データベースシステム 113－34
（1997. 7. 15）

ストーリの内容記述に基づく映像の検索と要約

堀内 直明　上原 邦昭

神戸大学 工学部 情報知能工学科

ストーリ性を持つ映像の記述をするには，ユーザの主観によって映像の受け取り方が変化するために，ユーザの主観を吸収，反映できるような記述形式が必要とされる．本研究では，オブジェクト指向プログラミングで提案されたプロトタイプの概念を用いて映像の典型的なシーンを階層的に表現し，ユーザの主観の差異を吸収している．用意したプロトタイプで記述できないシーンは，既存のプロトタイプを修正したものを新たなプロトタイプとして利用している．本論文では，プロトタイプの概念とプロトタイプの記述形式について述べ，プロトタイプによる内容記述を利用した因果関係の検索や映像の要約作成について述べる．また，プロトタイプに基づいた開発環境 AMULET を用いたシステムの構築について検討する．

Semantic Representation of Video Content based on Prototype Annotation

Naoaki Horiuchi　Kuniaki Uehara

Department of Computer and Systems Engineering,
Faculty of Engineering, Kobe University

In order to enable the search and retrieval of video from large archives, we need a representation of video content. In our representation, the typical scene is annotated by hierarchical annotations with causal relationship. Annotation is based on prototype-based model which is proposed in the field of object-oriented programming. Prototype is a prefabricated object with predefined structure, contents, and behavior, from which new objects can be created by copying the prototype. By using the prototype annotation, our system enables users to create semantic annotations of video content, group related sets of scene descriptors hierarchically, use these descriptors to annotate video content, and reuse descriptive effort.

1 はじめに

映像などのマルチメディアデータを利用するために，マルチメディアデータベースの研究が盛んに行われているが，ストーリを持つ映像を意味的に取り扱う研究はあまりなされていない．これは，映像の内容記述がユーザの主観によって大きく差異を生じ，統一的な内容記述形式を求めることが困難なことによる．本研究では，映像の典型的なシーンの内容記述をあらかじめプロトタイプとして用意し，適当なプロトタイプをユーザに選択させてシーンの記述を行ない，ユーザの主観の差異を吸収した記述の統一性を実現している．さらに，用意されたプロトタイプは階層化がなされており，ユーザがシーンの記述を行なう際のプロトタイプ選択の労力を軽減している．

しかし，映像には様々なシーンが含まれており，すべてのシーンを記述できるだけのプロトタイプをあらかじめ用意するのは不可能である．そこで，用意されたプロトタイプでは記述できないシーンについては，新たなプロトタイプをユーザに定義させて，記述形式に柔軟性を持たせている．なお，新たなプロトタイプを定義する際も，既存の似かよったプロトタイプを修正すれば良いので，ユーザに多大な労力を強いることはない．

以上の考え方を実現するために，本稿ではオブジェクト指向モデルのプロトタイプ [1] を導入した映像の内容記述形式について検討する．ここで，オブジェクトはシーン記述のプロトタイプとして利用される．さらに，ソフトウェア開発環境 AMULET [2] を用いてプロトタイプを用いた因果関係の検索，要約の機能を持ったビデオデータベースシステムの実装について検討する．

2 プロトタイプを利用したストーリの内容記述

一定の構造を持たないビデオデータの内容記述を行なうためには，柔軟な記述形式が必要である．Lehnert [3] は affect unit を用いてストーリの記述を行なっており，ストーリの記述の際に現れるいくつかの典型的なシーンの記述をまとめている．本研究では，Lehnert のまとめた典型的なシーンの affect unit を用いた記述に注目し，さらに典型的なシーンの記述を一つのオブジェクトとしてとらえ，オブジェクト指向モデルの導入をはかる．

一般に，オブジェクト指向モデルは，複雑なオブジェクトの管理の容易さ，データのカプセル化，クラス階層に基づいた属性やメソッドの継承などの利点を持っているといわれている．しかしながら，ユーザの主観によって内容記述が異なるために，柔軟な記述が求められるビデオデータの記述には，データ構造の動的な変更の必要性が指摘されている [4]．

