格納部１０に格納されている複数のＸＰａｔｈ式を個別に評価する場合に比べて効率の良い評価を行うことが可能となる。このデータ構造についての詳細、このデータ構造を用いた評価方法及びこのデータ構造の生成方法については後述する。

【００２６】

評価実行部４０は、ＸＭＬ文書格納部５０に格納されているＸＭＬ文書に対して、データ構造生成部２０にて生成されたデータ構造を用いてＸＰａｔｈ式の評価（マッチング）を行う。これにより、当該入力文書がＸＰａｔｈ格納部１０に格納されているＸＰａｔｈ式にて指定される特定部分を持つかどうかを、効率よく評価することができる。評価実行部４０による評価結果はメインメモリ１０３の所定の領域に格納される。

ＸＭＬ文書処理部６０は、メインメモリ１０３の所定領域から評価実行部４０の評価結果を読み出し、当該評価結果に基づいて、ＸＭＬ文書格納部５０に格納されているＸＭＬ文書がＸＰａｔｈ格納部１０に格納されているＸＰａｔｈ式にて指定される特定部分を持つならば、ＸＭＬ文書格納部５０から当該ＸＭＬ文書を処理対象として読み出し、目的とする所定の情報処理、すなわちアプリケーションとして用意された情報処理を実行する。

【００２７】

次に、本実施の形態におけるＸＰａｔｈ式の評価を効率化する方法を詳細に説明する。

本実施の形態では、複数のＸＰａｔｈ式における共通部分を共用するためのデータ構造を生成し、このデータ構造を用いてＸＭＬ文書に対する評価を行うことにより、評価処理の高速化を実現する。これを実現する方策として、本実施の形態では、次の３つの手法を提示する。

方法１．複数のロケーション・パスの間でのステップの評価の共有。

方法２．複数の演算式や関数の間での部分式や引数の評価の共有。

方法３．ロケーション・パスの述部の高速化。

【００２８】

以下、各手法について説明する。

方法１．複数のロケーション・パスの間でのステップの評価の共有

ＸＰａｔｈ格納部１０に、次の３つのＸＰａｔｈ式Ｐ１、Ｐ２、Ｐ３が格納されており、ＸＭＬ文書に対してこれらのＸＰａｔｈ式を評価する場合を例として説明する。

Ｐ１：/profile/demographics/age[text() < 20]

Ｐ２：/profile/interests/sport[text() = 'Soccer']

Ｐ３：/profile/demographics/age[text() >= 40 and text() < 50]

【００２９】

まず、データ構造生成部２０により、上記ＸＰａｔｈ式Ｐ１、Ｐ２、Ｐ３をまとめたデー

タ構造が生成される。
データ構造生成部２０は、このデータ構造を生成するため、上記のＸＰａｔｈ式をステップごとに分解する。ステップが軸指定子（axis）とノードテストの組と述部からなるときには、さらにこれらも分解する。すなわち、ＸＰａｔｈ式Ｐ１は次のＰ１’に、Ｐ２はＰ２’に、Ｐ３はＰ３’に、それぞれ分解される。
Ｐ１’：profile + demographics + age + [text() < 20]
Ｐ２’：profile + interests + sport + [text() = 'Soccer']
Ｐ３’：profile + demographics + age + [text() >= 40 and text() < 50]
Ｐ１’、Ｐ２’、Ｐ３’を比較すると、これら３つのＸＰａｔｈ式は「profile」を共通ステップとして持つ。また、Ｐ１’とＰ３’とは、さらに続くステップの「demographics」を共通に持つ。そこで、共通のステップをまとめることによって、これらのＸＰａｔｈ式を木構造で表現することができる。
すなわち、かかる処理においてデータ構造生成部２０は、ＸＰａｔｈ式をステップごとに分解するステップ分解手段と、各ステップに対応するノードを持つ木構造（データ構造）を生成する木生成手段として機能する。
図３は、データ構造生成部２０にて生成されるデータ構造（木構造）を示す図である。
【００３０】
次に、評価実行部４０により、上記のように生成された木データ構造を用いて、ＸＭＬ文書に対するＸＰａｔｈ式Ｐ１、Ｐ２、Ｐ３の評価が行われる。
ここでは、図４に示すＸＭＬ文書に対し、ＸＰａｔｈ式Ｐ１、Ｐ２、Ｐ３を評価する例を考える。
（１）ＸＰａｔｈ式Ｐ１の評価
最初に図４に示したＸＭＬ文書Ｄ１に対し、ＸＰａｔｈ式Ｐ１の先頭のステップ「profile」を評価する。評価結果は、次の４要素を持つノードセットになる。
Ｎ１：
{<name>Alan</name>,
<demographics><age>35</age></demographics>,
<location><city>Osaka</city></location>,
<interests><sport>Soccer</sport><music>Classical</music><book>History</book></interests>}
【００３１】
ノードセットＮ１の４つの要素それぞれに、ＸＰａｔｈ式Ｐ１の次のステップ「demographics」を適用すると、結果は
{} , {<age>35</age>} , {} , {}
となる。この４つのノードセットを結合して、このステップのノードセットを作ると、次のようになる。
Ｎ２：
{<age>35</age>}
このノードセットＮ２の要素に「age」を適用すると、次のようになる。
Ｎ３：
{35}
さらに、このノードセットＮ３に「[text() < 20]」適用すると、結果は空のノードセットになり、これがＸＰａｔｈ式Ｐ１の評価結果として得られる。
【００３２】
（２）ＸＰａｔｈ式Ｐ２の評価
次に、ＸＰａｔｈ式Ｐ２を評価する。このとき、ＸＭＬ文書Ｄ１に対するＸＰａｔｈ式Ｐ２の先頭のステップ「profile」の評価結果は、すでにノードセットＮ１として得られている。したがって、この評価のための処理は実際には行わず、すでに得られている評価結果を使用する。ノードセットＮ１の各要素に「interests」を適用すると、結果はそれぞれ

{} , {} , {} , {<sport>Soccer</sport>, <music>Classical</music>, <book>History</book>}
となる。この４つのノードセットを結合して、このステップのノードセットを作ると、次のようになる。
Ｎ４：
{<sport>Soccer</sport>, <music>Classical</music>, <book>History</book>}
【００３３】
さらに、このノードセットＮ４の３つの要素に対して、ＸＰａｔｈ式Ｐ２の残りの表現である「sport」を適用し、その結果に「[text() = 'Soccer']」を適用すると、結果は
{Soccer} , {} , {}
となる。これを結合すると
{Soccer}
となり、これがＸＰａｔｈ式Ｐ２の評価結果として得られる。
【００３４】
（３）ＸＰａｔｈ式Ｐ３の評価
最後にＸＰａｔｈ式Ｐ３を評価する。このとき、ＸＭＬ文書Ｄ１に対するＸＰａｔｈ式Ｐ３のロケーション・パスのうち、「/profile/demographics/age」の部分の評価値は、すでにノードセットＮ３として得られている。したがって、この部分に対する評価のための処理は実際には行わず、すでに得られている評価結果を使用する。ノードセットＮ３の各要素に対して、ＸＰａｔｈ式Ｐ３の最後の述部「[text() >= 40 and text() < 50]」を適用すると、結果は空のノードセットになり、これがＸＰａｔｈ式Ｐ３の評価結果として得られる。
【００３５】
以上のようにして、図３に示したデータ構造を用い、複数のＸＰａｔｈ式に共通するステップについては１度だけ実際の計算を行い、評価結果を共有することによって、複数のＸＰａｔｈ式の計算時間を短縮することができる。
上記の例では、任意の順番でＸＰａｔｈ式を評価するように記述したが、図３のデータ構造から導かれる深さ優先または幅優先の順番で評価することにより、評価を行う際に必要なメモリの記憶領域を減らすことができる。例えば、深さ優先で評価した場合、所定のノードを保持するために必要な記憶領域は、当該ノードに対して全ての部分木の評価が終わったならば回収できる。一方、幅優先で評価した場合、上位ノードの評価結果を保持するために必要な記憶領域は、当該評価結果が下位ノードの評価に使用された後に回収できる。
【００３６】
図５は、幅優先で複数のＸＰａｔｈ式を評価する場合の評価実行部４０による処理手順を説明するフローチャートである。
図５を参照すると、評価実行部４０は、まず、ＸＭＬ文書全体を１つの要素とする集合を、現在のノードセットにする（ステップ５０１）。そして、全てのロケーション・パスの先頭ステップを、ステップ集合の要素にする（ステップ５０２）。次に、全てのステップ集合の要素に対し、次の処理を行う。すなわち、まず、現在のノードセットの各要素に対して、ステップをそれぞれ評価する（ステップ５０４）。そして、全ての評価結果を結合する。これを現在のノードセットとする（ステップ５０５）。
【００３７】
ステップ５０４、５０５の処理を、全てのステップ集合の要素に対して行ったならば、次に、全てのステップ集合の要素に対して、次のステップの集合を求める。そして、評価結果を結合し、現在のステップ集合と置き換える（ステップ５０３、５０６）。
この後、ステップ集合に要素が残っていればステップ５０３～５０６の処理を繰り返し、要素が空になったならば処理を終了する。
すなわち、かかる処理で評価実行部４０は、各ＸＰａｔｈ式の評価において、データ構造のノードに対応するステップごとに評価を行う部分評価手段と、ステップごとの評価結果

を結合してＸＰａｔｈ式全体の評価結果を得る評価結果結合手段として機能する。そして、他のＸＰａｔｈ式の共通するステップの評価において、当該ステップに関して他のＸＰａｔｈ式を評価した際の評価結果が存在するならば、この評価結果を着目中のＸＰａｔｈ式の該当ステップの評価結果として用いることにより、当該ステップに対する評価処理を省略する。

【００３８】

この方法の基礎になるＸＰａｔｈの性質は、ロケーション・パスの途中までの表現を評価して得たノードセットに対して、そのノードセットの各要素をコンテクストノードとしてパスの残りを評価し、最後に結果を結合すると、パス全体を評価したものと同じものが得られることである。すなわち、上記の手順では、各ＸＰａｔｈ式のロケーション・パスのうちで共通部分を抽出し、当該共通部分と他の部分とを分けて評価した上で評価結果を結合する。この過程で、評価結果を共有できる部分（ロケーション・パスの共通部分）に対する評価結果が既に存在するならば、かかる部分に対する評価処理を省略して、その既存の評価結果を使用する。したがって、この性質を満たすようにパスが分解できる場合には、この方法が適用できることとなる。

上記の説明では、軸指定子（axis）を省略してデフォルトの「child」が選択される場合を例として用いたが、軸指定子が「decsendant::age」などのように「child」以外を明示的に書いてある場合も方法１で扱うことができる。また、ステップの区切りとして「/」だけでなく「//」の場合も扱うことができる。さらに、ノードテストとして要素名だけでなく、「.」「..」「*」「comment()」などを記述することもできる。例えば、次に示すような表現で記述されたＸＰａｔｈ式などが、方法１で扱うことができる。

```
/*//interest/../demographics/decsendant::age[text() < 20]
```

【００３９】

方法２．複数の演算式や関数の間での部分式や引数の評価の共有

上記の方法１では、ロケーション・パスの評価を高速化する手法を説明した。

ＸＰａｔｈにおいては、ロケーション・パス式が主要な式であるが、パスの述部の条件を記述したり、他のプログラムが使うためのデータを生成したりするために、四則演算式、比較演算式、論理演算式などの演算式や関数も使用される。そこで、以下に説明するロケーション・パス以外の式を高速化する手法を用いる。

【００４０】

ＸＰａｔｈ格納部１０に、次の３つのＸＰａｔｈ式Ｐ７、Ｐ８、Ｐ９が格納されており、ＸＭＬ文書に対してこれらのＸＰａｔｈ式を評価する場合を例として説明する。

Ｐ７：

```
(/CPEXMLv1/person/partyActivities/food[favoriteFood='Hamburger'] or
(not(/CPEXMLv1/person/partyActivities/hobby[typeName='SPORT']) and
  /CPEXMLv1/person/personDemographics/birthDate[year<1960])) and
/CPEXMLv1/person/partyActivities/hobby/startDate[year=1986]
```

Ｐ８：

```
(/CPEXMLv1/person/partyDemographics[gender ='MALE'] or
(not(/CPEXMLv1/person/partyActivities/hobby[typeName='SPORT']) and
  /CPEXMLv1/person/personDemographics/birthDate[year<1960])) and
/CPEXMLv1/person/partyActivities/hobby/startDate[year>1990]
```

Ｐ９：

```
/CPEXMLv1/person/partyActivities/food[favoriteFood ='Hamburger'] and
/CPEXMLv1/person/partyActivities/hobby/startDate[year>1990]
```

【００４１】

データ構造生成部２０は、まず、上記のＸＰａｔｈ式Ｐ７、Ｐ８、Ｐ９をロケーション・パスからなる部分式に分解する。すなわち、ＸＰａｔｈ式Ｐ７はＰ７１～Ｐ７４に、Ｐ８はＰ８１～Ｐ８４に、Ｐ９はＰ９１、Ｐ９２に、それぞれ分解される。

Ｐ７１：/CPEXMLv1/person/partyActivities/food[favoriteFood ='Hamburger']

Ｐ７２：/CPEXMLv1/person/partyActivities/hobby[typeName='SPORT']
Ｐ７３：/CPEXMLv1/person/personDemographics/birthDate[year<1960]
Ｐ７４：/CPEXMLv1/person/partyActivities/hobby/startDate[year=1986]
Ｐ８１：/CPEXMLv1/person/partyDemographics[gender ='MALE']
Ｐ８２：/CPEXMLv1/person/partyActivities/hobby[typeName='SPORT']
Ｐ８３：/CPEXMLv1/person/personDemographics/birthDate[year<1960]
Ｐ８４：/CPEXMLv1/person/partyActivities/hobby/startDate[year>1990]
Ｐ９１：/CPEXMLv1/person/partyActivities/food[favoriteFood ='Hamburger']
Ｐ９２：/CPEXMLv1/person/partyActivities/hobby/startDate[year>1990]

【００４２】

上記の部分式のうちで、Ｐ７１とＰ９１、Ｐ７２とＰ８２、Ｐ７３とＰ８３、Ｐ８４とＰ９２は、それぞれ同じ式である。したがって、ＸＰａｔｈ式Ｐ７、Ｐ８、Ｐ９は、有向非循環グラフ（DAG）にて表現することができる。

図６は、ＸＰａｔｈ式Ｐ７、Ｐ８、Ｐ９を表す有向非循環グラフを示す図である。

図６における左側の６個のロケーション・パスについては、上述した方法１によるステップの評価を共有する手法を用いることができるので、この部分を図３に示したような木構造のデータ構造に変換する。生成された有向非循環グラフのデータ構造は、方法１で生成された木構造のデータ構造と共にデータ構造格納部３０に格納される。

