論　文

Technical Papers

文章生成における推敲機能の実現について

Implementation of Revision in Text Generation

乾　健太郎*　徳永　健伸*　田中　穂積*
Kentaro Inui　Takenobu Tokunaga　Hozumi Tanaka

*　東京工業大学工学部情報工学科
Dept. of Computer Science, Faculty of Eng., Tokyo Institute of Technology, Tokyo 152, Japan.

1992年9月4日　受理

Keywords: text generation, how to say, text revision, dependency directed backtracking, functional unification grammar.

Summary

To generate good text, many kinds of decisions should be made. Many researchers have been engaged in searching for both an architecture that would determine a proper order for these decisions and heuristics that would make an appropriate decision locally at each decision point. However, even if such an architecture or heuristics were found, there are still certain kinds of problems that are difficult to consider during the generation process. Those problems include, for example, structural ambiguity and sentence complexity. Some of them can be easily detected and solved by introducing a revision process after generation.

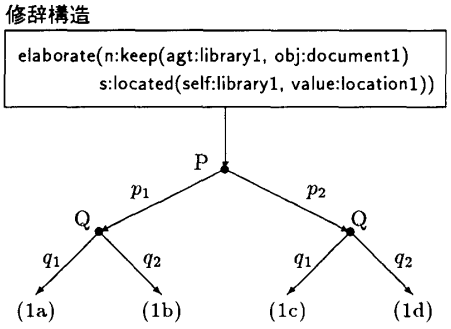
In this paper, we argue the importance of text revision with respect to natural language generation, and adopt a computational model of text revision. We also discuss its implementation issues and introduce dependency directed backtracking in order to realize efficient revisions. Our model can be implemented easily because the revision process allows the system to make a decision at each point without any anticipation of the future decisions. Finally, we evaluate our method on an experimental Japanese text generation system.

1. はじめに

文章生成システムは，一般に書き手の情報伝達ゴールを入力として受け取り，文章を出力する．文章生成に必要な処理は，どのような内容をどのような構成で述べるかに関する選択(what-to-sayの選択)とwhat-to-sayをどのように言語表現するかに関する選択(how-to-sayの選択)に大きく分けることができる[徳永 91]．多くのシステムでは，まずwhat-to-sayを選択し，その結果を修辞構造[Mann 87]と呼ばれる構造化された命題の集合で表現する．次に，how-to-sayを選択し，文字列を出力する．本論文では，特にhow-to-sayの選択に関する従来の手法の問題点を取り上げ，この問題を解決するために，推敲機能を備えた生成モデルを提案する．

how-to-sayの選択は以下のような探索問題として捉えることができる．how-to-sayの選択には，複数の命題を一文に統合するなどの文章レベルの統語的選択，文型，語順などの文内の統語的選択，および語選択が含まれる．これらの選択は図1のような探索空間の各アークに対応する．how-to-sayの生成過程では，システムは修辞構造を入力として受け取り，探索空間を根ノードから順にたどる．中間ノードはそれまでの選択を蓄積した状態を表し，これは次の選択に対する制約となる．各ノードから出るアークは，その状態で必要な統語的・語彙的制約を満たす選択肢を表す．システムは，最終的に葉ノードに到達し，文章を一つ出力する．図1には葉ノードが複数あるが，これは同じ入力に対応する言語表現が複数存在することを意味する．システムの目標は，統語的・語彙的制約を満たす複数のパスのなかから，質の高い文章を生成するパスを選択することである．

多くのシステムでは，個々の選択点で選択肢の局所



p_1: located を keep の節に埋め込む
p_2: located を keep の節に埋め込まない
q_1: location1 を building1 の「4 階」として表現
q_2: location1 を単に「4 階」として表現

図 1 文章生成の探索空間

的な優先度を求め，それに従って選択を行う．各選択点での局所的な優先度は一般にヒューリスティクスを用いて与える．本論文ではこのような規則を選好規則と呼ぶ．選好規則は文章の質を決めるさまざまな評価基準に基づいて作られる．文章生成の研究目的の一つは，どのような選好規則が適当かを究明することである．例えば，照応表現や態の選択などに関して，すでにさまざまな規則が提案されている[桃内 89]．しかしながら，文章の質を局所的な優先度だけで実現する方法には限界がある．

例として，図 1 に示した修辞構造からの生成を考えてみよう*1．生成に必要な選択の例として図に示した選択点 P, Q を取り上げる．それらの選択に依存して，以下のような複数の文章が生成できる．

(1 a) あなたが探している資料は正門に最も近い建物の 4 階にある資料室に保管されています．

(1 b) あなたが探している資料は 4 階にある資料室に保管されています．

(1 c) あなたが探している資料は資料室に保管されています．資料室は正門に最も近い建物の 4 階にあります．

(1 d) あなたが探している資料は資料室に保管されています．資料室は 4 階にあります．

P は命題の埋込みに関する選択である．例えば，p_1 を選択すると二つの命題 keep と located が一つの文として表出される．Q は，修辞構造中の個体 location1 に対する参照表現に関する選択を表している．読み手が location1 を同定するために必要な情報は，文脈，読み手の知識，談話の状況などに依存する．したがって，(1 a)のように多くの修飾句を必要とする場合もあれば，(1 b)のように 1 語で十分な場合もある．

ここで，文の統語的な複雑さを文章の評価基準の一つとして考えると，(1 a)は「正門に… 4 階にある」という修飾句が長すぎるため望ましい出力とはいえない．談話の状況から選択点 Q で q_1 が優先されると仮定した場合，システムは p_2, q_1 を選択し，(1 c)を生成するのが最も望ましいことになる．すなわち，最初の選択点 P では p_2 を選択すべきだったことがわかる．しかしながら，このことを選択点 P で判断するのは容易ではない．P の選択の時点では，選択 Q をはじめ語選択などが未定であり，文の複雑さの評価に必要な情報が十分に得られないからである．