動的なデータ構造の変更を実現しているオブジェクト指向モデルとして，プロトタイプを用いたデータモデルがある．プロトタイプを用いたデータモデルは，静的なクラス定義を持たず，オブジェクトそのものを新たなオブジェクトの生成のためのプロトタイプとして利用することができるという特徴を持っている．新たなオブジェクトの生成は，オブジェクトのコピーで実現され，生成されたオブジェクトもまたプロトタイプとして振舞うことができる．

本研究では，Lehnert の記述に現れるいくつかの典型的なシーンの記述にプロトタイプを用いたデータモデルを導入し，プロトタイプを映像の適切なシーンに当てはめ，内容記述を行なう．プロトタイプを用いたデータモデルを導入し，それぞれの典型的なシーンの記述をオブジェクトとしてとらえることができ，オブジェクト指向モデルの利点を保ちつつ映像の内容表現を行なうことが可能になる．また，動的なデータ構造の変更によって，ユーザの定義する概念を新たにプロトタイプとして加えることも可能となる．まず，Lehnert の提案する affect unit を用いたストーリの記述について説明する．

Lehnert の提案する affect unit では，3 つの unit (OO, XX, MM) を用いて物語を記述している．ここで，OO はその unit の所有者にとって望ましいイベント，XX は望ましくないイベントを表している．MM は精神状態を表す unit である．たとえば，桃太郎の物語で桃太郎や村人たちが主体の場合の「桃太郎が鬼を退治する」イベントは村を平和にするための望ましいイベントなので，OO で記述されるが，鬼達が主体の場合は望ましくないイベントなので XX で記述される．これら 3 つの unit は，4 つの link(m, a, t, e) で結ばれ，ス

トーリの因果関係を表現している．m は動機づけ (Motivation)，a は実現化 (Actualization)，t は終結 (Termination)，e は等価 (Equivalence) を表す link である．

以上にあげた unit と link を用いて，最も基本となる affect unit (二つの unit と一つの link からなる組合せ) が 15 個用意され，それぞれに適したシーンの記述に用いられる．たとえば，成功を表す Success は MM $\overset{a}{\rightarrow}$ OO と書き，失敗を表す Loss は OO $\overset{t}{\perp}$ XX と書く．さらに，この 15 個の affect unit を組み合わせてより大きなシーンの記述が行なわれる．

図 1: プロトタイプの例

affect unit を用いたプロトタイプの例を図 1 に示す．この例は，左から「かなえられた要求」,「拒否された要求」,「失敗した要求」を表すプロトタイプとなっている.「かなえられた要求」とは，ある人物 A (左側の unit の主体) の精神状態に人物 B (右側の unit の主体) が影響を受けて，積極的にあるイベントを起こしたとき，そのイベントが人物 A にとっても望ましいイベントであった場合を表している．また，人物 B の起こしたイベントが人物 A にとって望ましくないイベントであったときは「拒否された要求」のプロトタイプ，人物 B の起こしたイベントが両者にとって望ましくないイベントであったときは「失敗した要求」のプロトタイプが用いられる．

プロトタイプを用いたストーリの記述には，以下のような利点がある．

- 前後に長い因果関係を直接プロトタイプ内で記述することができる．つまり，あるシーンの因果関係を記述するのに 2 つ以上のプロトタイプを用いる必要がない．
- プロトタイプを挿入削除する際に，他のプロトタイプを考慮する必要がない．
- ストーリの記述を理解した部分から徐々に詳細にしていくことができ，不完全な記述でも検索などを行うことができる．

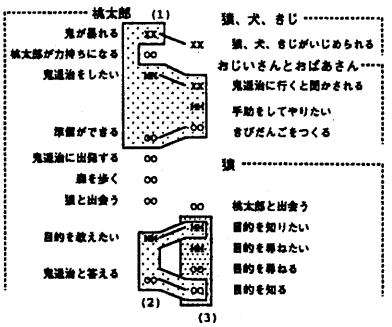

図 2: プロトタイプによる記述例

図 2 にプロトタイプを用いた記述例を示す．太い枠で囲まれた部分は 2 つ以上の unit からなるプロトタイプを表し，点線で囲まれた部分はその unit の主体を表している．たとえば，右下の unit OO は「猿が」「目的を知る」ことを表す unit である．また，(1) から (3) のプロトタイプは以下の場面を表現している．

