自然言語処理 92－3
（1992. 11. 19）

文章生成における推敲機能の実現について

乾健太郎, 徳永健伸, 田中穂積
東京工業大学 工学部
{inui,take,tanaka}@cs.titech.ac.jp

文章生成には語彙選択や語順などさまざまな要素に関する選択が必要である. 従来のシステムでは, 規則を用いて各選択点における選択肢の優先度を局所的に計算するのが一般的だった. しかしながら, 選択肢間の優先度の根拠となる評価基準のなかには, 評価に将来の選択の予測や大域的な情報を必要とするものもある. それらのすべてを局所的な規則で扱うアーキテクチャは, 規則が複雑になりすぎるため, 実現や保守の点で望ましくない. 本稿では, この問題に対して推敲機能の導入が有効であることについて議論し, 推敲機能を備えた生成モデルとその実装方法を提案する. また, その手法を実装した日本語文章生成システムについて述べ, その性能を評価する.

Implementation of Revision in Text Generation

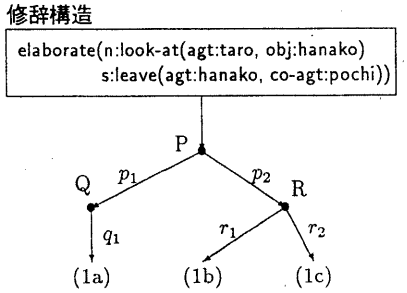
INUI Kentaro, TOKUNAGA Takenobu and TANAKA Hozumi
Department of Computer Science, Tokyo Institute of Technology
(2–12–1 Ôokayama Meguro-Ku Tokyo 152 Japan)

To generate good text, many kinds of decisions should be made. Many researchers have been engaged in searching for the heuristics that would make an appropriate decision locally at each decision point. However, even if such heuristics were found, there are still certain kinds of problems that are difficult to consider during the generation process. Those problems can be more easily 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 propose a computational model of text revision. We also discuss its implementation issues and introduce dependency directed backtracking in order to realize efficient revisions. 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 [10]. 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 [6, 10] と呼ばれる構造化された命題の集合で表現する. つぎに, how-to-say を選択し, 文字列を出力する. 本稿では, とくに 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 1 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中から, 質の高い文章を生成するパスを選択することである.



p_1: hanako を look-at の目的語にする
p_2: leave を look-at の目的語にする
q_1: 主題を文頭に置く
r_1: 従属節の語順を主格, 随伴格にする
r_2: 従属節の語順を随伴格, 主格にする

図 1 文章生成の探索空間

多くのシステムでは, 個々の選択点で選択肢の局所的な優先度を求め, それにしたがって選択を行なう. 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. 本稿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. 文章生成の研究の一つ目的はどのような選好規則が適当かを究明することである. たとえば, 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 [?, 10]. 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.

係り受けのあいまい性を文章の評価基準の一つとして考えてみよう. 入力として図 1のような修辞構造を仮定する. いま, システムが図の選択肢 p_1, q_1を選択して, 文 (1a) を生成したとしよう.

(1a) 太郎はポチと去っていく花子を見ていた.

(1b) 太郎は花子がポチと去っていくのを見ていた.

(1c) 太郎はポチと花子が去っていくのを見ていた.

この文は「ポチと」の係り先があいまいなため, 複数の解釈が可能であり, 望ましい文とは言えない. 一方, p_2, r_1を選択すると, あいまい性のない文 (1b) を生成することができる. この例では, p_1を選択するとあいまい性を持つ文を生成するので, あいまい性を避けるためには選択点 P で p_2を選択すべきだったことがわかる.

しかしながら, 出力文があいまい性を持つか持たないかは P 以降の選択にも依存する. たとえば, p_2を選択しても, R で r_2を選択すれば, 文 (1c) のようなあいまい性が生じてしまう. つまり, 選択点 P では, p_1よりも p_2の方が望ましいとは必ずしも言えない. 選択点 P では, まだ語順や語などが決まっていないため, 出力文のあいまい性を評価するために必要な情報が部分的にしか得られないからである. このように, 評価基準の中には, 選択肢の優先度を局所的に評価するのが困難なものがある. このような評価基準の例に, 係り受けのあいまい性, 文や節の長さなどの統語的複雑さなどがある.

この問題は選択の順序をうまく制御すれば解消するように見えるが, 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 [4], 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.

そこで, 本稿では, 一度生成した文章を推敲することができる生成モデルを提案し, その実現について述べる. 推敲機能を導入することによって, 係り受けのあいまい性や文の複雑さのような評価基準については, 文章を生成した後で評価することができる. 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.

