日本語シナリオからのアニメーションの生成

舟渡信彦† 吉川耕平††
花田恵太郎†† 宮本雅之††

本論文では，アニメーションの作成・修正を簡便化することを目的としたアニメーション作成支援システムを提案する．本システムは，アニメーションの作成作業の一部である「登場物の動き」の記述を日本語で行うことを可能にする．このためにシステムは，日本語のシナリオを入力とし，(1)アスペクトを用いた登場物の動作間に成り立つ順序関係の抽出，(2)動作に付随する状態変化知識を用いた順序関係を補正するための推論，(3)得られた関係からの事象駆動方式にもとづくアニメーション記述言語プログラムの生成，を行う．日本語によるシナリオの記述を許すことで，利用者が習得しなければならないシステムに関する知識は軽減される．また，従来のフレームごとの画像を作成する方法に比べて，作成したいアニメーションを直接的に表現することができるため，試行錯誤が容易になる．これらの特徴により，本システムは専門的な知識をもたない利用者が自らアニメーション作成作業を行う場合に有効である．

Automatic Animation Generation from Scenarios in Japanese

Nobuhiko Funato,† Kouhei Yoshikawa,††
Keitaro Hanada†† and Masayuki Miyamoto††

We introduce an animation authoring system which enables users to describe the motions of animation characters using natural language scenario texts in Japanese. The system takes the following steps: (1) It extracts the temporal ordering information between the motions of characters from the aspects of verbs in the input text provided by the user, (2) Extracted temporal orderings are then refined by the knowledge corresponding to the state changes caused by the motions, and (3) The ordering information is translated into a program text of EASY, an object-oriented event-driven animation language developed by the authors. Users of our system can directly specify their intentions and easily specify the motions of animation characters using natural language texts, without knowing complicated usage of conventional authoring systems.

1. はじめに

近年のマルチメディア機器の進歩に伴い，人間が直観的に理解しやすい動画（アニメーション）情報の活用に対する期待はますます高まっている．一般にアニメーション情報の作成コストは高いため，計算機を利用した作成支援システムの研究が盛んに行われている．計算機によるアニメーションの作成作業は，まず登場物の形状デザインを行い，次いで登場物やカメラの動きを指定し，最後にレンダリングを行う，という三つの過程からなる．このうち，アニメーションのストーリを決めるものは動きの指定であり，何度も試行錯誤が繰り返される．したがって，この部分の効率が全体の作成効率を決定する．

既存のアニメーション作成支援システムでは，フレームごとの画像の作成が，ストーリの記述に相当することが多い[1),2)]．このような記述方法では，ストーリの進行を時間どおりに表現できる反面，利用者は作成したいストーリの内容とは直接関係のない，画像作成に関するシステムの操作方法を知る必要がある．また，利用者はストーリ中の事象の発生をフレームごとの画像に置き換えて表現することになるので，後に事象の順序関係や因果関係を修正するときは，それが部分的な修正であっても全体的なフレームの作り直しを強いられるという問題がある．

本論文で提案するアニメーション作成支援システムでは，末端の利用者が自らアニメーションを作成できることを目指している．このようなシステムは以下の点を満たすべきである．

● システムに関する専門知識をもたない利用者もシ

† シャープ株式会社 A1175プロジェクト
A1175 P. T., SHARP Corporation
†† シャープ株式会社技術本部情報技術研究所
Information Technology Research Laboratories, Corporate R & D Group, SHARP Corporation

ナリオ（ストーリの記述）を作成できる．

●部分的なシナリオの修正を簡便に行える．

筆者らは，後者の要件を満たすアニメーション記述言語 EASY をすでに提案している[3]．EASY は事象駆動方式のプログラミング言語で，シナリオを時刻ではなく事象の同期条件にもとづいて記述できる．記述されたプログラムを EASY 処理系で実行することによって，対応するアニメーションが実時間生成される．しかし，EASY の利用にはプログラミングの知識が必要であり，単独で利用したのでは前者の要件を満たさない．

本論文では上記の要件を満たすために，日本語で記述されたシナリオから EASY のプログラムを自動生成することによって，アニメーションを生成するシステムを提案する．日本語によるシナリオの記述を可能にすることで，利用者が知らなければならない専門的な知識は大幅に軽減される．また，日本語のシナリオでは，従来のフレームごとの画像を直接作成する方法に比べて，登場物の動きをより直接的に表現できる．これによって，シナリオの作成が簡単になり，試行錯誤を行うことが容易になる．

また本論文では，EASY のプログラムを生成するために必要となる，日本語シナリオから動作事象の順序関係を認識するための一つの手法を示している．特にアニメーションでは動作が開始から終了までの間にわたって可視化されるため，すべての動作事象（開始と終了）とその間に成り立つ関係（順序関係や因果関係）を認識しなければならない．一方，自然言語の叙述は一般に不完全であることから，字面上からすべての関係を得ることはできない．このため，部分的に認識された関係から未知の関係を推論する必要がある．