1. Problem Resolution by Effective Coercion
   鬼退治に出かけて，鬼の乱暴を解決 (しようと) している．
2. Honored Request
   猿の要求 (桃太郎の目的を知る) が達成される．
3. Nested Subgoal
   第一の要求 (桃太郎の目的を知る) が，第二の要求 (桃太郎に目的をたずねる) の成功によって，達成される．

プロトタイプの中には，さらに他のプロトタイプが含まれているものも多くある．これは，あるプロトタイプの状況をより詳細に記述したプロトタイプが存在していることを示している．例えば，図 1 の網かけの部分は Request を表すプロトタイプであり，3 つのプロトタイプは Request を更に詳細に表したものである．このように，プロトタイプを階層化して大まかな記述からより細かな記述を選択すれば，記述の際のコストの軽減がなされ，多様な映像表現に対応できると考えられる．表 1 に Request を含むプロトタイプの例をあげる．

表の中で，Honored Request は「要求が満たされた」状態を表す Request の下位プロトタイプ，Promised Request Honored は「約束していた要

表 1: Request を含むプロトタイプ

- Request (を含むプロトタイプ)
  - Honored Request (を含むプロトタイプ)
    - Promised Request Honored
    - Obligation (を含むプロトタイプ)
      Serial Exchange
    - Simultaneous Exchange
    - Request Honored with Conditional Request
  - Denied Request (を含むプロトタイプ)
    - Double-Cross
  - Bungled Request (を含むプロトタイプ)
    - Promised Request Bungled

求が満たされた」状態を表す Honored Request の下位プロトタイプである.

ユーザがプロトタイプを用いて「犬が桃太郎に鬼ヶ島にいく約束をする」シーンの記述を行なう場合, まず Request の記述を参照する. Request の下位プロトタイプには Honored Request が存在するので, ユーザは目的のシーンを Honored Request を用いて記述することができる. 同じシーンを「犬が困っているところを桃太郎に鬼ヶ島へつれていってもらう」であると解釈した場合, ユーザは Unsolicited Help のプロトタイプを用いて記述することができる. Honored Request と Unsolicited Help は似た構造を持っており, 同じシーンを異なる解釈をして, プロトタイプを用いた記述形式がユーザの解釈の差異を吸収できるようになっている.

プロトタイプを利用したデータモデルでは, プロトタイプを修正してユーザの定義する概念を内容記述に反映させることができる. たとえば,「実際には失敗して欲しい要求」のシーンの記述を考えてみる. かぐや姫での, 求婚相手への結婚の申し出に対する交換条件などがこれにあたる. この場合,「かぐや姫は求婚相手に自分の要求を達成して欲しくない」ので, 図 1 の Honored Request や Bungled Request の記述をそのまま用いることはできず, 新たなプロトタイプを導入する必要がある. ユーザはプロトタイプ Bungled Request のコピーに修正を加え, 新しいプロトタイプ Wished Bungled Request を得ることができる (図 3).

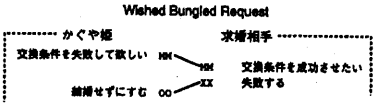

図 3: 新たなプロトタイプ

Wished Bungled Request は Bungled Request の持つ因果関係や階層構造において占める位置などをそのまま受け継ぎ, 新たに階層構造に追加される. 必要に応じて加えられた修正も含め, Wished Bungled Request の特性は, 将来 Wished Bungled Request から生成されるすべてのオブジェクトに継承される. さらに, 修正は動的に行なわれるので新しいプロトタイプ Wished Bungled Request を直ちに内容記述に利用することができる. このように, プロトタイプを利用したデータモデルを導入すれば, 必要とされるすべての概念を用意する必要がなくなり, ユーザの定義する概念を映像の内容記述に反映できるようになる.

3 プロトタイプを用いた機能

3.1 プロトタイプによる要約作成

プロトタイプによる要約を行うには, どのような unit がストーリ上で重要化を決定する必要がある. Lehnert によれば,「多くのプロトタイプから参照されている unit は重要な unit である」とされている. これは, さまざまな因果関係から参照されている unit が重要な unit であることを示している. 例えば, 図 2 では「(桃太郎が) 森を歩く」という unit より,「(猿が) 目的を知りたい」という unit の方がストーリ上で重要であることになる. 我々は,「ストーリの参照範囲が広いプロトタイプは重要なプロトタイプである」という基準を導入している.

