

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4326599号
(P4326599)

(45) 発行日 平成21年9月9日(2009.9.9)

(24) 登録日 平成21年6月19日(2009.6.19)

(51) Int.Cl.

F I

G 0 6 F 17/21 (2006.01)

G 0 6 F 17/21 5 8 6 A

請求項の数 6 外国語出願 (全 30 頁)

(21) 出願番号	特願平8-181114	(73) 特許権者	500046438
(22) 出願日	平成8年6月6日(1996.6.6)		マイクロソフト コーポレーション
(65) 公開番号	特開平9-101949		アメリカ合衆国 ワシントン州 9805
(43) 公開日	平成9年4月15日(1997.4.15)		2-6399 レッドモンド ワン マイ
審査請求日	平成15年6月2日(2003.6.2)		クロソフト ウェイ
審査番号	不服2006-2058(P2006-2058/J1)	(74) 代理人	100147485
審査請求日	平成18年2月6日(2006.2.6)		弁理士 杉村 憲司
(31) 優先権主張番号	08/486407	(74) 代理人	100143568
(32) 優先日	平成7年6月7日(1995.6.7)		弁理士 英 貢
(33) 優先権主張国	米国 (US)	(74) 代理人	100072051
			弁理士 杉村 興作
		(74) 代理人	100114292
			弁理士 来間 清志

最終頁に続く

(54) 【発明の名称】 対話型テキスト修正及びユーザーガイダンスをサポートする方法及びシステム

(57) 【特許請求の範囲】

【請求項 1】

ワード処理のドキュメントにおけるユーザー入力を自動修正するユーザー入力自動修正方法であって、プログラムされたコンピュータ、ディスプレイ装置、及び入力装置を含むワード処理システムで実行されるユーザー入力自動修正方法において、

ユーザーがワード処理のドキュメントを作成中に前記入力装置から受信したユーザー入力イベントを傍受する傍受ステップであって、前記入力装置が、キーボード及びポインティングデバイスを有し、ユーザー入力イベントが、前記キーボードで文字を打ち込むこと及び前記ポインティングデバイスを用いて前記ディスプレイ装置に対する制御を行うことを含む、傍受ステップと、

前記ユーザー入力イベントが予め定められた自動修正イベントに対応するか否かを判別するために、前記ユーザー入力イベントを分析するユーザー入力イベント分析ステップと

、
該ユーザー入力イベント分析ステップから、1又は複数の文字からなるテキストトークンで表される当該予め定められた自動修正イベントに対応するユーザー入力イベントの検出に応じて、ユーザー入力を自動修正するための自動修正規則の評価をスケジューリングするために当該自動修正規則を決定する自動修正規則決定ステップであって、前記自動修正イベントは、所定の規則ベース内の自動修正規則にリンクされており、且つ前記自動修正規則は、前記規則ベース内の予め定められたイベントインターバルセクションに格納されている、自動修正規則決定ステップと、

前記自動修正規則の決定に応じて、評価キューに前記自動修正規則を置くことによって、評価のための前記自動修正規則をスケジューリングするステップと、

前記イベントインターバルセクションに対応するイベントインターバルの生起の検出に応じて、前記自動修正規則を評価する自動修正規則評価ステップとを含み、

前記自動修正規則評価ステップは、

前記自動修正規則の評価に応じて、自動修正エントリーによって規定されている1又は複数の文字とドキュメントに入力された文字列とを比較し、前記自動修正エントリーが合致する場合は、当該ドキュメント中に入力されている文字列を前記自動修正エントリーによって規定されている置換文字列で置換するステップを含むユーザー入力自動修正方法。

【請求項2】

自動修正エントリーが自動修正リストに記憶されており、

前記ユーザーが前記自動修正リストに前記自動修正エントリーを加えることができるように前記ディスプレイ装置上にインタフェースを表示するステップと、

前記ユーザーが新しい自動修正エントリーを入力するのに応じて、新しい自動修正エントリーに対応する新しい自動修正イベントを創出するステップと、
を更に含む請求項1に記載のユーザー入力自動修正方法。

【請求項3】

自動修正エントリーが自動修正リストに記憶されており、

ユーザーが既存の前記自動修正エントリーを前記自動修正リストから削除することができるように前記ディスプレイ装置上にインタフェースを表示するステップと、

ユーザーが前記既存の自動修正エントリーを削除するのに応じて、既存の自動修正イベントと前記自動修正規則との間のリンクを消去し、既存の自動修正イベントを検出するのに応じて前記自動修正規則が評価されないようにするステップと、
を更に含む請求項1に記載のユーザー入力自動修正方法。

【請求項4】

前記ユーザーが前記置換文字列を前記ドキュメントに入力される文字列に戻すか否かを監視するステップと、

その結果が肯定の場合、前記自動修正規則に対する例外を創出するステップと、
を更に含む請求項1に記載のユーザー入力自動修正方法。

【請求項5】

前記評価キューは、遅延動作キューである請求項1に記載のユーザー入力自動修正方法。

【請求項6】

前記ユーザー入力イベント分析ステップが、テーブル中でユーザー入力イベントを探索し、前記ユーザー入力イベントが前記自動修正イベントに対応するか否か決定することを
含む請求項1に記載のユーザー入力自動修正方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、一般的には、プログラムを用いる際にユーザーを支援するように設計された、コンピュータシステムにおける自動化された機能に関するものであり、更に特定するならば、自動テキスト修正及びユーザーガイダンス機能に関するものである。

【0002】

【従来の技術】

今日のワード処理ソフトウェアは、未経験のユーザーでさえも精巧なドキュメントを作成することができるようになっている。ワードプロセッサは、通常、ユーザーが、ドキュメントのスタイル、レイアウト、又は文字フォントを決めることによって、ドキュメントのフォーマットを定めることができるという多数の特徴を具えている。これらの特徴を用いると、ユーザーは、種々のレイアウト、フォント、スタイル等を持つレポート、手紙、メモ等の様々なドキュメントを作成することができる。

【0003】

その反面、新しいフォーマット設定機能の氾濫により、ワードプロセッサの利用を困難なものにしている。今や、ユーザーはドキュメントのフォーマットを定めるための非常に多くのオプションを持っているので、継続してそれらの情報を得ることが極めて困難になっている。ドキュメントのフォーマットを決めるためには、ユーザーはフォーマットを決めるためのコマンドを思い出すか、又はワードプロセッサに対するユーザーインタフェースの多数のオプションから選択しなければならない。最も親切なユーザーインタフェースであっても、ドキュメントのフォーマット設定は面倒で時間のかかることである。

【0004】

ユーザーがフォーマット設定機能を選択すると、ワードプロセッサの挙動が時には混乱した状態になる。例えば、ユーザーが図表オプションを選択すると、テキストを入力するための複雑さのレベルが上がる。ユーザーは、時には図表にテキストを入れるための新しいコマンドを学習しなければならない。これらのコマンドは、ワードプロセッサ毎に変わることがあり、平均的なユーザーにとっては直観的に認識できるものではない。

10

【0005】

殆どのユーザーにとって、ドキュメントを作成する最も容易な方法は、タイプライターに打ち込むように単にテキストを入力することである。平均的なユーザーはワードプロセッサを用いてドキュメントを作成するために、時に彼の／彼女のタイプライターのキーに対する馴れに頼っている。例えば、アウトラインを作る時、通常のユーザーは手動で数字及び文字を入力し、タブキーを用いて種々のセクションのインデントを行う。多くのユーザーは、いつでもアウトライン機能を選択することができアウトラインフォーマットを自動的に発生するワードプロセッサを持っている時でさえ、このような方法でアウトラインを作っている。このワードプロセッサの能力の利点を利用することに対する抵抗は、他の多くのフォーマット設定機能にも同様に当てはまる。このように、ワードプロセッサは、かなり職業的なドキュメントを作成する際にユーザーを助ける目的で設計されているが、彼らはそれでも通常のタイプライターを使うのと同様にテキストを入力することがずっと快適だと感じている。

20

【0006】

フォーマット設定機能は、大部分のユーザーがワードプロセッサで用いる方法を完全に揃えていないので十分に役立つものではない。ユーザーは、フォーマット設定機能の選択の方法を学習するために時間を取ることを望まないし、それらを使うことに満足することもない。従って、ドキュメントのフォーマット設定の負担を軽減するワードプロセッサが必要であり、同時に、更に広い範囲のフォーマット設定オプションを具えることが必要である。

30

【0007】

実際に存在するワードプロセッサは、更に、ユーザーが誤りのないドキュメントを作成することを助ける付加機能を含むことによって改良することができる。特に能力が劣るタイプストにとって役に立つ1つの機能は、打ち込みのエラーを自動的に修正することである。自動的な修正は、エラーを発見すること、及びユーザーに対して停止及び変更の要求をせずにそれらを正しいテキストに置き換えることに関する。マイクロソフト社のワード処理プログラムであるワード6は自動修正機能を持っており、更に効率的に動作するように、且つ種々の型のエラーに対して容易に拡張することができるように改良することができる。

40

【0008】

【発明が解決しようとする課題】

上述のように、ワードプロセッサに対して新しい機能を追加しても、その新しい機能の使い方をユーザーが迅速に且つ容易に学習できなければ、それを容易に利用するようにはならない。ユーザーに対してガイダンスを提供するための1つの方法は、「ヘルプ」テキストに対するリンクを作ることである。多くのプログラムが、ユーザー命令及びプログラムの機能についての記録と共に、ヘルプスクリーンに対するクイックアクセスを具えている

50

。このオンラインガイドは有用であるが、それでもユーザーがドキュメントの作成を止め、手動で所望の情報の走査を行うことが必要である。ワードプロセッサ及びその他のアプリケーションプログラムは、所望の機能の使い方に対する有用なガイド、助言、及び近道を自動的に提供するように設計するならば、ずっと使い易くなるだろう。

【0009】

【課題を解決するための手段】

現存するワード処理システムに係わる問題に近づくため、1つの実施例は、対話型自動フォーマット設定機能を具え、ワード処理を簡易化し改良している。対話型自動フォーマット設定機能は、ユーザーが打ち込むと同時にワードプロセッサが自動的にフォーマットを決定する。

10

【0010】

1つの実施例によれば、対話型自動フォーマット設定機能は、ユーザーが打ち込む時にユーザー入力をモニターし、多数の自動フォーマット設定規則に従って自動的にドキュメントのフォーマットを設定する。自動フォーマット設定機能は、ユーザーがキーボードによって文字を打ち込む時にそれらを読取る。次に文字を分析して予め定められたイベントとの同定を行う。

【0011】

予め定められたイベントは、自動フォーマット設定規則に対するアーギュメントである。1つのイベントが生起すると、次に関連する規則が評価される。規則が満足されると対話型自動フォーマット設定機能が自動フォーマット設定動作を開始し、ユーザーが打ち込む時にこれらの動作を自動的に実行する。

20

【0012】

このシステムにおけるフォーマット設定は、ユーザーがドキュメントを編集している間に行われるという意味で「対話型」である。ユーザーがフォーマットのコマンド又はバッチフォーマット設定機能を個別に呼出す必要はない。このように、ユーザーはフォーマット機能の呼出し方を記憶する必要はなく、一時中止して規定されたコマンドを選択する必要もない。ユーザーは、単にタイプライターに打ち込むように、テキストを入力する方法を知ることが必要であるだけであり、ユーザーが打ち込む時に、対話型フォーマット設定機能が自動的にドキュメントのフォーマットを設定する。

【0013】

本発明の他の実施例は、改良された自動的な修正（「自動修正」）機能を具える。ユーザーがドキュメントを作成する時、ワードプロセッサはユーザー入力イベントを傍受し、それら进行分析し、それらが予め定められたイベントに対応するか否かを決定する。自動修正規則が満足されるような予め定められた自動修正イベント又はイベントのパターンが生起すると、ワードプロセッサが自動修正動作を実行し、1つの文字列を他の文字列で置き換える。

30

【0014】

この実施例においては、イベントモニター、規則ベースのイベントエンジン及び規則ベースが、これらの自動修正機能を実行するために用いられる。イベントモニターが自動修正イベントを識別した場合は、それらは評価のためにスケジューリングされる。これらのイベントの評価の間に、他のイベント又は規則をスケジューリングすることができる。規則の評価の間に、イベントエンジンが、規則の条件が満足されているか否かを決定する。この自動修正規則に対する条件が満足されると、自動修正動作が実行され、ユーザーが打ち込む時に又はプログラムを用いる時にドキュメントが自動的に修正される。

