(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-535034
(P2004-535034A)
(43)公表日　平成16年11月18日(2004.11.18)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 5/00	G06F 5/00	H	5B082
G06F 12/00	G06F 12/00	511A	5J064
H03M 7/40	G06F 12/00	547Z	
	H03M 7/40		

審査請求 有　予備審査請求 有　（全 83 頁）

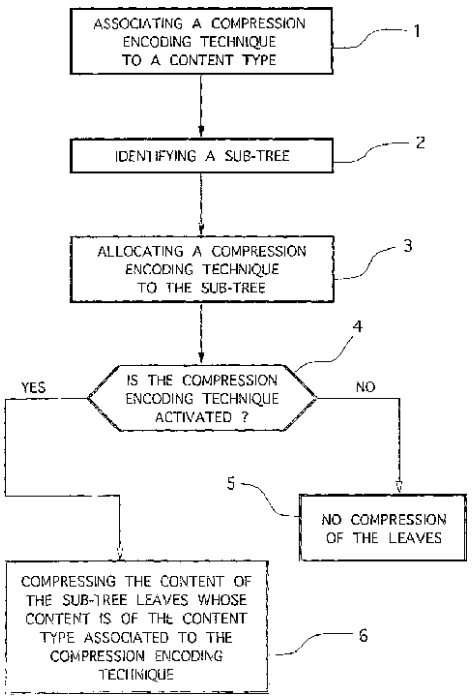
(21)出願番号　特願2003-513248(P2003-513248)
(86)(22)出願日　平成14年7月12日(2002.7.12)
(85)翻訳文提出日　平成16年1月13日(2004.1.13)
(86)国際出願番号　PCT/EP2002/008667
(87)国際公開番号　WO2003/007614
(87)国際公開日　平成15年1月23日(2003.1.23)
(31)優先権主張番号　01460047.2
(32)優先日　平成13年7月13日(2001.7.13)
(33)優先権主張国　欧州特許庁(EP)

(71)出願人　591034154
フランス　テレコム
ＦＲＡＮＣＥ　ＴＥＬＥＣＯＭ
フランス国、７５０１５　パリ、プラス・ダルレ、６
(71)出願人　504015263
グループ・デ・エコール・デ・テレコミュニカシオン
フランス国，７５６３４　パリ・セデクス　１３，リュ・バロー　４６
(71)出願人　504014691
エクスプウェイ
フランス国，７５００３　パリ，リュ・デュ・ポントシュー　１７

最終頁に続く

(54)【発明の名称】階層化ツリーを圧縮する方法及び圧縮されたマルチメディア信号をデコーディングする方法

(57)【要約】
本発明は、マルチメディア信号を記述する階層化ツリーを圧縮する方法に関する。このツリーは、少なくとも２つの明確なタイプのコンテキストに関連付けることができるノード及びリーフを含む。本発明によれば、この方法は、それぞれがコンテントタイプの少なくとも１つに選択的に関連付けられる少なくとも２つの圧縮コード化技術によって、リーフの少なくとも幾つかに対してコンテント圧縮を実行する。



【特許請求の範囲】
【請求項１】
マルチメディア信号を記述する階層化ツリーを圧縮する方法であって、
該ツリーが、データタイプと呼ばれる少なくとも２つの明確な特徴を有するデータに関連付けることができ、ノードとリーフとを含み、
それぞれが前記データタイプの少なくとも１つに選択的に関連付けられる少なくとも２つの圧縮コード化技術によって、前記リーフの少なくとも幾つかに対してデータ圧縮を実行するものである方法。
【請求項２】
少なくとも１つのサブツリーを識別するステップと、該サブツリーに対して前記圧縮コード化技術の１つを割り当てるステップとを含む請求項１に記載の方法。
【請求項３】
前記圧縮コード化技術に関連したタイプのデータをもつ前記サブツリーのリーフに対してのみ、該サブツリーに割り当てられた前記圧縮コード化技術を実行するステップを含み、前記サブツリーの他のリーフはどのような圧縮コード化も受けることはないものである請求項２に記載の方法。
【請求項４】
前記圧縮コード化技術のパラメータ記述を実行する請求項１から請求項３のいずれかに記載の方法。
【請求項５】
前記ツリーの構造を圧縮するステップをさらに含む請求項１から請求項４のいずれかに記載の方法。
【請求項６】
前記ツリーがＭＰＥＧ７規格によるＢｉＭ（２進ＭＰＥＧ）タイプである請求項１から請求項５のいずれかに記載の方法。
【請求項７】
前記圧縮コード化技術の１つが線形の量子化を実行する請求項１から請求項６のいずれかに記載の方法。
【請求項８】
前記圧縮コード化技術の１つが統計的圧縮アルゴリズムを実行する請求項１から請求項７のいずれかに記載の方法。
【請求項９】
前記アルゴリズムがGZipタイプである請求項８に記載の方法。
【請求項１０】
前記アルゴリズムが少なくとも２つのリーフのデータに対応するデータの集合に対して同時に実行されるものである請求項８及び請求項９のいずれかに記載の方法。
【請求項１１】
前記ツリーがＸＭＬ（拡張マークアップ言語）タイプの文書の構造を示すものである請求項１から請求項１０のいずれかに記載の方法。
【請求項１２】
前記階層化ツリーをデコーディングする間に前記サブツリーをスキップできるようにする幾つかの情報を含む少なくとも１つのコーディングコンテキストを前記サブツリーに関連付けるステップをさらに含む請求項１から請求項１１のいずれかに記載の方法。
【請求項１３】
前記幾つかの情報が、
使用された圧縮コード化技術を示す１つの情報、及び／又は、
対応するサブツリーが圧縮されているかどうかを示す１つの情報、及び／又は、
対応するサブツリーがスキップ可能かどうかを示す１つの情報、及び／又は、
使用された圧縮コード化技術の少なくとも１つのパラメータが修正されているかどうかを示す１つの情報

を含むものである請求項１２に記載の方法。
【請求項１４】
請求項１から請求項１３のいずれかの方法に従って圧縮されたマルチメディア信号をデコーディングする方法。
【請求項１５】
前記信号によって送られたコード化コンテキストの情報に基づいて、現在のデコーディングコンテキストをリフレッシュするステップを実行する請求項１４に記載の方法。
【請求項１６】
前記現在のコンテキストが少なくとも１つのデータタイプを定義し、該データタイプのデータを含むリーフに対して該データタイプに関連した圧縮デコーディング技術を実行するステップを含む請求項１５に記載の方法。
【請求項１７】
請求項１から請求項１３のいずれかの方法によって生成された信号。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の技術分野は、データを圧縮する分野である。より正確には、本発明はＸＭＬベースの文書（「拡張マークアップ言語規約（eXtended Markup Language）」）に関する。
【背景技術】
【０００２】
本発明には、特に下記の分野のアプリケーションがあるが、それらに限定されない。
－マルチメディア用アプリケーション
－インデクセーションツール（indexation tool）
－メタデータ（meta-data）取扱いツール
－ＭＰＥＧ－７規格
－ＳＭＩＬ
－常設ＴＶ（TV Anytime）
－第三世代の無線通信（３ＧＰＰ）
【０００３】
ＸＭＬに対する従来の圧縮技術には、幾つかの欠点がある。特に、それらの技術はデータへの高速アクセス、高い圧縮率及び文書のプログレッシブ構造に同時に対応していない。言い換えると、大抵の場合、上記の特徴の１つに対応すると、他の全ての特徴が失われてしまう。
【０００４】
従来の圧縮技術の１つは、ＢｉＭ（２進ＭＰＥＧ）として周知である。そのような技術は、文書の構造、すなわちＸＭＬ文書に関連したツリー構造のノードを２進化することによって、ＸＭＬ文書を圧縮する方法を提供する。このため、ＢｉＭ技術はデータへの高速アクセス、文書のプログレッシブ構造及びスキップ能力を可能にするが、ＢｉＭ技術を実行することによって得られる圧縮率は極めて小さい。
【発明の開示】
【発明が解決しようとする課題】
【０００５】
本発明の目的は、従来技術の欠点を補償することである。より正確には、本発明は、ＸＭＬベースの文書に対する効率的な圧縮技術を提供することを目的とする。本発明は、文書のスキップ能力、高い圧縮率、及びプログレッシブ構造を提供する、ＸＭＬ文書に対する圧縮技術を提供することも目的とする。本発明は、ＭＰＥＧ－７記述子を効率的に圧縮することも目的とする。本発明の別の目的はＸＭＬ文書を圧縮するための方法を実現することであり、この方法は、ＢｉＭ形の技術によって与えられた圧縮率を大いに向上させるが、ＢｉＭにより与えられる機能性と同じ機能性を提供する。
【課題を解決するための手段】

【０００６】
後に出てくる本発明の別の目的と同様に本発明の上記の目的は、本発明に基づいて、マルチメディア信号を記述する階層化ツリーを圧縮する方法によって実現される。このツリーは、少なくとも２つの明確なタイプの内容に結び付けることができるノード及びリーフを含むものである。この場合には、この方法は、それぞれが少なくとも１つのコンテントタイプに選択的に関連付けられた少なくとも２つの圧縮コード化技術によって、少なくとも幾つかのリーフに対してコンテント圧縮（content compression）を実現する。
【０００７】
本発明の好ましい実施形態によれば、そのような方法は少なくとも１つのサブツリー（sub-tree）を識別するステップと、圧縮コード化技術の１つをこのサブツリーに割り当てるステップとを含む。
【０００８】
有利には、そのような方法は、サブツリーのリーフのコンテンツ（又は、内容）が圧縮コード化技術に関連したタイプであり、サブツリーの他のリーフがいかなる圧縮コード化処理も受けないようなサブツリーのみに割り当てられた圧縮コード化技術を実行するステップを含む。
【０００９】
本発明の好ましい特徴によれば、そのような方法は、圧縮コード化技術のパラメータ記述を実行する。
そのような方法は、このツリーの構造を選択的に圧縮するステップも含む。
有利には、このツリーは、ＭＰＥＧ７規格によるＢｉＭ（２進ＭＰＥＧ）形である。
この圧縮コード化技術の１つは、線形の量子化を選択的に実行する。
有利には、この圧縮コード化技術の１つは統計的な圧縮アルゴリズムを実行する。
選択的には、このアルゴリズムはＧＺｉｐ形である。
有利には、このアルゴリズムは、少なくとも２つのリーフの内容に対応するデータの集合に対して同時に実行される。
このツリーは、ＸＭＬ（Extended markup language：拡張マークアップ言語規約）形の文書の構造を選択的に示す。
本発明は、階層化ツリーを圧縮するための上記の方法に基づいて圧縮されたマルチメディア信号をデコードする方法にも関係する。
【００１０】
有利には、そのような方法は、信号によって伝達されたコンテキスト情報（context information）のコード化に基づいて、現在のデコーディングコンテキスト（decoding context）をリフレッシュするステップを実行する。
この現在のコンテキストは、少なくとも１つのコンテントタイプを選択的に定義し、この方法は、リーフがこのコンテントタイプの内容を有するようなこのコンテントタイプに関連した圧縮デコーディング技術を実行するステップを含む。
【００１１】
本発明は、階層化ツリーを圧縮する前述した方法が発生する信号にも関係する。
【００１２】
本発明の他の特徴及び利点は、下記の説明及び以下の図面に照らしてより明確になるであろう。下記の説明は、単に説明するために示すものであり、実施例に限定するものではない。
【００１３】
本発明を実行する方法を詳細に説明する前に、ＢｉＭ技術としてよく知られた従来の圧縮技術の主な特徴を最初に思い出してみる。
【００１４】
文書を容易に理解するために、幾つかの引例をＡｎｎｅｘ９（表１６）の中に集めて、本明細書全体にわたって参照する。本明細書に含まれる全てのＡｎｎｅｘ（補遺）は、本発明の説明の一部をなすものである。

【００１５】
１．従来技術
（はじめに）
図１に示したコーディングコンテキストは、ビットストリームをデコーディングする間に必要なデコーディング情報の集合である。コーディングコンテキストは、定義されるノードのサブツリー全体に適用できる。ツリーの全てのノードにおいて、コーディングコンテキストを修正して、新しいコーディングコンテキストを作成し、対応するサブツリーに適用するようにできる。
【００１６】
コンテキストは、命令が関係するサブツリーに適用できることを特徴とする幾つかの情報を保持できる。現在のところ、ＢｉＭのサブツリーのコーディングフォーマット［１］では、これらの特徴は、（後方及び前方の互換性の特徴を提供するため、）サブツリーまたはコンテキストのスキップ能力及びサブツリーまたはコンテキストに対する複数のスキーマ（schema）のコード化である。最後に、このコンテキスト方法は、帯域幅をセーブするために全てのサブツリー内で無効にすることができる。これはコンテキスト凍結モードである。
【００１７】
このコーディングコンテキスト方法は、文書ツリーの全てのサブツリーの中で最大の柔軟性を提供し、ＢｉＭのコード化方法にプラグ接続されるという拡張可能な特徴を可能にする。
【００１８】
発明者らは、次に、ＢｉＭ内で使用されるコンテキスト方法を説明する。
【００１９】
[現在のコーディングコンテキスト方法]
（定義）
（コーディングコンテキスト（codingContext））
「コーディングコンテキスト」とは、デコーダがビットストリームをデコードするために必要とする情報の集合、すなわちコンテキスト情報である。コーディングコンテキストは、それが定義されているノード及びこのノードに対応するサブツリー全体に適用できる。現在のコーディングコンテキスト（つまり、ある記述の指定されたノードにおいて適用可能なコンテキスト）は、文書内で修正することができる（換言すれば、その下にある情報の集合の修正）。コーディングコンテキストが修正されるたびに、修正された情報の集合を保持する新しいコーディングコンテキストが作成される。全てのコーディングコンテキストは、デコーダがコンテキストに対応するサブツリーのデコーディングを終了したとき、それらを元に戻すためにスタックされるべきものである。
【００２０】
（デコーダ）
ＢｉＭのデコーダは、下記の２つのデコーダからなる、すなわち、
－コンテキストデコーダ（context decoder）。このデコーダは、コンテキスト情報のデコーディング専用である。前述したように、コンテキスト情報は記述の一部ではない。これは幾つかの外部形状、後方及び前方の互換性、高速スキッピング（fast skipping）などを保持する情報の集合である。
－要素デコーダ（element decoder）。このデコーダはＢｉＭ正規デコーダ［１］（BiM regular one）であり、要素情報のデコーディング専用である。
【００２１】
（チャンク）
記述の各要素は、図２に示した３つの部分にコード化される。この場合、ヘッダ部は大きさをゼロにできる２つのチャンクから構成される。
－ＭＣはメタコンテキストのチャンク（metacontext chunk）である。
－Ｃはコンテキストチャンクである。