参照範囲の広いプロトタイプとは, プロトタイプの unit が, ストーリ上の時間の流れの上で遠く離れているものをいう. たとえば, 図 2 では「(桃太郎が) 森を歩き, 猿と出会う」というプロトタイプよりも,「鬼が暴れるので, (桃太郎は)(おじいさんとおばあさんに) 準備してもらって鬼退治に出かけた」というプロトタイプの方が参照範囲の広いプロトタイプである. ストーリ上の時間の流れ

からみて，参照範囲の広いプロトタイプは参照範囲の狭いプロトタイプよりも上位のプロトタイプであり，これらの基準にそって要約をほどこすと，図 2 は図 4 のように書き換えることができる．

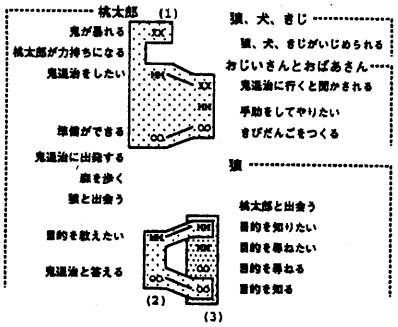

図 4: 要約されたプロトタイプによる記述

プロトタイプを用いた記述からの要約の予備実験として，他のプロトタイプからの参照数と参照範囲の広さにあたる合う一定の基準を設けてプロトタイプを残し，残ったプロトタイプの参照している映像のフレーム数を計算したところ,「桃太郎」の場合 15421 フレームから，7214 フレームになった．この時のプロトタイプ数の変化を表 2 に示す．

表 2: 要約におけるプロトタイプ数の変化

総数	参照数で選択	参照範囲で選択
84 個	24 個	11 個

表 2 の「参照範囲で選択」したプロトタイプのうち 7 個が「参照数で選択した」プロトタイプと同じプロトタイプであった．つまり,「参照範囲で選択」されたプロトタイプが，Lehnert の基準に照らしても重要度が高いことがわかる．

3.2 プロトタイプによる因果関係の検索

あるシーンの因果関係の検索は，そのシーンの属する unit とプロトタイプを参照し，基本的には，そのプロトタイプ内で因果関係を追うことになる．たとえば，図 5 では「(桃太郎が) 鬼退治と答えた」のは「(桃太郎が) (猿に) 目的を教えたかった」からであり,「(猿が) 目的を知りたい」ので「(猿が) 目的を尋ねた」ためである．

このように，プロトタイプ内での因果関係は明示されているので，プロトタイプ内の関係を追えば因果関係に関する検索が可能である．また，プロトタイプ内の関係には MM という精神状態を表す unit があるが，映像として現れる部分は，普通，あるキャラクターが起こしている行動であり，精神状態を表す MM の unit をある行動の理由とすることはない．たとえば,「(桃太郎が) 鬼退治と答えた」理由は上述した通りであるが,「(桃太郎が) (猿に) 目的を教えたかった」unit や「(猿が) 目的を知りたい」unit は映像を持たないので，実際に検索結果として現れる映像は「(猿が) 目的を尋ねた」unit の持つ映像である．

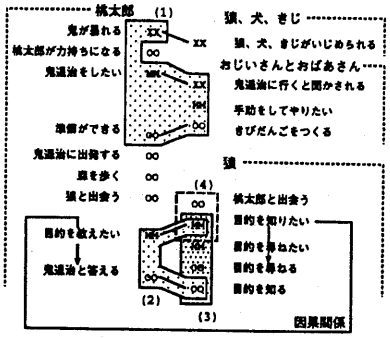

図 5: プロトタイプによる記述の因果関係

一方，原因や結果を複数のプロトタイプにまたがって追うような場合がある．たとえば,「(猿が) 目的を尋ねた」ことの原因を探る場合は，図 5 の (2) のプロトタイプだけでは原因となるシーンを探し当てることはできない．この場合，(2) のプロトタイプから「目的を知りたい」ことが原因であることは分かるので,「目的を知りたい」unit を共有するプロトタイプ (4) を参照すれば,「(猿が) 桃太郎と出会う」シーンが「(猿が) 目的を知りたい」ことの原因であることがわかる．したがって,「(猿が) 桃太郎と出会う」シーンが「(猿が) 目的を尋ねた」シーンの原因として参照される．