40

【0015】

ワードプロセッサは、検出された時に自動的に置き換えられるべき文字列を特定する自動修正入力のリストを保持する。ワードプロセッサは、新しい追加及び存在する入力の削除の両者を含むこのリストの変更をサポートする。自動修正リストに加えて、ワードプロセッサは、或る種の自動修正機能に対する例外のリストを保持し、これも変更することができる。ワードプロセッサは、自動修正規則に対する例外を検出し、これらの例外を自動的に

50

に自動修正リストに加える機能をサポートする。

【0016】

他の変形実施例は、予め定められたイベントの検出にตอบสนองして、ユーザーに対する「助言」と呼ばれる自動ガイダンスを具える。例えば、予め定められたイベントのパターンが、ユーザーがまごつき、助けを必要としていることを表示することができる。他の例としては、或るイベントが、プログラムが行っていることに対するユーザーの理解を助けるため、又は1つの動作を受入れるか又は拒否するかを選択をユーザーが行えるようにするための動作に対応する助言を含む動作を起動することができる。

【0017】

1つの実施例においては、助言機能は更にイベントモニター、規則ベースのイベントエンジン及び規則ベースと共にサポートされる。イベントモニターが規則ベースの予め定められたイベントを検出した場合は、評価のためにそれらをスケジューリングする。イベントエンジンは、イベントを評価し、続いてスケジューリングされるべき他のイベント又は規則をスケジューリングすることができる。「助言」動作に対応する或る型の規則が評価され、これらの条件が満足される場合は、「助言」動作が自動的に遂行される。助言動作は、一般的に助けになるテキストのボックスの表示を含む。しかし、更に、ユーザーインターフェース制御を含むことができ、これによりユーザーが、「実行」一動作を続ける一又は「実行せず」一プログラムが自動的に実行した動作を無効にする一等のオプションを選択することができるようになる。

【0018】

本発明のその他の利点及び特徴は、以下の詳細な説明及び添付の図面によって明らかになる。

【0019】

【発明の実施の形態】

先ず、本発明の1つの実施例に関する方法及びシステムが実現されるコンピュータシステムの各要素についての説明を行う。続いて、本発明の1つの実施例に関する対話型自動フォーマット設定機能の構成の概観を説明する。次に、この実施例の動作及びその実現方法について付加的な詳細の説明を行う。

【0020】

図1は、本発明の方法及びシステムの実施例を実現するために用いられるコンピュータシステム20のブロック図である。コンピュータシステム20は、基本的な要素としてコンピュータ22、入力デバイス24及び出力デバイス26を含む。

【0021】

コンピュータ22は通常、バス構成32によって通信する中央処理ユニット（CPU）28及びメモリーシステム30を含む。CPU28は、計算を実行するための算術的論理ユニット（ALU）33、データ及び命令を一時格納するためのレジスタ34、及び、アプリケーション又はオペレーティングシステムのようなコンピュータプログラムからの指示にตอบสนองしてコンピュータシステム20の動作を制御するための制御ユニット36を含む。

【0022】

メモリーシステム30は通常、ランダムアクセスメモリー（RAM）及びリードオンリーメモリー（ROM）半導体デバイスのような媒体の形の高速主メモリー38、及び、フロッピーディスク、ハードディスク、テープ、CD-ROM等のような媒体の形、及び光学的、磁氣的、又は記録材料を用いる他のデバイスの形の2次記憶装置40を含む。主メモリー38は、コンピュータのオペレーティングシステム及び現在走っているアプリケーションプログラムのようなプログラムを格納する。主メモリー38は、更に表示装置を通して画像を表示するためのビデオ表示メモリーを含む。

【0023】

入力デバイス24及び出力デバイス26は、通常、バス構成32によってコンピュータ22に接続される周辺デバイスである。入力デバイス24は、キーボード、モデム、ポインティングデバイス、ペン、その他の、入力データをコンピュータに供給するデバイスである。出力デ

10

20

30

40

50

バイス26は、ディスプレイ装置、モデム、プリンタ、音響装置、その他の、コンピュータからの出力データを供給するデバイスである。

【0024】

図1は、コンピュータシステムの基本的な要素を説明するためのブロック図であることを理解すべきである。図1は、コンピュータシステム20の特別なアーキテクチャを説明することを意図したものではない。例えば、コンピュータシステムの各要素を相互接続するために、コンピュータ設計の分野で既知の種々のバス構成が必要に応じて多くの方法で用いられているので、特定のバス構成は図示されていない。CPU28は、個別のALU33、レジスタ34及び制御ユニット36から構成されてもよいし、マイクロプロセッサのようにCPUの1又は複数のこれらの部分が統合されている単独のデバイスであってもよい。更に、コンピュータシステムの要素の数及び配置は、この分野で既知の方法で、図示され説明されたものから変更してもよい。

10

【0025】

本発明は、多数の既知のコンピュータシステムのいずれによっても実行することができる。例えば、本発明は、例えばIBM-AT互換のパーソナルコンピュータ(PC)、又はインテル社製の80386、80486、又はペンチウムプロセッサに基づくコンピュータシステムで実行することができる。他の例として、本発明は、アップル社製のアップルマッキントッシュコンピュータで実行することができる。マッキントッシュコンピュータシステムは、モトローラ社製のMC68000ファミリーのプロセッサに基づいている。これに代えて、本発明は、RISC(reduced instruction set computing)アーキテクチャに基づくマシンのような、幾つかのコンピュータワークステーションで実行することができる。上述のシステムは、例示としてのみ意味があり、本発明が実行されるコンピュータシステムの型を限定するものと解釈されてはならない。

20

【0026】

コンピュータシステムのメモリーにロードされたオペレーティングシステムは、本発明の実施例の動作をサポートするための多数の低レベル機能具える。一般的に、オペレーティングシステムはメモリー、CPUタイム、ディスク空間、及び周辺デバイスのようなハードウェアリソースの割当て及び使用を制御する。よく知られているように、オペレーティングシステムは、モジュール(プロセス及びダイナミックリンクライブラリ)管理、スケジューリング、プロセス間メッセージング、メモリー管理、ファイルシステム管理、及びグラフィックユーザーインターフェースサポートのような低レベル機能具えている。

30

【0027】

多数の既知のオペレーティングシステムが、上記に挙げたコンピュータに利用できる。例えば、マイクロソフトDOS及びウィンドウズオペレーティングシステムは、インテル社製のプロセッサのX86ファミリーに基づくコンピュータシステムに広く用いられている。他の例としては、ウィンドウズNTオペレーティングシステムは、インテル社製のプロセッサのX86ファミリーに基づくコンピュータシステムばかりでなく、RISCマシンにも用いることができる。更に他の例としては、UNIXオペレーティングシステム及びその変形版は、種々のコンピュータシステムに広く用いられている。

【0028】

本発明は、種々のオペレーティングシステムを用いて実行することができるが、以下に説明する実施例は、マイクロソフト社のウィンドウズオペレーティングシステムで動作するコンピュータシステムで実行される。この実施例は、マイクロソフト社の「ウィンドウズアプリケーションプログラムインターフェース」(Win 32 API)から図形上における窓開け環境についての利点が得られる。ウィンドウズAPIの解説は、マイクロソフト社の「ウィンドウズ95ソフトウェアデベロッパーズキット」(SDK)及びマイクロソフトプレス社発行の「マイクロソフトウィン32プログラマーズリファレンス」のような種々の解説書から得られる。

40

【0029】

実施例の実行に用いられるハードウェア及びオペレーティングシステムについて説明した

50

ので、次に実施例のアーキテクチャに進む。図2は対話型自動フォーマット設定システムのアーキテクチャを示す。このシステムは、3つの基本的な成分、即ち、イベントモニター50、規則ベースイベントエンジン52、及び規則ベース58について説明することができる。このシステムをこの一般的な構成に基づいて説明すると本発明についての明快な理解が得られる。当業者は、この構成が本発明の範囲を逸脱せずに変更することができる事が分かる筈である。

【0030】

図2の左側を見ると、イベントモニター50は、ユーザーが打ち込むか又は他の入力を行う時に入力をモニターし、自動的にドキュメント56のフォーマット設定を行う機能を起動するためのシステムの一部であることが分かる。イベントモニター50はユーザー入力を捕捉し、この入力からイベントを生成する。イベントモニター50は、次にこれらのイベントを規則ベースイベントエンジン52に送る。

10

【0031】

イベントモニターがイベントエンジン52にイベントを送ることにより、イベント及び規則に対する評価のためのスケジューリングが行われる。イベントモニター50は規則ベースイベントエンジン52を呼出し、規則ベース58のインターバルセクションでスケジューリングされた規則及びイベントを評価する。この評価の結果として、イベントエンジン52は、自動フォーマット設定規則を遂行するために生成されるべき遅延動作を生起することができる。いずれかの自動フォーマット設定規則が用いられると、イベントモニター50は、適当な機能に対して呼出しを行う判断を行い、これらの遅延動作を遂行する。

20

【0032】

説明を続ける前に、この実行例における規則に関する用語を明確にしておくことが重要である。「擬似コード規則」は、人間の言語での自動フォーマット設定動作のための条件を記述する高レベル規則に関する。コンピュータでこれらの擬似コード規則を実行するため、これらはCのような高水準ソースコードで書かれ、規則ソース60に置かれる。規則ソース60中の規則は、次にコンパイルされて規則ベース58を形成する。規則ソース60中の規則は、規則ベース58中の規則と1対1の対応を持っている。従って、一般的な用語「規則」は、規則ソース60及び規則ベース58中の規則の両者に用いられる。規則ベース58の構造は以下に詳細に説明される。

【0033】

再び図2を参照すると、規則ベースイベントエンジン（イベントエンジン）52は、規則ベース58中の規則に従ってユーザー入力を評価するシステムの一部であることが分かる。

30

【0034】

イベントエンジン52はイベントを処理するための2つのルーチン、即ち、スケジューリングルーチン及び評価ルーチンを含む。イベントモニター50がイベントをイベントエンジン52に送ると、イベントエンジンは、そのイベントに従属する規則及び他のイベントを決定し、評価のためにそれらのスケジューリングを行う。イベントモニターが評価ルーチンを呼出すと、イベントエンジンは、これらのスケジューリングされた規則及びイベントを評価する。規則の評価がスケジューリングされるべきイベント及び規則の追加の原因になることがあるため、スケジューリング及び評価のステップは相互に関係付けられる。例えば、規則が満足されると、それにより、それに従属する規則が、直接又は新しいイベントを通しての評価のためにスケジューリングされることになる。

40

【0035】

規則の評価が、他の規則又はイベントをスケジューリングする原因になることがあり得るが、一方で、スケジューリング及び評価ルーチンは分離されている。規則及びイベントは、上述のようにイベントモニター50により、又は評価の間はイベントエンジン52により、スケジューリングされ得る。規則ベース58は、以下に定義される複数のインターバルセクションに分割される。一般的には、規則は、イベントモニターが規則を評価するために評価ルーチンを呼出した時に評価され、イベントは、規則ベースの特定のインターバルセクションでスケジューリングされる。評価の間、評価されたイベント又は規則が追加のイベ

50

ント又は規則をスケジューリングする時は、評価ルーチンがスケジューリングルーチンを呼出す。インターバルセクションにおける規則及びイベントのスケジューリング及び評価の処理は、そのインターバルセクション中の評価のためにスケジューリングされた規則がなくなるまで続く。

【0036】

イベントモニターは、「イベントインターバル」の識別に応答して「インターバルセクション」のための評価ルーチンを呼出す。

【0037】

規則ベースは、それぞれがイベントの型に対応する複数のインターバルセクションに分割される。1つのインターバルセクションが、イベントの共通の型を処理するために専用になっている。ユーザー入力から、又は他の規則又はイベントの評価から、1つのイベントが生成されると、それがそのインターバルセクションでスケジューリングされる。

【0038】

「イベントインターバル」は、特定のインターバルセクションについてスケジューリングされた規則又はイベントを評価するために評価ルーチンが呼出される時期を決定する。イベントインターバルは、従ってインターバルセクションに対応する。