－エレメント（Element）は要素チャンクであり、これは正規の要素コーディングチャンクである（［１］を参照のこと）
このＭＣメタコンテキストチャンクは、デコーダが次のＣコンテキストチャンクをデコードするために必要とする情報を含む。すなわち、ＭＣチャンクは、Ｃコンテキストチャンクのコンテキストチャンクである。
このＣコンテキストチャンクは、デコーダが次の要素チャンクをデコードするために必要な、現在のコーディングコンテキストの情報の集合を変更できる情報を含む。言い換えると、このＣチャンクは、要素チャンクのコンテキストチャンクである。
【００２２】
（情報の集合）
現在のＢｉＭコーディングコンテキストは、情報の集合、すなわち、次の２つの主な部類に分割できるコンテキスト情報を保持する。
－（この節に含まれた情報がコンテキストデコーディング工程のみに影響する場合の）メタコンテキスト部。
－（この節に含まれた情報が要素デコーディング工程のみに影響する場合の）コンテキスト部。
【００２３】
現在の情報の集合は、下記の変数の集合である。
【表１】

クラス	変数	値	記述
メタコンテクスト	freezing_state	ブーリアン	偽：コンテクストはこのサブツリー内で変更できる。 真：このコンテクストはこのサブツリー内ではもはや変更できない。
コンテクスト	allows_skip	（必須、任意、禁止）	スキッピング機能はこのコンテクスト内では必須、任意又は禁止か？
	schema_mode	（モノラル、マルチ）	現在の要素は複数のスキーマでコード化されるか？
	allows_partial_instantiation	ブーリアン	部分的な例示はこのコンテクスト内で許可されるか？
	allows_subtyping	ブーリアン	サブタイピングはこのコンテクスト内で許可されるか？

定義されたクラスは、（ＭＣメタコンテキストチャンク及びＣコンテキストチャンク等の）前述したチャンクを正確にコーディングする。
【００２４】
[メタコンテキストチャンク]
（定義）
ＭＣメタコンテキストチャンクは、その大きさをゼロにすることができ、デコーダが以下の節で説明されるような次のＣコンテキストチャンクを読み取る必要があるかどうかを知るための情報を含む。
【００２５】
（関係する変数）
【表２】

変数	値	記述
freezing_state	ブーリアン	偽:コンテクストをこのサブツリーの中で変更できる。ＭＣチャンクはビットストリームへとコード化される。 真:このコンテクストはこのサブツリー内ではもはや変更できない。ＭＣチャンクはビットストリームへと変更されない。

【００２６】
（デフォルト値）
freezing_stateのデフォルト値は偽である。すなわち、デフォルトでは、根本的なコンテキストを動的に変更できるようになっている。
【００２７】
（伝搬規則）
新しいコンテキストが作成されるときには、
－freezing_state値は、その父親のコンテキスト（father's context）のfreezing_state値に設定される。
【００２８】
（動的修正規則）
新しいコンテキストを作成するために、記述の各ノードにおいて、
－freezing_state値を偽の値から真の値に切り換えることができる。
【００２９】
（デコーディング規則）
freezing_state値が真の場合には、ＭＣメタコンテキストチャンク（及び次のＣコンテキストチャンク）はビットストリームにコード化されない。真でない場合には、ヘッダ部分のＭＣメタコンテキストチャンクは下記のようにコード化される。
【表３】

MC () {	ビットの#	ニーモニック (Mnemonic)
freeze_type	１～３	ｖｌｃｌｂｆ
}		

context_chunkは偽で初期化されたローカル変数である。
【００３０】
【表４】

フリーズタイプ	含意 (Implication)
110	context_chunk = true and freezing_state = true
10	context_chunk = true
111	context_chunk = false and freezing_state = true
0	context_chunk = false

前の節で述べたように、現在のコンテキストの変数の修正は、新しいコンテキストの作成を意味する。
【００３１】
（コンテキストデコーディング工程に対する影響）
context_chunk値が真の場合には、デコーダは次のＣコンテキストチャンクを読み込む必要がある。
【００３２】

コンテキストチャンク
（定義）
Ｃコンテキストチャンクは、大きさをゼロにすることができ、現在のコンテキスト変数を動的に変更できる情報の集合を含む。これらの変数は、ＢｉＭ要素のデコーディング工程に影響を与えるため、コーディング特性（codingProperties）と呼ばれる。
（関係するコーディング特性）
【表５】

codingProperty	値	記述
allows_skip	（必須、任意、禁止）	スキッピング特性は、このコンテクストでは必須、任意又は禁止されているのか？
schema_mode	（モノラル、マルチ）	現在の要素は複数のスキーマでコード化されるか？
allows_partial_instantiation	ブーリアン	部分的な例示はこのコンテクスト内で許可されるか？
allows_subtyping	ブーリアン	サブタイピングはこのコンテクスト内で許可されるか？

【００３３】
（デフォルト値）
allows_skip変数は、ＦＤＣシステム文書［１］の中で定義されているように、特別な４ビットのビットフィールド（bitfield）の最初の２ビットによって、ビットストリームの始めで初期化される。
allows_partial_instantiation変数は、特別な４ビットのビットフィールドの第３の２ビットによって、ビットストリームの始めで初期化される。
allows_subtyping変数は、特別な４ビットのビットフィールドの第４の２ビットによって、ビットストリームの始めで初期化される。
schema_modeのデフォルト値は、モノラル（mono）である。すなわち、デフォルトでは、ルートのサブツリー／コンテキストは１つのスキーマを用いてコード化される。
【００３４】
（伝搬規則）
新しいコンテキストの作成時に、
－allows_skip値は、その父親のコンテキストのallows_skip値に設定される。
－allows_partial_instantiation値は、その父親のコンテキストの値に設定される。
－allows_subtyping値は、その父親のコンテキストの値に設定される。
－schema_mode値は、そのデフォルト値に設定される。
【００３５】
（動的修正規則）
新しいコンテキストを作成するために、記述の各ノードにおいて、
－allows_skipは動的に修正できる。
－allows_partial_instantiationは動的に修正できない。
－allows_subtypingは動的に修正できない。
－schema_modeは動的に修正できる。
【００３６】
（デコーディング規則）
Ｃコンテキストチャンクは、ＭＣメタコンテキストチャンクが既に存在し、その以前のローカル変数context_chunkが真の場合にのみ存在する。
現在のコンテキストの動的な修正は、ＢｉＭの正規のコード化方式を用いてコード化されるＸＭＬ要素を用いて記述される。ＢｉＭスキーマからの汎用要素modifyContextが使用される。http://www.mpeg7.org/2001/BiMCodingのコーディングスキーマは、付属書類１

に記載されている。
Ｃコンテキストチャンクは、上記のスキーマによるＢｉＭの正規の方式を用いて記述する必要がある。
前述したように、コンテキスト内の現在のコーディング特性の修正は、新しいコンテキストの作成を意味する。このため、Ｃコンテキストチャンクが存在することは、修正されたコーディング特性を保持する新しいコンテキストが作られたことを意味する。
allowsSkip要素がmodifyContext要素の中で例示される場合には、allows_skipの値は新しいコンテキストの中で更新される。
schema_mode要素がmodifyContext要素の中で例示される場合には、schema_modeの値は新しいコンテキストの中で更新される。
【００３７】
（要素デコーディング工程に対する影響）
allows_skip値及びschema_mode値は、スキッピング特性を処理する場合に要素デコーディング工程に影響を与える。この動作は［１］に記載されている。
schema_mode値は、要素が１つのスキーマだけにより又は幾つかのスキーマによりコード化されるのかどうかを知るために要素デコーディング工程に影響する。この仕組みは［１］に記載されている。
allows_partial_instantiation値は、１つの特殊なタイプのpartiallylnstantiatedタイプを要素の可能なサブタイプに加えることによって、要素デコーディング工程に影響を与える。［１］を参照のこと。
allows_subtyping値は、要素デコーディング工程に影響し、要素が多形性（polymorphism）（xsi:typeの属性を有する）又はユニオンの場合には、要素又は属性が異なる可能なタイプを持つことができるようにする。［１］を参照のこと。
【発明を実施するための最良の形態】
【００３８】
２．本発明の説明
２．１　リーフを圧縮するためのフレームワークを提供するコンテキスト方法の拡張
本発明は、新規で興味ある特徴、すなわち、結果として生ずるビットストリームの大きさを減らすために、ある文書のリーフを圧縮する局部圧縮機（local compressor）を使用して、現在のＢｉＭコンテキスト方法を拡張することを提案する。
【００３９】
この節では、局部圧縮機を使用できるように現在のＢｉＭコンテキスト方法を拡張する方法を説明する。これは、典型的な例では、特定の意味論規則、伝搬規則及びコーディング規則とリンクした新しい変数の集合のcodingPropertiesである。従って、この新たなcodingPropertiesの集合によって、現在のコンテキストチャンクが拡張される。
【００４０】
（はじめに）
サブツリーでは、１つ又は数個の指定された単純なタイプのインスタンスがあると、それらは全て１つ又は数個の指定された圧縮機を用いて圧縮することができる。これは基本的には、圧縮機と１つ又は数個の単純なタイプとの間のマッピングを定義する。
さらに、
－場合によっては、圧縮機は幾つかの外部パラメータを必要とすることができる。
－幾つかのサブツリー内で圧縮機を使用し、別のサブツリーでは使用しないようにするために、マッピングを起動又は停止することができる。
最後に、起動または停止するために、マッピングは参照可能でなければならず、このためコンテキストの中で固有の識別子を持つ必要がある。
その結果、各コンテキストは、ゼロ、１つ又は数個のcodecTypeMapperを保持できる。ここで、codecTypeMapperは、識別子、１つ又は数個の単純なタイプ、コーデック、選択可能な外部のコーデックパラメータ及び活性化状態から成る４プレット（4-plet）である。
【００４１】

（定義）
（CodecTypeMapper）
codecTypeMapperは、４プレットであり、
－サブツリーまたはコンテキストの中で固有の参照キーとして使用される識別子と、
－マッピングが適用できる１つ又は数個の単純なタイプと、
－コーデックと、
－任意選択的な外部コーデックパラメータ（コーデックに依存する）と、
－活性化状態と
を備えている。
【００４２】
（識別子）
識別子は、固有の番号であり、明白な方法でコンテキスト内のマッピングを識別する。ＢｉＭのコーディングスキーマは、コンテキスト内のcodecTypeMappersの最大数を３２に制限する。
【００４３】
（単純なタイプ）
あるスキーマにおいて定義された全ての単純なタイプ（simple type）は、全てのコーデックによって先験的に（a priori）コード化することができるが、各コーデックはこの選択を制限できる。例えば、以後文書の中で定義するような線形の量子化器（linear quantizer）は、数値的な単純なタイプと共にのみ使用できる。
【００４４】
単純なタイプは、その名前及びそれが属するスキーマのＵＲＬによって識別される。正しいスキーマを指し示すために、ＸＭＬスキーマのプレフィックス（prefix）を用いるべきである。ＢｉＭコーディングスキーマは、この対をコード化するためにある特殊なタイプを定義する、すなわち、このタイプは整数の対としてコード化されなければならない、第１の整数はわかっているスキーマの現在の数に制限され（この情報の部分は、DecoderConfig部［１］の中でフェッチすることができる）、第２の整数は対応するスキーマの中に存在するグローバルな単純なタイプの数に制限される。
【００４５】
（コーデック）
圧縮機または解凍機（compressor/decompressor）の役目をするコーデック（codec）は、入力ビットを取り入れて出力ビットを書き込むモジュールである。それは幾つかの選択可能な外部パラメータを必要とすることがある。
コーデックは、ＢｉＭのコーディングスキーマの中で定義された非抽象的なコーデックの名前の中の名前によって識別される。上記の節の中で定義された現在のＢｉＭのコーディングスキーマは、どの非抽象的なコーデックも定義しないが、本明細書の２．２節（段落番号[００６１]以下）では定義している。
【００４６】
（活性化状態）
活性化状態はブーリアンフラグである。
【００４７】
（意味論規則）
（CodecTypeMapper）
各コンテキストは、
－ゼロ、１つ又は数個のcodecTypeMapperを保持できる。
－１つ又は数個のcodecTypeMapperを定義できる。
－１つ又は数個の現行のcodecTypeMapperを起動又は停止できる。
あるcodecTypeMapperがコンテキストの中で定義されると、それは全てのそのサブコンテキストの中に残る。
コンテキスト内に存在するcodecTypeMapperを削除又は修正することはできない（活性化

状態は除く）。
【００４８】
（識別子）
マッピングの識別子は、コンテキストの全てのcodecTypeMapperの中で一意的でなければならない。
【００４９】
（単純なタイプ）
codecTypeMapperを１つ又は数個の単純なタイプ及びコーデックと結び付ける場合、このコーデックは単純なタイプ自体及びそれらから生じた全ての単純なタイプをコード化／デコードする。
コンテキストには、コーデックの場合とは異なり、たかだか１つの単純なタイプが存在する必要がある。
【００５０】
（コーデック）
２種類のコーデック、すなわち、メモリ無しのコーデックとコンテキスト式コーデックとがある。
メモリ無しのコーデックは、常に同じ入力バイトを同じバイト出力にコード化するモジュールであり、コーデックの履歴に無関係である。典型的なメモリ無しのコーデックは、線形の限定子である。ＢｉＭのリーフ圧縮（本明細書の２．２節を参照のこと）は、そのようなコーデックを説明している。
コンテキスト式コーデックは、送られた以前のバイトを使用する（このため、コーデックのコンテキストは変更される）モジュールである。そのようなコーデックは、受け取った同じ入力バイトに対して同じ出力バイトを発生しない。典型的なコンテキスト式コーデックは、Ｚｉｐのようなローカルコーデックであり、本明細書の２．２節に説明されている。
メモリ無しのコーデックは、現在のコンテキストのアーキテクチャでは何ら問題を引き起こさないが、コンテキスト式コーデックはスキップ可能なサブツリーの場合には問題を発生する。そのような場合には、この前者がサブツリーをスキップした場合にデコーダを混乱させないように、コンテキスト式コーデックはリセットされる。
【００５１】
（活性化状態）
全てのサブツリー又はコンテキストにおいて、codecTypeMapperを起動又は停止することができる。
この仕組みにより、codecTypeMapperを文書ツリーのより高いレベルで定義し、使用されているサブツリーの中でのみcodecTypeMapperを再定義せずに起動することができる。
【００５２】
（新しいcodingProperty：codecTypeMapper）
この部分では、新しいcodingPropertyを、前述したコンテキスト部の以前の変数の集合に加える。この新しいcodingPropertyはcodecTypeMapperと名付けられ、前の節の中で説明された以前のcodecTypeMapperのリストである。
【００５３】
（関係する新しいcodingProperty）
コンテキストはcodecTypeMapperのリストを保持する。
【００５４】
【表６】