このような問題に対してはいくつかの研究がなされている．Winograd[4]の SHRDLU は，英語の文章から，動作（積木の移動）によって変化する状態の開始時刻と終了時刻を認識する．この際，明示的に与えられない時刻は推論によって補う．ただし，SHRDLU は同時に複数の積木が動かないことを仮定しており，複数の登場物が動作するアニメーションのシナリオには対応できない．Kowalski[5]は，ある状態の開始や終了を表す事象の間に成り立つ半順序関係から，与えられた事象間の未知の関係を推論するアルゴリズム（イベント計算）を示した．しかし，イベント計算は後向き推論にもとづいており，すべての事象と関係を求めることには向いていない．Allen[6]は，時区間の間に成り立つ順序関係を考え，関係の追加にともない，任意の二つの時区間の間に成立しうる関係の集合を更新する拘束伝搬アルゴリズムを示した．このアルゴリズムはすべての順序関係を求めることはできるが，時間間隔等の定量的情報を表現しづらいという問題がある．

本システムの順序関係の認識手法では，アスペクトや動作の継続性および接続句の意味を利用して，動作事象間の順序関係を抽出する．次いで，抽出された関係の不完全な部分を，動作にともなう状態変化の知識を用いた推論によって補正する．本手法では，事象の時刻ではなく事象間の半順序関係を認識するため，複数の登場物が並行に動作するアニメーションのシナリオにも対応できる．また，事象間の時間間隔も自然に表現できる．

以下，2章ではアニメーション記述言語 EASY の概要について，3章ではシステムの全体構成について，4章で日本語シナリオから登場物の動作間に成り立つ半順序関係を認識する手法について，5章では実験システムの概要と実行例について述べる．

2. アニメーション記述言語 EASY の概要

EASY[3),7)] (Event driven Animation SYstem) は筆者らが提案したアニメーション記述用のプログラミング言語である．ここでは紙面の都合上，EASY におけるストーリの記述モデル（EASY モデル）とアニメーションの記述方法の概略についてのみ述べる．

2.1 EASY モデル

EASY モデルでは，アニメーションを自律的に動作する複数の登場物（アクタ）が相互に影響を及ぼし合いながら並行動作する系と見なす．個々のアクタの動作は，対応するプログラム手続き（メソッド）によって記述する．メソッドは，座標値や表示イメージ等の表示属性等を含むアクタの内部状態の更新を行う．

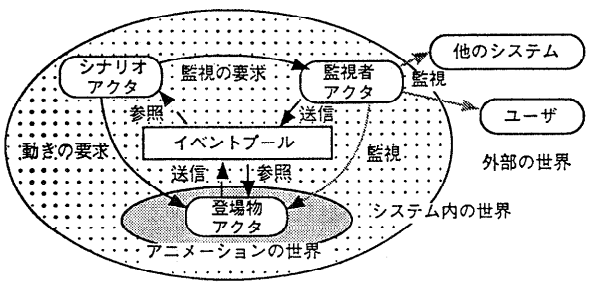


図 1 EASY モデル
Fig. 1 The EASY model.

複数のメソッドが大域的な時刻の進行に同期して並行に実行される．EASY の処理系は(1)すべてのアクタについて開始しているメソッドを微小時区間分ずつ実行する，(2)更新された表示属性を反映して画面の再表示を行う，という二つの処理を繰り返すことによってアニメーションを生成する．

アクタ間の相互作用を表現するために，アクタがメソッドを実行中に，互いに同期をとるためのメッセージ通信を行うことができる．このメッセージには以下の2種類がある：

デマンド型メッセージ（デマンド）

あるアクタに対して送信され，メソッドの開始・終了を要求するもの．受信アクタは要求に従い指定されたメソッドを開始または終了する．デマンドは動作の開始や終了の間に成り立つ順序関係を表現できるので，おもに大局的なストーリの記述に用いられる．

イベント型メッセージ（イベント）

送信されると大域的な領域（イベントプール）に蓄えられ，ある事象が発生したことを表すもの．他のアクタのメソッドは，蓄えられたイベントの内容を参照し，条件分岐を行うことで，自律的に次の処理を決定できる．アクタ間の関係変化（衝突等）や系外事象（キーボード入力等）はイベントによって表現することができる．

このように，EASY モデルでは2種類のメッセージによって事象の順序関係や因果関係を表現することで，事象駆動的にストーリを記述できる．このため，事象の発生時刻を明示的に記述しなくてもよい．

2.2 EASY によるアニメーションの記述

EASY によるアニメーションの記述は，以下のプログラミングを行うことに相当する．

個々の登場物の動きの記述

登場物をアクタに割り当て，その動作を表すメソッドを記述する．EASY はオブジェクト指向言語の一種であり，クラス間の継承によってプログラムの再利用性を高めている．そこでこのことは，適当なメソッドをもつクラスを探し，そのインスタンス（アクタ）を生成することで済む場合が多い．

登場物の動きの関係（順序関係・因果関係）の記述

シナリオアクタ（監督に相当し，他のアクタに動作の開始・終了を適宜要求する）のメソッドを記述する．記述の便宜を図るため，(1)シナリオアクタ，(2)衝突などの複数のアクタの関係変化を監視するアクタ，(3)キーボード入力やタイマ割り込みなどの系外事象を監視するアクタ，のクラスが用意されている．これらのアクタはアニメーションには表示されない．

前者は，関数法・キーフレーム法・モーションパス法等[2]の既存の CG 技術を用いて記述し，ライブラリ化して再利用することが想定されている．後者は大局的なストーリの進行を表すもので，個々のアニメーションごとに必要である．