【0039】

イベントインターバルの概念を明らかにするためには、ユーザー入力に関連付けて説明するのが有効である。イベントモニターは各イベント型の状態の変化を探索している。それぞれインターバルセクションに対応して種々のイベント型があるため、イベントモニターは、各イベント型について状態の変化を監視する。イベントモニターが特定のイベント型について状態の変化（即ちイベントインターバル）を識別すると、それは対応するインターバルセクションの評価を開始する。

【0040】

イベントインターバルは時間に類似する。秒針の正時が新しい秒インターバルを表す。次の分への分針の通過が新しい分インターバルの到来を表す。次の時間への時針の通過が他の時間インターバルの到来を表す。このシステムにおいては、イベントインターバルは時間とは異なる量の状態の変化を表すが、概念は同一である。ここでは、イベントインターバルは、特定のイベント型についての状態の変化を表す。この状態の変化が生起すると、イベントモニターが、対応するイベントセクションについての評価ルーチンを呼出す。

【0041】

この実行例においては、数個のイベントインターバルセクションがある。ユーザーインタフェース、選択、対話、文字、トークン、ライン、文節、オブジェクト及びドキュメントイベントを含む多種のイベントについてのインターバルセクションがある。選択インターバルセクションは、「選択」イベント（即ち、ユーザーがテキストを選択するか又はカーソルを動かす）を追跡する。対話インターバルセクションは、対話ボックス又はメニューの中のユーザー入力イベントを追跡する。ユーザーインタフェースインターバルセクションは、対話インターバルセクションより高いレベルのイベントを追跡し、押しボタン制御の始動、対話の呼出し等のようなイベントを含む。オブジェクトインターバルセクションは、複数文節要素、テーブル等についてのイベントを追跡する。ドキュメントインターバルセクションは、ドキュメントが変わるまで有効なイベントを追跡する。例えばヘッダーが個別のドキュメントと考えられ、従って、ユーザーがヘッダーを開くと、これがドキュメントイベントインターバルセクション中のドキュメントイベントになる。

【0042】

トークンインターバルセクションは、デリミッタ及びワードのような論理ユニットを形成する或る文字又は文字列の追跡を続ける。

【0043】

インターバルセクションの数は変化し得る。多種のユーザー入力の追跡を続けるため、イベントモニターが監視するイベントの数は増加することがあり、これらのイベントを分類するためのイベントインターバルセクションの数もやはり増加し得る。自動フォーマット

10

20

30

40

50

設定規則については、関連する基本的なインターバルセクションは、文字、トークン、ライン、文節、オブジェクト及びドキュメントである。イベントモニターは、主として自動フォーマット設定規則に妨害を与えるイベントが起きないことを確認するために、更に他の型のイベントを監視する。規則がワード及び文字の或るシーケンスに従属する場合は、その規則に、そのワードと文字との間に他のユーザー入力イベントが起きないという条件を含めることが重要であり得る。例えば、ユーザーがワードを打ち込み、ユーザーインタフェース又は対話インターバルセクションで他のイベントが起き、続いて最終的に文字の打ち込みに戻ると仮定しよう。この例においては、ユーザーのみがワードを及び続いて文字を打ち込むという望ましい条件が、もはや満足されない。要約すると、この規則はイベントの生起及び他のイベントの欠如に従う。

10

【0044】

イベントをこれらのイベントインターバルセクションに分離することにより、基礎であるイベントエンジンが、規則が評価される時期を制御することができる。他のインターバルセクション中のイベント及び規則が、現在のインターバルセクション中のイベント及び規則によってスケジューリングされ得る。これは、それが或る型のユーザー入力に基づく規則の評価を遅らせる1つの方法であることから、ワード処理に関しては明らかである。インターバルセクションの利用により、ユーザーが行うことについて規則が評価されるべき時期を、規則ベースが表すことが可能になる。例えば、ユーザーは、トークンインターバルセクションについてのイベントインターバルを生成する文字列の打ち込みを行うことができる。このイベントが、トークンインターバルセクション中の規則ベースを通して伝播する場合は、イベントエンジンは、文節インターバルセクションのような一層高いインターバルセクションで、規則をスケジューリングすることができる。この可能性は、規則は、単にスケジューリングされた場合、及び規則ベースにおいて規則が存在するインターバルセクションについてイベントインターバルが生起する場合にのみ評価されることが必要であるに過ぎないので、著しい効率化を生み出すことができる。このように、規則ベースの設計が、規則が満足されるか否かを決定するために必要な処理の量を減らす。他のイベントインターバル又はインターバルセクション中の規則をスケジューリングする能力は、極めて長時間後に起こり得るユーザー動作を予想する能力を規則に与えることができる。

20

【0045】

次に、図2の右側に移ると、規則ベース58が規則を実行するためのシステムの一部になっている。規則ソース60は、Cのような高水準言語で書かれ、次に規則コンパイラ62を用いて規則ベース58にコンパイルされた規則である。

30

【0046】

規則は、追加のイベント又は規則のスケジューリング又はコードの実行のような、動作がとられる前に生起する筈の多数のイベントに基づいて表現される。これらのイベントは、変数に基づいて表現され、規則の条件は、これらの変数をアークギュメントとして用いて論理的に表現される。例えば、規則は、論理的なコネクタを用いて、生起する筈の一連のイベントにおける部分を特定することができる。規則は、更に、「and」、「not」及び「or」、更に、ドキュメント又はワードプロセッサ内の全体的な状態のチェック、又は機能の呼出しを含む論理演算子を用いて多数の条件を定めることができる。

40

【0047】

高レベルの自動フォーマット設定機能の1つの例は、ユーザーが箇条書きで作成するリストをシステムが予測し、自動的にドキュメントの形式を整えることである。フォーマット機能と呼出す方法が分からない場合は、通常、ユーザーはアステリクスを使う。例えば、文節の最初の文字として、ユーザーは「*」を、続いてスペースを、続いてワードを打ち込むことによってリストを作成することができる。これらの文字に対する自動フォーマット設定のための擬似コード規則は次のようになる。即ち、ラインが「*」で始まり、スペースが続き、ワードが続く場合は、アステリクスを黒丸に変換し自動箇条書きモードを呼出す。ソースコードでは、このアブストラクト規則は一連の順序規則に基づいて記述される。

50

【0048】

規則コンパイラ62でコンパイルされると、規則はノードのネットワークに格納される。各イベント及び規則はノードによって規則ベース中に表示される。図3はこのネットワークの構成の一例を示す。このネットワークは、複数のインターバルセクション70a-eに配置された多数のノードを含む。各インターバルセクションはイベントインターバルに対応する。同様に、各イベントインターバルセクションは、1つのインターバルセクションの中におけるイベントと規則との内部従属性を反映する多数のレベルを持つことができる。1つのレベルのノードは、次に高いレベルのノード又は上の複数のレベルのノードとの結合を持つことができる。更に、1つのインターバルセクションのノードと高いか又は異なるインターバルセクションのノードとの間の結合も存在し得る。

10

【0049】

図3は、規則ベース中のノード間の関係を説明する図である。各ノードは多くのイベントに対して従属性を持つことができる。ノードは同一のイベント又は規則に従属するので、ノードの多くは相互従属性である。イベントと規則との間の従属性を表すために、規則コンパイラ62は各規則及びイベントをノードに変換し、ノードをレベルに分類する。規則ノードの1つは、「イベント1が生起したならば、イベント2に続け」であり得る。他の規則又はイベントに従属する規則は、それらが従属する規則又はイベントより高いか又はこれと異なるレベルに存在する。1つのセクションの中では、他のノードに従属するノードは、それらが従属するノードより高いレベルに存在する。

【0050】

ノード間の結合はノード間の従属性を表す。結合は、同一のインターバルにおける直接の結合、イベントを介しての間接の結合、又は他のインターバルにおける評価に対するスケジューリングによる遅延結合が可能である。規則の場合、結合は、規則とそれに従属するノードとの間の関係を表す。直接規則の一例は、「イベント1が生起したならば、イベント2をスケジューリングせよ」である。2つの間接相互従属規則の例は、「イベント1が生起したならば、イベント2を真に設定せよ」及び「イベント2が生起したならば、イベント3を真に設定せよ」である。第1の規則がイベント2の値を変えられるので、第2の規則は第1の規則に従属する。遅延規則の一例は、「イベント1が生起したならば、1つのインターバルの遅延の後でイベント2をスケジューリングせよ」である。

20

【0051】

図3に示されたネットワーク構成は、規則がスケジューリングされ、イベントエンジン52で評価される方法を説明するのに役立つ。上述のように、イベントが生起した時に、このイベントを契機として、ノードのスケジューリング及び評価の対話処理が開始される。図3においては、イベントインターバルは、ネットワークのインターバルセクション70a-eに対応する。1つのインターバルセクションに対して1つのイベントが生起すると、イベントエンジン52が、関連するノードをスケジューリングし、そのインターバルセクションの第1レベルで始まる時間にそのノードを1つのレベルに評価する。

30

【0052】

イベントエンジン52は、1つのレベルに対してスケジューリングされた全てのノードの評価が終了するまで、繰り返しノードを評価しスケジューリングを行う。次に、イベントエンジンは次のレベルに移動し、そのレベルでスケジューリングされたノードを評価する。イベントエンジンは、そのインターバルセクション中の全てのレベルに対してこの処理を続け、やがて停止する。この処理の間、他のインターバルセクションのノードは、図3の異なるインターバルセクションのノード間の結合によって表されるようにスケジューリングされ得る。しかしながら、これらのノードは、イベントモニターがこのインターバルセクションのための評価ルーチンと呼出すまでは評価されない。

40

【0053】

このネットワークに関しては、1次規則、2次規則、順序規則、及び動作規則を含む規則の多数の異なる型が存在する。1次規則は他の規則に従属しない規則である。1次規則が従属するイベントが値を変える時には、イベントエンジン52は自動的に1次規則をスケジ

50

ューリングする。2次規則は他の規則に従属する。2次規則は明確に他の規則によってスケジューリングされる。言い換えれば、規則の評価によって、2次規則のスケジューリングが行われる。

【0054】

順序規則は2次規則の1つの型であり、自動フォーマット設定動作がイベントの順序に従属することを可能にする。順序規則は、後のイベントが検出され得る場合に、それが後のイベントに対する合成規則の部分の評価を遅延させるので、イベントエンジンが一連のイベントを検出することを可能にする。順序規則は、特定の数のイベントインターバルの変化が生起するまではそれが評価されないような遅延を持ってスケジューリングされる。例えば、一連のイベントに基づいて、1つのトークンインターバルの中で1つの規則が評価される場合は、この規則は順序規則をスケジューリングし、次のトークンインターバルで評価することができる。2つの規則は、連続的にパターンを検出するための能力との合成規則のように振舞う。このようにして合成規則を構成することによって、2つのトークンイベントが連続的に生起することを、イベントエンジンが検出できる。

10

【0055】

順序規則は、規則ベース中の規則の評価を遅らせるための1つの方法である。他の方法としては、概略を上述したように、規則が他の規則をネットワーク中の高い又は異なるインターバルセクション70b-eにスケジューリングさせることである。両者の場合共、イベントエンジンは異なるイベントインターバル中で規則を評価することにより、評価を遅らせる。

20

【0056】

動作規則は遅延した動作を発生することができる規則である。動作規則が満足されると、続いてそれが遅延した動作を遅延動作キューに置く。この遅延した動作は1つの動作であり得るし、実行されるべき多数の動作を含む複合動作でもあり得る。例えば、箇条書きリストを作成する場合、リストを作成するためのドキュメントの編集に係わる多数の動作、及びユーザーにヘルプテキストを表示するようなその他の動作があり得る。これらの遅延した動作を開始するために、イベントモニターがそれらをキューから1度に1つずつ削除し、適当な機能と呼出し、それらを実行する。