文章生成における推敲機能の役割とその重要性についてはすでにいくつかの議論がある. Meteer[7], Yazdani[9] は, 一度生成した文章に対する評価・修正を繰り返すモデルをそれぞれ提案し, それについて次のような利点をあげている.

1. 推敲過程には心理学的妥当性がある.

2. 構造的に類似した2つの文を1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.

3. 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.

本稿では, 2と3の観点から推敲機能を導入し, より具体的なモデル化を行なう. また, 従来ほとんど議論されてこなかった推敲機能の実装について一つの手法を提案する.

以下, 2節で文章生成モデルについて述べ, 3, 4節で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. さらに, 5節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

図2に推敲機能を備えた生成モデルを示す. 本稿では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つづく推敲過程からなる.

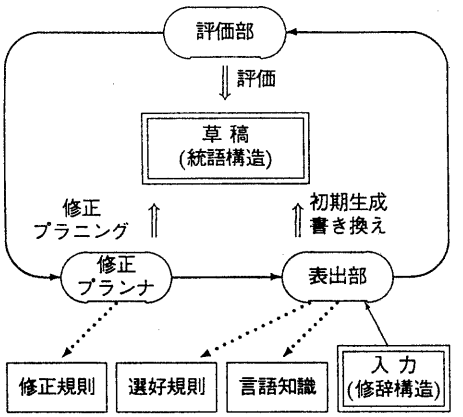


図2 文章生成モデル

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ない, 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なう. 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節で述べたように, 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なった結果であり, 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.

推敲過程は修正サイクルの繰り返しからなる. 各修正サイクルでは, 評価, 修正プラニング, 書き換えをおこなう. まず, 評価部が草稿を評価し, 問題点を検出する. つぎに, 修正プランナが検出された問題点の中から1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したがって実際に草稿を書き換える. これで一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.

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2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節で述べたように, 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中に押し込めると, 選好規則が複雑になり, システムの実現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, 局所的な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, 大域的な情報を必要とする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評価部の評価に基づく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. 本稿では, 評価部で扱う評価基準の例として, 係り受けのあいまい性と文の構造の複雑さ (4.1項) を取り上げるが, どのような評価基準を評価部で扱うべきかは今後の研究課題である.

3 言語知識と初期生成

3.1 機能単一化文法

1節で述べたように, 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, 草稿の書き換えは, 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. もとの草稿と書き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)[5] を採用している. FUG では, 意味構造や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なう.

入力となる WFD の例を図 3に, GFD の一部を図 4, 5, 7に示す. 修辞関係には Mann の RST[6] を用いる. n は主部 (nucleus) を, s は従属部 (satellite) を表す. 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 1 つの場合は自分自身を指し, 無い場合は絶対パスを示す. @で始まる素性名・素性値は型つき変数を表す.

