複合文書処理システム MOE におけるインクリメンタ ル・フォーマッター (その2)

1X-6

松下武史　中野秀紀　V.P.Shrestha　坂入隆

日本アイ・ビー・エム株式会社　東京基礎研究所

1. はじめに

コンピューターのユーザーインターフェイスの進歩にともない，文書フォーマッターも対話型で WYSIWYG タイプ（以後対話型と呼ぶ）のものが多くなってきた。対話型の文書フォーマッターでは，ユーザーはフォーマットした結果を画面上で見ながら，その上にテキストの入力，編集，あるいはフォーマットの指示を行うことができ，非常に使いやすい。フォーマット機能に関しては，従来からあるバッチ型の文書フォーマッターのほうが強力であると言われることもあるが，対話型の使いやすさは非常に大きな利点であり，今後の文書フォーマッターの主流になると思われる。我々はバッチ型の文書フォーマッターの強力な機能を，対話型タイプの文書フォーマッターで実現することをひとつの目標としている。

対話型の文書フォーマッターにとって重要なことは，ユーザーの入力，指示に対する応答の速さである。ユーザーの入力，指示に対して瞬時にフォーマット結果を表示すれば非常に使いやすいが，もし，応答に数十秒もかかると対話型の使用には耐えられない。より高速な応答の実現のために，われわれは，入力，編集，あるいはフォーマットの指示があった時，必要最小限の部分のみを再フォーマットし，再表示することにした。これをインクリメンタル・フォーマッティングと呼んでいる。[1]

2. 文書のフォーマッティング

文書フォーマッターは，まずバッチ型のものが開発された。バッチ型の文書フォーマッターで，例えば図1のような文書を作るには，ユーザーは通常のテキスト・エディターを使って，入力となる文章のなかにフォーマット・コマンドを埋めこんだ図2のようなファイルを用意する。ここではコマンドは，'：' で始まる文字列からなる。

フォーマッターは，入力テキストを読みこんでその中のコマンドを解釈し，コマンドが要求するフォーマット処理を行ない出力を作成する。その際に，図3のような状態変数あるいはスタックを使用する。例えば，番号付けを行うために，章，節などの見出し番号変数にそれまでの最後の章，節などの見出し番号を覚えておく。また，字体の変更のように入れ子になった修飾指定がある場合は，コマンドに出会ったならば，その情報をスタックに押し込む。バッチ処理ではフォーマッターは，入力テキストを先頭から順に読んで状態変数を更新し，その現在値を参照して，見出しの番号付けや，文字の修飾を容易に行うことができる。

3. インクリメンタル・フォーマッター実現における問題点

一般にユーザーが次に文章のどこを修正するかあるいはどこにフォーマットの指示を行うかは予測がつかない。したがってインクリメンタル・フォーマッターではユーザーの指示に対応して文書のどこからでもフォーマット処理を開始できなければならない。

フォーマットをしようとしているテキストの中に見出しコマンドがあれば，番号付けをするために，その直前にふった章や節の番号を知る必要があり，また，文字の修飾では，フォーマットしようとしている所より前にどのような修飾の開始の指示があったかを知る必要がある。これは，テキストを先頭から読んでコマンドを処理することにより得られる情報であり，バッチ型のフォーマッターでは，前述の状態変数（スタックを含む）の値として知ることができるが，ランダムな位置に対してこれらの値を求めることは簡単ではない。したがって，インクリメンタル・フォーマッターでは見出しの番号付けや文字修飾の処理に特別の工夫がいる。

従来の対話型のフォーマッターでは，例えば，文字修飾の場合では，ユーザーは開始点と終了点を指定するが，コンピューター内部のデータとしては，開始点，終了点を記録せず，各文字に直接ゴシック，下線付きなどの修飾情報を持たせたり，見出しの作成の指示があれば，そこから入力テキストを逆にさかのぼって直前の見出しコマンドを見つけ，その番号を知るなどの方法，あるいは，文書の論理的な構造を表現するデータ構造を作りそれを用いて番号付けをする方法などがある。文字に修飾情報を持たせる方式では，強調したテキストの途中に文字を挿入するとその文字は修飾されないなどユーザーの意図と異なった結果になることがある。また，入力テキストをさかのぼる方式では，場合によって応答速度が非常に悪くなる，ロジカルな構造を表現するデータ構造はデータの管理が難しいなどの問題がある。