【００４３】

評価実行部４０は、以上のようにして得られたデータ構造を用いて、ＸＰａｔｈ式Ｐ７、Ｐ８、Ｐ９を評価する。まず、ロケーション・パス部分を、方法１を用いて評価する。次に、図６の有向非循環グラフを使って各ＸＰａｔｈ式を評価する。このとき、共通の部分式を持つものについては、当該部分式を１度だけ評価し、共通箇所に対して当該評価結果を再使用するようにする。

上記の例では、引数がノードセットで、評価した値が真理値になる論理演算式を例として取り上げたが、四則演算式など他のデータ型の演算式でも同様の最適化が可能である。

【００４４】

方法３．ロケーション・パスの述部の高速化

上述した方法１では、述部が異なるステップは独立のステップとして処理を行うものとして説明した。しかしながら、類似する述部の共通性を利用して述部の実行を高速化することが可能である。

例えば「[age < 20]」などのように、「[(式) (比較演算子) (定数)]」の形になる述部が高速化される。表面的にはこの形になっていなくても、置き換えや部分計算によってこの形に変形できればこの手法が適用できる。例えば、ノードの位置による指定「[2]」は、「[position() = 2]」に置き換えられるので、この手法が適用できる。

【００４５】

ここでは、（１）比較演算子が「＝」の場合、（２）比較演算子が「！＝」の場合、（３）比較演算子が「＞」の場合について、具体的について説明する。

（１）比較演算子が「＝」の場合の高速化

ステップにおける述部の式の評価結果をキーとして、残りのロケーション・パスを検索するためのハッシュテーブルを作る。例えば、次に示すＸＰａｔｈ式Ｐ４１、Ｐ４２、Ｐ４３、Ｐ４４のように、/CPEXMLv1/person/partyActivities/hobby[typeName = 'XYZ']/...の「XYZ」部分が様々な値を取るようなＸＰａｔｈ式がある場合を考える。

Ｐ４１：
/CPEXMLv1/person/partyActivities/hobby[typeName = 'SPORT']/startDate/year
Ｐ４２：
/CPEXMLv1/person/partyActivities/hobby[typeName = 'MUSIC']/composer
Ｐ４３：
/CPEXMLv1/person/partyActivities/hobby[typeName = 'SPORT']/hobbyName
Ｐ４４：

/CPEXMLv1/person/partyActivities/hobby[typeName = 'COMPUTER']/software

【００４６】

データ構造生成部２０は、上記のＸＰａｔｈ式Ｐ４１、Ｐ４２、Ｐ４３、Ｐ４４に対して、次のようなハッシュテーブルを生成する。

・キーが'SPORT'のときに検索される組が {startDate/year, hobbyName}

・キーが'MUSIC'のときに検索される組が {composer}

・キーが'COMPUTER'のときに検索される組が {software}

また、/CPEXMLv1/person/partyActivities/hobbyまでの部分に関しては、方法１によりステップ評価の共有が可能であるので、この部分に対するデータ構造を生成する（ただし、この部分についてはＸＰａｔｈ式Ｐ４１、Ｐ４２、Ｐ４３、Ｐ４４が全て同一であるので、木構造としての分岐はない）。生成されたハッシュテーブルは、方法１で生成された木構造のデータ構造と共にデータ構造格納部３０に格納される。

【００４７】

評価実行部４０は、まず上述した方法１により、/CPEXMLv1/person/partyActivity/hobbyステップの評価を行い、ノードセットを得ておく。そして、ノードセットの各ノードに対して上記のハッシュテーブルを用いた処理を行う。

図７は、ハッシュテーブルを用いた評価処理の流れを説明するフローチャートである。

図７を参照すると、まず、述部の左辺式（この例の場合は「typeName」）を評価する（ステップ７０１）。次に、左辺式の値をキーとしてハッシュテーブルを検索し（ステップ７０２）、検索結果が得られた場合は、得られたロケーション・パスを当該ノードに対して評価し、評価結果を返す（ステップ７０３、７０４）。一方、検索結果が得られない場合は、空のノードセットを返す（ステップ７０３、７０５）。

【００４８】

以上説明した方法３を用いることにより、１つのノードに対してＸＰａｔｈの個数と同じ回数の述部を評価する必要が無くなり、１回のテーブル検索で１つのノードに対して全ての述部を評価したのと同じ結果が得られることとなる。例えば、typeNameが'MUSIC'の場合、ＸＰａｔｈ式Ｐ４２の述部が真になり、Ｐ４１、Ｐ４３、Ｐ４４の述部が偽になることが１回のテーブル検索で判断される。

ＸＰａｔｈ式の数がｎで、ノードセットに含まれるノード数がｍとすると、方法３を用いない場合、全てのノードに対して全てのＸＰａｔｈを評価するためには、述部のチェックにＯ(ｎ×ｍ)の時間を要する（Ｏ(ｎ×ｍ)は、ｎ×ｍの定数倍以内の時間で計算が可能であることを意味する）。一方、方法３を用いた場合、ほぼ定数時間で実行できるハッシュテーブルの検索がｍ回行なわれる。したがって、ｎが十分大きい場合、方法３を用いることにより、評価処理の大幅な高速化を図ることができる。

【００４９】

（２）比較演算子が「！＝」の場合の高速化

上記の比較演算子が「＝」の場合に、キーがＫｉのときに検索される残りのロケーション・パスの集合がＰｉであるとすると、比較演算子が「！＝」の場合には、キーがＫｊのときに検索される残りのロケーション・パスの集合が$\cup_{i \neq j}$Ｐｉになるようにハッシュテーブルを構成する。ＸＰａｔｈ式Ｐ４１、Ｐ４２、Ｐ４３、Ｐ４４の場合、次のような検索結果を返すハッシュテーブルを構成する。

・キーが'SPORT'のとき、 {composer, software}

・キーが'MUSIC'のとき、 {startDate/year, hobbyName, software}

・キーが'COMPUTER'のとき、 {startDate/year, composer, hobbyName}

以上のハッシュテーブルを用いる他、全体の処理については（１）比較演算子が「＝」の場合と同様であるため、説明を省略する。

【００５０】

（３）比較演算子が「＞」の場合の高速化

比較演算子が「＞」の場合、データ構造生成部２０は、ステップの述部の定数をキーとして検索を行う二分探索木を作る。例えば、次の２つのＸＰａｔｈ式Ｐ５、Ｐ６がある場合

、
Ｐ５：/CPEXMLv1/person/personDemographics/birthDate[year>1990]/gender
Ｐ６：/CPEXMLv1/person/personDemographics/birthDate[year>1976]/birthPlace
次のような計算を行なう二分探索木を構成する。

```
if (key 〉 1990) then
        {gender, birthPlace}
else
        if (key 〉 1976) then
                {birthPlace}
        else
                {}
        endif
endif
```

生成された二分探索木は、方法１で生成された木構造のデータ構造と共にデータ構造格納部３０に格納される。

【００５１】
評価実行部４０は、まず上述した方法１により、/CPEXMLv1/person/personDemographics/birthDateまでのステップの評価を行う。次に、述部の左辺式（この例の場合はyear）を評価し、その値をキーとして二分探索木の検索を行う。この探索の結果、キーが１９９０よりも大きい値の場合は、「gender」と「birthPlace」の両方が返される。キーが１９９０以下で１９７６よりも大きい値の場合には「birthPlace」が返される。そして、どの条件も満たさない場合には空集合が返される。
ＸＰａｔｈ式の数がｎで、ノードセットに含まれるノード数がｍとすると、方法３を用いない場合、全てのノードに対して全てのＸＰａｔｈを評価するためには、述部のチェックにO(ｎ×ｍ)の時間を要する。一方、方法３を用いた場合、O(log n)の時間を要する二分探索がｍ回行なわれる。したがって、ｎが十分大きい場合、方法３を用いることにより、評価処理の大幅な高速化を図ることができる。
なお、ここでは比較演算子が「＞」の場合について説明したが、比較演算子が「＞＝」、「＜」、「＜＝」の場合も同様に扱うことができるのは言うまでもない。

【００５２】
次に、データ構造生成部２０が、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈ式から評価実行部４０に用いられるデータ構造（図３参照）を生成する方法について説明する。
本実施の形態で用いられるデータ構造の生成には、原則的には木構造を生成する一般的な種々の方法を用いることができ、特にその生成方法は限定されない。しかしながら、本実施の形態では、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈが変更（追加または削除）された場合に、効率的にかかる変更をデータ構造に反映させることのできる生成方法を提案し、これについて説明する。

【００５３】
ＸＭＬ文書に対してＸＰａｔｈ式を評価する処理に要する時間を短縮するためには、処理を行うたびに毎回上述したようなデータ構造を生成するのではなく、生成されたデータ構造をデータ構造格納部３０に保存しておき、複数のＸＭＬ文書に対する処理において使い回すことが考えられる。
さらに、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈの集合が変化した場合に、データ構造を最初から生成し直すのではなく、追加されたＸＰａｔｈ式と削除されたＸＰａ

ｔｈ式とを、保存してあるデータ構造に反映するようにすれば、効率的に所望のＸＰａｔｈ集合のためのデータ構造を得ることができる。
すなわち、ＸＰａｔｈ式が新たに追加された場合には、そのＸＰａｔｈ式のうちで、既存のデータ構造に対応するノードがないステップについて新たなノードを生成し、このデータ構造に加える。一方、所定のＸＰａｔｈ式が削除された場合には、そのＸＰａｔｈ式のステップに対応するノードのうちでこのＸＰａｔｈ式に固有の部分に対応するノードのみを除去する。これにより、データ構造を効率よく更新する。

【００５４】

図８は、データ構造生成部２０によるデータ構造の生成方法を説明するフローチャートである。
本実施の形態では、ＸＰａｔｈ式を１つずつ順次追加していくことにより、所望のデータ構造を得ることとする。したがって、最初にデータ構造を生成する際には、ＸＰａｔｈ格納部１０に格納されているＸＰａｔｈ集合の中から所定のＸＰａｔｈ式を選択し、他のＸＰａｔｈ式を順次加えていくこととなり、ＸＰａｔｈ集合の変更により新たなＸＰａｔｈ式が追加された場合には、既に生成されたデータ構造格納部３０に格納されているデータ構造に、新たに追加されたＸＰａｔｈ式のステップを加えていくこととなる。

【００５５】

図８を参照すると、データ構造生成部２０は、まず追加するＸＰａｔｈ式Ｐをステップｓ１、ｓ２、・・・、ｓｋに分解する（ステップ８０１）。次に、パラメータｉ及びパラメータＳを初期化、すなわちｉ＝１、Ｓ＝「トップレベルのステップの集合」とし（ステップ８０２）、ステップの集合Ｓからステップｓｉを探す（ステップ８０３）。
ステップの集合Ｓにステップｓｉが含まれていれば、次に、ｉ＋１をパラメータｉの新しい値とし（ステップ８０４、８０５）、当該新たなｉがＸＰａｔｈ式Ｐにおけるステップの数ｋに達していないか調べる（ステップ８０６）。そして、ｉの値がｋに達していなければ、パラメータＳをＳ＝「データ構造中でステップｓｉの次のレベルにおけるステップの集合」とし、ステップ８０３から処理を繰り返す（ステップ８０７）。

【００５６】

一方、ステップ８０４で、ステップの集合Ｓにステップｓｉが含まれていないと判断されたならば、ｓｉ、・・・、ｓｋに対応するステップをデータ構造に追加し（ステップ８０８）、ＸＰａｔｈ式Ｐを当該データ構造の該当ステップに登録して処理を終了する（ステップ８０９）。
また、ステップ８０６で、新しいｉの値がＸＰａｔｈ式Ｐにおけるステップの数ｋに達した場合も、ＸＰａｔｈ式Ｐを当該データ構造の該当ステップに登録して処理を終了する（ステップ８０９）。

【００５７】

ここで、図３に示したデータ構造に、次のＸＰａｔｈ式Ｐ４を追加する場合を例として、処理を具体的に説明する。
Ｐ４：/profile/location/city[text() = 'Tokyo']
まず、ステップ８０１で、ＸＰａｔｈ式Ｐ４が、ステップごとに次のように分解される
s1 = profile
s2 = location
s3 = city
s4 = [text() = 'Tokyo']
なお、ロケーション・パスが省略形を使って表現してある場合には、省略形を使わない表現に変換してから分解する。例えば、ステップの区切り「//」は「decendant-or-self::node()」に、「.」は「self::node()」に変換しておく。

【００５８】

次に、ステップ８０２で、ｉ＝１、Ｓ＝｛profile｝に初期化される。
ステップｓ１がステップの集合Ｓに含まれ、かつ残りのステップが存在するので（ステップ８０３～８０６）、ステップ８０７に進み、データ構造中で「profile」の次のレベル

のステップ「{demographics, interests}」がＳに代入される。
すると、今度はステップ８０４の判断において、ステップｓ２の値「location」はステップの集合Ｓに含まれないので、ステップ８０８に進み、ｓ２、ｓ３、ｓ４に対応するステップがデータ構造に追加される。そして、ステップ８０９で、ＸＰａｔｈ式Ｐ４がデータ構造に登録される。
図９は、図３のデータ構造にＸＰａｔｈ式Ｐ４が追加された状態のデータ構造を示す図である。
【００５９】
以上で、上述した方法１のステップ評価の高速化を実現するためのデータ構造が生成（更新）された。次に、方法２による演算式評価の高速化を実現するためのデータ構造の生成（更新）について説明する。
まず、新たに加える演算式からロケーション・パスを抽出し、その中から既存のデータ構造に登録されていないロケーション・パスだけをデータ構造に追加する。ロケーション・パスの追加は、図８のフローチャートに示した手順にて行われる。
ロケーション・パス以外の部分式に対しては、共通部分式として既に現れているかどうかを検索する。そして、既に共通部分式として現れている場合には、その共通部分式を評価するデータ構造を再使用するようにする。共通部分式として現れていな場合は、その部分式を評価するためのデータ構造を新たに作り、既存のデータ構造に追加する。
【００６０】
次に、方法３による述部評価の高速化を実現するためのデータ構造の生成（更新）について説明する。
図８のステップ８０８において、ステップｓｉの述部が、[<式> = <定数>]、[<式> != <定数>]、[<式> < <定数>]、[<式> <= <定数>]、[<式> > <定数>]、[<式> >= <定数>]のいずれかである場合、まず、式に対応するハッシュテーブル（または二分木）があるかどうかを調べる。対応するハッシュテーブル（または二分木）がない場合は、これらを生成する。すなわち、定数をキーとした検索項目を既存のハッシュテーブル（または二分木）に追加する。そして、残りのステップ列（ｓｉ+1、・・・、ｓｋ）を、木構造にして既存のデータ構造に追加する。データ構造への追加は、図８のフローチャートに示した手順にて行われる。
【００６１】
図１０は、データ構造から所定のＸＰａｔｈ式を除去する際のデータ構造生成部２０の処理を説明するフローチャートである。
図１０を参照すると、データ構造生成部２０は、まずパラメータｓをｓ＝「データ構造上でＸＰａｔｈ式Ｐが登録されているステップ」とし（ステップ１００１）、ステップｓからＸＰａｔｈ式Ｐを取り除く（ステップ１００２）。
次に、ステップｓに登録されているＸＰａｔｈ式の数が０かどうかを調べ、０でないならば処理を終了する（ステップ１００３）。
一方、ステップｓに登録されているＸＰａｔｈ式の数が０になった場合、次に、パラメータｓｐをｓｐ＝「データ構造中でステップｓの１つ上位のステップ」とし（ステップ１００４）、ステップｓｐからステップｓを取り除く（ステップ１００５）。そして、ステップｓｐの１つ下位のステップ数が０かどうかを調べ、０でないならば処理を終了する（ステップ１００６）。
これに対し、ステップｓｐの１つ下位のステップ数が０になった場合、ｓ＝ｓｐとしてステップ１００４に戻り（ステップ１００７）、新たなステップｓに関して処理を繰り返す。
【００６２】
ここで、図９に示したデータ構造からＸＰａｔｈ式Ｐ３を除去する場合を例として、処理を具体的に説明する。
まず、ステップ１００１で、パラメータｓに、データ構造上の「[text() >= 40 and text < 50]」のステップが割り当てられる。ここからＸＰａｔｈ式Ｐ３を取り除くと、ステッ