3. アニメーション生成システムの構成

本システムでは，まず**日本語シナリオ解析部**が，日本語シナリオに記述されたストーリ（登場物の動きの順序関係・因果関係）を表す EASY のプログラムを生成する．次いで **EASY 処理系**が，このプログラムを実行することで，対応するアニメーションを実時間で生成する．利用者は，(1)日本語シナリオの作成・修正を行う，(2)必要に応じて EASY のプログラムを修正する，(3)出力アニメーションを検証する，という作業を繰り返すことで，アニメーションを作成できる．

本システムでは以下のことを仮定している．

- **日本語シナリオ**は，動作を表す動詞（動作動詞）とそれに対する連用修飾語からなる文（動作文），および順序関係・因果関係を表す接続詞・接続助詞から構成される．
- **辞書**中の名詞（登場物）に対しては**表示イメージ**が，動詞（動作）に対しては EASY の**メソッド**と後述する**状態に関する知識**が用意される．
- 動詞のテンス（時制）は非過去時制に限定する．

本システムは，日本語シナリオから既知の動作の間の関係をストーリとして認識するが，未知の動きや形状の描写は認識しない．また，情景描写の認識[8]も行わないので，背景の指定（例えば登場物の初期配置の

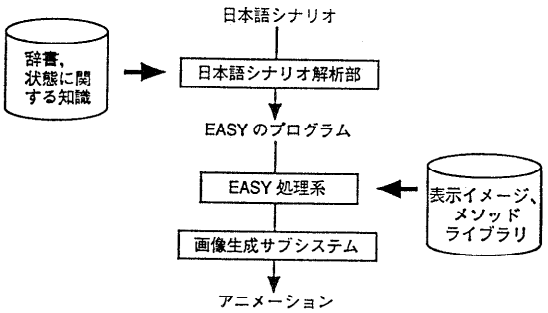


図 2 アニメーション生成システムの構成
Fig. 2 Structure of the Animation Generation System.

指定）が必要ならば，あらかじめ EASY で記述するなどして与える必要がある．

なお，テンスの限定は順序関係の認識処理の前提になっているが，実用に供されている絵コンテの注釈でもほとんどの文が非過去時制であり，ストーリの記述力を損なう仮定ではないと考えられる．

4. アニメーション生成のためのシナリオ理解

4.1 概　　要

アニメーションを生成するためには，すべての動作事象（開始と終了）とその間に成り立つ関係（順序関係や因果関係）を認識する必要がある．このためには，日本語シナリオに現れる以下の省略を補わなければならない．

(1)時点の省略

Aが歩き始める．…．そのときAは座っている．（「座る」の開始時点以前に「歩く」の終了時点があることは表層上には現れない）

(2)順序関係の省略

Aが論文をコピーし始めると，Bはコピー機へ歩き始める．その後，注文書をコピーする．（コピー機が1台しかない状況では「Aがコピーする」が終了した後に「Bがコピーする」が始まるが，明示的には表現されない）

(3)動作の省略

Aは自分の席に座っている．…．そのとき，Aは見積書をコピーしている．（「コピーする」の開始前に「Aがコピー機の場所に移動する」ことが省略されている）

日本語シナリオ解析部は，このような要求を満たすために以下に述べる処理を行う．まず，動作動詞のアスペクト，動作動詞句の継続性，および単文間の接続表現の意味から，動作の開始時点や終了時点の間に成り立つ順序関係を抽出する（4.2節，4.3節）．(1)の省略はこのとき補完される．次いで，状態に関する知識（動作にともなう状態変化，および矛盾を表す状態の組合わせ）を用いた推論を行って，(2)と(3)の省略を検知し補完する（4.4節）．ここでいう推論は，初期状態も目標状態も与えられずに中間の状態変化の系列が部分的に与えられたときに，省略を適当に補うことで，矛盾のない初期状態とすべての状態変化の系列を求める探索処理に相当する．最後に，省略の補完にともない補正された順序関係から，ストーリの記述を表す EASY のプログラムを生成する（4.5節）．

本手法では，矛盾のない状態を求めるために必要以上に省略の補完を行わないため，認識される時点間の順序関係も必要最小限の半順序関係である．このことによって，EASY の並行動作記述能力を生かした，複数の登場物が並行に動作するプログラムを生成することが可能になる．

4.2 動作文の解析

日本語シナリオ解析部は，入力される動作文を単文ごとに解析することで，以下に述べる情報を抽出する．

4.2.1 動作の意味を表す格フレーム

視覚的な動作の意味を抽出するために，特に空間や時間に関する始点・終点・間隔・経路（空間のみ）を格にもつ深層格フレームを生成する．

この深層格フレームは以下の処理に用いられる．

- パターン照合による二つの動作の同一性の判定（4.3節）
- 動作によって変化する状態の具体化（4.4.1節）
- EASY 言語プログラムに埋め込まれるメソッド呼び出しパターンの生成（4.5節）

4.2.2 アスペクトから認識できる順序関係

アスペクト（～ル，～テイル，等の動詞語尾に現れる表層上の形式）によって，時間軸上のある観測時点から動作時区間の開始前・実行中・終了後のうちのどの部分を観測しているかを解釈できる．

すなわち，動作動詞句は継続性に着目することで，瞬間型（完了の意味を含むもの）・限定継続型（継続時間を限定できるもの）・非限定継続型（継続時間を限定できないもの），の3種類に分類できるが，動作時区間のどの部分を観測しているかは，基本的にはアスペクトと動詞句の継続性の組合わせからわかる[9),10)]（**図3**）．

動作動詞は単独では上記のいずれかの継続性をもつが，共起語（その場所まで，30秒間，1分かかって，

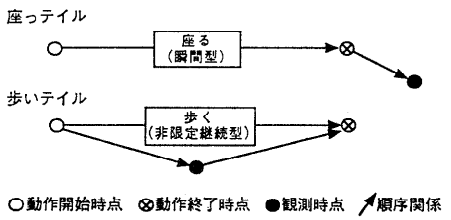


図3 アスペクトにもとづく順序関係の認識例
Fig. 3 Recognitions of temporal orderings from aspect information.