【0057】

イベントモニター50、イベントエンジン52、及び規則ベース58について上述したが、次にこれらの成分の実行例について更に詳細に説明する。

30

【0058】

図2に示すように、イベントモニター50は、ユーザー入力を捕捉するイベントフック80、ユーザー入力から種々のインターバルセクションに対するイベントを発生する複数のイベント発生器82、及び「トークン」に基づいてイベントを識別するレクサー84を含む多数の機能要素を含む。これらの要素に加えて、イベントモニターは、ラインフォーマットバッファからの一連のフォーマット設定されたテキストをアクセスする次のランの取得機能86を含む。レクサーはこのテキストから「テキストトークン」を識別し、これらをテキストトークンキャッシュ88に格納する能力を持つ。フォーマットレクサー90は、一連のテキストから「フォーマットトークン」を識別し、これらを一時的にフォーマットトークンキャッシュ92に格納する。遅延動作インタプリタ94は、機能95を開始して、動作規則の評価の間にイベントエンジンで発生された遅延動作に基づいてドキュメントのフォーマット設定を行う。これらの機能要素の各々について、以下に更に詳細に説明する。

40

【0059】

フック80は、ワードプロセッサ内の手順からイベント発生器に対する呼出しであり、これは、例えばキーボード上のキーの押下によりイベント発生器に送られるユーザー入力を傍受するためのものである。このイベントフックは、必要に応じて、規則で特定される新しいイベントをサポートするために加えられる。イベントは必要に基づいているため、イベントフックの数及び型は変化し得る。ワードプロセッサプログラムは、種々のユーザー入力イベントを捕捉するため、コードの種々のセクションに位置する多数のフックを含む。

50

フックの一例は、システム中でキーの押下が受信された時にこれを傍受するために用いられる文字イベントフックである。ワードプロセッサは、システムのユーザーインタフェースに対する他の型の入力を傍受するための他のフックを含むことができる。例えば、この実施例は更に、テキスト選択、ボタンを押すマウスクリックのようなユーザーインタフェースイベント、及び対話ボックス内部のユーザー入力からの対話型イベントを検出するためのフックを含む。

【0060】

イベントフックは、ワードプロセッサの動作状態及び傍受されるべき入力に応じて、ワードプロセッサの中の種々の位置に位置することができる。この実施例においては、ユーザーインタフェースイベントに対するフックは、「コマンド送出」と呼ばれる、コマンドを送出するためのルーチンを呼出すコードセクションに位置する。コマンド送出は更にキーボード又はマウスでのユーザー入力に直接対応しないマクロを処理するので、フックは、ユーザー入力をマクロから分離できる更に高次のコードレイヤに位置される。マクロは、本質的に、1又は複数のユーザー入力動作に応答して実行される多数のコマンドを潜在的に含むスクリプトである。従って、スクリプトに含まれるコマンドは、直接にユーザー入力動作にマップすることはない。規則ベース中の規則はユーザー入力の或る型に関連するので、フックは、所望のユーザー入力イベントを検出できるコードセクションに位置しなければならない。しかしながら、フックの位置及び動作は変化し得る。

【0061】

ユーザーインタフェースイベントに対するフックのように、対話型イベントに対するフックは、特に或る型のユーザー入力イベントを傍受するように設計される。この実施例においては、対話ボックスにおけるユーザー入力イベントを傍受するためには、1) 対話ボックスの「以前」の状態と「以後」の状態とを比較することによる方法、及び2) ワードプロセッサ中の全ての対話ボックスを処理するための中央集中手順にフックを加えることによる方法の2つの方法がある。第1の方法においては、イベントフックがイベント発生器を呼出し、対話の初期化の間に対話の状態を捕捉し、続いて第2イベントフックがイベント発生器を呼出し、イベント発生器がこの状態に対話が終了した時の対話の状態と比較する。対話ボックスに対する最終のユーザーの変化に従属するのみのイベントが、第1の方法における主流である。第2の方法においては、対話ボックスを処理するためのコード内のイベントフックが、例えばチェックボックスのチェック又は未チェック、押しボタンのクリック等のようなユーザー入力があった時にそれを傍受し、そのイベント発生器を呼出してイベントを発生する。規則が対話ボックス内の特定の連続のユーザー入力イベントを知る必要がある場合には、これらのイベントが追加されなければならない。

【0062】

イベントを選択するためのフックは同様に振舞う。このフックは、ユーザーがテキストを選択する時及び選択しない時にこれを検出するように設計される。

【0063】

各イベント発生器82は、ユーザー入力からイベントを発生し、これらのイベントをイベントエンジン52に送る役割を持つ。一般的に、複数のイベント発生器は、個別のルーチンのように実行することができ、又は、1つのルーチンに結合することができる。しかしながら、概念的には、複数の特定のイベント発生器は1つのイベント発生器のように見ることができる。この実施例においては、イベント発生器82は、ユーザーインタフェース、対話、文字、選択、及びトークンイベントに対するイベント発生器を含む。

【0064】

図4は、イベント発生器の概要を示すフロー図である。イベント発生器は、各イベントフックから傍受したイベントに応答して動作を開始する。イベント発生器の最初のステップ100は、関連するイベントを識別しそれらをスケジューリングすることである。イベント発生器は、フックから得られたイベントから規則ベースに関連するイベントを振り分けする。一般的に、イベント発生器は、規則ベースに関連するイベントをルックアップするためのテーブルを用いる。しかしながら、或る種のイベントは、後に説明するように、ルッ

クアップテーブル以外のものも含む。この実施例はテーブルを用いて関連するイベントを識別するが、他の方法を用いることもできる。

【0065】

一般的に、このテーブルはユーザー入力イベントを規則ベースに關係する1又は複数のイベントにマップすることができる。テーブル中の各イベントは、インデックスによって規則ベースに關係付けられ、イベントのスケジューリングをサポートする。1つのイベントフックからのイベントが関連するイベントを発生しないことも多数の関連するイベントを発生することもあり得るので、最初のステップ100は、ゼロか又は複数イベントのスケジューリングに関する。

【0066】

次のステップ102においては、イベント発生器がイベントエンジンの評価ルーチンを呼出し、1つのインターバルセクションについて規則及びイベントを評価する。例えば、ユーザーインタフェースイベント発生器は、ユーザーインタフェースイベントインターバルに応答して、ユーザーインタフェースインターバルセクションについての評価ルーチンを呼出し、文字イベント発生器は、文字イベントインターバルに応答して、文字インターバルセクションについての評価ルーチンを呼出し、その他も同様である。イベントの数のトラックを保持するため、イベントエンジンは、イベント発生器がそれぞれのインターバルセクションについての評価ルーチンを呼出す度に、毎回1つのインターバルについてカウンタを進捗させる。1つのインターバルセクションについての評価において、規則がこの他のインターバルセクションへの遷移を開始させる場合は、評価ルーチンは更に他のインターバルセクションを評価することができる。例えば、これは、トークン発生器が、ライン、文節、オブジェクト、及びドキュメントインターバルについての評価ルーチンを開始させる方法である。

【0067】

判断ステップ104は、イベント発生器が、遅延動作が評価ステップ102の後で実行される必要があるか否かの決定を行うことを示す。肯定の場合は、イベント発生器が遅延動作インタプリタ94を呼出し、通常ステップ106で表されるように遅延動作を実行する。この実施例においては、各イベント発生器は通常図4のステップに従って実行する。しかしながら、文字及びトークンイベント発生器についての以下の説明から明らかになるように、これと異なる特別な設計も可能である。

【0068】

図5は、文字及びトークンイベント発生器によって実行されるステップの概要を示す。この実施例においては、文字及びトークンイベント発生器が1つのルーチンで実行され効率を最高にする。文字発生器及びトークンイベント発生器の両者共に、ユーザーの1つの文字の打ち込みに応答してイベントを発生することができる。

【0069】

ユーザーが文字を打ち込む時、文字イベントフックが文字イベント発生器を呼出す際に処理が始まる。最初のステップ110においては、文字イベント発生器が文字フック80から文字を受信する。この文字をルックアップテーブル中で探すことにより、そこにある場合は、文字イベント発生器が文字フックから送られて来た各文字について文字イベントを識別する。

【0070】

次のステップ112においては、文字イベント発生器は文字イベントをスケジューリングする。文字イベントのスケジューリングにおいては、文字イベント発生器がイベントエンジン中のスケジューリングルーチンを呼出し、文字イベントについてのインデックスを文字インターバルセクションの規則ベースに送る。文字イベント発生器が文字イベントをモニターし、ドキュメント中のパターンを監視する。例えば、順序規則は、次に空白、その次に文字数字が続くワードに従属する。この例においては、合成順序規則は、空白文字及び文字数字についてのワード及び文字イベントに対するトークンイベントに基づいて書かれる。

10

20

30

40

50

【0071】

文字イベント発生器は、ステップ114 で示すように、次に文字インターバルについての評価ルーチンを呼出す。評価の間に何らかの遅延動作が生起されると、これらは遅延動作キューに置かれる。文字イベント発生器、キューをチェックして実行すべき遅延動作があるか否かを決定する（ステップ116）。肯定なら文字イベント発生器が遅延動作インタプリタ94を呼出し、ステップ118 で表されるように、遅延動作を実行する。

【0072】

ステップ110 から118 までは文字イベント発生器の挙動を要約している。打ち込まれたそれぞれの文字は潜在的にトークンを形成する。このように、文字イベント発生器がいずれかの文字イベントを一度処理すると、トークンイベント発生器がトークン関連イベントを処理する。ステップ120 -132 はトークンイベント発生器に関係する。

10

【0073】

トークンイベント発生器は、トークンに基づいてイベントを発生するためのものである。前述の説明のように、規則ベース58の規則はイベント及び他の規則の両者に従属する。このトークンは、規則ベース58に存在するイベントに、文字からなるユーザー入力と文字列との間の関係を確立する方法を提供する。トークンイベント発生器はレクサーを呼出し、「トークン」イベントを識別する。レクサー84が「トークン」を識別する時は、それは規則ベースに影響する或る型のイベントが生起したことを既に決定している。

【0074】

この実行例には、テキストトークン及びフォーマットトークンの2つの型のトークンがある。テキストトークンは、文字又は文字列を表す。これらのテキストトークンは、これらがワード又は区切り記号全体を表示するので、規則ベースにとって重要である。各々が規則ベースの対応するイベントと1対1の関係を有する。図3は、トークンインターバルセクション70a で、テキストトークンキャッシュから出ている一連の矢印によってノードにリンクされたテキストトークンを表示することにより、テキストトークンがコンパイルされた規則ベースに関係する様子を示す。イベント発生器82がイベントエンジン52に対してテキストトークンを送ると、それはどのイベントが生起したかを知らせることとなり、これにより、関係する規則又はイベントがスケジューリングされ評価される。

20

【0075】

フォーマットトークンは、テキストに適用されるフォーマットの型に対応し、複数のテキストトークンに亘ることができる。図3は更に、フォーマットトークンがコンパイルされた規則ベース58にどのように関係しているかを示している。テキストトークンと同様に、フォーマットトークンは、規則ベースイベントノードに対応する。フォーマットトークンは、フォーマットトークンキャッシュ92から出ている点線によって示されるように、トークンインターバルセクション、更にラインインターバル及び文節インターバルセクションでイベントに対応することができる。

30

【0076】

イベントエンジンに到達したそれぞれのトークンは、インデックス、文字位置データ、及び値と結合する。トークンのインデックスは規則ベース58内で1つのノードを特定する。イベント発生器がイベントエンジン52にトークンを送ると、それが規則ベース中のイベントノードに対してインデックスを供給する。イベントノードは、規則の従属リスト及びスケジューリングされるべき他のイベントノードと結合される。

40

【0077】

文字位置データは、ラインフォーマットバッファ及びドキュメントファイルにおけるトークンのスタートの文字位置を含む。それは更に両者における効果の範囲即ちトークンの長さを含む。例えばラインフォーマットバッファに現れるがドキュメントには現れない、作成されるテキスト、表示フィールド等、又は、ドキュメントに現れるがラインフォーマットバッファには現れない、隠されたテキストのような作成物によって文字位置が異なる場合があるので、ラインフォーマットバッファ及びドキュメントの両者に対する文字位置データが含まれる。

50

【0078】

トークンと共に送られる値は、それと結合しているイベントの値である。上述のように、1つのイベントが規則ベースの変数ノードによって表される。テキストトークンについては、その変数の値は1か又はゼロである。ここで1は、イベント発生器がイベントを検出したことを表す。フォーマットトークンについては、この値は非ゼロかゼロである。フォーマット設定では有用な情報を表すので、その値は非ゼロである。例えば、「インデント」に対応するフォーマットトークンの値は、インデントのための水平方向のスペースの合計を表す。フォーマット設定の目的のためにこの値がゼロである場合、それにも拘わらずイベントが検出されると、その値が一時的に強制的に非ゼロ値にされ、次に元に戻してイベントが生起したことをイベントエンジンに表示する。