プｓに登録されているＸＰａｔｈ式の数が０になるので、ステップ１００４に進む。そして、ステップ１００５で、ステップｓの上位ステップ「demographics」からステップｓを取り除く。この場合、ステップ１００６の判断において、「demographics」の下位ステップ数が０ではないので、ここで処理が終了する。
図１１は、図９のデータ構造からＸＰａｔｈ式Ｐ３を除去した状態のデータ構造を示す図である。

【００６３】
以上で、上述した方法１のステップ評価の高速化を実現するためのデータ構造が更新された。次に、方法２による演算式評価の高速化を実現するためのデータ構造の更新について説明する。
まず、取り除こうとする演算式からロケーション・パスを抽出し、その中からこの演算式だけが使っているロケーション・パスを求め、当該ロケーション・パスをデータ構造から取り除く。また、所定のロケーション・パスが取り除こうとする演算式だけから使われているかどうかの検査は、ロケーション・パスの参照数を管理するなどの手法を用いて行うことができる。
さらに、ロケーション・パス以外の部分式に対しても同様に、この演算式だけが使っている部分式を求めて、データ構造から取り除く。

【００６４】
次に、方法３による述部評価の高速化を実現するためのデータ構造の更新について説明する。
まず、述部の定数に対応する項目をハッシュテーブル（二分木）から取り除く。そして、この項目から検索されるロケーション・パスをデータ構造から削除する。ロケーション・パスの削除は、図１０のフローチャートに示した手順にて行われる。

【００６５】
次に、ＸＰａｔｈ式及びＸＭＬ文書の具体例を挙げて、本実施の形態の動作についてさらに説明する。
ＸＰａｔｈ格納部１０に、次の４つのＸＰａｔｈ式が格納されているものとする。
Ｐ１１：/profile/interests[sport/@type = 'Soccer']/music
Ｐ１２：/profile/interests[sport/@type = 'Baseball']/book
Ｐ１３：/profile/demographics/age
Ｐ１４：count(/profile/interests[sport/@type = 'Soccer']) > 1

【００６６】
データ構造生成部２０は、図８に示した手順で上記４つのＸＰａｔｈ式Ｐ１１、Ｐ１２、Ｐ１３、Ｐ１４に対するデータ構造を生成する。なお、以下では、空の木構造（初期状態）に対して、ＸＰａｔｈ式Ｐ１１、Ｐ１２、Ｐ１３、Ｐ１４を順次追加するものとして説明する。
Ｐ１１の追加：
（１）ロケーション・パスを３つのステップ「profile」、「interests[sport/@type = 'Soccer']」、「music」に分解する（ステップ８０１）。
（２）ステップ「profile」はデータ構造に登録されていないので、追加し登録する（ステップ８０４、８０８、８０９）。
（３）ステップ「interests[sport/@type = 'Soccer']」を「interests」と「[sport/@type = 'Soccer']」とに分解する。
（４）ステップ「interests」はデータ構造に登録されていないので、追加し登録する（ステップ８０４、８０８、８０９）。
（５）述語のためのハッシュテーブルがないので生成する。
（６）述語の左辺式であるロケーション・パス「sport/@type」を表現するデータ構造を作り、ハッシュテーブルに付加する。
（７）キーが'Soccer'で値が残りのステップ「music」となるエントリをハッシュテーブルに追加する。

【００６７】
Ｐ１２の追加：
（１）ロケーション・パスを３つのステップ「profile」、「interests[sport/@type = 'Baseball']」、「book」に分解する（ステップ８０１）。
（２）ステップ「profile」と「interests」とは、既にデータ構造に登録されており、また対応するハッシュテーブルに「sport/@type」のためのデータ構造も付加されているので、データ構造への追加は行わず、ＸＰａｔｈ式Ｐ１２の登録のみを行う（ステップ８０９）。
（３）キーが'Baseball'で、値が残りのステップ「book」となるエントリをハッシュテーブルに追加する。
【００６８】
Ｐ１３の追加：
（１）ロケーション・パスを３つのステップ「profile」、「demographics」、「age」に分解する（ステップ８０１）。
（２）ステップ「profile」は登録されているが、「demographics」と「age」は登録されていないので、「demographics」と「age」とをステップとしてデータ構造に追加し、ＸＰａｔｈ式Ｐ１３を登録する（ステップ８０４、８０８、８０９）。
【００６９】
Ｐ１４の追加：
（１）式を部分式に分解して、ロケーション・パス「/profile/interests[sport/@type = 'Baseball']/」を抽出する。
（２）ロケーション・パス「/profile/interests[sport/@type = 'Baseball']」は、既に登録されているので、その結果を関数countに渡し、さらに関数countの結果を比較演算式に渡すようにデータ構造を生成する。
【００７０】
以上の結果、４つのＸＰａｔｈ式Ｐ１１、Ｐ１２、Ｐ１３、Ｐ１４を表現するデータ構造が生成される。
図１２は、生成されたデータ構造を示す図である。
生成されたデータ構造はデータ構造格納部３０に格納され、保存される。
【００７１】
次に、図１３に示すＸＭＬ文書が入力され、ＸＭＬ文書格納部５０に格納されたものとする。評価実行部４０は、ＸＭＬ文書格納部５０からこのＸＭＬ文書を読み出し、データ構造格納部３０に格納されているデータ構造を用いて、ＸＰａｔｈ式Ｐ１１、Ｐ１２、Ｐ１３、Ｐ１４を評価する。
まず、最初のステップ「profile」を実行し、２つの要素からなるノードセットＮ１を得る。

```
N1:
 { <demographics>
        <age>19</age>
   </demographics>,
   <interests>
        <sport type@='Baseball'/>
        <book>History<book/>
   </interests>
}
```

【００７２】
次に、ノードセットＮ１の各要素にステップ「interests」を適用し、次の２つのノードセットを得る。
{}
{<sport type@='Baseball'/>, <book>History<book/>}
そして、この２つの併合し、「interests」の結果であるノードセットＮ２を得る。
Ｎ２：
{<sport type@='Baseball'/>, <book>History<book/>}
【００７３】
次に、ノードセットＮ２の２つの要素に対して、左辺式「sport/@type」を評価すると、それぞれ、{} と {'Baseball'} とが得られる。
そして、'Baseball'をキーとしてハッシュテーブルを検索すると、ステップ「book」と関数countが得られる。「book」をノードセットＮ２の２つの要素に対して評価すると、{} と {History} とが得られる。この２つを併合して
ＸＰａｔｈ式Ｐ２の評価値：{History}
を得る。
【００７４】
次に、{<sport type@='Baseball'/>,を引数として関数countを評価すると値は１となり、この値１との比較演算を評価するとfalseになるので
ＸＰａｔｈ式Ｐ４の評価値：false
を得る。
そして、ノードセットＮ１の各要素にステップ「demographics」を適用し、結果を併合すると「 {<age>19</age>} 」になり、さらにこの要素にステップ「age」を適用して、
ＸＰａｔｈ式Ｐ３の評価値：{19}
を得る。
さらに、データ構造を訪問し終わってもＸＰａｔｈ式Ｐ１にたどり着くノードがないので、
ＸＰａｔｈ式Ｐ１の評価値：空のノードセット
を得る。
【００７５】
次に、本実施の形態によるＸＭＬ文書処理システムによるＸＰａｔｈ式の評価処理の実施例を示す。
本実施の形態では、上述したようにＸＰａｔｈ式をステップごとに細分化し、細分化によってできたＸＰａｔｈの断片の実行に関して、実行順序を代えたり実行回数を減らしたりすることにより、全体の実行時間を短縮している。
ＸＰａｔｈ式の断片もまたＸＰａｔｈ式であり、この断片の実行には、Apache XML Projectにて提供されるＸＳＬＴプロセッサXalan-Java 2（以下、単にXalanと記す）に含まれるＸＰａｔｈプロセッサのパッケージorg.apache.xpathを使用した。例えば、ロケーション・パスの１つのステップの実行や算術式の評価のためにXalanの機能をそのまま使っている。このため、Xalanのみを使用して実行した場合と、本実施の形態によるＸＭＬ文書処理システムでXalanを使用し実行したものの差分によって、本実施の形態によるＸＰａｔｈの評価における効率化の度合いを直接知ることができる。
【００７６】
本実施例では、CPExchange（IDEAlliance, CPExchange Specification Version 1.0, 2000.（http://www.cpexchange.org/））で定義されるXML文書に対して、多くのＸＰａｔｈ式が用意されており、それらのＸＰａｔｈ式の中から与えられたXML文書に適合するものを選ぶことを行なう。ＸＰａｔｈ式としては、次のようなパターンでＸＹＺが異なるものを生成してシステムに登録する。
/CPEXMLv1/person/partyActivities/hobby[typeName='XYZ']
/CPEXMLv1/person/partyActivities/hobby[hobbyName='XYZ']

/CPEXMLv1/person/partyActivities/food[favoriteFood='XYZ']
/CPEXMLv1/person/personDemographics/gender[@enumtype='XYZ']
/CPEXMLv1/personName[firstName='XYZ']
/CPEXMLv1/person/partyActivities/hobby/startDate[year!='XYZ']
/CPEXMLv1/person/personDemographics/birthDate[year='XYZ']
/CPEXMLv1/person/partyActivities/newspaper/startDate[year='XYZ']
/CPEXMLv1/person/partyActivities/hobby/startDate[year='XYZ']
/CPEXMLv1/person/partyActivities/magazine/startDate[year='XYZ']
また、システムの動作環境としては、ＣＰＵ１０１に米国Intel社のモバイル Pentium IIIの８００ＭＨｚを用い、メインメモリ１０３を１２８ＭＢのＲＡＭ（Random Access Memory）、オペレーティングシステムを米国Microsoft社のWindows 2000とした。

【００７７】

図１４は、上記の条件で、本実施例によるＸＰａｔｈ式の評価に要した実行時間を示す図である。図１４には、（１）XalanだけでＸＰａｔｈ式を評価するシステムと（２）本発明の手法とXalanの組合わせで評価するシステムの２つの実行時間を測定した結果が示されている。

図１４を参照すると、Xalanだけを使用したシステムでは、ＸＰａｔｈ式の数の増加に対して直線的に実行時間が増大している。一方、本実施の形態においてXalanを使用したシステムでは、ＸＰａｔｈ式の数の増加量に対して実行時間の増大がほとんど見られない。したがって、ＸＰａｔｈ式の数が増加するにしたがって両社の性能の違いが大きくなっている。

【００７８】

〔第２の実施の形態〕

上記第１の実施の形態では、評価されるべき複数のＸＰａｔｈ式に関し、その共通部分を共用するデータ構造を用いてＸＭＬ文書に対する評価を行うことにより、評価結果を共用し、評価処理の高速化を実現した。これに対し、第２の実施の形態では、評価されるべき複数のＸＰａｔｈ式の間の依存関係を解析し、得られた依存関係に基づいて当該複数のＸＰａｔｈ式の冗長部分を省略したデータ構造を用いて評価を行うことにより、評価処理を簡単化し、評価処理の高速化を実現する。なお、本実施の形態では、ウェブページにアノテーションを付加する目的で、アノテーション内のＸＰａｔｈ式が対象のウェブページの所定のエレメントを正しく指しているかどうかを評価する場合を例として説明する。

第２の実施の形態によるウェブページ処理システムは、第１の実施の形態と同様に、例えば図１に示すようなハードウェア構成を有するコンピュータ装置にて実現される。

【００７９】

図１５は、第２の実施の形態によるウェブページ処理システムの機能を説明するブロック図である。

図１５を参照すると、本実施の形態は、処理対象であるウェブページ（ＨＴＭＬ文書）の特定部分を指し示す複数のＸＰａｔｈ式を格納したＸＰａｔｈ格納部１５１０と、ＸＰａｔｈ格納部１５１０に格納されているＸＰａｔｈ式から本実施の形態による効率的な評価を行うためのデータ構造を生成するデータ構造生成部１５２０と、データ構造生成部１５２０にて生成されたデータ構造を格納するデータ構造格納部１５３０と、データ構造格納部１５３０に格納されているデータ構造を用いて処理対象であるウェブページに対してＸＰａｔｈ式の評価を行う評価実行部１５４０とを備え、また処理対象であるウェブページを格納する文書格納部１５５０と、評価実行部１５４０による評価を経たウェブページを対象として所定の情報処理を実行する文書処理部１５６０とを備える。

【００８０】

図１５に示したシステム構成において、データ構造生成部１５２０、評価実行部１５４０及び文書処理部１５６０は、図１に示したプログラム制御されたＣＰＵ１０１にて実現される仮想的なソフトウェアブロックである。ＣＰＵ１０１を制御してこれらの機能を実現するプログラムは、磁気ディスクや光ディスク、半導体メモリ、その他の記録媒体に格納

して配布したり、ネットワークを介して配信したりすることにより提供され、メインメモリ１０３に読み込まれる。また、ＸＰａｔｈ格納部１５１０、データ構造格納部１５３０及び文書格納部１５５０は、メインメモリ１０３にて実現される。なお、メインメモリ１０３に保持されるデータやプログラムは、必要に応じてハードディスク１０５などの記憶装置に退避させることができる。