4 プロトタイプを用いたシステムの構築

本システムの構築には AMULET を用いている．AMULET はプロトタイプの概念をサポートし，グラフィカルユーザインタフェイスの容易な開発が可能な C++ の開発環境である．システムの構成を図 6 に示す．

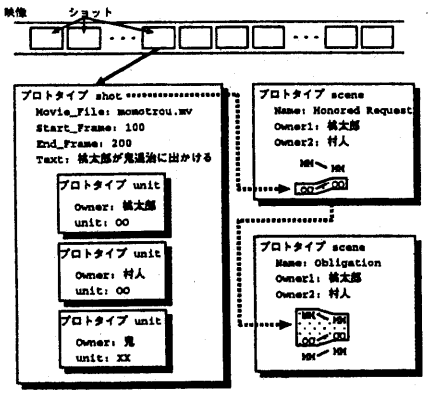

図 6: システムの構成

図 6 で，映像からの実線の先にあるのが shot を表すプロトタイプである．shot を表すプロトタイプは動画像ファイルを表す Movie_File，動画像の区間を表すフレーム番号の Start_Frame と End_Frame，shot についての記述 Text から構成されている．また，shot はプロトタイプ unit を parts として持っている．

また，プロトタイプ unit はプロトタイプ shot が登場人物にとってどのような shot であるかを表している．プロトタイプ scene の記述は，scene の表すプロトタイプ名 Name とプロトタイプの affect unit による記述，unit の所有者 Owner1, Owner2 からなる．さらに，scene の affect unit の記述はプロトタイプ unit を parts として持ち，プロトタイプ unit から実際の動画像を参照することができるようになっている．

因果関係に及ぶ検索の機能は各プロトタイプの持つメソッドに定義されており，メソッドを呼び検索が行なわれる．なお，要約の作成はシステムがプロトタイプを参照して行なうようになっている．

上述したプロトタイプは基本的にシステムの実行中のみ存在しており，システムの終了とともにそのデータは消去されるが，データベースとしてシステムを構成するには，プロトタイプを含んだすべてのオブジェクトが永続性を持っていなければならない．AMULET はデータベース構築のための言語ではなく，オブジェクトの永続性はサポートしていないが，宣言されたオブジェクトを記録する機能がある．したがって，システムを終了する前に存在するすべてのオブジェクトのデータをセーブし，再びシステムを起動する前に以前のデータをロードすれば，オブジェクトの永続性を実現することが可能となる．

5 おわりに

本研究では，映像上の典型的なシーンをプロトタイプとして，ユーザの主観を吸収，反映しつつ映像のストーリを記述するために手法についてのべた．また，プロトタイプの記述には Lehnert の提案する affect unit を用い，プロトタイプの表現による階層化について述べた．さらに，AMULET を用いたシステムの構築について検討した．今後は実際のシステムを完成させ，検索や要約などの実験を行う予定である．

謝辞

本研究は，文部省科学研究費重点領域「高度データベース」の援助を受けて行なわれたものである．本研究において，アニメ「まんが日本昔ばなし」を使用するにあたり，学術利用を許可して下さった愛企画センター 川内彩友美代表取締役，および大手前女子短期大学生活文化学科 浦畑育生助教授に感謝致します．

参考文献

[1] Blaschek, G.: Object-Oriented Programming with Prototypes, pp.91–119, Springer-Verlag (1994).

[2] Myers, B. A., et al.: The Amulet Enviroment: New Models for Effective User Interface Software Development, Technical Report CMU-CS-96-189, Department of Computer Science, Carnegie Mellon University (1996).

[3] Wendy, G. L.: Affect Units and Narrative Summarization, Report # 179, Department of Computer Science, Yale University (1980).

[4] Oomoto, E. and Tananka, K.: OVID: Design and Implementation of a Video-Object Database System, IEEE Transactions on Knowledge and Data Engineering, Vol.5, No.4, pp.629–623 (1993).