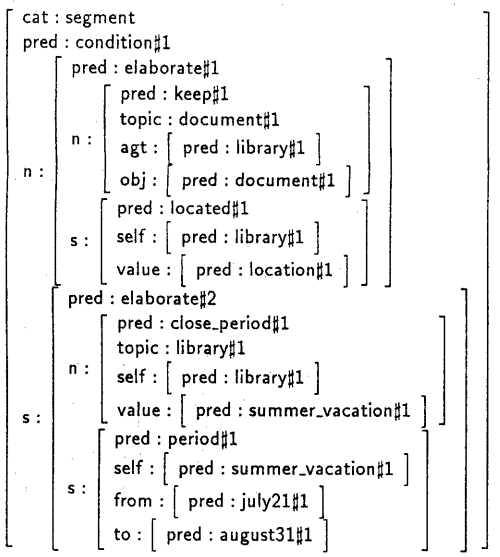


図 3　WFD の初期状態

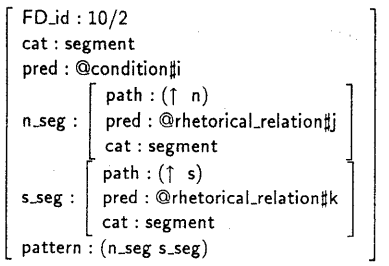


図 4　GFD の一部

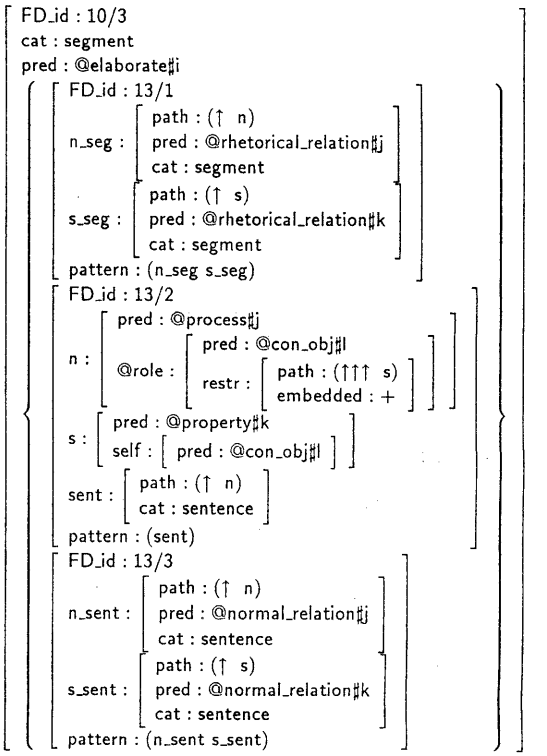


図 5　命題の埋め込みに関する GFD

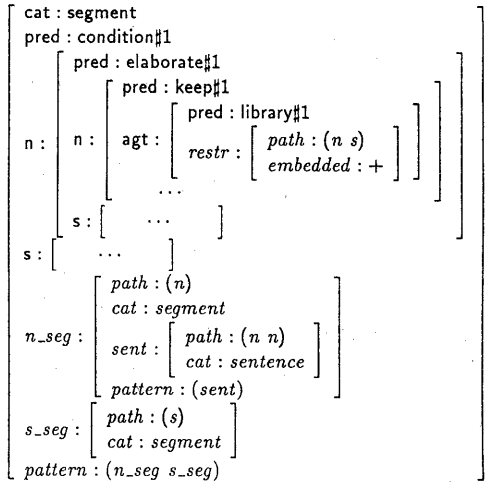


図 6　単一化後の WFD

生成過程では, まず WFD のパス () の下にある FD と図 4の GFD との単一化が成功し, つぎにパス (n_seg) の下にある FD と図 5の GFD との単一化が成功する. 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 1 つだけ選択する. ここでは FD(13/2) との単一化に成功し, 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. このように, 単一化は WFD 上をトップダウンに進み, 表層の文字列の生成に必要な統語的・語彙的素性がすべて決定した時点で終了する. この例では草稿 (2a) を生成する.

(2a) あなたが探している資料は正門にもっとも近い建物の 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.

生成過程には参照表現の選択も含まれる. たとえば, document♯1 が「あなたが探している資料」と表現されているが, これは, 読み手が個体 document♯1 を同定するために必要な情報を表出部が対象領域のモデルから取り出し, 表層化した結果である.

3.2 言語知識と生成過程

図 6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 [8] 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越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項で述べたように, 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節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, 文脈に依存して動的に優先度が変化する場合もある. たとえば, 草稿 (2a) 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, 省略する方が自然であろう (4.1項の草稿 (2b))[?]. この知識は図 7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[3] に導入したものと同じである. たとえば, 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はたし, 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.

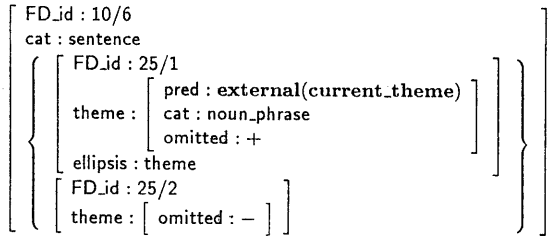


図 7 主題の省略に関する GFD

Appelt の TELEGRAM[1] は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 の機構の一般化と見なせる. external は, 読み手のモデルやそれまでの選択の結果など, 文法をとりま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 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項で示した例では, まず, 評価部が草稿 (2a) を評価し, 問題を検出する. 2節で述べたように, 本稿では評価部の評価基準として係り受けのあいまい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 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
中心埋め込みの深さ	3	0

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. つぎに, 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. さいごに, 表出部が FD(13/2) の代わりに FD(13/3) を選択し, 草稿 (2b) を生成する.