【００８１】

図１５に示したシステム構成において、ＸＰａｔｈ格納部１５１０には、ウェブページに対して評価を行うべき、予め用意された複数のＸＰａｔｈ式が格納されている。データ構造格納部１５３０には、データ構造生成部１５２０にて生成され、評価実行部１５４０にて用いられるデータ構造が格納される。文書格納部１５５０には、図１に示したネットワークインターフェイス１０６や、キーボード／マウス１０９等の入力デバイス、処理対象であるウェブページを格納した記録媒体から当該ウェブページを読み出すドライブ装置などを介して入力されたウェブページが格納されている。

【００８２】

データ構造生成部１５２０は、ＸＰａｔｈ格納部１５１０に格納されているＸＰａｔｈ式から、評価実行部１５４０による評価に用いるデータ構造を生成する。このデータ構造を用いることにより、ウェブページに対してＸＰａｔｈ格納部１５１０に格納されている複数のＸＰａｔｈ式を個別に評価する場合に比べて効率の良い評価を行うことが可能となる。このデータ構造についての詳細、このデータ構造を用いた評価方法及びこのデータ構造の生成方法については後述する。

【００８３】

評価実行部１５４０は、ＸＰａｔｈエンジンであり、文書格納部１５５０に格納されているウェブページに対して、データ構造生成部１５２０にて生成されたデータ構造を用いてＸＰａｔｈ式の評価（マッチング）を行う。これにより、当該入力文書がＸＰａｔｈ格納部１５１０に格納されているＸＰａｔｈ式にて指定される特定部分を持つかどうかを、効率よく評価することができる。評価実行部１５４０による評価結果はメインメモリ１０３の所定の領域に格納される。

文書処理部１５６０は、メインメモリ１０３の所定領域から評価実行部１５４０の評価結果を読み出し、当該評価結果に基づいて、文書格納部１５５０に格納されているウェブページがＸＰａｔｈ格納部１５１０に格納されているＸＰａｔｈ式にて指定される特定部分を持つならば、文書格納部１５５０から当該ウェブページを処理対象として読み出し、目的とする所定の情報処理、すなわちウェブページにアノテーションを付加する処理を実行する。

【００８４】

次に、本実施の形態におけるＸＰａｔｈ式の評価を効率化する方法を詳細に説明する。

ウェブページにアノテーションを付加する作業において、所定のアノテーションが所定のウェブページに対して適用可能かどうかを判断するには、当該アノテーション内の複数のＸＰａｔｈ式が対象のウェブページの所定のエレメントを正しく指しているかどうかを評価することが必要である。これは、複数のＸＰａｔｈ式の集合（以下、ＸＰａｔｈセットと称す）が当該ウェブページにマッチしているかどうかを評価する問題と言い換えることができる。ＸＰａｔｈセットがウェブページにマッチするとは、ＸＰａｔｈセットに含まれる全てのＸＰａｔｈ式が当該ウェブページの構造にマッチすることを意味する。本実施の形態では、特に、複数のアノテーションの中で所定のウェブページに適用できるものを探索する場合のように、複数のＸＰａｔｈセットのうちでウェブページにマッチするものを判定する場合の評価処理の効率化を対象とする。

【００８５】

本実施の形態は、複数のＸＰａｔｈ式の間の依存関係に基づいてＸＰａｔｈ式の評価を簡単化するためのデータ構造を生成し、このデータ構造を用いてウェブページに対する評価を行うことにより、上記のような場合の評価処理の高速化を実現する。本実施の形態では、評価実行部１５４０の種類（下記のタイプ１、タイプ２）に応じて異なるアルゴリズム

が適用される。
タイプ１：ＸＰａｔｈ評価関数の呼び出しオーバーヘッドが大きく、ＸＰａｔｈ式の評価の回数が少ない方がシステム全体のパフォーマンス（処理効率）を向上させることができる。
タイプ２：ＸＰａｔｈ評価関数の呼び出しオーバーヘッドが小さく、ＸＰａｔｈ式を分割して評価を行ってもシステム全体のパフォーマンス（処理効率）は大きく低下しない。
【００８６】
まず、評価実行部１５４０の種類がタイプ１である場合について説明する。
この場合、ＸＰａｔｈ式の評価回数が少ない方がシステムのパフォーマンスを向上させられるため、データ構造生成部１５２０は、生成するデータ構造においてＸＰａｔｈ式の分割を行わない。具体的には、
１．ＸＰａｔｈ式の共通性に基づくＸＰａｔｈ式の絞込み
２．ＸＰａｔｈ式の依存関係に基づく連結ＸＰａｔｈ式の簡略化
３．ＸＰａｔｈ式の依存関係に基づくＸＰａｔｈ式の省略
４．ＸＰａｔｈ式の統合
５．ＸＰａｔｈ式のツリー長とＤＯＭツリー上での位置に基づく評価処理の優先順位付け
６．実行時判断のためのデータ構造の構築
という処理が行われる。各処理について、以下に説明する。なお、以下の説明において、「ＸＰａｔｈ式が成立する」「ＸＰａｔｈ式が真になる」とは、「評価対象であるウェブページ中に、ＸＰａｔｈ式に対応するエレメントが存在している」ことを意味する。
【００８７】
１．ＸＰａｔｈ式の共通性に基づくＸＰａｔｈ式の絞込み
複数のＸＰａｔｈセットが共通に含んでいるＸＰａｔｈ式は、ウェブページにマッチするＸＰａｔｈセットの判定に関与しない。このようなＸＰａｔｈ式の評価結果は、各ＸＰａｔｈセットにおいて共通となる（差別化されない）ためである。したがって、複数のＸＰａｔｈセットに含まれないＸＰａｔｈ（specific XPath）に対する評価を先に行う。
図１６は、ＸＰａｔｈセットとこれに含まれるＸＰａｔｈ式とを一覧表示した図表である。図１６に示す例では、４種類のＸＰａｔｈセット（Ａ、Ｂ、Ｃ、Ｄ）にそれぞれ複数のＸＰａｔｈ式が含まれている。図１６に示すＸＰａｔｈ式には、重複するものが多く含まれている。例えば、番号５（ＸＰａｔｈセットＡ）のＸＰａｔｈ式と番号１４（ＸＰａｔｈセットＢ）のＸＰａｔｈ式とは、全く同一のＸＰａｔｈである。
図１７は、図１６に示したＸＰａｔｈ式のうち重複するものを整理して識別情報（ＸＰａｔｈＩＤ）を割り当てた様子を示す図である。図１７に示す例では、図１６に示した３１個のＸＰａｔｈ式が１８個に整理されている。
【００８８】
図１８は、図１７のように整理されたＸＰａｔｈ式をさらに解析し、複数のＸＰａｔｈセットに共通するＸＰａｔｈ式を抽出した様子をグラフ構造で示した図である。
図１７を参照すると、ＸＰａｔｈ式Ｘ７はＸＰａｔｈセットＣ、Ｄに共通して含まれ、またＸＰａｔｈ式Ｘ１、Ｘ２、Ｘ３、Ｘ１８は全てのＸＰａｔｈセットに共通して含まれる。図１８の例では、これら複数のＸＰａｔｈセットに共通して含まれるＸＰａｔｈ式を表すノードから個々のＸＰａｔｈセットを分岐させて、各ＸＰａｔｈセットに固有のＸＰａｔｈ式を記述している。したがって、図１８の例において、評価対象は、各ＸＰａｔｈセットを表すノードに付加されている１３種類のＸＰａｔｈ式（Ｘ４～Ｘ６、Ｘ８～Ｘ１７）に絞込まれることとなる。
【００８９】
２．ＸＰａｔｈ式の依存関係に基づく連結ＸＰａｔｈ式の簡略化
連結ＸＰａｔｈ式は、複数のＸＰａｔｈ式をＯＲ記号（｜）で結合したものである。例えば、所定のウェブページ（html[1]）の所定の位置（body[1]）に配置された２つのテーブルコンテンツ（table[1]、table[2]）のいずれかを指す連結ＸＰａｔｈ式、
/html[1]/body[1]/table[1]|/html[1]/body[1]/table[2]

を考える。これは、所定のウェブページに、/html[1]/body[1]/table[1]または/html[1]/body[1]/table[2]のいずれか一方が存在すれば当該ＸＰａｔｈ式が成立することを示している。このとき、ＸＰａｔｈ式の表記規則から、table[1]が存在しなければ、table[2]も存在しないことは自明である。したがって、この連結ＸＰａｔｈ式に関しては、前半の
/html[1]/body[1]/table[1]
のみを評価すれば良いことになる。このように所定のＸＰａｔｈ式（ＸＰａｔｈ１）が成立するとき、別のＸＰａｔｈ式（ＸＰａｔｈ２）も成立する場合、ＸＰａｔｈ２はＸＰａｔｈ１に依存していると定義する。この性質を利用して連結ＸＰａｔｈを簡略化する。

【００９０】

ＸＰａｔｈ式の間の依存関係には、主として次の２種類がある。

・同一タグ兄弟（sibling）関係

ＸＰａｔｈ１　/html[1]/body[1]/table[2]

ＸＰａｔｈ２　/html[1]/body[1]/table[1]

であるとき、ＸＰａｔｈ１とＸＰａｔｈ２とは兄弟の関係にあり、

ＸＰａｔｈ１が成立すればＸＰａｔｈ２も成立し、

ＸＰａｔｈ２が不成立ならばＸＰａｔｈ１も不成立である。

したがって、「ＸＰａｔｈ１はＸＰａｔｈ２に依存」する。

・親ノード関係

ＸＰａｔｈ１　/html[1]/body[1]/table[1]/tr[1]/td[1]

ＸＰａｔｈ２　/html[1]/body[1]/table[1]

であるとき、ＸＰａｔｈ１はＸＰａｔｈ２の子孫であり、

ＸＰａｔｈ１が成立すればＸＰａｔｈ２も成立し、

ＸＰａｔｈ２が不成立ならばＸＰａｔｈ１も不成立である。

したがって、「ＸＰａｔｈ１はＸＰａｔｈ２に依存」する。

【００９１】

連結ＸＰａｔｈ式の簡略化は、以下の手順で行う。

まず、連結ＸＰａｔｈ式を展開して新たにＸＰａｔｈＩＤを割り当てる。例えば図１７に示したＸＰａｔｈ式Ｘ１は、

/html[1]/body[1]/table[1]/tbody[1]/tr[1]|/html[1]/body[1]/table[1]/tbody[1]/tr[2]

であるから、次の２つのＸＰａｔｈ式に展開することができる。

Ｘ１－１：/html[1]/body[1]/table[1]/tbody[1]/tr[1]

Ｘ１－２：/html[1]/body[1]/table[1]/tbody[1]/tr[2]

同様に、図１８において評価対象とした１３種類のＸＰａｔｈ式を解析し展開する。そして、展開された各ＸＰａｔｈ式の依存関係を調べる。

【００９２】

図１９は、得られた展開後のＸＰａｔｈ式の依存関係をツリー構造で示す図である。なお、図中、円で囲まれたＸＰａｔｈ式Ｘ６－５とＸＰａｔｈ式Ｘ７とは、同一のＸＰａｔｈ式である。

図１９を参照すると、Ｘ１１－３、Ｘ１１－２、Ｘ１１－１のように、で本来１つの連結ＸＰａｔｈ式（Ｘ１１）に含まれている複数のＸＰａｔｈ式が依存関係を持つ場合がある。この場合、次の理由により、下位のＸＰａｔｈ式であるＸ１１－２、Ｘ１１－３は評価が不要となる。

すなわち、Ｘ１１－１が成立した場合、元の連結ＸＰａｔｈ式Ｘ１１も成立する。これは結合関係にあるＸＰａｔｈ式が１つでも成立すれば、かかるＸＰａｔｈ式を含む連結ＸＰａｔｈ式は成立するという性質を持つからである。一方、依存関係により、Ｘ１１－１が不成立の場合、Ｘ１１－２、Ｘ１１－３も不成立となる。したがって、Ｘ１１－２、Ｘ１１－３はＸＰａｔｈ式Ｘ１１の評価結果に影響を与えない。

以上のように、他のＸＰａｔｈ式に依存するＸＰａｔｈ式を削除することによって、連結ＸＰａｔｈ式が簡略化される。

図２０は、図１９に示したＸＰａｔｈ式の依存関係に基づいて連結ＸＰａｔｈ式を簡略化した様子を示す図である。

【００９３】

３．ＸＰａｔｈ式の依存関係に基づくＸＰａｔｈ式の省略

複数のＸＰａｔｈ式において、ウェブページの構造やエレメントのバリエーションなどから受ける制約のために、１つのＸＰａｔｈ式について評価結果が得られた場合に、他のＸＰａｔｈ式の評価結果が自明となる場合がある。例えば、ＸＰａｔｈセット中に所定のウェブページ（html[1]）の所定の位置（body[1]）に配置された２つのテーブルコンテンツ（table[1]、table[2]）を指す２つのＸＰａｔｈ式、

ＸＰａｔｈ１　/html[1]/body[1]/table[2]

ＸＰａｔｈ２　/html[1]/body[1]/table[1]

がある場合を考える。この場合、table[2]を指すＸＰａｔｈ１が存在するならば、table[1]を指すＸＰａｔｈ２も必ず存在する。したがって、ＸＰａｔｈ２は評価しなくても良いこととなる。この性質を利用して評価するＸＰａｔｈ式を削減する。

【００９４】

これらＸＰａｔｈの依存関係についてさらに詳しく説明する。

所定のＸＰａｔｈセットが成立するためには、当該ＸＰａｔｈセットに「含まれる全てのＸＰａｔｈ式の成立」が条件となる。上記のＸＰａｔｈ１、ＸＰａｔｈ２は、どちらか一方でも不成立の場合、ＸＰａｔｈセット全体が成立しない。すなわち、以下のような関係が成り立つ。

ＸＰａｔｈ１が成立すればＸＰａｔｈ２も成立する。

ＸＰａｔｈ１が不成立ならばＸＰａｔｈセット全体が成立しない。

これにより、ＸＰａｔｈ２の成立不成立を無視することができる。言い換えると、次のようになる。

「所定のＸＰａｔｈ式（上の例でＸＰａｔｈ２）が含まれる全てのＸＰａｔｈセットが依存関係ツリー（図１９、２０に示したようなツリー構造）の末端側に出現するとき、そのＸＰａｔｈ式（上の例でＸＰａｔｈ２）は成立不成立に関与しないため、省略できる。」