10

【0079】

トークンの概念がトークンインターバルと混同されないことが重要である。トークンは、そのトークンインターバルでスタートしドキュメントインターバルセクションまで範囲が広がる種々のインターバルセクションにおいて、イベントを表すことができる。テキスト及びフォーマットトークンは、ユーザー入力から識別されるイベントの型を表示する。他方、1つのイベントインターバルは、上述のように状態変化のユニットであり、「トークンインターバル」はこの実施例におけるイベントインターバルの型の1つである。変更のユニットを構成するものは設計の判断であり、変化し得る。トークンインターバルは、図3のトークンインターバルセクション70aに対応する。図3の例では、テキスト及びフォーマットトークンキャッシュからトークンインターバルセクション70aの第1レベル71に向かう矢印によって示されるように、テキストトークン及び或る種のフォーマットトークンが、トークンインターバルセクションで、規則ベースのノードにインデックスされる。更に、ラインインターバルセクション70bの第1レベル72でノードに、及び、同様に文節インターバルセクション70cの第1レベル73でノードに向かう矢印で示されるように、他のフォーマットトークンがライン及び文節インターバルセクションにインデックスされる。

20

【0080】

判断ステップ120で一般的に説明したように、トークン発生器が、文字又は文字の順序が完全なトークンを構成するか否かを最初の質問として決定する。この最初の質問は最適化である。最初に完全なトークンが存在するか否かを決定することによって、不必要な処理を避けるのである。この最適化を実行するため、トークン発生器は、レクサーのデジェネラティブのバージョンを用いる。トークン発生器は、文字翻訳テーブル及び状態マシンテーブルを用いて、文字又は文字列がトークンを構成するか否かを識別する。文字翻訳テーブルは、文字の型を分類する。状態マシンテーブルは、文字翻訳テーブルからの結果を取り、文字の型の遷移を識別し、文字又は文字列がトークンを構成するか否かを決定する。

30

【0081】

図6は、デジェネラティブレクサーが完全なトークンを識別する方法を説明するフロー図である。

【0082】

ステップ133では、デジェネラティブレクサーが、処理されているテキストの現在のランから取出した文字を文字翻訳テーブルに照会する。文字翻訳テーブルは、状態マシンテーブルにおける列番号を返す。状態マシンテーブルは文字翻訳テーブルなしでも利用できるが、別個の文字翻訳テーブルを用いると、状態マシンテーブルの必要なサイズが小さくなり、特定の文字セット（例えばロシア文字、ANSI、IBM）を扱うことができるようになるので好都合である。当業者にとっては、状態の変化及び処理されるテキスト要素の型の情報を得るために他の技術を用いることができることは明らかである。

40

【0083】

デジェネラティブレクサーは、初め状態マシンのゼロ行の文字翻訳テーブルによって指示された列にエンタリーする。各行はレクサーの異なった状態を表す。各行と列との交点には、現在の文字の処理の後のレクサーの次の状態である新しい状態値が存在する。次の状

50

態が行の全数より小さい場合は、状態値は次の文字のために使用される次の行を示す。この値が行の数より大きい場合は、そのトークンについての予備のトークンの型を表す。第1文字の後、デジェネラティブレクサーが状態マシンの現在の行の、文字翻訳テーブルから返された列にエントリーする。これはステップ134に示されている。各文字が打ち込まれる時には、新しい状態が行の数を超えるまで処理が継続する。この点における状態値はそのトークンの予備トークンの型の値である。好ましい実施例は、4行を有する有限の状態マシンテーブルを用い、行ゼロはエントリーの開始点として使用され、行1は空白又はホワイトスペースに対応し、行2はアルファベット文字に対応し、行3は数字に対応する。

【0084】

10

好ましい実施例においては、状態マシンテーブルのための列は0と11との間であり、列0は空白及びホワイトスペースに対応し、列1はアルファベット文字A-Zに対応し、列2は数字0-9に対応し、残余の列は句読点、記号等の他の型の文字に対応する。

【0085】

テキストのランの第1文字がアルファベット文字の場合は、文字翻訳テーブルが状態マシンの列1を示す。状態マシンの行0、列1の値は新しい状態を示し、行2、列1の値はレクサーの新しい状態を示す。文字トークンの連続するアルファベット文字が走査される時は文字翻訳テーブルは同一の列を指し続け、列1、行2の交点の新しい状態はアルファベット文字を処理する行2の状態に留まる。例えば、コンマのような句読点記号がイベントフックから到着すると、これはアルファベット文字ではないので、文字翻訳テーブルは、句読点記号のその型に使用される列を指示し、その列の新しい状態は列2以外になる。

20

【0086】

判断ステップ136においては、デジェネラティブレクサーが、ステップ135からの新しい行番号が状態マシンの行の全数より大きいかを決定し、肯定ならば、その文字又は文字列が完全なトークンを構成することの検出を完了する。状態マシンテーブルが状態の変化を識別するので、デジェネラティブレクサーは、テキストのワードが処理される時点を決めることができ、次の文字を、その前のワードの文字とは別に処理されるべき、コンマ、ピリオド、コロンの、又は他の型の句読点として識別することができる。

【0087】

ステップ136の後、デジェネラティブレクサーが、文字又は文字列が完全なトークンを構成するか否かの決定を完了する。肯定の場合、処理は図5の122に続く。

30

【0088】

デジェネラティブレクサーが完全なトークンを識別すると（図6の120）、トークン発生器82がフルレクサー84を呼出し、それに結合している特定のテキストトークン又はフォーマットトークンを識別する。トークンイベント発生器がレクサー84を呼出し、特定の範囲の文字についてのトークンインターバルイベントを発生する。レクサーの呼出しにおいては、トークンイベント発生器がトークンを返す文字の範囲を特定する。

【0089】

レクサー84は、ドキュメントから走査された文字からのトークンを識別する。レクサー84は、ラインフォーマットバッファをアクセスし、ドキュメントから走査された文字のラインを記憶する。次のランの取得機能86を呼出すことにより、レクサー84が、ラインバッファ中にある均一にフォーマットされたテキストのランに対するポインタを受信する。

40

【0090】

レクサー84はテキストトークンを識別し、それらを、テキストトークンを一時的に記憶するためのデータ記憶であるテキストトークンキャッシュ88に置く。文字又は文字列がテキストトークンを構成するか否かを決定するために、レクサー84は、デジェネラティブレクサーに関して上述した文字翻訳テーブル及び状態マシンテーブルを利用する。

【0091】

図7A及び7Bはレクサーによって遂行されるステップのフロー図を示す。レクサーは、デジェネラティブレクサーと同様に、ラインフォーマットバッファからの文字を走査し状

50

態マシンテーブルを更新する。ステップ138 は、レクサーがラインフォーマットバッファから次の文字を得る方法を示す。このステップにおいては、このランに走査すべき文字がない場合は、レクサーは次のランの取得機能と呼出し、他のランを引出し、ポインタをリセットする。ステップ139-142 は、デジェネラティブレクサーで実行されるステップ133-136 と同一である。レクサーは、状態マシンテーブル中のデジェネラティブレクサーと同一の状態を実行し、デジェネラティブレクサーが停止する箇所で停止することができる。

【0092】

現在の行番号が状態マシンテーブル中の行の数を超えていない場合は、レクサーが、次の文字を得る前に、走査し過ぎているか否かを決定する。これは、単に、レクサーがトークンイベント発生器によって特定された文字範囲を超えて走査しようとしているか否かをチェックすることを意味する。1つの例は、レクサーが挿入点に対して走査する時である。次の文字が挿入点を超えている場合は、レクサーが「トークン未完成」を返す。

10

【0093】

再び判断ステップ142 を参照すると、現在の行番号が状態マシンテーブル中の行の数を超えている場合は、レクサーは判断ステップ144 に進む。トークンの識別においては、レクサーはトークンの最後の文字を超えて走査することができる。このような場合、レクサーはステップ145 で説明した適当な数の文字をバックアップする。状態マシンテーブルの最終状態はトークンの予備の型を指示する。この点で、レクサーは予備のトークンの型（例えばアルファベット、数字、デリミタ）を持ち、文字の番号をバックアップする（1又は0文字）。レクサーは次に、ステップ156 に示すように、行番号をゼロにリセットする。

20

【0094】

次に図7Bを参照すると、レクサーは送出ルーチンと呼出し、予備のトークンの型（状態マシンテーブルからの最終状態値）に基づく1又は複数のトークンを識別する。

【0095】

アルファベット及び数字の文字列のような或る種のトークンについて、特定のトークンを識別する方法は、ルックアップテーブルを用いてトークンを識別することを含む。規則コンパイラ62が、規則ソース60をコンパイルする時にこのルックアップテーブルを発生する。アルファベット文字列については、送出ルーチンが大文字化をチェックし、「大文字でスタート」、「大文字あり」又は「全て大文字」を示すフォーマットトークンを発生することができる。従って、レクサーが最初にテキストトークンを識別するとしても、それは更に或る種のフォーマットトークンを識別することができ、それらをフォーマットトークンキャッシュに置くことができる。

30

【0096】

他のトークンに対して追加の処理が必要である。レクサーは、文節マークのような特定の文字について、その位置に基づいてトークンを分析しなければならない。

【0097】

送出ルーチンがリターンした後、レクサーは次にステップ148 に進み、現在のトークンの長さ及び次のトークンのスタート位置を計算する。

【0098】

トークンがトークンインターバルイベントに対応するか否かはその型に依存する。この実行において、「間接」トークンはトークンインターバルイベントに対応しない。間接トークンは空白、注釈マーク、及び脚注マークのような文字である。レクサーは、これらの間接トークンを直ちに返さずに、ステップ150 に示すように、テキストトークンキャッシュにそれらをキャッシュする。「直接」トークンはインターバルイベントに対応し、イベント発生器に返される。一般的に、直接トークンは、アルファベット又は数字の文字列、又は句読点のようなデリミタである。

40

【0099】

フォーマットレクサー90は、同一のフォーマット設定を持つ1又は複数の文字を表すフォーマットトークンを識別するために用いられる。フォーマットレクサー90は、次のランの取得機能86によって返されたテキストのそれぞれの新しいランに対して作用する。図8は

50

フォーマットトークンを識別する処理を詳細に示す。フォーマットレクサー90は、均一にフォーマットされたテキストのランからのフォーマットトークンを識別する。更に多くの文字を走査することが必要なためにレクサー84が次のランの取得機能86を呼出す場合、ラインフォーマットバッファ中の均一にフォーマットされたテキストのランへのポインタを返す。図8は、処理の第1ステップとしてこのステップ164を示す。文字の現在のランから、フォーマットレクサー90が、ドキュメント及びラインフォーマットバッファ両者、規則ベースノードインデックス、及びフォーマット特性の値に基づくフォーマット特性の変化の開始位置及び長さを識別する。フォーマットレクサー90は、ステップ166、168に示すように、この情報をフォーマットトークンキャッシュ92に格納する。フォーマットレクサー90は、フォーマットトークンを、現在文字位置の昇順に処理されているテキストのランのためのフォーマットトークンキャッシュ中に置く。フォーマットレクサーは、フォーマットトークンがドキュメント中に存在する順序に正しく対応するように文字位置の昇順に従ってフォーマットトークンをソートし、文字の次のランを処理する時にフォーマットトークンキャッシュを置き換える。

10

【0100】

再び図5を参照すると、トークンイベント発生器が、各トークンイベントインターバルについてのイベントエンジンを呼出す。トークンインターバルイベントを定める各トークンについて、トークンイベント発生器はステップ126及び128により処理する。レクサーがトークン未完成を返すと、判断ステップ124がステップ130に分岐する。

20

【0101】

トークン発生器は、ステップ126に示すように、イベントエンジン中のスケジューリングルーチンを呼出し、テキスト及びフォーマットトークンをスケジューリングする。トークン発生器は直接トークン、及び、キャッシュされた間接トークン及びスケジューリングのためのイベントエンジン52に関連するフォーマットトークンを送出する。レクサーは、キャッシュされた間接トークンをイベントインターバルを定めるトークンと共に送出する。