(2b) あなたが探している資料は資料室に保管されています. 資料室は正門にもっとも近い建物の 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なう選択点，すなわちバックトラックの戻り先をCCP (culprit choice point) とよぶ．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，草稿の書き換えは単一化の処理をCCPまでもどし，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) [2] 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2つの処理を効率的に行なう．DDBを用いる利点は次の2つに要約できる．第1に，バックトラックの戻り先をヒューリスティクス (修正規則) 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に抑制できる．第2に，草稿の書き換えの際，変更した選択に依存する選択だけを再計算するため，時系列にそ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．

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 (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．素性ノードは次のようにパスと素性の組⟨*path, feature*⟩で表現される．

⟨(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8は，依存関係ネットワーク中の3.1項で述べた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他に，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を単一化する度にネットワークを更新する．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．また各ノードは高々1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1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播によって *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，子ノードに伝播しない．

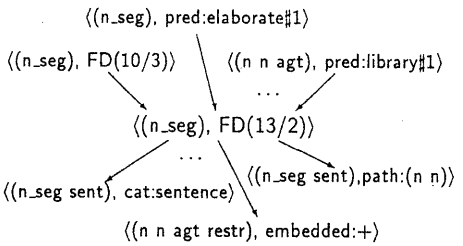


図8 依存関係ネットワーク

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項の例ではつぎの修正規則を適用する．Path, Path1は変数を表す．

```
If too_deep(Path) ∧
   state_in(⟨Path, cat:rel_clause⟩) ∧
   constituent_of(Path1, Path) ∧
   state_in(⟨Path1, embedded:+⟩)
then remove(⟨Path1, 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中からもっと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8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 草稿の書き換え

草稿の書き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播するものとする．

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．状態伝播後の*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き換え成功で，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他に，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の後で行な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き換えが実現できる．5は書き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．

つぎに，4.1項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2つに分割され，⟨(s_seg sent), FD(25/1)⟩(図7)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を介しての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，本節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の生成システムFUFでは，FUGの単一化に失敗したとき効率よく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[3]．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(2a)の生成を経て草稿(2b)を生成するまでの計算量を比較した．草稿(2b)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．

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**

本節で述べたように我々の手法では2度目の生成で草稿(2b)が得られる．しかしながら，bk-classを用いると，草稿(2b)を生成する前に(2b')のような不必要な草稿を生成してしまう．

(2b') あなたが探している資料は正門にもっとも近い建物の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-lcassに指定された場合，(2b')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，1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，時系列にそ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．

つぎに，CPU時間の比較を表3に示す．表は草稿(2b)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, 本手法の方が bk-class を使うより効率的であることがわかる.

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])

6 おわりに

本稿では, 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とその実装方法を提案した.

文章生成を探索問題と見なした場合, 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, 係り受けのあいまい性や文の複雑さなど, いくつかの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, 局所的な選好規則としては実現しにくい. 本稿では, 選好規則が与える不十分な優先度を修正規則によって修正するモデルを提案した. このモデルでは, 選好規則が局所的な情報しか用いないため, 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. また, 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, その有効性についても検証が必要である.

推敲機能を備えたモデル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稿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, あらゆるレベルの評価を仮定した他のモデルに比べ, 評価部の実装は比較的容易であると考えられる. ただし, 係り受けのあいまい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なっている.

草稿の書き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. 本稿では, 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. 書き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節では, FUF と我々の手法をおも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. 本稿で示した手法はこの問題も解決している.

参考文献

[1] D. E. Appelt. TELEGRAM: A grammar formalism for language planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 595–599, 1983.

[2] J. Doyle. A truth maintenance system. *Artificial Intelligence*, pp. 231–272, 1979.

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

[4] E. H. Hovy. *Generating Natural Language under Pragmatic Constraints.* Lawrence Erlbaum Associates, 1988.

[5] M. Kay. Functional Unification Grammar: A formalism for machine translation. In *Proceedings of the International Conference on Computational Linguistics*, pp. 75–78, 1984.

[6] W. C. Mann and S. A. Thompson. Rhetorical Structure Theory: Description and construction of text structures. In G. Kempen, editor, *Natural Language Generation*, chapter 7, pp. 85–96. Martinus Nijhoff, 1987.

[7] M. M. Meteer (Vaughan) and D. D. McDonald. A model of revision in natural language generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 90–96, 1986.

[8] D. R. Scott and C. S. de Souza. Getting the message across in RST-based text generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, chapter 3, pp. 47–74. Academic Press, 1990.

[9] M. Yazdani. Reviewing as a component of the text generation process. In G. Kempen, editor, *Natural Language Generation*, chapter 13, pp. 183–190. Martinus Nijhoff, 1987.

[10] 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.

[11] 徳永健伸, 乾健太郎. 1980年代の自然言語生成 1–3. 人工知能学会学会誌, Vol. 6, No. 3-5, 1991.