図２１は、図２０に示したＸＰａｔｈ式の依存関係に基づいてＸＰａｔｈ式を省略した様子を示す図である。図２１（Ａ）は、省略されるＸＰａｔｈ式（Ｘ２、Ｘ７、Ｘ９）をマークした状態、図２１（Ｂ）は、図２１（Ａ）でマークされたＸＰａｔｈ式を依存関係ツリーから消去した状態を示す。図中のＸＰａｔｈ式に付された「Ａ」、「Ｂ」、「ＡＢＣＤ」等の符号は、当該ＸＰａｔｈ式が含まれるＸＰａｔｈセットを示す。なお、図２１中では、Ｘ１１及びＸ１の添え字表記（Ｘ１－１、Ｘ１１－１の「－１」）をはずしてある。これは、２．の処理によって、これらのＸＰａｔｈ式が連結ＸＰａｔｈ式から単独ＸＰａｔｈ式に変わったためである。

【００９５】

４．ＸＰａｔｈ式の統合

ＸＰａｔｈ評価関数の呼び出しオーバーヘッドが大きいタイプ１の評価実行部１５４０を用いる場合、データ構造生成部１５２０は、評価を行うべきＸＰａｔｈ式の本数を減らすため、２．の処理においてＸＰａｔｈ式の依存関係を解析した際にばらばらにした連結ＸＰａｔｈ式を、再び結合する。

図２２は、図２１に示したＸＰａｔｈ式の依存関係ツリー中のＸＰａｔｈ式に対して再結合可能なものを再結合する様子を示す図である。

結合処理は、次の規則に基づいて行う。

（１）結合後のＸＰａｔｈ式の位置は、所属するＸＰａｔｈ式の全てが共有する親ノードの直下とする。

（２）共有する親ノードがない場合、結合後のＸＰａｔｈ式自体がルートノードとなる。

図２１、２２に示す例では、例えば、Ｘ５－＊というＸＰａｔｈ式は、Ｘ５－４、Ｘ５－１、Ｘ５－２の３つが存在する（図２１（Ｂ）参照）。これらのＸＰａｔｈ式を、それぞれルートに向かって辿っていくと、Ｘ４で合流する。そこで、Ｘ５－＊の各ＸＰａｔｈ式

をＸ４の直下のノードとして１つにまとめ（図２２（Ａ）参照）、再結合してＸＰａｔｈ式Ｘ５とする（図２２（Ｂ）参照）。同様にして、Ｘ６、Ｘ１７が再結合される。
ここで、仮にＸ５－２がＸ１８の子ノードであったとする。この場合、共有する親ノードは存在しないため、Ｘ５－＊は１つのルートノードとして、Ｘ１、Ｘ３、Ｘ４、Ｘ１８と同じレベルのノードとして結合されることとなる。
【００９６】
５．ＸＰａｔｈ式のツリー長とＤＯＭツリー上での位置に基づく評価処理の優先順位付け
評価対象であるウェブページにマッチしないＸＰａｔｈセットを効率よく決定しふるい落とすためには、「存在しにくいエレメント」を対象にしたＸＰａｔｈ式を先に処理することが好ましい。そこで、次のような基準を用いて評価するＸＰａｔｈ式の順番（優先順位）を決定する。
（１）ＸＰａｔｈ式のツリー長が長いＸＰａｔｈ式
（２）評価対象であるウェブページのＤＯＭツリー上で、より後ろに位置するＸＰａｔｈ式
（３）ＸＰａｔｈ式の依存関係に基づき、成立したときにより多くのＸＰａｔｈ式に影響を与えるものを優先する。
を先に評価する。例えば、/table[1]よりもより末端まで含んだ/table[1]/tr[1]/td[1]の評価を先に行う（規則（１））。また、/table[1]と/table[2]とでは、ＤＯＭツリー上を巡回したときに、より「後ろに」出現する、言い換えればＨＴＭＬのタグ順序で後ろに出現するエレメントを指しているので、/table[2]の評価を先に行う（規則（２））。これは、ＤＯＭツリー上で後ろに位置するＸＰａｔｈ式（/table[2]）の出現率は前に位置するＸＰａｔｈ式（/table[1]）の出現確率以下だからである。
【００９７】
図２３は、図２２に示した依存関係ツリーの各ノード（ＸＰａｔｈ式）に、上記の規則に従って優先順位（丸数字）を付加した様子を示す図、図２４は、図２３に示した優先順位に従ってＸＰａｔｈ式を並べた図表である。図２４には、さらに各ＸＰａｔｈ式が依存している他のＸＰａｔｈ式（依存ＸＰａｔｈ）、依存レベル、ＸＰａｔｈ式のツリー長が記載されている。ここで、依存レベルはルートノードを０としたときのツリーの深さを意味する。また、ツリー長は、ＸＰａｔｈ式自身のツリーの長さである。連結ＸＰａｔｈ式のツリー長は、当該連結ＸＰａｔｈ式を構成するＸＰａｔｈ式のうちでツリー長が最も長いＸＰａｔｈ式のツリー長となっている。
図２３において、例えばＸＰａｔｈ式Ｘ１０、Ｘ１１は、共に依存レベルが３であり、依存レベルが最も深い。すなわち成立した場合に、より多くのＸＰａｔｈ式に影響を与える。Ｘ１０とＸ１１とでは、Ｘ１０の方がツリー長が長い。そのため、Ｘ１０の優先順位が１、Ｘ１１の優先順位が２となっている。また、依存レベル及びツリー長がいずれも等しいＸＰａｔｈ式Ｘ１２、Ｘ１３では、/td[2]/table[1]を指すＸ１３の方が評価対象のウェブページのＤＯＭツリー上で後方に位置するので、Ｘ１３の方が優先順位が高くなっている。
【００９８】
６．実行時判断のためのデータ構造の構築
以上のようにしてＸＰａｔｈ式の依存関係に基づいて整理されたＸＰａｔｈセットを、ウェブページとのマッチングの処理において参照可能なデータ構造（例えば有効グラフ構造）に変換し、所定の記憶装置（例えば図１に示したメインメモリ１０３）に保存する。
【００９９】
以上のようにして、ＸＰａｔｈセットの評価を効率的に行うためのデータ構造が生成された後、評価実行部１５４０が、このデータ構造を用いて、文書格納部１５５０に格納されているウェブページに対する評価を行う。
上述したように、この評価実行部１５４０は、ＸＰａｔｈ式の評価の回数が少ない方がシステム全体のパフォーマンスを向上させることができるタイプ（タイプ１）であるため、優先順位の高いＸＰａｔｈ式から順に、次の手順で評価を行う。

１．未処理のＸＰａｔｈ式の中で優先順位の最も高いＸＰａｔｈ式を選択し、ウェブページに対する評価を行う。
２．評価対象のＸＰａｔｈ式がマッチした場合、当該ＸＰａｔｈ式に依存している他のＸＰａｔｈ式（優先順位の低いＸＰａｔｈ式）はマッチすることが保証されるため評価から除外する。
一方、評価対象のＸＰａｔｈ式がマッチしなかった場合、当該ＸＰａｔｈ式を含むＸＰａｔｈセットは、当該ウェブページにマッチしないため、当該ＸＰａｔｈセットに含まれる全てのＸＰａｔｈ式を評価対象から除外する。
３．各ＸＰａｔｈセットに固有のＸＰａｔｈ式（specific XPath）の全てについて処理がおわった段階で、全てのＸＰａｔｈ式のマッチしているＸＰａｔｈセットは、当該ウェブページに完全にマッチする可能性がある。そこで、そのようなＸＰａｔｈセットが存在するかどうかを調べる。
４．そのようなＸＰａｔｈセットが存在するならば、当該ＸＰａｔｈセットに関して、他のＸＰａｔｈセットと共有するＸＰａｔｈ式の評価を行う。
一方、そのようなＸＰａｔｈセットが存在しない場合は、当該ウェブページに完全にマッチするＸＰａｔｈセットは存在しないため、他のＸＰａｔｈセットと共有するＸＰａｔｈ式の評価を行わずに処理を終了する。
本実施の形態によるＸＰａｔｈセットの評価処理を、複数のアノテーションの中からウェブページに付加するアノテーションを決定する処理に用いる場合、上記の手順で全てのＸＰａｔｈ式がマッチすると判断されたＸＰａｔｈセットを持つアノテーションが、ウェブページに付加できるアノテーションとして決定されることとなる。

【０１００】

次に、評価実行部１５４０の種類がタイプ２である場合について説明する。
この場合、ＸＰａｔｈ式を分割して評価を行ってもシステム全体のパフォーマンスが大きく低下しないため、データ構造生成部１５２０は、生成するデータ構造においてＸＰａｔｈ式を分割し、個々のＸＰａｔｈ式を簡単化する。具体的には、
１．ＸＰａｔｈ式の共通性に基づくＸＰａｔｈ式の絞込み
２．ＸＰａｔｈ式の依存関係に基づく連結ＸＰａｔｈ式の簡略化
３．ＸＰａｔｈ式の依存関係に基づくＸＰａｔｈ式の省略
４．ＸＰａｔｈ式の分割
５．ＸＰａｔｈ式のツリー長とＤＯＭツリー上での位置に基づく評価処理の優先順位付け
６．実行時判断のためのデータ構造の構築
という処理が行われる。上記の処理のうち、１～３の処理は、評価実行部１５４０の種類がタイプ１の場合と同様であるので、説明を省略する。

【０１０１】

４．ＸＰａｔｈ式の分割
ＸＰａｔｈ評価関数の呼び出しオーバーヘッドが小さいタイプ２の評価実行部１５４０を用いる場合、データ構造生成部１５２０は、複数のＸＰａｔｈ式で共通する部分と固有の部分とを細分化してそれぞれを評価することにより、処理の高速化を図ることができる。
例えば、次の２つのＸＰａｔｈ式（ＸＰａｔｈ１、ＸＰａｔｈ２）は、/html[1]/body[1]/table[1]/tr[1]/td[1]の部分（ノード）を共有している。
ＸＰａｔｈ１　/html[1]/body[1]/table[1]/tr[1]/td[1]/font[1]
ＸＰａｔｈ２　/html[1]/body[1]/table[1]/tr[1]/td[1]/b[1]
したがって、この共通ノードの評価を行い、そのノードから相対パスをfont[1]及びb[1]だけ評価することにより、高速な処理を実現できる。ここで、/html[1]/body[1]/table[1]/tr[1]/td[1]と/html[1]/body[1]/table[1]/tr[1]/td[1]/font[1]との間の関係を「包含関係」と呼び、「/html[1]/body[1]/table[1]/tr[1]/td[1]/font[1]が/html[1]/body[1]/table[1]/tr[1]/td[1]を包含している」とする。この包含関係に基づいて、ＸＰａｔｈ式を共通部分と固有の部分とに分割し、それぞれを部分ＸＰａｔｈ式とする。
図２５は、依存関係に基づく簡略化、省略の後、この分割処理を経て得られたＸＰａｔｈ

式のリストを示す図表である。図中のＸＰａｔｈ式ＰＸ１、ＰＸ２、ＰＸ３が、分割処理に伴って追加された部分ＸＰａｔｈ式である。この３つのＸＰａｔｈ式を分割処理することにより、各ＸＰａｔｈ式が、図１７等に列挙したＸＰａｔｈ式に比して非常に簡潔になっていることがわかる。また、ＰＸ１、ＰＸ２、ＰＸ３を含めた依存関係ツリーを図２６に示す。

【０１０２】

５．ＸＰａｔｈ式のツリー長とＤＯＭツリー上での位置に基づく評価処理の優先順位付け

評価対象であるウェブページにマッチしないＸＰａｔｈセットを効率よく決定しふるい落とすためには、「存在しにくいエレメント」を対象にしたＸＰａｔｈ式を先に処理することが好ましい。そこで、次のような基準を用いて評価するＸＰａｔｈ式の順番（優先順位）を決定する。

優先順位を決定する処理は、上述した評価実行部１５４０の種類がタイプ１である場合の処理と同様であるが、ここでは３．の処理で新たに追加されたＸＰａｔｈ式ＰＸ１、ＰＸ２、ＰＸ３を含めて優先順位が決定される。

図２７は、図２６に示した依存関係ツリーの各ノード（ＸＰａｔｈ式）に、上記の規則に従って優先順位を付加した様子を示す図、図２８は、図２７に示した優先順位にしたがってＸＰａｔｈ式を並べた図表である。なお、上の４．の処理で複数のＸＰａｔｈ式における共通部分が分割されて個別のＸＰａｔｈ式とされていることから、ＸＰａｔｈ式自身のツリー長は優先順位を決定するために参酌しない。

【０１０３】

６．実行時判断のためのデータ構造の構築

以上のようにしてＸＰａｔｈ式の依存関係に基づいて整理されたＸＰａｔｈセットを、ウェブページとのマッチングの処理において参照可能なデータ構造（例えば有効グラフ構造）に変換し、所定の記憶装置（例えば図１に示したメインメモリ１０３）に保存する。この処理は、上述した評価実行部１５４０の種類がタイプ１である場合の処理と同様である。

【０１０４】

以上のようにして、ＸＰａｔｈセットの評価を効率的に行うためのデータ構造が生成された後、評価実行部１５４０が、このデータ構造を用いて、文書格納部１５５０に格納されているウェブページに対する評価を行う。

上述したように、この評価実行部１５４０は、ＸＰａｔｈ式の評価の回数が多くてもシステム全体のパフォーマンスが大きく低下しないタイプ（タイプ２）であるため、優先順位の高いＸＰａｔｈ式から順に、次の手順で評価を行う。

１．未処理のＸＰａｔｈ式の中で優先順位の最も高いＸＰａｔｈ式を選択し、ウェブページに対する評価を行う。このとき、評価対象のＸＰａｔｈ式が包含しているＸＰａｔｈ式の評価も行う。なお、包含しているＸＰａｔｈ式の評価が既になされている場合（より優先順位の高いＸＰａｔｈ式の評価に伴って評価されている場合）、その評価結果を利用する。

２．１．で評価したＸＰａｔｈ式がマッチした場合、当該ＸＰａｔｈ式に依存している他のＸＰａｔｈ式（優先順位の低いＸＰａｔｈ式）はマッチすることが保証されるため評価から除外する。

一方、評価対象のＸＰａｔｈ式がマッチしなかった場合、当該ＸＰａｔｈ式を含むＸＰａｔｈセットは、当該ウェブページにマッチしないため、当該ＸＰａｔｈセットに含まれる全てのＸＰａｔｈ式を評価対象から除外する。

また、評価対象のＸＰａｔｈ式が包含しているＸＰａｔｈ式がマッチしなかった場合、当該ＸＰａｔｈ式を包含する他の全てのＸＰａｔｈ式を原則として評価対象から除外する。ただし、連結ＸＰａｔｈ式については、当該連結ＸＰａｔｈ式を構成する全てのＸＰａｔｈ式が評価されるまでは、当該連結ＸＰａｔｈ式がマッチするかどうかを判断できないので、評価対象からは除外しない。