このように，表層に近いレベルの情報を必要とする評価基準を表層依存性評価基準と呼ぶことにする．このような評価基準の例として，文の統語的複雑さのほかに，係り受けの曖昧性などがあげられる．表層依存性評価基準に関する選択肢の優先度を生成の途中で計算するためには，将来の選択に関する慎重な予測が必要になり，複雑な規則を用意しなければならない．これは，システムの実装・保守の点で望ましくない．

この問題は，選択の順序をうまく制御すれば解消できるように見えるが，最適な選択の順序を静的に決めるのは一般に難しい．また，選択の順序を動的に決定する手法もいくつか提案されているが[Hovy 88]，処理が複雑になるため，その実現や保守の点で問題がある．

これに対し，推敲機能を導入することによってこの問題の解消を試みるアプローチもある．[Meteer 86]は，一度生成した文章に対し評価・修正を繰り返す機能を備えた生成モデルを提案し，工学的な立場から次のような利点をあげている．

- 構造的に類似した二つの文を一つに統合するなど，表層化したほうが考慮しやすい問題について，表層から途中の選択へのフィードバックが可能になる．
- 比較的単純な修正を繰り返すことによって，全体として複雑な処理を実現することができる．

また，Yazdani は，推敲過程の心理学的な妥当性という観点から同様な生成モデルを提案している[Yazdani 87]．これらのモデルでは，すべての選択を暫定的に決定した後で評価するため，表層依存性評価基準についても，評価に必要な情報を予測する必要がない．したがって，推敲機能を持たない従来のモデルに比べ表層依存性評価基準の考慮が容易となると考えられる．また，一度決めた選択を変更することができるため，選択の順序が必ずしも最適である必要はない．し

*1 修辞関係には Mann の RST[Mann 87]を用いる．**n** は主部（nucleus）を，**s** は従属部（satellite）を表す．

かしながら，いずれのモデルについても実装に関する議論はほとんどなく，これらのモデルを実装したシステムはいまだに報告されていない．

我々は表層依存性評価基準を扱うためにMeteerのモデルを採用しており，本論文では推敲機能の実装方法を中心に議論する．推敲機能は評価機能と修正機能に大別できるが，ここでは主として修正機能の実装について議論し，効果的かつ効率的な文章の修正を可能にする実装方法を提案する．以下，まず**2章**でモデルについて述べる．実装に関する議論を可能にするために，文章生成を探索問題とみなし，探索空間および探索手法を定義する．そして，その枠組みを用いて推敲機能の役割を明らかにする．**3章**では，推敲機能の実現に適した言語知識の記述方法について述べる．**4章**では，一度生成した文章の修正に依存指向性バックトラックを導入し，冒頭で例示した問題がどのように解決されるかを示す．さらに，**5章**で日本語の文章を生成する実験システムの定量的評価について述べる．

2. 生成モデル

推敲機能を備えた生成モデルを図2に示す．本論文ではhow-to-sayの選択に限って議論するので，入力としてwhat-to-sayすなわち修辞構造を仮定する．文章生成過程は初期生成過程とそれに続く推敲過程からなる．

初期生成過程では，表出部が修辞構造を受け取り，統語的選択と語選択を行い，文章を生成する．我々のモデルではこの文章を書き換える可能性があるので，これを草稿と呼ぶ．統語的選択と語選択は言語知識と選好規則を参照して行う．言語知識は統語的・語彙的制約を与える．一方，選好規則は参照表現や主題化などの語用論的・文脈的優先度を与える．**1章**で述べたように，ある選択点に統語的・語彙的制約を満たす選択肢が複数ある場合，表出部は選好規則を用いて優先度の高い選択肢を選ぶ．草稿は，すべての統語的選択と語選択を行った結果であり，統語構造として表現される．草稿は中間表現でなく，文字列であることに注意してほしい．

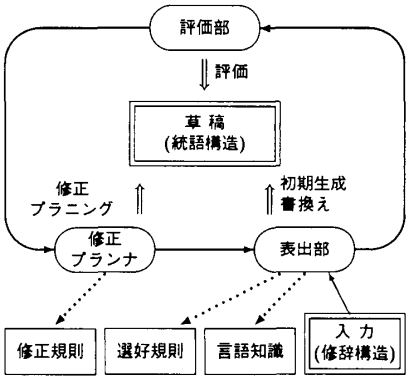


図2 文章生成モデル

推敲過程は修正サイクルの繰返しからなる．各修正サイクルでは，評価，修正プラニング，書換えを行う．まず，評価部が草稿を評価し，問題点を検出する．次に，修正プランナが検出された問題点のなかから一つを選択し，修正規則を参照して，その問題を解消するために有効な修正方法を決定する．最後に，表出部が修正プランナの決定に従って実際に草稿を書き換える．これで1回の修正サイクルが終わる．修正サイクルを繰り返すことによって，草稿の質が徐々に向上することが期待できる．