等の連用修飾句）をともなうと異なる継続性を意味することがある．そこで，動詞句の継続性を動詞と格の組合わせから検索できるように辞書を構成している．

4.2.3 前後の文との関係

動作文のテンスを非過去時制に限定していることから，観測時点は発話時点と同等になる．このため，観測時点の順序は同時か発話順（文の順番）のいずれかになる．

そこで，接続表現なしで単文が並ぶ場合は，二つの観測時点の関係を同時に近い発話順であるものと解釈している．接続表現が順序関係を表す場合（～と同時に，そして，その後，～したら，等）は，二つの文の観測時点の順序関係をそのまま表すものと解釈する．

接続表現が因果関係を表す場合（～したら，～すれば，等の条件文）は，アニメーションのシナリオでは条件の成立が含意されることが多い．そこでこの場合は，条件節の動作事象が起こった後に帰結節の動作が実行を開始する，という順序関係として解釈する．

本システムでは上記以外の場合として，(1)連続する条件文によって複数の帰結節からの選択を表す場合や，(2)条件節が衝突やキー入力である場合，を因果関係として解釈し，EASY のイベントで表現することも行うが，本論文では割愛する．

4.3 動作間の順序関係の認識

動作間の順序関係は，アスペクトから認識した順序関係を，連続する文の順序関係に従って順次接続していくことによって得られる．この関係を，時点をノード，順序関係をエッジとする DAG (Directed Acyclic Graph) によって表現し，動作順序グラフと呼ぶ(図4)．

動作順序グラフにおいて，ある動作の開始時点と終了時点の間にあるエッジを動作エッジと呼ぶ．そうでないものを遅延エッジと呼ぶ．エッジは継続時間（下限値と上限値の組で表される）を属性情報としてもつ．接続表現で指定される遅延時間（～秒後，等）や，時間間隔格で指定される動作の継続時間は，この属性に記憶される．エッジの継続時間属性は，後述の EASY のプログラムの生成の際に用いられ，アニメーションの時間的な制約を表す．

動作順序グラフの構築の際，「～しはじめる．…．そのとき，まだ～している．」のように，過去に同一の動作が観測されている場合，すでに生成された開始時点や終了時点を用いる必要がある．ここでは二つの同じ主格で同じ種類の動作について，(1)同じ格の値が異なる，(2)アスペクトから双方の時区間が重複しないとわかる，(3)間に並行に実行できない別の動作がある，という条件がすべて不成立であれば，二つの動作が同一と判断する．

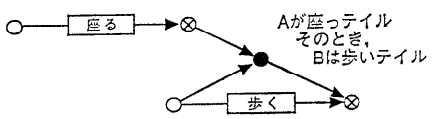


図 4 動作順序グラフの例
Fig. 4 An example of Motion-Order Graph.

また，4.1節の(1)の省略を補うために，連続する同一主格の異なる種類の動作について，(1)アスペクトから双方の時区間が重複する可能性がある，(2)双方の動作を並行に実行できない，という二つの条件が成立すれば，前の動作の終了時点と後ろの動作の開始時点の間に順序関係を追加する．

以上の二つの処理は，観測時点の関係が同時か発話順かに単純化されていることを利用しており，Kowalski のイベント計算[5]による継続性の推論と同じ効果が得られる．

4.4 状態制約による動作順序グラフの補正

前節で動作順序グラフとして得られる関係は不完全であるため，状態に関する知識を用いてグラフ上の状態の矛盾を調べることで，省略された順序関係や動作の検知と補正を行う．

4.4.1 状態に関する知識

状態に関する知識には以下の2種類がある．

- 動作にともなう状態変化の知識
- 状態間の矛盾を表す知識

```
(defchange motsu-1
  :caseframe          ((subj ?subj)
                       (verb MOTSU)
                       (obj  ?obj)
                       (pos  ?pos))
  :filter             ((isa  ?subj HUMAN))
  :preconds           ((not-hold ?subj)
                       (at ?obj ?pos))
  :dels               ((at ?obj ?pos))
  :holds              ()
  :adds               ((hold ?subj ?obj))
  :default-duration   (*instant-default-Lbound*
                       *instant-default-Ubound*)
  :method             Motsu-S/O/P
  :self-terminate-p t
)
```

図 5 動作にともなう状態変化の記述例
Fig. 5 An example of state change description for a motion.

