共有エディタの編集制御の実装アルゴリズム

5Q−5

都築功兒　池井　寧　大川善邦
(大阪大学工学部)

1．　はじめに

組織における我々の活動は基本的にグループによる共同作業である．同一の文書を複数の人間が関与して開発するのもその1つである．従来の文書開発では，グループの各個人がそれぞれ独自にワークステーションに向かって文書を作成する段階と，文書の内容についてグループで協議する段階が全く分離して行われている．この場合，文書の作成と協議による変更が不連続となるため，開発効率が低下する可能性がある．従って，文書開発時の共同作業を効率化するためには，作成段階と協議段階を同時に進める環境が必要となる．

そこで本研究では，同一の文書を同時に2つのサイトで編集するための支援システムを開発している．具体的には，編集中の同一文書を各々のワークステーション上に示し，各自の注目している部分を指示した上で，同時に変更もできる共有エディタを設計した．これにより実際に相手と対面して会話しながら紙の上で同時に書き込むのと同様な環境の実現を考えている．共有エディタでは，各サイトの局所的変更を時間遅れなく処理するために，一時的に両サイトのテキストが異なる状態となる．相手からどの様な編集入力が到着しても意図した正しい操作が行われ，両者のテキストが最終的に一致するためには，テキストが適切に構造化されていなければならない．[1]本稿では共有エディタのための共有テキストの構造を示し，この構造に基づいて編集要求を処理する編集制御のアルゴリズムについて述べる．

2．　共有エディタにおけるテキストの構造化

2．1　対象とする共有エディタの設計仕様

共有エディタを制御するにあたって，考察対象を以下のように限定する．①編集対象は1次元有限文字列である．②編集操作は，1文字単位の挿入と削除，および対象位置を示すキャレットの移動だけである．③テキストの各文字は一意に識別できる．④編集入力は2つのサイトで発生する．⑤各サイトでは2つのキャレットが表示される．⑥各サイトの同時入力は，個別に一括して行われ，各々の入力が終了した時点で，もう一方のサイトでの入力が到着する．

2．2　共有テキストの構造

2つのサイトにおける入力後のテキストを一致させるために，テキストは図1の概念図で表される構造を持つとする．一番上のレベルのi，j，kは，両サイトの編集前に一致している文字列で，基準文字と呼ぶことにする．

文字の位置はその前後の文字を指定することにより一意に限定できる．このエディタでは，挿入位置を前後2文字で表し，これをフォーカスと呼ぶことにする．画面の表示では，自分のフォーカスを|，相手の注目している位置を示すために相手のフォーカスを¦で表す．さらに同じiとjの間に文字を挿入する際に，①を低位サイトの，②を高位サイトの入力位置とし，サイト毎に入力位置を区別する．したがって低位サイト，高位サイトの連続入力も位置が明確に決まるので，両サイトで同一のテキストが作成できる．y_4はy_1の前に挿入された文字を，y_5はy_4とy_1の間に挿入された文字を表す．また同一サイトの文字間，例えば図1の時にy_4とy_1の間にはそのサイトの文字しか挿入できないので，y_4とy_1の間のフォーカスは1つしかない．更に，キャレット移動の際も，挿入と同様に新しいフォーカスを指定すると位置が限定できる．

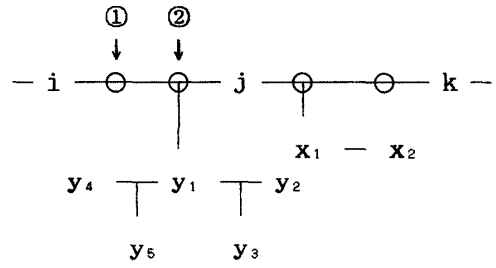
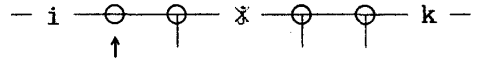


図1　共有テキストの概念図

3．　共有エディタの制御方法

テキストは逐次変化するが，異なったサイトの共有エディタにおいて編集されたテキストは，編集後一致しなければならない．そのために図1の概念図に基づき，相手が位置決めのために指定したフォーカスに従って自分のサイトにおいて位置づける．相手の指定したフォーカスが削除されている場合や，既に複数の文字を挿入した結果，相手フォーカスが移動させられている場合は，あらかじめ自分のサイトで保存した情報を参照しながら，概念図に基づいて処理する位置を決定する．

フォーカスとして与えられる文字は，基準文字と相手サイトで挿入された文字である．相手サイトで挿入された文字は，自分サイトで削除されることはないのでフォーカスとして使うことができる．問題となるのは基準文字がフォーカスとされたときである．この時は，自分のサイトで削除している場合があるので，保存した情報を参照して妥当な位置を決定する．また相手のサイトで基準文字が削除されている場合，例えば次の場合（相手サイトでjが削除されている）低位サイトに対してi−kの間の位置という指定がきた場合は，次図の↑の位置つまりiのすぐ後ろの位置に両サイトで統一する．