CodingProperties	値	記述
codecTypeMapper[i].identifier	整数	明白な方法でコンテクスト内のcodingProperty を参照するための数値識別子
codecTypeMapper[i].simple_type[j]	整数；整数	２プレットのリスト（スキーマの数値指数、現在のスキーマ内の単純なタイプの数値指数）
codecTypeMapper[i].codec	整数	現在のコーディングコンテクスト方式内のコーデックの数値指数
codecTypeMapper[i].codec_parameters		任意の外部のコーデックパラメータ
codecTypeMapper[i].activation_state	ブーリアン	codingProperty の活性化状態

【００５５】
（新しいデフォルト値）
デフォルトにより、サブツリー又はコンテキスト内にはcodecTypeMapperはない。codecTypeMapperをコンテキスト内で定義する場合には、その識別子、コーデック及びsimple_type値を定義する必要がある。指定されない場合は、新しく定義されたcodecTypeMapperの活性化状態はデフォルトにより真に設定される。すなわち、新しく定義されたcodecTypeMapperはデフォルトにより起動される。
【００５６】
（新しい伝搬規則）
（規則１）
新しいコンテキストの作成時には、codecTypeMapperのリストはその父親のコンテキストのコピーである。
－識別子の値がコピーされる。
－simple_typeの値がコピーされる。
－コーデックの値が規則２に従ってコピーされる。
－codec_parameterの値がコピーされる。
－activation_stateの値がコピーされる。
（規則２）
コーデックの値がコピーされ、下記が成立する場合、つまり、
－父親のコーデックのcodingProperty[i].codecがコンテキスト式コーデックであり、
－現在のコンテキストがスキップ可能である場合には、
デコーダは、父親のコーデックの事例（その値だけでなく）をコピーし、それをリセットすることによって、コーデックの新しい事例を作成することが期待される。
例えば、ZLibコーデックがコピーされ、スキップ可能なノードに入るときに再度初期化される。
【００５７】
（新しい動的な修正規則）
codecTypeMapperのリストを記述の中で下記のように動的に修正できる。
－新しいcodecTypeMapperを定義できる。
－現在の（次に参照可能な）activation_stateのcodecTypeMapperを動的に修正できる。（そのactivation_stateを除き、）現在のcodecTypeMapperを削除することはできず、そのメンバを動的に修正することもできない。
【００５８】
（新しいデコーディング規則）
同じ前の規則はＣコンテキストチャンクのデコーディングに適用するが、補遺２で説明される新しいスキーマが、新しいcodecTypeMapperの動的な修正機能を加えるために使用される。

【００５９】
（情報提供部）
（実施例）
補遺３に示した実施例は、記述する場合の１つの活性化された線形の限定子（本明細書の２．２節を参照のこと）の定義を示す。
【００６０】
補遺４に示した実施例は、記述する場合の１つの非活性化された線形の限定子（本明細書の２．２節を参照のこと）の定義を示す。
【００６１】
２．２　ＢｉＭのリーフ圧縮
異なるコーデックを用いてデータをコード化するために、本発明によって実現された仕組みをここで示す。より正確に言うと、ここで２つの実施例を示す、すなわち、１つは線形の量子化コーデックを使用して例えば浮動小数点の値を圧縮する実施例、及びもう１つはgzipのコーデックを使用して例えばストリング値を圧縮する実施例である。
【００６２】
そのような仕組みはコーディングコンテキストに密接に関連しており、幾つかの他の種類のコーデックを使用できるようにする。さらに、この仕組みは、例えば、ストリップ可能なサブツリーなどのコーディングコンテキストの機能を適切に処理できる。最後に、この仕組みは、別のコーディングコンテキスト内のコーデックを再度使用することができる。
【００６３】
（はじめに）
ＢｉＭのサブツリーのコーディング［１］は、記述のデータリーフを圧縮しない。現在は、リーフ値は、（ＩＥＥＥ７５４のフロート及びダブル（floats and doubles）、ＵＴＦストリングなどの）その種類に関連してコード化される。
【００６４】
多くの場合には、ＢｉＭの圧縮比をその主な機能を失わずに向上させるために、（流線型の構文解析（streamline parsing）、高速スキッピング機能、タイプドデコーディング（typed decoding）などの）線形の量子化又は統計的な圧縮のような幾つかの伝統的な圧縮技術を使用することが有用である。
【００６５】
次に、より優れた圧縮比を実現するために、２．１節で説明されたコンテキストコーディングの方法の中で、ある文書のデータリーフの圧縮を行うことができる方法を示す。
【００６６】
２．２．１　線形量子化
（定義）
線形量子化は、情報源が既知であり、このため損失を制御できる場合に、ビットストリーム内のコード化された数の大きさを減らすための通常の損失の多い方法である。
一例として、サンプリングされたオーディオ信号の包絡線は正確なビットサイズの量子化を有するとして良く知られており、この技術は、ＭＰＥＧ－７のオーディオ記述をコーディングするために効果的に使用することができる。
νが実数の場合には、νはnbitsビットのν_qにより下記のようにコード化することができる。
【数１】

$$\nu_q = \frac{\nu - \nu_{min}}{\nu_{max} - \nu_{min}}(2^{nbits} - 1)$$

ここで、
－ν_qは、νの量子化及びコード化された値である。
－nbitsは、ビットに必要な精度である。

－ν_{min}は、νが到達できる最小の包含的値である。
－ν_{max}は、νが到達できる最大の包含的値である。
【００６７】
νからのデコード値は、
【数２】

$$\bar{v}$$

であり、次式で与えられる。
【数３】

$$\bar{v} = v_{min} + v_q \frac{v_{max} - v_{min}}{2^{nbits} - 1}$$

ここで、
【数４】

$$\bar{v}$$

は、νのデコードされた近似値である。
【００６８】
（コンテキスト方法との統合：LinearQuantizerCodec）
線形の量子化は、本明細書の２．１節の中で説明されたコーディングコンテキスト方法で定義されたように、コーデックとして使用できる。この方法を用いて、線形の量子化を記述のどのようなサブツリーの中でも、望ましい単純なタイプの数値データのリーフに対して適用できる。
【００６９】
このように使用することにより、線形の量子化コーデックに関連するコーディングコンテキスト方法は、ＭＰＥＧ－４のＢＩＦＳ［３］の中で使用されるQuantizationParameterノードとして動作する。
【００７０】
（適用可能な単純なタイプに対する制約）
本明細書の２．１節のコーデックに対する定義によれば、このコーデックはメモリ無しのコーデックであり、これは全ての極小の数値及び極小でない数値の単純なタイプに適用できる。そのＸＭＬスキーマの基本タイプは、フロート、ダブル又はデシマルである。
【００７１】
（コーデックの外部パラメータ）
線形量子化器のコーデックは、下記の３つの必須のパラメータである
－上記のnbits変数であるbitSizeと、
－上記のν_{min}変数であるminInclusiveと、
－上記のν_{max}変数であるmaxInclusiveと
を必要とする。
【００７２】
（コーデックのスキーマの定義）
線形量子化のコーデックは、抽象的なCodecTypeタイプ（２．１節を参照のこと）に基づいて、コーディングコンテキストのnamespaceのＵＲＬ xmlns:cc=http://www.mpeg7.org/2001/BiMCodingにおける補遺５の中で与えられたスキーマで定義されたタイプLinearQuantizerCodecTypeの新しいコーデックである。
【００７３】
（コード化（情報提供））
値νの数値データのリーフは、nbitsビットの符号のない整数ν_qにより下記の式のように

コード化される。
【数５】

$$v_q = \frac{v - v_{min}}{v_{max} - v_{min}}(2^{nbits} - 1)$$

【００７４】
（デコーディング）
nbitsビットでコード化された符号のない整数v_qは、下記の式のように、
【数６】

$$\bar{v}$$

としてデコードされるべきである。
【数７】

$$\bar{v} = v_{min} + v_q \frac{v_{max} - v_{min}}{2^{nbits} - 1}$$

【００７５】
（実施例（情報提供））
補遺６で示された実施例は、記述における線形量子化器の定義を示す。
【００７６】
２．２．２　統計的な圧縮
伝統的な損失のない統計的な圧縮アルゴリズムは、コーディングコンテキスト方法（２．１節を参照）の中で定義されたようなコーデックとして使用できる。この方法を用いて、どの記述のサブツリーの中でも、望ましい単純なタイプのデータリーフを効率的に圧縮できる。
このコーデックは、特に記述が多くの反復的な又は同様のストリングを含む場合には、ビットストリームの大きさを著しく減少させるために有用である。
【００７７】
（定義）
（Zip又はGZipなどの）伝統的な損失のない統計的な圧縮アルゴリズムをＢｉＭの中で用いて、記述の任意のリーフを圧縮することができる。
しかし、大抵の場合、データリーフが１０文字よりも少ない短いストリングの場合には、通常の統計的な圧縮アルゴリズムは大きなルックアヘッドバッファ（lookahead buffer）を必要とするため、このアルゴリズムの性能は良くない。
最適な圧縮比を実現するために、文書のリーフは圧縮する前に、小さいバッファにバッファする必要がある。次の節は、根本的な損失のない統計的な圧縮アルゴリズムに依存するそのようなバッファ付き統計的コーダ（a buffered statistical coder）を定義する。
【００７８】
（バッファ付き統計的コーダの定義）
バッファ付き統計的コーダは、一般的な下記の基本的な方法である、
－ストリームの圧縮又は解凍動作を初期化するinitialize_stream()と、
－コーダの現在の統計的なモデルをリセットするreset_model()と、
－入力の解凍されたバイトを取り、圧縮ストリームの中に入れるfeed_input_bytes()と、
－既に処理された入力バイトを圧縮し、対応する圧縮された出力バイトを出すことによって、圧縮ストリームをフラッシュするflush_output_bytes()と、
－指定された量の圧縮された入力バイトを取り、対応する解凍された出力バイトを出すことによってそれらをデコードするdecompress_input_bytes()と
を必ず含む根本的な統計的コーダに依存する。
バッファ付きコーデックは、bufferSizeのバイト長、すなわちバイトアレイバッファ（byte array buffer）のＦＩＦＯ構造を有する。

エンコーダ側から見ると、bufferSize値はフラッシングの前にエンコーダがどの位の入力バイトを処理できるかを示す。デコーダ側からすると、これは根本的な統計的コーダのＡＰＩを通してビットストリームをデコードするために必要な最小のバッファ寸法である。このバッファはfillingLevel変数も有する。この変数は、バッファのバイトにおける実際の充填レベルを含む。

【００７９】

（統計的なコーダとしてのZLib ＡＰＩの使用）

GZip圧縮方式の中で使用されるZLibの公開ライブラリのＡＰＩ［４］は、文書リーフに対して統計的圧縮を行うための効率的で有用なＡＰＩを提供する。

【００８０】

ZLib ＡＰＩは、下記のマッピングを用いて前の一般的な方法を実行する。

－initialize_stream()は、Z_DEFAULT_COMPRESSIONの効率値のパラメータを用いて、ZLibのinflateInit()又はdeflateInit()機能によってマッピングすることができる。

－reset_model()は、ZLibのinflateEnd()又はdeflateEnd()コール及び次のinitialize_stream()コールを用いてマッピングすることができる。

－feed_input_bytes()は、Z_NO_FLUSHパラメータを用いてZLibのdeflate()方法によりマッピングすることができる。

－flush_output_bytes()は、Z_SYNC_FLUSHパラメータを用いてZLibのdeflate()方法によりマッピングすることができる。

－decompress_input_bytes()は、ZLibのdeflate()方法によりマッピングすることができる。

【００８１】

ZLibのバッファ付きコーデックは、［４］で定義されたように、Z_DEFAULT_COMPRESSIONの効率値を用いて初期化されるべきである。このことは、メモリのフットプリント（footprint）についての要求事項と圧縮効率との間の良いトレードオフになる。

【００８２】

（コンテキスト方法との統合：ZlibCodec）

この節では、コーディングコンテキスト方法の中で定義された、ZLib ＡＰＩに依存する、上記のバッファ付き統計的コーダの統合について説明する。

【００８３】

（適用可能な単純なタイプに対する制約）

２．１節のコーデックに対する定義によれば、このコーデックはコンテキスト式コーデックであり、これは全ての極小のストリングタイプ及び極小でないストリングタイプに適用できる。ZLibCodecは、［１］の中で説明したように、文書のリーフの根本をなす基本的なコード化に依存している。例えば、intリーフは３２ビットの符号のない整数によりコード化され、stringリーフはＵＴＦ－８のコード化により、float及びdoubleリーフはＩＥＥＥ７５４形式でコード化される。このため、ZLibCodecは、コード化されたリーフを圧縮できる。

【００８４】

（コーデックの外部パラメータ）

バッファ付きのZLibコーデックは、根本的なZLibの効率がZ_DEFAULT_COMPRESSIONに設定され、bufferSizeパラメータがデコーダ側からは必要とはされないため、どのような外部のパラメータも必要としない。

【００８５】

（コーデックのスキーマの定義）

このZLibコーデックは、抽象的なCodecTypeタイプ（２．１節を参照のこと）に基づいて、補遺７のコーディングコンテキストのnamespaceの中で説明されたスキーマによって定義されたZLibCodecTypeタイプの新しいコーデックである。