動作にともなう状態変化の知識は，動作に関する他の言語外知識とともに記述する（**図5**）．

状態変化の知識は，preconds・dels・holds・adds という4種類の状態の集合によって表現する．ここで，状態は以下のパターンで記述される．

(〈状態名〉〈引数〉…)

例えば図5で，adds は (hold ?subj ?obj) という一つの状態からなる集合である．？で始まるシンボルは変数で，この状態は「?subj（主格）が ?obj（目的格）をもっている」ということを表す．caseframe は深層格フレームのパターンを表しており，入力動作文を解析して得られた格フレームと単一化される．状態記述中の変数は，この単一化によって具体化される．

preconds・dels・adds の意味は STRIPS[11]の条件記述と同様である．これらに加えて，holds によって継続型の動作動詞を実行中の状態を表す．すなわち，動作を実行する際には(1)開始時点では preconds が成立している必要がある，(2)開始時点でそれまでの状態から dels が削除され holds が追加される，(3)終了時点では holds が削除され adds が追加される，ということを状態変化の知識によって表す．

状態間の矛盾を表す知識は以下の述語 contradicts によって記述する．

(contradicts 〈判定関数〉〈状態1〉〈状態2〉)

この記述は，〈状態1〉と〈状態2〉が同一時点で成立するとき〈判定関数〉が真ならば二つの状態が矛盾である，ということを表す．〈状態 n〉は状態の記述を表し，ある時点の状態と単一化可能な場合に，矛盾かどうかの判定が行われる．例えば，「同時に異なる位置にいることはできない」という知識は，組み込みの関数 not-equal を用いて以下のように記述する．

```
(contradicts (not-equal ?pos 1 ?pos 2)
             (at ?subj ?pos 1)
             (at ?subj ?pos 2))
```

4.4.2 動作順序グラフの補正アルゴリズム

省略された順序関係や動作の検知と補正は，以下の手順で行う．

(1)アニメーションの開始および終了時点の追加

開始時点よりも早い時点や，終了時点より遅い時点が認識されていない動作がある．これらの動作がいつ開始あるいは終了するかを明瞭にするため，動作順序グラフにアニメーション全体の開始時点と終了時点を加え，それぞれが順序関係の下限（開始側）と上限（終了側）になるようにエッジを追加する．

(2)状態空間の計算

動作順序グラフ上で，二つのノード（時点）間に経路があり，かつどの経路にも動作エッジが含まれないという関係があるとき，この2点間では状態変化がない．このような時点の集合（上記の関係の推移的閉包）を状態空間と定義する．グラフを走査することで，すべての状態空間を計算する．

(3)各状態空間の状態計算と矛盾の補正

状態空間をノード，動作エッジをエッジとするグラフ（DAG）を状態空間グラフと呼ぶ．各状態空間中に属する状態の集合は，動作にともなう状態の変化によって定まる．例えば，ある動作の終了後の状態空間では，直前の状態空間の状態集合から継続状態（holds）が削除され追加状態（adds）が追加される．また，状態空間では続いて起こる動作の前提状態（preconds）が成立している必要がある．

各状態空間に属する状態の集合を，状態空間グラフから計算する．このために，状態空間ノードを動作エッジで表される順序関係でトポロジカルソートして，開始側ノード（アニメーションの開始時点が含まれる）から順に走査する．走査する際，動作エッジ上の状態変化を模擬しながら，状態空間ごとの状態の集合を計算していく．こうして，すべての状態集合を計算できれば，補正処理は終了する．ただし，ある状態空間において，状態間の矛盾か前提集合の不足が検知された場合は，以下の補正を行う．

得られた状態間に矛盾があり，かつ状態空間ノードに複数の入力エッジがあるときは，すべてのエッジに対して回帰（ある状態に STRIPS 規則を後向きに適用することで，その状態が過去のどの時点から成立するか，あるいはどの時点で矛盾するかを検査する処理）[12]を行って，並行するエッジ間の状態変化の干渉[13]の有無を検査する．干渉があるときは，順序関係が省略されているものと解釈し，NOAH[13]と同様に順序関係を補うことで干渉を解消する．その後(2)から再処理を行う．干渉がないか，複数の入力エッジがない場合は，動作が省略されているものと解釈する．この場合，矛盾を削除し，矛盾を起こさない状態を追加する動作を，図5の動作に関する知識から検索する．検索された動作を具体化して動作順序グラフに挿入した後，(2)から再処理を行う．

得られた状態間に矛盾はないが，動作の前提状態が不足しているときは，回帰を行って，その状態が可能な限り早くから成立するという仮説を立てる．その後(3)から再処理を行う．

補正処理が終了したとき，動作順序グラフ上では矛盾のない動作間の半順序関係が，また状態空間グラフ上ではアニメーション開始時と終了時の状態集合が得られる．

なお，探索空間を限定するために，挿入される動作を選択する際に以下の制限を設けている．

- できるだけ多数の矛盾状態を解消するような動作を選ぶ．
- 複数の挿入候補があるときは利用者に選択を促す．
- 一つの動作を追加することで矛盾が解消できなければ，補正不能として処理を打ち切る．

例えば，以下の文章からは**図6**(a)の動作順序グラフが得られる．

(動作1) 研究者が論文をコピーし始めると，
(動作2) 庶務係はコピー機へ歩き始める．
(動作3) その後注文書をコピーする．

このグラフに処理(1)を施すと同図(b)が得られ，さらに処理(2)によって同図(c)の状態空間（曲線で囲まれた部分）が計算される．次いで，処理(3)によって各空間の状態集合を計算していくと，並行する(動作1)と(動作3)は，1台しかないコピー機を同時に使用するという矛盾を起こすことから，干渉していることがわかる．この干渉を回避するために，図6(d)のように二つの動作間に順序関係が追加され，(動作1)は(動作3)の開始前に終了するようになる．