【0102】

トークンインターバルを定める各テキストトークンについて、トークン発生器は、ステップ128に示すように、イベントエンジン中の評価ルーチンを呼出す。次に、イベントエンジンは、規則ベース中のトークンインターバルセクションについての全てのイベント及び規則を評価する。

30

【0103】

図9A及び9Bは、イベント発生器82から送られて来たイベントをイベントエンジン52が処理する過程を説明するフロー図である。イベントエンジン52に送られて来たテキスト及びフォーマットトークンは、規則ベース58中のノードに対するインデックスを含む。規則ベース58は、一次元のアレイからなり、各インデックスはこのアレイ中のセルに対応する。

【0104】

更に、アレイ中の各セルは、コア従属リスト又はボディ従属リストの2つのアレイの1つに属している。ノードは、対話型自動フォーマット設定機能のフェーズに基づいてこれらのリストの1つに属する。同時には1つのリストのみがアクティブである。ボディ従属リストはユーザーが打ち込みを行う時に適用される。コア従属リストは初期化フェーズの間及びユーザーが打ち込みを行う時に適用される。ボディ従属リストは、そのノードについてのコアリストへの全てのエントリー及び付加エントリーを含む。初期化は、ユーザーが挿入点を動かしした場合に、対話型自動フォーマット作成機能をリセットするための処理である。初期化の間に実行される走査は後に説明する。

40

【0105】

各ノードは2つの従属リストを有する。規則ベース中の各ノードは、それぞれの従属リストアレイ中に対応するセルを持つ。このアクティブ状態の従属リストアレイはノードアレイ中のノードと同一のインデックスを持つが、このアレイ中のセルは、そのノードについてのアクティブ従属リストである。

50

【0106】

ノードは規則又はイベントを表すデータ構造である。データ構造は22バイトの大きさであり、ノードに関連する種々の情報を含む。データ構造は、規則ベース中のインターバルセクションに対応するレベル番号を含む。それは更にノードの型、即ち規則又はイベントを特定する。それは、ノードが遅延評価のためにスケジューリングされるか否かを示す遅延マスクを含む。この遅延マスクは、順序規則に結合された遅延を実現する。

【0107】

このデータ構造は、更に、ノードが評価キューに存在する場合、リンクされたリストにおける次の項目へのポインタを含む。この値が空白の場合は、イベントエンジンはノードがいずれの評価キューにも存在しないことを知る。

10

【0108】

イベントノードについては、データ構造は、1つのインターバルの評価の後で自動的にクリアされるべき、リンクされたイベント変数のリスト中の次のイベント変数へのポインタを含む。1つのインターバルセクションについての全ての規則及びイベントの評価の後、評価ルーチンは、この「自動クリア」リスト中のイベント変数の値をクリアする。リンクされたリストを用いることにより、イベントエンジンは、ノードが自動クリアリストにあるか否かの決定を行うことができる。規則が自動クリアリストへのポインタを必要としないことを除いて、このノード及び規則についての構造は一般的に同一である。

【0109】

ノードは、ノードに結合され、規則又はイベントに従属する規則及びノードを掲載した、ノードに対応するリンクされたリスト、所謂従属リストを持つ。上述のように、アクティブ従属リストアレイ中にノードのインデックスを付けることによって、このリストが発見される。従属リストアレイの2つの型、即ちコア及びボディは、従属リストの2つのアレイとして実現される。初期化フェーズの間はコア従属リストアレイがアクティブである。ユーザーが打ち込む間はボディ従属リストアレイがアクティブである。この実施例においては、ボディ従属リストはコア従属リストを包含する。

20

【0110】

イベント発生器がスケジューリングルーチンを呼出してこれにトークンを送る場合、スケジューリングルーチンは、図9Aの最初のステップ170に示すように、トークンによって識別されたノードのための、アクティブな従属リスト中の全てのイベント及び規則のスケジューリングを行う。規則及びイベントをスケジューリングするため、イベントエンジン52が、従属リスト中の各イベント及び規則向けのポインタを、規則ベース中のノードのレベルにおける評価キュー上に押し出す。

30

【0111】

規則ベース中の各レベルについて、2つの個別の評価キューが存在する。再び図3を更に参照すると、各ノードが規則ベース中の1つのレベルと結合されている。各レベルは、アクティブ評価キュー及び遅延評価キューを持つ。アクティブ評価キューにスケジューリングされたノードは直ちに評価される。しかしながら、遅延評価キューにスケジューリングされたノードは、特定の数のイベントインターバルの間遅延する。

【0112】

遅延評価キューはイベント及び順序規則の遅延評価をサポートする。この遅延をサポートするため、各ノードは、評価されるノードがどのイベントインターバルに存在するかを示す遅延マスクを有する。遅延マスクは、そのノードに結合したインターバルにおける新しいイベントをイベント発生器82が送る時に1ビット右にシフトする。マスクにおける第1ビットが存在し、シフトがそれを動かすと、イベントエンジンがノードをノードのレベルのアクティブキューに転送する。例えば、順序規則がトークンインターバルセクションにおける遅延の2つのインターバルについてスケジューリングを行うと、2つのトークンインターバルイベントがイベントエンジン52に到達するまで順序規則が評価されない。従って、遅延評価キューにより、自動フォーマットについての条件を一連のイベントに基づいて書くことが可能になる。

40

50

【0113】

再び図9A及び9Bを参照すると、ステップ172-190はイベントエンジン中の評価ルーチンに属する。スケジューリングが完了すると、イベント発生器がそのインターバルセクションについての評価ルーチンと呼出す。この評価ルーチンは、遅延評価キュー中のノードをアクティブキューに転送するステップ172で始まる。

【0114】

次に評価ルーチンは、ステップ174に示すように、次のノードをキューから移動させてそれを評価する。ノードがイベントである場合(176)、次に評価ルーチンは、ステップ180に示すように、スケジューリングルーチンと呼出してそのノードの従属リスト中の規則及びイベントをスケジューリングする。ノードが規則の場合、判断ステップ178に示すように、その規則は、従属リストへのエントリーのスケジューリングの前に、「誤りではない」ことを試験しなければならない。

10

【0115】

規則が「誤りではない」ことを試験する場合、この論理は判断ステップ179に進む。規則がそれに結合された動作を持つ場合は、ステップ181に示すように、次にこの動作が実行される。ここで用いられる「動作」は、規則が満足される時に実行される規則に結合されたコードに属する。このコードは種々の機能を遂行する。特に重要なものは遅延動作を発生する機能である。諸動作が遅延動作を遅延動作キューに置くことができる。他の型の動作は新しい規則及び／又はイベントのスケジューリングであり、このステップはステップ180に示されている。

20

【0116】

判断ステップ182においては、評価ルーチンが、現在のレベルにおける評価キューが空であることを見出すためのチェックを行う。否定の場合、このルーチンはステップ174に戻る。肯定の場合、このルーチンは判断ステップ184に続く。最初の繰り返しの間、「現在」のレベルがインターバルセクションの最初のレベルである。例えば、イベント発生器がイベントをトークンインターバルセクションに送る場合、イベントエンジンは、トークンインターバルセクション中の第1レベル71で評価を開始する。図3に示された規則ベースの例においては、第1レベル71はトークンインターバルセクションの基底にある。

【0117】

判断ステップ184では、評価ルーチンが、インターバルセクションの終わりまで、各レベルについて評価キュー中のノードの評価を続ける。ステップ186に示されるように、現在のレベルで最後のノードが評価された後、評価ルーチンはそのインターバルセクションの次のレベルに進む。

30

【0118】

そのインターバルセクションが終わると、ステップ188に示されるように、評価ルーチンはそのインターバルについての全ての变化したイベントをクリアする。上述のように、イベントを表す各ノードは、自動クリアリスト中に次のノードへのポインタを持つ。このノードが变化した値を持たない場合は、それはこのリスト中に存在せず、ポインタは零を指示する。このポインタが零ではない場合、イベント変数の値がクリアされる。

【0119】

或る場合においては、評価ルーチンが他のインターバルセクションに遷移し、そのインターバルセクションについて評価処理を繰り返す。この型の遷移を実行するために、イベントインターバルが遷移規則を含む。遷移規則についての条件が満足されると、遷移規則は遷移変数の値を遷移先のインターバルセクションに対応する数に設定する。判断ステップ190においては、評価ルーチンが遷移変数の値をチェックする。遷移変数が他のインターバルセクションに対して設定されると、評価ルーチンはステップ172に戻る。そうではない場合は評価ルーチンは終了する。

40

【0120】

評価処理の間、イベントエンジンは「遅延動作」106と呼ばれる自動フォーマット動作を発生することができる。システムのこの観点は、図2に規則ベース58から遅延動作キュー

50

108 への矢印で示されている。動作規則が満足されると、イベントエンジン52が遅延動作を遅延動作キュー108 に置く。

【0121】

遅延動作キュー108 は遅延動作レコードのリンクされたリストである。遅延動作レコードは多数のフィールドを含む。1つのフィールドは、それが存在する場合は1つの編集範囲を特定する。編集範囲はドキュメント中の開始文字位置と終了文字位置で表される。それは、自動的な調整によって登録される。この調整は1つの動作が他の動作に影響する場合に敏感である。例えば、動作の遂行中、編集ルーチンがドキュメントに現在ある文字の位置を変えながら文字を加えることができる。他の遅延動作が調整のために登録されると、それに対応する編集範囲は、その編集範囲が特定の文字に対応することが明確になるために必要な調整を受けることができる。これは、更に、他の規則の変化の影響について調整を必要とせずに編集の決定を行うための規則を可能にする。しかしながら、全ての動作が編集範囲を含むわけではない。ドキュメントを編集しない動作については、編集範囲は不要である。

10

【0122】

遅延動作レコードの他のフィールドは動作レコードの型即ち「オブコード」を特定する。このオブコードは同一の作用を有する動作のクラスを記述するために利用することができる。

【0123】

更に他のフィールドは、キューの次の動作への次のポインタを含む。次のポインタが零を指示する場合は、それはそのキューの最後のレコードである。

20

【0124】

合成動作は動作のサブリストへのポインタを含む。このサブリストは、合成動作が遂行される時に実行されるべき動作を特定する。

【0125】

図9A及び9Bに示された評価ルーチンがリターンした後で、遅延動作キュー108 中の遅延動作のいずれかが遂行される。再び図5を参照すると、判断ステップ130 に示されるように、トークンイベント発生器が、遅延動作キュー中に遅延動作が存在するか否かを決定する。肯定ならば、トークンイベント発生器は遅延動作インタプリタ94を呼出し、遅延動作キュー中の動作を遂行する。

30

【0126】

イベントモニター50の遅延動作インタプリタ94の部分は、これらの遅延動作を実行するための開始機能95の役割を持つ。遅延動作インタプリタ94は適当なルーチンを呼出し、遅延動作を実行する。それは更に「取消し」スタックをセットアップする。取消しスタックは、既に実行された最後の動作又は合成動作をユーザーが取消することができるようにする。例えば、自動フォーマット規則がリストを自動黒丸リストに変換すると、ユーザーは編集メニューから取消しコマンドを選択してそれを取消することができる。合成動作であっても取消しスタックへの1つのエントリーとして現れる。これにより、ユーザーは、取消しコマンドを1度選択するだけで合成動作の各々の動作を取消することができる。

【0127】

遅延動作インタプリタは、キューに動作の残りがなくなるまで一時に動作を遂行する。合成動作については、遅延動作インタプリタが動作の残りがなくなるまでサブリスト中の各動作を反復し、実行する。次にそれはキュー中の次の動作を続ける。遅延動作キューが空の場合、イベント発生器は他のイベントが生起するまで停止する。

40

【0128】

次に、ユーザーが打ち込む間にこのシステムが如何にフォーマット設定を行うかについて説明する。ユーザーは頻繁にカーソルの挿入位置を変更するので、システムはこのユーザー入力の形式を検出し、適切に自動フォーマット設定機能をリセットしなければならない。