【００８６】

（コード化（情報提供））

コーデックの活性化又はインスタンシエーションにおいては、
－ＦＩＦＯのバッファ構造はクリアであると想定され、そのfillingLevelは０に設定される。
－グローバル変数referencable_chunkは０に初期化される。
このreferencable_chunkは、エンコーダが保持しなければならない参照可能なビットのチャンクを含むべきである。その理由は、その値がコード化工程の間に後で知られるからである。通信関数signal_reference_chunk_known()は、このチャンクが知られた場合には呼び出すことができる。
全てのゼロでないチャンクのバイトの大きさは、［１］で定義されたように、標準的な符号無しの無限の整数４＋１コーディング（infinite integer 4+1 coding）を用いて、flush_output_bytes()呼出しの間にチャンク自身の前に書き込まれるべきである。
入力リーフ（input leaf）は、その基本のタイプに関連したテキストのリーフのコード化された値である。このリーフのバイトの長さは、フィールドleaf.lengthによって与えられる。一例［１］を挙げれば、stringリーフは、ストリングのバイトの大きさが前に来る（無限整数のコーディング［１］を用いてコード化された）ＵＴＦ－８のコードであり、doubleリーフは、対応するＩＥＥＥ７５４規格の６４ビット値である。

【００８７】

下記のencode_leaf関数は、出力バイトのチャンク内のリーフをコード化することができる。

```
chunk encode_leaf(chunk leaf) [
    while (leaf is not empty) [
        if (fillingLevel + leaf.length < bufferSize) [
            feed_input_bytes(leaf,leaf.length)
            fillingLevel = fillingLevel + leaf.length
            if (referencable_chunk is null) [
                referencable_chunk = new chunk
                return referencable_chunk
            ] else [
                return nil_size_chunk
            ]
        ] else [
            remaining_bytes = bufferSize - fillingLevel - leaf.length
            feed_input_bytes(leaf,remaining_bytes)
            referencable_chunk = flush_output_bytes()
            signal_reference_chunk_known()
            fillingLevel = 0
            leaf = leaf.remove_beginning_bytes(remaining_bytes)
            referencable_chunk = null
        ]
    ]
]
```

【００８８】
（デコーディング）
string_fifoをストリングＦＩＦＯとする。
下記の方法get()及びput()は、それぞれ、ＦＩＦＯから要素を取り出し、ＦＩＦＯに要素を置く。
ＦＩＦＯが空かどうかを知る方法はEmpty()信号である。
subをサブストリングを取り出す関数とする。
concatを連結関数とする。
getData()を、リーフからのde-gzipデータを保持するchar[]を戻す関数とする。
split(char[] , char sep, Fifo, char[] remainder)を各セパレータ「sep」において文字のアレイを分割し、この分割されたストリング要素をＦＩＦＯに保管し残りを戻す方法とする。

```
[
    if (data==null) return;
    int BEGIN=0;
    for (int I=0;I(data.length;I++)
    [
        if (data[I]==sep)
        [
            char[] str= concat(remainder, sub(data,BEGIN,I));
            put(string_fifo, str);
            BEGIN=I+1;
        ]
    ]
    if(BEGIN!=data.length)
    [
        //there's a remainder.
        remainder = sub(data,BEGIN,data.length);
    ]
]
String decode()
[
    if (isEmpty(string_fifo)
    [
        data = getData();
        split(data,0x00,string_fifo,remainder);
    ]
    return get(string_fifo);
]
```

初期化されたとき、string_fifoは空である。
コーデックの活性化又はインスタンシエーションにおいては、
－ＦＩＦＯの構造はクリアであると想定され、そのnumberOfLeavesは０に設定される。
－変数first_chunkは真に設定される。
セパレータのバイトを0x00とする。
下記のdecode_leaf関数は、ビットストリームからの圧縮されたリーフをデコードすることができる。

```
string decode_leaf() [
    if (numberOfLeaves == 0) [
        read_and_decompressed_byte()
        numberOfLeaves = count_number_of_leave_in_buffer()
    ] else [
        ]
]
```

【００８９】
デコーディングは、下記により定義される。
１．ＦＩＦＯが空の場合には、
ａ．コード化データをデコードする。
ｂ．0x00により分離された全ての要素をＦＩＦＯの中にスタックする。
ｃ．最後の文字が0x00でない場合は、未完成のストリングを一時的に保管する。
ｄ．「last_element」が空でない場合は、それをＦＩＦＯ内の第１の要素の先頭に挿入する。
ｅ．このラウンドの未完成のストリングをlast_elementの中に入れる。
ｆ．第１の要素を取り除き戻す。
【００９０】
２．ＦＩＦＯが空でない場合には、第１の要素を取り除き戻す。
これは、「ＦＩＦＯは空ではない」及び「現在のリーフにはコード化されたデータはない」と言うのに等しい。
【００９１】
（実施例（情報提供））
補遺８で与えられた説明は、string及びanyURIタイプでマッピングされたZLibCodecTypeのコーデックを使用する場合の例である。
【００９２】
（結果（情報提供））
下記の図面は、記述（string及びanyURIのＸＭＬスキーマの基本のタイプから得られた記述）のテキストのリーフを圧縮するためにZLibCodecを使用する場合の性能を示す。bufferSize＝２５６バイトのバッファは、コード化工程の間に使用された。使用されたファイルは、ＭＰＥＧ－７のＭＤＳサブグループにより提供された。
【表７】

ファイル名	当初の寸法（バイト）	BiMの寸法（バイト）	ZLibコーデック付きのBiMの寸法（バイト）	ジップ処理されたファイルの寸法（バイト）
mdsExamplesClause11-12.xml	160658	22512	10602	17019
mdsExamplesClause13-15.xml	81133	11627	8538	9698
mdsExamplesClause17.xml	142426	57583	22489	21444
mdsExamplesClause4-7.xml	37208	6536	3748	7623
mdsExamplesClause8-10.xml	8179	2081	1389	2416

階層化ツリーのリーフの内容を圧縮するために、本発明に基づいて実行されたステップを、ここで手早く説明する。
【００９３】
ステップ１は、圧縮コード化技術をあるコンテントタイプに対応付ける工程から成る。例えば、線形の量子化を、浮動小数点の値に対応付けることができる。

【００９４】
ステップ２では、あるサブツリーが検討されたＸＭＬ文書の構造に対応する階層化ツリーの中で識別される。
【００９５】
ステップ３は、圧縮コード化技術を識別されたサブツリーに割り当てる工程から成る。
【００９６】
そして、ステップ４は、圧縮コード化技術を実行しているコーデックが起動されているかどうかを検査する工程から成る。起動されていない場合には、サブツリーのリーフの圧縮（ステップ５）は行われない。
【００９７】
起動されている場合には、本発明は、その内容が（ステップ１において）圧縮コード化技術に対応付けられたコンテントタイプであるような、サブツリーのリーフの内容の圧縮を実行する（ステップ６）。
【００９８】
［補遺１］
【表８】

ANNEX 1

```
<schema targetNamespace="http://www.mpeg7.org/2001/BiMCoding"
    xmlns:cc="http://www.mpeg7.org/2001/BiMCoding"
    xmlns="http://www.w3.org/2000/10/XMLSchema">

    <element name="modifyContext">
        <complexType>
            <sequence>
                <element name="allowsSkip" minOccurs="0">
                    <simpleType>
                        <restriction base="string">
                            <enumeration value="mandatory"/>
                            <enumeration value="optional"/>
                            <enumeration value="forbidden"/>
                        </restriction>
                    </simpleType>
                </element>

                <element name="schemaMode" minOccurs="0">
                    <simpleType>
                        <restriction base="string">
                            <enumeration value="mono"/>
                            <enumeration value="multi"/>
                        </restriction>
                    </simpleType>
                </element>
            </sequence>
        </complexType>
    </element>
</schema>
```

【００９９】
［補遺２］
【表９Ａ】

ANNEX 2

```
<schema targetNamespace="http://www.mpeg7.org/2001/BiMCoding"
xmlns:cc=" http://www.mpeg7.org/2001/BiMCoding"
xmlns="http://www.w3.org/2000/10/XMLSchema"
>
      <element name="context">
            <complexType>
                  <sequence>
                        <element name="allowsSkip" minOccurs="0">
                              <simpleType>
                                    <restriction base="string">
                                          <enumeration
value="mandatory"/>
                                          <enumeration
value="optional"/>
                                          <enumeration
value="forbidden"/>
                                    </restriction>
                              </simpleType>
                        </element>

                  <element name="schemaMode" minOccurs="0">
                        <simpleType>
                              <restriction base="string">
                                    <enumeration value="mono"/>
                                    <enumeration value="multi"/>
                              </restriction>
                        </simpleType>
                  </element>

                        <element name="codecTypeMappers"
minOccurs="0">
                              <complexType>
                                    <sequence>
                                          <element
name="codecTypeMapper"

      type="cc:codecTypeMapper"
                                           maxOccurs="unbounded"/>
                              </sequence>
                        </complexType>
                  </element>

            </sequence>
</complexType>
      </element>

      <!--  a type can be pointed with the help of a XML Schema
prefix, in order
                  to know the schema it is belonging to
   -->
      <simpleType name="coupleSchemaTypeType">
            <restriction base="string"/>
      </simpleType>
```

【表９Ｂ】

```
	<!-- restriction of 32 maximal codecTypeMappers in a context
-->
	<simpleType name="codecIDType">
		<restriction base="integer">
			<minInclusive="0"/>
			<maxInclusive="31"/>
		</restriction>
	</simpleType>


	<complexType name="codecTypeMapperType">
		<sequence>
			<element name="type" type="coupleSchemaTypeType"
maxOccurs="unbounded"/>
			<element name="codec" type="cc:codecType"/>
		</sequence>
		<attribute name="id" use="required"
type="codecIDType"/>
		<attribute name="state">
			<simpleType>
				<restriction base="string">
					<enumeration value="activated"/>
					<enumeration value="deactivated"/>
				</restriction>
			</simpleType>
		</attribute>
	</complexType>

	<complexType name="codecType" abstract="true"/>

</schema>
```

【０１００】
［補遺３］
【表１０】

ANNEX 3

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
  <AudioEnvelope>
    <cc:modifyContext>
      <codingProperties>
        <codingProperty id="1">
          <type>audioFloat</type>
          <codec xsi:type="LinearQuantifierType">
            <bitSize>8</bitSize>
            <minInclusive>-1.0</minInclusive>
            <maxInclusive>1.0</maxInclusive>
          </codec>
        </codingProperty>
      </codingProperties>
    </cc:modifyContext>
    <Values>
      <Raw>
        -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814
0.145544 -0.847
      </Raw>
    </Values>
  </AudioEnvelope>
</Example>
```

【０１０１】
［補遺４］
【表１１Ａ】

ANNEX 4

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
  <AudioEnvelope>
    <cc:modifyContext>
      <codingProperties>
        <codingProperty id="1" state="deactivated">
          <type>audioFloat</type>
          <codec xsi:type="LinearQuantifierType">
            <bitSize>8</bitSize>
            <minInclusive>-1.0</minInclusive>
            <maxInclusive>1.0</maxInclusive>
          </codec>
        </codingProperty>
      </codingProperties>
    </cc:modifyContext>
    <Values>
      <Raw> <!-- quantization here -->
        <cc:modifyContext>
          <codingProperties>
            <codingProperty id="1" state="activated"/>
          </codingProperties>
        </cc:modifyContext>
        -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814
0.145544 -0.847
      </Raw>
      <Variance> > <!-- but no quantization here -->
        0.1777441 0.2094511 0.349411 0.548444 –0.445445 –0.3654847
0.9541
      </Variance>
```

【表１１Ｂ】

```
        </Values>
      </AudioEnvelope>
    </Example>
```

【０１０２】
［補遺５］
【表１２】

ANNEX 5

```
<complexType name="LinearQuantizerCodecType">
  <complexContent>
    <extension base="cc:CodecType">
      <element name="bitSize">
        <simpleType>
          <restriction base="int">
            <minInclusive value="1"/>
            <maxInclusive value="32"/>
          </restriction>
        </simpleType>
      </element>
      <element name="minInclusive" value="float"/>
      <element name="maxInclusive" value="float"/>
    </extension>
  </complexContent>
</complexType>
```

【０１０３】
［補遺６］
【表１３】

ANNEX 6

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
    <AudioEnvelope>
        <cc:modifyContext>
            <codecTypeMappers>
                <codecTypeMapper id="1" state="deactivated">
                    <type>audioFloat</type>
                    <codec xsi:type="LinearQuantizerCodecType">
                        <bitSize>8</bitSize>
                        <minInclusive>-1.0</minInclusive>
                        <maxInclusive>1.0</maxInclusive>
                    </codec>
                </codecTypeMapper>
            </codecTypeMappers>
        </cc:modifyContext>
        <Values>
            <Raw> <!-- quantization here -->
                <cc:modifyContext>
                    <codecTypeMappers>
                        <codecTypeMapper id="1" state="activated"/>
                    </codecTypeMappers>
                </cc:modifyContext>
                -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814 0.145544 -0.847
            </Raw>
            <Variance> > <!-- but no quantization here -->
                0.1777441 0.2094511 0.349411 0.548444 –0.445445 –0.3654847 0.9541
            </Variance>
        </Values>
    </AudioEnvelope>
</Example>
```

【０１０４】
［補遺７］
【表１４】

ANNEX 7

```
<complexType name="ZLibCodecType">
    <complexContent>
        <extension base="cc:CodecType"/>
    </complexContent>
</complexType>
```

【０１０５】
［補遺８］
【表１５Ａ】

ANNEX 8

```
<MdsExampleTest xmlns="http://www.mpeg7.org/2001/MPEG-7_Schema"
    xmlns:cc="http://www.mpeg7.org/2001/BiMCoding"
    <cc:modifyContext>
        <codecTypeMappers>
            <codecTypeMapper id="1">
                <type>string</type>
                <type>anyURI</type>
                <codec xsi:type="ZLibCodecType"/>
            </codecTypeMapper>
        </codecTypeMappers>
    </cc:modifyContext>

    <!--  the termId attributes, Name elements and Definitions elements
          will be catched by the ZLibCodecType -->


    <ClassificationScheme uri="urn:mpeg:MPEG7AudioDomainCS" domain="/..">
        <Term termId="1">
            <Name xml:lang="en">Source</Name>
            <Definition xml:lang="en">Type of audio source</Definition>
            <Term termId="1.1">
                <Name xml:lang="en">Synthetic</Name>
            </Term>
            <Term termId="1.2">
                <Name xml:lang="en">Natural</Name>
            </Term>
        </Term>
        <Term termId="2">
            <Name xml:lang="en">Acquisition</Name>
```

【表１５Ｂ】