このモデルの特徴は，文章の質を判断するさまざまな評価基準を選好規則と修正規則という二つの資源を用いて記述する点にある．選好規則については，多くの研究者がさまざまなヒューリスティクスを提案し，その有効性を確認している．我々のモデルではそれらの成果を自然に利用できる．しかしながら，**1章**で述べたように，評価基準のなかには，生成過程の途中では評価に必要な情報が部分的にしか得られないもの(表層依存性評価基準)がある．そのような評価基準を生成過程の途中で評価するには将来の選択に関する慎重な予測が必要になる．このため，それらのすべてを選好規則のなかに押し込めると，選好規則が複雑になり，システムの実装や保守の観点から望ましくない．我々のモデルでは，選好規則を単純化するという立場から，将来の選択に関する予測を用いず，現在得られる情報だけで選択肢の優先度を計算するように選好規則の能力を限定する．一方，表層依存性評価基準に関する優先度については，選好規則だけでは十分に与えることができないので，選好規則が与えた優先度を修正規則の適用によって修正する．これにより，個々の規則を単純化できるだけでなく，選好規則だけでは十分に扱えない評価基準についても適切な優先度を与えることができる．このように推敲機能は「選好規則が与える不十分な優先度を修正規則によって補正する処理」として考えることができる．以下では，この考え方に基づいて推敲機能の実装について議論する．

現在の実験システムでは表層依存性評価基準として文の構造的複雑さと係り受けの曖昧性を扱っているが，どのような評価基準を評価部で扱うべきかは今後の課題である．例えば，Meteerは評価部の評価基準として一文を超えた大域的結束性を取り上げており，そのような議論も参考になるだろう．ただし，ある評価

基準について必要な情報がすべて得られたとしても，現在の技術では必ずしも的確に評価できるとは限らない．例えば，係り受けの曖昧性の評価については，高度な解析処理技術の開発を待たねばならず，現在のシステムでは人手で行っている．しかしながら，表層依存性評価基準のなかには文の構造的複雑さや大域的結束性など，表層の情報が完全に与えられれば比較的評価しやすいものもあり，それらに関する推敲機能については現在の技術でも自動化することができよう．

3. 言語知識と初期生成

3・1 機能単一化文法

1章で述べたように，文章を生成する過程は探索空間の1本の探索パスを選択することに対応する．一方，草稿の書換えは，一度選択したパス上のある選択点について選択を変更する処理とみなせる．もとの草稿と書換え後の草稿はどちらも同じ修辞構造から生成したものなので，草稿の意味は書き換えた後も保存される．

問題を解消するために変更すべき選択は，統語的な選択や語選択など，さまざまな言語知識に関係する．したがって，言語知識を統一的な表現形式で扱うことが望ましい．このような理由から，我々のシステムでは言語知識の表現形式として機能単一化文法（FUG）[Kay 84]を採用している．FUGでは，意味構造あるいは統語構造を表現する作業機能記述（WFD）と言語知識を表現する文法機能記述（GFD）とを単一化することによって生成または解析を行う．

入力となるWFDの例を図3に，GFDの一部を図4，図5，図8に示す．各素性は，対象とする領域中の個体（document#1など），統語的・語彙的素性値（sentenceなど），機能記述（FD）のいずれかを値にとる．素性path(…)はパス(…)で指されたFDと素性を共有することを表す．↑は親のFDに1段さかのぼることを示す．↑が一つの場合は自分自身を指し，ない

```
[ cat : segment
  pred : condition#1
  n : [ pred : elaborate#1
        n : [ pred : keep#1
              topic : document#1
              agt : [ pred : library#1 ]
              obj : [ pred : document#1 ] ]
        s : [ pred : located#1
              topic : library#1
              self : [ pred : library#1 ]
              value : [ pred : location#1 ] ] ]
  s : [ pred : elaborate#2
        n : [ pred : close_period#1
              topic : library#1
              self : [ pred : library#1 ]
              value : [ pred : summer_vacation#1 ] ]
        s : [ pred : period#1
              topic : summer_vacation#1
              self : [ pred : summer_vacation#1 ]
              from : [ pred : july21#1 ]
              to : [ pred : august31#1 ] ] ] ]
```

図3 WFDの初期状態

```
[ FD_id : 10/2
  cat : segment
  pred : @condition#i
  n_seg : [ path : (↑ n)
            pred : @rhetorical_relation#j
            cat : segment ]
  s_seg : [ path : (↑ s)
            pred : @rhetorical_relation#k
            cat : segment ]
  pattern : (n_seg s_seg) ]
```

図4 GFDの一部

```
[ FD_id : 10/3
  cat : segment
  pred : @elaborate#i
  { [ FD_id : 13/1
      n_seg : [ path : (↑ n)
                pred : @rhetorical_relation#j
                cat : segment ]
      s_seg : [ path : (↑ s)
                pred : @rhetorical_relation#k
                cat : segment ]
      pattern : (n_seg s_seg) ]
    [ FD_id : 13/2
      n : [ pred : @process#j
            @role : [ pred : @con_obj#l
                      restr : [ path : (↑↑↑ s)
                                embedded : + ] ] ]
      s : [ pred : @property#k
            self : [ pred : @con_obj#l ] ]
      sent : [ path : (↑ n)
               cat : sentence ]
      pattern : (sent) ]
    [ FD_id : 13/3
      n_sent : [ path : (↑ n)
                 pred : @normal_relation#j
                 cat : sentence ]
      s_sent : [ path : (↑ s)
                 pred : @normal_relation#k
                 cat : sentence ]
      pattern : (n_sent s_sent) ] } ]
```

図5 命題の埋込みに関するGFD

```
[ cat : segment
  pred : condition#1
  n : [ pred : elaborate#1
        n : [ pred : keep#1
              ... ]
        s : [ pred : located#1
              ... ] ]
  s : [ pred : elaborate#2
        n : [ pred : close_period#1
              ... ]
        s : [ pred : summer_vacation#1
              ... ] ]
  n_seg : [ path : (n)
            cat : segment ]
  s_seg : [ path : (s)
            cat : segment ]
  pattern : (n_seg s_seg) ]
```