4. 高速インクリメンタル・フォーマッターの実現

MOE では，ロジカルな文書構造に基づいたテキスト・フォーマッティング機能を，対話型で高速かつ安価に実現する方法として，フォーマットの途中におけるフォーマッターの状態変数の値を一定間隔で記録することにした。状態変数の値を記録する間隔としては，処理の速度，データ量，データの管理の容易さなどから，現在はページ単位としている。

我々の文書フォーマッターは，フォーマット・コマンドを含んだ入力テキストを入力としてフォーマッティングを行ない，その結果として

Incremental Formatting Algorithm for the Composite Document Processing System MOE (2)
T. Matsushita, H. Nakano, V. P. Shrestha, T. Sakairi
IBM Research, Tokyo Research Laboratory

フォーマット結果データを作成する。フォーマット結果データは，ページ，コラムに対応した構造を持ち，行，コラム，ページに分割したテキストの出力用紙上での位置，文字の大きさやフォント，見出しの番号などの情報を持っている。このフォーマット結果データの各ページに，そのページの先頭における状態変数の値を持たせている。

ユーザーが画面上でテキストを指し見出しの作成など編集操作を指示すると，フォーマッターはフォーマット結果データを参照してどの文字が指されたかを判断する。入力テキストとフォーマット結果データは段落単位でポインターでリンクされており，ユーザーが指した文字の入力テキスト中の位置を知ることができる。同時にフォーマッターは，フォーマット結果データから，そのページの先頭の入力テキストの位置と，その時点におけるフォーマッターの状態変数の値も知る。この状態変数の値と入力テキストを用いて，そのページの先頭から正しいフォーマッティングを行う。ただし，編集操作を指示された位置までは入力テキストに含まれるコマンドを解釈して状態変数と更新するだけで良く，フォーマット結果データを作り直す必要はない。編集操作によって出力フォーマットが影響される段落を再フォーマットした後，そのページの最後まで入力テキストを走査して状態変数を更新する。その時の状態変数の値が次のページが持っている状態変数の値に等ければ，そこでインクリメンタル・フォーマッティングは終了する。等しくなければ次のページの状態変数の値を更新し，改ページなどで必要になった時に，次のページを再フォーマットする。

5. まとめ

フォーマット結果データの各ページの先頭に，フォーマッターの状態変数の値を記録することにより，ロジカルな文書構造に基づいたテキスト・フォーマッティング機能を，対話型で高速かつ安価に実現することが可能になった。これは，見出しの番号付けだけでなく，マクロ機能などを含むバッチ型の文書フォーマッターが持つより高度なフォーマット機能の実現も可能にする。

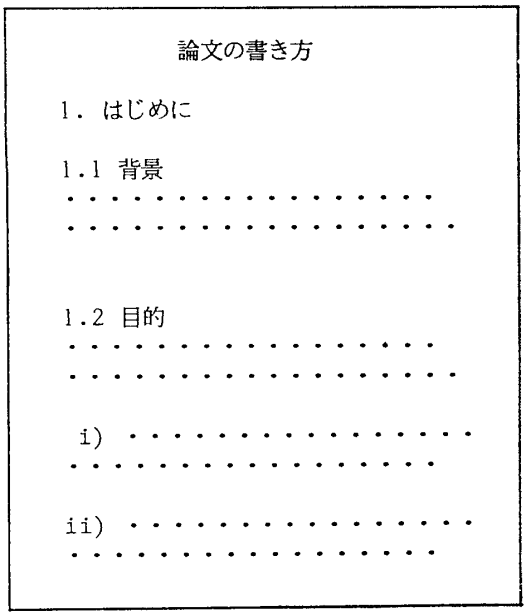

図 1: 文書フォーマッターの出力例

```
:title. 論文の書き方 :etitle.
:h1. はじめに :eh1.
:h2. 背景 :eh2.
:p. ・・・・・・・・・・・・
:h2. 目的 :eh2.
:p. ・・・・・・・・・・・・
:ol.
:li. ・・・・・・・・・・・・
:li. ・・・・・・・・・・・・
:eol.
```

図 2: バッチ型フォーマッターの入力テキスト

見出し番号	1.
小見出し番号	1.2
リスト番号	ii)
字体強調	ゴシック

図 3: 状態変数

文献

1. 坂入 et al.: 複合文書処理システム MOE における インクレメンタル・フォーマッター（その1）． 情報処理学会第37回全国大会.