```
<Definition xml:lang="en">Type of Content</Definition>
<Term termId="2.1">
    <Name xml:lang="en">Music</Name>
</Term>
<Term termId="2.2">
    <Name xml:lang="en">Speech</Name>
</Term>
<Term termId="2.3">
    <Name xml:lang="en">Mixed</Name>
</Term>
<Term termId="2.4">
    <Name xml:lang="en">Multi-track</Name>
</Term>
</Term>
...
```

【０１０６】
［補遺９］
【表１６】

ANNEX 9

References

1 – MPEG-7 Systems FCD, N4001, MPEG Singapore meeting, March 2001.

3 – ISO/IEC 14496-1, MPEG-4 Systems, N3850.

4 – The ZLib API, http://www.gzip.org/zlib/, RFC 1950, RFC 1951, RFC 1952 available at http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1950.html

【図面の簡単な説明】
【０１０７】
【図１】コーディングコンテキストの概念を説明する図である。
【図２】ＢｉＭ技術に基づいてコーディングされた要素の構造を説明する図である。
【図３】階層化ツリーのリーフの内容を圧縮するために、本発明に基づいて実行される幾つかのステップを示す図である。

【図１】

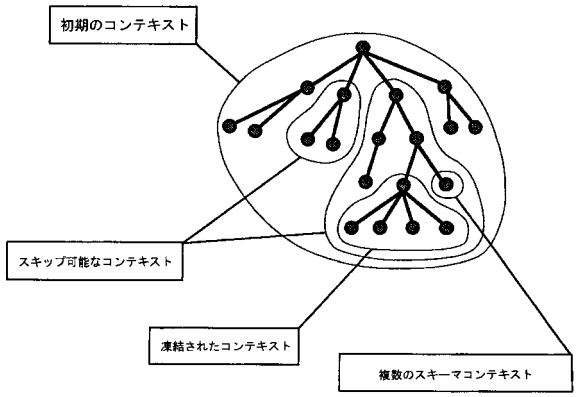


Fig. 1

【図２】

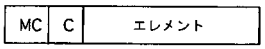


Fig. 2

【図３】

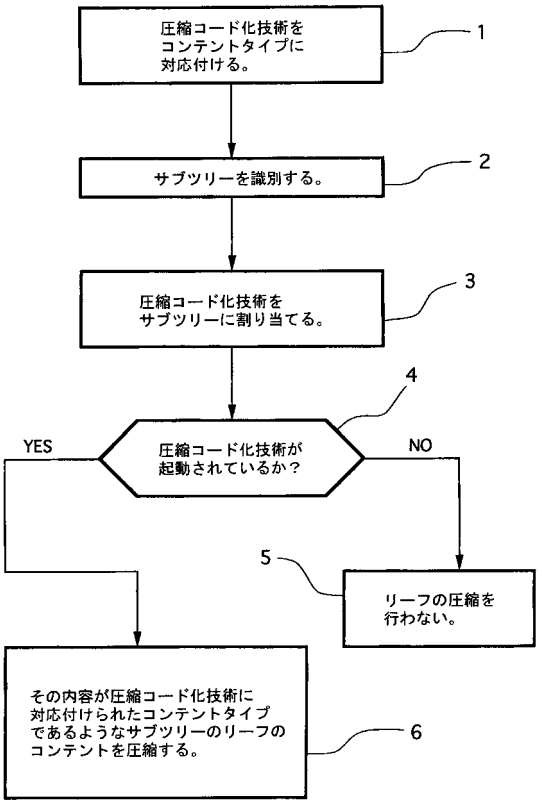


Fig. 3

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 January 2003 (23.01.2003)

PCT

(10) International Publication Number
WO 03/007614 A2

(51) International Patent Classification⁷: H04N 7/24

(21) International Application Number: PCT/EP02/08667

(22) International Filing Date: 12 July 2002 (12.07.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
01460047.2 13 July 2001 (13.07.2001) EP

(71) Applicants *(for all designated States except US)*: **FRANCE TELECOM** [FR/FR]; 6, Place d'Alleray, F-75015 Paris (FR). **GROUPE DES ECOLES DES TELECOMMUNICATIONS** [FR/FR]; 46, rue Barrault, F-75634 Paris Cedex 13 (FR). **EXPWAY** [FR/FR]; 16, rue Vauthier Le Noir, F-51100 Reims (FR).

(72) Inventors; and

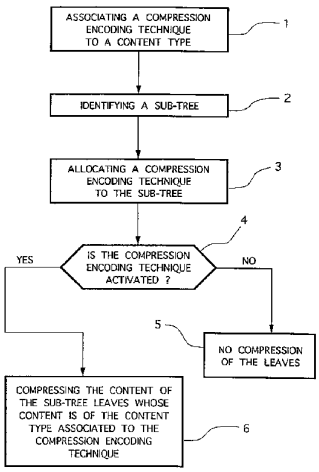
(75) Inventors/Applicants *(for US only)*: **CONCOLATO, Cyril** [FR/FR]; 8, rue Robert Marchand, F-94250 Gentilly (FR). **SEYRAT, Claude** [FR/FR]; 12, rue Rollin, F-75005 Paris (FR). **PAU, Grégoire** [FR/FR]; 32, rue du Cotentin, F-75015 Paris (FR). **THIENOT, Cédric** [FR/FR]; 115, rue Oberkampf, F-75011 Paris (FR). **COTARMANAC'H, Alexandre** [FR/FR]; 3, rue Paul Bert, F-35000 Rennes (FR).

(74) Agent: **VIDON, Patrice**; Le Nobel, 2, allée Antoine Becquerel, BP 90333, F-35703 Rennes Cedex 7 (FR).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG,

[Continued on next page]

(54) Title: METHOD FOR COMPRESSING A HIERARCHICAL TREE, CORRESPONDING SIGNAL AND METHOD FOR DECODING A SIGNAL.



(57) Abstract: The invention regards a method for compressing a hierarchical tree describing a multimedia signal, said tree comprising nodes and leaves, which can be associated to contents of at least two distinct types.According to the invention, said method implements a content compression for at least some of said leaves by means of at least two compression encoding techniques, each of said techniques being selectively associated to at least one of said content types.

WO 03/007614 A2

WO 03/007614 A2

SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Method for compressing a hierarchical tree, corresponding signal and method for decoding a signal.

The field of the invention is that of the compression of data. More precisely, the invention regards the compression of XML-based document ("eXtended Markup Language").

The invention has applications, in particular, but not only, in the following fields :
- multimedia applications ;
- indexation tools ;
- meta-data manipulation tools ;
- the MPEG-7 specification ;
- SMIL ;
- TV Anytime ;
- the third generation of radiocommunications (3GPP).

Compression techniques of the prior art for XML have several drawbacks. In particular, they do not support at the same time fast access to data, high compression ratios and progressive construction of the document. In other words, most of the time, when one of the above mentioned feature is supported, all other features are missing.

One of the compression techniques of the prior art is known as BiM (Binary MPEG). Such a technique provides a method for compressing a XML document by binarising the structure of the document, that it to say the nodes of a tree structure associated to the XML document. Hence, the compression ratio achieved by implementing the BiM technique is very poor, although the BiM technique allows a fast access to data, progressive construction of the document and skippability.

It is an aim of the invention to compensate for the drawbacks of the techniques of the prior art.

More precisely, the invention aims at providing an efficient compression technique for XML-based documents.

The invention also aims at providing a compression technique for XML which provides skippability, high compression ratios and progressive construction of the document.

The invention also aims at compressing efficiently MPEG-7 descriptors.

Another aim of the invention is to implement a method for compressing an XML document which enhances greatly the compression ratio provided by a technique of the BiM type, but which provides the same functionalities as that provided by BiM.

The above mentioned aims of the invention, as well as other aims of the invention which will appear later on, are achieved, according to the invention, by means of a method for compressing a hierarchical tree describing a multimedia signal, said tree comprising nodes and leaves, which can be associated to contents of at least two distinct types, wherein said method implements a content compression for at least some of said leaves by means of at least two compression encoding techniques, each of said techniques being selectively associated to at least one of said content types.

According to a preferred embodiment of the invention, such a method comprises a step of identifying at least one sub-tree and a step of allocating one of said compression encoding techniques to said sub-tree.

Advantageously, such a method comprises a step of implementing said compression encoding technique allocated to said sub-tree only for the leaves of said sub-tree whose content is of the type associated to said compression encoding technique, and the other leaves of said sub-tree do not undergo any compression encoding.

According to an advantageous feature of the invention, such a method implements a parametrical description of said compression encoding techniques.

Preferentially, such a method also comprises a step of compressing the structure of said tree.

Advantageously, said tree is of the BiM (Binary MPEG) type according to the MPEG7 standard.

Preferentially one of said compression encoding techniques implements linear quantization.

Advantageously, one of said compression encoding techniques implements a statistical compression algorithm.

Preferentially, said algorithm is of the GZip type.

Advantageously, said algorithm is simultaneously implemented for a set of data corresponding to the content of at least two leaves.

Preferentially, said tree represents the structure of an XML (Extended markup language) type document.

The invention also regards a method for decoding a multimedia signal compressed according to the above-mentioned method for compressing a hierarchical tree.

Advantageously, such a method implements a step of refreshing a present decoding context according to encoding context information conveyed by said signal.

Preferentially, said present context defines at least one content type, said method comprising a step of implementing a compression decoding technique associated to said content type for the leaves having a content of said content type.

The invention also regards a signal generated by the above-mentioned method for compressing a hierarchical tree.

Other features and assets of the invention will appear more clearly in light of the following description, given as a mere illustrative but non restrictive example, and of the following figures :

- figure 1 illustrates the concept of coding context ;
- figure 2 describes the structure of an element as coded according to the BiM technique ;
- figure 3 illustrates some of the steps implemented according to the invention for compressing the content of the leaves of a hierarchical tree.

Before describing in details how to implement the invention, we first recall the main features of the compression technique of the prior art, known as the BiM technique.

To ease the understanding of the document, some references are gathered in annex 9 and referred to throughout the document.

All the annexes provided with the present document are part of the present description of the invention.

1. Prior art

Introduction

A coding context, illustrated in figure 1, is a set of decoding information, needed while decoding the bitstream. A coding context is applicable to the whole sub-tree of the node where it is defined. At every nodes of the tree, the coding context can be modified ; leading to the creation of a new coding context, applicable to the corresponding sub-tree.

A context can carry several information which edict features applicable to the concerned sub-tree. Currently, in the BiM sub-tree coding format [1], these features are skippability of a sub-tree/context and multiple schema encoding of a sub-tree/context (in order to provide the backward and forward compatibility feature).At last, the context mechanism can be disabled in every sub-tree in order to save bandwidth ; this is the context frozen mode.

The coding context mechanism provides maximum flexibility in every sub-tree of a document tree and allows extensible features to be plugged into the BiM encoding mechanism.

We know presents the current context mechanism used in BiM.

Current coding context mechanism

Definitions

CodingContext

A *codingContext* is a set of information, the contextual information, needed by the decoder to decode the bitstream. A *codingContext* is applicable to

the node where it has been defined, and the whole sub-tree corresponding to this node.

The current *codingContext*, (i.e. the context applicable at a specified node of a description) can be modified within the document (that is to say, a modification of its underlying set of information). Each modification of a *codingContext* leads to the creation a of new *codingContext*, which will carry the modified set of information. All *codingContexts* are expected to be stacked, in order to get them back, when the decoder has finished decoding a sub-tree corresponding context.

Decoder

The BiM *decoder* is composed of two decoders :

- the context decoder ; this decoder is dedicated to decode the contextual information. As stated before, the contextual information is not part of the description. This is a set of information which carry some external features, backward and forward compatibility, fast skipping...
- the element decoder ; this decoder, the BiM regular one [1] is dedicated to decode the element information.

Chunks

Each element of a description is coded as the 3-plet illustrated in figure 2, where the header part is constituted of two chunks, whom sizes can be nil :

- MC is the metacontext chunk ;
- C is the context chunk ;
- Element is the element chunk, which is the regular element coding chunk, see [1].

The MC metacontext chunk contains the information needed by the decoder to decode the following C chunk. That is to say that the MC chunk is the context chunk of the C context chunk.

The C context chunk contains the information able to change the current coding context set of information, and needed by the decoder to decode the

following element chunk. That is to say that the C chunk is the context chunk of the element chunk.

Set of information

The current BiM coding context carries a set of information, the contextual information, which can be divided in the two following main classes :

- the metacontext section ; if the information contained in this section, influence only the context decoding process
- the context section ; if the information contained in this section, influence only the element decoding process

The current set of information is the following set of variables :

Class	Variable	Value	Description
Metacontext	freezing_state	boolean	false: The context changed within this sub-t true : The context ca changed anymore in tl tree.
Context	allows_skip	(mandatory, optional, forbidden)	Is skipping feature ma optional or forbidden context ?
	schema_mode	(mono, multi)	Is the current element co multiple schemas ?
	allows_partial_instantiation	boolean	Is partial instantiation all this context ?
	allows_subtyping	boolean	Is subtyping allowed context ?

The classes defined are exactly coding the chunks described above (the MC metacontext chunk and the C context chunk).

Metacontext chunk

Definition

The MC metacontext chunk, which size can be null, contains information to know if the decoder has to read the next C context chunk, described in the following section.

Variables involved

Variable	Value	Description
freezing_state	boolean	false: The context can be changed within this sub-tree. The MC chunk will be coded into the bitstream. true : The context cannot be changed anymore in this sub-tree. The MC chunk won't be coded into the bitstream.

Default values

The default value of freezing_state is false ; that is to say that, by default, the root context can be dynamically changed.

Propagation rules

At the creation of a new context :

- the freezing_state value is set to the freezing_state value of its father's context

Dynamic modification rules

At each node of a description and in order to create a new context :

- freezing_state value can be switched from the value false to the value true

Decoding rules

If the freezing_state value is true, the MC metacontext chunk (and the upcoming C context chunk) is not coded into the bitstream. Otherwise, the MC metacontext chunk part of the header is coded as follows:

MC () {	# of bits	Mnem
freeze_type	1-3	vlclbf
}		

The context_chunk is a local variable, initialised at false.

freeze type	**Implication**
110	context_chunk = true and freezing_state = true
10	context_chunk = true
111	context_chunk = false and freezing_state = true
0	context_chunk = false

As stated in the previous section, the modification of the variables of the current context implies the creation of a new context.

Influence on the context decoding process

If the context_chunk value is true, the decoder has to read the following C context chunk.

Context chunk

Definition

The C context chunk, which size can be null, contains a set of information able to dynamically change the current context variables. These variables are called codingProperties because they influence the BiM element decoding process.

CodingProperties involved

codingProperty	Value	Description
allows_skip	(mandatory, optional, forbidden)	Is skipping feature mandatory, optional or forbidden in this context ?
schema_mode	(mono, multi)	Is the current element coded with multiple schemas ?
allows_partial_instantiation	boolean	Is partial instantiation allowed in this context ?
allows_subtyping	boolean	Is subtyping allowed in this

		context ?

Default values

The allows_skip variable is initialized at the beginning of the bitstream by the first two bits of the special 4 bits bitfield, as defined in the FCD Systems document [1].

The allows_partial_instantiation variable is initialized at the beginning of the bitstream by the third two bits of the special 4 bits bitfield.

The allows_subtyping variable is initialized at the beginning of the bitstream by the fourth two bits of the special 4 bits bitfield.

The default value of schema_mode is mono ; that is to say that, by default, the root sub-tree/context is encoded with one schema.

Propagation rules

At the creation of a new context :

- allows_skip value is set to the allows_skip value of its father's context
- allows_partial_instantiation value is set to the value of its father's context
- allows_subtyping value is set to the value of its father's context
- schema_mode value is set to its default value

Dynamic modification rules

At each node of a description and in order to create a new context :

- allows_skip can be dynamically modified
- allows_partial_instantiation cannot be dynamically modified
- allows_subtyping cannot be dynamically modified
- schema_mode can be dynamically modified

Decoding rules

The C Context chunk is present only if the MC metacontext chunk is already present and its previous local variable context_chunk is true.

The dynamic modification of the current context is described with an XML element which is encoded with the BiM regular encoding scheme. The global element modifyContext from the BiM schema is used. The http://www.mpeg7.org/2001/BiMCoding coding schema is described in annex 1.

The C context chunk has to be decoded with the BiM regular scheme, with the above schema.

As stated before, the modification of the current codingProperties in the context implies the creation of a new context. Therefore, the presence of the C context chunk, implies the creation of a new context, which will carry the modified codingProperties.

If the allowsSkip element is instantiated within the modifyContext element, then the value of allows_skip will be updated in the new context.

If the schema_mode element is instantiated within the modifyContext element, then the value of schema_mode will be updated in the new context.

Influence on the element decoding process

The allows_skip and schema_mode values influence the element decoding process, when dealing with the skipping feature. This behavior is described in [1].

The schema_mode value influences the element decoding process, in order to know if the element is coded with only one schema or several ones. This mechanism is described in [1].

The allows_partial_instantiation value influences the element decoding process, by adding one special type partiallyInstantiated type to the possible subtypes of the element. See [1].

The allows_subtyping value influences the element decoding process, and allows an element or an attribute to have different possible types, in case of element polymorphism (with the xsi:type attribute) or union. See [1].

2. Description of the invention

2.1. Extension of the context mechanism to provide a framework for leaf compression

The invention proposes to extend the current BiM context mechanism in order to support a new and interesting feature : the use of local compressors to compress leaves of a document, in order to reduce the size of the resulting bitstream.

This section describes how to extend the current BiM context mechanism to support the use of local compressors. This is typically a new set of variables, codingProperties, linked with specific semantic, propagation and coding rules. Therefore, this new set of codingProperties will extend the current context chunk.

Introduction

In a sub-tree, all the instances of one or several specified simple type can be compressed with one or several specified compressor. This basically defines a mapping between a compressor and one or several simple types.

Moreover :

- in some cases, a compressor can need some external parameters
- a mapping can be activated/deactivated, in order to use a compressor in some sub-trees but not in another ones

At last, in order to be activated/deactivated, a mapping should be referencable and therefore, must have an unique identifier in a context.

Therefore, each context can carry zero, one or several codecTypeMapper ; where a codecTypeMapper is a 4-plet, consisting of an identifier, one or several simple types, a codec, optional external codec parameters and an activation state.

Definitions

CodecTypeMapper

A *codecTypeMapper* is a 4-plet, consisting of :

- an identifier, used as an unique reference key in a sub-tree/context
- one or several simple types, for which the mapping is applicable
- a codec
- optional external codec parameters (depends of the codec)
- an activation state

Identifier

The identifier is an unique number which identify a mapping within a context in an unambiguous way. The BiM coding schema restricts the maximal number of codecTypeMappers in a context to 32.

Simple type

All the simple types defined in a schema could be *a priori* encoded by every codec but, each codec can restrict this choice. For instance, a linear quantizer, as defined hereafter in the document, can only be used with numerical simple types.

A simple type is identified by its name, and the by the URL of the schema its belongs. XML Schema prefixes should be used, in order to point to the correct schema. The BiM coding schema defines a special type to encode this couple ; this type should be encoded as couple of integers ; the first integer is restricted to the current number of schemas known (this piece of information can be fetched in the DecoderConfig part [1]) and the second integer is restricted to the number of global simple types present in the corresponding schema.

Codec

A *codec*, standing for compressor/decompressor, is a module which takes input bits, and writes output bits. It can need some optional external parameters.

A codec is identified by a name, among the names of non-abstract codecs defined in the BiM coding schema. The current BiM coding schema, defined in a section above, doesn't define any non-abstracts codecs, but §2.2 of the present document does.

Activation state

The activation state is a boolean flag.

Semantic rules

CodecTypeMapper

Each context :

- can carry zero, one or several codecTypeMappers
- can define one or several codecTypeMappers
- can activate or deactivate one or several existing codecTypeMappers

If a codecTypeMapper is defined in a context, it remains in all its subcontexts.

An existing codecTypeMapper, within a context, cannot be deleted nor modified (except its activation state).

Identifier

The identifier of a mapping must be unique among all the codecTypeMappers of a context.

Simple type

When you associate in a codecTypeMapper one or several simple types and a codec, the codec will encode/decode the simple types themselves and all the simple types which derived from them.

In a context, there must be at most one simple type than can be applicable with a codec.

Codec

There are two types of codecs : memoryless codecs and contextual codecs.

A memoryless codec is a module which encodes always the same input bytes into the same bytes out ; independently of the history of the codec. A typical memoryless codec is a linear quantifier. The BiM leaf compression (see §2.2 of the present document) describes such a codec.

A contextual codec is a module which uses the previous bytes fed in it, (thus changing the context of the codec). Such a codec doesn't generate the same output bytes for the same input bytes it receives. A typically contextual codec is a Zip-like local codec, one is described in §2.2 of the present document.

A memoryless codec doesn't induce any problem in the current context architecture but a contextual codec does, in case of skippable sub-tree. In such cases, a contextual codec is reset, in order not to confuse the decoder, when this former has skipped the sub-tree.

Activation state

In every sub-tree/context, a codecTypeMapper can be activated or deactivated.

This mechanism allows to define a codecTypeMapper in a higher level of the document tree, and activate it only in the sub-trees it is used, without redefining the codecTypeMapper.

A new codingProperty : codecTypeMapper

In this part, we add a new codingProperty to the previous set of variables of the context section, described before. This new codingProperty is named codecTypeMapper and is a list of the previous codecTypeMapper described in the previous section.

New codingProperty involved

The context carries a list of codecTypeMapper :

CodingProperties	**Value**	**Description**
codecTypeMapper[i].identifier	int	Numerical identifie reference a codingPro in a context ir unambiguous way.
codecTypeMapper[i].simple_type[j]	int ; int	List of 2-plet (Nume index of a sch numerical index of a si type in the current sche
codecTypeMapper[i].codec	int	Numerical index of a c in the current co context scheme
codecTypeMapper[i].codec_parameters		Optional external c parameters.
codecTypeMapper[i].activation_state	boolean	State of activation codingProperty.

New default values

By default, there is no codecTypeMapper in a sub-tree/context.

If a codecTypeMapper is defined within a context, its identifier, codec and simple_type value must be defined. If not specified, the state of activation of a newly defined codecTypeMapper is set to true by default ; that is to say that a newly defined codecTypeMapper is activated by default.

New propagation rules

Rule 1: At the creation of a new context, the codecTypeMapper list is the copy of its father's context :

- the identifier value is copied
- the simple_type value is copied
- the codec value is copied according to the rule 2
- the codec_parameters value are copied
- the activation_state is copied

Rule 2: The codec value is copied and

if :

- the father's codec codingProperty[i].codec is a contextual codec
- and if the current context is skippable

Then,

the decoder is expected to create a new instance of the codec by copying the instance of the father's codec (not only its value), and resetting it.

For instance, a ZLib codec would be copied and re-initialized when entering a skippable node.

New dynamic modification rules

The list of codecTypeMapper can be dynamically modified, within the description :

- a new codecTypeMapper can be defined
- the activation_state of an existing (then referencable) codecTypeMapper can be dynamically modified

An existing codecTypeMapper cannot be deleted and its members cannot be dynamically modified (except its activation_state).

New decoding rules

The same previous rules apply to the C context chunk decoding, but the new schema, described in annex 2, should be used, in order to add the new codecTypeMapper dynamic modification functionalities.

Informative part

Examples

The example, illustrated in annex 3, presents the definition of one activated linear quantifier (see §2.2 of the present document) in a description.

The example, illustrated in annex 4, presents the definition of one deactivated linear quantifier in a description.

2.2 BiM Leaf compression

We now present the mechanism implemented by the invention to encode data with different codecs. More precisely, we now illustrate two examples, one where a linear quantization codec is used to compress, for example, floating point values, and the other where a gzip coedc is used to compress, for example, string values.

Such a mechanism is closely related to the coding context and allows the use of several other types of codecs. Moreover, it allows to deal properly with coding context features, for instance skippable subtrees. Finally it allows re-use of codecs in different coding contexts.

Introduction

The BiM sub-tree coding [1] doesn't compress the data leaves of a description. Currently, leaf values are encoded with respect of their types (IEEE 754 floats and doubles, UTF strings...).

In many cases, it could be useful to use some classical compression techniques like linear quantization or statistical compression to improve the compression ratio of BiM without losing its main features (streamline parsing, fast skipping feature, typed decoding).

This document presents how data leaves compression of a document can be done within the context coding mechanism described in §2.1, in order to achieve better compression ratios.

2.2.1. Linear quantization

Definition

Linear quantization is an usual and lossy way to reduce the size of encoded numbers in the bitstream, when the source of the information is known and therefore, when losses can be controlled.

As an example, the envelope of a sampled audio signal is often known with a precise bitsize quantization, and this technique could be fruitfully used for coding MPEG-7 audio descriptions.

If v is a real number, it can be encoded with v_q with *nbits* bits where :

$$v_q = \frac{v - v_{min}}{v_{max} - v_{min}}(2^{nbits} - 1)$$

where :

- v_q is the quantized, encoded value of v
- *nbits* is the precision required in bits
- v_{min} is the minimal inclusive value that v can reach
- v_{max} is the maximal inclusive value that v can reach

And the decoded value from v is $\bar{v}$, with :

$$\bar{v} = v_{min} + v_q \frac{v_{max} - v_{min}}{2^{nbits} - 1}$$

where :

- $\bar{v}$ is the decoded, approximated value of v

Integration with the context mechanism : the LinearQuantizerCodec

Linear quantization can be used as a codec, as defined in the coding context mechanism described in §2.1 of the present document. With this mechanism, linear quantization can be applied on numerical data leaves, of a desired simple type, in any sub-tree of a description.

Used like this, the coding context mechanism, associated with the linear quantization codec, is acting as the QuantizationParameter node, used in MPEG-4 BIFS [3].

Restriction on applicable simple types

According to the definition of a codec in §2.1 of the present document, this codec is a memoryless codec, which can be applied on every atomic and non-atomic simple numerical types ; whose XML Schema primitive type is float, double or decimal.

Codec external parameters

The linear quantizer codec needs the following 3 mandatory parameters :

- bitSize ; the *nbits* variable described above
- minInclusive ; the v_{min} variable described above
- maxInclusive ; the v_{max} variable described above

Schema definition of the codec

The linear quantization codec is a new codec of type LinearQuantizerCodecType, based on the abstract CodecType type (see §2.1) and defined by the schema given in annex 5, in the coding context namespace URL xmlns:cc=http://www.mpeg7.org/2001/BiMCoding.

Encoding (informative)

A numerical data leaf of value v is encoded with the unsigned integer v_q on *nbits* bits where :

$$v_q = \frac{v - v_{min}}{v_{max} - v_{min}}(2^{nbits} - 1)$$

Decoding

The unsigned integer v_q, coded on *nbits* bits, should be decoded as $\bar{v}$ where :

$$\bar{v} = v_{min} + v_q \frac{v_{max} - v_{min}}{2^{nbits} - 1}$$

Example (informative)

The example illustrated in annex 6 presents the definition of a linear quantizer in a description.

2.2.2. Statistical compression

Classical statistical lossless compression algorithms can be used as codecs, as defined in the coding context mechanism (see §2.1). With this mechanism, data

leaves, of a desired simple type, can be compressed efficiently in any sub-tree of a description.

This codec is useful for significantly reducing the size of the bitstream, especially when the description contains many repetitive or similar strings.

Definition

Classical lossless statistical compression algorithms (like Zip or GZip) can be used in BiM to compress any leaves of a description.

But, in most cases, when data leaves are short strings which contain fewer than 10 characters, this leads to poor performances because usual statistical compression algorithms requires a larger lookahead buffer.

In order to achieve optimal compression ratio, the leaves of a document have to be buffered into a small buffer before being compressed. The following section defines such a buffered statistical coder, relying on an underlying lossless statistical compression algorithm.

Definitions of a buffered statistical coder

A buffered statistical coder relies on an underlying statistical coder which should contain the generic following primitives methods :

- initialize_stream() ; which initializes a compressing or a decompressing stream
- reset_model() ; which resets the current statistical model of the coder
- feed_input_bytes() ; which takes input decompressed bytes and put them in the compressing stream
- flush_output_bytes() ; which flushes the compressing stream by compressing the input bytes already processed and by emitting the corresponding compressed output bytes
- decompress_input_bytes() ; which takes a specified amount of input compressed bytes and decodes them by emitting the corresponding decompressed output bytes

A buffered codec has a bufferSize bytes length, byte array buffer FIFO structure.

From the encoder side, the bufferSize value indicates how many input bytes the encoder can process before flushing. From the decoder side, this is the minimal buffer size in bytes, needed to decode the bitstream, through the underlying statistical coder API.

The buffer has also a fillingLevel variable, which contains the actual filling level, in bytes, of the buffer.

Using the ZLib API as a statistical coder

The ZLib public library API [4], used in the GZip compression scheme, provides an efficient and useful API for using statistical compression on document leaves.

The ZLib API fulfils the previous generic methods, with the following mapping :

- initialize_stream() can be mapped with the ZLib's inflateInit() or deflateInit() functions, with the Z_DEFAULT_COMPRESSION efficiency value parameter.
- reset_model() can be mapped with an ZLib's inflateEnd() or a deflateEnd() call and a following initialize_stream() call.
- feed_input_bytes() can be mapped with the ZLib's deflate() method with the Z_NO_FLUSH parameter.
- flush_output_bytes() can be mapped with the ZLib's deflate() method with the Z_SYNC_FLUSH parameter.
- decompress_input_bytes() can be mapped with the ZLib's inflate() method.

The ZLib buffered codec should be initialized with the Z_DEFAULT_COMPRESSION efficiency value, as defined in [4], which provides a good tradeoff between memory footprint requirements and compression efficiency.

Integration with the context mechanism : the ZLibCodec

This section describes the integration of the previously buffered statistical coder defined, relying on the ZLib API, within the coding context mechanism.

Restriction on applicable simple types

According to the definition of a codec in §2.1, this codec is a contextual codec, which can be applied on every atomic and non-atomic string types. The ZLibCodec is relying on the underlying primitive encoding of leaves of a document, as described in [1]. For instance, int leaves are encoded with a 32 bits unsigned integer, string with a UTF-8 encoding, float and double are encoded with the IEEE 754 format, ... Therefore, the ZLibCodec will compress the encoded leaf

Codec external parameters

The buffered ZLib codec doesn't need any external parameters, as the efficiency of the underlying ZLib is set at Z_DEFAULT_COMPRESSION and as the bufferSize parameter is not needed from the decoder side.

Schema definition of the codec

The ZLib codec is a new codec of type ZLibCodecType, based on the abstract CodecType (see §2.1) type and defined by the schema illustrated in annex 7, in the coding context namespace.

Encoding (informative)

At the activation/instantiation of the codec :

- the FIFO buffer structure is supposed to be clear, its fillingLevel is set to 0
- the global variable referencable_chunk is initialized to null

The referencable_chunk should contain a referencable chunk of bits, which must be hold by the encoder, because its value will be known later during the encoding process. The signaling function signal_reference_chunk_known() could be called when this chunk is known.

The size, in bytes, of every non-nil chunk should be write before the chunk itself, during the flush_output_bytes() call, with the standard unsigned infinite integer 4+1 coding, as defined in [1].

An input leaf is the encoded value of a textual leaf, with respect of its primitive type. The length of the leaf, in bytes, is given by the field leaf.length. For instance [1], a string leaf is an UTF-8 code, preceded by the size in bytes of the string

(coded with the infinite integer coding [1]) ; a double leaf is the 64-bits value of the corresponding IEEE 754 standard...

The following encode_leaf function is able to encode a leaf in a chunk of output bytes :