図6 単一化後のWFD(1)

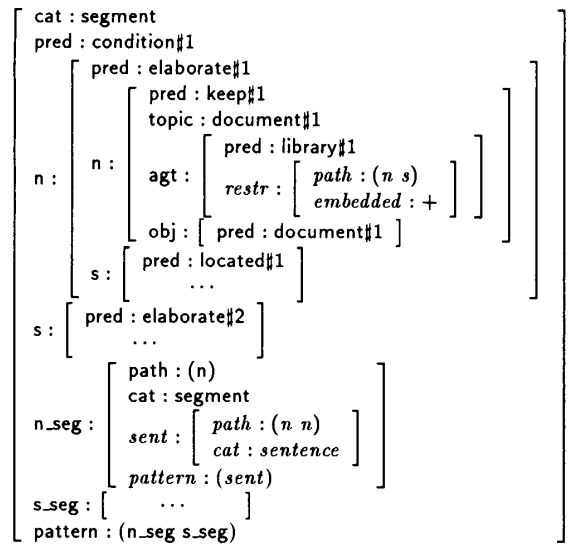


図 7　単一化後の WFD(2)

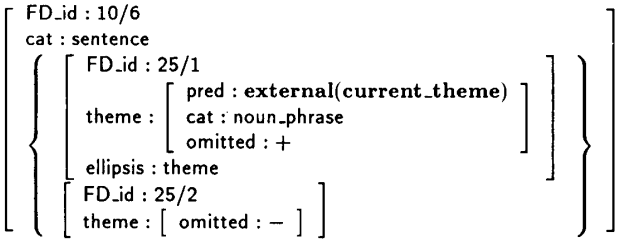


図 8　主題の省略に関する GFD

場合は絶対パスを示す．pattern 素性は各レベルの WFD の構成素とそれらの表層での順序を示す．@で始まる素性名・素性値は型つき変数を表す．また，{ } は FD の選言を表し，選択肢のなかの少なくとも一つが単一化に成功しなければならないことを意味する．

システムは WFD と GFD の単一化を統語的・語彙的素性がすべて決まるまで繰り返す．まず WFD のトップレベルと GFD との単一化から試みる．この場合，図 3 の WFD と図 4 の GFD の単一化が成功し，図 6 の WFD を得る．斜字体で示した素性が単一化によって付加された素性である．トップレベルの単一化が終了すると，次に素性 pattern 中に現れる構成素について左から順に単一化を進める．図 6 の例では，最初の構成素が n_seg であるから，まず WFD 中の (n_seg) と GFD(図 5)を単一化する((n_seg) 中の path：(n) によって，(n_seg) が図 7 の (n) と素性を共有することに注意)．図 5 のように GFD が選言を含む場合は，一番上の選択肢から順に単一化を試み，最初に成功するものを一つだけ選択する．ここでは FD(13/2) との単一化に成功し，図 7 の WFD を得る．このように，表出部は単一化をトップダウンに進め，表層の文字列の生成に必要な統語的・語彙的素性をすべて決定した時点で処理を終了する．この例では草稿(2 a)を生成する．

(2 a)　あなたが探している資料は正門に最も近い建物の 4 階にある資料室に保管されています．ただし，資料室は 7 月 21 日から 8 月 31 日までの夏休みの間は利用できません．

3・2　言語知識と生成過程

図 7 は，(n_seg) との単一化で FD (13/2) を選択したことにより，(n n agt) に restr が加わった状態である．これにより命題 located#1 が命題 keep#1 に埋め込まれることになる．修辞構造から表層の文章構造への写像規則に関しては，Scott らの研究[Scott 90]をはじめ，いくつかのヒューリスティクスが提案されている．それらの多くは図 5 のような GFD として記述することができ，文内レベルの統語的選択・語選択と同様，単一化を用いて処理できる．このように，本システムでは文を超えたレベルの選択と文内レベルの選択を統一的に扱う．

GFD が選言を含む場合，**3・1 節**で述べたように，表出部は上の選択肢から順に単一化を試みる．成功するものが複数存在する場合でも，最初に成功した選択肢だけを選んで先に進むため，選択肢の間には優先度が存在することになる．この優先度は **1 章**で述べた選好規則に相当する．

選言中の選択肢の位置は選択肢の間に静的な優先度を与えるが，照応表現の選択など，文脈に存在して動的に優先度が変化する場合もある．例えば，草稿(2 a)の第 2 文の主題 library#1 は「資料室は」という後置詞句として表層に現れているが，仮に直前の文ですでに「資料室は」という提題が起こっていたとすると，省略するほうが自然であろう(**4・1 節**の草稿(2b))[桃内 89]．この知識は，図 8 のような選言として記述できる．FD(25/1)中の external は，任意の外部手続きを呼び出し，その結果を値として返す特別な関数で，Elhadad らが生成システム FUF[Elhadad 92]に導入したものと同じである．例えば，FD(25/1)の external(curent_theme) は直前の文で主題になっていた個体を返す役割を果たし，この情報を用いることによって主題の省略を決定できる．Appelt の TELEGRAM[Appelt 83]も参照表現を選択するための外部手続きを GFD から呼び出す機構を備えているが，external は TELEGRAM の機構の一般化とみなせる．external は，読み手のモデルやそれまでの選択の結果など，文法を取り巻く環境とのインタフェースであり，これを利用すれば動的に変化する制約や優先度を扱うことができる．

