2I4-7in

# 物語における外的行動の背景知識と一貫性を管理するための状態－事象変換機構

# A State/Event Transformation Mechanism to Manage the Background Knowledge and Coherence of External Actions in a Story

栗澤 康成[*1]
Yasunari Kurisawa

福田 至[*2]
Itaru Fukuda

小方 孝[*2]
Takashi Ogata

[*1] 岩手県立大学大学院
Graduate School of Iwate Prefectural University

[*2] 岩手県立大学
Iwate Prefectural University

In this paper, we present a mechanism which deals with the relationship between elements in a story and stats as the backgrounds information of the elements in an integrated narrative generation system. An element in a story is equal to a case structure including a verb concept and several noun concepts, and attribute information in the noun concepts is constituted by the corresponding instance data. We have developed a state/event transformation mechanism to make states from a sequence of events. This paper provide a mechanism for managing the consistency of the progression of external events using an event complement mechanism according to the state/event transformation mechanism.

## 1. はじめに

本稿では，筆者らの統合物語生成システム[Akimoto 2012]において，物語内容を構成する事象とその背景の状態との関係を扱う一機構について述べる．今回は特に背景知識管理及び一貫性管理の側面に焦点を当てる．ここで事象（または事象概念）とは，動詞概念とその格によって表される出来事を意味し，状態とは物語内容の各時点における登場人物・物・場所の具体的要素（インスタンス）の属性情報を記述する知識構造に相当する．さらに状態は，特定のインスタンスの描写や説明を付与するためにも使用される．つまり描写や説明の対象となる知識が状態の知識構造の中に格納されている．事象と状態には，主に二つの関係がある――①事象は状態に何らかの変化を引き起こす，②ある時点で可能な事象はその時の状態によって制限される．状態と事象の関係は，これらの情報すなわち事象によって生じる状態変化及び事象の前提となる状態を定義した「状態－事象変換知識ベース」と，これを利用して事象から状態を生成する「状態管理機構」によって管理される．

物語内容の記述例とそこで利用される幾つかの知識ベースを図 1 に示す．物語内容は，物語内容木と状態列によって構成されている．物語内容木は，複数の事象が関係によって結合された木構造によって表される．終端要素が事象で中間要素が関係に相当する．この木構造は，物語知識ベース中に格納された，関係単位に組織化された物語の断片的情報を利用して拡張される．例えば，木構造中のある抽象的な事象を具体的な事象系列として展開したい場合（スクリプト化と呼ぶ），その事象に対応する具体的なスクリプト情報を物語知識ベースから検索することで木構造を拡張し，それらの事象列を束ねる関係に「継起」というラベルを付ける．このような処理を実際に行う関数を物語技法と呼ぶ（物語内容中の処理であることを明示する時は物語内容技法）．関係は個々の事象のみならず，ひとつ以上の関係を含む部分構造の結合節点ともなる．さらに，個々の事象の前後に状態が結び付く．ある事象の後の状態は，同時にその次の事象の前の状態となり，この状態情報の集合が状態列と呼ばれる．事象列を関係で結んだ物語内容木と各時点の状態を表した状態列は個別のデータで管理されるが，時間（time）を互いに参照することによって結びついている．

事象は，動詞概念とその深層格からなるフレーム構造により表現する．深層格には，time（事象の前後各時点の状態の番号），agent（主体），counter-agent（対象人物），object（対象物），instrument（道具），location（場所），from（始発地点），to（終着地点）の 8 種類があり，time 以外の格には人・物・場所を表すインスタンスの ID（固有記号）が記述される．事象を構成する動詞概念や名詞概念の情報は概念辞書[Oishi 2012]より参照される．インスタンスは，名詞概念辞書中の終端概念から生成される．一方，状態には，各インスタンスの属性情報が，スロット名と値の対の集合として記述される．