```
chunk encode_leaf(chunk leaf) {
    while (leaf is not empty) {
        if (fillingLevel + leaf.length < bufferSize) {
            feed_input_bytes(leaf,leaf.length)
            fillingLevel = fillingLevel + leaf.length
            if (referencable_chunk is null) {
                referencable_chunk = new chunk
                return referencable_chunk
            } else {
                return nil_size_chunk
            }
        } else {
            remaining_bytes = bufferSize – fillingLevel – leaf.length
            feed_input_bytes(leaf,remaining_bytes)
            referencable_chunk = flush_output_bytes()
            signal_reference_chunk_known()
            fillingLevel = 0
            leaf = leaf.remove_beginning_bytes(remaining_bytes)
            referencable_chunk = null
        }
    }
}
```

Decoding

Let string_fifo be a string FIFO.

The following methods get() and put() respectively take an element out resp. put an element from resp in the FIFO.

The method is Empty() signals whether the Fifo is empty
Let sub be the function that takes a substring.
Let concat be the concatenation function
Let getData() the function that returns the char[] holding de-gzip data from a leaf .
Let split(char[] , char sep, Fifo, char[] remainder) be the method that splits an array of characters at each separator 'sep' and stores the separated string elements into the Fifo and returns the remainder (i.e. the last chunk that has no 'sep' to finish it)