Implementational Algorithm of Edit Control for Shared Editor
Koji TSUZUKI, Yasushi IKEI, Yoshikuni OKAWA
Osaka University

［自分のサイトの編集入力に対する処理］

相手サイトで編集を実行した時と実質的に同一の場所で自分のサイトでも編集できるために，自分が編集操作を加えた履歴についての情報を保存しておく．以下にその情報の内容を示す．

<挿入>

挿入文字とその前後の文字を１組にして保存する．

INS buffer	a	b	c

a： 挿入文字の直前の文字
b： 挿入文字
c： 挿入文字の直後の文字

<削除>

削除文字とその前後の文字を１組にして保存する．

DEL buffer	d	e	f

d： 削除文字の直前の文字
e： 削除文字
f： 削除文字の直後の文字

［他サイトからの編集要求に対する処理］

現在既に変化している自分のテキストにおいて，相手が指示したフォーカスの実質的位置を決定する為に概念図を用いる．この概念図は両サイトにおいて共通に定義されているから，勿論両サイトで同一である．

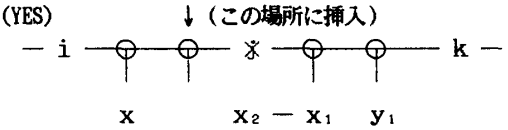
<挿入>

他サイトからの挿入の際には，挿入場所を指示するためのフォーカスと挿入文字が与えられる．実際の挿入位置は，既に述べたように高位サイトと低位サイトで区別する．

（低位サイトにおける挿入）

高位サイトから，（ｉｊ）をフォーカスとした挿入要求が到着した場合は，図１の②の位置に挿入する．この時はフォーカスの高位文字ｊを参照すればよい．テキストの状態とフォーカス文字の種類に応じて処理の方法が異なるので，以下にそれぞれの場合についての手順を示す．

テキストの文字列にｊがあるか?(step 1)
(YES)ｊの直前に挿入．
(NO) DEL buffer の{e}からｊを検索して，対応するfを取り出す．
INS buffer の{c}にfがあるか?(step 2)
(YES)



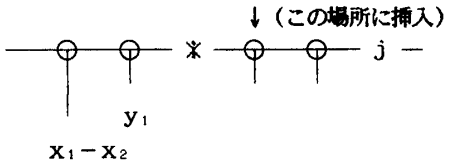
fに対応するbをツリーの最初の階層において検索し，その文字の直前に挿入．{c}にfが複数個ある場合は，その挿入文字がよりｊに近い文字の直前に挿入．
（この場合x_2の直前に挿入）

(NO) テキストの文字列にfがあるか?(step 3)
(YES) f の直前に挿入．
(NO) f を引数にして，(step 1)の(NO)の手順にジャンプ．

（高位サイトにおける挿入）

高位サイトから，（ｉｊ）をフォーカスとした挿入要求が到着した場合は，図１の①の位置に挿入する．この時はフォーカスの低位文字ｉを参照すればよい．具体的な手順を低位サイトの場合と同様に示す．

テキストの文字列にｉがあるか?(step 1)
(YES)ｉの直後に挿入．
(NO) DEL buffer の{e}からｉを検索して，対応するdを取り出す．
INS buffer の{a}にdがあるか?(step 2)
(YES)



dに対応するbをツリーの最初の階層において検索し，その文字の直後に挿入．{a}にdが複数個ある場合，その挿入文字がよりｉに近い文字の直後に挿入．
（この場合y_1の直後に挿入）

(NO) テキストの文字列にdがあるか?(step 3)
(YES) d の直後に挿入．
(NO) d を引数にして，(step 1)の(NO)の手順にジャンプ．

<フォーカス移動>

他サイトからのキャレット移動要求では，挿入の時と同様に位置を限定すればその新しい位置は一意に決まる．そこで挿入文字のかわりに相手のキャレットを出力すれば良い．

<削除>

他サイトから指定された文字を削除する．テキストの文字列を検索して，ない場合は既に削除されたことになるのでその時は何もしない．

４．おわりに

本稿では共有エディタを制御する際に，編集入力に対して３文字組の保存バッファを用いることにより，図１の概念図の文字列と等価な文字列を復元できるアルゴリズムについて述べた．このアルゴリズムにより，同時に編集したサイトのテキストのコンシステンシを保つことができる．実装上の問題となるバッファメモリのサイズは，サイト間の編集要求メッセージの伝達速度に応じて増減するが，両者のテキストが一致したときに基準文字を更新すれば，バッファサイズは有限とすることができる．今後の課題としては，入力が交互に到着する場合におけるバッファの更新制御や，編集単位が複数文字である場合の制御が挙げられる．

参考文献

１）池井，都築，大川，「共有エディタにおける編集制御とテキストの構造化」，情報処理学会第４１回全国大会講演論文集