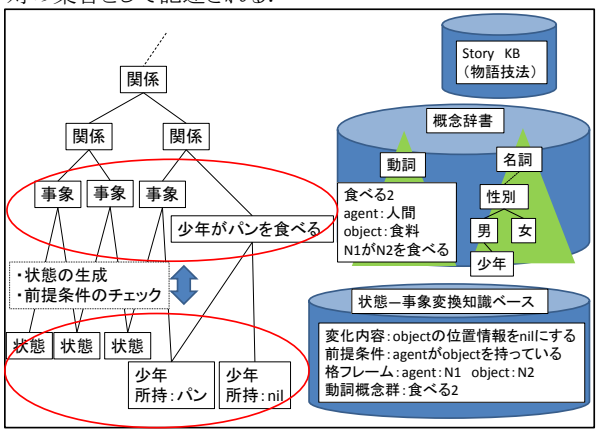


図 1 物語内容の記述例と関連する各種知識ベース

このインスタンスを構成するスロットの種類については，[Onodera 2012]を参考に人，物，場所の 3 タイプに分け，それぞれの構成スロット及びそのデフォルト値を定めた雛形を用意する．現状では，人タイプは 25 種類，物タイプは 17 種類，場所タイプは 14 種類のスロットを持つ．インスタンスの生成は，物語内容の生成過程中で随時行われる．その方法は，名詞終端概念名とインスタンスタイプ（人，物，場所の何れか）を入力とし，タイプに対応するスロットの雛形を獲得し，その中の「ID」（固有記号）と「instance-of」（対応する名詞概念名）のスロット値を，入力の概念名に基づき設定する．なお，構想では，名詞概念辞書に含まれる各概念に対して，スロットのデフォルト値を定義したフレームを付与し，インスタンスのスロットにそれを付与することを予

連絡先：栗澤康成，岩手県立大学大学院ソフトウェア情報学研究科，〒020-0193 岩手県岩手郡滝沢村滝沢字巣子 152-52，g231k013@s.iwate-pu.ac.jp

定しているが，現状ではこの仕組みは実現されていないため，多くのスロットのデフォルト値が空（nil）となる．

## 2. 統合物語生成システムにおける状態の役割

統合物語生成システムにおける状態が持つ役割を，以下の4つの項目に分けて説明する．

### (1) ある状態における「可能な事象」の制約

概念辞書における動詞概念の格の制約条件が，「物理的に可能」な範囲を定義するのに対して，状態に基づく制約は，格の値を，それが発生する時点の状態において可能な要素に制約する．例えば「少年が目覚める」という事象が発生するには，前提として少年が眠っている必要がある．このような条件を設定するには，状態情報の管理と事象が起きる前提条件を設定する必要がある．

### (2) 事象の流れの補完

事象列の中で，(1)の制約に反する事象が存在した場合，それを満たすように事象列を補う・追加を行う．これは事象列の一部を変化させた時に，それが他の部分における制約違反を招く場合が考えられる．例えば，物語の最初に「主人公が死ぬ」という事象を追加すると，その後に主人公の行動があった場合に制約違反となる．そのため，次に主人公が行動を行うまでに「主人公が生き返る」などの事象を生成することで，主人公が生きている状態を用意し，制約の条件を満たす事象の流れを生成する．

### (3) 並列的な事象進行の管理

同一の世界で複数の事象列が並列的に進行するような場合に，並列する事象間に不整合が生じないように管理することにも利用ができる．例えば，ある時点において，同一の人物が複数の場合に同時に存在するというようなことが生じないようにする．

### (4) 物語言説における利用

各インスタンスの状態を管理することで，その情報を描写や説明への利用することが可能となる．例えば，林檎を描写する際，林檎の状態情報に「色：赤」と記述されている場合には「林檎は赤い」などの表現が可能となる．