```
split(char [] data, char sep, FIFO string_fifo,char[] remainder)
{
    if (data==null) return;
    int BEGIN=0;
    for (int I=0;I<data.length;I++)
    {
        if (data[I]==sep)
        {
            char[] str= concat(remainder, sub(data,BEGIN,I));
            put(string_fifo, str);
            BEGIN=I+1;
        }
    }
    if (BEGIN!=data.length)
    {
        //there's a remainder.
        remainder = sub(data,BEGIN,data.length);
    }
}
String decode()
{
    if (isEmpty(string_fifo)
```

```
        {
                data = getData();
                split(data,0x00,string_fifo,remainder);
        }
        return get(string_fifo);
}
```

At initialization, string_fifo is empty.

At the activation/instantiation of the codec :

- the FIFO structure is supposed to be clear, its numberOfLeaves is set to 0
- the variable first_chunk is set to true

Let the separator byte be 0x00.

The following decode_leaf function is able to decode a compressed leaf from the bitstream :

```
string decode_leaf() {
        if (numberOfLeaves == 0) {
                read_and_decompressed_byte()
                numberOfLeaves = count_number_of_leave_in_buffer()
        } else {

        }
}
```

Decoding is defined by :

1. If the FIFO is empty :
    a. decode the coded data,
    b. stack in a FIFO all elements separated by 0x00
    c. if the last character is not 0x00 store the unfinished string temporarily.
    d. if "last_element" is not empty, insert it at the beginning of the first element in FIFO
    e. put the unfinished string of this round in last_element.

f. remove and return the first element.

2. If the FIFO is not empty, then remove and return the first element.

It is equivalent to say : "the FIFO is not empty" and to say "there is no encoded data in a the current leaf."

Example (informative)

The description given in annex 8 is an example of the usage of the ZLibCodecType codec, mapped with the string and the anyURI types.

Results (informative)

The following figures show the performances of using the ZLibCodec so as to compress textual leaves of descriptions (those derived from the string and the anyURI XML Schema primitive types). A buffer of bufferSize = 256 bytes was used during the encoding process.

The files used were provided by the MPEG-7 MDS sub-group.

Filename	Original size (bytes)	BiM size (bytes)	BiM with the ZLib codec size (bytes)	Zipped file siz (bytes)
mdsExamplesClause11-12.xml	160658	22512	10602	17019
mdsExamplesClause13-15.xml	81133	11627	8538	9698
mdsExamplesClause17.xml	142426	57583	22489	21444
mdsExamplesClause4-7.xml	37208	6536	3748	7623
mdsExamplesClause8-10.xml	8179	2081	1389	2416

We now quickly describe the steps implemented according to the invention for compressing the content of the leaves of a hierarchical tree.

Step 1 consists in associating a compression encoding technique to a content type. For example, linear quantization can be associated to floating point values.

In step 2, a sub-tree is identified within the hierarchical tree corresponding to the structure of the considered XML document.

Step 3 consists in allocating a compression encoding technique to the identified sub-tree.

Step 4 then consists in checking whether the codec implementing the compression encoding technique is or not activated. If no, no compression (5) of the leaves of the sub-tree is achieved.

If yes, the invention implements (6) compression of the content of the sub-tree leaves whose content is of the content type associated (1) to the compression encoding technique.

ANNEX 1

```
<schema targetNamespace="http://www.mpeg7.org/2001/BiMCoding"
    xmlns:cc="http://www.mpeg7.org/2001/BiMCoding"
    xmlns="http://www.w3.org/2000/10/XMLSchema">

    <element name="modifyContext">
        <complexType>
            <sequence>
                <element name="allowsSkip" minOccurs="0">
                    <simpleType>
                        <restriction base="string">
                            <enumeration value="mandatory"/>
                            <enumeration value="optional"/>
                            <enumeration value="forbidden"/>
                        </restriction>
                    </simpleType>
                </element>

                <element name="schemaMode" minOccurs="0">
                    <simpleType>
                        <restriction base="string">
                            <enumeration value="mono"/>
                            <enumeration value="multi"/>
                        </restriction>
                    </simpleType>
                </element>
            </sequence>
        </complexType>
    </element>
</schema>
```

ANNEX 2

```
<schema targetNamespace="http://www.mpeg7.org/2001/BiMCoding"
xmlns:cc=" http://www.mpeg7.org/2001/BiMCoding"
xmlns="http://www.w3.org/2000/10/XMLSchema"
>
      <element name="context">
            <complexType>
                  <sequence>
                        <element name="allowsSkip" minOccurs="0">
                              <simpleType>
                                    <restriction base="string">
                                          <enumeration
value="mandatory"/>
                                          <enumeration
value="optional"/>
                                          <enumeration
value="forbidden"/>
                                    </restriction>
                              </simpleType>
                        </element>

                  <element name="schemaMode" minOccurs="0">
                        <simpleType>
                              <restriction base="string">
                                    <enumeration value="mono"/>
                                    <enumeration value="multi"/>
                              </restriction>
                        </simpleType>
                  </element>

                        <element name="codecTypeMappers"
minOccurs="0">
                              <complexType>
                                    <sequence>
                                          <element
name="codecTypeMapper"

      type="cc:codecTypeMapper"
                                                maxOccurs="unbounded"/>
                              </sequence>
                        </complexType>
                  </element>

            </sequence>
</complexType>
      </element>

      <!--  a type can be pointed with the help of a XML Schema
prefix, in order
                  to know the schema it is belonging to
   -->
      <simpleType name="coupleSchemaTypeType">
            <restriction base="string"/>
      </simpleType>
```

```
        <!-- restriction of 32 maximal codecTypeMappers in a context
    -->
        <simpleType name="codecIDType">
              <restriction base="integer">
                    <minInclusive="0"/>
                    <maxInclusive="31"/>
              </restriction>
        </simpleType>



        <complexType name="codecTypeMapperType">
              <sequence>
                    <element name="type" type="coupleSchemaTypeType"
    maxOccurs="unbounded"/>
                    <element name="codec" type="cc:codecType"/>
              </sequence>
              <attribute name="id" use="required"
    type="codecIDType"/>
              <attribute name="state">
                    <simpleType>
                          <restriction base="string">
                                <enumeration value="activated"/>
                                <enumeration value="deactivated"/>
                          </restriction>
                    </simpleType>
              </attribute>
        </complexType>

        <complexType name="codecType" abstract="true"/>

    </schema>
```

ANNEX 3

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
  <AudioEnvelope>
    <cc:modifyContext>
      <codingProperties>
        <codingProperty id="1">
          <type>audioFloat</type>
          <codec xsi:type="LinearQuantifierType">
            <bitSize>8</bitSize>
            <minInclusive>-1.0</minInclusive>
            <maxInclusive>1.0</maxInclusive>
          </codec>
        </codingProperty>
      </codingProperties>
    </cc:modifyContext>
    <Values>
      <Raw>
        -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814
0.145544 -0.847
      </Raw>
    </Values>
  </AudioEnvelope>
</Example>
```

ANNEX 4

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
  <AudioEnvelope>
    <cc:modifyContext>
      <codingProperties>
        <codingProperty id="1" state="deactivated">
          <type>audioFloat</type>
          <codec xsi:type="LinearQuantifierType">
            <bitSize>8</bitSize>
            <minInclusive>-1.0</minInclusive>
            <maxInclusive>1.0</maxInclusive>
          </codec>
        </codingProperty>
      </codingProperties>
    </cc:modifyContext>
    <Values>
      <Raw> <!-- quantization here -->
        <cc:modifyContext>
          <codingProperties>
            <codingProperty id="1" state="activated"/>
          </codingProperties>
        </cc:modifyContext>
        -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814
0.145544 -0.847
      </Raw>
      <Variance> > <!-- but no quantization here -->
        0.1777441 0.2094511 0.349411 0.548444 –0.445445 –0.3654847
0.9541
      </Variance>
```

```
        </Values>
      </AudioEnvelope>
    </Example>
```

ANNEX 5

```
<complexType name="LinearQuantizerCodecType">
    <complexContent>
        <extension base="cc:CodecType">
            <element name="bitSize">
                <simpleType>
                    <restriction base="int">
                        <minInclusive value="1"/>
                        <maxInclusive value="32"/>
                    </restriction>
                    </simpleType>
            </element>
            <element name="minInclusive" value="float"/>
            <element name="maxInclusive" value="float"/>
        </extension>
    </complexContent>
</complexType>
```

ANNEX 6

```
<Example xmlns:cc="http://www.mpeg7.org/2001/BiMCoding">
    <AudioEnvelope>
        <cc:modifyContext>
            <codecTypeMappers>
                <codecTypeMapper id="1" state="deactivated">
                    <type>audioFloat</type>
                    <codec xsi:type="LinearQuantizerCodecType">
                        <bitSize>8</bitSize>
                        <minInclusive>-1.0</minInclusive>
                        <maxInclusive>1.0</maxInclusive>
                    </codec>
                </codecTypeMapper>
            </codecTypeMappers>
        </cc:modifyContext>
        <Values>
            <Raw> <!-- quantization here -->
                <cc:modifyContext>
                    <codecTypeMappers>
                        <codecTypeMapper id="1" state="activated"/>
                    </codecTypeMappers>
                </cc:modifyContext>
                -0.25411 0.88541 0.2141946 0.3652541 -0.148941 0.8814 0.145544 -0.847
            </Raw>
            <Variance> > <!-- but no quantization here -->
                0.1777441 0.2094511 0.349411 0.548444 -0.445445 -0.3654847 0.9541
            </Variance>
        </Values>
    </AudioEnvelope>
```

</Example>

ANNEX 7