３．各ＸＰａｔｈセットに固有のＸＰａｔｈ式（specific XPath）の全てについて処理が

おわった段階で、全てのＸＰａｔｈ式のマッチしているＸＰａｔｈセットは、当該ウェブページに完全にマッチする可能性がある。そこで、そのようなＸＰａｔｈセットが存在するかどうかを調べる。
４．そのようなＸＰａｔｈセットが存在するならば、当該ＸＰａｔｈセットに関して、他のＸＰａｔｈセットと共有するＸＰａｔｈ式の評価を行う。
一方、そのようなＸＰａｔｈセットが存在しない場合は、当該ウェブページに完全にマッチするＸＰａｔｈセットは存在しないため、他のＸＰａｔｈセットと共有するＸＰａｔｈ式の評価を行わずに処理を終了する。
本実施の形態によるＸＰａｔｈセットの評価処理を、複数のアノテーションの中からウェブページに付加するアノテーションを決定する処理に用いる場合、上記の手順で全てのＸＰａｔｈ式がマッチすると判断されたＸＰａｔｈセットを持つアノテーションが、ウェブページに付加できるアノテーションとして決定されることとなる。

【０１０５】

以上説明した本実施の形態は、ＸＰａｔｈ式の全機能に対して効率的な評価を行うためのデータ構造を生成（最適化）するものではなく、頻繁に利用される一部の機能について最適化する。
図２９は、本実施の形態による最適化が可能なＸＰａｔｈ式の構造を説明する図である。
図２９に示すＸＰａｔｈ式において、Ｅｘはｘ番目のエレメントである。また、Ｃｘはｘ番目のsibling内でのポジションを示す数値であり、predicateで表現すると、[position() = Cx]と等しい。すなわち、ＸＰａｔｈ式を２つの部分に分けることができ、前半部分（図２９のパート１）が次の条件に当てはまる場合に、本実施の形態の最適化を行うことができる。
・ｎ段目まで子孫方向に１段ずつツリーを移動している。
・軸（axis）は子（child）のみであり、他の軸（axis）は使用されていてはならない。また、descendant-or-self::node()の省略である「//」指定も使用されていてはならない。
・predicateは[position() = 番号]（省略形は[番号]）のみが使用可能。
・predicateの省略（/tag/）は無し。
後半部分（図２９のパート２）には、どのようなＸＰａｔｈ式が出現してもかまわないが、これらの部分は最適化されない。
言い換えれば、本実施の形態は、ウェブページを記述したＨＴＭＬドキュメントのようにツリー長深くかつ不定形であるツリーのノードを指すＸＰａｔｈ式には、上記のような構造（パート１、パート２に分けることができるという構造）を持ったものが多いという性質を利用している。

【０１０６】

なお、本実施の形態では、ＸＰａｔｈセット内の全てのＸＰａｔｈ式が成立することをＸＰａｔｈセット成立の条件として説明した。しかし、このような厳密なマッチングだけではなく、「最もマッチするものを必ず１つ」選択するという課題も存在する。例えば、ウェブページにアノテーションを付加するシステムにおいて所定のウェブページに付加すべきアノテーションを選択する場合、何らかの評価基準を用いてＸＰａｔｈ式の適合度を算出し、最も適しているものを選択することによって、全くトランスコードできないという状況を回避することができると考えられる。このような用途のために、本実施の形態を拡張し、ＸＰａｔｈセットの成立を判断する際に曖昧性を許容するアルゴリズムを導入することができる。

【０１０７】

このアルゴリズムの具体的な手順について説明する。
１．評価実行部１５４０によるマッチングの実行時に、所定のＸＰａｔｈ式がマッチしなかった場合、そのＸＰａｔｈ式に対してマッチしなかったことを示すフラグ（UNMATCHフラグ）を立てる。
２．マッチしないＸＰａｔｈ式と同じＸＰａｔｈセットに含まれることによって評価され

なかったＸＰａｔｈ式に対して、評価が行われていないこと示すフラグ（UNEVALUATEDフラグ）を立てる。
３．全てのＸＰａｔｈセットがマッチしないと評価された段階で、UNEVALUATEDフラグのたっている全てのＸＰａｔｈ式を評価する。（ただし、既に依存関係や包含関係によって評価結果の判明しているものは除外する）マッチしなかったＸＰａｔｈ式にはUNMATCHフラグを立てる。
【０１０８】
ＸＰａｔｈセットの成立を判断する際に曖昧性を許容するアルゴリズムを導入する場合、適合度の評価手法に関わらず、ここまでの処理は共通である。しかし、これ以降の処理は、適合度の評価手法に応じて異なる処理が行われる。ここでは一例として、「最もマッチしないＸＰａｔｈ式の数の少ないＸＰａｔｈセット」であり、かつ「DOMツリー上の距離で定義されるＸＰａｔｈ式の適合度が最も大きいもの」を選択するアルゴリズムを示す。
４．UNMATCHフラグの立っているＸＰａｔｈ式の最も少ないＸＰａｔｈセットを選択する。
５．UNMATCHフラグの立っているＸＰａｔｈ式の数が同数であるＸＰａｔｈセットが複数存在する場合、各ＸＰａｔｈセットに含まれているＸＰａｔｈ式の適合度を全て算出する。例えば、適合度Aは次のように算出することができる
ＡＸＰａｔｈ＝ｋ１＊Ｐ＋ｋ２＊Ｓ－ｋ３＊Ｌ
ここで、各パラメータは以下の通りである。
ｋ１、ｋ２、ｋ３：定数係数
Ｐ：着目中のＸＰａｔｈ式の親シーケンス（parent sequence）を辿ったときに、存在するノードに至るまでのステップ数。最悪でもbodyノードは共通であるため、必ず値を持つ。
Ｓ：着目中のＸＰａｔｈ式の兄弟シーケンス（sibling sequence）の距離。（親ノード）／ｔｒ［４］が存在せず（親ノード）／ｔｒ［３］が存在する場合距離を１、（親ノード）／ｔｒ［２］が存在する場合２と定義する。兄弟ノードが存在しない場合、Ｐの対象となる存在する親方向のノードにおける子ノード内での兄弟ノードを算出。対象となるノードが元々ファーストノード（／タグ［１］）である場合、この値には一定のデフォルト値を割り当てる。
Ｌ：着目中のＸＰａｔｈ式が指しているノードのツリー長。この距離が長い場合、Ｐの値などで不利になることが考えられるため、負の係数によって相殺する。
以上のような計算によって各ＸＰａｔｈ式の適合度が算出され、最終的に得られた各ＸＰａｔｈ式の適合度の総和をＸＰａｔｈセットの適合度とする。
ＡＸＰａｔｈセット＝ΣＡＸｐａｔｈ
最後に得られたＡの値が最も小さいＸＰａｔｈセットを選択することで最も適合するＸＰａｔｈセットを選択することができる。
【０１０９】
ただし、このような拡張を行った場合、ＸＰａｔｈセットが成立する場合の計算量に変化はないが、成立するＸＰａｔｈセットが存在しなかった場合の計算量が大幅に増加する可能性がある。そのため、全体のパフォーマンスは、成立するＸＰａｔｈセットが存在しない入力の割合に応じて低下してしまう。このような事態を防ぐためには、適合度算出のアルゴリズムを単純化し、高速に算出可能な「曖昧性」にとどめることが必要と考えられる。
【０１１０】
以上のように、第１、第２の実施の形態（以下、本実施の形態）では、多数のＸＰａｔｈ式の評価結果を効率よく求めることによって、ＸＭＬ文書に対する処理の速度を向上させることができる。例えば、ＸＳＬＴプロセッサに本実施の形態を組み込むことによって、スタイルシートに多数のＸＰａｔｈ式が含まれる場合の処理速度の向上が可能になる。
また、ＸＰａｔｈを、ＸＭＬ文書に対して特定の部分を持つかどうかのチェックをするために使うことによって、利用者毎に指定した配信条件でのニュース送信や、コンテンツ毎

に指定した条件でのコンテンツと利用者とのマッチングを、ＸＭＬをベースとして行うことができる。この場合、従来はＸＰａｔｈ式の数の増加に伴って処理時間が増大するので、大規模なサービスを行うことができなかったが、本実施の形態によって、処理時間の増大を抑えることができるため、かかるサービスを実現することが可能になる。

【０１１１】

さらにまた、ネットワーク上のウェブページ（ＨＴＭＬ文書）を所定のアノテーションに基づいてトランスコーディングするシステムにおいて、予め用意された多くのアノテーションパターンの中からトランスコーディングしようとするウェブページに適用可能なアノテーション（当該ウェブページ中のエレメントを正しく指すＸＰａｔｈセットを持ったアノテーション）を検索する場合に、本実施の形態を用いることにより、ＸＰａｔｈセットの評価に要する時間を短縮し、高速にアノテーションの検索処理を実行することが可能となる。

【０１１２】

なお、本実施の形態によるＸＭＬ文書処理システムは、複数のＸＰａｔｈ式があるときに、それらの中に類似なものや重複するもの、あるいは依存関係によって他のＸＰａｔｈ式の評価結果によっては評価を要しないものといった冗長な部分が含まれることを利用して効率を向上させている。したがって、そのような冗長な部分がない場合には効率が向上しないこととなる。

しかし、この場合でも、ロケーション・パスの計算はノードセットから別のノードセットを再帰的に求めることで実現されるのでXalanなどをそのまま使った従来の方式と同じ手続きで処理されることになる。また、１つの比較演算子はエントリ数１のテーブル検索で実現されるが、両者ともにノードセットの再構成に比べると無視できるぐらいに高速に実行できる。さらに、述部が決められたパターンでない場合には最適化の対象にならないが、その場合のテストはXalanなどをそのまま使った従来の方式と同じ手続きで計算を行えばよい。

したがって、冗長な部分がなく最適化できないＸＰａｔｈ式の集合でも、従来の方式とほぼ同等の効率でＸＰａｔｈ式の評価を行うことができる。そして、最適化可能なもの（相互に類似するＸＰａｔｈ式）が増えるにしたがって処理の実行効率が向上することとなる。

【０１１３】

【発明の効果】

以上説明したように、本発明によれば、ＸＭＬ文書に対して複数のＸＰａｔｈ式を評価する場合の実行効率を向上させることができるという効果がある。この効果は、当該複数のＸＰａｔｈ式の中に冗長な部分が増えれば増えるほど顕著となる。

【図面の簡単な説明】

【図１】　第１の実施の形態によるＸＭＬ文書処理システムを実現するのに好適なコンピュータ装置のハードウェア構成の例を模式的に示した図である。

【図２】　第１の実施の形態によるＸＭＬ文書処理システムの機能を説明する図である。

【図３】　第１の実施の形態のデータ構造生成部にて生成されるデータ構造を示す図である。

【図４】　第１の実施の形態の処理対象であるＸＭＬ文書の例を示す図である。

【図５】　幅優先で複数のＸＰａｔｈ式を評価する場合の評価実行部による処理手順を説明するフローチャートである。

【図６】　第１の実施の形態で評価されるＸＰａｔｈ式Ｐ７、Ｐ８、Ｐ９を示す有向非循環グラフを示す図である。

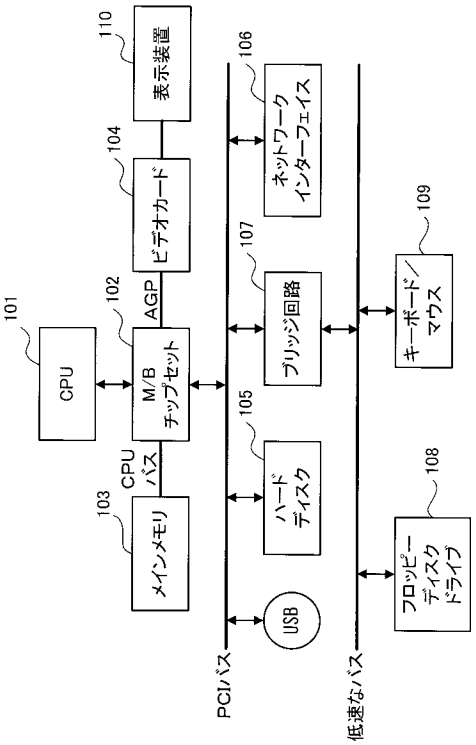
【図７】　第１の実施の形態においてハッシュテーブルを用いた評価処理の流れを説明するフローチャートである。

【図８】　第１の実施の形態のデータ構造生成部によるデータ構造の生成方法を説明するフローチャートである。

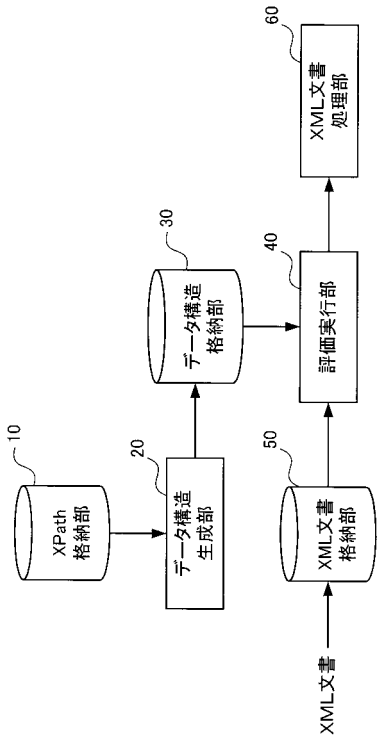
【図９】　図３のデータ構造にＸＰａｔｈ式Ｐ４が追加された状態のデータ構造を示す図