4. 依存指向性バックトラックによる草稿の修正

4・1 修正サイクル

初期生成が終了すると，システムは最初の修正サイクルに入る．**3・1節**で示した例では，まず，評価部が草稿(2 a)を評価し，問題を検出する．**2章**で述べたように，本論文では評価部の評価基準として係り受けの曖昧性と文の複雑さを取り上げる．日本語の文の複雑さとして表1に示す基準を考える．

表中の係り受けの深さは，“[[[太郎の]友達の]家]”のように係り受け関係が入れ子になるときの構造の深さを意味する．この基準に照らすと，第1文の「資料室」にかかる名詞修飾節「正門に…4階にある」の係り受け関係は深さが4なので深すぎる．次に，修正プランナが，この問題を解消するためには(n_seg)における FD(13/2)の選択を変更すればよいことを判断する．最後に，表出部が FD(13/2)の代わりに FD(13/3)を選択し，草稿(2 b)を生成する．

(2 b) あなたが探している資料は資料室に保管されています．資料室は正門に最も近い建物の4階にあります．ただし，[φハ]7月21日から8月31日までの夏休みの間は利用できません．

これで一つの修正サイクルが終了する．システムは以下同様の処理を繰り返す．

修正の際に選択の変更を行う選択点，すなわちバックトラックの戻り先を CCP(culprit choice point)と呼ぶ．上の例では，(n_seg)における選択点 FD(13)(図5)が CCP に当たる．上で述べたように，修正プラニングは CCP を特定する処理に相当し，草稿の書換えは単一化の処理を CCP まで戻し，新しいパスをたどることに相当する．本システムでは，真偽値維持システム(TMS)[Doyle 79]で用いられている依存指向性バックトラック（DDB）を応用することによって，これら二つの処理を効率的に行う．DDB を用いる利点は次の二つに要約できる．

- バックトラックの戻り先を修正規則と呼ばれるヒューリスティクスによって特定することができるため，単純な時系列バックトラックに比べ，CCP の同定に要するバックトラックの数を大幅を抑制できる．
- 草稿の書換えの際，変更した選択に依存する選択だけを再計算するため，時系列に沿って選択を取り消す手法に比べ，CCP から新たな草稿を生成する際の計算量が少ない．

表1 文の複雑さの評価基準

評価項目	上限	下限
文の長さ (語数)	40	5
係り受けの深さ	3	0
中心埋込みの深さ	3	0

4・2 依存関係ネットワーク

システムは，TMS と同様，依存関係ネットワークを用いて DDB を実現する．WFD 中の各素性に対し固有のノード（これを素性ノードと呼ぶ）を割り当て，素性ノードの集合として WFD を表現する．各素性ノードは次のようにパスと素性の組 〈*path, feature*〉で表される．

〈(n n agt), pred : library#1〉

初期生成を開始する時点での依存関係ネットワークは，入力の WFD が持つ素性ノードの集合である．生成過程では，素性ノードの集合として表現した WFD を GFD と単一化する．このとき，GFD 中の各選択点と WFD 中の素性との間の依存関係を，対応する FD ノードと素性ノードの間に正当化のアークを張ることによって記録する．FD ノードはパスと選択した FD の識別子の組 〈*path, FD_id*〉で表現する．

図9は，依存関係ネットワーク中の**3・1節**で述べた FD(13/2)の選択に対応する部分である．FD ノード〈(n_seg, FD(13/2)〉に向かうアークを持つ素性ノード(〈(n n agt), pred : library#1〉など）は，FD(13/2)に含まれる素性のうち単一化以前から WFD に存在した素性に相当し，これらの連言が FD(13/2)の正当化(必要条件)に当たる．他方，FD ノードから外に向かうアークの先の素性ノード（〈n n agt restr), embedded : +〉など）は単一化によって新たに WFD に付加された素性に相当する．システムは，これら FD ノードと素性ノードの間の依存関係のほかに，FD ノード間の依存関係もネットワーク上に記録する．〈(n_seg), FD(10/3)〉から〈(n_seg), FD(13/2)〉に張られたアークがその例である．

表出部は FD を単一化するたびにネットワークを更

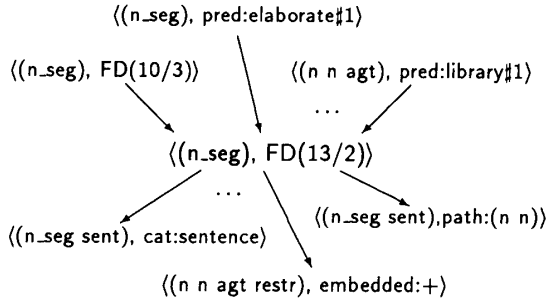


図9 依存関係ネットワーク

新する．単一化ではWFD上に生成された素性はそれ以後つねに制約として働くので，ネットワークは必ず非循環有向グラフになる．また，各ノードはたかだか一つの正当化しか持たない．FDノードは *in, out, unknown* のいずれかの状態を，素性ノードは *in, out* のいずれかの状態をとる．各ノードは，最初に生成されたとき *in* をとり，親ノードの少なくとも一つが *out* のとき *out* になる．アークで表現されるノードの正当化は必要条件にすぎないため，一般に *out* のノードを状態の伝搬によって *in* に変えることはできない．したがって，*out* のノードはネットワークから取り除く．*unknown* はexternalによって生じる依存関係を扱うための状態である．externalの値は文脈に依存して変化する可能性があるので，システムはexternalを含むFDが選択の変更後も単一化できるかどうか調べる必要がある．しかしながら，externalを含むFDの選言は一般に語用論的・文脈的優先度に関する選択点に相当するので，選択の変更後も再利用できる可能性が高い．そこで，選択の変更後，externalを含むFDに対応するFDノードに状態 *unknown* を与える．*unknown* は，*out* と異なり，子ノードに伝搬しない．