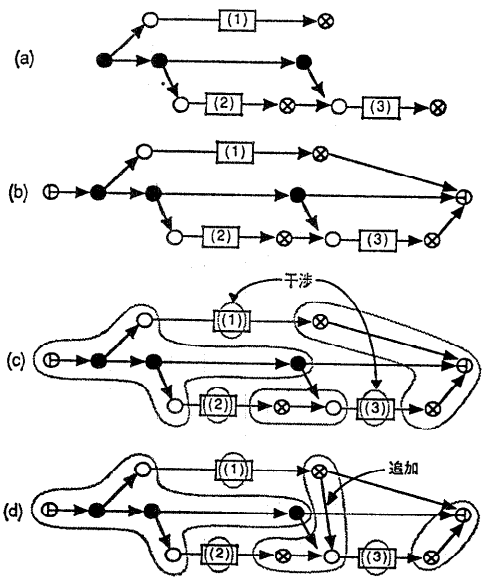


図 6 動作順序グラフの補正の例
Fig. 6 An example of refinement of a Motion-Order Graph.

4.5 EASY プログラムの生成

補正された動作順序グラフをデータフローグラフとして解釈することで，2.2節の要求にそったEASYのプログラムを容易に生成できる．すなわち，グラフのすべてのノードにシナリオアクタを割り当て，各シナリオアクタに対して以下の手順で動作するメソッドmark-nodeを生成する．

1. 発火条件の待ち合わせ
   すべての入力エッジの開始ノードからmark-nodeメッセージ(mark-nodeメソッドの実行要求)が送られてくるのを待つ．すべてのメッセージを受けとると，入力エッジに対応する継続中の動作(以下の2(c)で開始したもの)がある場合はその終了を要求してから，2の処理へ進む．
2. ノードの発火
   すべての出力エッジに対して，以下の処理を並行に実行する．
   (a)遅延エッジの場合
   指定された遅延時間(下限値)の経過を待って，エッジの終端ノードへmark-nodeメッセージを送信する．
   (b)自律的に終了する動作を表す動作エッジの場合
   このエッジは「座る」等の瞬間型あるいは限定継続型の動作を表す．対応するアクタの動作の開始を要求し，実行が終了してから，エッジの終端ノードへmark-nodeメッセージを送信する．
   (c)自律的に終了しない動作を表す動作エッジの場合
   このエッジは「歩く」等の非限定継続型の動作を表す．対応するアクタの動作の開始を要求し，指定された継続時間(下限値)が経過すると，エッジの終端ノードへmark-nodeメッセージを送信する．

アニメーションの開始時点に対応するシナリオアクタにmark-nodeメッセージを送信することで，アニ

```
(a-defmethod <node> mark-node-n1
  ((wait-count 1))
  ((start
     (send '研究者-1、'座る '(:pos '椅子-1)))
   (run
     (check-child-end '研究者-1 '座る 'n2))))

(a-defmethod <node> mark-node-n2
  ((wait-count 1))
  ((start
     (send '研究者-1 '立つ nil))
   (run
     (check-child-end '研究者-1 '立つ 'n3))))

(a-defmethod <node> mark-node-n3
  ((count-1 1) (count-2 1) (wait-count 2))
  ((start)
   (run
     (check-time-end count-1 'n11)
     (check-time-end count-2 'n4))))
```

図 7 生成される EASY プログラム（部分）
Fig. 7 A part of generated program text of EASY.

メーションの生成を開始できる．

生成されるプログラムの例を**図 7** に示す．ここで，登場物とアクタおよび動作とメソッドの割り当ては，深層格フレームを機械的に構文変換することで行う．

5. インプリメンテーション

5.1 実験システムの概要

3章で述べたアニメーション生成システムを Common Lisp を用いて Sun 4/260 上に試作した．なお，画像生成サブシステムはプロセス間通信で結ばれたグラフィクスワークステーション上に作成した．

辞書情報のうち，接続詞・接続助詞は約 20 種類，動作動詞は約 50 種類（対応する EASY のメソッドは 20 種類）である．また，文章中の話題を「研究所内の人の動き」に限定することで名詞の語彙を選び，対応する EASY の表示イメージやメソッドおよび背景記述を作成した．