```
<complexType name="ZLibCodecType">
    <complexContent>
        <extension base="cc:CodecType"/>
    </complexContent>
</complexType>
```

ANNEX 8

```
<MdsExampleTest xmlns="http://www.mpeg7.org/2001/MPEG-7_Schema"
    xmlns:cc="http://www.mpeg7.org/2001/BiMCoding"
    <cc:modifyContext>
        <codecTypeMappers>
            <codecTypeMapper id="1">
                <type>string</type>
                <type>anyURI</type>
                <codec xsi:type="ZLibCodecType"/>
            </codecTypeMapper>
        </codecTypeMappers>
    </cc:modifyContext>

    <!--  the termId attributes, Name elements and Definitions elements
          will be catched by the ZLibCodecType -->

    <ClassificationScheme uri="urn:mpeg:MPEG7AudioDomainCS" domain="/..">
        <Term termId="1">
            <Name xml:lang="en">Source</Name>
            <Definition xml:lang="en">Type of audio source</Definition>
            <Term termId="1.1">
                <Name xml:lang="en">Synthetic</Name>
            </Term>
            <Term termId="1.2">
                <Name xml:lang="en">Natural</Name>
            </Term>
        </Term>
        <Term termId="2">
            <Name xml:lang="en">Acquisition</Name>
```

```
<Definition xml:lang="en">Type of Content</Definition>
<Term termId="2.1">
    <Name xml:lang="en">Music</Name>
</Term>
<Term termId="2.2">
    <Name xml:lang="en">Speech</Name>
</Term>
<Term termId="2.3">
    <Name xml:lang="en">Mixed</Name>
</Term>
<Term termId="2.4">
    <Name xml:lang="en">Multi-track</Name>
</Term>
</Term>
...
```

ANNEX 9

References

1 – MPEG-7 Systems FCD, N4001, MPEG Singapore meeting, March 2001.

3 – ISO/IEC 14496-1, MPEG-4 Systems, N3850.

4 – The ZLib API, http://www.gzip.org/zlib/, RFC 1950, RFC 1951, RFC 1952 available at http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1950.html

CLAIMS

1. Method for compressing a hierarchical tree describing a multimedia signal, said tree comprising nodes and leaves, which can be associated to data of at least two distinct natures, called data types,

wherein said method implements a data compression for at least some of said leaves by means of at least two compression encoding techniques, each of said techniques being selectively associated to at least one of said data types.

2. Method according to claim 1, comprising a step of identifying at least one sub-tree and a step of allocating one of said compression encoding techniques to said sub-tree.

3. Method according to claim 2, comprising a step of implementing said compression encoding technique allocated to said sub-tree only for the leaves of said sub-tree whose data is of the type associated to said compression encoding technique, and wherein the other leaves of said sub-tree do not undergo any compression encoding.

4. Method according to any of claims 1 to 3, implementing a parametrical description of said compression encoding techniques.

5. Method according to any of claims 1 to 4, also comprising a step of compressing the structure of said tree.

6. Method according to any of claims 1 to 5, wherein said tree is of the BiM (Binary MPEG) type according to the MPEG7 standard.

7. Method according to any of claims 1 to 6, wherein one of said compression encoding techniques implements linear quantization.

8. Method according to any of claims 1 to 7, wherein one of said compression encoding techniques implements a statistical compression algorithm.

9. Method according to claim 6, wherein said algorithm is of the GZip type.

10. Method according to any of claims 8 and 9, wherein said algorithm is simultaneously implemented for a set of data corresponding to the data of at least two leaves.

11. Method according to any of claims 1 to 10, wherein said tree represents the structure of an XML (Extended markup language) type document.

12. Method according to any of claims 1 to 11, also comprising a step of associating at least one coding context to said sub-tree, said coding context comprising pieces of information allowing to skip said sub-tree while decoding said hierarchical tree.

13. Method according to claim 12, wherein said pieces of information comprise :

- a piece of information indicating the used compression encoding technique ; and/or
- a piece of information indicating if the corresponding sub-tree has been compressed ; and/or
- a piece of information indicating if the corresponding sub-tree is skippable; and/or
- a piece of information indicating that at least one parameter of the used compression encoding technique has been modified.

14. Method for decoding a multimedia signal compressed according to the method of any of claims 1 to 13.

15. Method according to claim 14, implementing a step of refreshing a present decoding context according to encoding context information conveyed by said signal.

16. Method according to claim 15, wherein said present context defines at least one data type, said method comprising a step of implementing a compression decoding technique associated to said data type for the leaves comprising data of said data type.

17. Signal generated by the method of any of claims 1 to 13.

1/2

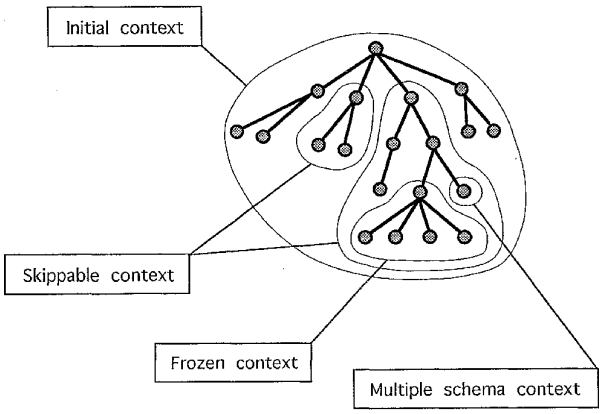

Initial context
Skippable context
Frozen context
Multiple schema context

Fig. 1

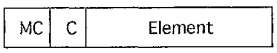

MC
C
Element

Fig. 2

2/2

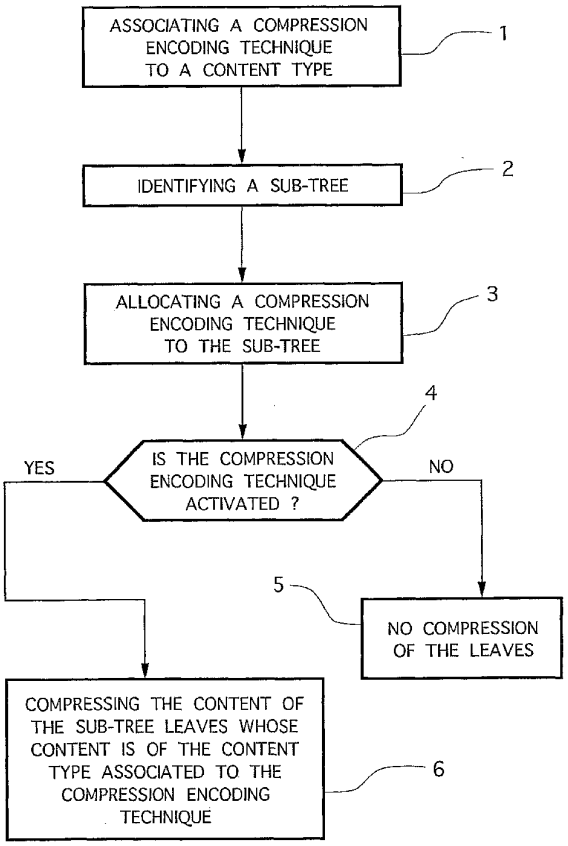

ASSOCIATING A COMPRESSION ENCODING TECHNIQUE TO A CONTENT TYPE
1
IDENTIFYING A SUB-TREE
2
ALLOCATING A COMPRESSION ENCODING TECHNIQUE TO THE SUB-TREE
3
4
IS THE COMPRESSION ENCODING TECHNIQUE ACTIVATED ?
YES
NO
5
NO COMPRESSION OF THE LEAVES
COMPRESSING THE CONTENT OF THE SUB-TREE LEAVES WHOSE CONTENT IS OF THE CONTENT TYPE ASSOCIATED TO THE COMPRESSION ENCODING TECHNIQUE
6

Fig. 3

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 January 2003 (23.01.2003)

PCT

(10) International Publication Number
WO 03/007614 A3

(51) International Patent Classification[7]: H04N 7/24, H03M 7/30

(21) International Application Number: PCT/EP02/08667

(22) International Filing Date: 12 July 2002 (12.07.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
01460047.2 13 July 2001 (13.07.2001) EP

(71) Applicants *(for all designated States except US)*: **FRANCE TELECOM** [FR/FR]; 6, Place d'Alleray, F-75015 Paris (FR). **GROUPE DES ECOLES DES TELECOMMUNICATIONS** [FR/FR]; 46, rue Barrault, F-75634 Paris Cedex 13 (FR). **EXPWAY** [FR/FR]; 16, rue Vauthier Le Noir, F-51100 Reims (FR).

(72) Inventors; and

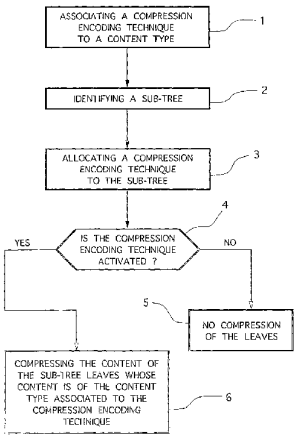
(75) Inventors/Applicants *(for US only)*: **CONCOLATO, Cyril** [FR/FR]; 8, rue Robert Marchand, F-94250 Gentilly (FR). **SEVRAT, Claude** [FR/FR]; 12, rue Rollin, F-75005 Paris (FR). **PAU, Grégoire** [FR/FR]; 32, rue du Cotentin, F-75015 Paris (FR). **THIENOT, Cédric** [FR/FR]; 115, rue Oberkampf, F-75011 Paris (FR). **COTARMANAC'H, Alexandre** [FR/FR]; 3, rue Paul Bert, F-35000 Rennes (FR).

(74) Agent: **VIDON, Patrice**; Le Nobel, 2, allée Antoine Becquerel, BP 90333, F-35703 Rennes Cedex 7 (FR).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

[Continued on next page]

(54) Title: METHOD FOR COMPRESSING A HIERARCHICAL TREE, CORRESPONDING SIGNAL AND METHOD FOR DECODING A SIGNAL.



(57) Abstract: The invention regards a method for compressing a hierarchical tree describing a multimedia signal, said tree comprising nodes and leaves, which can be associated to contents of at least two distinct types. According to the invention, said method implements a content compression for at least some of said leaves by means of at least two compression encoding techniques, each of said techniques being selectively associated to at least one of said content types.

WO 03/007614 A3

WO 03/007614 A3

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
16 October 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Interr Application No
PCT/EP 02/08667

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/24 H03M7/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
E	FR 2 820 563 A (EXPWAY) 9 August 2002 (2002-08-09) abstract page 2, line 24 -page 4, line 3 page 19, line 3 -page 20, line 22 page 22, line 24 -page 23, line 1 figure 1	1-5,7,8, 11-17
X	LIEFKE H ET AL: "XMILL: AN EFFICIENT COMPRESSOR FOR XML DATA" SIGMOD RECORD, ASSOCIATION FOR COMPUTING MACHINERY, NEW YORK, US, vol. 29, no. 2, June 2000 (2000-06), pages 153-164, XP001002286	1,2,4,5, 7,8,10, 11,14-17
Y	page 153, right-hand column, line 31 -page 154, left-hand column, line 10 page 156, right-hand column, line 31 -page 159, right-hand column, line 12	6,9

-/--

[X] Further documents are listed in the continuation of box C. [X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 January 2003	17/01/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Hampson, F

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter Application No
PCT/EP 02/08667

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	NACK F ET AL: "EVERYTHING YOU WANTED TO KNOW ABOUT MPEG-7: PART 2" IEEE MULTIMEDIA, IEEE COMPUTER SOCIETY, US, vol. 6, no. 4, October 1999 (1999-10), pages 64-73, XP000880605 ISSN: 1070-986X	6,9
A	page 70, left-hand column, line 30 -page 70, left-hand column, line 34	1
A	"TEXT OF ISO/IEC FCD 15938-1 INFORMATION TECHNOLOGY - MULTIMEDIA CONTENT DESCRIPTION INTERFACE - PART 1 SYSTEMS" ISO/IEC JTC1/SC29/WG11 MPEG01/N4001, XX, XX, March 2001 (2001-03), pages 1-2,I-V,6-58, XP001001465 cited in the application	
A	WO 97 34240 A (UNIV MASSACHUSETTS) 18 September 1997 (1997-09-18)	
A	CHENEY J: "Compressing XML with multiplexed hierarchical PPM models" PROCEEDINGS DCC 2001. DATA COMPRESSION CONFERENCE, PROCEEDINGS DCC 2001. DATA COMPRESSION CONFERENCE, SNOWBIRD, UT, USA, 27-29 MARCH 2001, pages 163-172, XP002187036 2001, Los Alamitos, CA, USA, IEEE Comput. Soc, USA ISBN: 0-7695-1031-0	
A	WIRELESS APPLICATION PROTOCOL FORUM: "WAP Binary XML Content Format" WAP BINARY XML CONTENT FORMAT DOCUMENT ID WAP-192-WBXML-20000515 VERSION 1.3, XX, XX, 15 May 2000 (2000-05-15), pages 1-25, XP002168490	

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter Application No
PCT/EP 02/08667

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
FR 2820563	A	09-08-2002	FR 2820563 A1		09-08-2002
			WO 02063776 A2		15-08-2002
WO 9734240	A	18-09-1997	AU 2585797 A		01-10-1997
			WO 9734240 A1		18-09-1997

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VN,YU,ZA,ZM,ZW

(74)代理人 100099623
弁理士 奥山 尚一
(74)代理人 100096769
弁理士 有原 幸一
(74)代理人 100107319
弁理士 松島 鉄男
(72)発明者 コンコラート，シリル
フランス国，９４２５０ ジャンティイ，リュ・ロベール・マルシャン ８
(72)発明者 セイラ，クロード
フランス国，７５００５ パリ，リュ・ロラン １２
(72)発明者 ポー，グレゴワール
フランス国，７５０１５ パリ，リュ・デュ・コタンタン ３２
(72)発明者 ティエノ，セドリック
フランス国，７５０１１ パリ，リュ・オベルカンフ １１５
(72)発明者 コタルマナク，アレクサンドル
フランス国，３５０００ レンヌ，リュ・ポール・ベール ３

Ｆターム(参考) 5B082 GA01 GA06
5J064 AA02 BA08 BC16 BD02