## 3. 状態管理機構の概要

### 3.1 状態－事象変換知識ベース

状態－事象変換知識ベースは，動詞概念辞書に含まれる各（終端）動詞概念に対して，その動詞概念による事象が引き起こす状態の変化を表す「変化内容」と，その事象が前提として必要とする状態の条件を表す「前提条件」の二つの要素を定義したルール（変換ルール）を格納する．なお，ある動詞概念は複数の格構造（深層格の組み合わせ）を持つ場合があるため，その格構造毎に変換ルールが定義される．図 2 は変換ルールの例である．現在の知識ベースには 3178 件のルールが格納されている（状態変化なしのルールも含む）．変換ルールの定義項目については[秋元 2013]に記述している．

```
((変化内容
((actor agent)(slot location)(op nil)(order (alt to))))
(前提条件
(条件 1(条件名 存在)(op nil)(actor agent)(slot location)
(val (n_v_eq to))))
(格フレーム ((agent N1) (to N2)))
(動詞概念群 (至る 1)))
```

図 2「至る 1」の変換ルール

### 3.2 状態管理機構

図 3 は物語内容生成の概念図である．構造生成機構は物語知識ベースや概念辞書を参照して，物語内容木とその中の事象を構成するインスタンスの生成を行う．その後，状態管理機構へこの物語内容木を入力する．状態管理機構は，一つの物語技法によって単一の事象もしくは複数の事象が新たに生成される度に，新たな状態列（背景知識）を生成する．処理手順は次のようになる．

最初に，各インスタンスのデフォルト値に基づき先頭の状態（先頭の事象の前の状態）が生成される．その後，以下の処理が先頭の事象から順に最後の事象まで繰り返される．

1) 対象の事象の動詞概念及び格構造をキーとして，状態－事象変換知識ベースからひとつの変換ルールを獲得する．

2) 前提条件に記述された各要素を順にチェックし，その事象の前状態が前提条件を満たすかどうかを調べる．前提条件を満たさない場合，その条件に該当するスロットの値が前提条件を満たす値に書き換えた状態を作成して状態列に追加する．なお，処理 2)の詳細な動作については 4 節で詳しく述べる．

3) その事象の前状態に対して，ルールの変化内容の各要素に基づく値を変化させる処理を順に実行し，事象の後状態を生成する．処理 2)で状態が追加された場合はその中の最後の状態に対し，同様の処理を行う．

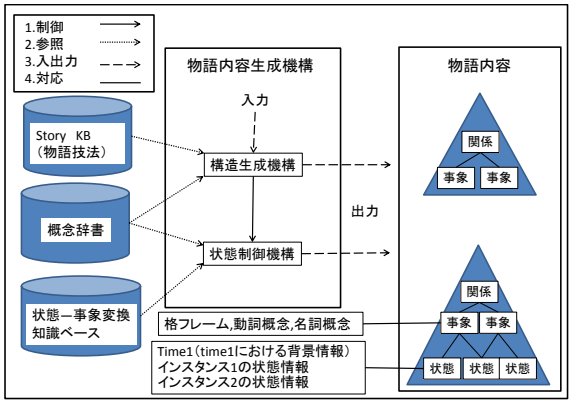


図 3 物語内容生成―概念図

### 3.3 変換ルールの定義における問題点

状態－事象変換知識ベースにおける変換ルールの定義作業を進める中で種々の問題が挙がった．現在は，内的・心理的な事象を除く登場人物の外的行動を伴う事象のみを扱っているが，大まかに分類すると三つの問題が挙がった．一つ目は，「太郎が父を訪ねる」のように，実際には太郎が父の居場所へ移動するのだが，事象にはその情報が直接的には記述されていない為に，このまま記述を行うと人インスタンスである「父」が太郎の位置情報となってしまう．この問題は事象概念や知識ベースの記述に拠るものであり，「agent の居場所」という情報の指定・記述方法を知識ベースに追加することで解決可能である．二つ目として，「太郎が父を殴る」のように，変化後の状態（怪我，死亡等）を一義的に決定できない事象が存在する．この問題は，変化後の状態候補を幾つか用意し，そこから何らかの方法で選択するという解決方法が考えられる．実際は，殆どの事象の結果は一義的に定義出来ないと言った方が良く，この複数の候補を何らかの方法で用意して利用する方式への拡張が必要になると思われる．三つ目は，「太郎が次郎を演じる」のように，事象による状態の変化が，現在扱っているような登場人物の外的な