4・3 修正プラニング

修正プラニングでは修正規則を参照してCCPを特定する．**4・1節**の例では次の修正規則を適用する．Path, Path 1は変数を表す．

```
If too_deep (Path)∧
   state_in (⟨Path, cat : rel_clause⟩)∧
   constituent_of (Path 1, Path)∧
   state_in (⟨Path 1, embedded : +⟩)
then remove (⟨Path 1, embedded : +⟩).
```

このように，修正規則には，現在の草稿からどの素性を取り除けばよいかをプロダクション規則の形式で記述する．修正プラニングでは，適用可能な修正規則のなかから最も優先度の高い規則を選択し，取り除くべき素性を特定する．あとは，依存関係ネットワークを参照すれば，容易にCCPが特定できる．この例では，⟨(n n agt restr), embedded : +⟩を取り除くべき素性として特定し，図9に示したネットワークを参照して，FDノード⟨(n_seg), FD(13/2)⟩をCCPとして特定する．

4・4 草稿の書換え

草稿の書換えアルゴリズムを示す．ノードの状態は各処理ごとに伝搬するものとする．

1. 修正プラニングでCCPとして特定されたFDノードの状態を *out* にする．状態伝搬後の *in* の素性ノードの集合が現在のWFDである．
2. externalを含むFDに対応するFDノードが存在し，そのパスが統語構造中でCCPのパスの下または右にある場合，そのFDノードの状態を *unknown* にする．
3. 未決定の選択点を統語構造についてトップダウン左優先に探す．すべての選択が決定していれば書換え成功で，処理を終える．未決定の選択点は，選択の変更で新たに生じた選択点のほかに，対応するFDノードが *out* または *unknown* であるものを含む．
4. 3で見つけたの選択点のFDノードが：
   a. CCPであるとき4dへ．
   b. *unknown* のとき，親のexternalノードについて単一化を試み，成功すればFDノードを *in* にして3へ，失敗すれば *out* にして4dへ．
   c. 存在しないとき，通常の単一化を試みる．成功すれば3へ，失敗すれば5へ．
   d. 残りの選択肢について通常の単一化を試みる．成功すれば3へ，失敗すれば5へ．
5. 修正プランナに失敗メッセージを出して終了．

主な処理は，1でCCPに依存する選択だけを取り消し，3と4でそれらの選択点について再計算することである．CCPに依存する選択だけをやりなおすため，CCPの後で行った選択のすべてを取り消す時系列バックトラックより効率的な書換えが実現できる．5は書換え失敗の場合に相当し，このときシステムは別の修正規則を適用するか，別の問題点の解消を試みる．

次に，**4・1節**の修正例を使って処理2,4bの意味を説明する．草稿(2b)では第2文の主題「資料室は」が省略されている．これは，選択⟨(n_seg), FD(13/2)⟩を変更した結果，(2a)の最初の文が二つに分割され，⟨(s_seg sent), FD(25/1)⟩（図8）中のexternalの評価値がdocument#1からlibrary#1に変わったためである．この場合，システムはFD(25/1)からFD(25/2)に選択を変更しなければならない．このように，選択間にはexternalを介して依存関係が存在するが，これらはネットワーク中に明示することができない．2と4bはexternalを介した依存関係を管理するための処理である．

5. 評　　価

一般に，TMSで用いられているDDBは，むだな再計算を防ぐことができる半面，ネットワーク管理のた

めのコストが無視できない．そこで，本稿で述べたアルゴリズムの有効性を実験システムを用いて定量的に評価した．

システムはすべてProlog上に実装し，ネットワークの更新にはassert, retractを用いた．比較対象として時系列バックトラックとElhadadの手法を用いた．ElhadadのFUFでは，FUGの単一化に失敗したとき効率良くCCPまで戻れるように，特定の素性が単一化に失敗したときの戻り先の候補をbk-classという特殊なクラスとして文法中に記述することができる[Elhadad 92]．この手法を我々の例題に応用するためには，「係り受けの深さが深すぎる」という問題が生じたときの戻り先の候補として図5の選択点FD(13)をbk-classに指定すればよい．

実験では，図3に示したWFDを入力し，草稿(2 a)の生成を経て草稿(2 b)を生成するまでの計算量を比較した．草稿(2 b)の生成に到達するまでに生成した草稿の数とそれに要した素性の単一化の回数を表2に示す．我々の手法では，2度目の生成で草稿(2 b)が得られる．しかしながら，bk-classを用いると，草稿(2 b)を生成する前に(2 b′)のような不必要な草稿を生成してしまう．

(2 b′) あなたが探している資料は正門に最も近い建物の4階にある資料室に保管されています．ただし，資料室は夏休みの間は利用できません．夏休みは7月21日から8月31日までです．

これは，bk-classに指定された選択点FD(13)が第2文にも存在するためである．また，名詞句に関する選択点のように頻繁に生じる選択点がbk-classに指定された場合，(2 b′)のようなむだな草稿の生成はさらに増えると予測できる．草稿の評価に要するコストを考慮に入れた場合，この問題は深刻である．これに対し，我々の手法ではWFDにおけるパスと選択肢の組を用いてCCPを特定するため，一度で適切なCCP(この例では第1文の選択点〈(n_seg), FD(13/2)〉)に戻ることができる．また，bk-classを用いる場合，時系列に沿って選択を取り消すため，CCPから新たな草稿を生成する過程でもむだな再計算が多くなる．