である。
【図１０】　データ構造から所定のＸＰａｔｈ式を除去する際のデータ構造生成部の処理を説明するフローチャートである。
【図１１】　図９のデータ構造からＸＰａｔｈ式Ｐ３を除去した状態のデータ構造を示す図である。
【図１２】　ＸＰａｔｈ式Ｐ１１、Ｐ１２、Ｐ１３、Ｐ１４を表現するデータ構造を示す図である。
【図１３】　処理対象であるＸＭＬ文書を示す図である。
【図１４】　実施例におけるＸＰａｔｈ式の評価に要した実行時間を示す図である。
【図１５】　第２の実施の形態によるウェブページ処理システムの機能を説明するブロック図である。
【図１６】　ＸＰａｔｈセットとこれに含まれるＸＰａｔｈ式とを一覧表示した図表である。
【図１７】　図１６に示したＸＰａｔｈ式のうち重複するものを整理して識別情報（ＸＰａｔｈＩＤ）を割り当てた様子を示す図である。
【図１８】　図１７に示したＸＰａｔｈ式のうち複数のＸＰａｔｈセットに共通するＸＰａｔｈ式を抽出した様子をグラフ構造で示した図である。
【図１９】　連結ＸＰａｔｈ式を展開後のＸＰａｔｈ式の依存関係をツリー構造で示す図である。
【図２０】　図１９に示したＸＰａｔｈ式の依存関係に基づいて連結ＸＰａｔｈ式を簡略化した様子を示す図である。
【図２１】　図２０に示したＸＰａｔｈ式の依存関係に基づいてＸＰａｔｈ式を省略した様子を示す図である。
【図２２】　図２１に示したＸＰａｔｈ式の依存関係ツリー中のＸＰａｔｈ式に対して再結合可能なものを再結合する様子を示す図である。
【図２３】　図２２に示した依存関係ツリーの各ノード（ＸＰａｔｈ式）に優先順位を付加した様子を示す図である。
【図２４】　図２３に示した優先順位に従ってＸＰａｔｈ式を並べた図表である。
【図２５】　依存関係に基づく簡略化、省略の後、この分割処理を経て得られたＸＰａｔｈ式のリストを示す図表である。
【図２６】　図２５に示したＰＸ１、ＰＸ２、ＰＸ３を含めた依存関係ツリーを示す図である。
【図２７】　図２６に示した依存関係ツリーの各ノード（ＸＰａｔｈ式）に、上記の規則に従って優先順位を付加した様子を示す図である。
【図２８】　図２７に示した優先順位にしたがってＸＰａｔｈ式を並べた図表である。
【図２９】　第２の実施の形態による最適化が可能なＸＰａｔｈ式の構造を説明する図である。
【符号の説明】
１０、１５１０…ＸＰａｔｈ格納部、２０、１５２０…データ構造生成部、３０、１５３０…データ構造格納部、４０、１５４０…評価実行部、５０…ＸＭＬ文書格納部、６０…ＸＭＬ文書処理部、１０１…ＣＰＵ、１０３…メインメモリ、１０５…ハードディスク、１０６…ネットワークインターフェイス、１０９…キーボード／マウス、１５５０…文書格納部、１５６０…文書処理部

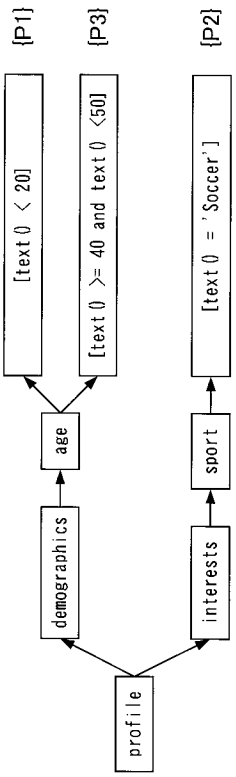
【図１】



【図２】



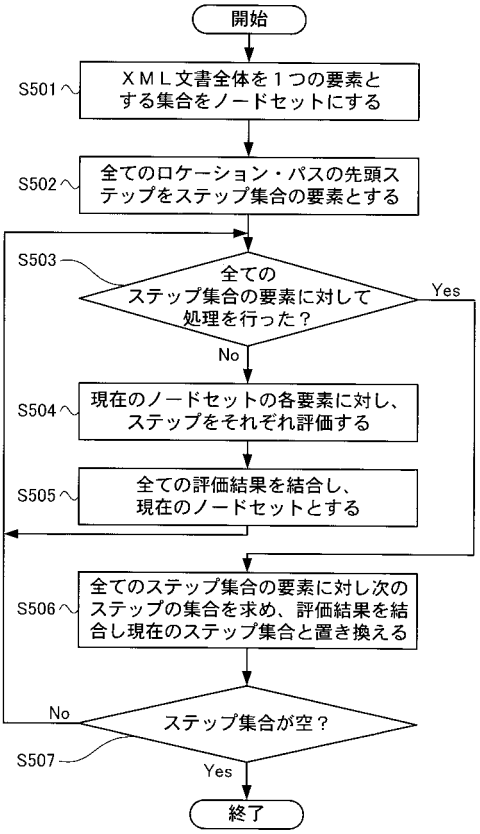
【図３】



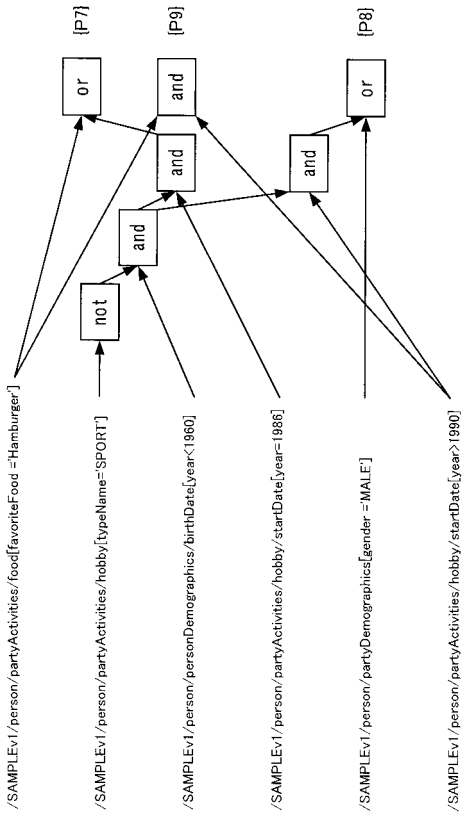
【図４】

```
D1:
<profile>
    <name>Alan</name>
    <demographics>
        <age>35</age>
    </demographics>
    <location>
        <city>Osaka</city>
    </location>
    <interests>
        <sport>Soccer</sport>
        <music>Classical</music>
        <book>History</book>
    </interests>
</profile>
```

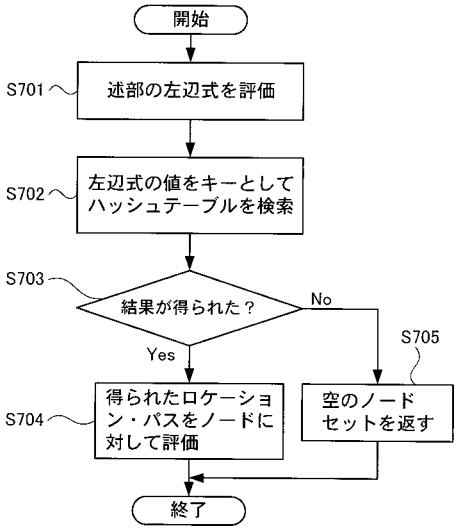
【図５】


開始
S501
ＸＭＬ文書全体を１つの要素とする集合をノードセットにする
S502
全てのロケーション・パスの先頭ステップをステップ集合の要素とする
S503
全てのステップ集合の要素に対して処理を行った？
Yes
No
S504
現在のノードセットの各要素に対し、ステップをそれぞれ評価する
S505
全ての評価結果を結合し、現在のノードセットとする
S506
全てのステップ集合の要素に対し次のステップの集合を求め、評価結果を結合し現在のステップ集合と置き換える
No
ステップ集合が空？
S507
Yes
終了

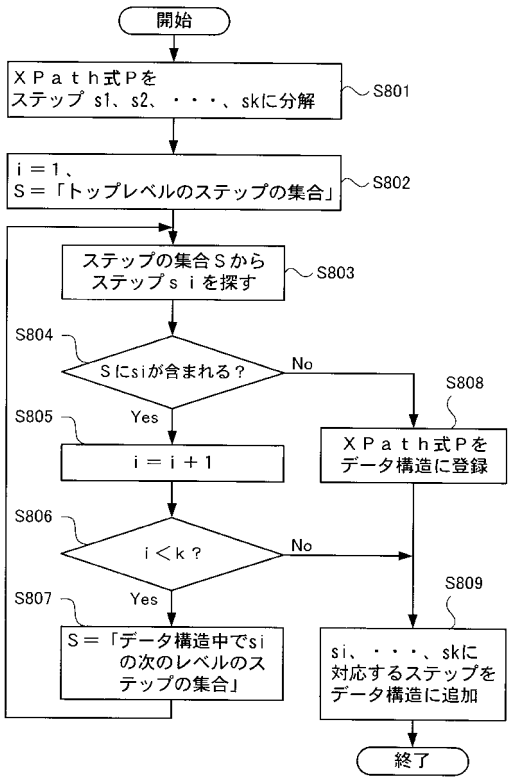
【図６】


[P7]
[P9]
[P8]
or
and
and
and
and
or
not
/SAMPLEv1/person/partyActivities/food[favoriteFood ='Hamburger']
/SAMPLEv1/person/partyActivities/hobby[typeName='SPORT']
/SAMPLEv1/person/personDemographics/birthDate[year<1960]
/SAMPLEv1/person/partyActivties/hobby/startDate[year=1986]
/SAMPLEv1/person/partyDemographics[gender ='MALE']
/SAMPLEv1/person/partyActivities/hobby/startDate[year>1990]

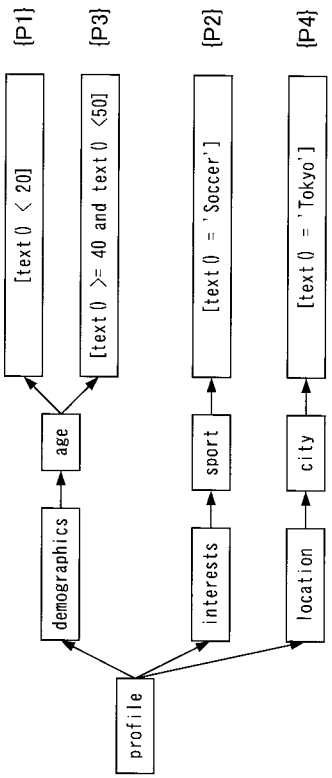
【図７】


開始
S701
述部の左辺式を評価
S702
左辺式の値をキーとしてハッシュテーブルを検索
S703
結果が得られた？
No
Yes
S705
S704
得られたロケーション・パスをノードに対して評価
空のノードセットを返す
終了

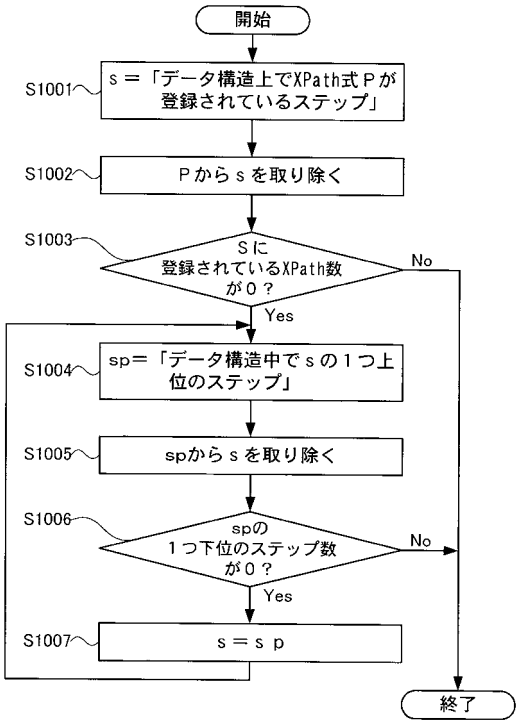
【図８】


開始
ＸＰａｔｈ式Ｐをステップ s1、s2、・・・、skに分解
S801
ｉ＝１、Ｓ＝「トップレベルのステップの集合」
S802
ステップの集合Ｓからステップｓｉを探す
S803
S804
Ｓにsiが含まれる？
No
Yes
S808
S805
ｉ＝ｉ＋１
ＸＰａｔｈ式Ｐをデータ構造に登録
S806
ｉ＜ｋ？
No
Yes
S809
S807
Ｓ＝「データ構造中でsiの次のレベルのステップの集合」
si、・・・、skに対応するステップをデータ構造に追加
終了

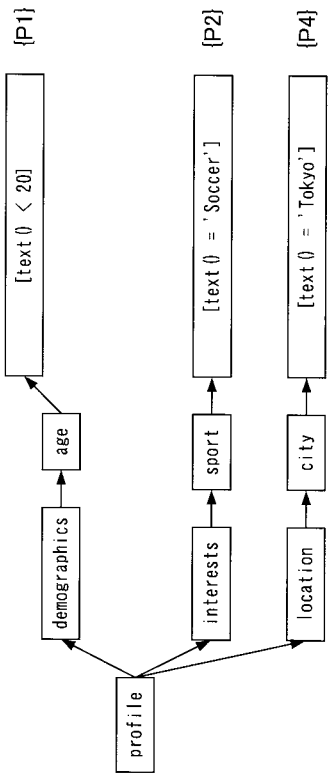
【図９】



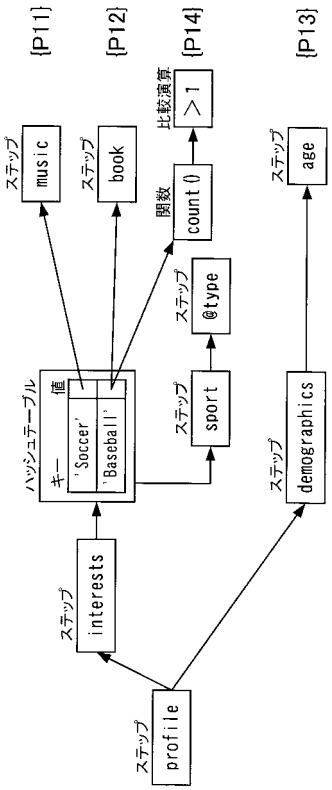
【図１０】



【図１１】



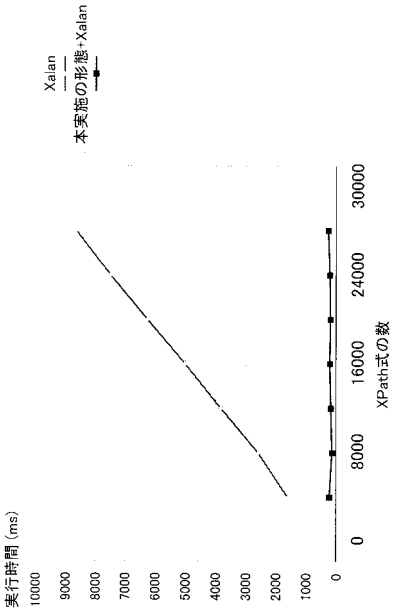
【図１２】



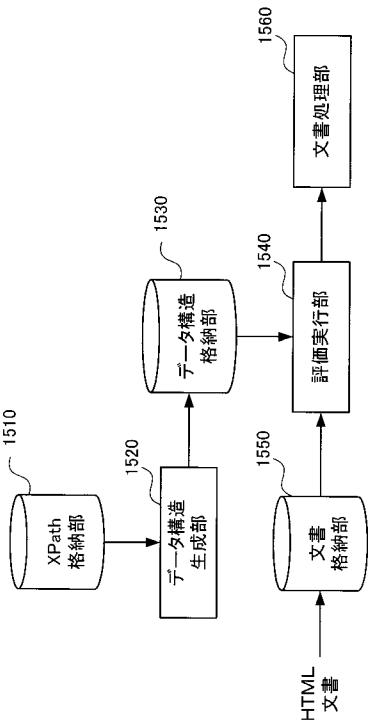
【図１３】

```
<profile>
  <demographics>
       <age>19</age>
  </demographics>
  <interests>
       <sport type@='Baseball'/>
       <book>History<book/>
   </interests>
</profile>
```