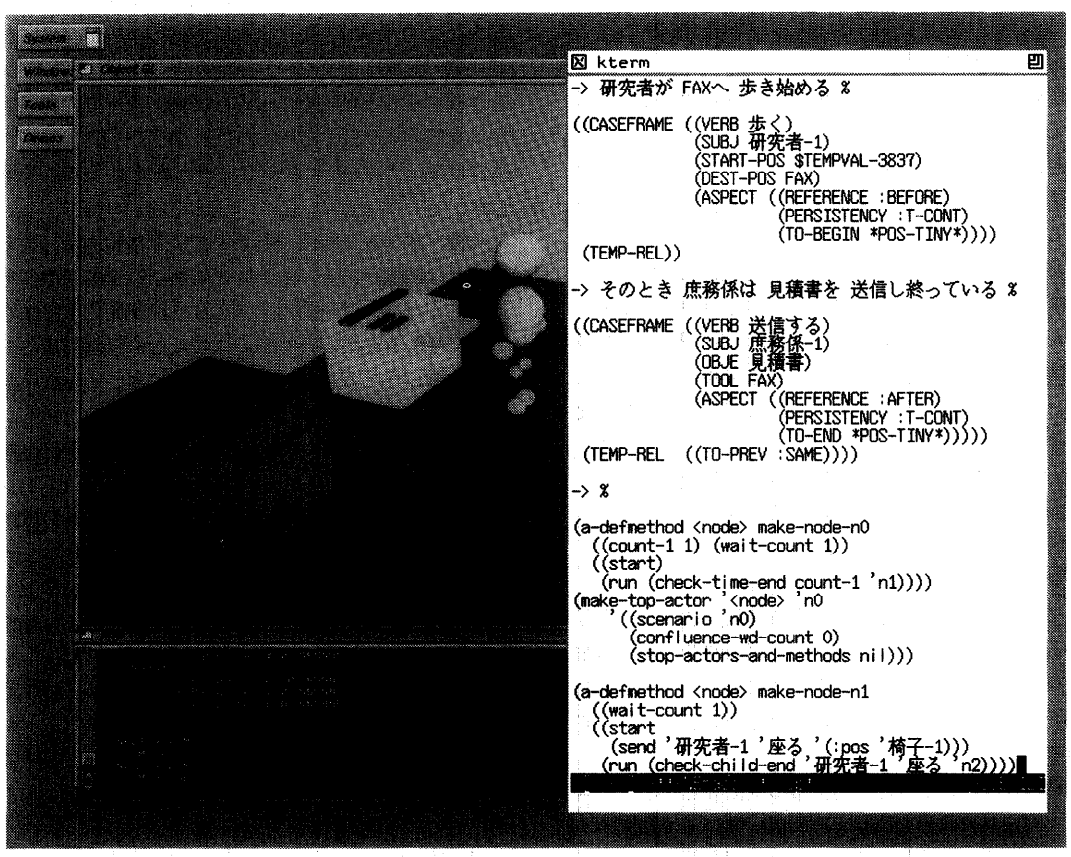


図 8 実験システムの動作例
Fig. 8 An example of the output of the experimental system.

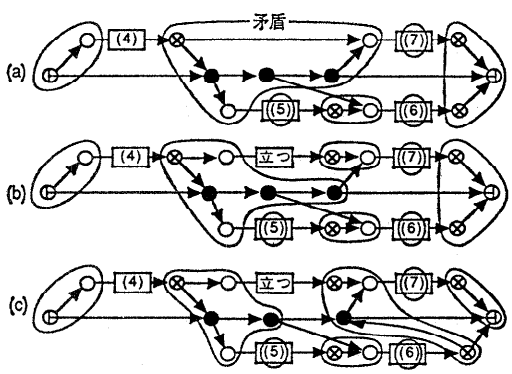


図 9 (動作4)～(動作7)から生成される動作順序グラフ
Fig. 9 A Motion-Order Graph generated from Motions (4)～(7).

日本語シナリオ解析部は，構文解析アルゴリズムに拡張 LINGOL[14]を用いており，構文解析と並行して，登場物や背景に関するフレーム型の知識ベースを用いた意味解析を行うことで，深層格フレームを抽出する．

5.2 実験結果

実験として，名詞・動詞の語彙と背景情報を知っている利用者が，4～18個の文からなる入力文章を10数例与えることで，実際にアニメーションの作成および修正を行った(**図8**)．この結果，期待する順序関係をもったアニメーションが生成されることを確認した．

例えば，実験システムに以下の文章を入力すると，**図9**(a)から補正処理を経て同図(b)の動作順序グラフが得られる．このグラフは，(動作4)と(動作7)の間に「研究者が立つ」という動作が補完されることや，(動作6)や(動作7)が並行に実行されることなどを表している．

(動作4) 研究者が自分の椅子に座っている．
(動作5) そのとき庶務係は FAX の前へ歩いて，
(動作6) 見積書を FAX に入れる．
(動作7) 研究者が FAX へ歩き始める．

この文章に次の文を加えた文章を入力すると，得られるグラフは図9(c)に変化し，(動作6)が(動作7)の開始以前に終了するようになる．

(動作 6′) そのとき庶務係は見積書を送信し終わっている．

利用者はこのような入力文章の変更にともなうアニメーションの変化を視覚的に確認しながら，動作間の同期の関係を詳細化していくことができる．

なお，図9(b)の動作順序グラフを導出する補正処理には約6秒，もっとも長い文章(18文)に対する補正処理には約47秒かかる．単文ごとの解析やプログラム生成に要する処理時間はいずれも1秒程度であるため，ほとんどの時間は補正処理に費やされている．

6. おわりに

本論文では，アニメーションの作成・修正のプロセスを簡便にするためのアニメーション作成支援システムについて報告した．

本システムでは，アニメーションのストーリ(登場物の動きの記述)を日本語のシナリオによって与える．このために，日本語のシナリオから動作事象間の半順序関係を認識するための一手法を提案した．事象間の半順序関係を認識することで，アニメーションにおける複数の登場物の並行動作を自然に表現できる．

本システムでは，日本語による記述を可能にしたことで，システムの操作知識を習得する必要性が軽減される．また，シナリオの記述は直接的で明瞭なものになる．これらの特徴によって，試行錯誤が容易になり，専門知識をもたない利用者が自らアニメーションを作成することが可能になる．実験によって，利用者が日本語のシナリオ上で動作間の関係を作成・修正するだけで，アニメーションを作成できることを示した．