状況の変化ではどう処理するのか不明な場合である．これは事象の性格付けをより詳細に考察し，状態変化の定義を精密にしなければ解決出来ない．

## 4. 物語内容の一貫性の管理

物語内容における物語の一貫性とは，物語の事象列において状態の断絶（前提条件を満たさない事象）を持たないことを指す．物語内容生成において，概念辞書や状態－事象変換知識ベースを始めとする様々な知識を用いた基本的な知識処理により，事象進行における一貫性の管理が行われる．本節では，状態管理機構及び状態－事象変換知識ベースを利用した一貫性の管理・補完の仕組みを説明する．

### 4.1 管理方法の概要

図 4 に状態を利用した補完処理の概要を示す．まず，物語内容機構において生成された物語内容木から状態列を生成する．この時，物語内容木終端の事象列において，状態－事象変換知識ベースに定義された前提条件を満たさない事象があった場合，3.2 節で述べたように，その条件を満たす状態（列）が補完される（図中の time2-1）．この時点では，補完によって追加された状態（列）部分の状態変化に対応する事象（time2 と time2-1 の間を結ぶ事象）が存在しない．そこでシステムは，この状態変化に対応する事象を新たに生成し，物語内容木に追加することによって，事象レベルでの補完を行う．

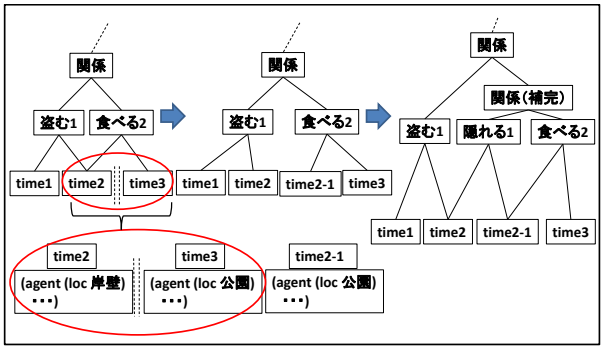


図 4 状態及び事象補完の流れ

### 4.2 実行例

以上の流れを，システムの処理手順に沿って，実際のデータを示しながら解説する．最初の入力となる物語内容木を図 5 に示す．この物語内容木は「丈夫が岸壁で少年から林檎を盗む」「丈夫が公園で林檎を食べる」という意味のふたつの事象概念からなる．

```
($継起
 (event 盗む 1 (type action) (ID 1) (time (time1 time2))
  (agent age%丈夫#1) (counter-agent nil)
  (location loc%岸壁#1) (object obj%林檎#1)
  (instrument nil) (from age%少年#1) (to nil))
 (event 食べる 2 (type action) (ID 2) (time (time2-1 time3))
  (agent age%丈夫#1) (counter-agent nil)
  (location loc%公園#1) (object obj%林檎#1)
  (instrument nil) (from nil) (to nil)))
```

図 5 入力した物語内容木

#### (1) 状態生成フェーズ

図 5 を入力として，3.2 節で述べた方法により状態列を生成した結果を図 6 に示す．なお，ここでの動作説明に関連する情報以外の大部分を省略してある（主に状態変化の生じないスロットを省略し，time2 以降は「丈夫」（インスタンス）のみを示す）．