【図１４】



【図１５】



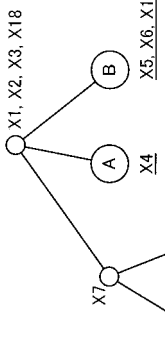
【図１６】

XPath ID	XPath セット	XPath
1	A	/html[1]/body[1]/table[1]/tbody[1]/tr[1]|/html[1]/body[1]/table[1]/tbody[1]/tr[2]
2		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]
3		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
4		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]
5		/html[1]/body[1]/table[3]
6	B	/html[1]/body[1]/table[1]/tbody[1]/tr[1]|/html[1]/body[1]/table[1]/tbody[1]/tr[2]
7		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]
8		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
9		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[6]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[8]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[7]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[1]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[8]
10		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/script[1]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/b[1]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/br[1]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[1]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[1]
11		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[3]
12		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[4]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[5]
13		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[7]|/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[1]]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[2]]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[12]
14		/html[1]/body[1]/table[3]
15	C	/html[1]/body[1]/table[1]/tbody[1]/tr[1]|/html[1]/body[1]/table[1]/tbody[1]/tr[2]
16		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]
17		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
18		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[1]
19		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]
20		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[2]
21		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[3]
22		/html[1]/body[1]/table[3]
23	D	/html[1]/body[1]/table[1]/tbody[1]/tr[1]|/html[1]/body[1]/table[1]/tbody[1]/tr[2]
24		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]
25		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
26		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[1]
27		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
28		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[2]
29		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[2]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[3]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[4]
30		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[2]/table[1]
31		/html[1]/body[1]/table[3]

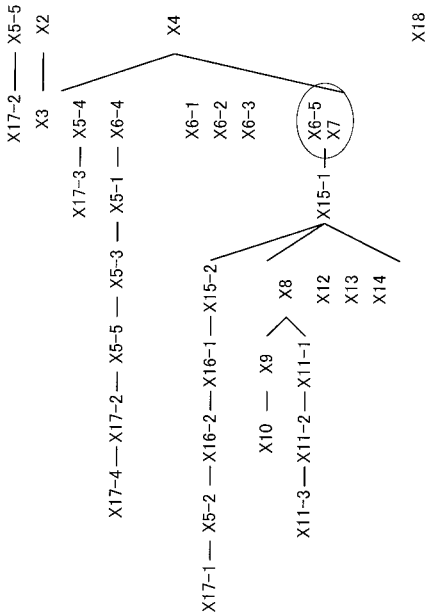
【図１７】

XPath ID	XPath
X1	/html[1]/body[1]/table[1]/tbody[1]/tr[1]/html[1]/tbody[1]/table[1]/tbody[1]/tr[2]
X2	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]
X3	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]
X5	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[6]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[6]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[7]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[8]
X6	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/script[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/b[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/br[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[1]
X7	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]
X8	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]
X9	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
X10	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[2]
X11	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[2]/tbody[1]/tr[2]/td[1]/table[2]/tbody[1]/tr[2]/td[3]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[3]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[4]
X12	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[2]
X13	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[2]/table[1]
X14	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[3]
X15	html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[3]
X16	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[4]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[5]
X17	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[7]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[11]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[2]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[2]
X18	/html[1]/body[1]/table[3]

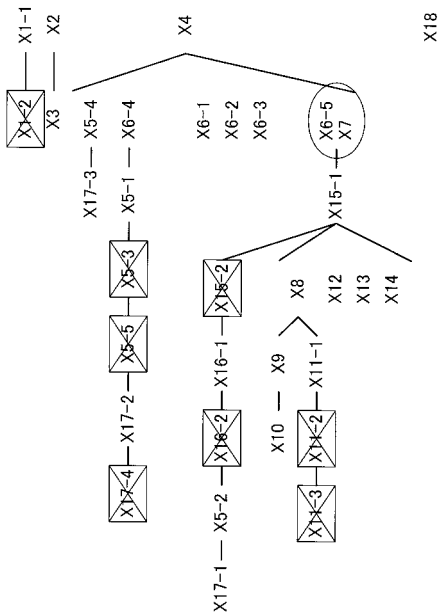
【図１８】



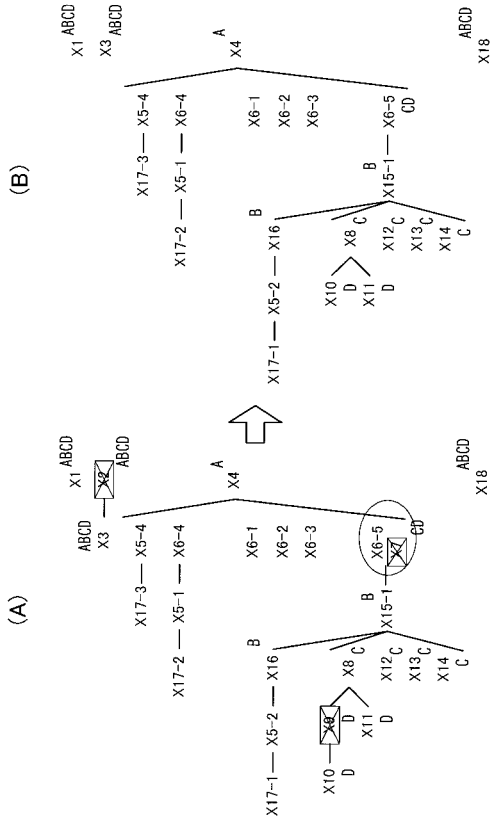
【図１９】



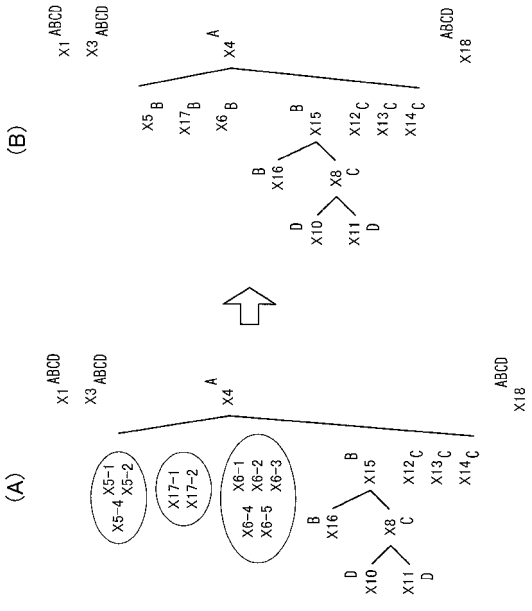
【図２０】



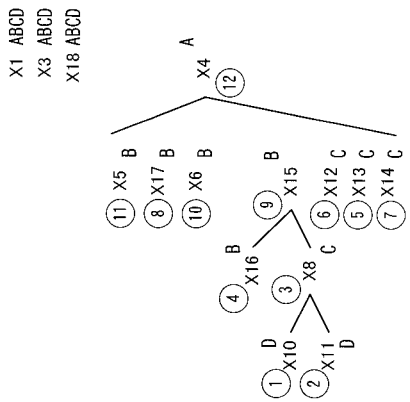
【図２１】



【図２２】



【図２３】



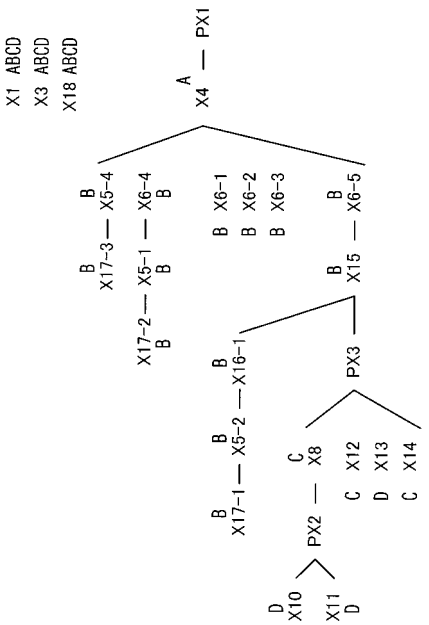
【図２４】

優先順位	XPath ID	依存 XPath	XPath	依存レベル	列一長
1	X10	X8	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[2]	3	15
2	X11	X8	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[2]	3	13
3	X8	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[2]/table[1]	1	11
4	X16	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[2]	1	11
5	X13	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/table[1]	2	11
6	X12	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[3]	1	10
7	X14		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/tbody[1]/table[2]/tbody[1]/tr[1]/td[3]/font[11]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[2]	1	7
8	X17	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[4]	2	7
9	X15	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[2]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[3]	1	7
10	X6	X4	/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/script[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[2]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[1]]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[1]	1	7
11	X5		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[6]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/table[6]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/a[1]/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]/font[8]	1	7
12	X4		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[3]	0	6
必須 XPart	X1		/html[1]/body[1]/table[1]/tbody[1]/tr[1]	0	5
	X3		/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]	0	11
	X18		/html[1]/body[1]/table[3]	0	13

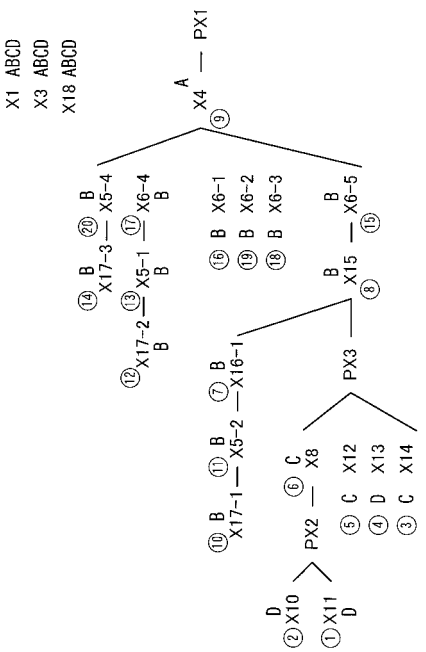
【図２５】

XPath ID	依存XPath	XPath
PX1		{body}/table[2]/tbody[1]/tr[1]
PX2	X8	{X8}/tbody[1]
PX3		{X15-1}/tbody[1]/tr[1]
X10	PX2	{PX2}/tr[1]/td[1]/table[2]
X1-1		{body}/table[1]/tbody[1]/tr[1]
X11-1	PX2	{PX2}/tr[2]
X12	PX3	{PX3}/td[1]/table[2]
X13	PX3	{PX3}/td[2]/table[1]
X14	PX3	{PX3}/td[3]
X15-1	X6-5	{X4}/table[2]
X16	X15-1	{X4}/table[4]
X17-1	X5-2	{X4}/table[7]
X17-2	X5-1	{X4}/font[11]
X17-3	X4	{X4}/a[2]
X18		{body}/table[3]
X3		{PX1}/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]
X4		{PX1}/td[3]
X5-1	X6-4	{X4}/font[6]
X5-2	X16	{X4}/table[6]
X5-4	X4	{X4}/a[1]
X6-1	X4	{X4}/script[1]
X6-2	X4	{X4}/b[1]
X6-3	X4	{X4}/br[1]
X6-4	X4	{X4}/font[1]
X6-5	X4	{X4}/table[1]
X8	PX3	{PX3}/td[1]/table[1]

【図２６】



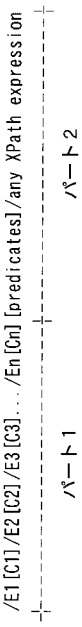
【図２７】



【図２８】

優先順位	XPath ID	依存XPath	XPath	依存レベル
1	X11	PX2	{PX2}/tr[2]	7
2	X10	PX2	{PX2}/tr[1]/td[1]/table[2]	7
3	X14	PX3	{PX3}/td[3]	5
4	X13	PX3	{PX3}/td[2]/table[1]	5
5	X12	PX3	{PX3}/td[1]/table[2]	5
6	X8	PX3	{PX3}/td[1]/table[1]	5
7	X16	X15	{X4}/table[4]	4
8	X15	X6-5	{X4}/table[2]	3
9	X4		{PX1}/td[3]	1
10	X17-1	X5-2	{X4}/table[7]	6
11	X5-2	X16	{X4}/table[6]	5
12	X17-2	X5-1	{X4}/font[11]	4
13	X5-1	X4	{X4}/font[6]	3
14	X17-3	X4	{X4}/a[2]	3
15	X6-5	X4	{X4}/table[1]	2
16	X6-1	X4	{X4}/script[1]	2
17	X6-4	X4	{X4}/font[1]	2
18	X6-3	X4	{X4}/br[1]	2
19	X6-2	X4	{X4}/b[1]	2
20	X5-4	X4	{X4}/a[1]	2
分割XPart	PX3		{X15-1}/tbody[1]/tr[1]	
	PX2	X8	{X8}/tbody[1]	
	PX1		{body}/table[2]/tbody[1]/tr[1]	
必須XPart	X1		{body}/table[1]/tbody[1]/tr[1]	
	X18		{body}/table[3]	
	X3		{PX1}/td[1]/table[1]/tbody[1]/tr[1]/td[1]/table[1]	

【図２９】


/E1[C1]/E2[C2]/E3[C3]...../En[Cn][predicates]/any XPath expression
パート１
パート２

フロントページの続き

(72)発明者　中村　宏明
神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内
(72)発明者　百合山　まどか
神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内
(72)発明者　高木　啓伸
神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

審査官　高瀬　勤

(56)参考文献　特開２０００－２９３１９１（ＪＰ，Ａ）
Aaron Skonnard, XPathを使ってInfosetにアクセスする, Microsoft Developer Network Magazine日本語版, 日本, 株式会社アスキー, ２０００年　8月１８日, 第5号, 第１４９～１５６頁
村上隆一, Mastering XML 第15回, Java WORLD, 日本, (株)IDGジャパン, ２００１年　5月１日, 第5巻, 第5号, 第179-183頁

(58)調査した分野(Int.Cl., ＤＢ名)
G06F　12/00
G06F　17/21-25
G06F　17/30