次に，CPU時間の比較を表3に示す*2．表は草稿(2 b)を生成するまでの実行時間を示しており，ネットワーク管理のコストを支払っても，本手法のほうがbk-classを使うより効率的であることがわかる．本手法の実行時間の内訳のうち，ネットワーク管理に要する時間を表4に示す．**4・2節**で述べたように，本システムはネットワークを生成しながらFUGの単一化を行う．表4の初稿と第2稿の数値はネットワーク生成に要した時間である．本システムではPrologのassert, retractを用いた単純な実装を行っているが，それでもネットワーク生成に要する時間は一つの草稿の生成に要する時間の1割に満たない．表4の状態伝搬の数値5.4 secは，初稿を生成した状態からCCPに戻るまでの処理(**4・4節**の処理1と処理2）に要した時間である．これに対し，表には示さなかったが，bk-classを用いた場合でも，第2稿の生成後CCPまで戻るのに2.2 secかかる．このように，ネットワーク管理のコストは，単一化の回数を減らすことによる効率化の効果に比べて，十分小さいことがこの実験で確かめられた．この理由として，我々のネットワークが非循環有効グラフであり，さらにノードが単一の正当化しか持たないため，状態伝搬が単純であることがあげられる．

表2 生成した草稿の数と必要な単一化の数の比較

	草稿の数	単一化の数
時系列バックトラック	17	10855
bk-class	3	2401
本手法	**2**	**1177**

表3 CPU時間の比較

	初稿	第2稿	第3稿	合計
時系列バックトラック	32.1	11.0	35.4	440.3
bk-class	32.0	30.5	31.5	94.0
本手法	**34.5**	**23.5**	—	**58.0**

(SONY NEWS 3860; [sec])

表4 ネットワーク管理に要する時間

初稿	状態伝搬	第2稿	合計
2.4	5.4	1.1	8.9

6. おわりに

本論文では，文章生成における推敲機能の重要性について議論し，how-to-sayに関する推敲機能を備えた生成モデルの実装方法を提案した．

文章生成を探索問題とみなした場合，各選択点で局所的に優先された選択の集合がシステムの出力に対応する．しかしながら，表層依存性評価基準については，評価に必要な情報が生成の途中では部分的にしか得られないため，選好規則だけでは扱いにくい．本論文では，この問題の解決方法として推敲機能を備えたMeteerの生成モデルを採用し，モデルを実装するために「選好規則が与える不十分な優先度を修正規則によって補正する処理」として推敲機能を捉えた．この

*2 Prolog上でコンパイルした場合のCPU時間はそれぞれ表3の値の42～44%である．

モデルでは，評価に必要なすべての情報が得られた状態で表層依存性評価基準について評価するので，それらの扱いが容易になる．また，選好規則が将来の選択の予測を用いないため，個々の規則を単純化することができ，システムの実装と保守に有利である．どのような評価基準を評価部で評価し，修正規則として実現するかについては，今後具体的な応用システムの構築を通じて考察する必要がある．

推敲機能を実装したシステムの代表的な例にはMannとMooreのKDS[Mann 81]，Gabrielの*Yh*[Gabriel 88]がある．しかしながら，これらのシステムでは，複数の命題を一つの文に統合する処理に推敲機能を限定し，生成過程の途中で生成される中間表現を規則に基づいて変形するにすぎない．すなわち，生成した文章を評価し，書き換えるシステムではない．また，一度決定した選択は変更しないという強い制約を与えているため，現在の選択と将来の選択が相互に依存する問題がうまく扱えない．このように，KDSや*Yh*の推敲機能は，本論文で述べた推敲機能とは性質が異なり，表層依存性評価基準を考慮する際に生じる問題を解消できない．

これに対し，本論文で述べたような推敲機能の実装についてはこれまでほとんど議論されていない．これについては文章の評価の難しさが一つの要因にあげられる．評価の実現については本論文でもほとんど議論しなかったが，我々のモデルではある特定の評価基準についてのみ草稿を評価するため，一般的な評価を仮定したYazdaniの心理学的モデルに比べ，評価部の実装はある程度容易であると考えられる．ただし，高度な解析処理技術を必要とする係り受けの曖昧性の評価については，現在のシステムでは人手で行っている．

草稿の書換えは探索空間における選択の変更として実現することができる．本論文では，選択を効率的に変更するために，DDBのメカニズムをFUGの単一化の制御に導入する手法を提案した．DDBの実現は選択の順序を動的に入れ換えることに相当する．この点で選択の順序を規定しない単一化文法はDDBに適している．書換えの際には照応表現など，選択の変更の文脈的な影響も考慮する必要があるが，これについてはexternalを介した選択間の依存関係を用いて対処できることを示した．DDBを導入した生成システムの例としてElhadadのFUFがあげられる．5章では，FUFと我々の手法を主に時間的効率の面から比較し，我々の手法が効率の向上に有効であることを示した．また，FUFでは，バックトラックの戻り先という制御に関する情報をbk-classとして文法中に記述する必要があるが，これは文法の開発や保守の点で望ましくない．

本論文で導入した修正規則は，選好規則と同様，経験から得られるヒューリスティクスであり，今後大規模な文法と多数の生成例を用いて有効性を検証する必要がある．また，修正規則の表現力についても議論の余地がある．例えば，動詞句の名詞化は文の構造の簡単化や係り受けの曖昧性の解消に有効な修正手段であるが，現在の手法ではそのような規則が書けない．現在の修正ルールは特定の素性を草稿から取り除くように指示するだけで，ある素性を別の素性と取り換えるよう指示することはできないからである．