ひとつ目の事象の後（time2）の状態において，「丈夫」は「岸壁」に存在する（“(location loc%岸壁#1)”）．ふたつ目の事象に対応する「食べる 2」の変換ルールの前提条件のひとつとして，「agent 格のインスタンスが事象の location 格のインスタンスに存在すること」がある（図 7）．ふたつ目の事象の agent 格は「丈夫」であり，location 格の値は「公園」となっているため，前提条件に違反する．そのため，「time2-1」の状態が補完された（「丈夫」の居場所が「公園」に変化）．これに伴い，ふたつ目の事象は，「time2-1」から「time3」の間に生じたものと見なされる．

```
(time1
  (time (ID time1) (名前 nil))
  (agents
    ((ID age%丈夫#1) (instance-of 丈夫@男)
     (location loc%岸壁#1) (所持 nil))
    ((ID age%少年#1) (instance-of 少年@男)
     (location loc%岸壁#1) (所持 obj%林檎#1)))
  (objects
    ((ID obj%林檎#1) (instance-of 林檎@果樹)
     (location loc%岸壁#1)))
  (locations
    ((ID loc%岸壁#1) (instance-of 岸壁@港))
    ((ID loc%公園#1) (instance-of 公園@庭園))))
(time2
  (agents ((ID age%丈夫#1) (instance-of 丈夫@男)
  (location loc%岸壁#1) (所持 obj%林檎#1))・・・)
(time2-1
  (agents ((ID age%丈夫#1) (instance-of 丈夫@男)
  (location loc%公園#1) (所持 obj%林檎#1))・・・)
(time3
  (agents ((ID age%丈夫#1) (instance-of 丈夫@男)
  (location loc%公園#1) (所持 nil))・・・)
```

図 6 出力された状態列

```
((変化内容
  ((actor agent)(slot 所持)(op nil)(order (del object)))
  ((actor object)(slot location)(op nil)(order (alt nil))))
(前提条件
  (条件 1(条件名 存在)(op nil)(actor agent)(slot 所持)
  (val (contain object))))
(条件 2(条件名 存在)(op nil)(actor agent)(slot location)
  (val (eq location))))
(格フレーム ((agent N1) (object N2)))
  (動詞概念群(食べる 2)))
```

図 7「食べる 2」の変換ルール

#### (2) 事象補完フェーズ

次に，状態生成の結果（図 6）及び元の物語内容木（図 5）を入力として，事象補完処理を行う．この機構は，状態列と物語内容木に含まれる事象列を照らし合わせて，状態変化に対応する事象が存在しない部分（この例では「time2」と「time2-1」の間）を

発見した場合に，次の方法で新たな事象概念を生成して，物語内容木に追加する．

まず，該当部分の前状態と後状態を比較し，スロットの値が変化しているインスタンス ID 及びそのスロットの変化前と変化後の値をそれぞれ抽出する（状態変化情報と呼ぶ）．状態－事象変換知識ベースから，状態変化情報と同じ変化内容が定義された変換ルールをすべて獲得する．今回の例では，「time2」と「time2-1」の間で，「丈夫」の「location」スロットの値が，「岸壁」から「公園」に変化しており，この情報をもとに，「agent 格の location スロットを別の値に書き換える」という変化内容を持つ変換ルールが獲得される．図 8 に獲得された変換ルールの一例を示す．その後，その中のひとつの変換ルールをランダムで選択し，さらにそこに含まれる動詞概念の中のひとつを同じくランダムで選択する．最後に，その動詞概念の格に，状態変化情報に基づき適切な値を設定した事象概念を生成し，物語内容木に追加する．以上の方法により補完が行われた結果を図 9 に示す．この例では「丈夫が公園に隠れる」という意味の事象が追加されている．

```
((変化内容
  ((actor agent)(slot location)(op nil)(order (alt to))))
 (前提条件
  (条件 1(条件名 存在)(op nil)(actor agent)(slot location)
         (val (n_v_eq to))))
 (格フレーム ((agent N1) (to N2)))
 (動詞概念群 (隠れる 1)))
```