今後，本システムを発展させていくためには，以下のような課題がある．本システムが正しく順序関係を認識するためには，動作にともなう状態変化の知識を正しく与える必要があるが，このことは語彙が増えるに従い難しくなる．このため，動作に関する知識の記述方法を階層化するなどの検討が必要である．また，試行錯誤をさらに容易にするためには，順序関係の補正処理に要する時間を短縮することが好ましい．本システムのアルゴリズムは一括的な手順であり，処理時間は文の長さとともに増加する．これを避けるために，Dean[15]のような増進的な順序関係の追加を許すアルゴリズムを適用することが考えられる．この種のアルゴリズムが提供する順序関係に関する非単調推論の基本機構を利用するためには，本システムに適した仮説生成などの推論戦略を検討する必要がある．さらに，動作の同一性の判定などの順序関係以外の曖昧さの処理を，このアルゴリズムとどのように共存させるかについても検討する必要がある．

謝辞 有益なコメントをいただきました査読者の方々に感謝いたします．また，日頃御指導頂くシャー

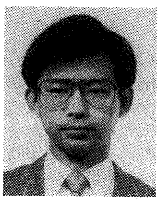
プ(株)技術本部河田亨副本部長，情報技術研究所大崎幹雄所長，千葉徹第1研究部長に感謝いたします．

本研究開発は，通商産業省工業技術院大型プロジェクト「電子計算機相互運用データベースシステムの研究開発」の一環として，新エネルギー・産業技術総合開発機構 (NEDO) より委託を受けて実施したものである．

参考文献

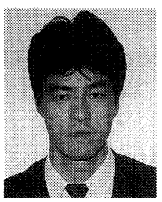
1) *VideoWorks II Manual*. Macro Mind Inc. (1987).
2) 沓沢：コンピュータアニメーション技法，情報処理，Vol. 29, No. 10, pp. 1090-1096 (1988).
3) Miyamoto, M., Hanada, K., Yoshikawa, K. and Sato, R.: EASY: A Model for Representing Computer Animation, *Proc. of ACM International Conference on Multimedia Information Systems '91*, pp. 321-332 (1991).
4) Winograd, T.: Understanding Natural Language, Academic Press (1971).
5) Kowalski, R.: A Logic-based Calculus of Events, *New Generation Computing*, Vol. 4, pp. 67-95 (1986).
6) Allen, J. F. and Koomen, J. A.: Planning Using a Temporal World Model, *Artif. Intell.*, Vol. 23, pp. 123-154 (1984).
7) 発電所用高度データベースシステムの開発：平成2年度委託研究成果報告書，(財)情報処理相互運用技術協会 (1991).
8) 山田，網谷，星野，西田，堂下：自然言語における空間描写の解析と情景の再構成，情報処理，Vol. 31, No. 5, pp. 660-672 (1990).
9) 町田：日本語の時制とアスペクト，アルク (1989).
10) 国立国語研究所：現代日本語動詞のアスペクトとテンス，秀英出版 (1985).
11) Fikes, R. and Nilsson, J.: STRIPS: A New Approach to Application of Theorem, *Artif. Intell.*, Vol. 2, pp. 189-205 (1971).
12) Nilsson, J.: *Principles of Artificial Intelligence*, Tioga Publishing (1980).
13) Sacerdoti, E. D.: Non-Linear Nature of Plans, *Proc. of the 4th IJCAI*, pp. 206-214 (1975).
14) 田中：拡張 LINGOL—自然言語処理のためのプログラミング・システム，電子技術総合研究所 (1978).
15) Dean, T.: Temporal imagery: An approach to reasoning about time for planning and problem solving, *Tech. Report*, Yale Univ. (1985).

(平成4年11月4日受付)
(平成5年4月8日採録)



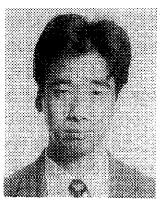
舟渡 信彦 (正会員)

1959年生．1983年大阪大学工学部電子工学科卒業．1985年同大学院工学研究科（電子工学専攻）前期課程修了．同年シャープ(株)入社．通信ソフトウェア，分散システム，プログラミング言語処理系の研究開発に従事．現在，同用 A 1175 プロジェクト所属．IEEE 会員．



吉川 耕平 (正会員)

1962年生．1985年京都大学工学部数理工学科卒業．同年シャープ(株)入社．人工知能，自然言語処理の研究開発に従事．1993年より奈良先端技術大学院大学情報科学研究科留学中．IEEE/CS 会員．



花田恵太郎 (正会員)

1962年生．1986年大阪府立大学工学部電気工学科卒業．1988年同大学院電気工学専攻修士課程修了．同年シャープ(株)入社．ソフトウェア工学，ユーザインタフェースの研究開発に従事．1993年より東北大学情報科学研究科後期課程留学中．日本ソフトウェア科学会，人工知能学会各会員．

宮本 雅之 (正会員)

1957年生．1979年神戸大学理学部数学科卒業．1984年大阪大学大学院理学研究科（数学専攻）後期課程単位取得退学．同年シャープ(株)入社．数値解析，ソフトウェア工学，人工知能，等の研究開発に従事．現在，同社情報技術研究所第2研究部所属．ACM, IEEE/CS各会員．