【0129】

50

ユーザーが挿入位置を移動し、打ち込みを開始する時、システムはそれ自体を初期化する。初期化処理は、ドキュメント中の位置を挿入位置の前に移動させ、この位置から挿入位置まで、トークンについて走査を行うことによって開始される。走査は前の文節の終端の前方5文字から開始され、挿入位置まで続く。最適にするには、カーソルが存在する文節の現在のラインを通して挿入位置の前256文字が存在する最初のラインのみが走査される。

【0130】

自動フォーマット作成機能は、初期化の間バッチモードで動作する。初期化ルーチンは、最初に現在の文節の開始の前の5文字をラインフォーマットバッファにロードすることによってその走査を開始する。続いて、上述のようにレクサーがトークンを分析する。各トークンインターバルについて、初期化ルーチンが評価ルーチンを呼出す。

10

【0131】

初期化走査は、数個の点について対話モードと異なる。この初期化フェーズの間、ボディ従属リストではなくコア従属リストが用いられる。更に、遅延動作は生成されない。初期化の目的はシステムをリセットすることである。前の文節の終端の前5文字から開始することによって、システムが挿入位置に至るイベントのパターンを検出することができる。走査される文字の特定の範囲は変更することができる。この場合、この範囲は、前の文節がどのように終わったかを識別できるように、又はこれに代えて前の文節が存在したか否かを識別できるように選ばれる。

20

【0132】

規則ベースイベントエンジンは、自動フォーマット設定をサポートするのに加えて、他の種々のワード処理機能をサポートする。イベントエンジンは、広い範囲の種々のユーザー入力イベントについて有効な規則分析をサポートするので、ユーザー入力イベントに依存する他の自動ワード処理機能をサポートすることができる。1つの機能は、ユーザーがしていることを検出して助けになるフィードバックを与える助言ウィザードである。他の機能は、自動的な修正即ち「自動修正」である。自動的な修正は、典型的なエラーを監視し、それらが起きた時にそれらを修正することである。イベントエンジンの重要な能力は、自動修正を強化することである。ユーザーが自動修正動作を覆すか否かを監視することによって自動修正規則に例外を設けるためにイベントエンジンを用いることができる。

30

【0133】

図2に示したシステムは助言ウィザードをサポートする。助言は、ディスプレイスクリーン上のボックスに表示される有用なテキストである。助言は、有用なテキストを簡単に提供し、ユーザーにフィードバックすることができる。テキストを表示するのに加えて、助言は、更にユーザーに、動作を選択するためのオプションを与えることができる。このシステムは、ユーザーの入力を促す押しボタン制御を助言ボックス中表示する。この押しボタンは、ユーザーが動作をさせるか又はシステムが自動的に実行した動作を取消することができるようにする。例えば、押しボタン制御を持つ助言ボックスは、合成遅延動作の部分として表示することができる。自動フォーマットモードにある間、ドキュメントを編集する合成遅延動作は助言ボックス及び押しボタン制御を表示し、ユーザーが編集動作を取消することができる。逆に、自動フォーマットモードが選択されない場合、可能性がある編集動作を記述するテキストを含む助言ボックス中で、ユーザーが「実行」押しボタンを選択するか否かに従って、編集動作が行われる。

40

【0134】

助言ウィザード動作をサポートするため、その動作を起動するユーザー入力イベントを確定し、同時に動作を確定するための規則を書込むことができる。これらの規則は、次に、規則コンパイラで規則ベース58中にコンパイルされる。助言をサポートする合成規則は、通常、ユーザーインタフェース又は対話型イベントに基づいて書込まれる。例えば、ユーザーが対話の開始のようなイベントのパターンを見て、何もオプションを選択せずに何回もそれを閉じるにより、ユーザーがまごついているか否かをシステムが検出することができる。助言ウィザード規則は、通常、ユーザーインタフェース及び対話イベントに基

50

づいて書込まれる一方、トークン及び文字イベントを含む他のイベントに基づいても同様に書込まれる。

【0135】

図2に示したシステムは更に自動修正をサポートする。自動修正は、通常の打ち込みエラーを検出し、自動的に修正を行うことを意味する。例えば、自動修正機能は、文の最初のワードを大文字にすること又は2つの大文字が連続している場合に2番目の大文字を小文字に変えることを含む。第1の機能は、文中の最初のワードの最初の文字が大文字ではないという通常の誤りに向けられる。第2の機能は、偶発的に、ユーザーがワードの最初の文字を大文字にする時に必要以上に長くシフトキーを押さえるという問題に向けられる。

【0136】

一般的に、自動修正機能をサポートするための合成自動修正規則と共に、この機能に対する例外を書込むことができる。最初のワードの大文字化規則に対する例外のクラスの1つは、ユーザーが略語の後にピリオドを、次に文の初めではない他のワードを打ち込む場合である。例えば、「etc.」は通常は文の中間に用いられる。この機能を実現する規則は、この規則が適用されない若干の例外を示す例外リストを含むことができる。基本的には、例外は自動修正規則に抵触する規則である。

【0137】

規則ベース中に予め定めておくことができる例外の数には限界がある。一方、システムは対話的に例外を作り出すこと、例えばユーザーの挙動に基づく例外を加えることをサポートする。

【0138】

合成規則は、自動修正規則を遂行するために必要な遅延動作を特定する動作規則を含む。動作規則が満足される場合、それが、自動フォーマット設定動作に関して上述した遅延動作と同様の遅延動作を発生する。特に、自動修正規則に基づいてテキストを編集するための動作を発生する。自動フォーマット動作と同様に、自動修正動作はドキュメントの編集を必要とせず、大部分の場合、必然的にそれが編集を含むことになる。

【0139】

例外を監視するため、動作規則は更に追加の動作を発生する。最初のワードの大文字化の例は概念を説明するための助けになる。ユーザーが「after」を「aft.」と省略し、「aft. five」という句を打ち込むとしよう。ワードの最後のピリオドの後にスペースが続くことにより、「five」を大文字化する自動修正規則が起動される。規則ベース中の動作規則が「誤りではない」を試験する時、合成動作が遅延動作キューに置かれる。

【0140】

合成動作は、それぞれが1又は複数の動作を含むことができる3つの基本的な動作を含む。1つの動作が最初の文字の大文字化である。他の動作が前のワード「aft.」及びワード「five」を捉える。そして第3の動作が監視ルーチンと呼出す。

【0141】

監視ルーチンは、ユーザーがバックスペースを行うこと等により自動修正を削除し、元のテキストでの置き換えを行う場合を検出するように設計される。監視ルーチンに対するアーギュメントとして、遅延動作インタプリタが2番目のワード「five」とこのワードのいずれかの変形との比較を実行する機能のIDを送出する。比較の結果はユーザーが自動修正を元のテキストで置き換えたか否かを示す。

【0142】

ワード「five」を変形するいかなる企てに対しても、監視ルーチンが比較機能と呼出す。比較の結果が一致する場合は、比較機能がワード「aft.」を例外リストに加える。この実施例においては、比較機能は、ユーザーが取消しコマンドと呼出す時に一体として用いられる規則と同一なので、規則として実現される。しかしながら、それはまた個別の機能として実現することもできる。ユーザーが、自動修正を「取消す」ために再度打ち込みをせずに取消しコマンドを選択した場合は、例外リストに例外を加えるために比較規則が起動される。

10

20

30

40

50

【0143】

この実施例においては、自動修正機能をサポートするため規則ベースをダイナミックに変更する多数の方法が存在する。自動修正機能は、1) 自動的に文字列を検出して他の文字列で置き換える機能、2) 文の最初の文字を自動的に大文字にする機能、及び3) ワードの2番目の文字を自動的に小文字にする機能を含む。第1の機能について、プログラムは監視し置き換えを行うべき多数の予め定義された文字列を持ち、「自動修正リスト」中のエントリーとしてリストされる。プログラムによって提供される自動修正リストを変更するために、ユーザーが自動修正対話を用いてリストにエントリーを加え又はリストからエントリーを削除することができる。

【0144】

第2及び第3の機能は、実際には第1の機能の特別な場合である。第2及び第3の機能に対する多数の例外が存在し、それらは各機能の例外リストに掲載される。ユーザーは、自動修正対話を用いて、例外リスト中のエントリーを変更することができる。この実施例においては、特別な場合を扱う対話ボックスは、自動修正対話の中からアクセスすることができる。これに加えて、対話ボックスを通して例外リストを変更するために、ユーザーが「取消し」ボタンを選択する場合又は上述のように特別な場合の機能を手動的に取消す時には、プログラムが追加の例外を発生することができる。

【0145】

次に、自動修正リスト及び例外リストに関係するプログラムの観点を更に詳細に説明する。

【0146】

この実施例においては、自動修正リスト及び例外リスト中の変更に基づいて規則ベースを変更することができる場合の3つの例が存在する。1つは、プログラムがファイルから自動修正リスト及び例外リストを走査するブート時間においてであり、そのリスト中のエントリーによって必然的に規則ベースが変更される。次の例は、自動修正対話を通して入力されたユーザーによる変更からである。そして第3は、ユーザーが「取消し」オプションを選択する時又は自動修正の特別な場合の1つを手動的に取消す時である。ブート時間においては、プログラムは自動修正及び例外リスト中の各エントリーを走査し、これらのリスト中のエントリーに従って規則ベースが変更されるべきが否かを識別する。自動修正対話で入力された変更については、プログラムが対話ボックス内の編集ボックス中に置かれた文字を走査する。ユーザーが特定の場合の1つを取消す場合は、第1文字大文字化の合成動作との関連で上述したように、例外に対応する文字列が捕捉される。

【0147】

走査が実行される時に文字列が特別に位置されていることは決定的なことではない。この実施例においては、レクサーは、ポインタが文字列の走査されるべき位置に向いて与えられた時に走査を実行する。従って、文字列がどこに位置しているかに拘わらず、レクサーは同様に動作する。しかしながら、トークンを特別に識別するために用いられる送出ルーチンは、文字列の走査が開始される箇所に依存して異なる動作を行うように設計することができる。例えば、文字列がドキュメントではなく編集ボックスから来るとすれば、文節及びラインマークのようなドキュメントに限って起きる種類のイベントを探す必要はない。

【0148】

文字列の走査は一般的に特定の文字列についてのレクサーの呼出しを含む。或る場合には、文字列の走査が一連のトークンの発生を含むことがある。しかしながら、多くの場合、文字列は1つの対応するトークンを持つただ1つのワードからなる。一連のトークンの場合、プログラムは一連のトークンの最後のトークンを採用し、それをルックアップテーブルで探し、そのトークンに対応するイベントノードが既に存在するか否かを決定する。1つのトークンの場合は、一連のトークンの最後のトークンを採用する場合より簡単にこのトークンを探し出す。

【0149】

トークンイベントがテーブルにある場合は、即ち既に存在する場合は、規則ベースが既に対応するイベントノードを持っている。この場合、イベントノードは簡単に他のイベントノードとリンクされる。リンクされるイベントノードの型は、走査が自動修正エントリーについて実行されるか、又は特定の場合の1つに対する例外について実行されるかに依存する。

【0150】

自動修正エントリーのためには、イベントノードは他の自動修正規則を起動するイベントノードにリンクされる。イベントノードは「自動修正」イベントといわれる。代替案として、自動修正エントリーに対応するイベントノードが自動修正規則に直接リンクされる。この代替案においては、自動修正エントリーに対応するイベントノードはまた自動修正イベントである。

10

【0151】

自動修正機能に対する例外については、イベントノードはこの機能を禁止する他のイベントノードにリンクする。例えば、この禁止イベントが真にセットされると、第1文字の大文字化のための自動修正動作が実行されない。

【0152】

或る場合において、例えば新しいエントリーが自動修正リストに加えられる場合は、新しいエントリーのための対応するイベントノードは存在しない。トークンイベントがテーブルに存在しない場合は、次に1つのイベントノードが規則ベースに加えられるべきである。イベントノードが加えられると、対応するトークンが更にレクサーのためのルックアップテーブルに加えられる。前述のように、ルックアップテーブルは、規則ベース中でインデックスによりトークンイベントをそのノードに対応させている。自動修正又は例外リストのいずれかに対する新しいエントリーのために、適切なリストが新しいエントリーを含むように構成される。

20

【0153】

新しいノードが作られた後、他のノードとのリンクが確立される。リンクを作るために、新しいイベントノードの従属リンクに1つのエントリーが加えられる。逆に、ユーザーが自動修正リストから自動修正エントリーを消す場合は、イベントノードと自動修正規則との間のリンクを消去する。この実施例においては、自動修正エントリーのためのイベントノードと自動修正規則を起動するイベントノードとの間のリンクが消去される。