本論文では，文章生成における表層からのフィードバックに焦点を当て，それを推敲機能と呼んできた．しかしながら，例えば文の構文的な複雑さについては，文章あるいは一文全体を表層化しなくても，部分的には評価できる可能性がある．部分的な評価で問題が検出された場合，ただちにバックトラックするほうが望ましい．このような生成と推敲の統合についてはWongが一つのモデルを提案している[Wong 88]．しかしながら，Meteerと同様，Wongも実装についてはほとんど議論していない．現在，本論文で述べた手法を一般化し，生成途中での単一化の失敗によるバックトラックを含めて，FUGの単一化の制御をDDBで統一する手法を検討している．

謝　辞

示唆に富む有益なコメントをいただきました査読者の方に感謝いたします．

◇参考文献◇

[Appelt 83] Appelt, D. E.: TELEGRAM: A grammar formalism for language planning, *Proc. Int. Joint Conf. on Artificial Intelligence*, pp. 595-599 (1983).

[Doyle 79] Doyle, J.: A truth maintenance system, *Artif. Intell.*, pp. 231-272 (1970).

[Elhadad 92] Elhadad, M. and Robin, J.: Controlling content realization, Dale, R., Hovy, E., Rösner, D. and Stock, O. (eds.), *Aspects of Automated Natural Language Generation*, pp. 89-105, Springer-Verlag, Lecture Notes in Artificial Intelligence, Vol. 587 (1992).

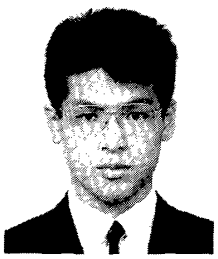
[Gabriel 88] Gabriel, R. P.: Deliberate writing, McDonald, D. D. and Bolc, L. (eds.), *Natural Language Generation Systems*, chapt. 1, pp. 1-46, Springer-Verlag (1988).

[Hovy 88] Hovy, E. H.: *Generating Natural Language under Pragmatic Constraints*, Lawrence Erlbaum Associ-

ates (1988).
[Kay 84] Kay, M.: Functional Unification Grammar: A formalism for machine translation, *Proc. Int. Conf. on Computational Linguistics*, pp. 75-78 (1984).
[Mann 81] Mann, W. C. and Moore, J. A.: Computer generation of multiparagraph English text, *Am. J. of Computational Linguistics*, Vol. 7, No. 1, pp. 17-29 (1981).
[Mann 87] Mann, W. C. and Thompson, S. A.: Rhetorical Structure Theory: Description and construction of text structures, Kempen, G. (ed.), *Natural Language Generation*, chapt. 7, pp. 85-96, Martinus Nijhoff (1987).
[Meteer 86] Meteer (Vaughan), M. M., and McDonald, D. D.: A model of revision in natural language generation, *Proc. Annual Meeting of the Association for Computational Linguistics*, pp. 90-96 (1986).
[桃内 89] 桃内佳雄, 高橋 晃: 生成文章の結束性の良さを考慮した文章生成に関する研究, 言語情報処理の高度化研究報告 7, 言語情報処理の高度化の諸問題, pp. 359-378 (1989).
[Scott 90] Scott, D. R. and de Souza, C. S.: Getting the message across in RST-based text generation, Dale, R., Mellish, C. and Zock, M. (eds.), *Current Research in Natural Language Generation*, chapt. 3, pp. 47-74, Academic Press (1990).
[徳永 91] 徳永健伸, 乾健太郎: 1980年代の自然言語生成(1)-(3), 人工知能学会誌, Vol. 6, No. 3-5 (1991).
[Wong 88] Wong, W. C. and Simmons, R. F.: A blackboard model of text production with revision, *Proc. AAAI Workshop on Text Planning and Realization*, pp. 99-106 (1988).
[Yazdani 87] Yazdani, M.: Reviewing as a component of the text generation process, Kempen, G. (ed.), *Natural Language Generation*, chapt. 13, pp. 183-190, Martinus Nijhoff (1987).

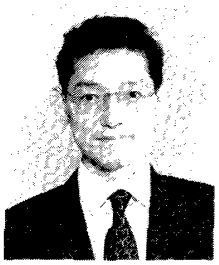
〔担当編集委員・査読者：石崎 俊〕

著者紹介



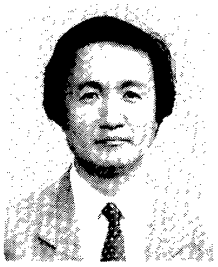
乾 健太郎（学生会員）

1967年生まれ. 1990年東京工業大学工学部情報工学科卒業. 1992年同大学院理工学研究科修士課程修了. 現在, 同大学院理工学研究科博士課程在学中. 自然言語生成に関する研究に従事, 特に文章の推敲に関心を持つ. 情報処理学会会員.



徳永 健伸（正会員）

1961年生まれ. 1983年東京工業大学工学部情報工学科卒業. 1985年同大学院理工学研究科修士課程修了. 同年, (株)三菱総合研究所入社. 現在, 東京工業大学工学部情報工学科助教授. 博士(工学). 自然言語処理, 計算言語学に関する研究に従事. 情報処理学会, 日本認知科学会, 人工知能学会, 計量国語学会, Association for Computational Linguistics 各会員.



田中 穂積（正会員）

1964年東京工業大学理工学部制御工学科卒業. 1966年同大学院修士課程修了. 同年, 電気試験所（現, 電子技術総合研究所）入所. 1983年東京工業大学工学部情報工学科助教授. 1986年同大学教授. 工学博士. 人工知能, 自然言語処理の研究に従事. 情報処理学会, 電子情報通信学会, 日本認知科学会, 計量国語学会各会員.