図 8 獲得された変換ルールの一例

```
($継起
  (event 盗む 1 (type action) (ID 1) (time (time1 time2))
   (agent age%丈夫#1) (counter-agent nil)
   (location loc%岸壁#1) (object obj%林檎#1)
   (instrument nil) (from age%少年#1) (to nil))
  ($－
   (event 隠れる 1 (ID nil) (time (time2 time2-1))
   (agent age%丈夫#1) (counter-agent nil)
   (location loc%岸壁#1) (object nil) (instrument nil)
   (from nil) (to loc%公園#1)
  (event 食べる 2 (type action) (ID 2)
  (time (time2-1 time3)) (agent age%丈夫#1)
  (counter-agent nil) (location loc%公園#1)
  (object obj%林檎#1) (instrument nil) (from nil) (to nil))))
```

図 9 出力された物語内容木

### 4.3 補完機能における問題点と解決案

補完機構は状態の断絶を発見した際にその直前へ状態と事象を追加するため，補完によって追加された事象が不自然な流れとして捉えられる場合が考えられる．例えば「勇者が怪物を倒しに行く」場合であるが，この場合に「勇者が剣で怪物を切る」という事象の直前に「勇者が剣を拾う」という事象の補完を行うと不自然な話の流れとなる（怪物を倒しに行くのに武器を用意せずに拾うのは不自然）．この問題は状態の断絶の直前だけでなく更に前の位置への補完を可能とする機能拡張を行い，他の知識ベースを用いた推論を利用して補完する位置の選択を行うなどの解決案が考えられる．

## 5. むすび

以上，統合物語生成システムにおける状態管理機構と状態－事象変換知識ベースを紹介し，特に背景知識と一貫性の管理に焦点を当てた議論を行った．知識ベースは，動詞概念によって生じる変化を形式的に記述した変換ルールの集合となる．現状で，物理的な行為を中心に 3178 個のルールを定義した．

物語中の各時点におけるインスタンス群の状態を表す背景知識，物語の事象列における一貫性（状態の断絶のチェック）は状態管理機構により管理・補完が行われる．状態管理機構は状態－事象変換知識ベースに記述された変換ルールを参照することで，事象による状態の変化を各時点で記述し，状態の断絶を確認した際には補完機能により事象と状態の補完を行う．将来的には，状態を利用した事象概念の生成も考えている．例えば，現在は事象列を作成した後に状態を生成しているが，逆に目標となる状態を先に設定する方法も考えられる．これによって目標の状態を目指す，問題解決的な物語内容の生成が可能となる．

今後は定義を進める中で浮上した問題について対応を行いながら，知識ベースの性能に対する評価も段階的に行い，知識ベース自体の完成度を高めていく予定である．状態－事象変換知識ベースの評価については以下のような段階で評価を行う予定である．

1)動作確認
2)変換ルールの妥当性
3)実際にルールを用いた際の物語の一貫性

1)は最も基本的な動作確認である．2)は変換ルールの定義が適切であるかどうかに焦点を当てた評価である．3)は実際に変換ルール用いて作成した物語と利用しなかった場合の物語を比較することで評価を行う．これらの評価結果を今後の知識ベース構築の作成方針や仕様拡張に反映していく．

### 参考文献

[Akimoto 2012] Akimoto, T. & Ogata, T.: Macrostructure and Basic Methods in the Integrated Narrative Generation System by Introducing Narratological Knowledge, Proc. of 11th IEEE International Conference on Cognitive Informatics & Cognitive Computing, 253-262, 2012.

[Oishi 2012] Oishi, K., Kurisawa, Y., Kamada, M., Fukuda, I., Akimoto, T. & Ogata,T.: Building Conceptual Dictionary for Providing Common Knowledge in the Integrated Narrative Generation System, Proc. of the 34th Annual Conference of the Cognitive Science Society, 2126-2131, 2012.

[Onodera 2012] Onodera, K., Akimoto, T. & Ogata, T.: A State-Event Transformation Mechanism for Generating Micro Structures of Story in an Integrated Narrative Generation System, Proc. of the 34th Annual Conference of the Cognitive Science Society, 2150-2155, 2012.

[秋元 2013] 秋元泰介，栗澤康成，福田至，小方孝: 物語内容における状態を管理する機構の構築，言語処理学会第 19 年次大会, 2013.