30

【0154】

代表的な例においては、自動修正規則は以下のように動作する。自動修正規則にリンクしたイベントノードが評価されると、それが自動修正規則を起動する。次に自動修正規則が比較ルーチンを呼出し、これがテキストトークンキャッシュ中の1又は複数のトークンを自動修正エントリーと比較し、一致するか否かを決定する。一連のトークンの場合は、そのシーケンスの最後のトークンを検出した後で自動修正規則が起動される。次に、このシーケンスを通して後方に進みながら、比較ルーチンが、テキストトークンキャッシュ中のトークンが一連の自動修正エントリーと一致するか否かを見るためにチェックする。これが一致する場合は、次にドキュメントが、現存の文字列を自動修正リスト中のその置換部分で置き換えを行うように適当に編集される。

40

【0155】

この方法を用いると、多数の自動修正エントリー及び自動修正エントリーに対する例外が、容易に規則ベースに追加され又はこれから消去される。自動修正機能をサポートする規則は或るイベントが発生された後にのみ起動するように設計され得るので、イベントエンジン及び規則ベースは、この機能に対して効果的なサポートを行う。

【0156】

本発明の特別な実施例が詳細に説明されたので、本発明が本発明の範囲を逸脱することなしに変形され得ることが理解される筈である。例えば、イベントエンジン及びイベントモニターを実現するためのルーチンが本発明の範囲を逸脱することなしに変形され得る。それを実現するために用いられるデータ構造を含む規則ベースの特別な実現例も変形するこ

50

とができる。規則は、コンピュータプラットフォーム、アプリケーションプログラム及び他の因子に依存して変化することができる種々のイベントに従属する構造を持つことができる。規則ベースは柔軟性を持ち、種々のイベント型、イベントインターバル、及び対応するインターバルセクションをサポートする。規則は、他のインターバルにおけるスケジューリング従属イベント又は規則による遅延を持って実現され得る。要約すれば、多数のバリエーションが可能なのである。

【0157】

本発明の原理が実現されている多くの可能な実施例を考慮して、ここに記載された詳細な実施例は単に説明のためのものであって、本発明の範囲を限定するものと考えられるべきではないことが強調されなければならない。特許請求の範囲は、発明の範囲及び精神がこの

10

【図面の簡単な説明】

【図1】本発明の実施例に用いられるコンピュータシステムのアーキテクチャを説明するためのブロック図である。

【図2】本発明の実施例による自動フォーマット作成機能のアーキテクチャのブロック図である。

【図3】規則ベースの構成の例を説明する図である。

【図4】本発明の実施例によるイベント発生器を概観するためのフロー図である。

【図5】本発明の実施例における文字及びトークンイベント発生器によって実行されるステップを示すフロー図である。

20

【図6】本発明の実施例による完全なトークンの検出において実行されるステップを示すフロー図である。

【図7】本発明の実施例によるテキストトークンの識別において実行されるステップを示すフロー図である。

【図8】本発明の実施例によるフォーマットトークンの識別において実行されるステップを示すフロー図である。

【図9】規則ベースを通してイベントを伝搬する際の規則ベースイベントエンジンによって実行されるステップを示すフロー図である。

【符号の説明】

20 コンピュータシステム

30

22 コンピュータ

24 入力デバイス

26 出力デバイス

28 中央処理ユニット (CPU)

30 メモリーシステム

32 バス構成

33 算術的論理ユニット (ALU)

34 レジスタ

36 制御ユニット

38 高速主メモリー

40

40 2次記憶装置

50 イベントモニター

52 規則ベースイベントエンジン

56 ドキュメント

58 規則ベース

60 規則ソース

62 規則コンパイラ

70 インターバルセクション

71 トークンインターバルセクションの第1レベル

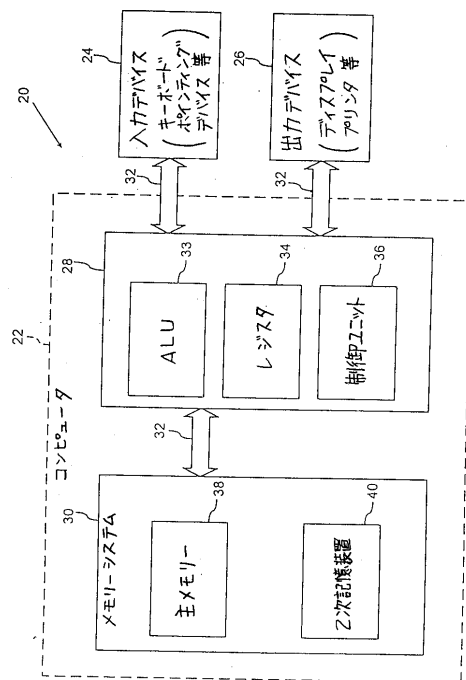
72 ラインインターバルセクションの第1レベル

50

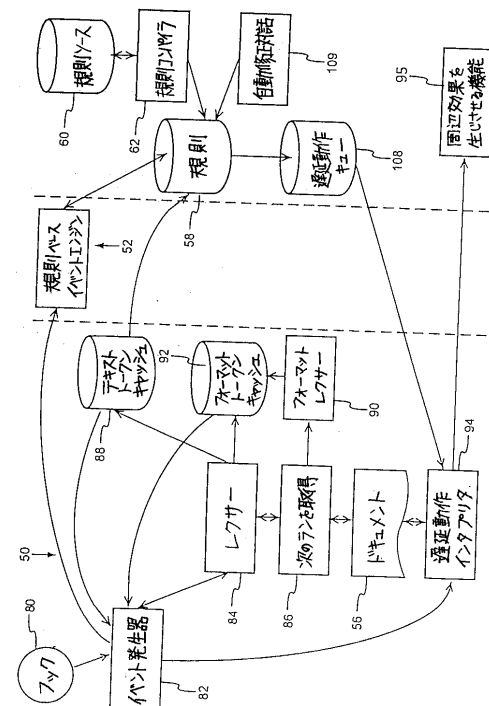
- 73 文節インターバルセクションの第1レベル
- 80 イベントフック
- 82 イベント発生器
- 84 レクサー
- 86 次のランの取得機能
- 88 テキストトークンキャッシュ
- 90 フォーマットレクサー
- 92 フォーマットトークンキャッシュ
- 94 遅延動作インタプリタ
- 95 機能
- 108 遅延動作キュー
- 109 自動修正対話

10

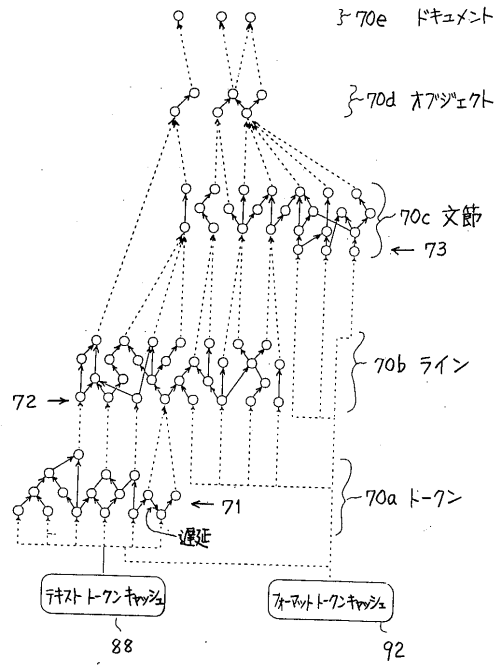
【図1】



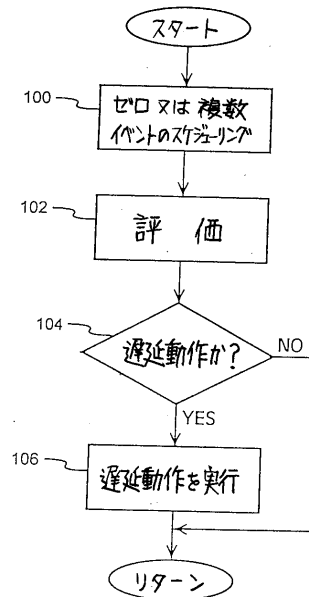
【図2】



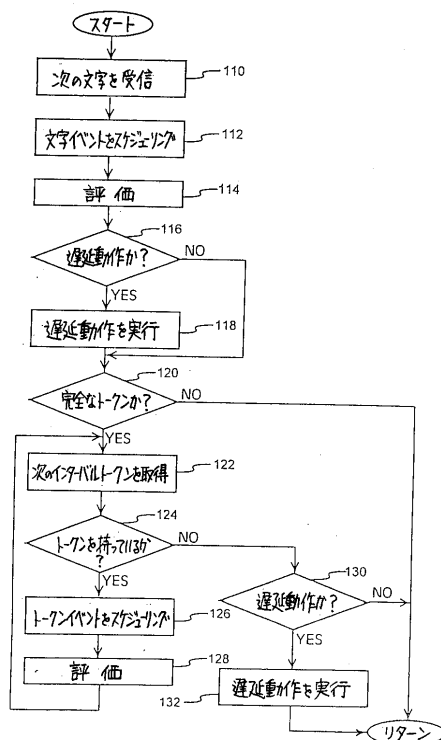
【図 3】



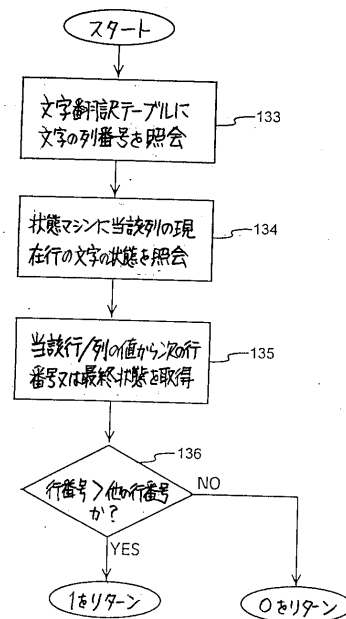
【図 4】



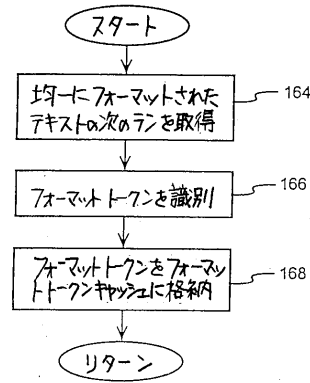
【図 5】



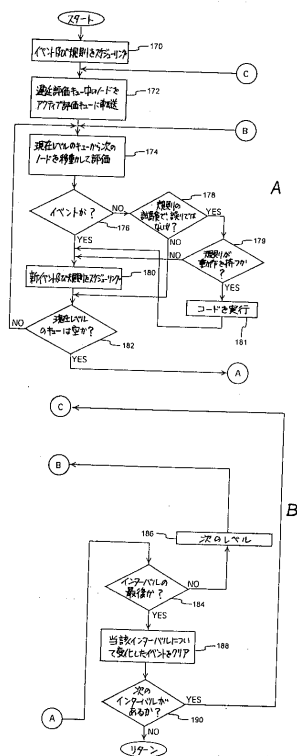
【図 6】



【图 8】



【图 9】



フロントページの続き

(72)発明者 デイル エル ジブソン

アメリカ合衆国 ワシントン州 98053 レドモンド トゥハンドレッドフォーティース ア
ヴェニュー エヌ イー 5216

合議体

審判長 田口 英雄

審判官 菅原 浩二

審判官 飯田 清司

(56)参考文献 特開平3-163623 (JP, A)

特開平5-282286 (JP, A)

特表平7-508116 (JP, A)

特開昭61-156466 (JP, A)

特開平7-85050 (JP, A)

特開平1-231167 (JP, A)

特開平3-271828 (JP, A)

特開平1-231166 (JP, A)

特開昭59-103132 (JP, A)

ワープロ666対決前哨戦、ASCII、日本、1995.03.01発行、第19巻、第3号
、305~320頁

Microsoft Word for the Macintosh ユーザーズガイド、日
本、1995.03.15発行、第1版、79-113頁

(58)調査した分野(Int.Cl., DB名)

G06F17/21